

Estimating Process Simulation Errors of an Industrial Fractionator Column by Using Data-Driven Modelling Techniques

By
Sadık Ödemiş

**A Thesis Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of**

**Master of Science
In
Industrial Engineering**

**Koc University
January 2017**

Koc University
Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Sadık Ödemiş

And have found that it is complete and satisfactory in all respects,
And that any and all revisions required by the final
examining committee have been made.

Committee Members:

Assoc.Prof. Özden Gür Ali (Advisor)

Assist. Prof. Mehmet Gönen

Assoc. Prof. Burak Alakent

Date:

ABSTRACT

Safe and efficient operation of refinery units remains the prime concern for refineries. The daily growth of fuel consumption and the necessity of proving it show that operating of a refinery unit in an economical way is a vital case. Unfortunately, the downstream refining industry faces many technical and operation difficulties due to the introduction or increase of environmental restrictions that lead to tighter product specification that require investing in new refinery technologies, improved process operations, as well as adaptation to the changes in demands.

For most of the refinery units it is usually possible to improve the operating conditions with small touches and these tiny operational improvements often mean a huge economic benefit and are worth the effort. Operational improvements depend on accurate estimates of process behaviours, which can be achieved both by using process simulators, which embody first-principle models of the processes, and/or by data-driven models derived with operation data. Due to the inevitable deviation of theoretical models from the actual behavior of the process, many process engineers prefer data-driven. On the other hand; compared to data-driven models, it is possible to extract much more information about the process by means of a process simulator.

In this thesis, we propose and test a novel approach that corrects process simulator outputs for product qualities of an industrial hydrocracker fractionator column located in Turkish Petroleum Refinery Co. with data-driven models. The data-driven models use Support Vector Regression, Artificial Neural Networks, Random Forest, and linear regression model. With such an approach, process engineers can both correct some crucial actual outputs (product qualities) of process simulator with an acceptable error and at the same time they can extract more information related to the process, which is not possible to obtain with only data-driven models. We also introduce a method for feed characterization with operational data, which is a critical input of the process simulators. This enables using the process simulator for any time. Finally, we show six different approaches for simulation error modelling and evaluate their results by considering the measurement uncertainties.

ÖZETÇE

Petrol rafinerileri için rafineri ünitelerinin güvenli ve verimli şekilde işletilmesi en temel unsur olmaya devam etmektedir. Günlük yakıt tüketimindeki artış ve bu tüketimin karşılanması ihtiyacının gerekliliği rafineri ünitelerinin ekonomik şekilde işletilmesinin son derece önemli olduğunu göstermektedir. Ne yazık ki, aşağı akım (downstream) rafineri endüstrisi, yeni rafineri teknolojilerine yatırım gerektiren, iyileştirilmiş proses operasyonlarını ve taleplerdeki değişikliklere adapte olmayı gerektiren sıkı ürün spesifikasyonlarının getirilmesinden dolayı bir çok teknik ve operasyonel zorluklar ile karşı karşıya kalmaktadır.

Bir çok rafineri ünitesi için küçük dokunuşlarla çalışma koşullarının iyileştirilmesi genellikle mümkündür ve bu küçük iyileşmeler genellikle büyük bir ekonomik fayda anlamına gelir ve çaba harcamaya değerdir. Bu operasyonel iyileştirmeler hem süreçlerin ilk ilkelerini (teorik ilkelerini) modelleyen proses simülatörlerinden hem de veri tabanlı modeller yoluyla prosesin davranışını tahmin etmek için kullanılan proses verilerinden yararlanılarak elde edilebilir. Teorik modellerin, prosesin gerçek davranışından kaçınılmaz olarak sapması nedeniyle bir çok proses mühendisi proses koşullarını iyileştirmek için veri bazlı modelleri tercih etmektedir. Öte yandan, veri bazlı modellere kıyasla, proses simülatörleri ile proses hakkında daha fazla bilgi edinmek mümkündür.

Bu tez çalışmasında, Türkiye Petrol Rafinerileri A.Ş.'de yer alan endüstriyel hidrokraker fraksiyonlandırma kolonunun ürün kalitesine ilişkin proses simülatör çıktılarını karar destek regresyonu, yapay sinir ağları, rasgele orman ve doğrusal regresyon gibi veri bazlı modeller ile düzeltecek yeni bir yaklaşımı öneriyor ve test ediyoruz. Böyle bir yaklaşım ile birlikte proses mühendisleri hem proses simülatörünün (ürün özellikleri gibi) bazı kritik kritik çıktılarını kabul edilebilir bir hata ile düzeltmekte ve aynı zamanda yalnızca data bazlı modeller ile elde edilemeyecek proses hakkındaki ilave bilgilere ulaşabilmektedir. Ayrıca, operasyonel veri ile şarj karakterizasyonu yapmaya olanak sunan bir metot sunarak, şarjın destilasyon verilerine ihtiyaç duyan proses simülatörlerinin herhangi bir zamanda çalıştırılmasının yöntemini sunuyoruz. Son olarak, simülatör hatalarının modellenmesi için 6 farklı yaklaşım sunarak, bu yaklaşımların sonuçlarını ölçüm belirsizlerini dikkate alarak değerlendiriyoruz.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor *Assoc. Prof. Özden Gür Ali* for the continuous guidance of my MSc. Study and research, encouraging support and patience and motivation during the challenging thesis period concurrent with my role as a process control supervisor in Production Department of Turkish Petroleum Refineries Co.

I would like to extend my grateful thanks to *Dr. Ahmet Murat Yıldırım and Görkem Oğur*, who made possible for me to begin Master of Science degree in Industrial Engineering at Koç University while working in Tüpraş.

I am grateful that in the beginning of my MSc. Studies, I started working as a process modelling engineer in 2013 and carry out many studies via the process simulators with *Dr. Mustafa Karakaya* who was my Modelling & Optimization Superintendent in R&D department of Tüpraş.

Last but not least, I would like to thank my wife *Derya Ödemiş* for her understanding and love. It would be much harder to finish this work without my wife's support. My mother, *Dilek*, my sister *Sinem* receive my deepest gratitude and love for their encouragement.

TABLE OF CONTENTS

LIST OF TABLES	VIII
LIST OF FIGURES	X
CHAPTER 1: INTRODUCTION	1
1.1. MOTIVATION	1
1.2. LITERATURE REVIEW ON DATA-BASED MODEL APPLICATIONS IN REFINERIES.....	3
1.3. LITERATURE REVIEW ON FIRST-PRINCIPLE MODEL APPLICATIONS IN REFINERIES	5
1.4. COMPARISON OF DATA-BASED MODELLING AND FIRST-PRINCIPLE MODELLING APPROACHES IN REFINERIES	7
1.5. CONTRIBUTIONS.....	8
1.6. OVERVIEW OF THESIS	11
CHAPTER 2: FRACTIONATOR COLUMN	12
2.1. FRACTIONATION PROCESS DESCRIPTION.....	12
2.2. PRODUCT QUALITIES	14
2.2.1. <i>True Boiling Point (TBP) Curve</i>	14
2.2.2. <i>ASTM Distillation</i>	14
2.2.3. <i>Flash Point</i>	15
2.2.4. <i>API Gravity</i>	15
2.3. PRODUCT SPECIFICATIONS FOR FRACTIONATOR COLUMN	16
CHAPTER 3: PROCESS SIMULATION OF FRACTIONATOR COLUMN	17
3.1. INTRODUCTION	17
3.2. PROCESS DESCRIPTION OF HYSYS SIMULATION.....	17
3.3. CHARACTERIZATION OF THE FEED	21
3.3.1. <i>Back-blending</i>	22
3.3.2. <i>Pseudo-component Generation</i>	25
3.4. SIMULATION INPUTS & OUTPUTS	28
3.4.1. <i>Inputs</i>	28
3.4.2. <i>Outputs</i>	29
CHAPTER 4: DATA-DRIVEN MODELLING TECHNIQUES USED IN THE THESIS	30
4.1. SUPPORT VECTOR MACHINES	30
4.1.1. <i>The Support Vector Classifier</i>	31
4.1.1.1. Separable Case	31
4.1.1.2. Non-Separable Case	35
4.1.2. <i>Support Vector Machines and Kernels</i>	37
4.1.2.1. Moving into Non-linearity	37
4.1.2.2. Kernel Functions	38
4.1.3. <i>Support Vector Regression</i>	40
4.1.3.1. Regression Problem Statement and Some Basic Principles.....	41
4.1.4. <i>Nonlinear Regression with Kernel Trick</i>	45
4.2. ARTIFICIAL NEURAL NETWORKS.....	47
4.2.1. <i>The Artificial Neuron Network Models Fundamentals</i>	47
4.2.2. <i>Types of Activation Functions</i>	50
4.2.3. <i>Training the Neural Net</i>	52
4.3. RANDOM FORESTS MODELS	54
4.3.1. <i>Decision Tree Basics</i>	54

4.3.2.	<i>Building a Regression Tree</i>	55
4.3.3.	<i>From Bagging to Random Forests</i>	57
4.3.4.	<i>Variable Scoring with Random Forest Technique</i>	58
CHAPTER 5: THE MODELLING APPROACH		60
5.1.	INTRODUCTION TO MODELLING APPROACH	60
5.2.	PROCESS SIMULATOR OUTPUTS	64
CHAPTER 6: FEED CHARACTERIZATION MODEL BUILDING AND THE MODEL RESULTS		69
6.1.	REVIEW OF THE APPROACH FOR FEED CHARACTERIZATION	69
6.2.	FEED CHARACTERIZATION MODEL BUILDING	69
6.2.1.	<i>Input Variable Selection</i>	69
6.2.2.	<i>Input Data</i>	74
6.2.3.	<i>Parameter Search</i>	76
6.3.	MODEL RESULTS	80
CHAPTER 7: SIMULATOR OUTPUT CALIBRATOR MODEL BUILDING AND THE MODEL RESULTS		88
7.1.	INPUT VARIABLE SELECTION FOR SIMULATION ERROR CALIBRATOR MODELS	88
7.2.	APPROACHES FOR THE PURPOSE OF MODELLING THE SIMULATION ERROR	92
7.2.1.	<i>Results for Approach I : Modelling Simulation Errors with Residuals Variables</i>	94
7.2.2.	<i>Results for Approach II : Modelling Simulation Errors with Simulation Output</i>	96
7.2.3.	<i>Results for Approach III : Modelling Simulation Errors with Simulation Error with 1 lag and Simulation Output</i>	98
7.2.4.	<i>Results for Approach IV : Modelling Simulation Errors with Operational Variables</i>	101
7.2.5.	<i>Results for Approach V : Modelling Simulation Errors with All Variables</i>	102
7.2.6.	<i>Results for Approach VI : Modelling Simulation Errors with Selected Variables</i>	103
7.3.	FINAL MODEL SELECTION AND PERFORMANCE EVALUATIONS	105
7.3.1.	<i>Sample Final Model Selection (Diesel %5 Boiling Point)</i>	105
7.3.2.	<i>Model Accuracy Assessment</i>	117
CHAPTER 8: DISCUSSION OF MODEL RESULTS		119
8.1.	DISCUSSION OF PROCESS SIMULATOR OUTPUTS	119
8.2.	DISCUSSION OF FEED CHARACTERIZATION MODEL RESULTS	120
8.3.	DISCUSSION OF SIMULATION ERROR MODEL RESULTS	124
8.4.	DISCUSSION ON MODEL SELECTION	126
CHAPTER 9: CONCLUSION		127
APPENDIX I – SIMULATION OUTPUTS		129
APPENDIX II – MODELLING CODES IN R		136
BIBLIOGRAPHY		154
VITA		158

LIST OF TABLES

TABLE 1.1 - SURVEY OF SOFT SENSOR APPLICATIONS REPORTED IN PETROLEUM REFINERY.....	5
TABLE 1.2 - COMMERCIAL PROCESS SIMULATION SOFTWARE LIST	6
TABLE 1.3 - SURVEY OF PROCESS SIMULATOR APPLICATIONS IN OIL REFINERY	7
TABLE 1.4 - COMPARISON BETWEEN SIMULATION MODELS AND DATA-BASED MODELS (Z.KÍSTER, 2002)	8
TABLE 1.5 – LITERATURE SURVEY OF THE WORKS CARRIED OUT TO IMPROVE OPERATION CONDITIONS	10
TABLE 2.1 – PRODUCT SPECIFICATIONS FOR FRACTIONATOR COLUMN	16
TABLE 3.1 – SUGGESTED NUMBER OF PSEUDO-COMPONENTS	26
TABLE 3.2 – OPERATIONAL INPUTS FOR SIMULATION.....	28
TABLE 3.3 – FEED CHARACTERIZATION INPUTS.....	29
TABLE 3.4 – ALL OF THE POSSIBLE SIMULATOR OUTPUTS.....	29
TABLE 5.1 – PROCESS SIMULATOR OUTPUTS.....	64
TABLE 5.2 – RESPONSE VARIABLES	65
TABLE 5.3 – GAGE R&R VALUES FOR PRODUCT QUALITIES CONSIDERED IN THE THESIS	66
TABLE 6.1 – FEED CHARACTERIZATION VARIABLES	70
TABLE 6.2 – SVM MODEL PARAMETERS.....	76
TABLE 6.3 – GRID SEARCH REGION FOR SVR PARAMETERS	78
TABLE 6.4 – INFORMATION FOR CROSS-VALIDATION CONFIGURATION	79
TABLE 6.5 – DEFINED PARAMETERS FOR ARTIFICIAL NEURAL NETWORK TRAINING	79
TABLE 6.6 - 10-FOLD CROSS VALIDATION ERROR (RMSE) FOR DIFFERENT MODELS.....	80
TABLE 6.7 – FEED TBP CURVE COMPARISON (ACTUAL VS PREDICTED) – TRAINING SAMPLE.....	81
TABLE 6.8 – FEED TBP CURVE COMPARISON (ACTUAL VS PREDICTED) – TEST SAMPLE.....	82
TABLE 6.9 – MODEL SUMMARIES FOR FEED TBP MODELLING	83
TABLE 6.10 – FINAL MODELS FOR FEED CHARACTERIZATION MODELS (SVR WITH RBF KERNELS).....	86
TABLE 6.11 – A-FACTORS FOR EACH PRODUCT QUALITY	86
TABLE 7.1 – RESPONSE VARIABLES	88
TABLE 7.2 – POSSIBLE INPUT VARIABLES FOR SIMULATION ERROR MODELS	91
TABLE 7.3 – RESULT SUMMARIES FOR APPROACH I.....	94
TABLE 7.4 – RESULT SUMMARIES FOR APPROACH II	96
TABLE 7.5 – RESULT SUMMARIES FOR APPROACH III	98
TABLE 7.6 – RESULT SUMMARIES FOR APPROACH IV	101
TABLE 7.7 – RESULT SUMMARIES FOR APPROACH V.....	102

TABLE 7.8 – RESULT SUMMARIES FOR APPROACH VI.....	104
TABLE 7.9 – MODEL RESULTS FOR DIESEL 5% BOILING POINTS FOR ALL APPROACHES	105
TABLE 7.10 – SUMMARY FOR FINAL DIESEL %5 SIMULATION ERROR MODEL.....	106
TABLE 7.11 – SUMMARY OF FINAL MODELS FOR PREDICTION SIMULATION ERRORS	107
TABLE 7.12 – REPEATABILITY AND REPRODUCIBILITY IN MEASUREMENTS AND B-FACTORS.....	118
TABLE 8.1 – CORRELATION MATRIX FOR FEED CHARACTERIZATION MODEL INPUTS	122
TABLE 10.1 – SIMULATION OUTPUTS	129



LIST OF FIGURES

FIGURE 1.1 - THE PROGRESSION OF EUROPEAN EMISSION STANDARDS	1
FIGURE 1.2 - PETROLEUM PRODUCT CONSUMPTIONS FROM 1950 TO 2011 IN UK	2
FIGURE 1.3 - OVERVIEW OF DATA-DRIVEN SOFT SENSORS IN THE PROCESS INDUSTRY	4
FIGURE 2.1 - HYDROCRACKER SIMPLE PROCESS FLOW DIAGRAM	13
FIGURE 2.2 – TBP CURVE FOR DIESEL PRODUCT	14
FIGURE 3.1 – MAIN CASE VIEW OF THE SIMULATION.....	18
FIGURE 3.2 – DETAILED VIEW OF THE SIMULATION (COLUMN ENVIRONMENT)	19
FIGURE 3.3 – TOP REFLUX AND PUMP AROUNDS (REPRESENTATIVE DRAWING)	20
FIGURE 3.4 – BACK-MIXING (BACK-BLENDING) METHOD	22
FIGURE 3.5 – DISTILLATION INFORMATION OF THE PRODUCTS.....	23
FIGURE 3.6 – API GRAVITY (@60°F) TRENDS FOR PRODUCTS (2016) –	24
FIGURE 3.7 – PRODUCT YIELDS (2016).....	24
FIGURE 3.8 – RELATIONSHIP BETWEEN PSEUDO-COMPONENT PROPERTIES AND TBP CURV	26
FIGURE 4.1. SAMPLE TRAINING DATA FOR BINARY CLASSIFICATION.	31
FIGURE 4.2 - A HYPERPLANE SEPARATING BINARY CLASSES WITH THE MAXIMUM MARGIN.	32
FIGURE 4.3 - SOFT MARGIN SUPPORT VECTOR CLASSIFIER	36
FIGURE 4.4 - LOSS FUNCTIONS FOR REGRESSION PURPOSE	40
FIGURE 4.5 - ϵ -INSENSITIVE LOSS FUNCTION	42
FIGURE 4.6 - QUADRATIC LOSS FUNCTION	44
FIGURE 4.7. BIOLOGICAL NEURON	48
FIGURE 4.8 - A SIMPLE NN STRUCTURE.....	49
FIGURE 4.9 - A NEURAL NETWORK STRUCTURE	49
FIGURE 4.10 - THRESHOLD ACTIVATION FUNCTION OF A PERCEPTRON	50
FIGURE 4.11. PIECEWISE LINEAR FUNCTION	51
FIGURE 4.12 – SAMPLE REGRESSION TREE AND ITS CORRESPONDING REGION PARTITION	54
FIGURE 5.1 – FEED OF THE FRACTIONATOR COLUMN IS THE PRODUCTS OF THE REACTORS.....	61
FIGURE 5.2 – ESTIMATION OF PRODUCT QUALITIES WITH THE SUMMATION OF ESTIMATED SIMULATION ERRORS AND SIMULATOR OUTPUTS.....	62
FIGURE 5.3 - MODELLING APPROACH SCHEME	63
FIGURE 5.4 – ACTUAL AND CALCULATED (BY PROCESS SIMULATOR) PRODUCT QUALITIES.....	67
FIGURE 6.1 – HYDROCRACKER UNIT SIMPLIFIED PROCESS FLOW DIAGRAM	71
FIGURE 6.2 – EFFECT OF HVGO RATE AND WART OF THE HYDROCRACKER REACTORS ON HSRN YIELD	72

FIGURE 6.3 – EFFECT OF HVGO RATE AND WART OF THE HYDROCRACKER REACTORS ON DIESEL YIELD	72
FIGURE 6.4 – EFFECT OF HVGO RATE AND WART OF THE HYDROCRACKER REACTORS ON UCO YIELD	73
FIGURE 6.5 – EFFECT OF HYDROGEN RATE ON THE PRODUCT YIELDS	74
FIGURE 6.6 – 10-FOLD CROSS VALIDATION ERROR (RMSE) FOR DIFFERENT MODELS.....	80
FIGURE 6.7 – SIMULATION OUTPUTS WITH ESTIMATED AND ACTUAL FEED PROPERTIES	87
FIGURE 7.1 – SIMULATION ERROR MODELLING APPROACHES	93
FIGURE 7.2 – VARIABLE RANKINGS OBTAINED FROM RANDOM FOREST RANKING	103
FIGURE 7.3 – ERROR HISTOGRAMS OF FINAL SIMULATION ERROR MODELS	107
FIGURE 7.4 – SIMULATION ERROR AND PRODUCT QUALITY PREDICTIONS	112
FIGURE 7.5 – B-FACTORS FOR SIMULATION ERROR MODELS	118

NOMENCLATURE

x_i	:	ith observation for input variable
y_i	:	ith observation for output variable
w	:	the normal vector to the hyperplane
H_i	:	ith hyperplane
M	:	Margin
α_i	:	Lagrange multipliers for ith constraint
L_p	:	Primal objective function
L_D	:	Dual objective function
ξ_i	:	Non-negative slack variable in SVM formulation
C	:	Cost parameter for tuning support vector model
$\mathcal{H}$	:	Euclidean space
$K(\cdot)$	:	Kernel Function
Φ	:	Mapping function
D	:	Dataset
$\mathcal{L}(\cdot)$	:	Loss function
ε	:	Epsilon tuning parameter for support vector model
λ	:	Regularization parameter
w	:	weight value of artificial neural network
θ	:	Bias value for neural node in ANN
$ T $	:	The number of terminal nodes in decision trees
T_{bi}	:	Boiling point of component I
K	:	Watson K-factor
SG	:	Specific Gravity
$\mathfrak{B}^t$	:	OOB samples for a tree t
$L(T_t(x_i), y_i)$	:	Prediction accuracy of t th tree for the i th observation
t_L	:	Resulting children nodes in the tree for left side
N_t	:	Number of observation reaching node t
$i(\cdot)$	:	Impurity measure
$R_{x_j x_{-j}}^2$	:	R ² from a regression of X _j onto all of the other predictors
Y_i	:	i th product property value
W_i	:	Simulator output for i th product property
R_i	:	Residual value (Simulator output – Laboratory Result) for i th product property

ABBREVIATIONS

ANN	Artificial Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
RBFNN	Radial Basis Function Neural Network
PCA	Principle Component Analysis
RON	Research Octane Number
PLS	Partial Least Square
GA	Genetic Algorithm
SQP	Sequential Quadratic Programming
FCC	Fluid Catalytic Cracking
BPANN	Back-propagation artificial neural networks
SRM	Structural Risk Minimization
ERM	Empirical Risk Minimization
RKHS	Reproducing Kernel Hilbert Spaces
MLP	Multilayer perceptron
RTO	Real Time Optimization
FP	First-principle
RPROP	Resilient back propagation
MeABP	Mean-Average Boiling Point
WABT	Weighted average bed temperature
CABP	Cubic average boiling point
MABP	Molal-average boiling point
HCR	Hydrocracking
VIM	Variable Importance Measure
OBB	Out-of-bag error for random forest modelling technique
VI	Variable Importance
MDI	Mean Decrease Impurity
VIF	Variance Inflation Factor
TBP	True Boiling Point

Chapter 1

Introduction

1.1. Motivation

Oil refineries play a very crucial role for the global oil supply chain process. Refineries process the crude oil to yield more useful products such as liquefied petroleum gas, gasoline, naphtha, kerosene, diesel fuel, asphalt base, heating oil, and petroleum coke (Gary, et al., 1984). Presently, the downstream refining industry¹ faces many technical and operational challenges due to the introduction or increase of environmental restrictions that lead to tighter product specifications that require investing in new refinery technologies, improved process operations and better crude oil feedstocks, as well as adaptation to the changes in demands and prices for end products. For example, Figure 1.1 shows how nitrogen oxides (NO_x) and particulate matter (PM) emission standards changed in Europe from 1992 to 2008, whereas Figure 1.2 shows how demands for refinery products are changed over time in UK. As Figure 1.2 shows, the largest markets at

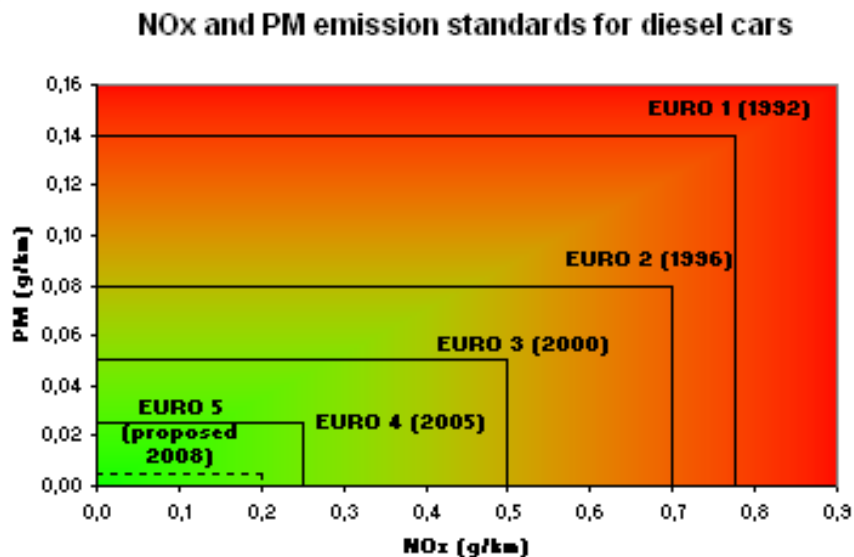


Figure 1.1 - Simplified chart showing the progression of European emission standards for Diesel cars
(European Union emission regulations, 2009)

¹ The downstream refining industry refers to the refining of crude oil and processing and purifying of the products, as well as the distribution of products to the market. The upstream sector refers to the searching of potential underground or underwater crude oil and natural gas sources, drilling of exploratory well and operating the well to recover and bring these sources to the surface. (Leffler, 1985)

the time were for fuel oil and gasoline contain around 50% residue ² (which is used to produce fuel oil in vacuum units), many refineries were built as simple plant with few conversion units and as time went by, the refined product fuels market changed and demand for gasoline, kerosene and diesel (gas oil) started to grow. At the same time, new product specifications were brought, such as removal of lead from gasoline, lower sulphur content in diesel, and so on. This situation was almost the same for the other countries, as well. To overcome these technical challenges, many refineries in UK and the other countries increased their complexity with new investments for the installation of more vacuum distillation, catalytic cracking³ and associated process units with desulphurization plants.

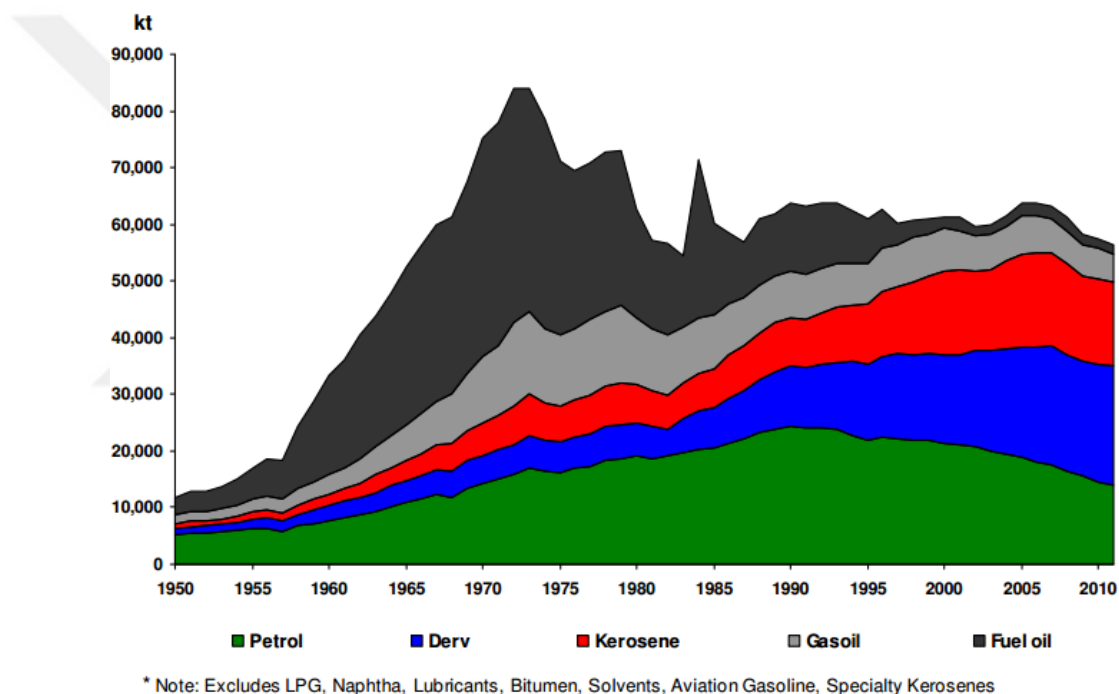


Figure 1.2 - Petroleum product consumptions from 1950 to 2011 in UK (UKpia)

Even though the investments for new units help the refineries to adapt their selves to the current product standards and demands, there is usually room for operational improvement and a tiny operational improvement often means a huge economic benefit and is worthy of effort. Operational improvement can be achieved through optimizing sub-processes, lowering energy/utility consumption, monitoring and controlling process variables and reducing emissions.

² Residue is one of the product from atmospheric distillation unit and is generally converted to other products at a reduced pressure and high temperature. (Leffler, 1985)

³ A refinery process to convert long chained hydrocarbons to short-chained hydrocarbons to produce lighter products

Practically, the performance of a refinery unit can be improved by using first-principle modelling of the process or exploring the optimal operation conditions based on the production data. Due to the inevitable deviation of any first principle model from the reality of a unit, data-driven unit models are often used to model the refinery processes by the process engineers. In Section 1.2 and Section 1.3, literature review on data-based model approach and first-principle model will be summarized, respectively.

1.2. Literature Review on Data-Based Model Applications in Refineries

To optimize the operational conditions of a refinery process through the use of data, one should gain knowledge about the process by collecting process data. This data should contain some measurements from the field, such as pressure, temperature, and flow, as well as the product quality. Even though it is easy to get information about operating conditions like pressure, temperature or flow through the transmitters located in the field, real-time monitoring of product quality is difficult or impossible due to lack of hardware sensors or their low reliability. Even though there are few hardware sensors available to monitor product quality, measurements are done with significant delays. At the same time, the laboratory measurement procedures are tedious and time consuming. The need for hardware sensors can be reduced by data-based models of processes, designed to estimate relevant process variables; thus, operation efficiency can be improved by implementing tight product quality control. The models used in such a framework are known as *Soft Sensors* or named *inferential estimators*. The basic idea of soft sensing is to choose some operational variables, which are easy to measure and related with the product quality, and then finding a mathematical model to reflect the relation between the product quality and operational variables. The mathematical model is found by adopting black box model, including regression analysis, back propagation neural network, support vector machines etc. (Yafen, et al., 2003).

Figure 1.3 summarizes the soft sensors applications in the process industry. In the context of petroleum refinery and petrochemical industries, soft sensing techniques have been proposed for prediction of different quality variables. A survey of different soft sensors reported in petroleum refinery is reported in Table 1.1.

In recent years, SVM regression is a new regression method based on the statistical learning theory and it is deduced based on the rule of structure risk minimization (Vapnik, 2000). According to Vapnik, the criterion of structural risk avoids the curse of dimensionality, and minimizes the empirical error and model complexity simultaneously, which effectively improves the generalization ability of SVR models and overcomes the inevitable drawbacks of empirical risk minimization of ANNs, where solutions may be captured in local minima and overfitting may happen. In Yafen Li (Li, et al., 2006), the implementation of estimation of kerosene dry point by least squared support vector regression (LS-SVM) is introduced and this model is compared with conventional radial basis function neural network (RBFNN) and traditional

SVM regression. The input variables of the models are selected via principle component analysis (PCA); and thus, the number of input variable is significantly reduced from 124 to 5. According to Yafen Li (Li, et al., 2006), soft sensing based on LS-SVM regression has better generalization capability compared to RBFNN.

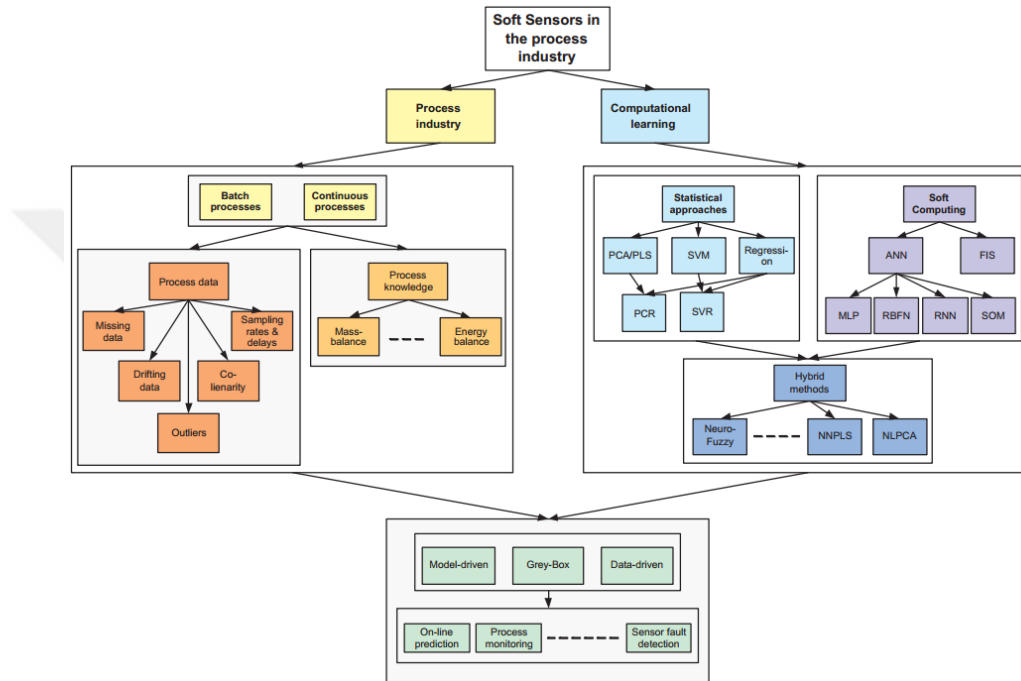


Figure 1.3 - Overview of Data-driven Soft Sensors in the Process Industry (Kadlec, et al., 2008)

Artificial Neural Network (ANN) offers as an alternative powerful and fast tool to model non-linear processes, such as hydrocracker reactor, distillation column, and etc. ANN has the ability to learn the relationship between the outputs and inputs for a system. In order to develop ANN model for a process, it requires suitable chosen network architecture and appropriate training data.

