UNIVERSITY OF CAPE TOWN

DEPARTMENT OF STATISTICS

Unravelling black box machine learning methods using biplots

MSc Dissertation

Adriaan Rowan
February 2019

Supervisors:

Professor Francesca Little

Professor Sugnet Lubbe

Declarations

Copyright:

I hereby grant the University of Cape Town permission to copy and disseminate this work, or any part thereof, using any means, for the purposes of study and research.

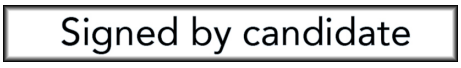
Plagiarism:

I, Adriaan Rowan, confirm that:

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the *Harvard Reference* for citation and referencing. Each contribution and quotation in this report from the work(s) of other people has been attributed and has been cited and referenced.

3. This dissertation titled, 'Unravelling black box machine learning methods using biplots', is my own work and is in my own words (except where I have attributed it to others).

Signature 

Date 2019/06/10

Abstract

Following the development of new mathematical techniques, the improvement of computer processing power and the increased availability of possible explanatory variables, the financial services industry is moving toward the use of new machine learning methods, such as neural networks, and away from older methods such as generalised linear models. However, their use is currently limited because they are seen as “black box” models, which gives predictions without justifications and which are therefore not understood and cannot be trusted.

The goal of this dissertation is to expand on the theory and use of biplots to visualise the impact of the various input factors on the output of the machine learning black box.

Biplots are used because they give an optimal two-dimensional representation of the data set on which the machine learning model is based. The biplot allows every point on the biplot plane to be converted back to the original p-dimensions – in the same format as is used by the machine learning model. This allows the output of the model to be represented by colour coding each point on the biplot plane according to the output of an independently calibrated machine learning model.

The interaction of the changing prediction probabilities – represented by the coloured output – in relation to the data points and the variable axes and category level points represented on the biplot, allows the machine learning model to be globally and locally interpreted.

By visualing the models and their predictions, this dissertation aims to remove the stigma of calling non-linear models “black box” models and encourage their wider application in the financial services industry.

Acknowledgements

"If I have seen further it is by standing on the shoulders of Giants" – Sir Isaac Newton[1]

I would like to thank the Giants on whose shoulders I stood on during the writing of this dissertation, and without whom it would not have been possible to even attempt it.

To my parents, for a lifetime of continuous support and motivation and their unfailing believe in my abilities to succeed.

To Professor Francesca Little, for all the help and support during the course work preceeding the dissertation.

To Professor Sugnet Lubbe, for suggesting the topic and for all the hours she spent explaining biplots to me and reviewing my notes. And through her, to Professor Niël le Roux, and through him, to Professor John Gower…

Soli Deo Gloria

[1] https://digitallibrary.hsp.org/index.php/Detail/objects/9792

Table of Contents

Chapter 1 Introduction

1.1 Background

In the field of insurance and banking machine learning is widely used, but mostly under the name of predictive analytic models.

For actuaries, a method often used to model the likelihood of various events occurring – be it determining whether a person qualifies for a loan or the optimal premium of an insurance policy – is called Generalised Linear Models, or GLMs (Boland, 2006). This is also the method that is explained in the syllabus of the Actuarial Society of South Africa (*ASSA syllabus*, 2018).

GLMs are popular because they have the advantage of providing coefficients that explain the impact of each variable on the final outcome of the model. This allows the actuary to explain why and how – for a particular instance – the machine learning model made the stated prediction. This is because each variable is assumed to be independent from the others and to have a linear impact on the prediction (Strumbelj and Kononenko, 2010). GLM theory also provides a range of statistical techniques that can justify which variables are statistically significant in predicting the targeted variable's outcome and therefore need to be included or excluded from the model.

Alternative models also have the ability to explain the impact of each variable on the model output. Examples are decision trees and naive Bayes models. These models are also very accurate in certain situations, just like GLMs are very accurate when the distribution of the data conforms to the assumptions underlying the GLM.

However, recent developments have resulted in machine learning methods[2], such as neural networks and gradient boosting, that have significantly better prediction accuracy than linear models. They are also not as dependent on the data conforming to certain assumptions that are required for traditional statistical prediction models. The improved accuracy of these new models is seen in the prediction accuracy of benchmark measures such as the accuracy of predicting hand written digits from the MNIST database (*'The MNIST database...'*, 1998). However, the drawback of using these more accurate models is that they often have less interpretable results. This is caused by the characteristics of the algorithms that make them so accurate, especially the use of ensemble models or allowing for non-linear interaction between the input features.

Ensemble methods combine the results of more than one model to ensure that inaccurate predictions of some models are compensated for by accurate predictions by the majority of the models.

[2] The machine learning methods refer to the class of the model. A machine learning model refers to a method that has been calibrated to a data set.

However, it means that there is no one exact model that can explain why a particular prediction was made. This method is used in random forest or gradient boosting models.

Models with non-linear interaction terms better capture the dependence of the output on all the input features. But that means that the impact of each input feature on the prediction is not fixed anymore and depends on the relative value of the other input features. This causes a trade-off between the interpretability and accuracy of machine learning methods.

1.2 The need for interpretability

Since ensemble and non-linear models are often more accurate, why has there not been a faster shift towards using these models in the financial services industry - for example, in predicting whether a person should qualify for a loan or an insurance policy? There are two main regulatory reasons for this.

The first reason concerns the statutory financial reporting requirements. In the context of an insurance company, the Statutory Actuary of an insurer must be able to justify the use of the model. Section 27 (1) of Board Notice 158 of the Financial Services Board of South Africa (Registrar of Long-term Insurance and Registrar of Short-term Insurance, 2014) states that: “The actuarial function must provide assurance to the board of directors regarding the accuracy of the calculations and the appropriateness of the assumptions underlying the insurance liabilities and the capital adequacy requirement.”

If results of the model used are not interpretable, the actuary will not be able to explain why the model has made the prediction. Whenever a situation arises where the model has made an incorrect prediction causing the company to lose money or to not be able to meet its liabilities, both the Statutory Actuary and the Board could be held responsible for the mistake. This is not a risk the Statutory Actuary or Board are willing to take.

The second reason concerns the need to explain to a potential client why their loan or insurance application was declined. New regulations in the European Union (Goodman and Flaxman, 2016) propose that individuals affected by algorithmic decisions have a right to an explanation of what factors caused the prediction to be made. In South Africa, section 62 of the National Credit Act states that the consumer has the “right to reasons for credit being refused.” (Government, 2006).

To give valid reasons, the results of the model for a particular person must be interpretable.

Definition of interpretability

Model interpretability is however not a clearly defined term and can therefore be understood differently in different contexts. As explained in Lipton (2017), interpretability can mean any of the following:

- Trust: There must be confidence that the model will perform well. This can extend beyond model accuracy and also mean that a person must feel at ease with the reasons for the predictions, even if the reasons are not used. The reason trust is required is that, as explained in Ribeiro, Singh and Guestrin (2016), “if the users do not trust a model or a prediction, they will not use it.” As an example, the accuracy of a doctor’s diagnoses is often accepted even

though he does not necessarily provide the exact reasons and thought process behind reaching his conclusions.

- Transferability: This refers to the ability of a model to be applied in the real world, without causing the resulting future predictions to be invalid. Therefore, the model must be robust against being manipulated by users who deliberately alter their input data to achieve specific model outcomes.

- Fair and ethical: This relates to the new European standards that require that models must produce interpretations for the purpose of assessing whether decisions produced automatically by algorithms conform to ethical standards (Goodman and Flaxman, 2016). For example, the model should not decline a loan application because of the person's race.

Lipton continues further to explain the properties of interpretable models. These properties fall into two broad categories:

- *Transparency,* i.e., how does the model work?
- *Post-hoc explanations,* i.e., what else can the model tell me?

Under transparency, properties can include:

- Simulatability: This refers to the ability of a human to take the input data together with the parameters of the model and in a reasonable time step through every calculation required to produce a prediction. This reflects both the number of model inputs and the computations required with the inputs. This implies that a sparse linear model is more transparent than a linear model with many input features.

- Decomposability: This requires that each input and calculation performed must be individually explainable. This therefore means that input transformations must be limited and must remain interpretable post transformation.

- Algorithmic transparency: This refers to the actual process by which the model's mathematics converges to a final calibrated model. Linear models and their coefficients are easy to explain, whereas the weights and calibration of a perceptron in a neural network are not.

Under post-hoc explanations, properties can include:

- Visualisations: There are many ways in which the results of a model can be visualised to let the user understand the impact of various features. One approach is using multi-dimensional scaling and biplots, which will be explored later in this dissertation.

- Local explanations: This refers to understanding why the model has made its prediction for a specific example. This is the approach followed by Ribeiro, Singh and Guestrin (2016), where the authors explain the decisions of the model by learning a new sparse linear model in a local region around the example.

- Explanation by example: This method identifies the *k*-nearest neighbours of similar examples to the specific case to give justification for the classification, as suggested by Caruana et al. (1999).

The definition discussed above has implications for how interpretable a model is considered to be. This means that the general statement that linear models are more interpretable than neural networks is not strictly true. For example, “with respect to algorithmic transparency, this claim seems uncontroversial, but given high dimensional or heavily engineered features, linear models lose simulatability or decomposability, respectively” (Lipton, 2017). In this regard the definition used for interpretability must be clarified.

In the case of humans, how the brain works is not fully understood, but a human can provide reasons for decisions made. The reasons given may not be the true reasons, as the decision maker is often influenced by many unconscious factors or biases. We do however trust the decision even if it is often incorrect. Similarly, the reason provided by transparent machine learning methods can be false or misleading. Therefore, “we should be careful when giving up predictive power, that the desire for transparency is justified and isn’t simply a concession to institutional biases against new methods.” (Lipton, 2017).

Definitions of accuracy

As explained before, accuracy should not necessarily be sacrificed to achieve interpretability. And if accuracy is sacrificed too much, there will not be trust in the model’s predictions.

Accuracy also does not have a single definition. It can be defined as the overall prediction accuracy of the model, often measured by the classification error rate of an out-of-sample test data set.

Accuracy could also be understood with regards to the interpretation provided. Ribeiro, Singh and Guestrin (2016) consider two definitions – local and global fidelity, or accuracy.

Local fidelity implies that the interpretation provided must be accurate in the area around the data point or prediction being investigated. Global fidelity implies interpretations that explain all the predictions made by the model. As explained by Ribeiro, Singh and Guestrin (2016), ”local fidelity does not imply global fidelity: features that are globally important may not be important in the local

context, and vice versa. While global fidelity would imply local fidelity, identifying globally faithful explanations that are interpretable remains a challenge for complex models.”

1.3 Scope of this dissertation

The goal of the dissertation is to develop a new technique with which to quantify and understand the impact of the various input factors on the output of machine learning models. Therefore, to generate *post-hoc interpretations that are globally or locally faithful.* The current techniques will be discussed as part of the literature study in Chapter 2. The new technique uses generalised biplots to visualise the data and provide explanations for the predictions of individual data points and is explained in Chapter 5 and Chapter 6.

To investigate a classification, a well-designed and accurate underlying prediction model is first needed. Therefore, part of this dissertation will focus on the methods of calibrating these models. These include feature selection and model selection. The models aim to achieve very high prediction accuracy, but the method of visualising the results should be model agnostic and therefore not depend on the type of model or the accuracy of the prediction. The goal of the dissertation does therefore not include finding the optimal or most accurate machine learning model for the data. Some background of the predictive model methods used will be given to provide algorithmic transparency.

The dissertation will compare the results using a linear and non-linear machine learning method. Using two different modelling methods will verify that the visualisation methods developed are model agnostic and will provide insight into both the local and global features of these two models.

These methods are:

- Generalised linear models (or GLMs), using the binomial family error distribution and logit link function. This results in the logistic regression model. The use of a GLM will allow for the benchmarking of the improvement in accuracy of new machine learning methods above more traditional methods.

- Gradient Boosting Machines (or GBMs). This is an example of an ensemble machine learning model. This model is traditionally seen as a “black box” model. Therefore, the proposed generalised biplots will be used to provide interpretations of how the model works and how it compares against the GLM.

The application of the generalised biplot to other machine learning methods will also be briefly discussed.

The technique will be applied to data provided by a South African insurance company. The predictive models will attempt to automate and replicate the results of the current underwriting process that was applied by insurance underwriters to individuals applying for life insurance policies.

Chapter 2 Literature Study

In recent years many published papers have discussed ways to improve the interpretability of predictive, or machine learning, models. As mentioned in the introduction, there are various ways to define interpretability. This section will review current and proposed methods for creating approximate post-hoc interpretations of model results, even if it does not improve the transparency of the underlying machine learning models.

Friedman (2001) developed variable importance and partial dependence plots. These are two instances of feature selection methods, which are very useful for giving an overall view of the impact and importance of a feature on the accuracy of a model. They provide global interpretations of how the machine learning models work and will be discussed and used later in this dissertation. However, they do not assist in providing locally faithful interpretations of individual predictions (Hastie, Tibshirani and Friedman, 2008).

To determine the impact of a specific target feature on the prediction of an individual data point, Robnik-Šikonja and Kononenko (2008) suggested analysing the difference between the individual data point prediction if the specific target feature is known or is unknown. If the model cannot calculate a prediction while ignoring the target feature, the output can be calculated as the weighted average of the prediction using each of the possible target feature values, weighted by the prior probability of the target feature having that value. This will show the contribution of the target feature to the prediction probability of an individual data point, and whether a change in the feature would cause a change in the predicted class.

This method was expanded by Strumbelj and Kononenko (2010), to address “serious errors if the feature values are conditionally dependent.” They suggested that searching all the combinations of attribute values is not feasible and developed an efficient way to allocate the overall impact of the change in all the features to each individual combination of features and therefore to each individual feature. The result shows both the magnitude and direction of the impact of the target feature on the output of a machine learning model. This method therefore also assists with “algorithmic transparency,” in addition to giving post-hoc individual data point explanations. The method may also assist in dealing with situations where data features are missing for an individual data point, but a prediction still needs to be made.

A very important area of research is concerned with explaining the result of convolutional neural networks (CNNs). CNNs have been very successfully used to classify objects in images or videos. Since CNNs and visual classification are not the focus of this dissertation, they were not researched any further.

One of the oldest methods of explaining machine learning output is case-based explanations. As explained in Caruana et al. (1999), “case-based methods explain their reasoning by presenting to users the cases in the case-base (the training set) that are most similar to a new case that needs to be explained. This allows users to assess the quality of the model's prediction by applying their own expertise to the new case while also seeing relevant empirical data from the training set.” One method is using a *k*-nearest neighbours method to identify prototype cases that serve as example cases for the specific data point. The advantage of case-based explanations is that they are model agnostic. The disadvantage is that the underlying training data or example prototypes need to be retained. This method also needs a distance measure to determine which prototypes are closest to the data point.

Along these lines, Kim, Rudin and Shah (2015) propose the Bayesian case model (BCM), which overcomes the problem of the prototype having too many features to allow easy comparison, and understanding why an instance was classified as it was. The BCM method also includes subspace selection to determine the most important features that determine similarity to the prototype feature. This simplifies the comparison and therefore makes it more interpretable to users.

Baehrens et al. (2010) propose a method to calculate a vector that characterises the flow away from the predicted class for each feature. This is presented as a gradient function. The measure changes as the data point being focussed on is changed. Setting the width of the Parzen window estimators specifies the area over which the change is measured. The results are the same for different models if they classify the individual data points in the same way. Therefore, this method does not give “algorithmic transparency”. Future work suggests allowing for the distance to a decision border.

Other papers focus on developing new prediction models that automatically provide individual data point explanations. Al-Shedivat, Dubey and Xing (2017) propose the use of simple probabilistic graphical models to create contextual explanation networks (CENs) – “a class of probabilistic models that learn to predict by constructing explanations.” Other methods include the use of interpretable decision sets, as explained in Lakkaraju, Bach and Leskovec (2016). Since these are not prediction model agnostic methods, they will not be considered in this dissertation.

Another popular approach is the use of rule-extraction techniques, which are explained and expanded on in Junque De Fortuny and Martens (2015). “Rule extraction is a technique that attempts to find a compromise between both requirements by building a simple rule set that mimics how the well-performing complex model (black box) makes its decisions. In the presented approach, rule sets are generated by a rule induction method, hereafter called the white-box technique.”

Rule based approaches allow for the prediction to be expressed as if…then expressions, m-to-n rules or fuzzy logic. It therefore attempts to combine the post-hoc explanation power of decision

trees with the accuracy (fidelity) of black box predictions. Rule-extraction techniques are often applied to support vector machines (SVM) models (Nunez, Angulo and Catala, 2006). It is a very promising area of research, especially around the use of meta-heuristics to find the decision boundaries and determine the decision rules. However, this will not be focussed on further in this dissertation.

Weng (2017) explains other methods and provides longer summaries of the current research on explaining machine learning model predictions.

An area of research that has not been touched on frequently is the visualisation of data points and decision boundaries. The easiest and most common method is applicable when there are only two input variables, and they are used to form the x- and y-axis of the plot. The data points and decision area are then overlaid on these coordinates. However, this straightforward method is not applicable when there are more than two input variables to visualise. Having multiple two-variable plots using a trellis, represents all the combinations of input variables but does not show the true interaction between these variables.

To overcome this problem, a way of representing multi-dimensional data on a single two-dimensional plane, called *biplots*, will be explored in this dissertation.

Chapter 3 PCA Biplots

3.1 Introduction to biplots

This dissertation focuses on the use of biplots to visualise the output of a prediction model. Before applying the method, an introduction to biplots is given in this chapter. Biplots are explained further in Gower, Lubbe and le Roux (2011).

Biplots can be understood as an extension of multi-dimensional scaling (MDS), which is a means of giving an optimal two-dimensional representation of multi-dimensional data. This allows the distance between the data points in two dimensions to act as a measure of similarity. The closer two plotted points are, the more similar their multi-dimensional values will be. Biplots add to this by representing the variables of the MDS as axes on the two-dimensional representation, and therefore allows the actual values of the data points in each dimension to be approximately viewed by projecting the data points onto the axes and reading off the values. The 'bi' in biplots therefore refers to representing both variables and data points at the same time.

Having all the variables shown in a single two-dimensional representation has obvious benefits to other ways of understanding how multi-dimensional data interact. The most common way of presenting multi-dimensional data is by using a trellis of multiple two-by-two scatter plots, also called scatter-plot matrices. Trellis charts were popularised by Edward Tufte, according to whom (*'A small multiple...'*, 1990): "At the heart of quantitative reasoning is a single question: Compared to what? Small multiple designs, multivariate and data bountiful, answer directly by visually enforcing comparisons of changes, of the differences among objects, of the scope of alternatives. For a wide range of problems in data presentation, small multiples are the best design solution."

The problem with this representation is that non-linear interactions over multiple dimensions often remain hidden in trellis charts. Also, if the number of dimensions is large, the number of charts that are needed becomes too high:To compare n variables would require n*-combination-2* charts.

There do exist other ways to represent the interactions in multi-dimensional data, but the ability to visually represent data via distance, colour and shapes is a very powerful method that allows a human to very easily compare and assimilate a multitude of factors at once. This motivates the desire for presenting the data visually with biplots.

An example of representing multi-dimensional data using biplots is given in Figure 3-1. The data are a sample of the height, weight, age and gender of 544 individuals (Howell, 1960). The first three graphs show a trellis representation of weight vs height, weight vs age and height vs age, with the data of males displayed in black and that of females in red.

They show a very narrow distribution for ages below 20, after which age becomes uncorrelated with weight and height.

The same representation of these three trellis graphs can be captured in a single biplot. In this biplot, each of the four variables is plotted as individual axes plotted in blue. The vertical and horizontal sides of the graphs become irrelevant and are called *scaffolding axes.*

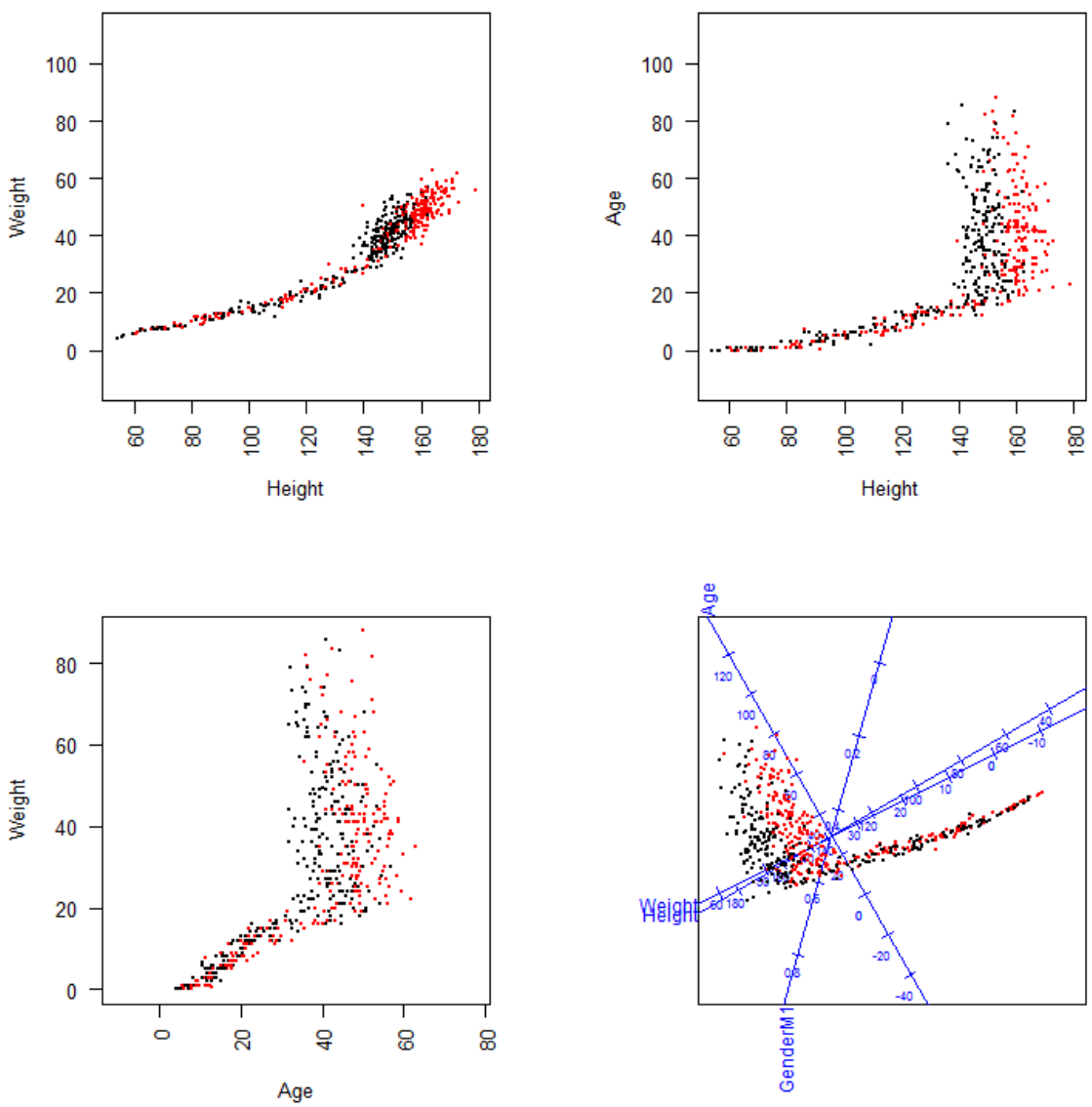


Figure 3-1: Trellis charts of age, height, weight and gender data compared to a biplot of the same data. Males are displayed in black, females in red.

The direction of increase for each of the biplot axes is indicated by the side on which the name is plotted. Therefore, moving from right to left, the weight and height increases.

The relative directions of the axes indicate correlation, with axes that increase in the same direction being positively correlated (weight and height), axes that are orthogonal being uncorrelated (weight and height compared to age) and axes that increase in the opposite direction being negatively correlated (gender and age).

For each data point, the value of its variables can be found by projecting the point orthogonally onto the respective variable axis and reading the closest axis label.

Individuals who have very similar combinations of the four variables are also plotted close together, which visualises the clusters of data points.

The biplot shows that, generally, males are taller and heavier than females, since the gender, indicated with a “1” for males, increases in roughly the same direction as height and weight. Weight and height are also very closely correlated, and the variation seen in the first graph with only weight and height can be explained by the increase in age for higher values of weight and height. Females are on average older than males. Therefore, the biplot can also be interpreted as a multi-dimensional scatterplot.

The next section explains how to construct a biplot.

3.2 The mathematics of PCA biplots

Introducing Principal Components

The basis for the construction of a biplot is the *singular value decomposition* (SVD) of the data matrix $\boldsymbol{X}: n \times p$, with n the number of rows of data and p the number of explanatory variables.

The SVD of any matrix $\boldsymbol{X}: n \times p$ can be expressed as

$$\boldsymbol{X} = \boldsymbol{UDV}' \text{ , with}$$

$$\boldsymbol{U}: n \times p \text{ an orthogonal matrix with } \boldsymbol{U}'\boldsymbol{U} = \boldsymbol{I}_p$$
$$\boldsymbol{V}: p \times p \text{ and orthonormal matrix with } \boldsymbol{V}'\boldsymbol{V} = \boldsymbol{I}_p$$

$$\boldsymbol{D}: p \times p \text{ a diagonal matrix.}$$

Equation 1: Calculating the singular value decomposition (SVD) of X

Note that $\boldsymbol{U}$ is not square. The columns of $\boldsymbol{U}$ are mutually orthogonal, but the rows are not orthogonal, so $\boldsymbol{U}'\boldsymbol{U} = \boldsymbol{I}_p$ and $\boldsymbol{U}\boldsymbol{U}' \neq \boldsymbol{I}_n$. $\boldsymbol{V}$ is a square orthogonal matrix. The columns of $\boldsymbol{V}$ are linearly independent vectors of unit length, with $\boldsymbol{V}'\boldsymbol{V} = \boldsymbol{I}_p = \boldsymbol{V}\boldsymbol{V}'$**.** The columns of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are called the left and right *singular vectors,* respectively. The matrix $\boldsymbol{D}$ is a diagonal matrix containing the *singular values* on the diagonal. It is assumed unless stated otherwise that the singular values are ordered in decreasing order. Without loss of generality, the singular values are always positive. For more information, see Harville (1997).

Linked to the idea of singular vectors and -values are *eigenvectors and -values*. For a given matrix $\boldsymbol{X}$, the singular vectors of $\boldsymbol{X}$ (given by $\boldsymbol{U}$ and $\boldsymbol{V}$) will be the same as the eigenvectors of $\boldsymbol{X}'\boldsymbol{X}$ (also given by $\boldsymbol{U}$ and $\boldsymbol{V}$)**.** In addition, if the singular values of $\boldsymbol{X}$ is given by $\boldsymbol{D}$, then the eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$ is given by $\boldsymbol{D}^2$*.* In general, if $\boldsymbol{X}: n \times n$ is a real symmetric matrix with non-negative eigenvalues, then its eigenvalues and singular values coincide.

Therefore, the SVD of the matrix $\boldsymbol{X}: n \times p$ provides a convenient way of calculating the eigenvectors and -values of $\boldsymbol{X}$**.** The rest of this dissertation will refer to the eigenvector and -value interpretation of $\boldsymbol{U}$**,** $\boldsymbol{D}$ and $\boldsymbol{V}$**.**

The *correlations* between the columns of $\boldsymbol{X}$ are given by $\boldsymbol{X}'\boldsymbol{X}$ (where $\boldsymbol{X}$ is a centred matrix).

$$\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}'\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}' = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}'$$

Equation 2: Calculating the correlation matrix $X'X$ using the SVD of X

This then extends to a key result by Hotelling (1933) that, for $\boldsymbol{Y} = \boldsymbol{XV}$, the columns of $\boldsymbol{Y}$ are uncorrelated, and the variance of each column of $\boldsymbol{Y}$ is given by the eigenvalues $\boldsymbol{D^2}$ that correspond to each eigenvector of $\boldsymbol{X'X}$.

Therefore, $cov(\boldsymbol{Y}) = \boldsymbol{V'}cov(\boldsymbol{X})\boldsymbol{V} = \boldsymbol{V'VD^2V'V} = \boldsymbol{D}^2$, which is the diagonal matrix of the eigenvalues of $\boldsymbol{X'X}$. $\boldsymbol{\Lambda}$ is used to represent $\boldsymbol{D}^2$, i.e. $\boldsymbol{\Lambda} = \boldsymbol{D}^2$.

Visually, the difference between the $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices are shown in Figure 3-2. Consider only the height and weight data, which represent matrix $\boldsymbol{X}$. The correlation between the two variables can be seen on the left side graph, where weight on average increases as height increases (or height increases as weight increases – correlation does not imply causation). Therefore, height and weight are said to be positively correlated.

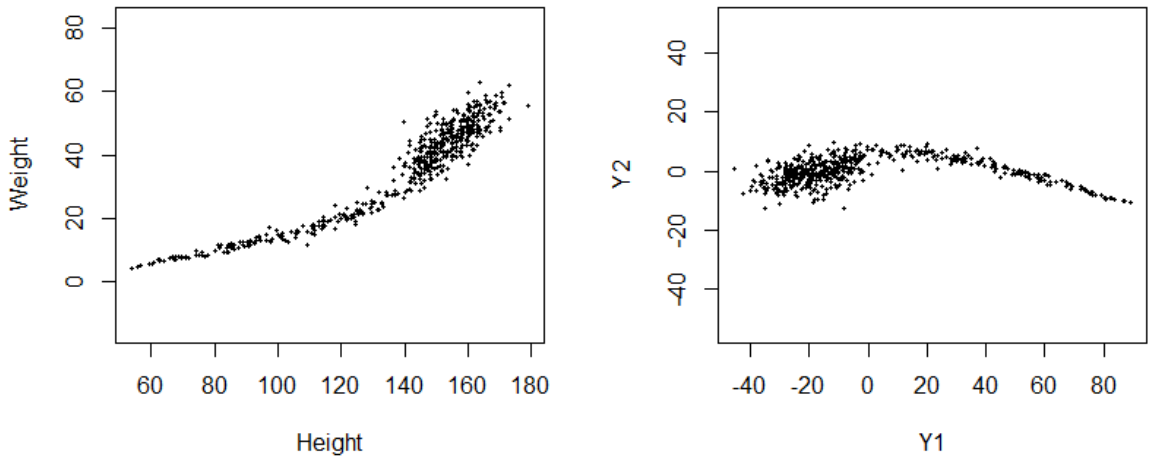


Figure 3-2: Comparing the matrix of correlated columns X with the matrix of uncorrelated columns Y

This can be compared to the plot of the two columns of $\boldsymbol{Y} = \boldsymbol{XV}$. In this case, there is no correlation between the two variables, as seen on the right-side graph. As variable $\boldsymbol{Y_2}$ increases, $\boldsymbol{Y_1}$ first increases and then decreases. But on average, the change in the values of $\boldsymbol{Y_1}$ is zero as $\boldsymbol{Y_2}$ increases. Similarly, as $\boldsymbol{Y_1}$ increases, the change in the values of $\boldsymbol{Y_2}$ is on average zero.

However, there is still variance within the data of columns $\boldsymbol{Y_1}$ and $\boldsymbol{Y_2}$. $\boldsymbol{Y_2}$ varies much less than $\boldsymbol{Y_1}$, so $\boldsymbol{Y_1}$ is said to capture the most variance in the data. This variance in shown in the size of the eigenvalues that correspond to the eigenvectors.

The difference between the $\boldsymbol{X}$ and $\boldsymbol{Y}$ matrices can further be seen by comparing their respective covariance matrices. The covariances matrix of the centred matrix $\boldsymbol{X}$ contains non-zero off-diagonal values, which indicates the covariances between the columns:

$$cov(\boldsymbol{X}) = \begin{bmatrix} 761 & 382 \\ 382 & 216 \end{bmatrix}$$

The covariance matrix of $\boldsymbol{Y}$ has only zeros for off-diagonal values, which indicates zero covariance between the columns (which are uncorrelated):

$$cov(\boldsymbol{Y}) = \boldsymbol{Y'Y} = \boldsymbol{V'X'XV} = \boldsymbol{D}^2 = \begin{bmatrix} 959 & 0 \\ 0 & 20 \end{bmatrix}$$

The eigenvalues of $\boldsymbol{X'X}$ are 959 and 20 respectively. Therefore, the first column of $\boldsymbol{Y}$ captures 959 / (959 + 20) = ~97% of the variance in the data.

Notice that the points in the left- and right-side graphs in Figure 3-2 are the same, except that the right-side is a rotation of the left-side. The relationships of the individual points are unchanged, but the columns are no longer uncorrelated. The $\boldsymbol{Y}$ values of each of the data points can be read off the $\boldsymbol{Y_1}$ and $\boldsymbol{Y_2}$ axes by projecting the points orthogonally onto the two axes. To get back to the original $\boldsymbol{X}$ values using the right-side graph, the two eigenvectors in $\boldsymbol{V}$ are added to the graph and the values are read off from these two eigenvectors. This is because:

$$\boldsymbol{X} = \boldsymbol{YV'} = \boldsymbol{XVV'} \text{ with } \boldsymbol{V} = \begin{bmatrix} -0.89 & 0.46 \\ -0.46 & -0.89 \end{bmatrix} \text{ and } \boldsymbol{VV'} = \boldsymbol{I}.$$

The successive (and ordered by the corresponding eigenvalues) columns of the matrix V are called the *principal components*. They convert the values of $\boldsymbol{Y}$ into the values of $\boldsymbol{X}$. In a regression sense, $\boldsymbol{V}$ gives the coefficients or *regression betas* ($\boldsymbol{B}$) that allow $\boldsymbol{X}$ to be estimated from the values of $\boldsymbol{Y}$.

The columns of $\boldsymbol{V}$ can be seen to be orthogonal (because $\boldsymbol{V'V} = \boldsymbol{I}$) and of unit length ($\boldsymbol{VV'} = \boldsymbol{I}$), so that, after extending their length and adding them to the right side graph of Figure 3-2, Figure 3-3 is obtained:

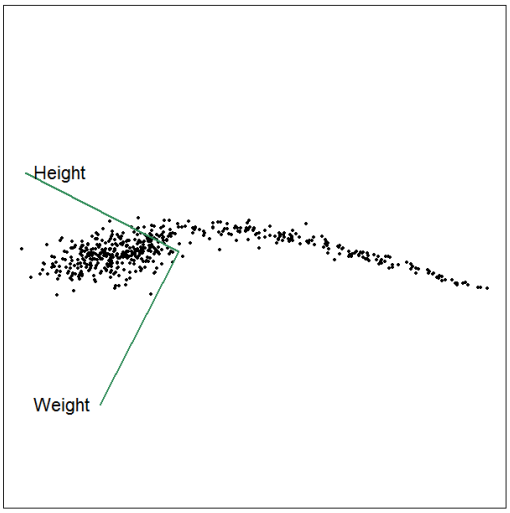


Figure 3-3: Adding the eigenvectors *V* – also called the principal component loadings – to the uncorrelated data

Figure 3-3 is therefore exactly the same as the left-side of Figure 3-2, representing the original $\boldsymbol{X}$ values. However, in this case the plot has been rotated and reflected to ensure that, in terms of the

values of $\boldsymbol{Y} = \boldsymbol{XV}$, the data points are uncorrelated and the values of $\boldsymbol{X}$ can be found by projecting the data points onto the $\boldsymbol{V}$ axis instead of $\boldsymbol{Y}$ axis. The data are also centred, and hence the data points are grouped around the origin of the axes.

For a single data point x_{ij} in $\boldsymbol{X}$, this is the same as calculating the *inner product* between the two vectors within the matrices $\boldsymbol{Y}$ and $\boldsymbol{V}$, represented by $\boldsymbol{y}_i{}'\boldsymbol{v}_j = x_{ij}$.

The two axes represented by $\boldsymbol{Y}$ now becomes irrelevant and are only needed to allow the software to plot the points. They are therefore called *scaffolding axes* and are never plotted explicitly on any biplots.

Reflection and rotation have no impact on the relative position of the data points towards each other or to the axes. By rotating the data and principal components in Figure 3-3 the same representation as the original $\boldsymbol{X}$ values is shown without any loss in accuracy, with the resulting Figure 3-4 identical to the left-side of Figure 3-2.

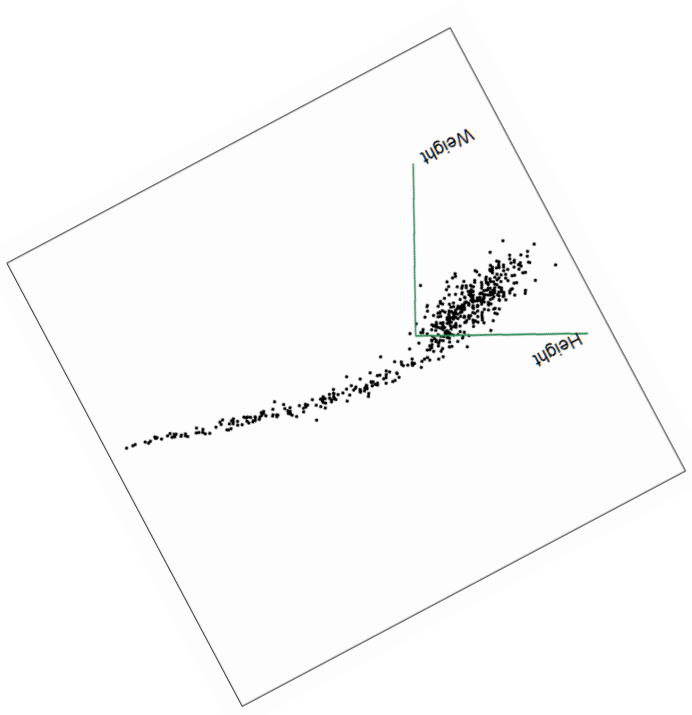


Figure 3-4: Rotated plot of the matrix *Y* with variable axes added

Principal components for dimension reduction

The previous section explained how principal components are calculated, by first introducing the singular value decomposition of $\boldsymbol{X}$, which is the same as the eigenvalue decompostion of the covariance matrix $\boldsymbol{X}'\boldsymbol{X}$ of the centred matrix $\boldsymbol{X}$. The eigenvector matrix $\boldsymbol{V}$ corresponds to the principal components of $\boldsymbol{X}$.

The principal components yield the optimal two-dimensional representation of the multi-dimensional data. This is achieved by approximating $\boldsymbol{X}$ with a two-column matrix $\widehat{\boldsymbol{X}}_{[r]}: r = 2$:

$$\widehat{\boldsymbol{X}}_{[r]} = \boldsymbol{UDJV'} = \boldsymbol{UJDV'} = \boldsymbol{UJDJV'} \text{ where } \boldsymbol{J}: p \times p = \begin{bmatrix} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} and\ r \leq p.$$

Equation 3: Calculation of r-dimensional representation of X

$\widehat{\boldsymbol{X}}_{[r]}$ is an $n \times p$ matrix, but since the last $p - r$ columns contain only zero values they can be dropped without any loss of information, resulting in the $n \times r$ matrix $\widehat{\boldsymbol{X}}_r$ of rank r.

As proved by Eckart and Young, (1936), $\widehat{\boldsymbol{X}}_{[r]}$ is an optimal approximation of $\boldsymbol{X}$ the sense that this representation of $\boldsymbol{X}$ minimizes:

$$\left\|\boldsymbol{X} - \boldsymbol{X}_{[r]}\right\| = tr\left\{(\boldsymbol{X} - \boldsymbol{X}_{[r]})(\boldsymbol{X} - \boldsymbol{X}_{[r]})'\right\}$$

This is the same as minimizing the sum of the squares between the original data and the data projected onto the biplot plane (the residuals). Therefore, the distance from the multi-dimensional data points in $\boldsymbol{X}$ to the two-dimensional data points in $\widehat{\boldsymbol{X}}_{[r]}$, is on average a minimum. The two-dimensional plane created in $\widehat{\boldsymbol{X}}_{[2]}$ does not consist of any two individual variables – it is a linear combination of the p variables in $\boldsymbol{X}$. The best combinations of variables in $\boldsymbol{X}$ needed to create $\widehat{\boldsymbol{X}}_{[r]}$ are represented by the first r principal components of $\boldsymbol{X}$. The p values in each column of $\boldsymbol{V}$ (in each principal component) indicate the factor (*loading*) that is applied to each of the p variables in $\boldsymbol{X}$ in order to calculate the new latent variables $\boldsymbol{XV'}$.

Therefore, the first r principal components give the *principal component loadings* from the columns of the matrix $\boldsymbol{VJ}$ and the *principal component scores* from $\boldsymbol{Y} = \boldsymbol{XVJ}$**.** A biplot created using the principle components is called a PCA Biplot, since it uses the principal component analysis of the data matrix to calculate the variable axes and data points.

This can also be understood in terms of creating a plane in *r* dimensions that represents the most variance in the data across all p dimensions, as summarised by the eigenvalues that correspond to each eigenvector or principal component. As a consequence of Huygens' principle (Huygens, 1657), this plane must pass through the centroid of the points in $\boldsymbol{X}$, because the sum of squares about the mean is smaller than the sum of squares about any other point. Therefore, whenever $\boldsymbol{X}$ is referred to, it implies the *centred* $\boldsymbol{X}$, with the original $\boldsymbol{X}$ replaced by:

$$\left(\boldsymbol{I} - \frac{\boldsymbol{1}}{\boldsymbol{n}}\boldsymbol{1}\boldsymbol{1}'\right)\boldsymbol{X}$$

ensuring that the centroid is at the origin. This implies that the origin of each of the axes plotted represents the mean value of the data for the variables represented.

Returning to our explanatory data set, the three-dimensional data of weight, height and age will be reduced to two dimensions and in the process the biplot shown in Figure 3-5 is obtained.

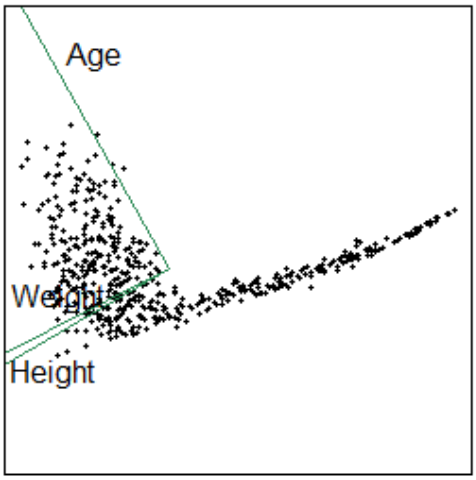


Figure 3-5: Biplot with three variables axes for the age, weight and height data

The full value of $\boldsymbol{X}$ is represented by $\boldsymbol{XVV'}$. In this case, $\boldsymbol{V}$ is a 3 x 3 matrix, with

$$\boldsymbol{V} = \begin{bmatrix} -0.77 & -0.45 & -0.44 \\ -0.40 & -0.20 & 0.90 \\ -0.49 & 0.87 & 0.02 \end{bmatrix}$$

Reducing the dimensions of $\boldsymbol{X}$ to two results in:

$\widehat{\boldsymbol{X}}_{[2]} = \boldsymbol{UDJV'}$ where $\boldsymbol{J}: 3 \times 3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $\boldsymbol{JV'}: 3 \times 3 = \begin{bmatrix} -0.77 & -0.40 & -0.49 \\ -0.45 & -0.20 & 0.87 \\ 0 & 0 & 0 \end{bmatrix}$

Again, $\boldsymbol{V'}$ gives the coefficients (loadings) that are applied to the data points $\boldsymbol{UD} = \boldsymbol{XV}$ (the scores)**.** The two values in each column of $\boldsymbol{JV'}$ give the coordinates $(\boldsymbol{x}, \boldsymbol{y})$ of the vector from the origin of each axis. Each row in $\boldsymbol{V'}$ represents a principal component with the values of the row being the principal component loadings. The number of principal components is reduced to two (and only two coordinates are used), to give a two-dimensional representation of the data.

The data points $\boldsymbol{Y} = \boldsymbol{XV}$ are still uncorrelated, with

$$cov(\boldsymbol{Y}) = \begin{bmatrix} 1210 & 0 & 0 \\ 0 & 179 & 0 \\ 0 & 0 & 20 \end{bmatrix}$$

The biplot shows how weight and height increase in the same direction. After a certain age, the relationship between age and height and age and weight breaks down. This is the same conclusion as was made from the three individual plots shown in Figure 3-1.

Accuracy of the biplot data points and axes

Using the notation explained above, the coordinate $\boldsymbol{y}'$ on the two-dimensional biplot plane from the projection of any data point $\boldsymbol{x}$ is given by $\boldsymbol{y}' = \boldsymbol{x}'\boldsymbol{V}_2$, with $\boldsymbol{JV} = \boldsymbol{V}_2$. This means that any new data point not included in the original data set can be added to the biplot plane. This process of adding new data points onto the biplot plane not included in the original matrix is called *interpolation.*

The projected two-dimensional coordinate can also be converted back to the original p-dimensional value with $\hat{\boldsymbol{x}}' = \boldsymbol{x}'\boldsymbol{V}_2\boldsymbol{V}_2{}' = \boldsymbol{y}'\boldsymbol{V}_2{}'$. This represents the orthogonal projection of the coordinate $\boldsymbol{y}'$ onto the biplot axes $\boldsymbol{V}_2{}'$. This relationship allows us to convert any coordinate on the biplot plane into the p-dimension presentative value $\hat{\boldsymbol{x}}$ by orthogonally projecting the coordinate onto the biplot axes $\boldsymbol{V}_2{}'$. Since some of the information was lost by reducing the number of principal components to two, $\hat{\boldsymbol{x}}' \neq \boldsymbol{x}'$**.** The process of converting a point on the biplot plane to the original p-dimensional space is called *prediction*.

Any form of approximation is known to result in a loss of accuracy. Therefore, although the biplot yields an optimal two-dimensional representation of multi-dimensional data, the reduction in dimensions will imply that the biplot relationship between the variables, between the data points, and between the variables and data points, will not be 100% accurate.

The *accuracy measures* of the biplot helps to explain the reliability of the representation.

The measures of fit below use the ratios of the fitted to the total *sum of squares* (s.s), with the key requirement that the decomposition is orthogonal, with $s.s\,(\boldsymbol{X}) = s.s(\hat{\boldsymbol{X}}) + s.s(\boldsymbol{X} - \hat{\boldsymbol{X}})$ – that is, the total sum of squares is equal to the fitted plus the residual sum of squares. The corrected total sum of squares of the data is given by the sum of the diagonal values of $\boldsymbol{X}'\boldsymbol{X} = trace(\boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}') = tr(\boldsymbol{\Lambda})$**.** The fitted sum of squares is given by the trace of $\hat{\boldsymbol{X}}'\hat{\boldsymbol{X}}$, with $\hat{\boldsymbol{X}} = \boldsymbol{XVJV}'$.

The goal of the biplot approximation, as stated by Eckart and Young (1936), is to minimize the residual sum of squares. If the total sum of squares cannot be orthogonally decomposed, then the ratio of fitted sum of squares to total sum of squares gives no insight into the extent to which the residual sum of squares has been minimized. Therefore, before using the accuracy measures, it is important to show that they are underpinned by the required orthogonality condition.

For a data matrix $\boldsymbol{X}$, there are two possible *types of orthogonality* arising from $\boldsymbol{X} = \hat{\boldsymbol{X}} + (\boldsymbol{X} - \hat{\boldsymbol{X}})$:

$$\text{Type A: } \boldsymbol{X}\boldsymbol{X}' = \hat{\boldsymbol{X}}\hat{\boldsymbol{X}}' + (\boldsymbol{X} - \hat{\boldsymbol{X}})(\boldsymbol{X} - \hat{\boldsymbol{X}})'$$

$$\text{Type B: } \boldsymbol{X}'\boldsymbol{X} = \hat{\boldsymbol{X}}'\hat{\boldsymbol{X}} + (\boldsymbol{X} - \hat{\boldsymbol{X}})'(\boldsymbol{X} - \hat{\boldsymbol{X}})$$

This is explained in more detail in Gardner-Lubbe, le Roux and Gower (2008) and in Appendices

Appendix A: Orthogonality of predictions.

The following accuracy measurements can be used for a biplot:

- Quality: The overall reduction in accuracy is due to the projected points not being the same as the original points. The quality measure reflects the eigenvalues included or variance accounted for, with:

$$Quality = \frac{tr(\boldsymbol{\Lambda J})}{tr(\boldsymbol{\Lambda})} = \frac{\sum_r \boldsymbol{\lambda}_j}{\sum_p \boldsymbol{\lambda}_j} = \frac{\sum_r \boldsymbol{\sigma}_j^2}{\sum_p \boldsymbol{\sigma}_j^2}$$

  By excluding the third principal component to create Figure 3-5., the quality of the biplot is calculated as (1210+179) / (1210+179+20) = 99%. This measure is independent of the orthogonality of the sum of squares decomposition and can be used for all types of biplots.

