A Machine Learning Model for Octane Number Prediction

Victor Spencer

Thesis presented for the degree of Master of Science in the Department of Statistical Sciences at the University of Cape Town.

Supervised by Prof. Klaus Möller and Dr Juwa Nyirenda

April 1, 2022

Plagiarism Declaration

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 scientific convention for citation and referencing. Each significant contribution to, and quotation in this thesis from the

   work(s) of other people has been attributed, cited and and referenced.
3. This thesis in my own work.
4. I have not allowed, and will not allow anyone to copy my work with the intention of passing it of as his or her own work.

Signature:

Signed by candidate

April 1, 2022

Abstract

Assessing the quality of gasoline blends in blending circuits is an important task in quality control. Gasoline quality however , cannot be measured directly on a process stream. Therefore a quality indicator which can be determined from the stream composition is required. Various quality indicators have been used in the existing body of literature but the indicator in this study will be the Research Octane Number (RON). This is an indicator which measures the ignition of gasoline relative to pure octane (Abdul-Gani et al. 2018). Previous research has used empirical models in the form of phenomeno-logical and machine learning models (González 2019). Phenomeno-logical models have been used in the past as a way of programming an engineer's thought process in the form of differential equations put together. Machine learning models are data driven with primarily regression and deep learning methods being used in literature as prediction models.

This study aims to develop a parsimonious machine learning model which can be used to predict the RON from the molar composition of the gasoline product stream. Regression, ensemble learning and Artificial Neural Networks (ANN) will be used specifically in this study. The ensemble learning models which will be trained are Bayesian Additive Regression Trees (BART) and Gradient Boosting Machines (GBM). The raw data will be scraped from multiple journals online and the data frame will be comprised of volume compositions of the reference compounds and the RON of each blend. The existing data frame will be extended to include the molar composition of the structural groups present in each of the blends. The structural groups which may be referred to as functional groups are specific substituents within molecules which may be responsible for the characteristic chemical reactions of the respective molecules. This addition of structural groups adds a layer of information to differentiate between blends with different compound compositions but similar RON.

It was hypothesised that the molar compositions of the additives and their substituent structural groups would rank highest and the molar composition of n-heptane would have the lowest ranking. For the Multiple Linear Regression (MLR) models, two cases were trained; one with interaction parameters and another without. Both of these cases were trained with and without the composition constraints on the compound compositions. For the ensemble learning case, a BART model with 200 trees and a GBM model with 1998 trees were trained. Four Single Layer Feed-forward Neural Network (SLFN) models were trained, each with 3, 5, 10 and 15 nodes. The choice of neural network architecture was made because the data frame was small, with only 12 input variables and 350 observations. Prior to training the models, an Explanatory Data Analysis was carried out to assess the potential dimensionality reduction, correlations and outliers.

The final regression model was the interaction model with a test MSE of 7.54 and an adjusted R^2 of 0.986. The BART model obtained a test MSE of 13.74 and an adjusted R^2 of 0.983. The GBM model had a test MSE of 38.12 and an adjusted R^2 of 0.917. Lastly the best performing ANN was the 10 node SLFN which obtained a test MSE of 11.26 and an adjusted R^2 of 0.969. For each model, a variable importance was carried out and it was observed that the molar composition of n-Heptane consistently ranked high in the variable importance. In addition to these predictive statistics; the parity plots, residual plots and Analysis of Variance (ANOVA) were analysed and taken into consideration in evaluating the performance of each of the models trained.

It was concluded that the MLR model performed best followed by the BART model. The ANN models ranked third and the GBM model ranked last. The hypothesis that the molar compositions of the additives and their substituent structural groups would rank highest and

n-heptane would be the lowest ranking component was disproved as the molar composition of n-heptane and its substituent structural groups consistently ranked high .

The recommendation for this study is to train the models with a more representative data set in future and to use a hybrid model which comprises of a phenomeno-logical model and a machine learning model for best results and to reduce the bias of the model in the regions with few data points. With the next step of the study being the integration of the new model into the plant-wide Advanced Process Control (APC).

Contents

Acknowledgements

I would like to express my deepest appreciation to my supervisors Prof Klaus Möller and Dr Juwa Nyirenda whose charisma and passion for numerical modelling and machine learning inspired my enthusiasm in their classes. I am grateful for their assistance and guidance in making this project enjoyable and fruitful in my pursuit to understand machine learning.

I would like to extend my sincere thanks to my mother Sharon and my wife Anotida, without their support, I would not have completed this dissertation. Their consistent encouragement in times of discouragement was instrumental in keeping my passion for this topic alive.

Glossary

bootstrap Any test or metric that uses random sampling with replacement, and falls under the broader class of resampling methods. 49, 50

cetane number a quantity indicating the ignition properties of diesel fuel relative to cetane as a standard. 16

correlation Statistical relationship which may or may not be causal between two random variables. Commonly used to refer to the degree to which a pair of variables are linearly related. 26

cross validation A resampling procedure used to evaluate machine learning models on a limited data sample. 45, 46, 62, f

four-stroke cycle A four-stroke cycle engine is an internal combustion engine that utilizes four distinct piston strokes (intake, compression, power, and exhaust) to complete one operating cycle. The piston make two complete passes in the cylinder to complete one operating cycle. 6

functional group See structural group. 6

generalization The ability of a model to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model. 59

heteroskedasticity When the standard deviations of a predicted variable, monitored over different values of an independent variable are non-constant. 44, 68, 73

homoskedasticity When the standard deviations of a predicted variable, monitored over different values of an independent variable are constant. 43, 56, 73, 75

outlier a point which falls more than 1.5 times the interquartile range above the third quartile or below the first quartile. ii, vi, 1, 14, 17, 22, 26, 31, 32, 49, 72, ap

over-fitting the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably. 47, 52, 62, 70, n

parsimonious The simplest model/theory with the least assumptions and variables but with greatest explanatory power. ii, 50, 56, 68

phenomeno-logical The thought process of a scientist or engineer which is derived from observable scientific phenomenon and can be programmed as a group of differential equations. ii, iii, 4, 7

regularization Any modification made to a learning algorithm that is intended to reduce its generalization error but not its training error. 70, f, n

structural group Recognizable groups of bound atoms in organic compounds. ii, 2, 41, 42, 61, 62, 74

weak learner a model that can be used as a building block for designing more complex models by combining several of them. A weak learner performs poorly on its own due to a high bias resulting from under-fitting. 56, e, f, g

List of Abbreviations

AI Artificial Intelligence. xi, 4, 5, 19, k

ANN Artificial Neural Networks. ii, 1, 4, 6, 13, 15, 16, 17, 18, 20, 26, 58, 60, 61, 62, 64, 70, 73, l, m, n

ANOVA Analysis of Variance. ii, vi, xii, 38, 39, 47, 53, 54, 63, 64, 65, 71, aw, be

APC Advanced Process Control. iii, 2, 20, 21, 22, 75

Back-fitting MCMC Bayesian Back-Fitting Monte Carlo Markov Chains. e, i

BART Bayesian Additive Regression Trees. ii, vi, xi, xii, 26, 45, 46, 47, 48, 49, 50, 51, 53, 56, 57, 70, 71, 72, 73, 74, e, f, g, h, bh

BART-MCMC Bayesian Additive Regression Trees with Monte Carlo Markov Chains. h

BPNN Back Propagation Neural Networks. 18, s

CN Cetane Number. 16, 17

DAE De-noising Auto-Encoder. 18

EDA Exploratory Data Analysis. vi, 2, 3, 12, 14, 24, 25, 26, 27, an

ELM Extreme Learning Machine. 19, 62, o, p, r

ELM-RBF Extreme Learning Machine with Radial Basis Function. 19

GBM Gradient Boosting Machines. ii, vi, xi, xii, 26, 52, 53, 54, 55, 56, 57, 70, 71, 72, 73, 74, bk

IDT Ignition Delay Time. xiii, 5, 8, 20

IID Independent and Identically Distributed. 44, 50, 56, 68, 73, 75, g

MAE Mean Absolute Error. 47, 53

MCMC Monte Carlo Markov Chains. 45, 47, 49, 50

MH Metropolis Hastings. i

MLFN Multiple Layer Feed-forward Neural Network. xiv, 59, 60, 61, 62, 64, o, s

MLR Multiple Linear Regression. ii, vi, xii, 13, 26, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 70, 71, 72, 73, 74, 75, a, ar, au, aw, az, bc, be

MON Motor Octane Number. 15, 17

MSE Mean Square Error. ii, 15, 18, 37, 45, 52, 60, 61, 70, 71, t

NIR Near Infrared Resonance. 14, 18, 19, 21, 47

NMR Nuclear Magnetic Resonance. 15, 16, 17, 21, 59, 60, 62

ORACL **O**xygen, **R**ings, **A**romatics, aliphatic **C**hains and O**L**efins. 15, 59, 60

OS-ELM Online Sequential Extreme Learning Machine. 19, p, q

PCA Principal Component Analysis. 11, 12

PCR Principal Component Regression. 13

PIANO Paraffines, Iso-Paraffins, Aromatics, naphthenes, Olefins. 12, 13, 15, 23, 60, 61, 62

PPR Projection Pursuit Regression. 13, 14, 15, 20

QSPR Quantitative Structure-Property Relationship. 10, 16, 17, 61

RMSE Root Mean Square Error. 13, 15, 19, 45, 46, 59, 70, r

RON Research Octane Number. ii, xi, xiii, 1, 2, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 31, 33, 34, 36, 38, 42, 43, 44, 45, 47, 48, 49, 50, 53, 55, 56, 60, 61, 62, 65, 66, 67, 68, 71, 72, 73, 74, 75

SaDE-ELM Self Adaptive Evolutionary Extreme Learning Machine. 19, 62, 63, r

SDAE Stacked De-noising Auto-Encoder. 18

SDAE-BP Stacked De-noising Auto-Encoder with Back Propagation. 18, 19

SLFN Single Layer Feed-forward Neural Network. ii, vi, xi, xii, xiv, 19, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, l, m, o, p, bo, bt, by, cd

TanH Hyperbolic Tangent Activation Function. 58

List of Figures

List of Tables

List of Equations

1 Introduction

1.1 Subject and Motivation of Study

This report details the investigation made into developing a model which can be used to predict the Research Octane Number (RON) of different blends of gasoline given the composition of compounds in the gasoline blends. RON is a variant of the octane number which is simulated at low severity. The result of the study will be a machine learning model with the least amount of error in predicting the RON from the composition of each gasoline blend. Machine learning is a good approach to octane number modelling because the models can be independent of an engineer's bias in modelling different ranges of octane number and machine learning also allows for data manipulations to make models simpler and less computationally expensive.

1.2 Background to Study

Research Octane Number is a measure of the ignition quality of a petrol blend and the most significant quality indicator. RON measures the ignition quality of a petrol blend and the resistivity of the blend to knocking. Knocking is the phenomena by which auto-ignition of the fuel due to hot spots, compression heating, and wrong chemical mixture result in an incorrect and faster reaction path.

With more policies being implemented regarding vehicle emissions, it is important to ensure the quality of the fuel is known before it is sent to the market. Having said this, refineries would need to know how to mix the fuel components available in a cost effective manner without jeopardising the quality of the fuel. There have been several models which have been used to perform this task, ranging from parametric and quadratic forms to machine learning methods. The RON shall be predicted using three different types of models; regression, ensemble learning and Artificial Neural Networks (ANN). The models will be built on data sourced from different academic journals. The Research Octane Number is a physical property which cannot be measured directly but can be determined in a similar fashion to flash point prediction using a thermodynamic equation of state.

1.3 Hypotheses

The proposed hypothesis is that the molar composition of the additives and their respective structural groups will rank highest and the molar composition of n-heptane would rank the lowest in terms of variable importance.

1.4 Objective of Study

The objectives of this investigation are:

- Determine molar composition of structural groups from the composition of compounds in each of the blends.
- Explore data properties for potential dimension reduction and perform outlier analysis.
- Determine the order of significance of each structural group and compound.

- Develop models for predicting the RON from the structural group and compound compositions.
- Compare the results of each model using relevant error metrics and model diagnostics and discuss causal effect of structural groups on the RON.
- Determine either the best performing model or a combination of the models which form the best performing single model.
- Consider integration of final model to the plant-wide Advanced Process Control (APC).

1.5 Scope and Limitation of the Study

The molar composition of the compounds and their substituent structural group composition in each of the blends will be used as explanatory variables. Although there are more parameters which can be included in this investigation, these are omitted in this study. These include; vapor pressure, boiling point and other physical properties. The single limitation therefore is the use of only chemical properties as input variables. The second limitation may be presented in the error analysis of the models. Not all models have the same error metrics for assessing prediction accuracy. This may present a limitation in how the error metrics may be compared and interpreted on the same scale. Another limit of machine learning models is consistent feature analysis. Regression models are fairly simple to interpret causal effects of variables while ensemble learning models and artificial neural networks cannot be interpreted as easily.

1.6 Plan of Development

Chapter 1: Introduction

This chapter provides a background to the investigation. The objectives, scope and hypotheses are stated in this chapter.

Chapter 2: Literature Review

This chapter reviews the literature which is relevant to and inspires this investigation. In this section topics including importance of octane number and previous work attempting to model the octane number are reviewed.

Chapter 3: Research Motivation

In this chapter the gap analysis will be conducted to define how this investigation extends the body of knowledge. The aims and objectives will be outlined in this chapter as well. Finally the hypotheses will be listed.

Chapter 4: Methodology

This chapter details the experimental procedure taken to obtain the results of this investigation. The data and all pre-processing is described, followed by the Exploratory Data Analysis

(EDA) and all the models which will be built are listed.

Chapter 5: Results and Discussion

The models produced and their respective results are presented by means of plots and tables detailing the results obtained. The predictive statistics and model diagnostics will be produced and the results obtained for each model will be evaluated and compared to the other models trained and to the models trained in literature, using the error metrics and plots.

Chapter 7: Conclusions

Conclusions will be derived from the results and discussion and presented in this chapter.

Chapter 8: Recommendations

After deriving conclusions, ways of expanding and improving this investigation for future work will be presented in this chapter.

Chapter 9: Bibliography

The list of cited and non-cited material which was useful for this investigation will be listed in this chapter.

Chapter 10: Appendices

Any additional supplementary information used in this investigation will be presented in this chapter.

2 Literature Review

2.1 Artificial Intelligence in Chemical Engineering

Himmelblau 2000 wrote the earliest paper discussing the applications of Artificial Neural Networks (ANN) in chemical engineering. Himmelblau 2000 makes the conclusion that an ANN can provide good empirical models for complex non-linear processes. Octane number prediction is a complex non-linear prediction because the octane number does not change linearly with the explanatory variables. This phenomena makes neural networks a potential model for predicting the octane number. Himmelblau 2000 describes several uses of neural networks, one use being polymer quality prediction which would be similar to gasoline quality prediction. This study will be focusing on fuel quality prediction, particularly that of the Research Octane Number (RON) so his method may be extended on to be applicable to this problem.

This study will inform process control in the blending stage at a refinery. The composition of the blend will be run through a machine learning algorithm and the octane number predicted. Once the octane number is predicted, if it does not lie within the desired range, a signal is sent and the blend is changed. Himmelblau 2000 used a neural network to speed up the operation and control of reactors. The feedback was to the reactor to control the operating conditions but in this study, the feedback will be to the blending circuit.

Venkatasubramanian 2019 describes three phases which have been experienced in the implementation of Artificial Intelligence (AI) in chemical engineering and describes them as follows:

1. Expert systems era (1983 - 1995).

   This era involved programming the human algorithmic process. Multiple if statements and psychological mimicry were used in a top down design paradigm.

2. Neural Networks era (1990 - 2008).

   In this era, data was used to gain insights about the process, making model building easier and less complex. Venkatasubramanian 2019 argues that the biggest break through was development of a non-linear function approximation. This may be true for non linear machine learning methods but not the case in development of non-linear phenomeno-logical models as these have been used since the 1960's.

3. Deep Learning and Data Science (2008 - Present).

   Venkatasubramanian 2019 argues that process engineering has entered the era of predictive analytics, where the machine learning models used are predominantly of a deep learning nature. Deep learning algorithms differ from the neural networks used in era 2 in the sense that the neural networks have shifted from single to multi-layer networks in light of computers becoming more powerful.

To remain impactful in the current era of AI in process engineering, predictive models should be one of deep learning, reinforcement learning or statistical learning (Venkatasubramanian 2019). Of the methods relevant to the current era, deep neural networks are applicable to the problem being addressed in this study. Considering the descriptions given by Venkatasubramanian 2019, it may be concluded that Abdul-Gani et al. 2018, Tian, You, and X. Huang 2018 and Wang, Yang, and Kalivas 2020 are some of the researchers who have used era two models in their work.

Venkatasubramanian 2019 defines a process of automating a knowledge extraction framework for a catalyst design using an iterative cyclical methodology shown in Figure 2.1. Rather than using a linear approach which has been used traditionally as shown in Figure 2.2.

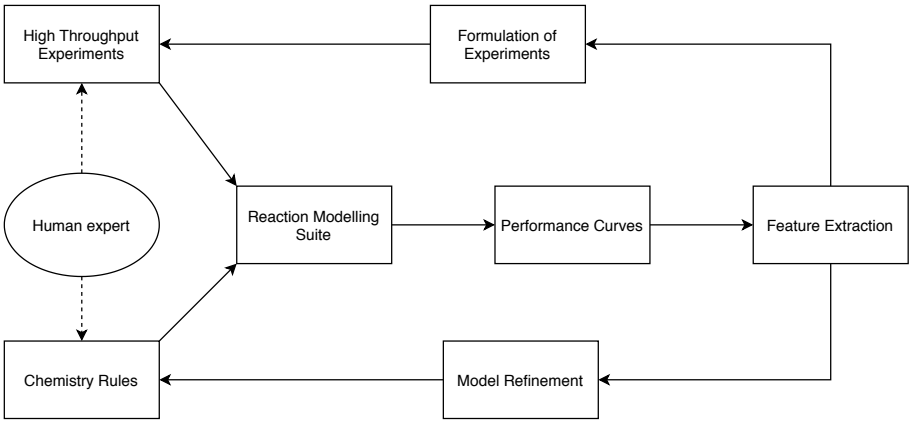


Figure 2.1: Iterative method of AI model design taken from Venkatasubramanian 2019

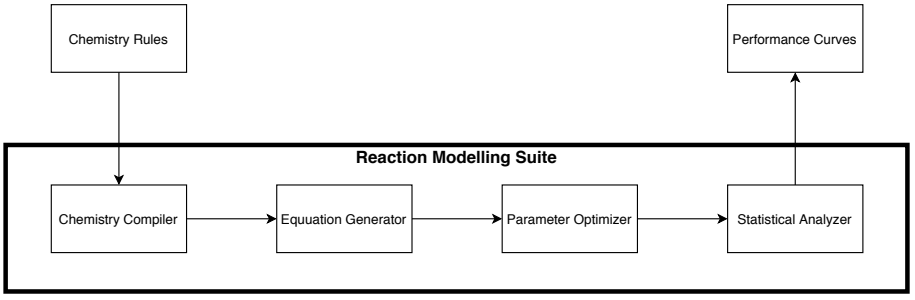


Figure 2.2: Straight line method of AI model design taken from Venkatasubramanian 2019

Although the approach shown in Figure 2.1 was developed for a catalyst framework, the principle will be the same for this study. In this study the rules and experiment results from known blending experiments will be combined in building the basis model which will be assessed using error metrics in place on performance curves. Given the results of the basis model and some heuristics, the model will be refined in an iterative manner in an attempt to reduce the prediction error. This study will be adopted as shown in Experimental Procedure, where further details of the step by step approach will be discussed.

2.2 Parameters Measuring Fuel Quality

In a study carried out by Shah et al. 2019, auto-ignition and flame propagation were said to be the two main parameters considered in engine combustion design. The flame propagation is affected by laminar flame speed which is the speed at which the flame propagates though the un-burnt fuel in the engine. Auto-ignition is affected by Ignition Delay Time (IDT) and octane rating, which is the measure of anti-knock. IDT is the time between an arrival of a shock coming from a flame front and the onset of an ignition. IDT shows the amount of time it takes for the fuel to burn and serves as an indicator of resistance to combustion.

2.3 Octane Number and its Importance

Octane number is the single most important fuel quality indicator because it is a measure of the anti-knock properties of a gasoline blend (Ibrahim and Al-Kassmi 1997). Knocking occurs in spark ignition engines and occurs when the combustion of an air and fuel mixture in the engine is not a result of flame propagation from the flame front. Flames in the engine must only be a result of the propagation of the front resulting from the flame front. Knocking of the engine will therefore happen when combustion does not occur in the precise moment of the four-stroke cycle. Increasing the compression ratio in a combustion engine improves the efficiency of the engine but this efficiency requires higher octane rating fuel to prevent knocking as the tendency of knocking increases at higher compression ratios(Ibrahim and Al-Kassmi 1997).

Given the implication of the octane number on the anti-knock abilities of a combustion engine, octane number can therefore be considered a reasonable proxy for the other parameters discussed earlier in this review. Knocking is representative of the four-stroke cycle not happening efficiently and this affects the flame propagation and flame ignition time. The factors affecting knocking therefore make octane number the most significant factor to asses fuel quality.

Because engine knocking may have destructive consequences to a combustion engine, it is important to ensure RON is maximised or produced to be within a particular range. Having said this, octane number is to be determined for every blend produced at a refinery to inform process control and ensure that good quality gasoline is produced.

2.4 Factors Affecting Octane Number

There are several factors which affect how high the octane number of a given gasoline blend may get. Ibrahim and Al-Kassmi 1997 suggest that there is a strong correlation between the octane number of a blend and its boiling point. The octane number is observed to decrease with increasing boiling point (Ibrahim and Al-Kassmi 1997). The rate of this decrease will vary among different gasoline blends. The strong correlation between boiling point and octane number show that boiling point is a parameter which must be considered when building a model for predicting the octane number of a fuel blend.

Abdul-Gani et al. 2018 carried out an investigation where a ANN was used to predict octane number. The variables they considered were the weight fractions of hydrocarbon functional groups, the branching index and the molecular weight. The chemical composition of the fuel is expressed in terms of the functional groups present in hydrocarbons. The functional groups will dictate the physical properties such as flame speed and flash point. The molecular weight is considered because the heavier molecules will reduce octane number. The branching index is a new parameter which describes the degree of branching in a molecule by considering the positions of methyl groups in the molecule (Abdul-Gani et al. 2018). The functional groups present in a blend are sufficient to reflect the reactivity of the fuel which ultimately affects the ignition of any air/fuel mixtures.

Physical properties can also be used to predict octane number, Ibrahim and Al-Kassmi 1997 considered only the boiling point of the blend as a variable for use in predicting octane number while Al Fahemi, Albis, and Gad 2014 used a wider array of physical properties some of which include; hydration energy, critical pressure and molar refractivity. The contrast between Al Fahemi, Albis, and Gad 2014 and Abdul-Gani et al. 2018 is the approaches of using physical and chemical properties respectively. A potential extension of the framework

of the two investigations may be to consider how both physical and chemical properties affect the prediction of the octane number. Meusinger and Moros 2001 also list that octane number can be increased by the following molecular factors.

1. Shorter straight chains alkanes and olefins.
2. Increased side chains (branching).
3. Smaller rings in naphthenes.
4. Shorter branched chains.

2.5 Methods of Predicting Octane Number

2.5.1 Data-driven Predictive Modelling in Quality Control

Previous attempts at predictive fuel quality and combustion parameters of multi-component fuels were phenomeno-logical models. The models as described by Venkatasubramanian 2019, were largely a program of the engineer's thinking process, coupled with complex kinetics and differential equations. Although these models could be solved quickly, the biases of the engineer were included in the model and in many instances some important reaction pathways were overlooked. This made the models less accurate and with the advent of machine learning, data driven models have been deployed to train models which learn from trends found in data. This does not in any way render phenomeno-logical models useless but suggests that they be used together to increase prediction accuracy.

2.5.2 Phenomeno-logical Models

Phenomeno-logical models tend to focus more on describing the observable phenomena derived from experiments. Many combustion properties rely on the reaction kinetics and mechanisms of the many species making up the fuel blend. Shah et al. 2019 argue that some of these reaction kinetics are approximations at best and are very difficult to obtain if they are visible, making construction of a phenomeno-logical model time consuming and difficult. This may have been the case at the advent of reaction modelling, but given the technology available today, many of the models are multiple differential equations, which can be solved computationally in a matter of minutes. One drawback of phenomeno-logical modelling is that many of the models only work in certain ranges of temperature, pressure and other conditions, meaning that different models may need to be built for different ranges. Another problem with phenomeno-logical modelling is the high uncertainty associated with parameters such as rate constants and activation energies in the models and possible omission of significant reaction pathways (Shah et al. 2019).

One of the earliest attempts at developing a phenomeno-logical model which only included the composition of the blend was done by Stewart 1959. Volume compositions were used to predict the octane rating. Stewart 1959 proposed that the octane rating of a gasoline blend can be given by the following equation.

$$R_{mix} = \frac{\sum_{i=1}^{n} V_i D_i (R_i + CP_i)}{\sum_{i=1}^{n} V_i D_i} \tag{2.1}$$

Where:

- V_i = Volume of component i added to the blend.
- P_i = Difference between olefin volume composition in component i and the blend.
- D_i = Weighting factor calculated from Equation 2.2.
- R_i = Octane rating of component i.
- C = 0.130 for RON and 0.097 for MON. It is a constant.

$$D_i = \frac{AP_i}{1 - e^{-AP_i}} \tag{2.2}$$

Stewart 1959 make two major assumptions:

- Linear dependence of RON on the volume composition and individual octane number of each component.
- Additivity of the component volumes.

The above assumptions would weaken the model because it is widely known that the gasoline volumetric composition has a non-linear effect on the octane number (Abdul-Gani et al. 2018), (Twu and Coon 1996) . The assumption of additivity is not always valid for volume additions. Although hydrocarbon mixtures behave in an ideal manner, mixtures including oxygenates do not. The additivity assumption is therefore only weakened when the mixture contains oxygenates. The additivity can however be modelled more accurately by using a thermodynamic equation of state to take into account he inter-molecular forces introduced to the mixture by the oxygenates.

Another model where octane number was modelled as a function of Ignition Delay Time (IDT) was determined by Singh et al. 2017 and is shown in Equation 2.3. IDT is defined in Parameters Measuring Fuel Quality.

$$RON = 98.21 - 64.91 \exp\left(-\frac{IDT}{0.00363}\right) - 903.7 \exp\left(-\frac{IDT}{0.00056}\right) \tag{2.3}$$

Equation 2.3 above shows the relationship between RON and ignition delay time is exponential as shown in Figure 2.3. The octane numbers of the blends are measured in a lab while ignition delay time is determined by simulation to generate the empirical relationship between the RON and the IDT. A simulation was used because the test for IDT is quite complex. In their study, Singh et al. 2017 would have developed kinetic models for each gasoline blend but this model is not shown.

The plot of shown in Figure 2.3 below also shows the straight line relationship between the measured and predicted values. It was found that the plot had an R^2 value of 0.9932. This shows that the simulation is performing very well.

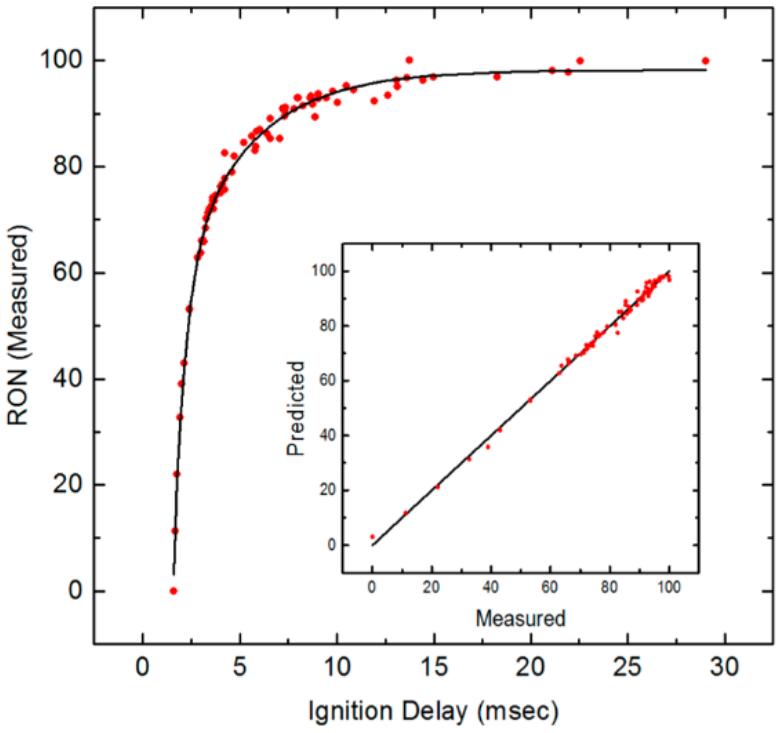


Figure 2.3: Graph showing the proposed relationship between IDT and RON and the respective regression plot taken from Singh et al. 2017

The input variables used in the study by Singh et al. 2017 where physical values; temperature and pressure in the engine coupled with the compositions of each of the blends studied. The temperature and pressure where the parameter used in developing the reaction kinetics which were not explicitly defined in their paper. C. Huang, Golovitchev, and Lipatnikov 2010 carried out a study with a similar approach to Singh et al. 2017 where fuel composition, temperature and pressure were used to determine an empirical model for predicting flame speed and IDT.

2.5.3 Multiple Linear Regression

Twu and Coon 1996 attempted to predict the octane number of a gasoline blend using a linear regression model with binary interaction parameters. They conclude that the interaction method will accurately predict the octane number of the gasoline blends. Twu and Coon 1996 extend the work done by Schoen and Mrstik 1955 from a simple linear regression model to a regression model with binary interaction parameters shown in Equation 2.4.

$$a = \sum_{k} a_k x_k + \sum_{i} \sum_{j} x_i x_j a_{ij} \tag{2.4}$$

Where:

- x_i and x_j are the volume fraction of any two components considered in each binary combination.
- a is the RON of the mixture.

- a_k is the RON of component k
- x_k is the volume fraction of component k
- a_{ij} is the mathematical average mean rule of the octane numbers of the pure components Given by Equation 2.5

$$a_{ij} = \frac{1}{2}(a_i + a_j)(1 - k_{ij}) \tag{2.5}$$

Where:

- a_i and a_j are the octane number of the pure components.
- k_{ij} is the binary interaction parameter between pure components.

Twu and Coon 1996 extend the formulation above to incorporate the non linearity between gasoline cuts as described by using a different method to determine the binary interaction parameter a_{ij} as shown in Equation 2.6. Where K_{ij} becomes the non-linear binary interaction parameter between gasoline cuts.

$$a_{ij} = \frac{1}{2}(a_i + a_j)(1 - K_{ij}) \tag{2.6}$$

Where:

$$K_{XY} = 1 - \frac{\sum_i^m \sum_j^m \left(x_i y_j + x_j y_i\right) \left[\frac{1}{2}\left(a_i + a_j\right)\right] \left(1 - k_{ij}\right)}{\sum_i \sum_j^m \left(x_i x_j + y_i y_j\right) \left[\frac{1}{2}\left(a_i + a_j\right)\right] \left(1 - k_{ij}\right)} \tag{2.7}$$

Instead of using an interaction parameter between pure components, the interaction parameter is developed between different gasoline cuts. Equation 2.7 shows iterative combination of the different components by means of a sort of polynomial to capture the non-linearity of the factors affecting octane number. Twu and Coon 1996 use the above model to predict octane number given only the composition of the fuel. It is known that the structure and branching of the components also affect the octane number and should have been included. Twu and Coon 1996 also do not have polynomial terms in their regression model, only the interaction parameters, perhaps a combination of both polynomial and interaction terms may increase the efficiency of the regression model.

Another approach to predicting the octane number was taken by Al Fahemi, Albis, and Gad 2014. Quantitative Structure-Property Relationship (QSPR) was developed and is used to predict the octane number of hydrocarbons by correlating physical properties with the octane number (Al Fahemi, Albis, and Gad 2014). This model is not a very strong and useful model as the physical properties have no significant link to the RON The explanatory variables are listed below:

- Molecular mass M
- Hydration energy E_H
- Boiling Point B_P

- Octanol/Water distribution coefficient $log(P)$
- Molar Refractivity M_R
- Critical Pressure C_P
- Critical Volume C_V
- Critical Temperature C_T

Al Fahemi, Albis, and Gad 2014 developed a linear regression model for the above variables to predict the octane number.

$$RON = \beta_0 + \sum_{i=1}^{n} \beta_i X_i \tag{2.8}$$

Equation 2.8 above is a multiple linear regression model where the physical properties are regressed onto the octane number of the blend.

$$\begin{aligned} RON = & -(193.53 \pm 319.19) \\ & +(1.47 \pm 1.01)M \\ & -(53.06 \pm 31.47)E_H \\ & -(8.67 \pm 2.73)B_P \\ & -(24.94 \pm 19.44)M_R \\ & -(50.52 \pm 26.92)\log P \\ & +(4.33 \pm 3.09)C_P \\ & +(3.72 \pm 2.04)C_V \\ & +(5.17 \pm 2.08)C_T \end{aligned} \tag{2.9}$$

Al Fahemi, Albis, and Gad 2014 assume that there is linearity between the physical properties of the fuel and its octane number. This may not always be true and their model may have been more reliable if interaction parameters had been added to the regression model. Although their model does not explicitly include it, structure of the components is implied in some of the physical properties such as hydration energy and boiling point. In exploring the data, Al Fahemi, Albis, and Gad 2014 used Principal Component Analysis (PCA) shown in Figure 2.4 to determine correlations between variables. This is useful to see how the variables change with respect to one another.

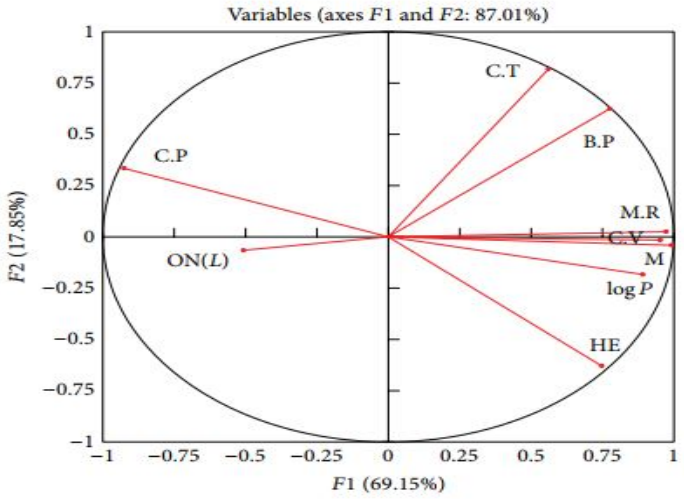


Figure 2.4: Bi-plot of descriptor variables taken from Al Fahemi, Albis, and Gad 2014

The bi-plot in Figure 2.4 above shows that Molecular mass, molar refractivity and critical volume are highly correlated. PCA is a form of unsupervised learning in which correlations between variables are determined by linear methods (Friedman, Hastie, and Tibshirani 2001). PCA is a linear method and is not very reliable for an attempt to show these correlations, a more accurate non-linear method such as kernel PCA would have produced a more reliable bi-plot. More unsupervised learning methods may be used to cluster the different blends to see the pure components which are similar in their descriptors and may be used in this study as part of the Exploratory Data Analysis (EDA).

Albahri 2003 assigned a RON to the structural groups present in each of the PIANO compounds. The final model proposed by Albahri 2003 is shown in Equation 2.10. Where $\sum_i(\text{ON})_i$ is the sum of the RON of the structural groups present in each of the PIANO groups. These then form the contributions to the RON.

$$\begin{aligned}\text{RON} = a + b\left(\sum_i(\text{RON})_i\right) + c\left(\sum_i(\text{RON})_i\right)^2 + d\left(\sum_i(\text{RON})_i\right)^3 + \\ e\left(\sum_i(\text{RON})_i\right)^4 + f\left(\sum_i(\text{RON})_i\right)\end{aligned} \tag{2.10}$$

This method of determining RON is similar to the method taken by Abdul-Gani et al. 2018 with the difference being Abdul-Gani et al. 2018 used weight fractions of structural groups present whereas Albahri 2003 assigned a RON to each structural group and used that to train their algorithms. Albahri 2003 then carry on with their investigation by training a neural network for comparison of results. The polynomial regression performed well with an R^2 value of 0.975. The results and architecture of the neural network they build is discussed in Deep Learning.

2.5.4 Projection Pursuit Regression

The above regression methods are all parametric models. Parametric machine learning models are models whereby the algorithm simplifies a function to a known form (Russell and Norvig 2016) such as the form shown in Equation 2.4. Parametric models do not change with growing data and are typically more logical in their interpretation. While parametric models

exist, there are also non-parametric models which have been used to model the RON. A non parametric model does not assume an underlying function and assumes very little about the data (Russell and Norvig 2016). A non-parametric model typically takes linear combinations of the input variables forming derived features, a non linear function of these derived features is determined and used to model the target variable (Friedman, Hastie, and Tibshirani 2001). Non-parametric models are best used on very large data sets to minimize the amount of sparsity in the data (Friedman and Stuetzle 1981).

One of the earliest work on predicting the RON from gasoline composition using non-parametric models was done by Van Leeuwen, Jonker, and Gill 1994. In their study Van Leeuwen, Jonker, and Gill 1994 used 2 of these non-parametric models; ANN Projection Pursuit Regression (PPR). Composition of PIANO groups was used in the study.

In the above study, gas chromatographic analysis of gasolines was carried out and the results of the mass distributions of the gasolines over individual components were combined into the compositions of their respective PIANO groups. The resulting compositions were then used to predict the RON of the gasoline. The data used had 824 different gasolines. The gas chromatogram of each blend produced 89 structural groups and these were used as input variables.

Projection pursuit regression is a model in which the regression surface is modelled as a sum of general smooth functions of linear combinations of the input variables iteratively as opposed to parametric models where one assumes a functional form and the problem becomes one of estimating the parameters of the function (Friedman and Stuetzle 1981). The projection pursuit regression model is shown in Equation 2.11 and the detailed algorithm of PPR is discussed in Regression.

$$f(X) = \sum_{m=1}^{M} g_m \left(\omega_m^T X\right) \tag{2.11}$$

Where:

- X is the vector of a single input variable.
- ω_m is the unit vector which is the projection vector for smooth m.
- g_m is a smooth function for the mth projection vector.

Before Van Leeuwen, Jonker, and Gill 1994 attempted to use non-parametric methods, they used Principal Component Regression (PCR) and Multiple Linear Regression (MLR) which they concluded to work poorly unless a large number of variables was used. This can however, introduce the curse of dimensionality which makes parametric models perform poorly due to the increased sparsity in the data.

Before a model was built on the entire data set, Van Leeuwen, Jonker, and Gill 1994 carried out cross validation on 4 disjunct randomly sampled validation sets of size 100. This resulted in 4 models being built using each validation set as a test set. Model adequacy was then evaluated using a parameter called the Root Mean Square Error of Prediction (RMSEP) which determined by Equation 2.12 below, which is referred to as Root Mean Square Error (RMSE) in more recent research.

$$RMSEP = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{2.12}$$

It is then observed that when two smooth functions are used in the PPR model, the RMSEP rises, this shows that more noise is being incorporated in the model and to ensure that the model performs well, Van Leeuwen, Jonker, and Gill 1994 used one smooth function instead. The noise also suggests that there is a lot of variance in the target variable (RON). The entire data set was then split into a test and training set on which the PPR model was built. When each smoothing function is produced, the model is still not suitable to use for prediction and the smoothing function g_m is transformed to a Cartesian equation f_m. From one smooth function, Van Leeuwen, Jonker, and Gill 1994 obtained 2 straight line fits of the type $y = \beta_0 + \beta_1 V$ with a RMSEP of 0.34 where $V_m = \omega_m^T X$.

Outliers were found to greatly impact the model. Outliers are determined by means of EDA. PPR was observed to be trained more accurately when outliers had been removed. The resulting two linear functions were defined as steps in the model and these were interpreted as two different groups of gasolines. Each group having similar compositions forms different projections on the projection vector.

2.5.5 Ensemble Learning

Ensemble learning can be described as a single model which is a combination of simpler base models (Friedman, Hastie, and Tibshirani 2001). Tree-based methods such as random forests have been the most popular methods used in literature for predicting fuel quality parameters. Random forests can be considered an ensemble learning algorithm because they are a collection of trees weighed together to make a prediction.

An attempt to predict the octane number using a tree based method was made by Lee et al. 2013. They used Random Forests (RF) which is an ensemble learning method. A random forest is a collection of a number of de-correlated trees(James et al. 2013). Each of the trees are built by randomly sampling a fraction of the data each time to build each tree, making each tree independent. More details on how random forests work is shown in Ensemble Learning.

Lee et al. 2013 build trees for the pure components whose spectral properties are known. They state in their study, that random forests remain effective in predicting RON when there is highly overlapping Near Infrared Resonance (NIR) spectra from complex samples. This suggests that random forests will be able to work well in instances where the data being modelled is very similar and may serve as an advantage for the purposes of trying to predict RON when the different blends have similar properties and compositions. Random forests are indeed useful in predicting gasoline properties because gasoline has a high degree of compositional complexity (Lee et al. 2013). The random forest, which is a combination of trees is then built to represent the calibration samples for predicting the RON. The method proposed by Lee et al. 2013 works by tuning the trees to Near Infrared spectra. Their random forest is grown and trained from data obtained from NIR. The NIR spectrum of the sample can be used to determine the RON of a sample.

In their study, Lee et al. 2013 propose that the NIR can be done very quickly in real time and the RON can be predicted from the NIR spectra. This method is therefore not a direct determination of RON using compositions. NIR can however be done in real time on

a process stream to obtain its spectra and then the RON of the stream may be determined from the corresponding spectra.

González 2019 attempted to model how multi-component blends affect the properties of gasoline. In González 2019's study, the RON, MON and another parameter S which is the difference between RON and MON were predicted using several machine learning methods. González 2019 built a decision tree and a random forest for both a mole basis and a volume basis. González 2019 trains trees with depth of 17 levels and 15 levels respectively for volume and mole basis respectively and the Mean Square Error (MSE) is used as a quality indicator. The decision tree has the advantage of being able to visualise the split at each node. Random forests were also grown and the trees had a depth of 12 and 13 respectively for volume and molar basis.The interesting result was the number of estimators for each basis which were 24 and 128 respectively for volume and mole basis. The volumetric basis had a lower performance than the molar basis with a R^2 of 0.9701 while the molar basis had an R^2 of 0.9852. González 2019 predicts the RON directly from the compositions.

2.5.6 Deep Learning

Deep learning is a class of machine learning algorithms where multiple layers are used to extract high level features from raw input and are inspired by the biological neural networks found in the cranial system of animals (Friedman, Hastie, and Tibshirani 2001).

