

MACHINE LEARNING PREDICTIVE MODEL DEVELOPMENT
FOR VACUUM DISTILLATION UNIT IN
LUBRICANT BASE OIL PRODUCTION

PRAKASH SARAVANAN

CHEMICAL ENGINEERING
UNIVERSITI TEKNOLOGI PETRONAS

JANUARY 2021

**Machine Learning Predictive Model Development for Vacuum Distillation Unit
in Lubricant Base Oil Production**

by

Prakash Saravanan

24394

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Chemical Engineering)

JANUARY 2021

Universiti Teknologi PETRONAS,
32610, Bandar Seri Iskandar,
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

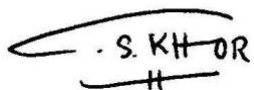
**Machine Learning Predictive Model Development for Vacuum Distillation Unit
in Lubricant Base Oil Production**

by

Prakash Saravanan
24394

Dissertation submitted to the
Chemical Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(CHEMICAL ENGINEERING)

Approved by,

A handwritten signature in black ink, consisting of a stylized loop followed by the text ". S. KH OR" and a horizontal line underneath.

Ir Dr Khor Cheng Seong

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR, PERAK

January 2021

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



PRAKASH SARAVANAN

ABSTRACT

Predictive modelling through machine learning is utilized to accurately determine the predictive variables for a given target variable in the operation of a Vacuum Distillation Unit in a lubricant base oil plant. This is done by training the model using 70% of the past plant data from 2018 to 2020 and enables the model to capture the correlations between the target variables and the predictor variables. The algorithms, such as XGBoost, Random Forest and Decision Tree, have different approaches for training the model and their correlations but XGBoost has emerged as the best of the rest. In this report, two source codes were developed, one using XGBoost algorithm and the other using Random Forest algorithm. The breakdown of the source code development is explained in depth and some of the functions and packages used were also highlighted. Upon generating the model, the remaining 30% of the plant data is used as the test dataset to analyse the performance of the models. Based on the results obtained, it is identified that 19 out of the 21 models developed showed good fit with the data whereas the two variables, namely V2SS Density and V2SS Sulphur have fluctuations in its data, thus making it difficult to determine the outliers and selectively remove them. Comparison with the Random Forest models further enforces that the target variables except V2SS Density and V2SS Sulphur show excellent fit with the data. When the predictions of XGBoost and Random Forest are compared against each other, it can be observed that Kinematic Viscosity@100°C, Nitrogen and Aromatic content of all three side streams shows agreeable result. Further work could be done to fine tune certain parameters in the XGBoost to improve its performance comparable to that of Random Forest since XGBoost has the best computing performance. One such way is to use exhaustive tuning to vet through all through possible options which would take extremely long if Random Forest algorithm were to be used. Obtaining recent plant data could also help in analysing the performance and reliability of the developed models.

ACKNOWLEDGEMENTS

Throughout my journey in completing this dissertation, I am grateful to have received support and assistance from lecturers, friends and family. First and foremost, I would like to extend my utmost gratitude to Ir Dr Khor Cheng Seong, my project supervisor who has provided me with valuable resources and information for me to execute this project successfully. His constant supervision on my progress and his keen interest on my weekly findings has kept me motivated to continue pushing myself further and contribute the best I could for this project.

I would also like to extend my gratitude to Dr Mohd Hilmi bin Noh and Dr Mohd Dzul Hakim bin Wirzal for their support in arranging the relevant seminars to aid us in the project and their coordination and guidance throughout this project. Without their assistance, it would have been a difficult journey achieving what I had achieved now with the tight deadlines.

Next, I would like to extend my deepest gratitude towards the data science community for providing the resources and the fundamental knowledge regarding machine learning and data science available for public use. Their contributions have helped beginners like me to understand machine learning and data science with ease and allow us to challenge ourselves further knowing that the community could support in times of need. Finally, I would like to convey my appreciation to my family and friends for their moral support and motivation throughout executing this project.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL	ii
CERTIFICATION OF ORIGINALITY	iii
ABSTRACT	iv
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF APPENDICES	x
CHAPTER 1: INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	3
1.3 Objectives	3
1.4 Scope of Study	4
CHAPTER 2: LITERATURE REVIEW	5
2.1 Decision Tree	5
2.2 Random Forest	6
2.3 Extreme Gradient Boost (XGBoost)	7
CHAPTER 3: METHODOLOGY/PROJECT WORK	10
3.1 Softwares and Tools Used	10
3.2 Packages and Key Functions	10
3.3 Source Code Overview	15
CHAPTER 4 RESULTS AND DISCUSSION	17
4.1 XGBoost Model Results	17
4.1.1 Predictive Model Fit	17
4.1.2 Importance Plots	19
4.1.3 Predicted vs Actual Plots	23
4.2 Random Forest Model Results	26

4.2.1	Predictive Model Fit	26
4.2.2	Predicted vs Actual Plots	27
4.3	Comparison between XGBoost and Random Forest	30
CHAPTER 5	CONCLUSION AND RECOMMENDATIONS	34
5.1	Conclusion	34
5.2	Recommendations	35
REFERENCES		36
APPENDICES		38

LIST OF FIGURES

Figure 1.1	Overview of the processes within MRCSB.	2
Figure 1.2	Overview of the Vacuum Distillation Unit (Unit 18).	2
Figure 3.1	Snapshot of function masking when adding packages.	12
Figure 3.2	Flowchart of the Source Code.	16
Figure 4.1	Importance plots of V1SS using XGBoost.	19
Figure 4.2	Graph of Predicted vs Actual of V1SS Yield using XGBoost.	23
Figure 4.3	Graphs of Predicted vs Actual using XGBoost.	24
Figure 4.4	Graph of Predicted vs Actual of V1SS Yield using Random Forest.	27
Figure 4.5	Graphs of Predicted vs Actual using Random Forest.	28
Figure 4.6	Chart of R^2 for XGBoost and Random Forest.	30
Figure 4.7	Graphs of Random Forest vs XGBoost.	31

LIST OF TABLES

Table 2.1	Modelling Results for Predicting Marine Debris (Wu, 2018).	8
Table 2.2	Attributes of XGBoost Algorithm.	9
Table 3.1	List of Softwares.	10
Table 3.2	List of Packages and Functions.	11
Table 3.3	Hyperparameter description.	13
Table 3.4	Learning Task Parameters – “objective”.	14
Table 3.5	Learning Task Parameters – “eval_metric”.	14
Table 4.1	Coefficient of Determination, Main Feature and Gain for Respective Target Variables using XGBoost.	18
Table 4.2	Coefficient of Determination for Respective Target Variables using Random Forest.	26

LIST OF APPENDICES

Appendix A	List of Abbreviations	38
Appendix B	Input Variables	42
Appendix C	Output Variables	43
Appendix D	Gantt Chart (FYP I)	44
Appendix E	Gantt Chart (FYP II)	45
Appendix F	Source Code (XGBoost)	46
Appendix G	Source Code – Random Forest	50

CHAPTER 1

INTRODUCTION

1.1 Background

This project entitled “Machine Learning Predictive Model Development for Vacuum Distillation Unit in Lubricant Base Oil Production” is a collaborative project between Malaysian Refining Company Sdn. Bhd. (MRCSB) and Centre for Process Systems Engineering (CPSE), Universiti Teknologi PETRONAS.

Figure 1.1 provides a brief overview of the processes within MRCSB. A Crude Distillation Unit (CDU) fractionates the crude into fuel gases, Liquefied Petroleum Gas (LPG), naphtha, kerosene and diesel. The crude residue is then sent to an intermediary tank. Referring to Figure 1.2, the Low Sulphur Waxy Residue (LSWR) from the CDU makes up the hot feed and the LSWR from the intermediate tank represents the cold feed. Throughout the operation of the shown units, two modes are run, namely MG3 and Non-MG3 where the first is a combination of both the hot and cold feed and the latter constitutes of only cold feed from the intermediate tank. The feed then enters into a Vacuum Distillation Unit (VDU) (Unit 18) where it is further separated into light ends, light vapor gas oil (LVGO), medium vapor gas oil (MVGO), vapor residue (VR) and three side draws (V1SS, V2SS and V3SS). These three side draws are then stored in an intermediate tank before being blended and further processed in Unit 19 to form Group III base oil.

MRCSB CDU1 and MG3 Overview

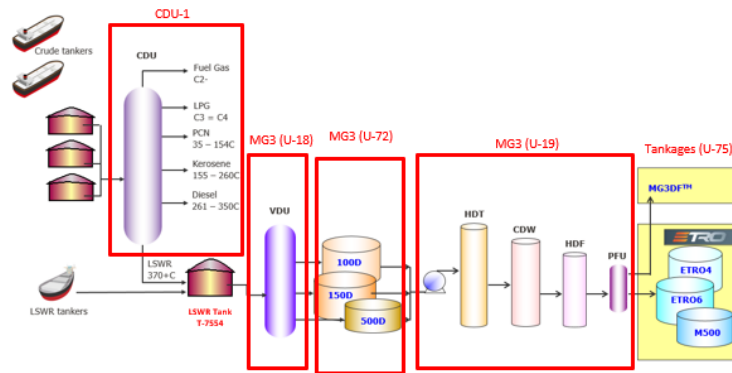


Figure 1.1: Overview of the processes within MRCSB.

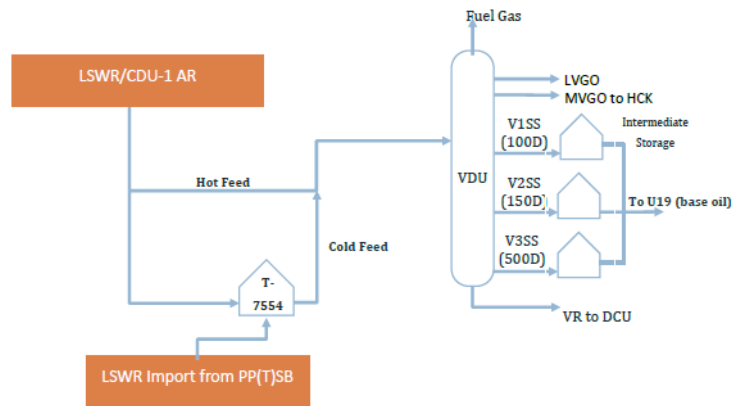


Figure 1.2: Overview of the Vacuum Distillation Unit (Unit 18).

Extensive work was done on developing and modifying the source code for predicting the predictor variables for respective target variables of the VDU which will be explained further in Chapter 3.

1.2 Problem Statement

The primary problem that we are trying to solve through this project is to identify the key predictor variables which influence the physical properties such as density, kinematic viscosity, pour point and etc. of the desired streams within the lubricant base oil production process, especially for the CDU and VDU. This would be crucial in understanding the process better and possibly setting up better control structures to maintain optimal value of the product physical properties, which could produce high quality products consistently, thus maximising profitability.

1.3 Objectives

The main objectives of this project include:

- i. Development of predictive models for the Vacuum Distillation Unit.

Predictive models for target variables in the vacuum distillation unit will be developed through XGBoost algorithm only using R. These models are to be developed based on a set number of input variables, such as draw temperature of the streams, pump around flow rates, feed flow rates and etc., to predict and understand the correlation between these input variables and the target variables.

- ii. Comparison between algorithms using available dataset.

Source codes for machine learning algorithms (Decision Tree, Random Forest and Extreme Gradient Boosting) will be developed and these algorithms are compared in terms of their computation time, accuracy of the prediction and other relevant data as deemed necessary for a set number of target variables. This will serve as a benchmark for further machine learning works with regards to the refinery processes property prediction.

1.4 Scope of Study

The scope of this project is limited to:

- i. Developing the source codes to produce predictive models for 21 target variables within the Vacuum Distillation Unit using both XGBoost and Random Forest algorithms and perform necessary hyperparameter tuning to achieve better predictive performance of the model.
- ii. Analysing and comparing the results from both algorithms and recommend appropriate measures for future work.

CHAPTER 2

LITERATURE REVIEW

In this project, multivariable regression is used to develop the model based on the target variables. However, the selection of the appropriate algorithm is key in developing the regression model. Among the most common algorithms which are used in the era of machine learning includes Decision Tree, Random Forest and Extreme Gradient Boost. Each of these algorithms are further explained below.

2.1 Decision Tree

Decision Tree, specifically the Classification and Regression Tree (CART) is the most common type of data mining method used to develop prediction algorithms. This method classifies a given data into a flowchart-like tree structure with a root node, internal node and a leaf node. The root node splits all of the data into two or more specific subsets based on the feature that has the best split and the internal node splits the current tree level into respective subdivisions similarly. The leaf node represents the terminal node and contains the output label. In order to identify the right feature with the best split at each node, Gini Index is used. (Song & Lu, 2015)

Gini Index is a measure of impurity (the amount of inaccuracy in separating the data) and lower Gini index value shows better separation of the data using a particular feature. The feature with the lowest Gini Index becomes the root of the decision tree. Then, each node of the root is further separated using a feature that has the lowest Gini index in splitting the node. However, if the Gini index of the node is

much lower as compared to that of the feature with the lowest score, then the separation of the node is not needed. (Saxena, 2017) Since the Decision Tree forms a single tree and depends on the lowest Gini Index to determine the respective features on the nodes of tree, the results of the Decision Tree model would be skewed for datasets with significant outliers and noise. The model also would only work well for a small sample size, thus making this algorithm undesirable for this study.

2.2 Random Forest

Random Forest is a widely used machine learning algorithm and could be used to develop both regression and classification type models. For a given dataset, Random Forest algorithm first randomly extracts rows of data from the primary dataset with replacement and adds them into a bootstrap dataset. Since the samples are taken randomly with replacements, the bootstrap data would roughly contain about 60% of the actual rows from the primary dataset and have several duplicate rows. Following this, the algorithm would randomly select a number of features (or columns) from the bootstrap dataset to create individual trees. This method of bootstrapping and aggregating the data is known as “bagging”.

The number of trees (ntree) is generally defined by the user and a higher number of tree would result in a better prediction accuracy at the expense of computation time. The final result of the algorithm is done based on a majority vote of the output values when a test data is introduced. Thus, Random Forest algorithm generally performs better in classification type models. (Nikulski, 2016)

Aside from ntree, another variable that is important in Random Forest is mtry, which represents the number of random variables used in each tree within the ensemble. Fine tuning both these values allows us to obtain a fairly accurate model in most situations. Bhalla (n.d.) has stated the optimal way to tune a Random Forest model is by using default mtry value and manipulating the ntree value to achieve a low Out-of-Bag (OOB) error. Upon achieving a stable OOB error rate, a number of mtry tuning values are used to find the optimal mtry value where the OOB error is at the

minimum. Initial mtry values can be approximated to be the square root of the number of total input variables or predictors, half of the square root value and twice of the square root value. The OOB error stated earlier is Random Forest algorithm's way of performing internal model validation. It utilises the data which are not taken into the bootstrap dataset (the bootstrap data is analogous to a bag) and performs cross validation of the model to compute the performance of the model.