  Two other measurements of the accuracy – for the axes and the data points – require Type B and Type A orthogonality to hold. These two measures explain the variance in each variable or data point that has been accounted for in the biplot, and therefore indicate whether the sum of squares of the fitting error is small or not.

- Axes predictivity: $\boldsymbol{diag}(\widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}})\boldsymbol{I}\ [\boldsymbol{diag}(\boldsymbol{X}'\boldsymbol{X})]^{-\mathbf{1}}$, with $p \times p$ elements

  Where Type B orthogonality holds, the accuracy of the biplot axes can be measured. If a variable is orthogonal to the biplot plane, the axis predictivity will be zero – the approximation provided by the variable axis will not be reliable at all. If the variable is on the biplot plane, the axis predictivity will be one – the biplot axis will correctly represent the variable. If no eigenvector is removed then the axis predictivity for all the variables will be one. For the biplot of the age, weight and height data set, the axes predictivities are:

Height	Weight	Age
99.50%	92.80%	99.90%

  Therefore, only the weight variable has a significant reduction in accuracy when represented on the biplot.

- Data point predictivity: $\boldsymbol{diag}(\widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}')\boldsymbol{I}\ [\boldsymbol{diag}(\boldsymbol{X}\boldsymbol{X}')]^{-\mathbf{1}}$, with $n \times n$ elements

  Where Type A orthogonality holds, the accuracy of the biplot data points can be measured. This measures how far each sample is from the biplot plane, with a value of one indicating a data point that is correctly presented.

The data point with the lowest accuracy has a predictivity of 48.4%. This data point has a height value of 139 cm, a weight value of 50 kg and an age value of 38 years. For the entire data set, the averages are 138 cm, 35 kg and 30 years. He was therefore heavier than normal given his length.

In this data set, weight and height are positively correlated. If a data point therefore has a higher than average value for the weight and a lower than average (or the same) value for height, the biplot prediction will be less accurate.

The predicted values for the lowest prediction accuracy point were 145 cm, 39 kg and 38 years respectively. Therefore, the height was overpredicted and the weight underpredicted,. This is the best approximation that could be found for a data point that has values that do not correspond with the correlation structure of the overall data set.

Lastly, where Type A orthogonality holds, the accuracy of a newly interpolated data point x onto the biplot plane is measured as:

$$\widehat{\boldsymbol{x}}'\widehat{\boldsymbol{x}}(\boldsymbol{x}'\boldsymbol{x})^{-\mathbf{1}} \text{ , with } \widehat{\boldsymbol{x}}' = \boldsymbol{x}'\boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'$$

A data point that does not conform to the structure of the overall data set (i.e., has a low value for variable 1 and a high value for variable 2 where variable 1 and 2 are highly correlated) will have a low predictive accuracy. However, this will not happen often in the data since if it did the correlation structure would reflect this reality.

3.3 Improving the visual representation of a PCA biplot

The following topics are not critical to the understanding and interpretation of biplots, but show the range of ways that the visual interpretability of large data sets can be improved.

Showing accurate axis correlation of data point values

The ability to display the correlation between variables from the SVD was the original idea behind biplots, as introduced by Gabriel (1971). As stated by Gower and Harding (1988), three things may be approximated in a biplot:

- The data point value x_{ij} of the inner-product
- The Euclidean distances between the data points
- The covariances between the variables in the data

Variant forms of biplots allow the simultaneous approximation of any two of these three, but all three cannot be achieved optimally in one diagram. The difference between optimally presenting either Euclidean distances or the covariances between the variables is explained below:

Any matrix $\widehat{\boldsymbol{X}}_{[r]}$ of rank r can be written as the product of two matrices:

$$\underset{n \times p}{\widehat{\boldsymbol{X}}_{[r]}} = \underset{n \times r}{\boldsymbol{G}} \quad \underset{r \times p}{\boldsymbol{H}'}$$

An r-dimensional biplot is produced by simultaneously plotting the rows of $\boldsymbol{G}$ and the rows of $\boldsymbol{H}$ on the same set of scaffolding axes. By tracing lines from the origin to the points in $\boldsymbol{H}$**,** the variable axes are created. As explained above, $\boldsymbol{X}_{[r]}$ is represented by:

$$\widehat{\boldsymbol{X}}_{[r]} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{J}^*\boldsymbol{V}' = \boldsymbol{U}\boldsymbol{J}^*\boldsymbol{D}\boldsymbol{V}' = \boldsymbol{U}\boldsymbol{J}^*\boldsymbol{D}\boldsymbol{J}^*\boldsymbol{V}' \text{ , with } \boldsymbol{J}^* = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{0}: (p-r) \times r \end{bmatrix}.$$

This means that the biplot can be constructed from two possible decompositions:

Decomposition A

$$\widehat{\boldsymbol{X}}_{[r]} = \begin{matrix} \boldsymbol{G} & \boldsymbol{H}' \\ \boldsymbol{U}\boldsymbol{D}\boldsymbol{J}^* & (\boldsymbol{V}\boldsymbol{J}^*)' \end{matrix}$$

or *Decomposition B*

$$\widehat{\boldsymbol{X}}_{[r]} = \begin{matrix} \boldsymbol{G} & \boldsymbol{H}' \\ \boldsymbol{U}\boldsymbol{J}^* & (\boldsymbol{V}\boldsymbol{D}\boldsymbol{J}^*)' \end{matrix}$$

The difference is in the allocation of the diagonal entries of the matrix $\boldsymbol{D}$. In the example shown in Figure 3-5 above, the matrix $\boldsymbol{D}$ was allocated to the matrix $\boldsymbol{U}$, yielding Decomposition A. However, the matrix $\boldsymbol{D}$ could equally be allocated to the matrix $\boldsymbol{V}$.

If Decomposition A is followed, then:

$\widehat{\boldsymbol{X}}_{[r]}\widehat{\boldsymbol{X}}'_{[r]}$ can be simplified to $\boldsymbol{GG}'$, since $\boldsymbol{H}'\boldsymbol{H} = (\boldsymbol{VJ}^*)'(\boldsymbol{VJ}^*) = \boldsymbol{I}_r$.

However, $\widehat{\boldsymbol{X}}'_{[r]}\widehat{\boldsymbol{X}}_{[r]}$ cannot be simplified, since the matrix $\boldsymbol{D}$ implies $\boldsymbol{G}'\boldsymbol{G} \neq \boldsymbol{I}_r$.

Since $\widehat{\boldsymbol{X}}_{[r]}\widehat{\boldsymbol{X}}'_{[r]}$ measures the interpoint distances between the rows of the data, this decomposition of $\widehat{\boldsymbol{X}}_{[r]}$ optimally approximates the distances between the rows of $\boldsymbol{X}$, with

$$\left\|\boldsymbol{x}_{(i)} - \boldsymbol{x}_{(j)}\right\| \cong \left\|\widehat{\boldsymbol{x}}_{(i)} - \widehat{\boldsymbol{x}}_{(j)}\right\| = \left\|\boldsymbol{g}_{(i)} - \boldsymbol{g}_{(j)}\right\|$$

If Decomposition B is followed, then:

$\widehat{\boldsymbol{X}}'_{[r]}\widehat{\boldsymbol{X}}_{[r]}$ can be simplified to $\boldsymbol{HH}'$, since $\boldsymbol{G}'\boldsymbol{G} = (\boldsymbol{UJ}^*)'(\boldsymbol{UJ}^*) = \boldsymbol{I}_r$.

However, $\widehat{\boldsymbol{X}}_{[r]}\widehat{\boldsymbol{X}}'_{[r]}$ cannot be simplified, since the matrix $\boldsymbol{D}$ implies $\boldsymbol{H}'\boldsymbol{H} \neq \boldsymbol{I}_r$.

Since $\widehat{\boldsymbol{X}}'_{[r]}\widehat{\boldsymbol{X}}_{[r]}$ measures the inter-variable covariance structure, this decomposition of $\widehat{\boldsymbol{X}}_{[r]}$ optimally approximates the correlations between the columns of $\boldsymbol{X}$, with

$$\boldsymbol{r}_{ij} = \frac{\boldsymbol{x}'_i\boldsymbol{x}_j}{\sqrt{\boldsymbol{x}'_i\boldsymbol{x}_i}\sqrt{\boldsymbol{x}'_j\boldsymbol{x}_j}} = \frac{\widehat{\boldsymbol{x}}'_i\widehat{\boldsymbol{x}}_j}{\sqrt{\widehat{\boldsymbol{x}}'_i\widehat{\boldsymbol{x}}_i}\sqrt{\widehat{\boldsymbol{x}}'_j\widehat{\boldsymbol{x}}_j}} = \cos\langle\boldsymbol{h}_{(i)}, \boldsymbol{h}_{(j)}\rangle$$

The two possible biplots representations, depending on allocating the matrix $\boldsymbol{D}$ to either the matrix $\boldsymbol{U}$ or $\boldsymbol{V}$, are compared in Figure 3-6. They are often referred to as *Form Biplots* and *Correlation Biplots* (Greenacre, 2010).

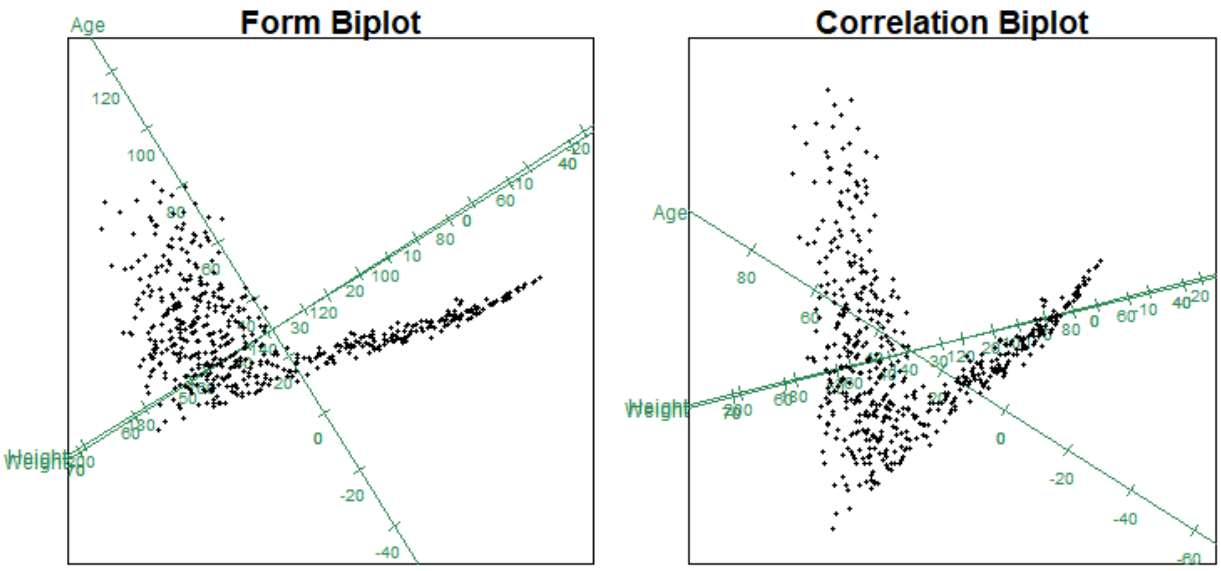


Figure 3-6: The form and correlation biplots for the age, weight and height data

The graph on the left-hand side is the same as before. In this case the age axis is perpendicular to the height and weight axes, which would imply that age is uncorrelated from height and weight. This is incorrect, however, since the data points accurately show that most people with a height and weight less than 140cm and 35kg are younger than 20 years.

In the graph on the right-hand side, the age axis is at a very close angle to the height and weight axes, which would imply that age is positively correlated with height and weight, which is correct and can be verified when compared to:

$$cor(\boldsymbol{X}) = \begin{matrix} \\ Height \\ Weight \\ Age \end{matrix} \begin{matrix} \begin{matrix} Height & Weight & Age \end{matrix} \\ \begin{bmatrix} 1 & 0.94 & 0.68 \\ 0.94 & 1 & 0.67 \\ 0.68 & 0.67 & 1 \end{bmatrix} \end{matrix}$$

However, the distribution of the data points themselves is only approximately correct and shows that some people with height and weight less than 140cm and 35kg respectively are older than 40 years, which is not a true reflection of the underlying data.

The overall interpretation remains the same between the two graphs. The entries in the matrix $\boldsymbol{X}$ are determined from the axes by orthogonally projecting each point onto one of the biplot axes. The correlations between the variables in the matrix $\boldsymbol{X}$ can be estimated from the angles between the respective axes. The only difference is that different aspects are shown with different levels of accuracy, namely the distribution of and distances between the data points or the correlation between the variables.

Axes markers and normalising the data

Axes markers

The variable value of an individual data point is determined by orthogonally projecting that data point onto the variable axis and calculating the length of the variable axis up to that point. This can be verified by combining two aspects of the biplot:

1) The variable value (say variable number 1) of a data point value (say data point number 2) in the data matrix $\boldsymbol{X}$ is equal to the scalar product of the two corresponding vectors ($\boldsymbol{y'}_2\boldsymbol{v'_1}$) of the matrix into which $\boldsymbol{X}$ has been decomposed, namely, $\boldsymbol{Y} = \boldsymbol{XV}$ and $\boldsymbol{V'}$

2) The scalar product of the two corresponding vectors is equal to the length of the projection of the first vector onto the second (the length of the point representing $\boldsymbol{y}_2$ on the biplot, projected onto variable axis $\boldsymbol{v'_1}$), multiplied by the length of the second variable:

$$\boldsymbol{y'}_2\boldsymbol{v'_1} = \|\boldsymbol{y}_2\|.\|\boldsymbol{v'_1}\|.\cos\theta$$

Since the viewer of the biplot will not be able to easily calculate the length of the data point, the length of the variable axis, or the angle between the two, it is not intuitively possible for the viewer to calculate the original variable value of that data point. To facilitate reading the values, the biplot axes have *markers* indicated on them that are calibrated such that the variable value of a data point can be read off by simply orthogonally projecting the data point on the variable axes.

Greenacre (2010) explains this in more detail and describes how the markers are calibrated. Depending on the calculation of the markers, the axes are called either *predictive* or *interpolation* axes. Predictive axes are calibrated such that the variable values of any point on the biplot plane can be accurately measured by projecting that point onto the variable axes. Interpolative axes are calibrated such that a new data point can be accurately placed onto the biplot plane relatively to the other existing data points. This dissertation only makes use of predictive axes.

Normalising the data

Before plotting data in a biplot, it needs to be determined whether the various variables are on commensurable scales and how the distances between data points on the biplot plane will be interpreted.

Figure 3-7 shows two bivariable biplots (and therefore scatterplots) of three data points with their height and weight values indicated on the axes. Based on the left-side graph in Figure 3-7, data point 1 is just as similar to data point 3 as data point 2 is to data point 3, due to the fact that the Euclidean distance between data points 1 and 3 is similar to that between data points 2 and 3.

In the data, the height variable is on average 138 units (in cm) and the weight variable is 35 units (in kg). However, one standard deviation in height is 27 units and in weight is 15 units. Therefore, the 10 kg weight difference between points 2 and 3 results in a much bigger difference in terms of multiples of standard deviations than the 10 cm height difference between points 1 and 3. To reflect this difference, point 2 should in fact be plotted closer to point 3 than the plot of point 1 to point 3.

This can be accomplished by *normalising*[3] the data before plotting it, by dividing each column of the data by the standard deviation of the data. Plotting the normalised data and adjusting the markers on the x and y axes to reflect the new normalised data scale results in the display on the right-side graph in Figure 3-7.

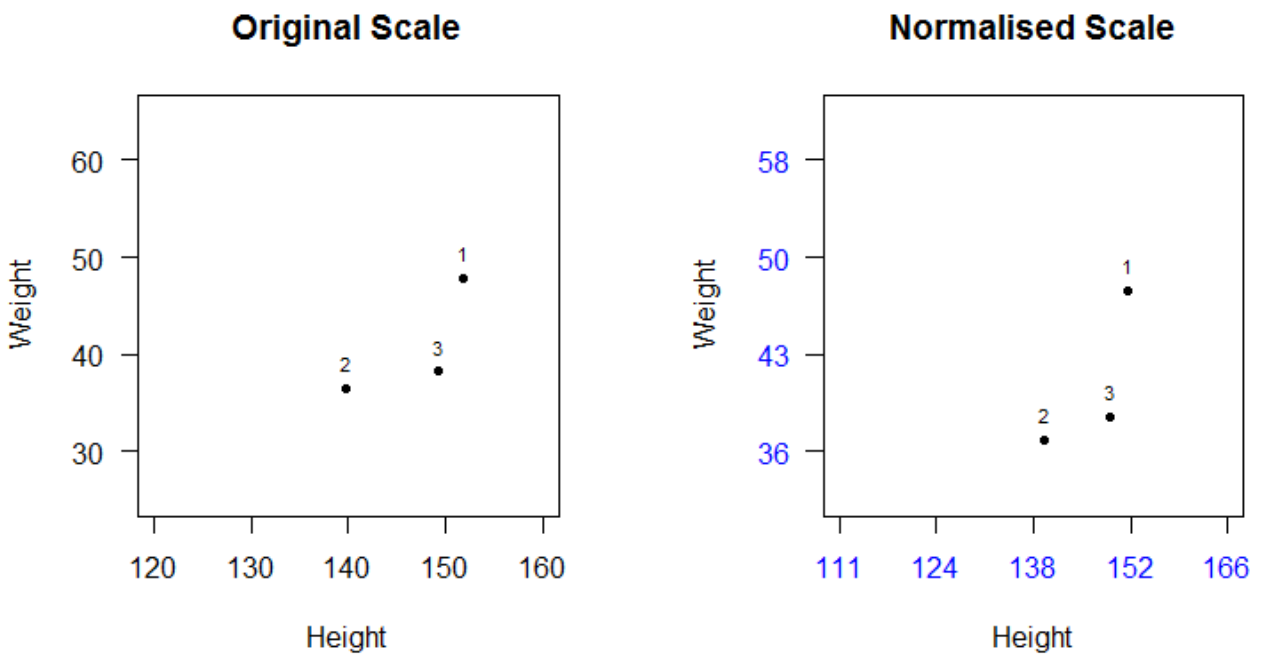


Figure 3-7: Plot of the same data points before and after normalising the data points by the variance of each variable

Normalising of the data is needed because PCA biplots use Euclidean distance to measure the dissimilarity between data. Euclidean distance is sensitive to the relative scale of the individual components of the distance and overweights the contribution of variables that are measured in a larger scale. After normalising the data, the height and weight values are said to be on

[3] Normalising can also refer to dividing the vector by its length: $\boldsymbol{x}^* = \boldsymbol{x} / ||\boldsymbol{x}||$.

commensurable scales and their representation in Euclidean distance is not distorted by the relative sizes of the variable scales.

After centring and normalising, the correlation and covariance matrices of $\boldsymbol{X}$ will be the same, and $diag(\boldsymbol{X}'\boldsymbol{X}) = (\boldsymbol{n} - \boldsymbol{1}) \times \boldsymbol{I}$

The same adjustment needs to be made for all biplots. Certain biplot types (for example the CVA biplots) are *scale invariant* since they automatically adjust the data to reflect their normalised distances. Other biplots do not use Euclidean distance between the data points to measure similarity and are therefore not distorted by the scale of the variables. This will be discussed in later sections.

There are many possible ways of *standardising* data, which refers to the processing of transforming variables to ensure they are measured on similar scales and therefore are commensurate. Normalising is a specific way of standardising data, by dividing each column by its standard deviation. If not otherwise stated, the term standardised will imply normalised in this dissertation. Alternative methods include:

- *Unit range*, or *interval scaling*, where each column is divided by the range of the variable, resulting in a rescaled variable that has a range between 0 and 1

- Unit sum of squares, where each column is divided by the sum of squares ($\sqrt{(n-1)} \times s$, with s the sample standard deviation of the column data), and each variable has a sum of squares equal to the number of data points

For biplots it is important to remember that the data should always first be centred to create the optimal two-dimensional representation before it is standardised to ensure that the scales of the different variables are commensurable. The markers on the axes should then be adjusted to ensure that the correct value is still read when projecting a data point onto the axes and reading off its value.

In the case of the age, height and weight data, the impact of scaling on the representation is not significant, due to the fact that the scales of the different variables are similar before normalisation, as shown in Figure 3-8.

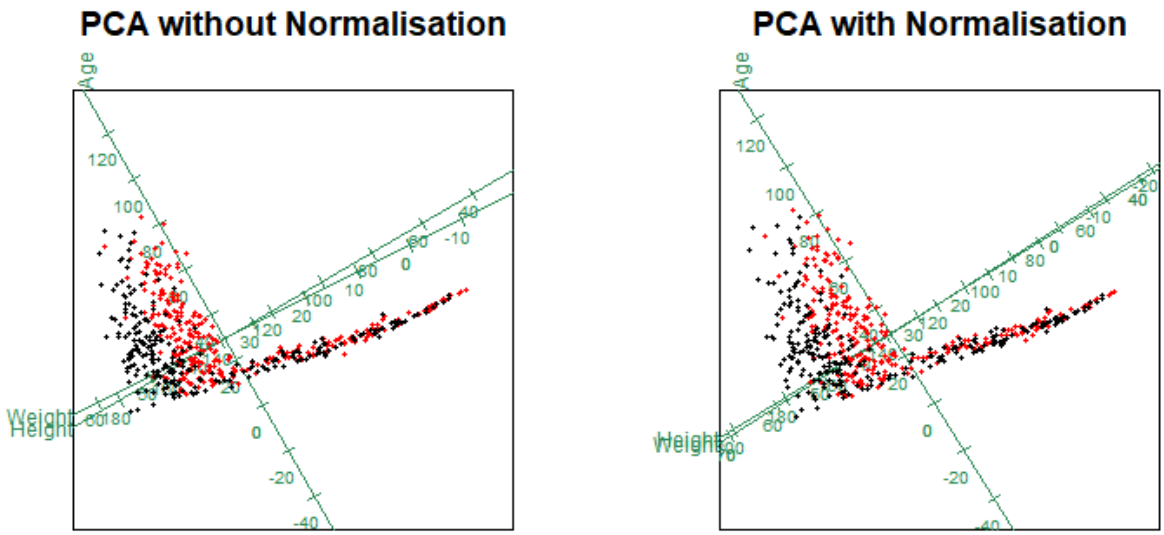


Figure 3-8: PCA biplots before and after normalisation of the data

Other two-dimensional biplots and one- or three-dimensional biplots

Two-dimensional biplots can be represented in different ways to improve the visualisation of the results. This is done by using two different eigenvectors (other than the first two) to create the scaffolding axes $\boldsymbol{V}'$.

The optimal two-dimensional representation is created using the first two eigenvectors, but using the second and third eigenvector, or using the first and third eigenvector, will yield an alternative view that may provide additional insights into the data. This is however at the expense of a lower accuracy of the presentation.

Biplots can also be improved by adding a categorical colour classification to the different groups of data points, allowing easy visual inspection of the groups. An example with the male and female data points shown in green and blue respectively is shown in Figure 3-9.

Another alternative representation is obtained by using three eigenvectors to create three-dimensional scaffolding axes for representing the data, shown on the right-side of Figure 3-9. (Here the three-dimensional figure has no variable axes, so is technically only a three-dimensional MDS plot.)

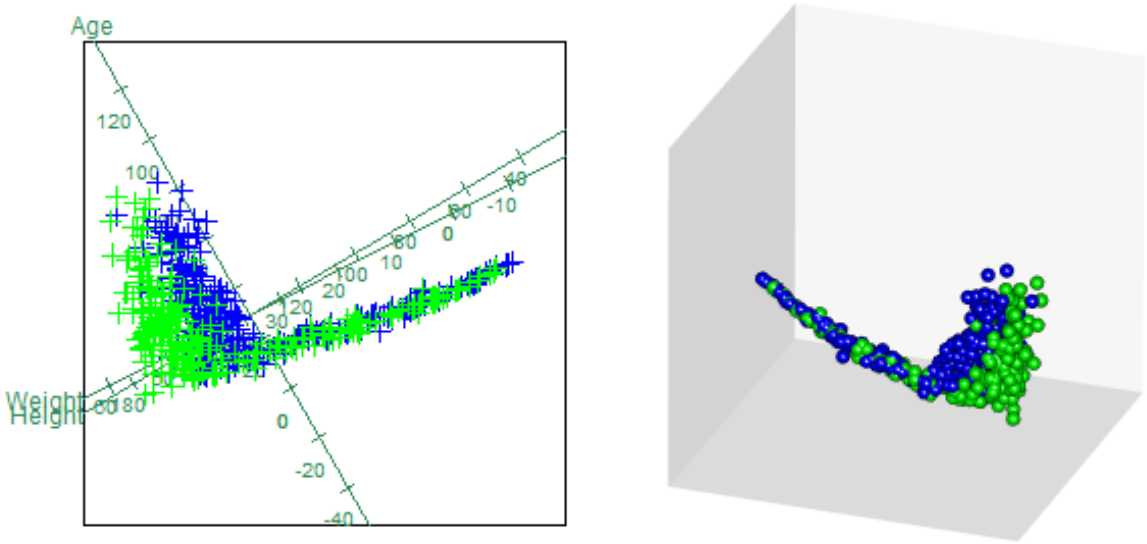


Figure 3-9: Two-dimensional biplot and three-dimensional MDS plot with males and females separated by green and blue colour classification

The additional eigenvector increases the accuracy of the biplot. It also better illustrates the interaction of data points and variables. However, three-dimensional data plotted on a two-dimensional screen or piece of paper is always less easily understood and results in more potential misinterpretations compared to a two-dimensional plot.

One-dimensional biplots allow for very easy projection onto the variable axes. However, this is at the expense of much lower biplot accuracy, and does not show the correlation between the variables. An example is shown in Figure 3-10.

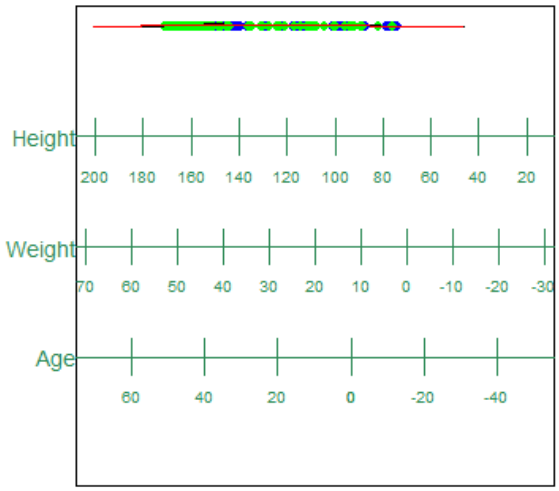


Figure 3-10: One-dimensional biplot with males and females separated by green and blue colour classification

This dissertation will always make use of two-dimensional biplots using the first two eigenvectors only.

Improving interpretability of the biplot

The biplots shown thus far work well for representing the individual data points and how they relate to their variables. A problem with the visualisation occurs when there are either too many data points or too many variables for the representation of the entire data set at once. The data points will form a single blob, with different categories of data points being plotted over each other and obscuring the visualisation of the structure of the different groups. The variable biplot axes will obscure each other and cause their axes' labels to be unreadable.

Better ways of representing the data points and axes in large data sets are discussed below:

- Spanning and Concentration ellipses

  A spanning ellipse is defined as the smallest ellipse that covers all selected data points. This ellipse is very sensitive to outliers, which can increase the size of the ellipse unnecessarily and cause the shape to not be representative of the true underlying structure of the data.

  To overcome this, *concentration ellipses* can be drawn instead, which ensure that a specified percentage of the data points will be enclosed by the ellipse. This ellipse assumes that the data points are distributed normally with the mean and covariance based on the data presented on the two-dimensional biplot plane. The ellipse will then be centred at the mean, and the angle and shape are determined by the covariance matrix.

- Convex hulls

  Although the shape of a concentration ellipse is less sensitive to outliers in the data than a spanning ellipse, it can still only show the general shape of the data points. A *convex hull* connects the smallest convex set of points around all the data points and more accurately reflects the shape of the data. It is still sensitive to outliers, and therefore peeling is used to yield smaller convex hulls, by removing all the data points that form the outer convex hull and fitting a new convex hull on the remaining data points.

- Bagplots and alpha bags

  One problem with a convex hull is that it ignores the distribution of the data points within the convex hull, since the shape is only determined by the outer points. One way to visualise the density and bivariate data is through box plots and bagplots. *Bagplots* are created to contain 50% of the data points, being inflated outward from the centre of the data set.

Alpha bags are similar, where the size of the bag is set to include α% of the data.

The shape of an alpha bag is not smooth and will incrementally expand to include an additional point until the specified percentage of the data is included. Therefore, alpha bags capture the distribution of the data and are less influenced by outliers, while still only using a single fence line, as illustrated in Figure 3-11, which clearly shows the higher average weight and height for males vs females in the data set, independent of age. By not plotting the individual data points, this association is more clearly illustrated.

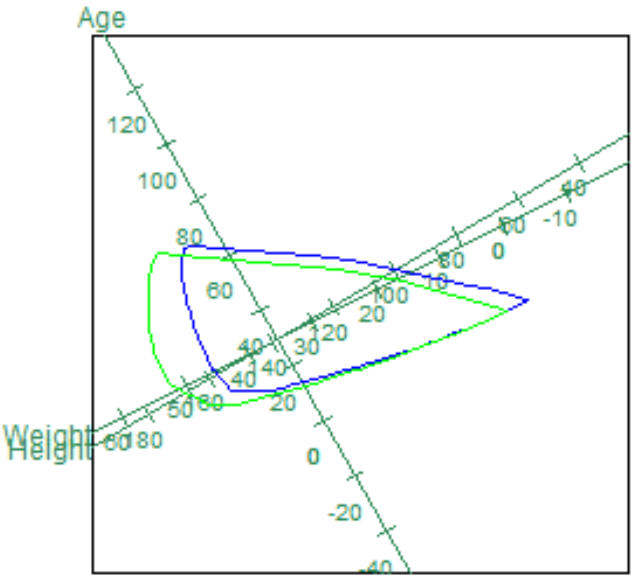


Figure 3-11: Biplot with 50% alpha bags around the male and female data

- Bivariate density plots

The next method considered is *bivariate density plots*, which are shown in Figure 3-12. This method represents the data as a cloud of deepening colour luminance as the data points become denser, in contrast with the previous methods that only have a single fence line to show the data shape.

Bivariate density plots are not influenced by outliers and can accurately reflect data with different areas of concentration. However, it uses another colour channel with makes visualising other features of the data with colour difficult.

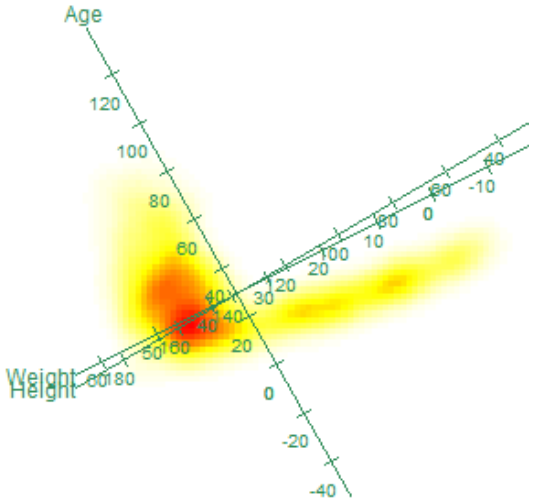


Figure 3-12: Bivariate density biplot

It there are too many axes on the biplot, it can be dealt with by simply hiding the projection of subsets of axes. Therefore, axes used in the construction of the biplot but are either not the focus of the analysis or are not significant for understanding the difference between groups of data, are simply not shown.

For more information on how to more clearly represent the data set on a biplot, please refer to Gower, Lubbe and le Roux (2011).

3.4 Conclusion to PCA biplots

This chapter introduced the theoretical background of biplots and considered practical issues for improving the visualisation.

In summary, the PCA biplot allows for:

- Plotting of data points to reflect the Euclidean distances between them, to see which data points are similar or different across multiple dimensions

- Plotting of more than three continuous variables on the same graph to show correlation between the variables and explain which variables cause the difference between data points

- Projection of the data points onto the variable axes to approximate the original values of the data and measure the accuracy of the biplot projection.

The traditional PCA biplots need to be extended in order to represent distance measures other than Euclidean distance and to allow for categorical data. This will be the focus of the following chapters.

Chapter 4 Multi-Dimensional Scaling and Biplots

4.1 Introduction to multi-dimensional scaling

Biplots are in many ways merely a specific implementation of *multi-dimensional scaling* (MDS). The main difference between biplots and MDS is that biplots also add biplot axes to the plot, which allow the variable values of each of the individual data points to be read off the axes. The other difference is that MDS uses an explicit underlying distance calculation, whereas with traditional biplots the distance calculation is implicit in the PCA calculation.

This chapter explains the connection between biplots and MDS, and focuses on combining the distance measure feature of MDS with the variable axes of biplots to allow for the creation of alternative types of biplots that uses different underlying distance measures.

Brief introduction to MDS

Given a set of objects $\boldsymbol{X}$: $n \times p$, let $\boldsymbol{D}$: $n \times n$ denote the pairwise dissimilarities between each pair of objects. The goal of MDS is to find coordinates (an ordination) in r dimensions $\boldsymbol{Y}$: $n \times r$, such that the Euclidean distances between the n coordiantes, denoted by $\boldsymbol{\Delta}$: $n \times n$, optimally approximates $\boldsymbol{D}$: $n \times n$. If $r = 2$, then the plotted coordinates $\boldsymbol{Y}$ will provide a visual representation of the similarity or dissimilarity of the n different data points in $\boldsymbol{X}$ (Groenen and Van De Velden, 2016).

Therefore, MDS consists of two steps:

1) Calculate the d*istance matrix* $\boldsymbol{D}$, (with entries d_{ij}) from the data matrix $\boldsymbol{X}$: $n \times p$, using a measure of dissimilarity[4]. The matrix $\boldsymbol{D}$ can be provided directly, in which case $\boldsymbol{X}$ is not needed.

2) Calculate the coordinates $\boldsymbol{Y}$, with a distance matrix between the coordinates $\boldsymbol{\Delta}$: $n \times n$ (with entries $\boldsymbol{\delta_{ij}}$), such that it minimizes the stress:

$$Stress = \sum_{k=1}^{P} \left(\boldsymbol{d}_{ik} - \boldsymbol{\delta}_{jk}\right)^2$$

This is called *metric least squares MDS* and was first proposed by Kruskal (Kruskal, 1964). *Non-metric MDS* is not discussed in this dissertation but refers to minimizing the difference between $\boldsymbol{\delta}_{jk}$ and a monotone transformation of $\boldsymbol{d}_{ik}$, denoted by $\boldsymbol{f}\left(\boldsymbol{d}_{ij}\right)$.

[4] The distance measure should be an additive Euclidean embeddable measure - this will be defined later

This process allows the information in $\boldsymbol{X}$ to be represented in the map of $\boldsymbol{Y}$, with $\boldsymbol{\Delta}$ (generated by$\boldsymbol{Y}$) approximating $\boldsymbol{D}$ (generated by $\boldsymbol{X}$).

Different metric MDS methods can be used to calculate the coordinates $\boldsymbol{Y}$ that approximate $\boldsymbol{D}$. One is the SMACOF method (de Leeuw and Mair, 2011). However, this dissertation will focus on *Classical Scaling*, as proposed by Gower (1966).

Metric MDS and classical scaling

One method of calculating $\boldsymbol{Y}$ is by calculating:

$$\boldsymbol{B} = (\boldsymbol{I} - \boldsymbol{1s}')\boldsymbol{D}^*(\boldsymbol{I} - \boldsymbol{s1}') \text{ , with } \boldsymbol{D}^* = \left\{-\frac{1}{2}d_{ij}{}^2\right\} \text{ and } \boldsymbol{s}'\boldsymbol{1} = 1$$

If $\boldsymbol{B}$ is positive semi-definite it can be expressed as $\boldsymbol{B} = \boldsymbol{YY}'$ and the rows of $\boldsymbol{Y}$ provide the MDS coordinates, with $\boldsymbol{Y}$ centred at the origin. For the proof and further discussion on the necessary and sufficient conditions, see Gower and Harding (1988). A necessary and sufficient condition for the matrix $\boldsymbol{B}$ to be positive semi-definite is for the distance measure used to calculate $\boldsymbol{D}$ (and therefore $\boldsymbol{B}$) to be so-called *Euclidean embeddable*. If the distance measure is not Euclidean embeddable and $\boldsymbol{B}$ therefore not positive semi-definite, an optimal approximation of $\boldsymbol{D}$ by $\boldsymbol{Y}$ is not possible.

$\boldsymbol{D}^*$ is a matrix derived from the pairwise distances of all n samples with the ij-th element $\left\{-\frac{1}{2}\boldsymbol{d}_{ij}{}^2\right\}$. These quantities are termed *ddistances*.

A distance measure is called *additive* if the contribution of each variable to the distance between two points is calculated independently, that is, adding another variable to the distance calculation will not change the contribution of the existing variables.

Classical scaling solves $\boldsymbol{B} = \boldsymbol{YY}'$ by calculating the SVD of $\boldsymbol{B} = \boldsymbol{V\Lambda V}'$ and setting $\boldsymbol{Y} = \boldsymbol{V\Lambda}^{\frac{1}{2}}$, the eigenvectors and eigenvalues of the symmetric matrix $\boldsymbol{B}$. The two-dimensional MDS will then be created by the first two columns of $\boldsymbol{Y}$. Therefore, the first two columns of $\boldsymbol{Y}$ are represented by:

$$\boldsymbol{Z}_{n\times 2} = \boldsymbol{V\Lambda}^{\frac{1}{2}}\boldsymbol{J}, \text{ with } \boldsymbol{J}: p \times p = \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \text{ and } 2 \leq p$$

Equation 4: Calculation of the MDS coordinates using classical scaling

A positive semi-definite matrix $\boldsymbol{B}$ - therefore a Euclidean embeddable distance matrix $\boldsymbol{D}$ - implies that all the eigenvalues $\boldsymbol{\Lambda}$ are greater than or equal to zero (otherwise the square root of $\boldsymbol{\Lambda}$ will not give real values). If this is not the case, then one way to still use classical scaling to find $\boldsymbol{Y}$ is to drop the eigenvectors and -values that correspond to the negative values in $\boldsymbol{\Lambda}$.

Then $\boldsymbol{Y} = \boldsymbol{V}_d \boldsymbol{\Lambda}_d^{\frac{1}{2}}$, with d $(d < p)$ representing the number of columns with non-negative eigenvalues. This implies that classical scaling can be used with any distance measure to give two-dimensional MDS results. The accuracy of this approximation may just be too low to be useful. For more information see Cox and Cox (2000).

Distance measures

The three measures of distance that will be used in this chapter to calculate the matrix $\boldsymbol{D}$ from $\boldsymbol{X}$ are summarised below. All three of these measures are Euclidean embeddable.

Euclidean distance between two data points i and j over p variables is measured as:

$$\boldsymbol{d}_{ij} = \left[\sum_{k=1}^{P} \left(\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}\right)^2\right]^{1/2}$$

Euclidean distance is the most intuitive way to measure the multi-dimensional distance between two data points. (Euclidean distance is called Pythagorean distance when measured over two dimensions). As explained later, classical scaling using Euclidean distance will yield the same results as PCA biplots.

Euclidean distance is sensitive to the scale of the variables and overweigh the contribution of the variables measured in larger scales, that have larger standard deviations or have large outliers.

Square root of Manhattan distance:

$$\boldsymbol{d}_{ij} = \left[\sum_{k=1}^{P} \left|\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}\right|\right]^{1/2}$$

The main benefit of the square root of Manhattan distance[5] is that it is less sensitive to the impact of outliers in the data. However, if there is sufficient data and the data is standardised in some way, the benefits become less obvious. When referring to Manhattan distance, the square root distance is implied.

Mahalanobis distance:

$\boldsymbol{d}_{ij} = \left(\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right)^{\frac{1}{2}}$, with $\boldsymbol{\Sigma}$ the covariance matrix between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Mahalanobis distance between two points takes into account the variance within each variable used to measure the distance. If a variable has a high variance, then the probability of the distance between two points along that variable being far apart is low. By dividing the distance along each variable by the standard deviation of that variable, the data is standardised for the variance in the data and therefore the probability of the point being far away along each variable is incorporated in calculating the distance. This reduces the overlap of data points along high variance dimensions. If

[5] Manhattan distance is not an Euclidean embeddable metric but the square root of Manhattan distance, also known as L1-norm, is

the variance is measured using the *within-group covariance matrix*, then it will result in better separation between the group means.

The terms 'distance' and 'dissimilarity' are used equivalently. Additional types of dissimilarities can be measured for categorical or ordinal variables and will be considered in later chapters when biplots are adjusted for categorical or ordinal variables.

Classical scaling and MDS biplots

If the measure of distance used to calculate $\boldsymbol{D}$ from $\boldsymbol{X}$ is Euclidean, then the coordinates $\boldsymbol{Y}$ created by classical scaling will be equal to the principal coordinate scores $\boldsymbol{XV}$ created by the PCA of $\boldsymbol{X}$. This is the reason why classical scaling with Euclidean distance is also known as *principal coordinate analysis*, or PCO. This can be shown as follows:

If $\boldsymbol{D}$ measures the Euclidean distance of the matrix $\boldsymbol{X}$ (with $\boldsymbol{X}$ already centred), then

$$\boldsymbol{D}^* = -\frac{\mathbf{1}}{\mathbf{2}}\left(\mathbf{11}'\boldsymbol{diag}(\boldsymbol{X}'\boldsymbol{X}) + \boldsymbol{diag}(\boldsymbol{X}'\boldsymbol{X})\mathbf{11}' - \mathbf{2}\boldsymbol{X}'\boldsymbol{X}\right) \text{ and } \boldsymbol{D}^* = \boldsymbol{B}$$

$\boldsymbol{V}$ is the matrix of eigenvectors and $\boldsymbol{S}^2$ the eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$ since $(\boldsymbol{X}'\boldsymbol{X})\boldsymbol{V} = \boldsymbol{V}\boldsymbol{S}^2$ (with $\bar{\boldsymbol{x}} = \mathbf{0}$). Then $\boldsymbol{V}'(\boldsymbol{X}'\boldsymbol{X})\boldsymbol{V} = \boldsymbol{S}^2$, with $\boldsymbol{S}^2 = \boldsymbol{\Lambda}$. Therefore, $(\boldsymbol{XV})'(\boldsymbol{XV}) = \boldsymbol{\Lambda}$. If in the classical scaling solution the matrix $\boldsymbol{Y}$ is scaled such that $\boldsymbol{Y}'\boldsymbol{Y} = \boldsymbol{\Lambda}$, then $\boldsymbol{Y} = \boldsymbol{XV}$ and both represent the principal coordinate scores.

To illustrate the impact of different distance measures, the three groups from the well-known *Iris data set* is visualised in Figure 4-1. The Iris data (Becker, Chambers and Wilks, 1988) set is used instead of the gender, age, height and weight data since for Mahalanobis distance a minimum of three groups are needed to separate the data in two dimensions, as will be explained later. Therefore, to show the difference between the different distance measures requires a data set with at least three groups.

Figure 4-1 first shows the results using the PCA biplot method and then using the MDS classical scaling method with Euclidean, square root Manhattan and Mahalanobis distance measurements. The PCA, Euclidean, and Manhattan data are first standardised. The Mahalanobis data do not need to be standardised, since the method is scale-invariant.

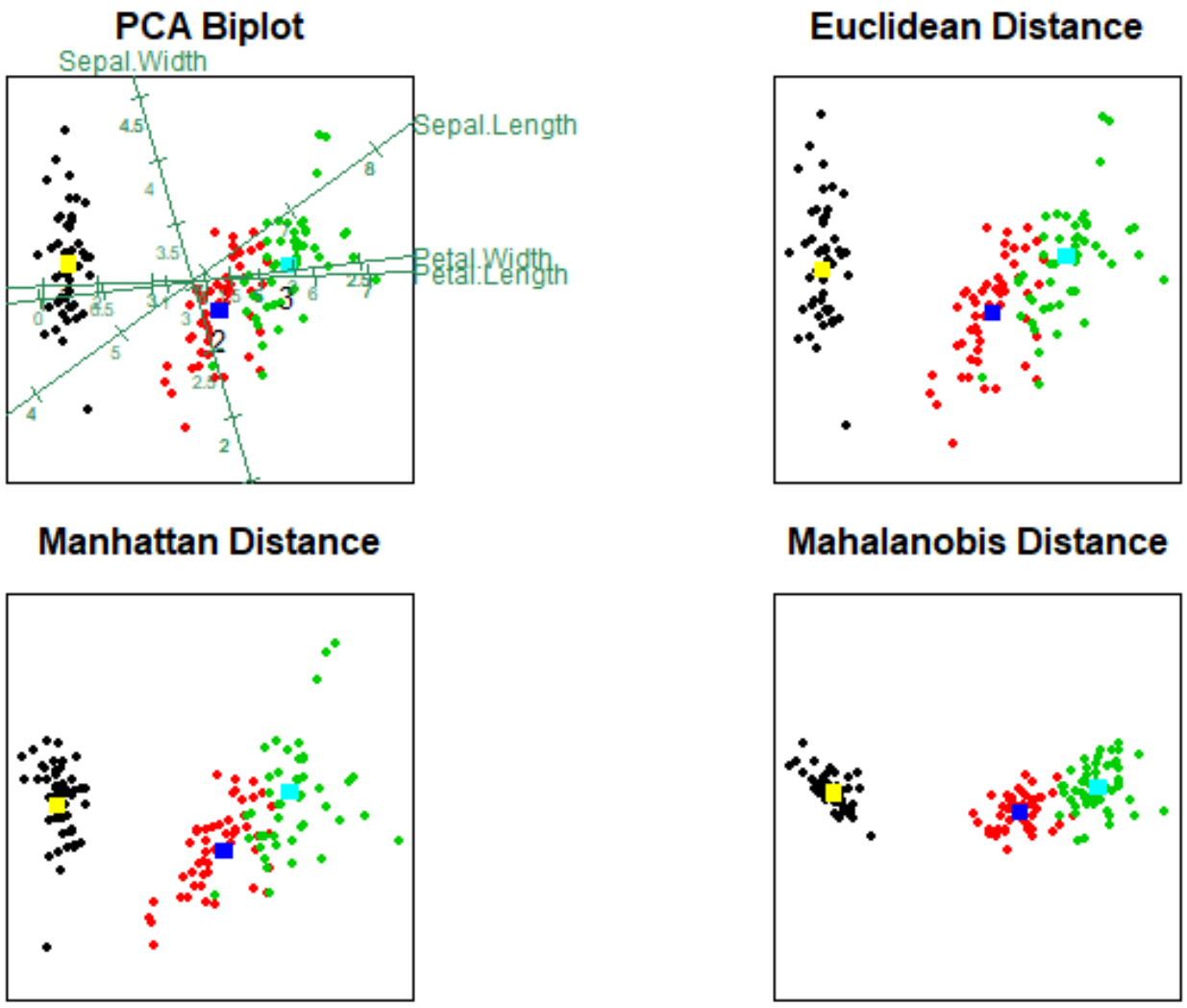


Figure 4-1: Comparing PCA biplots with MDS using various measures of distance for the Iris data points and group means. The yellow, purple and aqua-blue square data points represent the means of the black, red and green data groups, respectively.

The graph shows how the multi-dimensional (in this case, four-dimensional) data is represented in two dimensions. Points close to each other in two dimensions will also have similar values over the original dimensions. The mean of each group is indicated by a square data point.

The PCA and the MDS with Euclidean distance yield the same results, whereas Manhattan distance does not give a better separation between the red and green data points.

The Mahalanobis distance MDS results in much better separation of the data compared to Euclidean or Manhattan distance MDS, with very little overlap between the red and green data points.

The covariance matrix uses the average *sum-of-squares-and-products* (SSP) within each of the three groups of data. If the covariance matrix used to calculate the Mahalanobis distance is set equal

to the overall covariance matrix, the results would be the same as using PCA or Euclidean MDS with normalised data.

Non-metric MDS

An alternative to the metric classical scaling discussed above is *non-metric* MDS, which can also give Euclidean embeddable coordinates on which to perform regression for the calculation of the biplot axes. The non-metric MDS approach may result in a better separation of the data points and clearer visual representation in two dimensions. A popular non-metric MDS method is called *t-SNE* (Maaten and Hinton, 2008). Non-metric MDS will not be investigated any further in this dissertation.

4.2 Regression method for creating biplot axes

We have seen above that both Euclidean MDS (via PCO) and PCA results in the same optimal two-dimensional representation of data points. If the original $\boldsymbol{X}$ matrix is available for the MDS method the biplot axes can be created for the MDS representation, as provided by the eigenvector matrix $\boldsymbol{V}$ in PCA.