The second method used by Van Leeuwen, Jonker, and Gill 1994 to model RON prediction is neural networks. Their study was one of the earliest works using neural networks to predict RON. The same cross validation method was used as the one they used in building the PPR model. The difference between the PPR and neural network approaches is that in PPR one must obtain the best number of smooth functions while with neural networks one must determine the number of hidden layers and the number of neurons in each layer. Van Leeuwen, Jonker, and Gill 1994 used a single hidden layer ANN and determined the best number of neurons in the hidden layer which was found to be 4. They also determined an RMSE of 0.35 for the ANN. The ANN is also a model which works by over-parameterizing the model where as PPR is a dimension reduction technique. Van Leeuwen, Jonker, and Gill 1994 conclude that PPR and ANN can indeed be used to capture the non-linearity between RON and PIANO groups with PPR performing slightly better than a single layer ANN. They also conclude that even smaller non-linearities in RON can be incorporated well by non-parametric models.

Another approach to modelling the octane number as a function of structural groups present was taken by Meusinger and Moros 2001. In their study, they used both the presence of structural groups and the presence of the different families of compounds present in the blend. The families of compounds investigated were; **O**xygen, **R**ings, **A**romatics, aliphatic **C**hains and O**L**efins (ORACL).

The data used in Meusinger and Moros 2001 consisted of 323 pure compounds. The data set was then split 85%, 10% and 5% into training test and validation sets respectively. Instead of dividing the compounds into their different ORACL families, the NMR spectra showed 28 chemical shift regions corresponding to each of the structural groups present. The matrix for prediction was populated by the number of times a structural group occurred in the compound. The presence of each ORACL group on the compound was recorded as 1 regardless of the frequency of the ORACL group on the compound or a 0 if a particular ORACL group was not present. This resulted in a matrix of 323 rows and 33 input variables. They trained a neural network with two hidden layers but they do not specify the number of nodes in each hidden layer. Meusinger and Moros 2001 achieve a R^2 value of 0.944 and

conclude that NMR spectra are not sufficient to use as input to a machine learning algorithm because structure relationships would be captured poorly. Meusinger and Moros 2001 also mention that the neural network results cannot be interpreted directly which is expected and well known.

Most of the approaches taken in the reviewed work thus far was done in a chemistry lab context from experimental data. Pasadakis, Gaganis, and Foteinopoulos 2006 takes a more process engineering type of approach to model the RON. Pasadakis, Gaganis, and Foteinopoulos 2006 model the volumetric composition of gasoline fraction obtained from different process units on a Greek refinery as inputs to the model. The process units from which gasoline fractions are produced are listed below.

- Fluidized Catalytic Cracking (FCC).
- Reforming (REF).
- Isomerization (ISO).
- Alkylation (ALK).
- Dimersol (DIM).
- Butanes (C_4)
- Oxygenate Additives (MTBE)

One would be concerned that with differing feed and other system disturbances, the concentration of the output from each process unit may not always be consistent. This may make the idea of using composition of gasoline fractions directly a bit less reliable although the model may be more easily implemented in the plant operations. A data frame with 173 observations is used as input in this study. The ANN developed in the studies contained 12, 5, 3 and 1 nodes respectively in the input, first hidden layer, second hidden layer and 1 output layer. Pasadakis, Gaganis, and Foteinopoulos 2006 split the data into 160 training observations and 13 test observations. The model had a R^2 value of 0.989. after having trained a single ANN, 9 others were trained with the same architecture but different initial weights to verify that the ANN produced was not over-fit. Pasadakis, Gaganis, and Foteinopoulos 2006 conclude that ANN models can describe multivariate spaces (such as those formed by the data in their study) where the data is not uniformly distributed.

Another approach to using descriptors derived from the molecular structure of compounds called Quantitative Structure-Property Relationship (QSPR) has been used by several researchers to model chemical and physical properties of hydrocarbons. A QSPR model was built by Kubic et al. 2017 in an attempt to predict cetane number CN, RON and MON of hydrocarbons and oxygenated organic compounds found in bio-fuels. They used a data set of 218 compounds to predict octane number. The complexity of the octane properties is to be handled cautiously and to do this, Kubic et al. 2017 used the Smolenskii's concept of complexity. Smolenskii concludes that RON is a complex property because of large variability within groups of isomers. Group contribution methods such as those used by Albahri 2003, Abdul-Gani et al. 2018 and Meusinger and Moros 2001 simplify the approach to describing the molecular structure of organic compounds (Kubic et al. 2017).

Kubic et al. 2017 justify the use of a ANN as a suitable method for RON by stating that the ANN is the best model to capture the non-linearities in the interactions among the different structural groups present in organic molecules. The data used by Kubic et al.

2017 consists of structural groups of carbon compounds ranging from 1 to 12 carbons and 38 structural groups which form the input variables. Some of the observations had multiple octane numbers and these would bias the model. To prevent data skewing, only three points were used for a given compound; the best estimate, the maximum and the minimum. The next data cleaning step was to remove outliers and numbers reported as inequalities. In their modelling approach, Kubic et al. 2017 trained a ANN with a small sample of the data and then built a final ANN using all the data. They used a logistics function shown in Equation 2.13 as the activation function for the ANN.

$$y_k = \frac{1}{1 + e^{-z_k}} \tag{2.13}$$

In the initial small ANN, Kubic et al. 2017 use 3 of the structural groups as input variables in the input layer; $-CH_3$, $-CH_2-$ and $-OH$. The model had one hidden layer with 3 nodes. The final model produced had an input layer, a hidden layer with 8 nodes and 3 output nodes corresponding to CN, RON and MON. The correlation between CN and both RON and MON was found to be weak as the ANN had 4 of the hidden nodes predicting CN while the remaining 4 predicted both MON and RON. Their model was then validated using two methods; 10-fold cross validation and an extrapolation test. 10-fold cross validation is simply a method of testing the final mode on 10 different randomly samples sub-data sets. The extrapolation method on the other hand was to test the model on data where the number of carbon chains was outside the range of the training data.

The difference between the regression error and the 10-fold validation error using an F-test with a null hypothesis stating "Errors of the validation set are significantly greater than the regression errors of the same set of compounds." The null hypothesis was supported by the result of the F-test. The use of the F-test here suggests a potential use of a statistical testing method to compare the differences between models as opposed to using only the correlation coefficients. Kubic et al. 2017 conclude that The mathematical structure of a ANN allows them to out-perform most empirical QSPR models and that ANN-based group contribution models can be applicable to a broad range of hydrocarbons to predict physical and chemical properties of new bio-fuels being produced.

Machine learning algorithms have been increasing in their application to modelling the RON because of the low cost associated with using them and their speed (Abdul-Gani et al. 2018). In their study, Abdul-Gani et al. 2018 generated the data from NMR spectroscopy and trained a ANN. The data generated was the functional groups present in the compounds. This is another structural group method where gasoline composition was transformed to structural group composition. The weight fraction of each functional group and the other parameters which make up the input variables are listed below.

1. Paraffinic $-CH_3$.
2. Paraffinic $-CH_2-$ groups.
3. Paraffinic $-\underset{\diagdown\ \diagup}{CH}$ groups.
4. Olefinic $-CH=CH_2$ groups.
5. Naphthenic $\diagdown\ \diagup CH-CH_2$ groups.

6. Aromatic $\overset{}{=}\!\!\overset{}{=}\text{CH}-\text{CH}\overset{}{=}\!\!\overset{}{=}$ groups.
7. Ethanolic $\overset{}{-}\!\!\overset{}{-}\text{OH}$ groups.
8. Molecular weight.
9. Branching Index.

The above parameters were determined for each of the 281 available blends, 224 points were randomly sampled as the training set and 57 as a test set. K-fold validation was used to verify the model on different randomly generated sets as tuning the model on the test set will lead to mis-representative error metrics (Abdul-Gani et al. 2018). K-fold cross validation also advantageous as it can overcome the variance in the data set. They tuned the ANN by varying the number of nodes per layer, regularization coefficients, and the number of layers. The final ANN presented by Abdul-Gani et al. 2018 consisted of an input layer with 9 nodes, 2 hidden layers with 540 and 314 nodes respectively and one node in the output layer. The ANN built by Abdul-Gani et al. 2018 has an accuracy rate of 98.2% and an R^2 value of 0.99. This shows that the model captures the non-linearity of the input variables well. They conclude that the functional groups can be used to predict both blends and pure hydrocarbon properties. Similar conclusions were made by Kubic et al. 2017 and Meusinger and Moros 2001. Abdul-Gani et al. 2018 are the only researchers who explored the concept of the branching index. Other researchers have used the positions of side chains as different functional groups and that resulted in the input variables getting to as much as 30. Using the branching index allows the data set to be smaller as fewer functional groups would be required. This reduces the potential down falls which may result from the curse of dimensionality.

Tian, You, and X. Huang 2018 used an ensemble of 2 deep learning methods; Stacked De-noising Auto-Encoder (SDAE) and Back Propagation Neural Networks (BPNN). The combination of the two algorithms is referred to as Stacked De-noising Auto-Encoder with Back Propagation (SDAE-BP) The SDAE-BP consists of a cascade of De-noising Auto-Encoder (DAE)s. The output of the SDAE is the initial weight used in the BPNN. Tian, You, and X. Huang 2018 acknowledge that a ANN performs very well in addressing the non-linearity between RON and any input variables. The other problem being solved by Tian, You, and X. Huang 2018 is reducing the time it takes for the model to run. The two reasons for their investigation make their method worthwhile to be considered for use on a chemical plant as a soft sensor model for control purposes.

The SDAE is trained with unlabelled data and the weights in each DAE are used as initial weights in the BPNN. The SDAE weights can avoid local minima in the BPNN and accelerate convergence. Tian, You, and X. Huang 2018 used sigmoid activation functions to build the SDAE and BPNN. Tian, You, and X. Huang 2018 then use gradient descent to tune the parameters of the BPNN and optimize the loss function. Regularization was then used to minimize variance and reduce over-fitting by optimizing the loss function.

The model used by Tian, You, and X. Huang 2018 is based on NIR spectra. The model has 201 input variables which are derived from the peaks of the spectra and 466 observations. The observations are split 400 for the training set and 66 for the testing set. 5 different SDAE-BP were trained by ranging the number of hidden layers from 1 to 5. In all 5 models, all the hidden layers had 9 nodes. The best performing SDAE-BP was the one with 4 hidden layers; with an R^2 value of 0.94 and an MSE of 0.023.

Tian, You, and X. Huang 2018 conclude that obtaining the RON in the gasoline blending process can be done fairly quickly by carrying out the NIR spectroscopy on the process stream at the refinery and running the resulting spectra through the SDAE-BP. This approach was

also taken by Lee et al. 2013 who trained random forests using the SDAE-BP using NIR spectra. Both approaches taken by Tian, You, and X. Huang 2018 and Lee et al. 2013 are not composition based methods. Having an extra layer of NIR spectroscopy running may be capital intensive and may slow down the model; making the process control less effective. A composition based AI model may outperform the NIR method because there is no additional layer of data extraction being done.

The final deep learning algorithm which shall be explored for this study is the Extreme Learning Machine (ELM). ELM is a special case of Single Layer Feed-forward Neural Network (SLFN); where instead of specifying the initial weights, the weights are randomly set. Wang, Yang, and Kalivas 2020 explored 3 cases of ELM; Extreme Learning Machine with Radial Basis Function (ELM-RBF), Online Sequential Extreme Learning Machine (OS-ELM) and Self Adaptive Evolutionary Extreme Learning Machine (SaDE-ELM).

In their study, Wang, Yang, and Kalivas 2020 compared the performance of the 3 ELM algorithms mentioned above by forecasting the RON. Similar to Tian, You, and X. Huang 2018 and Lee et al. 2013, Wang, Yang, and Kalivas 2020 built predictive methods from NIR spectra of gasoline blends. 60 different blends were sampled and scanned. Each sample had 401 variables and 1 output variable (RON). 50 of these samples were randomly selected for the training set while the remaining 10 samples were left as a test set. Each of the three models where trained repeatedly with different activation functions; sigmoid, sine, hardlim and RBF. The performance attributes used were Root Mean Square Error (RMSE), R^2, the correlation co-efficient and the time. In all ELM models, Wang, Yang, and Kalivas 2020 observed that the sigmoid function was the best performing activation function. The R^2 value of all the methods was determined to be in the range [0.7,1], which shows that generally the ELM performs well.

The SaDE-ELM with 20 nodes in the hidden layer performed best for all the activation functions but had the drawback of taking approximately 10 times longer to make a prediction because of the iterative optimization of weights and biases. Although the speed is much slower, it is worthwhile to compromise on speed for increased accuracy (Wang, Yang, and Kalivas 2020). The best performing SaDE-ELM had an R^2 of 0.93 which is a good value and shows that the model approximates the actual data reasonably well. Wang, Yang, and Kalivas 2020 conclude that SaDE-ELM and OS-ELM increase the prediction accuracy, generalization ability and stability of the ELM. This conclusion is expected because the OS-ELM continuously updates the output matrix of the hidden layer and the weight matrix of the output layer. The SaDE-ELM is also expected to perform better because it uses differential evolution to optimize the weights of the input layer and the bias of the hidden layer. This eliminates the randomness of the weight assignments while overcoming the risk of converging at a local minimum. Using the ELM, RON prediction directly on the composition data of a process stream would be much faster than conducting NIR of a process stream and making predictions on NIR data.

3 Research Motivation

3.1 Problem Statement

This study aims to produce a parsimonious model which can be implemented at a refinery as part of the Advanced Process Control (APC) system for the blending circuit. The model will be a machine learning model which will be used to predict the Research Octane Number of process streams in the blending section of the refinery. The final model will predict the RON of the gasoline product stream given the compositions of the stream.

None of the quality indicators can be measured directly from the product stream. The gasoline quality must therefore be determined from the measurable properties of the gasoline product stream. The most commonly used indicator, Research Octane Number will be used in this study. RON however, cannot be measured online. The models will be built with the assumption that the stream compositions are known and can be measured rapidly. The target will therefore be to blend the streams effectively to achieve the required RON.

3.2 Gap Analysis

Multiple studies have been conducted in pursuit of a suitable method to predict RON either in real time on a plant or from experimental data in a chemistry lab. The earliest attempts at modelling the RON, where in the time Venkatasubramanian 2019 referred to as the expert systems era. During this period, the methods used were majority phenomeno-logical models, which describe the observable phenomena and in some cases a human algorithmic process and multiple IF statements (Venkatasubramanian 2019). One of the earliest phenomeno-logical models was proposed by Stewart 1959; their model assumed linear dependence on the volume compositions and individual RON of each of the components of the gasoline blend. As time went on, some phenomeno-logical models involved complex differential equations to model the reaction kinetics. The reaction kinetics were often approximations at best, with high uncertainties in the parameters of the models. The high uncertainty and the computational expense of solving multiple complex differential equations made phenomeno-logical models less attractive (Shah et al. 2019). Another recent attempt at phenomeno-logical modelling was made by Singh et al. 2017 where the RON was modelled as a function of IDT. This model which may look simple, was disadvantageous because Ignition Delay Time (IDT) is difficult to measure in real time and therefore cannot be used in real time for process control in a blending circuit.

As time progressed and the industry entered the neural networks and machine learning era (Venkatasubramanian 2019) and data driven models were employed. The earliest attempts were made by Twu and Coon 1996 who extended the work by Schoen and Mrstik 1955 who had modelled RON using linear regression without considering the non linear effect of the explanatory variables on the RON. Twu and Coon 1996 built a regression model which was non-linear in its response but linear in the coefficients of the model. Van Leeuwen, Jonker, and Gill 1994 used Projection Pursuit Regression (PPR) and ANNs in their study and realised that both methods performed well in capturing the non linearity of the explanatory variables, this was an early attempt at using non-parametric models to predict RON.

Now that the machine learning algorithms are developed in what Venkatasubramanian 2019 refers to as the deep learning and data science era, more attempts have been made to predict the RON using more complex transformations and models. Many researchers used

either Nuclear Magnetic Resonance (NMR) or Near Infrared Resonance (NIR) spectra as input data to their models and some have claimed that NIR spectra can be done quickly on process streams in real time. Using NIR spectra however, can be capital intensive in terms of installation. This is why this study argues that some of the methods of predicting RON provided may only be useful for a lab context and not a chemical plant context. Some of these approaches have predicted RON using the variables described below.

1. Composition of individual compounds.
2. Composition of structural groups.
3. Presence of structural groups.
4. Arbitrary assignment of RON to each of the structural groups.
5. Process fractions.
6. Physical properties.

Approaches 2-4 listed above were using NIR or NMR spectra to obtain the structural group composition. This study aims to develop a transformation function, in the form of an AI model to map compound composition to structural group composition. This will be more easily implemented into a plant wide control system as composition data for streams is always known and the need for installation of additional equipment will be done away with. The predictive model will be used to predict RON using both compound and structural group compositions. It has been observed in literature that with increased variables, more information can be captured and the predictions can be more accurate. Physical properties will not be considered because all physical properties can be derived from chemical compositions and other information contained in composition may be omitted in the predictive model. Using individual process streams, although potentially the fastest method will not be considered because process streams vary with other parameters such as feed compositions and process conditions and would cause too much variability in the compositions of the blending streams.

This study will take the composition of the final gasoline stream and transform that composition into molar composition of structural groups present in the blend. This transformation will be parsed into a predictive model which will output the RON of the gasoline blend.

Once the models has been trained, integration into Advanced Process Control (APC) system will be recommended for the blending model to be implemented in the bending circuit.

3.3 Hypotheses

One hypotheses can be formulated to set the trajectory of this study. This hypothesis can be made regarding the explanatory variables. Since RON is set at zero for n-heptane and 100 for i-octane, one can hypothesise that the compositions of substituent structural groups of i-Octane and the additives will rank higher in variable importance than the composition of n-heptane. It is therefore hypothesised that the composition of i-octane and its substituent structural groups will rank highest, followed by the additives and their substituent groups and then n-heptane will have the lowest variable importance.

3.4 Aims and Objectives

This study aims to produce a parsimonious model which can be used to predict the RON of the gasoline product stream. To meet the aims of this study, the following objectives have been defined:

- Determine molar composition of structural groups from the composition of compounds in each of the blends.
- Explore data properties for potential dimension reduction and perform outlier analysis.
- Determine the order of significance of each structural group and compound.
- Develop models for predicting the RON from the structural group and compound compositions.
- Compare the results of each model using relevant error metrics and model diagnostics and discuss causal effect of structural groups on the RON.
- Determine either the best performing model or a combination of the models which form the best performing single model.
- Explore integration of the final model to the plant-wide Advanced Process Control (APC).

4 Methodology

4.1 Data

4.1.1 Data Description

Table B.1 in Database shows the data which will be used for this study. The data set shows the different blends which will be explored and their volumetric compositions. The variables of the blend data shown in Table B.1 are the compositions of the **PIANO** groups and the additives; alcohol and ether. The RON is the output variable. In this study, the gasoline blends used are composed of the 7 compounds described below. Each of the compounds listed below are used as proxies for measuring the compositions of their respective class of compound.

1. Paraffin: The paraffins are measured by the composition of n-heptane.
2. Iso-Paraffin: The iso-paraffins are measured by the composition of iso-octane.
3. Aromatics: The Aromatics are measured by the composition of toluene.
4. Naphthene: The Naphthenes are measured by the compositions of cyclopentane.
5. Olefin: The olefins are measured by the composition of 1-hexene.
6. Alcohol: The alcohol content will be measured by the composition of ethanol.
7. Ether: The ether will be measured by the composition of Ethyl tert-butyl ether (ETBE).

4.1.2 Data Acquisition

The data used in this study was obtained from multiple sources. The pure component data will be obtained from the supporting information provided by Kubic et al. 2017. The blend data was taken from González 2019, Singh et al. 2017, Al Fahemi, Albis, and Gad 2014, Abdul-Gani et al. 2018 and Yuan et al. 2017.

4.2 Experimental Procedure

Figure 4.1 below shows how the approach proposed by Venkatasubramanian 2019 will be adapted to build each of the models above in this study.

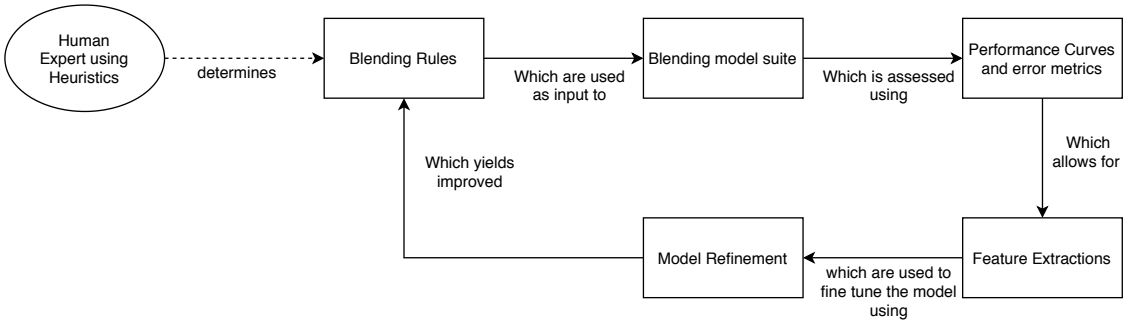


Figure 4.1: Adaption of cyclic modelling approach proposed by Venkatasubramanian 2019

4.2.1 Data Pre-processing and Transformation

After extraction from several sources, the data is combined in to a single data set. The compounds which make up each of the gasoline blends have a known molar structural group composition, these will be used to determine the molar structural group composition of each blend. This will result in a new data set with 18 input variables; 11 of which will be the structural group composition and the remaining 7 will be the compound composition and one output variable (RON) as shown in Table 4.1. The Exploratory Data Analysis (EDA) will then be carried out on the transformed variables (shown in table 4.1).

Table 4.1: Structural Groups, Compounds and their respective variables names

Variable Number	Variable Name	Structural Group/ Compound
1	x_1	—CH_3
2	x_2	—CH_2—
3	x_3	—CH< (C# $\geq$ 4)
4	x_4	>C< (C# = 2)
5	x_5	=CH_2
6	x_6	=CH— (C# = 2)
7	x_7	—CH_2— (Cyclic)
8	x_8	=CH— (Aromatic)
9	x_9	=C< (ortho)
10	x_{10}	—OH
11	x_{11}	—O— (2nd)
12	x_{12}	n-Heptane
13	x_{13}	i-Octane
14	x_{14}	Toluene
15	x_{15}	Cyclopentane
16	x_{16}	1-Hexene

Table 4.1 continued from previous page

Variable Number	Variable Name	Structural Group/ Compound
17	x_{17}	Ethanol
18	x_{18}	ETBE

4.2.2 Mixture Analysis

The molar structural group composition of each blend is determined using procedure outlined below.

1. List the blend components and their respective volume compositions.
2. Pick a basis of 1L of the blend so to obtain the volume of each compound present.
3. Multiply the compound volumes by their respective densities to obtain the mass of each compound present.
4. Divide the mass of each compound by its molar mass to obtain the moles of each compound present.
5. Determine the molar structural group composition of each compound.
    5.1. Count the number of occurrences of each structural group in each compound.
    5.2. Take the occurrences as the molar ratios of the structural groups in each compound.
    5.3. Sum the ratios to get the total number of moles of structural group present in a single compound.
    5.4. Divide the ratios by the total number of moles to obtain molar structural composition of each compound.
6. Multiply the moles of each compound by its molar structural group composition to get the moles of each structural group in the compound.
7. Sum the moles of the structural groups present in each compound to obtain the total number of moles present in the compound.
8. Divide the number of moles of each structural group by the total moles present in a single compound to obtain the molar structural group composition of each blend.

A detailed sample calculation of the above procedure is shown in Mixture Analysis Sample Calculation.

4.2.3 Exploratory Data Analysis

The questions to be answered by the EDA, are listed below:

1. Are there any missing entries?
2. How are each of the blend components correlated with the RON?

3. Are there any correlations between the variables?
4. Are there any outliers and how will they be handled?

The EDA procedure will be as follows.

1. Read the data frame into python.
2. Check the column names and their respective variable types (integer, numerical, categorical, etc).
3. Check top and bottom 5 entries of the data for some consistency.
4. Check that all the columns are relevant to the output variable and rename and delete some if necessary.
5. Check for duplicate rows and delete the duplicates.
6. Check for missing entries and delete them.
7. Correlation analysis of the heatmap.
8. Multivariate Outlier analysis using the Mahalanobis distance metric.

4.2.4 Model Building

Once the EDA procedure defined above is complete, the methods listed below will be built for each of the molar structural group composition and the best performing model will be determined by assessing the respective error metrics.

1. Multiple Linear Regression (MLR).
2. Ensemble Learning.
    2.1. Bayesian Additive Regression Trees (BART).
    2.2. Gradient Boosting Machines (GBM).
3. Artificial Neural Networks (ANN).

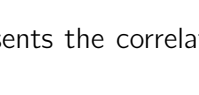
5 Results and Discussion

5.1 Exploratory Data Analysis (EDA)

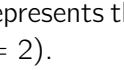
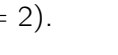
5.1.1 Correlation Analysis

To start the EDA, a correlation analysis was carried out. This is because some of the variables were expected to be identically correlated as a result of some of the structural groups being present in only some of the compounds. Identifying these identical correlations will serve as a dimensionality reduction measure as the structural groups which are only in one compound will be removed from the data set. Figure 5.1 below shows the heat map of the variable correlations. The variables with identical correlation and their respective descriptions are listed below. These identical correlations are a result of the structural groups being found in only the compounds they are identically correlated with.

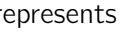
1. $x_3 : x_{13}$

   This represents the correlation between the molar compositions of  (C# $\geq$ 4) and i-Octane.

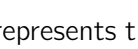
2. $x_5 : x_6$

   This represents the correlation between the molar compositions of  and  (C# = 2).

3. $x_5 : x_{16}$

   This represents the correlation between the molar compositions of  and 1-Hexene.

4. $x_6 : x_{16}$

   This represents the correlation between the molar compositions of  (C# = 2) and 1-Hexene.

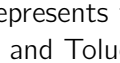
5. $x_7 : x_{15}$

   This represents the correlation between the molar compositions of and Cyclopentane.

6. $x_8 : x_9$

   This represents the correlation between the molar compositions of and  (ortho).

7. $x_8 : x_{14}$

   This represents the correlation between the molar compositions of  (Aromatic) and Toluene.

8. $x_9 : x_{14}$

   This represents the correlation between the molar compositions of  (ortho) and Toluene.

9. $x_{10} : x_{17}$

   This represents the correlation between the molar compositions of —OH and Ethanol.

10. $x_{11} : x_{18}$

    This represents the correlation between the molar compositions of —O— (2nd) and ETBE.

Having considered the identical correlations above, the following variables (structural groups) were removed from the data set. Reducing the number of input variables fro 18 to 10. The heat map of the correlations between the final variables is shown in Figure 5.2.

1. x_3: —CH< (C# $\geq$ 4)
2. x_5: =CH_2
3. x_6: =CH— (C# = 2)
4. x_7: —CH_2— (Cyclic)
5. x_8: =CH— (Aromatic)
6. x_9: =C< (ortho)
7. x_{10}: —OH
8. x_{11}: —O— (2nd)

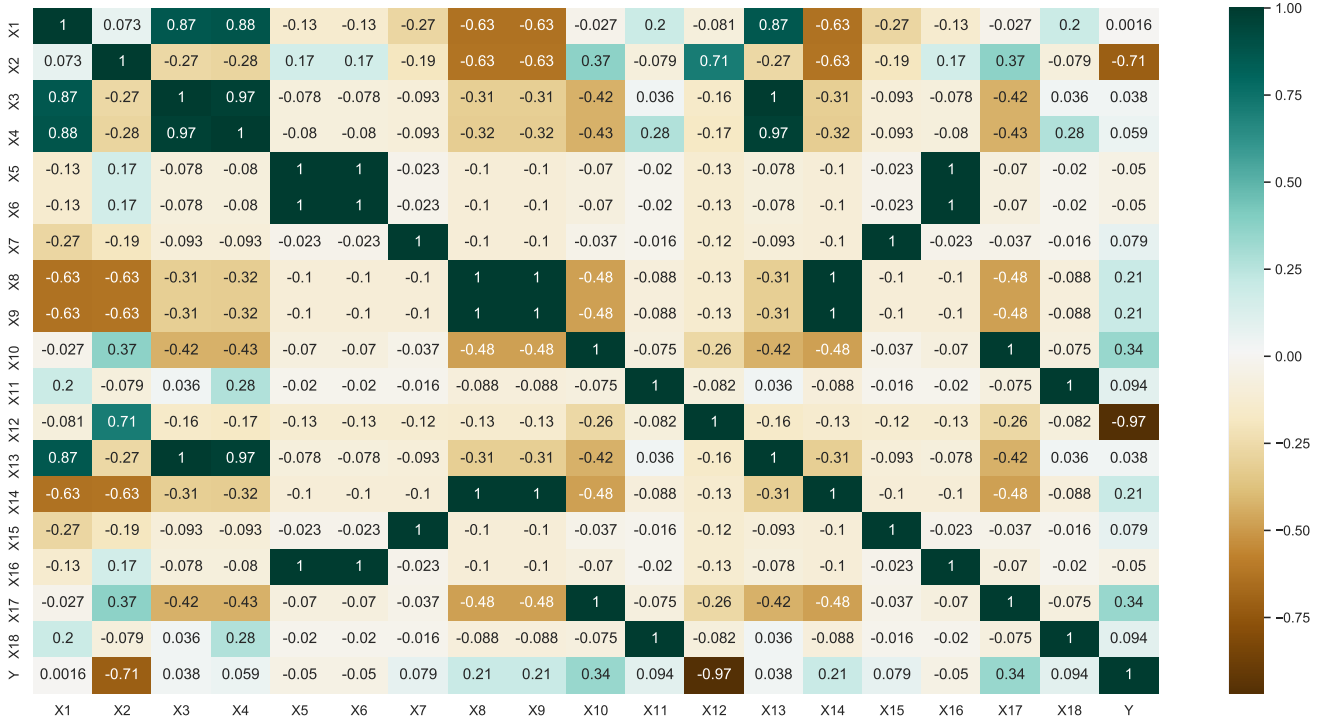


Figure 5.1: Heat Map of Variables

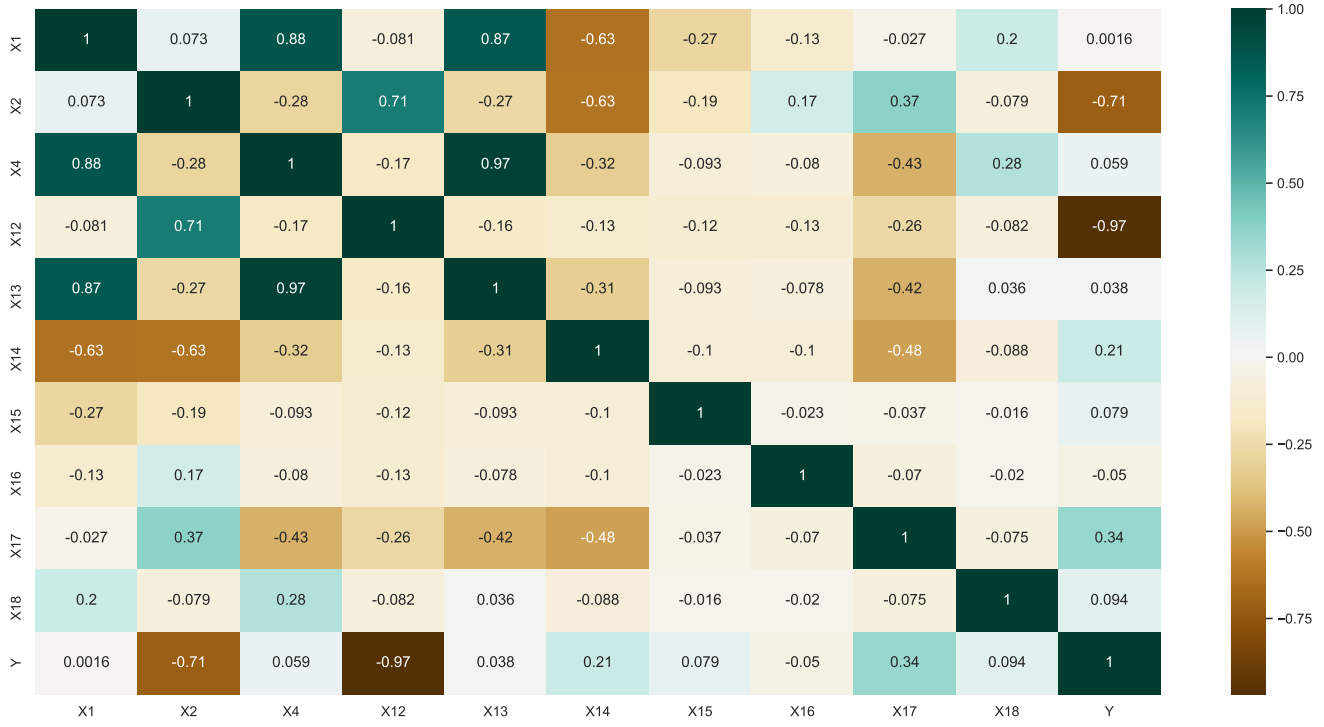


Figure 5.2: Heat Map of Variables

5.1.2 Outlier Analysis

Table 5.1: Blend Molar Compositions Determined to be outliers

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
1	0.46	0.00	0.00	0.00	0.54	0.00	0.00	40.30
2	0.00	0.00	0.00	0.00	1.00	0.00	0.00	73.60
3	0.00	0.00	0.00	0.00	0.81	0.19	0.00	81.00
4	0.00	0.43	0.00	0.00	0.57	0.00	0.00	88.20
5	0.00	0.00	0.00	0.00	0.58	0.42	0.00	89.20
6	0.00	0.00	0.00	1.00	0.00	0.00	0.00	100.00
7	0.00	0.00	0.00	0.85	0.00	0.15	0.00	101.00
8	0.00	0.00	0.00	0.65	0.00	0.35	0.00	102.30
9	0.00	0.00	0.00	0.48	0.00	0.52	0.00	103.30
10	0.13	0.12	0.74	0.00	0.00	0.00	0.00	110.00
11	0.00	0.00	0.00	0.00	0.00	0.00	1.00	117.00

Table 5.1 above shows the molar compositions of the blends which were considered as outliers. The measure of determining the multivariate outliers was the mahalanobis distance. The 11 blends shown in the Table 5.1 can be expected to be outliers not only because of the RON being outside of the common range of RON's which is between 90 and 100 but because blends are not often mixed with these compositions. All the blends in Table 5.1 except for blends 1, 4 and 10 do not have any paraffins in their composition. This means that all the other outliers contain neither n-Heptane nor i-Octane. Petrol blends are not produced without paraffins so these points are indeed outliers.

Blends 1 and 4 contain either n-Heptane or i-Octane and 1-Hexene which is the olefin compound. Petrol blends are not typically blended with only these two compounds so the points are indeed outliers. Point 10 on the other hand has n-Heptane, i-Octane and toluene. This is a blend without any of the additives and may indeed be an outlier.

5.1.3 Final Dataset

Table 5.2: Final input Variables after dimensionality reduction

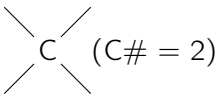
Variable Number	**Variable Name**	**Structural Group/ Compound**
1	x_1	—— CH_3
2	x_2	—— CH_2 ——
3	x_4	 (C# = 2)
4	x_{12}	n-Heptane
5	x_{13}	i-Octane
6	x_{14}	Toluene

Table 5.2 continued from previous page

Variable Number	Variable Name	Structural Group/ Compound
7	x_{15}	Cyclopentane
8	x_{16}	1-Hexene
9	x_{17}	Ethanol
10	x_{18}	ETBE

Table 5.2 above shows the final list of variables in the data set. The input variables have reduced from 18 to 10 and the number of observations has reduced from 500 to 361 after duplicate entries were removed and then from 361 to 350 after the outliers were removed from the data. This was a result of the data cleaning, correlation analysis and outlier analysis above.

5.2 Regression

5.2.1 Model Building

Tables 5.3 to 5.6 below show the results of the regression model. These results will be used to test the null hypothesis that the input variables (compositions) have no correlation with the dependent variable (RON). This null hypothesis tests one of the major assumptions of the regression model which states that the variables must be correlated with the output variable. This null hypothesis will be tested with a 95% confidence level.

The standard errors and t values will be used to asses the precision with which the regression coefficient is estimated . The standard error will show the standard deviation of the coefficients and the amount with which the coefficient varies. The t value is the value of the coefficient divided by its standard error. The larger the absolute value of the t-value, the further the coefficient is from 0.

For the models which were trained with mass balance constraints, the compound compositions were set to sum to one in input of the regression model constraints in the models. The detail as to how this was done are shown in the code extracts in Python Code for MLR with mass balance constraint and Python Code for Interaction MLR with mass balance constraint

5.2.1.1 Multiple Linear Regression Without Mass Balance Constraint

Equation 5.1 below shows the proposed regression model. The model will be trained initially without consideration of the mass balance effects.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_4 + \beta_4 x_{12} + \beta_5 x_{13} + \beta_6 x_{14} + \beta_7 x_{15} + \beta_8 x_{16} + \beta_9 x_{17} + \beta_{10} x_{18} \quad (5.1)$$

Table 5.3 below shows the model parameters of the MLR model without the mass balance constraints.

Table 5.3: MLR without mass balance constraint model parameters

	Estimate	**Std. Error**	**t value**	**Pr(>|t|)**
Intercept	107.519	3.444	31.217	0
x_1	127.621	10.984	11.618	0
x_2	-108.178	11.233	-9.63	0
x_4	-285.98	59.257	-4.826	0
x_{12}	-66.134	4.589	-14.412	0
x_{13}	-35.2688	6.412	-5.5	0
x_{14}	11.487	3.149	-3.648	0
x_{15}	-	-	-	-
x_{16}	-	-	-	-
x_{17}	-	-	-	-
x_{18}	-	-	-	-

Table 5.3 above shows the results of the MLR which was trained without the mass balance constraint on the compound composition. It can be observed that each of the variables which

had non zero coefficients where found to be statistically significant by scoring p-values of nearly zero. Each of these values also had large t values suggesting strong correlation between each of these variables and the output variable. The molar compositions of cyclopentane, hexene, ethanol and ETBE were determined to be zero. These compounds are added to the petrol blends as additives and are sometimes not added to the blends. The result of these compounds having zero coefficients is therefore a sensible observation. The variables x_1 and x_{14} which correspond to the molar compositions of $\text{---}CH_3$ and Toluene respectively. This observation is counter intuitive as toluene is expected to contribute to higher RON as an additive for increasing RON. The molar composition of the $\text{---}CH_3$ structural group is a composite of the compositions of all the compounds besides cyclopentane and the combined effect of these compounds is resulting in the negative effect the composition of the structural group has on the RON.

5.2.1.2 Multiple Linear Regression With Mass Balance Constraint

The MLR model with mass balance constraints was trained as per Equation 5.1. The results of the model are shown in Table 5.4 below.

Table 5.4: MLR with mass balance constraint model parameters

	Estimate	**Std. Error**	**t value**	**Pr($>$|t|)**
Intercept	-3.89×10^8	1.57×10^8	-2.476	0.013
x_1	1.40×10^9	4.03×10^8	3.48	0.001
x_2	-2.55×10^8	3.86×10^8	-0.661	0.508
x_4	7.52×10^8	4.69×10^8	1.603	0.109
x_{12}	1.70×10^8	1.78×10^8	0.957	0.338
x_{13}	-5.51×10^8	1.46×10^8	-3.772	0
x_{14}	1.88×10^8	1.19×10^8	1.585	0.113
x_{15}	3.89×10^8	1.57×10^8	2.476	0.013
x_{16}	2.83×10^8	1.11×10^8	2.55	0.011
x_{17}	5.87×10^6	3.64×10^7	0.161	0.872
x_{18}	-4.85×10^8	1.27×10^8	-3.805	0

From Table 5.4 above it can be observed that the regression model with constraints yielded non zero coefficients for all the variables. It can be observed that the variables x_1, x_{13}, x_{15}, x_{16}, x_{18} have the most significance and correspond to the molar compositions of $\text{---}CH_3$, i-Octane, cyclopentane, hexene and ETBE respectively. Toluene and ethanol have very little significance on the output variable in this model. This suggests that ethanol and toluene are additives which have no effect in this particular model as opposed to the rest of the compounds.

5.2.1.3 Interaction Multiple Linear Regression Without Mass Balance Constraint

The proposed interaction regression model which was trained had interaction parameters as shown in Equation 5.2 below.

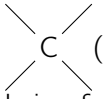
$$
\begin{aligned}
y = {} & \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_4 + \beta_4 x_{12} + \beta_5 x_{13} + \\
& \beta_6 x_{14} + \beta_7 x_{15} + \beta_8 x_{16} + \beta_9 x_{17} + \beta_{10} x_{18} + \beta_{11} x_1 x_{12} + \\
& \beta_{12} x_2 x_4 + \beta_{13} x_2 x_{12} + \beta_{14} x_2 x_{13} + \beta_{15} x_2 x_{18} + \\
& \beta_{16} x_{12} x_4 + \beta_{17} x_{12} x_{14}
\end{aligned} \tag{5.2}
$$

The interaction parameters were determined by performing a grid search on all the possible binary interactions to find the model with the best fit. The variables whose interaction was considered are listed and described below:

1. $x_1 : x_{12}$

   This represents the interaction between the molar compositions of $\text{—}\,\text{CH}_3$ and n-Heptane. These variables may interact as a result of the $\text{—}\,\text{CH}_3$ structural group being found in n-Heptane.

2. $x_2 : x_4$

   This represents the interaction between the molar compositions of $\text{—}\,\text{CH}_2\,\text{—}$ and  (C# = 2). These variables may interact as a result of these structural groups being found in i-Octane molecules.

3. $x_2 : x_{12}$

   This represents the interaction between the molar compositions of $\text{—}\,\text{CH}_2\,\text{—}$ and n-Heptane. This variables may interact as a result of the $\text{—}\,\text{CH}_2\,\text{—}$ structural group being found in n-Heptane.

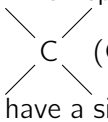
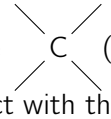
4. $x_2 : x_{13}$

   This represents the interaction between the molar compositions of $\text{—}\,\text{CH}_2\,\text{—}$ and i-Octane. This structural group can also be found in i-Octane molecules and would result in an interaction effect between the two variables.

5. $x_2 : x_{18}$

   This represents the interaction between the molar compositions of $\text{—}\,\text{CH}_2\,\text{—}$ and ETBE. This structural group can also be found in ETBE molecules and would result in an interaction effect between the two variables.