The literature reported some work on refinery processes using neural networks. In Enrique Arce-Medina (Enrique, et al., 2009), sulfur content of petroleum naphtha is predicted by artificial neural networks and according to his claim, ANN models can be used as tools for prediction purposes and they present a strong alternative to the classical modelling techniques using first-principles. In Sepehr Sadighi and Mohaddecy (Sadighi, et al., 2013), industrial catalytic naphtha reforming reactors⁴ are modelled by a layered-recurrent artificial neural network using back-propagation algorithm. Their artificial neural network model could predict the research octane number (RON) of gasoline with mean absolute error of 0.52%. In Ramli et al.

⁴ Catalytic naphtha reforming reactors are used in refineries to yield octane gasoline and hydrogen

(Ramli, et al., 2014), artificial neural network models are implemented to predict the product compositions of a debutanizer column and the composition predictions are compared with partial least square (PLS) and linear regression analysis methods to show its superiority over these conventional methods.

Table 1.1 - Survey of soft sensor applications reported in petroleum refinery

Author(s)	Year	Quality Variable Predicted	Soft Sensor Technique
Kresta et al. (Kresta, et al., 1994)	1994	Heavy key components in distillate	PLS
Chen and Wang (Chen, et al., 1998)	1998	Condensation temperature of light diesel oil	BPNN
Park and Han (Park, et al., 2000)	2000	Toulene composition	Multivariate locally weighted regression
Bhartiya and Whiteley (Bhartiya, et al., 2001)	2001	ASTM 95% end point of kerosene	BPNN
Fortuna et al. (Fortuna, et al., 2003)	2003	Hydrogen sulfide and sulfur dioxide	BPNN and RBFNN
Yan et al. (Yan, et al., 2004)	2004	Freezing point of light diesel oil	SVR, LS-SVR, BPNN
Ramli et al. (Ramli, et al., 2014)	2004	Top and bottom product composition for debutanizer	ANN
Dam and Saraf (Dam, et al., 2006)	2006	Specific gravity, flash point and ASTM temperature of crude fractionator products	BPNN
Yafen Li (Yafen, et al., 2006)	2006	Kerosene dry point	LS-SVM, RBFNN,SVM
Yan (Yan, 2008)	2008	Naphtha 95% cut point	Ridge regression
Kaneko et al. (Kaneko, et al., 2009)	2009	Distillation unit bottom product composition	PLS
Enrique Arce-Medina (Enrique, et al., 2009)	2009	Sulfur content of naphtha	ANN
Wang et al. (Wang, et al., 2010)	2010	ASTM 90% distillation temperature of the distillate	PLS
Sepehr et al. (Sadighi, et al., 2013)	2013	RON of gasoline	BPANN
Ramli et al. (Ramli, et al., 2014)	2014	Product composition of debutanizer column	ANN, PLS, linear regression

1.3. Literature Review on First-Principle Model Applications in Refineries

The other way of improving the operations within the oil refineries are the use of models built through first-principles. These models are also named rigorous models. This method is highly complex due to use of thermodynamic, physical, and chemical laws and embodies comprehensive physical property calculations, energy and material balance calculations. Process simulators embody these first-principle models. Table 1.2 denotes a list of software used to simulate the material and energy balances of chemical processing plants. These commercial process simulation applications enable optimization of conceptual design and operations. The main difference among these simulators are the flowsheeting technique used (sequential-modular or equation-oriented), the type of thermodynamic equations used, optimization techniques to optimize the process and graphical user interfaces (GUI). Since, these are not within the scope of the thesis, the details for these differences are not explained.

Table 1.2 - Commercial Process Simulation Software List

Software	Developer	Applications
Aspen Plus	Aspen Technology	Process simulation and optimization
Aspen Hysys	Aspen Technology	Process simulation and optimization
CHEMCAD	Chemstations	Software suite for process simulation
EMSO	ALSOC Project	Modelling, simulation and optimization
GPROMS	PSE Ltd	Advanced process simulation and modelling
PROSUMPLUS	ProSim	Process simulation and optimization
PRO/II	SimSci	Process simulation and optimization
Unisim	Honeywell	Modelling, simulation and optimization

A survey of different process simulator applications in petroleum refinery is reported in Table 1.3. In More et al. (More, et al., 2010), the crude distillation unit including a pre-flash tower, an atmospheric distillation unit and a vacuum distillation unit was optimized so that raw materials and energy costs are minimized. Parametric study pertaining to feed selection (feed stock considered are Bombay Crude, Araby crude and Nigeria Crude) and composition has also been conducted in this work to further benefit refinery planning and scheduling. In Zhou et al. (Zhou, et al., 2011), industrial hydrocracking unit is optimized by adjusting the flow rate of quench hydrogen through genetic algorithm(GA) and sequential quadratic programming(SQP) to improve the yield of the lighter products. In Raji et al. (Raji, et al., 2012), the performance of an existing industrial fluid catalytic cracking⁵(FCC) unit in a refinery is simulated and optimized by manipulating various process variables such as molar flow rate of recycle bottom oil, reflux ratio, and so on. According to the Raji's results, tremendous improvement of 17% in net profit is achieved. In Wang et al. (Wang, et al., 2013), Aspen Plus software is used to optimize an integrated hydrogen production process in a refinery based on the coupled feed of refinery off-gas and natural gas. The optimal operating conditions are obtained by performing a multi-objective optimization using a non-dominated sorting genetic algorithm-II (NSGA-II) technique. In M.A.Samborskaya et al. (Samborskaya, et al., 2014), optimum fractionation of products of refining gasoline on zeolite catalyst with account of the catalyst deactivation is obtained and according to the authors' claim optimal conditions were found to differ considerably from those of designed for fresh catalyst. In Pauladi et al. (Pouladi, et al., 2016), a refinery process called "gas sweetening process" is improved by using dynamic simulation of CO₂ absorber column and this research revealed that pressure variation does not have significant changes in the CO₂ absorption

⁵ Fluid catalytic unit is one of the most crucial conversion unit within petroleum refineries. It is used to convert high-boiling hydrocarbons to low-boiling hydrocarbons at high temperature and moderate pressure, with a fluidized powdered catalyst. (Gary, et al., 1984)

process efficiency, while both amine solution inlet temperature and total circulation rate of amine solution enhance the process efficiency.

Table 1.3 - Survey of process simulator applications in oil refinery

Authors	Year	Process Simulator	Application
More et al.	2010	Aspen Plus	Optimization of crude distillation system Crude feedstock selection
Zhou et al.	2011	Aspen Plus	Modeling and optimization of an industrial hydrocracking unit to improve the yield of diesel or kerosene
Raji et al.	2012	Aspen HYSYS	Modelling and optimization of Fluid Catalytic Cracking Unit
Wang et al.	2013	Aspen Plus	Simulation and multi-objective optimization of an integrated process for hydrogen production from refinery off-gas
M.A.Samborskaya et al.	2014	Aspen HYSYS	Optimal fractionation of products of refining straight-run gasoline on zeolite catalyst with account of its deactivation
M.A.Samborskaya et al.	2015	Aspen Plus	Pre-design optimization of crude oil distillation flowsheet
Pauladi et al.	2016	Aspen HYSYS	Modelling and optimization of CO ₂ absorber column

1.4. Comparison of Data-based Modelling and First-Principle Modelling Approaches in Refineries

Large-scale computer based simulations that use first-principles are being used to predict the behavior of refinery systems, as mentioned in Section 1.3. First-principle models can be used to obtain many information about the process that may not be possible to obtain in plant operation, such as compositions of flows, partial pressures of components, heat capacities or enthalpies of flows, and so on. However, the cost of an error between the predicted and actual performance of a refinery process unit can be very high. Therefore, it is crucial that simulation results should come with a high level of confidence when used to guide high-consequence decisions. Scientific foundations and assumptions on which the predictions rest, including limitations on the experimental and calculation data and numerical methods used, as well as errors in the measurements on the fields due to the reasons related to the hardware itself may change the level of accuracy of the predictions done by the simulations based on first-principle models. On the other hand, data-based models (sometimes called as black box models) can be very successful for estimating any system response like product quality of a refinery process, with properly chosen inputs. However, different from the first-principle models, it is not possible to have any knowledge of internal workings of a system with these data-based empirical models, and to explain the relation between inputs and outputs of the model. Table 1.4 shows the comparison between simulation models based on first-principle and data-based models based on empirical data.

Table 1.4 - Comparison between simulation models and data-based models (Z.Kister, 2002)

	Simulation Model (Based on first-principles)	Data-driven Models(Based on empirical data)
Advantages	- Full knowledge of internal workings - Explains the causal relationships between input and output - Helps in finding out an optimum design - Easy to explore process sensitivity - Possible to check the process performance with different thermodynamic model - Helps to estimate equipment cost - Helps in investigating feed-stock flexibility	- Preferable in cases where the modeller's knowledge about the process is limited - Useful when primary interest is in fitting the data regardless of a particular mathematical structure of the model - Easy to implement for process control purposes
Disadvantages	- Requires high-quality of input data - High level of computation complexity - Computations take more time - Results may alter according to solver used - Uncertainty would increase due to an overly complex system, since each separate part induces some amount of variance in the model	- Difficult to explain the relations between the inputs and outputs just by analyzing the model - Deals only with what is measurable - Can be time consuming due to feature parameter selection, feature selection, model assessment, and etc.

1.5. Contributions

In the literature review sections, we showed that some of the researchers prefer the process simulators, whereas some of them prefer the data-driven models to improve the process efficiency of a refinery process. (Recall that the advantages and disadvantages of these two approaches are also given in Section 1.4). Table 1.5 summarizes all the papers investigated in the literature review by giving the purpose of the paper, the modeling approach used (Data-driven or first-principle modelling), input data type (Operating conditions, feed properties), and the outcomes of the study.

In Table 1.5, it can be seen that some of the works (mostly the works that use the first-principle models) require feed property information. One way to obtain the feed property information is to laboratory measurements. However, the laboratory measurements are infrequent (one sample per day), slow (time delay of hours), and sometimes unreliable. Another way is the use of online analyzer measurements located in the field. Unfortunately, the online analyzers are quite expensive, the accuracy of the measurements can be problematic, and the maintenance of these analyzers can be time consuming. Reliable prediction of the feed properties from the operational data, which is available on any time, would be extremely beneficial in that sense. By achieving that, the process engineers can use the process simulators, which requires the property feed information, to carry the process conditions one step further. The first contribution of the thesis, we provide a method to estimate the feed properties by building feed characterization models, whose input variables are the process variables. The method is described in detail in Chapter 6.

According to the same table (Table 1.5), there is no study conducted to adjust the outcomes of the first-principle models, which are used by the process simulator applications. The outcomes of the process simulator can be misleading due to several reasons, such as incorrect input variable measurements (in other words, incorrect of flows, temperatures, pressures and so on), the model assumptions, increasing uncertainty due to increasing complexity in the system (since each separate part induces some amount of variability in the model), and so on. The second and the main contribution of the thesis is to correct the outcomes of the process simulators with the simulator output calibrator models, which are basically data-driven models like artificial neural networks. By correcting the outputs of the process simulator, process engineers can both obtain more accurate and realistic outputs from the simulator and benefit from the extra properties of the simulators, such as case study, cost/profit estimations, and full-knowledge of internals. In Chapter 7, the details about how the simulator outputs are corrected are given.

The third and the final contribution is that this study is applied on a real industrial fractionator column of the hydrocracker unit in İzmit Refinery located in Turkey. The outcomes from the simulator for the product qualities of the fractionator column are corrected by the proposed methodology. The results show that methodology followed in the work is quite successful to correct the outcomes of fractionator column simulation, built in Aspen HYSYS v8.0. To understand the process and simulation of interest, the fractionator column process is explained in detail in Chapter 2 and its corresponding simulation is explained in Chapter 3.

Table 1.5 – Literature survey of the works carried out to improve operation conditions

Paper	Year	Purpose			Model Type		Input Data Type		Outputs of the study
		Estimation of product quality	Optimization of the process	Fault Detection	Data-Driven	First-Principle	Operating Conditions	Feed Properties	
Kresta et al. (Kresta, et al., 1994)	1994	✓			✓		✓		Heavy key components in distillate
Chen and Wang (Chen, et al., 1998)	1998	✓			✓		✓		Condensation temperature of light diesel oil
Park and Han (Park, et al., 2000)	2000	✓			✓		✓		Toluene composition
Bhartiya and Whiteley (Bhartiya, et al., 2001)	2001	✓			✓		✓		ASTM 95% end point of kerosene
Fortuna et al. (Fortuna, et al., 2003)	2003	✓			✓		✓		Hydrogen sulfide and sulfur dioxide
Yan et al. (Yan, et al., 2004)	2004	✓			✓		✓		Freezing point of light diesel oil
Dam and Saraf (Dam, et al., 2006)	2006	✓			✓		✓	✓	Specific gravity, flash point and ASTM temperature of crude fractionator products
Yafen Li (Yafen, et al., 2006)	2006	✓			✓		✓		Kerosene dry point
Yan (Yan, 2008)	2008	✓			✓		✓		Naphtha 95% cut point
Kaneko et al. (Kaneko, et al., 2009)	2009	✓			✓		✓		Distillation unit bottom product composition
Enrique Arce-Medina (Enrique, et al., 2009)	2009	✓			✓		✓	✓	Sulfur content of naphtha
Wang et al. (Wang, et al., 2010)	2010	✓			✓		✓		ASTM 90% distillation temperature of the distillate
More et al.	2010		✓			✓	✓	✓	Optimum operational conditions for CDU column
Azadeh et al.	2010			✓	✓		✓		Failure diagnosis of centrifugal pumps
Zhou et al.	2011		✓			✓	✓	✓	Optimum operational conditions for Hydrocracking Unit
Raji et al.	2012		✓			✓	✓	✓	Optimum operational conditions for FCC Unit
Sepehr et al. (Sadighi, et al., 2013)	2013	✓			✓		✓		RON of gasoline
Wang et al.	2013		✓			✓	✓	✓	Optimum operational conditions for H2 ring
Ramli et al. (Ramli, et al., 2014)	2014	✓			✓		✓		Top and bottom product composition for debutanizer
M.A.Samborskaya et al.	2014		✓			✓	✓	✓	Optimum fractionation of products on zeolite catalyst
M.A.Samborskaya et al.	2015		✓			✓	✓	✓	Optimum operational conditions for CDU column
Askarian et al.	2016			✓	✓		✓		Detection of operator action consequences

1.6. Overview of Thesis

Chapter 1 introduces the literature review sections. In this chapter we showed that some of the researchers prefer the process simulators, whereas some of them prefer the data-driven models to improve the process efficiency of a refinery process. The advantages and disadvantages of these two approaches are also given in this chapter.

Chapter 2 presents the information about the hydrocracker fraction column operation and important product qualities that must be monitored and enhanced. In this chapter, hydrocracker unit process flow diagram is explained in detail and some terminology, such as true boiling point, ASTM distillation, flash point are explained to facilitate to understand the further chapters.

Chapter 3 presents how the process simulations are done by using Aspen HYSYS software. Simulator inputs and outputs are listed in this chapter. Moreover, the functionality of some important equipments of the fractionator column, such as pump arounds, side-stripper columns, are introduced and how these equipments are represented in the simulation is explained.

Chapter 4 presents the background information about the modelling techniques used in the thesis. The modelling algorithms are explained in detail for each model techniques. Some information about feature selection, model validation and selection are also given in this chapter.

Chapter 5 explains the scheme that combining the process simulation and data-based models to estimate the process simulator error. In Chapter 6 and Chapter 7, the feed characterization model building and simulation output calibrator model building methodology is explained in detail, respectively. The model results are given and evaluated in detail inside these chapters. In chapter 8, the results are discussed and Chapter 9 concludes with a brief summary of the study.

Chapter 2

Fractionator Column

In this chapter, the fractionator column process is described briefly to create a knowledge base on the purpose of the process and the feeds and the products of the process. Beside these, the product qualities considered during the thesis are explained. More details about the process are given in Chapter 3.

2.1. Fractionation Process Description

Hydrocracker is one of the units of a complex refinery in which residue of crude oil remaining after atmospheric and vacuum distillations is converted into middle distillates and lighter products in presence of high temperature, pressure and excess hydrogen (Speight, 2006). In a hydrocracking plant, reactor effluent is fed to fractionation section which includes distillation columns in which product mixture is separated into final products.

In Turkish Petroleum Refineries Corporation there is a hydrocracking plant in İzmit Refinery which takes feed from three different vacuum columns. The unit can be considered as two parts, reaction section and fractionation section. In the reaction sections, removal of sulphur and nitrogen and hydrocracking reactions take place. In the fractionation section, the reactor effluent is separated into final products via distillation columns. There are certain number of specifications on the products that all companies must satisfy. According to those specifications, production planning departments optimize the blends composed of products of different units. The specifications given to the process units are such that the resulting product that is to be sold meets the specifications that are defined in regulations. The products obtained from reaction section must meet the sulphur specification. Other specifications, flash points and 95% distillation points, are controlled in fractionation part. The control of these specifications is done via base layer control, which is simple cascade loops and PID controllers in DCS system.

In hydrocracking plant, heavy vacuum gasoil (HVGO) feed coming from three different vacuum columns is processed. Feed is first sent to a hydrotreater reactor after it is heated up to reaction temperature. In hydrotreater reactor, sulphur, nitrogen, oxygen and other undesired impurities are removed from hydrocarbon. The impurities leads to catalyst poisoning so they must be cleaned before the feed goes to the hydrocracker reactors. The treated feed is then sent to hydrocracker reactors. The reactor effluent is first sent to a high pressure separator in order to separate hydrogen and sent it back to the system as recycle hydrogen. In the separation section, there are fractionator, naphtha splitter and debutanizer columns to

separate diesel, kerosene, heavy and light naphtha and liquefied petroleum gas from unconverted oil. A simple process flow diagram is given in Figure 2.1.

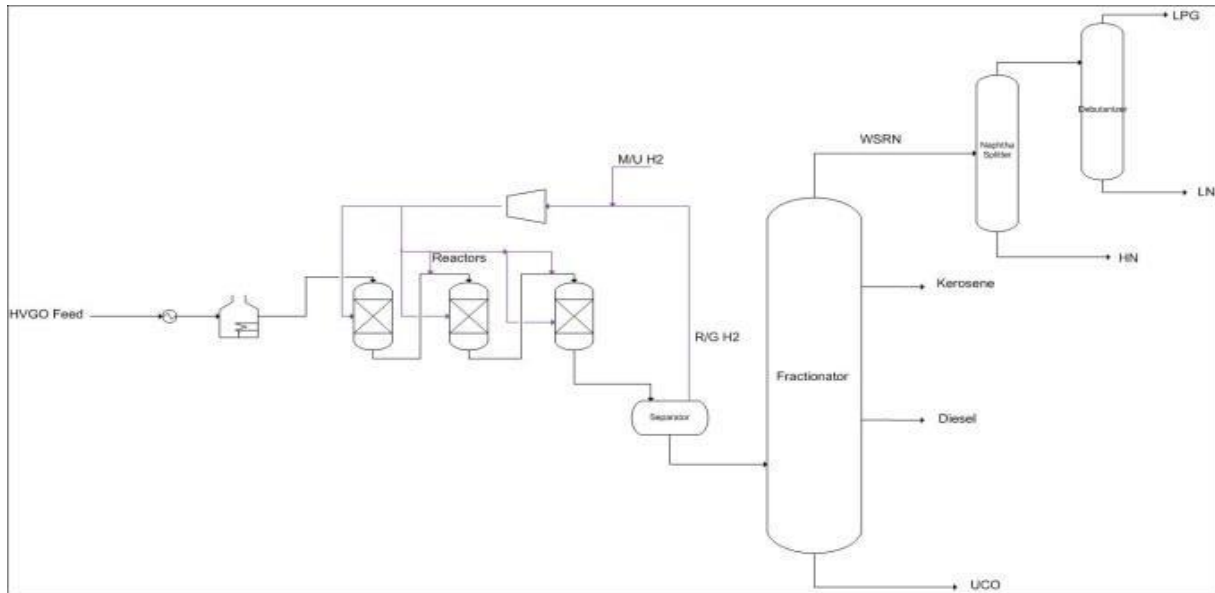


Figure 2.1 - Hydrocracker Simple Process Flow Diagram

In this thesis, fractionator column is the distillation column of interest. The fractionator is the main column in which reactor effluent is separated into products that meet market specifications. The most crucial products of hydrocracker unit comes from this column. It consists of four main products; namely, unconverted oil (bottom product), diesel, kerosene and fractionator top (heavy straight run naphtha-HSRN). Fractionator top stream is taken into a reflux drum after cooled down. The stream is separated into two parts at the outlet of the drum. One is sent back to the column as reflux and the other one is sent to naphtha splitter column for further separation. There are two total draw trays in the fractionator, from which kerosene and diesel products are obtained. Kerosene, which is usually counted as the most valuable product of the unit is taken from the n th tray of the fractionation column and sent to the product stripper. In the product stripper, lighter parts of the kerosene is stripped and sent back to column. Finally, the product kerosene is cooled and sent to the storage. Diesel, which is taken from the tray below kerosene, has the same procedure with kerosene before it is sent to storage. From the bottom of the fractionator column unconverted oil (UCO) is obtained and sent to another unit for further processing.

2.2. Product Qualities

Some test methods and terminologies will be covered in this section to understand the product qualities.

2.2.1. True Boiling Point (TBP) Curve

The composition of any crude oil sample is approximated by a true boiling point (TBP) curve. The method used is basically a batch distillation operation, using a large number of trays, usually greater than 60 and high reflux to distillate ratio (greater than 5). The temperature at any point on the temperature-volumetric yield curve represents the true boiling point of the hydrocarbon material present at the given volume percent point distilled. TBP curves are generally run only on the crude oil and not on petroleum products, such as HSRN, kerosene or diesel. (Parkash, 2003). However, to see how the TBP curve looks like, Figure 2.2 shows a sample of TBP curve for a diesel sample.

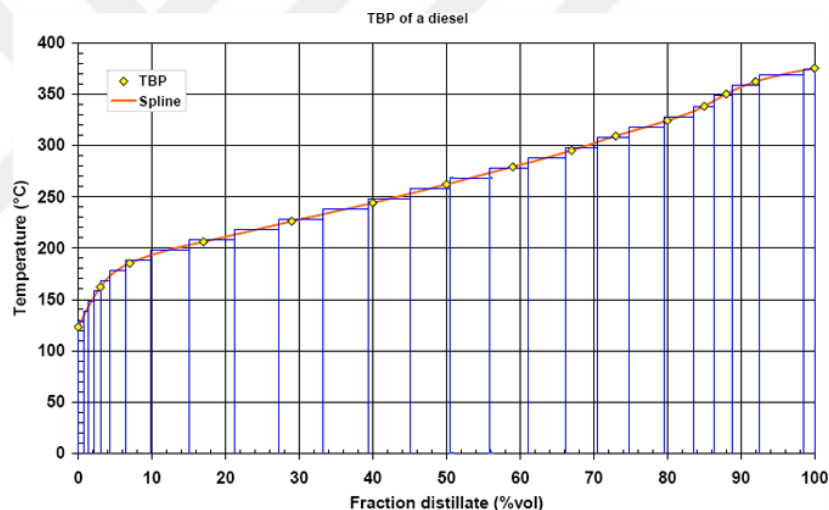


Figure 2.2 – TBP Curve for diesel product (Hemptinne et al., 2011)

2.2.2. ASTM Distillation

For petroleum products such as HSRN, kerosene or diesel, a more rapid distillation procedure is used. This is a procedure developed by the American Society for Testing and Materials (ASTM), employs a batch distillation procedure with no trays or reflux between the still pot and the condenser.

ASTM D86 is a specific test method of determining the boiling range of a petroleum product by performing a simple batch distillation has been in use as long as the petroleum industry has existed. It is one of the oldest test methods under the jurisdiction of ASTM Committee D02, dating from the time when it was still referred to as the Engler distillation. Since the test method has been in use for such an extended period, a tremendous number of historical data bases exist for estimating end-use sensitivity on products and processes.

ASTM D1160 is the test method covers the determination, at reduced pressures, of the range of boiling points for petroleum products and biodiesel that can be partially or completely vaporized at a maximum liquid temperature of 400 °C. Both a manual method and an automatic method are specified. Through the thesis, %95 boiling point of a product will refer to the temperature at which %95 of a product will be vaporized. (Parkash, 2003)

2.2.3. Flash Point

The flash point of a chemical is the lowest temperature where enough fluid can evaporate to form a combustible concentration of gas.

The flash point is an indication of how easy a chemical may burn. Materials with higher flash points are less flammable or hazardous than chemicals with lower flash points.

IP 170 is the test method to measure the flash point of a petroleum fraction. This International Standard specifies a method for the determination of the closed-cup flash point of combustible liquids having flash points between -30,0 °C and 70,0 °C, inclusive. However, the precision given for this method is only valid for flash points in the range -5,0 °C to 66,5 °C. (Parkash, 2003)

2.2.4. API Gravity

API Gravity, or namely American Petroleum Institute gravity, is a measure of how heavy or light a petroleum product is compared to water. API gravity is a dimensionless quantity. This quantity is calculated as follows (ASTM, 1960):

$$API = \frac{141.5}{Specific\ gravity} - 131.5 \quad (2.1)$$

Note that it is an inverse measure of density relative to that of water. For example, if one petroleum liquid is less dense than another, it has a greater API gravity.

2.3.Product Specifications for Fractionator Column

The product qualities that must be satisfied are listed below.

Table 2.1 – Product Specifications for Fractionator Column

Product	Product Quality	Test Method
HSRN	%95 Boiling Point	ASTM D86
Kerosine	%95 Boiling Point	ASTM D86
Kerosine	Flash Point	IP 170
Diesel	%95 Boiling Point	ASTM D86

Note that even though the other boiling points, such as 5%, 55%, %90 and the API gravities of the products are not product specifications, they are considered in the next chapters because of the fact that the product specification can alter according to the refinery. For instance, some refineries can take 90% boiling point as a product specification instead of 95% boiling point.

Chapter 3

Process Simulation of Fractionator Column

In the previous chapter, the location of the fractionator column in hydrocracker unit, the input and output streams, the products, and the product qualities are explained. Chapter 3 focuses on the product fractionator column itself by giving more detail over simulation. Then, it gives the necessary inputs and the possible outputs that can be taken from the simulator. Chapter 3 also gives brief theoretical information for how the feed properties, such as heat capacity, molecular weight, density, vapor fraction and so on, can be extracted from the feed TBP curve by pseudo-component generation. Note that in Chapter 5, the feed TBP is estimated by using data-driven feed characterization models, whose inputs are the operational inputs. The estimated feed TBP will be given to the simulator to estimate the physical properties of the feed. Thus, understanding how feed properties can be calculated from TBP information can be beneficial. The first principles-based physicochemical models used within the simulator are not described in this work since it is not within the scope of the thesis.

3.1. Introduction

This chapter starts with a process description of the HYSYS Simulation of the fractionator column and then continues with the feed characterization, and simulation inputs and outputs.

3.2. Process Description of HYSYS Simulation

Our process is basically called “fractional distillation”, which is the separation of a mixture into its component parts, or fractions, separating chemical compounds by their boiling point by heating them to a temperature at which one or more fractions of the compound will vaporize (Perry, et al., 1984). The following figure (Figure 3.1) shows the main case process flow diagram, which only shows the inputs and outputs of the process. There are more equipment within the system and they are shown in an environment called “column environment”. The distillation column enframed with a black box also contains 2 distillation column inside, 4 heat exchangers, 1 three-phase separator, 1 reboiler. This detailed system is shown in the Figure 3.2.

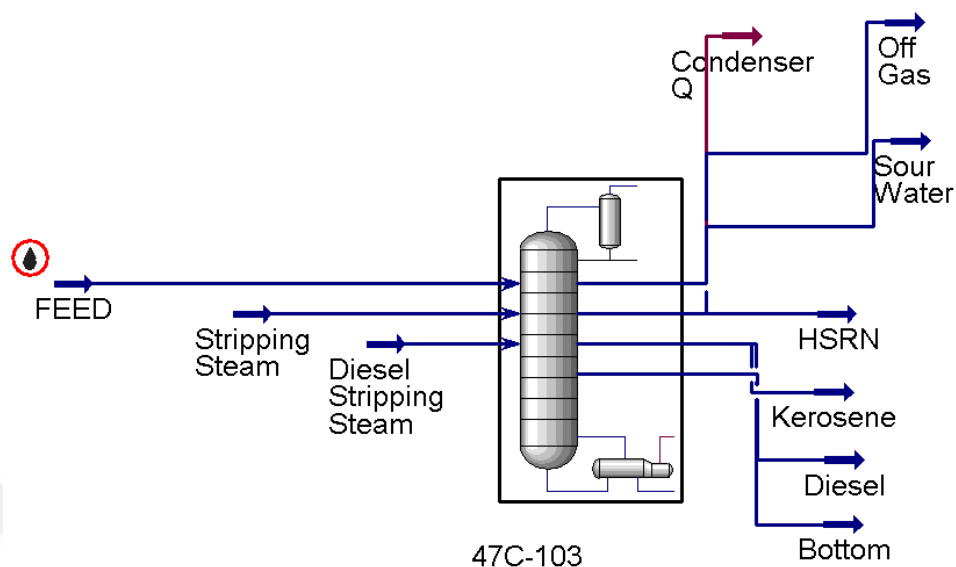


Figure 3.1 – Main Case view of the simulation

The streams with a dark lilac-color shows energy streams, whereas the streams with a blue color show the material streams. As can be seen from the figure above, the column has 3 input streams, namely “FEED”, “Stripping Steam” and “Diesel Stripping Steam”. Similarly, it has 6 output streams, namely “Off Gas”, “Sour Water”, “HSRN”, “Kerosene”, “Diesel” and “Bottom”.

The main products of the column are HSRN (Heavy-straight run naphtha), Kerosene, Diesel and unconverted oil (Bottom product). The off gas and sour water streams are the by-product of the column and there are sent to another refinery units to be used for another purposes.

It is very crucial to characterize the input streams to get the correct outputs from the simulation. Since stripping steam and diesel stripping steams just include water, they are easy to characterize. However, the “Feed” stream composed of many components in order of the thousands and hence there is no current laboratory technique to measure the composition of the feed. Therefore, the technique called “pseudo-component generation” is used to characterize the feed, which will be discussed in detail in the following sections.

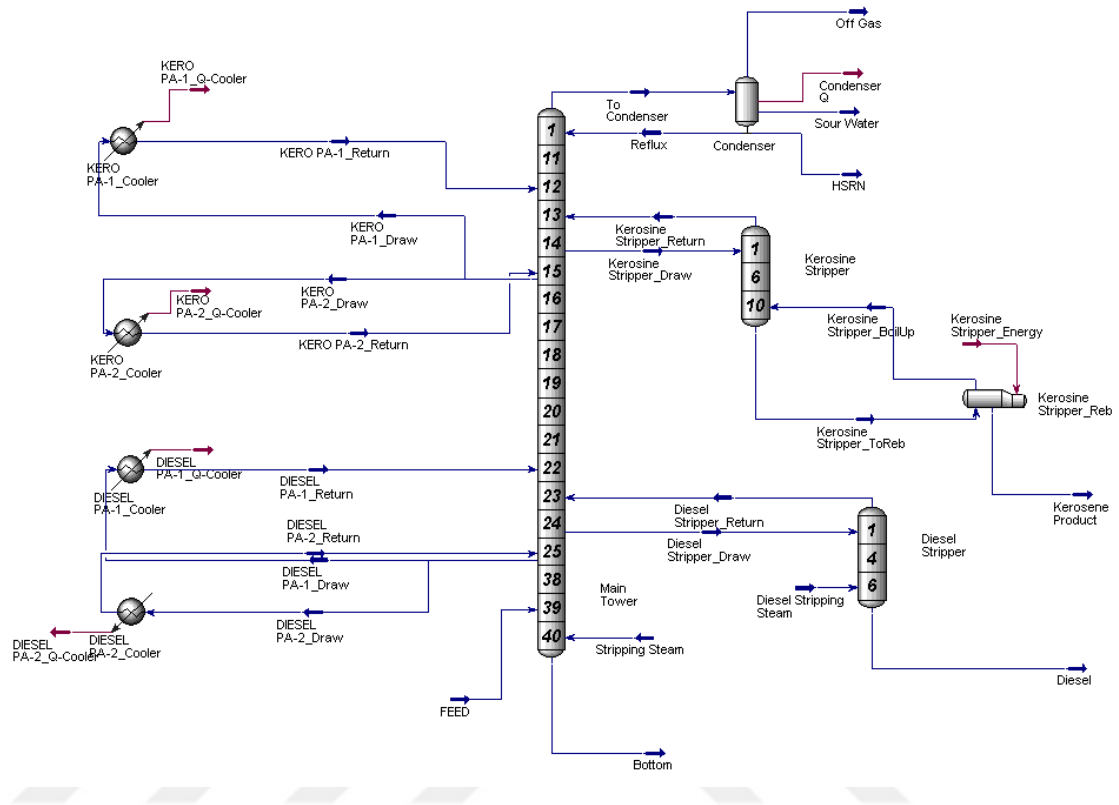


Figure 3.2 – Detailed view of the simulation (Column environment)

The figure above shows the detailed view of the column environment. FEED enters the column from the bottom section (39th tray). The stripping steam enters the column from 40th tray. The stripping steams reduce the hydrocarbon partial pressure and thus allows more components to vaporize and to escape from the liquid phase into the vapor phase. In other words, by reducing the partial pressures the hydrocarbons vaporize easily. We have 2 options to improve the performance of the distillation column: increase the feed temperature or increase the stripping steam amount. Increasing the feed temperature requires more duty load in the feed furnace, which increase the utility cost of fuels consumed in the furnace and similarly increasing the stripping steam requires more production of steams, which also incurs some costs. There are trade-offs between these two options needed to be optimized.