Creating biplot axes

In the absence of an eigenvector matrix $\boldsymbol{V}$, the projection of the data matrix $\boldsymbol{X}$ onto the biplot plane, $\boldsymbol{Z}$, can be approximated with a linear regression transformation:

$$\boldsymbol{X} = \boldsymbol{Z\Gamma}.$$

Here $\boldsymbol{X}$ is the original data matrix, $\boldsymbol{Z}$ is the MDS coordinates and $\boldsymbol{\Gamma}$ plays the role of a linear regression matrix of coefficients.The aim is therefore to find the projection $\widehat{\boldsymbol{X}}$ onto the two-dimensional biplot plane created by the basis $\boldsymbol{\Gamma}$, using the coordinates $\boldsymbol{Z}$.

Using the least squares formulation (Gower, Lubbe and le Roux, 2011, p206), $\widehat{\boldsymbol{\Gamma}} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}$.

If the MDS calculation of $\boldsymbol{Z}$ uses an Euclidean embeddable distance measure, then $\boldsymbol{Z}$ is defined in Equation 4 as:

$$\boldsymbol{Z}_{n\times 2} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{J}$$

and $\widehat{\boldsymbol{X}} = \boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}$

Equation 5: Regression of the data matrix X onto the biplot plane

In the case of PCA, $\widehat{\boldsymbol{X}}_{[r]} = \boldsymbol{UDJV}' = \boldsymbol{XVJV}'$ so that the plotted points (coordinates) are given by $\boldsymbol{Z} = \boldsymbol{XV}_r$.

Therefore, if the MDS distance measure is Euclidean, then $\boldsymbol{\Gamma}$ can be estimated by:

$$\widehat{\boldsymbol{\Gamma}} = \left((\boldsymbol{XV}_r)'(\boldsymbol{XV}_r)\right)^{-1}(\boldsymbol{XV}_r)'\boldsymbol{X} = \left((\boldsymbol{V}_r)'\boldsymbol{X}'\boldsymbol{X}(\boldsymbol{V}_r)\right)^{-1}(\boldsymbol{V}_r)'\boldsymbol{X}'\boldsymbol{X}(\boldsymbol{V}_r\boldsymbol{V}_r{}') = \boldsymbol{\Lambda}_r{}^{-1}\boldsymbol{\Lambda}_r(\boldsymbol{V}_r)' = (\boldsymbol{V}_r)'$$

Equation 6: Equivalence of Euclidean distance MDS regression coefficients $\widehat{\Gamma}$ and the PCA $V_r{}'$ matrix

$(\boldsymbol{V}_r)'$ is defined as the matrix consisting of the first *r* columns of $\boldsymbol{V}$, with $(\boldsymbol{V}_r{}'\boldsymbol{V}_r) = \boldsymbol{I}$. $\boldsymbol{\Lambda}_r$ is defined as the first *r* eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$, such that $\boldsymbol{V}_r{}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{V}_r = \boldsymbol{\Lambda}_r$. This gives:

$$\hat{\boldsymbol{X}} = \boldsymbol{Z}\hat{\boldsymbol{\Gamma}} = \boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}'_r$$

Equation 7: Predicting the continuous variable values represented by the coordinates on the biplot plane

The MDS method creates the linear biplot axes $\hat{\boldsymbol{\Gamma}}$, by using the MDS coordinates $\boldsymbol{Z}$ and the original data $\boldsymbol{X}$, with $\boldsymbol{x}_k = \boldsymbol{Z}\boldsymbol{\gamma}_k$ and $\boldsymbol{\gamma}_k$ the k'th column of $\boldsymbol{\Gamma}$, the axis of the k'th variable.

MDS allows for the creation of many types of fitted coordinates $\boldsymbol{Z}$, depending on the measure of distance and method of approximating the distance in two dimensions that are used. If the distance measure used is not Euclidean, then the axes trajectory will not be linear. However, the distance measure still needs to be Euclidean embeddable to allow for an optimal two-dimensional representation of the data and trajectories.

The details of the various methods of calculating the non-linear biplot axes are explained in Gower, Lubbe and le Roux (2011), and were first proposed by Gower and Hand (1996). It will not be explained here, but the key result is that the values of individual data points can still be determined by orthogonally projecting the data points onto the non-linear axes.

The functionality of the non-linear or regression biplot is the same as that of the PCA biplot. When the distance measure is based on Euclidean distances, then the nonlinear biplot reduces to a PCA biplot and the nonlinear trajectories become linear.

An example of non-linear biplot trajectories provided by le Roux and Lubbe (2013) is shown in Figure 4-2.

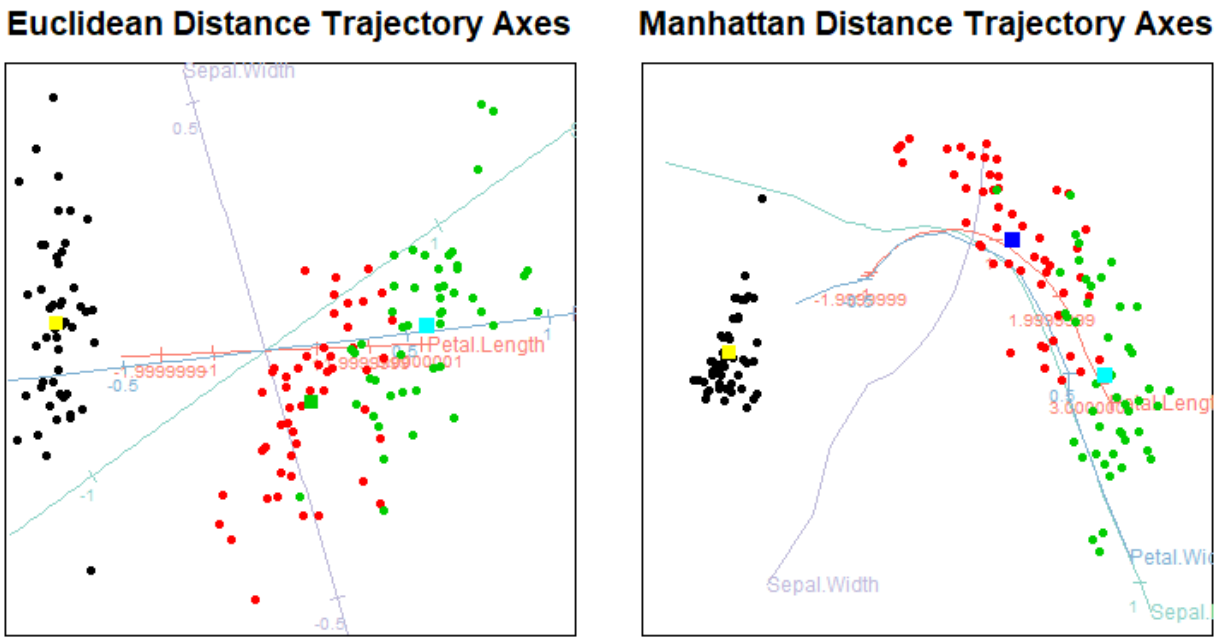


Figure 4-2: Euclidean and Manhattan distance biplots, with resulting trajectory axes

The non-linear axes give interesting insight into how the three Iris types vary by the four measurements. The Manhattan distance trajectory axes show more intuitively that Sepal Width is the least significant variable in separating the green and red groups. This is less evident on the Euclidean distance biplot - the orthogonal projection of the points onto the Sepal Width axis indicates that the green and red groups have very similar values for this variable.

For both distance measures, Pedal Width and Length can be seen to be the main variables separating all three of the groups.

Plotting the Form and Correlation method PCA biplots in Figure 4-3 shows that Sepal Length is in fact closely correlated with Pedal Length and Width, which is the intuitive interpretation from the Manhattan distance non-linear regression biplot.

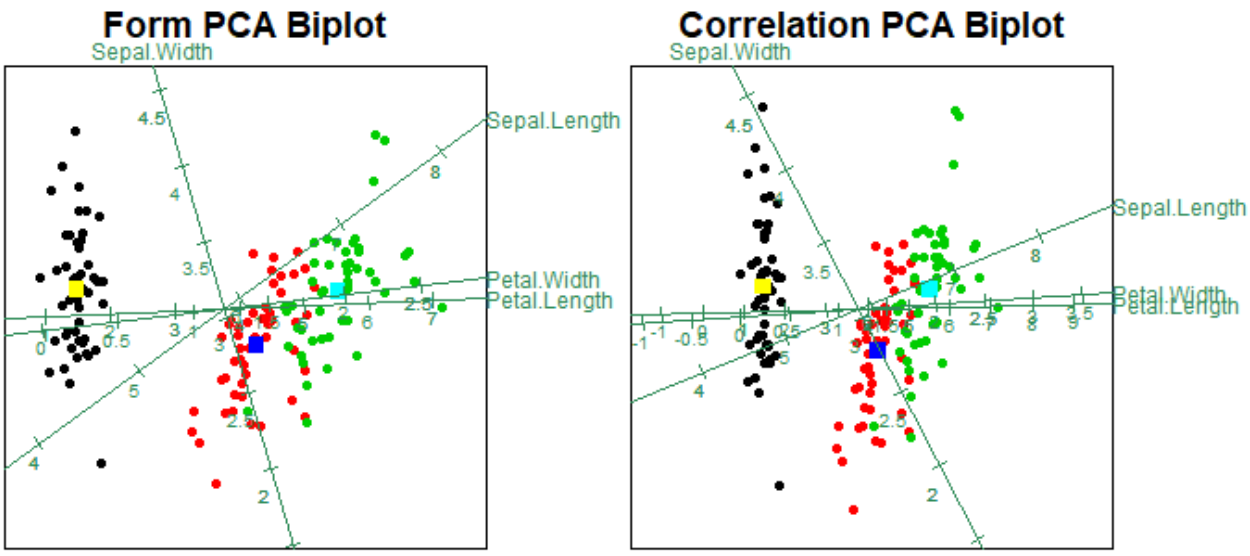


Figure 4-3: The form and Correlation biplots for the Iris data set

Since the non-linear axes can create very dramatic trajectories over the biplot surface, this method will not be used any further. Combining non-linear decision models with non-linear distance measures would be an interesting area for future investigation.

Adding new data points and predicting the values of data points on the biplot

Once the biplot and axes have been created, new data points can be interpolated onto the biplot space by calculating the vector of distances to each of the original data points in the original multi-dimensional space. The data point will then be placed in the two-dimensional space such that it optimally represents those original distances. For more details, see Gower, Lubbe and le Roux (2011, p213).

As with regression, the basis matrix $\widehat{\boldsymbol{\Gamma}}$ created using the MDS biplot points allows any coordinate on the biplot plane to be converted into a p-dimensional estimation $\widehat{\boldsymbol{x}}$ using:

$$\widehat{\boldsymbol{x}} = \boldsymbol{z}'\widehat{\boldsymbol{\Gamma}}.$$

4.3 Plotting of group means and the implication for biplot construction

We have seen from the previous sections that constructing the biplot plane and biplot axes requires a matrix $\boldsymbol{V}_r$ (with $r = 2$). However, as the number of rows and dimensions in $\boldsymbol{X}$ grows with bigger data sets, the full eigen-decomposition of $\boldsymbol{X}$ can become computationally difficult.

Also, the interest of the analysis is not necessarily to represent the full data set accurately, but rather to clearly separate and visualise only the differences between groups in the data set, that is, males vs females or different Iris types.

Therefore, an alternative approach is to rather calculate the eigen-decomposition of $\bar{\boldsymbol{X}}$ $(k \times p)$**,** which represents the matrix of k centroids (or group means) over p variables. The selection of the number of groups k can be based on an intuitive value such as the number of subject types in the data. (Note that k should be greater than or equal to three, as will be explained shortly.) Alternatively, k can be a larger value selected such that the centroids represent an accurate sample of the total data set, with $k \ll n$.

Once the projection matrix $\bar{\boldsymbol{V}}_r$ has been calculated for the centroids, it can be applied to the individual data points in order to project all the data points onto the biplot plane. This will result in an optimal separation of the centroids on the biplot plane, while still showing all the data and reducing the size of the eigen-decomposition matrix to k instead of n rows.

How the data points and centroids are presented depends on the distance measure used to calculate the distances between the centroids. When n is large, calculating appropriate centroids can also be a computationally difficult exercise.

The section above explained two ways of projecting the centroids – using PCA biplots (and implying Euclidean distance) or using MDS and adding the axes using regression. Along with different distance measures that can be used for MDS, this could result in many permutations of creating biplots. This section discusses two options, namely Canonical Variance Analysis (CVA) and Analysis of Distance (AoD).

Canonical variance analysis

Canonical Variance Analysis (CVA) is the PCA of centroids in the canonical space, and therefore requires the calculation of the principal component matrix $\boldsymbol{V}$ for the matrix of centroids after they have been transformed into the canonical space. Transforming the data into the canonical space requires the calculation of the matrix $\boldsymbol{L}$, such that $\boldsymbol{y}' = \boldsymbol{x}'\boldsymbol{L}$, with $\boldsymbol{x}$ being in the original space and $\boldsymbol{y}$ in the canonical space.

Once the transformation and projection matrix $\boldsymbol{M} = \boldsymbol{LV}$ has been calculated for the centroids, the individual data points can be transformed and projected onto the biplot using the same matrix $\boldsymbol{M}$.

Because of the focus on the matrix of group means, and due to the fact that the original matrix was first centred such that the mean of each variable is zero, only $k - 1$ of the k groups are needed to calculate the last group's mean value. Therefore, a plane intersecting the k groups' mean data points will occupy at most $m = min(k - 1, p)$ dimensions. If there are three $(k = 3)$ groups in the data, the matrix of group means will consist of a maximum of two dimensions. Therefore, a two-dimensional biplot will represent the three mean points without any approximation caused by the projection of data – a two-dimensional biplot plane can go exactly through any three points. However, the additional data points projected onto this biplot plane will only be approximately accurate.

This also implies that a two-dimensional biplot based on the centroids needs at least three groups.

As first developed by Sir Ronald Fisher and then expanded on by Rao (1948) as part of *linear discriminant analysis*, $\boldsymbol{L}$ is calculated as an eigenvector matrix that solves the eigenvector equation

$$\boldsymbol{WL} = \boldsymbol{L\Lambda}$$

Here $\boldsymbol{LL'} = \boldsymbol{W^{-1}}$ and $\boldsymbol{L'L} = \boldsymbol{\Lambda}$, where $\boldsymbol{W}$ is the within-class SSP matrix

$$\boldsymbol{W} = \boldsymbol{X'X} - \overline{\boldsymbol{X}}'\boldsymbol{N}\overline{\boldsymbol{X}}$$

and $\boldsymbol{N}$ is a diagonal matrix of the group sizes such that $\boldsymbol{N} = \boldsymbol{G'G^{-1}}$.

Without going into detail, it can be shown that by first applying the transformation $\boldsymbol{y'} = \boldsymbol{x'L}$ and then performing PCA, implies that if the measure of distance in the original space is Mahalanobis then the measure of distance in the canonical space is Euclidean (as is always the case when performing PCA). The same result would therefore have been found by first calculating the Mahalanobis distances between the group means with the covariance matrix equal to $\boldsymbol{W}$, and subsequently performing classical scaling MDS followed by regression to determine the biplot axes.

In the PCA biplot, the group means do not contribute to the scaffolding axes. In the CVA biplot, the scaffolding axes are determined by the group mean points and the individual data points are then interpolated onto the biplot plane using the canonical transformation matrix $\boldsymbol{L}$ and the scaffolding axes $\boldsymbol{V}$, represented by $\boldsymbol{M}$.

CVA focuses on optimally separating the group means, and therefore aims to maximise the between–class, relative to the within–class, variance. CVA is optimal in the case where the within–class covariances are equal and the pooled within–class covariance matrix is an accurate estimate

of within–class covariance of all k different groups of data. If this is not the case, then AoD (explained below) will in theory yield better results.

CVA improves the visual representation of the centroids compared to PCA by:

1) Calculating the optimal biplot plane for the group means, which causes the group means to be optimally separated and visualised. If there are only three groups, the biplot plane will fit the three mean points exactly and the axes and group means predictivities for CVA will achieve the maximum value of unity. This will not be the case for the group means of the PCA biplot.

2) Automatically dealing with incommensurable variable unit sizes by standardising the data with the standard deviation of each variable. In PCA the data needs to be pre-processed by standardising each variable, otherwise the results are scale-dependent. If the CVA covariance matrix is set equal to the variance matrix, the results would be the same as using PCA.

The relationship $\boldsymbol{M} = \boldsymbol{LV}$ allows any data point $\boldsymbol{x}$ to be plotted in the canonical biplot plane. This is reduced to a coordinate $\boldsymbol{z}$ by multiplication with $\boldsymbol{J}$. $\boldsymbol{M}$ is also invertible and therefore $\boldsymbol{M}^{-1}$ can be used to convert any coordinate from the biplot plane to the multi-dimensional canonical space and then from the canonical space to the original space. Therefore:

$$\boldsymbol{z}' = \boldsymbol{x}'\boldsymbol{MJ} \text{ and } \hat{\boldsymbol{x}}' = \boldsymbol{z}'\boldsymbol{M}^{-1}$$

$$\hat{\boldsymbol{X}} = \boldsymbol{XMJM}^{-1} \text{ , where } \boldsymbol{J}: p \times p = \begin{bmatrix} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \text{ and } r \leq p.$$

Analysis of distance

An alternative approach to represent the results of different groups of data optimally is to perform the PCO of $\boldsymbol{D}_k$, with $\boldsymbol{D}_k$ the distance between each of the k centroids. This will result in the centroids being optimally represented in two dimensions through the MDS of $\boldsymbol{D}_k$. The matrix $\boldsymbol{V}$ calculated from the classical scaling of $\boldsymbol{D}_k$, can then project the multi-dimensional data points onto the biplot plane, with $\boldsymbol{z}' = \boldsymbol{x}'\boldsymbol{VJ}$. The biplot axes can be added through regression of the data onto the biplot plane, and individual points can be added using a method described by Gower and Hand (1996) and Gower and Krzanowski (1999).

If Euclidean distance is used to calculate $\boldsymbol{D}_k$, the PCO of $\boldsymbol{D}_k$ will give the same results as PCA of the group means. This method is also called *analysis of distance*, or AoD. Therefore, AoD tries to maximise the variation between the group means. If the Mahalanobis measure of distance used with the covariance matrix set equal to the within-class SSP, then AoD will yield the same results as CVA.

The Iris data, as represented using the PCO of the standardised group means and the CVA, is shown in Figure 4-4.

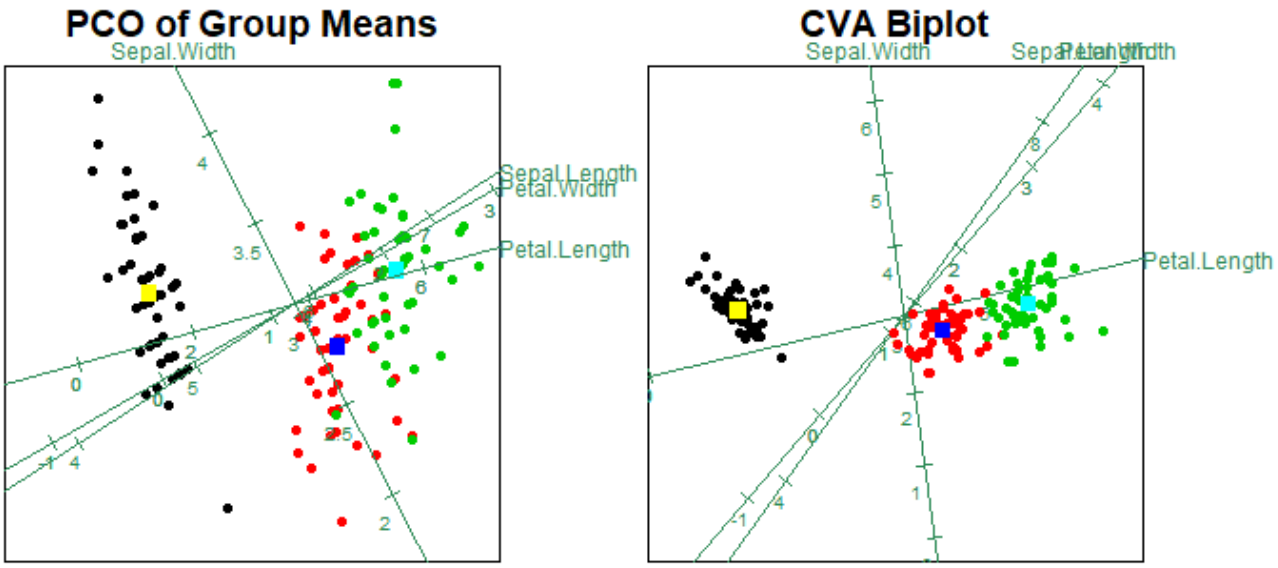


Figure 4-4: PCO of group means biplot vs CVA biplot

The CVA biplots produce a much better separation of the different groups of data. The data within each group is also more spherical, which means that the correlation of the variables within the groups have been successfully removed.

This is not the case for the PCO of group means, because the assumption of uniform within-group variance is appropriate for this data. When this is true, the CVA will produce better results compared to the PCO of group means.

Compared to using CVA, the benefits of using AoD are:

- AoD reduces the size of the eigen-decomposition, since the scaffolding axes are based on the decomposition of the group means matrix with k columns instead of the full data matrix with p column, where $k < p$.

- AoD is more flexible in terms of the underlying assumptions. It can use different types of distance measurers (preferably Euclidean embeddable) and does not assume that the within group covariance is the same between all the groups in the data.

- AoD provides an optimal separation of the group means and will have axes and group means predictivities of unity in the case of having three groups. The individual data points projected onto the biplot will however have axes and data point predictivities below 100%.

However, AoD requires that the variables be standardised when calculating the distance and group means to deal with incommensurable variables. The results are therefore scale-dependent.

Accuracy of the PCA, Euclidean MDS, CVA and AoD biplots

This section concludes by comparing the accuracy of the four types of biplots discussed so far. For the PCA biplot:

- Quality is 96%
- Axes predictivity is:

Sepal Length	Sepal Width	Petal Length	Petal Width
92%	99%	98%	94%

- Data point predictivity ranges between 20% and 99%, with an average of 91%

For the Euclidean MDS biplot, the regression vector $\widehat{\boldsymbol{\Gamma}}$ is orthogonal with $\widehat{\boldsymbol{\Gamma}}\,\widehat{\boldsymbol{\Gamma}}' = \boldsymbol{I}$. Therefore, Type A orthogonality holds and the data point predictivities can be used. The Euclidean MDS and PCA biplots give the same quality and the same axes and data point predictivity. See Appendix A for more details on how this conclusion is determined.

For CVA and AoD biplots for data with three groups, the quality and predictivities of the axes and group mean points are equal to 100%. This is because there are three groups for a two-dimensional biplot. However, because the projection matrices $\boldsymbol{VJ}$ for AoD and $\boldsymbol{MJ}$ for CVA are based on the group means and not on the original data points, the residuals and fitted data matrix $\boldsymbol{X} - \widehat{\boldsymbol{X}}$ and $\widehat{\boldsymbol{X}}$ are not orthogonal and the Type A decomposition is not possible. Therefore, the predictivities for the projected data points cannot be calculated.

Using AoD biplots to reduce the distance calculation requirements

As explained above, the AoD method enables the calculation of a smaller eigen-decomposition or distance calculation of the matrix $\overline{\boldsymbol{X}}\,(k \times p)$**,** with $k \ll n.$ The original data points can be added afterwards to give a biplot of the full data set, although they will be less accurately represented than the points in a full PCA biplot. This approach allows for the creation of biplots for much larger data sets than would otherwise be computationally feasible. The overall structure and interaction of the data points and variables is still accurately represented and interpreted, even if only the k centroids were plotted.

This can be seen in Figure 4-5 where a PCA biplot of the full data set is compared to the PCO biplot of nine representative sample points, with the rest of the data set projected onto the PCO biplot plane.

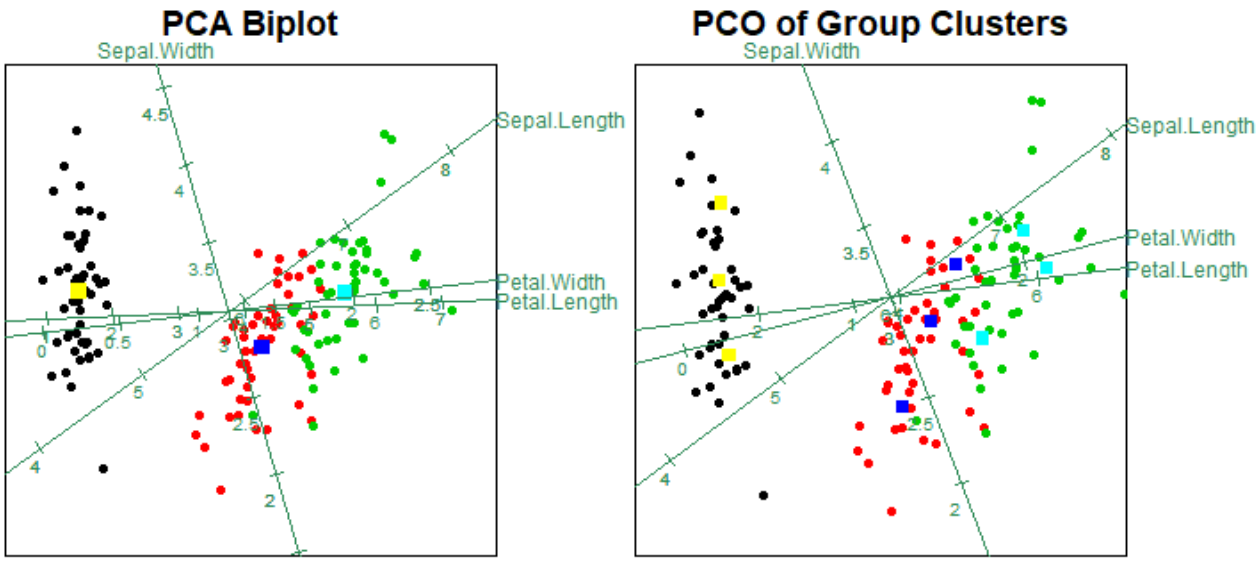


Figure 4-5: Comparison of a PCA biplot using all the data vs the PCO of only 9 representative sample points

However, as noted in the section above and shown in the graph, there will always be a loss of quality and change in the representation of the data points and variable axes when using this method. This is especially true for outliers, which would not be represented by the sample points.

4.4 Conclusion on the MDS and the impact of distance measures

This chapter has explained how different distance measures can be used to yield different biplots, provided the distance measures are Euclidean embeddable. The best distance measure depends on the structure of the data and what aspects of the data need to be illustrated. Certain distance measures will result in non-linear biplot axes, which can be useful for illustrating the non-linear interaction between the data points and variables.

The chapter also explained how representative samples of the data set can still result in biplots that represent the structure and interaction of the data points and variables, but significantly reduce the time required to calculate the distance matrix.

The methods explained up to now can only represent continuous variables. The next chapter explains how categorical data can be visualised with a biplot.

Chapter 5 Generalised Biplots

5.1 Introduction to generalised biplots

The previous sections explained how biplots can be created to reflect different distance measures between data points, including the advantages of focusing on the means or representative samples of the data instead of the full data set.

Up to now, only continuous variable data – age, height and weight – was used. The categorical variable – gender – was used to group and colour the data, but the biplot data points and axes only reflected the relationships and differences in the continuous variables.

Continuous variables can be either interval-scaled or ratio-scaled and are naturally represented by the biplot axes, which represent the scale and the direction of the data. However, ordinal categorical data[6] does not have a scale, even if it can be ordered and ranked. For example, shirts that are marked as small, medium or large have an ordering that represents their relative size, but does not indicate how much bigger a large vs medium shirt is compared to a medium vs small shirt. Nominal categorical data have no intrinsic scale or order. Green eyes are not better or worse than blue eyes. Representing these two data types with an axis will result in representations that are both misleading and inconsistent, implying an order and comparative scale that do not exist.

Categorical data – either ordinal and nominal – therefore require a different approach if it is to be represented accurately in biplots. Normal PCA or CVA biplots are not feasible options, since SVD decomposition assumes that all variables are continuous and only Euclidean or Mahalanobis distance measures can be used to measure the distance between data points.

There are many different methods that can be used to overcome these shortfalls, including *categorical PCA biplots*. The method that will be used in this dissertation is called *generalised biplots*, which build on the multi-dimensional scaling (MDS) biplot method explained in Chapter 4.

To incorporate categorical data, generalised biplots require additional features to:

- Standardise the data to avoid distortions due to incommensurate value
- Measure distance and create explanatory axes from ordinal and nominal variable types
- Predict the original multi-dimensional data point represented by a point on the biplot plan

After explaining these differences between MDS and generalised biplots, an example of representing continuous and categorical data on the same biplot will be provided.

[6] Ordinal data is a type of categorical data. But in the rest of the dissertation it is treated as a type of continuous data

5.2 Additional considerations for representing categorical data

This section will describe all the changes to the continuous data MDS biplots method that are needed to allow for the representation of categorical data.

Previously, p was defined as the number of explanatory variables. To indicate whether the data that is being referred to is continuous or categorical, p can be separated into p_1 and p_2, with:

- p_1 representing the number of continuous and ordinal variables
- p_2 the number of nominal categorical variables
- $p = p_1 + p_2$

Equation 8: Separating the number of variables into the number of continuous and the number of categorical variables

Standardising data for incommensurable variables

The first issue is that continuous, ordinal and nominal data types are by nature incommensurable. The MDS method captures the variance in the data in the same way that PCA would, namely, variables that have a large variance will contribute more to the distance and spread of the data points. This implies that if a categorical variable has a lower variance because of the way that it is recorded and that its distance is measured, the contribution to the spread of the data will be lower. This is similar for a variable that is recorded on a larger scale and has a distance measure with larger variance.

Therefore, different types of standardisation are needed for different data types to ensure that they all contribute an appropriate amount to the distribution of the data on the biplot. One way to ensure this is to scale all of the variables to ensure that each variable have a unit sums of squares across all data points, which will result in each variable contributing the same amount to the total variance.

Achieving unit sums of squares when calculating distances vary by data type. For continuous variables, this is achieved by dividing the variable by $\sqrt{(n-1)} \times s$ before calculating the Euclidean embeddable distance, with s the sample standard deviation. Over n samples this results in a contribution towards the total distance from each varible of n. For categorical variables, unit sum of squares is achieved by dividing the *ddistance* matrix $\boldsymbol{D}^*$ by $-\mathbf{1}'\boldsymbol{D}\mathbf{1} / n$, to ensure a contribution towards the total distance of n.

This is not the only method and can cause certain variables to be under- or over-represented compared to their true variance. However, without any prior knowledge to how much variance one variable should contribute, this is the most stable approach.

An alternative method is to scale the continuous variables to have unit variance and the categorical variables to unit sum of squares. This will ensure that, within the continuous variables, and within the categorical variables, they are commensurable. This scaling will however not ensure commensurability between the two group of variables.

A third alternative is to use interval scaling for continuous variables to ensure they have unit range. Then the categorical data distance measure that is used must also be comparable to interval-scaled data.

The quality and variable predictivities discussed earlier can be used to decide which of the above three methods will be best given the data types and purpose of visualising the data using biplots. More information on distance measures for categorical data can be found in Gower (1971).

Ordinal data

Ordinal categorical data shares properties of both nominal categorical and continuous data. Because they can be ordered, a linear variable axis can be used to represent their relative position. Data points with higher ranked variable values will be presented further along the axis.

A *normalised-rank* distance measure is used for ordinal data, and this ensures that every additional higher rank adds the same amount to the distance measured and ensures equal weight is given to each rank. This measure is also Euclidean embeddable. The distance is multiplied by $-\frac{1}{2}$ but not squared (compared to the *ddistance* measure used in PCO biplots) to avoid data points with larger differences in rank having a too big difference in distance measured.

This method is directly comparable with an interval-scaled continuous variable and does not need to be standardised first, allowing for an easy way to remove the incommensurability between different variables.

Note that some variables vary over a very small integer range and can be either ordinal or continuous. The choice of how the variable is classified then needs to be made based on expert opinion since there is no single rule that determines if such a variable is ordinal or continuous.

Predictive axes can now be created for the continuous and ordinal variables using the regression method that was explained for the MDS biplot method. The p-dimensional basis matrix $\widehat{\boldsymbol{\Gamma}}$ is created using the regression of the original values for the p_1 continuous and ordinal variables onto the $\boldsymbol{Z}$-coordinates created by the classical scaling of the new distance matrix.

Since the Euclidean distance measure is used to represent and interpret the continuous and ordinal variables, the biplot axes created using the regression method will be linear.

Nominal data and category level points

The key to representing nominal categorical variables is the use of *categorical level points* (CLPs). Each category can consist of different levels, for example, green and blue are two levels in the eye colour category. These individual levels must be represented on the biplot plane.

A CLP is defined as the centre point of all of the data points on the biplot plane that are part of the specified category level. The CLP is not calculated in the original n-dimensional space of the data points but is calculated as the k-means centroid of the MDS biplot data points, with k set to 1.

CLPs allow nearest neighbour regions to be created around each CLP. The estimated category level value of any point on the biplot plane depends on the region in which they are plotted and is set equal to the value of their nearest CLP. This ensures that only one of the finite number of distinct levels of a categorical variable can be allocated to any coordinate. Fixed cut-off boundaries - called *category level boundaries* - indicate when a point will move from being nearest to one CLP and therefore be classified in one level or another. Continuous axes are not used because they imply a continuum of possible values with no fixed boundary for moving from one level to the next.

As an alternative, a CLP can be calculated as the centre point in the original multi-dimensional space. The CLPs could then be projected onto the biplot plane and the corresponding nearest neighbour region calculated in the biplot space. This method is not followed here, however, due to the additional complexity of projecting the CLP onto the biplot plane that this method requires.

CLPs allow categorical variables to be shown on the biplot plane in a way that:

- Is consistent with the distance measure and MDS method used to create the biplot
- Creates distinct and finite category levels to be predicted for a coordinate
- Shows how the categorical variables relate to the continuous variables
- Does not imply an ordering where there is none

The CLP method, however, also poses certain problems. The variable axes for all of the continuous variables can be shown simultaneously, allowing the correlation between all of the variables and all the variable values of the data points to be seen on one biplot. With CLPs entire convex regions of points that are nearest to each CLP are created, called *prediction regions*, which implies that:

- For categories with a large number of levels (for example five levels, as will be used in Chapter 6), the prediction regions for some levels can be very small.
- If there is more than one category, the category level prediction regions will overlap. This is overcome by only drawing the individual boundary lines and not the actual regions, but it can

still cause multiple and possibly confusing boundaries, similar to having too many continuous axes.

- Many data points do not fall within their prediction region, lowering the sample predictivity of the biplot space.

For categorical variables, the data set is usually coded in pseudo-numerical form, where the k'th variable is recorded in an $\boldsymbol{n} \times \boldsymbol{L}_k$ matrix $\boldsymbol{G}_k$, and $\boldsymbol{L}_k$ is the number of category levels in category *k*. The i'th row of $\boldsymbol{G}_k$ is zero, apart from a single 1 in the column relating to the actual category level taken by the i'th data point.

$\boldsymbol{G}_k$ is termed an *indicator matrix,* with $\boldsymbol{G}$ the matrix for all p_2 categorical variables, calculated as:

$$\boldsymbol{G} = [\boldsymbol{G}_1, \boldsymbol{G}_2, \dots, \boldsymbol{G}_{p_2}]$$

As explained earlier, to calculate the MDS biplot, a distance matrix $\boldsymbol{D}$ must be calculated using additive Euclidean embeddable distance measures. To incorporate the nominal data, a dissimilarity coefficient that respects the structure of the $\boldsymbol{G}$ matrix and that is Euclidean embeddable is needed.

A dissimilarity coefficient measure for multi-level categories with zeros and ones leads to a situation where the number of negative matches (0-0) dominate the number of matches (1-1). The number of mismatches is twice the number of disagreements, since one disagreement leads to two mismatches (0-1) and (1-0). However, one agreement only leads to one match (1-1). Hence, Gower and Hand (1996) suggest the extension of the *Jaccard* coefficient, called the *extended matching coefficient* (EMC). If the variables are binary, then the EMC coincides with the *simple matching coefficient*. The EMC measure ensures that the number of (0-0) matches for a category with multiple levels does not dominate the distance measure (Gower, Lubbe and le Roux, 2011, p 365, 376). The EMC distance for a single categorical variable is calculated as:

$$\boldsymbol{D}_k = -\frac{1}{2} \times (\mathbf{11'} - \boldsymbol{G}_k \boldsymbol{G}_k') \text{ for each category.}$$

Multiplying by the factor of $-\frac{1}{2}$ is needed to make it a *ddistance* measure.

The EMC contributes zero to the overall similarity of two data points if the k'th categorical variable is mismatched, and unity if matched, with a resulting maximum value of 1. It is also additive Euclidean embeddable and can be combined with the distance contribution from the quantitative variables that have been interval-scaled.

Clustering methods

As explained previously for the AoD method, a representative sample of $k << n$ points can be used to calculate the MDS and resulting regression matrix and CLPs. This is necessary since the additive distance matrix calculation time increases with a factor of $n \times n \times p$. A total of k representative samples can be found by using a clustering algorithm to cluster the data into k clusters. The data point closest to the cluster centroid can be used a representative of the whole cluster. The MDS biplot is constructed based on the k representative points to represent the complete data set. (See also principal points).

However, the clustering algorithm often also requires a distance calculation. Therefore, the above technique just defers the computation problem to another earlier calculation. It will however reduce the calculation time needed for the subsequent regression and CLPs.

Care must be taken to ensure the distance measures used to calculate the clustering centroids are the same as the distance measure used to calculate the MDS of the resulting *k* data points.

MDS and interpolative and predictive axes

Assuming that all of the distance measures are Euclidean embeddable and additive, the measures for each of the variable types can be added together to calculate the overall distance matrix:

$$\boldsymbol{D} = \sum_{continuous} \boldsymbol{D}_i + \sum_{ordinal} \boldsymbol{D}_j + \sum_{nominal} \boldsymbol{D}_k$$

Using the overall distance matrix, classical scaling can be used to create the two-dimensional MDS of the original (or k-clustered sample) data points. Adding the continuous and ordinal variable data axes and nominal variable CLPs creates the generalised biplot.

Once the biplot and axes have been created, new points can be interpolated onto the biplot plane by calculating the vector of distances between the new data points and each of the original data points in the original multi-dimensional space. However, the effect of the different prediction regions on the distance calculation makes it very hard to accurately add a point to the biplot plane. Using only a sample to create an AoD biplot will also reduce the accuracy of the interpolated data points.

Similarly to the AoD biplot, the multi-dimensional value represented by each point on the generalised biplot plane can be predicted. Further, the continuous and ordinal variable values are predicted using the basis matrix $\widehat{\boldsymbol{\Gamma}}$. Any coordinate on the biplot plane is converted into the p_1 continuous and ordinal variables of $\hat{\boldsymbol{x}}$ using:

$$\hat{\boldsymbol{x}}_{p_1} = \boldsymbol{z}'\hat{\boldsymbol{\Gamma}}$$

Equation 9: Calculate the p_1 continuous and ordinal variable values represented by the biplot

For each of the p_2 nominal categorical variables, the Euclidean distance from the coordinate to each of the CLPs will be measured and the level of the closest CLP used to estimate the categorical level values for $\hat{\boldsymbol{x}}_{p_2}$. However, this can result in the levels of some data points being different from their prediction region. If the data points with a certain category level cluster closely around its CLP, the CLPs and prediction regions will be well spread over the biplot plane and result in accurate predictions for most of the data points. If not, the CLPs will all be clustered close to the origin and the predictions will be inaccurate.

Even if the predictions are not accurate, the spread or clustering of CLPs will indicate the relationships between the CLPs, and between the CLPs and the axes, similar to the correlation relationship shown between the axes.

The final predicted value of each point on the biplot plane is then $\hat{\boldsymbol{x}}_p = \{\hat{\boldsymbol{x}}_{p_1}, \hat{\boldsymbol{x}}_{p_2}\}$.

Accuracy measures for ordinal and nominal data

It is still possible to measure the accuracy of the biplot projection when incorporating categorical variables. The overall quality only requires the SVD of the distance matrix, which now incorporates both the additive continuous and categorical distance measures. The SVD provides the eigenvectors with which to calculate the quality measure.

Type B orthogonality does hold for all types of MDS biplots, as explained in Appendix A. Therefore, axes predictivities can be calculated for the continuous and ordinal data variables as normal.

For nominal data, the axes predictivity depends on the accuracy of the CLPs, and is measured by the percentage of data points that fall within the correct prediction regions for that category. If the category levels clearly separate the data, then most of the data points will be closest to the correct CLP.

Type A orthogonality does not hold for MDS biplots since the combined distance measure is not Euclidean and is non-linear.

5.3 Generalised biplot example

Based on the changes explained above, the generalised biplot method can now be applied to the original height, weight, age and gender data set. Previously, gender was used to separate and colour the data points or as a binary continuous variable. If the data, including gender, is first normalised, the resulting biplot is shown on the left-side of Figure 5-1, including the gender axes with males being indicated by a 1 and females by a 0. The data was not normalised for the PCA biplot shown in Figure 3-1.

Next, gender will be used as a categorical variable and included in the measure of distance between the data points. The data is standardised to unit sum of squares per variable. Euclidean distance is used to measure the distance between the continuous variables, and the simple matching coefficient (that is, the binary case of EMC) is used for the categorical variable. The CLP for each of the categories, male and female, is plotted and the gender will still be used to colour the data points. The resulting biplot is shown on the right-side of Figure 5-1.

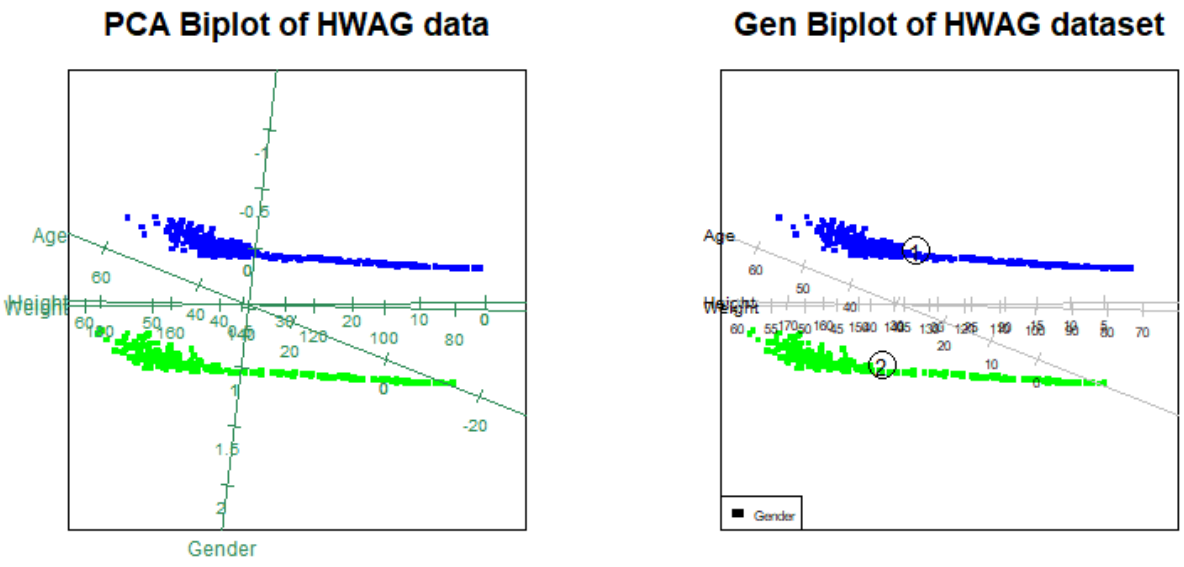


Figure 5-1: PCA biplot with gender a continuous binary variable and all variables normalised compared to a generalised biplot with gender a categorical variable and variables standardised to unit sum of squares. Males are indicated by green data points and CLP 2 and females are indicated by blue data points and CLP 1

The two biplots yield a similar representation of the data, since the squared distance is the same as the simple matching coefficient for a binary variable. The location of the CLPs implies the same correlation between the categorical gender variable and the other three variables as the PCA biplot. The CLPs remove the need for the gender variable axis as well as the need to imply a ranking of male gender data points as a 1.

If the female CLP is projected onto the age axes, it results in a slightly higher value than the projected male CLP. This is consistent with the linear regression interpretation of a model predicting a data point's age, where the female category level causes a higher predicted age. The male CLP projects onto higher weight and height values, which is also consistent with the other biplots and with a linear regression model for weight and height.

The data could also have been standardised to have a unit range. The resulting biplot would have a much bigger separation between the male and female data points, because the continuous variable data would be spread over the interval and would therefore have smaller squared distances compared to a 0-1 distance from a binary variable.

Alternatively, gender could have been treated as an ordinal variable. However, it would have had the same contribution towards the distance measure as the binary categorical variable, since a binary interval scaled distance measure would have the same effect as the simple matching coefficient distance used for the binary categorical variables.

The quality of the generalised biplot is 89% (compared to 89% for the PCA biplot). Because the distance measures are non-linear, the data point predictivities cannot be measured. The axes predictivities are:

Height	Weight	Age	Gender
100%	99%	91%	100%

The axes predictivities of the PCA biplot are:

Height	Weight	Age	Gender
91%	91%	74%	99%

Overall it indicates a significant improvement in the amount of variance explained by the axes.

5.4 Conclusion to generalised biplots

This chapter explained how a data set containing continuous, ordinal and categorical variables can be optimally represented in two dimensions. The generalised biplot accurately represents the overall structure and interaction between the data points and the variables.

The section has also indicated how any point on the two-dimensional plane can be projected to calculate the multi-dimensional values that the points represent.

Therefore, generalised biplots provide a tool with which to represent and interpret the predictions of any machine learning model, as will be explained in the next chapter.

Chapter 6 Interpreting Machine Learning Models

6.1 Introduction to interpreting black box machine learning models

The goal of this dissertation is to show how biplots can be used to generate post-hoc interpretations of machine learning models that are both globally and locally faithful for any type of machine learning model. As explained in Chapter 1, post-hoc explanations include visualising the features of the model and how they relate to individual data points and predictions.

For *global fidelity,* the biplot aims to visualise why one machine learning model provides more accurate or appropriate results than another model for the same data. For *local fidelity,* the biplot aims to visualise why the model predicted a certain outcome for individual data points.

Underwriting data and insurance application outcome prediction models

In Chapter 5 it was described how generalised biplots are created using the height, weight, age and gender data set and the Iris data set. For a real-world application the analysis in this section will focus on predicting the underwriting outcome of an insurance data set.

After a person fills in an insurance policy application form, there are broadly two possible outcomes. The first is that the policy gets accepted at standard rates. The second is that the policy gets declined, the decision gets deferred until a later date or extra premiums are loaded on the policy to reflect higher than normal risk. Often, this decision is only made after the insurers verify the policyholder's health by performing expensive and time-consuming medical tests.

If a machine learning model can predict with high enough certainty, using only the data provided on the insurance policy application form, whether the policyholder will be accepted at standard rates or not, it can save the company significant time and money, and significantly simplify the insurance application process. For the purpose of this dissertation, all cases where the person was not accepted at standard rates are classified as declined.

This chapter will explain how:

- The variables for predicting the underwriting outcome are selected, the machine learning models are calibrated, and their resulting accuracy is measured
- The two machine learning methods that are being considered work and how they are calibrated
- Generalised biplots visualise the predictions of any machine learning model
- Biplots give post-hoc globally and locally faithful interpretations of the two machine learning models

6.2 Variable selection and model calibration

The output of the machine learning model

The output of the machine learning model will always be a *prediction probability* between 0 and 100%, which represents the probability according to the model that the policy will be accepted at standard rates. The closer to 100%, the more confident the model is about the prediction. *A decision boundary* represents the prediction probability that must be exceeded before the policy is predicted to be accepted at standard rates.

The combination of the prediction probabilities and the decision boundary results in two measures of the model's accuracy.

The first is the *overall accuracy*: the number of policies that were declined and that have a model prediction score below the decision boundary, and the number of policies that were accepted having a model prediction score above the decision boundary, as a fraction of all the policies considered. The overall accuracy does not necessarily improve as the decision boundary increases, as this will lead to an increase in the number of false negative and false positive predictions – that is, the number of policies that are predicted to be declined but were accepted at standard rates, or that were predicted to be accepted but were not accepted at standard rates. The decision boundary should aim to maximize the overall accuracy, subject to the requirements of the next measure of accuracy.

The second is the *positive prediction accuracy*: Out of all the policies that have a prediction probability above the decision boundary, how many were accepted at standard rates. This measure is important because the accuracy of the policies that will be automatically accepted is of primary concern. Policies below the decision boundary will continue to follow the current manual underwriting process. Positive predictivity is expected to increase as the decision boundary value increases, since as the model's prediction probability increases, the model is increasingly confident that the policy would be accepted at standard rates. Only those policies will then be included in the measure. However, the number of policies with a score above the decision boundary will decrease as the decision boundary value increases. This number is called the *data prevalence* above the decision boundary.