6. $x_{12} : x_4$

   This represents the interaction between the molar compositions of n-Heptane and  (C# = 2). Although the  (C# = 2) structural group is observed to have a significant interaction effect with the composition of n-Heptane, this structural group is not found in n-Heptane molecules.

7. $x_{12} : x_{14}$

   This represents the interaction between the molar compositions of n-Heptane and Toluene. These two compounds would have an interaction effect as toluene is an additive.

Table 5.5: Interaction MLR without constraints model parameters

	Estimate	**Std. Error**	**t value**	**Pr(>|t|)**
Intercept	1.18×10^2	5.50	21.407	0
x_1	1.07×10^2	1.28×10^1	8.349	0
x_2	-1.27×10^2	1.14×10^1	-11.138	0
x_4	-3.37×10^2	1.59×10^2	-2.125	0.035
x_{12}	-9.41×10^1	2.69×10^1	-3.504	0
x_{13}	-3.29×10^1	1.98×10^1	-1.665	0.098
x_{14}	-1.67×10^1	4.35	-3.844	0
x_{15}	-	-	-	-
x_{16}	-	-	-	-
x_{17}	-	-	-	-
x_{18}	-	-	-	-
x_1x_{12}	1.97×10^2	7.66×10^1	2.556	0.011
x_2x_4	2.18×10^9	1.60×10^9	1.363	0.174
x_2x_{12}	-3.41×10^1	1.23×10^1	-2.772	0.006
x_2x_{13}	-2.73×10^8	2.00×10^8	-1.363	0.174
x_2x_{18}	-3.12×10^8	2.29×10^8	-1.363	0.174
x_4x_{12}	-8.77×10^2	1.83×10^2	-4.794	0
$x_{12}x_{14}$	5.94	1.79×10^1	0.331	0.741

Table 5.5 above shows the results of the MLR with interaction effects which was trained without the mass balance constraint on the compound composition. This model can be observed to have the same variables failing to obtain coefficients as its MLR counterpart which was trained without interaction effects. The variables x_1, x_2 and x_{14} are the only variables in the model which were found to be significant. These variables correspond to the molar compositions of $-\!\!-CH_3$, $-\!\!-CH_2-\!\!-$ and toluene. This may be a result of the two structural groups being found each of the compounds besides toluene. Suggesting that the information contained in the composition of the compounds is contained in the compositions of the two structural groups.

Both the MLR models which were trained without the mass balance constraint show that the coefficients of the additives cannot be estimated. This shows that these variables are confounded as these are spuriously correlated with each other due to the fact that they are all additives which are added blended into gasoline to increase the RON.

5.2.1.4 Interaction Multiple Linear Regression With Mass Balance Constraint

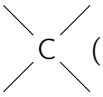
The interaction MLR model with mass balance constraints was trained as per Equation 5.2. The results of the model are shown in Table 5.6 below.

Table 5.6: Interaction MLR with constraints model parameters

	Estimate	**Std. Error**	**t value**	**Pr(>|t|)**
Intercept	-2.16×10^8	1.50×10^8	-1.436	0.151
x_1	1.22×10^9	3.70×10^8	3.302	0.001
x_2	-4.11×10^8	3.47×10^8	-1.183	0.237
x_4	-1.28×10^9	9.65×10^8	-1.322	0.186

Table 5.6 continued from previous page

	Estimate	Std. Error	t value	Pr(> |t|)
x_{12}	1.60×10^8	1.62×10^8	0.984	0.325
x_{13}	-3.37×10^8	1.57×10^8	-2.143	0.032
x_{14}	4.11×10^7	1.15×10^8	0.357	0.721
x_{15}	2.16×10^8	1.50×10^8	1.436	0.151
x_{16}	2.17×10^8	1.06×10^8	2.053	0.040
x_{17}	-5.48×10^7	4.56×10^7	-1.203	0.229
x_{18}	-2.42×10^8	1.52×10^8	-1.591	0.112
x_1x_{12}	1.42×10^4	5.10×10^4	0.278	0.781
x_2x_4	8.27×10^9	3.42×10^9	2.414	0.016
x_2x_{12}	1.75×10^3	6.38×10^3	0.274	0.784
x_2x_{13}	-1.03×10^9	4.28×10^8	-2.414	0.016
x_2x_{18}	-1.18×10^9	4.89×10^8	-2.414	0.016
x_4x_{12}	-3.05×10^4	1.08×10^5	-0.281	0.779
$x_{12}x_{14}$	3.27×10^3	1.18×10^4	0.276	0.782

Table 5.6 above shows the results of the interaction MLR model with the mass balance constraint imposed on the compound compositions. As opposed to the unconstrained interaction MLR model, the model with mass balance constraints yielded non zero coefficients for all the variables including the interaction effects. The variables x_1, x_{13}, x_{16}, x_2x_4, x_2x_{13}, x_2x_{18} were observed to have the greatest significance on the model. The molar compositions of the $-CH_3$ group, i-Octane and 1-Hexene were found to have the greatest significance in this model. The most significant interactions in this model were the interactions between the molar composition of the $-CH_2-$ structural group and the  ($C\# = 2$) structural group, i-Octane and ETBE.

5.2.2 Predictive Statistics

Table 5.7: Summary of Regression Predictive Statistics

Model	Validation MSE	Test MSE	Adjusted R^2
MLR without mass balance constraint	3.44	4.31	0.987
MLR with mass balance constraint	4.47	7.24	0.972
Interaction MLR without constraint	3.20	4.30	0.991
Interaction MLR with constraint	3.03	7.54	0.981

The interaction MLR models outperformed their counterparts which were trained without interaction parameters. Both MLR models which were trained without the mass balance constraint had similar predictive statistics as shown in Table 5.7. The addition of the mass balance constraint does increase the test MSE by a similar margin for both the interaction and non-interaction MLR models. The model fit on the other hand does not decrease by the same margin when the mass balance constraint is added to the interaction MLR as can be seen from the adjusted R^2 values. The interaction MLR models performed better than the models without interaction as expected with both interaction models yielding similar performance.

Twu and Coon 1996 developed a linear regression model using binary interaction parameters to determine the RON of 2 gasoline streams mixed together. They determine the interaction parameters of the regression model using the average mean rule of the octane numbers of the components present in the gasoline stream. The interaction model trained in this study did not involve using pure component information but by proposing a model with interaction parameter shown in Equation 5.2. The problem in this study was to determine the parameters of the proposed model. The model proposed by Twu and Coon 1996 took into account all the possible binary interactions between each of the pure components present in the gasoline blend.

The proposed regression model with interaction parameters only took into account causal interactions which could be seen to increase the accuracy of the MLR model. The interaction MLR model proposed in this study differs from the model proposed by Twu and Coon 1996 because the model proposed in this study considers causal interaction between structural groups and interactions between structural groups and compound composition. The model developed by Twu and Coon 1996 performs well with an adjusted R^2 value of 0.990. The interaction MLR model developed in this study has an adjusted R^2 value of 0.981. Both models perform very well and explain different aspects of the data while the interaction MLR model in this study considers the mass balance constraints of the compounds.

The second regression model considered was developed by Albahri 2003 who assigned an arbitrary RON to each of the structural groups present in the blend and built a regression model by taking polynomial combinations of the sums of the RON of each of the functional groups present in the blend. The form of the model proposed by Albahri 2003 is shown in Equation 2.10. The model proposed by Albahri 2003 in a sense takes into account the causal interactions between structural groups by summing the RON's of the structural groups present. This model however, can be flawed by not considering the weights of each structural group as done in the regression models developed in this study shown in Table 5.7. The interaction MLR model proposed in this study obtained at adjusted R^2 value of 0.981 and outperforms the one developed by Albahri 2003 which obtained an R^2 value of 0.975. Both models attempt to consider the non-linearity of the data but Albahri 2003 specifies a quadratic model which may not be the best descriptor of the non linearity and variability in the data.

5.2.3 Model Diagnostics

5.2.3.1 Analysis of Variance (ANOVA)

Tables 5.8 to 5.11 below show the results of the one way ANOVA carried out on the MLR models built in this study. The null hypothesis for the ANOVA test in regression states that all the coefficients are equal to zero. The results of the ANOVA which was carried out for all the models shows that we reject this null hypothesis. The F values in Tables 5.8 to 5.11 show the extents of the deviation from zero and with the exception of the confounded variables, the F values are large and show that the coefficients are indeed non zero for those variables while the coefficients are 0 for the variables x_{15} to x_{17}.

Table 5.8: MLR without constraints ANOVA

	Mean Sq	**F value**	**Pr(>F)**
x_1	0	3.24×10^{24}	0
x_2	17769	1.10×10^{31}	0
x_4	13104	8.12×10^{30}	0

Table 5.8 continued from previous page

	Mean Sq	F value	Pr(>F)
x_{12}	441	2.73×10^{29}	0
x_{13}	5	3.21×10^{27}	0
x_{14}	4	2.34×10^{27}	0
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	0	2.82×10^{1}	0.06

Table 5.9: MLR with constraints ANOVA

	Mean Sq	F value	Pr(>F)
x_1	190	9.30×10^{2}	0
x_2	15764	7.71×10^{4}	0
x_4	13794	6.75×10^{4}	0
x_{12}	855	4.19×10^{3}	0
x_{13}	6	2.85×10^{1}	0
x_{14}	11	5.44×10^{1}	0
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	1	7.34	0.009

Table 5.10: Interaction MLR without constraints ANOVA

	Mean Sq	F value	Pr(>F)
x_1	0	5.65×10^{1}	0
x_2	17935	4.37×10^{6}	0
x_4	13639	3.32×10^{6}	0
x_{12}	479	1.17×10^{5}	0
x_{13}	1	2.61×10^{2}	0
x_{14}	3	7.11×10^{2}	0
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	0	3.58×10^{1}	0
x_1x_{12}	19	4.62×10^{3}	0
x_2x_4	10	2.52×10^{3}	0
x_2x_{12}	9	2.19×10^{3}	0
x_4x_{12}	67	1.63×10^{4}	0
$x_{12}x_{14}$	0	1.55	0.226

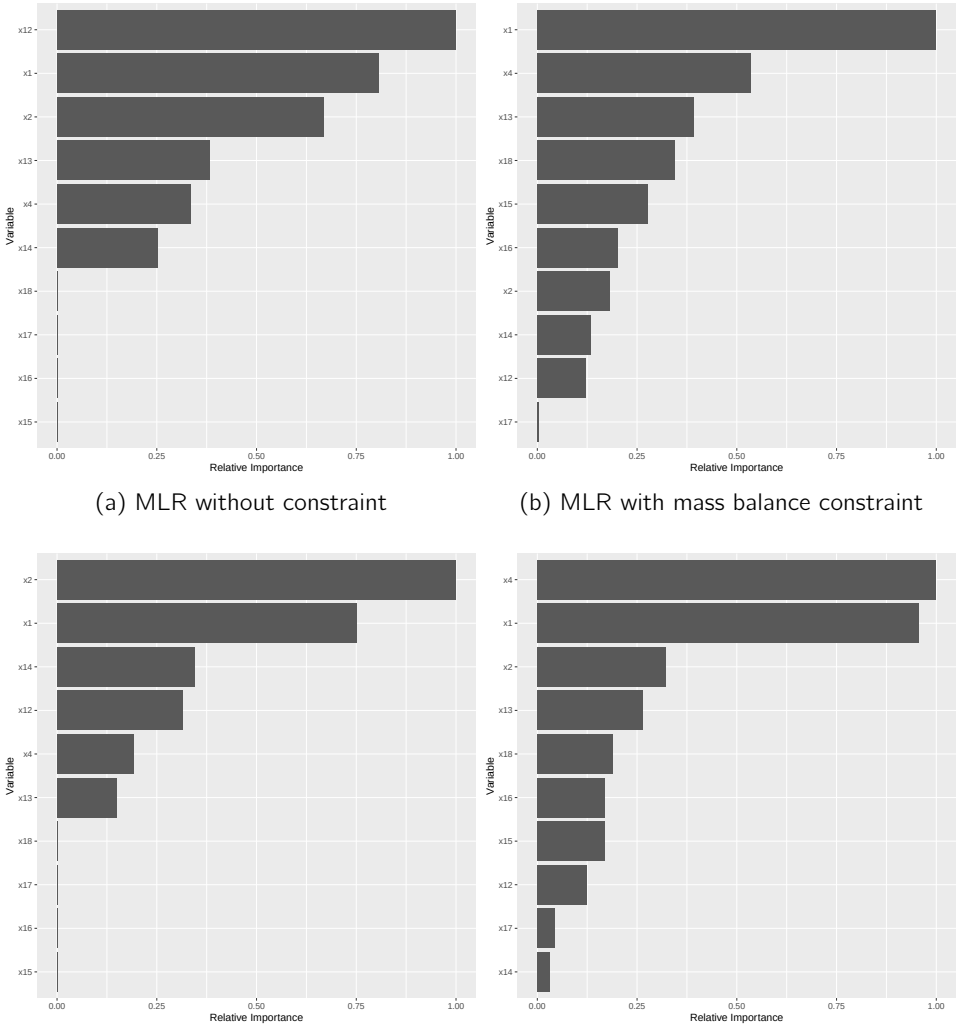
Table 5.11: Interaction MLR with constraints ANOVA

	Mean Sq	F value	Pr(>F)
x_1	223	6.69×10^{2}	0
x_2	15794	4.75×10^{4}	0

Table 5.11 continued from previous page

	Mean Sq	F value	Pr(>F)
x_4	13416	4.03×10^4	0
x_{12}	1191	3.58×10^3	0
x_{13}	5	1.50×10^1	0
x_{14}	51	1.53×10^2	0
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	1	2.35	0.13
x_1x_{12}	0	2.85×10^{-1}	0.60
x_2x_4	18	5.51×10^1	0
x_2x_{12}	0	7.14×10^{-1}	0.40
x_4x_{12}	23	6.93×10^1	0
$x_{12}x_{14}$	2	4.65	0.04
$x_{12}x_{15}$	1	4.44	0.04

5.2.3.2 Variable Importance



(a) MLR without constraint

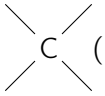
(b) MLR with mass balance constraint

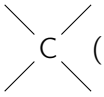
(c) Interaction MLR without constraint

(d) Interaction MLR with constraint

Figure 5.3: Variable Importance Plots of regression models

In the final regression model with interaction parameters, a variable importance analysis was carried out. The most significant variables in descending order were as follows.

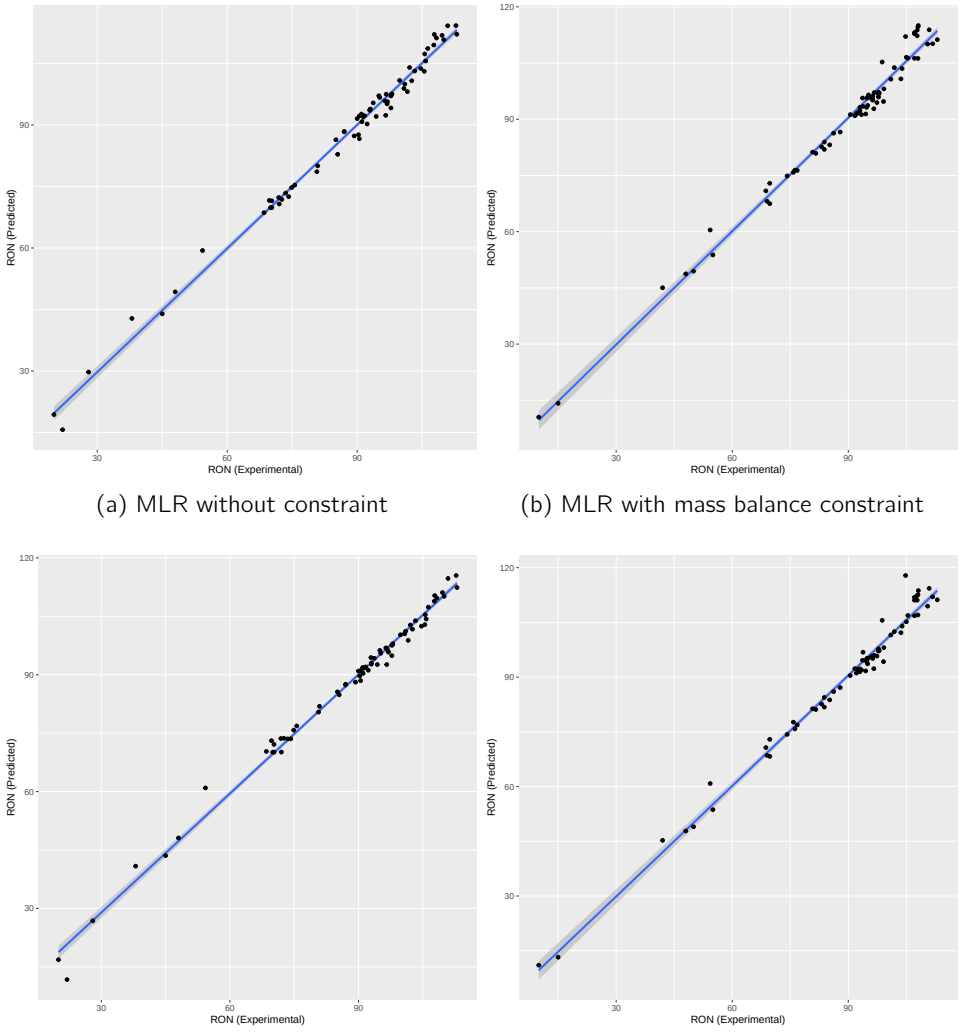
1. x_4:  (C# = 2)
2. x_1: $-CH_3$
3. x_2: $-CH_2-$
4. x_{13}: i-Octane
5. x_{18}: ETBE

The iso-paraffinic  (C# = 2) structural group had the highest ranking in variable

significance in the interaction MLR with mass balance constraints. This is expected as iso-paraffininc groups tend to have a higher impact on increasing the RON as the branching allows for more carbon to carbon bonds which yield a higher combustion energy.

The paraffinic $-CH_3$ and $-CH_2-$ structural groups molar composition ranked second and third respectively. As paraffininc structural groups, it is an expected result as these have the highest impact on RON because they are known to have a negative effect on the RON. I-Octance and ETBE also have a high variable importance in the model because they are components of the fuel which can increase the RON with small quantities.

5.2.3.3 Parity



(a) MLR without constraint

(b) MLR with mass balance constraint

(c) Interaction MLR without constraint

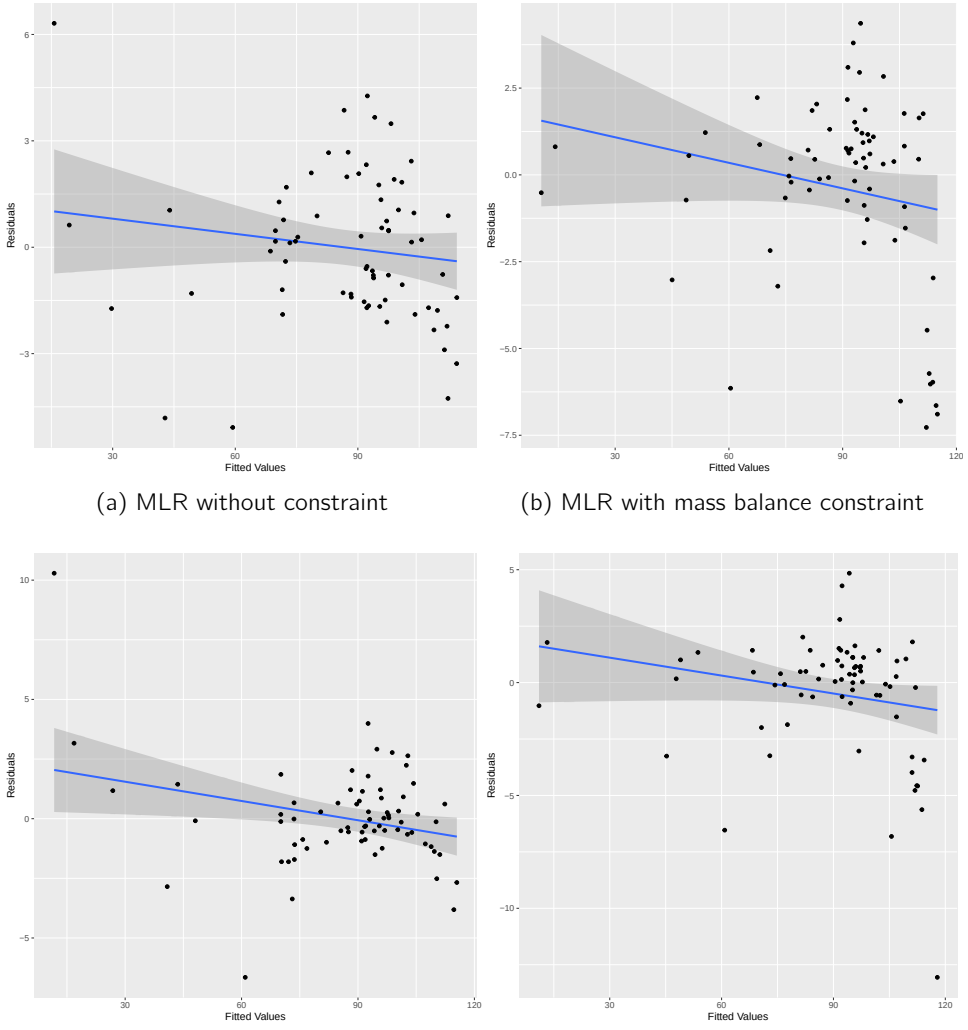
(d) Interaction MLR with constraint

Figure 5.4: Parity Plots of regression models

The parity plots of the regression models shown in Figure 5.4 above show that all the regression models had very narrow uncertainty ranges for all the values of RON. For a RON above 60, the models were observed to have an even narrower uncertainty range. This can be alluded to the range were the RON is above 60, having more data available. Fewer data points were available for instances where the RON was below 60, hence the larger uncertainty

ranges as the models had fewer points to learn from. All the points, including the points were RON was less than 60 appeared to be reasonable close to the line $y = x$ suggesting that the model learnt well in spite of the lack of data. All the regression models demonstrate a strong fit in their parity plots, particularly the regression model with interaction parameters which has the narrowest uncertainty range in the regions with few data points.

5.2.3.4 Residuals



(a) MLR without constraint

(b) MLR with mass balance constraint

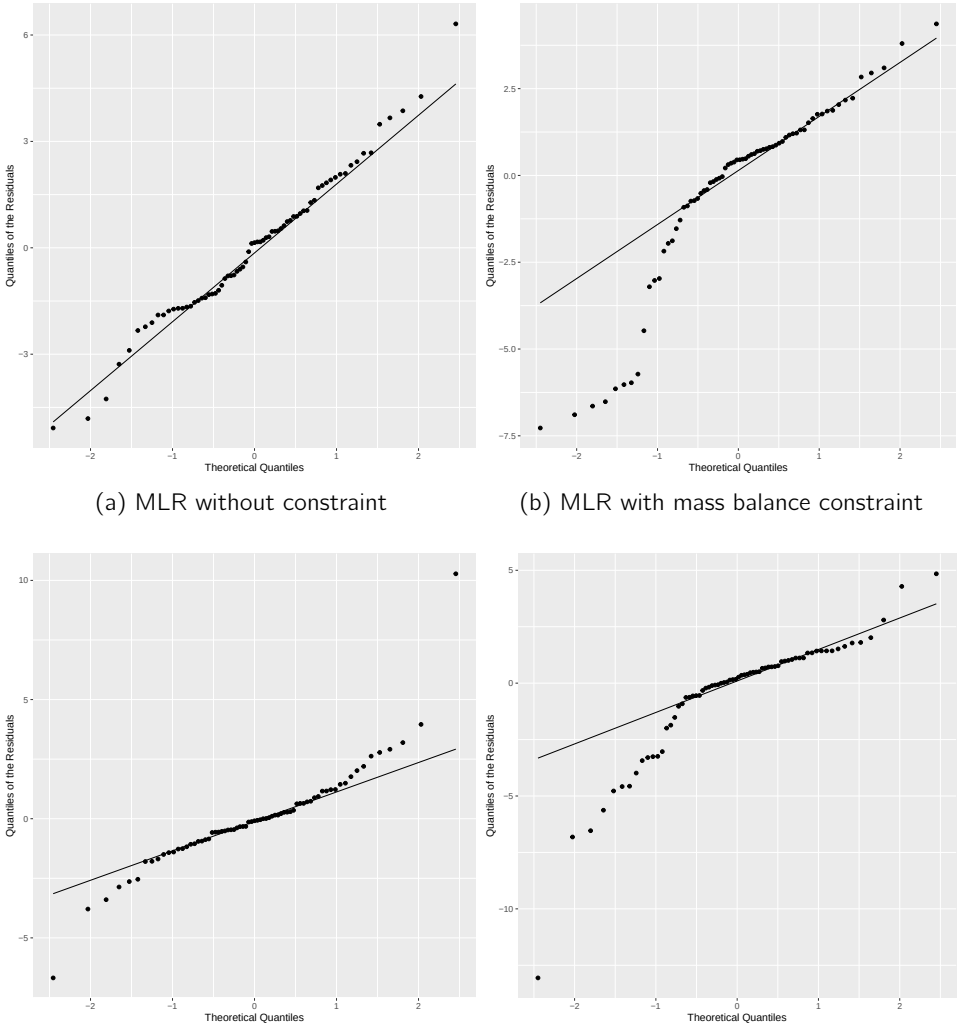
(c) Interaction MLR without constraint

(d) Interaction MLR with constraint

Figure 5.5: Residual Plots of regression models

The residual plots shown in Figure 5.5 above show the residual plots for the regression models trained in this study. In all three regression models, it can be observed that when the RON lies between 90 and 100, the uncertainty range of the residuals is the smallest. This is can be alluded to most of the data being found in this range. In the range where the RON is greater than 60, it can be observed that the residuals are scattered fairly equally around the line $y = 0$. This suggests that the residuals demonstrate homoskedasticity in this region for all four models. The residuals are observed to fall below zero when the RON is higher than 100. This shows that the models fail to capture all the information available from the

data as this is evidence of some information being left in the residuals. Demonstrating some heteroskedasticity in this range. This suggests that the regression models are a good fit for instances where the RON is less than 105. This suggests that the distribution of the residuals is skew and may not be fully IID.



(a) MLR without constraint

(b) MLR with mass balance constraint

(c) Interaction MLR without constraint

(d) Interaction MLR with constraint

Figure 5.6: QQ Plots of Regression Residuals

Figure 5.6 above shows the QQ plots of the residuals of the regression models. The MLR model without the mass balance constraints showed the greatest extent of normality of residuals as the plot had the points scattered about the diagonal line. Both MLR models which were trained with mass balance constraints have the bottom ends of the QQ plots deviating from the straight lines and the residuals are therefore left-skewed. The interaction MLR model produced residuals which were over-dispersed. It can be observed from the QQ plots above that imposing the mass balance constraint resulted in the residuals becoming left-skewed.

5.3 Ensemble Learning: Bayesian Additive Regression Trees (BART)

5.3.1 Model Building

The BART model was trained by varying the number of trees from 0 to 200 trees, with 250 Monte Carlo Markov Chains (MCMC) samples drawn from the posterior distribution. The initial model had a MSE of 0.75 on the training set and an adjusted R^2 of 0.9976. With the residuals being normally distributed as per the Shapiro-Wilk test for normality of residuals. When this initial model was used to make predictions on the validation set, the MSE was observed to be worse than the initial MSE, with a value of 5.74 and an adjusted R^2 of 0.983. Given the sparsity of the data in the low RON ranges, this is a reasonable validation MSE to obtain. Ten fold cross validation using a separate data set was used to tune and validate the BART model. The out of sample RMSE was the metric used to evaluate the accuracy of the model as the number of trees was increased as shown in Figure 5.7. The following grid search parameters were varied in the process of tuning the model.

1. Number of trees [m].

   The number of trees was varied from 0 to 200.

2. The prior probability that $\mathbb{E}(Y|X)$ is contained in the interval (y_{min}, y_{max}), based on a normal distribution [k].

   This parameter was varied from 2 to 5. When $k = 2$, the prior probability is 0.95. As k increases by 1, the prior probability decreases by 0.05.

3. Degrees of freedom for the inverse X^2 prior [ν].

   The degrees of freedom were varied between 3 and 10.

4. Quantile of the prior on the error variance at which the data-based estimate is placed [q].

   The quantile of the prior was varied between 3 and 10.

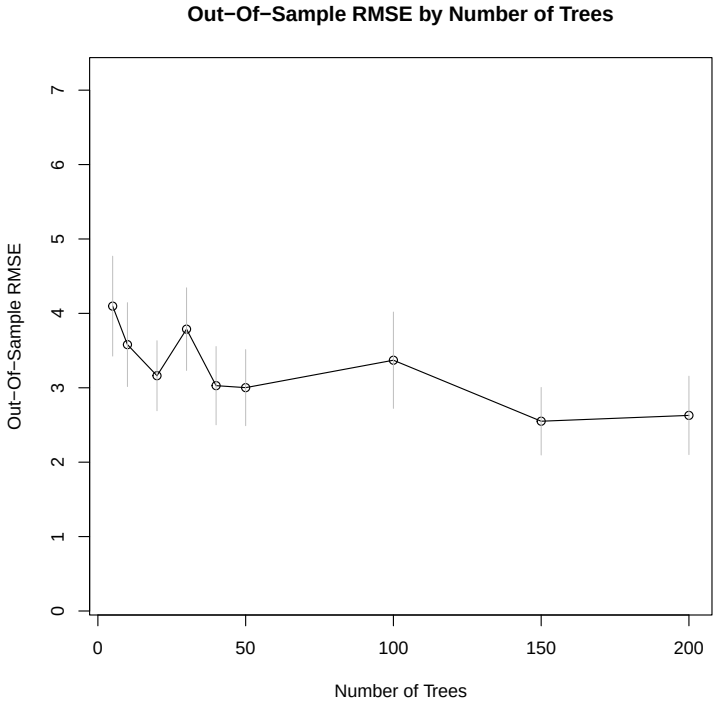


Figure 5.7: BART cross validation results

Figure 5.7 above shows how the number of trees used to train the BART model affected the accuracy of the model using the RMSE diagnostic. It can be observed that when the number of trees reaches 50, the model starts to stabilize. this shows that the model was learning the data well and quickly. At 50 tress, the RMSE is approximately 3, and the model starts to converge at this value of RMSE. As the number of trees is increased, the model RMSE reaches its minimum point at 150 trees and starts to increase. Showing that the BART model obtained the best performance at 150 trees. The grid search, which produced these results was a result of cross validation on a separate validation data set.

5.3.2 Predictive Statistics

The best performing BART model with diagnostics shown in Table 5.12 was determined to have the parameters listed below.

- $m = 200$
- $k = 5$
- $\nu = 3$
- $q = 0.99$

Table 5.12: Model Diagnostics of BART Model

Model	Validation MSE	Test MSE	Adjusted R^2
BART	5.74	13.74	0.983

An ensemble learning method was used by Lee et al. 2013 to predict RON from NIR spectra. They used random forests to predict the RON from NIR spectra. Lee et al. 2013 used a data set of 379 samples and their gasoline blends were composed of paraffins, naphthenes and aromatics. The major difference between random forests and Bayesian Additive Regression Trees is that random forests use bootstrap samples to train the learners while the BART model uses MCMC simulations to train the learners. Although both models are ensemble learning models, the nuance can be observed from how the models are trained. Lee et al. 2013 used random forests as they present a low risk of over-fitting and they learn well from very similar data, which was useful in learning from the overlapping spectra. Using ten fold cross validation, Lee et al. 2013's model converged at 2000 trees and achieved an MAE of 0.22. Their model outperforms the BART model trained in this study which has 200 trees and achieved a MAE of 1.49. This discrepancy in results may be alluded to the data which was used in this study, the data had few learning points in the extreme values of the RON and this affected the ability of the model to learn the data well.

5.3.3 Model Diagnostics

5.3.3.1 Analysis of Variance (ANOVA)

The ANOVA was done and the results in Table 5.13 below. From the results below, we reject the null hypothesis that all the parameters of the model are zero values. This is because only the confounded variables x_{15} to x_{17} have zero values on the results of the ANOVA.

Table 5.13: BART model ANOVA

	Mean Sq	F value	Pr(>F)
x_1	5	1.04	0.312
x_2	16908	3.70×10^3	0
x_4	13041	2.85×10^3	0
x_{12}	491	1.07×10^2	0
x_{13}	2	3.60×10^{-1}	0.551
x_{14}	2	3.58×10^{-1}	0.552
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	1	1.58×10^{-1}	0.692

5.3.3.2 Variable Importance

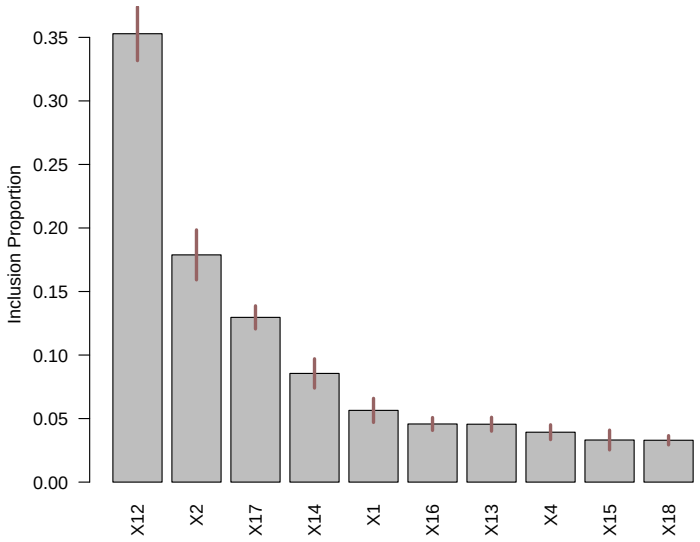


Figure 5.8: BART Variable Importance

The variable importance plot in Figure 5.8 above shows that the following variables corresponding to their respective component compositions have the greatest influence on the BART model in descending order of significance.

1. x_{12}: i-Octane
2. x_2: —CH_2—
3. x_{17}: ETBE
4. x_{14}: Cyclopentane
5. x_1: —CH_3

The highest ranking variable was the molar composition of i-Octane. This is expected as increasing the composition of i-Octane results in an increase in the RON. This demonstrates that an increase in i-Octane composition will increase the RON by a large magnitude. the composition of the —CH_2— structural group ranked second in importance and this is an expected observation as it is a substituent of both i-Octane and ETBE which are the two highest ranking compounds in this model. The molar composition of cyclopentane ranked fourth in the BART model and this is expected as it is an addidtive to increase the RON. It was also observed that the molar composition of the —CH_3 structural group ranked fifth and this is expected as it is a substituent of both ETBA and i-Octane molecules. All the other variables have a similar importance with an inclusion proportion of approximately 0.05 and do not have as much significance as the top five variables.

5.3.3.3 Parity

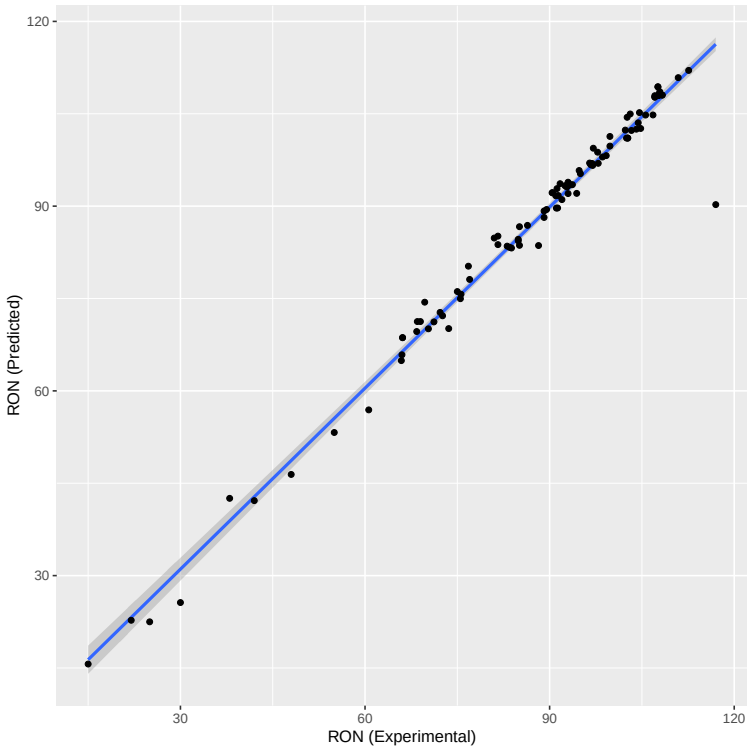


Figure 5.9: Parity Plot of BART Model

The parity plot for the BART model is shown in Figure 5.9 above. In this parity plot, it can be observed that when the RON is below 60, the uncertainty range in the prediction becomes larger. This large uncertainty range can be alluded to a lack of data points where the RON is below 60. The BART model however, observes a narrow range in the uncertainty range as the Monte Carlo Markov Chains (MCMC) bootstrap procedure allows for the points to be equally represented across the whole range of RON. When the RON is between 60 and 100 the points can be observed to be lying reasonably along the diagonal $y = x$ line. An outlier can be observed when the experimental RON is 117 and the BART model predicts a RON of 90. This is the point furthest from the line $y = x$ and this is a result of there being no other points in that range from which the MCMC bootstrap could sample. The parity plot in Figure 5.9 shows that the model has a good fit to the data and may be more parsimonious in representing the ranges with few data points because of the bootstrap sampling method.

5.3.3.4 Residuals

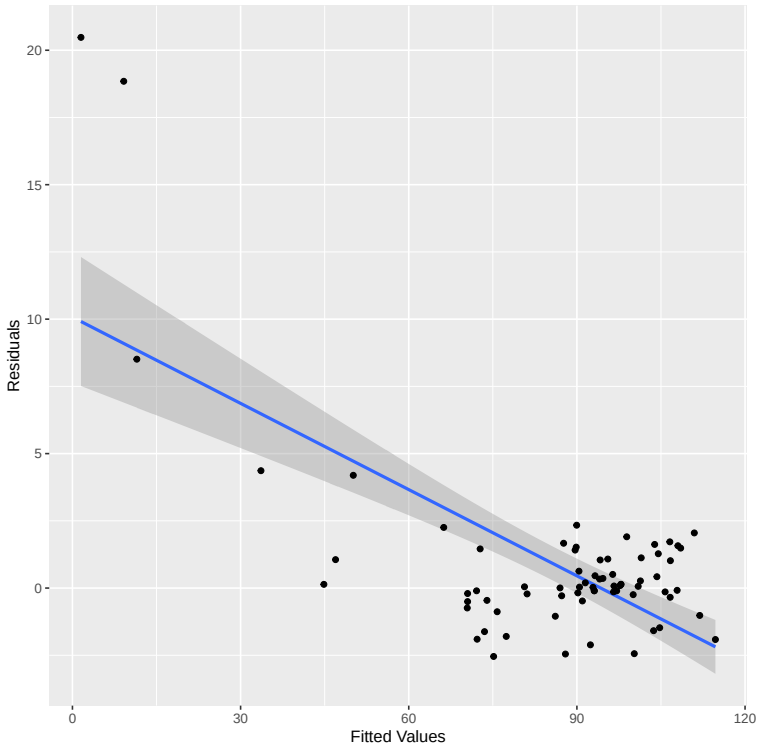


Figure 5.10: Residuals Plot of BART Model

Figure 5.10 shows the residual plot for the BART model. As the RON decreases from 70, the uncertainty range of the residuals can be observed to increase. This observation is a result of the few data points available in this range. The BART model however, observes good distribution around the line $y = 0$ because it can learn more from the data using the MCMC bootstrap sampling method. In the range above 70, The residuals are scattered almost equally around the line $y = 0$. When the RON is below 70, the residuals can be observed to be large and in the range from 0 to 70, there is still some unexplained variance in the model. The residuals when the predicted RON is below 70 can be said to not be IID, while the distribution of the residuals when the RON is above 70 seems to be fair around the line $y = 0$ and in this range the residuals can be said to be IID. The BART model is not parsimonious across the entire range of RON.

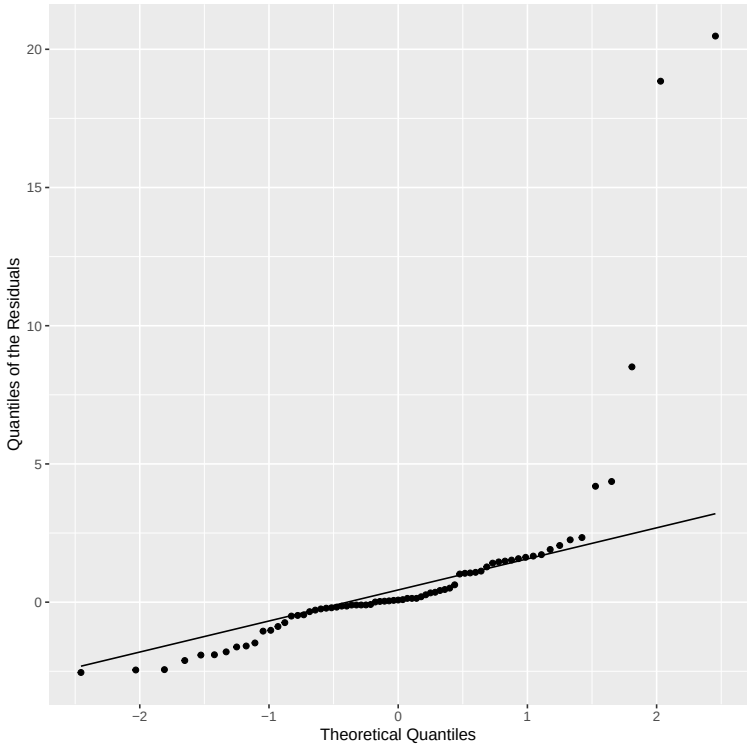


Figure 5.11: QQ Plot of BART Model Residuals

The QQ plot of the residuals of the BART model which was trained are shown in Figure 5.11 above. The residuals were observed to be skewed right.

5.4 Ensemble Learning: Gradient Boosting Machines (GBM)

5.4.1 Model Building

The initial Gradient Boosting Machines (GBM) were trained using 1000 trees and a Gaussian loss function. The maximum depth was specified as 1. This meant that there were no variable interactions considered in the initial model. The learning rate was set to 0.1. The initial GBM model had a validation MSE of 22.5 and an adjusted R^2 value of 0.917. A separate validation set was used to tune and validate the GBM model. The error plot of the GBM model is shown in Figure 5.12 below.