There are 2 *side-stripper* in the fractionator column, one for stripping kerosene and one for stripping diesel. The purpose of the *side stripper* is to remove some light hydrocarbons by using steam injection or an external heater called 'reboiler' (Jechura, 2016). It essentially helps to meet the properties specified for kerosene and diesel, since in a multi-component distillation there is overlap of constituents of various cuts. The stripping steam rate, or reboiled duty is controlled such as to meet the flashpoint specification of the product. Similarly to the main column, the side stripper has fractionating trays for providing contact between

vapour and liquid. The vapours produced from the top of the side stripper are routed back via pipe into the fractionating column. Kerosene stripper has 10 sieve trays, whereas that of diesel has 6 trays. The bottom of kerosene stripper column is heated with a reboiler to remove lighter components from the kerosene product so that it becomes possible to adjust flash point of kerosene. Even though diesel stripper column is designed such that the bottom of diesel stripper is heated with diesel stripping steam, there is no diesel stripping steam entering to the diesel stripper column, since even without the diesel steam, the flash point of diesel is satisfied. Therefore, there is no need to consume steam.

There are two ways to remove heat from a distillation tower: top reflux and circulating reflux (pump around) (see Figure 3.2). The vast majority of fractionator columns have top reflux. Cooled liquid from the condenser drum(vessel) is pumped onto the top tray of the tower. The alternate cooling method is pump-around (circulating reflux). The purpose of the pump-around is to cool and partially condense the up-flowing vapors. One can wonder the reason beyond the removal of heat from the column. This can be described as follows:

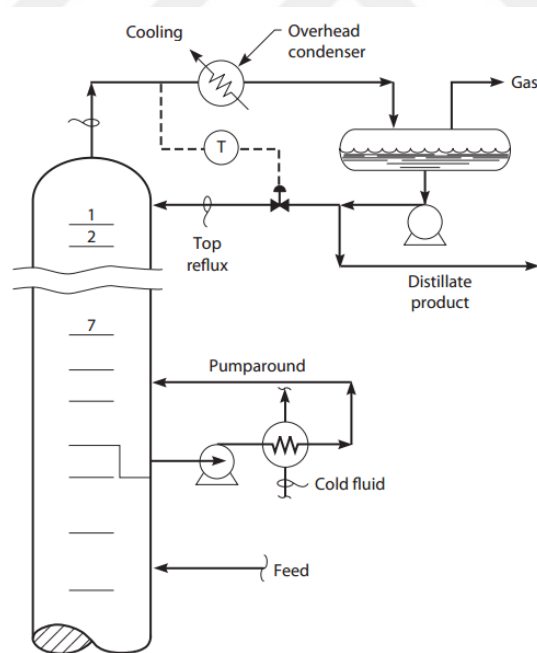


Figure 3.3 – Top Reflux and pump arounds (Representative drawing)

Liquid is drawn from a tray and cooled with another fluid in the cooler heat exchanger and then sent to the upper trays as can be seen from Figure 3.3. Note that tower-top reflux is controlling the tower-top temperature. If we were to reduce the pump-around circulation rate, less heat would be extracted from pump-around draw tray. In such a case, more and hotter vapor would flow up the tower. Top reflux temperature

control valve would open and so top reflux rate would go up. The vaporization of reflux on the top tray would increase. The overhead(top) condenser duty would increase. In conclusion, the decrease in the heat duty of the pumparound heat exchanger would equal the increase in the heat duty of the overhead condenser. Thus, it is possible to say that heat balance of the tower is preserved. Some of the heat that was being recovered to the cold fluid is now lost to cooling water (or cooler fans as in our case). This shows the most important function of pump-around: recovering heat to another process stream that would otherwise be lost.

There are 2 pump-arounds within our system, one for kerosene and one for diesel.

Kerosene Pumparounds : Kerosene product is drawn from 15th tray of the main fractionator column and then sent to coolers. One part of the cooled liquid is sent to 12th tray and the other part is sent to 15th tray. Thanks to this pumparounds, operator can manipulate the heat removed from the column.

Diesel Pumparounds : Similar to kerosene pumparound, diesel product is drawn from 25th tray of the main fractionator column and then sent to coolers. One part of the cooled liquid is sent to 22th tray and the other part is sent to 25th tray.

Pumparounds allow the utilization of available column heat via the column feed heat exchanger network, maintaining the stability of the vapor loading of the column. Maintaining the amount of liquid traffic in the column for effective fractionation is another important concept that needs to be considered. Vapor-liquid load can be quantified by reviewing the average vapor-liquid molar traffic in any zone of the column.

Pumparounds also control and distribute fractionation between products, since they adjust the flow of vapor and liquid within the column. If the rate of pumparound (that is, heat removal by pumparound) is decreased by the operator, the vapor load to the top will be increased. The top reflux rate will also increase in order to keep the top temperature constant. If there were no pumparounds, the highest liquid load would be at the top of the column due to an increase in top reflux. In addition, the amount of heat lost to the atmosphere from the column's top condenser would be higher.

3.3. Characterization of The Feed

This section introduces the method for characterizing the column feed stream and for estimating the physical and chemical properties. We begin by back-blending method to obtain the feed TBP. Next, we discuss the generation of pseudo-components of the feed based on boiling-point ranges and the estimation of density and molecular weight distributions of the resulting pseudo-components.

It is important to note that we do not have feed TBP information of the fractionator feed from the laboratory so it is calculated by back-blending method.

3.3.1. Back-blending

The most important factor in the success of any model of distillation columns is an accurate representation of the feed. There are two ways to quantify the feed to the column. The first method relies on the distillation information of the feed. The other method uses the current column product yield (product amount) and qualities to back-mix or “back-blend” these products. The goal of back-blending is to recover the composition of the feed to the column in terms of pseudo-components. This method is very useful when we have little information of the feed. At a minimum, we must obtain a detailed distillation curve and density information. The following figure shows the back-blending procedure:

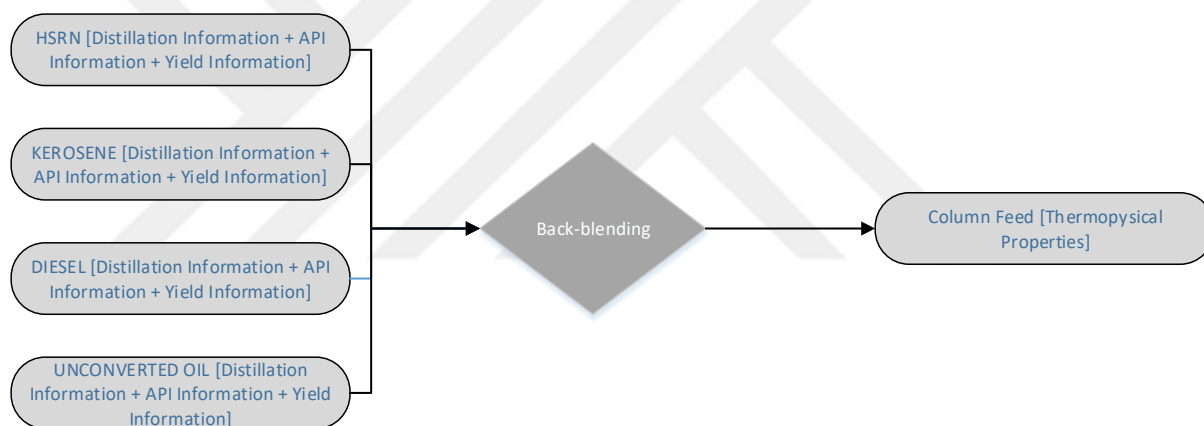


Figure 3.4 – Back-mixing (Back-blending) Method

The distillation information include ASTM D86 or ASTM D1160 distillation curves. The following figure shows a sample distillation data for the product on 23.02.2016:

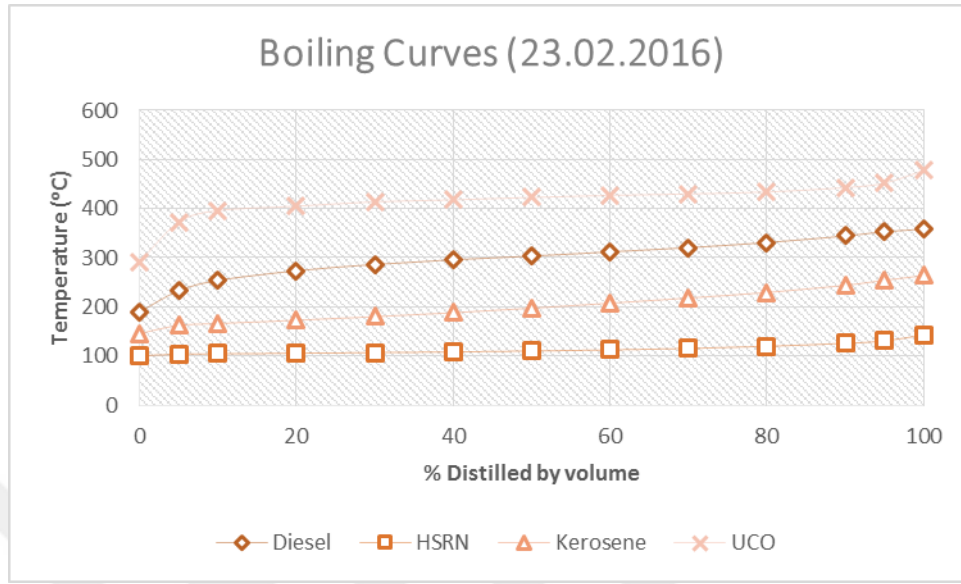


Figure 3.5 – Distillation information of the products

In addition to distillation information, one or more bulk properties will be good to accurately estimate the properties, but it is not mandatory. Bulk properties provide a quick understanding of the type of oil fraction sample, such as sweet and sour (according to sulphur content), or light and heavy (according to API gravity), etc. Bulk properties include specific gravity, sulfur content, nitrogen content, metal content, C/H ratio, refractive index, carbon residue and so on. However, in general API(American Petroleum Institute) gravity to specify the specific gravity (SG) of the oil fractions is used:

$$API\ Gravity = \frac{141.5}{SG} - 131.5 \quad (3.1)$$

Where, SG is the specific gravity defined as the ratio of the density of the oil fraction to the density of water both at 15.6 °C (60 °F). Even though there are many bulk properties to characterize the feed, in general only API gravity is used. The following figure shows the API gravities of the products (oil fractions):

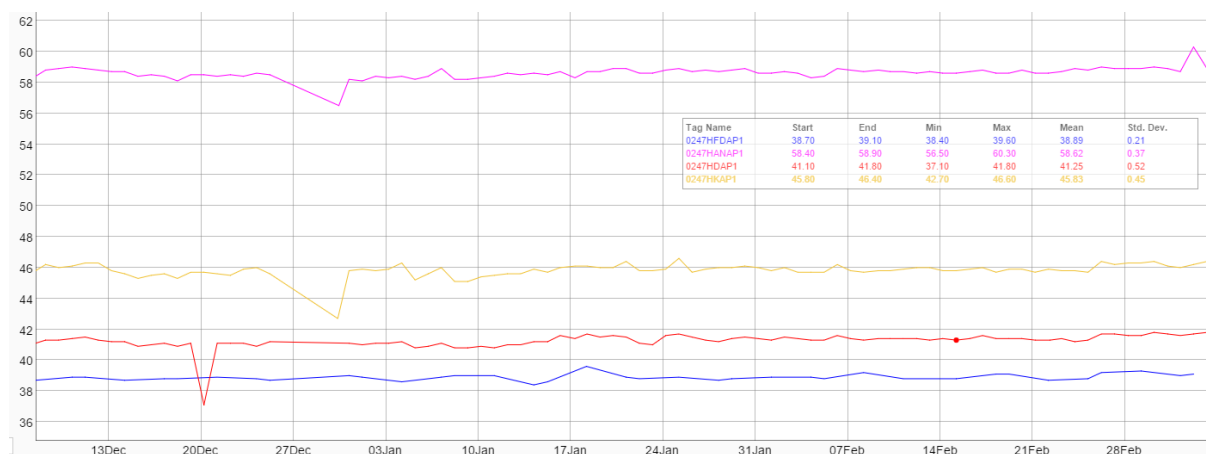


Figure 3.6 – API Gravity (@60°F) Trends for products (2016) – Pink: HSRN, Yellow: Kerosene, Red: Diesel, Blue: Unconverted oil (Bottom Product)

As can be seen from the Figure 3.6, the heaviest product of the column is unconverted oil (bottom product) and it has the smallest API gravity value. The lightest product of the column is HSRN (top product) and it has the largest API gravity value.

Product yield is another information needed for back-blending process. Yield is defined as the amount of product obtained in the distillation column. Figure 3.7 shows the yield trends for the products.

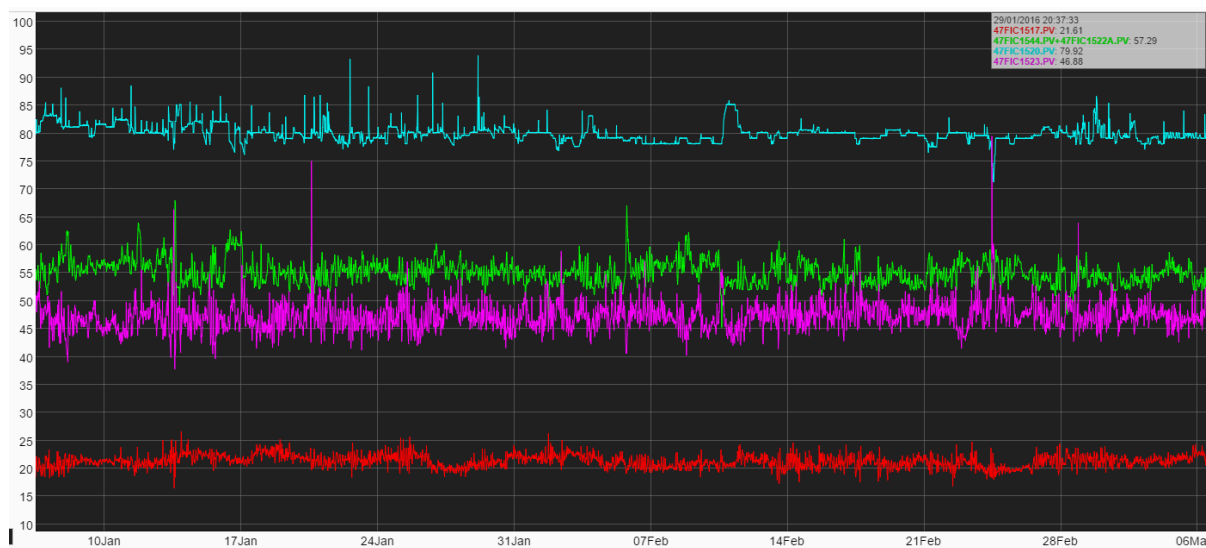


Figure 3.7 – Product yields (2016) – Turquoise: Kerosene product flow(m3/h) , Green: Unconverted oil flow(m3/h), Pink: Diesel Product Flow (m3/h), Red: HSRN Product Flow (m3/h)

As can be seen from the Figure 3.5, 3.6, and 3.7, all the necessary information to apply back-blending method is available. So, the TBP curve for the feed of the fractionator column is obtained by applying back-blending method.

3.3.2. Pseudo-component Generation

Since there are many components inside the feed, it is not possible to get the molecular composition information by laboratory analyses. Therefore, it is assumed that the feed is composed of pseudo-components. To simulate refining processes, the first task is to construct a pseudo-component scheme to characterize the feedstock. Data requirement and definition of the pseudo-components depend on the type of the refining process to be modeled. There are different concerns to be addressed when building pseudo-components for fractionation and reaction units. The pseudo-components for fractionation units have to accurately characterize volatilities of the hydrocarbons in the feedstock in order to calculate vapor-liquid-equilibrium in distillation columns. Therefore, refiners use pseudo-components based on boiling-point ranges to represent the feedstock and model fractionation units. For modeling of reaction units, refiners partition the hydrocarbons into multiple lumps (or model compounds) based on molecular structure or/and boiling-point ranges, and assume the hydrocarbons of each lump to have an identical reactivity in order to develop the reaction kinetics for reaction units. This section deals with pseudo-component generation based on boiling-point ranges for fractionation unit.(Chang et al., 2012)

Most commercial process simulators include the capability to generate pseudo-components based on boiling-point ranges representing the oil fractions. Conventionally, there are four steps to develop pseudo-components based on boiling-point ranges to represent petroleum fraction: (Chang et al., 2012)

Convert distillation information (ASTM D86/ASTM D1160/ASTM D2887) into TBP (true boiling curve) if TBP curve is not available.

1. Cut the entire boiling range into a number of cut-points ranges which are used to define pseudo-components. Note that the determination of the number of cuts is arbitrary. The following table lists the typical boiling-point ranges for pseudo-components in commercial simulators:

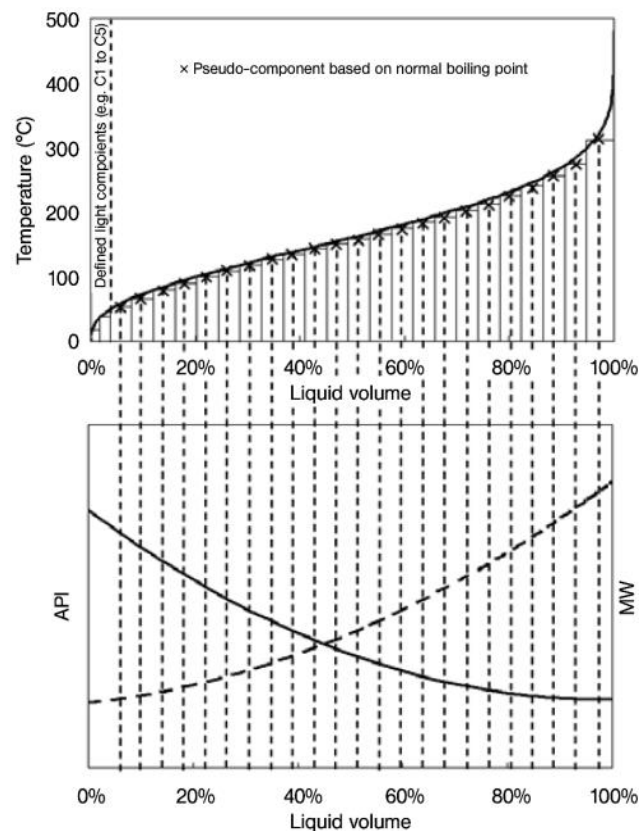


Figure 3.8 – Relationship between pseudo-component properties and TBP Curve (Kaes, G.L. Refinery Process Modeling – A Practical Guide to Steady State Modeling of Petroleum Processes; The Athens Printing Company: Athends, GA, 2000)

Table 3.1 – Suggested number of pseudo-components

Boiling Point Range of Oil Fraction(°C)	Suggested Number of Pseudo-components
425	30
650	10
900	8

2. Estimate the density distribution of pseudo-components if only the bulk density is available.
3. Assume Watson-Murphy “characterization factor” or K-factor to be constant throughout the entire boiling range and calculate mean-average boiling point (MeABP). Dissimilar to weight-average boiling

point(WABP), MeABS is defined as the average of molal-average boiling point(MABP) and cubic-average boiling point(CABP). The following equations define these four boiling point indications:

$$WABP = \sum_{i=1}^n x_i T_{bi} \quad (3.2)$$

$$MABP = \sum_{i=1}^n x_i T_{bi} \quad (3.3)$$

$$CABP = \left(\sum_{i=1}^n x_i T_{bi}^{\frac{1}{3}} \right)^3 \quad (3.4)$$

$$MeABP = \frac{MABP + CABP}{2} \quad (3.5)$$

Where T_{bi} indicates boiling point of component I and x_i in Equation (3.2) to (3.4) indicate weight fraction (varying from 0 to 1), molar fraction and volume fraction of component i, respectively.

Watson K-factor is calculated as follows:

$$K_{avg} = \frac{\sqrt[3]{MeABP}}{SG} \quad (3.6)$$

Where, SG is the bulk specific gravity at 60 °F.

The density distribution of the entire boiling range is calculated with the following formula:

$$SG_i = \frac{\sqrt[3]{T_{i,b}}}{K_{avg}} \quad (3.7)$$

After calculating these factor, any physical property of pseudo-component can be calculated with some equations can be found in the literature. For instance, critical temperature can be estimated with the following equation :

$$T_{c,i} = 189.8 + 450.6 \times SG_i + (0.4244 + 0.1174 \times SG_i) \times T_{b,i} + \frac{(0.1441 - 1.0069 \times SG_i) \times 10^5}{T_{b,i}} \quad (3.8)$$

3.4. Simulation Inputs & Outputs

3.4.1. Inputs

This section introduces the basic requirements for the column model. The following table shows the required operational inputs:

Table 3.2 – Operational inputs for simulation

INPUT	DESCRIPTION	TYPE	UNIT
X ₁	Feed Flow Rate	Flow Rate	m ³ /h
X ₂	Feed Temperature	Temperature	°C
X ₃	Stripping Steam Rate	Flow Rate	kg/h
X ₄	Stripping Steam Temperature	Temperature	°C
X ₅	Column Bottom Pressure	Pressure	kg-f/cm ²
X ₆	Column Top Pressure	Pressure	kg-f/cm ²
X ₇	Column Condenser Pressure Drop	Pressure	kg-f/cm ²
X ₈	Kerosene PA-1 Rate	Flow Rate	m ³ /h
X ₉	Kerosene PA-1 Return Temperature	Temperature	°C
X ₁₀	Kerosene PA-2 Rate	Flow Rate	m ³ /h
X ₁₁	Kerosene PA-2 Return Temperature	Temperature	°C
X ₁₂	Diese PA-1 Rate	Flow Rate	m ³ /h
X ₁₃	Diesel PA-1 Return Temperature	Temperature	°C
X ₁₄	Diese PA-2 Rate	Flow Rate	m ³ /h
X ₁₅	Diesel PA-2 Return Temperature	Temperature	°C
X ₁₆	Off Gas Rate	Flow Rate	Nm ³ /h
X ₁₇	HSRN Product Flow	Flow Rate	m ³ /h
X ₁₈	Kerosene Product Flow	Flow Rate	m ³ /h
X ₁₉	Diesel Product Flow	Flow Rate	m ³ /h
X ₂₀	Frac.bottom Product Flow	Flow Rate	m ³ /h
X ₂₁	E-108 Kerosene Strip. Outlet T.	Temperature	°C
X ₂₂	Condenser Outlet Temperature	Temperature	°C

In addition to these operational inputs, the user must also enter the feed characterization inputs and they are TBP distillation information of the feed. Note that this information is obtained by back-blending method, which is discussed in the previous section.

Table 3.3 – Feed Characterization Inputs

TBP (°C)	FEED
5%	T ₁
10%	T ₂
20%	T ₃
30%	T ₄
40%	T ₅
50%	T ₆
60%	T ₇
70%	T ₈
80%	T ₉
90%	T ₁₀
95%	T ₁₁

Since the initial and final boiling point temperatures can be different significantly from that of the actual ones, they are skipped and not entered within the simulation. Moreover, API information of the feed is also skipped by considering the further implementations. In other words, since API information is only available once a day, only TBP distillation information is fed to the simulation and this TBP information is estimated from selected operational variables. The estimation of feed TBP is discussed in detail in the following chapters of the thesis.

3.4.2. Outputs

The simulation application can excessive information about the process. The following table summarizes the outputs of a process simulator:

Table 3.4 – All of the possible simulator outputs

Output	Example
Product Distillation Properties	ASTM D86 Boiling Points
Product Physical Properties	Enthalpy, Heat Capacity, Density, Flash Point, Molecular Weight and so on.
Column Temperature Profile	Temperature of any tray
Column Pressure Profile	Pressure of any tray
Column Vapor Flows	Vapor flow in any tray
Column Liquid Flows	Liquid flow in any tray
Duties	Pump around duties, condenser duties
Internal or external stream flows	Reflux flow rate, pump around return rate, product flow rate and etc.

Even though there are many outputs that user can extract from the simulation, only product qualities will be extracted due to the scope of the thesis. All simulator outputs are given in Appendix I.

Chapter 4

Data-Driven Modelling Techniques Used in the Thesis

The main objective of statistical learning is to propose a description of an unknown relationship or dependency between measurements of objects, namely inputs, and certain properties of these objects, namely outputs. Once a description of this dependency is found, then it is possible to determine the values of output variables for any object of interest. In pattern recognition terminology, this relates to finding an estimation function $f : \mathbb{R}^N \rightarrow \{\mp 1\}$ that can approximately classify or regress new examples based on the past observations.

As mentioned in the introduction section the problem of data-driven empirical modelling is involved in many engineering applications. There are many data sources available in industries that can be used for building data-based empirical models to deduce responses of the systems. Optimization algorithms that are used for the parameter selection process and the statistical measures used for the selection of best model are very crucial and affect the generalization performance of the data-driven models.

In the following sections, theory of support vector machines, artificial neural networks and random forest algorithms will be introduced.

4.1. Support Vector Machines

Vapnik developed the foundations of Support Vector Machines (SVM) and SVMs are gaining popularity due to many attractive features and promising empirical performance. Even though SVMs were developed to solve the classification problems, these techniques have been successfully extended to regression problems recently (Vapnik, 2000). The SVM formulation comprises the Structural Risk Minimization (SRM) principle, which has been shown to be superior, to the conventional Empirical Risk Minimization (ERM) principle that is employed by traditional neural networks (Vapnik, et al., 1999). ERM minimizes the error on the training data whereas SRM minimizes the upper bound on the expected risk. SRM principle equips SVM with a greater ability for generalization performance, which is one of the most important goals in statistical learning.

In a nutshell, Support Vector Machines are one of the most popular classification and regression algorithms for such tasks and are well-known for their strong theoretical foundations, generalization performance and ability to handle high dimensional data. The terminology of SVMs can be confusing, so

the following terminology will be used. The term SVM will refer to both regression and classification, SVM will be used for only classification tasks and SVR will be used for regression tasks. This section continues with an overview of support vector learning, starting with linear SVMs, followed by their extension to the nonlinear case by introducing a kernel idea. This chapter then continues with regression formulations.

4.1.1. The Support Vector Classifier

4.1.1.1. Separable Case

Let our training data to be in the binary classification setting, given in the following form $((x_1, y_1), \dots, (x_n, y_n))$ where x_i 's are the feature vectors (or observations, inputs) and y_i 's are the label vectors of samples and $y_i \in \{-1, +1\}$. Representative data for separable case is given in the following figure.



Figure 4.1. Sample training data for binary classification. Black points are positive examples and white points are negative examples.

The goal of support vector learning is to find a separating hyperplane that creates the biggest margin between the training points for class +1 and -1 (See Figure 4.1). The shortest distance between the positive and negative examples that are closest to the hyperplane is called as the margin of the hyperplane. The intuition behind searching for the hyperplane with a large margin is that a hyperplane with the largest margin should be more resistant to noise than a hyperplane with a smaller margin.

Suppose that all the data given in the Figure 4.1 satisfies the following two constraints

$$w \cdot x_i + b \geq +1 \quad y_i = +1 \quad (4.1)$$

$$w \cdot x_i + b \leq -1 \quad y_i = -1 \quad (4.2)$$

where $|b|/||\mathbf{w}||$ is the perpendicular distance from the separating hyperplane to the origin point, $\mathbf{w}$ is the normal vector to the hyperplane and $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$. It is possible to combine Equation 4.1 and Equation 4.2 into the following form :

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for } \forall i \quad (4.3)$$

The training samples for which Equation 4.3 holds are called support vectors and they lie on the canonical hyperplanes (H_1 and H_2 in Figure 4.2).

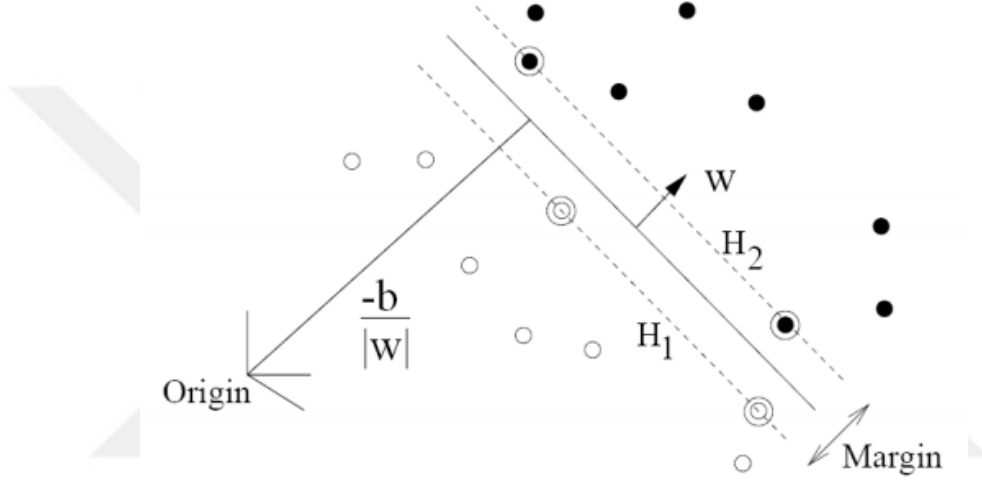


Figure 4.2 - A hyperplane separating binary classes with the maximum margin. The decision boundary is the solid line, while broken lines bound maximal margin of width = $2/||\mathbf{w}||$. The circled examples that lie on the canonical hyperplanes are support vectors.

The margin $2M$ is the distance between canonical hyperplanes H_1 and H_2 and can be calculated as follows

$$Margin = 2M = \frac{|1 - b|}{||\mathbf{w}||} - \frac{|-1 - b|}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||} \quad (4.4)$$

Thus, the separating hyperplane with the maximum margin can be found by solving the following Primal optimization problem:

$$\min_{\mathbf{w}} \quad \tau(\mathbf{w}) = \frac{1}{2} ||\mathbf{w}||^2$$

Subject to

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for } \forall i$$

(4.5)

Since the training data ($\mathbf{x}_i$'s) appears as a dot product ($\mathbf{w} \cdot \mathbf{x}_i$) between vectors and the constraints are easier to handle, it is possible to write Equation 4.5 in the Lagrangian formulation as follows :

$$\min_{\mathbf{w}, b} L_P(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \alpha_i y_i (\mathbf{x}_i \mathbf{w} + b) + \sum_{i=1}^l \alpha_i \quad (4.6)$$

Where $\alpha_i \geq 0$ are Lagrange multipliers for each constraint that are given in Equation 4.5. The objective here is to minimize $L_P(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b and it require that the derivatives of $L(\mathbf{w}, b, \alpha)$ with respect to all α vanish.

The KKT (Karush-Kuhn-Tucker) conditions establish the requirements needed to be satisfied by an optimum solution to a general optimization problem (Karush, 1939). The following conditions must be satisfied by the solutions $\mathbf{w}^*$, b^* and α^* for the given primal problem in Equation 4.6 according to the KKT conditions :

$$\begin{aligned} \frac{\partial L_P(\mathbf{w}^*, b^*, \alpha^*)}{\partial \mathbf{w}} &= w_v - \sum_i \alpha_i y_i x_{iv} = 0 \quad \text{for } v = 1, 2, \dots, d \\ w_v &= \sum_i \alpha_i y_i x_{iv} \quad \text{for } v = 1, 2, \dots, d \end{aligned} \quad (4.7)$$

$$\begin{aligned} \frac{\partial L_P(\mathbf{w}^*, b^*, \alpha^*)}{\partial b} &= - \sum_i \alpha_i y_i = 0 \\ \sum_i \alpha_i y_i &= 0 \end{aligned} \quad (4.8)$$

$$y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad , \forall i \quad (4.9)$$

$$\alpha_i \geq 0 \quad , \forall i \quad (4.10)$$

$$\alpha_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0 \quad , \forall i \quad (4.11)$$

The support vector classifier problem is convex and these KKT conditions are necessary and sufficient for $\mathbf{w}^*$, b^* , α^* to be a solution (Fletcher, 1987). Together these equations (4.7)-(4.11) uniquely characterize the solution to the primal problem. It means that solving the support vector classifier problem is equivalent to finding a solution to the KKT conditions.