From the above, it follows that a balance must be found between having accurate predictions and having enough qualifying policies to make it worthwhile to automate the underwriting process. Finding the decision boundary level that optimizes the measures will not be discussed in this dissertation and either a 70% decision boundary level or a 50% decision boundary will be used.

The overall accuracy and model predictivity measures should always be compared to the "*no information rate*", which is the percentage of policies that were accepted at standard rates and

therefore is the benchmark rate that the model accuracy must exceed. In the underwriting data used in this chapter, 69% of the policies have been accepted at standard rate. Therefore, if the model predicts that all policies should be accepted at standard rates, the model would be 69% accurate. The machine learning model and decision boundary calibration should result in a higher accuracy than 69% to show that it adds value.

Machine learning models also use other accuracy measures to decide between models or select hyper-parameters of the models. A common measure is the *receiver operating characteristic* (ROC) curve and corresponding *area under this curve* (AUC). The ROC curve is created by plotting the true positive rate against the false positive rate at various decision boundary values. The AUC measure is used to select the optimal hyper-parameters as part of the model calibration process, and gives an overview of the model accuracy across all potential decision boundaries.

More information on accuracy measures can be found in Yachen (2016).

The modelling process requires two data sets, namely, the *training set* and *out-of-sample test set*. The model is calibrated using the training data. The test set estimates the likely accuracy, or the generalisation error, that the model will have on new and independent data. If the training data was also to be used to measure the generalisation error, the true accuracy would be overstated (Hastie, Tibshirani and Friedman, 2008). K*-fold cross-validation* with the training data is used for estimating the expected generalisation error. The model fitting process uses $k-1$ parts of the training data, and estimates the error with the remaining k'th part. The cross-validation error is the average error after fitting the model k times and estimating the error with a different k'th part. This allows for the efficient use of the training data. Ideally, the hyper-parameters would be set using their own set of data, but having separate data sets for estimating each hyper-parameter would result in too small training subsets for accurate estimation.

Variable selection and model fitting

The underwriting outcome prediction models calibrated in this dissertation will use the application form data of ~32000 policies with 13 answered questions and the result of the current underwriting process. The *target variable* value is the result of the current underwriting process – "1" for policies that were accepted at standard rates and "0" if not (declined). The 13 questions form the *input variables* and consist of:

- 4 continuous variables marked as X_1 through to X_4
- 7 ordinal variables marked as X_5 through to X_{11}
- 2 categorical variables marked as Z_1 and Z_2, the first with 2 levels and the second with 5 levels

The axes and CLPs indicated on the biplot are renamed to X_1 through to X_p for the continuous and ordinal variables, and Z_1 through to Z_p for the categorical variables, to keep the underlying data anonymous. The j-th level of the i-th categorical variable will be indicated as Z_{ij}.

Not all 13 input variables necessarily provide information that improves the accuracy of the prediction of the target variables. Therefore, an important part of the modelling process is the selection of the appropriate variables for the machine learning method. As explained in Guyon and Elisseeff (2003) and Hastie, Tibshirani and Friedman (2008), the objective of variable selection includes improving the predictive performance of machine learning models. Variable selection falls under three categories, namely filters, wrappers and embedded methods.

Filters select subsets of variables independently of the chosen machine learning method and include the following possible pre-processing steps:

- Data checks

  These include checking for missing data, outliers and incorrect data types. The data provided to use for this dissertation was complete and correct and excessively large and small values were capped to realistic values given the range of most of the data.

- Dealing with unbalanced or insufficient data

  If a very large proportion of the target variable belongs to only one class then it is very hard for machine learning models to be calibrated such that it improves the prediction accuracy over classifying all of the data as the dominant class. To improve the model in these situations, possible data and algorithmic solutions have been proposed and summarised in Kotsiantis, Kanellopoulos and Pintelas (2006). A common method for dealing with unbalanced data is SMOTE: Synthetic Minority Over-sampling Technique, as proposed by Chawla et al. *(*2002). Since the dominant class of the underwriting target variable in this dissertation represents only 69% of the data, the data is not considered to be imbalanced, and no transformations are required to improve the model accuracy.

  The ~32000 data points are also sufficient, given that the data is balanced and only two classes in the target variable and 13 input variables are considered. Therefore, no data augmentation is required.

- Variable transformation

  In many cases variables that do not improve the prediction accuracy in their original form can after being transformed. Variable transformations can take various forms, which include:

  Standardisation: As with the distance issues discussed earlier, data that is commensurate in scale results in more accurate predictions. However, no standardisation was performed on the data. Due to confidentiality, the original values cannot be discussed, but the variables were in commensurable units. If standardisation is performed, another step is required in the biplot axes calibration to ensure that the original variable scales are shown on each axes, instead of the values of the resulting transformed variable. Alternatively, the biplot axes must be interpreted in terms of the resulting transformed variable axes.

  Feature discretization: Some algorithms do not model continuous data well. This is especially the case with linear models that do not allow for different interactions of a variable with other variables at different levels. In this case, it makes sense to discretize continuous values into a finite set of discrete units and separately model the interactions in each.

  Linear and non-linear space embedding methods: These techniques project or embed high-dimensional data into a lower-dimensional space while retaining as much information as possible. Lower-dimensional data allows for faster model calibration and removes noise contained in the higher-dimensional data. A classic example is principal component analysis (PCA), which, due to the low dimensionality of the available data, was not used in this dissertation.

  Signal enhancement: The signal-to-noise ratio may be improved by applying baseline or background removal, de-noising, smoothing, or sharpening. Fourier and wavelet transforms are popular methods. For this dissertation, a log transformation was applied to the variable with the largest and most right-skewed distribution to normalise the distribution and reduce its range. This improved the model accuracy and the visualisation of the variable as a biplot axis.

- Variable selection

  Filters also assist in selecting subsets of variables, to ensure that the selection is independent of the machine learning method. This includes ranking variables according to the accuracy of single variable classifiers, or the correlation between the target and input variables. The

highest-ranking variables are then retained, and low-ranking variables excluded from further modelling.

This can assist in removing large numbers of redundant variables to improve the speed of fitting the machine learning models. However, there remains a risk that a variable that is completely redundant by itself can be significantly predictive when combined with other variables. Therefore, a balance is required between the time and cost of modelling all combinations of variables against the total model accuracy required.

Wrappers select variables based on the output of machine learning models that score subsets of variables according to a measure of accuracy or predictive power. A wide range of search strategies can be used, including best-first, branch-and-bound, simulated annealing and genetic algorithms. The most popular are forward selection and backward elimination.

Forward selection progressively incorporates larger and larger subsets of variables. Backward elimination starts with the complete set of variables and progressively eliminates the least promising ones. Both methods yield nested subsets of variables.

Wrappers use generic measures of accuracy that can be applied to any machine learning method. *Variable importance plots (VIPs)* (Friedman, 2001), also called Relative Influence or Permutation Tests, is an example of a backward variable selection wrapper function. VIPs can be model specific, but a generic method randomly permutes each predictor variable at a time and computes the associated reduction in predictive performance (Greenwell et al., 2018). The more the predictive performance reduces, the more important the variable for accurately predicting the target variable.

Another type of wrapper is a partial dependence plot (Friedman, 2001; Hastie, Tibshirani and Friedman, 2008). Partial dependence plots (PDP) represent the effect of a variable X_i on the model's prediction probability after accounting for the average effect of all the other variables. Plotting how the prediction probability changes as the value of X_i increases over its range shows the dependence of the prediction of the variable. This gives a global interpretation of the impact of the variable. The PDPs used in Section 6.5 plot the change in the log-odds of the target variable as the input variable X_i changes. Therefore, it plots the change in:

$$f(x) = \log \frac{\Pr(\text{Accepted} \mid x)}{\Pr(\text{Declined} \mid x)}.$$

Whereas wrappers use generic measures, *embedded methods* have unique measures that reflect the way that the model is designed. Embedded methods are often used to calibrate the hyper-parameters of the model, and the performance assessments are usually done using a test data set or by cross-validation. Examples include the error rate of the out-of-bag data set used in the calibration of random forest machine learning models, or p-values generated by linear models.

Different measures of accuracy and different machine learning methods can cause wrapper or embedded methods to select different subsets of variables. Therefore, the variable selection process is simplified by using the same measure of accuracy in the wrapper function to select the variables included in different models. Different embedded methods will only be used to calibrate the hyper-parameters of the specific models.

For evaluating and selecting the final model, a single measure of accuracy must be used based on the hold-out test data set.

Calibrating machine learning models in R

All of the data modelling for this dissertation were fitted using R in R-Studio. In particular, the filters, wrappers and embedded methods were implemented using the '*Caret'* package, which is short for classification and regression training (Kuhn, 2008). The 'Caret' package aims to provide a uniform syntax for the implementation of different machine learning models and provide built-in functions to automate the modelling steps discussed above.

The plots of the output were generated with either R's internal *'plot'* function or the *'ggplot2'* package (Wickham *et al.*, 2018).

The models were fitted using a 75% training data set. The hyperparameters were set by using 5-fold cross-validation out of the 75% training data set, using AUC as the accuracy measure. The final overall accuracy and positive prediction accuracy were tested using the 25% hold-out test data set.

The data set used for creating the generalised biplot of the underwriting data was reduced to only 500 data points. In order to allow for faster computation of the biplots and to make the illustration of the biplot points more legible. The roughly 24000 training data points would have resulted in a blob of points that cannot be individually identified and separated.

The clustering, as explained in Section 5.2, was performed using the '*cluster'* package (Rousseeuw et al., 2018) and the '*clara*' function. The '*clara'* function provides an efficient way of finding clusters by only considering sub-data sets of a fixed size such that the increase in time and storage requirements are linear to the number of data points, rather than quadratic.

The R code also makes use of the '*myintegrate2016'* (le Roux, 2016) and '*pdist'* functions (Wong, 2016).

6.3 Calibrating two machine learning models

Recall that the purpose of this dissertation is to create and compare the visualisation of different machine learning models. Different machine learning methods have their own underlying assumptions that may result in better or worse results for certain problems, depending on the distribution of the underlying data and the type of predictions that are required.

Therefore, to find the method that will result in the highest prediction accuracy will require an analysis of a wide range of options, which can be narrowed down by selecting methods that have underlying assumptions that match the data and output required. Rather than perform this analysis, this dissertation will focus on comparing the visualisation of only two methods, namely, a *logistic regression model* – a method that is used most often in the insurance industry (Boland, 2006) – and a *stochastic gradient boosting model* – a non-linear model that is known to produce very accurate predictions (Hastie, Tibshirani and Friedman, 2008). The methods described here can however be applied to any classification method with posterior probabilities as output.

This section provides a basic introduction to these two methods, explaining the underlying assumptions of the models and how their parameters are calibrated in RStudio. This is required to understand how and why their predictions and the visualisation of their output are different.

Introduction to generalised linear models (GLMs)

The basic assumption a linear model is that the regression function $\boldsymbol{f(x) = E(Y|X)}$ is linear, or that the linear model is a reasonable approximation. Therefore, the predicted output $\boldsymbol{Y}$ is modelled as a linear function of the data matrix $\boldsymbol{X}$, with:

$$\boldsymbol{\hat{Y} = X'\hat{B}.}$$

The estimates of the coefficients $\boldsymbol{B}$ are chosen such that they minimize the residual sum of squares. If $\boldsymbol{X'X}$ is non-singular, and $\boldsymbol{Y}$ a matrix of 0's and 1's, the unique solution is given by:

$$\boldsymbol{\hat{B} = (X'X)^{-1}X'Y.}$$

However, this formulation of a linear model implies that the predicted value of $\boldsymbol{f(x)}$ can be greater than one or negative. Since the goal is to predict a value between 0 and 1, this is not an acceptable outcome. This problem is solved by applying a monotone *logit* transformation to the formulation that will ensure that the predicted values are between 0 and 1 (Hastie, Tibshirani and Friedman, 2008). The logit transformation implies that:

$\boldsymbol{\hat{Y} = \frac{e^{X'\hat{B}}}{1+e^{X'\hat{B}}}}$, or alternatively, that $\boldsymbol{logit(\hat{Y}) = X'\hat{B}}$

This formulation results in the *logistic regression model*, which has a binomial error distribution and is suitable for predicting binary outcomes.

The coefficients $\boldsymbol{B}$ are estimated using the iteratively re-weighted least squares (IWLS) algorithm, with $\boldsymbol{B} = \mathbf{0}$ used as the starting estimates. This method is required because the logit transformed equation above now implies that the log-likelihood score equations are non-linear in $\boldsymbol{B}$ and therefore do not have a closed form solution.

Linear models, and monotone transformations of linear models, imply a linear smooth and single decision boundary. Therefore, the decision boundary between two classes is the set of points where the $\widehat{\boldsymbol{Y}} = 50\%$ (if the 50% decision boundary is used), which divides the input space into two regions of constant classification.

The benefit of the linear assumption is that the model will have very low sampling variance. The model parameter estimates are unlikely to change significantly from one sample of data to the next, since the model is also not significantly influenced by outliers in the data. Further, the model has no hyper-parameters, which simplifies the calibration and yields reliable predictions in situations where data is sparse. Therefore, if the assumption of a linear separation between the two classes is accurate, then the method is likely to provide good out-of-sample prediction accuracy.

Another benefit is that interaction terms between the variables can be added to the model, which will allow for a quadratic, but still single, decision boundary.

The estimated coefficients $\widehat{\boldsymbol{B}}$ also explain the impact of each variable on the predictions. With the logit transformation, the impact of each coefficient is now multiplicative and not additive.

The main disadvantage of using the linear model is most pronounced if there are multiple regions in the data that result in the same class outcome, since the linear model only creates a single decision boundary. Therefore, if what is perceived by the model as outliers are instead sub-spaces in the data representing different classes, the model will incorrectly classify those points.

In addition, the model requires the matrix $\boldsymbol{X'X}$ to be non-singular. Therefore, strong co-linearity in the data will prevent the estimation algorithm from converging and will result in no or unstable coefficient estimates. This also implies that the model is not suitable for situations with a large number of input variables, as co-linearity is then more likely to occur.

The problems of co-linearity and selection of variables are more effectively dealt with using the '*GLMNET'* package (Jerome et al, 2018), but this will not be further explained here.

Lastly, another benefit of using the logistic linear model is that it provides a built-in method for variable selection. This follows from the central limit theorem, which states that the resulting

coefficient estimates will be normally distributed. This allows each individual coefficient to be separately tested against the null-hypothesis of $H_0: \boldsymbol{B_i} = 0$. Since this is a two-sided test, the corresponding z-score at the 95% percentile will be 1.96. Therefore, $\hat{B}_i \pm 2 * se(\hat{B})$ will approximate the 95% estimate confidence interval, and if the confidence interval includes zero, then the null-hypothesis cannot be rejected.

The R function uses the Wald test (Wald, 1943) to determine the significance of individual variables. A backward stepwise variable selection process starts with all of the variables and then systematically removes the variables that are shown to be not significant – those with a p-value greater than 0.05. This is the process that will be used in the model building process explained in Section 6.5.

Introduction to gradient boosting machines (GBMs)

Gradient boosting is a non-parametric method that additively and sequentially adds new functions to explain the residuals of predictions from previously fitted functions.

Where normal regression aims to create an estimate $\hat{f}(x)$ that will minimize the selected loss function, boosting modifies the current estimate of $f(x)$ by adding a new function $f_i(x)$ in a greedy fashion. This is done by fitting the new function that will cause the greatest decrease in the measure of loss. For classification problems, the Bernoulli loss function is recommended (Ridgeway, 2018). Friedman, (2001) suggested selecting a class of functions that use the covariate information to approximate the gradient, hence calculating $\hat{f}(x)$ as a tree. Therefore, gradient boosting machines sequentially fit trees to the residuals of the predictions from the previous trees.

Gradient Boosting Machines are often compared to Random Forests methods, which is an ensemble method where each individual decision tree is grown independently. The final prediction is then made by majority voting between the trees.

Gradient Boosting Machines have four hyper-parameters that need to be selected to optimize the accuracy for a given problem. These hyper-parameters are:

- The size of the tree that is iteratively added: This represents the level of interactions between variables that are captured. If the tree has one split, it implies an additive model. If it has two splits, it allows for two-variable interactions.

- The number of trees or functions that are combined to give the final model: Unlike random forests, adding too many functions will cause the model to overfit and reduce the out-of-sample test prediction accuracy, due to the fact that the model will start trying to fit a model to smaller and smaller residuals.

- The shrinkage parameter: This parameter determines the scale of the contribution of each tree to the predicted outcome. A smaller shrinkage parameter results in “slower” learning, which means that each tree will explain less of the residuals, and therefore more trees are required to explain all the variation in the data. The shrinkage parameter and number of trees need to be considered together. The shrinkage parameter reduces the problem of overfitting caused by having too many trees.

- The amount of sub-sampling of the data at each step. At each iteration of fitting a function, only a certain percentage of the total data will be considered. This improves the speed at which each tree is built and results in a more accurate model. If data sub-sampling is used, the method is called a *stochastic gradient boosting model.*

Decision trees produce discontinuous piecewise constant models. This carries over to sums of trees, with of course many more pieces, since no smoothness requirement is placed on the results. Therefore, the decision boundaries produced by GBMs will not be smooth.

The benefit of boosting is that it can approximate various types of machine learning models through the basis expansion implied by the additive fitting of functions and the interaction depth implied by the size of each tree. It can therefore capture any non-linear interactions between the variables. In addition, the model can allow for different sub-spaces in the data representing different classes. These two features increase the potential prediction accuracy of the model.

The main disadvantage is that fitting the model can be time consuming and computationally expensive. This problem is increased by the need to test the outcome of various combinations of the hyper-parameters. Also, outliers can be modelled as separate prediction areas, which will reduce the out-of-sample prediction accuracy of the model.

Gradient boosting machines make use of variable importance plots (VIPs) and *partial dependence plots* (PDPs) to help determine the relative importance or contribution of each input on predicting the response. This information is used to assist with the backward stepwise variable selection process. Although not unique to GBMs, they do have built-in functionality to simplify the calculation of the VIPs and PDPs, compared to other machine learning models. These can be called in RStudio to interpret the results of the model.

6.4 Visualising the output of a machine learning model

Chapter 5 illustrated how the generalised biplot can be used to visualise the variables and data points of a data set with continuous and categorical data. This section explains how the generalised biplot can visualise the predictions of a machine learning model. The generalised biplot is model agnostic and can visualise the predictions of any machine learning method.

Visualising the output of a machine learning model using the generalised biplot requires three broad steps.

Selecting the biplot data: The first step is selecting the data that should be visualised and used to create the biplot. The biplot and machine learning model do not need to be created with the same data set, but the two data sets should contain the same variables, in order to ensure that predictions from the biplot can be used as valid input into the machine learning model.

In this dissertation, 500 sample points from either the training or the test data were used. Since additional points are not interpolated onto the biplot plane, the loss of accuracy from using only 500 sample points is not a concern. The only requirement is that the sample points create an accurate representation of the biplot axes and CLPs. The choice between using training or test data depends on the data set for which the generalised biplot should provide global and local interpretations. The initial analysis and variable selection were done using the training data. The interpretation of the final models was done using the test data.

Creating the biplot: The second step is creating a generalised biplot using the selected data. As explained in Chapter 5, this requires the following steps:

- Identify which variables should be treated as ordinal data

  Ordinal data is treated differently to continuous data and should not be incorrectly classified as continuous by the software.

- Standardise the continuous variables to ensure that all of the variables are commensurable

  Different approaches are possible, but the recommended approach is to standardise the continuous variables to unit sum of squares. The distances between categorical variables (ordinal and nominal) is typically between 0 and 1, thus each contribute a unit sum of squares. The choice between different distance measures and standardisations can be informed by looking at the resulting accuracy measures of the biplot data points and variables.

- Calculate the distance matrix

  The calculation of each variable's contribution to the distance matrix is performed separately. While continuous data uses a Euclidean distance measure, ordinal data makes use of the normalised rank distance measure, and categorical data first needs to be converted to 0's and 1's in the $\boldsymbol{G}$ indicator matrix, after which the distance is measured using the extended matching coefficient measure.

- Calculate the multi-dimensional scaling coordinates using the classical scaling method

  The *colour of the points* depends on the purpose of the biplot (for example, having separate colours for predicted and actual data points, or for training data and specific test data points). The biplot can also be zoomed into if the centre of the data set is the focus of the investigation.

- Create the variable axes and CLPs

  Adding the variable information is necessary to create a biplot instead of just an MDS plot. Since the distance measure used for the continuous data is Euclidean, the axes trajectory will be linear. The categorical data uses CLPs to plot all of the category levels. The CLPs will be marked by a number for variable category levels and each category will use a different colour.

Plot the output of the machine learning model onto the biplot: The third step is to colour the biplot plane to reflect the output of the machine learning model, to visualise how the machine learning model output varies over the full range of possible multi-dimensional values represented on the biplot plane. This is done using the following steps:

- Convert every pixel on the biplot plane to the predicted multi-dimensional approximation

  The p_1 continuous and ordinal variables are predicted by multiplying the biplot coordinate with the regression matrix $\widehat{\boldsymbol{\Gamma}}$, with $\widehat{\boldsymbol{x}}_{p_1} = \boldsymbol{z}'\widehat{\boldsymbol{\Gamma}}$, as explained in Equation 9, where p_1 represents the total number of continuous and ordinal variables, p_2 the number of nominal categorical variables, and $p = p_1 + p_2$, as defined in Equation 8.

  The $\widehat{\boldsymbol{\Gamma}}$ matrix is estimated using the continuous and ordinal variables of the original data and the data point coordinates matrix $\boldsymbol{Z}$, such that:

$$\widehat{\boldsymbol{\Gamma}}_{[p_1]} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}_{[p_1]}$$

The p_2 categorical variables' levels are predicted by determining the closest CLP. The predicted level can be found by looking at the *category level boundaries* (or CLP boundaries) that are created for each categorical variable. The categorical level boundaries represent the points on the biplot plane where the classification of the points move from one category level to another because the distance to the nearest CLP moves from one CLP to another.

- Plot the machine learning model prediction for each pixel

  Every pixel is now a p-dimensional value that has the data structure required by the calibrated machine learning model. Therefore, every pixel is a model input value that then gets converted into an output score between 0 and 100%. This score can be converted into a colour gradient. The closer the value is to 100%, the more confident the model is that the data point is a positive prediction and the deeper blue the pixel will be coloured. If the score is close to 50%, the model is uncertain of the classification and the pixel will be coloured white. If the score is close to 0%, the model is confident that the data point is a negative prediction and the pixel will be coloured deep red.

An example of the resulting biplot is shown in Figure 6-1.

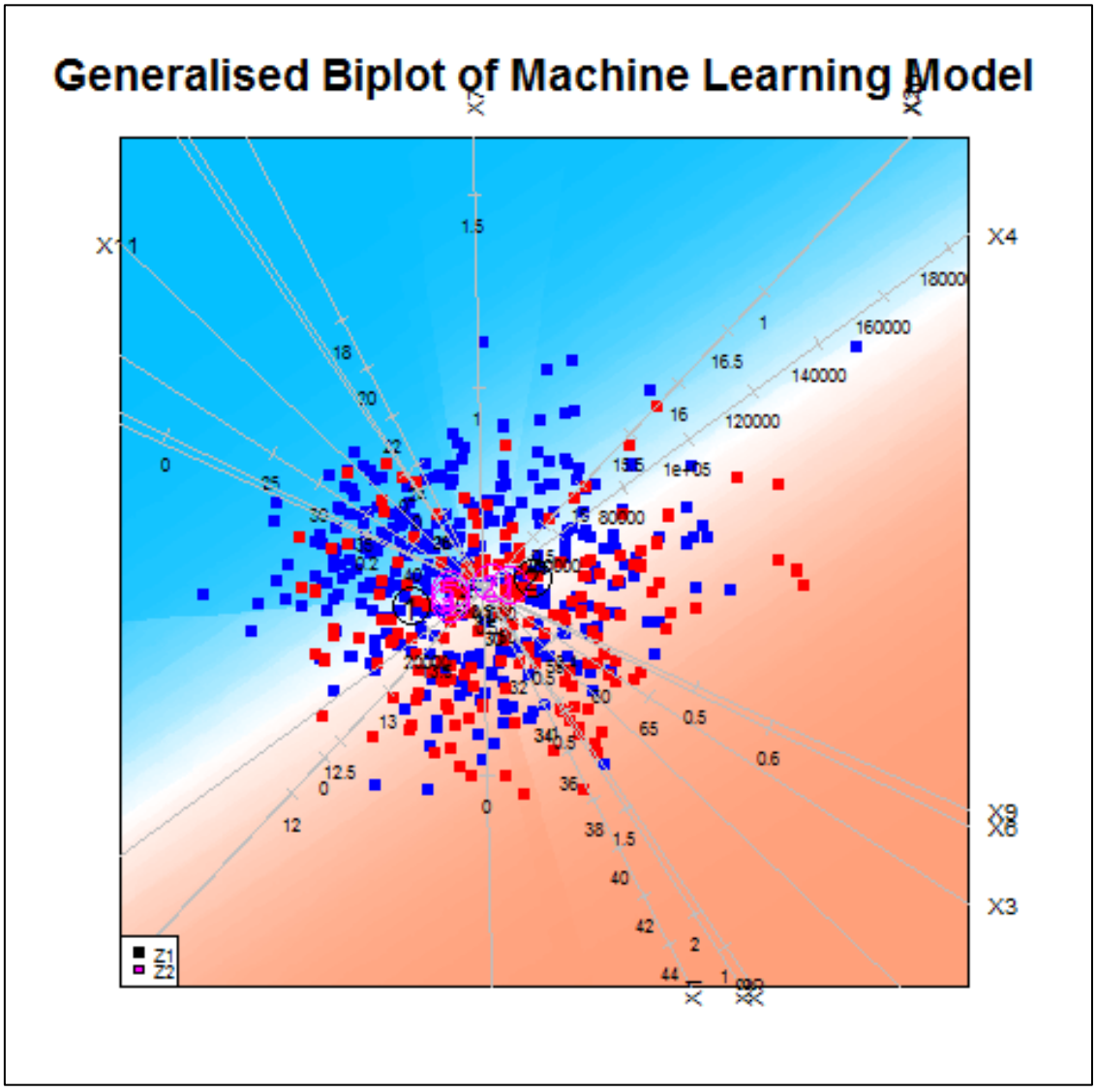


Figure 6-1: Example of a generalised biplot with the output of the machine learning model represented by the colour gradient of red-white-blue representing the range from 0 to 100% prediction

Using this process to visualise the data set and machine learning model output, the generalised biplot can be used to provide global and local fidelity for the two machine learning models.

6.5 Global interpretations using generalised biplots

This section shows how generalised biplots provide globally accurate post-hoc interpretations of the impact of the modelling method and the variables on the predictions for any machine learning model. The generalised biplots will be used to visualise:

- The interaction between the data points and variables in the data set.
  e
- The impact of individual variables and their interactions with the model predictions and whether a variable should be included or excluded from the model. The generalised biplot will be compared against traditional methods of variable selection for GLM and GBM methods, which can assist in the selection of methods and variables that will result in more accurate predictions.

- The output of the machine learning model to give globally faithful interpretations of any machine learning model.

Visualising the structure of, and interactions between, data points and variables

The generalised biplot can be used to give a better understanding of the interaction between the various input variables, and to show if there are any patterns in the distribution of the target variable over the input variables. This will allow us to better interpret the results of the machine learning models.

Biplot visualisations can be used to investigate the following:

- Impact of the distance measure

  The representation of the data points and variables depends on the distance measure that is used and the contribution of each variable to the total distance. Four possible distance measures, each resulting in a different biplot, are considered in this dissertation, and are illustrated in Figure 6-2. Since the choice of distance measure will impact all the subsequent biplots and visualisations, it is important to decide on the most appropriate measure at the start.

  The *first* measure uses the normalised Euclidean distance for continuous variables, normalised rank for ordinal variables and EMC for nominal categorical variables. The contribution of each variable is then weighted such that each variable has a sum of squares

equal to the number of data points n, ensuring that every variable has the same contribution towards the total distance. This is shown in the top left biplot.

The *second* measure uses the normalised Euclidean distance for continuous and ordinal variables but does not adjust their weight. It subsequently uses EMC and adjusts these variables to ensure the sum of squares of the nominal categorical variables are equal to the number of data points n. This is shown in the top right biplot.

The *third* measure uses the normalised Euclidean distance for continuous variables but does not adjust their weight. It subsequently uses normalised rank for ordinal variables and EMC for nominal categorical variables and adjusts the contribution towards the total distance from each varible to n . This is shown in the bottom left biplot.

The *fourth* measure standardises the continuous variables such that each only has a unit range. It subsequently uses normalised rank for ordinal variables and EMC for nominal categorical variables and adjusts the contribution towards the total distance from each varible to n. This is shown in the bottom right biplot.

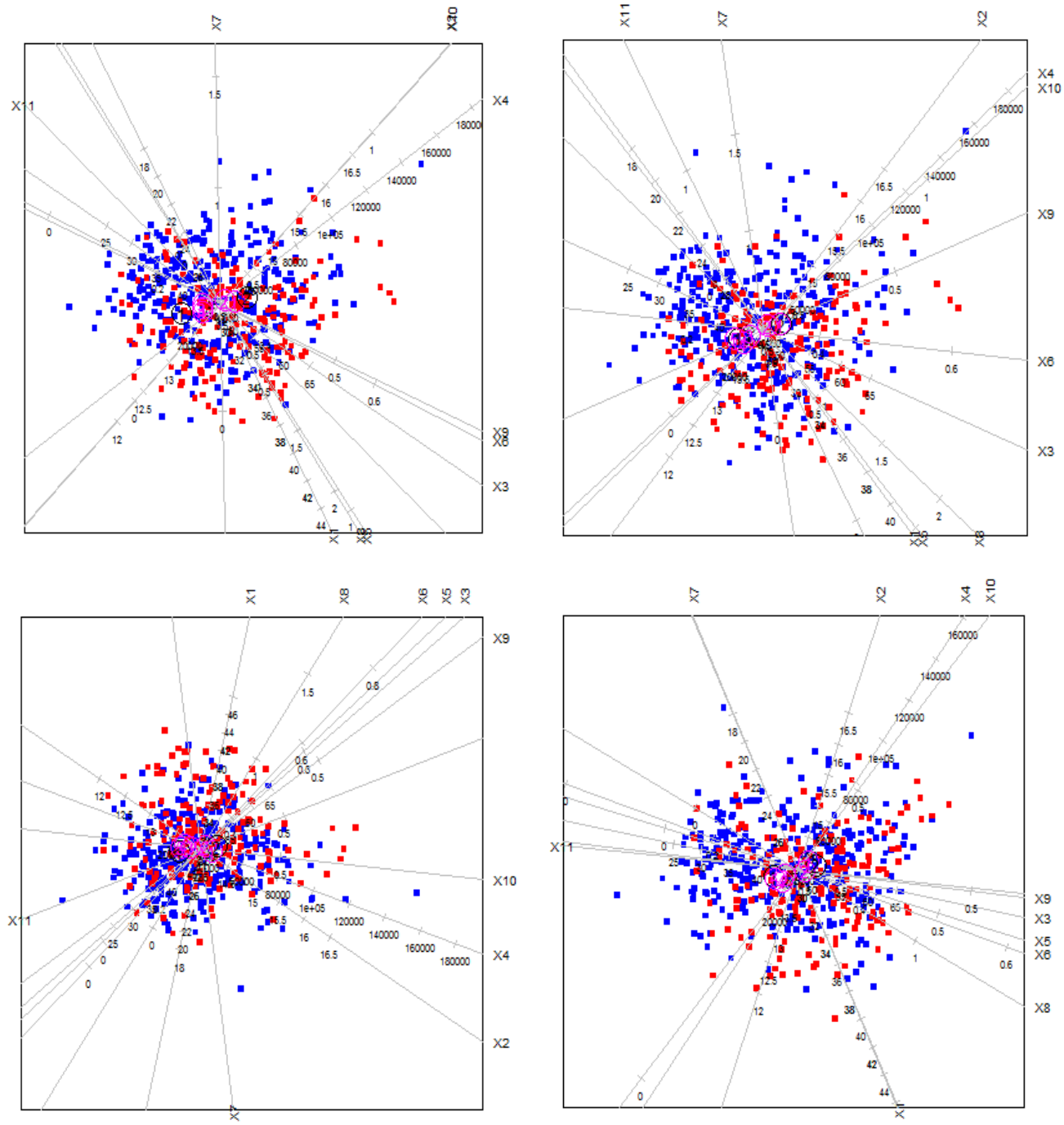


Figure 6-2: Impact on the biplot of different distance measures for continuous, ordinal and categorical variables

The choice between different measures can be informed by looking at the resulting distribution of the data points and the measures of the *biplot quality* and *axes predictivity*. These are shown in Table 6-1 below.

In general, the biplot axes follow the same pattern and can be interpreted in a similar way, although the implied correlations between the variables are different. The bottom left biplot

is transposed compared to the other graphs, due to the way that the software has calculated the SVD of the distance matrix.

Comparing the accuracy measures, the quality of the top two biplots are lower. However, their axes predictivities are higher.

The first measure, with all of the variables reweighted to contribute an equal amount to the distance measure (n sum of squares), is our preferred measure and will be used as the basis of all subsequent biplots. This method is used because in absence of any prior information on the relative importance of the different variables, it is uncertain if giving preference to any one variable will distort the results or not.

Using different measures may give better insights in certain situations, especially if more weight is given to more important variables.

Table 6-1: Quality and axes predictivity accuracy measures of biplots using various distance measures

	Measure 1	**Measure 2**	**Measure 3**	**Measure 4**
Quality of display	**27%**	**30%**	**69%**	**63%**
Continuous Axis predictivity				
X1	98%	98%	99%	98%
X2	100%	100%	100%	100%
X3	98%	98%	98%	100%
X4	91%	90%	91%	85%
X5	34%	35%	26%	28%
X6	34%	34%	34%	33%
X7	50%	56%	43%	44%
X8	68%	68%	53%	54%
X9	36%	36%	35%	36%
X10	40%	41%	36%	36%
X11	47%	50%	47%	47%
Categorical Axis predictivity				
Z1	80%	66%	68%	65%
Z2	28%	28%	28%	26%

- Distribution of data points

Next, the data points can be added to the biplot. Two plots are created and shown in Figure 6-3, namely, one showing the accepted policies highlighted in blue, and the other showing the declined policies highlighted in red. These are the actual values for the data, instead of their predicted values based on the machine learning model.

There is significant overlap of the policy classes, which indicates that no machine learning model is likely to be able to perfectly separate the two classes. However, the accepted policies are much more concentrated in the upper left side of the biplot. There also seems to be very few declined policies in the upper right side of the biplot. The declined polices are more concentrated in the bottom right. The decision model that better separates these areas is likely to be more accurate.

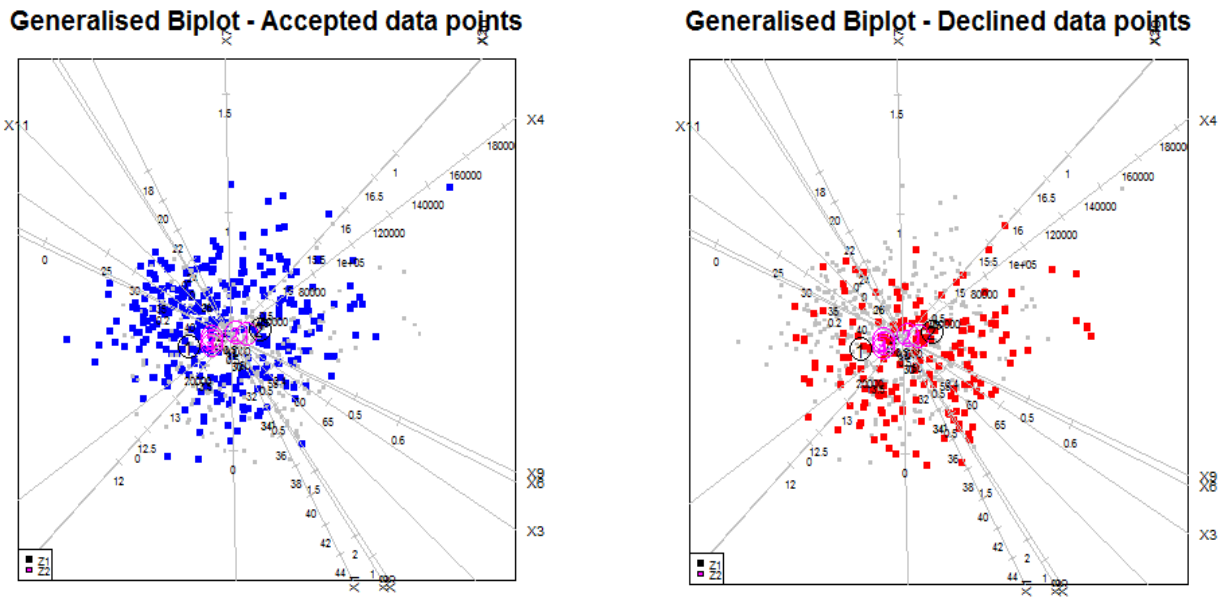


Figure 6-3: Distribution of the two target variable classes across the biplot plane, indicating the correlation between the target class and input variables. The accepted policies are coloured in blue, declined policies in red

- The interaction between the continuous and categorical variables

The biplot of only the variables and CLPs, without any data points, as shown in Figure 6-4, can be used to interpret the correlation between the continuous and categorical variables.

This biplot indicates that each of the variable pairs X_6 and X_9, X_5 and X_8, and X_2 and X_{10}, are very closely correlated. X_{11} is negatively correlated with most of the variables. Variables X_2, X_4, and X_{10} are uncorrelated with the rest of the data. The strong correlation between the pairs of variables may indicate co-linearity, which implies that removing one of the variables in any pair may increase the accuracy of the machine learning model.

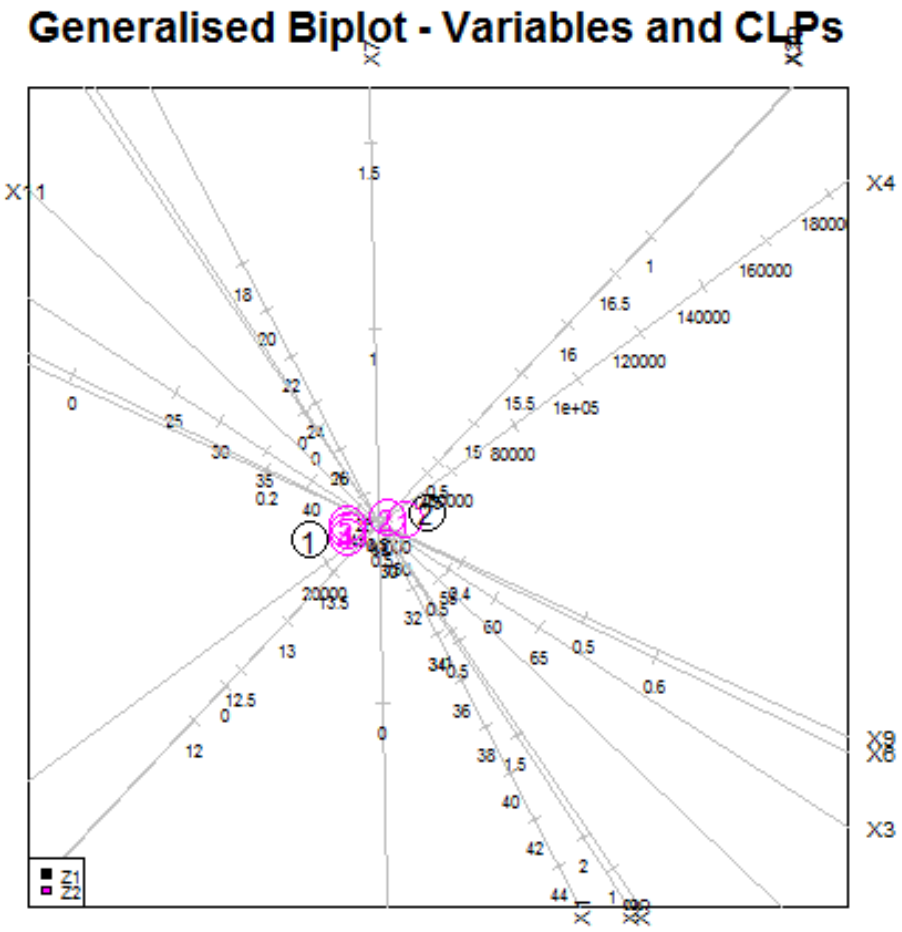


Figure 6-4: A generalised biplot of insurance data without the data points, showing the interaction between the continuous and categorical variables

The two CLPs of Z_1 are clearly separated, indicating that there is a clear pattern in the data – Z_{12} is more positively correlated with variables X_2, X_4, and X_{10}, and slightly correlated with variables X_3, X_6, and X_9. Both CLPs look uncorrelated with the other variables.

On the other hand, the five CLPs of Z_2 are grouped very closely together. This indicates that there is no clear correlation between these CLPs and other variables. Some of the levels

cover a very small area of the biplot plane and are unlikely to be predictive. They could potentially be grouped with other levels to better help separate different groups in the data.

If this correlation structure looks incorrect based on prior knowledge of the data set, this plot could help identify possible data errors that should be corrected before further analysis is done.

- CLP boundaries and the importance of individual category levels

The categorical variables can be further investigated by plotting the different categorical level boundaries, as shown in Figure 6-5 below:

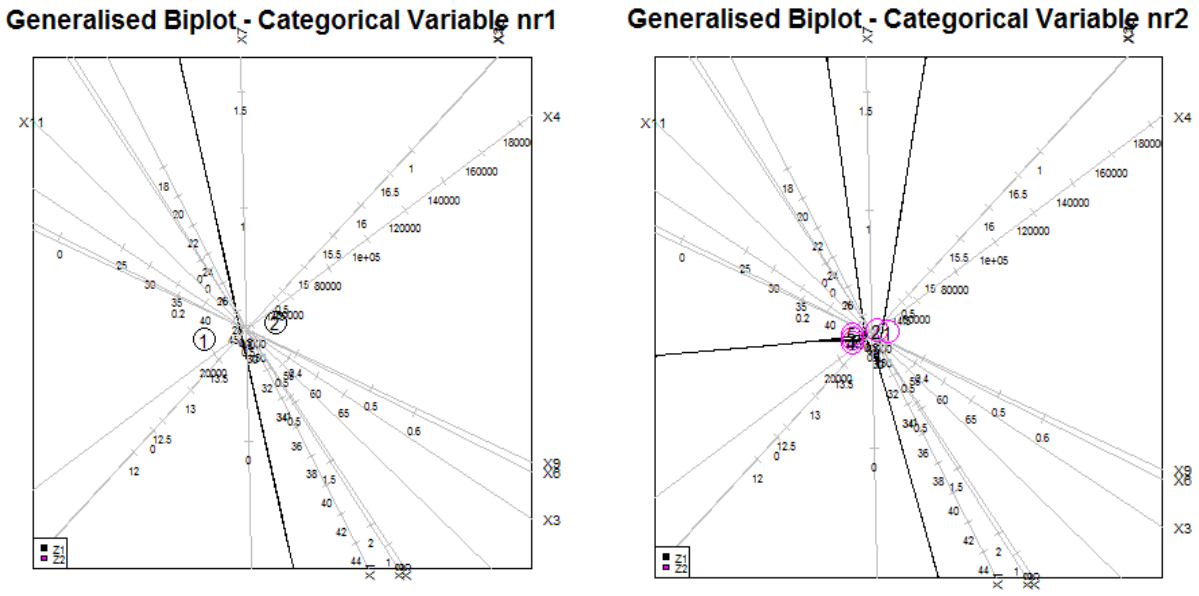


Figure 6-5: The CLPs for categorical variables nr 1 and 2, with their separate category level boundaries

The fact that Z_{11} and Z_{12} both lie far from their boundary indicates that for the first category variables the CLPs are clearly separated.

The boundaries of Z_{23} cover only a very small area of the biplot, which indicates that this category level is unlikely to have significant predictive power and should rather be grouped with either Z_{24} or Z_{25}, which are the closest CLPs. Z_{22} may also be potentially grouped with Z_{21} or Z_{25}.

The different prediction areas that are separated by the CLP boundaries are more clearly visualised by filling in their decision areas, as shown in Figure 6-6:

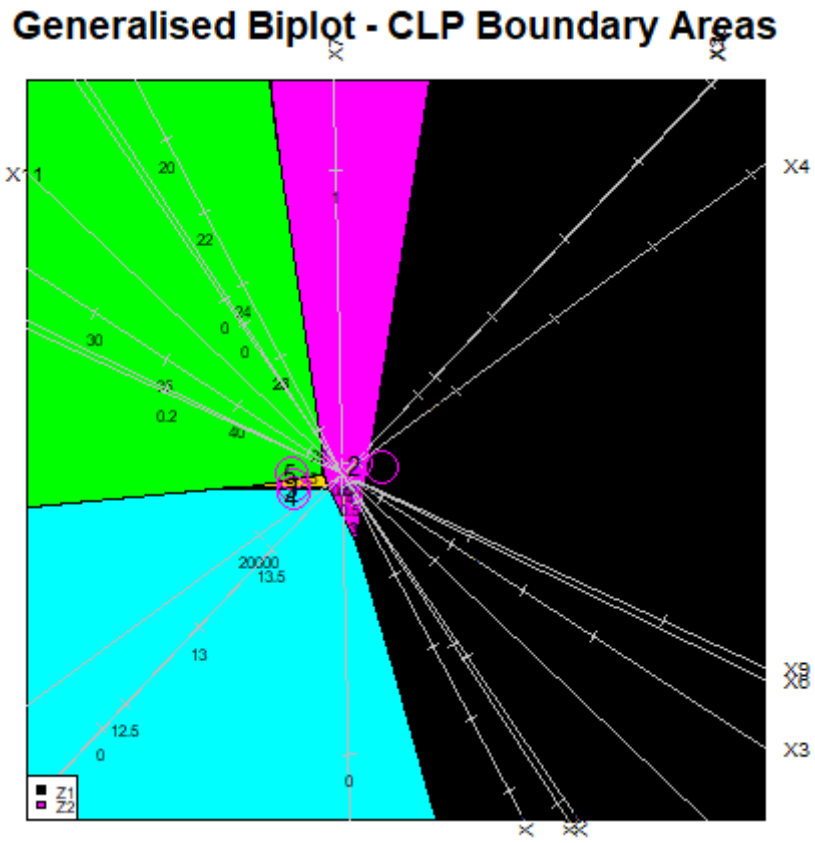


Figure 6-6: CLPs of categorical variable nr 2, with the decision areas filled in to indicate the category level in which each point on the biplot will be predicted.

Having investigated the data points and variables on the biplot, the output of the two machine learning models can be added and the generalised biplot used to provide interpretations of the model that are globally and locally accurate.

GLM model output and variable selection using generalised biplots

In Section 6.3 the assumptions underlying the GLM and GBM were explained and compared.

This section will first explain how traditionally the appropriate variables are selected and the prediction accuracy of a calibrated GLM is measured. This will form a benchmark for the comparison of the benefits of using generalised biplots to perform the same tasks.

Later in this section, a logistic regression model will first be fitted to all of the variables, called the *full model*, after which various accuracy measures will be summarised.

Following this, the p-values and variable importance plots will be compared, and the insignificant variables removed, which will result in an updated model calibration, with corresponding updated accuracy measures. This is the traditional approach to model fitting.

The section will conclude by showing how generalised biplots can be used to investigate the full model, in order to decide which of the variables to include in an updated model, with the resulting accuracy measures shown to compare the results. The final model is the model based on the analysis of the biplot, and will then be used in the subsequent discussions on the output of the GLM.

Step 1.1: Fitting a GLM model with all available variables

The full model uses all 11 continuous and ordinal variables as well as the 2 categorical variables. The key results, using a 70% decision boundary, are as follows:

- Overall accuracy: 69.31%
- Positive prediction accuracy: 82.94%
- Data prevalence above decision boundary: 58.55%
- AUC value: 74.6%

The overall accuracy of 69.31% compares very poorly against the no-information-rate of 69%. On this measure, the model does not perform better than random predictions. However, the positive prediction accuracy value of 82.94% illustrates that the model does provide insight into the factors driving the acceptance of policies. This is at the expense of only applying the model to 58.55% of the test data.

Step 1.2: The results of the GLM model after selecting the significant variables

The model summary provided by R gives the model coefficient estimates and p-values used to determine which variables are significant or not. These are shown in Table 6-2.

Table 6-2: Model summary for the full GLM

Variable	Estimate	p-value
X1	-0.13	0%
X2	0.10	0%
X3	-0.02	0%
X4	**-0.00**	**27%**
X5	-0.93	0%
X6	**0.02**	**80%**
X7	**-0.01**	**88%**
X8	-0.75	0%
X9	-0.22	0%
X10	-0.17	0%
X11	-0.12	0%
Z1-2	-0.18	0%
Z2-2	-0.22	0%
Z2-3	**0.07**	**19%**
Z2-4	-0.48	0%
Z2-5	**0.23**	**9%**

The variable importance plot is shown next in Figure 6-7. Note that the base levels of each categorical variable – Z_{11} and Z_{21} – are not shown separately. They need to be removed from the coefficient estimation process in order to avoid over-specifying the model and causing co-linearity.