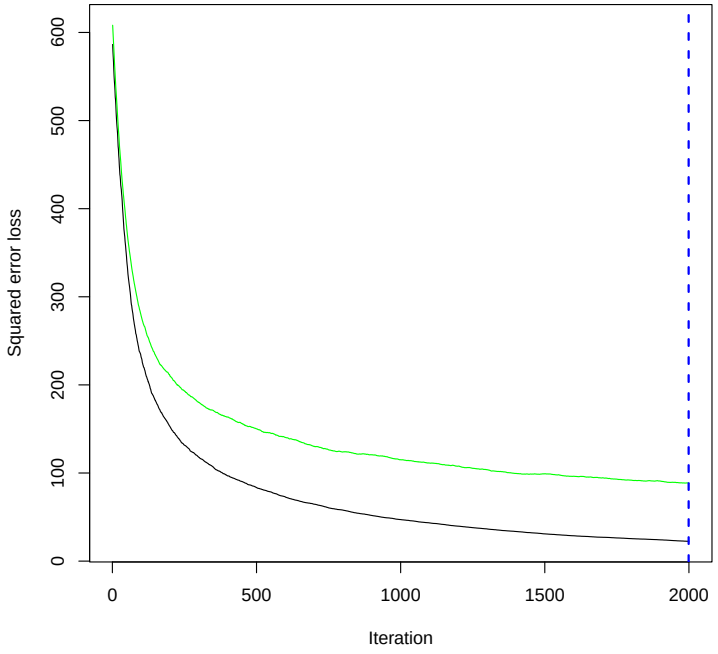


Figure 5.12: Cross Validation error plot for GBM

In Figure 5.12 above, the green line shows the validation error while the black line shows the training error. Cross validation was carried out by monitoring how the error reduced when the number of trees was increased from 0 to 2000. The interaction depth was also increased to 4. Allowing for the models to consider the interactions for up to four variables. It was observed that 1998 trees were the optimal number of trees to use without over-fitting but allowed the error to drop sufficiently while training the model. The horizontal axis shows the iterations of the model and these are equivalent to the trees as the GBM algorithm iteratively adds trees to the model.

Figure 5.12 shows how the squared error loss of the GBM model decreased as the number of iterations (trees) in the model increased. The error loss was observed to decrease in a steep fashion within the first 200 iterations, thereafter the error loss decreases rather slowly. The validation and training errors decrease at the same rate, with the validation error being slightly higher than the training error as expected. The GBM model takes 1998 iterations to converge to a constant error loss. The model converges a bit more slowly as expected after the steep drop in error loss. Although the ensemble of weak learners is improving the error,

most of the variance could be explained by the model in the first 200 iterations. Thereafter, the model fails to increase the amount of variance it can explain as the iterations increase. A separate validation set was used to validate the GBM model.

5.4.2 Predictive Statistics

The diagnostics of the final GBM model which had an interaction depth of 4 and 1998 trees are shown in Table 5.14 below.

Table 5.14: Model Diagnostics of GBM Model

Model	**Validation MSE**	**Test MSE**	**Adjusted R^2**
GBM	22.5	38.12	0.917

There is currently a limited amount of research where ensemble learning was used in RON prediction (Lee et al. 2013). Therefore both the BART and GBM models will be evaluated against the research done by Lee et al. 2013. Both Gradient Boosting Machines's and random forests are ensemble learning techniques which are premised on building a set of decision trees which are the weak learners. The differences emerge in the training processes of the models. Random forests train each of the trees independently and simultaneously, then combine the results at a later stage by taking averages. Gradient Boosting Machiness on the other hand train the trees one at a time and with each tree that is trained, combine the results as each learner is added to the model.

The GBM model trained in this study obtained a MAE of 0.362 as opposed to the model trained by Lee et al. 2013 which had a MAE of 0.22. The GBM model was trained and converged with 1998 trees which is similar to Lee et al. 2013's model which converged with 2000 trees. The slight difference in the diagnostics in the models can be alluded to the types of data used in training both models. The data sourced by Lee et al. 2013 was sampled in real time at a Korean refinery over a period of two years with careful repeats and balance in the data. The data in this study on the other hand was unbalanced; with most of the data being in the range were the RON was between 60 and 105. One can therefore say the GBM model did a good job at modelling the unbalanced data set. It can be said that the GBM model performs better than the random forests because on an unbalanced data set, the GBM model observes similar accuracy to the random forests which were built on a more representative data set.

5.4.3 Model Diagnostics

5.4.3.1 Analysis of Variance (ANOVA)

The ANOVA was done and the results in Table 5.15 below. From the results below, we reject the null hypothesis that the parameters of all the variables are equal to 0. Only the confound variables x_{15} to x_{17} have zero valued parameters based on the F values of the ANOVA.

Table 5.15: GBM model ANOVA

	Mean Sq	**F value**	**Pr(>F)**
x_1	1	1.80×10^{-2}	0.893
x_2	15016	4.25×10^{2}	0
x_4	9252	2.62×10^{2}	0
x_{12}	141	4.00	0.050
x_{13}	27	7.76×10^{-1}	0.382
x_{14}	2	5.40×10^{-2}	0.817
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	68	1.92	0.170

5.4.3.2 Variable Importance

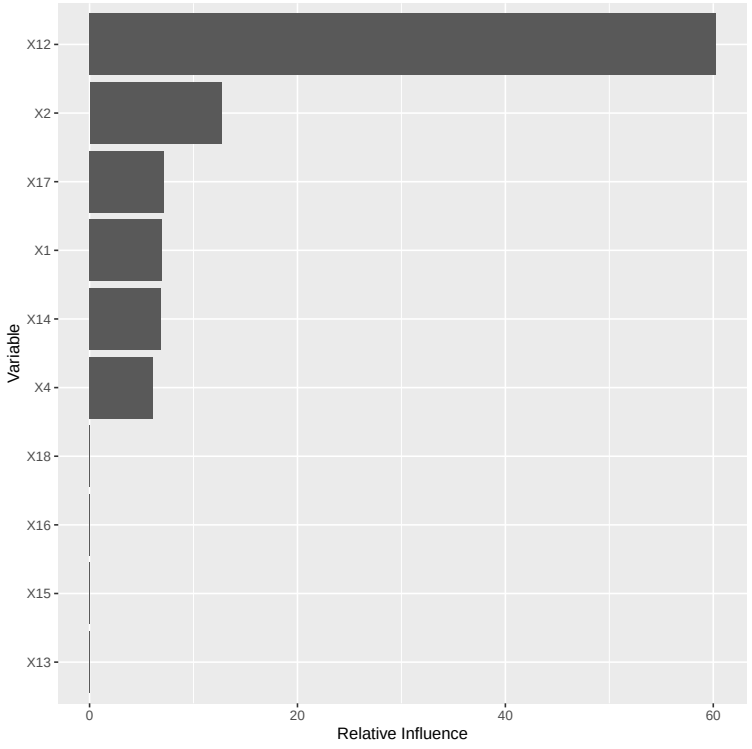


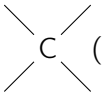
Figure 5.13: Variable Importance for GBM model

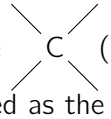
The variable importance plot in Figure 5.13 above shows that the following variables corresponding to their respective component compositions have the greatest influence on the model in descending order of significance.

1. x_{12}: n-heptane
2. x_2: —— CH_2 ——
3. x_{17}: Ethanol

4. x_1: —CH_3

5. x_{14}: Toluene

6. x_4:  (C# = 2)

The variable importance obtained shows that the molar composition of n-Heptane and its substituent structural groups; —CH_2— and —CH_3 rank highly in the GBM model. The molar composition of ethanol and toluene also ranked highly and both of these compounds contain the —CH_3 substituent. The interesting observation is that in this model, the molar composition of ethanol was found to rank higher than that of toluene which is not expected as ethers are known to have a greater effect on increasing the RON than alcohols. The molar composition of the  (C# = 2) structural group also ranked highly in the model which is also not expected as the molar composition of i-Octane is not found to rank high in the model.

5.4.3.3 Parity

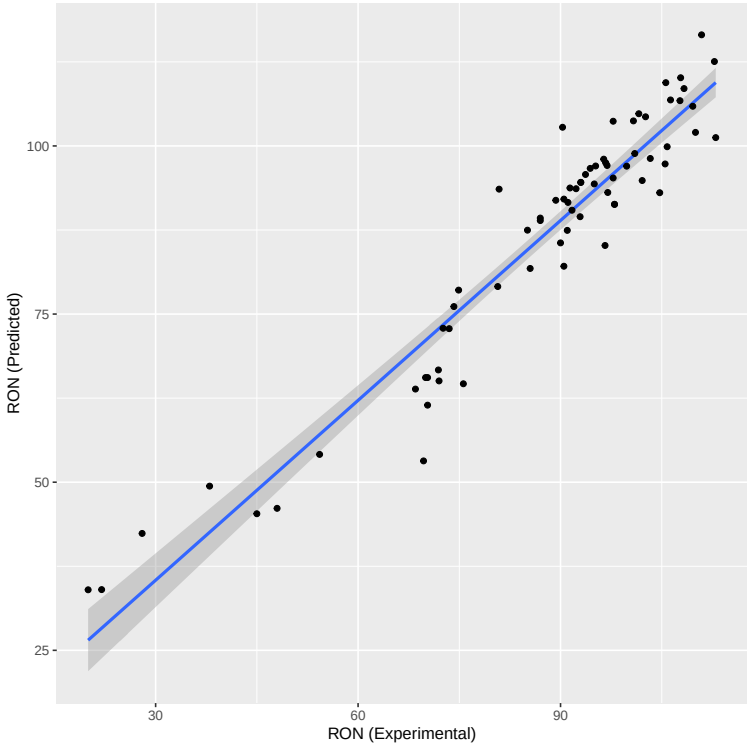


Figure 5.14: Parity plot of GBM model

Figure 5.15 above shows the parity plot for the GBM model. The region where the RON values are between 75 and 105 shows the most stability by having the smallest error margin. The RON prediction becomes more unreliable outside that range with a larger error margin for the RON values below 75. In the parity plot, it can be observed that the fit of the GBM model is poor with a wide scatter about the line $y = x$. When the RON is below 75, the

model appears to perform poorly as opposed to all the other models which observe poor performance at a RON of 60. The increase in the uncertainty range is a result of few data points in those ranges of RON.This shows that using an ensemble of weak learners which is done by GBM's is failing to learn all the facets of the data accurately. When the RON value is between 75 and 105, there is less scatter about the line $y = x$ but there are still a considerable number of points further away from the line thna in the other models, more specifically in comparison to the BART model which is also a tree based ensemble learning model. The model cannot be said to be reliable and parsimonious as the fit is rather poor across the entire range of RON values.

5.4.3.4 Residuals

Figure 5.15 below shows the residuals of GBM the model. This plot shows that the residuals have a fair scatter about the line $y = 0$ with a large uncertainty range. This fair scatter suggests that most of the variance is explained by the model and the residuals demonstrate a reasonable extent of homoskedasticity. Overall, the residuals can be said to be IID. Because of the observed homoskedasticity, the model can be said to be a good fit but not reliable because of the large error metrics and uncertainty range of the residuals. The GBM model shows that the model explains most of the variability in the data but the model performs poorly and this may be a result of there being too little data for the model to learn from.

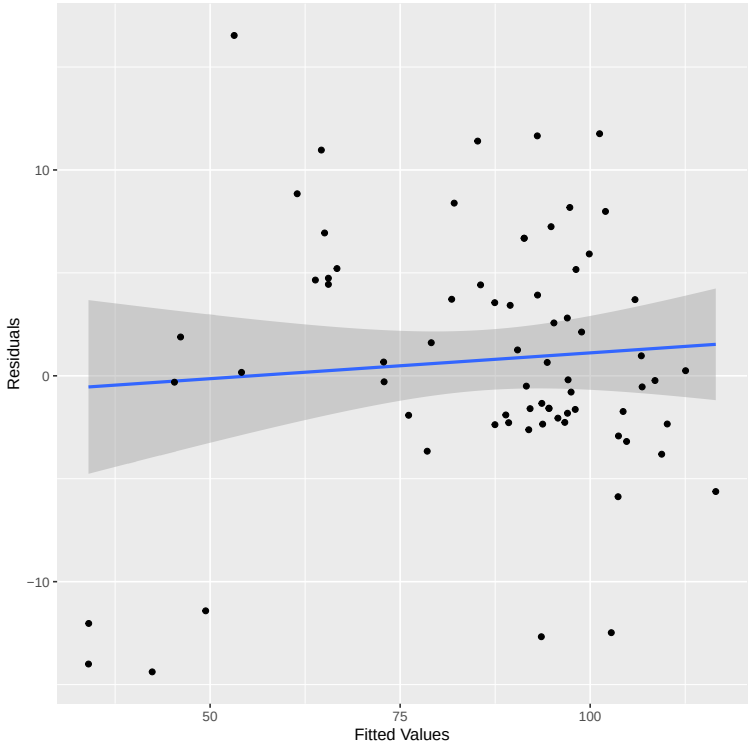


Figure 5.15: Residuals plot of GBM model

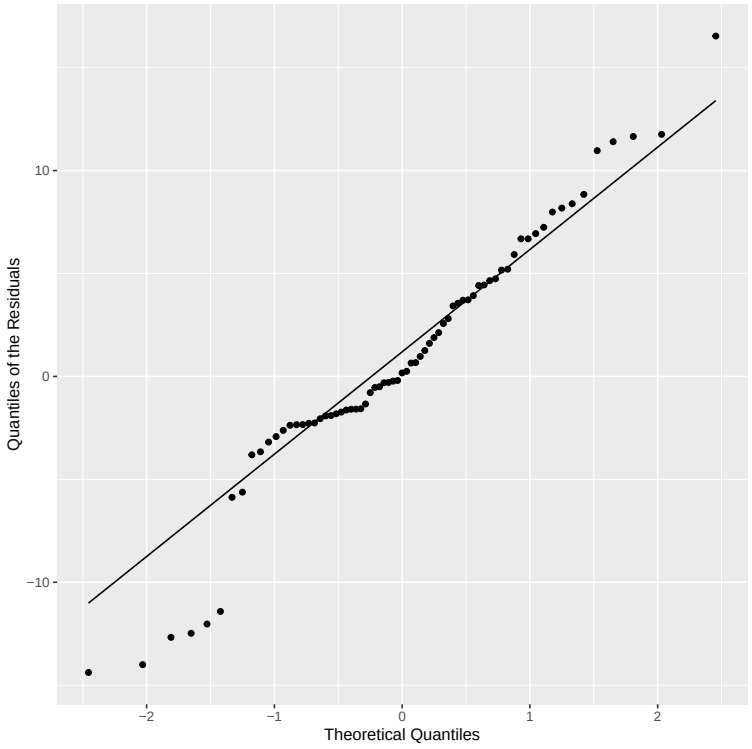


Figure 5.16: QQ Plot of GBM Model Residuals

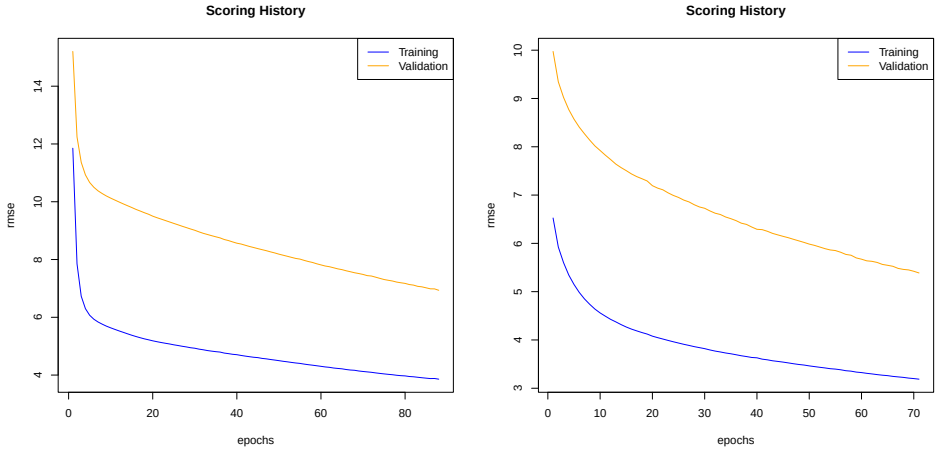
The QQ plot of the residuals of the GBM model can also be skewed left as opposed to the BART model which is skewed right.

5.5 Artificial Neural Networks (ANN)

5.5.1 Model Building

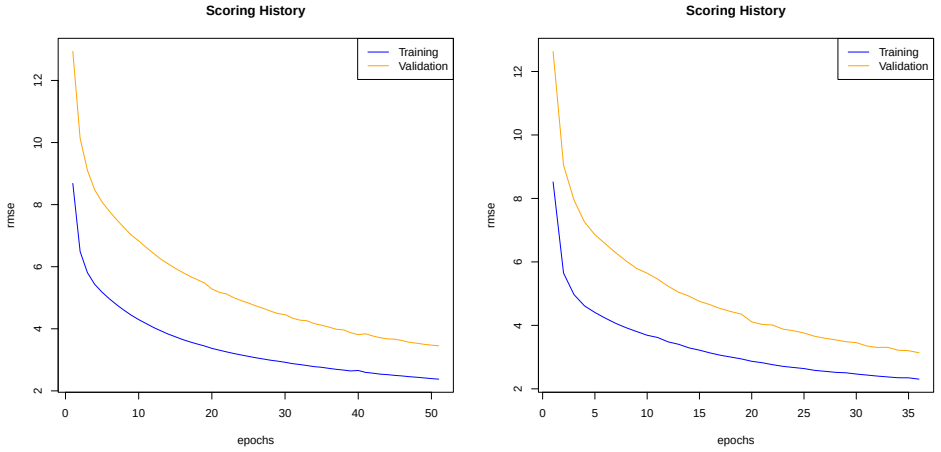
To develop an appropriate ANN model for this problem, 4 different Single Layer Feed-forward Neural Network (SLFN)'s were trained. Nielsen 2015 argues that for small data sets, SLFN's are appropriate. Considering the argument made by Nielsen 2015, it was expected that a SLFN would be sufficient since the matrix contained only 18 input variables. There were 210 training observations, 70 validation observations and 70 test observations.

All the ANN's trained in this study were trained using the Hyperbolic Tangent Activation Function (TanH). TanH was the choice of activation function because of its zero centeredness which makes it easier to model inputs with different correlations. The TanH function also has a smooth gradient and can allow for clear predictions to be made over a larger range of input values (Nielsen 2015). Each ANN was validated using a separate validation set.



(a) 3 Node Scores (b) 5 Node Scores

Figure 5.17: Scoring Plots for 3 and 5 node Single Layer Feed-forward Neural Network



(a) 10 Node Scores (b) 15 Node Scores

Figure 5.18: Scoring Plots for 10 and 15 node Single Layer Feed-forward Neural Network

Figures 5.17 and 5.18 above show the RMSE vs epochs plot for each of the SLFN's which were trained. An epoch represents the number of passes the entire training set has gone though the neural network in training the network till the error converges. The 10 and 15 node networks showed much faster convergence to a low error metric than both 3 and 5 node networks. In the 10 and 15 node networks, the SLFN is observed to stabilize within approximately 50 and 35 epochs respectively as opposed to 100 and 80 epochs in the 3 and 5 node SLFN's respectively. The 15 node SLFN appears to train the best as both the training and validation set RMSE's stabilize to similar values.

The scoring history of the RMSE metric is the type of learning curve chosen to diagnose the model performance of the SLFN's and how representative the data set is. The scoring history will be used to evaluate the model fit, generalization and how representative the training and validation sets are. A model has good generalization if the error metric being evaluated decreases to stability as the training epochs increase and if the error metric of the validation set is close to that of the training set. A data set is unrepresentative if the validation set achieves better performance than the training set and if the validation set has noisy movement around the training set.

The scoring history plots of the SLFN's trained are shown in Figures 5.17 and 5.18. All the models have a decreasing RMSE as the epochs increase, which shows that all the SLFN's are learning well. The SLFN's also have a steep decrease at first, then the RMSE starts to stabilize, with both the validation and training set RMSE's converging as the epochs increase. The 15 node SLFN learns fastest and this shows that it is the model with the best generalization.

5.5.2 Predictive Statistics

The diagnostics of the SLFN's architectures explored are summarized in Tables 5.16 below. It can be observed that the SLFN with 15 nodes in the hidden layer obtained the best performance in terms of accuracy and model fit.

Table 5.16: Single Layer Feed-forward Neural Network (SLFN) Predictive Statistics

Architecture	**Validation MSE**	**Test MSE**	**Adjusted** R^2
15	5.143	15.71	0.9713
10	6.910	11.26	0.9691
5	8.540	13.64	0.9590
3	13.17	16.54	0.9440

Meusinger and Moros 2001 trained a MLFN with 2 hidden layers using Nuclear Magnetic Resonance (NMR) data. The intensities of the carbon signals and their chemical shift values were recorded from the NMR spectra. To simplify the data and obtain a more manageable data set, the shift regions were split into 28 bins. These bins are an arbitrary division of similar regions of the same size. To add some qualitative information about the gasoline compounds, they included the presence of the **O**xygen, **R**ings, **A**romatics, aliphatic **C**hains and O**L**efins (ORACL) components which was the system used to categorize the gasoline components.

Although the approach taken by Meusinger and Moros 2001 is different to the approach taken in this study, it inspires the foundation of the approach taken. Meusinger and Moros

2001 argue that instead of using a hierarchical classification of the compounds by their types such as olefins and naphthenes, the presence of the major structural groups encoded as ORACL is sufficient to add the qualitative information of the gasoline. This study on the other hand, uses the molar composition of the gasoline reference fuels to add some qualitative information to the data.

The bins of the shift regions in the work done by Meusinger and Moros 2001 take into account the quantitative information of the structural composition of the gasoline compounds. In this study, the Paraffines, Iso-Paraffins, Aromatics, naphthenes, Olefins (PIANO) reference compounds and the additives present were broken down into their constituent structural group components and the molar composition of each of the groups was determined. Each of the two approaches has advantages and disadvantages. Binning the NMR spectra may result in some chemical structure groups being found in several bins, this overlap may affect some of the variable correlations and introduce bias to potential models. This approach would work well in a lab as NMR spectra may not be measured in real time on a process stream.

To train their ANN, Meusinger and Moros 2001 used an train: validation: test rations of 85%: 5%: 10% as opposed to the split used in this study of 60%: 20%: 20%. Their data also comprised of 320 pure compounds while this study builds a model on blend data of 350 different blends. The data in this study comprised of blends with 5 to 8 carbon compounds whereas Meusinger and Moros 2001 used data for 3 to 16 carbon compounds. The final diagnostics for the ANN trained by Meusinger and Moros 2001 are shown in Table 5.17 below. Meusinger and Moros 2001 observed that for the compound with extreme values of RON, their MLFN performed poorly and in one instance, a compound with a RON of 25 was predicted to have a RON of 79. They conclude that the MLFN that they train is not appropriate as the MSE needs to be below 0.3 measures of RON for the model to be used.

Table 5.17: Predictive Statistics of Meusinger and Moros 2001

Model	**Validation MSE**	**Test MSE**	**Adjusted** R^2
MLFN	20.1	15.7	0.933

Pasadakis, Gaganis, and Foteinopoulos 2006 took an approach more closely aligned to process engineering. By taking the volumetric flow of process streams entering the blending circuit, they determined the compositions of each of the types of compounds present in the blend. The process streams and the respective types of compounds present in them are listed below.

1. Fluidized cracking: Olefins.
2. Reforming: Naphthenes.
3. Isomerization: Isoparaffins.
4. Alkylation: Paraffins.
5. Dimersol: Olefins.
6. Butanes: Additives.
7. Oxygenates: Alcohol and ether additives.

In addition to using the compositions of the above process fractions, Pasadakis, Gaganis, and Foteinopoulos 2006 added arbitrary RON values of each of the components listed above to the input variables used in their model. They proceeded to train a MLFN with the architecture and diagnostics described in table 5.18 below.

Table 5.18: Predictive Statistics of Pasadakis, Gaganis, and Foteinopoulos 2006

Hidden Layer Architecture	**Validation MSE**	**Test MSE**	**Adjusted R^2**
5, 3	0.0289	0.0324	0.989

The data set used by Pasadakis, Gaganis, and Foteinopoulos 2006 had 173 observations and 12 input variables. This was a small data set in comparison to the one used in this study. Their data collection was done over a period of a year to obtain a representative data set which could be used to train a model accurately. Their MLFN outperformed all the ANN's trained in this study and this can be attributed to a larger data representation in the extreme values of the RON. They also conclude that the volumetric compositions of olefins, naphthenes and iso-paraffins have the highest impact on the RON. The only variable which had a high ranking of variable importance in both this study and the work done by Pasadakis, Gaganis, and Foteinopoulos 2006 was the composition of the iso-paraffins in the gasoline blends.

Kubic et al. 2017 built a Quantitative Structure-Property Relationship (QSPR) model using the structural groups present for estimating RON of hydrocarbons and oxygenated organic compounds such as alcohols and ethers. Their objective was to express RON of each organic compound as a function derived from molecular structure by dividing organic molecules in to their constituent structural groups.

The data set used by Kubic et al. 2017 consisted of 218 compounds. This study considered the group contribution method used by Kubic et al. 2017 and extended it to be used to determine the RON of blends. They did not sample the data themselves but collated the data from multiple journals as was done in the data collection of this study. The data set they used comprised of a wider range of compounds as they considered additional compounds such as esters, furans, aldehydes and ketones in addition to the PIANO group compounds. Their objective was to build a parsimonious model which worked well to predict RON for the oxygenated compounds, while this study aimed to train a model which could be used to consider the group interactions resulting from the presence of the oxygenated compounds. The data set used by Kubic et al. 2017 had RON in the range 80 to 120. This suggests that their model would also preform poorly in predicting the RON outside this range as observed in the results of this study.

The biggest drawback when training an ANN is the lack of guaranteed convergence to a global minimum (Kubic et al. 2017). To ensure convergence, Kubic et al. 2017 started with a 4 hidden node SLFN using a logistics activation function using sub samples of the data set. Their data set had 38 input variables. They validated their model using a 10 fold validation scheme on the training data. After validating the model, Kubic et al. 2017 tuned the model to be an 8 node SLFN which obtained as R^2 of 0.94. The SLFN's trained in this model achieved a better fit with but had less accuracy as the MSE's were higher in the models trained in this study. To train their model better, Kubic et al. 2017 suggests that more data in the lower values of RON be added to the data set to ensure the model is more robust. Although Kubic et al. 2017 trained their model for pure compounds, their approach

to use the molecular structure to predict RON was extended to model the RON of blends. This approach allows for the training of a model which can use the information such as inter molecular forces, which is embedded in the structural groups present in the blend.

Table 5.19: Predictive Statistics of Kubic et al. 2017

Hidden Layer Architecture	**Validation MSE**	**Test MSE**	**Adjusted R^2**
8	6.9	7.2	0.94

In a similar approach to Meusinger and Moros 2001, Abdul-Gani et al. 2018 predicted the RON of blends and pure compounds using NMR spectra. They derived the structural groups from the PIANO compounds present in the blends. They also determined a property called the branching index to capture the extent of branching in the blends. As discussed in the literature review, the concept of branching index would be useful for pure compound predictions and not for blends. Using the branching index would also present some bias in the model as the branching index is determined directly form the composition of the structural groups in the compound.

Abdul-Gani et al. 2018 trained their model using a data set consisting of 281 entries. Their data set is biased as these 251 entries are made up of both blends and pure compounds. the data set is transformed from compound to structural group composition. This may introduce some bias in the model as the ANN may not be able to capture the information regarding molecular interactions in the blends. Abdul-Gani et al. 2018 then added the molecular mass of the compounds and in the case of the blends it was a weighted molecular weight. this resulted in a data set with 9 input variables; 7 structural groups, molecular weight and branching index. This study on the other hand, used structural group and compound compositions as input variables. The composition of the compounds would in a sense have the role of the branching index as different blends may have the same structural group compositions with different compositions.

Abdul-Gani et al. 2018 used an 80/20 train to test split and 10 fold cross validation using the training set. In this study, a separate validation set was used to allow the model to learn better and achieve better results when used on unseen data. They trained a MLFN with the architecture and diagnostics shown in Table 5.20 below.

Table 5.20: Model Diagnostics of Abdul-Gani et al. 2018

Hidden Layer Architecture	**Validation MSE**	**Test MSE**	**Adjusted R^2**
540, 314	-	4.84	0.99

The architecture shown in Table 5.20 has many hidden nodes. This suggests that the model may be susceptible to over-fitting because the ANN has only 9 input variables and too many hidden nodes.

The final model explored as a deep learning architecture was trained by Wang, Yang, and Kalivas 2020. Wang, Yang, and Kalivas 2020 trained a structure called Extreme Learning Machine (ELM). an ELM is a specialized SLFN and is unique in the way described in Extreme Learning Machines. The data set used by Wang, Yang, and Kalivas 2020 had a 60 observations with 401 input variables which were obtained from NMR spectra. The best performing ELM was the Self Adaptive Evolutionary Extreme Learning Machine (SaDE-ELM) with the

diagnostics and architecture shown in Table 5.21 below. These results show that a neural network can obtain good prediction with few hidden nodes relative to the input variables. Their SaDE-ELM achieves better accuracy than the best performing SLFN trained in this study but the model fit was observed to be better in the SLFN's trained in this study. This is expected because of the different styles of learning employed by the two models as discussed in Extreme Learning Machines.

Table 5.21: Model Diagnostics of Wang, Yang, and Kalivas 2020

Hidden Layer Architecture	**Validation MSE**	**Test MSE**	**Adjusted** R^2
20	0.22	0.21	0.931

5.5.3 Model Diagnostics

5.5.3.1 Analysis of Variance (ANOVA)

Tables 5.22 to 5.25 below show the ANOVA results of each of the SLFN's trained in this study. It can be observed that we reject the null hypothesis that the parameters for each of the variables was equal to zero. The observations of the ANOVA of the SLFN's shows that only the confounded variables x_{15} to x_{17} have zero valued parameters.

Table 5.22: 3 node SLFN ANOVA

	Mean Sq	**F value**	**Pr(>F)**
x_1	46	4.193	0.045
x_2	12995	1193	0
x_4	9658	887	0
x_{12}	622	57.1	0
x_{13}	9	0.886	0.355
x_{14}	19	1.72	0.194
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	39	3.58	0.063

Table 5.23: 5 node SLFN ANOVA

	Mean Sq	**F value**	**Pr(>F)**
x_1	72	8.51	0.005
x_2	14578	1714	0
x_4	10745	1263	0
x_{12}	560	65.8	0
x_{13}	9	1.06	0.306
x_{14}	0	0.002	0.966
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-

Table 5.23 continued from previous page

	Mean Sq	F value	Pr(>F)
x_{18}	29	3.468	0.067

Table 5.24: 10 node SLFN ANOVA

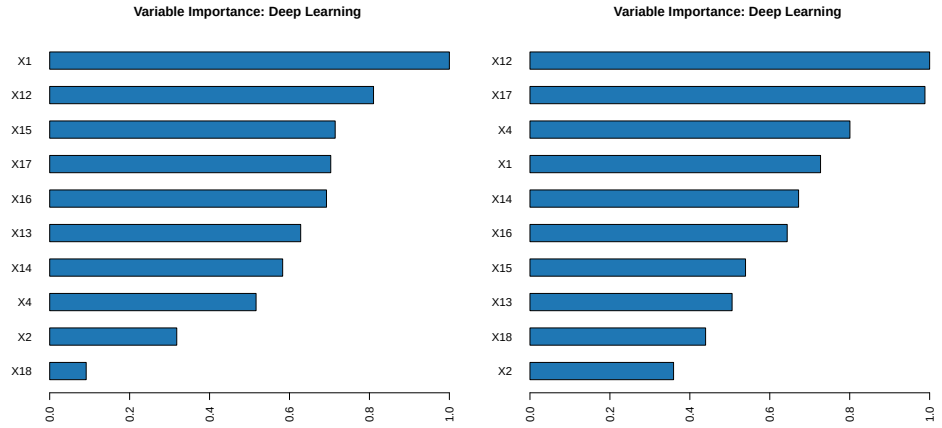
	Mean Sq	F value	Pr(>F)
x_1	21	4.227	0.044
x_2	15463	3063	0
x_4	11640	2305	0
x_{12}	609	120.6	0
x_{13}	1	0.168	0.683
x_{14}	0	0.023	0.881
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	24	4.853	0.031

Table 5.25: 15 node SLFN ANOVA

	Mean Sq	F value	Pr(>F)
x_1	29	7.345	0.009
x_2	15495	3915	0
x_4	12461	3149	0
x_{12}	632	159.7	0
x_{13}	0	0.047	0.830
x_{14}	3	0.829	0.366
x_{15}	-	-	-
x_{16}	-	-	-
x_{17}	-	-	-
x_{18}	17	4.231	0.044

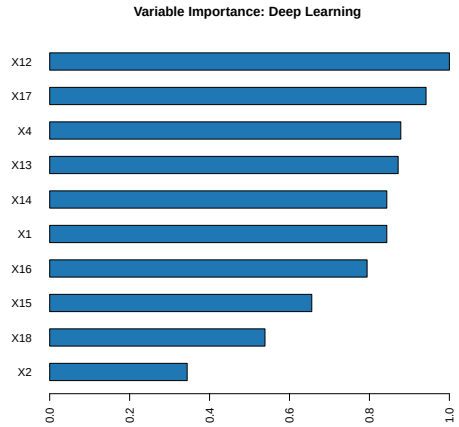
5.5.3.2 Variable Importance

The importance of each variable was determined for each ANN which was trained. the variable importances of the SLFN and the MLFN are shown in Figure 5.19 below.

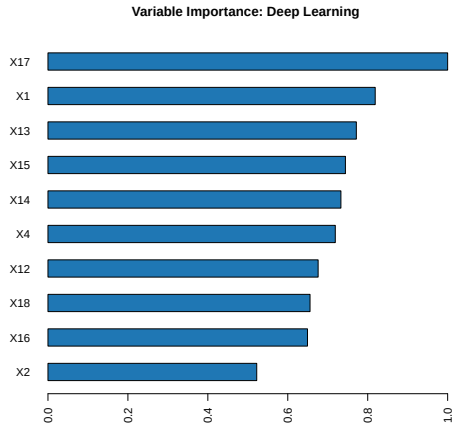


(a) 3 Node Variable Importance

(b) 5 Node Variable Importance



(c) 10 Node Variable Importance



(d) 15 Node Variable Importance

Figure 5.19: Variable Importance plots for Single Layer Feed-forward Neural Network's

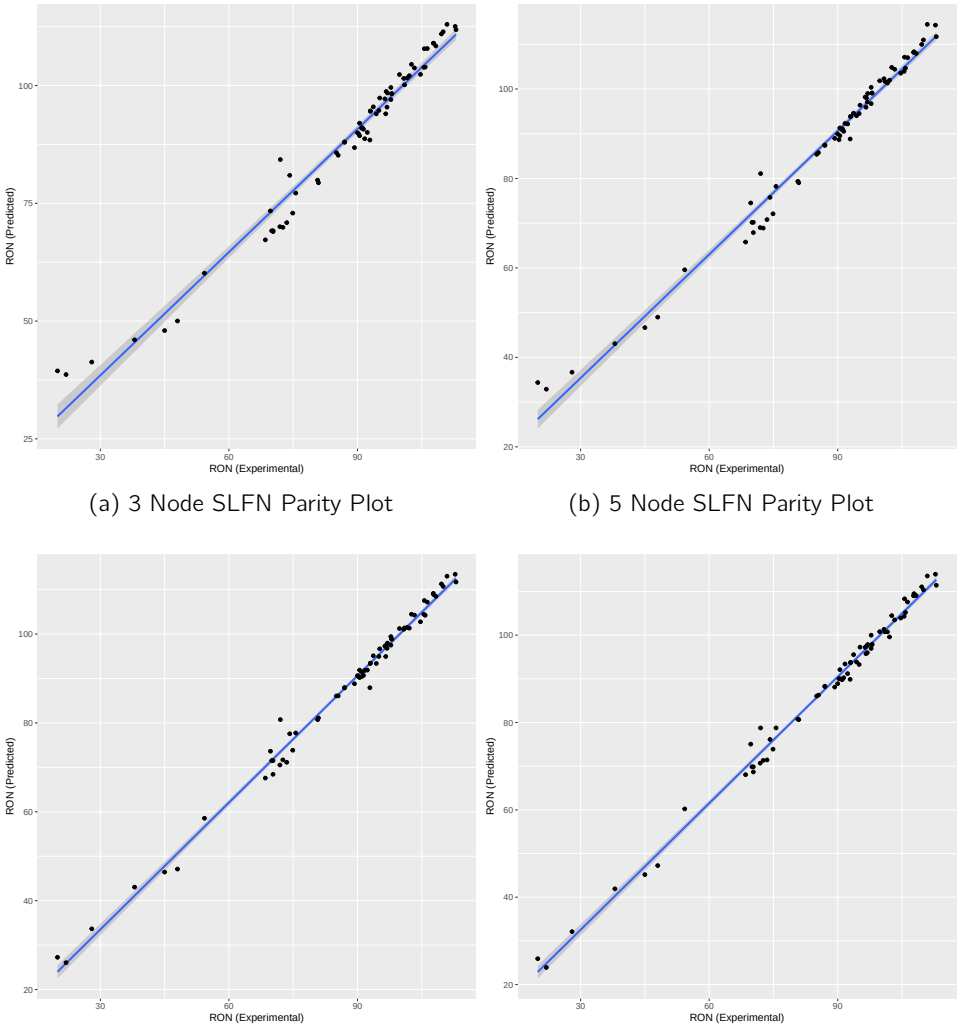
Figure 5.19 above shows that the variables x_{17} and x_{12} had a high significance in the SLFN models. This observation suggests that the molar compositions of the ethanol and n-heptane were generally the most significant in the prediction of RON. Which is an odd observation as the composition of ethanol was actually omitted from the ANOVA. In the 3 node SLFN, the n-heptane composition ranked second below the molar composition of the $-\!\!-CH_3$. While the composition of the ETBE was less significant in the 3 node SLFN than in the rest of the SLFN's, it ranked fourth in the 3 node SLFN which is still a high ranking. The variable x_{12} which corresponds to the molar composition of i-octane was found to have the highest importance in ech of the SLFN's except for the 15 node SLFN. The variable x_1 corresponding to $-\!\!-CH_3$ structural also ranked highly in all the SLFN's, this can be expected since all the compounds contained in the blend except cyclopentane contain this structural group. The variable x_2 which corresponds to the structural group $-\!\!-CH_2-\!\!-$ can be observed to consistently rank the lowest. This shows that this structural group has in insignificant impact on the RON in the SLFN models.

Figure 5.19 show that the following variables have a consistently high importance in all of the SLFN models trained.

- x_{12}: molar composition of n-heptane.
- x_{17}: molar composition of ethanol.
- x_1: molar composition of $\_\_CH_3$.
- x_{14}: molar composition of toluene.

The molar compositions of n-heptane, ethanol land toluene were found to have the consistently high ranking in each of the SLFN models trained. This is a reasonable observation because n-heptane is a major component in gasoline and is in many cases found to have the greatest composition and can be observed to rank highly even in the other models as well. Ethanol and toluene on the other hand are additives which are used to ensure that the fuel burns effectively in combustion engines. These compounds would therefore be expected to rank highly in the models trained as adding these in small quantities will significantly increase the RON. The molar composition of the $\_\_CH_3$ group on the other hand also had a high ranking. This can be alluded to the structural group being contained in all of the compounds which have been found to have a high ranking.

5.5.3.3 Parity



(a) 3 Node SLFN Parity Plot

(b) 5 Node SLFN Parity Plot

(c) 10 Node SLFN Parity Plot

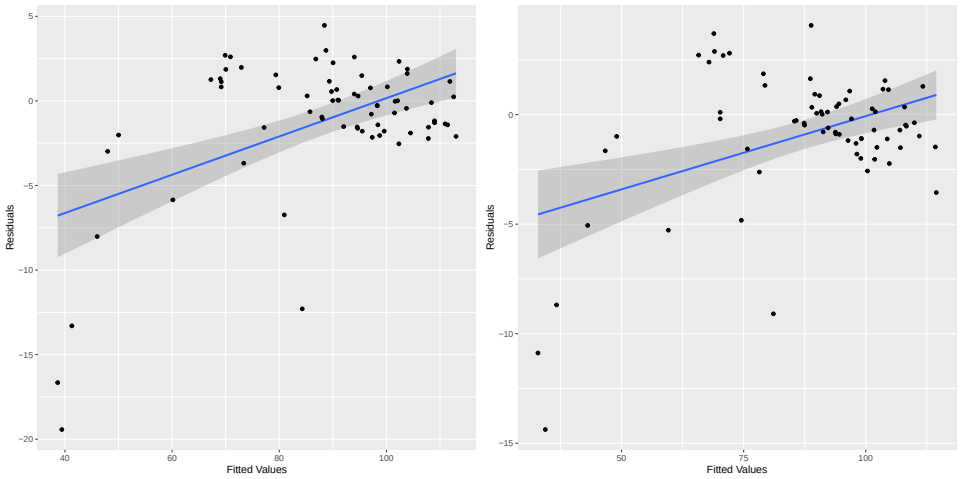
(d) 15 Node SLFN Parity Plot

Figure 5.20: Parity Plots for the Single Layer Feed-forward Neural Network's

In the parity plots of the SLFN's shown in Figure 5.20 above, the uncertainty ranges are narrowest when the experimental RON is between 80 and 90. In the extremities below 80 and above 90, the SLFN's have an increasing uncertainty range. The uncertainty ranges of the models can be observed to be wider when the RON is below 80 than when the RON is above 90. The uncertainty ranges can also be observed to decrease when the number of hidden nodes increases. In the ranges were the uncertainty ranges of the parity plots can be observed to increase, there were few data points. Which shows that the SLFN's learned poorly in the ranges with few data points as expected. The parity plots also show that the SLFN models were a good fit to the data which supports the observations made from the predictive statistics although the models had poor predictive accuracy.

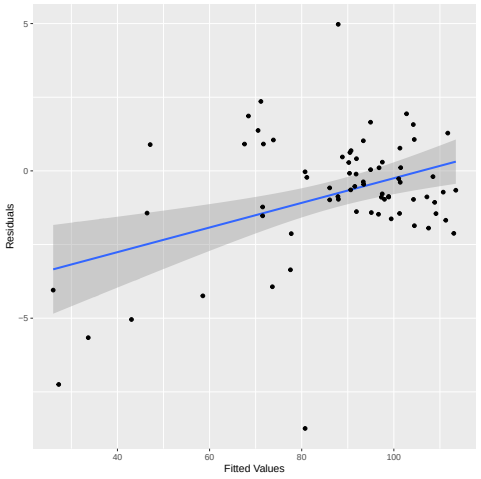
5.5.3.4 Residuals

Figure 5.21 below shows the residual plots for the SLFN models trained. All the SLFN models trained show large uncertainty ranges in their residual plots. The uncertainty range of the residuals can be observed to increase when the RON decreases below 80 and increases from 90 similar to the observations made in the parity plots. It can also be observed that the residuals of each of the models are not IID and demonstrate heteroskedasticity. The SLFN models are only appropriate for the RON ranges between 80 and 90 and are therefore not a good fit and are not parsimonious.