One other variable that impacts the performance of the Random Forest model is the node size. Node size is the minimum size of the tree nodes and represents the minimum number of sample or data points that it can hold at the node. Having a high node size would result in a less complex and shallow tree while a lower node size would create a complex and deep tree. Altogether, the optimal value of these three parameters could produce a model that is robust against noise and outliers present in the data and produces accurate predictions. (Boehmke & Greenwell, 2020)

2.3 Extreme Gradient Boost (XGBoost)

XGBoost which was introduced by Tianqi Chen and Carlos Guestrin in 2016 is a relatively new algorithm which is based on "boosting" where weak learners are used sequentially to develop the final model. It uses the second-order Taylor's approximation to simplify the objective loss function. This function is then minimised and produces an output value for the residuals on respective leaf nodes. To minimise split loss, XGBoost calculates the gain of respective nodes and this is controlled by the hyperparameter, gamma for pruning of the tree. XGBoost is ought to be the best performing predictive model algorithm available as of now based on its computing performance and prediction accuracy. (Leventis, 2018) One benefit of using XGBoost is that upon constructing the model, it is relatively simple to extract the importance scores of respective features, which are used to develop the model. This enables the user to identify the predictor variables with the highest contribution factor or gain value and understand its significance in a process.

$$\text{Gain} = \text{Output (Parent Node)} - (\text{Output (Left Node)} + \text{Output (Right Node)}) \quad (1)$$

Wu (2018) has suggested that XGBoost resulted in the lowest Mean Absolute Error with Random Forest up next for the prediction of marine debris using the data from International Coastal Cleanup based on Table 2.1.

Table 2.1: Modelling Results for Predicting Marine Debris (Wu,2018).

Models	Ordinary Least Square	Decision Tree	Random Forest	Generalized Boosted Model	Extreme Gradient Boosting
MAE	1382.50	998.29	836.01	910.00	807.83

This superior performance by the XGBoost algorithm is made possible by a number of enhancements which improved its computation speed and model development performance, which are tabulated in Table 2.2. (Chen & Guestrin, 2016) (Morde & Setty, 2019)

Table 2.2: Attributes of XGBoost algorithm.

Attributes	Description
Exact Greedy Algorithm	For manageable data sizes, the algorithm identifies the best split of data by running through every datapoint.
Approximate Greedy Algorithm	Most trees use exact greedy algorithm to identify the best split of data and to minimise losses. However, this poses a difficulty when the file size is large and it does not fit in the system memory for computations. This is where approximate greedy algorithm kicks in where instead of enumerating all the data points, it selects splitting points based on percentile of feature distribution.
Weighted Quantile Sketch	To identify the optimal splitting point, an evenly distributed data could be split into percentile of $1/n$ using the quantile sketch to identify the splitting points. However, there is no quantile sketch for weighted datasets and this addition to XGBoost allows it to handle weighted datasets.
Sparsity-aware Split Finding	In real world, the dataset would be filled with missing values and the XGBoost algorithm is developed to recognise this missing value and get around to the non-missing data.
Cache-aware Access	Limits the size of data within a block based on the available CPU cache to balance the cache property and parallelisation
Out-of-core Computation	Optimises the algorithm to maximise the usage of the device's hardware. It allows the data which does not fit into the system memory to be compressed or sharded into multiple blocks to the disks and runs an algorithm to read the data from the respective disks.
Regularization	Penalizes complex models using both Lasso and Ridge regression using alpha and gamma values respectively.
Built-in Cross Validation	Cross validation partitions the training dataset into n number and trains the model using $n-1$ partitions and runs a test using the untrained data. The model is trained and tested for n number of times and the hyperparameter tunes could be retrieved.

CHAPTER 3

METHODOLOGY / PROJECT WORK

3.1 Softwares and Tools Used

The list of software used throughout the project period is described in Table 3.1.

Table 3.1: List of Softwares.

No.	Software	Description
1	Microsoft Office (Microsoft Word, Microsoft PowerPoint, Microsoft Excel)	A collection of softwares used for documentations, data analysis, data visualisation using graphs and tables, and presentation slides.
2	Adobe Acrobat Reader	A software used to view and print digital documents.
3	RStudio Version 4.0.3 (2020-10-10)	An interactive interface to develop and run R-based source codes

3.2 Packages and Key Functions

Upon installation, RStudio runs of a *base* package which contains a large number of basic functions. However, this is not sufficient to achieve the desired result. Thus, additional packages could be installed as needed from RStudio itself or from the Comprehensive R Archive Network (CRAN) repository. Some of the key packages and functions which are used in this project are described in Table 3.2.

Table 3.2: List of Packages and Functions.

No.	Packages	Functions and Description
1	caret (Miscellaneous functions for training and plotting classification and regression models)	<ul style="list-style-type: none"> • createDataPartition Used to split the data into training and testing datasets • confusionMatrix Calculates a cross-tabulation of observed and predicted values with associated statistics
2	mice (Multivariate Imputation using Chained Equations)	<ul style="list-style-type: none"> • mice Performs imputation (replaces missing data with predicted data)
3	xgboost (Extreme Gradient Boosting)	<ul style="list-style-type: none"> • xgboost Running the XGBoost algorithm and generates a model • xgb.importance Creates an importance list with the gains of each predictor variables with respect to the target variable • xgb.importance.plot Plots an importance plot that represents the gains of respective predictor variables • xgb.save Save XGBoost model in a binary model file • xgb.load Load XGBoost model from the binary model file
4	mlr (Interface to a large number of classification and regression techniques)	<ul style="list-style-type: none"> • makeRegrTask Create a regression task • makeLearner Create learner object • train Train a learning algorithm • makeTuneControlRandom Create control object for hyperparameter tuning with random search • makeResampleDesc Create a description object for a resampling strategy • tuneParams Hyperparameter tuning
5	randomForest	<ul style="list-style-type: none"> • randomForest Generates the Random Forest model. • varImpPlot Generates the importance plot for the Random Forest model
6	parallelMap	<ul style="list-style-type: none"> • parallelStartSocket Enables parallelization backend

An important aspect to take note here is that the sequence of the packages matters especially when two or more packages to be used has the same function name. Figure 3.1 displays an example where the function “train” is present in both the caret and mlr packages. Since the mlr package is added after the caret package, the “train” function from the caret package is masked and it could only be used if the caret package is re-added. In this situation, the “train” function from the mlr package is utilised.

```
> library(mlr)      # ML package (also some data manipulation)
Loading required package: ParamHelpers
'mlr' is in maintenance mode since July 2019. Future development efforts will go
into its successor 'mlr3' (<https://mlr3.ml-org.com>).

Attaching package: 'mlr'

The following object is masked from 'package:caret':

  train
```

Figure 3.1: Snapshot of function masking when adding packages.

The arguments within the XGBoost function also requires special attention as each parameter is critical in developing a good model. Table 3.3 describes the common hyperparameters which are defined in the XGBoost algorithm. Only rounds is required to be defined by the user and other parameters have default values when the algorithm is run. However, the default values could lead to overfitting models in some cases which could result in poor prediction. Thus, hyperparameter tuning is significant to achieve a model with the ability to predict with high accuracy. (XGBoost Parameters, n.d.)

Table 3.3: Hyperparameter description.

No	Hyperparameters	Description
1	nrounds	Maximum number of iteration/ trees grown Value: Must be user defined
2	eta	Learning rate of the model based on the tree output Default value: 0.3 (Range: 0 – 1)
3	gamma	Pruning of the leave nodes Default value: 0 (Range: 0 – Inf)
4	max_depth	Depth of the tree Default value: 6 (Range: 0 – Inf) *Deeper trees tend to overfit
5	min_child_weight	Minimum number of residuals on each leaf node Default value: 1 (Range: 0 – Inf)
6	subsample	Number of samples (rows) supplied to the tree Default value: 1 (Range: 0 – 1)
7	colsample_bytree	Number of features (column) supplied to the tree Default value: 1 (Range: 0 – 1)
8	lambda	L2 regularization (Ridge regression) Default value: 0 (Range: 0 – Inf)
9	alpha	L1 regularization (Lasso regression) Default value: 1 (Range: 0 – Inf)

When the algorithm builds each tree and works on achieving the best model, it requires an objective to develop a classification or regression model and an evaluation metric to compare the actual data and the output of the tree to improve on and build a tree with a better fit. Table 3.4 and Table 3.5 displays some of the types of objectives and evaluation metrics respectively. (XGBoost Parameters, n.d.)

Table 3.4: Learning Task Parameters – “objective”.

No	Options	Description
1	reg:squarederror	Regression with squared loss
2	reg:squaredlogerror	Regression with squared log loss
3	reg:logistic	Logistic regression
4	reg:pseudohubererror	Regression with Pseudo Huber loss
5	binary:logistic	Logistic regression with binary classification. Outputs probability
6	binary:hinge	Hinge loss for binary classification. Outputs 0 or 1 instead of probabilities
7	rank:pairwise	Performs pairwise ranking where pairwise loss is minimised
8	rank:map	Perform list-wise ranking where Mean Average Precision is maximised

Table 3.5: Learning Task Parameters – “eval_metric”.

No	Options	Description
1	rmse	Root mean square error Default metric for reg:squaredloss
2	rmsle	Root mean square log error Default metric for reg:squaredlogloss
3	mae	Mean absolute error
4	mphe	Mean Pseudo Huber error Default metric for reg:pseudohubererror
5	error	Binary classification error rate where split occurs at 0.5

6	error@t	Binary classification error rate where split occurs at “t” (user-defined value)
7	map	Mean Average Precision

3.3 Source Code Overview

The developed source code (refer Appendix IV) initially starts off with the packages specifically XGBoost and mlr packages. The data is read from a comma-separated value file (.csv) and irrelevant variables are removed from the RStudio environment. The input variables (Appendix II) and one of the output variable or target variable (Appendix III) is extracted from the master dataset and added into a new dataset. The data from the new dataset is then split into training and testing dataset with a ratio of 70:30. Following that, a regression task for training and testing dataset is created alongside a learner object which results in a model. This model, however, is derived from the mlr package and uses the argument “reg:xgboost” in the learner to execute the XGBoost algorithm for a regression model. Then, the hyperparameter boundary limits were set and the control structure was set to run 200 random combinations of hyperparameters with a 10-fold cross validation. Upon the completion of the hyperparameter tuning, a XGBoost model was developed using the tuned hyperparameter values. The results from the model and the actual test data were stored in a .csv file and a residual plot is made to identify the coefficient of determination or R^2 value of the predicted values. A plot of predicted against actual was also made for each target variable to check for the fit of the model using a test data. An importance plot, which displays the key predictor variables, is also attained before the XGBoost model was saved as a binary model file. Figure 3.2 highlights the key steps within the source code.

To develop the Random Forest model, similar source code to that of XGBoost is used and the relevant parameters are specified with their boundaries set to perform parameter tuning before generating the model. The test data is run through the model and the results are saved as a CSV file in Excel before further analysis are performed.

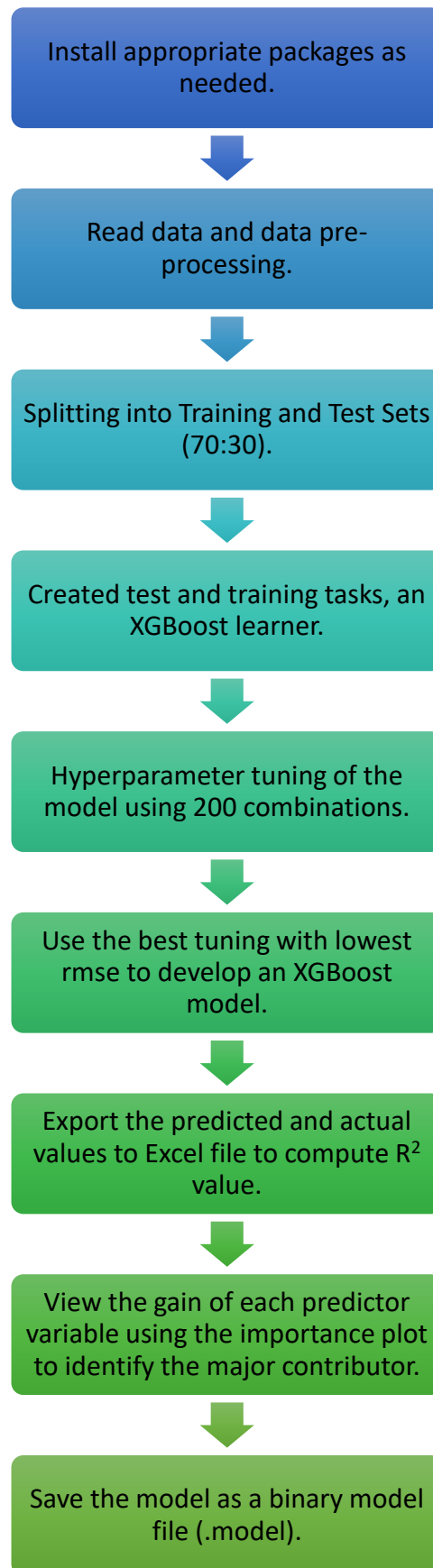


Figure 3.2: Flowchart of the Source Code.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 XGBoost Model Results

4.1.1 Predictive Model Fit

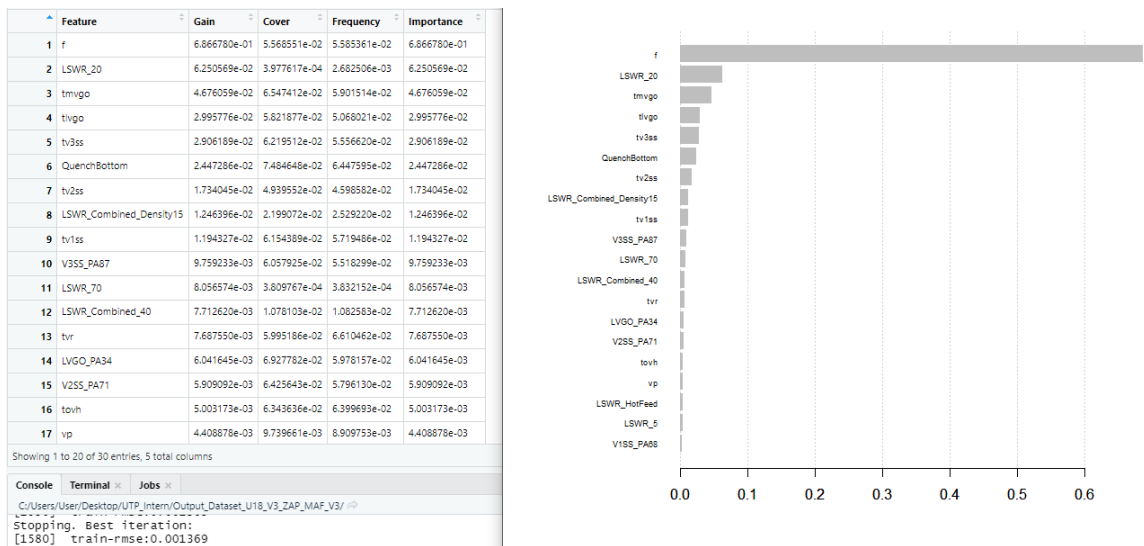
In this project, 21 predictive models are developed using the Extreme Gradient Boosting (XGBoost) algorithm for the target variables shown in Table 4.1 using 33 input variables as stated in Chapter 3. The list of abbreviations could be referred in Appendix I. Based on Table 4.1, the total feed flow rate, f and the Low Sulphur Waxy Residue (LSWR) simulated distillation at 70%, LSWR_70 are seen to be the primary predictor for a number of target variables. However, both V2SS_Density and V2SS_Sulphur models show poor coefficient of determination, R^2 when the test data is passed through the developed model.