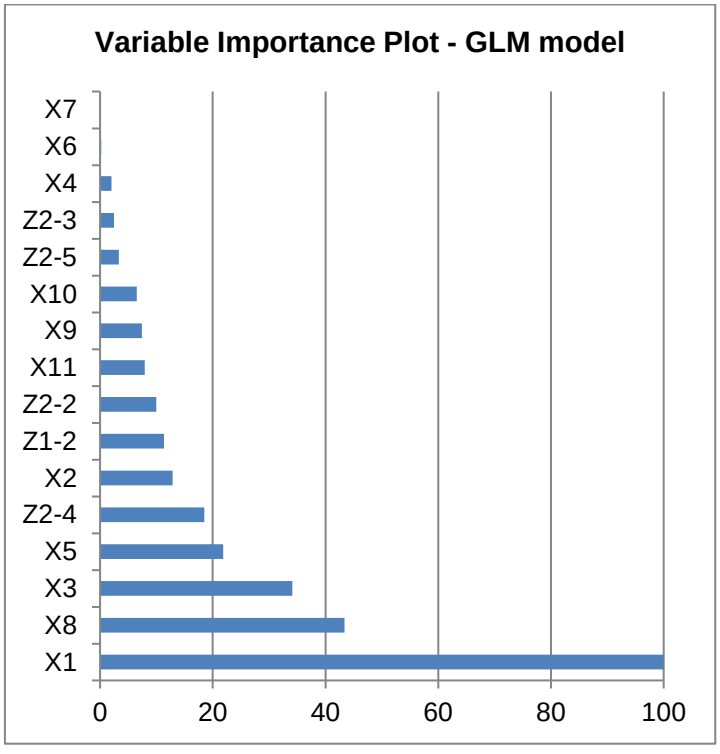


Figure 6-7: Variable importance plot for the full GLM

The significance tests and the variable importance plot both indicate that variables X_7, X_6, X_4, Z_{23}, and Z_{25} should be dropped, as their p-values are above 5% and they have the lowest impact on the model error rate. Therefore, the updated model will exclude these 5 variables. The key results for the updated model, again using a 70% decision boundary, are as follows:

- Overall accuracy: 69.19%
- Positive prediction accuracy: 82.70%
- Data prevalence above decision boundary: 58.79%
- AUC value: 74.6%

Therefore, the updated model resulted in a slightly worse performing model, based on its positive prediction accuracy. This reflects the problem with using p-values and VIP output as the basis for model selection. Although these variables are not statistically significant, they do help explain small parts of the distribution of the target variable that are lost by removing these input variables. A better method for variable selection is to attempt various combinations of variables and measure the cross-validation error of each until an optimal combination is found. This is possible with 13 variables, but it becomes more and more computationally unfeasible as the number of variables increases.

For this reason, the generalised biplot will be used next, in order to determine whether it can simplify and improve the variable selection process.

Step 1.3: Selecting variables by inspecting the generalised biplot of the full model

The first steps for using the generalised biplot include plotting the data for different distance measures, separately investigating the two target classes, and investigating the distribution of the variables, especially the CLPs. These steps were all completed in the first part of this section.

Next, the output of the full model should be plotted, as shown in Figure 6-8. Subsequently, the continuous and ordinal variables axes, followed by the categorical variable CLPs, must be investigated to determine which variables seem to be significant in predicting which class a data point belongs to and which variables can be removed from the model.

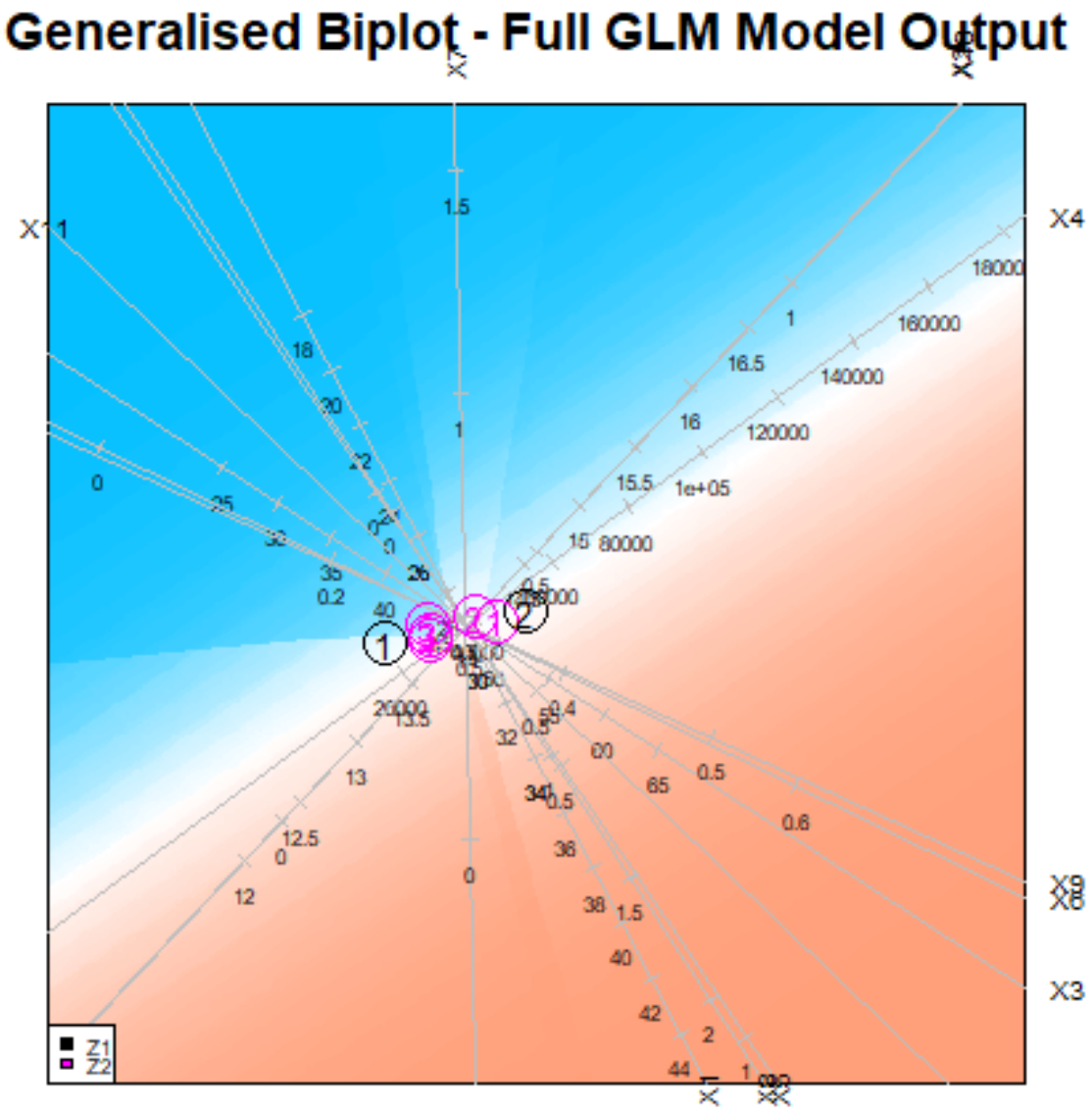


Figure 6-8: Generalised biplot of the full GLM, without the data points and with all the CLPs

The biplot above clearly shows the linear decision boundary that is expected from a linear model, running from the top right to bottom left. The direction of the boundary meets our expectations based on the initial investigation, where it was seen that the accepted policies were grouped in the top left side and the declined polices were grouped in the bottom right. Therefore, the model is functioning as expected.

Next, the direction of the variable axes can be compared to the changing colour of the GLM output and the direction of the linear boundary. The following conclusions can be made:

- All variables cause a gradual change in colour, which implies that the exclusion of any variables is difficult.

- The strong correlation between pairs of variables could represent co-linearity, so that, removing one of each of the pairs of variables may improve the model prediction accuracy. Other distance measures indicate that only the pair of X_4 and X_{10}, and of X_1 and X_5 may be co-linear.

- Variable X_4 runs almost parallel to the decision boundary. Therefore, a change in variable X_4 has very little impact on the decision output and indicates a lack of significance in the prediction of the model output. It will therefore be removed from the final model.

Finally, the categorical variables will be investigated. From the initial investigation, it was seen that Z_{23} only covered a small area of the biplot, which indicates that it is unlikely to have any predictive power and should rather be grouped with either Z_{24} or Z_{25}. This can be investigated further by adding the decision boundaries of the second categorical variables to the biplot, as shown in Figure 6-9.

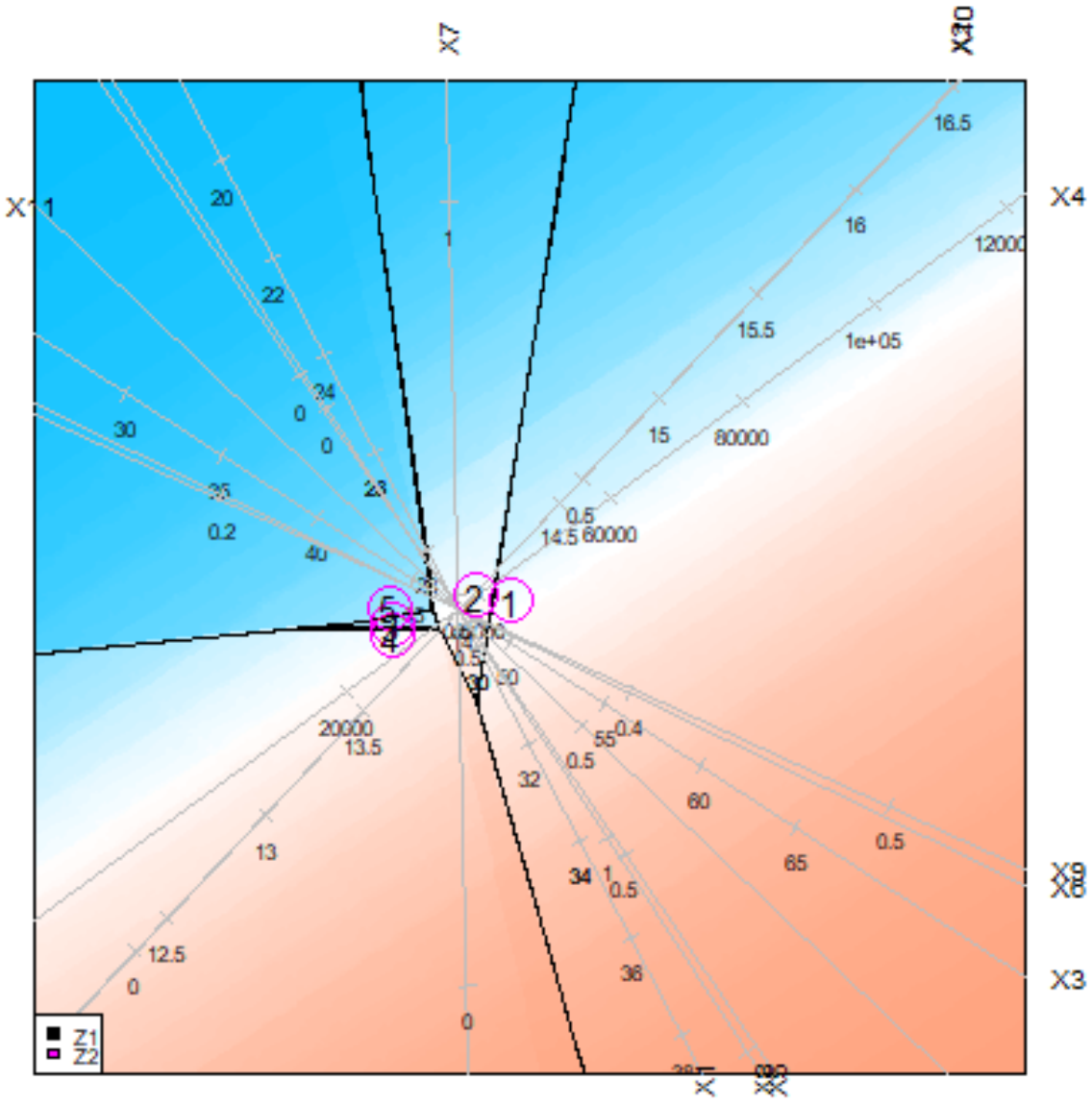


Figure 6-9: Full GLM with classification boundaries for the second categorical variable

The boundaries between Z_{21}, Z_{22}, Z_{24}, and Z_{25}, match the changing colours of the GLM, indicating that these four category levels impact the model predictions and should be retained. Only Z_{23} should be removed and combined with Z_{24}.

The same analysis can be done for the categorical variable Z_1, as shown in Figure Figure 6-10. Although this variable seems significant because each level covers a wide area, its classification boundary doesn't match a change in the colour of the GLM. This indicates that this variable is less significant than would be suspected, and could indicate that the two category levels should be combined and therefore the variable removed from the analysis. However, both variable levels will be retained.

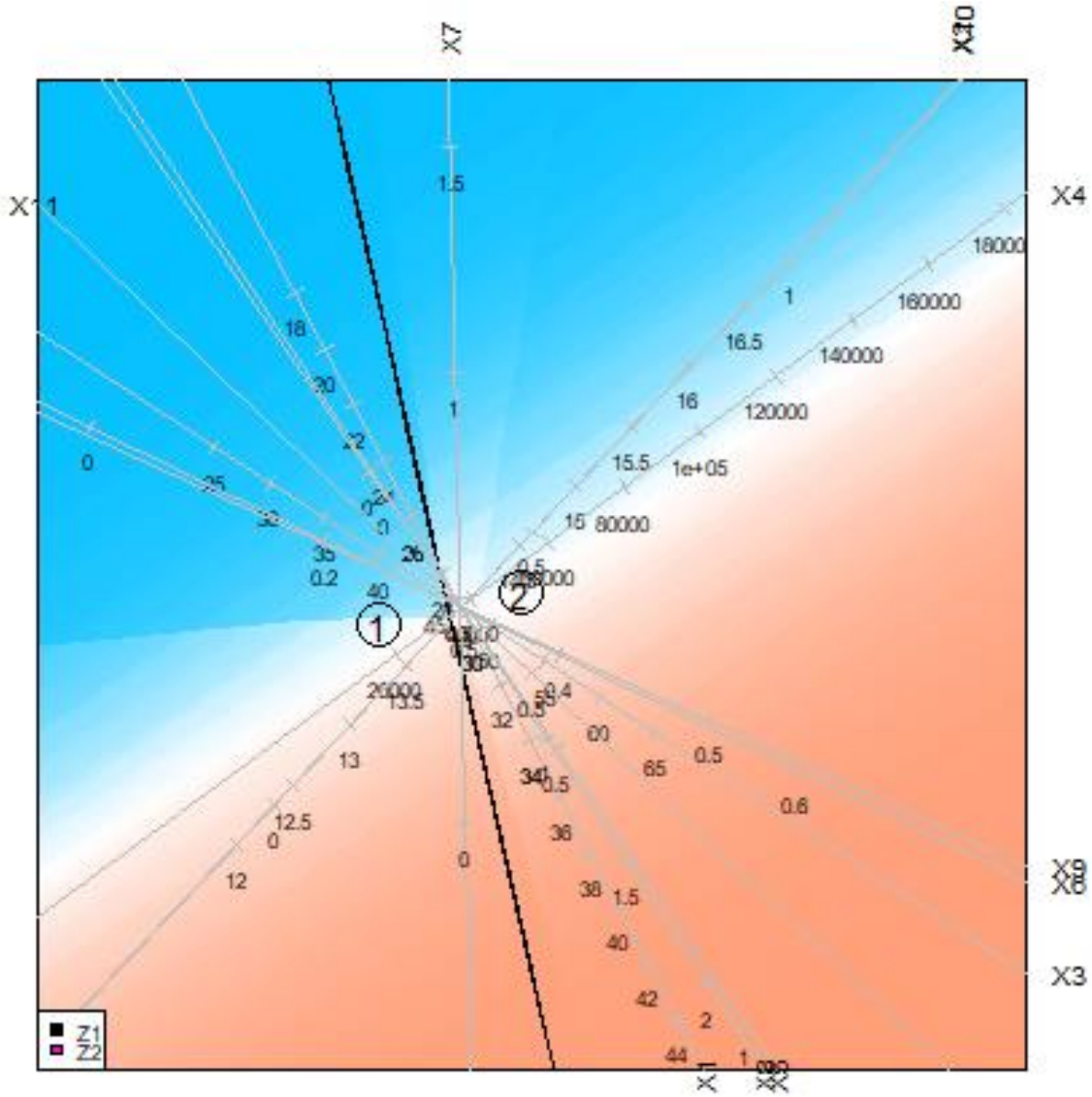


Figure 6-10: Full GLM model classification boundaries for the first categorical variable

Based on the analysis above, only the variables X_4 and Z_{23} will be removed before updating the GLM. The key results for the updated model, again using a 70% decision boundary, are as follows:

- Overall accuracy: 69.14%
- Positive prediction accuracy: 82.71%
- Data prevalence above decision boundary: 58.70%
- AUC value: 74.5%

This is very similar to the results of the reduced model that used the p-values for variable selection and indicates that the generalised biplot can help with the variable selection process and can be interpreted in a similar way to p-values produced by linear models. However, this variable selection methodology is more intuitive and easily understood than p-values.

The biggest advantage is that this method can be applied to any machine learning model. In the next section, the same process will be applied to the stochastic gradient boosting model. The p-values will not be available, but variable selection using VIPs will be compared to using the biplot.

GBM model output and variable selection using generalised biplots

Step 2.1: Fitting a GBM model with all available variables

This section starts by measuring the improvement of using a stochastic gradient boosting model (GBM) compared to the logistic regression model (GLM). The hyper-parameters were selected by comparing the AUC score of a 5-fold cross-validated model assessment of a range of possible hyper-parameters. The final selection was a model with 6 000 trees, each tree having an interaction depth of 7 nodes and a shrinkage parameter of 0.005. For all of the models, only 50% of the data has been considered in each of the trees.

The key results, using a 70% decision boundary, are as follows:

- Overall accuracy: 73.21%
- Positive prediction accuracy: 83.49%
- Data prevalence above decision boundary: 63.40%
- AUC value: 77.98%

This is a significant improvement over the results of the GLMs and shows the benefit of using boosting compared to linear models.

Step 2.2: The results of the GBM model after selecting significant variables

The variable importance plot of the full GBM is shown in Figure 6-11.

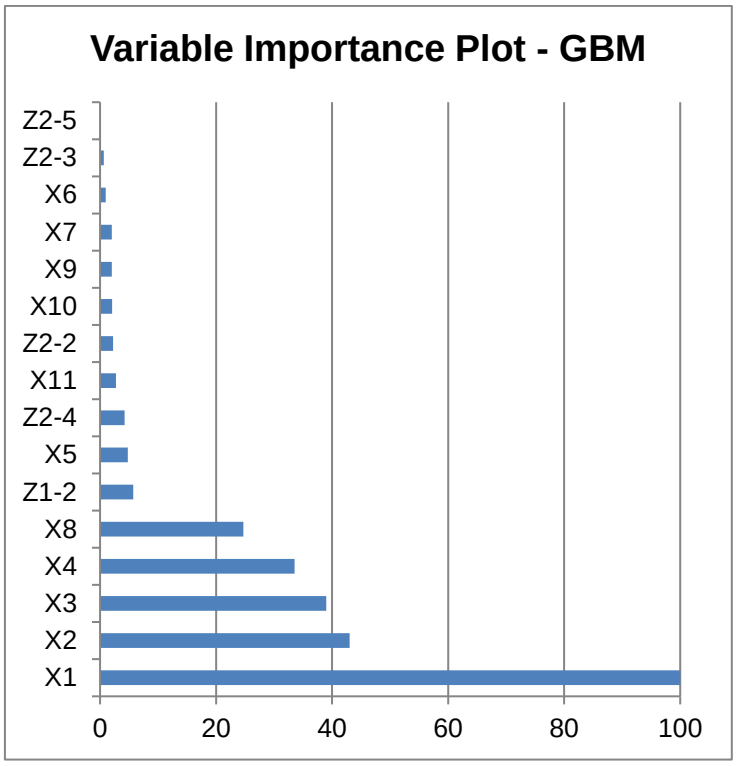


Figure 6-11: Variable importance plot for the full GBM

The VIP for the GBM has a very different profile compared to the VIP of the GLM shown before. The main change is that variable X_4, which was removed from the GLM based on both the p-value and the generalised biplot, is now the 4th most important variable. Variables X_6, X_7, Z_{23}, and Z_{25} are still the least important.

Based on the VIP, variables X_6, X_7, X_9, X_{10}, and X_{11} were removed before updating the model. CLP Z_{22} was combined with Z_{21}, and CLPs Z_{23} and Z_{25} were combined with Z_{24}.

The new model had 10 000 trees, each of which had an interaction depth of 6 nodes and a shrinkage parameter of 0.005.

The key results, with the usual 70% decision boundary, are as follows:

- Overall accuracy: 73.43%
- Positive prediction accuracy: 83.28%
- Data prevalence above decision boundary: 64.12%
- AUC value: 77.78%

The overall accuracy and data prevalence have increased slightly, while the positive prediction accuracy has reduced slightly. This implies that the selection of the appropriate machine learning

method has a much more significant impact on the accuracy of the predictions than the selection of the variables to include in the model.

Step 2.3: Selecting variables by inspecting the generalised biplot of the full model

As with the GLM, the output of the full GBM can be visualised with the biplot, as shown in Figure 6-12.

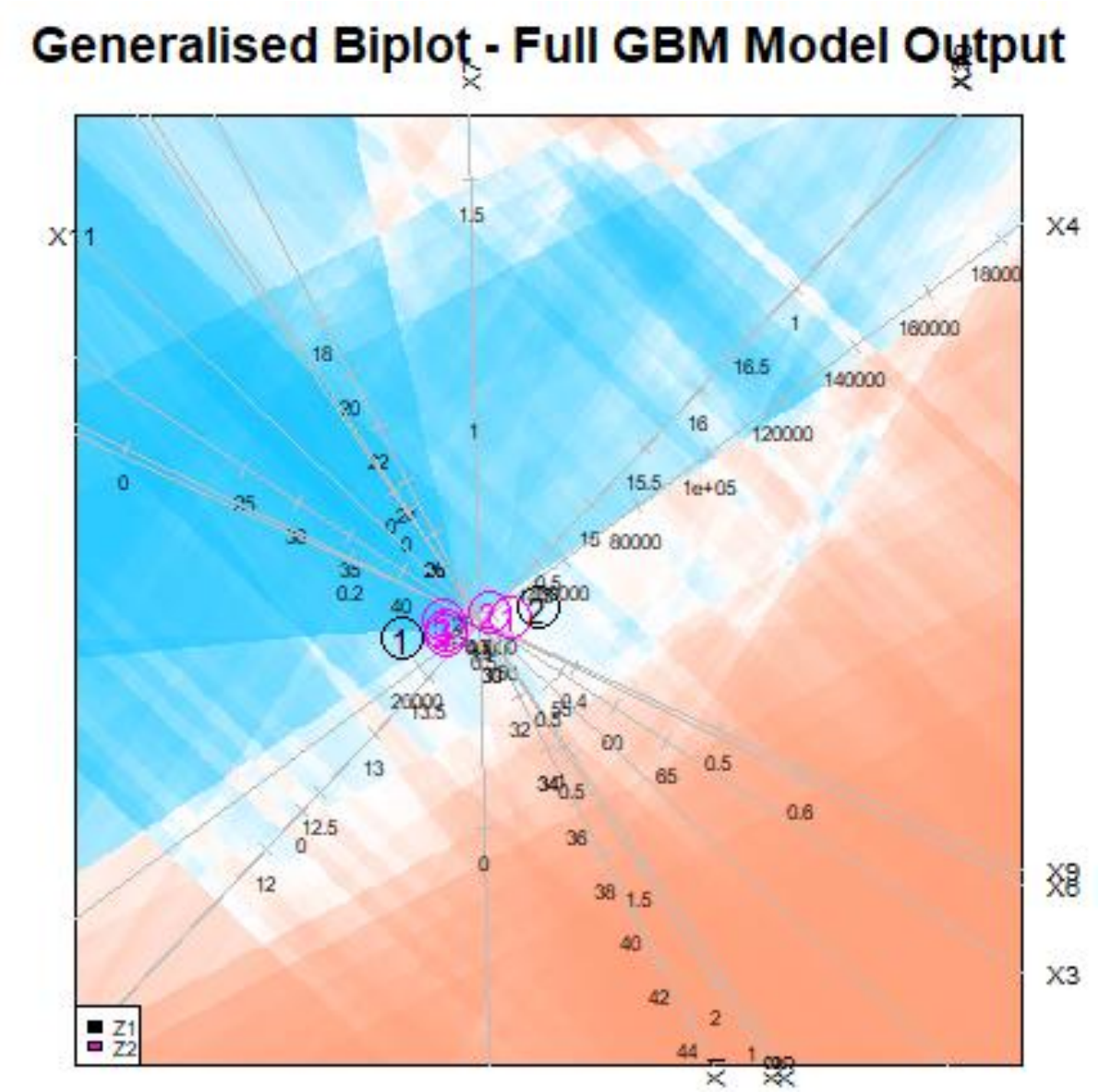


Figure 6-12: Generalised biplot of the full GBM, without the data points and with all the CLPs

The biplot of the output of the GBM shows a convex shape on the boundary where the colour of the model ouput predicts declines (red area) and where it is uncertain (white area). The areas that predict accepted (blue colour) policies are also sometimes surrounded by the output of the decline areas. This is expected from a model that can allow for the non-linear interactions between variables. The convex shape and dark blue predictions in the top left of the biplot follows the distribution of the data points as shown earlier in this section. Therefore, the model is functioning as expected.

Next, the direction of the variable axes can be compared to the changing colours of the output of the GBM and the implied linear decision boundary, leading to the following conclusions:

- Many of the variables are at some point perpendicular to the convex shape of the changing decision boundary, which could indicate that in that particular region, these variables are important in the prediction model. Two notable exceptions are X_7 and X_{11}, which do not seem to have any impact on the model output, and these two variables will therefore be removed from the calculation of the reduced final model.

  As with the GLM, variable X_4 runs almost parallel to the main decision boundary. However, in the case of the GBM, the gradient of the colour of the model output does indeed change as the variable changes, and therefore, this variable will be kept in the model.

- The potential co-linearity between the correlated variables will again be ignored and these variables retained.

As with the GLM, the category level boundaries of the different categories show if a change in the category level has an impact on the model output. The category level boundaries for variable Z_2 are shown in Figure 6-13. The model output does not seem to change between Z_{21} and Z_{22} and these two levels can be combined. Z_{23} is again removed since it covers a small area of the biplot and can be combined with Z_{24}.

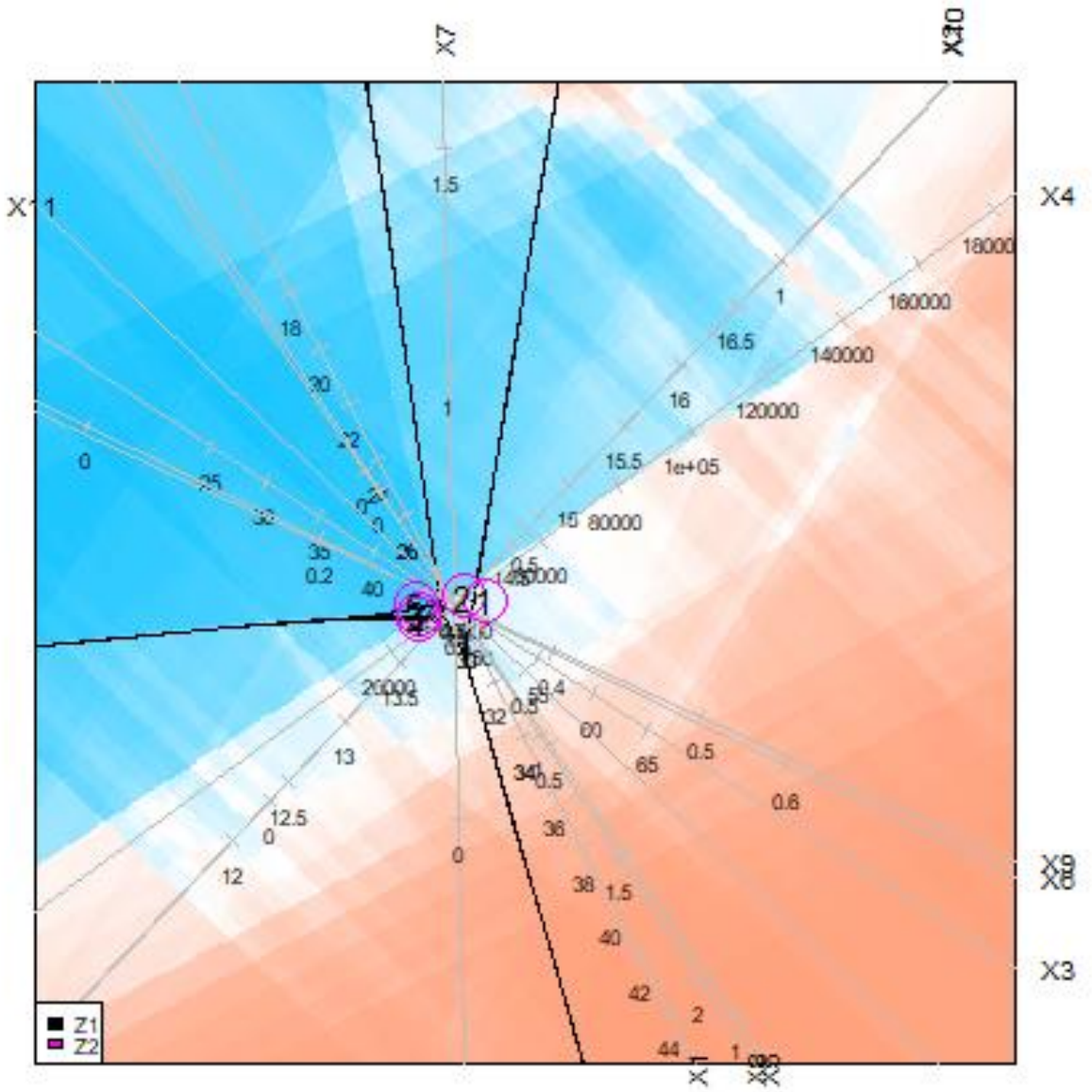


Figure 6-13: Full GBM model classification boundaries for the second categorical variable

The category level boundaries for variable Z_1 are shown in Figure 6-14. For the full GBM, the change from Z_{11} to Z_{12} has an impact, especially in the areas with relatively high predicted acceptance rates, and this variable should therefore be retained.

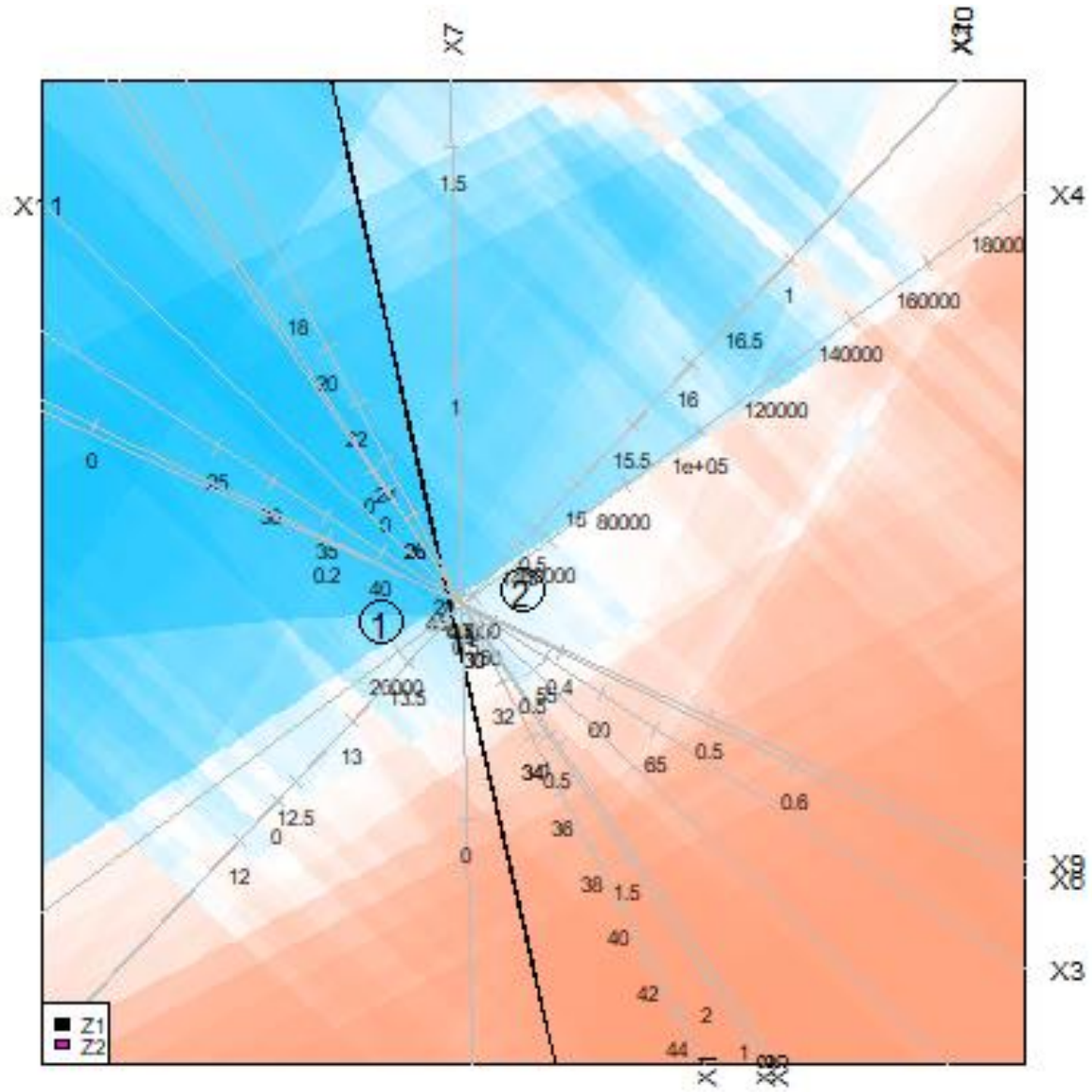


Figure 6-14: Full GBM with classification boundaries for the first categorical variable

Based on this analysis, variables Z_{22}, Z_{23}, X_7, and X_{11} were removed from the final model. The new model had 10 000 trees, each of which had an interaction depth of 6 nodes and a shrinkage parameter of 0.002. The key results, using a 70% decision boundary, are as follows:

- Overall accuracy: 73.52%
- Positive prediction accuracy: 83.32%
- Data prevalence above decision boundary: 64.19%
- AUC value: 77.92%

These results are almost the same as those that were obtained using the VIP to select variables, with each of the measures slightly higher. This observation again illustrates how the generalised biplot can be used for variable selection, resulting in similar results as using traditional variable selection processes.

This is only an illustration of the process followed to find the best model and optimal combination of variables. Improved models using further variable transformations and selections can result in more accurate results. However, what has been illustrated here is the process of finding and understanding the optimal models and variables using the output of generalised biplots. The results are consistent with using variable importance plots or p-values.

Visualising the output of the model to give globally faithful interpretations

The last section will explain the ability of generalised biplots to determine globally faithful interpretations of the output of any machine learning model. Therefore, the biplots will:

- Explain the global impact that the variables have on the predictions, and compare it to traditional methods of giving global interpretations
- Visualise why one machine learning model results in more accurate or appropriate results than another model based on the same data

The analysis will use the final GLM and GBM that were calibrated in the previous section, and the biplots are created using the 500 sample test data points.

Explaining the global impact of the variables

The biplot can be used to provide a global overview of the impact of each variable on the model predictions. If the decision boundary is linear, then there is a clear distinction between accepted and declined regions in the biplot. This simplifies the interpretation of the biplot compared to biplots of non-linear machine learning models.

If a variable axis increases in the direction of the blue area of the biplot, it in general indicates that an increase in the variable value will increase the prediction probability of the model. This interpretation will be incorrect if other variables increase in the same direction but have a bigger but opposite impact on the prediction probability of the model, causing the overall prediction probability to increase even if the impact of the less important variable decreases.

Therefore, the impact on the prediction probability of very positively or very negatively correlated variables should be considered together, and not in isolation.

Variables that are perpendicular to the decision boundary have the strongest impact on the prediction probabilities; that is, a small change in their values will cause a big change in the prediction probabilities. Variables that are parallel to the decision boundary are more difficult to interpret. Overall, they indicate variables that have a small impact on the decision model; that is, a big change in the variable implies a small impact on the prediction probability. Their impact is therefore also more likely to be masked or confused by the impact of other variables.

For non-linear models, the interpretation of the impact of a variable varies depending on how the decision areas that are visualised on the biplot vary. The same variable can cause the predicted probabilities of the model to increase or decrease, depending on the relative values of the other variables in an area. In addition, there will be less clear perpendicular or parallel intersections between the variable axes and possible decision boundaries or colour gradients. But this is representative of the nature of non-linear models, namely, especially complex models with ensembles of outcomes or multiple levels of interaction. Therefore, the overall impact and interactions of variables will still be indicated by the biplot, but they will be more open to interpretation.

The CLPs of categorical variables produce clear category level boundaries between variables. This makes it easier to see where they have an impact on the prediction probabilities – this is true for linear and non-linear models. The main problem is that the CLPs only predict their category level values in the area of the biplot that they represent. Therefore, the category level impact and interactions are only shown in that area. If the non-linear interactions will cause the impact of the category level value to be different in areas that are outside of the category level boundary, this will not be shown by the biplot. However, the biplot will place the CLP in the area where it most often occurs with the values of the other variables in the data. Consequently, the global interpretation of the impact of the CLP will still be accurate.

For linear models, the impact of the base category level of each categorical variable is included in the intercept term, so that the coefficients of the categorical levels in the GLM are always relative to the base category level, and the impact of the different levels should be considered relatively to each other.

Linear coefficient estimates and the generalised biplot of the final GLM model

For the GLM, the linear coefficient estimates provide both a global and local interpretation of the impact of the variables on the prediction.

The coefficients of the final GLM, after removing X_4 and combining Z_{23} and Z_{24} into Z_{23}, are shown Table 6-3.

Table 6-3: Coefficient estimates of the final GLM

Variable	Estimate
X1	-0.13
X2	0.11
X3	-0.02
X5	-0.93
X6	0.02
X7	-0.01
X8	-0.74
X9	-0.22
X10	-0.17
X11	-0.12
Z1-2	-0.22
Z2-2	-0.21
Z2-3	-0.18
Z2-5	0.25

If a coefficient is positive, an increase in the variable value causes an increase in the prediction probability of the model, with the biplot output becoming more blue. For this model, only variables Z_{25}, X_2, and X_6 have positive coefficients.

All of the other coefficients are negative, so that an increase in the variable decreases the prediction probability, causing the biplot output to become more red.

A coefficient should also be considered in relation to the scale of the variable. The coefficient for X_3 is -0.02, but the data ranges from 25 to 65. The biggest possible contribution is therefore +1.3. The value for X_5 is -0.93, but the value only ranges roughly between -1 and 1, and hence the biggest possible contribution is only -0.93. If the variables are standardised before being included in the calibration of the machine learning model, these variable coefficients and their interpretations will change.

These interpretations can be compared to the interpretation of the biplot of the final GLM, as shown in Figure 6-15. The rotation of the biplot is different from the biplot shown earlier, due to the MDS calculation of the reduced final model rotating the data in a different way. However, the interpretation and relationships that are discussed are unchanged by the rotation.

Variables X_{10}, X_8, X_3 and X_2 (and to an extent X_1) are perpendicular to the decision boundary and increases in these variables result in decreases in the prediction probability. This is consistent with the interpretation based on the coefficients above, except for variable X_2. Therefore, the impact of variable X_2 is masked by that of the other three correlated variables.

Variables X_5, X_6, X_7, X_9, and X_{11} are all nearly parallel with the decision boundary. Apart from variable X_5, this is consistent with the small coefficients in Table 6-3. The axis direction of variables X_6 and X_7 relative to the change in colour of the model output is the opposite to that implied by the coefficients. The impact of these two variables are masked by their surrounding variables.

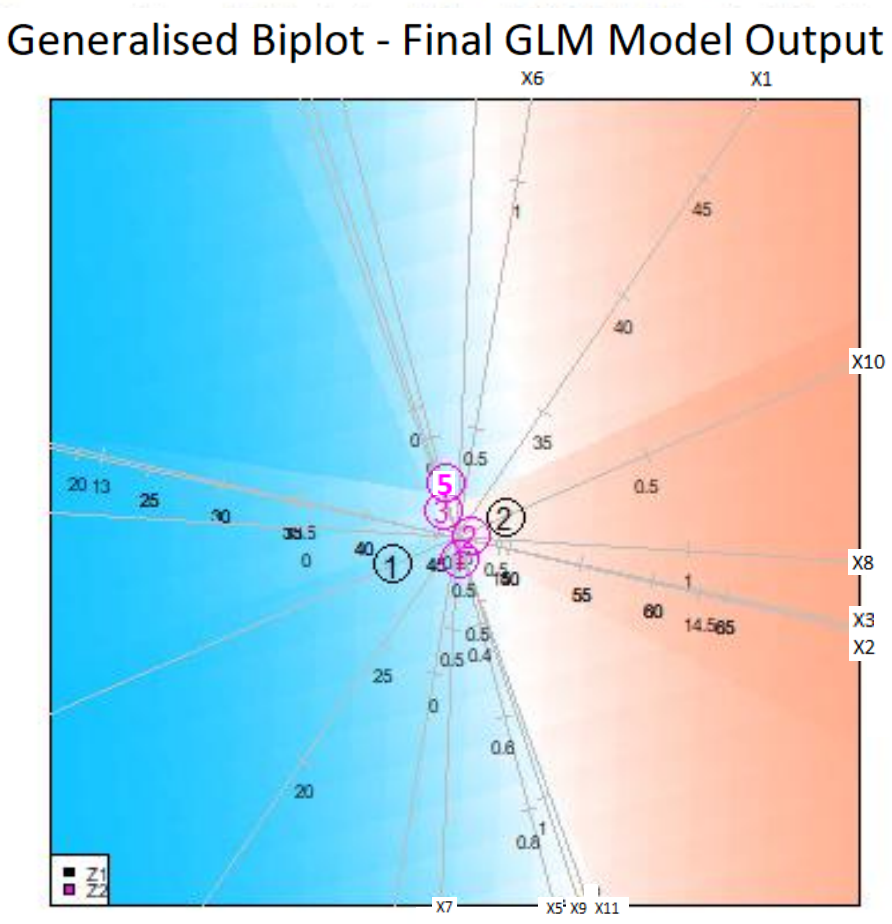


Figure 6-15: Generalised biplot of the final GLM

The impact of the two categorical variables can be investigated by plotting their category level boundaries, as shown in Figure 6-16.

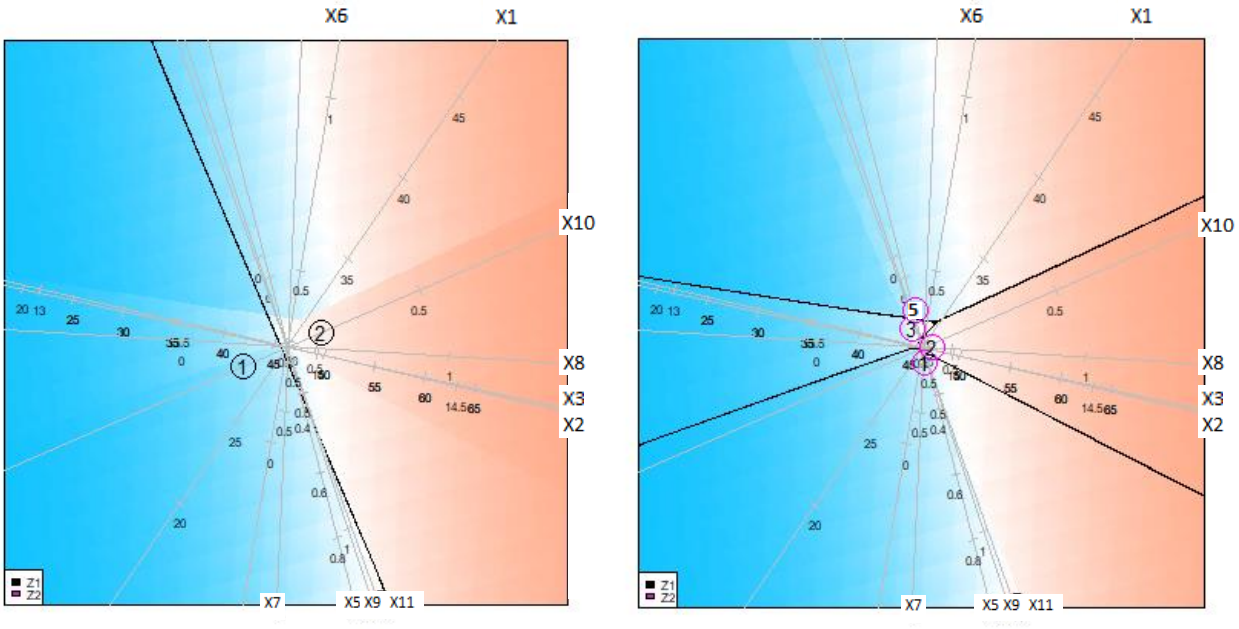


Figure 6-16: Generalised biplot of the category level boundaries of the final GLM

Category level Z_{12} is on the red side of the biplot, which is consistent with the negative estimate of the variable's coefficient. The CLPs of category Z_2 are parallel to the decision boundary, which makes their interpretation difficult. Relative to Z_{21}, Z_{22} is on the red side of the biplot, which is consistent with the negative estimate of the variable's coefficient. The coefficient of Z_{23} is less negative than Z_{22}, which is also consistent with the biplot. The coefficient of Z_{25} is positive, which cannot be interpreted from the biplot. However, Table 6-4 of the categorical variable axes predictivity shows that Z_{25} is very poorly represented by the biplot. Only 11 of the 149 data points that are in category level 4 are correctly predicted as such by the biplot. Overall, the biplot only predicts 20 of the data points to be in category level 4. Therefore, the interpretation from the biplot for this CLP will not be reliable.

Table 6-4: The biplot categorical variable axes predictivity table for categorical variable nr2

		Actual Values			
		1	**2**	**3**	**5**
Predicted Values	**1**	85	55	44	51
	2	38	21	20	34
	3	39	20	20	53
	5	2	6	1	11

To conclude, the interpretation of the impact of the variables from the biplot is similar to those provided by the coefficients of the GLM. The biplot has the advantage of also showing the impact of the scale and range of the variables, as represented by the tick marks on the variable axes and the colour gradient change of the model output along the axes. This allows the actual impact of two variables to be more easily compared, as the coefficient estimate without considering the variable scale and range can be misleading.

The biplot also indicates the correlation between the variables, which enables the biplot to provide a globally faithful indication of the combined effect of variables on the prediction probabilities, and to show where the impact of some variables overpowers and therefore mask the impact of others.

Partial dependence plots and the generalised biplot of the final GBM model

For the GBM, there are no coefficients to give global interpretations of the model's predictions. An alternative way to obtain these, is to use partial dependence plots.

Partial dependence plots (PDP) provide a global interpretation of the impact of a variable. An increasing or decreasing PDP indicates whether an increase in the variable will cause an increase or decrease in the prediction probability. Therefore, the PDPs of the most important variables (based on the VIP) will be compared to the output of the model in the biplot.

The main drawback of PDPs is that the visualisation of the interaction with other variables is limited to low-dimensional views. Showing the interaction between three or more variables requires a trellis of plots. The underlying trees produce discontinuous piecewise constant models and no smoothing function is applied, resulting in sharp discontinuities in the PDPs. (Hastie, Tibshirani and Friedman, 2008).

The final GBM was created after removing X_7 and X_{11} and combining Z_{21} and Z_{22} into Z_{21} and Z_{23} and Z_{24} into Z_{23}. The top six variables for the final GBM, based on the VIP, are X_1, X_8, X_3, X_2, X_4 and Z_{12}. Their PDPs can be seen in the trellis plot in Figure 6-17.

The PDP plots show a non-linear impact for X_1, X_2, and X_4. The continuous variable X_1 shows a significant reduction in the prediction probability once the variable value exceeds 32. After variable X_2 increases past 16 and X_4 increases past 50 000, the prediction probability falls significantly and then stays level. Variable X_2 is already log transformed and based on the PDP variable X_5 should also have been log-transformed in order to ensure that the entire range of the variable has an impact on the model.