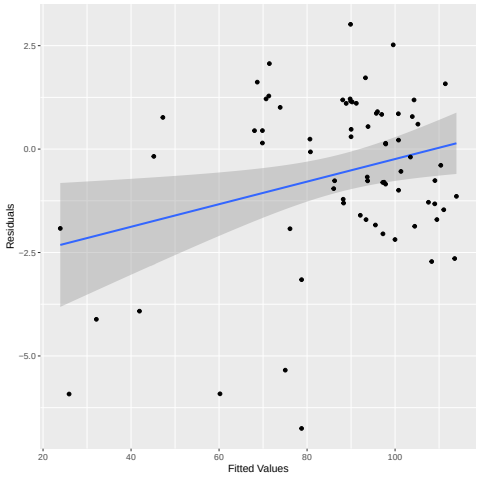


(a) 3 Node SLFN Residual Plot

(b) 5 Node SLFN Residual Plot



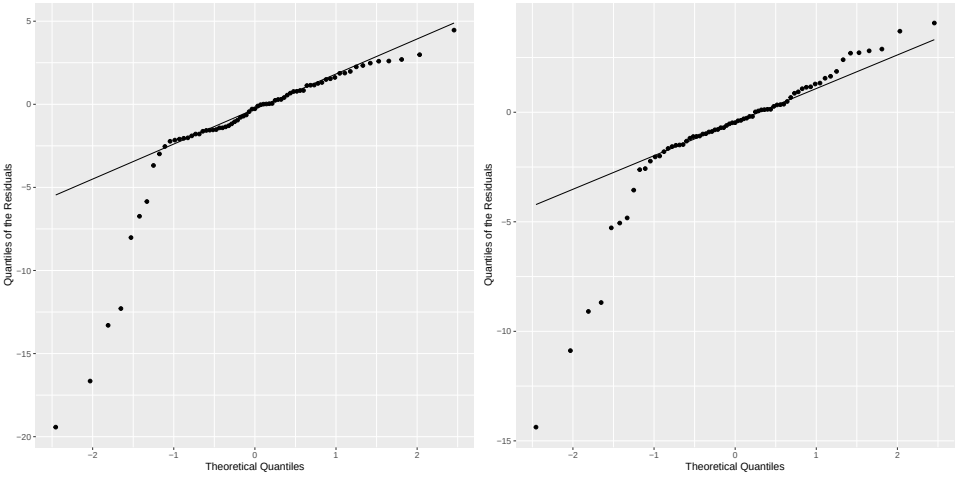
(c) 10 Node SLFN Residual Plot



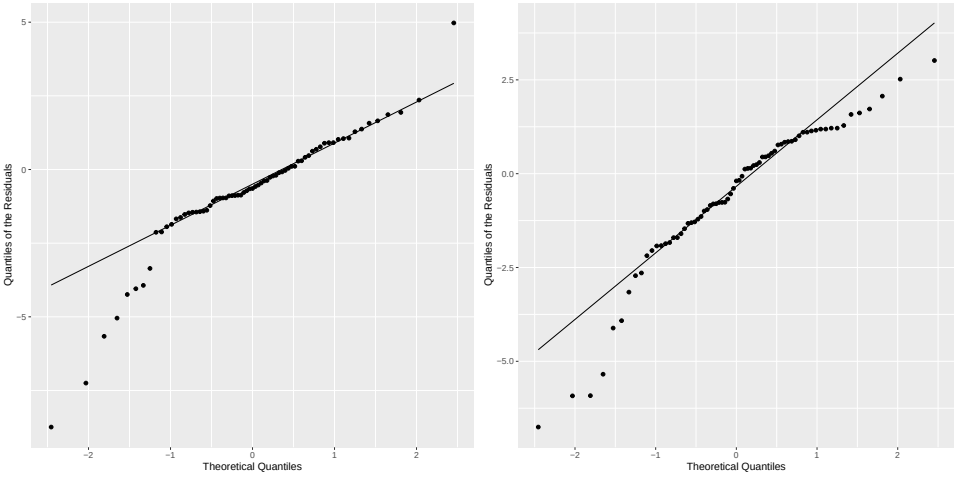
(d) 15 Node SLFN Residual Plot

Figure 5.21: Residual Plots for Single Layer Feed-forward Neural Network's

The QQ plots of the residuals of each of the SLFN models are shown in Figure 5.22 below. It can be observed that the residuals are left skewed and show non-normality. This further suggests that the residuals of the SLFN models are not IID.



(a) 3 Node SLFN QQ Plot of Residuals (b) 5 Node SLFN QQ Plot of Residuals



(c) 10 Node SLFN QQ Plot of Residuals (d) 15 Node SLFN QQ Plot of Residuals

Figure 5.22: Residual Plots for Single Layer Feed-forward Neural Network's

5.6 Model Comparison

5.6.1 Model Building

To train the regression model, two scenarios where considered; one with causal interactions and the other without interactions. Both scenarios had cases with and without the mass balance constraint imposed on the compound composition. Interactions were considered to allow the regression models to learn the data more accurately while preventing over-fitting. All the regression models performed well and obtained similar error metrics as shown in Table 5.7. The constrained MLR models were observed to have a slightly less fit and higher MSE's than their unconstrained counterparts. The interaction MLR models were observed to obtain better fit and model accuracy than their counterpart with no interaction. We can therefore observe that adding the mass balance constraint served as regularization.

The Bayesian Additive Regression Trees (BART) model learnt well by converging to a minimum error when 200 trees were used as shown in Figure 5.7. This shows that the model learns from the data well and fairly quickly. The model had a test MSE of 13.74 which is higher as than that of the constrained interaction MLR which was 7.54. The interaction model on the other hand obtained an adjusted R^2 of 0.981 as opposed to the BART model which obtained 0.983. While the BART model had a similar fit to the interaction MLR, the interaction MLR model out-performed the BART model.

Because of the way Gradient Boosting Machiness work, one can say that they performed well as they are naturally slow because of their method of training the ensemble model iteratively by adding a weak learner with each iteration. The GBM model converged at 2000 trees as opposed to the BART model which converged at 200 trees. This shows that the BART model learns better from the data using fewer learners. The BART model also outperforms the GBM model from both a model accuracy and model fit perspective.

When training Artificial Neural Networkss, the scoring history of each of the epochs is used as the metric to evaluate how well the model is learning. A single epoch can be defined as a single iteration which the entire training or validation set would have gone through the ANN during the training process. It can be observed that of the Single Layer Feed-forward Neural Network's, the 15 node model learns fastest in 37 iterations and has the smallest discrepancy between the validation and training set RMSE. The 10 node SLFN however, obtains the smallest test MSE and can be said to be the bets performing SLFN since its model fit is not significantly smaller than that of the 15 node SLFN.

Given the validation metrics and the learning iterations for each model, the following order of ease of training can be concluded.

1. Multiple Linear Regression.
2. Bayesian Additive Regression Trees.
3. Artificial Neural Networks.
4. Gradient Boosting Machines.

5.6.2 Predictive Statistics Evaluation

Table 5.26 below shows that when considering the test MSE and the fit of the model which is measured by the adjusted R^2 value, the interaction MLR and the BART models perform

best. While the SLFN and the GBM models have the lowest performance when considering the test MSE. This large error may be a result of the models failing to make good predictions on the training data with few points. All the models however, appear to have a reasonable fit to the data as the adjusted R^2 of all the model exceeds 0.9.

Table 5.26: Diagnostics of Best performing Models

Model	**Test MSE**	**Adjusted R^2**
Interaction MLR with mass balance constraint	7.54	0.981
BART	13.74	0.983
10 node SLFN	11.26	0.969
GBM	38.12	0.917

When choosing a model an important diagnostic to consider is the MSE as this shows the magnitude of the absolute error scaled through the entire range of the RON. The MSE is reported as a constant measure throughout the entire range. Considering the models in order of their uncertainty ranges, the models have the following order of performance in descending order.

1. Interaction Multiple Linear Regression with mass balance constraint.
2. Bayesian Additive Regression Trees.
3. 10 node Single Layer Feed-forward Neural Network.
4. Gradient Boosting Machines.

5.6.3 Model Diagnostics Evaluation

5.6.3.1 Analysis of Variance (ANOVA)

The variables x_{15}, x_{16}, x_{17} were found to consistently fail to reject the hypothesis that the model parameters for the variables were all equal to zero. Thos shows that these variables were confounded and their parameters could not be estimated. These variables correspond to the components listed below.

1. x_{15}: Cyclopentane.
2. x_{16}: 1-Hexene.
3. x_{17}: Ethanol.

The molar composition of cyclopentane, 1-Hexene and ethanol and its structural groups can be observed to have parameters which cannot be estimated and their p-values are effectively 0. The observation that they would reject the null hypothesis that all the variables have parameters which cannot be estimated is expected as these variables would be spuriously correlated as they are confounded.

5.6.3.2 Variable Importance

The variables x_{12}, x_1, x_2, x_{17} and x_{14} which correspond to the molar compositions of n-heptane, $-\!\!-CH_3$, $-\!\!-CH_2-\!\!-$ ethanol and toluene respectively, were found to rank highly in all the models. N-heptane and its substituent structural groups; $-\!\!-CH_2-\!\!-$ and $-\!\!-CH_3$ were found to consistently rank highly as n-heptane is the components with generally the largest composition in the blends. In addition to this, the molar composition of $-\!\!-CH_3$ also ranks highly and this is expected as all the components contain this structural group except for cyclopentane. Ethanol and ETBE are additives which are added to the blends to increase RON and are expected to rank highly in all the models as observed. Given the rankings of each of the models, the variables with the highest importance can be said to have the following rankings.

1. x_{12}: N-Heptane.
2. x_1: $-\!\!-CH_3$.
3. x_2: $-\!\!-CH_2-\!\!-$.
4. x_{17}: Ethanol.
5. x_{14}: Toluene.

5.6.3.3 Parity Analysis

The parity plots of each of the models were used to visually assess the fit of each of the models trained in this study. The parity plots of the regression models in Figure 5.4 show a good general fit of the data with no outliers and the points lying close to the line $y = x$. This shows that the regression models had a good fit to the data.

The parity plot of the BART model shows a generally good fit in Figure 5.9 but starts to have an increasing uncertainty range as the value of RON falls below 60. The increase in the uncertainty range as the RON falls below 60 is similar to that of the MLR models and no outliers can be observed for the BART model. The GBM model on the other hand has a much larger scatter about the line $y = x$ than both the MLR and BART models, suggesting that this model under-performs the BART model and the MLR models. The poor fit of the GBM model is evident from in the increase in the uncertainty range which is larger than in the MLR and BART models.

Figure 5.20 shows the parity plot of the the SLFN's trained in this study. It can be observed that as the complexity of the SLFN's increases, the fit of the SLFN models increases as shown by a narrower uncertainty range. The problem however, is that outside the range where the RON is between 80 and 90, the SLFN models can be observed to have poor performance as there is very little data for the models to learn from outside of this range. The choice of the model with the best fit will therefore be a trade off between these two observations. These SLFN's can be observed to have a better fit than the GBM model as they have less scatter but have a poorer fit than the MLR and BART models. The order of fit from the looking at the parity plots is as follows.

1. Multiple Linear Regression.
2. Bayesian Additive Regression Trees.

3. Artificial Neural Networks.

4. Gradient Boosting Machines.

5.6.3.4 Residual Analysis

The residual analysis was carried out to assess the models which have the least variance by determining the extent of homoskedasticity in the models. Figure 5.5 shows the residual plots of the MLR models. The residuals have the narrowest uncertainty ranges when the RON is between 80 and 90. The increasing uncertainty ranges when the RON decreases below 80 can be supported by the left-skewed distribution of the residuals as shown in Figure 5.6. The residuals of the MLR models were non IID and did not demonstrate homoskedasticity.

The BART model can be observed to have a poorer distribution of residuals as shown in Figure 5.10 about the line $y = 0$ than the MLR models. This is a greater extent of heteroskedasticity than the MLR models. The BART model shows right-skewed distribution of residuals as opposed to the MLR models which had left-skewed residuals. The GBM model was observed to have a larger scatter about the horizontal axis than the BART and MLR models. The shape of the residual plot of the GBM model in Figure 5.15 shows the highest extent of homoskedasticity in the residuals. The scatter about the line $y = 0$ however, is very large and the distribution of the residuals can be observed to be almost IID with a slight left skew.

The SLFN's can be observed to have a wide range of uncertainty when the RON is above 90 and below 80. The residuals where also found to be left-skewed in a similar fashion to the MLR and GBM models and were not observed to demonstrate and are therefore not IID. All the models can be observed to be left-skewed with the exception of the BART model which was right-skewed. Considering the extent of normality of the residuals it can be said that the models have the following order of performance.

1. Gradient Boosting Machines.

2. Multiple Linear Regression.

3. Bayesian Additive Regression Trees.

4. Single Layer Feed-forward Neural Network.

6 Conclusions

Considering all the metrics discussed in the previous chapter, the overall performance of the models trained in this study is listed below. The Interaction MLR with mass balance constraint will be taken as the final model to be used in the process control.

1. Interaction MLR with mass balance constraint.
2. Bayesian Additive Regression Trees.
3. 10 node Single Layer Feed-forward Neural Network.
4. Gradient Boosting Machines.

The hypothesis stated that the molar compositions of i-Octane and the additives will rank highest in variable importance followed by their substituent structural groups and n-Heptane would rank the lowest. Considering the results of the variable importance of each of the models trained, one can conclude that the order of the significance of the highest ranking models is as follows.

1. x_{12}: N-Heptane.
2. x_1: $-\!\!-CH_3\cdot$
3. x_2: $-\!\!-CH_2-\!\!-\cdot$
4. x_{17}: Ethanol.
5. x_{14}: Toluene.

It can be observed that the structural groups considered in this study were all the same size. Contrary to the hypothesis, the molar composition of n-heptane and its substituent structural groups was found to have the highest variable importance. This suggests that the negative effect n-heptane has on the RON is larger than the positive effect of the additives on the RON.

7 Recommendations and Future Work

The biggest constraint which limited the performance of the models trained in this study was the lack of representative data across the entire range of RON values. To ensure the models are trained more easily and perform better, future data samples need to be more representative to ensure the bias in the models is minimized. More representative data points in low RON ranges may be sample from low RON streams on blending circuits or in chemical labs. Future models should also take into consideration the correlations and potential dimension reduction techniques to make the model built simpler to interpret.

Another recommendation to minimize bias would be to attempt to make a model which is a hybrid of phenomeno-logical models and the MLR model. This will result in the models explaining more of the variance in the data and reduce the residuals. The models trained by making a hybrid model may result in more homoskedasticity and IID residuals.

Since the model built in this study is to be used to inform the fuel quality of the blend, the next stage of the study would be to design the integration of the model into the plant-wide Advanced Process Control.

References

Abdul-Gani, Abdul-Jameel et al. (2018). "Predicting octane number using nuclear magnetic resonance spectroscopy and artificial neural networks". In: *Energy & fuels* 32.5, pp. 6309–6329.

Al Fahemi, Jabir, Nahla Albis, and Elshafie Gad (2014). "QSPR models for octane number prediction". In: *Journal of Theoretical Chemistry* 2014.

Albahri, Tareq A (2003). "Structural group contribution method for predicting the octane number of pure hydrocarbon liquids". In: *Industrial & engineering chemistry research* 42.3, pp. 657–662.

Amirante, Riccardo et al. (2017). "Laminar flame speed correlations for methane, ethane, propane and their mixtures, and natural gas and gasoline for spark-ignition engine simulations". In: *International Journal of Engine Research* 18.9, pp. 951–970.

Baldi, Pierre (2012). "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49.

Blurock, Edward (1995). "Automatic learning of chemical concepts: Research octane number and molecular substructures". In: *Computers & chemistry* 19.2, pp. 91–99.

Broomhead, D. S. and D. Lowe (1988). "Multivariable functional interpolation and adaptive networks, complex systems, vol. 2". In.

Cancino, Leonel et al. (2011). "Ignition delay times of ethanol-containing multi-component gasoline surrogates: Shock-tube experiments and detailed modeling". In: *Fuel* 90.3, pp. 1238–1244.

Cao, Jiuwen, Zhiping Lin, and Guang-Bin Huang (2012). "Self-adaptive evolutionary extreme learning machine". In: *Neural processing letters* 36.3, pp. 285–305.

Chipman, Hugh A, Edward I George, Robert E McCulloch, et al. (2010). "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* 4.1, pp. 266–298.

Church, Russell M (1979). "How to look at data: A review of John W. Tukey's Exploratory Data Analysis". In: *Journal of the experimental analysis of behavior* 31.3, p. 433.

Doicin, Bogdan and I Onutu (2014). "Octane number estimation using neural networks". In: *Revista de Chimie* 65, pp. 599–602.

Friedman, Jerome (2001). "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics*, pp. 1189–1232.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York.

Friedman, Jerome and Werner Stuetzle (1981). "Projection pursuit regression". In: *Journal of the American statistical Association* 76.376, pp. 817–823.

González, Sandra Correa (Sept. 2019). "Modeling the effect of blending multiple components on gasoline properties". MA thesis. Aalto Univeristy School of engineering.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Himmelblau, David (2000). "Applications of artificial neural networks in chemical engineering". In: *Korean journal of chemical engineering* 17.4, pp. 373–392.

— (2008). "Accounts of experiences in the application of artificial neural networks in chemical engineering". In: *Industrial & Engineering Chemistry Research* 47.16, pp. 5782–5796.

Huang, Chen, Valeri Golovitchev, and Andrei Lipatnikov (2010). *Chemical model of gasoline-ethanol blends for internal combustion engine applications*. Tech. rep. SAE Technical Paper.

Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew (2004). "Extreme learning machine: a new learning scheme of feedforward neural networks". In: *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*. Vol. 2. IEEE, pp. 985–990.

Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew (2006). "Extreme learning machine: theory and applications". In: *Neurocomputing* 70.1-3, pp. 489–501.

Ibrahim, Hassan Al-haj and Mourhaf Al-Kassmi (1997). "Correlation between the octane number of motor gasoline and its boiling range". In: *Journal of King Saud University-Engineering Sciences* 9.2, pp. 311–317.

James, Gareth et al. (2013). *An introduction to statistical learning*. Vol. 112. Springer.

Jin, David and Sally Lin (2012). *Advances in computer science and information engineering*. Springer.

Khan, Ahmed Faraz, Philip John Roberts, and Alexey Burluka (2019). "Modelling of Self-Ignition in Spark-Ignition Engine Using Reduced Chemical Kinetics for Gasoline Surrogates". In: *Fluids* 4.3, p. 157.

Kubic, William et al. (2017). "Artificial neural network based group contribution method for estimating cetane and octane numbers of hydrocarbons and oxygenated organic compounds". In: *Industrial & Engineering Chemistry Research* 56.42, pp. 12236–12245.

Lee, Sanguk et al. (2013). "Random forest as a potential multivariate method for near-infrared (NIR) spectroscopic analysis of complex mixture samples: Gasoline and naphtha". In: *Microchemical Journal* 110, pp. 739–748.

Li, jing et al. (2012). "Brief Introduction of Back Propagation (BP) Neural Network Algorithm and its Improvement". In: *Advances in Computer Science and Information Engineering* 2, pp. 553–558.

Liang, Nan-Ying et al. (2006). "A fast and accurate online sequential learning algorithm for feedforward networks". In: *IEEE Transactions on neural networks* 17.6, pp. 1411–1423.

Meusinger, R and R Moros (2001). "Determination of octane numbers of gasoline compounds from their chemical structure by 13C NMR spectroscopy and neural networks". In: *Fuel* 80.5, pp. 613–621.

Morgan, Neal et al. (2010). "Mapping surrogate gasoline compositions into RON/MON space". In: *Combustion and Flame* 157.6, pp. 1122–1131.

Nielsen, Michael (2015). *Neural networks and deep learning*. Vol. 2018. Determination press San Francisco, CA, USA:

Pasadakis, Nikos, Vassilis Gaganis, and Charalambos Foteinopoulos (2006). "Octane number prediction for gasoline blends". In: *Fuel Processing Technology* 87.6, pp. 505–509.

Rumelhart, DE, GE Hinton, and RJ Williams (1986). "Learning internal representations by error propagation, Parallel Distributed Processing, Vol. 1". In: *Foundations. MIT Press, Cambridge*.

Russell, Stuart and Peter Norvig (2011). *Artificial Intelligence: A Modern Approach*.

— (2016). "Artificial intelligence: a modern approach (global 3rd edition)". In: *Essex: Pearson*.

Schoen, WF and AV Mrstik (1955). "Calculating gasoline blend octane ratings". In: *Industrial & engineering chemistry* 47.9, pp. 1740–1742.

Shah, Neel et al. (2019). *Prediction of autoignition and flame properties for multicomponent fuels using machine learning techniques*. Tech. rep. SAE Technical Paper.

Singh, Eshan et al. (2017). "Chemical kinetic insights into the octane number and octane sensitivity of gasoline surrogate mixtures". In: *Energy & Fuels* 31.2, pp. 1945–1960.

Stewart, Warren (1959). "Predict octanes for gasoline blends". In: *Petroleum Refiner* 38.12, pp. 135–139.

Tan, Yaoyuan Vincent and Jason Roy (2019). "Bayesian additive regression trees and the General BART model". In: *Statistics in medicine* 38.25, pp. 5048–5069.

Teixeira, Ana, João Leal, and Andre Falcao (2013). "Random forests for feature selection in QSPR Models-an application for predicting standard enthalpy of formation of hydrocarbons". In: *Journal of cheminformatics* 5.1, pp. 1–15.

Tian, Ying, Xinyu You, and Xiuhui Huang (2018). "SDAE-BP based octane number soft sensor using near-infrared spectroscopy in gasoline blending process". In: *Symmetry* 10.12, p. 770.

Tukey, John (1977). *Exploratory data analysis*. Addison-Wesley.

Tukey, John W (1993). *Exploratory data analysis: past, present and future*. Tech. rep. PRINCETON UNIV NJ DEPT OF STATISTICS.

Twu, Chorng and John Coon (1996). "Predict octane numbers using a generalized interaction method". In: *Hydrocarbon Processing* 75.2.

— (Mar. 1997). "Estimate octane numbers using an enhanced method". In: *Hydrocarbon Processing* 76.3.

Van Leeuwen, JA, RJ Jonker, and R Gill (1994). "Octane number prediction based on gas chromatographic analysis with non-linear regression techniques". In: *Chemometrics and intelligent laboratory systems* 25.2, pp. 325–340.

Venkatasubramanian, Venkat (2019). "The promise of artificial intelligence in chemical engineering: Is it here, finally". In: *AIChE* 65.2, pp. 466–78.

Vincent, Pascal, H Larochelle, et al. (2008). "Proceedings of the 25th International Conference on Machine Learning". In.

Vincent, Pascal, Hugo Larochelle, et al. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec, pp. 3371–3408.

Wang, Xiaoyu, Kan Yang, and John Kalivas (2020). "Comparison of extreme learning machine models for gasoline octane number forecasting by near-infrared spectra analysis". In: *Optik* 200, p. 163325.

Westbrook, Charles, Magnus Sjöberg, and Nicholas Cernansky (2018). "A new chemical kinetic method of determining RON and MON values for single component and multi-component mixtures of engine fuels". In: *Combustion and Flame* 195, pp. 50–62.

Wildhaber, Shawn Nicholas (2011). "Impact of combustion phasing on energy and availability distributions of an internal combustion engine". MA thesis. Missouri University of Science and Technology.

Yuan, Hao et al. (2017). "Optimal octane number correlations for mixtures of toluene reference fuels (TRFs) and ethanol". In: *Fuel* 188, pp. 408–417.

A Supervised Learning Theory

A.1 Regression

A.1.1 Linear Regression

Linear regression seeks to answer the following question. "Is there a relationship between the response variable and the predictor?" (James et al. 2013). Multiple Linear Regression (MLR) models the relationship as shown in Equation A.1 below.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{A.1}$$

And in a more compact form, Equation A.1 can be denoted as follows.

$$Y = \beta_0 + \sum \beta_i X_i + \epsilon \tag{A.2}$$

Where:

- β_0 is the intercept of the model.
- β_i are the model coefficients.
- ϵ is a normally distributed random error.

To determine whether a relations exists between a target variable and its predictors, the following hypothesis is tested.

H_0: There is no relationship between the target variable and its predictors.

The above hypothesis is then expressed in terms of the MLR model as shown below.

$$H_0: \beta_1 = \beta_2 = ... = \beta_p = 0.$$

The alternative hypothesis is shown below:

H_a: There is a relationship between the target variable and its predictors.

And can be expressed as:

$$H_a: \text{At least one } \beta_i \text{ is non-zero.}$$

In testing the above hypothesis, the F-statistic is computed.

$$F = \frac{(TSS - RSS)/p}{RSS/(n - p - 1)} \tag{A.3}$$

The parameters of Equation A.3 are defined in Error Metrics. Small values of F shows evidence to accept the null hypothesis while large values suggest that the null hypothesis be accepted. Before the F-statistic is determined, a model is built with the objective of minimising the Residual Sum of Squares (RSS) shown in Error Metrics. Once a model is built its F-statistic, RSS and prediction accuracy are used to determine whether or not the model is appropriate.

A.1.2 Polynomial Regression

Polynomial Regression works the same way as linear regression, the only difference being some of the X_i terms in equation A.2 are raised to a power (James et al. 2013). The parametric nature of the model requires that the analyst set the powers to which the model is raised and the coefficients be determined. Some programs allow for a gridsearch of the powers to be carried out but that may be computationally expensive.

A.1.3 Projection Pursuit Regression

Projection Pursuit Regression (PPR) is a concept which was initially introduced by Friedman and Stuetzle 1981 and takes the form shown in Equation A.4 (Friedman and Stuetzle 1981). PPR is a non-parametric prediction model based on dimension reduction. The model has to main steps. The first step is the projection and the second step is smoothing. Equation A.4 below shows the first step by determining the projection $V = \omega_m^T X$ and then the smoothing function takes that projection V as its input. The PPR model is a sum of functions and this means that it is an additive model.

$$\hat{y} = f(X) = \sum_{m=1}^{M} g_m(\omega_m^T X) \tag{A.4}$$

Where:

- X is the data frame of input variables.
- M is the number of iterations or smooths.
- ω_m is the unit projection vector of the m_{th} smooth.
- g_m is the non-linear smoothing function.

The fundamental difference between PPR and MLR is that in MLR, the regression surface is estimated from a proposed function and the problem becomes one of estimating the coefficients of the function. PPR on the hand, takes empirically determined uni-variate ridge functions (g_m) of linear combinations of predictors (James et al. 2013). The final predicted values will therefore be a sum of functions of linear combinations of the input variables. MLR on the other hand predicts the output variable directly from linear combinations of the input variables.

PPR is a 2 step algorithm described in the steps below (Van Leeuwen, Jonker, and Gill 1994). The first step is the reduction of the space (projection).

1. Produce a projection unit vector ω_1 of the first smooth.

2. Project the p-dimensional X space onto a projected vector $V_m = \omega_m^T X$.
3. Select the suitable projection vector ω_m by maximising the criterion $I(\omega_m)$ which is an R^2 equivalent which is the variance explained by the smooth.

$$I\left(\boldsymbol{\omega}_m\right) = 1 - \frac{\sum_{i=1}^{n}\left[r_{i,m} - g_m\left(x_i\boldsymbol{\omega}_m\right)\right]^2}{\sum_{i=1}^{n} r_{i,m}^2} \tag{A.5}$$

Where:

$$r_{i,m} = y_i - \hat{y}_i - \sum_{j=1}^{m-1} g_j\left(V_{ij}\right) \tag{A.6}$$

4. Repeat steps 2 and 3 until $I(\omega_m)$ is maximised.

The second step is the smoothing. Which follows the algorithm below (Van Leeuwen, Jonker, and Gill 1994).

1. Iteratively vary g_m and ω_m to maximize $I(\omega_m$.
2. Use gradient descent based cross validation to determine $I(\omega_m)$ maximising parameters.
3. Determine the residual from the $I(\omega)$ maximising g_1 and ω_1.
4. Substitute the residuals into the second iteration to get g_2 and ω_2
5. Continue iterations until $I(\omega)$ has been maximised.
6. Produce a scatter plot of $\hat{y}$ vs V

The plot produced will be a 2D plot, then simple linear regression can be used to get an actual function of V. PPR is a useful technique when the number of smooths to include in the model is correctly estimated. This is because the model prioritizes explanation variance therefore including noise in the model may cause the model to perform poorly (Van Leeuwen, Jonker, and Gill 1994).

A.2 Ensemble Learning

Ensemble Learning is the concept of building predictive models which are a collection of simpler base models (Friedman, Hastie, and Tibshirani 2001). Multiple learning algorithms are combined to increase the prediction accuracy of the constituent algorithms. The concepts leading to ensemble learning are described below.

A.2.1 Regression Tree

A regression tree makes predictions via stratification of the feature space (James et al. 2013). There are two major steps in building a regression tree.

1. Divide the predictor space of the p input variables $X_1, X_2, ..., X_p$ into J distinct, non-overlapping regions, $R_1, R_2, ..., R_J$.
2. Take the mean of the response variable of each of the predictors present in in each region. This mean value becomes the prediction value of all the predictors in the range covered by R_j.

To construct the region $R_1, ..., R_J$, The predictor space is divided into high dimensional rectangles, each minimising the RSS using Equation A.7. Where $\hat{y}_{R_j}$ is the is the mean response for the training observations within the j_th box.

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{A.7}$$

It is computationally expensive to consider all the possible partitions of the feature space. The partitioning is therefore done using a top-down greedy approach called Recursive Binary Splitting. Recursive Binary Splitting start from the top of a tree where the entire feature space belongs to one region and each split forms two branches. It is a greedy algorithm because the best split is made for a particular step and not for the remaining steps. Recursive Binary Splitting is carried out by selecting the predictor X_j and the cut-point s such that splitting the space into 2 regions $\{R_1(j, s) = X|X_j < s\}$ and $\{R_2(j, s) = X|X_j \geq s\}$ yields the largest reduction in RSS. The objective is therefore to seek the values of j and s which minimise Equation A.8 below.

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{A.8}$$

The above process is repeated by iteratively carrying out the splits within each region until a stopping criterion is reached. For instance splitting may carry on until there are no more than 5 observations in each region.The above process may produce a tree which fits the training data well but may be overfit. This problem is solved by pruning the tree. Tree pruning makes the tree more generalizable at the expense of increasing the bias in the model. One way to prune would be to select a sub-tree and determine its validation error, which computationally expensive to do for every possible sub-tree. To make the pruning more feasible, cost complexity pruning can be used. In cost complexity pruning, a sequence of trees indexed by a non-negative tuning parameter α is considered. Cost Complexity Pruning uses the concept that for each value of α, there is a corresponding subtree $T \subset T_0$ such that Equation A.9 below is satisfied .

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{A.9}$$

In Equation A.9 above, $|T|$ is the number of terminal nodes. The tuning parameter α controls the trade off between the complexity of a subtree and its fit to the training data. As α increases in Equation A.9 above, more branches are pruned and the model becomes more biased but more generalizable. The most optimal value of α can be determined using cross validation.

A.2.2 Random Forests

Random forests are a special case improvement of an ensemble learning technique called Bagging. Bagging is a procedure in which an ensemble of identical trees is grown from different bootstrapped samples of the data set. The drawback of bagging technique is that the trees are highly correlated and do not significantly reduce the variance of a single tree.

Random forests on the other hand, tweak the bagging method to de-correlate the trees. When growing bagged trees on a data set of p observations, p randomly bootstrapped samples are considered at each split whereas when growing random forests, m randomly sampled observations are considered (where $m < p$). The sample considered in a random forest has $m \approx \sqrt{p}$. Making m much less than p de-correlates the trees in a random forest. This is useful because when there are strong predictors in a data set, bagged tress will be highly correlated because the top splits of each tree will have the same variable. Because of a smaller sample being taken, random forests will have different splits at the top of each tree and this will reduce variance in the model (James et al. 2013). The procedure of growing a random forest can be summarised as follows (Friedman, Hastie, and Tibshirani 2001):

1. For $b = 1$ to B (where b is the number of trees in the random forest:
   1.1. Draw a bootstrap sample Z^* of size N from the training data.
   1.2. Grow a random forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      1.2.1. Select m observations at random from the p observations.
      1.2.2. Pick the best variable/split-point among the m.
      1.2.3. Split the node into 2 daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$

The regression function for random forests is shown in Equation A.10 below.

$$\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{A.10}$$

Equation A.10 shows that random forests are a semi-parametric model. This is so because, the structure of the ensemble of trees T_b is not made to fit a proposed function, but this parameter T_b is used as an input to the regression function.

A.2.3 Bayesian Additive Regression Trees

Bayesian Additive Regression Trees (BART) are an ensemble method which uses a "sum of trees approach". The trees are weak learners, which are a result of regularization. An iterative Bayesian Back-Fitting Monte Carlo Markov Chains (Back-fitting MCMC) algorithm is used to generate samples from a posterior distribution. Bayesian Additive Regression Trees (BART), like other tree based methods, is an ensemble method and is defined by a statistical method defined by a prior and a likelihood (Chipman, George, McCulloch, et al. 2010).

The fundamental problem of predictive modelling is the inference that an unknown underlying function f predicts on output Y from a p-dimensional vector of inputs. $x = (x_1, ..., x_p)$ as shown in Equation A.11

$$Y = f(x) + \varepsilon, \quad \varepsilon \sim \mathbb{N}(0, \sigma^2) \tag{A.11}$$

Predictive models which are built on the back of Bayes' theorem have the functions being expressed as conditional probabilities. $f(x)$ in Equation A.11 is approximated as the mean of Y given x, $f(x) = E(Y|x)$. This is expressed as a sum of m regression trees $f(x) \approx h(x) \equiv \sum_{j=1}^{m} g_j(x)$ where each g_j is a regression tree. The functional form is shown in Equation A.12 below.

$$Y = h(x) + \varepsilon, \quad \varepsilon \sim \mathbb{N}(0, \sigma^2) \tag{A.12}$$

BART differs from the other ensemble methods because Bayesian averaging is applied to the posterior sample to weight the trees before taking their sum. The weights used are posterior probabilities of each single tree. The prior distribution regularizes the fit, making each of the trees (g_j) weak learners explaining a portion of f (Chipman, George, McCulloch, et al. 2010). Building a BART model consists of two parts: a sum-of trees model and a regularization part. A single tree takes the form shown in Equation A.13 below.

$$Y = g(x; T, M) + \varepsilon, \quad \varepsilon \sim \mathbb{N}\left(0, \sigma^2\right) \tag{A.13}$$

The model in Equation A.13 is extended to a sum of trees as shown in Equation A.14 below (Chipman, George, McCulloch, et al. 2010).

$$Y = \sum_{j=1}^{m} g\left(x; T_j, M_j\right) + \varepsilon, \quad \varepsilon \sim \mathbb{N}\left(0, \sigma^2\right) \tag{A.14}$$

Where:

- T is a binary tree made up of interior node decision rules and a set of terminal nodes.
- $M = \{\mu_1, \mu_2, ..., \mu_b\}$, a set of parameter values associated with each of the b terminal nodes of T.
- μ_i is the average value of the target variable in terminal node i.

The average of the target variables in the ith terminal node of the jth tree; μ_{ij} will represent a main effect when there is a single input variable and interaction when there are multiple input variables. This makes the BART model, a strong candidate when both interaction and main effects are present in the data regardless of their order.

The second step in building the BART model is regularization. In the BART model, a prior is imposed over all the parameters of the sum of trees model. One way of carrying out the regularization, is to set the parameters using cross validation which is computationally expensive. The prior parameters to be determined are listed and described below.

1. Prior Independence and Symmetry.

$$\begin{aligned} p\left(\left(T_1, M_1\right), \ldots, \left(T_m, M_m\right), \sigma\right) &= \left[\prod_j p\left(T_j, M_j\right)\right] p(\sigma) \\ &= \left[\prod_j p\left(M_j|T_j\right) p\left(T_j\right)\right] p(\sigma) \end{aligned} \tag{A.15}$$

f

and

$$p\left(M_j|T_j\right) = \prod_i p\left(\mu_{ij}|T_j\right) \tag{A.16}$$

Under the above priors, the tree components (T_j, M_j) are independent of each other and of σ. The independence restriction above, simplifies the problem to a specification of forms for $p(T_j)$, $p(\mu_{ij}|T_j)$ and $p(\sigma)$.

2. The T_j prior.

   There are three aspects to specifying this prior;

   2.1. The probability that a node at depth $d(= 0, 1, 2...)$ is non-terminal, given by:

   $$\alpha(1+d)^{-\beta}, \quad \alpha \in (0,1), \beta \in [0,\infty) \tag{A.17}$$

   2.2. The distribution on the splitting variable assignments at each interior node.

   2.3. The distribution on the splitting rule assignment in each interior node, conditional on the splitting variable.

3. The $\mu_{ij}|T_j$ prior.

   To determine $p(\mu_{ij}|T_j)$, the conjugate normal distribution $\mathbb{N}(\mu_\mu, \sigma_\mu^2)$ is used. $E(Y|x)$ is the sum pf m μ_{ij}'s when using the sum of trees model. Each μ_{ij} is apriori Independent and Identically Distributed (IID), the induced prior of $E(Y|x)$ is $\mathbb{N}(m\mu_\mu, m\sigma_\mu^2)$. Because $y_{min} \leq E(Y|x) < y_{max}$, μ_μ and σ_μ are chosen such that $\mathbb{N}(m\mu_\mu, m\sigma_\mu^2)$ assigns substantial probability to the interval (y_{min}, y_{max}). Y is then shifted and re-scaled such that the range (y_{min}, y_{max}) = (-0.5,0.5). The prior for μ_{ij} is centered at zero with $\mu_\mu = 0$ and $k\sqrt{m}\sigma_\mu = 0.5$. Where k is a constant which is determined such that there would be a 95% prior probability that $E(Y|x)$ is in the interval (y_{min}, y_{max}). This yields:

   $$\mu_{ij} \sim \mathbb{N}\left(0, \sigma_\mu^2\right) \quad \text{where } \sigma_\mu = 0.5/k\sqrt{m} \tag{A.18}$$

   In a sum of trees model, the parameter μ_{ij} is a shrinkage parameter, which weakens the individual trees to make them weak learners.

4. The σ prior.

   To determine $p(\sigma)$, a conjugate prior of the inverse chi-square distribution is used, $\sigma^2 \sim v\lambda/\chi_v^2$. The parameters v and λ are determined such that substantial probability is assigned to the entire region of plausible values of σ. v is calibrated and σ is scaled to obtain an estimate $\hat{\sigma}$ of σ. σ is taken to be the sample standard deviation of Y

5. The choice of m.

   One way of determining the number of trees m is to specify a prior and use Bayes' theorem to determine m. Another way is setting m from a range of choices using cross validation. Both of these approaches are computationally expensive. The quickest way of determining the number of trees is to set a default value of 200 and compare results with a few other values. Performance of BART is observed to increase as m increases from 1, until a plateau is reached.

Having determined the parameters of the BART, the posterior distributions can be induced using the Bayesian Additive Regression Trees with Monte Carlo Markov Chains (BART-MCMC) algorithm. The posterior distribution of the unknowns of the sum of trees model is shown in Equation A.19 below.

$$p\left(\left(T_1, M_1\right), \ldots, \left(T_m, M_m\right), \sigma | y\right) \tag{A.19}$$

The algorithm samples a tree and its respective terminal nodes conditionally on the the remaining trees, their respective terminal nodes and σ. This can be written more concisely as m successive draws of (T_j, M_j) conditionally on $(T_{(j)}, M_{(j)}, \sigma)$:

$$(T_j, M_j) | T_{(j)}, M_{(j)}, \sigma, y \tag{A.20}$$

$j = 1, ..., m$, followed by a draw of σ from an inverse gamma distribution of the full conditional:

$$\sigma | T_1, ..., T_m, M_1, ..., M_m, y \tag{A.21}$$

Where $T_{(j)}$ is the sum of all the trees in the sum except T_j and similarly for $M_{(j)}$. The next step is to sample T_j, which is done by noting that the conditional distribution $p\left(T_j, M_j | T_{(j)}, M_{(j)}, \sigma, y\right)$ depends on $(T_{(j)}, M_{(j)}, y)$ only through the residuals R_j shown in Equation A.22 which is a vector partial residuals based on a fit which excludes the jth tree.

$$R_j \equiv y - \sum_{k \neq j} g\left(x; T_k, M_k\right) \tag{A.22}$$

The draws of $T_j, M_j | T_{(j)}, M_{(j)}, \sigma, y$ are equivalent to m draws from Equation A.23.

$$\left(T_j, M_j\right) | R_j, \sigma \tag{A.23}$$

This is equivalent to the posterior of a single tree model and because a conjugate prior was used for M_j, the following stochastic integral shown in Equation A.24 is obtained (Chipman, George, McCulloch, et al. 2010).

$$p\left(T_j | R_j, \sigma\right) \propto p\left(T_j\right) \int p\left(R_j | M_j, T_j, \sigma\right) p\left(M_j | T_j, \sigma\right) dM_j \tag{A.24}$$

The closed form of the above integral allows for the draws of Equation A.23 in two steps as in Equations A.25 and A.26 below.

$$T_j | R_j, \sigma \tag{A.25}$$

$$M_j | T_j, R_j, \sigma \tag{A.26}$$

The draw of T_j can be obtained from a Metropolis Hastings (MH) algorithm which is the Bayesian Back-Fitting Monte Carlo Markov Chains procedure used for this model. The draw of T_j is in essence a tree growing and pruning procedure carried out using one of the processes listed below.

1. Growing a terminal node.
2. Pruning a pair of terminal nodes.
3. Changing a non-terminal rule.
4. Swapping a rule between parent and child nodes

The draw of M_j is done next and is a set of independent draws of the terminal node μ_{ij}'s from a normal distribution.The draw M_j is used to determine the residual R_{j+1} which is required for the next draw of T_j. These iterations are repeated until satisfactory convergence is obtained.