1st KKT condition :

Optimal hyperplane is a linear combination of the vectors in the training set :

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i \quad (4.12)$$

2nd KKT condition :

The α_i coefficients of the training examples should satisfy the following :

$$\sum_{i=1}^n \alpha_i^* y_i = 0, \quad \alpha_i^* \geq 0 \quad (4.13)$$

The Lagrange(primal) function given in the Equation 4.6 can be written in dual form. Actually, the support vector classifier problem is solved by its dual definition in practice. In this case, it is desired to maximize Equation 4.6 with respect to α , and minimize with respect to $\mathbf{w}$ and b . By substituting Equation 4.7 and 4.8 into Equation 4.6, we obtain the Lagrangian (Wolfe) dual formulation as follows:

$$\max_{\alpha} \quad L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Subject to (4.14)

$$\forall i \left\{ \begin{array}{l} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{array} \right.$$

One can obtain the coefficients α_i by solving the dual optimization given above. The samples with nonzero coefficients α_i are called “support vectors” and they lie on the edge of the margin. It is also worth keeping in mind that only the samples with $\alpha_i > 0$ effect the support vector solution and thus only support vectors are needed to express the solution $\mathbf{w}$.

The decision function is

$$G(x) = \text{sign}[f(x)] = \text{sign}[\mathbf{w}^T \mathbf{x}_i + b] = \text{sign}\left[\sum_{i=1}^M y_i \alpha_i (\mathbf{x}_i^T \mathbf{x}) + b\right] \quad (4.15)$$

4.1.1.2. Non-Separable Case

Up to this section, the cases where it is possible to linearly separate the feature space that belong to different classes are discussed. However, this support vector classifier formulation will not find a solution if the feature space overlaps. SVM may overfit to the training data even if the data is linearly separable. Even an individual outlier in the dataset can significantly affect the hyperplane. These concerns mentioned above conduce to the development of *soft margin SVMs* (Vapnik, et al., 1995), which can handle linearly non-separable data by introducing non-negative slack variables ξ_i that relax the constraints given in the Equations 4.1 and 4.2 at a cost proportional to the value of this slack variable.

Suppose now that the classes overlap in feature space. One way to deal with the overlap is to still maximize M , but allow for some points to be on the wrong side of the margin. The natural way to modify the constraints in 4.1 and 4.2 is:

$$\forall i \quad \begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 - \xi_i & y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 - \xi_i & y_i = -1 \\ \xi_i \geq 0 \end{cases} \quad (4.16)$$

The idea of this formulation can be explained as follows. The value ξ_i in the constraints given in 4.16 is the proportional amount by which the prediction $f(\mathbf{x})$ is on the wrong side of its margin. These constraints permit some training data samples to lie inside the margin or even cross further among the samples of the opposite class (Figure 4.3). Even though this relaxed constraints lead to flexibility to decrease the effect of outliers, large values for slack variables are not desirable from optimization perspective since large slack variables would result in sub-optimal solutions. The objective function given in Equation 4.5 is then rearranged such that the relaxations are constrained in the objective function:

$$\min_{\mathbf{w}, \xi} \tau(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i$$

Subject to

(4.17)

$$\forall i \quad \begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 - \xi_i & y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 - \xi_i & y_i = -1 \\ \xi_i \geq 0 \end{cases}$$

where the “cost” parameter C replaces the constant in Equation 4.5. The cost coefficient can be considered as a tuning parameter of support vector classifier and is a hyperparameter that specifies the misclassification penalty and is tuned by the user based on the classification task and dataset characteristics. The optimal value for C can be estimated by cross-validation. Interestingly, the leave-one-out cross-validation error can be bounded above by the proportion of support points in the data. The reason beyond it is that leaving out

an observation will not change the solution if it is not a support vector. Hence these observations, being classified correctly by the original boundary, will be classified correctly in the cross-validation process. One should also see that for separable case, $C = \infty$.

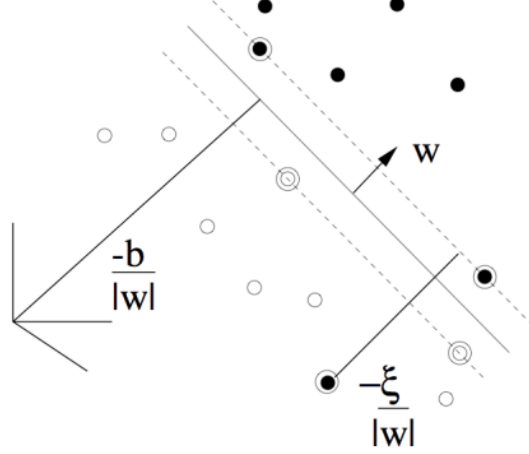


Figure 4.3 - Soft margin support vector classifier

The solution to Equation 4.17 can be shown as follows :

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0$$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad (4.18)$$

The solution is similar to the solution in the separable case. In Equation 4.18 the samples with $\alpha_i \geq 0$ are the support vectors. The dual formulation of Equation 4.17 is :

$$\max_{\alpha} \quad L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

Subject to

$$(4.19)$$

$$\forall i \left\{ \begin{array}{l} \sum_i \alpha_i y_i = 0 \\ C \leq \alpha_i \leq 0 \end{array} \right.$$

The only difference between this dual formulation and the dual formulation for the separable case is existence of extra upper bound of C for the coefficients α_i .

4.1.2. Support Vector Machines and Kernels

4.1.2.1. Moving into Non-linearity

The support vector classifier described so far are discussed in the specific context of learning linear decision boundaries. However, the striking property of SVMs can be fully realized when the linear SVM is allowed for more general decision surfaces. In this section of the report the method that can be used to construct a mapping into a high dimensional feature space by the use of reproducing kernels will be discussed. Lagrangian transformation of the support vector classifier problem includes the training data as a dot product between vectors (See Equation 4.19). Thus, it is possible to use “kernel trick” to allow SVM to generate nonlinear decision boundaries.

The idea of kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner product does not need to be evaluated in the feature space. This provides a way of addressing the curse of dimensionality. Notwithstanding the computation it still depends significantly on the number of training patterns and a large training dataset is required to provide a good data distribution for a high-dimensional problem.

The training data can be mapped into some other Euclidean space $\mathcal{H}$ using a mapping $\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$. The algorithm would now depend on the training data through the dot products in $\mathcal{H}$, that is on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ instead of computing the dot products $(\mathbf{x}_i \cdot \mathbf{x}_j)$ in $\mathbb{R}^d$. There is no need to perform computationally expensive Φ transformation for each instance. We can make the support vector procedure more flexible by enlarging the feature space using basis expansion with a kernel function K that corresponds to a dot product in expanded feature space. An inner product in feature space has an equivalent kernel in input space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (4.20)$$

As can be seen from Equation 4.20, one should only compute the value of K to find dot products in Φ , without mapping $\mathbf{x}_i$ and $\mathbf{x}_j$ from $\mathbb{R}^d \rightarrow \mathcal{H}$.

The Lagrange dual function in Equation 4.19 has the following form now:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (4.21)$$

The new dual formulation with kernel has the same constraints with the constraints given in Equation 4.19. These constraints are $\sum_i \alpha_i y_i = 0$ and $C \leq \alpha_i \leq 0$ for each i .

4.1.2.2. Kernel Functions

The basic idea with nonlinear SVMs is to map training data into a higher dimensional feature space through some mapping $\Phi(x)$ and then construct a separating hyperplane with maximum margin in the input space. The following theory is based upon Reproducing Kernel Hilbert Spaces (RKHS) (Aronszajn, 1950). An inner product in feature space has an equivalent kernel input space as given in Equation 4.20 provided certain conditions hold. To ensure that a kernel function actually corresponds to some feature space, it must satisfy some conditions:

- Symmetric ($K(x, y) = \langle \Phi(x), \Phi(y) \rangle = \langle \Phi(y), \Phi(x) \rangle = K(y, x)$)
- Kernel matrix $\mathbf{K}$ has no non-negative eigenvalues ($\mathbf{K}$ must be positive semi-definite)

$$\text{Note : } \mathbf{K} = (K(x_i, x_j))_{i,j=1}^n$$

If so, K satisfies Mercer's conditions and the kernel represents a legitimate inner product in feature space. Valid kernel functions are given in the following section.

Linear Kernel

The Linear kernel is the simplest kernel function. Kernel algorithms using a linear kernel are often equivalent to their non-kernel counterparts, i.e. KPCA with linear kernel is the same as standard PCA.

$$K(x, x') = \langle x, x' \rangle \quad (4.22)$$

Polynomial Kernel

A polynomial mapping is a popular method for non-linear modelling. The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized.

$$K(x, x') = \langle x, x' \rangle^d \quad (4.23)$$

$$K(x, x') = (\langle x, x' \rangle + 1)^d \quad (4.24)$$

The second kernel is usually preferable since it eliminates problems with the hessian becoming zero.

Gaussian Radial Basis Function Kernel

The kernel is given in the following equation :

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \quad (4.25)$$

Alternatively, this equation can also be implemented using

$$K(x, x') = \exp(-\gamma \|x - x'\|^2) \quad (4.26)$$

The tuning parameter in Equation 4.26 is gamma and it plays a major role in the performance of the kernel. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. In the other hand, if underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noise in training data.

Exponential Kernel

Exponential kernel is also a radial basis function kernel. The exponential kernel is closely related to the Gaussian kernel, with only the square of the norm left out. It produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right) \quad (4.27)$$

Hyperbolic Tangent (Sigmoid) Kernel

The Hyperbolic Tangent Kernel is also known as the Sigmoid Kernel and as the Multilayer Perceptron (MLP) kernel. The Sigmoid Kernel comes from the Neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons.

$$K(x, x') = \tanh(\gamma\langle x, x' \rangle - \theta) \quad (4.28)$$

It is interesting to note that a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network. This kernel was quite popular for support vector machines due to its origin from neural network theory. Also, despite being only conditionally positive definite, it has been found to perform well in practice.

There are two adjustable parameters in the sigmoid kernel, the slope gamma and the intercept constant theta. A common value for gamma is $1/N$, where N is the data dimension.

Laplacian Kernel

The Laplace Kernel is completely equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter. Being equivalent, it is also a radial basis function kernel.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{\sigma}\right) \quad (4.29)$$

It is important to note that the observations made about the sigma parameter for the Gaussian kernel also apply to the Exponential and Laplacian kernels.

4.1.3. Support Vector Regression

In this section it is going to be shown how SVMs can be adapted for regression with a quantitative response, in ways that inherit some of the properties of the SVM classifier. Support vector machines can be used for regression purposes by the introduction of an alternative loss functions (A.J.Smola, 1996) and the loss function must be modified to comprise a distance measure. The following figure shows possible loss functions.

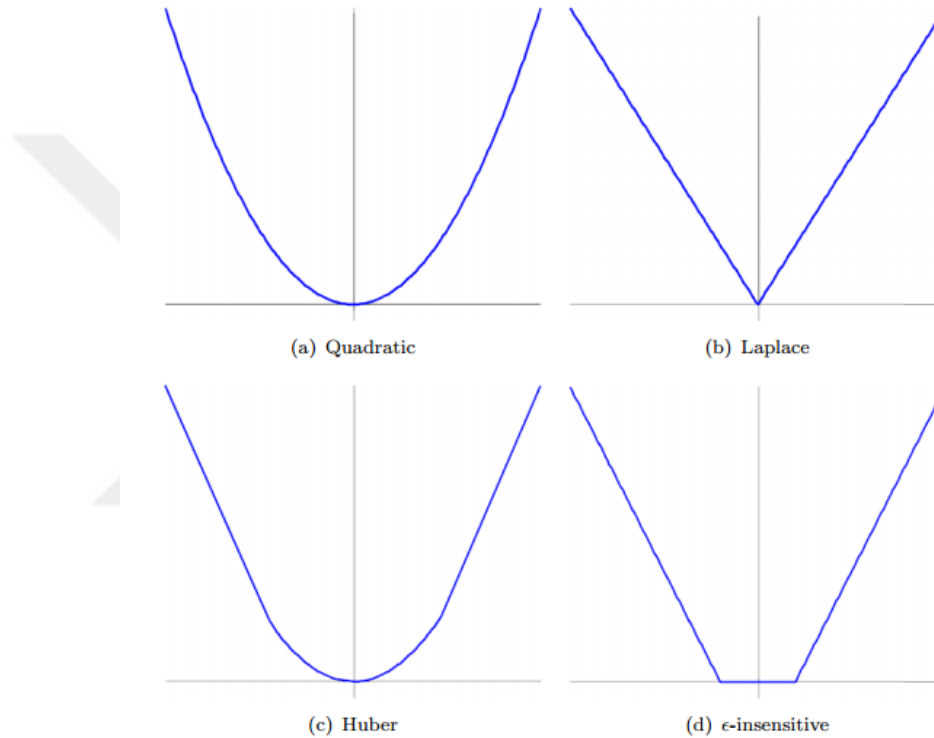


Figure 4.4 - Loss functions for regression purpose

The first loss function in Figure 4.4 (quadratic loss function) is usually used as the conventional least squares error criterion. The second loss function (laplacian loss function) is a loss function that is less sensitive to outlier instances than the quadratic loss function. If the underlying distribution of the data is unknown, one can use the Huber loss function as a robust loss function. However, these three loss functions mentioned will not produce sparseness in the support vectors. Vapnik proposed the ϵ – *insensitive* loss function as an approximation to Huber’s loss function that enables a sparse set of support vectors to be obtained (Huber, 1964).

4.1.3.1. Regression Problem Statement and Some Basic Principles

The regression problem can be stated as follows, given a set of training data D ,

$$D = \{ (\mathbf{x}_i, y_i) \mid i = 1, 2, \dots, l \}, \quad \mathbf{x} \in \mathbb{R}^l, \quad y \in \mathbb{R} \quad (4.30)$$

where $\mathbf{x}_i$ are input vectors and associated targets or labels are y_i . Our goal is to fit a function $f(\mathbf{x})$ that approximates the relation between the data set points to infer the output y for a new input data point $\mathbf{x}$. Any regression algorithm has a loss function $\mathcal{L}(y, f(\mathbf{x}))$ that describes how the estimated function is deviated from the true function. Some forms of loss functions are given in the Figure 4.4. For the sake of simplicity, the following discussion begins by describing the case of linear functions f , taking the form:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \quad (4.31)$$

where $\mathbf{x} \in X$, X is the input space, $b \in \mathbb{R}$ is the bias (constant term), and $\langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w} \cdot \mathbf{x}$ is the dot product of the vectors $\mathbf{w}$ and $\mathbf{x}$. As stated before, our goal is to fit a flat function to the data points for linear case. Flatness in Equation 4.31 means that one look for a small $\mathbf{w}$. In order to guarantee this flatness, one can minimize the norm $\|\mathbf{w}\|^2$. This regression problem can be handled with the following convex optimization problem:

$$\begin{aligned} \text{minimize} \quad & \Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{Subject to} \quad & \begin{cases} y_i - (\langle \mathbf{w}, \mathbf{x} \rangle + b) \leq \varepsilon \\ (\langle \mathbf{w}, \mathbf{x} \rangle + b) - y_i \leq \varepsilon \end{cases} \end{aligned} \quad (4.32)$$

The assumption related with the constraints given in Equation 4.32 is that a function $f(\mathbf{x})$ actually exists and it approximates all pairs of $(\mathbf{x}_i, y_i)$ with ε precision, or in other words, that convex optimization problem is feasible. However, to prevent this noise one can want to allow for some errors. To allow some errors, slack variables ζ_i, ζ_i^* can be introduced to deal with possible infeasible constraints of the optimization problem given in Equation 4.32. Then the new formulation can be stated as follows :

$$\begin{aligned} \text{minimize} \quad & \Phi(\mathbf{w}, \zeta) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l (\zeta_i + \zeta_i^*) \\ \text{Subject to} \quad & \begin{cases} y_i - (\langle \mathbf{w}, \mathbf{x} \rangle + b) \leq \varepsilon + \zeta_i \\ (\langle \mathbf{w}, \mathbf{x} \rangle + b) - y_i \leq \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \geq 0 \end{cases} \end{aligned} \quad (4.33)$$

where the constant C is a pre-specified value that determines the trade-off between the flatness of $f(\mathbf{x})$ and the amount up to which errors (or deviations) larger than ε are permitted. Hence, C corresponds to dealing

with the so-called ε – *insensitive loss function* (See Equation 4.34). Slack variables ζ_i, ζ_i^* represent upper and lower constraints on the outputs of the system, respectively.

$$\varepsilon - \text{insensitive loss function}, L_\varepsilon(x) = \begin{cases} 0 & \text{for } |f(x) - y| < \varepsilon \\ |f(x) - y| - \varepsilon & \text{otherwise} \end{cases} \quad (4.34)$$

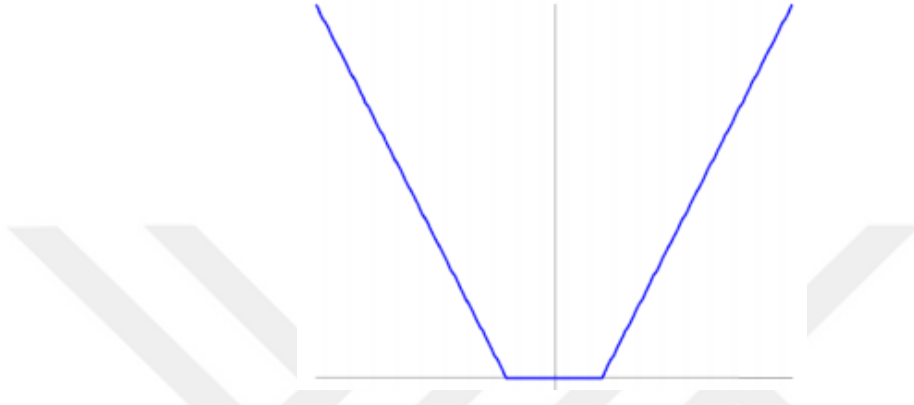


Figure 4.5 - ε -insensitive loss function

The minimization problem given in Equation 4.33 is called the *primal* objective function. We can construct a Lagrange function from the primal objective function and the corresponding constraints by introducing a dual set of variables. Lagrange function has a saddle point with respect to the primal and dual variables at the solution (R.J.Vanderbei, 1997). The primal objective function given in Equation 4.33 with its constraints are first transformed to the Lagrange function as shown below:

$$L = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l (\zeta_i + \zeta_i^*) - \sum_{i=1}^l \alpha_i (\varepsilon + \zeta_i - y_i + (\langle \mathbf{w}, \mathbf{x} \rangle + b)) - \dots \quad (4.35)$$

$$\dots \sum_{i=1}^l \alpha_i^* (\varepsilon + \zeta_i^* + y_i - (\langle \mathbf{w}, \mathbf{x} \rangle + b)) - \sum_{i=1}^l \lambda_i \zeta_i - \sum_{i=1}^l \lambda_i^* \zeta_i^*$$

In Equation 4.35, L is the Lagrangian and $\alpha_i, \alpha_i^*, \lambda_i, \lambda_i^*$ are the Lagrange multipliers. The dual variables in this equation must be greater than or equal to zero ($\alpha_i, \alpha_i^*, \lambda_i, \lambda_i^* \geq 0$). The partial derivatives of Lagrangian with respect to the primal variables ($\mathbf{w}, b, \zeta_i, \zeta_i^*$) have to vanish for optimality due to the saddle point condition:

$$\frac{\partial L}{\partial b} = 0 = \sum_{i=1}^l (\alpha_i^* - \alpha_i) \quad (4.36)$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 = \sum_{i=1}^l (\alpha_i^* - \alpha_i) \mathbf{x}_i \quad (4.37)$$

$$\frac{\partial L}{\partial \zeta_i^{(*)}} = 0 = C - \alpha_i^{(*)} - \lambda_i^{(*)} \quad (4.38)$$

In Equation 4.38, $\alpha_i^{(*)}$ refers to α_i and α_i^* . Equations 4.36-37-38 are substituted into Equation 4.35 will yield the following dual optimization problem:

$$\begin{aligned} & \text{maximize} -\frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \varepsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) + \sum_{i=1}^l \mathbf{x}_i (\alpha_i - \alpha_i^*) \quad (4.39) \\ & \text{Subject to} \sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C] \end{aligned}$$

or alternatively,

$$\bar{\alpha}, \bar{\alpha}^* = \operatorname{argmin} \frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \varepsilon \sum_{i=1}^l (\alpha_i + \alpha_i^*) + \sum_{i=1}^l \mathbf{x}_i (\alpha_i - \alpha_i^*)$$

The so-called *Support Vector Machines regression expansion*, i.e. $\mathbf{w}$ can be completely described as a linear combination of the training samples $\mathbf{x}_i$, is as follows :

$$\bar{\mathbf{w}} = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \mathbf{x}_i \quad (4.40)$$

$$f(\mathbf{x}) = \sum_{i=1}^l (\alpha_i - \alpha_i^*) \langle \mathbf{x}_i, \mathbf{x} \rangle + b \quad (4.41)$$

$$\bar{b} = -\frac{1}{2} \langle \bar{\mathbf{w}}, (\mathbf{x}_r + \mathbf{x}_s) \rangle \quad (4.42)$$

In Equation 4.42 $\mathbf{x}_r$ and $\mathbf{x}_s$ are the support vectors (i.e. any vector which has nonzero value of either α_i or α_i^* respectively).

The complexity of a function's representation by support vectors does not depend of the dimension of the input space X and it depends only on the number of support vectors. In addition to that, the complete algorithm can be described in terms of dot products between the data. This will come in handy for the formulation of a non-linear extension with kernel trick.

The Karush-Kuhn-Tucker (KKT) conditions are the essential for the Lagrangian solution. The product between dual variables and constraints has to vanish at the solution point according to KKT conditions.

$$\alpha_i (\varepsilon + \zeta_i - y_i + \langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 0$$

$$\alpha_i^* (\varepsilon + \zeta_i + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b) = 0 \quad (4.43)$$

$$(C - \alpha_i) \zeta_i = 0 \text{ and } (C - \alpha_i^*) \zeta_i^* = 0 \quad (4.44)$$

Some conclusions can be drawn from KKT conditions given in Equation 2.43 and 2.44.

- Only instances (x_i, y_i) with corresponding $\alpha_i^{(*)} = C$ lie outside the ε -insensitive tube.
- There can never be a set of dual variables $\alpha_i, \alpha_i^* = 0$, which are both simultaneously nonzero.
- Only for $|f(\mathbf{x})| \geq \varepsilon$ the Lagrange multipliers may be nonzero, or equivalently, α_i and α_i^* vanish for all samples inside the ε -tube. For $|f(\mathbf{x})| < \varepsilon$, α_i, α_i^* has to be zero such that KKT conditions are satisfied.

The training instances with non-zero coefficients, α_i, α_i^* , are called *Support Vectors*.

Quadratic Loss Function

The quadratic loss function is given in the Figure 4.6 and it has the following form :

$$L_{quad} (f(\mathbf{x}) - y) = (f(\mathbf{x}) - y)^2 \quad (4.45)$$

And the solution is given by,

$$\bar{\alpha}, \bar{\alpha}^* = \operatorname{argmin} \frac{1}{2} \sum_{i,j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle x_i, x_j \rangle + \sum_{i=1}^l (\alpha_i - \alpha_i^*) y_i - \frac{1}{2C} \sum_{i=1}^l (\alpha_i^2 + (\alpha_i^*)^2) \quad (4.46)$$

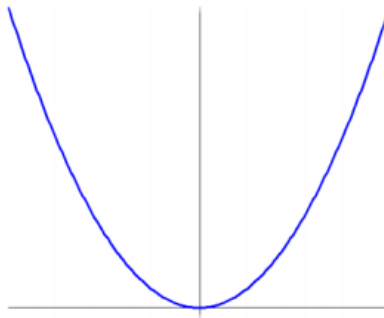


Figure 4.6 - Quadratic Loss Function

KKT conditions and Equation 4.47 can be used to simplify this optimization problem.

$$\bar{\alpha}_i \bar{\alpha}_i^* = 0 \quad , i = 1, 2, 3, \dots, l \quad (4.47)$$

The resultant optimization problem is :

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2} \sum_{i,j=1}^l \beta_i \beta_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^l \beta_i y_i + \frac{1}{2C} \sum_{i=1}^l \beta_i^2 \\ \text{Subject to} \quad & \sum_{i=1}^l \beta_i = 0 \end{aligned} \quad (4.48)$$

The regression function is the same with the one found by ε -insensitive loss function :

$$f(x) = \langle \mathbf{w}, \mathbf{x} \rangle + b \quad (\text{Recall 4.31})$$

$$\bar{\mathbf{w}} = \sum_{i=1}^l \beta_i \mathbf{x}_i \quad (4.49)$$

$$\bar{b} = -\frac{1}{2} \langle \bar{\mathbf{w}}, (\mathbf{x}_r + \mathbf{x}_s) \rangle \quad (\text{Recall 4.21})$$

4.1.4. Nonlinear Regression with Kernel Trick

For some problems, a non-linear model can be required due to the system to be modelled. In such a case, a non-linear mapping can be used to map the data into a high dimensional feature space where again linear regression is performed. The kernel trick is employed to address the curse of dimensionality as in the case of classification problems. An important note here is that in the nonlinear setting, the optimization problem corresponds to finding the flattest function in feature space, not in input space. The non-linear support vector regression solution using ε -insensitive loss function is given by the following equation:

$$\max_{\alpha, \alpha^*} \quad \sum_{i=1}^l \alpha_i^* (y_i - \varepsilon) - \alpha_i (y_i + \varepsilon) - \frac{1}{2} \sum_{i,j=1}^l (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j) \quad (4.50)$$

With constraints,

$$0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, 2, \dots, l$$

$$\sum_{i=1}^l (\alpha_i - \alpha_i^*) = 0 \quad (4.51)$$

The solution of this optimization problem gives α_i and α_i^* . The regression function is given by,

$$f(x) = \sum_{\substack{\text{Support} \\ \text{Vectors}}} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(\mathbf{x}_i, \mathbf{x}) + \bar{b} \quad (4.52)$$

where,

$$\langle \bar{\mathbf{w}}, \mathbf{x} \rangle = \sum_{i=1}^l (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x}_j) \quad (4.53)$$

$$\bar{b} = -\frac{1}{2} \sum_{i=1}^l (\alpha_i - \alpha_i^*) (K(\mathbf{x}_i, \mathbf{x}_r) + K(\mathbf{x}_i, \mathbf{x}_s)) \quad (4.54)$$

If the chosen kernel contains a bias term, the constraint $\bar{b}$ may be dropped and the regression function becomes then,

$$f(x) = \sum_{\substack{\text{Support} \\ \text{Vectors}}} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(\mathbf{x}_i, \mathbf{x}) \quad (4.55)$$

One can replace the dot product with a kernel function to obtain the optimization criteria for the other loss functions. The support vector solution can be sparse if the ε -insensitive loss function is used. However, for the quadratic and Huber lost functions all the data points will be support vectors. It is interesting to note that the quadratic loss function gives a solution equivalent to ridge regression, or zeroth order regularization, where the regularization parameter is $\lambda = \frac{1}{2c}$.

4.2. Artificial Neural Networks

Mathematical models are critical in refinery engineering, both in the design and operation of oil refineries. These models are needed for analysis, control and optimization purposes in refinery operation. Rigorous steady-state refinery plant models are required for real time optimization (RTO), which is the continuous evaluation and adjustment of process operating conditions to optimize the economic productivity and process efficiency subject to some constraints. Robust process models are critical for a successful implementation of RTO. Currently most RTO applications make use of rigorous first principles mathematical models (theoretical models), which are not only complex and costly to develop, but also awkward and difficult to maintain because of skill and time requirements. The high computation time needed in solving the FP mathematical models is one of the major problems for on-line applications. This leads the academicians and technical people to be in search of other types of suitable models, such as artificial neural network (ANN) and support vector machines (SVMs).

Artificial neural network models had generated much interest in the chemical process engineering community since the late eighties as an alternate approach to model a chemical process. An artificial neuron is a computational model inspired in the biological neurons and is a black box model that consists of layers of nodes with non-linear basis functions and weighted connections that link the nodes. The input space is mapped to the output space using the nodes in the layers and weights. It is also interesting to note that multilayer feedforward ANNs were mathematically proven to be a universal approximator (Hornik, et al., 1989). One should, however, keep in mind that the resulting models can only be good as the data provided to the network since this modelling technique is data-driven.

In this section of the thesis the scope is to give information about Artificial Neural Networks (ANNs).

4.2.1. The Artificial Neuron Network Models Fundamentals

The most elementary unit of a neural system is called a neuron in both artificial networks and biological systems. As mentioned before, an artificial neuron is a computational model inspired in the biological neurons. Biological neurons receive signals through synapses located on the dendrites or membrane of the neuron (See Figure 4.7). In other words synapses are responsible for transmitting information (stimulus in terms of neuroscience science terminology). Several information can cumulate in a neuron since a neuron can be connected to many other neurons.

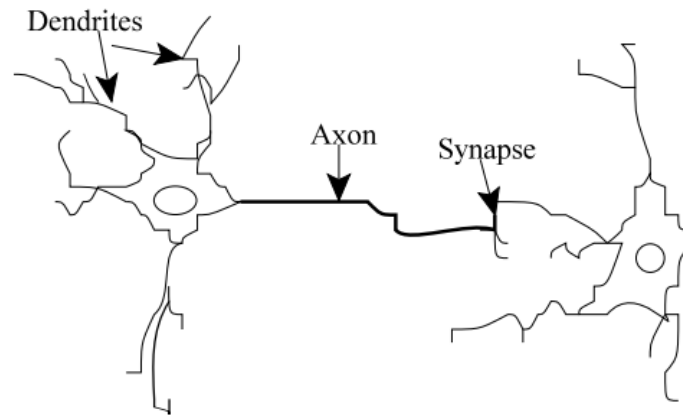


Figure 4.7. Biological neuron

One can consider the stimulus as the incoming signal x_i and the synapses as the connections w_i . In practice, w_i 's are referred as weights that scale the incoming inputs according to their importance. If the signals received are strong sufficiently, in other words, surpass a certain threshold, the neuron is activated and emits a signal through the axon. For an ANN, these weighted inputs accumulate inside the artificial neuron and some function of the sum is given as an output, y . This function of the sum is called *activation function*. This output value (or signal) might be sent to another synapse, and might activate other neurons. In general there is also a bias (threshold value) term, τ , for each neuron.

Artificial neurons are inspired from the structure of real neurons and for the modelling of artificial neurons, the functions of natural neuron components like (synapses, axon and so on) are imitated. Artificial neurons consist of inputs that are multiplied by weights, and then computed by a mathematical functions that determines the activation of the neuron. Output of the neuron is computed by another function, or it can be identity. Artificial neural networks combine all neuron to process the information. To make the input stronger, a weight of an artificial neuron must be higher. On the other side, these weights can also be negative, if so it is possible to say that the signal is inhibited by the negative weights.

It may be quite computationally costly to find all the necessary weights if we have a network of hundreds or thousands of neurons. However, there are such algorithms that can adjust the weights of the network to obtain the desired output from the network and this process of adjusting the weights is called as learning or training.

In mathematical terms, the output of a single artificial neuron can be calculated as follows: (Detienne et al., 2003):

$$y_i = g_i(n_i) = g\left(\sum_{j=1}^K w_{ji}x_j + \theta_i\right) \quad (4.56)$$

The following figure shows a simple neural network structure.

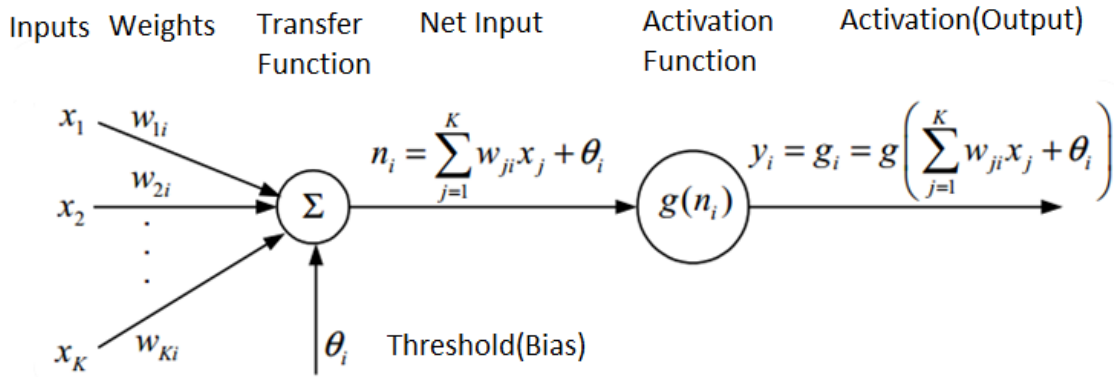


Figure 4.8 - A simple NN structure

The figure below shows a more complicated network with K input neurons, one layer with 3 neuron and 2 output neurons.

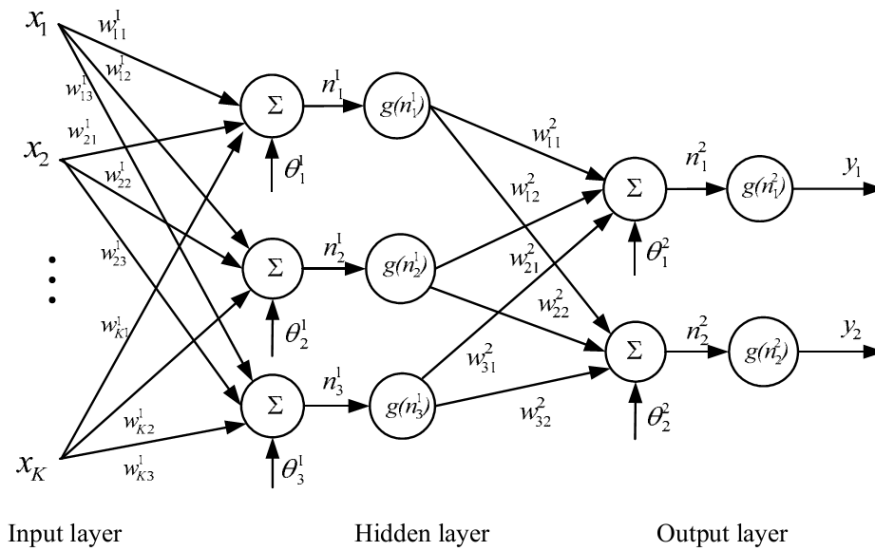


Figure 4.9 - A Neural Network structure with K input neuron, 1 hidden layer with 3 neurons and 2 output neurons

4.2.2. Types of Activation Functions

Activation functions are represented as $g(n_n^m)$ as indicated in Figure 4.9, where m denotes the layer number (1 for first hidden layer, 2 for second layer if exists, and so on. If there is only 1 hidden layer, 2 denotes for the output layer) and n denotes the neuron number. The activation function of a node defines the output of that node given an input or set of inputs. The most common activation functions are the threshold function, the piecewise linear function and the sigmoid function.