Variable X_3 has a near linear impact on the predictions. Variables X_8 and Z_{12} are ordinal and binary variables. The step change in the PDP shows that, once these variables increase over 0.5, their impact on the prediction probability falls significantly.

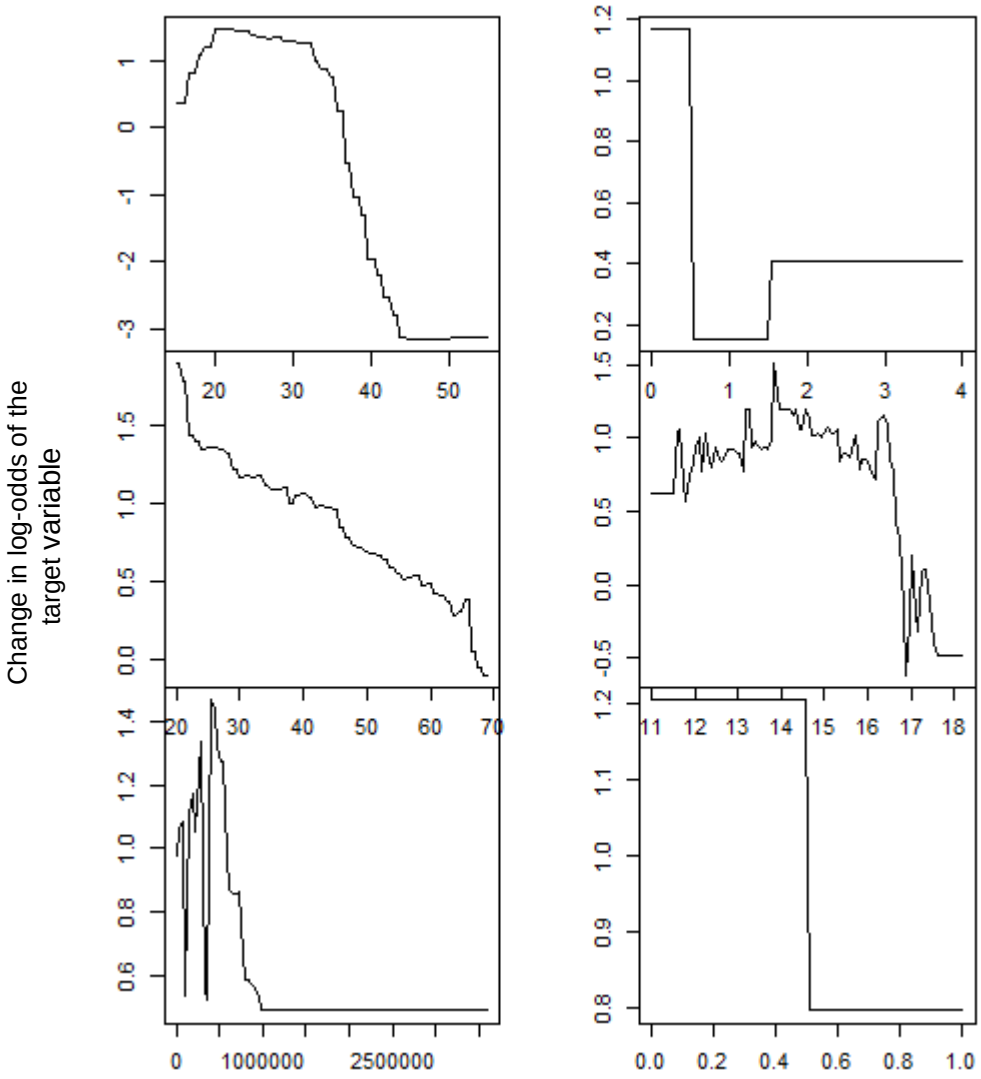


Figure 6-17: Partial dependence plots of the top six most important variables of the final GBM: From the top left to bottom right by rows, they represent the PDPs of X_1, X_8, X_3, X_2, X_4 and Z_{12}

The above interpretations can be compared to the interpretation of the biplot of the final GBM, shown in Figure 6-18. This biplot shows two almost linear decision boundaries that separate the accepted and declined predictions, with a wide area in which the model is uncertain about the two classifications.

The horizontal decision boundary is almost perpendicular to the variable axes of X_3 and X_8. Although the biplot cannot help with determining which of these two variables are the most significant, either or both will have a big impact on the prediction probabilities. Within the red area at the top of the biplot, there are also parallel lines of deepening red. The variable axes of X_{10}, X_9, X_1, and X_5 are almost perpendicular to these lines, indicating that at low prediction probabilities, some of these four variables are significant.

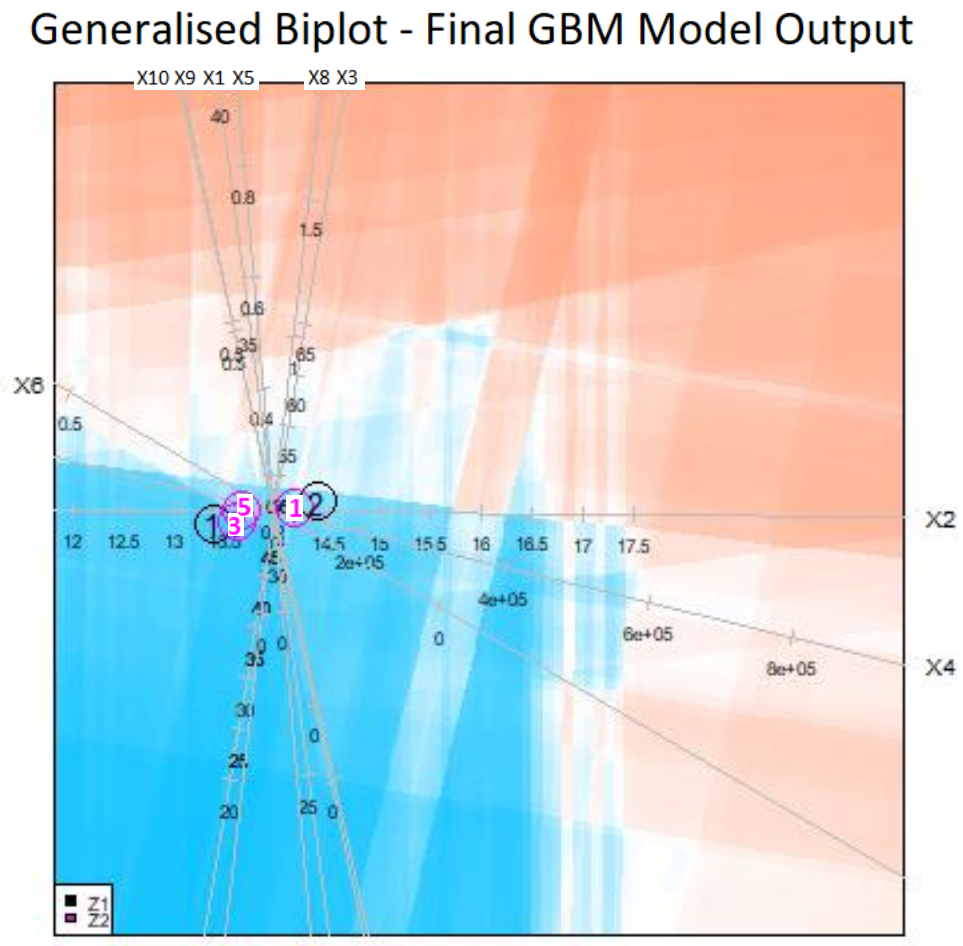


Figure 6-18: Generalised biplot of the final GBM

The vertical decision boundary is almost perpendicular to the variable axes of X_2 and X_4. Once these variables exceed a certain value, their prediction probabilities become very low, irrespective of the contribution of the other variables. This is the same interpretation as can be made from their partial dependence plots.

The impact of the two categorical variables can be investigated by plotting their category level boundaries, as shown in Figure 6-19. Moving across categorical level boundaries does not seem to have a significant impact on the prediction probabilities. Z_{25} does seem to be more closely correlated with the top left red area of the biplot, but the change to variable value Z_{25} does not change the colour grading.

The category level boundary between Z_{11} and Z_{12} is perpendicular to the horisontal decision boundary, therefore the impact of these two variables on the prediction probability is uncertain.

However, Z_{12} is closer to the vertical decision boundaries, which could be the reason why the PDP of variable Z_{12} shows the prediction probability decreasing if variable Z_1 is in category level 2.

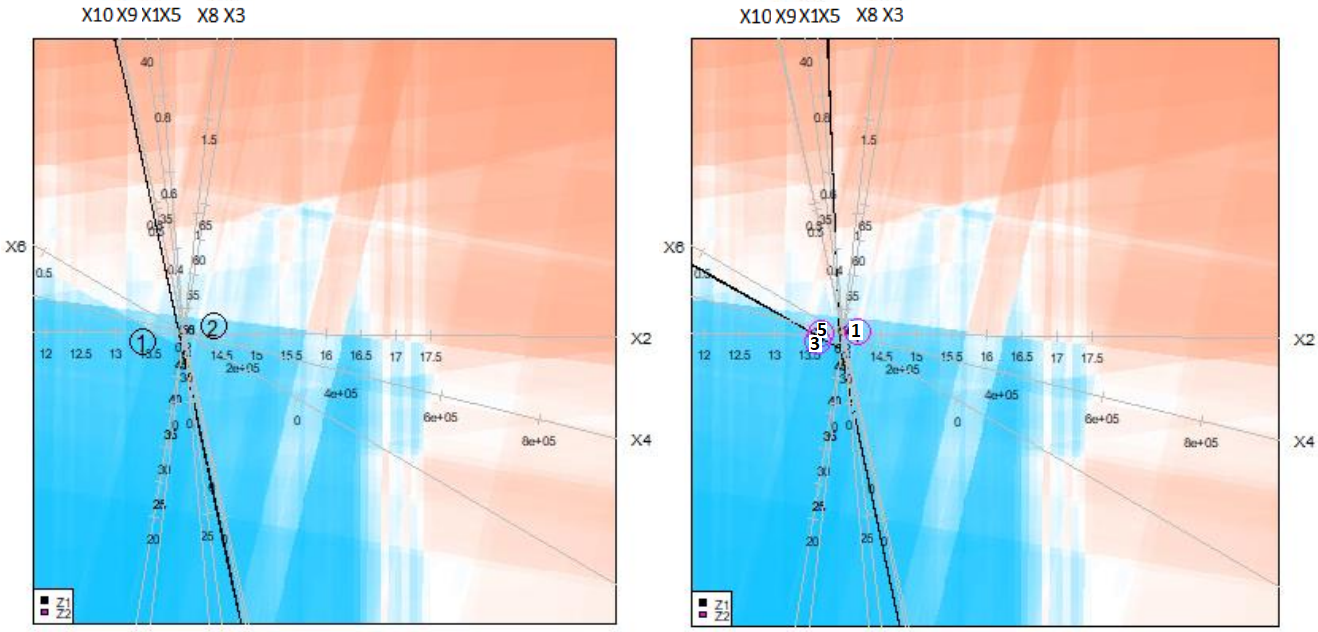


Figure 6-19: Generalised biplot of the category level boundaries of the final GBM

To conclude, the interpretations from the biplot of the impact of the variables is similar to those provided by the partial dependence plot.

Although the individual contribution of each of the variables cannot be determined from the biplot, the direction as well as the magnitude of their combined impact can be interpreted on a global level. The biplot shows at what combined value in the range of the variables the prediction will change sufficiently to move across the decision boundary, which is not indicated by the PDP output. This is due to the fact that the PDP only shows the impact of a single variable, and the output measure is the relative change in the prediction, instead of the absolute value. Where the variables are not closely correlated, the individual direction and impact on the prediction probabilities are shown.

As with the GLM, the biplot also indicates the correlation between the variables. This allows the biplot to give a globally faithful indication of the combined effect of variables on the prediction probabilities.

Lastly, the predictions of any model are only accurate in the multi-dimensional space in which it was calibrated. Therefore, the biplot should only be used to give global interpretations of the impact of the variables in the areas around the data points on which the biplot is based. This requires the data to be plotted along with the model output and variable axes on the biplot. However, the biplot data consist of only 500 sampled data points and they therefore do not capture the full range and distribution of the variables. This is the reason for the data points not being plotted as part of the analysis above. Ensuring that the sample points capture the full distribution, especially the tails of the distribution are areas where these biplots can be improved.

Explaining the accuracy of different models

This section on globally faithful interpretations of the machine learning model will conclude by using the biplot to compare the following two aspects of the model that determine its prediction accuracy:

- How the distribution of the future data on which the model will be applied compares to the training data
- How the distribution of the accepted and declined data points compare to the model's decision boundary

Distribution of the training vs out-of-sample test data

A common requirement for machine learning methods in order to give accurate predictions is that the training data and the future data on which the model is applied must follow the same distribution (Zadrozny, 2004).

To determine if this is the case, the biplots of the training and future application data can be compared. If their biplots have a similar structure, it could be assumed that the data follows the same distribution, which might suggest that the out-of-sample test prediction accuracy of the model will also hold for the future data. Since we do not have any application data with which to create a biplot, the biplots of the training and test data will be compared. The biplots are expected to be similar, since they were sampled from the same original data set.

However, both sets of data were further sub-sampled with a clustering model to get the 500 representative biplot data points. Therefore, this comparison serves to determine whether the clustering model has captured the same distribution in the data. The comparison, with the data points, is shown in Figure 6-20 below.

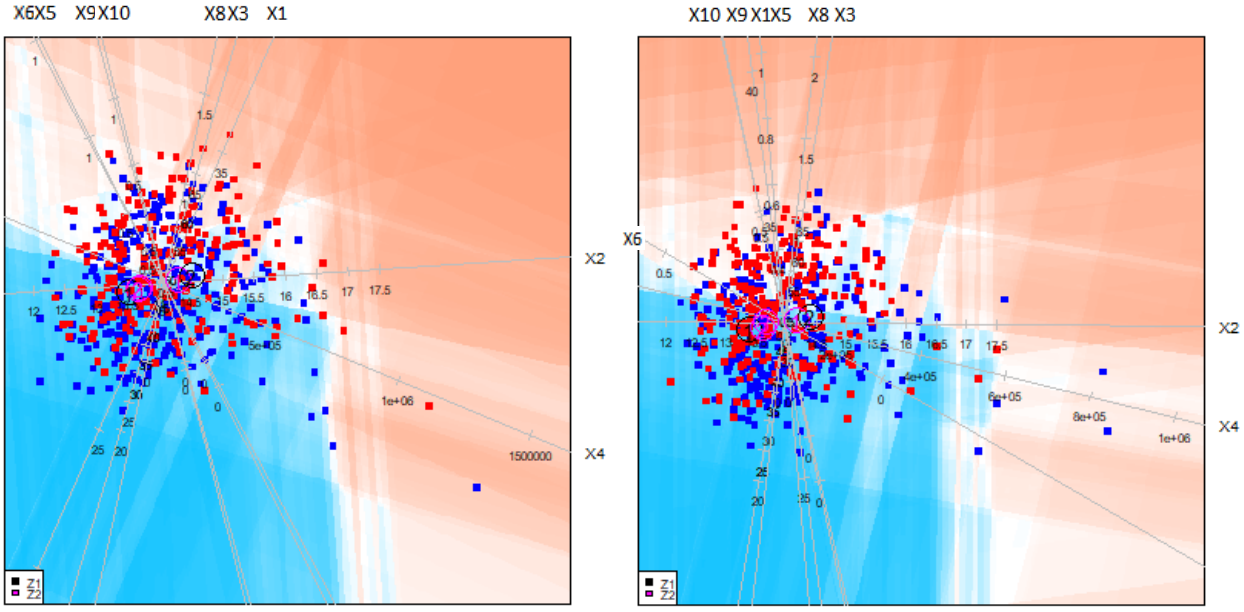


Figure 6-20: Comparison of the distribution of the training data set on the left and the test data set on the right

There are differences between the two data sets and the resulting biplots. However, the overall distribution of the data - especially the correlations between the most important variables - and the shape of the decision boundaries are the same.

Shape of the decision boundary compared to the distribution of the test data

The section will conclude by comparing the biplots of the GLM and the GBM to explain why the GBM accuracy is better than the GLM accuracy, by separately plotting the accepted policies and the declined policies over the prediction areas to illustrate how they interact with the decision boundary. This is shown in Figure 6-21 below.

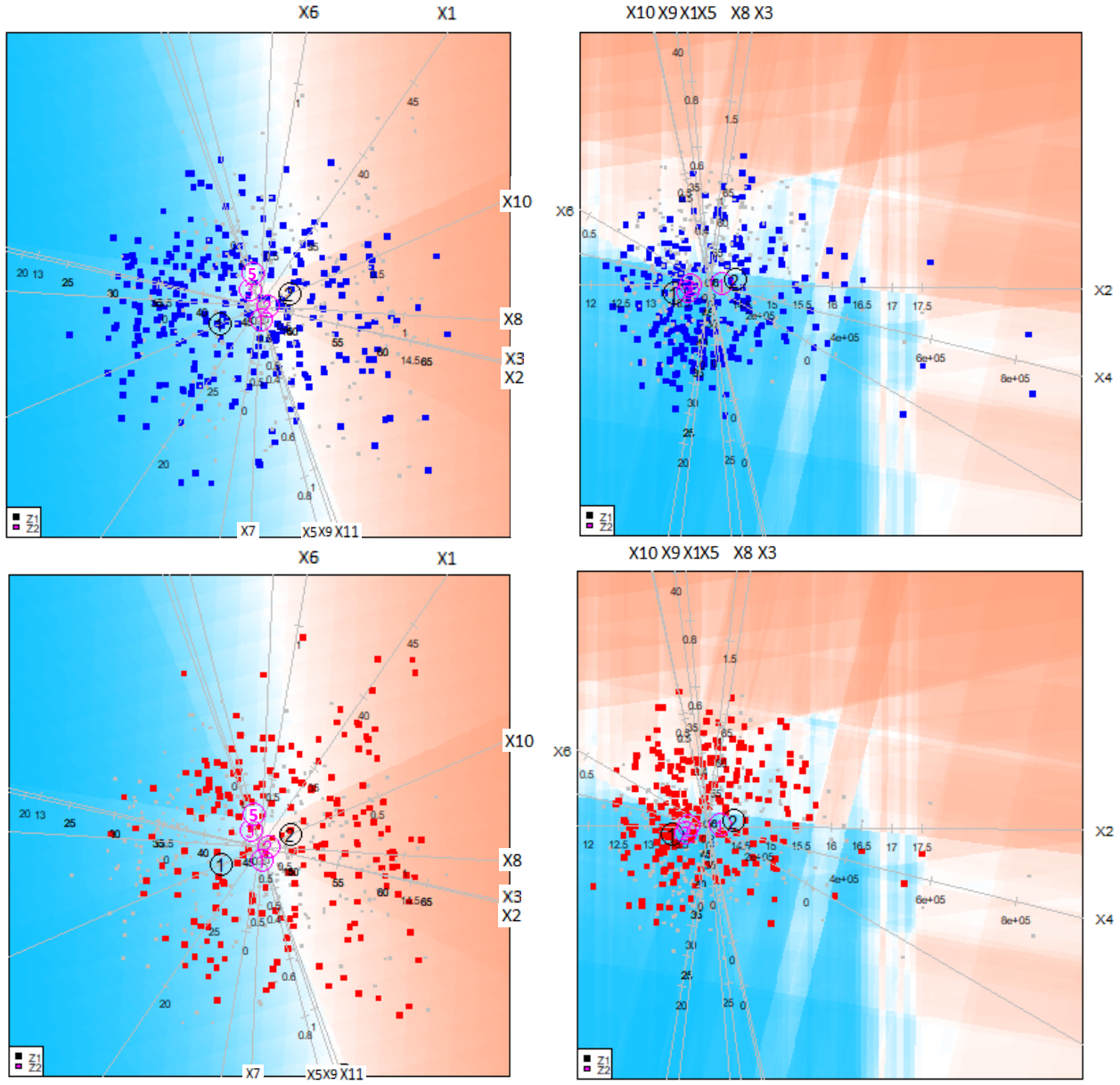


Figure 6-21: Impact of the shape of the distribution boundary on the accepted and declined predictions. The GLM is on the left and the GBM is on the right

As discussed earlier in this chapter, there is significant overlap of the accepted and declined policies. With the GLM, a significant number of accepted policies are in the declined area of the linear boundary. Many fewer declined policies are in the accepted area of the boundary, which will reduce the overall accuracy of the model. Because the decision boundary is linear, even if the decision boundary is increased to a prediction probability of 70% – that is, deeper into the blue area of the biplot – the accuracy will still be low. This will improve the positive prediction accuracy, as only a few declined policies will still be included, while many accepted policies will remain. This will however be at the expense of a significantly reduced data prevalence.

The above explains why the GLM gives such a low overall accuracy, but still manages to have a high positive prediction rate at the expense of a lower data prevalence.

The GBM has a very wide area of uncertainty represented by white output on the biplot, and most of the declined policies are in this area. This shape reduces the number of accepted policies into the declined area of the biplot, which causes the higher AUC value and increases the overall accuracy.

However, the GBM has a very strong linear element that separates the deep blue and high prediction probabilities from the rest of the biplot. If the decision boundary is increased beyond these prediction probabilities, the model will act very similarly to the GLM. This results in the positive prediction accuracy of 83.3% for the GBM against 82.7% for the GLM. However, the GBM has a higher data prevalence of 64.2% compared to 58.7%.

Therefore, the GLM's underlying assumption that the data is linearly separable does not completely hold. This reduces the accuracy of the linear model and suggests that a non-linear model will be more appropriate. However, some of the variables do have a very linear and independent impact, which means that the non-linear model will not significantly outperform the linear model.

6.6 Local interpretations using generalised biplots

This section shows how generalised biplots yield locally accurate post-hoc interpretations of why the model predicted a certain outcome for an individual data point. Biplots will be used to visualise the following:

- The impact of variables and their interactions on the prediction for an individual data point
- For a specific data point, the direction and magnitude of a change in a variable that are needed to change the classification of the point
- Which data points are outliers in terms of the overall data structure

This section will only compare the local interpretations given by the biplot against those provided by the coefficients of the final GLM, and not against the GBM. This is because the GBM does have a method for automatically providing local interpretations, where as GLM do.

Local interpretations of individual predictions

The first step towards interpreting the predictions for an individual data point, is to plot the point on the biplot, with the colour of the output of the machine learning model added to the biplot plane. Four examples points will be plotted and compared to the coefficients and prediction outcomes of the GLM using a 50% decision boundary. These four were selected to illustrate and compare the interpretations of the four types of machine learning predictions, namely true positive, true negatve, false positive and false negative predictions.

The 50% boundary was used to ensure that the sample points are all close to the middle of the biplot while still having all four types of machine learing prediction results.

The biplot and table of predictions are shown in Figure 6-22 and Table 6-5.

Based on their location on the biplot, point 1 and point 2 are correctly indicated as accepted and declined policies. Point 1 is the blue square furthest to the left on the biplot, and is far from the decision boundary in the blue area of the biplot, which corresponds to the GLM's 68% prediction probability. Point 2 is the red square furthest to the right on the biplot, and is far from the decision boundary in the red area of the biplot, which corresponds to the GLM's 10% prediction probability.

These two points are on the opposite ends of the axis of variable X_1, which corresponds to the difference in the variable's values for these two points, and explains why the model yields different predictions for these two points.

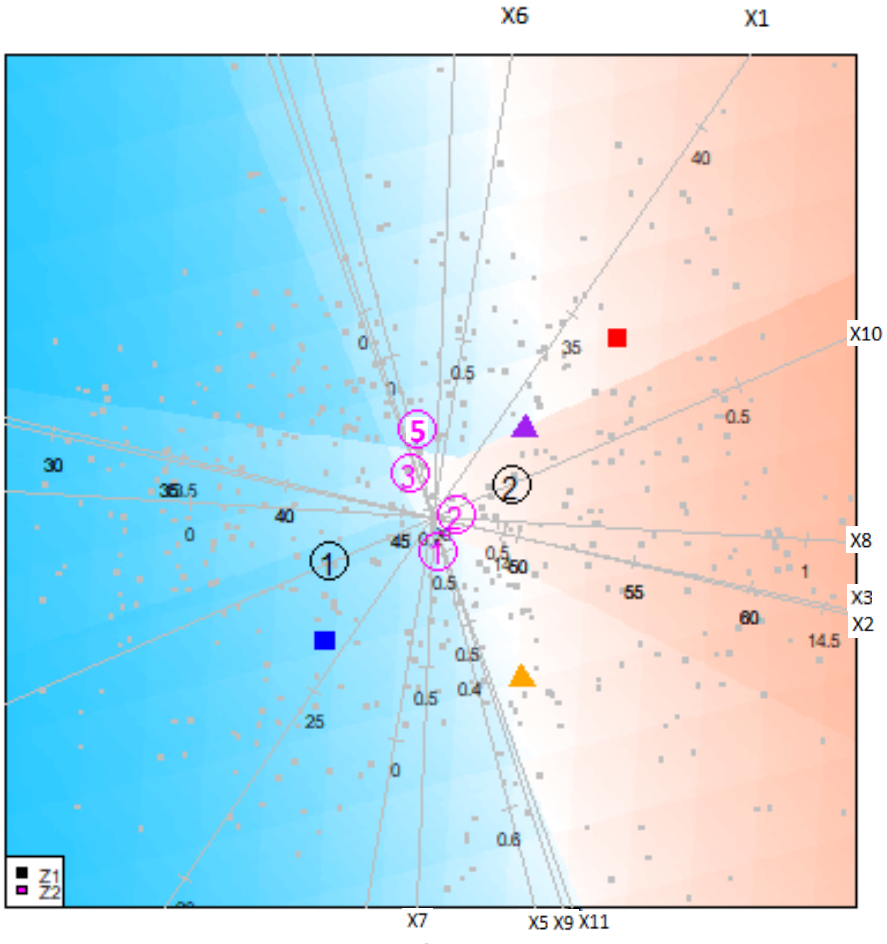


Figure 6-22: Using a biplot to provide local interpretation for the predictions made for four data points using the final GLM

Table 6-5: Actual and the prediction probability and outcome based on the GLM for four data points

	Point 1	Point 2	Point 3	Point 4
Actual Value	1	0	0	1
Predicted Value	68%	10%	64%	48%
Prediction Probability	1	0	1	0
Classification Outcome	True Positive	True Negative	False Positive	False Negative
Colour indication	Blue	Red	Orange	Purple

Points 3 and 4 are both incorrectly classified.

Point 3 is a declined policy classified as being accepted by the model. The point is represented on the biplot by an *orange triangle*. It is in the red area and is in fact a declined policy. Based on its position on the biplot, the policy is expected to have been correctly classified as declined. Therefore, the data point's values fit the pattern of a declined policy, as indicated by its location on the biplot.

This misclassfication could indicate a possible mistake in the calibration of the GLM and should be a focus area for further improving the model. This interpretation does however depend on the quality of the biplot display.

Point 4 is an accepted policy classified as being declined by the model. It is represented on the biplot by a *purple triangle*. It is in the red area and is predicted to be a declined policy by the GLM. Therefore, the prediction and location on the biplot correspond. Point 4 has high values for variables X_1, X_2, and X_3, which cause the predicted probability to be low – hence, the GLM output predicting the policy to be declined. However, the policy was accepted at standard rates.

This is unexpected since, based on the data point's values and location on the biplot, the decline prediction seems to be correct. This issue could indicate a mistake in the original classification of the policy as an accepted policy and indicates a policy that should be investigated to determine if it was correctly classified by the underwriter.

Factors that will cause a change to the model's prediction

As explained above, the biplot visualises the impact of each variable on the prediction probability of the machine learning model. The variable values of a data point, and therefore its location on the biplot and the surrounding colour gradient of the model output, also indicate the direction and impact that a change in a variable value will have on the prediction probabilities and classification of that point.

But a more specific question is to determine which variable values require the smallest change for the policy's classification to change.

This determination could be used to explain to a person for whom the prediction applies, why the black box machine learning model has predicted a certain outcome, as well as what factors the person would need to change in order to enable the prediction to change.

This determination is easily achieved by considering the coefficients of the GLM, since they indicate the independent *direction and magnitude* of the impact of each variable on the predicted outcome. This indicates what increase or decrease in a variable value is required in order to change the predicted outcome. However, this method is not possible in the case of non-linear models where there are no coefficients available.

VIPs only indicate the importance of a variable, and do not determine the impact that a change of one unit of a variable will have on the prediction probabilities; that is, whether an increase in the variable will increase or decrease the prediction probability and by how much. PDPs indicate the direction of the change due to a change in the variable, but they do not allow for the interaction between variables and do not indicate the magnitude of the impact of the change in the variable.

The biplot visualises the direction and magnitude of a change in a variable value that are needed in order to change the predicted outcome of the model. The motivation for this is that the biplot shows

the direction along the axes that the variable needs to change, in order to move towards the blue or red prediction areas. The axes are also marked with the unit scale of each variable, which indicates the magnitude of the change needed to move from one prediction area to another – that is, how many units a variable needs to move along the axis toward the decision boundary.

The relative position of the data point on the biplot allows for the interaction of terms, since different points will need to move into different directions in order to find the shortest path to a blue area. Moving along two axes at once may also shorten the total distance, which therefore indicates that the interaction of two terms require less change for each variable.

Therefore, the shortest Euclidean distance along a variable axis towards the classification boundary indicates the smallest change required to change the classification of a point.

Based on the biplot of the GLM, the shortest distance for data point 4 to the blue area is down along variable axes X_2 and X_3. An alternative is to move down variable axis X_1. The impact of each of these options is shown in Figure 6-23. The blue circle is the change from reducing variable X_1 by two units. The dark blue square is the change from reducing variables X_2 and X_3 by 0.15 and 4 units, respectively.

These two possible changes increase the prediction probabilities and changes the classification from declined to accepted. Movements along other variable axes will either further decrease the prediction probability or require a much longer distance on the biplot, and therefore bigger variable change, to change the classification.

Of course, certain variables cannot change for an individual person. The person cannot reduce his age. But he can apply reapply for the same policy a year later. Or he can do something to improve his health before reapplying. The biplot allows only the variable that can be changed to be identified and the change needed interpreted.

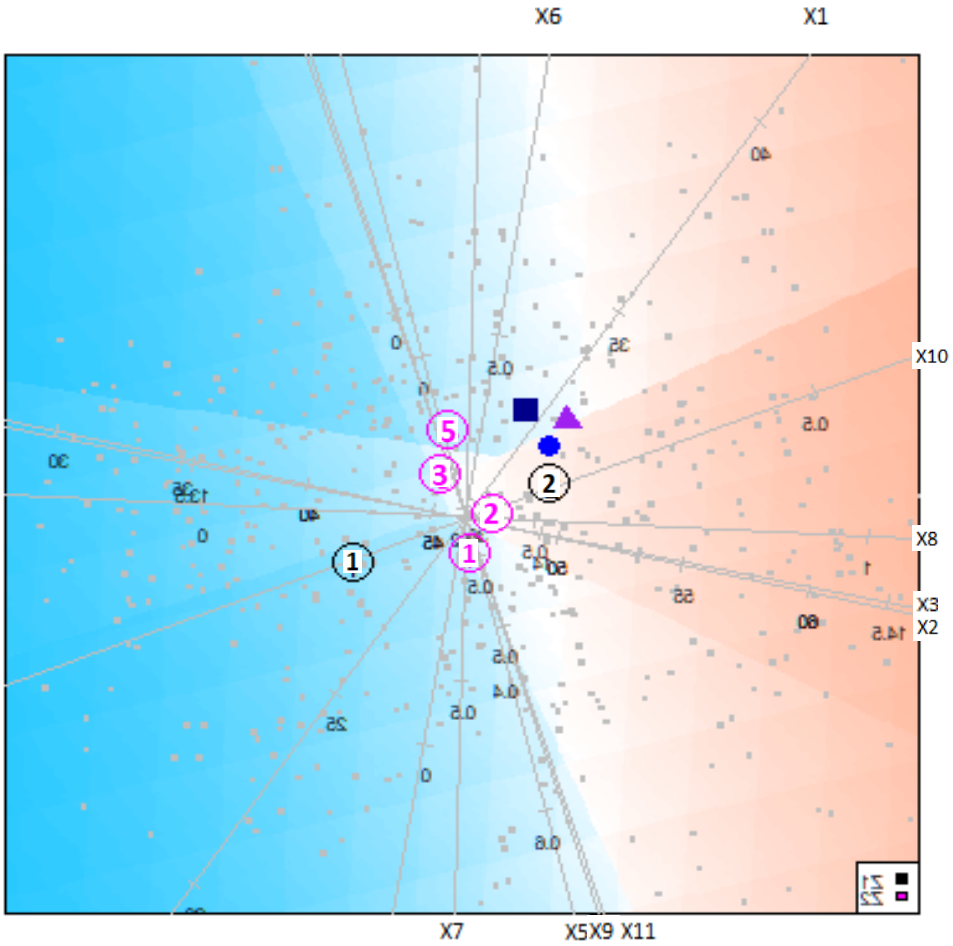


Figure 6-23: Changing the position on the biplot by reducing the value of variable X_1 (blue circle) and variables X_2 and X_3 (dark blue square). The points move into the blue area and the predicted outcome changes from declined to accepted.

The problems with using the biplot to determine the smallest change required are the following:

- The masking of the colour of the output by other variables distorts the perceived change required in a variable to change the classification. For variable X_1, a much bigger move in position of the point on the biplot was expected, based on the colour of the area around the point. This is because the impact of variable X_1 on the output colour was masked by other variables.

- Although both changes – along variable axes X_2 and X_3 and along variable axis X_1 – have resulted in the required change in classification, the biplot does not indicate which of these changes was the shorter route. The measure of distance would have needed to quantify the change in the variable that was needed to change the classification.

Using the biplots to identify outliers in the data

The final use of the biplot is to identify individual outliers in the data that could indicate potential fraud or incorrect data points that should be excluded from the calibration of the model. Two types of outliers are indicated in Figure 6-24.

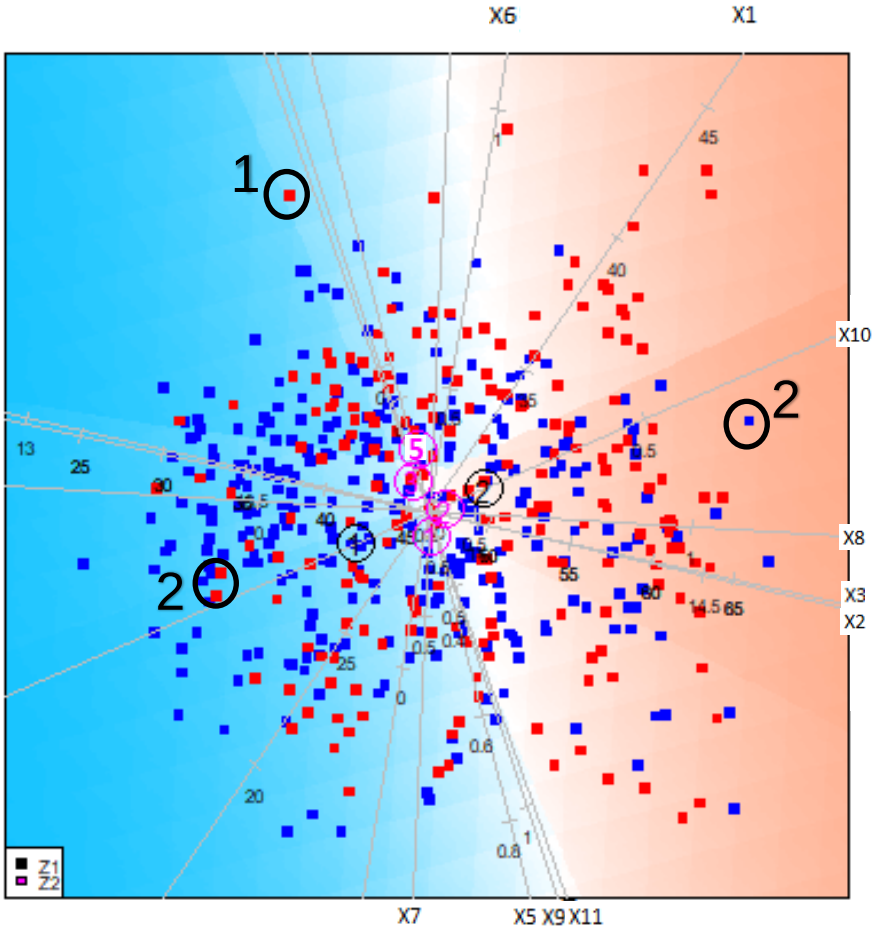


Figure 6-24: Using the biplot to identify outliers in the data

- Data points that are very far away from any other data points. An example of such an outlier is shown at the top of the biplot, marked with a “1”.

  The distance from one point to any other point indicates how unique the variable values are for this data point. A data point with no nearby neighbours is therefore unique and unexpected.

  Although outliers are always expected in a large data set, they could indicate data points that contain mistaken variable values that cause the data point to look very different from all the other points. This mistake can be either accidental or fraudulent, but in either case,

visually finding these policies on the biplot simplifies the process of knowing which policies an internal data control team should start investigating first.

- Data points which are classified completely differently from what the model has predicted. An example of such an outlier is shown at the left and right of the biplot, marked with a "2".

  These are the polices that have very high prediction probabilities but are declined policies, and/or that have very low prediction probabilities but are accepted policies. They can be found using the biplot by looking at declined policies deep in the blue area or accepted policies deep in the red area of the biplot.

  If this occurs very often, it could indicate a quality control problem in the company's processes that allow these policies to have an outcome that is counter-intuitive to all other similar policies. This will need to be investigated by the company to confirm whether these policies are indeed classified correctly. Removing the incorrect policies from the training data could result in more accurately calibrated machine learning models.

Chapter 7 Conclusion, Limitations and Future Developments

7.1 Summary of findings

The goal of this dissertation has been to develop a new technique with which to generate post-hoc interpretations that are globally or locally faithful for any machine learning method. This dissertation focussed on the use of generalised biplots to visualise both continuous and categorical variables, which in turn allows the method to be agnostic of the machine learning method.

Biplots offer an optimal two-dimensional representation of the data set on which the machine learning model is based. A biplot also allows every point on the biplot plane to be converted back to the original p dimensions, in the same format as was used by the machine learning model. Using this as the input data, the model then gives the prediction probability between 0 and 100% for every point. The prediction probabilities are subsequently used to colour every point on the biplot plane.

The interaction of the changing prediction probabilities – represented by the coloured output – in relation to the data points and the variable axes and category level points represented on the biplot, allows the machine learning model to be globally and locally interpreted.

Global interpretations of black box machine learning models

The generalised biplot allows for a wide range of global interpretations to be made about the data set and the machine learning model. They include interpretations of:

- The distribution of the data points and determining whether the assumptions underlying the machine learning method concerning the required data distribution are applicable

  The biplot allows the shape of the decision boundary created by the machine learning model to be visualised based on the pattern in the colour of the output. If the distribution of the data does not allow the decision boundary to separate the different classes of output, the model is unlikely to be accurate.

  The visualisation of the distribution of the data can help with the selection of the most appropriate modelling method, for a given data set. As seen in the comparision of the accuracy of the GLM and GBM, the choice between an linear and a non-linear model has a much bigger impact on the prediction accuracy than optimising the model's hyper-parameters or selecting the most appropriate set of input variables.

- The correlations and relationships between the different variables and between the data points and the variables

  Highly correlated variables, or category levels that only cover a small area of the biplot, could indicate variables or levels that will not be significant in accurately predicting the target variable, and should therefore be removed.

  It could also indicate if there are any co-linearity between variables, which could contribute to the selection of machine learning methods that can better accommodate co-linearity between variables.

- The relative importance of and the impact that a variable has on the prediction probabilities of the model

  If a variable axis runs parallel to the decision boundary of the model, or if a change in the category level does not result in a change in the colour of the model output, then the variable is less likely to be important for creating the prediction probabilities of the model. A variable axis that runs perpendicular to the decision boundary, or a change in the category level that results in a big change in the output colour, is more likely to be important.

  This analysis could be used to determine if specific variables should be removed to either simplify the model or improve the out-of-sample accuracy of the model.

- The direction and magnitude of a change in a variable that is required in order to change the prediction probabilities, allowing for the non-linear interactions between the variables

  The relationship between the variable and the coloured output of the model indicates the direction in which a variable must change in order to increase or decrease the prediction probabilities of the model. The required direction of change can also vary depending on where on the biplot the analysis starts, in relation to the location of the decision boundary.

  The gradient of the change in the output colour along the variable axes or over a category decision boundary, along with the variable scale markers on the axis, give an indication of the magnitude of change that is required to move from one classification area to another.

  This allows for the investigation of the combined impact of changing two or more correlated variables.

- The distribution of the test data over the coloured output

This distribution also gives an indication of how the positive or negative prediction accuracy will change as the decision boundary level is increased or decreased. This can help explain why and for what type of applications one model will outperform another model.

- Comparing the distribution of the training data and the data on which the model is applied

  The machine learning model will only be as accurate in the real-world application as predicted in the analysis above if the data on which it is applied has a similar distribution to the data on which the model was built.

  Therefore, to ensure that implementation of the model will be successful, the biplots of the training and future application data can be compared. If they have the same shape, it could indicate that the underlying data distribution might be the same and the machine learning model can be transferred to the real world with confidence.

Local interpretations of black box machine learning models

The generalised biplot also allows the following local interpretations to be made:

- Which variables have had the biggest impact on the classification of an individual data point

  This can be seen by looking at the position of the data point on the biplot relative to the variable axes and CLPs, and by considering the gradient of the coloured output that surrounds the data point, which will show which axes or CLP has had the biggest impact on moving the data point deeper into a certain classification area.

  This can also be used to understand why the machine learning model has mis-classified a data point. If many of the variables of a data points has values that indicate that the data point should have a certain predicted classification, but its actual classification is different, it could indicate that either the machine learning model is not calibrated accurately enough, or the data point is incorrect and should be excluded from the training data.

- Which variable requires the smallest change in order to change the classification of the data point

  This can be determined by considering the direction and magnitude of the change caused by each variable and the relative position of the decision boundary. Therefore, the shortest distance from the data point to the decision boundary along a variable axis indicates which variable requires the smallest amount of change in order to change the classification.

- Which data points are outliers and should be investigated to determine the accuracy of the input variables or the validity of the target variables

  A data point could be an outlier because it is far removed from all of the other data points, which may occur if the variable values do not follow the pattern seen in most of the other data points. It could indicate possible errors in the input data that need to be corrected.

  A data point is also considered an outlier if it is plotted deep in the incorrect classification area. This data point should be investigated to understand why it was given a counterintuitive classification and can indicate an error in the training data.

Conclusion

The insights and interpretations in this dissertation can assist with justifying the use of black box machine learning models compared to generalised linear models. The dissertation has explained how the biplot can help to increase trust in the model by:

- Providing local explanations as to why a specific point has been classified in a certain way, and what can be done to change the classification.

- Providing algorithmic transparency by showing the impact of the model on the decision boundary, and therefore illustrating why different models give different classifications of the same data points.

- Visualising how the model interacts with the different variables, which improves the understanding of why the model makes certain predictions. These visualisations are more intuitive and easily understood than p-values, variable importance plots, partial dependence plots, and other methods currently being used to visualise the output of machine learning models.

The improved trust, along with the increased prediction accuracy of non-linear models, will hopefully remove the stigma of calling non-linear models “black box” models, and encourage the wider application of these more advanced models.

7.2 Limitations to using generalised biplots

Using generalised biplots achieves the objective of being able to provide global and local interpretations of any machine learning model. However, there are still many limitations to its use. Some of these are listed below:

- Computational requirement to calculating the distance matrix

  Calculating the distance matrix underlying the generalised biplot is computationally very demanding. For this dissertation, this problem was simplified by using only 500 representative sample points, but this reduces the accuracy of the biplot and the interpretations made from it.

  Once the distance matrix is calculated, it can be saved and reused, which reduces the subseqencent time required to draw the biplot. However, every time that a data point changes, the distance matrix calculation must be repeated.

- Interpolating new data points onto the biplot plane

  PCA, CVA, and AoD biplots have the benefit of easily calculating the positions on the biplot plane of new individual data points, a function called *interpolating*. However, when using a distance matrix, the position of any new point needs to be set such that it optimally presents the distance to every other existing point. The current code cannot perform this calculation, and the calculation of the CLPs in the reduced two-dimensional space further complicates the mathematics of finding the optimal representation of a new data point.

  However, a suitable method is required in order to accurately visualise the impact of changing a variable's value of an individual data point in order to simulate the change that is needed to cross the decision boundary.

  The example shown in Section 6.6 was produced by including the updated variable values as new data points and creating the biplot for the extended data set. This can result in a different biplot, as occurred in the example where the biplot was transposed, which complicate the comparison and analysis of the change.

- Finding the shortest distance to the decision boundary

  Another local interpretation uses the biplot to find the shortest distance to a decision boundary for any data point, and therefore explain the easiest way to change the classification of the data point. The visual interpretations of the biplot does give a possible answer, but because the biplot is only an approximation of the distance to the decision

boundary, and because of the masking of the impact of variables by other variables, the answer that is given by the biplot is not optimal. In addition, it is difficult to implement the visual investigation to a large set of points.

- Categorical level values are only represented in the area closest to the CLP

  The categorical variables, represented by the CLPs on the biplot, predict the same category level for all of the the points on the biplot plane within the category level boundaries. This means that many areas of the biplot have no representation on that category level, and the interaction of that category level with the continuous variables and the other categories outside the category level boundary is ignored and not shown by the biplot. This in turn could lead to incorrect interpretations regarding the global and local impact of the variables.

  Incorrect interpretations of the direction and magnitude of the impact of continuous variables can also occur if the impact of a variable is masked by a correlated variable that has an opposite but bigger impact on the prediction probabilities. Therefore, correlated variables should always be investigated together, as the biplot can distort the the interpretation of their individual impact.

- The predictions of any model are only accurate in the multi-dimensional space in which it was calibrated

  The biplot should only be used to determine global interpretation of the impact of the variables in the areas around the data points on which the biplot is based.

  This requires the data to be plotted along with the model output and variable axes on the biplot. However, only 500 sampled data points are used to create the biplot, and they therefore do not capture the full range and distribution of the variables that was used to calibrate the machine learning model. Ensuring that the sample points capture the full distribution, especially the tails of the distribution, is an area where these biplots can be improved.

- PCA, CVA, and AoD biplots

  This dissertation has also explained the use of PCA, CVA, and AoD biplots. The dissertation has focussed on generalised biplots because it can visualise and interpret the impact of categorical variables on the machine learning model. Generalised biplots also allow different distance measures that are not Euclidean embeddable to be used as the basis of the multi-dimensional scaling of the data points.

If the input data only consists of continuous data, and a Euclidean embeddable distance measure is used, then generalised biplots are not needed and PCA, CVA, and AoD biplots can be used instead. As explained in this document, the p-dimensional value of any point on the biplot can be predicted and used in a similar way as with the generalised biplot.

The benefit of these methods is that they do not require the calculation of a distance measure, which makes them computationally more scalable to larger data sets. They also allow for more accurate interpolation of a new data point onto the biplot plane, which makes it easier and more accurate to interpret the local impact of variables on the prediction.

7.3 Further research in using biplots to interpret machine learning models

The final section in this chapter explains possible areas for future research. The topics mentioned cover some of the limitations mentioned above, but also include ways in which the current code can be updated to improve the functionality and extend the type of investigations that can be performed using the generalised biplots.

- Adding the decision boundary line to the biplot

  The investigations in this dissertation have used either a 70% or a 50% decision boundary to determine the class into which to classify a data point. These decision boundaries were implied by the colour gradient of the output of the machine learning model.

  However, an actual line representing the exact boundary was not shown, because very few points on the biplot plane actually represent 70% or 50% exactly and highlighting these points in black will result in a discontinuous set of points on the biplot.

  Therefore, an area of improvement would be to create a smoothed line to connect the various points and represent it as an approximation of the required decision boundary line.

- Applying non-linear variable axis trajectories

  Non-linear variable axes were discussed in Chapter 4, but were not applied in generalised biplots, because the distance measure between the continuous variables is Euclidean, which always results in linear variable axes.

  Combining non-linear decision models with non-linear distance measures for the continuous variables would be an interesting area for future investigation.

- Fixing the category levels across the biplot plane

  One limitation of using CLPs is that they only represent a category level within its category boundary. One way to see the impact of a category level across all of the variable axes and all of the other possible category levels, is to have the ability to fix the category levels across the entire biplot.

  Fixing and analysing all of the possible category level interactions will require a trellis of graphs. Although this will increase the number of biplots and complicate the analysis, it will reduce the extent to which category levels are being masked, and allow all of the non-linear interactions between a category level and other variables to be investigated.

- Other maching learning methods

  This dissertation has only focussed on comparing the output of a GLM and a GBM. Another area of research would be to compare the output of a range of machine learning methods to understand how their underlying assumptions relate to the training data set, and investigate in which situations one method outperforms the other.