Table 4.1: Coefficient of Determination, Main Feature and Gain for Respective Target Variables using XGBoost.

Target Variable	R ²	Main Feature / Predictor Variable	Gain
fv1ss	0.96	f	0.69
fv2ss	0.94	f	0.68
fv3ss	0.68	V2SS_PA71	0.18
y1	0.69	tlvgo	0.23
y2	0.83	LSWR_5	0.19
y3	0.75	f	0.24
V1SS_Density15	0.872	LSWR_Combined_40	0.91
V2SS_Density15	0.0001	LVGO_PA34	0.27
V3SS_Density15	0.9283	LSWR_Combined_Density15	0.25
V1SS_kv100	0.9469	LSWR_20	0.40
V2SS_kv100	0.8443	tmvgo	0.22
V3SS_kv100	0.9037	tmvgo	0.28
V1SS_Sulphur	0.8619	LVGO_PA34	0.18
V2SS_Sulphur	0.0015	tv1ss	0.21
V3SS_Sulphur	0.83	V3SS_PA87	0.13
V1SS_Nitrogen	0.9181	LSWR_90	0.24
V2SS_Nitrogen	0.9409	LSWR_70	0.46
V3SS_Nitrogen	0.9364	LSWR_70	0.42
V1SS_Aromatic	0.9436	LSWR_70	0.38
V2SS_Aromatic	0.962	LSWR_70	0.41
V3SS_Aromatic	0.9387	LSWR_50	0.22

4.1.2 Importance Plots

The XGBoost algorithm develops a tree ensemble and is able to identify the gain value or contribution factor of respective input variables by removing or adding the features and observing whether the accuracy of the model increases or decreases. These variables are then sequenced in terms of the gain values in descending order in an importance plot such as the example shown in Figure 4.1 for all of the target variables of side stream V1SS. The importance plot highlights the gain, cover and the frequency. Gain represents the contribution of a feature as explained previously; cover represents the relative number of observations related to a feature; and frequency is the number of times an input variable is used at the node to split the data.



(a) V1SS Flowrate

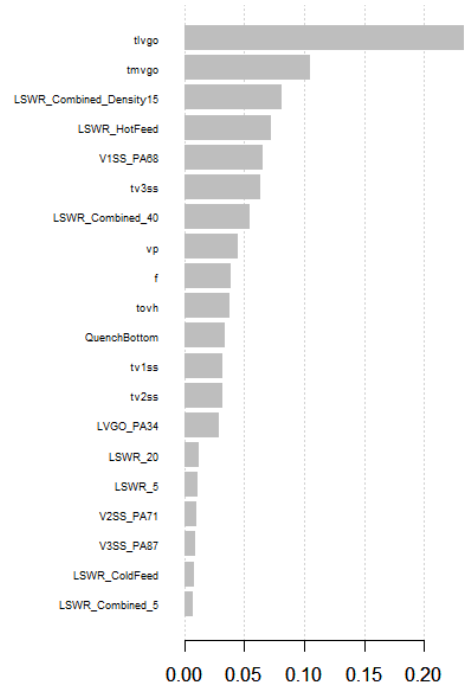
Feature	Gain	Cover	Frequency	Importance
1 tvlgo	0.233471460	0.073922968	0.062934818	0.233471460
2 tmvgo	0.104537100	0.070925621	0.071283314	0.104537100
3 LSWR_Combined_Density15	0.080713106	0.038629626	0.038531521	0.080713106
4 LSWR_HotFeed	0.071968921	0.051900428	0.049983945	0.071968921
5 V1SS_PA68	0.065374098	0.048624229	0.050947233	0.065374098
6 tv3ss	0.063262100	0.058180474	0.059509793	0.063262100
7 LSWR_Combined_40	0.054370791	0.014322669	0.013807128	0.054370791
8 vp	0.044136495	0.012123755	0.012843840	0.044136495
9 f	0.038868937	0.044899144	0.048057369	0.038868937
10 tovh	0.037373658	0.047353289	0.048699561	0.037373658
11 QuenchBottom	0.033280228	0.059032624	0.057797281	0.033280228
12 tv1ss	0.031934040	0.069925000	0.064861394	0.031934040
13 tv2ss	0.031702336	0.033716976	0.035962753	0.031702336
14 LVGO_PA34	0.029139772	0.064299513	0.062613722	0.029139772
15 LSWR_20	0.012043768	0.011511202	0.008027400	0.012043768
16 LSWR_5	0.011288477	0.003419447	0.005137536	0.011288477
17 V2SS_PA71	0.009928720	0.052105463	0.054800385	0.009928720

Showing 1 to 20 of 26 entries, 5 total columns

```

Console Terminal Jobs
C:/Users/User/Desktop/UTP_Intern/Output_Dataset_U18_V3_ZAP_MAF_V3/ >
Stopping. Best iteration:
[3144] train-rmse:0.001747

```



(b) V1SS Yield

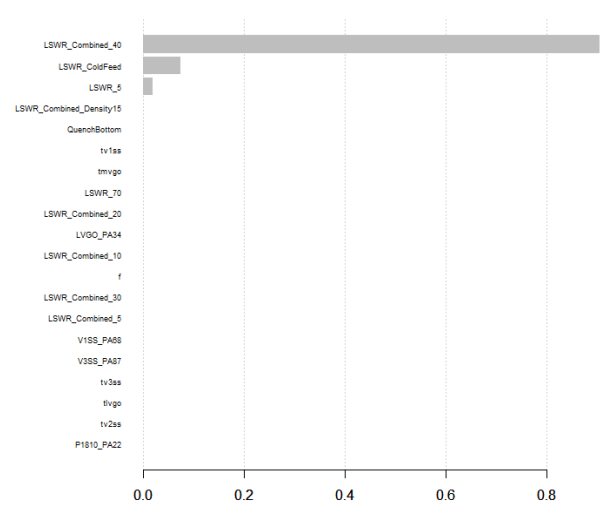
Feature	Gain	Cover	Frequency	Importance
1 LSWR_Combined_40	9.062376e-01	3.781769e-02	0.1262309758	9.062376e-01
2 LSWR_ColdFeed	7.463580e-02	2.122456e-01	0.1038495971	7.463580e-02
3 LSWR_5	1.912656e-02	1.436485e-01	0.0653536258	1.912656e-02
4 LSWR_Combined_Density15	2.111331e-09	1.108604e-01	0.0975828111	2.111331e-09
5 QuenchBottom	4.748942e-10	4.423833e-02	0.0546105640	4.748942e-10
6 tv1ss	3.384676e-10	3.465516e-02	0.0429722471	3.384676e-10
7 tmvgo	3.068935e-10	3.833092e-02	0.0510295434	3.068935e-10
8 LSWR_70	2.308373e-10	4.666619e-03	0.0026857654	2.308373e-10
9 LSWR_Combined_20	2.030888e-10	2.101219e-02	0.0232766338	2.030888e-10
10 LVGO_PA34	1.990166e-10	3.082189e-02	0.0358102059	1.990166e-10
11 LSWR_Combined_10	1.831582e-10	3.600901e-02	0.0259623993	1.831582e-10
12 f	1.783955e-10	2.490209e-02	0.0349149508	1.783955e-10
13 LSWR_Combined_30	1.750715e-10	2.438709e-02	0.0223813787	1.750715e-10
14 LSWR_Combined_5	1.481241e-10	4.221551e-02	0.0501342883	1.481241e-10
15 V1SS_PA68	1.359221e-10	1.977337e-02	0.0313339302	1.359221e-10
16 V3SS_PA87	1.307570e-10	4.541698e-02	0.0367054611	1.307570e-10
17 tv3ss	1.212778e-10	2.467733e-02	0.0367054611	1.212778e-10

Showing 1 to 20 of 28 entries, 5 total columns

```

Console Terminal Jobs
C:/Users/User/Desktop/UTP_Intern/Output_Dataset_U18_V3_ZAP_MAF_V3/ >
Stopping. Best iteration:
[71] train-rmse:0.000569

```



(c) V1SS Density

Feature	Gain	Cover	Frequency	Importance
1 LSWR_20	3.954280e-01	0.0037713863	0.0046900191	3.954280e-01
2 V1SS_PA68	9.698930e-02	0.0636009228	0.0482192584	9.698930e-02
3 LSWR_70	5.873704e-02	0.0058493367	0.0016121940	5.873704e-02
4 V3SS_PA87	5.555681e-02	0.0556447995	0.0512970834	5.555681e-02
5 LSWR_60	5.409222e-02	0.0027396236	0.0008793786	5.409222e-02
6 LSWR_Combined_Density15	5.083165e-02	0.0390148036	0.0411842298	5.083165e-02
7 LVGO_PA34	4.489672e-02	0.0580517164	0.0474864429	4.489672e-02
8 LSWR_HotFeed	4.239867e-02	0.0388427082	0.0438223655	4.239867e-02
9 LSWR_ColdFeed	2.509424e-02	0.0632468545	0.0794371977	2.509424e-02
10 tv1ss	2.208402e-02	0.0559075470	0.0483658215	2.208402e-02
11 tv3ss	1.971902e-02	0.0446510585	0.0480726953	1.971902e-02
12 LSWR_Combined_40	1.848125e-02	0.0162785978	0.0150959988	1.848125e-02
13 tmvgo	1.600587e-02	0.0433757717	0.0470467536	1.600587e-02
14 LSWR_Combined_5	1.508605e-02	0.0294448123	0.0631686941	1.508605e-02
15 f	1.224110e-02	0.0486467434	0.0507108310	1.224110e-02
16 tvgo	1.005354e-02	0.0444011279	0.0444086179	1.005354e-02
17 tv2ss	8.370483e-03	0.0635715506	0.0423567346	8.370483e-03

Showing 1 to 20 of 30 entries, 5 total columns

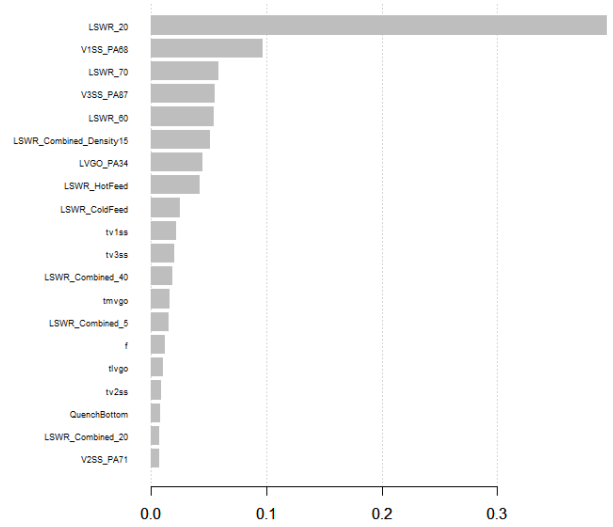
Console Terminal Jobs

C:/Users/User/Desktop/UTP_Intern/Source_Code/

```

Stopping. Best iteration:
[216] train-rmse:0.000969

```



(d) V1SS Kinematic Viscosity

Feature	Gain	Cover	Frequency	Importance
1 LVGO_PA34	0.1803802457	0.0809506023	0.076160991	0.1803802457
2 LSWR_Combined_Density15	0.1400498332	0.0931193733	0.080495356	0.1400498332
3 tmvgo	0.0904934349	0.0581978630	0.057585139	0.0904934349
4 LSWR_90	0.0635091839	0.0219132129	0.006191950	0.0635091839
5 tvgo	0.0516899229	0.0550160400	0.051393189	0.0516899229
6 V3SS_PA87	0.0463274596	0.0794617954	0.063157895	0.0463274596
7 LSWR_80	0.0462671095	0.0215589013	0.005572755	0.0462671095
8 tv1ss	0.0433766756	0.0478582474	0.047055824	0.0433766756
9 f	0.0421284759	0.0362253073	0.059442724	0.0421284759
10 LSWR_Combined_5	0.0415177294	0.0501918170	0.068114455	0.0415177294
11 tv3ss	0.0328195159	0.0528552628	0.056965944	0.0328195159
12 P1810_PA22	0.0284117023	0.0308286005	0.027863777	0.0284117023
13 tv2ss	0.0268781236	0.0464671817	0.034674923	0.0268781236
14 tovh	0.0264456828	0.0304271637	0.040247678	0.0264456828
15 V1SS_PA68	0.0205408175	0.0400791005	0.038390093	0.0205408175
16 LSWR_Combined_10	0.0177006022	0.0242502732	0.026006192	0.0177006022
17 LSWR_Combined_30	0.0165612735	0.0232554081	0.022910217	0.0165612735

Showing 1 to 20 of 28 entries, 5 total columns

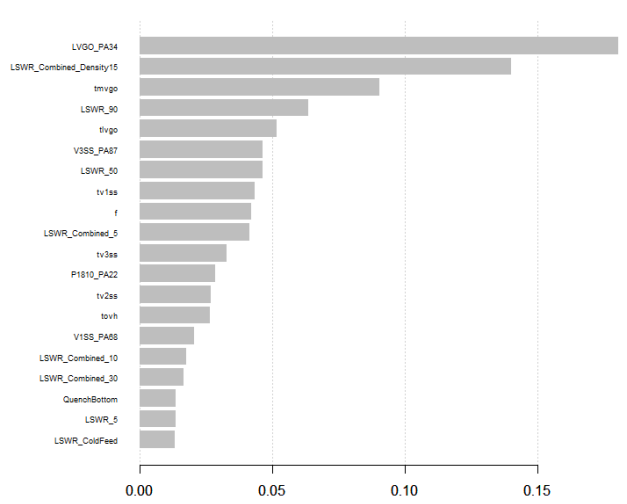
Console Terminal Jobs

C:/Users/User/Desktop/UTP_Intern/Output_Dataset_U18_V3_ZAP_MAF_V3/

```

[88] train-rmse:0.000603
Stopping. Best iteration:
[78] train-rmse:0.000603