Threshold function (Step Function)

The formula for this activation function is,

$$g(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases} \quad (4.57)$$

The output value is a certain value, say O1 if the net input is greater than a certain threshold and O2 if the net input is less than a certain threshold. For a perceptron, O1 is 1 and O2 is 0 and threshold value is zero.

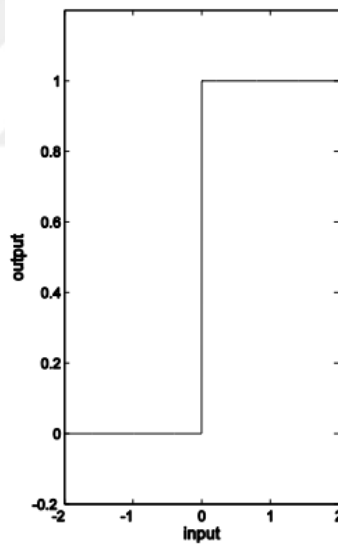


Figure 4.10 - Threshold activation function of a perceptron

These kinds of activation functions are usually used for binary classification tasks.

Piecewise Linear Function

The following function refers to this function,

$$g(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ x & \text{if } -0.5 \leq x < 0.5 \\ 0 & \text{if } x < -0.5 \end{cases} \quad (4.58)$$

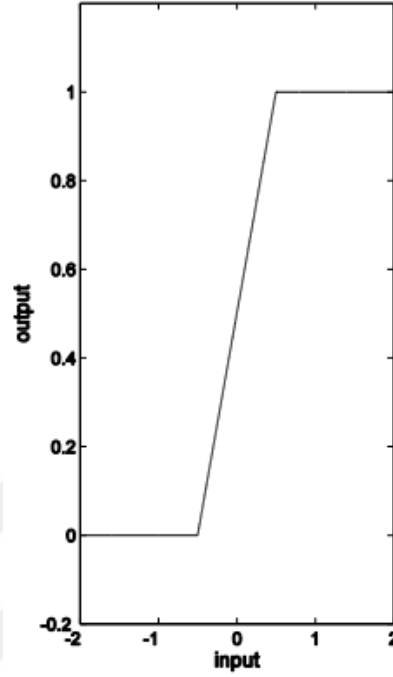


Figure 4.11. Piecewise Linear Function

Sigmoid and Logistic Function

The sigmoid function will only produce positive numbers between 0 and 1. The sigmoid activation function is most useful for training data that is also between 0 and 1. They are often used in networks to introduce nonlinearity in the model.

$$g(x) = \frac{L}{1 + e^{-\beta(x-x_0)}} \quad (4.59)$$

Where β is a slope parameter. This function is called logistic function. L is the maximum value of the function can take. The parameter β controls how steep the change from the minimum to the maximum value is. x_0 controls where on the x axis the growth should be. Assume that $x = x_0$, in such a case you end up with $g(x_0) = \frac{L}{2}$, which is the midpoint of the growth. If $L = 1, x_0 = 0$, then the function becomes “log-sigmoid function” (or simply “sigmoid function”) since a sigmoid can also be constructed using hyperbolic tangent function, in which case it would refer to a “tan-sigmoid function”. It is quite similar to step function, but it has an additional property of having a region for uncertainty. Sigmoids have popularity in neural networks because they satisfy a property between the derivative and itself such that it is computationally easy to perform.

$$\frac{d}{dx} \text{sig}(x) = \text{sig}(x)(1 - \text{sig}(x)) \quad (4.60)$$

Where $\text{sig}(x) = 1/(1 + e^{\beta x})$.

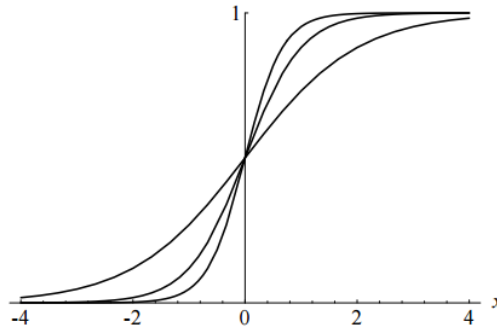


Figure 4.11 – Three sigmoids (for $\beta = 1$, $\beta = 2$, and $\beta = 3$)

4.2.3. Training the Neural Net

In training process of the neural nets, the correct value for each observation is known (supervised training). Thus, it is possible to compare the network's calculated values for the output nodes to these correct values and calculate an error term for each node (the "Delta Rule"). Adjusting the network weights in the hidden layers using these error terms is simply called as training a neural net. There are many algorithms to train a neural network model. The most popular neural network algorithm is back-propagation algorithm proposed in the 1980's.

Backpropagation algorithm is an abbreviation for "backward propagation of errors" and is used in conjunction with gradient descent optimization technique. In this algorithm, the gradient of a loss function with respect to the weights in the network is calculated so that optimization method can update the weights using these gradients in attempt to minimize the loss function. It is a generalization of the delta rule to multi-layered feedforward networks, made possible by using the chain rule to iteratively compute gradients for each layer. The algorithm can be decomposed in the following steps:

- i) Feed-propagation computation
- ii) Backpropagation to the output layer
- iii) Backpropagation to the hidden layers
- iv) Weight updates

The backpropagation algorithm can be summarized as follows:

Algorithm 4.1 : Backpropagation Algorithm for ANN models

-
- Initialize all weight to small random numbers. Until satisfied, do
 - For each training observation, do
 1. Input the training observation to the network and compute network outputs (Feed-propagation computation)
 2. For each output unit k ,

$$\delta_k^l \leftarrow o_k^l(1 - o_k^l)(y_k^l - o_k^l)$$
 3. For each hidden node h

$$\delta_h^l \leftarrow o_h^l(1 - o_h^l) \sum_{k \in \text{outputs}} w_{h,k} \delta_k^l$$
 4. Update each network weight $w_{i,j}$

$$\Delta w_{i,j}^l = \eta \delta_j^l o_i^l$$

$$w_{i,j} := w_{i,j} + \Delta w_{i,j}^l$$
-

Where, y_k is the observed value for k^{th} output label, $o_{k(h)}$ unit output obtained by forward propagation (if i is input variable, $o_i = x_i$), $w_{i,j}$ is the weight from i^{th} node to j^{th} node.

A number of algorithms have been developed to train the nets except back-propagation, such as the conjugate gradient method, steepest descent method, and the levenberg-marquardt algorithm. However, in practice, back-propagation algorithm often works well compared to the other algorithms. On the other hand, it requires some values for the learning rate and usually an optional momentum term. To handle this requirement, resilient back propagation (Rprop) algorithm can be used to train a neural network, which is similar to back-propagation algorithm, but it has two main advantages over back-propagation: First, the training time is much less than that of the back-propagation and secondly it does not require to specify any free parameter values. A small learning rate which, along with the magnitude of the gradient, is often used in regular back-propagation and this effects the weight delta in a training iteration. Hence, it is not likely to miss the optimal weights, but it leads to longer training durations. Rprop algorithm does not use the magnitude of the gradient to determine a weight delta; instead, it uses only the sign of the gradient. Moreover, Rprop maintains separate weight deltas for each weight and bias, and adapts these deltas during training. The main disadvantage of this algorithm is that it is a more complex algorithm to implement compared to back propagation.

The original version of the Rprop algorithm was published in a 1993 paper, "A Direct Adaptive Method for Faster Back Propagation Learning: The Rprop Algorithm," by M. Riedmiller and H. Braun (Riedmiller, et al., 1993). This algorithm is implemented in "neuralnet" package in R. In this thesis, back-propagation and resilient back-propagation algorithms are used for supervised learning purposes.

4.3. Random Forests Models

In this section, we describe random forest models, which is produced by multiple trees to yield a single consensus prediction. In order to understand these models clearly, we first introduce tree-based learning methods for regression, involving stratifying or segmenting the input space into a number of simple regions. The set of splitting rules used to segment the feature space is discussed, as well. After that, it is shown that combining a large number of trees can often leads to improvements in prediction accuracy at the expense of loss interpretation for the model.

4.3.1. Decision Tree Basics

The following graph shows a sample tree for a class and the corresponding partitions of regions.

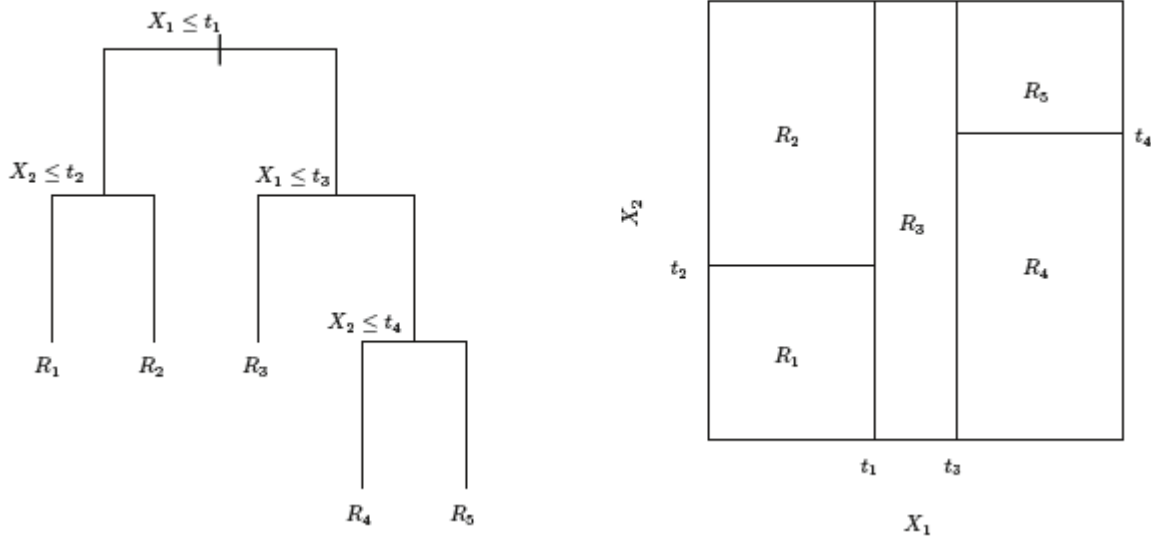


Figure 4.12 – Sample regression tree and its corresponding region partition (James et al., 2009)

In this figure, $R_1, R_2, \dots, R_5$ are called as *terminal nodes* or *leaves* of the tree. These regions can be written as $R_1 = \{X | X_1 < t_1, X_2 < t_2\}$, $R_2 = \{X | X_1 < t_1, X_2 \geq t_2\}$, $R_3 = \{X | t_1 \leq X_1 < t_3\}$, $R_4 = \{X | X_1 \geq t_3, X_2 < t_4\}$, $R_5 = \{X | X_1 \geq t_3, X_2 \geq t_4\}$. The notation $\{X | X_j < t\}$ means the region of input space in which X_j takes on a value less than t . As can be seen from this figure, decision trees are typically drawn upside down, in sense that the leaves are at the bottom of the tree. Internal nodes are the points where the input space is split. In this figure there are 4 internal nodes.

4.3.2. Building a Regression Tree

Basically, there are two steps in regression tree building process:

1. Division of input space : Input variables $X_1, X_2, \dots, X_p$ are split into J distinct and non-overlapping regions $R_1, R_2, \dots, R_J$.
2. The same prediction is yielded for all observations that falls into the region R_j and it is the mean response values for the training samples in R_j .

Even though it is possible to have any shape for division of input space, for ease of interpretation of the model, these regions are chosen as high-dimensional rectangles or boxes. In the division of input or feature space, the ultimate goal is to seek out for the regions $R_1, R_2, \dots, R_J$ that minimize the RSS given by the following equation:

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \quad (4.61)$$

where $\hat{y}_{R_j}$ is the mean response for the training samples within the j^{th} box. Since it is infeasible to consider every possible partition of input space into J boxes, *recursive binary splitting* method is applied. This method is *top-down, greedy* approach due to the following reasons. The approach starts at the top of the tree and successively splits the input space further down via two new branches at each split so that it is called as *top-down* approach. It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Recursive Binary Splitting

In the first step of this technique, we seek the value of j and s that minimize the following equation

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \quad (4.62)$$

where, $R_1(j, s) = \{X | X_j < s\}$ and $R_2(j, s) = \{X | X_j \geq s\}$. Basically, we are considering all predictors $X_1, X_2, \dots, X_p$ and all possible values of the cut point s for each of the predictors, and then chose the predictor and cut point such that the resulting tree has the lowest RSS.

In the next step, we follow the similar methodology, namely looking for the best prediction and best cut point s to split the training data to minimize the RSS within each of the resulting regions. At that step, instead of splitting the whole input space, only one of the two previously identified regions are split. At that point, we have three regions. Again, one of these three regions are split further to minimize the RSS. The

process continues until a stopping criterion is reached; for instance, the process can be stopped at a time when no region contains more than three observations. It is now possible to predict the response for any test observation using the mean of the training observations in that regions to which that test sample belongs. This process may lead to over-fitting and bad generalization performance due to the fact that the resulting tree may be too complex. To deal with such a problem, an approach in which a large tree T_0 is grown and *pruned* back in order to obtain a *subtree* is applied to the previous process. Given a subtree, it is possible to estimate the test error by using cross-validation or the validation set approach. Unfortunately, it can be cumbersome to do it for every possible subtree. Hence, it is needed to select a small set of subtrees for consideration and *cost complexity pruning method* is one of the ways to do this. A sequence of trees indexed by a nonnegative tuning parameter α is considered, instead of considering all possible subtrees. For each value of this tuning parameter, there corresponds a subtree $T \subset T_0$ such that the following is as small as possible.

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \quad (4.63)$$

In this equation $|T|$ is the number of terminal nodes of tree T , R_m is the box corresponding to the m^{th} terminal node, and $\hat{y}_{R_m}$ is the mean of the training observations in R_m . Basically, tuning parameter α controls the model complexity and generalization. As it increases, there is a cost for having a tree with more terminal nodes so that complexity of the model is decreased and over fitting is decreased to some extent. This tuning parameter can be obtained by using a cross-validation. The algorithm of tree building model is summarized in the following:

Algorithm 4.2 : Tree Building Algorithm

1. Until each terminal node has fewer than some minimum number of observations, use *recursive binary splitting* to grow a large tree.
2. Obtain a sequence of best subtrees as a function of tuning parameter α by applying *the cost complexity pruning* method.
3. Apply K-fold cross-validation to choose α . The way to do this is listed below:
 - 3.1. Divide the training data into K-fold.
 - 3.2. Repeat step 1 and 2 for each k^{th} fold of the training data. Evaluate the mean squared prediction error on the data in the left-out k^{th} fold, as a function of α .
 - 3.3. Average the results for each value of α and pick the one with the minimum average CV error.
4. The final tree is the subtree from Step 2 that corresponds to the chosen value of α , which is found at the previous step.

4.3.3. From Bagging to Random Forests

In bagging technique, B different bootstrapped training observations are generated and then $\hat{f}^{*b}(x)$ prediction is obtained by using b^{th} bootstrapped training set, and finally all predictions are averaged to obtain bagged estimation, $\hat{f}_{bag}(x)$.

$$\hat{f}_{bag}(x) = \frac{\sum_{b=1}^B \hat{f}^{*b}(x)}{B} \quad (4.64)$$

where, B is the number of decision trees. In this technique, trees are grown deep, but are not pruned. Hence each individual tree has high variance but low variance, averaging these B trees reduces variance. The number of samples/trees, B, is a free parameter. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample x_i , using only the trees that did not have x_i in their bootstrap sample.

Random forest is quite similar to bagging, but now a random sample of m predictors is chosen as split candidates from the full set of p predictors in building these B decision trees. Again, a number of decision trees are built with bootstrapped training observations. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and in general $m \approx \sqrt{p}$.

Assume that there is a one very strong input variable in the data set. Then most or all of the trees will use this variable in the top split in bagging technique. Consequently, all of the bagged trees will be similar so than the predictions from the bagged trees will be highly correlated. Random forest handles this

problem by forcing each split to consider only a subset of the predictors. On average $(p - m)/p$ of the splits will not use this strong predictors. This advantage leads to decorrelating the trees, and thereby the final estimation will be more reliable. At the same time, random forest will not over fit if the number of total trees are increased a lot.

In a nutshell, random forest makes the modelers have improved version of bagged trees by way of a small trick that *decorrelates* the trees.

4.3.4. Variable Scoring with Random Forest Technique

Even though random forest models is commonly used as a predictor function, many reports claim that random forest variable importance measure (VIM) as a means to find a ranking of important variables. In the random forest framework, the increase in mean of the error of a tree (Mean squared error for regression and misclassification rate for classification) in the forest when the observed values of the variable of interest are randomly permuted in the out-of-bag (OOB) samples is the commonly used scoring measure of importance of a given variable. *Permutation importance indices* term can also be used for such random forest variable importance (VI) in opposition to total decrease of node impurity, which introduced in the seminal book by Breiman et al (Breiman, 2001). The permutation importance measure is also introduced by Breiman (Breiman, 2001). Permutation importance for variable X_i is measured such that all values of X_i is permuted and the difference in prediction accuracy caused by the permutation is then used to indicate the importance for this variable. It is highly probable that the prediction accuracy will not be affected by permuting the values of this variable if the variable consists of purely random noise.

This permutation importance index is calculated as follows. Let $\mathcal{B}^t$ denote the OOB samples for a tree t and $L(T_t(\mathbf{x}_i), y_i)$ denote the prediction accuracy at the i^{th} training sample. Therefore, the variable importance (VI) for variable X_j in tree t is defined as

$$VI^{(t)}(X_j) = \sum_{i \in \mathcal{B}^t} L(T_t(\mathbf{x}_i), y_i) - L(T_t(\mathbf{x}_{i, \pi_j}), y_i) \quad (4.65)$$

where, $\mathbf{x}_{i, \pi_j} = (\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i, \pi_j(i)j}, \mathbf{x}_{i, j+1}, \dots, \mathbf{x}_{i,p})$ and where π_j is a random permutation of n integers. In regression settings the prediction accuracy $L(\hat{y}, y)$ is defined as the root mean squared error (RMSE). The variable importance measure for variable X_j is computed as the sum of importances over all trees in the forest as follows:

$$VI(X_j) = \frac{\sum_{t \in B} VI^{(t)}(X_j)}{\text{total number of tree}} \quad (4.66)$$

Another variable ranking parameter is Mean Decrease Impurity (MDI). Random forest consist of many decision trees and every node in decision trees is a condition on a single variable, designed to split the dataset into two (left and right) so that similar response values (or classes for classification) end up in the same set. The measure depends on which optimal condition is chosen is called impurity and for regression trees it is variance. Thus, it is possible to calculate how much each variable decreases the weighted impurity in a tree in training process. For a random forest, the impurity decrease from each variable can be averaged and ranked according to this measure. In other words, *Mean Decrease Impurity* (MDI) can also be used to measure the importance measure for variable X_j and is computed by the sum of all decreases in node impurities where variable X_j is used to partition the data.

Let t_L and t_R denote the two resulting children nodes when partitioning the data at node t , and let N_t denote the number of observations reaching node t , the decrease in impurity is defined as,

$$\Delta i(s, t) = i(t) - p_L \times i(t_L) - p_R \times i(t_R) \quad (4.67)$$

where $i(\cdot)$ is impurty measure used to guide the partitioning of input variables such that the associated target outcomes, which are similar, are grouped into shared hyper rectangles. MSE is chosen as impurity measure in regression settings. $p_L = N_{t_L}/N_t$ and $p_R = N_{t_R}/N_t$. The resulting children nodes are obtained when the data is partitioned by the parent node at $s = (X_m < c)$. Hence, the MDI measure is defined by averaging over all trees T and all nodes t ,

$$VI(X_m) = \frac{\sum_T \sum_{t \in T: v(s_t)=X_m} p(t) \times \Delta i(s_t, t)}{\text{total number of tree}} \quad (4.68)$$

where $p(t)$ denotes the proportion N_t/N of samples reaching node t and $v(s_t)$ denotes the variable used to split node t .

Both variable ranking methods (permutation importance and MDI measure with the RME are available in the package “*randomForest*” in R[®] system for statistical computing and it provides an R interface of the Fortran program developed by Breiman and Cutler.

In short, random forests are very popular method for variable importance scoring because they are easy to apply due to simple parameter tuning process and is capable of differentiating informative variables from noise variables. In the thesis, permutation importance index is used to rank the variables for the cases in which input selection is needed.

Chapter 5

The Modelling Approach

Recall that the main contributions of the work are

- The estimation of feed TBP curve,
- Estimating the simulation output error.

The first contribution helps the process engineers to run the process simulator in any time, since once you have the feed TBP, you can calculate the other physical properties of the feed by using pseudo-component generation techniques given in Chapter 3. The second contribution helps the process engineers to calibrate the simulator outputs to obtain more realistic product quality estimations. Once you estimate the simulator error, you can estimate what the product quality will be just by adding the estimated simulation output error to the corresponding simulation output.

Both of these contributions can be achieved by following the modelling approach that is going to be explained in this chapter. Chapter 5 will give the framework that explains the modelling approach is given in detail. In the following chapters, namely Chapter 6 and Chapter 7, feed characterization model building and simulator output calibrator model building will be explained, respectively.

5.1. INTRODUCTION TO MODELLING APPROACH

In Figure 5.3, the flowchart summarizes the modelling approach. According to this flowchart, Aspen HYSYS© process simulator requires two types of inputs, i.e. “Operational variables of distillation process” and “Operational variables of reactor process”. Throughout the thesis, the following terminology will be used for these variables:

- “*Operative inputs*” instead of “operational variables of distillation process”
- “*Feed characterization inputs*” instead of “Operational variables of reactor process”

Operative inputs, X_i 's, are the variables that specifies the state of the fractionator process and these are temperature, pressure, flow measurements obtained from the field instruments. There are 22 operative inputs and they are listed in Table 3.2. Operative input data (Flows, temperatures, pressure) is available for any second and can be read from DCS system.

Feed characterization inputs, F_i 's, are the variables used to estimate the feed TBP curve. As mentioned in Chapter 3, once TBP curve is known, then it is possible to calculate any other physical property, such as

density, heat capacity, molecular weight, and so on, by using pseudo-component generation technique. The following figure indicates that the feed of the fractionator column is basically the reactor outlet stream. In Figure 5.1, the feed for the hydrocracker unit combines with the recycle and make-up hydrogen and then enters to the furnace before entering the reactor. In the reactors, treating and hydrocracking reactors occur so that products are obtained at the outlet of the reactor; however they are mixed and contains undesired components. These undesired components are eliminated in the HPS (High pres. Separator), LPS and H₂S stripper column. Thereafter, the products are separated from each other in the fractionator column. Except the undesired components (Purge gases, LPS vapor, Sour Gases, LPG) and hydrogen, the content of the reactor outlet and fractionator feed are the same. Hence, the process variables of the reactor process directly affects the feed properties of the fractionator column. This is the reason why we use “feed characterization inputs” instead of “operational variables of reactor process”.

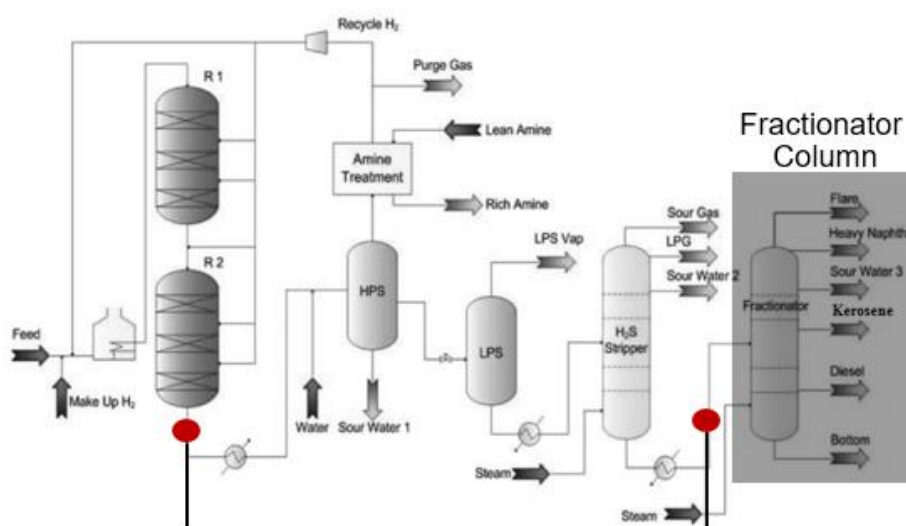


Figure 5.1 – Feed of the fractionator column is the products of the reactors

After explaining the operational inputs for distillation column and reactors, let's return to Figure 5.3. The gray box, labelled as “Process simulator” requires two types of inputs, operative inputs and feed TBP. Operative input values can be extracted from DCS system for any time. However, since feed TBP information can only be calculated once a day, it is estimated for any time with data-driven models which use feed characterization inputs. The blue box labelled as “Feed Characterization Models” takes feed characterization inputs to estimate the feed TBP. Feed characterization inputs are the process variables that directly affects the feed TBP and other feed properties. Similar to operative inputs, feed characterization inputs are also available for any second and can be exported from DCS system. After supplying operative inputs and feed TBP data to the process simulator, it can return many information related the process itself, such as the product qualities, column temperature/pressure profile, column internal and external flows and so on. By considering the scope of the thesis, only some of the product qualities are taken from the simulation and these qualities are the most crucial qualities that both customer and producer consider. More detail about the feed characterization models are given in Chapter 6.

Once the process simulator is run, the product qualities, W_i 's, are calculated by the simulator. As expected, there must be some error between the actual product qualities and calculated product qualities and there can be many reasons which are explained in Chapter 1.

The simulation output calibration models can be built by using the information gained from the process simulation, estimated feed TBP, operational state of the process (both reactor and distillation sections). However, it is needed to apply data preprocessing techniques, such as data normalization, data transformation, and feature selection to reduce the training times, simplify the models and enhance generalization performances by reducing overfitting. These models estimate the simulation error so that it will be possible to calculate the estimated product quality with the summation of simulator output and estimated simulation error. This is shown in Figure 5.2. Once the simulation output calibration models are built, then it is possible to estimate the simulation errors for a completely new data, which is unseen before. In Chapter 7, how the simulation output calibration models are build and the results of simulation output calibration models are given.

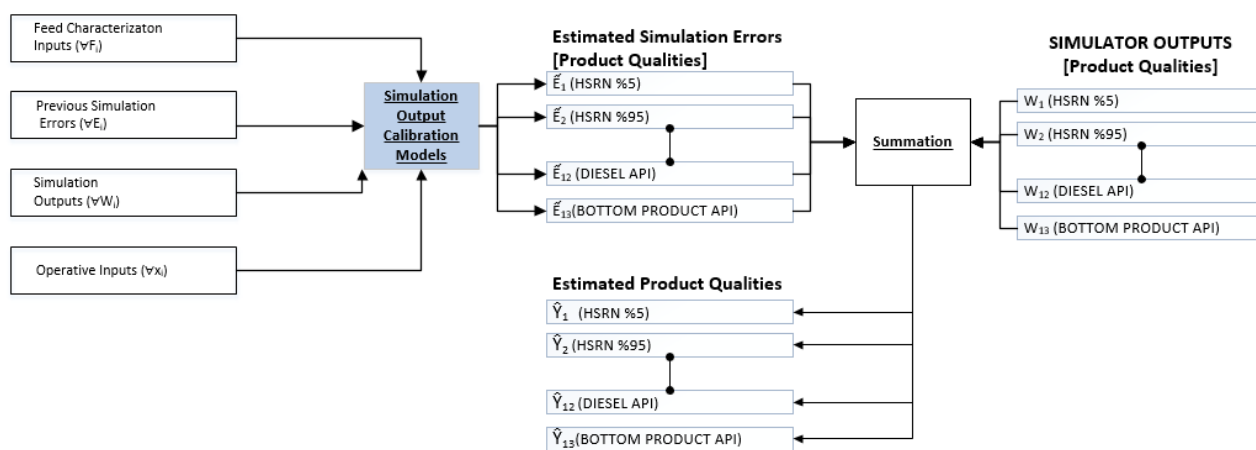


Figure 5.2 – Estimation of product qualities with the summation of estimated simulation errors and simulator outputs

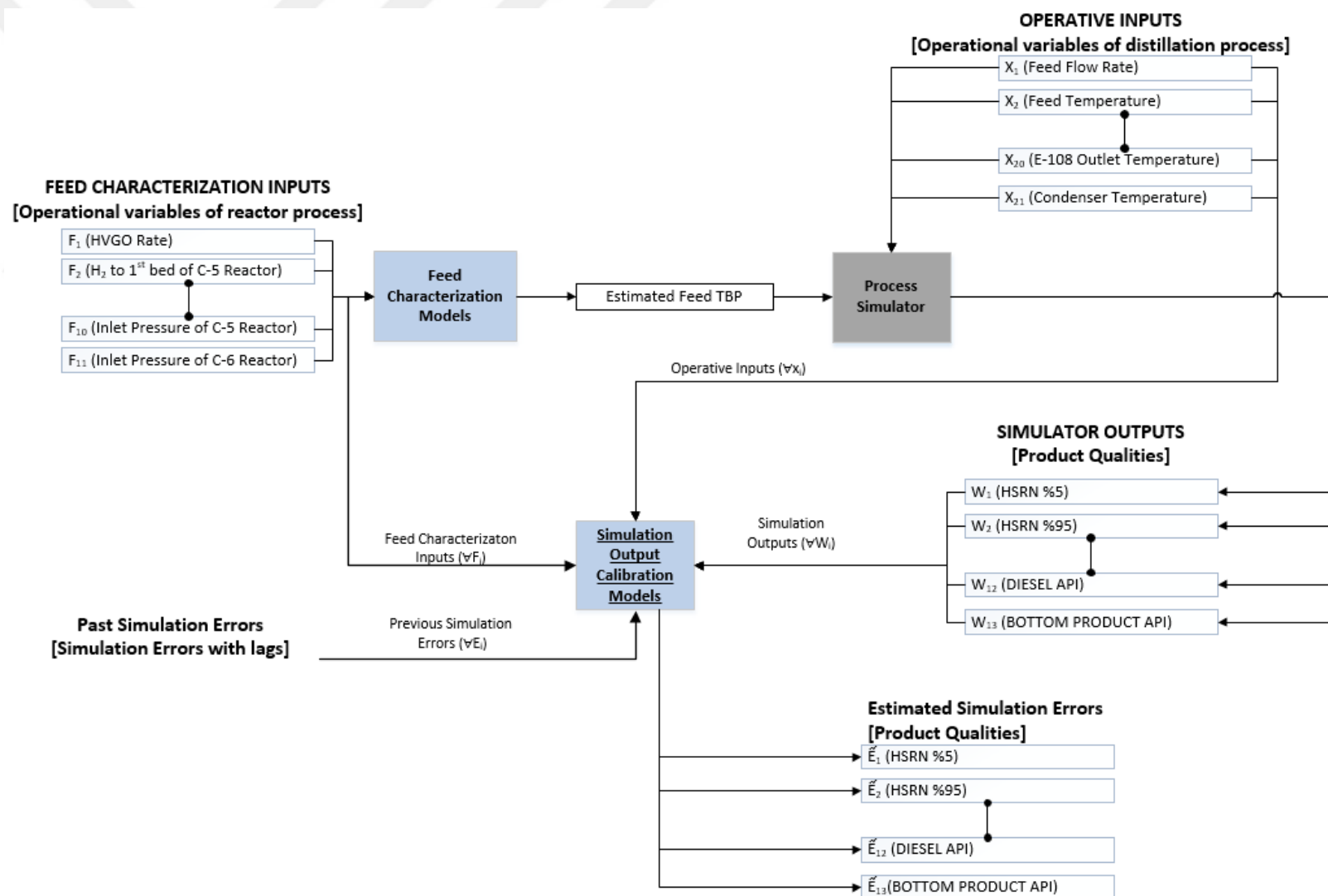


Figure 5.3 - Modelling Approach Scheme

5.2.PROCESS SIMULATOR OUTPUTS

In Chapter 3, the required data for a process simulator and the simulator outputs are given with details. After giving 22 operative and 11 feed characterization variables data to the simulator, the following product qualities given in Table 5.1 are estimated by the simulator even though there are many outputs that user can extract from the simulation, only product qualities will be extracted due to the scope of the thesis.