- Regression or multiclassification models

  This dissertation only focussed on a binary classification example, which allowed the clear distinction of blue and red prediction areas.

  However, many machine learning problems require multi-classification prediction models that means the change in colour gradient will not give an intuitively accurate visualisation of how the predictions changes.

- Other distance measures

  This dissertation has compared the biplots created using four different types of distance measures. However, it has not always been an exhaustive analysis, as the distance measures were not selected to highlight the impact of specific variables. The generalised biplot could be improved by using different columns of the MDS matrix $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}$, where only the first two columns are currently being used. Although the first two columns represent most of the variance in the data, other columns could represent more of the variance of specific variables. By selecting the appropriate columns, the biplot would better represent the impact of individual variables.

  Another consideration would be to weigh the contribution of each variable by the output of the variable importance plot, or some other measure of the relative importance of the different variables. This would ensure that the biplot better represents the impact of the most important variables.

  In the case of PCA biplots, different linear combinations of the original variables can be used than those implied by the eigenvectors. For example, the same weights as used in a linear prediction model.

  Finally, the benefits of improving the interpretability of the biplot using non-metric MDS methods could be investigated.

- Using alpha bags to improve global interpretability

  In Section 3.3, alpha bags were mentioned as a way to improve visualisation of the data, but they where not implemented in the generalised biplots. Incorporating them into the generalised biplot code would improve the global interpretability of the biplot.

- Quantifying the exact distance between a point and a decision boundary

  The last possible area of future research mentioned here is to quantify the exact distance of any point on the biplot to the nearest decision boundary. This distance can either be along a single variable axis, or can allow for the combined impact of multiple axes or categories.

  This research is needed to give a more definitive answer to the question highlighted in the literature study: “that the consumer has the right to reasons for credit being refused.” This question can currently only be subjectively answered by the generalised biplot.

--- The End ---

Appendices

Appendix A: Orthogonality of predictions

For the PCA method:

As defined in Equation 3, $\widehat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}'_r$ then $\widehat{\boldsymbol{X}}' = \boldsymbol{V}_r\boldsymbol{V}'_r\boldsymbol{X}'$, with $\boldsymbol{V}_r = \boldsymbol{V}\boldsymbol{J}$

As defined in Equation 2, $\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}'$

And $\begin{bmatrix} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} = \boldsymbol{J}$

Type A orthogonality

$$\boldsymbol{X}\boldsymbol{X}' = (\widehat{\boldsymbol{X}} + \boldsymbol{X} - \widehat{\boldsymbol{X}})(\widehat{\boldsymbol{X}} + \boldsymbol{X} - \widehat{\boldsymbol{X}})' = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}' + \left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right)\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right)' + \widehat{\boldsymbol{X}}(\boldsymbol{X} - \widehat{\boldsymbol{X}})' + (\boldsymbol{X} - \widehat{\boldsymbol{X}})\widehat{\boldsymbol{X}}'$$

$$\widehat{\boldsymbol{X}}\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right)' = \widehat{\boldsymbol{X}}\boldsymbol{X}' - \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}' = \boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{X}' - \boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{X}'$$

Since $\boldsymbol{V}_r'\boldsymbol{V}_r = \boldsymbol{I}$, $\widehat{\boldsymbol{X}}\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right)' = \boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{X}' - \boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{X}' = \boldsymbol{0}$

Therefore, Type A orthogonality holds, with $\boldsymbol{X}\boldsymbol{X}' = \widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}' + (\boldsymbol{X} - \widehat{\boldsymbol{X}})(\boldsymbol{X} - \widehat{\boldsymbol{X}})'$

Type B orthogonality

$$\boldsymbol{X}'\boldsymbol{X} = (\widehat{\boldsymbol{X}} + \boldsymbol{X} - \widehat{\boldsymbol{X}})'(\widehat{\boldsymbol{X}} + \boldsymbol{X} - \widehat{\boldsymbol{X}}) = \widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}} + \left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right)'\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right) + \widehat{\boldsymbol{X}}'(\boldsymbol{X} - \widehat{\boldsymbol{X}}) + (\boldsymbol{X} - \widehat{\boldsymbol{X}})'\widehat{\boldsymbol{X}}$$

$$\widehat{\boldsymbol{X}}'\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right) = \widehat{\boldsymbol{X}}'\boldsymbol{X} - \widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}} = \boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{X}'\boldsymbol{X} - \boldsymbol{V}_r\boldsymbol{V}_r'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{V}_r\boldsymbol{V}_r'$$

$$= \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'\boldsymbol{X}'\boldsymbol{X} - \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'$$

Since $\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}'$, $\widehat{\boldsymbol{X}}'\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right) = \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'\boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}' - \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'\boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}'\boldsymbol{V}\boldsymbol{J}\boldsymbol{V}'$

$$= \boldsymbol{V}\boldsymbol{J}\boldsymbol{D}^2\boldsymbol{V}' - \boldsymbol{V}\boldsymbol{J}\boldsymbol{D}^2\boldsymbol{J}\boldsymbol{V}'$$

With $\boldsymbol{J}\boldsymbol{D}^2 = \boldsymbol{J}\boldsymbol{D}^2\boldsymbol{J}$, $\widehat{\boldsymbol{X}}'\left(\boldsymbol{X} - \widehat{\boldsymbol{X}}\right) = \boldsymbol{0}$

Therefore, Type B orthogonality holds, with $\boldsymbol{X}\boldsymbol{X}' = \widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}} + (\boldsymbol{X} - \widehat{\boldsymbol{X}})'(\boldsymbol{X} - \widehat{\boldsymbol{X}})$

(Gardner-Lubbe, le Roux and Gower, 2008)

For the MDS method:

As defined in Equation 5**,** $\hat{\boldsymbol{X}} = \boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}$, $\hat{\boldsymbol{\Gamma}} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}$ and $\hat{\boldsymbol{X}} = \boldsymbol{Z}\hat{\boldsymbol{\Gamma}}$

For the Euclidean MDS biplot, the regression matrix $\hat{\boldsymbol{\Gamma}}$ is orthogonal, with $\hat{\boldsymbol{\Gamma}}\,\hat{\boldsymbol{\Gamma}}' = \boldsymbol{I}$

Type A orthogonality

$$(\boldsymbol{X} - \hat{\boldsymbol{X}})\hat{\boldsymbol{X}}' = \boldsymbol{X}\hat{\boldsymbol{X}}' - \hat{\boldsymbol{X}}\hat{\boldsymbol{X}}' = \boldsymbol{X}\boldsymbol{X}'\boldsymbol{Z}((\boldsymbol{Z}'\boldsymbol{Z})^{-1})'\boldsymbol{Z}' - \boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}\boldsymbol{X}'\boldsymbol{Z}((\boldsymbol{Z}'\boldsymbol{Z})^{-1})'\boldsymbol{Z}'$$

$$= \boldsymbol{X}\hat{\boldsymbol{\Gamma}}'\boldsymbol{Z}' - \boldsymbol{Z}\hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Gamma}}'\boldsymbol{Z}'$$

If this equation does not equal zero, Type A orthogonality does not hold.

With Euclidean MDS, $\boldsymbol{Z} = \boldsymbol{X}\hat{\boldsymbol{\Gamma}}'$**,** with $\hat{\boldsymbol{\Gamma}} = \boldsymbol{V}_r'$, and $\hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Gamma}}' = \boldsymbol{I}$

Therefore $\hat{\boldsymbol{X}}(\boldsymbol{X} - \hat{\boldsymbol{X}})' = \boldsymbol{X}\,\hat{\boldsymbol{\Gamma}}'\hat{\boldsymbol{\Gamma}}\boldsymbol{X}' - \boldsymbol{X}\hat{\boldsymbol{\Gamma}}'\hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Gamma}}'\hat{\boldsymbol{\Gamma}}\boldsymbol{X}' = \boldsymbol{0}$

Therefore, Type A orthogonality does hold for Euclidean distance MDS.

Other distance types imply non-linear projections of the data onto the biplot with $\hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Gamma}}' \neq \boldsymbol{I}$**.** This causes Type A orthogonality to not apply in those cases.

Type B orthogonality

$$\hat{\boldsymbol{X}}'(\boldsymbol{X} - \hat{\boldsymbol{X}}) = \hat{\boldsymbol{X}}'\boldsymbol{X} - \hat{\boldsymbol{X}}'\hat{\boldsymbol{X}} = \boldsymbol{X}'\boldsymbol{Z}((\boldsymbol{Z}'\boldsymbol{Z})^{-1})'\boldsymbol{Z}'\boldsymbol{X} - \boldsymbol{X}'\boldsymbol{Z}((\boldsymbol{Z}'\boldsymbol{Z})^{-1})'\boldsymbol{Z}'\,\boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}$$

Since $\boldsymbol{Z}'\boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1} = \boldsymbol{I}$

$$\hat{\boldsymbol{X}}'(\boldsymbol{X} - \hat{\boldsymbol{X}}) = \boldsymbol{X}'\boldsymbol{Z}((\boldsymbol{Z}'\boldsymbol{Z})^{-1})'\boldsymbol{Z}'\boldsymbol{X} - \boldsymbol{X}'\boldsymbol{Z}((\boldsymbol{Z}'\boldsymbol{Z})^{-1})'\boldsymbol{Z}'\boldsymbol{X} = \boldsymbol{0}$$

This equals zero so that Type B orthogonality holds for all MDS biplots.

Appendix B: Mathematical Definitions and Equations

For ease of reference, the most important mathematical definitions and equations used in the dissertation are summarised in this appendix.

Equation 1:

The SVD of any matrix $\boldsymbol{X}: n \times p$ can be expressed as

$$\boldsymbol{X} = \boldsymbol{UDV'}\text{ , with}$$

$\boldsymbol{U}: n \times p$ an orthogonal matrix with $\boldsymbol{U'U} = \boldsymbol{I}_p$

$\boldsymbol{V}: p \times p$ and orthonormal matrix with $\boldsymbol{V'V} = \boldsymbol{I}_p$

$\boldsymbol{D}: p \times p$ a diagonal matrix.

Equation 2:

The correlation between the columns of $\boldsymbol{X}$ is given by $\boldsymbol{X'X}$ (where $\boldsymbol{X}$ is a centred matrix). Therefore,

$$\boldsymbol{X'X} = \boldsymbol{VDU'UDV'} = \boldsymbol{VD^2V'}$$

Equation 3:

$$\widehat{\boldsymbol{X}}_{[r]} = \boldsymbol{UDJV'} = \boldsymbol{UJDV'} = \boldsymbol{UJDJV'} \text{ where } \boldsymbol{J}: p \times p = \begin{bmatrix} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} and\ r \leq p.$$

Equation 4:

Classical scaling solves $\boldsymbol{B} = \boldsymbol{YY'}$ by calculating the SVD of $\boldsymbol{B} = \boldsymbol{V\Lambda V'}$ and setting $\boldsymbol{Y} = \boldsymbol{V\Lambda}^{\frac{1}{2}}$, the eigenvectors and eigenvalues of the symmetric matrix $\boldsymbol{B}$. The two-dimensional MDS will then be created by the first two columns of $\boldsymbol{Y}$. Therefore, the first two columns of $\boldsymbol{Y}$ are represented by:

$$\boldsymbol{Z}_{n\times 2} = \boldsymbol{V\Lambda}^{\frac{1}{2}}\boldsymbol{J}, \text{ with } \boldsymbol{J}: p \times p = \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \text{ and } 2 \leq p$$

Equation 5:

The projection of data onto the biplot plane can be seen as a regression problem, where:

$$\boldsymbol{X} = \boldsymbol{Z\Gamma}.$$

If the MDS calculation of $\boldsymbol{Z}$ uses an Euclidean embeddable distance measure, then $\boldsymbol{Z}$ is defined in Equation 4 as:

$$\boldsymbol{Z}_{n\times 2} = \boldsymbol{V\Lambda}^{\frac{1}{2}}\boldsymbol{J}$$

The aim is therefore to find the projection $\widehat{\boldsymbol{X}}$ of the target variables $\boldsymbol{X}$ onto the two-dimensional biplot plane created by the basis $\boldsymbol{\Gamma}$, using the coordinates $\boldsymbol{Z}$.

Using the least squares formulation (Gower, Lubbe and le Roux, 2011, p206),

$$\widehat{\boldsymbol{\Gamma}} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X} \text{ and } \widehat{\boldsymbol{X}} = \boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{X}$$

Equation 6:

In the case of PCA $\widehat{\boldsymbol{X}}_{[r]} = \boldsymbol{UDJV}' = \boldsymbol{XVJV}'$ so that the plotted points (coordinates) are given by

$$\boldsymbol{Z} = \boldsymbol{XV}_r.$$

Therefore, if the MDS distance measure is Euclidean, then $\boldsymbol{\Gamma}$ can be estimated by:

$$\widehat{\boldsymbol{\Gamma}} = \left((\boldsymbol{XV}_r)'(\boldsymbol{XV}_r)\right)^{-1}(\boldsymbol{XV}_r)'\boldsymbol{X} = \left((\boldsymbol{V}_r)'\boldsymbol{X}'\boldsymbol{X}(\boldsymbol{V}_r)\right)^{-1}(\boldsymbol{V}_r)'\boldsymbol{X}'\boldsymbol{X}(\boldsymbol{V}_r\boldsymbol{V}_r') = \boldsymbol{\Lambda}_r{}^{-1}\boldsymbol{\Lambda}_r(\boldsymbol{V}_r)' = (\boldsymbol{V}_r)'$$

Equation 7:

$(\boldsymbol{V}_r)'$ is defined as the matrix consisting of the first *r* columns of $\boldsymbol{V}$, with $(\boldsymbol{V}_r{}'\boldsymbol{V}_r) = \boldsymbol{I}$. $\boldsymbol{\Lambda}_r$ is defined as the first *r* eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$, such that $\boldsymbol{V}_r{}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{V}_r = \boldsymbol{\Lambda}_r$. This gives:

$$\widehat{\boldsymbol{X}} = \boldsymbol{Z}\widehat{\boldsymbol{\Gamma}} = \boldsymbol{XV}_r\boldsymbol{V}'_r$$

Equation 8:

Previously, p was defined as the number of explanatory variables. To indicate whether the continuous or categorical data is being referred to, p can be separated into p_1 and p_2, with:

p_1 representing the number of continuous and ordinal variables and
p_2 the number of nominal categorical variables, with
$p = p_1 + p_2$.

Equation 9:

$$\widehat{\boldsymbol{x}}_{p_1} = \boldsymbol{z}'\widehat{\boldsymbol{\Gamma}}$$

Appendix C: Table of Figures

Appendix D: List of Tables

Appendix E: List of Acronyms

AoD – Analysis of Distance

AUC – Area Under the Curve

CLP – Categorical Level Points

CVA – Canonical Variance Analysis

EMC – Extended Matching Coefficient

GBM – Gradient Boosting Machines

MDS – Multi-Dimensional Scaling

PCA – Principal Component Analysis

PCO – Principal Coordinate Analysis

PDP – Partial dependence plots

ROC – Receiver Operating Characteristic

SSP – Sum-of-Squares-and-Products

SVD – Singular Value Decomposition

VIP – Variable Importance Plots

Appendix F: Bibliography

'A small multiple...' (1990). Available at: https://en.wikipedia.org/wiki/Small_multiple.

Al-Shedivat, M., Dubey, A. and Xing, E. P. (2017) 'Contextual Explanation Networks', *arxiv*, 1.

ASSA syllabus (2018). Available at: https://www.actuarialsociety.org.za/student-zone/e.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K. and Mueller, K.-R. (2010) 'How to Explain Individual Classification Decisions', *Journal of Machine Learning Research*, 11, pp. 1803–1831. Available at: http://www.jmlr.org/papers/volume11/baehrens10a/baehrens10a.pdf.

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Boland, P. J. (2006) 'Statistical methods in general insurance', in *International Conference on Teaching Statistics*, pp. 1–6.

Caruana, R., Kangarloo, H., David, J., Sinha, U. and Johnson, D. (1999) 'Case-Based Explanation of Non-Case-Based Learning Methods', *AMIA*, pp. 212–215.

Chawla, N. V, Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002) 'SMOTE : Synthetic Minority Over-sampling Technique', *Journal of Artificial Intelligence Research*, 16, pp. 321–357.

Cox, T. and Cox, M. (2008) *Handbook of Data Visualization - Multidimensional Scaling*. Springer Handbooks Comp.Statistics. Springer, Berlin, Heidelberg. doi: https://doi.org/10.1007/978-3-540-33037-0_14.

Eckart, C. and Young, G. (1936) 'The Approximation of One matrix by another of Lower Rank', *Psychometrika*, 1(3), pp. 211–218.

de Leeuw, J. and Mair, P. (2011) 'Department of Statistics Papers Multidimensional Scaling Using Majorization : SMACOF in R', *UCLA Department of Statistics Papers*.

Friedman, J. H. (2001) 'Greedy Function Approximation : A Gradient Boosting Machine', *The Annals of Statistics*, 29(5), pp. 1189–1232. Available at: http://www.jstor.org/stable/2699986.

Gabriel, K. R. (1971) 'The Biplot Graphic Display of Matrices with Application to Principal Component Analysis', *Biometrika*, 58(3), pp. 453 – 467.

Gardner-Lubbe, S., le Roux, N. J. and Gower, J. C. (2008) 'Measures of fit in principal component and canonical variate analyses', *Journal of Applied Statistics*, 35(9), pp. 947–965. doi: 10.1080/02664760802185399.

Goodman, B. and Flaxman, S. (2016) 'European Union regulations on algorithmic decision-making and a "right to explanation"', *arxiv*, 1, pp. 26–30. doi: 10.1609/aimag.v38i3.2741.

Government, S. (2006) 'National Credit Act 34 of 2005', *Government Gazette*, 469. doi: 102GOU/B.

Gower, J. C. (1966) 'Some Distance Properties of Latent Root and Vector Methods Used in Multivariate Analysis', 53(3), pp. 325–338. Available at: https://www.jstor.org/stable/2333639.

Gower, J. C. (1971) ‘A General Coefficient of Similarity and Some of Its Properties’, *Biometrics*, 27(4), pp. 857–871.

Gower, J. C. and Hand, D. J. (1996) *Biplots*. London: Chapman & Hall.

Gower, J. C. and Harding, S. (1988) ‘Nonlinear Biplots’, *Biometrika Trust Non-Normality and Tests on Variances Published by : Oxford University Press on behalf of Biometrika Trust*, 75(3), pp. 445–455.

Gower, J. C. and Krzanowski, W. J. (1999) ‘Analysis of Distance for Structured Multivariate Data and Extensions to Multivariate Analysis of Variance’, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 48(4), pp. 505–519. Available at: https://www.jstor.org/stable/2680759.

Gower, J. C., Lubbe, S. and le Roux, N. (2011) *Understanding Biplots*, *John Wiley & Sons, Ltd*. doi: 10.1002/9780470973196.

Greenacre, M. (2010) *Biplots in Practice*. Fundacion BBVA. Available at: https://www.fbbva.es/wp-content/uploads/2017/05/dat/DE_2010_biplots_in_practice.pdf.

Greenwell, B., Boehmke, B., Cunningham, J. and Developers, G. (2018) ‘Generalized Boosted Regression Models’, *CRAN*. Available at: https://github.com/gbm-developers/gbm.

Groenen, P. J. F. and Van De Velden, M. (2016) ‘Multidimensional Scaling by Majorization : A Review’, 73(8). doi: 10.18637/jss.v073.i08.

Guyon, I. and Elisseeff, A. (2003) ‘An Introduction to Variable and Feature Selection’, *Journal of Machine Learning Research*, 3, pp. 1157–1182.

Harville, D. (1997) *Matrix Analysis from a Statistician’s Perspective*. Harville DA.

Hastie, T., Tibshirani, R. and Friedman, J. H. (2008) *The Elements of Statistical Learning*. Springer-Verlag New York. doi: 10.1007/978-0-387-84858-7.

Hotelling, H. (1933) ‘Analysis of a complex of statistical variables into principal components’, *Journal of Educational Psychology*, 24.

Howell, N. (1960) *Height Weight Age Gender data set*. Available at: https://github.com/rmcelreath/rethinking/blob/master/data/Howell1.csv.

Huygens, C. (1657) ‘Huygens’ Proof of the Theorem of Pythagoras’, *The Mathematical Gazette*, 20(240).

Jerome, F., Hastie, T., Tibshirani, R. and Simon, N. (2018) ‘Lasso and Elastic-Net Regularized Generalized Linear Models’, *CRAN*. Available at: http://www.jstatsoft.org/v33/i01.

Junque De Fortuny, E. and Martens, D. (2015) ‘Active Learning-Based Pedagogical Rule Extraction’, *IEEE Transactions on Neural Networks and Learning Systems*, 26(11), pp. 2664–2677. doi: 10.1109/TNNLS.2015.2389037.

Kim, B., Rudin, C. and Shah, J. (2015) ‘The Bayesian Case Model : A Generative Approach for Case-Based Reasoning and Prototype Classification’, *arxiv*, 1, pp. 1–9.

Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006) ‘Handling Imbalanced Datasets : A review’, *GESTS International Transactions on Computer Science and Engineering*, 30.

Kruskal, J. B. (1964) 'Multi-dimensional Scaling by Optimizing goodness', *Psychometrika*, 29(1), pp. 1–27.

Kuhn, M. (2008) 'Building Predictive Models in R Using the Caret Package', *Journal Of Statistical Software*, 28(5), pp. 1–26. doi: 10.1053/j.sodo.2009.03.002.

Lakkaraju, H., Bach, S. H. and Leskovec, J. (2016) 'Interpretable decision sets: A Joint Framework for Description and Prediction', *ACM*, 1, pp. 1675–1684. Available at: http://dx.doi.org/10.1145/2939672.2939874.

Lipton, Z. C. (2017) 'The Mythos of Model Interpretability', *arxiv*, 3.

Maaten, L. Van Der and Hinton, G. (2008) 'Visualizing Data using t-SNE', *Journal of Machine Learning Research*, 9, pp. 2579–2605.

Nunez, H., Angulo, C. and Catala, A. (2006) 'Rule-Based Learning Systems for Support Vector Machines', *Neural Processing Letter*, 24, pp. 1–18. doi: 10.1007/s11063-006-9007-8.

Rao, C. R. (1948) 'The Utilization of Multiple Measurements in Problems of Biological Classification', *Wiley for the Royal Statistical Society*, 10(2), pp. 159–203. Available at: https://www.jstor.org/stable/2983775.

Registrar of Long-term Insurance and Registrar of Short-term Insurance (2014) *Board Notice Notice 158 of 2014 Financial Services Board*.

Ribeiro, M. T., Singh, S. and Guestrin, C. (2016) '"Why Should I Trust You?": Explaining the Predictions of Any Classifier'. doi: 10.1145/1235.

Ridgeway, G. (2018) 'Generalized Boosted Models : A guide to the gbm package', *CRAN*, pp. 1–15. Available at: https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf.

Robnik-Šikonja, M. and Kononenko, I. (2008) 'Explaining classifications for individual instances', *IEEE Transactions on Knowledge and Data Engineering*, 20(5), pp. 589–600. doi: 10.1109/TKDE.2007.190734.

Rousseeuw, P., Struyf, A., Hubert, M., Studer, M., Roudier, P. and Gonzalez, J. (2018) 'Finding Groups in Data: Cluster Analysis Extended', *CRAN*. Available at: https://cran.r-project.org/web/packages/cluster/cluster.pdf.

R version 3.4.4, R Core Team (2018). R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria. URL: https://www.R-project.org/

RStudio version 1.1.414, RStudio Team (2016). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL: http://www.rstudio.com/

le Roux, N. J. (2016) 'myintegrate2016: My function integrate. R package version 1.0.'

le Roux, N. J. and Lubbe, S. (2013) 'UBbipl: Understanding Biplot: Data sets and functions. R package version 3.0.4.' Available at: http://www.wiley.com/go/biplots.

Strumbelj, E. and Kononenko, I. (2010) 'An Efficient Explanation of Individual Classifications using Game Theory', *Journal of Machine Learning Research*, 11, pp. 1–18. doi: 10.1145/2858036.2858529.

'The MNIST database...' (1998). Available at: http://yann.lecun.com/exdb/mnist/.

Wald, A. (1943) 'Tests of Statistical Hypotheses Concerning Several Parameters When the Number of Observations is Large', *Transactions of the American Mathematical Society*, 54(3), pp. 426–482. Available at: http://www.jstor.org/stable/1990256.

Weng, L. (2017) *'How to Explain...'* Available at: https://lilianweng.github.io/lil-log/2017/08/01/how-to-explain-the-prediction-of-a-machine-learning-model.html.

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C. and Woo, K. (2018) 'Create Elegant Data Visualisations Using the Grammar of Graphics', *CRAN*.

Wong, J. (2016) 'Partitioned Distance Function', *CRAN*, pp. 1–4. Available at: https://github.com/jeffwong/pdist.

Yachen, Y. (2016) 'Machine Learning Evaluation Metrics', *CRAN*. Available at: http://github.com/yanyachen/MLmetrics.

Zadrozny, B. (2004) 'Learning and Evaluating Classifiers under Sample Selection Bias', in *ICML '04 Proceedings of the twenty-first International Conference on Machine Learning*.

Appendix G: Generalised biplots R code

The code for each of the functions used to create the biplots are shown below. It does not include the code used to calibrate and test the underlying machine learning model, which should be created using the 'caret' package and underlying functions. The necessary libraries must also be installed before the code will function.

0 Main - General Biplot - documented.R

```
## Load the functions+models+data:
# Load prediction model
load("ML model - glm - final.Rdata")
Model_Name <- Glm_Fit1_normal
load("dummy_model.Rdata")

# Load biplot calc functions
library(myintegrate2016)
library(pdist)
library(UBbipl)

# Load Data
load("data_train.RData")
load("clara_out.RData")
clara.med <- rownames(clara.out$medoids)

# Plotting data
plot_data <- data_train[clara.med , -1]

#  Set G grouping of the data
G <- indmat(data_train[clara.med , 1])

# add new test point
load("data_test.RData")
new_data <- data_test[120 , -1] # exclude the Target Var Y
new_data_Y <- data_test[120 , 1] # include the Target Var Y

# Load distance matrix if already available: For D in gen_biplot
load("D_alt - SSSS.RData")

# Load underlying functions and data load

source("1 GenBiplot function.R")
source("11 pred_accuracy.R")
source("2 GenBiplot drawbipl function.R")
source("31 Xgrid function.R")
source("32 prob grid function for Gen Biplot.R")
source("32 prob boundary grid.R")
source("32 Cate_class_boundary.R")
```

```
GenData <- Genbipl(X = plot_data # data must be sorted with Continuous variables first
      , G = G #NULL to make all the points the first colour in data.colours
      , specify.data =  T #T or #NULL to not plot the data
      , Title = "" # NULL

      ## New data point: List herethe data, training data will be in Grey
      # new sample will require a new D calc to plot in new position
      #, X_new_samples = new_data # NULL # Specify data must be TRUE
      #, Y_new_sample = new_data_Y # NULL # Specify data must be TRUE
      , pch.samples.size = c(0.65, 0.65) # # rep(1, 10) : Specify different size for new data

      ## plotting
      , cont.scale = c("unitSS") #c("none", "unitVar", "unitSS", "unitRange")
      , cate.scale = c("unitSS")  #c("none", "unitSS") #Standardise the cate and ordinal distance calc to give unit SS

      # Axes lables direction
      , pos = c("Orthog") # pos = c("Orthog", "Hor", "Paral")

      ## Set variables
      #, calc.dist.base = T # OR
      , calc.dist.alt = T # ordinal distance is treated differently: Must have ordinal data
      , save.dist.calc = T # save the selected distance calc
      #, D = D # will be overriden if the calc.dist functions are T
      #, zoomval = c(-1.5,1.5,-1.5,1.5)
      , exp.factor = 1.3 # overridden by zoomval

      , data.colours = c("red", "blue")
      , prob.colour = colorRampPalette(c("lightsalmon","white", "deepskyblue"))(100+1)
      , cate.colour = c("black","magenta","gold","cyan","green","grey","purple","salmon")

      # Use original or generic dummy variable names
      , use.dummy.names = T

      ## colour probabilities
      , draw.pred.area = F
      , Model_Name = Model_Name, Dummy_Model = Dummy_Model

      ## Categorical variable boundaries
      , class.bound.plot = F # plot the  the level boundaries: also use for CLP plot
      , all.class.bound = T #TRUE: plot all; FALSE: plot the category set in cate.plot
      , cate.plot = 2 # plot the stated(1,2,..) individual level boundary
      , cate.plot.region = F # TRUE fill in the region: must have: draw.pred.area = FALSE; class.bound.plot = TRUE

      ## Prediction Boundary
      , draw.pred.boundary = F
      , boundary_val = 0.6, boundary_margin = 0.002

      ## plot category classes medoids and the legend of the categories
      , cate.med.plot = T # to plot all the CLPs set: class.bound.plot = FALSE , all.class.bound = TRUE
      , cate.legend = F
```

```
        ## print prediction accuracy output
        , plot.pred.acc = F
)
```

1 GenBiplot function.R

```
Genbipl <- function (X = Remuneration.data.genbipl.2002
        , G = NULL
        , X_new_samples = NULL
        , Y_new_sample = NULL
        , specify.data = TRUE

        # Plotting method options
        , cont.scale = c("none", "unitVar", "unitSS", "unitRange")
        , cate.scale = c("none", "unitSS")

        # in Genbipl use default only for:
        , pos = c("Orthog", "Hor", "Paral")
        , label = FALSE
        , label.size = 0.6

        , pch.samples = rep(15, 10), pch.samples.size = rep(0.7, 10) # rep(1, 10)

        # plot presentation options
        , e.vects = 1:2, ax = 1:ncol(X), ax.name.size = 0.65
        , offset = rep(0.5, 4), ort.lty = 1, parplotmar = rep(3, 4)
        , Tukey.median = FALSE
        , x.grid = 0.25, y.grid = 0.25, plot.symbol = 20, plot.symbol.size = 1
        , ax.col = list(ax.col = rep(8, ncol(X)), tickmarker.col = rep(8, ncol(X)), marker.col = rep(1, ncol(X)))
        , line.length = c(1, 1), line.type = 1, line.width = 1, markers = TRUE, marker.size = 0.5
        , max.num = 2500
        , c.hull.n = 10
        , alpha = 0.95
        , zoomval = NULL #c(-1,1,-1,1)
        , exp.factor = 1.0
        , data.colours = c("red", "blue", "green", "black","grey","purple")  #  # UBcolours gives the default R colours
        , cate.colour = c("blue","green","gold","cyan","magenta","black","red","grey","purple","salmon")
        , prob.colour = colorRampPalette(c("deepskyblue","white", "lightsalmon"))(100+1) #"darkorange","deepskyblue"
        , Title = NULL
        , max.ord = 10 # specify when a continuous variable is ordinal or not
        , max.ticks = 10

        ## Distance matrix calc or import
        , calc.dist.base = FALSE
        , calc.dist.alt = FALSE
        , save.dist.calc = FALSE
        , D = NULL

        ## For drawbiplot
        # draw Biplot grid
        , use.dummy.names = TRUE
```

```
        ## Gridplot! values must be provided
        , draw.pred.area = FALSE
        , Model_Name = FALSE
        , Dummy_Model = FALSE

        ## categorical classification boundaries
        , class.bound.plot = FALSE
        , all.class.bound = FALSE
        , cate.plot = 1
        , cate.plot.region = FALSE

        ## Prediction Boundary
        , draw.pred.boundary = FALSE
        , boundary_val = 0.5, boundary_margin = 0.005

        ## plot category classes medoids
        , cate.med.plot = FALSE

        ## plot the legend
        , cate.legend = FALSE

        ## print output
        , plot.pred.acc = FALSE
        )

{

  ## load underlying functions
  source("11 pred_accuracy.R")
  source("2 GenBiplot drawbipl function.R")
  source("31 Xgrid function.R")
  source("32 prob grid function for Gen Biplot.R")
  source("32 prob boundary grid.R")
  source("32 Cate_class_boundary.R")

  dist.cont = c("Pythagoras", "Clark", "SqrtL1")
  ax.type = c("predictive", "interpolative")
  prediction.type = c("circle", "back", "normal")

  dist.cont <- dist.cont[1]
  ax.type <- ax.type[1]
  prediction.type = prediction.type[1]

  ## Debugging steps

  if (plot.pred.acc & exists("pred_accuracy") == F)
    stop("pred_accuracy function not loaded")

  if (exists("drawbipl_genbipl") == F)
    stop("Gen_Biplot Draw function not loaded")
```

```
if (exists("X_grid_function_genbi") == F)
  stop("X_grid_function_genbi function not loaded")

if (draw.pred.area & exists("prob_grid_function_genbi") == F)
  stop("prob_grid_function_genbi function not loaded")

if (class.bound.plot & exists("cate_class_boundary") == F)
  stop("cate_class_boundary function not loaded")

if (draw.pred.area  & exists("Model_Name") == F)
  stop("No prediction model loaded for prediction area calc")

if (is.na(match(cont.scale, c("none", "unitVar", "unitSS", "unitRange"))))
  stop("cont.scale must be one of: 'none','unitVar','unitSS','unitRange' \n")

if (is.na(match(cate.scale, c("none", "unitSS"))))
  stop("cate.scale must be one of: 'none','unitSS" \n")

if (calc.dist.base == F & calc.dist.alt == F & (is.null(D) || class(D) != "matrix"))
  stop("No distance matrix has been provided or calculated \n")

if (calc.dist.base == T & calc.dist.alt == T)
  stop("Please only select one distance calc method \n")

## Set variables
cont.scale <- cont.scale[1]
cate.scale <- cate.scale[1]

# Now set the Center points: CLPs = Category Level Points: therefore:
CLPs <- NULL

# plotting setting
pos <- pos[1]
old.par <- par(no.readonly = TRUE)
par(pty = "s", mar = parplotmar)

## Distance calc functions
# Only use "Pythagoras"
if (dist.cont == "Clark")
  dist.expr <- function(xik, xjk) {
    ((xik - xjk)/(xik + xjk))^2
  }
if (dist.cont == "Pythagoras")
  dist.expr <- function(xik, xjk) {
    ((xik - xjk)^2) # if sqrt then not EE anympre: should also be added to the manhattan to make it euclidean embeddable
  }
if (dist.cont == "SqrtL1")
  dist.expr <- function(xik, xjk) {
    abs(xik - xjk)  # no sqrt?
  }
```

```
D.mat <- function(X) {
  n <- nrow(X)
  D <- matrix(0, nrow = n, ncol = n)
  for (i in 1:(n - 1)) for (j in (i + 1):n) D[i, j] <- -0.5 *
    sum(dist.expr(X[i, ], X[j, ]))
  D + t(D)
}

D.cat.mat <- function(Gklist, numcat, cat.scale) {
  Dk.list <- vector("list", numcat)
  for (k in 1:numcat) {
    nn <- nrow(Gklist[[k]])
    tempmat <- matrix(1, nrow = nn, ncol = nn) - Gklist[[k]] %*% t(Gklist[[k]])
    if (cat.scale == "unitSS")  # scale to ensure total distance = -n: good for setting points, bad for setting CLPs
      tempmat <- tempmat/(sum(tempmat)/(2 * nn))

    Dk.list[[k]] <- -0.5 * (tempmat)
  }
  Dk.list
}

# Similarity function
dist.ord <- function(xik, xjk, Ri) {
  sijk = 1 - (abs(xik - xjk) / Ri)
}

# ordinal data distance function
D.ord <- function(X1, cat.scale) {
  n <- nrow(X1)
  po <- ncol(X1)
  Dt <- matrix(0, nrow = n, ncol = n)

  for (k in 1:po) {
    D <- matrix(0, nrow = n, ncol = n)
    for (i in 1:(n - 1)) for (j in (i + 1):n)
      D[i, j] <- (1 - sum(dist.ord(X1[i,k], X1[j,k], range(X1[,k])[2] )))

    if (cat.scale == "unitSS")  # scale to ensure total distance = -n: good for setting points, bad for setting CLPs
      D <- D/(sum(D)/(1 * n))

    Dt <- -0.5 * D + Dt
  }
  Dt + t(Dt)
}

dnplus1 <- function(X, x.new) {
  X1 <- X[, !categorical, drop = F]
  X2 <- X[, categorical, drop = F]
  d1 <- apply(X1, 1, function(x, x.new) -0.5 * sum(dist.expr(x, x.new)), x.new = x.new[!categorical, drop = F])
  d2 <- D.cat.mat(rbind(x.new[categorical, drop = F], X2))[1, -1]
  d1 + d2
```

```
}

ddmu.expr <- function(x) {
  if (dist.cont == "Pythagoras" || dist.cont == "Clark") {
    dist.text <- deparse(dist.expr)[3]
    dist.text <- paste("-0.5", dist.text, sep = "*")
    dist.text <- gsub("xik", x, dist.text)
    dist.text <- gsub("xjk", "mu", dist.text)
    dist.expression <- parse(text = dist.text)
    AA <- D(dist.expression, "mu")
  }
  if (dist.cont == "SqrtL1")
    AA <- expression(-(sign(x - mu)))
  return(AA)
}

d2dmu2.expr <- function(x) {
  first <- ddmu.expr(x)
  D(first, "mu")
}

sumlist <- function(listmat) {
  sum <- 0
  for (i in 1:length(listmat)) sum <- sum + listmat[[i]]
  sum
}

##################### START ########################

## Combine new data: Create new data, G and colour
if (!is.null(X_new_samples)) {
  G <- indmat(c(rep(0,nrow(X)), nrow(X_new_samples)))
  #data.colours <- c("grey", "green")
  if (!is.null(Y_new_sample)) {
    data.colours <- c("grey", data.colours[Y_new_sample+1])
  }
  X <- rbind(X, X_new_samples) # exclude the Target Var Y
}

if ( !is.null(D) )
  if ( (nrow(D) != nrow(X))  )
    stop("Distance matrix dimension <> data dimension")

## Start Calculations
n <- nrow(X)
p <- ncol(X)
categorical <- rep(F, p)
for (j in 1:p) if (is.factor(X[, j]))
  categorical[j] <- T

p2 <- sum(categorical)
```

```
p1 <- p - p2
unscaled.X <- X  # store the original data

# If the variable is too skew and ordinal in nature but large difference in unique values, make it a leveled ordinal variable
for (i in 1:p1)
  if ( ( (range(X[, i])[2] - range(X[, i])[1]) > 2 * length(unique(X[, i])) ) & length(unique(unscaled.X[, i])) < max.ord ) {
    print(paste(colnames(X)[i],": Range of ordinal variable too long given values; Can give plotting problems later"))
    print(table(X[, i]))
  }

if (p1 == 0)
 stop("No continuous variables in data: Currently model needs some \n")

if(cate.plot > p2 & class.bound.plot == TRUE)
 stop("Category boundary plot nr > than nr of categories \n")

# Ensure that there is tick marks on the axes for slow range variables: mostly for ordinal variables
# Have less than max.ticks when the range of integers in the variable is < max.ticks
n.int <- rep(max.ticks, p1)
maxval <- rep(0, p1)
for (i in 1:p1) {
 maxval[i] <- length(unique(unscaled.X[, i]))
}
n.int[1:p1] <- pmin(maxval*2, max.ticks)   # ensure there is enough points on plot

# if no continuous or categorical variables, still need Delta 1 and 2
Delta1 <- matrix(0, nrow = n, ncol = n)
Delta2 <- matrix(0, nrow = n, ncol = n)

## Standardisation of Continuous + Ordinal data
gems <- rep(0, p1)
sds <- rep(1, p1)

if (cont.scale == "unitVar") {
 gems <- apply(unscaled.X[, !categorical], 2, mean)
 sds <- sqrt(apply(unscaled.X[, !categorical], 2, var))
 X[, !categorical] <- scale(unscaled.X[, !categorical, drop = FALSE], center = TRUE, scale = sds)
}
if (cont.scale == "unitSS") {
 gems <- apply(unscaled.X[, !categorical, drop = FALSE], 2, mean)
 sds <- sqrt((n - 1) * apply(unscaled.X[, !categorical, drop = FALSE], 2, var))
 X[, !categorical] <- scale(unscaled.X[, !categorical, drop = FALSE], center = TRUE, scale = sds)
}
if (cont.scale == "unitRange") {
 gems <- apply(unscaled.X[, !categorical], 2, mean)
 sds <- apply(unscaled.X[, !categorical], 2, function(x) max(x) - min(x))
 X[, !categorical] <- scale(unscaled.X[, !categorical, drop = FALSE], center = TRUE, scale = sds)
}

Xmat <- X # combine standardised continuous and the normal categorical
Gmat <- NULL
```

```
Gk.list <- vector("list", p2)
L <- 0

# create Gk data set for each categorical variable
if(p2 > 0) {
  for (k in 1:p2) {
    Gk <- indmat(Xmat[, categorical, drop = F][, k])
    Gk.list[[k]] <- Gk
    Gmat <- cbind(Gmat, Gk)
    L <- L + ncol(Gk)
  }
}

## Distance measure calc:
#1 standard method
if(calc.dist.base) {
  Delta1 <- D.mat(Xmat[, !categorical, drop = F])

  # Categorical variables
  if(p2 > 0) {
    Delta2.list <- D.cat.mat(Gk.list, numcat = p2, cat.scale = cate.scale)
    Delta2 <- sumlist(Delta2.list)
  }

  D <- Delta1 + Delta2

  if(save.dist.calc) {
    save(D, file = "D_base.RData")
    # save(Delta1, file = "Delta1_base.RData")
  }
}

#2 alternative distance
if(calc.dist.alt) {

  # set ordinal var values for alt dist measure
  ordinal <- rep(F, p)
  continuous <- rep(F, p)
  # treat as ordinal all continuous variable with a max value < cut.off.cont
  for (i in 1:p1) {
    if (length(unique(unscaled.X[, i])) < max.ord) {
      ordinal[i] <- T
    }

    if (length(unique(unscaled.X[, i])) >= max.ord) {
      continuous[i] <- T
    }
  }

  # Continuous variables
  Delta1c <- D.mat(Xmat[, continuous, drop = F])
```

```
  # Ordinal variable:
  Delta1o <- matrix(0, nrow = n, ncol = n)
  if(sum(ordinal) > 0) {
    # D will be scaled by unitSS distance: so when axes are used for projection
    # then the same scaling is needed, and the unitSS gems and sds calculated
    Delta1o <- D.ord(unscaled.X[, ordinal, drop = F], cat.scale = cate.scale)
  }

  # Categorical variables
  if(p2 > 0) {
    Delta2.list <- D.cat.mat(Gk.list, numcat = p2, cat.scale = cate.scale)
    Delta2 <- sumlist(Delta2.list)
  }

  # Add data together
  Delta1 <- Delta1c + Delta1o
  D <-   Delta1 + Delta2   # sum(Delta1c); sum(Delta1o);  sum(Delta2)

  if(save.dist.calc) {
  save(D, file = "D_alt.RData")
  }
}

## Remove variable axes names:
cont.names <- dimnames(Xmat)[[2]][!categorical]
cate.names <- dimnames(Xmat)[[2]][categorical]
if (use.dummy.names) {
  cont.names <- sprintf("X%s", seq(1:(length(dimnames(Xmat)[[2]][!categorical]))))
  cate.names <- sprintf("Z%s", seq(1:(length(dimnames(Xmat)[[2]][categorical]))))
}

## Classical Scaling:
s <- rep(1/n, n)
one <- rep(1, n)
N <- one %*% t(s)
I <- diag(n)
B <- (I - N) %*% D %*% (I - N)
  # Data is centred
  # B symmetrix, so SVD(B) gives U = V
swd <- svd(B)
U <- (swd$u %*% diag(swd$d^0.5))[, -n]
lambda <- t(U) %*% U  #U = Y, Y'Y gives D
lambda.inv <- diag(ifelse(zapsmall(diag(lambda)) > 0, 1/diag(lambda), 0))
eigval.pos <- sum(zapsmall(swd$d) > 0)
eigval <- diag(lambda)
eigval.r <- eigval[e.vects[1:2]]

Z <- U[, e.vects[1:2]] # plotted 2d points

if (is.null(G)) J <- 0 else J <- ncol(G)
```

```
Z <- data.frame(Z, pch.sampl = pch.samples[1], colr = as.character(data.colours[1]),
                line.type = line.type[1], pch.sampl.size = pch.samples.size[1],  stringsAsFactors = FALSE)
if (J > 0) {
  for (j in 1:J) {
    Z[G[, j] == 1, 4] <- as.character(data.colours[j])
    Z[G[, j] == 1, 3] <- pch.samples[j]
    Z[G[, j] == 1, 5] <- line.type[j]
    Z[G[, j] == 1, 6] <- pch.samples.size[j]
  }
}
rownames(Z) <- rownames(Xmat)

if (ax.type == "predictive")
{
  z.axes <- lapply(1:p1, function(k, Xmat, unscaled.X, means, sd, n.int, D, G, lambda, s, e.vecs, prediction.type)
  {
    number.points <- 50

    std.markers <- pretty(unscaled.X[, k], n = n.int[k])
    std.markers <- std.markers[std.markers >= min(unscaled.X[,  k]) & std.markers <= max(unscaled.X[, k])]
    interval <- (c(std.markers, min(unscaled.X[, k]), max(unscaled.X[, k])) - means[k])/sd[k]
    axis.vals <- seq(from = min(interval), to = max(interval), length = number.points)
    axis.vals <- sort(unique(c(axis.vals, interval)))
    axis.vals <- zapsmall(axis.vals,10)

    axis.vals <- axis.vals[!axis.vals == 0]
    number.points <- length(axis.vals)
    n <- nrow(unscaled.X)
    p <- ncol(unscaled.X)

    axis.points <- matrix(0, nrow = number.points, ncol = 4)
    mu <- axis.vals
    m <- length(mu)
    ddmu.dnplus1.mat <- t(sapply(Xmat[, k], function(x) eval(ddmu.expr(x))))

    A.mat <- t(ddmu.dnplus1.mat) %*% G[, e.vects] %*% solve(lambda[e.vects, e.vects])
    m.star.vec <- apply(ddmu.dnplus1.mat, 2, sum)/(-n)
    a1.sq.plus.a2.sq <- apply(A.mat, 1, function(a) sum(a^2))

    LL.mat <- cbind(A.mat, m.star.vec, A.mat/sqrt(a1.sq.plus.a2.sq),
                    m.star.vec/sqrt(a1.sq.plus.a2.sq))

    L.a <- function(mu.val, X, k) {
      ddmu.dnplus1.vec <- sapply(X[, k], function(x,  mu) eval(ddmu.expr(x)), mu = mu.val)
      t(ddmu.dnplus1.vec) %*% G[, e.vects] %*% solve(lambda[e.vects, e.vects])
    }
    L.m <- function(mu.val, X, k) {
      ddmu.dnplus1.vec <- sapply(X[, k], function(x, mu) eval(ddmu.expr(x)), mu = mu.val)
      m.star <- sum(ddmu.dnplus1.vec)/(-nrow(X))
      m.star/sum(L.a(mu.val, X, k)^2)
    }
```

```
# Prediction methods:
if (prediction.type == "circle") {
  axis.points[, 1:2] <- cbind(LL.mat[, 4] * LL.mat[, 6], LL.mat[, 5] * LL.mat[, 6])
  # what is in the 4*6 = l1(mu)*t(mu); 5*6 = l2(mu)*t(mu) p223
}

if (prediction.type == "back") {
  mat <- sapply(mu, function(mm, unscaled.X, k) {
    my.vek <- rep(0, ncol(unscaled.X))
    my.vek[k] <- mm
    dnplus1(unscaled.X, my.vek)}, unscaled.X = Xmat, k = k)

  xsi.k <- solve(lambda[e.vects, e.vects]) %*% t(G[, e.vects]) %*% (mat - apply(D, 1, sum)/n)
  axis.points[, 1:2] <- t(sapply(1:m, function(j, xsi.k, LL.mat) {
    l.vec <- LL.mat[j, 4:5]
    m.mu <- LL.mat[j, 6]
    xsi <- xsi.k[, j]
    (diag(2) - l.vec %*% t(l.vec)) %*% xsi + m.mu * l.vec}, xsi.k = xsi.k, LL.mat = LL.mat))
} #end back

if (prediction.type == "normal") {
  epsilon <- 0.001
  pos <- order(abs(LL.mat[, 6]))[1]
  if (LL.mat[pos, 6] < epsilon)
    mu.0 <- axis.vals[pos]
  else {
    interval.begin <- pos - 1
    if (interval.begin < 1)
      interval.begin <- 1
    interval.einde <- pos + 1
    if (interval.einde > number.points)
      interval.einde <- number.points
    uit <- uniroot(L.m, axis.vals[c(interval.begin, interval.einde)], X = Xmat, k)
    mu.0 <- uit$root
  }

  skep.integrand.1 <- function(mu, XX, k) {
    sapply(mu, function(mu, Xmat, k) {
      ddmu.dnplus1.vec <- sapply(XX[, k], function(x,
                                          mu) eval(ddmu.expr(x)), mu = mu)
      a.vec <- t(ddmu.dnplus1.vec) %*% G[, e.vects] %*%
        solve(lambda[e.vects, e.vects])
      wortel <- sqrt(sum(a.vec^2))
      l1.mu <- a.vec[1]/wortel
      mster.mu <- sum(ddmu.dnplus1.vec)/(-nrow(XX))
      d2.dmu2.dnplus1.vec <- sapply(Xmat[, k],
                                    function(x, mu) eval(d2dmu2.expr(x)), mu = mu)
      ddmu.mster.mu <- sum(d2.dmu2.dnplus1.vec)/(-nrow(XX))
      ddmu.a1 <- t(G[, 1]) %*% d2.dmu2.dnplus1.vec/lambda[1, 1]
      ddmu.a2 <- t(G[, 2]) %*% d2.dmu2.dnplus1.vec/lambda[2, 2]
```

```
      ddmu.wortel <- (a.vec[1] * ddmu.a1 + a.vec[2] *
                      ddmu.a2)/wortel
      ddmu.mmu.l2mu <- ((ddmu.mster.mu * wortel -  ddmu.wortel * mster.mu) * a.vec[2]
                      - (ddmu.a2 * wortel - ddmu.wortel * a.vec[2]) * mster.mu)/(wortel * a.vec[2]^2)
      l1.mu * ddmu.mmu.l2mu
    }, Xmat = Xmat, k = k)
  }

  f1.int <- sapply(mu, function(mu, Xmat, k) my.integrate(skep.integrand.1, mu.0, mu, Xmat, k)$value, Xmat = Xmat, k = k)
  axis.points[, 1] <- LL.mat[, 5] * f1.int
  skep.integrand.2 <- function(mu, XX, k) {
    sapply(mu, function(mu, Xmat, k) {
      ddmu.dnplus1.vec <- sapply(XX[, k], function(x, mu) eval(ddmu.expr(x)), mu = mu)
      a.vec <- t(ddmu.dnplus1.vec) %*% G[, e.vects] %*%
        solve(lambda[e.vects, e.vects])
      wortel <- sqrt(sum(a.vec^2))
      l2.mu <- a.vec[2]/wortel
      mster.mu <- sum(ddmu.dnplus1.vec)/(-nrow(XX))
      d2.dmu2.dnplus1.vec <- sapply(Xmat[, k],
                                    function(x, mu) eval(d2dmu2.expr(x)), mu = mu)
      ddmu.mster.mu <- sum(d2.dmu2.dnplus1.vec)/(-nrow(XX))
      ddmu.a1 <- t(G[, 1]) %*% d2.dmu2.dnplus1.vec/lambda[1, 1]
      ddmu.a2 <- t(G[, 2]) %*% d2.dmu2.dnplus1.vec/lambda[2, 2]
      ddmu.wortel <- (a.vec[1] * ddmu.a1 + a.vec[2] * ddmu.a2)/wortel
      ddmu.mmu.l1mu <- ((ddmu.mster.mu * wortel - ddmu.wortel * mster.mu) * a.vec[1]
                      - (ddmu.a1 * wortel - ddmu.wortel * a.vec[1]) * mster.mu)/(wortel * a.vec[1]^2)
      l2.mu * ddmu.mmu.l1mu
    }, Xmat = Xmat, k = k)
  }
  f2.int <- sapply(mu, function(mu, Xmat, k) my.integrate(skep.integrand.2, mu.0, mu, Xmat, k)$value, Xmat = Xmat, k = k)
  axis.points[, 2] <- LL.mat[, 4] * f2.int
 }

  axis.points[, 3] <- zapsmall(axis.vals * sd[k] + means[k])
  for (i in 1:number.points) if (any(zapsmall(axis.points[i, 3] - std.markers) == 0))
    axis.points[i, 4] <- 1
  return(axis.points)
 }, Xmat = Xmat[, !categorical, drop = F], unscaled.X = unscaled.X[,  !categorical, drop = F], means = gems, sd = sds,
 n.int = n.int, D = D, G = U, lambda = lambda, s = s,
 e.vecs = e.vects, prediction.type = prediction.type) #close for continuous variables

## Regression value: Use MDS data points to get the Axes vectors
Z.points <-  as.matrix(Z[, 1:2])
X.contval <- as.matrix(Xmat[, !categorical])
R.betas <- solve(t(Z.points) %*% Z.points) %*% t(Z.points) %*% X.contval

## For categorical variables, create medoids in the biplot space:
Z.cate <-  cbind(Z[, 1:2],unscaled.X[, categorical, drop = F])
cate.med.points <- data.frame(1,2,3,4,5,6)

if(p2 > 0) {
```

```
  r <- 0
  for ( k in 1:p2) {
    cat.vals <- levels(Xmat[, categorical, drop = F][, k])
    clevels <- length(cat.vals)
    for (l in 1:clevels) {
      r <- r+1
      z.cate.points <- Z.cate[Z.cate[, k+2] == cat.vals[l], 1:2]
      cate.med.points[r,1:2] <- c(0,0)
      if(nrow(z.cate.points) > 0) {
        med.z.cate <- kmeans(z.cate.points, centers = 1)
        cate.med.points[r,1:2] <- med.z.cate$centers
      }
      cate.med.points[r,3] <- colnames(unscaled.X[, categorical, drop = F])[k]
      cate.med.points[r,4] <- k
      cate.med.points[r,5] <- cat.vals[l]
      cate.med.points[r,6] <- l
    }
  }
}

## Draw
uit <- drawbipl_genbipl(Z,
                        z.axes = z.axes,
                        z.axes.names = cont.names,
                        cate.names = cate.names,
                        p = p1,
                        ax = ax[1:p1]

                        , specify.data = specify.data

                        , exp.factor = exp.factor
                        , zoomval = zoomval
                        , data.colours = data.colours
                        , cate.colour = cate.colour
                        , prob.colour =  prob.colour
                        , label = label
                        , marker.size = marker.size
                        , label.size = label.size
                        , markers = markers, Title = Title
                        , ax.col = ax.col, ax.name.size = ax.name.size
                        , Tukey.median = Tukey.median, offset = offset
                        , pos = pos, strepie = line.length, max.num = max.num
                        , line.width = line.width, class.vec = class.vec
                        , ort.lty = ort.lty, parplotmar = parplotmar
                        , x.grid = x.grid, y.grid = y.grid, plot.symbol = plot.symbol
                        , plot.symbol.size = plot.symbol.size
                        , c.hull.n = c.hull.n
                        , alpha = alpha
                        , p1 = p1, p2 = p2
                        , categorical = categorical
                        , Xmat = Xmat
```

```
                , cate.med.points = cate.med.points
                , R = R.betas
                , sds = sds, gems = gems

                ## colour probabilities
                , draw.pred.area = draw.pred.area
                , Model_Name = Model_Name, Dummy_Model = Dummy_Model

                ## categorical classification boundaries
                , class.bound.plot = class.bound.plot
                , all.class.bound = all.class.bound
                , cate.plot = cate.plot
                , cate.plot.region = cate.plot.region

                ## Prediction Boundary
                , draw.pred.boundary = draw.pred.boundary
                , boundary_val = boundary_val, boundary_margin = boundary_margin

                ## plot category classes medoids
                , cate.med.plot = cate.med.plot

                ## plot the legend
                , cate.legend = cate.legend
                )

  pred.accuracy.output <- 0
  if (plot.pred.acc)
    pred.accuracy.output <- pred_accuracy(unscaled.X, X.contval, R.betas, Z.points, p1, p2, categorical, cate.med.points, sds, gems,
eigval, uit$usr)

  # Output
    list(Z = Z, Z.axes = z.axes, e.vals = eigval, R.betas = R.betas,
         normalized.X.cont = Xmat[, !categorical],
         centred.vec = gems, scaling.vec = sds, usr = uit$usr,
         predictions = pred.accuracy.output, p2 = p2,
         var.names <- dimnames(Xmat)[[2]],
         cate.med.points = cate.med.points
         ,xgrid = uit$X_grid_out
         )
}
```