```

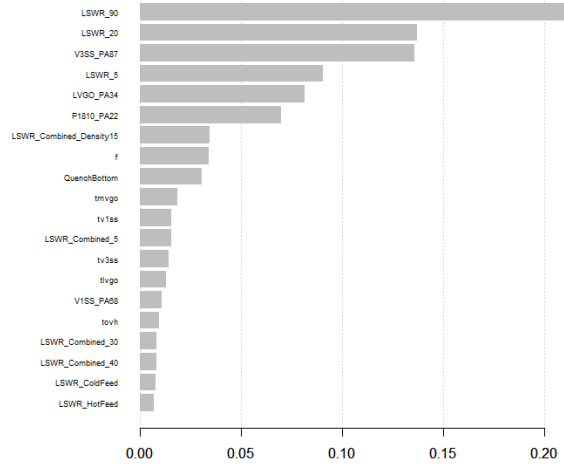


(e) V1SS Sulphur

Feature	Gain	Cover	Frequency	Importance
1 LSWR_90	2.351667e-01	1.964977e-03	0.0008647173	2.351667e-01
2 LSWR_20	1.372266e-01	2.070203e-03	0.0058254636	1.372266e-01
3 V3SS_PA87	1.359192e-01	6.194644e-02	0.0496757310	1.359192e-01
4 LSWR_5	9.056750e-02	6.688750e-03	0.0322903630	9.056750e-02
5 LVGO_PA34	8.153592e-02	7.452040e-02	0.0520195699	8.153592e-02
6 P1810_PA22	6.972854e-02	3.388483e-02	0.0224143816	6.972854e-02
7 LSWR_Combined_Density15	3.472035e-02	2.721505e-02	0.0325406758	3.472035e-02
8 f	3.428324e-02	5.640961e-02	0.0537945159	3.428324e-02
9 QuenchBottom	3.060100e-02	7.017022e-02	0.0547730117	3.060100e-02
10 tmygo	1.877121e-02	7.190812e-02	0.0505176926	1.877121e-02
11 tv1ss	1.569265e-02	4.682896e-02	0.0467402435	1.569265e-02
12 LSWR_Combined_5	1.556378e-02	2.073635e-02	0.0641938787	1.556378e-02
13 tv3ss	1.424715e-02	5.585480e-02	0.0448287632	1.424715e-02
14 tvigo	1.308202e-02	6.000224e-02	0.0492433724	1.308202e-02
15 V1SS_PA68	1.093360e-02	5.410235e-02	0.0427807487	1.093360e-02
16 tovh	9.506047e-03	7.386894e-02	0.0700876095	9.506047e-03
17 LSWR_Combined_30	8.325805e-03	6.994576e-03	0.0108999866	8.325805e-03

Showing 1 to 20 of 30 entries, 5 total columns

Console Terminal Jobs
C:/Users/User/Desktop/UTP_Intern/Output_Dataset_U18_V3_ZAP_MAF_V3/ />
[2004] train-rmse: 0.001426

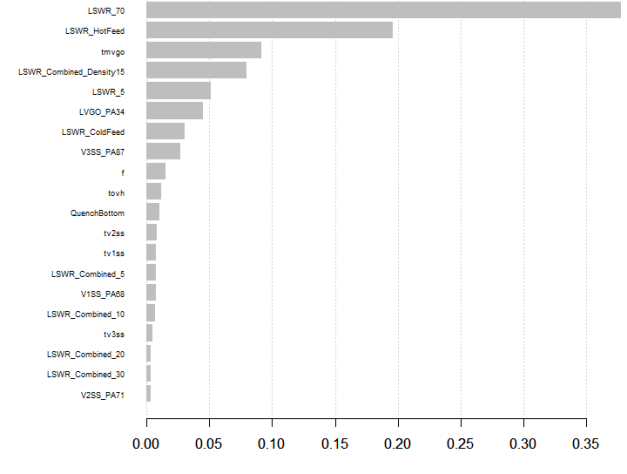


(f) V1SS Nitrogen

Feature	Gain	Cover	Frequency	Importance
1 LSWR_70	3.774959e-01	5.539178e-03	1.579703e-03	3.774959e-01
2 LSWR_HotFeed	1.958753e-01	3.352094e-02	4.155096e-02	1.958753e-01
3 tmygo	9.126109e-02	6.013761e-02	5.136429e-02	9.126109e-02
4 LSWR_Combined_Density15	7.990383e-02	2.521468e-02	3.389181e-02	7.990383e-02
5 LSWR_5	5.127263e-02	3.567421e-03	3.958832e-02	5.127263e-02
6 LVGO_PA34	4.519059e-02	7.493572e-02	5.088559e-02	4.519059e-02
7 LSWR_ColdFeed	3.075800e-02	7.936488e-02	6.544758e-02	3.075800e-02
8 V3SS_PA87	2.731834e-02	6.198334e-02	4.423169e-02	2.731834e-02
9 f	1.509235e-02	5.829599e-02	5.730014e-02	1.509235e-02
10 tovh	1.162220e-02	6.088675e-02	5.773097e-02	1.162220e-02
11 QuenchBottom	1.051867e-02	7.176963e-02	4.906654e-02	1.051867e-02
12 tv2ss	8.174225e-03	5.085576e-02	4.064146e-02	8.174225e-03
13 tv1ss	7.741859e-03	6.480019e-02	4.758258e-02	7.741859e-03
14 LSWR_Combined_5	7.434528e-03	1.988816e-02	8.118717e-02	7.434528e-03
15 V1SS_PA68	7.341867e-03	4.009764e-02	3.465773e-02	7.341867e-03
16 LSWR_Combined_10	7.066833e-03	1.638375e-02	1.780756e-02	7.066833e-03
17 tv3ss	5.061386e-03	5.589901e-02	4.681666e-02	5.061386e-03

Showing 1 to 20 of 30 entries, 5 total columns

Console Terminal Jobs
C:/Users/User/Desktop/UTP_Intern/Output_Dataset_U18_V3_ZAP_MAF_V3/ />
[627] train-rmse: 0.001288



(g) V1SS Aromatic

Figure 4.1: Importance plots of V1SS using XGBoost.

4.1.3 Predicted vs Actual Plots

Upon developing the models, the models are tested using a test data to observe its performance. Figure 4.2 shows the actual against predicted graph of target variable y_1 , where the data points follow the 45° reference line which indicates that the predicted data closely approximates the actual data of y_1 . The data has a coefficient of determination, R^2 value of 0.6923 as shown in Table 4.1.

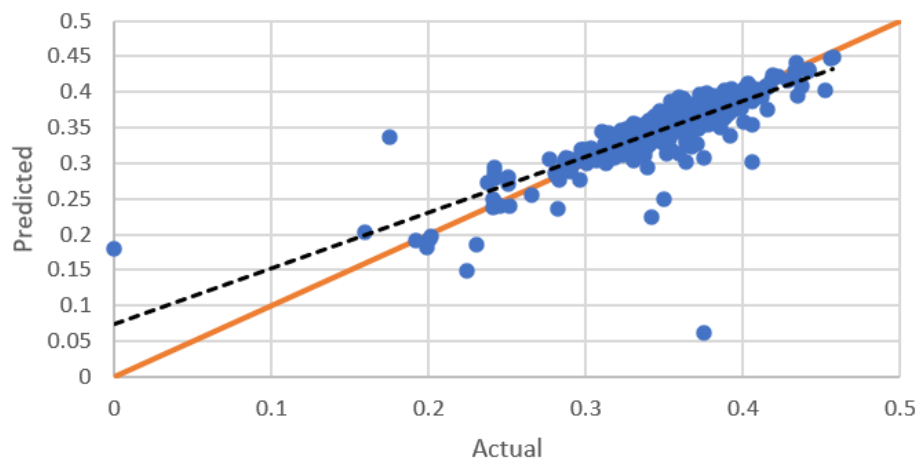
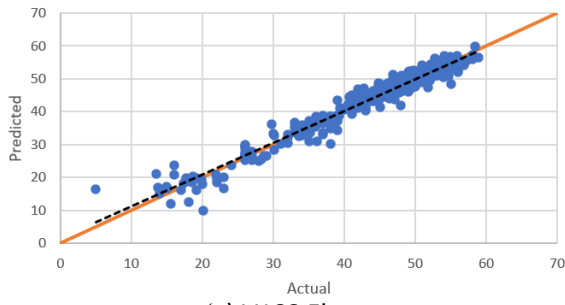
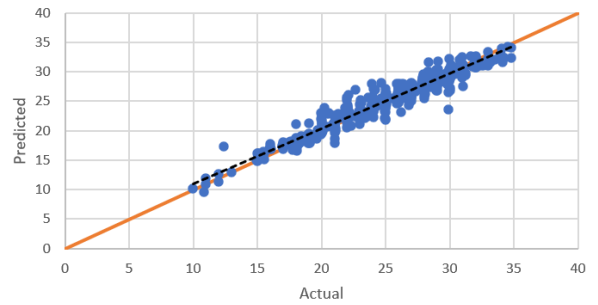


Figure 4.2: Graph of Predicted vs Actual of V1SS Yield using XGBoost.

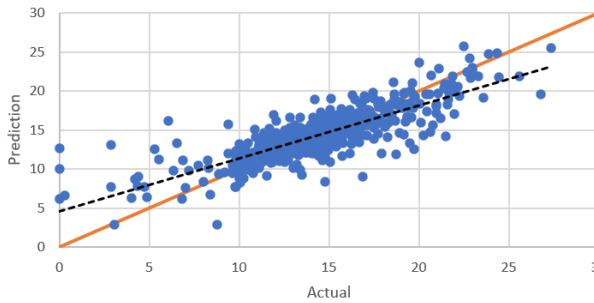
Figure 4.3 displays the actual versus predicted graphs of all the other 20 variables besides y_1 . All of these graphs except Figure 4.3 (g) and Figure 4.3 (m) highlights that the model has a good fit with the data and is reliable in terms of its prediction and accuracy. However, for Figure 4.3 (g) and Figure 4.3 (m), the discrepancy in the data of density and sulphur content respectively has resulted in a poor model fit. These values were not treated as outliers since the large values occurred several times within the dataset.