Table 5.1 – Process Simulator Outputs

Simulator Output	Description	Unit
W_1	HSRN %5 Boiling Point	°C
W_2	HSRN %95 Boiling Point	°C
W_3	KEROSENE %5 Boiling Point	°C
W_4	KEROSENE %95 Boiling Point	°C
W_5	KEROSENE FLASH Point	°C
W_6	DIESEL %5 Boiling Point	°C
W_7	DIESEL %95 Boiling Point	°C
W_8	DIESEL FLASH Point	°C
W_9	BOTTOM %5 Boiling Point	-
W_{10}	HSRN Density	API (@60F)
W_{11}	KEROSENE Density	API (@60F)
W_{12}	DIESEL Density	API (@60F)
W_{13}	BOTTOM Density	API (@60F)

The outputs from the simulator are compared with the actual product qualities and as expected there are some deviation of estimated qualities from the actual qualities due to several reasons given in Chapter 1. The process simulator errors are labeled as “ E_i ” for i^{th} product quality and calculated as follows:

$$E_i(t) = Y_i(t) - W_i(t) \quad (5.1)$$

where, $Y_i(t)$ is the actual product quality for t^{th} observation and W_i is the calculated product quality for t^{th} observation given by simulator.

Suppose that we observe a quantitative response Y and p different predictors, $X_1, X_2, \dots, X_p$. In this thesis, the response variables are the product qualities that are listed in Table 5.2.

Table 5.2 – Response Variables

Response Variable	Description	Unit
Y_1	HSRN %5 Boiling Point	°C
Y_2	HSRN %95 Boiling Point	°C
Y_3	KEROSENE %5 Boiling Point	°C
Y_4	KEROSENE %95 Boiling Point	°C
Y_5	KEROSENE FLASH Point	°C
Y_6	DIESEL %5 Boiling Point	°C
Y_7	DIESEL %95 Boiling Point	°C
Y_8	DIESEL FLASH Point	°C
Y_9	BOTTOM %5 Boiling Point	-
Y_{10}	HSRN Density	API (@60F)
Y_{11}	KEROSENE Density	API (@60F)
Y_{12}	DIESEL Density	API (@60F)
Y_{13}	BOTTOM Density	API (@60F)

This quantitative response variable Y (product qualities) can be calculated with the following equation:

$$Y_i(t) = W_i(t) + E_i(t) \quad (5.2)$$

Since the actual deviation of product quality from that of estimated by the simulator is never known for unseen data, it must be estimated to calculate actual $Y_i(t)$. Therefore, some fixed but unknown function of $X_1, X_2, \dots, X_p$, let this function to be shown as f , should be estimated.

$$E_i = f(X) + \epsilon \quad (5.3)$$

$$\hat{E}_i = \hat{f}(X) \quad (5.4)$$

$$\hat{Y}_i(t) = W_i(t) + \hat{E}_i(t) \quad (5.5)$$

where $\hat{f}(X)$ represents our estimate for f , and $\hat{E}_i$ represents the resulting prediction for error and ϵ is a random error term, which is independent of X . In this thesis, a dataset of inputs X and errors are readily available since simulations are performed and the obtained product qualities from the simulator are compared with the actual laboratory results so that it is possible to have a data of $\hat{E}_i(t)$. In this equation (Eq'n 5.4), $\hat{f}$ is treated as a black box, in the sense that the exact form of $\hat{f}$ are not within the scope of the thesis, provided that it yields accurate predictions of E_i .

The accuracy of $\hat{E}_i$ as a prediction for E_i depends on reducible and irreducible error. $\hat{f}$ will not be a perfect map for f leading to some error that can be reduced by implementing more proper machine learning techniques to estimate f and this error is called as reducible error. Even though it will be possible to form an perfect analytical equation to estimate for f , our prediction would still some error in it that cannot

be predicted by using the input variables X and this error is called as irreducible error, ϵ . Moreover, simulation outputs are also some inherent errors due to the assumptions applied to first-principle models. These are leading to irreducible error in f . This irreducible error is always larger than zero since there are some unmeasured variables that are useful for predicting E_i and since they are not measured, the perfect map f cannot use these unmeasured variables to predict E_i . A data for E_i have some inherent error since laboratory results, Y_i , are not a hundred percent correct.

Repeatability is the variability of the measurements obtained by one person while measuring the same item repeatedly. Even though the same technician retests the sample sample using the same laboratory instrument, there will be some variance in the results and this variance can be shown as $\sigma_{laboratory}^2$. Besides repeatability, reproducibility is another concern that is related to the agreement between results obtained from different laboratory instruments, different technicians and different laboratories. Reproducibility is the variability of the measurement system caused by differences in operator behavior. Mathematically, it is the variability of the average values obtained by several operators while measuring the same item. The variability of the individual operators are the same, but because each operator has a different bias, the total variability of the measurement system is higher when three operators are used than when one operator is used. The repeatability and reproducibility concepts are crucial since due to existence of these variances, there are always uncertainties in the laboratuary results. This uncertainties will be important when the performances of the simulation error models are evaluated.

The measurement system repeatabilty and reproducibility is given in Equation 5.6 and in Table 5.3, Gage R&R values for the product qualities are listed.

$$Gage\ R\&R = \sqrt{Repeatability^2 + Reproducibility^2} \quad (5.6)$$

Table 5.3 – Gage R&R values for product qualities considered in the thesis

PRODUCT QUALITY	GAGE R&R (IN CELCIUS)
HSRN %5	0.5
HSRN %95	1.0
KEROSENE %5	1.5
KEROSENE %95	2.0
KEROSENE FLASH	1.0
DIESEL %5	2.5
DIESEL %95	3.0
DIESEL FLASH	1.5
BOTTOM %5	6.0
HSRN API	0.3
KEROSENE API	0.3
DIESEL API	0.3
BOTTOM API	0.3

The following figure , Figure 5.4, shows the actual product quality values , Y_i 's, obtained from the laboratory and the calculated product quality values , W_i 's, obtained from the process simulator.

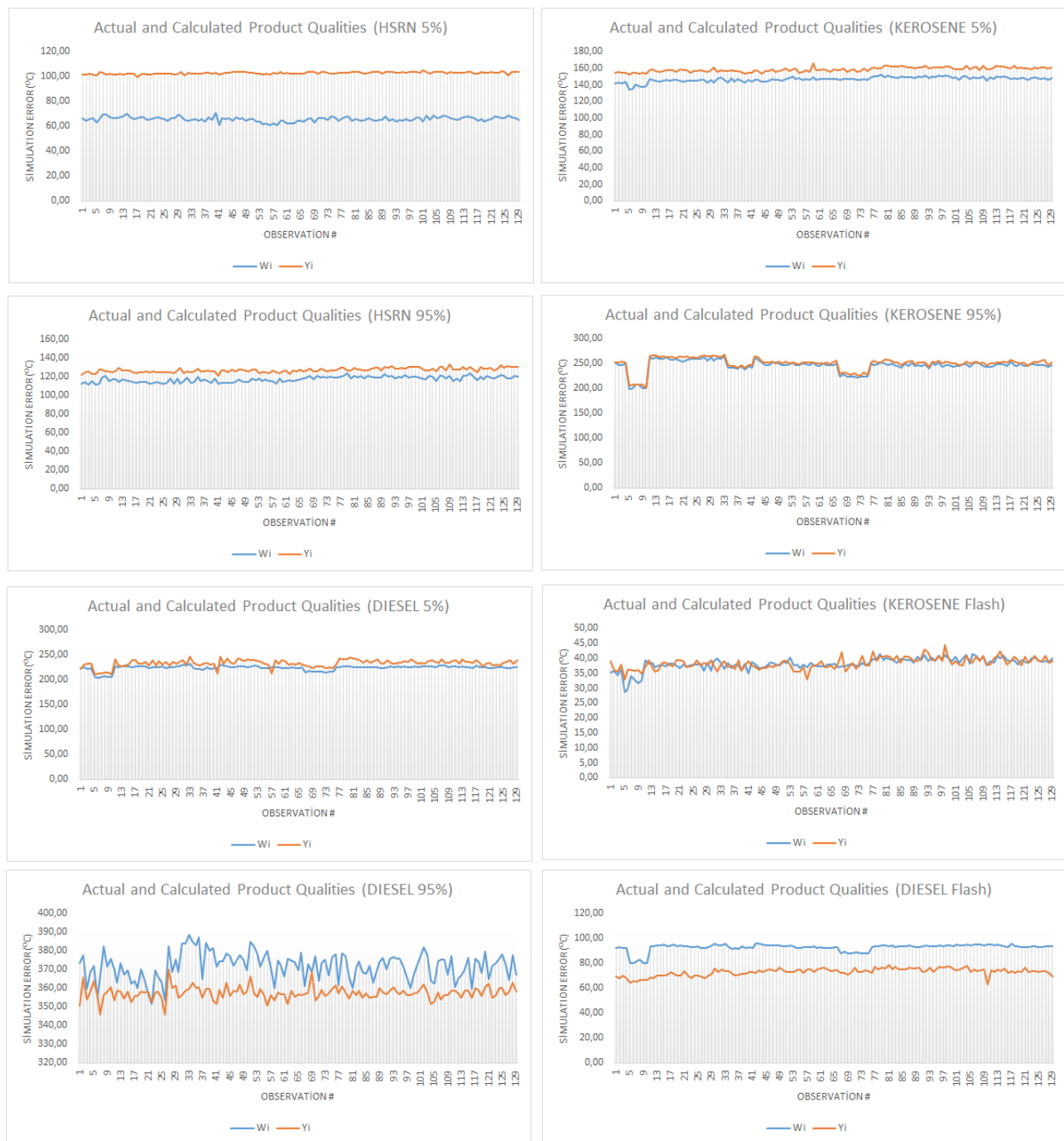


Figure 5.4 – Actual and calculated (by process simulator) product qualities

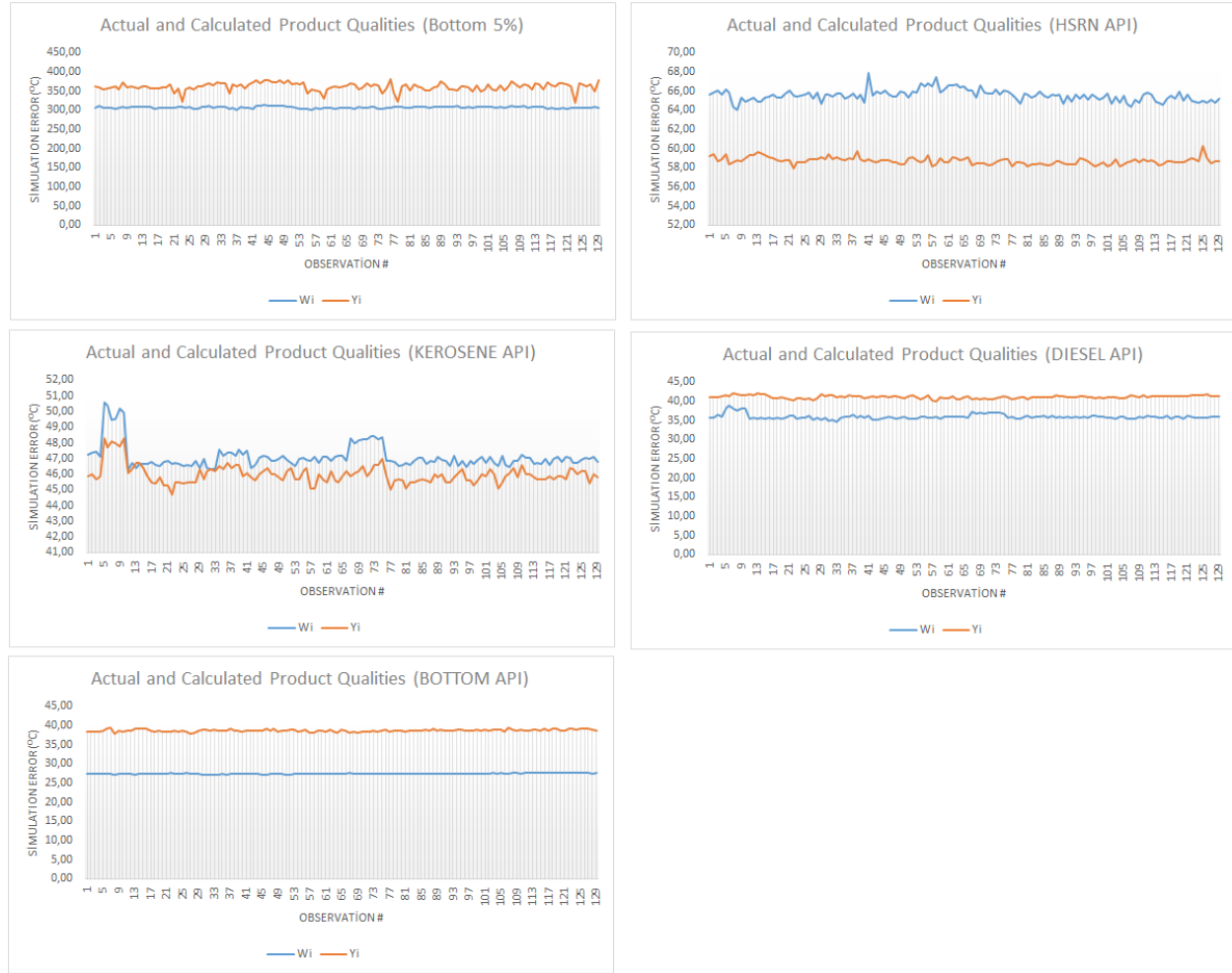


Figure 5.4 (Cont'd) – Actual and calculated (by process simulator) product qualities

As can be seen from the figure above, the error between the actual product quality and the calculated product quality is not constant. The details for how the mapping function, $\hat{f}$, to estimate this error is found and the predictors $X_1, X_2, \dots, X_p$ are given in detail in Chapter 7.

Chapter 6

FEED CHARACTERIZATION MODEL BUILDING AND THE MODEL RESULTS

6.1. Review of the approach for feed characterization

As shown in Figure 5.3, in order to run a process simulator for any time, the user must enter the feed TBP information to the simulator and to satisfy this requirement. To achieve this requirement feed characterization models are developed in this chapter. Input variables, F_i 's, for these models are selected by considering the process and they are listed in Table 6.1. 3 different types of model techniques are used in the training process and these are listed below:

- Linear regression
- Artificial Neural Networks
- Support Vector Regressions (with 3 different kernels : Linear, RBF, and Polynomial)

Artificial neural networks and support vectors with different kernels are tried to capture the nonlinearity between the feed TBP and input variables, F_i 's. Linear regression model is a simple and interpretable model and it is used as a benchmark model. Model configurations (such as error functions, training algorithms, cross-validation configuration, constant model parameters like learning rate for neural nets) and input data will be explained in the following sections.

6.2. Feed Characterization Model Building

6.2.1. Input Variable Selection

As mentioned in Section 6.1, Aspen HYSYS simulator requires feed TBP information; however, it is only measured a few times in a year or may not be measured. Therefore, it should be estimated by data-driven models to run the simulation application at any desired time.

The following variables are the operational feed characterization parameters that affect the feed properties. These variables are listed below:

Table 6.1 – Feed Characterization Variables

Variable	Description	Unit
F ₁	HVGO Rate	m ³ /h
F ₂	H ₂ to 1 st bed of C-5 reactor	Nm ³ /h
F ₃	H ₂ to 2 nd bed of C-5 reactor	Nm ³ /h
F ₄	H ₂ to 1 st bed of C-6 reactor	Nm ³ /h
F ₅	H ₂ to 2 nd bed of C-6 reactor	Nm ³ /h
F ₆	WABT of 1 st bed of C-5 reactor	°C
F ₇	WABT of 2 nd bed of C-5 reactor	°C
F ₈	WABT of 1 st bed of C-6 reactor	°C
F ₉	WABT of 2 nd bed of C-6 reactor	°C
F ₁₀	Inlet pressure of C-5 reactor	kg/cm ² _g
F ₁₁	Inlet pressure of C-6 reactor	kg/cm ² _g

These variables are selected by using the process knowledge and in the following paragraphs the relationship between the fractionator column feed properties and the selected variables are explained in detail.

Figure 6.1 shows a conventional the hydrocracker unit of a large-scale refinery. Heavy vacuum gas oil (HVGO) feed coming from the vacuum distillation unit (VDU) is mixed with a hydrogen-rich gas and then heated by a furnace. In the treating reactor, undesired contents like nitrogen and sulfur are significantly eliminated and in the following hydrocracker reactor the larger molecules are cracked into smaller ones so that lighter fractions are produced: heavy naphtha, kerosene, diesel and unconverted oil. After leaving the reactors, the products are fed to a high pressure separator (HPS) to recover unreacted hydrogen and then fed to a low pressure separator (LPS) to separate the light gases from the liquid. The remaining liquid is sent to H₂S stripper column, in which dissolved H₂S from light hydrocarbons and the bottom of this column is sent to the main fractionator column. In short, the products from the reactors are firstly treated (cleaned) and then sent to the fractionator distillation column. Therefore, it is possible to say that the most critical parameters for the reactors directly influence the feed properties like TBP. In such a case, one should know the crucial parameters for the hydrocracker reactors.

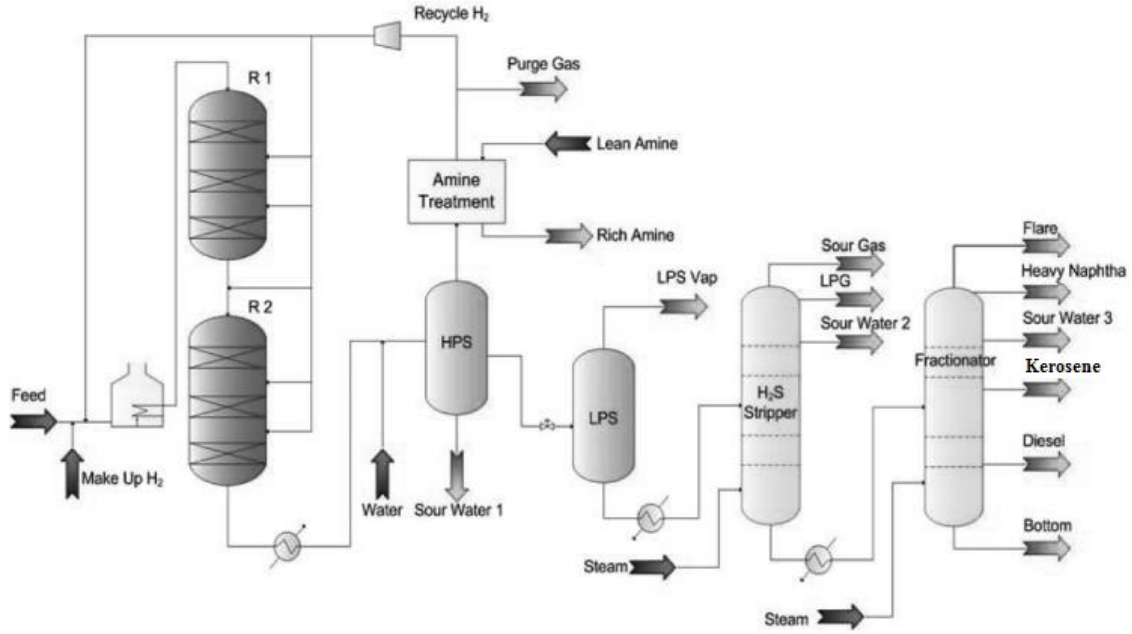


Figure 6.1 – Hydrocracker unit simplified process flow diagram

The most important operating variable for reactors is the reactor temperature. Increasing the reactor temperature increases reaction conversion and shifts product distribution from heavier to lighter products. The refiners intend to raise the reactor temperature gradually to produce a desirable product distribution. Most hydrocracking reactions are exothermic (heat releasing) in nature, meaning that commercial reactor temperature increases as the reactor feed HVGO passes through the reactor. In other words, outlet reactor temperature will be higher than that of the inlet section. In order to determine the average temperature of the reactors, the weight-average bed temperature (WABT) is commonly used and can be calculated with the following formula if there exists various temperature measurement devices located in different zones of the reactor :

$$WABT = \sum_{i=1}^N WABT_i \times Wc_i \quad (6.1)$$

$$WABT_i = \frac{T_i^{in} - 2T_i^{out}}{3} \quad (6.2)$$

where, N is the number of catalyst bed within the reactor and Wc_i is the weight fraction of catalyst in each bed with respect to the total.

Figure 6.2 to 6.4 illustrate the effects of the feed flow rate (HVGO flow rate) and WABT (Weighted average bed temperature) on the product distribution (yields) of an hydrocracker unit. The heavy naphtha yield increases significantly while WABT (increases and/or feed flow decreases, because the rising HCR

(Hydrocracker) reactor temperature enhances HCR reactions, and the decreasing feed flow implies a longer residence time that also enhances HCR reactions. On the other side, the bottom oil yield presents an opposite trend to that of the heavy naphtha yield. This follows because the bottom oil is the heaviest product and the higher severity of cracking reactions resulted from a rising WART and/or a falling feed flow rate, with the lower the bottom oil yield. However, the most interesting observation comes from Figure 6.4 that represents the diesel fuel yield reaching a maximum value at some operating point. Tippet et al. (Tippet, et al., 1978) reports that, in a HCR process, the yield of the middle distillate fraction (diesel fuel in this case) would approach a maximum value with an increasing reactor temperature, because of secondary HCR reactions of middle distillate paraffins. We can conclude that, with a lower feed flow rate, the diesel fuel yield tends to approach a maximum when increasing WART of the HCR reactor.

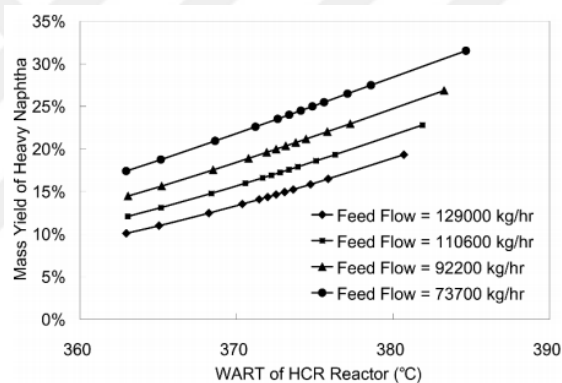


Figure 6.2 – Effect of HVGO rate and WART of the hydrocracker reactors on HSRN yield (Chang et al., 2012)

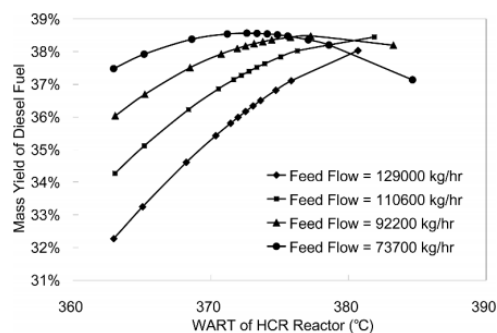


Figure 6.3 – Effect of HVGO rate and WART of the hydrocracker reactors on Diesel Yield (Chang et al., 2012)

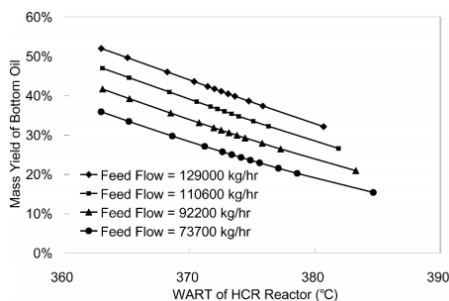


Figure 6.4 – Effect of HVGO rate and WART of the hydrocracker reactors on UCO Yield (Chang et al., 2012)

Therefore, as can be understood from the previous graphs, the HVGO Rate and WART of the reactors and the composition of HVGO (which is correlated with the distillation information) are key variables effecting the product yields in the main fractionator column. Fortunately, most of the time HVGO composition is more or less the same, since it usually comes from vacuum units or exported from other refineries.

Another important parameters is the reactor pressures. The pressure influences considerably the equilibrium of dehydrogenation-hydrogenation reactions that occurs on the metallic sites. The pressure increase for a specific H₂/feed molar ratio will lead to an increase in the partial pressure of hydrogen, resulting in an increase of conversion of the aromatic structures to saturated products which will improve the quality of the products.

The amount of hydrogen entering the reactors are also crucial. The following figure shows the product yields versus hydrogen consumed for a specific reactor with a specific feed properties. Therefore, hydrogen rate should be taken as an input variable, as well.

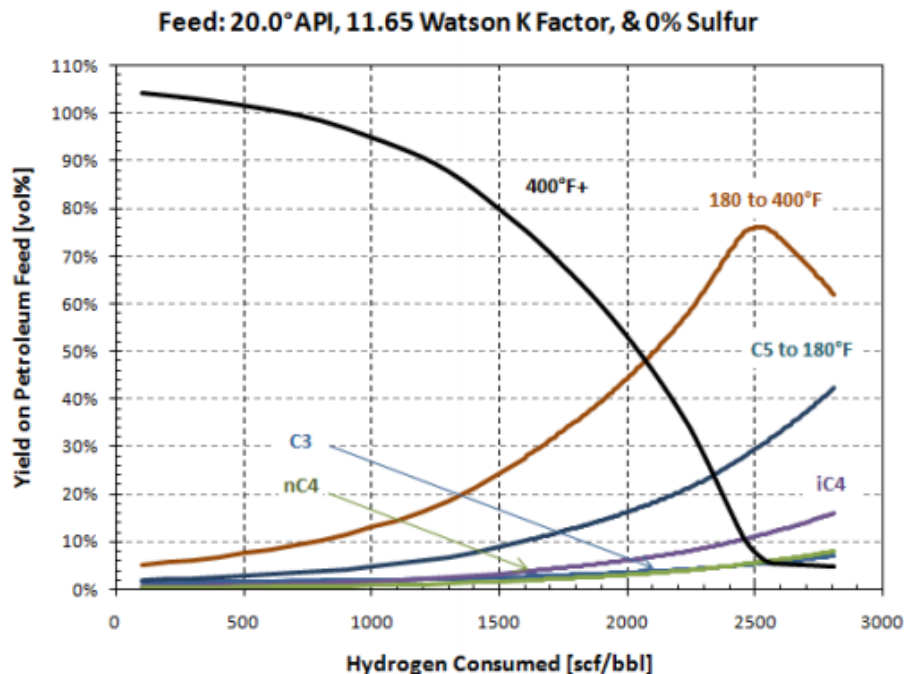


Figure 6.5 – Effect of hydrogen rate on the product yields (Jechura, 2016)

After deciding the input variables for model development using the process knowledge mentioned in the previous paragraphs, one can apply supervised learning methodology to infer a function (or map) from labeled training data. In supervised learning, each sample is a pair consisting of an input vector (those elements are listed in Table 6.1) and a desired output value (feed TBP values). By implementing any supervised learning algorithm, one can produce an inferred function, which can be used for mapping new examples. In this thesis, linear regression, artificial neural networks and support vector regression techniques are used and theoretical backgrounds for these techniques are given in Chapter 4.

6.2.2. Input Data

As mentioned in the previous sections, data for flows, temperatures, pressure are available for any second and can be obtained from DCS system. Unfortunately, since the label variable (estimated variable) is the true boiling curve (TBP) of the feed and is not measured, this TBP is obtained by using the process simulator application by using back-blending method (The details for back-blending method is given in Section 3.3.1). Once the TBP curve is supplied to the simulator, it can calculate physical properties by using the correlations embedded inside the simulator. This method requires the boiling information of the products, which is available for each day at around 06:00 A.M. Therefore, the true boiling curve of the feed can be calculated for each day. Since the label variable (or estimated variable) is the feed TBP and can be only calculated once a day for around 06:00 A.M, data for feed characterization input variables are collected by averaging the values between the sample collection time (the time at which the sample is collected in the

field) and 30 minutes before this time. It is important to note that the operative variable values needed to be almost constant, otherwise the feed qualities may change significantly. 135 sample data are collected and these are between 16th February 2015 and 18th March 2016. During this period, the feed characterization input variables are almost the same around the sample collection time.

Data Normalization

The rate of convergence in model training both for support vectors regression and neural network is affected considerably if the data is normalized (Graf, et al., 2003) . In addition to that, normalization is required for particular kernels (for Support vector regression) due to their restrictive domains (e.g. B splines) and may be helpful for unrestricted kernels by avoiding the problems related with Hessian matrix in the optimization problem. One can increase the rate of convergence by decreasing the difference between the largest and smallest eigenvalue (Gunn., 1998). In Herbrich 2001 (Herbrich, et al., 2001), it is revealed that the type of normalization has also important influence in SVM model performance, especially for the classification purposes.

Similar to support vector training, data scaling is also considered as a requirement for the application of multilayer Perceptrons. Even though, in theory it is not necessary to normalize the input data, practice has shown that when the input data values are normalized, neural network training is often more efficient and leads to a better predictor (Kaastra, et al., 1996).

In this thesis, normalization is applied to the input data for neural network model development, whereas standardization is applied to the input data for support vector model development. Normalization can be applied by using Equation 6.3 and standardization is carried out by using Equation 6.4, where μ is the mean of the data and σ is the standard deviation:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (6.3)$$

$$x_{new} = \frac{x - \mu}{\sigma} \quad (6.4)$$

Normalization scales all numeric variables in the range [0,1]. On the other hand, standardization will transform the variables to have zero mean and unit variance. It should be also noted that both of these techniques will be inefficient if there exists an outlier.

6.2.3. Parameter Search

There are some parameters that must be optimized for support vector regression, artificial neural network and linear regression models. In feed TBP modelling with support vector regression, the following kernels are used and the parameters that must be optimized are given in the Table 6.2.

Table 6.2 – SVM Model Parameters

	SVM Kernel Types		
	Linear	Radial Basis Function	Polynomial
Parameter 1	ϵ	ϵ	ϵ
Parameter 2	C	C	C
Parameter 3		γ	γ

The ϵ -insensitive loss function are used as a loss function in all support vector models. Therefore, user must define the value for this parameters, which describes the width of the insensitive range. For all kernels, epsilon value (ϵ) must be defined. This parameter is crucial since the number of support vectors is directly affected with this value. Small epsilon values will increase the model complexity so that model will have more support vectors and it will lead to memorization of the training data and decrease in generalization performance. Similarly, larger values for epsilon parameter will lead to fewer support vectors; however, it may lead to decrease in prediction performance of the model due to flatter estimation function. Therefore, there should be an optimum value for this parameter. Optimal setting of ϵ requires noise level knowledge. Fortunately, this noise variance can also be estimated directly from the training data.

Another SVR parameter is C (cost) and it is used to overcome the problem of outliers by creating a cost to estimation errors larger than epsilon. This parameter controls the trade off between errors of the SVM on the training data and margin maximization (Note that if C is infinity, it leads to hard margin SVM). A small value for C will increase the number of training errors, while a large C will lead to a behavior similar to that of a hard-margin SVM. In practice the parameter C is varied through a wide range of values and the optimal performance assessed using a separate validation set or a technique known as cross-validation for verifying performance using only a training set.

When using RBF kernel, separating surface will be based on a combination of bell-shaped surfaces centered at each support vector. The width of each bell-shaped surface will be inversely proportional to γ (gamma). If this width is smaller than the minimum pair-wise distance for data, overfitting is possible. If this width is larger than the maximum pair-wise distance for data, all points fall into one class and it won't have good performance either. So the optimal width should be somewhere between these two extremes.

In Libsvm toolbox in R, polynomial kernel is described as the following, $(\gamma u'v + 1)^{degree}$. Therefore, gamma is also a parameter for this type of kernel that must be optimized. Degree is chosen as 3(default value in Libsvm) for all polynomial kernels in this study.

It is not known beforehand which parameters are the best for a given regression problem. In such a case, some kind of model selection(parameter search) must be performed to identify good parameters so that the predictor model can accurately predict a new data (i.e. testing data). It is important to note that it may not be useful to achieve low training errors due to generalization issue, i.e. lowering the training error may result in overfitting. Common strategy is to divide the data set into two parts, of which one is considered as unknown. The prediction accuracy obtained from the unknown data division more precisely shows the actual performance of a model. Cross-validation is one of the improved version of this procedure. Hence, the user-defined parameters of SVR are optimized by cross validation methods with grid search method.

Cross-validation and Grid Search

Cross validation methods are used for partitioning the data set into training and testing sets. The training set is used in training the estimation function while testing set is used for monitoring the performance of generalization of the estimation function. There are many kinds of cross validation methods and this study focuses on k-fold cross validation method, because it is a non-exhaustive method with reasonable computational cost. k-fold cross validation method randomly partitions the data set into k equally sized subsets. One of the subsets is selected as the testing set and the rest of the subsets are used as training set. The cross validation process is repeated k times thus naming the method k-fold, where each subset is used exactly once for testing. The squared error from each iteration is averaged to give a single mean squared error (MSE). The parameter k is user-defined and increasing k provides diminishing value in terms of source size creation.

Grid Search (GS) method is a tool popularly used for optimizing SVR parameters. This method applies a mesh on the search space and re-meshes around the best known solutions until the best solution improves under a given tolerance or the mesh size is under a given tolerance. The performance of the grid search methods highly depends on the resolution of the mesh and search method. The grid search method may miss a good solution due to low resolution of the mesh. However a high resolution will have high computational cost since each edge of the mesh should be evaluated. Grid-search on the parameters using cross-validation is used to determine the best model parameters. Various combination of parameters are tried and the one with the best-cross validation accuracy is picked in this study. Moreover, exponentially growing sequences of C and γ is a good practical method to identify good parameters. Similarly linearly growing sequence of ε is tried during the parameter determination process.