A.2.4 Gradient Boosting Machines

The objective of gradient boosting machines is to carry out an optimization over a function space. This optimization obtains an approximation $F(\mathbf{x})$, of the function $F^*(\mathbf{x})$ mapping $\mathbf{x}$ onto y. The function must minimize the expected value of some specified loss function $L(y, F(\mathbf{x}))$ over the joint distribution of all $(y, \mathbf{x})$ values as shown in Equation A.27 (Friedman 2001). The objective function can be maximised using either parametric or non-parametric equations.

$$F^* = \arg\min_F E_{y,\mathrm{x}} L(y, F(\mathbf{x})) = \arg\min_F E_{\mathrm{x}} \left[E_y(L(y, F(\mathbf{x})))|\mathbf{x} \right] \tag{A.27}$$

$F(x)$ can be restricted to be a member of a parameterized class of functions $F(\mathbf{x}; \mathbf{P})$, where $\mathbf{P} = \{P_1, P_2, \cdots\}$ is a set of parameters with values identifying individual class members. Additive expansions usually take the form;

$$F\left(\mathbf{x}; \{\beta_m, \mathbf{a}_m\}_1^M\right) = \sum_{m=1}^{M} \beta_m h\left(\mathbf{x}; \mathbf{a}_m\right) \tag{A.28}$$

The function $h(\mathbf{x}; \mathbf{a})$ is a parameterized function of the input variables $\mathbf{x}$ characterized by the parameters $\mathbf{a} = \{a_1, a_2, \cdots\}$. When a parameterized model $F(\mathbf{x}; \mathbf{P})$ is chosen, the problem shifts from one function to parameter optimization:

$$\mathbf{P}^* = \arg\min_{\mathrm{P}} \Phi(\mathbf{P}) \tag{A.29}$$

where:

$$\Phi(\mathbf{P}) = E_{y,\mathrm{x}} L(y, F(\mathbf{x}; \mathbf{P}))$$

The optimization becomes;

$$F^*(\mathbf{x}) = F\left(\mathbf{x}; \mathbf{P}^*\right)$$

Numerical optimization methods are used to solve Equation A.29. The solution is expressed as follows:

$$\mathbf{P}^* = \sum_{m=0}^{M} \mathbf{p}_m$$

where $\mathbf{p}_0$ is the initial guess and

$$\{\mathbf{p}_m\}_1^M$$

are successive boosts. The steepest descent method is used to define the boosts

$$\{\mathbf{p}_m\}_1^M$$

by first computing the current gradient $\mathbf{g}_m$ as follows;

$$\mathbf{g}_m = \{g_{jm}\} = \left\{ \left[\frac{\partial \Phi(\mathbf{P})}{\partial P_j} \right]_{\mathrm{P}=\mathrm{P}_{m-1}} \right\}$$

Where:

$$\mathbf{P}_{m-1} = \sum_{i=0}^{m-1} \mathbf{p}_i$$

and the boost is taken to be

$$\mathbf{p}_m = -\rho_m \mathbf{g}_m$$

Where the line search along the steepest descent direction ρ_m is determined from the steepest-descent direction $-\mathbf{g}_m$ using the following equation:

$$\rho_m = \arg\min_{\rho} \Phi\left(\mathbf{P}_{m-1} - \rho \mathbf{g}_m\right)$$

In the non-parametric approach, numerical optimization can be parameterized in the function space by minimizing the following function directly with respect to $\mathbf{x}$

$$\Phi(F) = E_{y,\mathrm{x}} L(y, F(\mathbf{x})) = E_{\mathrm{x}} \left[E_y (L(y, F(\mathbf{x})))|\mathbf{x} \right]$$

The optimal solution takes the form

$$F^*(\mathbf{x}) = \sum_{m=0}^{M} f_m(\mathbf{x})$$

where $f_0(\mathbf{x})$ is an initial guess and $\{f_m(\mathbf{x})\}_1^M$ are the boosts defined by steepest descent as

$$f_m(\mathbf{x}) = -\rho_m g_m(\mathbf{x})$$

where

$$g_m(\mathbf{x}) = \left[\frac{\partial \phi(F(\mathbf{x}))}{\partial F(\mathbf{x})} \right]_{F(\mathrm{x})=F_{m-1}(\mathrm{x})} = \left[\frac{\partial E_y[L(y, F(\mathbf{x}))|\mathbf{x}]}{\partial F(\mathbf{x})} \right]_{F(\mathrm{x})=F_{m-1}(\mathrm{x})}$$

and

$$F_{m-1}(\mathbf{x}) = \sum_{i=0}^{m-1} f_i(\mathbf{x})$$

The steepest-descent becomes

$$g_m(\mathbf{x}) = E_y \left[\frac{\partial L(y, F(\mathbf{x}))}{\partial F(\mathbf{x})} |\mathbf{x} \right]_{F(\mathrm{x})=F_{m-1}(\mathrm{x})}$$

and the line search parameter becomes

$$\rho_m = \arg\min_{\rho} E_{y,\mathrm{x}} L\left(y, F_{m-1}(\mathbf{x}) - \rho g_m(\mathbf{x})\right)$$

When a finite data sample $\{y_i, \mathbf{x}_i\}_1^N$ is used the parametric approach is adapted as follows to minimize the expected loss.

$$\{\beta_m, \mathbf{a}_m\}_1^M = \arg\min_{\{\beta'_m, \mathbf{a}'_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta'_m h\left(\mathbf{x}_i; \mathbf{a}'_m\right)\right)$$

A stage-wise matching pursuit approach is used to carry out the minimization as opposed to the step-wise boosts used previously.

$$(\beta_m, \mathbf{a}_m) = \arg\min_{\beta,\mathrm{a}} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \beta h\left(\mathbf{x}_i; \mathbf{a}\right)\right)$$

and then

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h\left(\mathbf{x}; \mathbf{a}_m\right)$$

The function $h(\mathbf{x}; \mathbf{a})$ is the weak learner. The direction of steepest-descent becomes

$$-g_m\left(\mathbf{x}_i\right) = -\left[\frac{\partial L\left(y_i, F\left(\mathbf{x}_i\right)\right)}{\partial F\left(\mathbf{x}_i\right)}\right]_{F(\mathrm{x})=F_{m-1}(\mathrm{x})}$$

which is not generalized across the data set but is determined individually for every point on the data set. To ensure generalizability, the parameterized basis function $h(\mathbf{x}; \mathbf{a})$ most highly correlated with $-g_m(\mathbf{x})$ is obtained from the solution

$$\mathbf{a}_m = \arg\min_{\mathrm{a},\beta} \sum_{i=1}^{N} \left[-g_m\left(\mathbf{x}_i\right) - \beta h\left(\mathbf{x}_i; \mathbf{a}\right)\right]^2$$

and the line search is performed as

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \rho h\left(\mathbf{x}_i; \mathbf{a}_m\right)\right)$$

and the approximation is updated

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h\left(\mathbf{x}; \mathbf{a}_m\right)$$

A.3 Deep Learning

Deep learning is a class of algorithms which has been inspired by the human nervous system and was developed in pursuit of soling complex AI problems such as recognizing speech and objects (Goodfellow, Bengio, and Courville 2016). The name "Deep Learning" comes from the idea that the input data goes through a cascade of functions to provide a prediction. This cascade has a certain depth, bringing about the concept of "Deep Learning. There are several deep learning models which exist but this study only employs feed forward deep learning models.

A.3.1 Artificial Neural Networks

An Artificial Neural Networks uses links from one unit to the next to propagate the activation a_i. Each unit models a type of artificial neuron called a perceptron (Nielsen 2015). A perceptron can output a decision given its factors(input variables and their respective weights.

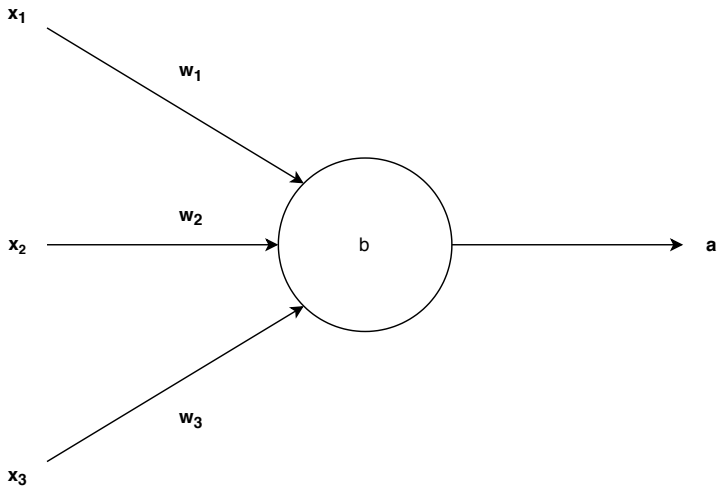


Figure A.1: Representation of a perceptron

Figure A.1 above shows the representation of a sigma neuron, an extension of the perceptron to non-binary outputs. Where x_i is each of the input variables, w_i are the corresponding weights and some bias b. The output $a = \sigma(z)$ from the neuron is the propagation of the activation function through the neuron. $z = \sum_j w_j b_j + b$.

A.3.1.1 Single Layer Feed-forward Neural Network (SLFN)

A single neuron would not be a particularly strong learner so the concept is extended to have multiple neurons working together in a SLFN, this is represented in Figure A.2 below.

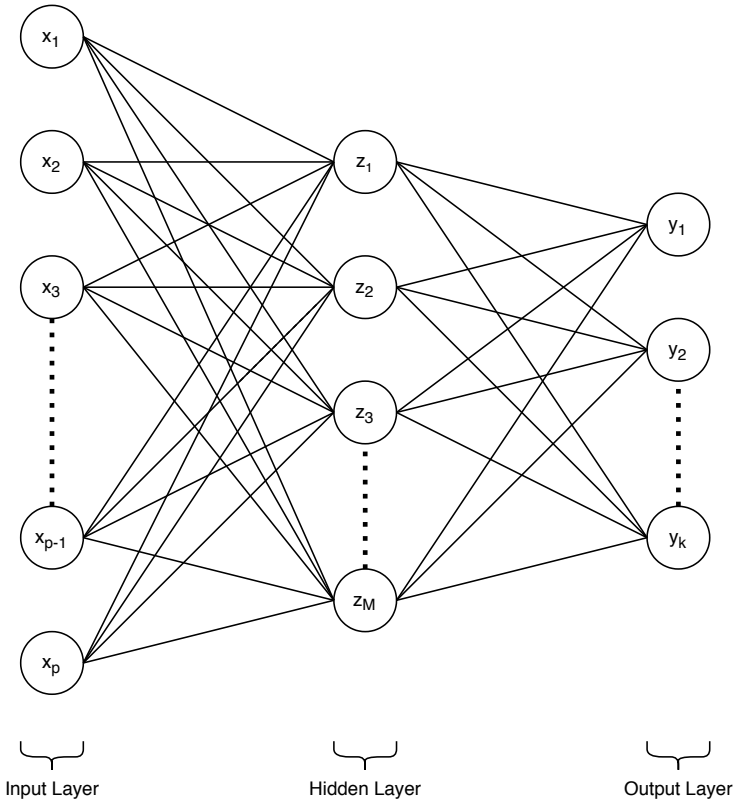


Figure A.2: Representation of the Single Layer Feed-forward Neural Network

Each unit z_i in the representation above is in the hidden layer. A SLFN consists of three layers; the input layer (input variables x_i), a single hidden layer (activation propagations z_i) and the output layer (output variables y_i). An ANN has two major specifications; the activation finctions and the cost function. The activation function as mentioned earlier, propagates the input through the ANN to obtain a prediction. For best performance, the sigmoid activation function is typically used for deep learning regression (Nielsen 2015). The sigmoid activation function is shown as a function of the hidden layer z_i in Equation A.30.

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} \tag{A.30}$$

The cost function on the other hand, measures the performance of a deep learning model by quantifying the error of prediction as a single real number. The cost function is usually minimized and must always be differentiable with respect to the weights and biases. The cross-entropy cost function for the sigmoid neuron in Figure A.1 is shown in Equation A.31. Cross entropy cost is used for most regression tasks because of its differentiability and ability to converge quickly (Nielsen 2015).

$$C = -\frac{1}{n}\sum_{x}[y \ln a + (1-y)\ln(1-a)] \tag{A.31}$$

The partial derivatives of the cross entropy cost function with respect to the weights and biases are shown in Equations A.32 and A.33 respectively.

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y) \tag{A.32}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y) \tag{A.33}$$

Minimizing the derivatives of the cost function does not give sufficient generalizability to the deep learning model. To solve this, regularization is carried out. The most commonly used technique is L2 regularization; where an extra term is added to the cost function. The regularization term is a sum of squares of all the weights in the network scaled by a factor $\frac{\lambda}{2n}$ where λ is a positive regularization parameter. The regularized cost function is shown in Equation A.34 below. Where C_0 is the unregularized cost function. Regularization will change the derivatives and suppress over-fitting (Nielsen 2015).

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2 \tag{A.34}$$

To ensure that the initialized weights do not slow the learning rate, they will be initialized as Gaussian random variables with mean 0 and standard deviation $1/\sqrt{n_{in}}$.

The biggest problem with training an ANN properly, is the selection of hyper-parameters. The hyper parameters and their respective definitions are listed below.

1. Activation function.
2. Cost function.
3. Architecture.
4. Epochs.
5. Learning Rate η. Controls the step size in the gradient descent method.
6. Regularization Parameterλ.
7. Mini-batch size.

This process shall be summarized below (Nielsen 2015).

1. Sample a small training set and an even smaller validation set. Note that these samples are not the same as the main training and validation sets.
2. Use a basic simple network which can provide meaningful results.
3. Use a large number of epochs on the small ANN.
4. Start with a regularization parameter $\lambda = 0$ (no regularization) and increase it as necessary to obtain a change in the training accuracy.
5. Assess how the results change with tuning the learning rate η (note that a small η results in over-fitting and the opposite for a large η.

6. Once a good value of η is obtained, obtain a value of λ. A good value of η is one where the the cost on the data decreases immediately and does not oscillate or increase during the first few epochs.

7. Increase complexity of the structure and iterate for η and λ.

A.3.1.2 Multiple Layer Feed-forward Neural Network (MLFN)

A Multiple Layer Feed-forward Neural Network (MLFN) extends the architecture of the SLFN to have multiple hidden layers as shown in Figure A.3. The algorithm of each neuron will work in exactly the same fashion as the SLFN. The cross entropy function becomes a sum across all the output neurons as shown in Equation A.35. The training and hyper-parameter tuning remains the same as that of a SLFN.

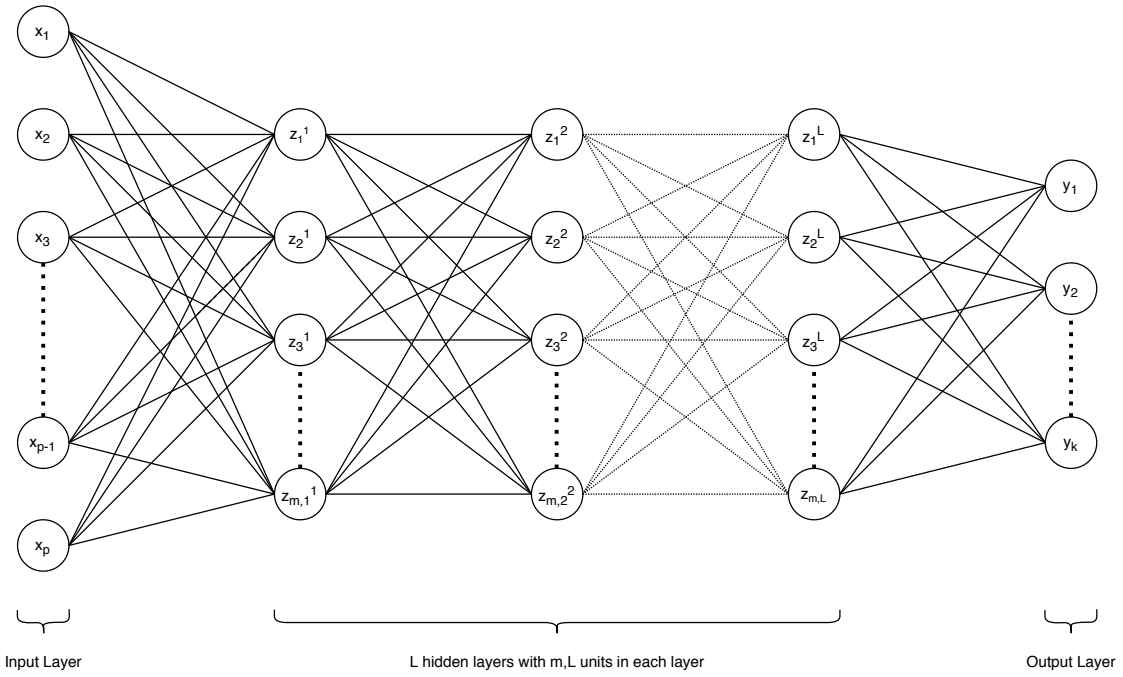


Figure A.3: Representation of the Multiple Layer Feed-forward Neural Network

$$\sum_j \left[y_j \ln a_j^L + (1 - y_j) \ln \left(1 - a_j^L\right) \right] \tag{A.35}$$

A.3.2 Extreme Learning Machines

The Extreme Learning Machine (ELM) is a special case of the SLFN based on generalized inverse matrix theory. As discussed in the previous section, a SLFN is a non-parametric model which takes the form shown in Equation A.36 below.

$$y_i = \sum_i^K \beta_i \sigma(z_i) \tag{A.36}$$

Where z_i is the output of the activation propagation through node i of the hidden layer, calculated from the input weights into node i and the biases, b of the node. $z = \sum_j w_j b_j + b$ and σ is the activation function. β_i is the vector of weights from hidden node i. G.-B. Huang, Zhu, and Siew 2006 argue that the input layer weight matrix and the hidden layer biases can be generated randomly. G.-B. Huang, Zhu, and Siew 2006 also argue that gradients based learning is slow and has the drawback of local maxima and random allocation of the weights and biases solves these problems as a cost function will not be used for regularization. When a neural network has a number of hidden layers which is equal to the number of input variables, the SLFN trains with zero error for any weight and bias. Meaning $\sum_{i=1}^{N} \|y_i - \hat{y}_i\| = 0$, where $\hat{y}_i = \sum_i^K \beta_i \sigma(z_i)$. The problem becomes one of using a least squares solution , $\hat{\beta}$ to the linear system shown in Equation A.37 below.

$$\mathbf{H}\beta = \mathbf{T} \tag{A.37}$$

Where:

- **H** is the matrix of hidden layer activations.
- β is the matrix made of the output weights from each hidden node(estimated using least squares.
- T is the expected output matrix (comprised of the $\hat{y}$ values of each output variable.

The least square equation to be solved becomes;

$$\|H\hat{\beta} - T\| = \min_{\beta} \|H\beta - T\| \tag{A.38}$$

If the number of hidden nodes does not equal the number of input variables, a square matrix cannot be obtained for **H**. The Moore-Penrose generalized inverse would be used in such a case and the equation would become $\hat{\beta} = H^{\dagger}T$. The procedure for building an ELM can be summarised as follows:

1. Randomly assign weights and biases.
2. Calculate the activations and populate the matrix **H**.
3. Calculate the output weights β

A.3.2.1 Online Sequential Extreme Learning Machine (OS-ELM)

Nielsen 2015 proposes that a neural network be trained iteratively with Small sub samples of the training set. When a neural network is being trained with increasing training observations, some hyper-parameters change. Instead of manually tuning hyperparameters, Liang et al. 2006 propose a method to learn new hyperparameters as the model is trained, the OS-ELM. The number of hidden nodes is the only parameter needing to be set when training an OS-ELM. The OS-ELM is a two step iterative process; calculating the weight matrix of the output layer according to the method of the basic ELM (initialization phase) and adding new

training data to the model and updating the weights of the output layer (sequential learning phase). Output layer weights are recalculated until all the data has been trained. an OS-ELM uses radial basis functions as activation functions and is trained as follows.

1. Given a mini batch from an initial training set $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$ solve for the output layer weights as shown in Equation A.38.
2. Obtain the initial output weights $\beta^{(0)} = \mathbf{K}_0^{-1}\mathbf{H}_0^T\mathbf{T}_0$ where $\mathbf{K}_0 = \mathbf{H}_0'\mathbf{H}_0$.
3. Determine the output layer weights for a new batch $\aleph_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$ where N_1 is the number of observations in the batch. TO do this one would minimize

$$\left\| \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \boldsymbol{\beta} - \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \right\|$$

Taking into account both $\aleph_0$ and $\aleph_1$, the output weight β becomes:

$$\boldsymbol{\beta}^{(1)} = \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}$$

Where

$$\mathbf{K}_1 = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}$$

Sequential learning uses the logic of $B^{(1)}$ and a function of $B^{(0)}$ and not a function of the previous batch so $\mathbf{K}_1$ becomes:

$$\begin{aligned} \mathbf{K}_1 &= \begin{bmatrix} \mathbf{H}_0^T \mathbf{H}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1 \end{aligned}$$

and

$$\begin{aligned} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} &= \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_0 \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_0 \boldsymbol{\beta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \left(\mathbf{K}_1 - \mathbf{H}_1^T \mathbf{H}_1\right) \boldsymbol{\beta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1 \\ &= \mathbf{K}_1 \boldsymbol{\beta}^{(0)} - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\beta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1 \end{aligned}$$

The final expression for $\beta^{(1)}$ becomes:

$$\begin{aligned} \boldsymbol{\beta}^{(1)} &= \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\ &= \mathbf{K}_1^{-1} \left(\mathbf{K}_1 \boldsymbol{\beta}^{(0)} - \mathbf{H}_1^T \mathbf{H}_1 \boldsymbol{\beta}^{(0)} + \mathbf{H}_1^T \mathbf{T}_1 \right) \\ &= \boldsymbol{\beta}^{(0)} + \mathbf{K}_1^{-1} \mathbf{H}_1^T \left(\mathbf{T}_1 - \mathbf{H}_1 \boldsymbol{\beta}^{(0)} \right) \end{aligned}$$

where

$$\mathbf{K}_1 = \mathbf{K}_0 + \mathbf{H}_1^T \mathbf{H}_1$$

The recursive algorithm can be adapted as follows for any $(k+1)$th batch of data.

$$\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=\left(\sum_{j=0}^{k} N_j N_j\right)+1}^{k+1} N_j$$

where

$$\mathbf{K}_{k+1} = \mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1}$$
$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{K}_{k+1}^{-1} \mathbf{H}_{k+1}^T \left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^{(k)} \right)$$

and T is the expected output layer matrix. Note that $\mathbf{K}_{k+1}^{-1}$ instead of $\mathbf{K}_{k+1}$ is used to compute $\boldsymbol{\beta}^{(k+1)}$ and is calculated using the Woodbury formula:

$$\begin{aligned}\mathbf{K}_{k+1}^{-1} =& \left(\mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1} \right)^{-1} \\ =& \mathbf{K}_k^{-1} - \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T \right)^{-1} \\ & \times \mathbf{H}_{k+1} \mathbf{K}_k^{-1}\end{aligned}$$

To simplify the notation, let $\mathbf{P}_{k+1} = \mathbf{K}_{k+1}^{-1}$. The equations for updating $\beta^{(k+1)}$ become

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T \right)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k$$
$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T \left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^{(k)} \right)$$

The above process is carried out iteratively until all the data has been trained.

A.3.2.2 Self Adaptive Evolutionary Extreme Learning Machine (SaDE-ELM)

The ELM procedure described in the previous section may in some cases obtain sub-optimal results because of the random estimation of the parameters (Wang, Yang, and Kalivas 2020). To optimize the parameter selection, the weights and biases are trained as the fitness of the differential evolution algorithm in the 4-step procedure described below (Cao, Z. Lin, and G.-B. Huang 2012).

1. Initialization. Present a training set with N_0 observations and a number of hidden neurons K. The initialization population is composed of NP set vectors made of the weights and biases and these are represented as follows:

   $$\theta_{s,G} = \left[w_{1,(s,G)}^T, \cdots, w_{K,(s,G)}^T, b_{1,(s,G)}, \cdots, b_{K,(s,G)} \right]$$

   Where G is the evolutionary generation and s is the population scale of each of the set vectors.

2. Determine the hidden layer output weights and verify the fitness.

   2.1. For each vector s in the population, calculate the output matrix of the hidden layer $H_{s,G}$ using the ELM.

   2.2. Calculate the weight matrix of the output layer using the Moore-Penrose inverse of the output weight matrix.

   $$\tilde{\beta}_{s,G} = H_{s,G}^{\dagger} T$$

   2.3. Determine the RMSE as an objective function of the differential evolution algorithm to check the fitness.

   $$RMSE_{s,G} = \sqrt{\frac{\sum_{i=1}^{N} \left| \sum_{j=1}^{K} \beta_j \cdot g\left(\omega_{j,(s,G)} \cdot x_1 + b_{j,(s,G)} - t_i \right) \right|}{m \cdot N}}$$

2.4. Determine the $(G+1)$th vector $\theta_{s,G+1}$ using the discriminant of ther RMSE as follows:

$$\theta_{s,G+1} = \begin{cases} u_{s,G+1} \text{ if } RMSE_{\theta_{s,C}} - RMSE_{u_{s,G+1}} > \lambda \cdot RMSE_{\theta_{s,G}} \\ u_{s,G+1} \text{ if } \left|RMSE_{\theta_{s,G}} - RMSE_{u_{s,G+1}}\right| < \lambda \cdot RMSE_{\theta_{s,G}} \\ and \left\|\beta_{u_{s,G+1}}\right\| < \left\|\beta_{\theta_{s,G}}\right\| \\ \theta_{s,G} \text{ else} \end{cases}$$

Where $u_{s,G+1}$ is the $(G+1)$th generation evolutionary individual vector for testing and the discriminant coefficient λ is preset to 0.015.

3. Mutation and cross-over. There are four potential mutation operations, and the choice of mutation strategy is determined by evaluating the probability parameter $P_{l,G}$. The probability parameter is the probability that strategy $l(l = 1, 2, 3, 4)$ should be chosen at the Gth generation. A threshold called the learning period, LP is used to determine the probability parameter as follows.

   3.1. When $G \leq \text{LP}$, each strategy has an equal chance off being chosen, i.e. $p_{l,G} = \frac{1}{4}$

   3.2. When $G \geq \text{LP}$, $p_{l,G} = \frac{S_{l,G}}{\sum_{l=1}^{4} S_{l,G}}$, where

$$S_{l,G} = \frac{\sum_{g=G-\text{L}}^{G-1} ns_{l,g}}{\sum_{g=G-\text{LP}}^{G-1} ns_{l,g} + \sum_{g=G-\text{LP}}^{G-1} nf_{l,g}} + \varepsilon, (l = 1, 2, 3, 4)$$

   Where:

   - $ns_{l,g}$ is the number of trial vectors generated by the lth strategy at the gth generation that can enter the next generation.
   - $nf_{l,g}$ is the number of trial vectors generated by the lth strategy at the gth generation that are discarded in the next generation.

4. Evaluation. The trial vectors $u_{s,G+1}$ generated at the $(G+1)$th generation are evaluated as shown in step 2. and steps 3 and 4 are repeated iteratively until the preset goal is reached or he maximum learning iterations are reached.

A.3.3 Back-Propagation Learning

A Back Propagation Neural Networks (BPNN) is a special case of the MLFN. In a MLFN, forward propagation happens until either a specified threshold of the cost is met or the the cost is minimised by the partial derivatives discussed earlier. Back propagation is an algorithm in which the error of prediction is propagated backwards from the output layer to the input layer, thereby adjusted the weights of each of the neurons. This is done using stochastic gradient descent. The back-propagation algorithm can be summarised as follows:

1. Set the activation function and parse the data into the input layer.
2. Start the MLFN and calculate all the activations a_i and the outputs of each node z_i.
3. Output the error of the activation in the output layer.
4. Back-propagate the error of each output neuron.
5. Output the gradient of the cost function determined using stochastic gradient descent.

A.4 Error Metrics

For regression tasks regardless of the algorithm used, there are some generic error metrics which can be used to assess model reliability and accuracy. Error metrics are useful for quantifying the extent of the difference between the predicted and true value for a given observation (James et al. 2013). Some are discussed in the sections below. The methods discussed in this section have been used by the researchers who have published the reviewed literature.

A.4.1 Mean Squared Error (MSE)

This is one of the most commonly used metrics for regression. MSE is the average of the squared difference between the predicted value and the true value. The equation for the MSE is shown below in Equation A.39. All the equations in this section have their variables defined as follows:

- n is the number of observations.
- y_i and $\hat{y}_i$ are the true and predicted values respectively.

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2 \tag{A.39}$$

A.4.2 Mean Absolute Error (MAE)

The MSE is the absolute difference between the predicted and true values a shown in Equation A.40. It is more robust to outliers as it penalizes errors less than MSE. MAE is less appropriate for situations where outliers are of importance (James et al. 2013).

$$MAE = \frac{1}{n}\sum|y_i - \hat{y}_i| \tag{A.40}$$

A.4.3 Residual Sum of Squares (RSS)

The RSS is the sum of the squares of residuals shown in Equation A.41. The residuals being the difference between the true value and the predicted value (James et al. 2013).

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{A.41}$$

A.4.4 Residual Standard Error (RSE)

The RSE is a measure of lack of fit of the model to the data and also estimates the standard deviation of the residuals (James et al. 2013). The formula is shown in Equation A.42 and is derived from the RSS.

$$RSE = \sqrt{\frac{1}{n-2}RSS} \tag{A.42}$$

A.4.5 Root Mean Squared Error (RMSE)

RMSE which is shown in Equation A.43 is the most widely used metric for regression. RMSE is the average squared difference between the true values and the predicted values. The RMSE penalises large errors severely because of the squared term. RMSE will therefore be useful when the large errors are undesirable (James et al. 2013).

$$RSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}} \tag{A.43}$$

A.4.6 Co-efficient of Determination

The coefficient of Determination (R^2) is determined as a proportion. This is the proportion of variance explained. To calculate the R^2 value, the formula shown in Equation A.44 below (James et al. 2013).

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \tag{A.44}$$

Where $TSS = \sum(y_i - \bar{y}_i)$ is the total sum of squares.

A.4.7 Adjusted Coefficient of Determination

The Adjusted Coefficient of Determination ($\bar{R}^2$)value is variant of the coefficient of determination which takes into account the number of variables and data points in the data as shown in Equation A.45 (James et al. 2013).

$$\bar{R}^2 = 1 - \left(1 - R^2\right)\frac{n-1}{n-p-1} \tag{A.45}$$

B Database

B.1 Blend Data

Table B.1: Gasoline Blend Data (González 2019), (Singh et al. 2017), (Al Fahemi, Albis, and Gad 2014), (Abdul-Gani et al. 2018), (Yuan et al. 2017)

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
1	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	95.0	5.0	0.0	0.0	0.0	0.0	0.0	5.0
3	90.0	10.0	0.0	0.0	0.0	0.0	0.0	10.0
4	85.0	15.0	0.0	0.0	0.0	0.0	0.0	15.0
5	80.0	20.0	0.0	0.0	0.0	0.0	0.0	20.0
6	90.0	0.0	10.0	0.0	0.0	0.0	0.0	22.0
7	75.0	25.0	0.0	0.0	0.0	0.0	0.0	25.0
8	80.0	0.0	20.0	0.0	0.0	0.0	0.0	28.0
9	70.0	30.0	0.0	0.0	0.0	0.0	0.0	30.0
10	72.0	18.0	10.0	0.0	0.0	0.0	0.0	32.0
11	65.0	35.0	0.0	0.0	0.0	0.0	0.0	35.0
12	70.0	0.0	30.0	0.0	0.0	0.0	0.0	38.0
13	66.6	16.7	16.7	0.0	0.0	0.0	0.0	39.0
14	64.0	17.0	19.0	0.0	0.0	0.0	0.0	39.0
15	66.7	16.7	16.7	0.0	0.0	0.0	0.0	39.0
16	60.0	40.0	0.0	0.0	0.0	0.0	0.0	40.0
17	50.0	0.0	0.0	0.0	50.0	0.0	0.0	40.3
18	64.0	16.0	20.0	0.0	0.0	0.0	0.0	42.0
19	55.0	45.0	0.0	0.0	0.0	0.0	0.0	45.0
20	72.0	8.0	0.0	0.0	0.0	20.0	0.0	45.9
21	63.0	27.0	0.0	0.0	0.0	10.0	0.0	46.5
22	54.0	36.0	10.0	0.0	0.0	0.0	0.0	48.0
23	54.0	36.0	10.0	0.0	0.0	0.0	0.0	48.0
24	50.0	50.0	0.0	0.0	0.0	0.0	0.0	50.0
25	60.0	0.0	40.0	0.0	0.0	0.0	0.0	51.4
26	56.0	14.0	30.0	0.0	0.0	0.0	0.0	53.2
27	56.0	14.0	30.0	0.0	0.0	0.0	0.0	53.2
28	64.0	16.0	0.0	0.0	0.0	20.0	0.0	53.3
29	70.0	0.0	0.0	0.0	0.0	30.0	0.0	54.3
30	45.0	55.0	0.0	0.0	0.0	0.0	0.0	55.0
31	54.0	36.0	0.0	0.0	0.0	10.0	0.0	55.0
32	54.0	36.0	0.0	0.0	0.0	10.0	0.0	55.6
33	54.2	5.6	40.2	0.0	0.0	0.0	0.0	57.5
34	48.0	32.0	20.0	0.0	0.0	0.0	0.0	58.0
35	48.0	32.0	20.0	0.0	0.0	0.0	0.0	58.0
36	46.5	33.7	19.8	0.0	0.0	0.0	0.0	59.5
37	40.0	60.0	0.0	0.0	0.0	0.0	0.0	60.0
38	40.0	60.0	0.0	0.0	0.0	0.0	0.0	60.6
39	42.5	47.3	10.2	0.0	0.0	0.0	0.0	60.8
40	56.0	24.0	0.0	0.0	0.0	20.0	0.0	60.8

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
41	63.0	7.0	0.0	0.0	0.0	30.0	0.0	61.1
42	48.0	12.0	40.0	0.0	0.0	0.0	0.0	63.7
43	48.0	12.0	40.0	0.0	0.0	0.0	0.0	63.7
44	45.0	45.0	0.0	0.0	0.0	10.0	0.0	63.8
45	50.0	0.0	50.0	0.0	0.0	0.0	0.0	64.1
46	50.0	0.0	50.0	0.0	0.0	0.0	0.0	64.1
47	40.0	45.0	15.0	0.0	0.0	0.0	0.0	64.3
48	35.0	65.0	0.0	0.0	0.0	0.0	0.0	65.0
49	50.0	0.0	50.0	0.0	0.0	0.0	0.0	65.1
50	49.9	0.0	50.1	0.0	0.0	0.0	0.0	65.9
51	50.0	0.0	50.0	0.0	0.0	0.0	0.0	65.9
52	50.0	0.0	50.0	0.0	0.0	0.0	0.0	65.9
53	50.0	0.0	50.0	0.0	0.0	0.0	0.0	65.9
54	50.0	0.0	50.0	0.0	0.0	0.0	0.0	65.9
55	36.0	54.0	10.0	0.0	0.0	0.0	0.0	66.0
56	36.0	54.0	10.0	0.0	0.0	0.0	0.0	66.0
57	36.0	54.0	10.0	0.0	0.0	0.0	0.0	66.0
58	42.0	28.0	30.0	0.0	0.0	0.0	0.0	66.1
59	42.0	28.0	30.0	0.0	0.0	0.0	0.0	66.1
60	42.0	28.0	30.0	0.0	0.0	0.0	0.0	66.1
61	40.0	47.5	7.5	0.0	0.0	5.0	0.0	66.6
62	35.0	57.5	7.5	0.0	0.0	0.0	0.0	67.3
63	56.0	14.0	0.0	0.0	0.0	30.0	0.0	67.4
64	40.0	30.0	30.0	0.0	0.0	0.0	0.0	68.4
65	40.0	30.0	30.0	0.0	0.0	0.0	0.0	68.4
66	40.0	30.0	30.0	0.0	0.0	0.0	0.0	68.4
67	48.0	32.0	0.0	0.0	0.0	20.0	0.0	68.5
68	48.8	5.0	36.2	0.0	0.0	10.0	0.0	68.7
69	48.0	32.0	0.0	0.0	0.0	20.0	0.0	69.0
70	40.0	50.0	0.0	0.0	0.0	10.0	0.0	69.7
71	60.0	0.0	0.0	0.0	0.0	40.0	0.0	69.7
72	30.0	70.0	0.0	0.0	0.0	0.0	0.0	70.0
73	30.0	70.0	0.0	0.0	0.0	0.0	0.0	70.0
74	30.0	70.0	0.0	0.0	0.0	0.0	0.0	70.0
75	41.9	30.3	17.8	0.0	0.0	10.0	0.0	70.3
76	30.0	70.0	0.0	0.0	0.0	0.0	0.0	70.3
77	35.0	60.0	0.0	0.0	0.0	5.0	0.0	70.5
78	43.0	7.0	30.0	20.0	0.0	0.0	0.0	70.9
79	33.0	52.0	15.0	0.0	0.0	0.0	0.0	71.2
80	33.0	52.0	15.0	0.0	0.0	0.0	0.0	71.2
81	33.0	52.0	15.0	0.0	0.0	0.0	0.0	71.2
82	38.3	42.5	9.2	0.0	0.0	10.0	0.0	71.9
83	35.0	52.5	7.5	0.0	0.0	5.0	0.0	71.9
84	31.0	44.0	15.0	0.0	10.0	0.0	0.0	72.0
85	29.0	36.0	15.0	0.0	20.0	0.0	0.0	72.0
86	31.0	44.0	15.0	0.0	10.0	0.0	0.0	72.0
87	29.0	36.0	15.0	0.0	20.0	0.0	0.0	72.0
88	31.0	44.0	15.0	0.0	10.0	0.0	0.0	72.0

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
89	29.0	36.0	15.0	0.0	20.0	0.0	0.0	72.0
90	40.0	35.0	15.0	0.0	0.0	10.0	0.0	72.2
91	36.0	54.0	0.0	0.0	0.0	10.0	0.0	72.6
92	35.0	45.0	15.0	0.0	0.0	5.0	0.0	73.5
93	0.0	0.0	0.0	0.0	100.0	0.0	0.0	73.6
94	32.0	48.0	20.0	0.0	0.0	0.0	0.0	73.6
95	0.0	0.0	0.0	0.0	100.0	0.0	0.0	73.6
96	32.0	48.0	20.0	0.0	0.0	0.0	0.0	73.6
97	30.0	55.0	15.0	0.0	0.0	0.0	0.0	74.2
98	25.0	65.0	0.0	10.0	0.0	0.0	0.0	74.2
99	49.0	21.0	0.0	0.0	0.0	30.0	0.0	74.2
100	25.0	65.0	0.0	0.0	10.0	0.0	0.0	74.2
101	25.0	65.0	0.0	0.0	10.0	0.0	0.0	74.2
102	23.0	57.0	0.0	0.0	20.0	0.0	0.0	74.6
103	23.0	57.0	0.0	0.0	20.0	0.0	0.0	74.6
104	23.0	57.0	0.0	0.0	20.0	0.0	0.0	74.6
105	35.0	47.5	7.5	0.0	0.0	10.0	0.0	74.9
106	25.0	75.0	0.0	0.0	0.0	0.0	0.0	75.0
107	36.0	24.0	40.0	0.0	0.0	0.0	0.0	75.1
108	36.0	24.0	40.0	0.0	0.0	0.0	0.0	75.1
109	40.0	10.0	50.0	0.0	0.0	0.0	0.0	75.5
110	40.0	10.0	50.0	0.0	0.0	0.0	0.0	75.5
111	41.9	0.0	58.1	0.0	0.0	0.0	0.0	75.6
112	42.0	0.0	58.0	0.0	0.0	0.0	0.0	75.6
113	54.0	6.0	0.0	0.0	0.0	40.0	0.0	75.6
114	42.0	0.0	58.0	0.0	0.0	0.0	0.0	75.6
115	40.0	40.0	0.0	0.0	0.0	20.0	0.0	75.8
116	33.3	33.4	33.3	0.0	0.0	0.0	0.0	76.2
117	33.3	33.3	33.3	0.0	0.0	0.0	0.0	76.2
118	33.3	33.3	33.3	0.0	0.0	0.0	0.0	76.2
119	30.0	57.5	7.5	0.0	0.0	5.0	0.0	76.8
120	40.0	0.0	60.0	0.0	0.0	0.0	0.0	77.0
121	43.3	4.5	32.2	0.0	0.0	20.0	0.0	78.3
122	30.0	60.0	0.0	0.0	0.0	10.0	0.0	78.7
123	28.0	42.0	30.0	0.0	0.0	0.0	0.0	79.0
124	28.0	42.0	30.0	0.0	0.0	0.0	0.0	79.0
125	20.0	80.0	0.0	0.0	0.0	0.0	0.0	80.0
126	38.0	0.0	62.0	0.0	0.0	0.0	0.0	80.5
127	42.0	28.0	0.0	0.0	0.0	30.0	0.0	80.6
128	42.0	28.0	0.0	0.0	0.0	30.0	0.0	80.6
129	42.0	28.0	0.0	0.0	0.0	30.0	0.0	80.7
130	48.0	12.0	0.0	0.0	0.0	40.0	0.0	80.7
131	48.0	12.0	0.0	0.0	0.0	40.0	0.0	80.7
132	39.9	10.0	30.1	0.0	0.0	20.1	0.0	80.8
133	27.0	63.0	0.0	0.0	0.0	10.0	0.0	80.9
134	37.2	27.0	15.8	0.0	0.0	20.0	0.0	81.0
135	0.0	0.0	0.0	0.0	90.0	10.0	0.0	81.0
136	0.0	0.0	0.0	0.0	90.0	10.0	0.0	81.0