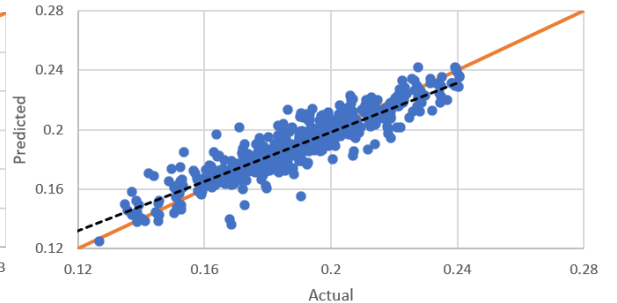
(a) V1SS Flowrate



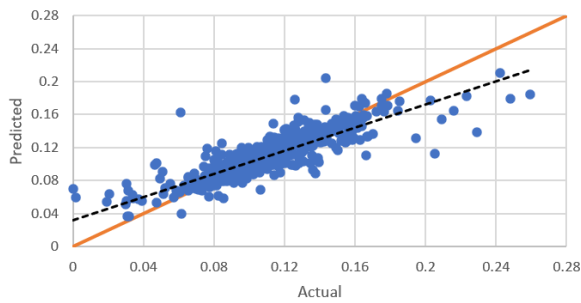
(b) V2SS Flowrate



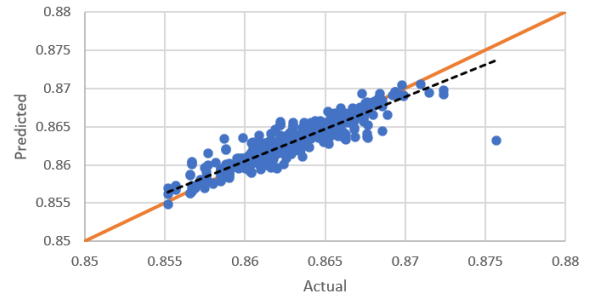
(c) V3SS Flowrate



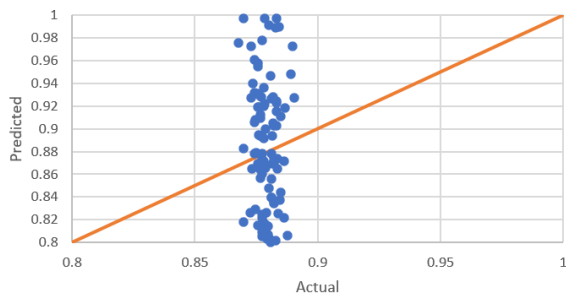
(d) V2SS Yield



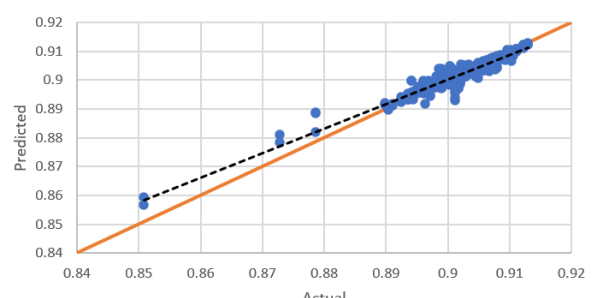
(e) V3SS Yield



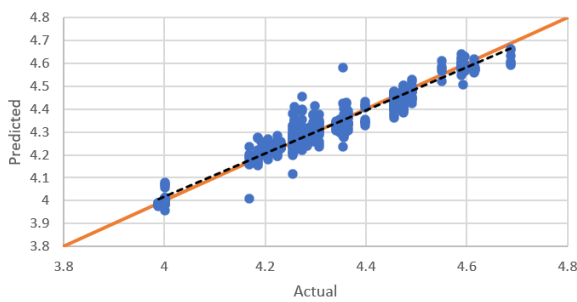
(f) V1SS Density



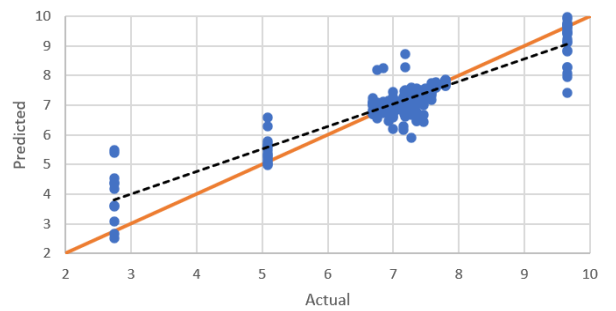
(g) V2SS Density



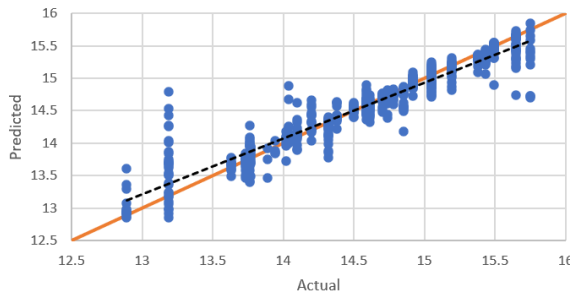
(h) V3SS Yield



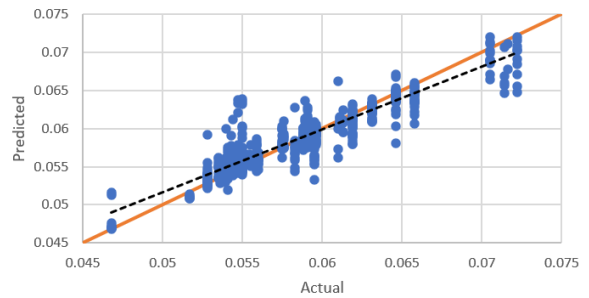
(i) V1SS kv100



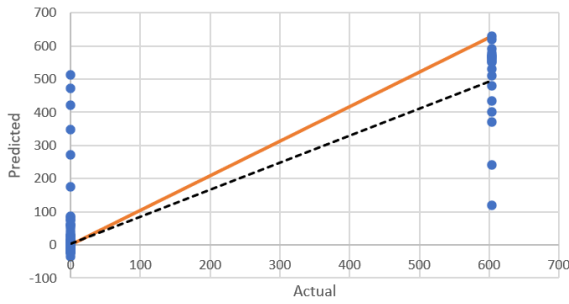
(j) V2SS kv100



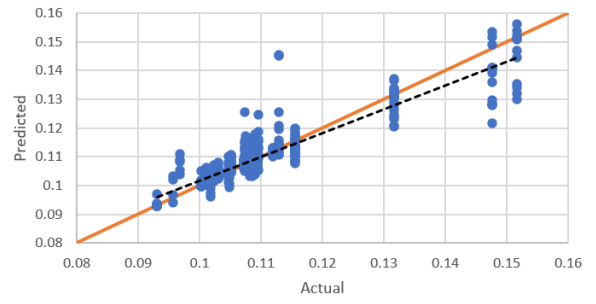
(k) V3SS kv100



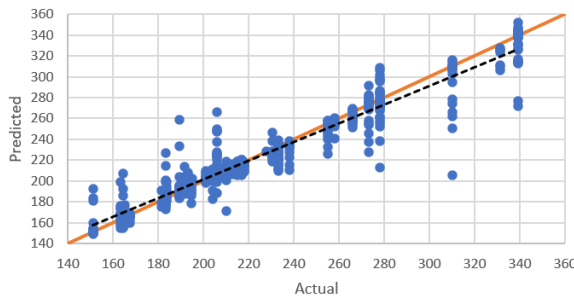
(l) V1SS Sulphur



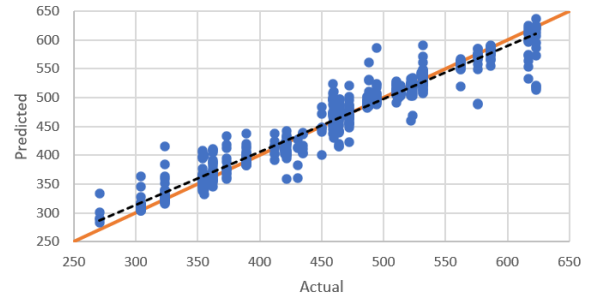
(m) V2SS Sulphur



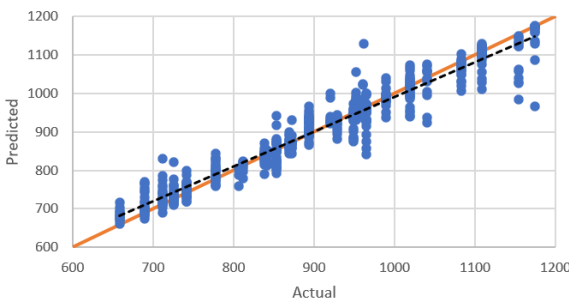
(n) V3SS Sulphur



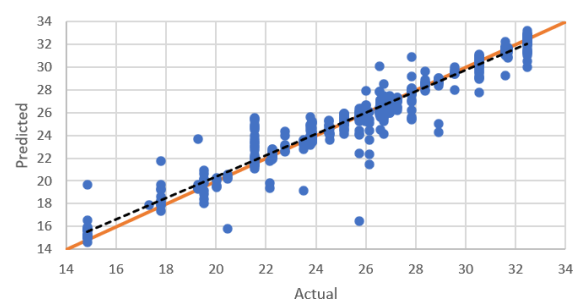
(o) V1SS Nitrogen



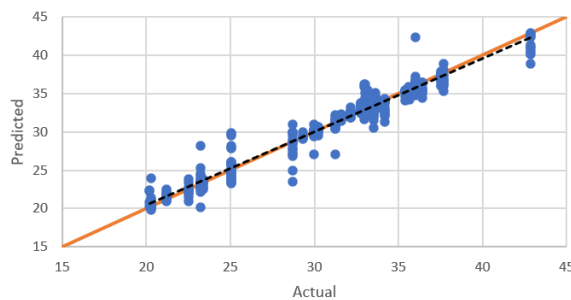
(p) V2SS Nitrogen



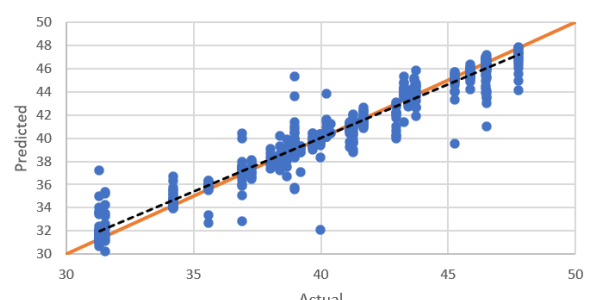
(q) V3SS Nitrogen



(r) V1SS Aromatic



s) V2SS Aromatic



t) V3SS Aromatic

Figure 4.3: Graphs of Predicted vs Actual using XGBoost.

4.2 Random Forest Results

4.2.1 Predictive Model Fit

Table 4.2 shows the coefficient of determination, R^2 value of the respective target variables for the Random Forest models. However, both V1SS Density and V2SS Density models show poor coefficient of determination, R^2 when the test data is passed through the developed model. V2SS Sulphur displays a good fit using the Random Forest algorithm but further tests with new test data is needed to verify this model.

Table 4.2: Coefficient of Determination for Respective Target Variables using Random Forest.

Target Variable	R^2
fv1ss	0.9907
fv2ss	0.9921
fv3ss	0.9559
y1	0.9825
y2	0.9722
y3	0.9483
V1SS_Density15	0.0001
V2SS_Density15	0.0009
V3SS_Density15	0.9915
V1SS_kv100	0.9940
V2SS_kv100	0.9761
V3SS_kv100	0.9872
V1SS_Sulphur	0.9773
V2SS_Sulphur	0.9589
V3SS_Sulphur	0.9703
V1SS_Nitrogen	0.9857
V2SS_Nitrogen	0.9908
V3SS_Nitrogen	0.9879

V1SS_Aromatic	0.9908
V2SS_Aromatic	0.9937
V3SS_Aromatic	0.9909

4.2.2 Predictive vs Actual Plots

When the models are developed using the Random Forest algorithm, the trends shown by the models are similar to that of XGBoost where all of the target variables except V1SS Density and V2SS Density show good performance. This can be attributed to the poor dataset that is obtained for the specific target variables. Figure 4.4 shows the graph of actual versus predicted of V1SS Yield using the Random Forest algorithm which shows a better fit and has a better R^2 value.

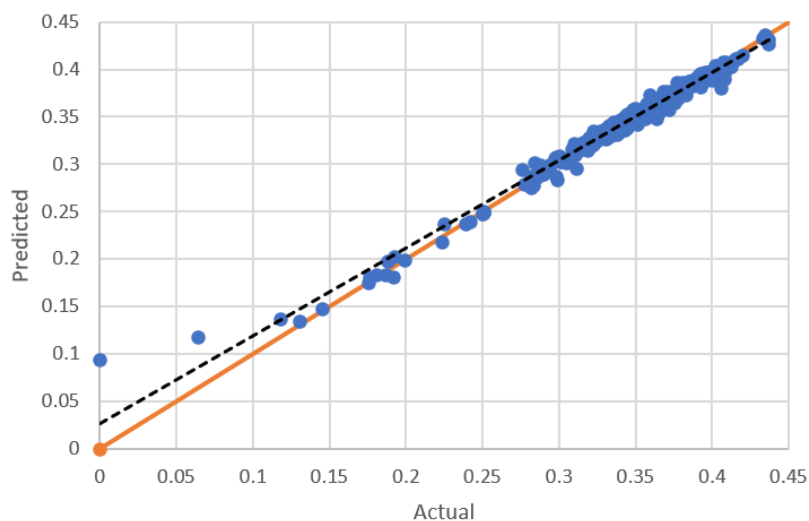
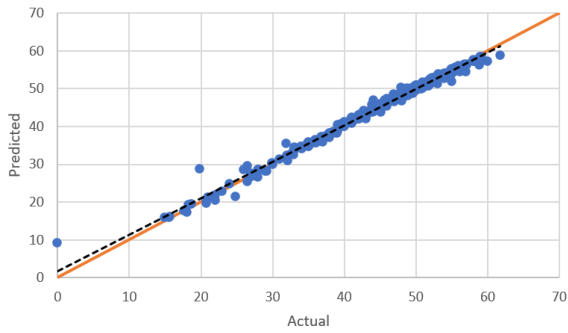
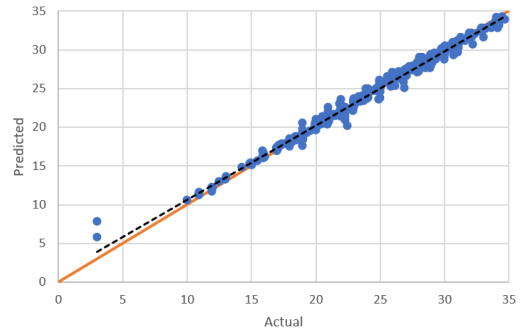


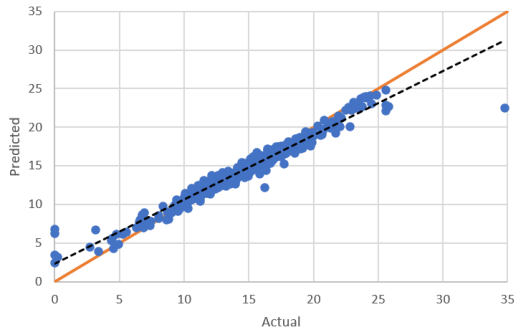
Figure 4.4: Graph of Predicted vs Actual of V1SS Yield using Random Forest.