Cross-validation can be used to estimate the quality of a neural network. If it is applied to different neural networks with different model parameters, like as the number of hidden node, the result of cross-validation process can be used to choose the best parameter values. In this study, backpropagation algorithm is used to train the networks. Learning rate for back-propagation algorithm is chosen as 0.01 and a differentiable function that is used for the calculation the errors is sum of the squared errors. Different networks (Networks with different number of hidden nodes) are tried within the study and the best number of hidden nodes is determined by using the cross-validation.

Methodology

All the data preprocessing (such as, data scaling and outlier elimination), model training and model parameter search algorithms was carried out in R programming©. For support vector model development, Libsvm library with e1071 package (Misc Functions of the Department of Statistics, Probability Theory Group) was used, while neuralnet package was used to train the neural networks.

As mentioned in the previous section, we have 135 sample sets with 11 input variables dating from 16.02.2015 to 18.03.2016. The input data is scaled before the training process. Therefore the dataset is split into training and testing sets randomly, having 115 observation and 20 observations, respectively. Linear, polynomial and radial basis function kernels are employed for support vector training. The grid search region for parameters and other constant model parameters are given in Table 6.3.

Table 6.3 – Grid Search Region for SVR parameters

	Cost		
	Values	Gamma Values	Epsilon Values
1	0.100	0.00010	0
2	0.183	0.00026	0.2
3	0.336	0.0007	0.4
4	0.616	0.0018	0.6
5	1.129	0.0048	0.8
6	2.069	0.013	1
7	3.793	0.034	1.2
8	6.952	0.089	1.4
9	12.7	0.234	1.6
10	23.4	0.616	1.8
11	42.8	1.624	2
12	78.5	4.281	
13	143.8	11.288	
14	263.7	29.764	
15	483.3	78.5	
16	885.9	206.9	
17	1623.8	545.6	
18	2976.4	1438.5	
19	5455.6	3792.7	
20	10000	10000	

As can be seen from the table, cost and gamma values are growing exponentially, whereas the epsilon range is increasing linearly. Then grid search algorithm combined with cross-validation was employed (Cross validation configuration is given in the following table). $20 \times 20 \times 11 = 4400$ different combination of parameters are tried for different kernels and model parameters.

Table 6.4 – Information for Cross-Validation Configuration

Partitioning method	k-fold
Partition number	10

Artificial neural network models was trained with the training data set and the default parameters used for ANN is given in Table 6.5. The number of hidden nodes is ranged from 1 to 20.

Table 6.5 – Defined Parameters for Artificial Neural Network Training

Parameter	Value
Number of hidden layer	1
Number of hidden node range	1 to 20
Training method	Back-propagation
Learning Rate	0.01
Hidden Layer Function	Logistic
Output Layer Function	Linear
Error Function	Sum of squared errors

Training data set with 115 observation is used for parameter optimization with cross-validation. After the best parameters for a certain model is decided, the model is trained with the best parameters and tested on the test data for evaluation of model performance.

The parameter optimization results and model performances are summarized in the following section. After the model parameters are optimized and the best models are selected, feed characterization models can be used to estimate the feed true boiling point information, which will be needed within the process simulator.

6.3. Model Results

For each boiling points of feed (5%, 10%, 20%, ..., 90%, 95%), 10-fold cross validation error are used as a measure of model performance and the one with the lowest CV error is chosen to be a model for the boiling point of interest. The following table and figure shows the 10-fold CV errors of the feed boiling points.

Table 6.6 - 10-Fold cross validation error (RMSE) for different models

Boiling point of the feed	Linear Regression	ANN	SVM (Linear Kernel)	SVM (RBF Kernel)	SVM (Polynomial Kernel)
5%	2,08	1,2	1,2	1,1	1,2
10%	2,44	2,5	2,5	2,3	2,4
20%	2,62	2,1	2,1	2,0	2,0
30%	2,15	2,1	2,2	2,1	2,1
40%	2,14	2,1	2,0	2,1	2,1
50%	3,02	2,6	2,8	2,6	2,7
60%	5,13	4,8	4,8	4,4	4,5
70%	4,98	5,1	4,9	4,7	4,7
80%	5,02	4,5	4,4	4,4	4,4
90%	4,61	3,8	3,8	3,8	3,8
95%	4,15	4,7	4,8	4,7	4,7
Average	3,49	3,23	3,23	3,11	3,13

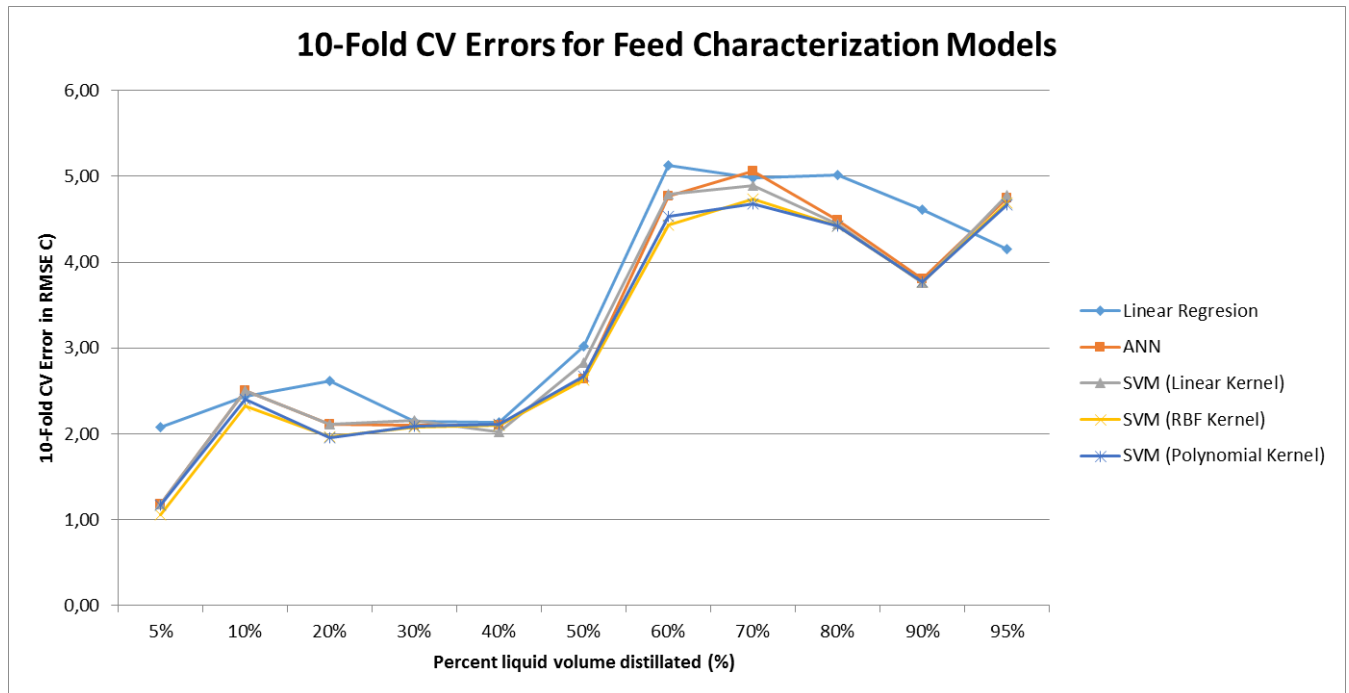


Figure 6.6 – 10-Fold cross validation error (RMSE) for different models

As can be seen from Figure 6.6, in general 10-fold CV error are quite close to each other for all models and as the percent liquid volume distilled increases, the 10-fold CV error also increases and it means that it is more difficult to characterize the higher boiling fractions. To choose between the models, we simply compute the mean cross-validated errors for each modelling technique, and then take the model with the minimum cross-validated error. The final row of Table 6.6 shows the mean cross-validated errors and according to this table, SVM with RBF kernel has the lowest mean of 10-fold CV errors. Hence, this model is chosen for estimating all boiling points of the feed.

Table 6.7 shows the actual feed TBP information and corresponding estimated TBP information for one of the training observations. This observation refers to the 8th observation (corresponding date: 5th of March 2015). Table 6.8 gives similar information given in Table 6.7 for the 51st observation (corresponding date: 7th of August of 2015), but now for testing observation. As can be seen from both table, feed information is estimated well by considering the repeatability and reproducibility in measurement systems.

In Table 6.9, training, testing, and cross-validation errors in RMSE, as well as the optimum model parameters found by grid search for different model types are listed. Note that the training data (with 115 sample) is used to decide model parameters with cross-validation. Once the model parameters are set, then the model is again trained with all training data. The resulting model is tested on the test data (with 20 observation). By considering that it is possible to say that in Table 6.9, the support vectors for SVR model with RBF kernels are included in training data.

Table 6.7 – Feed TBP curve comparison (Actual vs predicted) for 8th observation (Training Sample)

LIQ.VOLUME PERCENT DISTILLED (%)	ACTUAL BOILING POINT (°C)	PREDICTED BOILING POINT (°C)	ERROR (°C)
5%	100.6	101.0	-0.3
10%	115.7	115.7	0.0
20%	155.0	155.8	-0.8
30%	188.4	188.0	0.5
40%	221.3	220.3	1.0
50%	253.3	253.2	0.1
60%	296.6	297.3	-0.7
70%	354.9	355.8	-0.9
80%	411.5	408.7	2.8
90%	433.7	430.5	3.3
95%	449.4	448.1	1.3

Table 6.8 – Feed TBP curve comparison (Actual vs predicted) for 51st observation (Test Sample)

LIQ.VOLUME PERCENT DISTILLED (%)	ACTUAL BOILING POINT (°C)	PREDICTED BOILING POINT (°C)	ERROR (°C)
5%	103.2	102.0	1.2
10%	119.8	118.1	1.8
20%	157.4	156.9	0.5
30%	189.0	189.2	-0.2
40%	221.6	221.3	0.3
50%	252.7	252.7	-0.1
60%	293.0	294.1	-1.1
70%	348.2	351.0	-2.8
80%	410.9	409.6	1.3
90%	428.3	429.3	-1.0
95%	447.0	446.1	0.9

Table 6.9 – Model summaries for feed TBP modelling

Boiling point	Model	Validation Error [10-Fold CV- RMSE]	Train Error [RMSE - Whole Train Data:115 Sample]	Test Error [RMSE -Test Data: 20 Sample]	Model Parameters				
					# of SVs	C	Epsilon	Gamma	# of nodes
5%	Linear Regression	2.08	1.06	0.86	-	-	-	-	-
	ANN	1.18	1.09	0.99	-	-	-	-	2
	SVM(Linear Kernel)	1.18	1.10	0.99	42	1.1	0.6	-	-
	SVM(RBF Kernel)	1.06	0.55	1.03	92	2.1	0.2	0.2336	-
	SVM(Poly. Kernel)	1.18	1.09	0.98	42	263.7	0.6	0.0007	-
10%	Linear Regression	2.44	2.31	2.01	-	-	-	-	-
	ANN	2.51	2.14	1.65	-	-	-	-	13
	SVM(Linear Kernel)	2.50	2.43	1.96	115	2976.4	0.0	-	-
	SVM(RBF Kernel)	2.32	0.70	2.13	115	3.8	0.0	0.2335	-
	SVM(Poly. Kernel)	2.40	1.57	3.06	83	10000.0	0.2	0.0018	-
20%	Linear Regression	2.62	1.89	1.83	-	-	-	-	-
	ANN	2.11	1.88	1.85	-	-	-	-	11
	SVM(Linear Kernel)	2.11	1.91	1.85	60	3.8	0.4	-	-
	SVM(RBF Kernel)	1.97	1.55	1.78	78	1623.8	0.2	0.0018	-
	SVM(Poly. Kernel)	1.96	1.56	1.77	79	885.9	0.2	0.0018	-
30%	Linear Regression	2.15	1.91	2.13	-	-	-	-	-
	ANN	2.11	1.95	1.99	-	-	-	-	1
	SVM(Linear Kernel)	2.16	2.01	2.22	115	0.3	0.0	-	-
	SVM(RBF Kernel)	2.08	1.74	1.99	84	143.8	0.2	0.0048	-
	SVM(Poly. Kernel)	2.09	2.01	2.23	115	885.9	0.0	0.0001	-

Table 6.9 (Cont'd) – Model summaries for feed TBP modelling

40%	Linear Regression	2.14	1.86	2.01	-	-	-	-	-
	ANN	2.10	1.89	1.92	-	-	-	-	1
	SVM(Linear Kernel)	2.02	1.92	2.09	115	10000.0	0.0	-	-
	SVM(RBF Kernel)	2.10	1.90	1.85	51	10000.0	0.6	0.0001	-
	SVM(Poly. Kernel)	2.11	1.89	1.91	115	10000.0	0.0	0.0001	-
50%	Linear Regression	3.02	2.60	2.11	-	-	-	-	-
	ANN	2.64	2.23	2.04	-	-	-	-	3
	SVM(Linear Kernel)	2.83	2.64	2.12	76	7.0	0.4	-	-
	SVM(RBF Kernel)	2.62	1.89	1.81	81	23.4	0.4	0.0336	-
	SVM(Poly. Kernel)	2.67	2.43	1.74	45	263.7	0.8	0.0018	-
60%	Linear Regression	5.13	4.52	2.84	-	-	-	-	-
	ANN	4.77	4.09	2.88	-	-	-	-	3
	SVM(Linear Kernel)	4.80	4.60	3.02	78	42.8	0.4	-	-
	SVM(RBF Kernel)	4.43	3.19	3.16	95	3.8	0.2	0.0886	-
	SVM(Poly. Kernel)	4.53	3.90	2.22	54	78.5	0.6	0.0048	-
70%	Linear Regression	4.98	4.57	2.53	-	-	-	-	-
	ANN	5.06	4.69	2.81	-	-	-	-	15
	SVM(Linear Kernel)	4.89	4.71	2.89	30	3.8	1.0	-	-
	SVM(RBF Kernel)	4.73	4.11	3.70	115	7.0	0.0	0.0336	-
	SVM(Poly. Kernel)	4.68	4.24	3.68	115	0.2	0.0	0.0886	-
80%	Linear Regression	5.02	4.14	4.88	-	-	-	-	-
	ANN	4.49	4.29	4.62	-	-	-	-	7
	SVM(Linear Kernel)	4.45	4.29	5.33	35	0.1	0.8	-	-
	SVM(RBF Kernel)	4.42	4.24	5.02	35	7.0	0.8	0.0048	-
	SVM(Poly. Kernel)	4.42	4.17	5.23	115	10000.0	0.0	0.0000	-

Table 6.9 (Cont'd) – Model summaries for feed TBP modelling

90%	Linear Regression	4.61	3.57	4.00	-	-	-	-	-
	ANN	3.81	3.49	3.54	-	-	-	-	7
	SVM(Linear Kernel)	3.76	3.64	3.69	47	0.3	0.6	-	-
	SVM(RBF Kernel)	3.77	3.64	3.66	44	885.9	0.6	0.0001	-
	SVM(Poly. Kernel)	3.77	3.63	3.68	47	885.9	0.6	0.0001	-
95%	Linear Regression	4.15	4.47	4.14	-	-	-	-	-
	ANN	4.74	4.48	3.94	-	-	-	-	11
	SVM(Linear Kernel)	4.78	4.56	4.21	66	0.1	0.4	-	-
	SVM(RBF Kernel)	4.69	4.37	3.64	85	2976.4	0.2	0.0003	-
	SVM(Poly. Kernel)	4.66	4.34	3.60	83	2976.4	0.2	0.0026	-

In Appendix I, the simulation outputs for product qualities with both the actual feed and the predicted feed, which are estimated with the feed characterization models are tabulated. The ratio of the sum of squared errors (SSE) of the simulation outputs with the actual feed TBP and the sum of squared errors (SSE) of the simulation outputs with the estimated feed TBP is termed as “A”. This term denotes the percent of error in predicting laboratory (actual) result with the estimated feed is not due to the estimation of feed characterization. This variable is calculated by the following equation:

$$A = \frac{\sum_1^{135} r_{2,i}^2}{\sum_1^{135} r_{1,i}^2} \quad (6.5)$$

where,

$r_{1,i}$ = Actual Product Quality – Simulation output for the product quality with estimated feed properties

$r_{2,i}$ = Actual Product Quality – Simulation output for the product quality with actual feed properties

Table 6.10 – Final Models for feed characterization models (SVR with RBF Kernels)

Boiling point	10-CV Error [RMSE]	Train Error [RMSE]	Test Error [RMSE]	R ² Train	R ² Test	# of SVs	C	Epsilon	Gamma
5%	1.06	0.55	1.03	0.76	0.71	92	2.1	0.2	0.2336
10%	2.32	0.7	2.13	0.95	0.93	115	3.8	0	0.2335
20%	1.97	1.55	1.78	0.82	0.79	78	1623.8	0.2	0.0018
30%	2.08	1.74	1.99	0.84	0.78	84	143.8	0.2	0.0048
40%	2.1	1.9	1.85	0.72	0.71	51	10000	0.6	0.0001
50%	2.62	1.89	1.81	0.63	0.59	81	23.4	0.4	0.0336
60%	4.43	3.19	3.16	0.68	0.65	95	3.8	0.2	0.0886
70%	4.73	4.11	3.7	0.62	0.60	115	7	0	0.0336
80%	4.42	4.24	5.02	0.71	0.65	35	7	0.8	0.0048
90%	3.77	3.64	3.66	0.83	0.78	44	885.9	0.6	0.0001
95%	4.69	4.37	3.64	0.75	0.68	85	2976.4	0.2	0.0003
Average	3.1	2.5	2.7			79.5	1425.2	0.3	0.0577
Min	1.1	0.6	1.0			35.0	2.1	0.0	0.0001
Max	4.7	4.4	5.0			115.0	10000.0	0.8	0.2336

The A-ratio values for each product quality is given in Table 6.11. As can be seen from the table, A ratio for each property is almost equal to 1 (except kerosene flash point), meaning that the simulation output does not change in a considerable amount if the feed properties are estimated with the feed characterization models. In addition to A-ratio, the percentage of the response variable variation that is explained by the final models, R^2 's, are also calculated for the final models with the following equation and R^2 values are given for training and testing data in Table 6.10.

$$R^2 = \frac{\text{Explained Variation}}{\text{Total Variation}} = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} \quad (6.6)$$

Table 6.11 – A-factors for each product quality

	Property	sum of r_1^2 's	sum of r_2^2 's	A
Y ₁	HSRN %5	184988	185089	1.001
Y ₂	HSRN %95	1429748	1427160	0.998
Y ₃	KERO %5	19585	19185	0.980
Y ₄	KERO %95	14318	13710	0.958
Y ₅	KERO Flash	413	345	0.834
Y ₆	DIESEL %5	54531	53849	0.987
Y ₇	DIESEL %95	33002	32209	0.976
Y ₈	DIESEL Flash	54500	55220	1.013
Y ₉	BOTTOM %5	407242	408642	1.003
Y ₁₀	HSRN API	6345	6356	1.002
Y ₁₁	KERO API	217	206	0.948
Y ₁₂	DIESEL API	3702	3691	0.997
Y ₁₃	BOTTOM API	17125	17082	0.997

In Figure 6.7, the actual laboratory results for 5% percent boiling point of Kerosene and flash point of Diesel are given as sample trends with the simulator outputs with estimated and actual feed properties. These graphs shows that the feed estimation models performs well so that the simulation outputs for product qualities does not deviate a considerable from that of calculated with the actual feed properties.

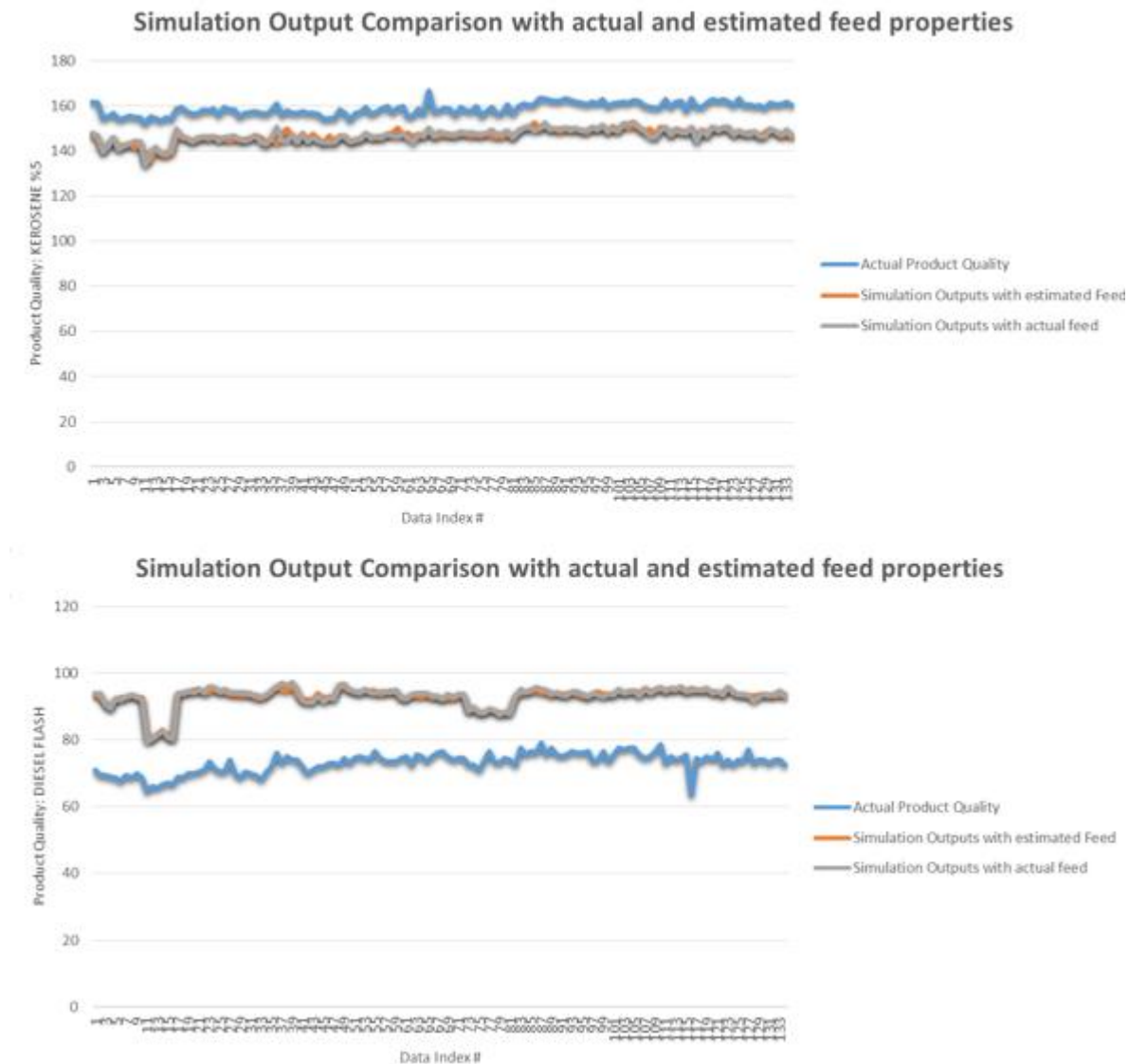


Figure 6.7 – Simulation outputs with estimated and actual feed properties (For Kerosene 5% boiling point and Diesel Flash point)

Chapter 7

Simulator Output Calibrator Model Building and The Model Results

7.1. Input Variable Selection for Simulation Error Calibrator Models

The outputs from the simulator are compared with the actual product qualities and as expected there are some deviation of estimated qualities from the actual qualities due to several reasons given in Chapter 1. In the following section, the methodology followed to estimate these errors are explained in the following sections.

The process simulator errors are labeled as “ R_i ” for i^{th} product quality and calculated as follows:

$$R_i(t) = Y_i(t) - W_i(t) \quad (7.1)$$

where, $Y_i(t)$ is the actual product quality for t^{th} observation and W_i is the estimated product quality for t^{th} observation given by simulator.

Suppose that we observe a quantitative response Y and p different predictors, $X_1, X_2, \dots, X_p$. In this thesis, the response are the product qualities that are listed in Table 7.1.

Table 7.1 – Response Variables

Response Variable	Description	Unit
Y_1	HSRN %5 Boiling Point	°C
Y_2	HSRN %95 Boiling Point	°C
Y_3	KEROSENE %5 Boiling Point	°C
Y_4	KEROSENE %95 Boiling Point	°C
Y_5	KEROSENE FLASH Point	°C
Y_6	DIESEL %5 Boiling Point	°C
Y_7	DIESEL %95 Boiling Point	°C
Y_8	DIESEL FLASH Point	°C
Y_9	BOTTOM %5 Boiling Point	-
Y_{10}	HSRN Density	API (@60F)
Y_{11}	KEROSENE Density	API (@60F)
Y_{12}	DIESEL Density	API (@60F)
Y_{13}	BOTTOM Density	API (@60F)

This quantitative response variable Y (product qualities) can be calculated with the following equation:

$$Y_i(t) = W_i(t) + R_i(t) \quad (7.2)$$

The outputs from the simulator are compared with the actual product qualities and as expected there are some deviation of estimated qualities from the actual qualities due to several reasons given in Chapter 1. In the following section, the methodology followed to estimate these errors are explained in the following sections.

The process simulator errors are labeled as “ E_i ” for i^{th} product quality and calculated as follows:

$$E_i(t) = Y_i(t) - W_i(t) \quad (7.3)$$

where, $Y_i(t)$ is the actual product quality for t^{th} observation and W_i is the estimated product quality for t^{th} observation given by simulator. It should be noted that $E_i(t)$ has no lag.

As well as the simulation error, $E_i(t)$, new simulation errors with lags are derived and calculated to be used in the simulation error modelling stage. These variables can be called as “simulation errors with lags” and the formulations for these derived variables are as follows:

$$E_{i,t-1}(t) = Y_i(t-1) - W_i(t-1) \quad (7.4)$$

$$E_{i,t-2}(t) = Y_i(t-2) - W_i(t-2) \quad (7.5)$$

$$E_{i,t-3}(t) = Y_i(t-3) - W_i(t-3) \quad (7.6)$$

$$E_{i,t-4}(t) = Y_i(t-4) - W_i(t-4) \quad (7.7)$$

$$E_{i,t-5}(t) = Y_i(t-5) - W_i(t-5) \quad (7.8)$$

$$E_{i,t-6}(t) = Y_i(t-6) - W_i(t-6) \quad (7.9)$$

Where,

- i: the product quality index (For example, $i=1$ refers to HSRN %5 boiling point. $i=1,2,\dots,13$)
- t: Observation number ($t=1,2,3,\dots,135$)
- E: Simulation error
- Y: Actual product quality
- W: Process simulator estimation for product quality

According to this nomenclature, for example, $E_{i,t-3}(10)$ is equal to $Y_i(7) - W_i(7)$, and it has 7 observation lag. Using these simulation errors with lag variables that inherently give information about the residual trend are derived. The formulations for these variables are:

$$\Delta_{i,2}(t) = \frac{E_{i,t-4}(t) + E_{i,t-3}(t)}{2} - \frac{E_{i,t-2}(t) + E_{i,t-1}(t)}{2} \quad (7.10)$$

$$\Delta_{i,3}(t) = \frac{E_{i,t-6}(t) + E_{i,t-5}(t) + E_{i,t-4}(t)}{3} - \frac{E_{i,t-3}(t) + E_{i,t-2}(t) + E_{i,t-1}(t)}{3} \quad (7.11)$$

Beside the residual variables, operational variables, feed characterization variables and simulation outputs can give information about the simulation error. Hence, the following table gives all input variables for simulation error models.

All possible input variables are given in Table 7.2. Since there are 63 input variable, it is needed to select a subset of relevant variables for use in model construction due to following reasons:

- Shorter training times,
- Improved generalization performance by reducing overfitting
- Enhanced interpretability of models by researchers/users

Some or many of these 63 variables used in the simulation error models are in fact not related with the response. Thus, including such irrelevant variables results in unnecessary complexity in the resulting model and it is possible to obtain a model that is more easily interpreted by removing these irrelevant variables. If number of observation is quite larger than the number of variables, the model will probably perform well on test observations, assuming that there is a relationship between the response and the predictors. However, if number of observations is not much larger than p as in our case, therefore a lot of variability is expected in the fit, leading to overfitting and consequently poor performance on test observations. By selecting proper input variables, we can enhance the generalization performance of the model so that substantial improvement in predicting the response for observations not used in model training is expected. In addition to enhanced generalization performance, shorter training times are achieved with fewer variables due to decreased model complexity.

In Section 7.2, 6 different approaches with different input variables and different modelling techniques are explained.

Table 7.2 – Possible Input Variables for Simulation Error Models

	Description	Original Variable	Corresponding Variable		Description	Original Variable	Corresponding Variable
Operative Inputs (Operational variables of fractionator process)	Feed flow rate	X ₁	X ₁	ASPEN Estimations	HSRN %5	W ₁	X ₃₄
	Feed Temperature	X ₂	X ₂		HSRN %95	W ₂	X ₃₅
	Stripping Steam Rate(kg/h)	X ₃	X ₃		KEROSENE %5	W ₃	X ₃₆
	Stripping Steam Temperature	X ₄	X ₄		KEROSENE %95	W ₄	X ₃₇
	P_bot	X ₅	X ₅		KEROSENE FLASH	W ₅	X ₃₈
	P_top	X ₆	X ₆		DIESEL %5	W ₆	X ₃₉
	dP_top	X ₇	X ₇		DIESEL %95	W ₇	X ₄₀
	Kerosene PA-1 Rate	X ₈	X ₈		DIESEL FLASH	W ₈	X ₄₁
	KERO PA-1_TRet(Pa)	X ₉	X ₉		BOTTOM %5	W ₉	X ₄₂
	KERO PA-2_Rate(Pa)	X ₁₀	X ₁₀		BOTTOM %95	W ₁₀	X ₄₃
	KERO PA-2_TRet(Pa)	X ₁₁	X ₁₁		HSRN API	W ₁₁	X ₄₄
	DIESEL PA-1_Rate(Pa)	X ₁₂	X ₁₂		KEROSENE API	W ₁₂	X ₄₅
	DIESEL PA-1_TRet(Pa)	X ₁₃	X ₁₃		DIESEL API	W ₁₃	X ₄₆
	DIESEL PA-2_Rate(Pa)	X ₁₄	X ₁₄		BOTTOM API	W ₁₄	X ₄₇
	DIESEL PA-2_TRet(Pa)	X ₁₅	X ₁₅	E _i (t-1)	HSRN %5	E ₁ (t-1)	X ₄₈
	OFF GAS Rate	X ₁₆	X ₁₆		HSRN %95	E ₂ (t-1)	X ₄₉
	HSRN Rate	X ₁₇	X ₁₇		KEROSENE %5	E ₃ (t-1)	X ₅₀
	KEROSENE Rate	X ₁₈	X ₁₈		KEROSENE %95	E ₄ (t-1)	X ₅₁
	DIESEL Rate	X ₁₉	X ₁₉		KEROSENE FLASH	E ₅ (t-1)	X ₅₂
	Frac.bottom Flow	X ₂₀	X ₂₀		DIESEL %5	E ₆ (t-1)	X ₅₃
	E-108 Outlet T	X ₂₁	X ₂₁		DIESEL %95	E ₇ (t-1)	X ₅₄
	Condenser T	X ₂₂	X ₂₂		DIESEL FLASH	E ₈ (t-1)	X ₅₅
Operative Inputs (Operational variables of reactor process)	HVGO Rate	F ₁	X ₂₃		BOTTOM %5	E ₉ (t-1)	X ₅₆
	C5 1.Yatak'a H2	F ₂	X ₂₄		BOTTOM %95	E ₁₀ (t-1)	X ₅₇
	C5 2.Yatak'a H2	F ₃	X ₂₅		HSRN API	E ₁₁ (t-1)	X ₅₈
	C6 1.Yatak'a H2	F ₄	X ₂₆		KEROSENE API	E ₁₂ (t-1)	X ₅₉
	C6 2.Yatak'a H2	F ₅	X ₂₇		DIESEL API	E ₁₃ (t-1)	X ₆₀
	C-5 1.Yatak WABT	F ₆	X ₂₈		BOTTOM API	E ₁₄ (t-1)	X ₆₁
	C-5 2.Yatak WABT	F ₇	X ₂₉		Delta 2 for Y _i	Δ1, 2	X ₆₂
	C-6 1.Yatak WABT	F ₈	X ₃₀		Delta 3 for Y _i	Δ1, 3	X ₆₃
	C-6 2.Yatak WABT	F ₉	X ₃₁				
	C-5 Giriş P	F ₁₀	X ₃₂				
	C-6 Giriş P	F ₁₁	X ₃₃				

7.2. Approaches for the Purpose of Modelling the Simulation Error

In this section, six different approaches to estimate the simulation error will be introduced and the corresponding performances will be given. The First approach is to use only simulation errors with lags and the other derived variables built from the simulation errors with lags in order to model the error between the actual and simulation output for the product quality of interest. In the second approach, only simulation output for i^{th} product quality will be used as an input variable. Linear regression, support vector regression with radial basis function kernel and artificial neural networks trained with resilient back propagation algorithm are used in the first and the second approaches. In the third approach, both simulation errors with lags and simulation output are taken as an input variable for prediction purpose. Similar to the first and second approach, the same algorithms, as well as the support vector regression with linear regression, will be used to train the models. In the fourth approach, in order to make a comparison with the models available in the literature, which use only operational variables to predict the product quality, operational variables will be used to estimate the simulation error. With this approach, the product quality will not be directly estimated; but, will be estimated by adding a simulation output to the estimated simulation error. Therefore, it will be analyzed if there is any improvement in predicting the product qualities with this approach. In the fifth approach, all variables including simulation errors with lags, the derived variables comprised of the simulation errors with lags, operational variables and simulation outputs will be chosen as an input variable and the simulation error will be modelled by the random forest algorithm. Since there are many features in the fourth and fifth approach, then random forest algorithm is chosen as a model training algorithm, which inherently include feature ranking property, are ease of use and robust. In the final approach, the random forest variable importance (Ranking according to the increase in mean of the error of a tree) graph will be considered to select the predictor variables. If there is no significant increase in mean of the error of a tree for a variable, then it will not be considered. In this approach, linear regression, SVR-RBF, and ANN with back transient algorithms are the model training algorithms.