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
137	30.0	45.0	15.0	0.0	0.0	10.0	0.0	81.6
138	30.0	45.0	15.0	0.0	0.0	10.0	0.0	81.6
139	30.0	47.5	10.0	0.0	0.0	12.6	0.0	81.6
140	36.0	0.0	64.0	0.0	0.0	0.0	0.0	82.3
141	34.0	37.8	8.2	0.0	0.0	20.0	0.0	82.5
142	19.0	76.0	0.0	0.0	0.0	0.0	5.0	83.1
143	32.0	48.0	0.0	0.0	0.0	20.0	0.0	83.5
144	32.0	48.0	0.0	0.0	0.0	20.0	0.0	83.5
145	30.0	20.0	50.0	0.0	0.0	0.0	0.0	83.8
146	30.0	20.0	50.0	0.0	0.0	0.0	0.0	83.8
147	50.0	0.0	0.0	0.0	0.0	50.0	0.0	83.8
148	50.0	0.0	0.0	0.0	0.0	50.0	0.0	83.8
149	29.9	37.5	20.0	0.0	0.0	12.6	0.0	83.8
150	35.0	0.0	65.0	0.0	0.0	0.0	0.0	83.9
151	42.9	19.0	0.0	0.0	0.0	38.1	0.0	84.4
152	18.0	72.0	10.0	0.0	0.0	0.0	0.0	84.5
153	18.0	72.0	10.0	0.0	0.0	0.0	0.0	84.5
154	19.0	76.0	0.0	0.0	0.0	5.0	0.0	84.9
155	19.0	73.0	4.0	0.0	0.0	4.0	0.0	84.9
156	15.0	85.0	0.0	0.0	0.0	0.0	0.0	85.0
157	17.5	70.5	12.0	0.0	0.0	0.0	0.0	85.1
158	20.0	58.0	22.1	0.0	0.0	0.0	0.0	85.1
159	22.5	62.0	7.5	0.0	0.0	8.0	0.0	85.1
160	34.0	0.0	66.0	0.0	0.0	0.0	0.0	85.2
161	34.0	0.0	66.0	0.0	0.0	0.0	0.0	85.2
162	34.0	0.0	66.0	0.0	0.0	0.0	0.0	85.2
163	34.0	0.0	66.0	0.0	0.0	0.0	0.0	85.2
164	30.0	10.0	60.0	0.0	0.0	0.0	0.0	85.3
165	30.0	10.0	60.0	0.0	0.0	0.0	0.0	85.3
166	29.9	27.5	30.1	0.0	0.0	12.5	0.0	85.3
167	42.0	18.0	0.0	0.0	0.0	40.0	0.0	85.5
168	16.2	74.2	9.7	0.0	0.0	0.0	0.0	85.7
169	16.0	74.0	10.0	0.0	0.0	0.0	0.0	85.7
170	16.2	74.1	9.7	0.0	0.0	0.0	0.0	85.7
171	20.0	65.0	10.0	0.0	0.0	5.0	0.0	86.1
172	24.0	36.0	40.0	0.0	0.0	0.0	0.0	86.2
173	24.0	36.0	40.0	0.0	0.0	0.0	0.0	86.2
174	18.0	72.0	0.0	0.0	0.0	0.0	10.0	86.3
175	35.0	35.0	0.0	0.0	0.0	30.0	0.0	86.4
176	16.6	69.3	14.1	0.0	0.0	0.0	0.0	86.6
177	17.0	69.0	14.0	0.0	0.0	0.0	0.0	86.6
178	16.6	69.2	14.2	0.0	0.0	0.0	0.0	86.6
179	16.6	66.7	16.7	0.0	0.0	0.0	0.0	87.0
180	17.0	67.0	16.0	0.0	0.0	0.0	0.0	87.0
181	21.2	50.5	28.3	0.0	0.0	0.0	0.0	87.0
182	17.0	69.0	14.0	0.0	0.0	0.0	0.0	87.0
183	16.7	66.7	16.7	0.0	0.0	0.0	0.0	87.0
184	17.0	69.0	14.0	0.0	0.0	0.0	0.0	87.0

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
185	16.7	66.7	16.7	0.0	0.0	0.0	0.0	87.0
186	45.0	5.0	0.0	0.0	0.0	50.0	0.0	87.6
187	29.9	29.9	20.0	0.0	0.0	20.1	0.0	87.9
188	17.0	63.0	20.0	0.0	0.0	0.0	0.0	88.0
189	0.0	50.0	0.0	0.0	50.0	0.0	0.0	88.2
190	16.0	64.0	20.0	0.0	0.0	0.0	0.0	89.1
191	16.0	64.0	20.0	0.0	0.0	0.0	0.0	89.1
192	17.0	68.0	0.0	0.0	0.0	0.0	15.0	89.1
193	0.0	0.0	0.0	0.0	75.0	25.0	0.0	89.2
194	0.0	0.0	0.0	0.0	75.0	25.0	0.0	89.2
195	30.0	0.0	70.0	0.0	0.0	0.0	0.0	89.3
196	30.0	0.0	70.0	0.0	0.0	0.0	0.0	89.3
197	30.0	0.0	70.0	0.0	0.0	0.0	0.0	89.3
198	30.0	0.0	70.0	0.0	0.0	0.0	0.0	89.3
199	17.5	52.5	30.0	0.0	0.0	0.0	0.0	89.5
200	18.0	72.0	0.0	0.0	0.0	10.0	0.0	89.5
201	17.5	52.5	30.0	0.0	0.0	0.0	0.0	89.5
202	17.5	52.5	30.0	0.0	0.0	0.0	0.0	89.5
203	18.0	72.0	0.0	0.0	0.0	10.0	0.0	89.5
204	18.0	72.0	0.0	0.0	0.0	10.0	0.0	89.7
205	10.0	90.0	0.0	0.0	0.0	0.0	0.0	90.0
206	20.0	44.9	30.1	0.0	0.0	5.0	0.0	90.2
207	24.0	56.0	0.0	0.0	0.0	20.0	0.0	90.3
208	36.0	24.0	0.0	0.0	0.0	40.0	0.0	90.4
209	36.0	24.0	0.0	0.0	0.0	40.0	0.0	90.4
210	36.0	24.0	0.0	0.0	0.0	40.0	0.0	90.5
211	12.5	72.5	15.0	0.0	0.0	0.0	0.0	90.5
212	12.5	72.5	15.0	0.0	0.0	0.0	0.0	90.5
213	12.5	72.5	15.0	0.0	0.0	0.0	0.0	90.5
214	11.0	39.0	30.0	0.0	20.0	0.0	0.0	90.9
215	10.0	31.3	37.5	0.0	21.2	0.0	0.0	90.9
216	11.0	39.0	30.0	0.0	20.0	0.0	0.0	90.9
217	11.0	39.0	30.0	0.0	20.0	0.0	0.0	90.9
218	12.4	72.6	15.0	0.0	0.0	0.0	0.0	91.0
219	9.0	91.0	0.0	0.0	0.0	0.0	0.0	91.0
220	9.0	91.0	0.0	0.0	0.0	0.0	0.0	91.0
221	12.0	73.0	15.0	0.0	0.0	0.0	0.0	91.0
222	20.0	35.0	45.0	0.0	0.0	0.0	0.0	91.0
223	20.3	34.7	45.0	0.0	0.0	0.0	0.0	91.1
224	10.0	65.0	15.0	0.0	10.0	0.0	0.0	91.2
225	10.0	65.0	15.0	0.0	10.0	0.0	0.0	91.2
226	10.0	65.0	15.0	0.0	10.0	0.0	0.0	91.2
227	17.0	53.2	29.8	0.0	0.0	0.0	0.0	91.3
228	14.0	46.5	32.0	0.0	7.5	0.0	0.0	91.4
229	13.9	46.5	31.9	0.0	7.7	0.0	0.0	91.4
230	13.9	46.5	31.9	0.0	7.7	0.0	0.0	91.4
231	17.0	53.0	30.0	0.0	0.0	0.0	0.0	91.4
232	40.0	10.0	0.0	0.0	0.0	50.0	0.0	91.5

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
233	40.0	10.0	0.0	0.0	0.0	50.0	0.0	91.5
234	7.0	58.0	15.0	0.0	20.0	0.0	0.0	91.7
235	17.0	43.0	30.0	10.0	0.0	0.0	0.0	91.7
236	7.0	58.0	15.0	0.0	20.0	0.0	0.0	91.7
237	7.0	58.0	15.0	0.0	20.0	0.0	0.0	91.7
238	28.0	42.0	0.0	0.0	0.0	30.0	0.0	92.0
239	19.3	36.6	44.1	0.0	0.0	0.0	0.0	92.0
240	18.0	62.0	0.0	0.0	0.0	20.0	0.0	92.0
241	28.0	42.0	0.0	0.0	0.0	30.0	0.0	92.0
242	20.0	30.0	50.0	0.0	0.0	0.0	0.0	92.1
243	20.0	30.0	50.0	0.0	0.0	0.0	0.0	92.1
244	11.6	54.9	18.9	0.0	4.7	9.9	0.0	92.1
245	26.6	0.0	73.4	0.0	0.0	0.0	0.0	92.3
246	27.0	0.0	73.0	0.0	0.0	0.0	0.0	92.3
247	26.6	0.0	73.4	0.0	0.0	0.0	0.0	92.3
248	20.0	47.4	20.0	0.0	0.0	12.6	0.0	92.3
249	0.0	0.0	50.0	0.0	50.0	0.0	0.0	92.5
250	9.5	85.5	0.0	0.0	0.0	0.0	5.0	92.5
251	14.0	56.0	30.0	0.0	0.0	0.0	0.0	92.8
252	14.0	56.0	30.0	0.0	0.0	0.0	0.0	92.8
253	5.1	84.7	0.0	0.0	10.2	0.0	0.0	92.9
254	18.0	32.0	30.0	20.0	0.0	0.0	0.0	92.9
255	5.1	84.7	0.0	0.0	10.2	0.0	0.0	92.9
256	5.1	84.7	0.0	0.0	10.2	0.0	0.0	92.9
257	16.5	43.5	39.9	0.0	0.0	0.0	0.0	93.0
258	14.6	51.7	33.7	0.0	0.0	0.0	0.0	93.0
259	3.0	77.0	0.0	0.0	20.0	0.0	0.0	93.0
260	13.0	62.0	15.0	10.0	0.0	0.0	0.0	93.0
261	16.8	43.6	39.6	0.0	0.0	0.0	0.0	93.0
262	14.9	51.5	33.7	0.0	0.0	0.0	0.0	93.0
263	3.0	77.0	0.0	0.0	20.0	0.0	0.0	93.0
264	3.0	77.0	0.0	0.0	20.0	0.0	0.0	93.0
265	16.5	43.5	40.0	0.0	0.0	0.0	0.0	93.0
266	14.7	51.5	33.8	0.0	0.0	0.0	0.0	93.0
267	26.0	0.0	74.0	0.0	0.0	0.0	0.0	93.4
268	26.0	0.0	74.0	0.0	0.0	0.0	0.0	93.4
269	26.0	0.0	74.0	0.0	0.0	0.0	0.0	93.4
270	26.0	0.0	74.0	0.0	0.0	0.0	0.0	93.4
271	9.8	72.3	17.9	0.0	0.0	0.0	0.0	93.7
272	10.0	72.0	18.0	0.0	0.0	0.0	0.0	93.7
273	17.0	68.0	0.0	0.0	0.0	15.0	0.0	93.7
274	9.9	72.2	17.9	0.0	0.0	0.0	0.0	93.7
275	7.0	53.0	12.0	14.0	14.0	0.0	0.0	93.8
276	9.5	85.5	0.0	0.0	0.0	5.0	0.0	94.0
277	9.0	81.0	0.0	10.0	0.0	0.0	0.0	94.1
278	9.5	85.5	0.0	0.0	0.0	5.0	0.0	94.1
279	9.5	85.5	0.0	0.0	0.0	5.0	0.0	94.1
280	25.0	0.0	75.0	0.0	0.0	0.0	0.0	94.2

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
281	32.5	3.4	24.1	0.0	0.0	40.0	0.0	94.4
282	14.0	51.0	15.0	20.0	0.0	0.0	0.0	94.5
283	30.0	30.0	0.0	0.0	0.0	40.0	0.0	94.5
284	40.0	0.0	0.0	0.0	0.0	60.0	0.0	94.7
285	35.0	15.0	0.0	0.0	0.0	50.0	0.0	94.7
286	40.0	0.0	0.0	0.0	0.0	60.0	0.0	94.7
287	20.0	49.9	10.0	0.0	0.0	20.1	0.0	94.7
288	13.5	48.4	38.1	0.0	0.0	0.0	0.0	94.8
289	9.0	81.0	0.0	0.0	0.0	0.0	10.0	94.9
290	20.0	20.0	60.0	0.0	0.0	0.0	0.0	95.0
291	5.0	95.0	0.0	0.0	0.0	0.0	0.0	95.0
292	20.0	20.0	60.0	0.0	0.0	0.0	0.0	95.0
293	10.0	65.1	24.9	0.0	0.0	0.0	0.0	95.2
294	10.0	65.0	25.0	0.0	0.0	0.0	0.0	95.2
295	10.0	65.0	25.0	0.0	0.0	0.0	0.0	95.2
296	15.0	53.9	21.1	0.0	0.0	10.0	0.0	95.5
297	18.3	31.2	40.5	0.0	0.0	10.0	0.0	96.0
298	18.3	31.2	40.5	0.0	0.0	10.0	0.0	96.0
299	8.5	69.4	22.1	0.0	0.0	0.0	0.0	96.1
300	14.9	35.0	50.0	0.0	0.0	0.0	0.0	96.3
301	13.7	42.8	43.5	0.0	0.0	0.0	0.0	96.3
302	15.0	35.0	50.0	0.0	0.0	0.0	0.0	96.3
303	13.9	42.6	43.6	0.0	0.0	0.0	0.0	96.3
304	15.0	35.0	50.0	0.0	0.0	0.0	0.0	96.3
305	13.7	42.8	43.5	0.0	0.0	0.0	0.0	96.3
306	10.5	75.5	6.0	0.0	0.0	8.0	0.0	96.3
307	10.0	80.0	0.0	0.0	0.0	10.0	0.0	96.4
308	0.0	0.0	0.0	0.0	60.0	40.0	0.0	96.5
309	0.0	0.0	0.0	0.0	60.0	40.0	0.0	96.5
310	36.0	4.0	0.0	0.0	0.0	60.0	0.0	96.6
311	11.9	48.0	40.0	0.0	0.0	0.0	0.0	96.7
312	12.0	48.0	40.0	0.0	0.0	0.0	0.0	96.7
313	27.9	20.2	11.9	0.0	0.0	40.0	0.0	96.8
314	21.0	5.0	73.9	0.0	0.0	0.0	0.0	96.9
315	21.0	5.0	74.0	0.0	0.0	0.0	0.0	96.9
316	21.0	5.0	74.0	0.0	0.0	0.0	0.0	96.9
317	21.0	5.0	74.0	0.0	0.0	0.0	0.0	96.9
318	15.3	47.9	26.8	0.0	0.0	10.0	0.0	97.0
319	16.0	64.0	0.0	0.0	0.0	20.0	0.0	97.0
320	16.0	64.0	0.0	0.0	0.0	20.0	0.0	97.0
321	15.3	47.9	26.8	0.0	0.0	10.0	0.0	97.0
322	19.9	29.9	30.0	0.0	0.0	20.1	0.0	97.0
323	8.5	76.5	0.0	0.0	0.0	0.0	15.0	97.1
324	21.0	49.0	0.0	0.0	0.0	30.0	0.0	97.4
325	9.0	71.0	0.0	20.0	0.0	0.0	0.0	97.5
326	9.0	81.0	0.0	0.0	0.0	10.0	0.0	97.6
327	9.0	81.0	0.0	0.0	0.0	10.0	0.0	97.6
328	20.8	0.0	79.2	0.0	0.0	0.0	0.0	97.7

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
329	21.0	0.0	79.0	0.0	0.0	0.0	0.0	97.7
330	20.8	0.0	79.2	0.0	0.0	0.0	0.0	97.7
331	25.5	28.4	6.1	0.0	0.0	40.0	0.0	97.8
332	11.2	65.3	13.5	0.0	0.0	10.0	0.0	97.8
333	9.0	81.0	0.0	0.0	0.0	10.0	0.0	97.8
334	11.2	65.3	13.5	0.0	0.0	10.0	0.0	97.8
335	30.0	20.0	0.0	0.0	0.0	50.0	0.0	97.9
336	30.0	20.0	0.0	0.0	0.0	50.0	0.0	97.9
337	30.0	20.0	0.0	0.0	0.0	50.0	0.0	97.9
338	16.6	16.7	66.6	0.0	0.0	0.0	0.0	98.0
339	16.7	16.7	66.7	0.0	0.0	0.0	0.0	98.0
340	16.7	16.7	66.7	0.0	0.0	0.0	0.0	98.0
341	16.7	16.7	66.7	0.0	0.0	0.0	0.0	98.0
342	20.0	0.0	80.0	0.0	0.0	0.0	0.0	98.6
343	8.1	81.9	0.0	0.0	0.0	10.0	0.0	98.7
344	8.1	81.9	0.0	0.0	0.0	10.0	0.0	98.7
345	10.2	37.8	12.0	0.0	0.0	40.0	0.0	98.8
346	24.0	36.0	0.0	0.0	0.0	40.0	0.0	98.9
347	24.0	36.0	0.0	0.0	0.0	40.0	0.0	98.9
348	32.0	8.0	0.0	0.0	0.0	60.0	0.0	99.1
349	32.0	8.0	0.0	0.0	0.0	60.0	0.0	99.1
350	24.4	11.2	24.9	0.0	0.0	39.4	0.0	99.2
351	16.0	10.0	74.0	0.0	0.0	0.0	0.0	99.8
352	9.9	40.0	50.0	0.0	0.0	0.0	0.0	99.8
353	16.0	10.0	74.0	0.0	0.0	0.0	0.0	99.8
354	10.0	40.0	50.0	0.0	0.0	0.0	0.0	99.8
355	16.1	10.3	73.6	0.0	0.0	0.0	0.0	99.8
356	16.0	10.0	74.0	0.0	0.0	0.0	0.0	99.8
357	16.0	10.0	74.0	0.0	0.0	0.0	0.0	99.8
358	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0
359	0.0	100.0	0.0	0.0	0.0	0.0	0.0	100.0
360	0.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0
361	16.2	27.8	36.0	0.0	0.0	20.0	0.0	100.2
362	16.2	27.8	36.0	0.0	0.0	20.0	0.0	100.2
363	8.5	76.5	0.0	0.0	0.0	15.0	0.0	100.8
364	0.0	0.0	0.0	90.0	0.0	10.0	0.0	101.0
365	16.0	4.0	80.0	0.0	0.0	0.0	0.0	101.0
366	0.0	0.0	0.0	90.0	0.0	10.0	0.0	101.0
367	13.6	42.6	23.8	0.0	0.0	20.0	0.0	101.4
368	13.6	42.6	23.8	0.0	0.0	20.0	0.0	101.4
369	10.0	30.0	60.0	0.0	0.0	0.0	0.0	101.6
370	30.0	0.0	0.0	0.0	0.0	70.0	0.0	101.6
371	16.2	38.6	21.7	0.0	0.0	23.5	0.0	101.6
372	30.0	0.0	0.0	0.0	0.0	70.0	0.0	101.6
373	0.0	0.0	0.0	0.0	40.0	60.0	0.0	101.7
374	0.0	0.0	0.0	0.0	40.0	60.0	0.0	101.7
375	0.0	89.9	10.1	0.0	0.0	0.0	0.0	101.9
376	0.0	90.0	10.0	0.0	0.0	0.0	0.0	102.0

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
377	0.0	90.0	10.0	0.0	0.0	0.0	0.0	102.0
378	0.0	95.0	0.0	0.0	0.0	0.0	5.0	102.1
379	0.0	0.0	0.0	75.0	0.0	25.0	0.0	102.3
380	0.0	0.0	0.0	75.0	0.0	25.0	0.0	102.3
381	24.0	16.0	0.0	0.0	0.0	60.0	0.0	102.5
382	9.9	58.1	12.0	0.0	0.0	20.0	0.0	102.6
383	9.9	58.1	12.0	0.0	0.0	20.0	0.0	102.6
384	24.0	16.0	0.0	0.0	0.0	60.0	0.0	102.7
385	24.0	16.0	0.0	0.0	0.0	60.0	0.0	102.7
386	12.0	8.0	80.0	0.0	0.0	0.0	0.0	103.1
387	0.0	0.0	0.0	60.0	0.0	40.0	0.0	103.3
388	11.0	15.0	74.0	0.0	0.0	0.0	0.0	103.3
389	0.0	0.0	0.0	60.0	0.0	40.0	0.0	103.3
390	11.0	15.0	74.0	0.0	0.0	0.0	0.0	103.3
391	11.0	15.0	74.0	0.0	0.0	0.0	0.0	103.3
392	21.7	2.2	16.1	0.0	0.0	60.0	0.0	103.4
393	8.0	72.0	0.0	0.0	0.0	20.0	0.0	103.6
394	8.0	72.0	0.0	0.0	0.0	20.0	0.0	103.6
395	7.2	72.8	0.0	0.0	0.0	20.0	0.0	103.8
396	7.2	72.8	0.0	0.0	0.0	20.0	0.0	103.8
397	0.0	95.0	0.0	0.0	0.0	5.0	0.0	103.9
398	0.0	80.0	20.0	0.0	0.0	0.0	0.0	104.1
399	0.0	80.0	20.0	0.0	0.0	0.0	0.0	104.1
400	0.0	90.0	0.0	0.0	0.0	0.0	10.0	104.2
401	18.6	13.5	7.9	0.0	0.0	60.0	0.0	104.4
402	12.2	20.8	27.0	0.0	0.0	40.0	0.0	104.6
403	12.2	20.8	27.0	0.0	0.0	40.0	0.0	104.6
404	20.0	0.0	0.0	0.0	0.0	80.0	0.0	104.7
405	20.0	0.0	0.0	0.0	0.0	80.0	0.0	104.7
406	18.0	12.0	0.0	0.0	0.0	70.0	0.0	104.8
407	0.0	0.0	0.0	40.0	0.0	60.0	0.0	104.8
408	0.0	0.0	0.0	40.0	0.0	60.0	0.0	104.8
409	0.0	70.0	30.0	0.0	0.0	0.0	0.0	105.0
410	17.0	18.9	4.1	0.0	0.0	60.0	0.0	105.2
411	12.0	8.0	0.0	0.0	0.0	80.0	0.0	105.3
412	8.0	12.0	80.0	0.0	0.0	0.0	0.0	105.4
413	16.0	24.0	0.0	0.0	0.0	60.0	0.0	105.5
414	16.0	24.0	0.0	0.0	0.0	60.0	0.0	105.5
415	0.0	70.0	30.0	0.0	0.0	0.0	0.0	105.6
416	0.0	70.0	30.0	0.0	0.0	0.0	0.0	105.6
417	12.0	48.0	0.0	0.0	0.0	40.0	0.0	105.7
418	12.0	48.0	0.0	0.0	0.0	40.0	0.0	105.7
419	16.0	4.0	0.0	0.0	0.0	80.0	0.0	105.8
420	16.0	4.0	0.0	0.0	0.0	80.0	0.0	105.8
421	10.0	0.0	90.0	0.0	0.0	0.0	0.0	106.0
422	10.2	31.9	17.9	0.0	0.0	40.0	0.0	106.0
423	10.2	31.9	17.9	0.0	0.0	40.0	0.0	106.0
424	0.0	85.0	0.0	0.0	0.0	0.0	15.0	106.0

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
425	8.1	13.9	18.0	0.0	0.0	60.0	0.0	106.3
426	8.1	13.9	18.0	0.0	0.0	60.0	0.0	106.3
427	10.0	0.0	0.0	0.0	0.0	90.0	0.0	106.5
428	10.0	0.0	0.0	0.0	0.0	90.0	0.0	106.5
429	12.0	8.0	0.0	0.0	0.0	80.0	0.0	106.6
430	12.0	8.0	0.0	0.0	0.0	80.0	0.0	106.6
431	0.0	90.0	0.0	0.0	0.0	10.0	0.0	106.8
432	0.0	90.0	0.0	0.0	0.0	10.0	0.0	106.8
433	6.8	21.3	11.9	0.0	0.0	60.0	0.0	107.1
434	7.4	43.6	9.0	0.0	0.0	40.0	0.0	107.1
435	4.1	6.9	9.0	0.0	0.0	80.0	0.0	107.1
436	7.4	43.6	9.0	0.0	0.0	40.0	0.0	107.1
437	6.8	21.3	11.9	0.0	0.0	60.0	0.0	107.1
438	4.1	6.9	9.0	0.0	0.0	80.0	0.0	107.1
439	3.4	10.6	6.0	0.0	0.0	80.0	0.0	107.5
440	3.4	10.6	6.0	0.0	0.0	80.0	0.0	107.5
441	6.0	20.0	74.0	0.0	0.0	0.0	0.0	107.6
442	8.0	12.0	0.0	0.0	0.0	80.0	0.0	107.6
443	6.0	20.0	74.0	0.0	0.0	0.0	0.0	107.6
444	6.0	20.0	74.0	0.0	0.0	0.0	0.0	107.6
445	8.0	12.0	0.0	0.0	0.0	80.0	0.0	107.6
446	0.0	60.0	40.0	0.0	0.0	0.0	0.0	107.7
447	0.0	60.0	40.0	0.0	0.0	0.0	0.0	107.7
448	5.0	29.0	6.0	0.0	0.0	60.0	0.0	107.7
449	8.0	32.0	0.0	0.0	0.0	60.0	0.0	107.7
450	8.0	32.0	0.0	0.0	0.0	60.0	0.0	107.7
451	5.0	29.0	6.0	0.0	0.0	60.0	0.0	107.7
452	2.5	14.5	3.0	0.0	0.0	80.0	0.0	107.8
453	2.5	14.5	3.0	0.0	0.0	80.0	0.0	107.8
454	0.0	0.0	20.0	0.0	0.0	80.0	0.0	107.9
455	0.0	0.0	20.0	0.0	0.0	80.0	0.0	107.9
456	0.0	0.0	0.0	0.0	0.0	100.0	0.0	108.0
457	8.0	2.0	90.0	0.0	0.0	0.0	0.0	108.0
458	5.4	54.6	0.0	0.0	0.0	40.0	0.0	108.0
459	0.0	0.0	0.0	0.0	0.0	100.0	0.0	108.0
460	5.4	54.6	0.0	0.0	0.0	40.0	0.0	108.0
461	0.0	0.0	40.0	0.0	0.0	60.0	0.0	108.1
462	0.0	0.0	40.0	0.0	0.0	60.0	0.0	108.1
463	0.0	50.0	50.0	0.0	0.0	0.0	0.0	108.2
464	4.0	16.0	0.0	0.0	0.0	80.0	0.0	108.3
465	4.0	16.0	0.0	0.0	0.0	80.0	0.0	108.3
466	3.6	36.4	0.0	0.0	0.0	60.0	0.0	108.4
467	1.8	18.2	0.0	0.0	0.0	80.0	0.0	108.4
468	3.6	36.4	0.0	0.0	0.0	60.0	0.0	108.4
469	1.8	18.2	0.0	0.0	0.0	80.0	0.0	108.4
470	6.0	4.0	0.0	0.0	0.0	90.0	0.0	108.5
471	2.0	18.0	80.0	0.0	0.0	0.0	0.0	108.5
472	0.0	49.9	50.1	0.0	0.0	0.0	0.0	108.5

Table B.1 continued from previous page

	Paraffin	**I**soParaffin	**A**romatic	**N**aphthene	**O**lefin	Alcohol	Ether	RON
473	0.0	0.0	60.0	0.0	0.0	40.0	0.0	108.6
474	0.0	0.0	60.0	0.0	0.0	40.0	0.0	108.6
475	0.0	20.0	0.0	0.0	0.0	80.0	0.0	109.0
476	0.0	20.0	0.0	0.0	0.0	80.0	0.0	109.0
477	0.0	80.0	0.0	0.0	0.0	20.0	0.0	109.4
478	0.0	80.0	0.0	0.0	0.0	20.0	0.0	109.4
479	6.0	4.0	90.0	0.0	0.0	0.0	0.0	109.5
480	0.0	40.0	0.0	0.0	0.0	60.0	0.0	109.6
481	0.0	40.0	0.0	0.0	0.0	60.0	0.0	109.6
482	0.0	40.0	60.0	0.0	0.0	0.0	0.0	110.0
483	16.7	16.7	66.7	0.0	0.0	0.0	0.0	110.0
484	0.0	50.0	50.0	0.0	0.0	0.0	0.0	110.0
485	0.0	50.0	50.0	0.0	0.0	0.0	0.0	110.0
486	0.0	60.0	0.0	0.0	0.0	40.0	0.0	110.2
487	0.0	60.0	0.0	0.0	0.0	40.0	0.0	110.2
488	0.0	50.0	50.0	0.0	0.0	0.0	0.0	110.5
489	0.0	0.0	80.0	0.0	0.0	20.0	0.0	110.9
490	0.0	0.0	80.0	0.0	0.0	20.0	0.0	110.9
491	4.0	6.0	90.0	0.0	0.0	0.0	0.0	111.8
492	0.0	20.0	80.0	0.0	0.0	0.0	0.0	112.6
493	0.0	0.0	90.0	0.0	0.0	10.0	0.0	112.8
494	0.0	0.0	90.0	0.0	0.0	10.0	0.0	112.8
495	0.0	26.0	74.0	0.0	0.0	0.0	0.0	113.0
496	0.0	26.0	74.0	0.0	0.0	0.0	0.0	113.0
497	0.0	25.9	74.1	0.0	0.0	0.0	0.0	113.0
498	0.0	10.0	90.0	0.0	0.0	0.0	0.0	115.3
499	0.0	0.0	0.0	0.0	0.0	0.0	100.0	117.0
500	0.0	0.0	100.0	0.0	0.0	0.0	0.0	118.0

B.2 Mixture Analysis Sample Calculation

1. List the blend components and their respective volume compositions. The blend to be sampled has the following volume composition:
   - 72% n-Heptane.
   - 18% i-Octane.
   - 10% Toluene.
2. Pick a basis of 1L of the blend so to obtain the volume of each compound present.
   - $V_{\text{n-Heptane}} = 0.72 \times 1\text{L} = 0.72\text{L}$
   - $V_{\text{i-Octane}} = 0.18 \times 1\text{L} = 0.19\text{L}$
   - $V_{\text{Toluene}} = 0.10 \times 1\text{L} = 0.10\text{L}$
3. Multiply the compound volumes by their respective densities to obtain the mass of each compound present.
   - $M_{\text{n-Heptane}} = 0.72\text{L} \times 684\text{g/L} = 492.48\text{g}$

- $M_{\text{i-Octane}} = 0.18\text{L} \times 690\text{g/L} = 124.2\text{g}$
- $M_{\text{Toluene}} = 0.10\text{L} \times 867\text{g/L} = 86.7\text{g}$

4. Divide the mass of each compound by its molar mass to obtain the moles of each compound present.

   - $N_{\text{n-Heptane}} = 492.48\text{g} \div 100.21\text{g/mol} = 4.91\text{mol}$
   - $N_{\text{i-Octane}} = 124.2\text{g} \div 114.23\text{g/mol} = 1.087\text{mol}$
   - $N_{\text{Toluene}} = 86.7\text{g} \div 92.14\text{g/mol} = 0.941\text{mol}$

5. Determine the molar structural group composition of each compound (Calculation for n-Heptane shown as an example below).

   5.1. Count the number of occurrences of each structural group in each compound. The structure of n-Heptane is as follows: $CH_3 - CH_2 - CH_2 - CH_2 - CH_2 - CH_2 - CH_3$

   5.2. Take the occurrences as the molar ratios of the structural groups in each compound.

   - **2**: $-CH_3$
   - **5**: $-CH_2-$

   $-CH_3$ to $-CH_2-$ ratio $= 2 : 5$

   5.3. Sum the ratios to get the total number of moles of structural group present in a single compound. $N_{\text{Total}} = 2 + 5 = 7\text{mol}$

Table B.2: Summation of structural group molar ratios

Structural Group	**n-Heptane**	**i-Octane**	**Toluene**
$-CH_3$	2	3	1
$-CH_2-$	5	4	0
$-CH<$ (C# = 2)	0	1	0
$=CH-$(Aromatic)	0	0	5
$-CH<$ (Aromatic)	0	0	1
Total moles of structural groups in compound	7	8	7

   5.4. Divide the ratios by the total number of moles to obtain molar structural composition of each compound (n-Heptane as example).

   - $C_{-CH_3} = \frac{2}{7} = 0.286$
   - $C_{-CH_2-} = \frac{5}{7} = 0.714$

Table B.3: Molar Structural Group Compositions

Structural Group	**n-Heptane**	**i-Octane**	**Toluene**
—CH_3	0.286	0.375	0.143
—CH_2—	0.714	0.5	0
—CH< (C# = 2)	0	0.125	0
=CH— (Aromatic)	0	0	0.714
—CH< (Aromatic)	0	0	0.143
Total Composition	1	1	1

6. Multiply the moles of each compound by its molar structural group composition to get the moles of each structural group in the compound. For n-Heptane
   - $N_{-CH_3} = 0.286 \times 4.91\text{mol} = 1.404\text{mol}$
   - $N_{-CH_2-} = 0.714 \times 1.087\text{mol} = 3.506\text{mol}$

Table B.4: Moles of structural groups in each compound for 1L of gasoline

Structural Group	**n-Heptane**	**i-Octane**	**Toluene**
—CH_3	1.404	0.4080	0.135
—CH_2—	3.506	0.5435	0
—CH< (C# = 2)	0	0.1355	0
=CH— (Aromatic)	0	0	0.672
—CH< (Aromatic)	0	0	0.134
Total Composition	4.91	1.087	0.941

7. Sum the moles of the structural groups present in each compound to obtain the total number of moles present in the compound. For example —CH_3 $T_{-CH_3} = 1.404\text{mol} + 0.4080\text{mol} + 0.135\text{mol}$

Table B.5: Moles of structural group compositions in 1L of gasoline

Structural Group	**Total moles in blend**
—CH_3	1.947
—CH_2—	4.0495
—CH< (C# = 2)	0.1355
=CH— (Aromatic)	0.672
—CH< (Aromatic)	0.134
Total Moles	6.938

8. Divide the number of moles of each structural group by the total moles present in a single compound to obtain the molar structural group composition of each blend.

Table B.6: Molar structural group composition of gasoline blend

Structural Group	**Total moles in blend**
—CH_3	0.28
—CH_2—	0.584
—CH< (C# = 2)	0.0195
=CH— (Aromatic)	0.0969
—CH< (Aromatic)	0.0196
Total Moles	1

C R and Python Codes Used

C.1 R Code for Data conversion

```
#1. Load libraries.
library(tidyverse)
library(dplyr)

#2. Setwd and Load Data.
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/1. Data
    Conversion")
groups <- read.csv("Blend_Structural_Groups.csv")
data <- read.csv("Blend_Data.csv")
compounds <- read.csv("Blend_Compounds.csv")

#3. Mixture Analysis
##3.1 List the Blends and Compositions
groups_1 <- groups[3:9,]
colnames(groups_1) <-
    c("Compound","X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11","Y")
View(groups_1)
View(compounds)
##3.2 Pick a 1L Basis
colnames(data) <-
    c("heptane","octane","toluene","cyclopentane","hexene","ethanol","etbe","Y")
View(data)
compounds_1 <- compounds %>% select(Compound, MolarMass, Density)
colnames(compounds_1) <- c("Compound", "Molar_Mass","Density")
View(compounds_1)
##3.3 Multiply the compound volumes by their respective densities to obtain the
    mass of each compound present.
data_mass <- as.data.frame(matrix(0, nrow = 500, ncol = 7))
colnames(data_mass) <-
    c("heptane","octane","toluene","cyclopentane","hexene","ethanol","etbe")
for (i in 1:500) {
  data_mass[i,] <- data[i,1:7] * compounds_1$Density
}
##3.4 Divide the mass of each compound by its molar mass to obtain the
    moles of each compound present.
data_mol <- as.data.frame(matrix(0, nrow = 500, ncol = 7))
colnames(data_mol) <-
    c("heptane","octane","toluene","cyclopentane","hexene","ethanol","etbe")
for (i in 1:500) {
       data_mol[i,] <- data_mass[i,1:7] / compounds_1$Molar_Mass
}
###3.4.1 Find total moles of compound in each blend
mol_tot <- rowSums(data_mol)
###3.4.2 Divide Moles by total moles to obtaint the molar composition
mol_comp <- as.data.frame(matrix(0, nrow = 500, ncol = 7))
colnames(mol_comp) <-
    c("heptane","octane","toluene","cyclopentane","hexene","ethanol","etbe")
for (i in 1:500) {
  mol_comp[i,] <- data_mol[i,1:7] / mol_tot[i]
```

```
}
View(mol_comp)
##3.5 Determine the molar structural group composition of each compound.
###3.5.1 Count the number of occurrences of each structural group in each
    compound.
counts <- groups_1[,1:12]
View(counts)
###3.5.2 Take the occurrences as the molar ratios of the structural groups in
    each compound.
####Implicit in the counts data frame
###3.5.3 Sum the ratios to get the total number of moles of structural group
    present in a single compound.
compound_mol <- as.data.frame(matrix(0, nrow = 7, ncol = 2))
colnames(compound_mol) <- c("Compound","Total_Mol")
compound_mol$Compound <- counts$Compound

####Convert the compositions to numeric variables
cols.num <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
counts[cols.num] <- sapply(counts[cols.num],as.numeric)
sapply(counts, class)


for (i in 1:7) {
  compound_mol$Total_Mol[i] <- sum(counts[i,2:12])
}


View(compound_mol)


###3.5.4 Divide the ratios by the total number of moles to obtain molar
    structural group composition of each compound.
str_comp <- as.data.frame(matrix(0,nrow = 7, ncol = 12))
colnames(str_comp) <-
    c("Compound","X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:7) {
  str_comp[,1] <- counts[,1]
  str_comp[i,2:12] <-
      as.matrix(as.numeric(counts[i,2:12])/compound_mol$Total_Mol[i])
}

##3.6 Multiply the moles of each compound by its molar structural group
    composition to get the moles of each structural group in the compound.
heptane <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(heptane) <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  heptane[i,] <- data_mol[i,1]*str_comp[1,2:12]
}

octane <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(octane) <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  octane[i,] <- data_mol[i,2]*str_comp[2,2:12]
}
```

```
toluene <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(toluene) <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  toluene[i,] <- data_mol[i,3]*str_comp[3,2:12]
}

cyclopentane <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(cyclopentane) <-
    c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  cyclopentane[i,] <- data_mol[i,4]*str_comp[4,2:12]
}

hexene <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(hexene) <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  hexene[i,] <- data_mol[i,5]*str_comp[5,2:12]
}

ethanol <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(ethanol) <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  ethanol[i,] <- data_mol[i,6]*str_comp[6,2:12]
}

etbe <- as.data.frame(matrix(0, nrow = 500, ncol = 11))
colnames(etbe) <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11")
for (i in 1:500) {
  etbe[i,] <- data_mol[i,7]*str_comp[7,2:12]
}


##3.7 Sum the moles of the structural groups present in each compound to obtain
    the total number of moles present in the compound.
data_str <- heptane+octane+toluene+cyclopentane+hexene+ethanol+etbe


##3.8 Divide the number of  moles of  each structural group by  the  total moles
    present in  a single compound to obtain the molar structural group
    composition of each blend.
data_2 <- data_str/rowSums(data_str)
data_2$Y <- data$Y

#4. Save as new dataframe
write.csv(data_2,"Blend_Molar_Structural_Group_Composition.csv")
write.csv(mol_comp,"Blend_Molar_Compound_Composition.csv")

#5. Combine the structural group compositions and the compound compositions into
    1 data frame
compound_comp <- mol_comp
colnames(compound_comp) <- c("X12","X13","X14","X15","X16","X17","X18")


structural_comp <- data_2
comp_data <- cbind(structural_comp,compound_comp)
```

```r
comp_data <- comp_data[,c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10",
    "X11","X12","X13","X14","X15","X16","X17","X18","Y")]

#6. Save the new data frame
write.csv(comp_data,"Composition_Data.csv")
```

C.2 Code for Exploratory Data Analysis (EDA)

C.2.1 Python Code for Correlation Analysis

```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 8 06:08:19 2021

@author: vtspe
"""

# 1. Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns #visualisation
sns.set(color_codes=True)
import matplotlib.pyplot as plt #visualisation
from matplotlib.ticker import NullFormatter
import statsmodels.api as sm
import numpy as np
from statsmodels.graphics.gofplots import qqplot_2samples
import os

# 2. Set the working directory and import the files
abspath = os.path.abspath(r'C:\Users\vtspe\OneDrive - University of Cape
    Town\MSc Data Science\Data Science Research\STA5079Z - Dissertation\V2\Code
    and Data\2. EDA - Correlation Analysis') ## String which contains absolute
    path to the script file
os.chdir(abspath) ## Setting up working directory
df = pd.read_csv("Composition_Data.csv")

# 3. Check column names, variable types and consistency
print(df.dtypes)
print(df.head(5))
print(df.tail(5))

# 4. Drop the irrelevant unnamed columns
df = df.drop(["Unnamed: 0"], axis = 1)
print(df.head(5))

# 5. Check for duplicate rows and delete them
print(df.shape)
duplicate_rows_df = df[df.duplicated()]
print(duplicate_rows_df.shape)
df = df.drop_duplicates()
print(df.shape)

# 6. Check fro missing values and null values
print(df.isnull().sum())

# 7. Produce the heatmap
c = df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(c, cmap = "BrBG", annot = True)
```

```
plt.savefig('heat_map.pdf')

# 8. Sub select the columns of the df which are still valid and not identical
c2 = df[["X1","X2","X4","X12","X13","X14","X15","X16","X17","X18","Y"]].corr()
plt.figure(figsize=(20,10))
sns.heatmap(c2, cmap = "BrBG", annot = True)
plt.savefig('heat_map2.pdf')
df[["X1","X2","X4","X12","X13","X14","X15","X16","X17","X18",
    "Y"]].to_csv(r'Reduced_Data.csv')
```

C.2.2 Python Code for outlier Analysis

```
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 12 15:19:24 2021

@author: vtspe
"""

# 1. Import required libraries
import os

import pandas as pd
import numpy as np
from scipy.stats import chi2
from matplotlib import patches
import matplotlib.pyplot as plt

# 2. Set the working directory and import the files
abspath = os.path.abspath(r'C:\Users\vtspe\OneDrive - University of Cape
    Town\MSc Data Science\Data Science Research\STA5079Z - Dissertation\V2\Code
    and Data\4. EDA - Outlier Windsorization') ## String which contains absolute
    path to the script file
os.chdir(abspath) ## Setting up working directory
df = pd.read_csv("Reduced_Data.csv")
df.to_numpy()

# 3. Get the necessary values to calculate the distances between center and point
# Covariance matrix
covariance = np.cov(df , rowvar=False)

# Covariance matrix power of -1
covariance_pm1 = np.linalg.matrix_power(covariance, -1)

# Center point
centerpoint = np.mean(df , axis=0)

# Distances between center point and
distances = []
for i in range(len(df)):
      p1 = df.iloc[i,:]
      p2 = centerpoint
      distance = (p1-p2).T.dot(covariance_pm1).dot(p1-p2)
      distances.append(distance)
distances = np.array(distances)

# Cutoff (threshold) value from Chi-Sqaure Distribution for detecting outliers
cutoff = chi2.ppf(0.95, df.shape[1])

# Index of outliers
outlierIndexes = np.where(distances > cutoff )

print('--- Index of Outliers ----')
print(outlierIndexes)
```

```
##array([ 0,  16,  70, 100, 142, 145, 179, 225, 260, 265, 276, 282, 347, 358]

outliers = df.index.isin([16, 70, 100, 142, 145, 260, 265, 276, 282, 347, 358])
outs = df[outliers]
final = df[~outliers]

outs.to_csv('outliers.csv', index = False)
final.to_csv('finaldata.csv', index = False)
```

C.3 Code for Multiple Linear Regression (MLR)