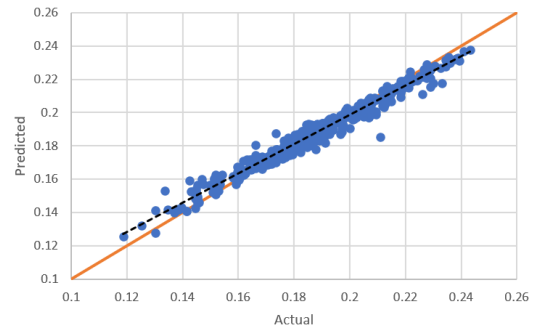
(a) V1SS Flowrate



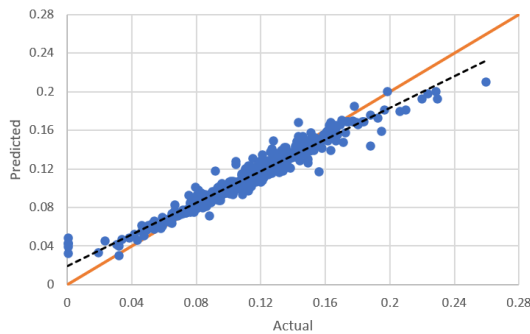
(b) V2SS Flowrate



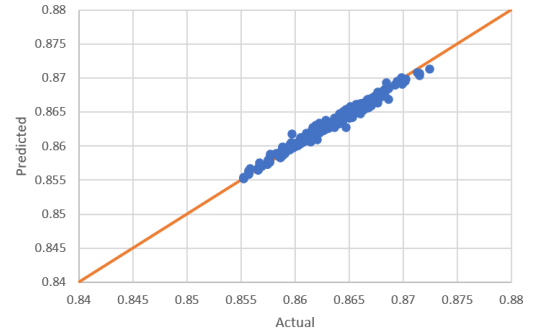
(c) V3SS Flowrate



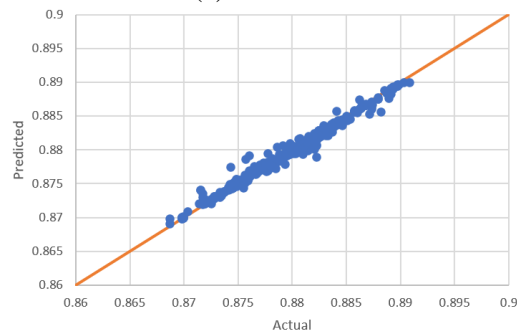
(d) V2SS Yield



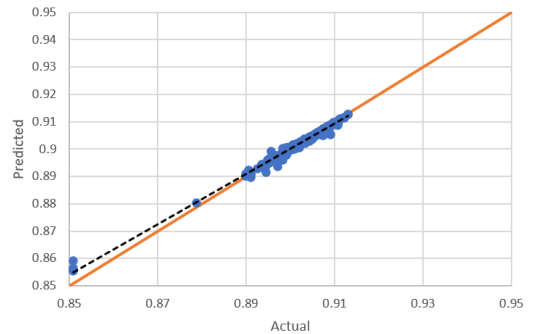
(e) V3SS Yield



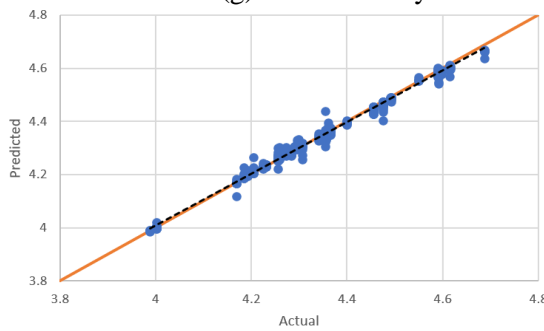
(f) V1SS Density



(g) V2SS Density

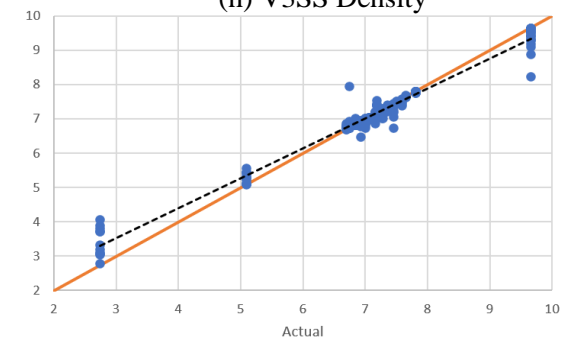


(h) V3SS Density

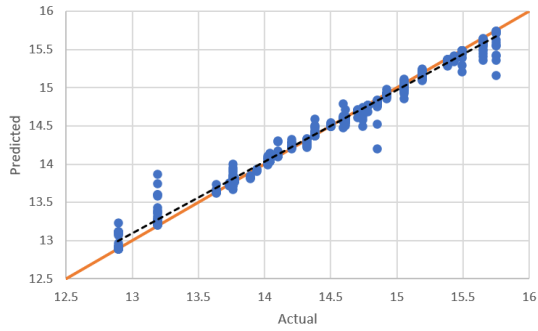


(i) V1SS kv100

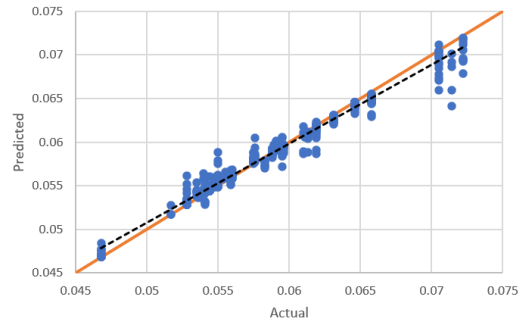
28



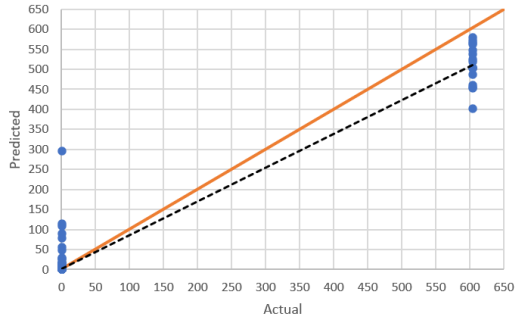
(j) V2SS kv100



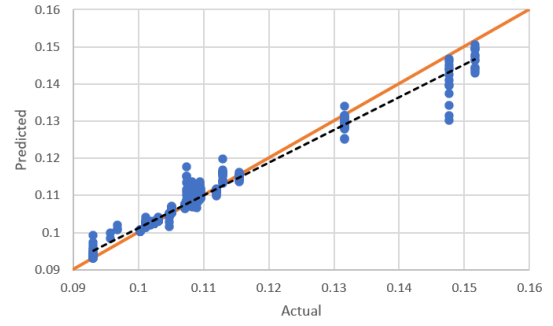
(k) V3SS kv100



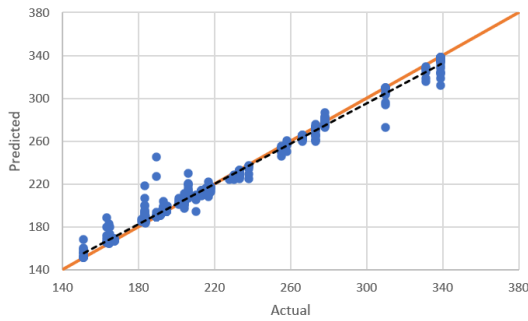
(l) V1SS Sulphur



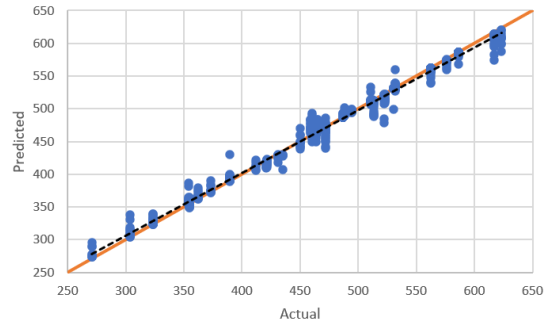
(m) V2SS Sulphur



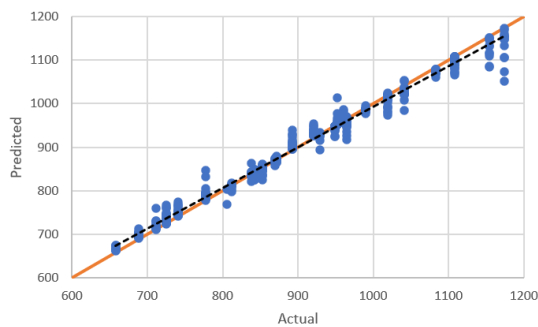
(n) V3SS Sulphur



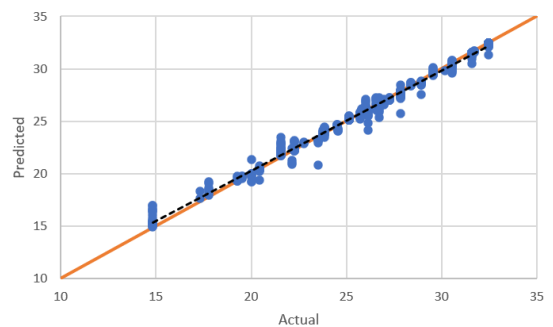
(o) V1SS Nitrogen



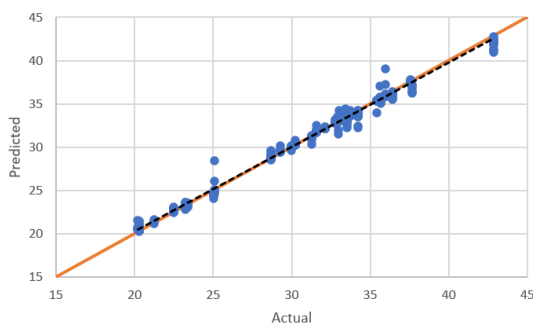
(p) V2SS Nitrogen



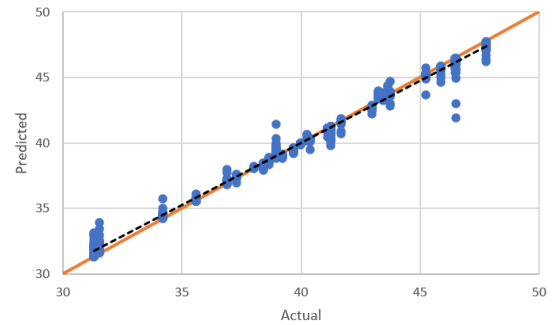
(q) V3SS Nitrogen



(r) V1SS Aromatic



(s) V2SS Aromatic



(t) V3SS Aromatic

Figure 4.5: Graphs of Predicted vs Actual using Random Forest.

4.3 Comparison between XGBoost and Random Forest

Based on Figure 4.6, it could be observed that the Random Forest algorithm is able to achieve better R^2 values due to its ability in randomly subsampling the data during training which prevents overfitting of the model towards the training data. Thus, Random Forest is a better option if it was to be used with minimal hyperparameter tuning. XGBoost has the advantage in terms of computing power and the overfitting issue could be corrected by introducing a lower sampling rate of columns and features to build each tree to avoid any overfitting. Referring to Figure 4.7, both algorithms achieved similar performance in predicting the kinematic viscosity at 100°C and the aromatic content of all three streams, V1SS, V2SS and V3SS as shown by charts (j), (k), (l), (s), (t) and (u).

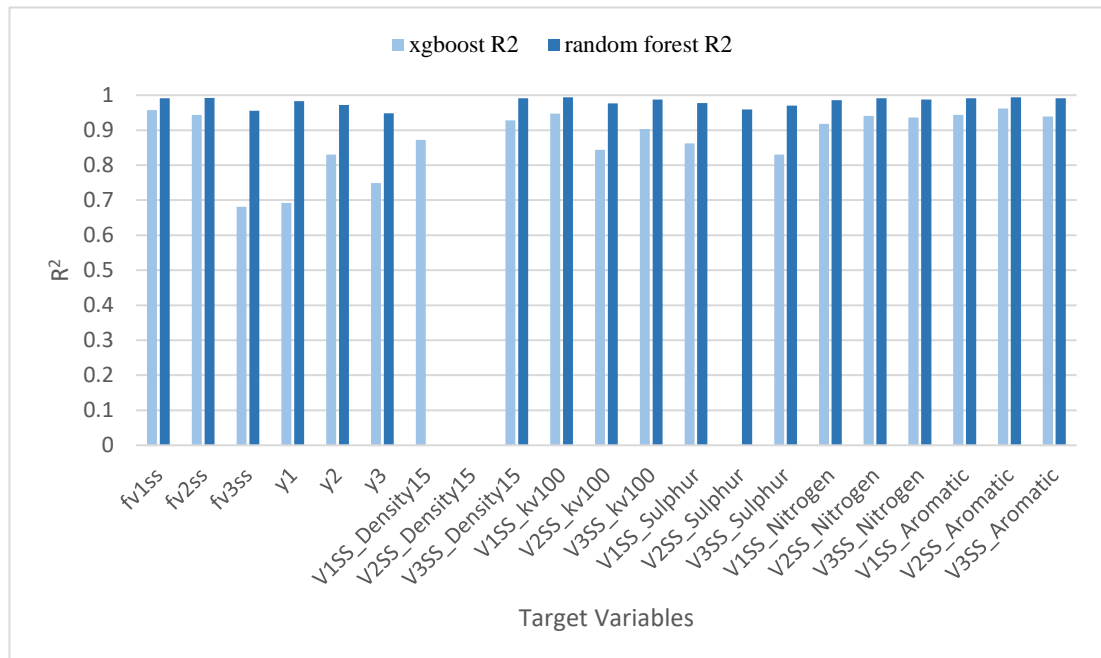
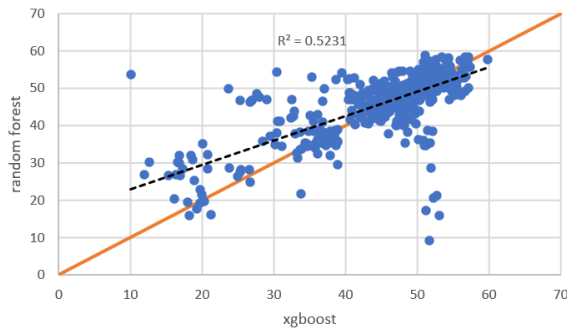
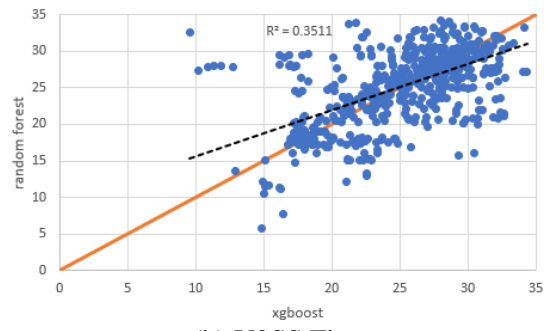


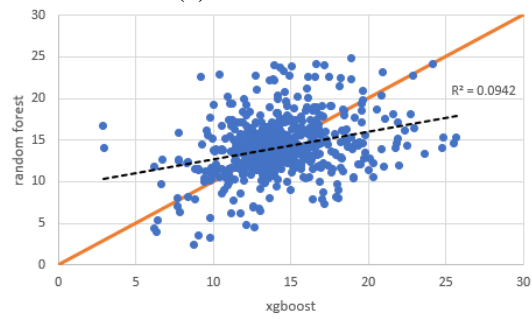
Figure 4.6: Chart of R^2 for XGBoost and Random Forest.