11 pred_accuracy.R

```
pred_accuracy <- function(Xmat, X.contval, R, Zs, p1, p2, categorical, CLPs, sd, gem, eigval, usr ) {
  library(pdist)
  n <- nrow(Xmat)
  Xall <- as.matrix(Xmat[, 1:p1])

  #usr: c(x1, x2, y1, y2)
  print(paste("Plot area xmin: ",par("usr")[1]))
  print(paste("Plot area xmax: ",par("usr")[2]))
  print(paste("Plot area ymin: ",par("usr")[3]))
```

```
print(paste("Plot area ymax: ",par("usr")[4]))

## overall quality that uses combined distance matrix eigvals
fit.quality <- round(((eigval[1] + eigval[2])/sum(eigval)) * 100, digits = 2)
print(paste("Quality of display = ", fit.quality, "%"))
print("--------")

## Predict continuous variables
  predict.cont <- (Zs[1:n, ] %*% R)

  predict.cont <- sweep(predict.cont, 2, sd, FUN = "*")
  predict.cont <- sweep(predict.cont, 2, gem, FUN = "+")
  predict.cont <- as.matrix(predict.cont)

  fit.cont.predictivity.mat <- NULL
  fit.cont.predictivity.axes <- round( diag(t(predict.cont) %*% (predict.cont)) / diag(t(Xall) %*% (Xall)), 2)
  fit.cont.predictivity.sample <- round( diag((predict.cont) %*% t(predict.cont)) / diag((Xall) %*% t(Xall)), 2)
  fit.cont.adequacy <- NULL

  print(paste("Continuous Axis predictivity"))
  for (k in 1:p1) {
    cont.names <- colnames(Xmat[, !categorical, drop = F])[k]
    print(paste(cont.names," = ", fit.cont.predictivity.axes[k] * 100, "%"))
  }

## Predict categorical variables
predict.cat <- matrix(0, ncol = p2, nrow = n )
colnames(predict.cat) <- dimnames(Xmat)[[2]][categorical]
predict.cat <- as.data.frame(predict.cat)

predict.class <- matrix(0, ncol = p2, nrow = n )
colnames(predict.class) <- dimnames(Xmat)[[2]][categorical]
predict.class <- as.data.frame(predict.cat)

if(p2 > 0)
  for (k in 1:p2) {
    CLPs.k <- CLPs[CLPs[, 4] == k, 1:2]
    cat.names <- levels(Xmat[, categorical, drop = F][, k])
    cat.levels <- length(cat.names)

    point.dist.matrix <- as.matrix(pdist(Zs, CLPs.k), nrow = n, ncol = cat.levels)
    point.classify <- apply(point.dist.matrix,1,which.min)
    predict.class[, k] <-  point.classify
    predict.cat[, k] <-  cat.names[point.classify]
  }

# combine two sets of predictions
pred.all <- cbind(round(predict.cont,2),  predict.cat)
# tables of categorical data
fit.cate.predictivity.axes <- matrix(0, nrow = 1, ncol = p2)
fit.cate.predictivity.sample.all <- matrix(0, nrow = n, ncol = p2)
```

```
  if(p2 > 0) {
    print("--------")
    print(paste("Categorical Axis predictivity"))
    colnames(fit.cate.predictivity.axes) <- dimnames(Xmat)[[2]][categorical]
    for (k in 1:p2) {
      correct <- 0
      correct <- correct + sum(Xmat[, categorical, drop = F][, k] == predict.cat[, k])

      print(dimnames(Xmat)[[2]][categorical][k])
      print(table(Xmat[, categorical, drop = F][, k], predict.cat[, k]))
      print(paste(round(correct / n, 2) * 100, "%"))
      print("--------")

      fit.cate.predictivity.axes[1, k] <- round(correct / n, 2)
      fit.cate.predictivity.sample.all[, k] <- Xmat[, categorical, drop = F][, k] == predict.cat[, k]
    }
  }

  if(p2 > 0)
    fit.cate.predictivity.sample <- apply(fit.cate.predictivity.sample.all, 1, sum) / p2
  if(p2 == 0)
    fit.cate.predictivity.sample <- 0

  predictivity.axes <- cbind(t(as.matrix(fit.cont.predictivity.axes)), fit.cate.predictivity.axes)
  predictivity.sample <- (fit.cont.predictivity.sample * p1 + fit.cate.predictivity.sample * p2) / (p1 + p2)

  list(fit.quality = fit.quality
       , predictivity.axes = predictivity.axes
       , predictivity.sample = predictivity.sample
       , pred.all = pred.all
       )
}
```

2 GenBiplot drawbipl function.R

```
drawbipl_genbipl <- function (Z, z.axes, z.axes.names = NULL, cate.names = NULL
          , p
          , ax = NULL
          , pos
          , specify.data = NULL
          , alpha = 0.5
          , c.hull.n = 10
          , ax.col, ax.name.size,  class.vec
          , label = label, label.size, line.width = 1, markers = TRUE
          , marker.size, max.num, offset = rep(0.5, 4), ort.lty = 1
          , parplotmar = rep(2, 4)
          , strepie = c(1, 1)
          , Title = NULL
          , Tukey.median = TRUE
          , x.grid = 0.25, y.grid = 0.25
          , plot.symbol = 20
```

```
        , plot.symbol.size = 1
        , zoomval = NULL
        , exp.factor = 1.1
        , data.colours = UBcolours
        , cate.colour = c("blue","green","gold","cyan","magenta","black","red","grey","purple","salmon")
        , prob.colour = colorRampPalette(c("deepskyblue","white", "lightsalmon"))(100+1) #"darkorange","deepskyblue"

        ## X Grid data: Calculated in GenBipl
        , p1 = NULL, p2 = NULL
        , categorical = NULL
        , Xmat = NULL
        , cate.med.points = NULL
        , R = NULL
        , sds = NULL, gems = NULL

        ## Prediction grid
        , draw.pred.area = FALSE
        , Model_Name = NULL, Dummy_Model = NULL

        ## Prediction Boundary
        , draw.pred.boundary = FALSE
        , boundary_val = 0.5, boundary_margin = 0.005

        ## categorical classification boundaries
        , class.bound.plot = FALSE
        , all.class.bound = FALSE
        , cate.plot = 1
        , cate.plot.region = FALSE

        ## plot category classes medoids
        , cate.med.plot = FALSE

        ## plot the legend
        , cate.legend = FALSE

        )

{

    specify.classes = NULL
    straight = TRUE # for axes prediction
    predictions.mean = NULL
    predictions.sample = NULL
    pch.means = 15
    pch.means.size = 1
    Z.means.mat = NULL
    CLPs = NULL
    pch.samples = 1
    pch.samples.size = 1

  if (!is.null(Z.means.mat)) {
```

```
  if (ncol(Z.means.mat) == 5)
    Z.means.mat <- data.frame(Z.means.mat[, 1:5], pch.means.size = pch.means.size, stringsAsFactors = FALSE)
}

.orthog.pred.line <- function(x1, y1, x2, y2, val1 = NULL, val2 = NULL, px, py, ort.lty) {
  if (x1 - x2 == 0) gradient <- Inf  else {
    if (y1 - y2 == 0) gradient <- 0 else gradient <- (y1 - y2)/(x1 - x2)
  }

  if (gradient == 0) {
    x.star <- px
    y.star <- y1
    prediction <- val1 + (x.star - x1) * (val2 - val1)/(x2 -  x1)
  } else {if (gradient == Inf) {
    x.star <- x1
    y.star <- py
    prediction <- val1 + (y.star - y1) * (val2 - val1)/(y2 - y1)
  }  else {
    intcept <- y1 - gradient * x1
    intcept.star <- py + (1/gradient) * px
    x.star <- (intcept.star - intcept)/(gradient + 1/gradient)
    y.star <- intcept + gradient * x.star
    prediction <- val1 + (x.star - x1) * (val2 - val1)/(x2 - x1)
  }
  }
  lines(x = c(px, x.star), y = c(py, y.star), lty = ort.lty)
  prediction
}

.draw.marker <- function(x, y, grad, expand = 1, both.sides = TRUE, col = 1) {
  usr <- par("usr")
  uin <- par("pin")/c(usr[2] - usr[1], usr[4] - usr[3])
  mm <- 1/(uin[1] * 25.4)
  d <- expand * mm
  b <- d * sqrt(1/(1 + grad * grad))
  a <- b * grad
  if (is.infinite(grad)) {
    if (sign(grad) > 0)
      lines(c(x, x), c(y, y + d), col = col)
    else lines(c(x, x), c(y, y - d), col = col)
  }
  else {
    if (both.sides == FALSE)
      lines(c(x, x + b), c(y, y + a), col = col)
    else lines(c(x - b, x + b), c(y - a, y + a), col = col)
  }
}

.marker.value <- function(x, y, grad, mark, expand = 1, marker.size, col = 1, pos = 1) {

  #print(mark)
```

```
  usr <- par("usr")
  uin <- par("pin")/c(usr[2] - usr[1], usr[4] - usr[3])
  mm <- 1/(uin[1] * 25.4)
  d <- (expand + 2) * mm
  b <- d * sqrt(1/(1 + grad * grad))
  a <- b * grad
  if (is.infinite(grad)) {
    if (sign(grad) > 0)
      text(x = x, y = y + d, labels = mark, cex = marker.size,
           col = col, pos = pos)
    else text(x = x, y = y - d, labels = mark, cex = marker.size,
              col = col, pos = pos)
  }
  else {
    text(x = x, y = y, labels = mark, cex = marker.size,
         col = col, pos = pos)
  }
}


.draw.axis <- function(marker.mat, line.name = NULL, offset = c(0.5, 0.5, 0.5, 0.5),
                       pos = "Orthog", axis.col = "black", ax.name.col = "black", ax.name.size = 0.65) {

  if (!is.element(pos[1], c("Hor", "Orthog", "Paral")))
    stop("Argument pos must be one of 'Hor','Orthog' or 'Paral' ")
  if (pos[1] == "Hor") {
    par(las = 1)
    adjust <- c(0.5, 1, 0.5, 0)
  }
  if (pos[1] == "Orthog") {
    par(las = 2)
    adjust <- c(0, 0, 0, 0)
  }
  if (pos[1] == "Paral") {
    par(las = 0)
    adjust <- c(0.5, 0.5, 0.5, 0.5)
  }

  marker.mat <- marker.mat[order(marker.mat[, 1]), , drop = FALSE]
  x.vals <- marker.mat[, 1]
  y.vals <- marker.mat[, 2]
  marker.vals <- marker.mat[, 3]
  test <- rep(NA, 4)
  test[1] <- min(x.vals) < usr[1]
  test[2] <- max(x.vals) > usr[2]
  test[3] <- min(y.vals) < usr[3]
  test[4] <- max(y.vals) > usr[4]

  if (y.vals[1] == y.vals[length(y.vals)] & x.vals[1] ==
      x.vals[length(x.vals)])
    type.line <- "plot.5"
```

```
  if (y.vals[1] == y.vals[length(y.vals)] & x.vals[1] !=
      x.vals[length(x.vals)]) {
    gradient <- 0
    intercept <- y.vals[1]
    type.line <- "plot.1"
  }
  if (y.vals[1] != y.vals[length(y.vals)] & x.vals[1] ==
      x.vals[length(x.vals)]) {
    gradient <- Inf
    intercept <- x.vals[1]
    type.line <- "plot.2"
  }
  if (y.vals[1] != y.vals[length(y.vals)] & x.vals[1] !=
      x.vals[length(x.vals)]) {
    gradient <- (y.vals[1] - y.vals[length(y.vals)])/(x.vals[1] - x.vals[length(x.vals)])
    intercept <- y.vals[1] - gradient * x.vals[1]

    # function for drawing a straight line: a + bx
    y1.ster <- gradient * usr[1] + intercept
    y2.ster <- gradient * usr[2] + intercept
    x1.ster <- (usr[3] - intercept)/gradient
    x2.ster <- (usr[4] - intercept)/gradient
    if (gradient > 0)
      type.line <- "plot.3"
    if (gradient < 0)
      type.line <- "plot.4"
  }

  # 5 types of lines
  # abline draws a line for you in R based on formula a + bx
  plot.1 <- function(marker.mat, line.name = NULL, offset = NULL, adjust = NULL, axis.col = "black", ax.name.col = "black",
ax.name.size = 0.65) {
    abline(h = marker.mat[1, 2], col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        if (marker.mat[1, 3] - marker.mat[nrow(marker.mat), 3] < 0)
          mtext(text = line.name, side = 4, line = 0 + offset[4], adj = adjust[4], at = marker.mat[1, 2], col = ax.name.col, cex =
ax.name.size)
        else mtext(text = line.name, side = 2, line = 0 + offset[2], adj = adjust[2], at = marker.mat[1, 2], col = ax.name.col, cex =
ax.name.size)
      }
    }
  }
  plot.2 <- function(marker.mat, line.name = NULL, offset = NULL, adjust = NULL, axis.col = "black", ax.name.col = "black",
ax.name.size = 0.65) {
    abline(v = marker.mat[1, 1], col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        test <- order(marker.mat[, 2])[c(1, nrow(marker.mat))]
        if (marker.mat[test[2], 3] - marker.mat[test[1],
                                                3] > 0)
```

```
      mtext(text = line.name, side = 3, line = 0 +
            offset[3], adj = adjust[3], at = marker.mat[1, 1], col = ax.name.col, cex = ax.name.size)
    else mtext(text = line.name, side = 1, line = 0 +
               offset[1], adj = adjust[1], at = marker.mat[1,  1], col = ax.name.col, cex = ax.name.size)
   }
  }
}
plot.3 <- function(y1.ster, y2.ster, x1.ster, x2.ster,
                   marker.mat, line.name = NULL, offset = NULL, adjust = NULL,
                   axis.col = "black", ax.name.col = "black", ax.name.size = 0.65) {
 if (y1.ster >= usr[3] & y2.ster >= usr[4]) {
  lines(c(usr[1], x2.ster), c(y1.ster, usr[4]),
        col = axis.col)
  if (!is.null(line.name)) {
   if (ncol(marker.mat) == 3) {
    if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                      3] > 0)
     mtext(text = line.name, side = 2, line = 0 +
           offset[2], adj = adjust[2], at = y1.ster,
           col = ax.name.col, cex = ax.name.size)
    else mtext(text = line.name, side = 3, line = 0 +
               offset[3], adj = adjust[3], at = x2.ster,
               col = ax.name.col, cex = ax.name.size)
   }
  }
 }
 if (y1.ster > usr[3] & y2.ster < usr[4]) {
  lines(c(usr[1], usr[2]), c(y1.ster, y2.ster),
        col = axis.col)
  if (!is.null(line.name)) {
   if (ncol(marker.mat) == 3) {
    if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                      3] > 0)
     mtext(text = line.name, side = 2, line = 0 +
           offset[2], adj = adjust[2], at = y1.ster,
           col = ax.name.col, cex = ax.name.size)
    else mtext(text = line.name, side = 4, line = 0 +
               offset[4], adj = adjust[4], at = y2.ster,
               col = ax.name.col, cex = ax.name.size)
   }
  }
 }
 if (y1.ster < usr[3] & y2.ster > usr[4]) {
  lines(c(x1.ster, x2.ster), c(usr[3], usr[4]),
        col = axis.col)
  if (!is.null(line.name)) {
   if (ncol(marker.mat) == 3) {
    if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                      3] > 0)
     mtext(text = line.name, side = 1, line = 0 +
           offset[1], adj = adjust[1], at = x1.ster,
```

```
            col = ax.name.col, cex = ax.name.size)
        else mtext(text = line.name, side = 3, line = 0 +
                     offset[3], adj = adjust[3], at = x2.ster,
                   col = ax.name.col, cex = ax.name.size)
      }
    }
  }
  if (y1.ster < usr[3] & y2.ster < usr[4]) {
    lines(c(x1.ster, usr[2]), c(usr[3], y2.ster),
          col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                          3] > 0)
          mtext(text = line.name, side = 1, line = 0 +
                  offset[1], adj = adjust[1], at = x1.ster,
                col = ax.name.col, cex = ax.name.size)
        else mtext(text = line.name, side = 4, line = 0 +
                     offset[4], adj = adjust[4], at = y2.ster,
                   col = ax.name.col, cex = ax.name.size)
      }
    }
  }
}
plot.4 <- function(y1.ster, y2.ster, x1.ster, x2.ster,
                   marker.mat, line.name = NULL, offset = NULL, adjust = NULL,
                   axis.col = "black", ax.name.col = "black", ax.name.size = 0.65) {
  if (y1.ster > usr[4] & y2.ster > usr[3]) {
    lines(c(x2.ster, usr[2]), c(usr[4], y2.ster),
          col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                          3] > 0)
          mtext(text = line.name, side = 3, line = 0 +
                  offset[3], adj = adjust[3], at = x2.ster,
                col = ax.name.col, cex = ax.name.size)
        else mtext(text = line.name, side = 4, line = 0 +
                     offset[4], adj = adjust[4], at = y2.ster,
                   col = ax.name.col, cex = ax.name.size)
      }
    }
  }
  if (y1.ster < usr[4] & y2.ster > usr[3]) {
    lines(c(usr[1], usr[2]), c(y1.ster, y2.ster),
          col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                          3] > 0)
          mtext(text = line.name, side = 2, line = 0 +
```

```
                offset[2], adj = adjust[2], at = y1.ster,
              col = ax.name.col, cex = ax.name.size)
        else mtext(text = line.name, side = 4, line = 0 +
                   offset[4], adj = adjust[4], at = y2.ster,
                 col = ax.name.col, cex = ax.name.size)
      }
    }
  }
  if (y1.ster > usr[4] & y2.ster < usr[3]) {
    lines(c(x2.ster, x1.ster), c(usr[4], usr[3]),
          col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                          3] > 0) {
          mtext(text = line.name, side = 3, line = 0 +
                  offset[3], adj = adjust[3], at = x2.ster,
                col = ax.name.col, cex = ax.name.size)
        }
        else mtext(text = line.name, side = 1, line = 0 +
                     offset[1], adj = adjust[1], at = x1.ster,
                   col = ax.name.col, cex = ax.name.size)
      }
    }
  }
  if (y1.ster < usr[4] & y2.ster < usr[3]) {
    lines(c(usr[1], x1.ster), c(y1.ster, usr[3]),
          col = axis.col)
    if (!is.null(line.name)) {
      if (ncol(marker.mat) == 3) {
        if (marker.mat[1, 3] - marker.mat[nrow(marker.mat),
                                          3] > 0)
          mtext(text = line.name, side = 2, line = 0 +
                  offset[2], adj = adjust[2], at = y1.ster,
                col = ax.name.col, cex = ax.name.size)
        else mtext(text = line.name, side = 1, line = 0 +
                     offset[1], adj = adjust[1], at = x1.ster,
                   col = ax.name.col, cex = ax.name.size)
      }
    }
  }
}
plot.5 <- function(marker.mat, axis.col = 1) {
  abline(h = marker.mat[1, 2], col = axis.col)
  abline(v = marker.mat[1, 1], col = axis.col)
}
switch(type.line, plot.1 = plot.1(marker.mat = marker.mat,
                                  line.name = line.name, offset = offset, adjust = adjust,
                                  axis.col = axis.col, ax.name.col = ax.name.col, ax.name.size = ax.name.size),


       plot.2 = plot.2(marker.mat = marker.mat, line.name = line.name,
```

```
                offset = offset, adjust = adjust, axis.col = axis.col,
                ax.name.col = ax.name.col, ax.name.size = ax.name.size),

      plot.3 = plot.3(marker.mat = marker.mat, line.name = line.name,
                offset = offset, y1.ster = y1.ster, y2.ster = y2.ster,
                x1.ster = x1.ster, x2.ster = x2.ster, adjust = adjust,
                axis.col = axis.col, ax.name.col = ax.name.col,
                ax.name.size = ax.name.size), plot.4 = plot.4(marker.mat = marker.mat,
                                        line.name = line.name, offset = offset, y1.ster = y1.ster,
                                        y2.ster = y2.ster, x1.ster = x1.ster, x2.ster = x2.ster,
                                        adjust = adjust, axis.col = axis.col, ax.name.col = ax.name.col,
                                        ax.name.size = ax.name.size), plot.5 = plot.5(marker.mat = marker.mat),
      axis.col = axis.col)
  list(gradient = gradient, intercept = intercept)
}

  .axes.plot <- function(ax, z.axes, z.axes.names, predictions.sample,
                predictions.mean,
                Z, Z.means.mat, p, ax.col, ax.name.size,
                markers, label, label.size, marker.size, offset, pos,
                strepie, ort.lty) {

  if (is.null(ax)) axes <- NULL else axes <- (1:p)[ax]

  predictions <- NULL
  if (!is.null(predictions.sample)) {
    predictions <- data.frame(matrix(NA, nrow = length(axes), ncol = length(predictions.sample)))
    dimnames(predictions) <- list(1:nrow(predictions), paste("s", predictions.sample, sep = ""))
  }

  if (!is.null(predictions.mean)) {
    predictions <- data.frame(matrix(NA, nrow = length(axes), ncol = length(predictions.mean)))
    dimnames(predictions) <- list(1:nrow(predictions), paste("m", predictions.mean, sep = ""))
  }

  r.names <- NULL
  for (i in axes) {
    marker.mat <- z.axes[[i]][z.axes[[i]][, 4] == 1, 1:3]
    if (is.null(nrow(marker.mat)))
      stop(paste("Increase n.int for axis ", i))
    x.vals <- marker.mat[, 1]
    y.vals <- marker.mat[, 2]
    if (is.null(z.axes.names)) {
      axis.name <- paste("v", i, sep = "")
    }      else {axis.name <- z.axes.names[i]}
    r.names[i] <- axis.name
    x.invals <- x.vals[x.vals < usr[2] & x.vals > usr[1] &
                y.vals < usr[4] & y.vals > usr[3]]
    y.invals <- y.vals[x.vals < usr[2] & x.vals > usr[1] &
                y.vals < usr[4] & y.vals > usr[3]]
    tick.labels <- marker.mat[x.vals < usr[2] & x.vals > usr[1]
```

```
                    & y.vals < usr[4] & y.vals > usr[3], 3]
    if (length(x.invals) < 2)
      warning(paste("Less than 2 markers on axis ",
                    i, ". Increase n.int."))

    uit <- .draw.axis(marker.mat, line.name = axis.name,
                      ax.name.size = ax.name.size, axis.col = ax.col$ax.col[i],
                      ax.name.col = ax.col$ax.name.col[i], offset = offset,
                      pos = pos)

    if (!is.null(predictions.sample)) {
      for (ss in predictions.sample) {
        predictions[i, paste("s", ss, sep = "")] <- round(.orthog.pred.line(x1 = x.invals[1],
                                                    y1 = y.invals[1], x2 = x.invals[length(x.invals)],
                                                    y2 = y.invals[length(y.invals)], val1 = tick.labels[1],
                                                    val2 = tick.labels[length(x.invals)], px = Z[ss,  1], py = Z[ss, 2], ort.lty = ort.lty), digits =
4)
      }
    }
    if (!is.null(predictions.mean)) {
      for (ss in predictions.mean) {
        predictions[i, paste("m", ss, sep = "")] <- round(.orthog.pred.line(x1 = x.invals[1],
                                                    y1 = y.invals[1], x2 = x.invals[length(x.invals)],
                                                    y2 = y.invals[length(y.invals)], val1 = tick.labels[1],
                                                    val2 = tick.labels[length(x.invals)], px = Z.means.mat[ss, 1], py = Z.means.mat[ss, 2],
ort.lty = ort.lty), digits = 4)
      }
    }
    gradient <- uit$gradient
    for (j in 1:length(x.invals)) .draw.marker(x = x.invals[j], y = y.invals[j], grad = -1/gradient, expand = strepie[1],
                                    both.sides = TRUE, col = ax.col$tickmarker.col[i])
    if (markers == TRUE) {
      x.labvals <- x.invals
      y.labvals <- y.invals
      tick.labels <- tick.labels
    }
    else {
      x.labvals <- x.invals[c(1, length(x.invals))]
      y.labvals <- y.invals[c(1, length(y.invals))]
      tick.labels <- tick.labels[c(1, length(tick.labels))]
    }
    for (j in 1:length(x.labvals)) {
      #print(tick.labels[j])
      .marker.value(x = x.labvals[j],  y = y.labvals[j], grad = -1/gradient
                    , mark = tick.labels[j], expand = strepie[2], marker.size = marker.size
                    , col = ax.col$marker.col[i])
    }
  }

  if (is.null(predictions.sample) & is.null(predictions.mean))
    predictions <- NULL
```

```
  if (!is.null(predictions)) {
    predictions <- na.omit(predictions)
    predictions <- as.matrix(predictions)
    dimnames(predictions)[[1]] <- r.names[!is.na(r.names)]
  }
  return(predictions)
} # end  .axes.plot

.trajectories.plot <- function(z.axes, z.axes.names, p, ax, pos) {
  if (is.null(ax)) axes <- NULL else axes <- (1:p)[ax]
  if (!is.element(pos[1], c("Hor", "Orthog", "Paral")))
    stop("Argument pos must be one of 'Hor','Orthog' or 'Paral' ")

  if (pos[1] == "Hor") {
    par(las = 1)
    adjust <- c(0.5, 1, 0.5, 0)
  }
  if (pos[1] == "Orthog") {
    par(las = 2)
    adjust <- c(0, 0, 0, 0)
  }
  if (pos[1] == "Paral") {
    par(las = 0)
    adjust <- c(0.5, 0.5, 0.5, 0.5)
  }
  axis.outofbounds <- rep(F, p)
  for (i in axes) {
    if (is.null(z.axes.names)) {axis.name <- paste("v", i, sep = "")}  else {   axis.name <- z.axes.names[i]}
    lines(z.axes[[i]][, 1], z.axes[[i]][, 2])

    tick.pos <- (1:nrow(z.axes[[i]]))[z.axes[[i]][, 4] == 1]
    tick.pos.voor <- tick.pos - 1
    tick.pos.voor[tick.pos.voor < 1] <- 1
    tick.pos.na <- tick.pos + 1
    tick.pos.na[tick.pos.na > nrow(z.axes[[i]])] <- nrow(z.axes[[i]])

    gradient <- (z.axes[[i]][tick.pos.na, 2] - z.axes[[i]][tick.pos.voor, 2])/(z.axes[[i]][tick.pos.na, 1] - z.axes[[i]][tick.pos.voor,1])
    gradient[is.na(gradient)] <- 0
    marker.mat <- z.axes[[i]][tick.pos, 1:3, drop = FALSE]
    x.vals <- marker.mat[, 1]
    y.vals <- marker.mat[, 2]
    x.invals <- x.vals[x.vals < usr[2] & x.vals > usr[1] &
                       y.vals < usr[4] & y.vals > usr[3]]
    y.invals <- y.vals[x.vals < usr[2] & x.vals > usr[1] &
                       y.vals < usr[4] & y.vals > usr[3]]
    tick.labels <- marker.mat[x.vals < usr[2] & x.vals >
                          usr[1] & y.vals < usr[4] & y.vals > usr[3], 3]
    gradient <- gradient[x.vals < usr[2] & x.vals > usr[1] &
                         y.vals < usr[4] & y.vals > usr[3]]
    if (length(gradient) == 0) axis.outofbounds[i] <- T else {
      for (j in 1:length(x.invals)) {
```

```
        .draw.marker(x.invals[j], y.invals[j], -1/gradient[j])
        pos1 <- 1
        if ((-1/gradient[j] > -1) & (-1/gradient[j]) <
            1)
          pos1 <- 4
        .marker.value(x.invals[j], y.invals[j], gradient[j],
                      tick.labels[j], marker.size = marker.size,
                      pos = pos1)
      }
      pos2 <- 2
      if (gradient[length(x.invals)] > -1 & gradient[length(x.invals)] <
          1)
        pos2 <- 3
      text(x.invals[length(x.invals)], y.invals[length(x.invals)],
           axis.name, pos = pos2, cex = ax.name.size)
    }
  }
  write.line <- usr[3] + 0.05 * (usr[4] - usr[3])
  for (i in 1:p) if ((axis.outofbounds[i])) {
    if (is.null(zoomval)) {
      text(usr[1], write.line, paste(paste("Axis", i), "out of bounds"), pos = 4)
      write.line <- write.line + 0.05 * (usr[4] - usr[3])
    }
  }
} # end .trajectories.plot

.samples.plot <- function(Z, pch.samples.size, pch.samples, Z.means.mat, specify.classes, label, label.size, class.vec) {
  if (ncol(Z) == 2)
    Z <- cbind(Z, pch.samples)
  if (ncol(Z) == 3)
    Z <- cbind(Z, 1)
  if (ncol(Z) == 4)
    Z <- cbind(Z, 1)
  if (ncol(Z) == 5)
    Z <- cbind(Z, pch.samples.size)
  if (is.null(dimnames(Z)[[1]]))
    dimnames(Z) <- list(paste("s", 1:nrow(Z)), NULL)
  classes <- unique(Z[, 4])
  class.vec <- Z[, 4]
  legend.labs <- classes
  for (j in classes) {
    Z.class <- Z[class.vec == j, , drop = FALSE]
    if (label == TRUE)
      text(Z.class[, 1], Z.class[, 2] - 0.015 * (usr[4] - usr[3]), labels = dimnames(Z.class)[[1]], cex = label.size)
    Z.class <- data.frame(Z.class[, 1:2], pch.samples.samp = Z.class[1, 3], colr = (Z.class[1, 4]), lty = Z.class[1, 5], pch.samples.size =
Z.class[1, 6], stringsAsFactors = FALSE)
    #if (!is.null(specify.classes)) # always make the plots: unless the functions are not called

      for (i in 1:nrow(Z.class)) points(x = Z.class[i, 1], y = Z.class[i, 2], pch = Z.class[i, 3], col = Z.class[i, 4], cex = Z.class[i, 6])

  }
```

```
} #end .samples.plot

#######################

if(T) {
  par(pty = "s", mar = parplotmar)
  if (alpha < 0 | alpha > 0.99)
    stop(message = "alpha not to be negative or larger than 0.99")
  alpha.entered <- alpha
  alpha <- round(alpha, digits = 2)
  if (abs(alpha.entered - alpha) > 0)
    cat("alpha has been rounded to ", alpha, "\n")
  p <- length(z.axes)
  alpha1 <- 100 * alpha
  if (is.null(zoomval))
    plot(Z[, 1] * exp.factor, Z[, 2] * exp.factor, xlim = range(Z[, 1] * exp.factor), ylim = range(Z[, 2] * exp.factor),
         xaxt = "n", yaxt = "n", xlab = "", ylab = "", type = "n",
         xaxs = "i", yaxs = "i", asp = 1)
  if (is.numeric(zoomval))
    plot(Z[, 1] * exp.factor, Z[, 2] * exp.factor, xlim = zoomval[1:2], ylim = zoomval[3:4],
         xaxt = "n", yaxt = "n", xlab = "",
         ylab = "", type = "n", xaxs = "i", yaxs = "i", asp = 1)
  usr <- par("usr")
  legend.lab.bags <- NULL
  legend.labs <- unique(dimnames(Z.means.mat)[[1]])
  predictions <- NULL
}

# calc x (n dim) value of every pixel
X_grid_out <- X_grid_function_genbi(m = 400, R = R, p1 = p1, p2 = p2, categorical = categorical
                                    , sd = sds, gem = gems, CLPs = cate.med.points
                                    , Xmat = Xmat)

# add the colour grid before data points, axes and CLPs
if (draw.pred.area == T ) {

  prob_grid_out <- prob_grid_function_genbi(model.type = model.type, Model_Name = Model_Name
                                    , p1 = p1, p2 = p2, Dummy_Model = Dummy_Model, categorical = categorical
                                    , col.vec = prob.colour, X_grid_out$Xgrid)

  points(X_grid_out$Zgrid, col = prob_grid_out$col.value, cex=0.1, pch=16)
  box(which = "plot", lty = "solid")
}

# Add the boundary line
if (draw.pred.boundary == T ) {

  prob_boundary_out <- prob_boundary_function_genbi(model.type = model.type, Model_Name = Model_Name
                                    , p1 = p1, p2 = p2, Dummy_Model = Dummy_Model, categorical = categorical
                                    , X_grid_out$Xgrid
                                    , boundary_val = boundary_val, boundary_margin = boundary_margin)
```

```
  points(X_grid_out$Zgrid[prob_boundary_out$bound.grid,], col = "black", cex=0.1, pch=16)
  box(which = "plot", lty = "solid")
}

if (class.bound.plot == T & cate.plot.region == T & all.class.bound == F & draw.pred.area == T)
  print("Categorical regions will not be shown: otherwise override the prediction scores")

## plot categorical prediction boundaries
if (class.bound.plot == T) {

  if(all.class.bound == F) {
    cate_bound_out <- cate_class_boundary(cate.plot, categorical
                                          , p2, Xmat, X_grid_out)

    for (i in 1:cate_bound_out$cat.levels[1]) {
      region <- NULL
      # cate.plot.region == T then colour in the regions
      if(cate.plot.region == T & draw.pred.area == F) region <- cate.colour[i]
      polygon(cate_bound_out$cat.list.x[[i]], cate_bound_out$cat.list.y[[i]]
              , border = "black", col = region, lwd = 1, lty = 1)
    }
  }

  if(all.class.bound == T) {
    if(cate.plot.region == T & all.class.bound == T)
       print("Cannot plot level regions of all categories")
    for (j in 1:p2) {
      cate_bound_out <- cate_class_boundary(j, categorical
                                          , p2, Xmat, X_grid_out)

      for (i in 1:cate_bound_out$cat.levels[1]) {
        polygon(cate_bound_out$cat.list.x[[i]], cate_bound_out$cat.list.y[[i]]
                , border = "black", col = NULL, lwd = 1, lty = 1)
      }
    }
  }
  box(which = "plot", lty = "solid")
}

if (!is.null(specify.data))
  .samples.plot(Z = Z, pch.samples.size = pch.samples.size,
                pch.samples = pch.samples, Z.means.mat = Z.means.mat,
                specify.classes = specify.classes, label = label,
                label.size = label.size, class.vec = class.vec)

# plot axes after the samples
if (straight)
  predictions <- .axes.plot(Z = Z, Z.means.mat = Z.means.mat,
                            z.axes = z.axes, z.axes.names = z.axes.names, p = p,
                            ax = ax, ax.name.size = ax.name.size, ax.col = ax.col,
```

```
                    markers = markers, label = label, label.size = label.size,
                    marker.size = marker.size, offset = offset, pos = pos,
                    strepie = strepie, predictions.sample = predictions.sample,
                    predictions.mean = predictions.mean, ort.lty = ort.lty
  )

 # plot category classes medoids
 if(cate.med.plot & (p2 > 0)) {

  if(all.class.bound == F) {
   points(cate.med.points[cate.med.points[,4] == cate.plot, 1:2], pch = 1 , cex = 2.4,  lwd = 1.5, col =
cate.colour[cate.med.points[cate.med.points[,4] == cate.plot, 4]] )
   text(cate.med.points[cate.med.points[,4] == cate.plot, 1:2], labels = cate.med.points[cate.med.points[,4] == cate.plot, 6], cex = 0.8,
col = "black" )
  }

  if(all.class.bound == T) {
   points(cate.med.points[,1:2], pch = 1 , cex = 2.4, lwd = 1.5, col = cate.colour[cate.med.points[,4]] )
   text(cate.med.points[,1:2], labels = cate.med.points[,6], cex = 0.8, col = cate.colour[cate.med.points[,4]] )
  }
 }

 # plot Legend
 if(cate.legend & (p2 > 0) & cate.med.plot)
  legend(x = "bottomleft", legend = cate.names, col = cate.colour[1:p2], fill = cate.colour[1:p2], cex=0.5 )

 # Title
 title(main = (ifelse(is.null(Title), paste("Generalised Biplot"), Title)))

 list(usr = usr, X_grid_out = X_grid_out$Xgrid)
}
```

31 Xgrid function.R

```
X_grid_function_genbi <- function(m = 400, R, p1, p2, categorical, sd, gem, CLPs, Xmat) {
 library(pdist)
 # Determine plot boundaries, in units of the data
 mm <- m*m
 minx <- par("usr")[1]
 miny <- par("usr")[3]
 maxx <- par("usr")[2]
 maxy <- par("usr")[4]

 xseq <- seq(from = minx, to = maxx, length.out = m)
 yseq <- seq(from = maxy, to = miny, length.out = m)
 Zgrid <- as.matrix(expand.grid(xseq,yseq))

 ## Predict continuous variables:
 predict.cont <- (Zgrid %*% R)
 predict.cont <- sweep(predict.cont, 2, sd, FUN = "*") # the sd and gems are already set to correctly for ordinal, where ordinal is used
 predict.cont <- sweep(predict.cont, 2, gem, FUN = "+")
```

```
  predict.cont <- round(predict.cont, 1)
  colnames(predict.cont) <- dimnames(Xmat)[[2]][!categorical]

  ## Predict categorical variables
  predict.cat <- matrix(0, ncol = p2, nrow = mm )
  colnames(predict.cat) <- dimnames(Xmat)[[2]][categorical]
  predict.cat <- as.data.frame(predict.cat)

  predict.class <- matrix(0, ncol = p2, nrow = mm )
  colnames(predict.class) <- dimnames(Xmat)[[2]][categorical]
  predict.class <- as.data.frame(predict.cat)

  if(p2 > 0)
  for (k in 1:p2){
    CLPs.k <- CLPs[CLPs[, 4] == k, 1:2]
    cat.names <- levels(Xmat[, categorical, drop = F][, k])  # get the names of each level in a category
    cat.levels <- length(cat.names)

    point.dist.matrix <- as.matrix(pdist(Zgrid[, 1:2 ], CLPs.k[, 1:2]), nrow = mm, ncol = cat.levels)
    point.classify <- apply(point.dist.matrix,1,which.min)
    predict.class[, k] <-  point.classify
    predict.cat[, k] <-  cat.names[point.classify]
  }

  ## Combine two sets of predictions
  Xgrid <- cbind(predict.cont,  predict.cat)

  list(Zgrid = Zgrid, Xgrid = Xgrid, predict.class = predict.class ) #predict.cat = predict.cat)
}
```

32 Cate_class_boundary.R

```
cate_class_boundary <- function(cate_plot = 1, categorical
                                , p2, Xmat, X_grid_out) {

  # Select a categorical variable to plot: cate_plot
  cat.names <- levels(Xmat[, categorical, drop = F][, cate_plot])
  cat.levels <- length(cat.names)
  cat.list.x <- vector("list", cat.levels)
  cat.list.y <- vector("list", cat.levels)

  # for each class in the selected cate_plot, get the hull boundaries
  for (class in 1:cat.levels) {
    xh <- X_grid_out$Zgrid[X_grid_out$predict.class[, cate_plot] == class, 1]
    yh <- X_grid_out$Zgrid[X_grid_out$predict.class[, cate_plot] == class, 2]

    flush.console()
    hull <- chull(xh, yh) # the pixels row nr that form the hull
    cat.list.x[[class]] <- xh[hull]
    cat.list.y[[class]] <- yh[hull]
```

```
  }
  list(cat.list.x = cat.list.x, cat.list.y = cat.list.y, cat.levels = cat.levels)
}
```

32 prob boundary grid.R

```
prob_boundary_function_genbi <- function(model.type = "glm", Model_Name , p1, p2
                                , Dummy_Model, categorical
                                , Xgrid, boundary_val = 0.5, boundary_margin = 0.005) {

        ## Combine two sets of predictions
        if (p2 > 0)  {
          dummy.output <- predict(Dummy_Model, newdata = Xgrid[, categorical, drop = F])
          Xgrid01 <-cbind(Xgrid[, !categorical, drop = F], dummy.output)
        }
        if (p2 == 0) Xgrid01 <- Xgrid[, 1:p1] #class is matrix: class(Xgrid01)

        ## predict the score between 0 and 1 for each pixel
          grid.prob <- predict(Model_Name,  Xgrid01, type="prob")[ ,2]

        # find where it is between the boundary values
          bound.grid <- (grid.prob > boundary_val - boundary_margin) & (grid.prob < boundary_val + boundary_margin)

        list(bound.grid = bound.grid)
}
```

32 prob grid function for Gen Biplot.R

```
prob_grid_function_genbi <- function(model.type = "glm", Model_Name , p1, p2
                                , Dummy_Model, categorical
                                , col.vec, Xgrid) {

        ## Probability colour scheme
        colbands <- 100
        ## Combine two sets of predictions
        # The convert columns of Categories into the 0/1 values as used in the prediction models
        if (p2 > 0)  {
          dummy.output <- predict(Dummy_Model, newdata = Xgrid[, categorical, drop = F])
          Xgrid01 <-cbind(Xgrid[, !categorical, drop = F], dummy.output)
        }
        if (p2 == 0) Xgrid01 <- Xgrid[, 1:p1] #class is matrix: class(Xgrid01)

        ## predict the score between 0 and 1 for each pixel
        grid.prob <- predict(Model_Name,  Xgrid01, type="prob")[ ,2]
        col.value <- col.vec[floor(grid.prob*colbands)+1] #col.vec[floor(grid.prob[,model.cls]*100)+1]

        list(col.value = col.value)
}
```