C.3.1 R Code for MLR without mass balance constraint

```
#1. Load libraries
library(tidyverse)
library(glmnet)
library(ggplot2)
library(leaps)
library(corrr)
library(caret)
library(broom)
library(plotmo)
library(glinternet)

#2. Set working directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/6. Regression")
data <- read.csv("finaldata.csv")
View(data)

#3. Train test split

n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x.train <- train %>% select(-Y)
y.train <- train %>% select(Y)



x.val <- val %>% select(-Y)
y.val <- val %>% select(Y)


x.test <- test %>% select(-Y)
y.test <- test %>% select(Y)


#4.1 Train the model
mod1 <- lm(Y~., data = train)
sum.mod1 <- summary(mod1)
```

```
sum.mod1

#4.2 Validate the model
val.mod1 <- predict(mod1, newdata = val)
val.mod1.MSE <- mean((val.mod1-val$Y)^2)
val.mod1.MAE <- mean(abs(val.mod1-val$Y))
val.mod1.MSE
val.mod1.MAE

#4.3 Test the model
test.mod1 <- predict(mod1, newdata = test)
test.mod1.MSE <- mean((test.mod1-test$Y)^2)
test.mod1.MAE <- mean(abs(test.mod1-test$Y))
test.mod1.MSE
test.mod1.MAE

#4.4 Evaluate R squared and adjusted R squared
#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

mod1.rsq <- R.squared(val.mod1, val$Y)
k <- dim(train)[2]-1
n <- dim(val)[1]
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
mod1.adjrsq

View(mod1$coefficients)
View(mod1$residuals)

#4.5 Do the ANOVA
anovadata <- cbind(x.test,as.data.frame(test.mod1))
anovadata <- anovadata %>%
  rename(yhat = test.mod1)
View(anovadata)

mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)

#4.6 Variable importance
mod1.varimp <- varImp(mod1)
mod1.varimp

#4.7 Parity plots
plot_dat <- as.data.frame(cbind(test$Y, test.mod1))
```

```
names(plot_dat) <- c("Y_actual", "Y_hat")
View(plot_dat)

parity <- ggplot(plot_dat, aes(x=Y_actual, y=Y_hat)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))

pdf(file = "parity_MLR_no_Constraints.pdf")
parity
dev.off()

#4.8 Residual plot

res <- y.test - test.mod1
res_dat <- as.data.frame(cbind(res, as.data.frame(test.mod1)))
names(res_dat) <- c("res", "pred")
residual <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf(file = "residual_MLR_no_constraints.pdf")
residual
dev.off()

#4.9 QQ plot of residuals
res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "MLR_no_constraints_Residual_qq.pdf")
res_qq
dev.off()
```

C.3.2 Python Code for MLR with mass balance constraint

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import os
from statsmodels.formula.api import glm
import pandas as pd
from sklearn.model_selection import train_test_split

# 2. Set the working directory and import the files
abspath = os.path.abspath(r'C:\Users\vtspe\OneDrive - University of Cape
    Town\MSc Data Science\Data Science Research\STA5079Z - Dissertation\V2\Code
    and Data\6. Regression') ## String which contains absolute path to the
    script file
os.chdir(abspath) ## Setting up working directory
df = pd.read_csv("finaldata.csv")




# 3. Train test split
X_train, X_testval, y_train, y_testval =
    train_test_split(df[['X1','X2','X4','X12','X13','X14','X15','X16','X17','X18']],
    df[['Y']], test_size=0.4, random_state=11)
X_val, X_test, y_val, y_test = train_test_split(X_testval, y_testval,
    test_size=0.5, random_state=11)

# 4. Build the model

train = pd.concat([X_train, y_train], axis=1)
mod = glm("Y ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18", data=train)
res = mod.fit_constrained(["X12+X13+X14+X15+X16+X17+X18 = 1"])
res.summary()
mod_val = res.predict(X_val)
mod_val = pd.DataFrame(mod_val)

mod_val = mod_val.rename(columns={0: 'Y'})

val_MSE = ((mod_val-y_val)**2).mean()
val_MAE = (abs(mod_val-y_val)).mean()

mod_test = res.predict(X_test)
mod_test = pd.DataFrame(mod_test)

mod_test = mod_test.rename(columns={0: 'Y'})

test_MSE = ((mod_test-y_test)**2).mean()
test_MAE = (abs(mod_test-y_test)).mean()

res_params = pd.DataFrame(res.params)
```

```
res_params = res_params.rename(columns={0: 'Y'})


res_params = res_params.drop(labels='Intercept', axis=0)

varimp = abs(res_params['Y'])/max(abs(res_params['Y']))

val = pd.concat([X_val, y_val], axis = 1)
test = pd.concat([X_test, y_test], axis = 1)

train.to_csv('mod2_train.csv')
val.to_csv('mod2_val.csv')
test.to_csv('mod2_test.csv')

val_pred = pd.concat([X_val, mod_val], axis = 1)
test_pred = pd.concat([X_test, mod_test], axis = 1)

val_pred.to_csv('mod2_valpred.csv')
test_pred.to_csv('mod2_testpred.csv')
```

C.3.3 R Code for MLR with mass balance constraint ANOVA

```
#1. Load libraries
library(tidyverse)
library(glmnet)
library(ggplot2)
library(leaps)
library(corrr)
library(caret)
library(broom)
library(plotmo)


#2. Set working directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/6. Regression")
train <- read.csv("mod2_train.csv")
val <- read.csv("mod2_val.csv")
test <- read.csv("mod2_test.csv")
valpred <- read.csv("mod2_valpred.csv")
testpred <- read.csv("mod2_testpred.csv")


#3. Train test split

x.train <- train %>% select(-Y,-X)
y.train <- train %>% select(Y)

x.val <- val %>% select(-Y,-X)
y.val <- val %>% select(Y)

x.test <- test %>% select(-Y,-X)
y.test <- test %>% select(Y)


#4.1 Train the model


#4.2 Validate the model
val.mod1 <- valpred$Y
val.mod1.MSE <- mean((val.mod1-val$Y)^2)
val.mod1.MAE <- mean(abs(val.mod1-val$Y))
val.mod1.MSE
val.mod1.MAE

#4.3 Test the model
test.mod1 <- testpred$Y
test.mod1.MSE <- mean((test.mod1-test$Y)^2)
test.mod1.MAE <- mean(abs(test.mod1-test$Y))
test.mod1.MSE
test.mod1.MAE

#4.4 Evaluate R squared and adjusted R squared
#R-Squared
```

```
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

mod1.rsq <- R.squared(val.mod1, val$Y)
k <- dim(train)[2]-1
n <- dim(val)[1]
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
mod1.adjrsq


#4.5 Do the ANOVA
anovadata <- testpred
anovadata <- anovadata %>%
  rename(yhat = Y)
View(anovadata)

mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)


#4.7 Parity plots
plot_dat <- as.data.frame(cbind(test$Y, testpred$Y))
names(plot_dat) <- c("Y_actual", "Y_hat")
View(plot_dat)

parity <- ggplot(plot_dat, aes(x=Y_actual, y=Y_hat)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))

pdf(file = "parity_MLR_Constraints.pdf")
parity
dev.off()

#4.8 Residual plot

res <- y.test - testpred$Y
res_dat <- as.data.frame(cbind(res, testpred$Y))
names(res_dat) <- c("res", "pred")
residual <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
```

```
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf(file = "residual_MLR_constraints.pdf")
residual
dev.off()

#4.9 QQ plot of residuals
res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "MLR_constraints_Residual_qq.pdf")
res_qq
dev.off()
```

C.3.4 R Code for Interaction MLR without mass balance Constraint

```
#1. Load libraries
library(tidyverse)
library(glmnet)
library(ggplot2)
library(leaps)
library(corrr)
library(caret)
library(broom)
library(plotmo)
library(glinternet)

#2. Set working directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/6. Regression")
data <- read.csv("finaldata.csv")
View(data)

#3. Train test split

n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x.train <- train %>% select(-Y)
y.train <- train %>% select(Y)



x.val <- val %>% select(-Y)
y.val <- val %>% select(Y)


x.test <- test %>% select(-Y)
y.test <- test %>% select(Y)


#4.1 Train the model
mod0 <- lm(Y~.^2, data = train)
sum.mod0 <- summary(mod0)
sum.mod0
```

```
mod1 <- lm(Y~X1 +X2 +X4 +X12 +X13 +X14 +X15+ X16 + X17+ X18+ X1:X12 + X2:X4 +
    X2:X12 + X2:X13 + X2:X18 + X4:X12 + X12:X14 + X12:X15 + X12:X16, data =
    train)
sum.mod1 <- summary(mod1)
sum.mod0
sum.mod1

#4.2 Validate the model
val.mod1 <- predict(mod1, newdata = val)
val.mod1.MSE <- mean((val.mod1-val$Y)^2)
val.mod1.MAE <- mean(abs(val.mod1-val$Y))
val.mod1.MSE
val.mod1.MAE

#4.3 Test the model
test.mod1 <- predict(mod1, newdata = test)
test.mod1.MSE <- mean((test.mod1-test$Y)^2)
test.mod1.MAE <- mean(abs(test.mod1-test$Y))
test.mod1.MSE
test.mod1.MAE

#4.4 Evaluate R squared and adjusted R squared
#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

mod1.rsq <- R.squared(val.mod1, val$Y)
k <- dim(train)[2]-1
n <- dim(val)[1]
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
mod1.adjrsq



#4.5 Do the ANOVA
anovadata <- cbind(x.test,as.data.frame(test.mod1))
anovadata <- anovadata %>%
  rename(yhat = test.mod1)
View(anovadata)

mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18+X1:X12 + X2:X4 +
    X2:X12 + X2:X13 + X2:X18 + X4:X12 +
                X12:X14, data = anovadata)
summary(mod1.aov)
```

```
#4.6 Variable importance
mod1.varimp <- varImp(mod1)
mod1.varimp

#4.7 Parity plots
plot_dat <- as.data.frame(cbind(test$Y, test.mod1))
names(plot_dat) <- c("Y_actual", "Y_hat")
View(plot_dat)

parity <- ggplot(plot_dat, aes(x=Y_actual, y=Y_hat)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))

pdf(file = "parity_MLR_interaction.pdf")
parity
dev.off()

#4.8 Residual plot

res <- y.test - test.mod1
res_dat <- as.data.frame(cbind(res, as.data.frame(test.mod1)))
names(res_dat) <- c("res", "pred")
residual <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf(file = "residual_MLR_interaction.pdf")
residual
dev.off()

#4.9 QQ plot of residuals
res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "MLR_interaction_qq.pdf")
res_qq
dev.off()
```

C.3.5 Python Code for Interaction MLR with mass balance constraint

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import os
from statsmodels.formula.api import glm
import pandas as pd
from sklearn.model_selection import train_test_split

# 2. Set the working directory and import the files
abspath = os.path.abspath(r'C:\Users\vtspe\OneDrive - University of Cape
    Town\MSc Data Science\Data Science Research\STA5079Z - Dissertation\V2\Code
    and Data\6. Regression') ## String which contains absolute path to the
    script file
os.chdir(abspath) ## Setting up working directory
df = pd.read_csv("finaldata.csv")




# 3. Train test split
X_train, X_testval, y_train, y_testval =
    train_test_split(df[['X1','X2','X4','X12','X13','X14','X15','X16','X17','X18']],
    df[['Y']], test_size=0.4, random_state=11)
X_val, X_test, y_val, y_test = train_test_split(X_testval, y_testval,
    test_size=0.5, random_state=11)

# 4. Build the model

train = pd.concat([X_train, y_train], axis=1)
mod = glm("Y ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18+X1*X12+
    X2*X4+X2*X12+X2*X13+X2*X18+X4*X12+X12*X14+X12*X15+X12*X16", data=train)
res = mod.fit_constrained(["X12+X13+X14+X15+X16+X17+X18 = 1"])
res.summary()
mod_val = res.predict(X_val)
mod_val = pd.DataFrame(mod_val)

mod_val = mod_val.rename(columns={0: 'Y'})

val_MSE = ((mod_val-y_val)**2).mean()
val_MAE = (abs(mod_val-y_val)).mean()

mod_test = res.predict(X_test)
mod_test = pd.DataFrame(mod_test)

mod_test = mod_test.rename(columns={0: 'Y'})

test_MSE = ((mod_test-y_test)**2).mean()
test_MAE = (abs(mod_test-y_test)).mean()
```

```
res_params = pd.DataFrame(res.params)
res_params = res_params.rename(columns={0: 'Y'})


res_params = res_params.drop(labels='Intercept', axis=0)
res_varimp
    =res_params.loc[['X1','X2','X4','X12','X13','X14','X15','X16','X17','X18']]

varimp = abs(res_varimp['Y'])/max(abs(res_varimp['Y']))

val = pd.concat([X_val, y_val], axis = 1)
test = pd.concat([X_test, y_test], axis = 1)

train.to_csv('mod2_train_int.csv')
val.to_csv('mod2_val_int.csv')
test.to_csv('mod2_test_int.csv')

val_pred = pd.concat([X_val, mod_val], axis = 1)
test_pred = pd.concat([X_test, mod_test], axis = 1)

val_pred.to_csv('mod2_valpred_int.csv')
test_pred.to_csv('mod2_testpred_int.csv')
```

C.3.6 R Code for Interaction MLR with mass balance constraint ANOVA

```
#1. Load libraries
library(tidyverse)
library(glmnet)
library(ggplot2)
library(leaps)
library(corrr)
library(caret)
library(broom)
library(plotmo)


#2. Set working directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/6. Regression")
train <- read.csv("mod2_train.csv")
val <- read.csv("mod2_val_int.csv")
test <- read.csv("mod2_test_int.csv")
valpred <- read.csv("mod2_valpred_int.csv")
testpred <- read.csv("mod2_testpred_int.csv")


#3. Train test split

x.train <- train %>% select(-Y,-X)
y.train <- train %>% select(Y)

x.val <- val %>% select(-Y,-X)
y.val <- val %>% select(Y)

x.test <- test %>% select(-Y,-X)
y.test <- test %>% select(Y)


#4.1 Train the model


#4.2 Validate the model
val.mod1 <- valpred$Y
val.mod1.MSE <- mean((val.mod1-val$Y)^2)
val.mod1.MAE <- mean(abs(val.mod1-val$Y))
val.mod1.MSE
val.mod1.MAE

#4.3 Test the model
test.mod1 <- testpred$Y
test.mod1.MSE <- mean((test.mod1-test$Y)^2)
test.mod1.MAE <- mean(abs(test.mod1-test$Y))
test.mod1.MSE
test.mod1.MAE

#4.4 Evaluate R squared and adjusted R squared
#R-Squared
```

```
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

mod1.rsq <- R.squared(val.mod1, val$Y)
k <- dim(train)[2]-1
n <- dim(val)[1]
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
mod1.adjrsq


#4.5 Do the ANOVA
anovadata <- testpred
anovadata <- anovadata %>%
  rename(yhat = Y)
View(anovadata)

mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18
    +X1:X12+X2:X4+X2:X12+X2:X13+X2:X18+X4:X12+X12:X14+ X12:X15 + X12:X16, data =
    anovadata)
summary(mod1.aov)


#4.7 Parity plots
plot_dat <- as.data.frame(cbind(test$Y, testpred$Y))
names(plot_dat) <- c("Y_actual", "Y_hat")
View(plot_dat)

parity <- ggplot(plot_dat, aes(x=Y_actual, y=Y_hat)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))

pdf(file = "parity_MLR_Constraints_int.pdf")
parity
dev.off()

#4.8 Residual plot

res <- y.test - testpred$Y
res_dat <- as.data.frame(cbind(res, testpred$Y))
names(res_dat) <- c("res", "pred")
residual <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
```

```
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf(file = "residual_MLR_constraints_int.pdf")
residual
dev.off()

#4.9 QQ plot of residuals
res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "MLR_constraints_Residual_int_qq.pdf")
res_qq
dev.off()
```

C.4 R Code for Bayesian Additive Regression Trees (BART)

```
# Load Libraries
library(dplyr)
library(tidyverse)
library(ggplot2)
library(bartMachine)
library(rJava)
# Working Directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/7. BART")
data <- read.csv("finaldata.csv")
View(data)

# Train test split
n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x_train <- train %>% select(-Y)
y_train <- train %>% select(Y)

x_val <- val %>% select(-Y)
y_val <- val %>% select(Y)

x_test <- test %>% select(-Y)
y_test <- test %>% select(Y)

# Model Building
## Train the model
mod1 <- bartMachine(X = x_train,y = as.numeric(as.matrix(y_train)))

## Validate by using a grid search on a seperate validation set for 15 and 50
    trees
mod1_cv <- bartMachineCV(x_train, as.numeric(as.matrix(y_train)), k_folds = 10)

##"#winning model has: k = 5, nu = 3, q = 0.99, num_trees = 200

### Evaluate the extent of the number of trees
pdf("BART_CV.pdf")
rmse_by_num_trees(mod1_cv, num_replicates = 20)
dev.off()
```

```
## Produce diagnostics of the winning ensemble
val.mod1 <- bart_predict_for_test_data(mod1_cv, Xtest = x_val, ytest =
    as.numeric(as.matrix(y_val)))
val.mod1.MSE <- (val.mod1$rmse)^2
val.mod1.MSE

test.mod1 <- bart_predict_for_test_data(mod1_cv, Xtest = x_test, ytest =
    as.numeric(as.matrix(y_test)))
test.mod1.MSE <- (test.mod1$rmse)^2
test.mod1.MSE

# Evaluate R squared and adjusted R squared
#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

mod1.rsq <- R.squared(as.numeric(val.mod1$y_hat), as.numeric(y_val$Y))
k <- dim(train)[2]-1
n <- dim(val)[1]
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
mod1.adjrsq

#Do the ANOVA
anovadata <- test
anovadata <- anovadata %>%
  rename(yhat = Y)
View(anovadata)

mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)

#4.7 Parity plots
plot_dat <- as.data.frame(cbind(test$Y, test.mod1$y_hat))
names(plot_dat) <- c("Y_actual", "Y_hat")
View(plot_dat)

parity <- ggplot(plot_dat, aes(x=Y_actual, y=Y_hat)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))

pdf(file = "parity_BART.pdf")
parity
```

```
dev.off()

#4.8 Residual plot

res <- y_test - test.mod1$y_hat
res_dat <- as.data.frame(cbind(res, test.mod1$y_hat))
names(res_dat) <- c("res", "pred")
residual <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf(file = "residual_BART.pdf")
residual
dev.off()

#4.9 QQ plot of residuals
res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "res_qq_BART.pdf")
res_qq
dev.off()

#5.0 Variable importance
pdf("BART_varimp.pdf")
investigate_var_importance(mod1_cv, num_replicates_for_avg = 20)
dev.off()
```

C.5 R Code for Gradient Boosting Machines (GBM)

```
# Load libraries
library(gbm)
library(dplyr)
library(tidyverse)
library(ggplot2)

# Working Directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/8. GBM")
data <- read.csv("finaldata.csv")

View(data)

# Train test split
n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x_train <- train %>% select(-Y)
y_train <- train %>% select(Y)

x_val <- val %>% select(-Y)
y_val <- val %>% select(Y)

x_test <- test %>% select(-Y)
y_test <- test %>% select(Y)



# Model Building
## Train the model
mod1 <- gbm(formula = Y ~.,
            distribution = "gaussian",
            data <- train,
            n.trees = 1000)

mod1
summary(mod1)

## Tune the GBM
```

```
mod1_cv <- gbm(formula = Y~.,
               distribution = "gaussian",
               data = val,
               n.trees = 2000,
               interaction.depth = 4,
               shrinkage = 0.01,
               cv.folds = 10)

## Show tuning results
mod1_opt_cv <- gbm.perf(mod1_cv, method = "cv")
mod1_opt_oob <- gbm.perf(mod1_cv, method = "OOB")
print(mod1_opt_cv)
print(mod1_opt_oob)

# Model Results and Diagnostics
## CV and OOB Plot of iterations

pdf("gbm_cv.pdf")
gbm.perf(mod1_cv, method = "cv")
dev.off()

pdf("gbm_oob.pdf")
gbm.perf(mod1_cv, method = "OOB")
dev.off()

## Variable Importance fixing

mod1_varimp <- as.data.frame(summary(mod1_cv))
colnames(mod1_varimp) <- c("Variable", "Rel_inf")
rownames(mod1_varimp) <- NULL

mod1_varimp_plot <- ggplot(mod1_varimp, aes(x=reorder(Variable, Rel_inf), y =
    Rel_inf)) +
  geom_bar(stat="identity") +
  coord_flip() +
  ylab("Relative Influence") +
  xlab("Variable")

pdf("gbm_varimplot.pdf")
mod1_varimp_plot
dev.off()




## Make Predictions

y_hat_val <- predict.gbm(object = mod1_cv,
                         newdata = val,
                         n.trees = mod1_opt_cv,
                         type = 'response')

y_hat_test <- predict.gbm(object = mod1_cv,
                          newdata = test,
                          n.trees = mod1_opt_cv,
```

```
                        type = 'response')
## Summary Diagnostics

mod1_val_MSE <- mean((y_hat_val-y_val$Y)^2)
mod1_test_MSE <- mean((y_hat_test-y_test$Y)^2)

#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherrry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}
mod1.rsq <- R.squared(y_hat_val, val$Y)
k <- dim(train)[2]-1
n <- dim(val)[1]
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
mod1.adjrsq

# Anova

anovadata <- as.data.frame(cbind(x_test, y_hat_test))
anovadata <- anovadata %>%
  rename(yhat = y_hat_test)
View(anovadata)
mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)

# Parity
plot_dat <- as.data.frame(cbind(y_test, y_hat_test))
names(plot_dat) <- c("Y_actual", "Y_GBM")

parity_gbm <- ggplot(plot_dat, aes(x=Y_actual, y=Y_GBM)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))

pdf(file = "parity_gbm.pdf")
parity_gbm
dev.off()

# Residual
res_gbm <- y_test - y_hat_test
res_dat_gbm <- as.data.frame(cbind(res_gbm, as.data.frame(y_hat_test)))
names(res_dat_gbm) <- c("res", "pred")
residual_gbm <- ggplot(res_dat_gbm, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
```

```
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf(file = "residual_gbm.pdf")
residual_gbm
dev.off()

#Residual QQ plot

res_qq <- ggplot(res_dat_gbm, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "res_qq_GBM.pdf")
res_qq
dev.off()
```

C.6 Code for Single Layer Feed-forward Neural Network (SLFN)

C.6.1 R Code for 3 Node SLFN

```
# Load Libraries

library(h2o)
library(dplyr)
library(tidyverse)

# Initialize h2o
localH2O <- h2o.init()

# Working Directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/9. ANN")

data <- read.csv("finaldata.csv")
View(data)

# Train test split
n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x_train <- train %>% select(-Y)
y_train <- train %>% select(Y)

x_val <- val %>% select(-Y)
y_val <- val %>% select(Y)

x_test <- test %>% select(-Y)
y_test <- test %>% select(Y)

# Convert data frames to H2O objects

train.h2o <- as.h2o(train)
val.h2o <- as.h2o(val)
test.h2o <- as.h2o(test)

x_train.h2o <- as.h2o(x_train)
y_train.h2o <- as.h2o(y_train)
```

```
x_val.h2o <- as.h2o(x_val)
y_val.h2o <- as.h2o(y_val)

x_test.h2o <- as.h2o(x_test)
y_test.h2o <- as.h2o(y_test)

# Train the Single Layer ANN
## Initial model

mod1 <- h2o.deeplearning(x = 1:10,
                         y = 11,
                         training_frame = train.h2o,
                         activation = "Tanh",
                         #validation_frame = val.h2o,
                         hidden = c(3),
                         seed = 1,
                         reproducible = TRUE,
                         nfolds = 10,
                         epochs = 1000,
                         variable_importances = TRUE,
                         export_weights_and_biases = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         #keep_cross_validation_fold_assignment = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         score_each_iteration = TRUE
                         #hidden_dropout_ratios = 0.1,
                         #l2 = 1e-4
)

## Check the model summary
mod1

## Plot model performance

plot(mod1)

## Retrieve the variable importance for model 1

h2o.varimp(mod1)
h2o.varimp(mod1)
pdf("03_node_var_imp.pdf")
h2o.varimp_plot(mod1)
dev.off()


## Retrieve the weights and biases

h2o.weights(mod1, matrix_id = 1)
h2o.biases(mod1, vector_id = 1)

# Obtain predictions for model 1
## Check the model parameters

mod1@parameters
```

```
## Evaluate model performance

h2o.performance(mod1, train = TRUE)


## Make predictions on model 1

mod1.val <- h2o.predict(mod1, val.h2o)
mod1.test <- h2o.predict(mod1, test.h2o)
mod1.MSE.val.h2o <- mean((mod1.val- y_val.h2o)^2)
mod1.MSE.test.h2o <- mean((mod1.test - y_test.h2o)^2)
mod1.MSE.val <- as.data.frame(mod1.MSE.val.h2o)
mod1.MSE.test <- as.data.frame(mod1.MSE.test.h2o)


## Set up function for R squared

#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherrry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

k <- dim(x_val)[2] ## number of variables in validation set
n <- dim(x_val)[1] ## number of observations in validation set ; Use the
    validation set to determine the r squared value


## Evaluate adjusted Rsq for model 1

mod1.rsq <- R.squared(as.data.frame(mod1.val), y_val)
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
print(" The Adjusted R^2 for model 1 is ")
mod1.adjrsq


# Train the model with the validation set

mod1cv <- h2o.deeplearning(x = 1:10,
                           y = 11,
                           training_frame = train.h2o,
                           activation = "Tanh",
                           validation_frame = val.h2o,
                           hidden = c(3),
                           seed = 1,
                           reproducible = TRUE,
```

```
                          nfolds = 10,
                          epochs = 1000,
                          variable_importances = TRUE,
                          export_weights_and_biases = TRUE,
                          #keep_cross_validation_predictions = TRUE,
                          #keep_cross_validation_fold_assignment = TRUE,
                          #keep_cross_validation_predictions = TRUE,
                          score_each_iteration = TRUE
                          #hidden_dropout_ratios = 0.1,
                          #l2 = 1e-4
)

mod1cv

pdf("03_score.pdf")
plot(mod1cv,
     title = "Scoring history for 3 node ANN")
dev.off()

plot_dat <- cbind(as.data.frame(y_test), as.data.frame(mod1.test))
names(plot_dat) <- c("Y_actual", "Y_03")
Parity_03 <- ggplot(plot_dat, aes(x=Y_actual, y=Y_03)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))
pdf("Parity_03.pdf")
Parity_03
dev.off()

res <- y_test - as.data.frame(mod1.test)
res_dat <- as.data.frame(cbind(res, as.data.frame(mod1.test)))
names(res_dat) <- c("res", "pred")
residual_03 <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf("residual_03.pdf")
residual_03
dev.off()

#Residual QQ plot

res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "res_qq_3_nodes.pdf")
res_qq
dev.off()
```

```r
# Anova

anovadata <- cbind(as.data.frame(x_test.h2o), as.data.frame(mod1.test$predict))
anovadata <- anovadata %>%
  rename(yhat = predict)
View(anovadata)
mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)
```

C.6.2 R Code for 5 Node SLFN

```
# Load Libraries

library(h2o)
library(dplyr)
library(tidyverse)

# Initialize h2o
localH2O <- h2o.init()

# Working Directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/9. ANN")

data <- read.csv("finaldata.csv")
View(data)

# Train test split
n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x_train <- train %>% select(-Y)
y_train <- train %>% select(Y)

x_val <- val %>% select(-Y)
y_val <- val %>% select(Y)

x_test <- test %>% select(-Y)
y_test <- test %>% select(Y)

# Convert data frames to H2O objects

train.h2o <- as.h2o(train)
val.h2o <- as.h2o(val)
test.h2o <- as.h2o(test)

x_train.h2o <- as.h2o(x_train)
y_train.h2o <- as.h2o(y_train)

x_val.h2o <- as.h2o(x_val)
y_val.h2o <- as.h2o(y_val)
```

```
x_test.h2o <- as.h2o(x_test)
y_test.h2o <- as.h2o(y_test)

# Train the Single Layer ANN
## Initial model

mod1 <- h2o.deeplearning(x = 1:10,
                         y = 11,
                         training_frame = train.h2o,
                         activation = "Tanh",
                         #validation_frame = val.h2o,
                         hidden = c(5),
                         seed = 1,
                         reproducible = TRUE,
                         nfolds = 10,
                         epochs = 1000,
                         variable_importances = TRUE,
                         export_weights_and_biases = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         #keep_cross_validation_fold_assignment = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         score_each_iteration = TRUE
                         #hidden_dropout_ratios = 0.1,
                         #l2 = 1e-4
)

## Check the model summary
mod1

## Plot model performance

plot(mod1)

## Retrieve the variable importance for model 1

h2o.varimp(mod1)
h2o.varimp(mod1)
pdf("05_node_var_imp.pdf")
h2o.varimp_plot(mod1)
dev.off()


## Retrieve the weights and biases

h2o.weights(mod1, matrix_id = 1)
h2o.biases(mod1, vector_id = 1)

# Obtain predictions for model 1
## Check the model parameters

mod1@parameters


## Evaluate model performance
```

```
h2o.performance(mod1, train = TRUE)


## Make predictions on model 1

mod1.val <- h2o.predict(mod1, val.h2o)
mod1.test <- h2o.predict(mod1, test.h2o)
mod1.MSE.val.h2o <- mean((mod1.val- y_val.h2o)^2)
mod1.MSE.test.h2o <- mean((mod1.test - y_test.h2o)^2)
mod1.MSE.val <- as.data.frame(mod1.MSE.val.h2o)
mod1.MSE.test <- as.data.frame(mod1.MSE.test.h2o)


## Set up function for R squared

#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherrry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

k <- dim(x_val)[2] ## number of variables in validation set
n <- dim(x_val)[1] ## number of observations in validation set ; Use the
    validation set to determine the r squared value


## Evaluate adjusted Rsq for model 1

mod1.rsq <- R.squared(as.data.frame(mod1.val), y_val)
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
print(" The Adjusted R^2 for model 1 is ")
mod1.adjrsq


# Train the model with the validation set

mod1cv <- h2o.deeplearning(x = 1:10,
                           y = 11,
                           training_frame = train.h2o,
                           activation = "Tanh",
                           validation_frame = val.h2o,
                           hidden = c(5),
                           seed = 1,
                           reproducible = TRUE,
                           nfolds = 10,
                           epochs = 1000,
```

```
                        variable_importances = TRUE,
                        export_weights_and_biases = TRUE,
                        #keep_cross_validation_predictions = TRUE,
                        #keep_cross_validation_fold_assignment = TRUE,
                        #keep_cross_validation_predictions = TRUE,
                        score_each_iteration = TRUE
                        #hidden_dropout_ratios = 0.1,
                        #l2 = 1e-4
)

mod1cv

pdf("05_score.pdf")
plot(mod1cv,
     title = "Scoring history for 5 node ANN")
dev.off()

plot_dat <- cbind(as.data.frame(y_test), as.data.frame(mod1.test))
names(plot_dat) <- c("Y_actual", "Y_05")
Parity_05 <- ggplot(plot_dat, aes(x=Y_actual, y=Y_05)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))
pdf("Parity_05.pdf")
Parity_05
dev.off()

res <- y_test - as.data.frame(mod1.test)
res_dat <- as.data.frame(cbind(res, as.data.frame(mod1.test)))
names(res_dat) <- c("res", "pred")
residual_05 <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf("residual_05.pdf")
residual_05
dev.off()

#Residual QQ plot

res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "res_qq_5_nodes.pdf")
res_qq
dev.off()

# Anova
```

```
anovadata <- cbind(as.data.frame(x_test.h2o), as.data.frame(mod1.test$predict))
anovadata <- anovadata %>%
  rename(yhat = predict)
View(anovadata)
mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)
```

C.6.3 R Code for 10 Node SLFN

```
# Load Libraries

library(h2o)
library(dplyr)
library(tidyverse)

# Initialize h2o
localH2O <- h2o.init()

# Working Directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/9. ANN")

data <- read.csv("finaldata.csv")
View(data)

# Train test split
n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x_train <- train %>% select(-Y)
y_train <- train %>% select(Y)

x_val <- val %>% select(-Y)
y_val <- val %>% select(Y)

x_test <- test %>% select(-Y)
y_test <- test %>% select(Y)

# Convert data frames to H2O objects

train.h2o <- as.h2o(train)
val.h2o <- as.h2o(val)
test.h2o <- as.h2o(test)

x_train.h2o <- as.h2o(x_train)
y_train.h2o <- as.h2o(y_train)

x_val.h2o <- as.h2o(x_val)
y_val.h2o <- as.h2o(y_val)
```

```
x_test.h2o <- as.h2o(x_test)
y_test.h2o <- as.h2o(y_test)

# Train the Single Layer ANN
## Initial model

mod1 <- h2o.deeplearning(x = 1:10,
                         y = 11,
                         training_frame = train.h2o,
                         activation = "Tanh",
                         #validation_frame = val.h2o,
                         hidden = c(10),
                         seed = 1,
                         reproducible = TRUE,
                         nfolds = 10,
                         epochs = 1000,
                         variable_importances = TRUE,
                         export_weights_and_biases = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         #keep_cross_validation_fold_assignment = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         score_each_iteration = TRUE
                         #hidden_dropout_ratios = 0.1,
                         #l2 = 1e-4
)

## Check the model summary
mod1

## Plot model performance

plot(mod1)

## Retrieve the variable importance for model 1

h2o.varimp(mod1)
h2o.varimp(mod1)
pdf("10_node_var_imp.pdf")
h2o.varimp_plot(mod1)
dev.off()


## Retrieve the weights and biases

h2o.weights(mod1, matrix_id = 1)
h2o.biases(mod1, vector_id = 1)

# Obtain predictions for model 1
## Check the model parameters

mod1@parameters


## Evaluate model performance
```

```
h2o.performance(mod1, train = TRUE)


## Make predictions on model 1

mod1.val <- h2o.predict(mod1, val.h2o)
mod1.test <- h2o.predict(mod1, test.h2o)
mod1.MSE.val.h2o <- mean((mod1.val- y_val.h2o)^2)
mod1.MSE.test.h2o <- mean((mod1.test - y_test.h2o)^2)
mod1.MSE.val <- as.data.frame(mod1.MSE.val.h2o)
mod1.MSE.test <- as.data.frame(mod1.MSE.test.h2o)


## Set up function for R squared

#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherrry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

k <- dim(x_val)[2] ## number of variables in validation set
n <- dim(x_val)[1] ## number of observations in validation set ; Use the
    validation set to determine the r squared value


## Evaluate adjusted Rsq for model 1

mod1.rsq <- R.squared(as.data.frame(mod1.val), y_val)
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
print(" The Adjusted R^2 for model 1 is ")
mod1.adjrsq


# Train the model with the validation set

mod1cv <- h2o.deeplearning(x = 1:10,
                           y = 11,
                           training_frame = train.h2o,
                           activation = "Tanh",
                           validation_frame = val.h2o,
                           hidden = c(10),
                           seed = 1,
                           reproducible = TRUE,
                           nfolds = 10,
                           epochs = 1000,
```

```
                          variable_importances = TRUE,
                          export_weights_and_biases = TRUE,
                          #keep_cross_validation_predictions = TRUE,
                          #keep_cross_validation_fold_assignment = TRUE,
                          #keep_cross_validation_predictions = TRUE,
                          score_each_iteration = TRUE
                          #hidden_dropout_ratios = 0.1,
                          #l2 = 1e-4
)

mod1cv

pdf("10_score.pdf")
plot(mod1cv,
     title = "Scoring history for 10 node ANN")
dev.off()

plot_dat <- cbind(as.data.frame(y_test), as.data.frame(mod1.test))
names(plot_dat) <- c("Y_actual", "Y_10")
Parity_10 <- ggplot(plot_dat, aes(x=Y_actual, y=Y_10)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))
pdf("Parity_10.pdf")
Parity_10
dev.off()

res <- y_test - as.data.frame(mod1.test)
res_dat <- as.data.frame(cbind(res, as.data.frame(mod1.test)))
names(res_dat) <- c("res", "pred")
residual_10 <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf("residual_10.pdf")
residual_10
dev.off()

#Residual QQ plot

res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "res_qq_10_nodes.pdf")
res_qq
dev.off()

# Anova
```

```
anovadata <- cbind(as.data.frame(x_test.h2o), as.data.frame(mod1.test$predict))
anovadata <- anovadata %>%
  rename(yhat = predict)
View(anovadata)
mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)
```

C.6.4 R Code for 15 Node SLFN

```
# Load Libraries

library(h2o)
library(dplyr)
library(tidyverse)

# Initialize h2o
localH2O <- h2o.init()

# Working Directory and load data
setwd("C:/Users/vtspe/OneDrive - University of Cape Town/MSc Data Science/Data
    Science Research/STA5079Z - Dissertation/V2/Code and Data/9. ANN")

data <- read.csv("finaldata.csv")
View(data)

# Train test split
n_data <- nrow(data)
n_train <- 0.6*n_data
n_val <- 0.2*n_data
n_test <- 0.2*n_data
set.seed(1)
train_index <- sample(1:n_data, n_train)
train <- data[train_index,]

data_2 <- data[-train_index,]
set.seed(2)
val_index <- sample(1:(n_val + n_test), n_val)
val <- data_2[val_index,]

test <- data_2[-val_index,]

x_train <- train %>% select(-Y)
y_train <- train %>% select(Y)

x_val <- val %>% select(-Y)
y_val <- val %>% select(Y)

x_test <- test %>% select(-Y)
y_test <- test %>% select(Y)

# Convert data frames to H2O objects

train.h2o <- as.h2o(train)
val.h2o <- as.h2o(val)
test.h2o <- as.h2o(test)

x_train.h2o <- as.h2o(x_train)
y_train.h2o <- as.h2o(y_train)

x_val.h2o <- as.h2o(x_val)
y_val.h2o <- as.h2o(y_val)
```

```
x_test.h2o <- as.h2o(x_test)
y_test.h2o <- as.h2o(y_test)

# Train the Single Layer ANN
## Initial model

mod1 <- h2o.deeplearning(x = 1:10,
                         y = 11,
                         training_frame = train.h2o,
                         activation = "Tanh",
                         #validation_frame = val.h2o,
                         hidden = c(15),
                         seed = 1,
                         reproducible = TRUE,
                         nfolds = 10,
                         epochs = 1000,
                         variable_importances = TRUE,
                         export_weights_and_biases = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         #keep_cross_validation_fold_assignment = TRUE,
                         #keep_cross_validation_predictions = TRUE,
                         score_each_iteration = TRUE
                         #hidden_dropout_ratios = 0.1,
                         #l2 = 1e-4
)

## Check the model summary
mod1

## Plot model performance

plot(mod1)

## Retrieve the variable importance for model 1

h2o.varimp(mod1)
h2o.varimp(mod1)
pdf("15_node_var_imp.pdf")
h2o.varimp_plot(mod1)
dev.off()


## Retrieve the weights and biases

h2o.weights(mod1, matrix_id = 1)
h2o.biases(mod1, vector_id = 1)

# Obtain predictions for model 1
## Check the model parameters

mod1@parameters


## Evaluate model performance
```

```
h2o.performance(mod1, train = TRUE)


## Make predictions on model 1

mod1.val <- h2o.predict(mod1, val.h2o)
mod1.test <- h2o.predict(mod1, test.h2o)
mod1.MSE.val.h2o <- mean((mod1.val- y_val.h2o)^2)
mod1.MSE.test.h2o <- mean((mod1.test - y_test.h2o)^2)
mod1.MSE.val <- as.data.frame(mod1.MSE.val.h2o)
mod1.MSE.test <- as.data.frame(mod1.MSE.test.h2o)


## Set up function for R squared

#R-Squared
R.squared <- function(x,y){
  R2 <- cor(x,y)^2
  return(R2)
}

#Adjusted R Squared using Wherrry's equation
adj.R.squared <- function(R,n,k){
  numerator <- (1-R^2)*(n-1)
  denominator <- n-k
  adj.R2 <- 1-(numerator/denominator)
  return(adj.R2)
}

k <- dim(x_val)[2] ## number of variables in validation set
n <- dim(x_val)[1] ## number of observations in validation set ; Use the
    validation set to determine the r squared value


## Evaluate adjusted Rsq for model 1

mod1.rsq <- R.squared(as.data.frame(mod1.val), y_val)
mod1.adjrsq <- adj.R.squared(mod1.rsq, n, k)
print(" The Adjusted R^2 for model 1 is ")
mod1.adjrsq


# Train the model with the validation set

mod1cv <- h2o.deeplearning(x = 1:10,
                           y = 11,
                           training_frame = train.h2o,
                           activation = "Tanh",
                           validation_frame = val.h2o,
                           hidden = c(15),
                           seed = 1,
                           reproducible = TRUE,
                           nfolds = 10,
                           epochs = 1000,
```

```
                        variable_importances = TRUE,
                        export_weights_and_biases = TRUE,
                        #keep_cross_validation_predictions = TRUE,
                        #keep_cross_validation_fold_assignment = TRUE,
                        #keep_cross_validation_predictions = TRUE,
                        score_each_iteration = TRUE
                        #hidden_dropout_ratios = 0.1,
                        #l2 = 1e-4
)

mod1cv

pdf("15_score.pdf")
plot(mod1cv,
     title = "Scoring history for 15 node ANN")
dev.off()

plot_dat <- cbind(as.data.frame(y_test), as.data.frame(mod1.test))
names(plot_dat) <- c("Y_actual", "Y_15")
Parity_15 <- ggplot(plot_dat, aes(x=Y_actual, y=Y_15)) +
  geom_smooth(method = 'lm') +
  geom_point() +
  xlab("RON (Experimental)") +
  ylab("RON (Predicted)") +
  theme(text = element_text(size=11))
pdf("Parity_15.pdf")
Parity_15
dev.off()

res <- y_test - as.data.frame(mod1.test)
res_dat <- as.data.frame(cbind(res, as.data.frame(mod1.test)))
names(res_dat) <- c("res", "pred")
residual_15 <- ggplot(res_dat, aes(x=pred, y=res)) +
  geom_smooth(method=stats::lm) +
  geom_point() +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme(text = element_text(size=11))
pdf("residual_15.pdf")
residual_15
dev.off()

#Residual QQ plot

res_qq <- ggplot(res_dat, aes(sample = res)) +
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical Quantiles") +
  ylab("Quantiles of the Residuals") +
  theme(text = element_text(size=11))
pdf(file = "res_qq_15_nodes.pdf")
res_qq
dev.off()

# Anova
```

```
anovadata <- cbind(as.data.frame(x_test.h2o), as.data.frame(mod1.test$predict))
anovadata <- anovadata %>%
  rename(yhat = predict)
View(anovadata)
mod1.aov <- aov(yhat ~ X1+X2+X4+X12+X13+X14+X15+X16+X17+X18, data = anovadata)
summary(mod1.aov)
```