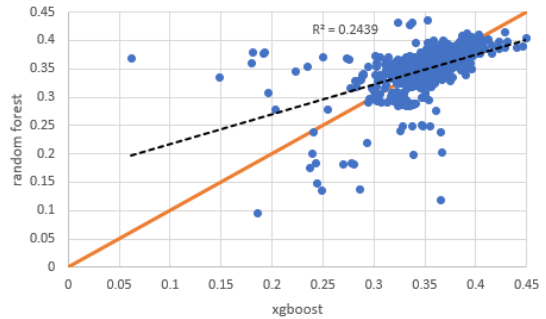
(a) V1SS Flowrate



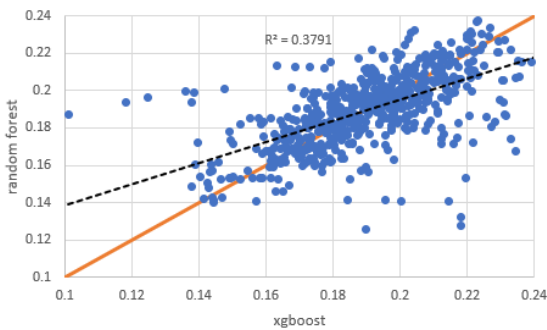
(b) V2SS Flowrate



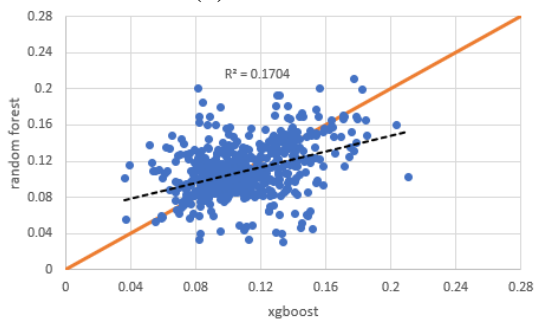
(c) V3SS Flowrate



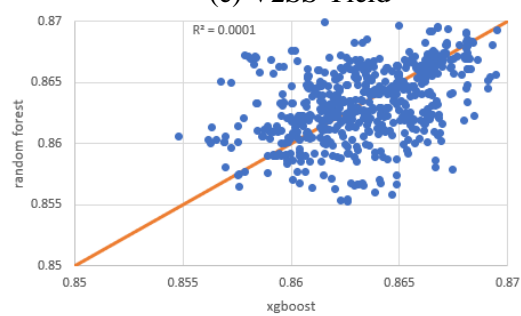
(d) V1SS Yield



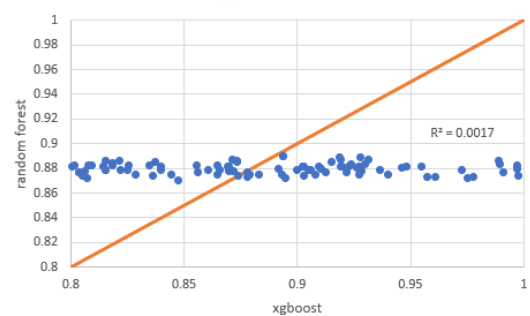
(e) V2SS Yield



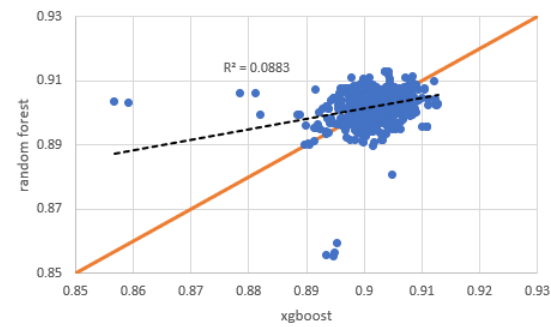
(f) V3SS Yield



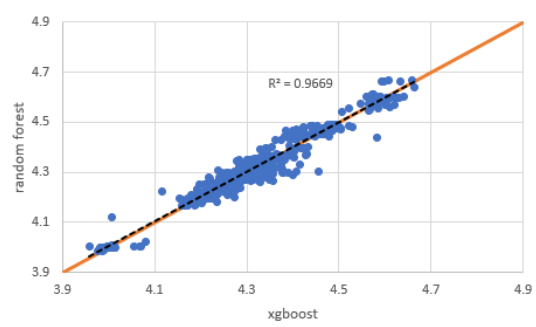
(g) V1SS Density



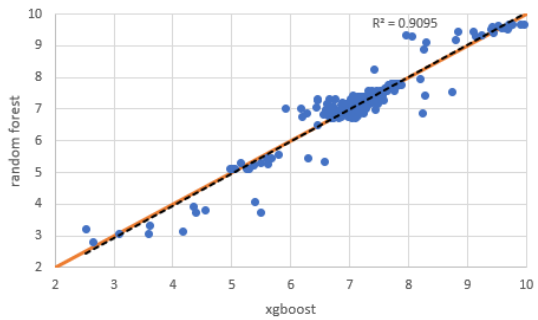
(h) V2SS Density



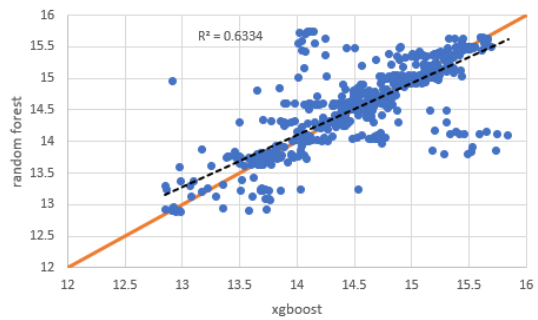
(i) V3SS Density



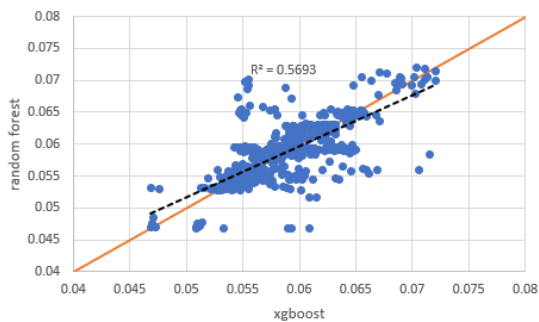
(j) V1SS kv100



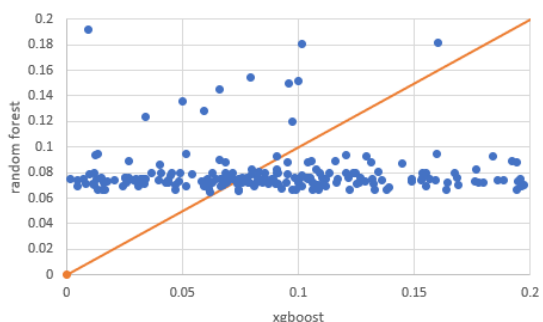
(k) V2SS kv100



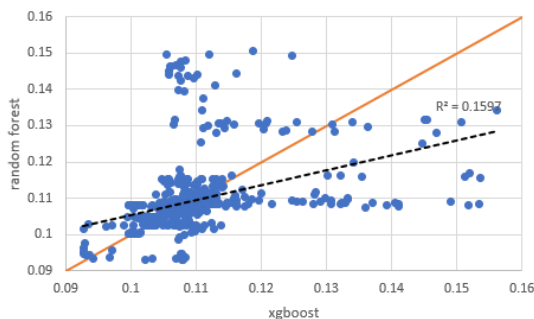
(l) V3SS kv100



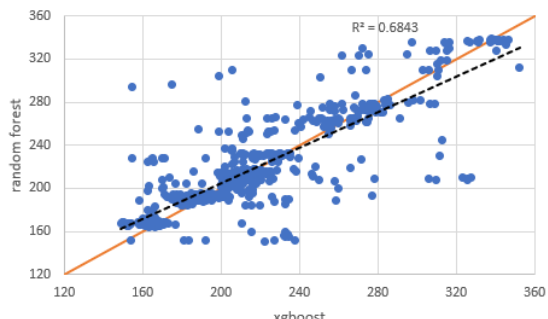
(m) V1SS Sulphur



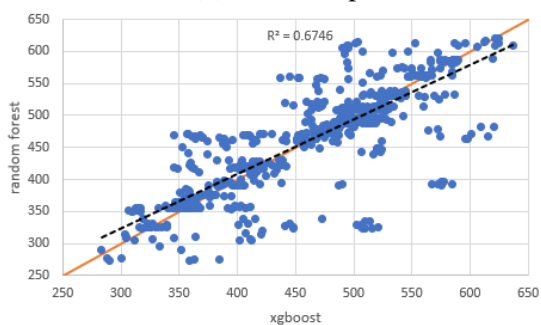
(n) V2SS Sulphur



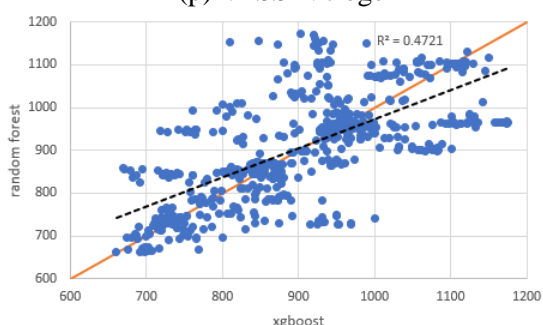
(o) V3SS Sulphur



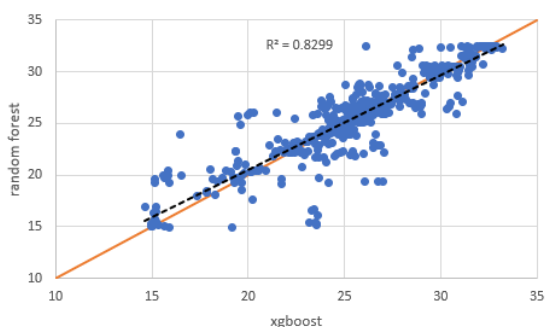
(p) V1SS Nitrogen



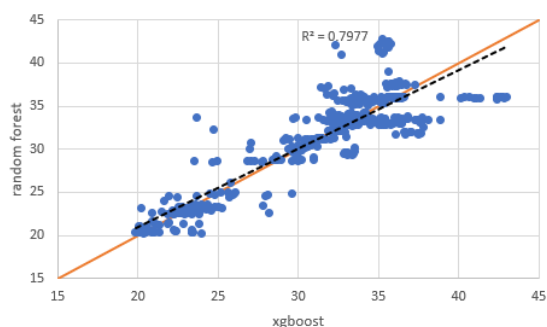
(q) V2SS Nitrogen



(r) V3SS Nitrogen



(s) V1SS Aromatic



(t) V2SS Aromatic

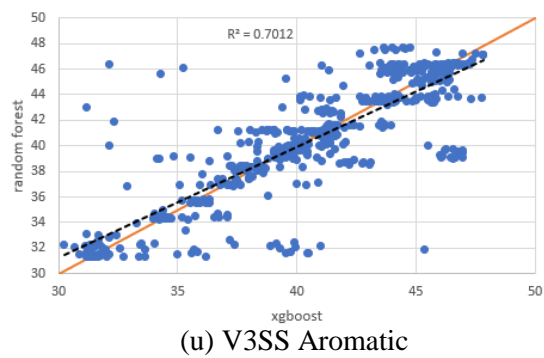


Figure 4.7: Graphs of Random Forest vs XGBoost

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project, which is based on the refinery in MRCSB is focused on developing predictive model for the VDU using the XGBoost algorithm. Through literatures, the superior performance of the XGBoost algorithm in terms of computation time and prediction accuracy over other algorithms such as Random Forest and Decision Tree algorithms is emphasised. In this report, the researcher has primarily worked on the first research objective, which is to develop and modify the source code to develop the predictive models. A detailed breakdown of the source code, major functions and packages used were also highlighted. The outputs of the developed model were explained, where the both the Random Forest and XGBoost models show good fit to 19 out of the 21 target variables. When the predicted results of XGBoost was compared against the predicted results from Random Forest, the Kinematic Viscosity@100°C, Nitrogen and Aromatic content values of streams V1SS, V2SS and V3SS closely approximate each other.

5.2 Recommendation

The performance of both the algorithm is not entirely conclusive since the test data is limited and is from the actual data itself. Thus, recent plant data is required to test the performance of these models and their accuracy in prediction as well as to analyse their reliability in predicting live data.

Another recommendation would be to explore the route of exhaustive hyperparameter tuning to obtain the best coefficient of determination. XGBoost is the appropriate algorithm to be used for this case since sufficient computing power is needed if such task was to be taken and Random Forest is not recommended because of its slow computing performance. Aside from that, the detailed understanding of the effects of manipulating each hyperparameter is one aspect to be considered for having a detailed analysis.

REFERENCES

- Bhalla, D. (n.d.). *A complete guide to random forest in R*. Retrieved from Listen Data: <https://www.listendata.com/2014/11/random-forest-with-r.html#What-is-Random-Forest->
- Boehmke, B., & Greenwell, B. (2020, February 1). *Random Forest*. Retrieved from Hands-on Machine Learning with R: <https://bradleyboehmke.github.io/HOML/random-forest.html>
- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 785-794). doi:10.1145/2939672.2939785
- Leventis, D. (2018, November 11). *XGBoost Mathematics Explained*. Retrieved from Towards Data Science: <https://towardsdatascience.com/xgboost-mathematics-explained-58262530904a>
- Morde, V., & Setty, V. A. (2019, April 8). *XGBoost Algorithm: Long May She Reign!* Retrieved from Towards Data Science: <https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d>
- Nikulski, J. (2016, March). *The ultimate guide to adaboost, random forests and xgboost*. Retrieved from Towards Data Science: <https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f>
- Saxena, R. (2017, January 30). *How decision tree algorithm works*. Retrieved from Dataaspirant: <https://dataaspirant.com/how-decision-tree-algorithm-works/#:~:text=Decision%20Tree%20Algorithm%20Pseudocode,same%20value%20for%20an%20attribute.>
- Song, Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2), 130-135. doi:10.11919/j.issn.1002-0829.215044

Wu, X. (2018, October 8). *Evaluating Machine Learning Models in R: Predicting Marine Debris*. Retrieved from Azavea: <https://www.azavea.com/blog/2018/10/08/marine-debris-machine-learning-models-r/>

XGBoost Parameters. (n.d.). Retrieved from <https://xgboost.readthedocs.io/en/latest/parameter.html#general-parameters>

APPENDICES

Appendix A: List of Abbreviations

List of Abbreviations Used

LSWR	Low Sulphur Waxy Residue	VR_5	VR_Simdist_5
LVGO	Light Vacuum Gas Oil	VR_10	VR_Simdist_10
MVGO	Medium Vacuum Gas Oil	VR_20	VR_Simdist_20
V1SS	Vacuum 1 Side Stream	VR_30	VR_Simdist_30
V2SS	Vacuum 2 Side Stream	VR_40	VR_Simdist_40
V3SS	Vacuum 3 Side Stream	VR_50	VR_Simdist_50
VR	Vacuum Residue	VR_60	VR_Simdist_60
f	Total Flowrate	VR_70	VR_Simdist_70
flvgo	LVGO Flowrate	VR_80	VR_Simdist_80
fmvgo	MVGO Flowrate	VR_90	VR_Simdist_90
fv1ss	V1SS Flowrate	VR_95	VR_Simdist_95
fv2ss	V2SS Flowrate		
fv3ss	V3SS Flowrate		
fvr	VR Flowrate		

List of Abbreviations Used

tovh	Overhead Draw Temperature	V1SS_Nitrogen	V1SS Nitrogen Content
tlvgo	LVGO Draw Temperature	V2SS_Nitrogen	V2SS Nitrogen Content
tmvgo	MVGO Draw Temperature	V3SS_Nitrogen	V3SS Nitrogen Content
tv1ss	V1SS Draw Temperature	V1SS_Sulphur	V1SS Sulphur Content
tv2ss	V2SS Draw Temperature	V2SS_Sulphur	V2SS Sulphur Content
tv3ss	V3SS Draw Temperature	V3SS_Sulphur	V3SS Sulphur Content
tvr	VR Draw Temperature	V3SS_MCRT	V3SS Micro Carbon Residue Tester
vp	Vapour Pressure	LVGO_Density15	LVGO Density@15°C
V1SS_kv100	V1SS Kinematic Viscosity @100°C	MVGO_Density15	MVGO Density@15°C
V2SS_kv100	V2SS Kinematic Viscosity @100°C	V1SS_Density15	V1SS Density@15°C
V3SS_kv100	V3SS Kinematic Viscosity @100°C	V2SS_Density15	V2SS Density@15°C
V1SS_Aromatic	V1SS Aromatic Content	V3SS_Density15	V3SS Density@15°C
V2SS_kv100	V2SS Aromatic Content	VRES_Density15	VRES Density@15°C
V3SS_kv100	V3SS Aromatic Content	QuenchBottom	Quench Bottom Flowrate

List of Abbreviations Used

LSWR_5	LSWR_Simdist_5	LVGO_5	LVGO_Simdist_5	MVGO_5	MVGO_Simdist_5
LSWR_10	LSWR_Simdist_10	LVGO_10	LVGO_Simdist_10	MVGO_10	MVGO_Simdist_10
LSWR_20	LSWR_Simdist_20	LVGO_20	LVGO_Simdist_20	MVGO_20	MVGO_Simdist_20
LSWR_30	LSWR_Simdist_30	LVGO_30	LVGO_Simdist_30	MVGO_30	MVGO_Simdist_30
LSWR_40	LSWR_Simdist_40	LVGO_40	LVGO_Simdist_40	MVGO_40	MVGO_Simdist_40
LSWR_50	LSWR_Simdist_50	LVGO_50	LVGO_Simdist_50	MVGO_50	MVGO_Simdist_50
LSWR_60	LSWR_Simdist_60	LVGO_60	LVGO_Simdist_60	MVGO_60	MVGO_Simdist_60
LSWR_70	LSWR_Simdist_70	LVGO_70	LVGO_Simdist_70	MVGO_70	MVGO_Simdist_70
LSWR_80	LSWR_Simdist_80	LVGO_80	LVGO_Simdist_80	MVGO_80	MVGO_Simdist_80
LSWR_90	LSWR_Simdist_90	LVGO_90	LVGO_Simdist_90	MVGO_90	MVGO_Simdist_90
LSWR_95	LSWR_Simdist_95	LVGO_95	LVGO_Simdist_95	MVGO_95	MVGO_Simdist_95
P1810_PA22	P1810 Pump Around	V1SS_PA68	V1SS Pump Around	V3SS_PA87	V3SS Pump Around
LVGO_PA34	LVGO Pump Around	V2SS_PA71	V2SS Pump Around		

List of Abbreviations Used

V1SS_5	V1SS_Simdist_5	V2SS_5	V2SS_Simdist_5	V3SS_5	V3SS_Simdist_5
V1SS_10	V1SS_Simdist_10	V2SS_10	V2SS_Simdist_10	V3SS_10	V3SS_Simdist_10
V1SS_20	V1SS_Simdist_20	V2SS_20	V2SS_Simdist_20	V3SS_20	V3SS_Simdist_20
V1SS_30	V1SS_Simdist_30	V2SS_30	V2SS_Simdist_30	V3SS_30	V3SS_Simdist_30
V1SS_40	V1SS_Simdist_40	V2SS_40	V2SS_Simdist_40	V3SS_40	V3SS_Simdist_40
V1SS_50	V1SS_Simdist_50	V2SS_50	V2SS_Simdist_50	V3SS_50	V3SS_Simdist_50
V1SS_60	V1SS_Simdist_60	V2SS_60	V2SS_Simdist_60	V3SS_60	V3SS_Simdist_60
V1SS_70	V1SS_Simdist_70	V2SS_70	V2SS_Simdist_70	V3SS_70	V3SS_Simdist_70
V1SS_80	V1SS_Simdist_80	V2SS_80	V2SS_Simdist_80	V3SS_80	V3SS_Simdist_80
V1SS_90	V1SS_Simdist_90	V2SS_90	V2SS_Simdist_90	V3SS_90	V3SS_Simdist_90
V1SS_95	V1SS_Simdist_95	V2SS_95	V2SS_Simdist_95	V3SS_95	V3SS_Simdist_95

APPENDIX B: Input Variables

Input Variables

LSWR_5	LSWR_Combined_5	tvr
LSWR_10	LSWR_Combined_10	tlvgo
LSWR_20	LSWR_Combined_20	tmvgo
LSWR_30	LSWR_Combined_30	tv1ss
LSWR_40	LSWR_Combined_40	tv2ss
LSWR_50	LSWR_Combined_Density	tv3ss
LSWR_60	LSWR_ColdFeed	QuenchBottom
LSWR_70	LSWR_HotFeed	P1810_PA22
LSWR_80	f	V3SS_PA87
LSWR_90	vp	V2SS_PA71
LSWR_95	tovh	V1SS_PA68
		LVGO_PA34

APPENDIX C: Output Variables

Output Variables

LVGO_5	MVGO_5	V1SS_5	V2SS_5	V3SS_5	VR_5
LVGO_10	MVGO_10	V1SS_10	V2SS_10	V3SS_10	VR_10
LVGO_20	MVGO_20	V1SS_20	V2SS_20	V3SS_20	VR_20
LVGO_30	MVGO_30	V1SS_30	V2SS_30	V3SS_30	VR_30
LVGO_40	MVGO_40	V1SS_40	V2SS_40	V3SS_40	VR_40
LVGO_50	MVGO_50	V1SS_50	V2SS_50	V3SS_50	VR_50
LVGO_60	MVGO_60	V1SS_60	V2SS_60	V3SS_60	VR_60
LVGO_70	MVGO_70	V1SS_70	V2SS_70	V3SS_70	VR_70
LVGO_80	MVGO_80	V1SS_80	V2SS_80	V3SS_80	VR_80
LVGO_90	MVGO_90	V1SS_90	V2SS_90	V3SS_90	VR_90
LVGO_95	MVGO_95	V1SS_95	V2SS_95	V3SS_95	VR_95
flvgo	fmvgo	fv1ss	fv2ss	fv3ss	fvr
LVGO_Density	MVGO_Density	V1SS_Density	V2SS_Density	V3SS_Density	VRES_Density
V1SS_kv100	V2SS_kv100	V3SS_kv100	V1SS_Aromatic	V2SS_Aromatic	V3SS_Aromatic
V1SS_Nitrogen	V2SS_Nitrogen	V3SS_Nitrogen	V1SS_Sulphur	V2SS_Sulphur	V3SS_Sulphur
V3SS_MCRT					

APPENDIX D: Gantt Chart (FYP I)

Task	1	2	3	4	5	6	7	8	9	10	11	12	13
Context setting and briefing													
Work on literature review regarding XGBoost													
Develop the source code to develop XGBoost models													
Develop predictive model for 94 target variables													
Report Preparation													
Developed model submission													
Report Submission and Presentation													

APPENDIX E: Gantt Chart (FYP II)

Task	1	2	3	4	5	6	7	8	9	10	11	12	13
Develop Random Forest model													
Trial run on 60/40 data split for training and test data													
Correlation studies													
Draft Dissertation Submission													
Develop predictive model for 21 target variables													
Report Submission and Presentation													

Milestones:

- 1 – Develop a source code that could generate predictive model for XGBoost and Random Forest
- 2 – Identify correlations between features wherever applicable
- 3 – Submit the developed models in a .zip file as well as the finalised report and other related documents.

APPENDIX F: Source Code (XGBoost)

```
#Prakash Saravanan

# Packages

library(tidyverse) # data manipulation

library(caret)

library(mlr) # ML package (also some data manipulation)

library(knitr) # just using this for kable() to make pretty tables

library(xgboost)

library(caTools)

require(fastmatch)

# Data preparation

#For version 2 of the data

# Importing the dataset

dataset = read.csv('C:\\Users\\User\\Desktop\\UTP_Intern\\Source_Code\\Dataset_Latest_v2_1.csv')

dataset$mode<-as.factor(dataset$mode)

#dataset<-na.omit(dataset)#Remove rows with missing data

dataset<-dataset[,-c(1,2)] #Used for current code to remove only sequence, date,

dataset = select(dataset,"V3SS_kv100",1:10,13:29,121:126) #Dataset for inputs only. Target variable to be added in the
beginning

str(dataset)

summary(dataset)

mydata1<-dataset

mydata1<-mydata1[,-c(34)]

# Splitting the dataset into the Training set and Test set

set.seed(12345)

split1=createDataPartition(mydata1$V3SS_kv100,p=0.7,list=FALSE)

train1 <- mydata1[split1,]

test1 <- mydata1[-split1,]

#fitting XGBoost to the training set
```

```

trainTask1 <- makeRegrTask(data = train1, target = "V3SS_kv100")
testTask1 <- makeRegrTask(data = test1, target = "V3SS_kv100")

# Timer
start_time = Sys.time()
Sys.sleep(0.5)

#set.seed(1)

# Create an xgboost learner that is classification based and outputs
# labels (as opposed to probabilities)
xgb_learner <- makeLearner(
  "regr.xgboost",
  predict.type = "response",
  par.vals = list(
    objective = "reg:squarederror",
    eval_metric = "error",
    nrounds = 200
  )
)

# Create a model
xgb_model1 <- train(xgb_learner, task = trainTask1)
result1 <- predict(xgb_model1, testTask1)

head(result1$data) %>%
  kable()

# XGBoost hyperparameter tuning
xgb_params <- makeParamSet(
  # The number of trees in the model (each one built sequentially)
  makeIntegerParam("nrounds", lower = 500, upper = 2000),
  # Number of splits in each tree
  makeIntegerParam("max_depth", lower = 1, upper = 6),
  # "shrinkage" - prevents overfitting
  makeNumericParam("eta", lower = .1, upper = .5),

```

```

# L2 regularization - prevents overfitting
makeNumericParam("lambda", lower = -1, upper = 0, trafo = function(x) 10^x)
)

control <- makeTuneControlRandom(maxit = 200)

# Create a description of the resampling plan
resample_desc <- makeResampleDesc("CV", iters = 10)

tuned_params1 <- tuneParams(
  learner = xgb_learner,
  task = trainTask1,
  resampling = resample_desc,
  par.set = xgb_params,
  control = control
)

#view importance
classifier1 = xgboost(data = as.matrix(train1[-1]),
  label = train1$V3SS_kv100,
  nrounds = 125000,
  max_depth = tuned_params1$x$max_depth,
  eta = tuned_params1$x$eta,
  lambda = tuned_params1$x$lambda,
  seed = 1,
  nfolds = 10,
  eval_metric = "rmse",
  objective = "reg:squarederror",
  early_stopping_rounds = 10
)

print(c(tuned_params1$x$max_depth,tuned_params1$x$eta,tuned_params1$x$lambda, classifier1$best_score))

importance_matrix1 <- xgb.importance(model = classifier1)

print(importance_matrix1)

xgb.plot.importance(importance_matrix = importance_matrix1,top_n = 20)

# save model to binary local file
xgb.save(classifier1 , "Project_Unsorted_V3SS_kv100.model")

```

```

end_time = Sys.time()
end_time - start_time

#Used for model retrieval and residual plot
#classifier1<-xgb.load("Project_MG3_QuenchBottom_103.model")#Load developed model

num<-1
num
testlabel1<-as.matrix(test1[,num])
test1<-test1[,~num]

y_pred1 <- predict(classifier1, as.matrix(test1))
y_pred1 = as.data.frame(y_pred1)
df_VDU1 = data.frame( Actual = testlabel1 , Predicted = y_pred1$y_pred1)
write.csv(df_VDU1, 'Project_Unsorted_V3SS_kv100.csv')

summary(y_pred1)
mean(testlabel1)
mean(y_pred1$y_pred1)
y_pred1<-as.integer(y_pred1>mean(y_pred1$y_pred1))
testlabel1<-as.integer(testlabel1>mean(testlabel1))
confusionMatrix(as.factor(y_pred1),as.factor(testlabel1))

```

APPENDIX G: Source Code (Random Forest)

```
#Prakash Saravanan
#28/8/2020

# Packages

library(tidyverse) # data manipulation
library(caret)
library(mlr) # ML package (also some data manipulation)
library(knitr) # just using this for kable() to make pretty tables
library(parallelMap)
library(parallel)
library(caTools)
library(ranger)
library(tuneRanger)
require(fastmatch)
library(randomForest)

#Parallel computing
no_cores <- detectCores()-0

.onLoad = function(libname, pkgname) {
  configureMlr()
  backports::import(pkgname)
}

.onAttach = function(libname, pkgname) {
  configureMlr()
  parallelRegisterLevels(package = "mlr", levels = c("benchmark", "resample", "selectFeatures", "tuneParams", "ensemble"))
}

parallelStartSocket( cpu = no_cores , level = "mlr.resample")

#For version 2 of the data
#Check the working directory to save files
getwd()
setwd("C:\\Users\\User\\Desktop\\UTP_Intern\\Output_Dataset_U18_V3_ZAP_MAF_V3")
```

```

# Importing the dataset
dataset = read.csv('C:\\Users\\User\\Desktop\\UTP_Intern\\Source_Code\\Dataset_U18_V3_ZAP_MAF_V3_1.csv')
dataset$mode<-as.factor(dataset$mode)
#dataset<-na.omit(dataset)#Remove rows with missing data
dataset<-dataset[,-c(1,2)] #Used for current code to remove only sequence, date, Nitrogen and Sulphur
dataset1 = select(dataset,"V3SS_Sulphur",1:10,13:29,121:126) #Dataset for inputs only. Target variable to be added in the
beginning

set.seed(12345)

split1=createDataPartition(dataset1$V3SS_Sulphur,p=0.7,list=FALSE)
train1 <- dataset1[split1,]
test1 <- dataset1[-split1,]

trainTask1 <- makeRegrTask(data = train1, target = "V3SS_Sulphur")
testTask1 <- makeRegrTask(data = test1, target = "V3SS_Sulphur")

rf_learner <- makeLearner(
  "regr.randomForest",
  predict.type = "response",
  par.vals = list(
    ntree=2000,
    importance=TRUE,
    proximity=TRUE
  )
)

rf_model1 <- train(rf_learner, task = trainTask1)

result1 <- predict(rf_model1, testTask1)

getParamSet(rf_learner)

rf_params <- makeParamSet(
  # The number of columns sampled in each tree
  makeIntegerParam("mtry", lower = 2, upper = 10),

```

```

# Observations in terminal nodes

makeIntegerParam("nodesize", lower = 10, upper = 50)#,
# Number of trees
#makeNumericParam("ntree", lower = 500, upper = 2000)
)

control <- makeTuneControlRandom(maxit = 200)

resample_desc <- makeResampleDesc("CV", iters = 10L)

tuned_params1 <- tuneParams(
  learner = rf_learner,
  task = trainTask1,
  resampling = resample_desc,
  par.set = rf_params,
  control = control
)

rf1 <- randomForest(
  dataset1$V3SS_Sulphur ~ .,
  data=as.matrix(dataset1),
  ntree=2000,
  mtry=tuned_params1$x$mtry,
  nodesize=tuned_params1$x$nodesize,
  importance=TRUE,
  proximity=TRUE
)

plot(importance(rf3))
varImpPlot(rf3)
print(rf3)
MDSplot(rf1,dataset1$y1)
proximity.plot(
  rf2,
  dim.x = 1,
  dim.y = 2,

```



```

legend.loc = c("top", "bottom", "left", "right"),
point.size = 2,
circle.size = 8,
circle.border = 1,
hull.alpha = 0.3,
plot = TRUE
)

pred1 <- predict(rf1, as.matrix(test1))

num <- 1
num
testlabel1 <- as.matrix(test1[, num])
test1 <- test1[, -num]
actual1 <- testlabel1
predicted1 <- pred1
R2_1 <- 1 - (sum((actual1 - predicted1)^2) / sum((actual1 - mean(actual1))^2))
df_VDU1 = data.frame( Actual = testlabel1 , Predicted = pred1, R2_1)
write.csv(df_VDU1, 'RandomForest_V3SS_Sulphur.csv')

```