	<u>Label Variable</u>	<u>Predictors</u>	<u># of Predictors</u>	<u>Model Types</u>
Approach 1 [Lagged simulation errors as inputs]	Simulation error for i^{th} product quality	Simulation errors with lags and derived variables comprised of them	3	Linear Regression
				SVR (RBF Kernel)
				ANN (With Back transient Algorithm)
Approach 2 [Simulation output as input]	Simulation error for i^{th} product quality	Corresponding simulation output for modeling i^{th} product quality	1	Linear Regression
				SVR (RBF Kernel)
				ANN (With Back transient Algorithm)
Approach 3 [Lagged simulation errors and simulation outputs as inputs]	Simulation error for i^{th} product quality	Simulation errors with lags and derived variables comprised of them + Corresponding simulation output for modeling i^{th} product quality	4	Linear Regression
				SVR (RBF Kernel)
				SVR (Linear Kernel)
				ANN (With Back transient Algorithm)
Approach 4 [Operational variables only as inputs]	Simulation error for i^{th} product quality	All Operational variables (Reactor + Fractionator Processes)	33	Random Forest
Approach 5 [All variables as inputs]	Simulation error for i^{th} product quality	All variables	63	Random Forest
Approach 6 [Selected variables from all variables set as inputs]	Simulation error for i^{th} product quality	Selected variables from all variable set	10 to 1	Linear Regression
				SVR (RBF Kernel)
				ANN (With Back transient Algorithm)

Figure 7.1 – Simulation error modelling approaches

Figure 7.1 summarizes all approaches by tabulating all input predictors and model types for each approach. In the following sections of the report, the prediction accuracies of the models will be summarized. Therefore, for each product property, models will be evaluated by using certain criteria and the prediction trends for the selected models for each property will be shown both for training and testing data sections.

7.2.1. Results for Approach I : Modelling Simulation Errors with Residuals Variables

Recall that in this approach, a study was conducted to see if the simulation errors, E_i 's, are related with the previous simulation errors, $E_i(t - 1)$, $\Delta_{2,i}$, and $\Delta_{3,i}$, and how the prediction accuracy will be by only using this variables. In Table 7.3, training/testing/10-Fold CV errors in root mean squared error, and the final model parameters after the best model parameters are found by grid search algorithm combined with 10-fold cross validation are be tabulated.

Table 7.3 – Result Summaries for Approach I

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3
É1	HSRN %5	Linear Regression	3	1.73	1.83	1.82					X48	X62	X63
É1	HSRN %5	SVR-RBF	3	1.37	1.86	1.73	0.8	4.64E+00	3.59E-01	31.0	X48	X62	X63
É1	HSRN %5	ANN	3	1.60	2.12	1.76	5				X48	X62	X63
É2	HSRN %95	Linear Regression	3	2.36	1.99	2.50					X49	X62	X63
É2	HSRN %95	SVR-RBF	3	1.37	1.86	1.73	0.8	4.64E+00	3.59E-01	31	X49	X62	X63
É2	HSRN %95	ANN	3	2.00	2.14	2.14	5				X49	X62	X63
É3	KEROSENE %5	Linear Regression	3	1.99	1.54	2.14					X50	X62	X63
É3	KEROSENE %5	SVR-RBF	3	1.99	1.53	2.01	0.8	7.74E+02	1.00E-04	24	X50	X62	X63
É3	KEROSENE %5	ANN	3	1.46	1.63	1.94	3				X50	X62	X63
É4	KEROSENE %95	Linear Regression	3	2.53	3.14	2.60					X51	X62	X63
É4	KEROSENE %95	SVR-RBF	3	1.32	3.13	2.55	0.6	4.64E+00	2.15E+01	54	X51	X62	X63
É4	KEROSENE %95	ANN	3	1.83	6.14	2.66	5				X51	X62	X63
É5	KEROSENE Flash	Linear Regression	3	1.82	1.42	1.91					X52	X62	X63
É5	KEROSENE Flash	SVR-RBF	3	1.83	1.38	1.88	0.6	7.74E+02	1.00E-04	51	X52	X62	X63
É5	KEROSENE Flash	ANN	3	1.80	1.45	1.69	12				X52	X62	X63
É6	DIESEL %5	Linear Regression	3	5.13	3.15	5.23					X53	X62	X63
É6	DIESEL %5	SVR-RBF	3	4.79	3.41	5.11	0	7.74E+02	5.99E-03	94	X53	X62	X63
É6	DIESEL %5	ANN	3	4.77	3.46	4.24	12				X53	X62	X63

Table 7.3 (Cont'd) – Model Summaries for Approach I

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3
É7	DIESEL %95	Linear Regression	3	6.62	5.75	7.12					X54	X62	X63
É7	DIESEL %95	SVR-RBF	3	6.77	5.43	6.70	2	1.67E+01	4.64E-02	2	X54	X62	X63
É7	DIESEL %95	ANN	3	5.50	6.41	6.25	1				X54	X62	X63
É8	DIESEL Flash	Linear Regression	3	2.19	3.02	2.87					X55	X62	X63
É8	DIESEL Flash	SVR-RBF	3	2.09	2.98	2.20	2	5.99E+01	5.99E-03	37	X55	X62	X63
É8	DIESEL Flash	ANN	3	2.16	3.12	2.25	11				X55	X62	X63
É9	Bottom %5	Linear Regression	3	9.73	11.54	10.40					X56	X62	X63
É9	Bottom %5	SVR-RBF	3	9.75	11.29	9.92	0.8	1.29E+00	4.64E-02	27	X56	X62	X63
É9	Bottom %5	ANN	3	9.74	11.52	9.83	2				X56	X62	X63
É10	HSRN API	Linear Regression	3	0.64	0.58	0.70					X58	X62	X63
É10	HSRN API	SVR-RBF	3	0.65	0.59	0.66	0.4	5.99E+01	7.74E-04	62	X58	X62	X63
É10	HSRN API	ANN	3	0.63	0.57	0.56	14				X58	X62	X63
É11	KEROSENE API	Linear Regression	3	0.49	0.37	0.57					X59	X62	X63
É11	KEROSENE API	SVR-RBF	3	0.46	0.37	0.50	0.4	7.74E+02	5.99E-03	62	X59	X62	X63
É11	KEROSENE API	ANN	3	0.44	0.34	0.42	16				X59	X62	X63
É12	DIESEL API	Linear Regression	3	0.67	0.48	0.77					X60	X62	X63
É12	DIESEL API	SVR-RBF	3	0.65	0.48	0.68	0.2	5.99E+01	5.99E-03	76	X60	X62	X63
É12	DIESEL API	ANN	3	0.55	0.53	0.59	16				X60	X62	X63
É13	BOTTOM API	Linear Regression	3	0.30	0.24	0.33					X61	X62	X63
É13	BOTTOM API	SVR-RBF	3	0.25	0.26	0.31	1.2	1.29E+00	2.78E+00	23	X61	X62	X63
É13	BOTTOM API	ANN	3	0.20	0.36	0.30	1				X61	X62	X63

7.2.2. Results for Approach II : Modelling Simulation Errors with Simulation Output

In the second approach the simulation errors E_i 's are modelled by using corresponding simulation outputs W_i 's. In Table 7.4, all model results are tabulated.

Table 7.4 – Result Summaries for Approach II

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1
Ē1	HSRN %5	Linear Regression	1	0.78	0.99	0.79					X34
Ē1	HSRN %5	SVR-RBF	1	0.76	0.94	0.77	0.4	2.15E+02	5.99E-03	24	X34
Ē1	HSRN %5	ANN	1	0.76	0.97	0.71	5				X34
Ē2	HSRN %95	Linear Regression	1	1.61	2.38	1.69					X35
Ē2	HSRN %95	SVR-RBF	1	1.62	2.47	1.64	0.6	1.00E+04	1.00E-04	31	X35
Ē2	HSRN %95	ANN	1	1.61	2.35	1.59	20				X35
Ē3	KEROSENE %5	Linear Regression	1	1.79	1.78	1.85					X36
Ē3	KEROSENE %5	SVR-RBF	1	1.61	1.69	1.66	0.6	2.15E+02	3.59E-01	38	X36
Ē3	KEROSENE %5	ANN	1	1.66	1.70	1.59	4				X36
Ē4	KEROSENE %95	Linear Regression	1	2.58	2.96	2.65					X37
Ē4	KEROSENE %95	SVR-RBF	1	1.57	3.16	1.97	0	1.29E+00	1.67E+02	94	X37
Ē4	KEROSENE %95	ANN	1	2.57	2.91	2.55	14				X37
Ē5	KEROSENE Flash	Linear Regression	1	1.57	1.45	1.61					X38
Ē5	KEROSENE Flash	SVR-RBF	1	1.49	1.31	1.56	0.8	2.78E+03	3.59E-01	32	X38
Ē5	KEROSENE Flash	ANN	1	1.55	1.37	1.33	14				X38
Ē6	DIESEL %5	Linear Regression	1	5.13	3.15	5.18					X39
Ē6	DIESEL %5	SVR-RBF	1	4.76	3.71	5.08	0	3.59E-01	2.15E+01	94	X39
Ē6	DIESEL %5	ANN	1	5.13	3.16	3.89	10				X39

Table 7.4 (Cont'd) – Result Summaries for Approach II

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1
É7	DIESEL %95	Linear Regression	1	3.44	2.21	3.51					X40
É7	DIESEL %95	SVR-RBF	1	3.44	2.18	3.45	0	1.67E+01	5.99E-03	94	X40
É7	DIESEL %95	ANN	1	3.43	2.21	2.69	14				X40
É8	DIESEL Flash	Linear Regression	1	2.52	2.67	2.57					X41
É8	DIESEL Flash	SVR-RBF	1	2.42	2.72	2.46	2	1.67E+01	5.99E-03	44	X41
É8	DIESEL Flash	ANN	1	2.39	2.73	2.21	15				X41
É9	Bottom %5	Linear Regression	1	10.25	10.99	10.40					X42
É9	Bottom %5	SVR-RBF	1	10.03	10.61	10.09	0	7.74E+02	5.99E-03	94	X42
É9	Bottom %5	ANN	1	9.98	10.95	7.76	6				X42
É10	HSRN API	Linear Regression	1	0.34	0.38	0.35					X44
É10	HSRN API	SVR-RBF	1	0.34	0.37	0.34	0.2	5.99E+01	5.99E-03	70	X44
É10	HSRN API	ANN	1	0.34	0.40	0.30	1				X44
É11	KEROSENE API	Linear Regression	1	0.44	0.31	0.45					X45
É11	KEROSENE API	SVR-RBF	1	0.37	0.33	0.38	0.4	7.74E+02	3.59E-01	52	X45
É11	KEROSENE API	ANN	1	0.39	0.34	0.36	16				X45
É12	DIESEL API	Linear Regression	1	0.41	0.32	0.42					X46
É12	DIESEL API	SVR-RBF	1	0.36	0.36	0.37	0.2	1.00E+04	4.64E-02	61	X46
É12	DIESEL API	ANN	1	0.38	0.34	0.33	3				X46
É13	BOTTOM API	Linear Regression	1	0.28	0.47	0.29					X47
É13	BOTTOM API	SVR-RBF	1	0.27	0.43	0.28	0.2	1.29E+00	3.59E-01	72	X47
É13	BOTTOM API	ANN	1	0.26	0.52	0.25	5				X47

7.2.3. Results for Approach III : Modelling Simulation Errors with Simulation Error with 1 lag and Simulation Output

In the third approach, feature sets of approach I and II are used as predictor variable set. The results for this approach are shown in Table 7.5.

Table 7.5 – Result Summaries for Approach III

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3	VAR 4
É1	HSRN %5	Linear Regression	4	0.70	0.92	0.74					X34	X48	X62	X63
É1	HSRN %5	ANN	4	0.58	0.94	0.66	9				X34	X48	X62	X63
É1	HSRN %5	SVR-RBF	4	0.68	0.87	0.70	0	774.2636827	0.000774264	94	X34	X48	X62	X63
É1	HSRN %5	SVR-Linear Kernel	4	0.71	0.87	0.73	0.4	1.291549665		22	X34	X48	X62	X63
É2	HSRN %95	Linear Regression	4	1.61	2.42	1.75					X35	X49	X62	X63
É2	HSRN %95	ANN	4	1.54	2.34	1.85	6				X35	X49	X62	X63
É2	HSRN %95	SVR-RBF	4	1.62	2.21	1.69	0.2	774.2636827	0.0001	70	X35	X49	X62	X63
É2	HSRN %95	SVR-Linear Kernel	4	1.61	2.36	1.69	0.2	774.2636827		70	X35	X49	X62	X63
É3	KEROSENE %5	Linear Regression	4	1.76	1.73	1.86					X36	X50	X62	X63
É3	KEROSENE %5	ANN	4	1.56	1.67	1.53	4				X36	X50	X62	X63
É3	KEROSENE %5	SVR-RBF	4	1.56	1.65	1.83	0	16.68100537	0.046415888	94	X36	X50	X62	X63
É3	KEROSENE %5	SVR-Linear Kernel	4	1.79	1.64	1.85	0.4	0.1		56	X36	X50	X62	X63
É4	KEROSENE %95	Linear Regression	4	2.53	3.12	2.65					X37	X51	X62	X63
É4	KEROSENE %95	ANN	4	1.39	4.69	2.80	1				X37	X51	X62	X63
É4	KEROSENE %95	SVR-RBF	4	2.56	3.05	2.51	0	4.641588834	21.5443469	94	X37	X51	X62	X63
É4	KEROSENE %95	SVR-Linear Kernel	4	2.58	3.11	2.62	1.6	1.291549665	13		X37	X51	X62	X63

Table 7.5 (Cont'd) – Result Summaries for Approach IV

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3	VAR 4
É5	KEROSENE Flash	Linear Regression	4	1.56	1.46	1.65					X38	X52	X62	X63
É5	KEROSENE Flash	ANN	4	1.30	1.54	1.30	19				X38	X52	X62	X63
É5	KEROSENE Flash	SVR-RBF	4	1.47	1.31	1.58	0.6	215.443469	0.005994843	44	X38	X52	X62	X63
É5	KEROSENE Flash	SVR-Linear Kernel	4	1.57	1.46	1.60	0.8	0.359381366	34	34	X38	X52	X62	X63
É6	DIESEL %5	Linear Regression	4	5.09	3.14	5.26					X39	X53	X62	X63
É6	DIESEL %5	ANN	4	4.76	3.38	3.88	7				X39	X53	X62	X63
É6	DIESEL %5	SVR-RBF	4	5.16	3.16	5.18	1.4	0.1	0.0001	10	X39	X53	X62	X63
É6	DIESEL %5	SVR-Linear Kernel	4	5.12	3.17	5.25	0.6	0.359381366		46	X39	X53	X62	X63
É7	DIESEL %95	Linear Regression	4	3.44	2.21	3.61					X40	X54	X62	X63
É7	DIESEL %95	ANN	4	2.93	2.45	2.94	2				X40	X54	X62	X63
É7	DIESEL %95	SVR-RBF	4	3.07	2.18	3.52	0.2	4.641588834	0.046415888	57	X40	X54	X62	X63
É7	DIESEL %95	SVR-Linear Kernel	4	3.45	2.22	3.53	0.2	0.1		54	X40	X54	X62	X63
É8	DIESEL Flash	Linear Regression	4	1.93	2.85	2.06					X41	X55	X62	X63
É8	DIESEL Flash	ANN	4	1.61	4.01	1.85	11				X41	X55	X62	X63
É8	DIESEL Flash	SVR-RBF	4	1.66	2.83	1.75	0	16.68100537	0.005994843	94	X41	X55	X62	X63
É8	DIESEL Flash	SVR-Linear Kernel	4	1.94	2.84	2.02	2	0.1		32	X41	X55	X62	X63
É9	Bottom %5	Linear Regression	4	9.71	11.61	10.3					X42	X56	X62	X63
É9	Bottom %5	ANN	4	8.54	12.09	10.15	1				X42	X56	X62	X63
É9	Bottom %5	SVR-RBF	4	9.80	11.37	9.86	0.8	10000	0.0001	31	X42	X56	X62	X63
É9	Bottom %5	SVR-Linear Kernel	4	9.81	11.35	9.85	0.8	1.291549665		32	X42	X56	X62	X63

Table 7.5 (Cont'd) – Result Summaries for Approach IV

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10- FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3	VAR 4
Ē10	HSRN API	Linear Regression	4	0.32	0.38	0.34					X44	X58	X62	X63
Ē10	HSRN API	ANN	4	0.30	0.38	0.31	7				X44	X58	X62	X63
Ē10	HSRN API	SVR-RBF	4	0.32	0.36	0.33	0.2	774.2636827	0.0001	64	X44	X58	X62	X63
Ē10	HSRN API	SVR-Linear Kernel	4	0.32	0.40	0.33	0.2	10000		61	X44	X58	X62	X63
Ē11	KEROSENE API	Linear Regression	4	0.41	0.33	0.45					X45	X59	X62	X63
Ē11	KEROSENE API	ANN	4	0.34	0.36	0.36	3				X45	X59	X62	X63
Ē11	KEROSENE API	SVR-RBF	4	0.38	0.32	0.40	0.8	10000	0.000774264	23	X45	X59	X62	X63
Ē11	KEROSENE API	SVR-Linear Kernel	4	0.42	0.33	0.44	0.2	0.1		71	X45	X59	X62	X63
Ē12	DIESEL API	Linear Regression	4	0.40	0.33	0.44					X46	X60	X62	X63
Ē12	DIESEL API	ANN	4	0.33	0.33	0.34	17				X46	X60	X62	X63
Ē12	DIESEL API	SVR-RBF	4	0.34	0.33	0.37	0.4	2782.559402	0.000774264	40	X46	X60	X62	X63
Ē12	DIESEL API	SVR-Linear Kernel	4	0.40	0.32	0.43	0.2	0.359381366		64	X46	X60	X62	X63
Ē13	BOTTOM API	Linear Regression	4	0.27	0.41	0.29					X47	X61	X62	X63
Ē13	BOTTOM API	ANN	4	0.17	0.53	0.25	4				X47	X61	X62	X63
Ē13	BOTTOM API	SVR-RBF	4	0.26	0.36	0.27	0.4	1.291549665	0.046415888	58	X47	X61	X62	X63
Ē13	BOTTOM API	SVR-Linear Kernel	4	0.27	0.41	0.28	0.4	0.1		58	X47	X61	X62	X63

7.2.4. Results for Approach IV : Modelling Simulation Errors with Operational Variables

In literature, many studies were conducted to estimate the product qualities and some of the studies are given in Chapter 1. A study was conducted to see whether it is possible to estimate the simulation errors accurately by only using operational variables. Hence, we can measure the extent of improvement in prediction accuracies with the addition of other variables, namely simulation errors with lags and simulation outputs. The following table summarizes the model performances, whose model inputs are only operational variables, such as temperature, flow, and pressure readings.

Table 7.6 – Result Summaries for Approach IV

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10-FOLD CV ERROR [RMSE]	RF: MTRY	RF: NTREE	VARIABLES
É1	HSRN %5	Random Forest	33	0.64	1.65	1.53	10	100	X1 , X2, ..., X33
É2	HSRN %95	Random Forest	33	0.85	2.06	1.99	15	1000	X1 , X2, ..., X33
É3	KEROSENE %5	Random Forest	33	0.75	1.52	1.69	5	500	X1 , X2, ..., X33
É4	KEROSENE %95	Random Forest	33	0.96	2.44	2.37	33	50	X1 , X2, ..., X33
É5	KEROSENE Flash	Random Forest	33	0.65	1.79	1.61	20	500	X1 , X2, ..., X33
É6	DIESEL %5	Random Forest	33	2.92	3.02	4.80	1	1000	X1 , X2, ..., X33
É7	DIESEL %95	Random Forest	33	1.89	3.00	4.61	33	100	X1 , X2, ..., X33
É8	DIESEL Flash	Random Forest	33	0.58	2.83	1.40	15	1000	X1 , X2, ..., X33
É9	Bottom %5	Random Forest	33	5.58	11.21	9.27	1	100	X1 , X2, ..., X33
É10	HSRN API	Random Forest	33	0.17	0.63	0.43	33	1000	X1 , X2, ..., X33
É11	KEROSENE API	Random Forest	33	0.15	0.32	0.39	25	500	X1 , X2, ..., X33
É12	DIESEL API	Random Forest	33	0.18	0.41	0.43	10	100	X1 , X2, ..., X33
É13	BOTTOM API	Random Forest	33	0.12	0.26	0.27	10	50	X1 , X2, ..., X33

7.2.5. Results for Approach V : Modelling Simulation Errors with All Variables

The case that includes all variables is the fifth approach and in that approach random forest models are trained since there are many predictor and we do not have enough observation. Table 7.7 gives the results for the fifth approach. After model parameters for random forests , namely the number of variables randomly sampled as candidates at each split and number of total trees in the forest, are found by cross-validation, then the forest is retrained with the final parameters. At the same time, variable importance factors (VI) are calculated at this stage to be used in sixth approach for variable ranking and selection purposes.

Table 7.7 – Result Summaries for Approach V

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10-FOLD CV ERROR [RMSE]	RF: MTRY	RF: NTREE	VARIABLES
É1	HSRN %5	Random Forest	63	0.32	1.10	0.75	50	1000	X1 , X2, ..., X63
É2	HSRN %95	Random Forest	63	0.73	1.93	1.73	30	100	X1 , X2, ..., X63
É3	KEROSENE %5	Random Forest	63	0.67	1.53	1.58	15	1000	X1 , X2, ..., X63
É4	KEROSENE %95	Random Forest	63	0.94	2.27	2.28	20	50	X1 , X2, ..., X63
É5	KEROSENE Flash	Random Forest	63	0.59	1.22	1.49	63	1000	X1 , X2, ..., X63
É6	DIESEL %5	Random Forest	63	2.65	3.07	4.69	1	1000	X1 , X2, ..., X63
É7	DIESEL %95	Random Forest	63	1.51	2.14	3.75	63	500	X1 , X2, ..., X63
É8	DIESEL Flash	Random Forest	63	0.57	2.82	1.39	15	1000	X1 , X2, ..., X63
É9	Bottom %5	Random Forest	63	4.48	11.05	9.25	3	100	X1 , X2, ..., X63
É10	HSRN API	Random Forest	63	0.14	0.39	0.33	50	50	X1 , X2, ..., X63
É11	KEROSENE API	Random Forest	63	0.14	0.28	0.34	15	1000	X1 , X2, ..., X63
É12	DIESEL API	Random Forest	63	0.15	0.35	0.35	20	500	X1 , X2, ..., X63
É13	BOTTOM API	Random Forest	63	0.11	0.25	0.26	40	100	X1 , X2, ..., X63

7.2.6. Results for Approach VI : Modelling Simulation Errors with Selected Variables

Variable ranking information is taken from the previous case, namely case five, and then all simulation errors for each property is modelled with by considering these variable ranking. For each property, simulation errors are modelled with linear regression, ANN (with back transient algorithm) and SVR-RBF algorithm whose input variables are decided by analyzing the variable ranking information. The following figure shows the variable ranking information by giving the increase in mean of the error of a tree in the forest (%IncMSE) for HSRN 5%(Figure on the left) and kerosene flash (Figure on the right).According to this figure, it is logical to use only X_{34} and X_{44} variables as input variables for HSRN 5%, and to use X_{38} , X_{21} , X_{36} , and X_{46} for kerosene flash, since after these variables, there is no significant variable and the curve almost flattening. Similarly, Similar approach is applied to the other product qualities and the input variables are chosen according to this rule. Table 7.8 summarizes all model results.

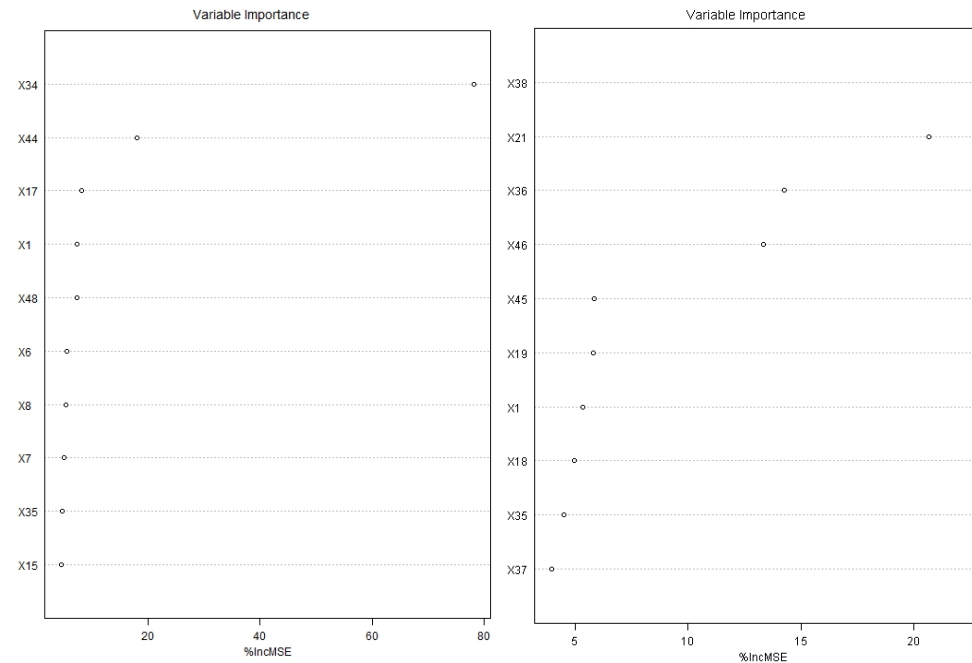


Figure 7.2 – Variable rankings obtained from random forest ranking for simulation error model (Figure on the left: HSRN 5% boiling point, Figure on the right: Kerosene Flash)

Table 7.8 – Result Summaries for Approach VI

	PRODUCT PROPERTY	MODEL	# OF INPUT VAR	RMSE TRAIN	RMSE TEST	10-FOLD CV ERROR [RMSE]	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3	VAR 4	VAR 5
É1	HSRN %5	L.Regression	2	0,77	0,89	0,79					X34	X44			
É1	HSRN %5	SVR-RBF	2	0,76	0,90	0,77	0	59,94842503	0,005994843	94	X34	X44			
É1	HSRN %5	ANN	2	0,75	0,94	0,68	13				X34	X44			
É2	HSRN %95	L.Regression	2	1,56	1,85	1,63					X35	X24			
É2	HSRN %95	SVR-RBF	2	1,59	1,86	1,58	0,8	774,2636827	0,0001	19	X35	X24			
É2	HSRN %95	ANN	2	1,56	1,88	1,53	17				X35	X24			
É3	KEROSENE %5	L.Regression	5	1,63	2,55	1,76					X40	X26	X36	X38	X46
É3	KEROSENE %5	SVR-RBF	5	1,42	1,48	1,63	0,4	4,641588834	0,046415888	55	X40	X26	X36	X38	X46
É3	KEROSENE %5	ANN	5	1,19	2,11	1,51	4				X40	X26	X36	X38	X46
É4	KEROSENE %95	L.Regression	2	2,26	2,47	2,34					X40	X37			
É4	KEROSENE %95	SVR-RBF	2	1,96	2,29	2,27	0	0,359381366	2,782559402	94	X40	X37			
É4	KEROSENE %95	ANN	2	1,96	2,24	2,20	3				X40	X37			
É5	KEROSENE Flash	L.Regression	4	1,28	1,22	1,34					X38	X21	X36	X46	
É5	KEROSENE Flash	SVR-RBF	4	1,30	1,22	1,31	0	774,2636827	0,0001	94	X38	X21	X36	X46	
É5	KEROSENE Flash	ANN	4	1,26	1,17	1,23	3				X38	X21	X36	X46	
É6	DIESEL %5	L.Regression	1	5,04	3,39	5,09					X55				
É6	DIESEL %5	SVR-RBF	1	4,87	6,74	4,98	1,2	774,2636827	0,359381366	13	X55				
É6	DIESEL %5	ANN	1	4,87	3,21	3,70	10				X55				
É7	DIESEL %95	L.Regression	1	3,44	2,21	3,51					X40				
É7	DIESEL %95	SVR-RBF	1	3,44	2,18	3,45	0	16,68100537	0,005994843	94	X40				
É7	DIESEL %95	ANN	1	3,43	2,21	2,69	14				X40				
É8	DIESEL Flash	L.Regression	5	1,32	2,72	1,43					X18	X10	X8	X17	X37
É8	DIESEL Flash	SVR-RBF	5	1,22	3,19	1,31	0,4	774,2636827	0,0001	67	X18	X10	X8	X17	X37
É8	DIESEL Flash	ANN	5	1,18	3,83	1,36	5				X18	X10	X8	X17	X37
É9	Bottom %5	L.Regression	1	10,03	11,93	10,10					X21				
É9	Bottom %5	SVR-RBF	1	10,05	11,79	10,08	0,8	59,94842503	0,005994843	27	X21				
É9	Bottom %5	ANN	1	10,03	11,53	7,76	6				X21				
É10	HSRN API	L.Regression	5	0,27	0,39	0,29					X44	X17	X34	X20	X46
É10	HSRN API	SVR-RBF	5	0,24	0,42	0,27	0	10000	0,000774264	94	X44	X17	X34	X20	X46
É10	HSRN API	ANN	5	0,26	0,40	0,27	7				X44	X17	X34	X20	X46
É11	KEROSENE API	L.Regression	3	0,39	0,28	0,40					X46	X37	X45		
É11	KEROSENE API	SVR-RBF	3	0,35	0,25	0,36	0,4	774,2636827	0,005994843	53	X46	X37	X45		
É11	KEROSENE API	ANN	3	0,35	0,29	0,31	19				X46	X37	X45		
É12	DIESEL API	L.Regression	1	0,41	0,32	0,42					X46				
É12	DIESEL API	SVR-RBF	1	0,36	0,36	0,37	0,2	10000	0,046415888	61	X46				
É12	DIESEL API	ANN	1	0,38	0,34	0,33	3				X46				
É13	BOTTOM API	L.Regression	1	0,28	0,47	0,29					X47				
É13	BOTTOM API	SVR-RBF	1	0,27	0,43	0,28					X47				
É13	BOTTOM API	ANN	1	0,26	0,52	0,25					X47				

7.3. Final Model Selection and Performance Evaluations

In the following sections, the way the final models are chosen for each product quality will be described and the results for training and testing zones will be shown. In Section 7.3.2, the errors will be compared with the repeatability and reproducibility in measurements to discuss whether it is possible to enhance the model performance.

The model selection criteria are as follows:

- Choose the model with lowest 10-fold CV error
- If the 10-fold CV errors are close to each other, choose the model with less complexity

7.3.1. Sample Final Model Selection (Diesel %5 Boiling Point)

Table 7.9 shows the model performances for estimating the simulation error for diesel %5 by giving training, testing and 10-fold CV errors, all in RMSE. Cross validation errors are ordered in the table. According to this table, due to having smaller 10-fold CV error and less complexity, it should be proper to choose the first model found by approach 6.

Table 7.9 – Model results for Diesel 5% boiling points for all approaches

Approach	Label	PREDICTED PROPERTY	Model	# of Input Variable	RMSE Train	RMSE Test	10-Fold CV Error [RMSE]
Approach 6	$\hat{E}_6$	DIESEL %5	ANN	1	4,87	3,21	3,70*
Approach 3	$\hat{E}_6$	DIESEL %5	ANN	4	4,76	3,38	3,88
Approach 2	$\hat{E}_6$	DIESEL %5	ANN	1	5,13	3,16	3,89
Approach 1	$\hat{E}_6$	DIESEL %5	ANN	3	4,77	3,46	4,24
Approach 5	$\hat{E}_6$	DIESEL %5	Random Forest	63	2,65	3,07	4,69
Approach 4	$\hat{E}_6$	DIESEL %5	Random Forest	33	2,92	3,02	4,80
Approach 6	$\hat{E}_6$	DIESEL %5	SVR-RBF	1	4,87	6,74	4,98
Approach 2	$\hat{E}_6$	DIESEL %5	SVR-RBF	1	4,76	3,71	5,08
Approach 6	$\hat{E}_6$	DIESEL %5	L.Reggression	1	5,04	3,39	5,09
Approach 1	$\hat{E}_6$	DIESEL %5	SVR-RBF	3	4,79	3,41	5,11
Approach 3	$\hat{E}_6$	DIESEL %5	SVR-RBF	4	5,16	3,16	5,18
Approach 2	$\hat{E}_6$	DIESEL %5	Linear Regression	1	5,13	3,15	5,18
Approach 1	$\hat{E}_6$	DIESEL %5	Linear Regression	3	5,13	3,15	5,23
Approach 3	$\hat{E}_6$	DIESEL %5	SVR-Linear Kernel	4	5,12	3,17	5,25
Approach 3	$\hat{E}_6$	DIESEL %5	Linear Regression	4	5,09	3,14	5,26

Model parameters for final model of Diesel %5 is given in the following table.

Table 7.10 – Summary for final Diesel %5 simulation error model

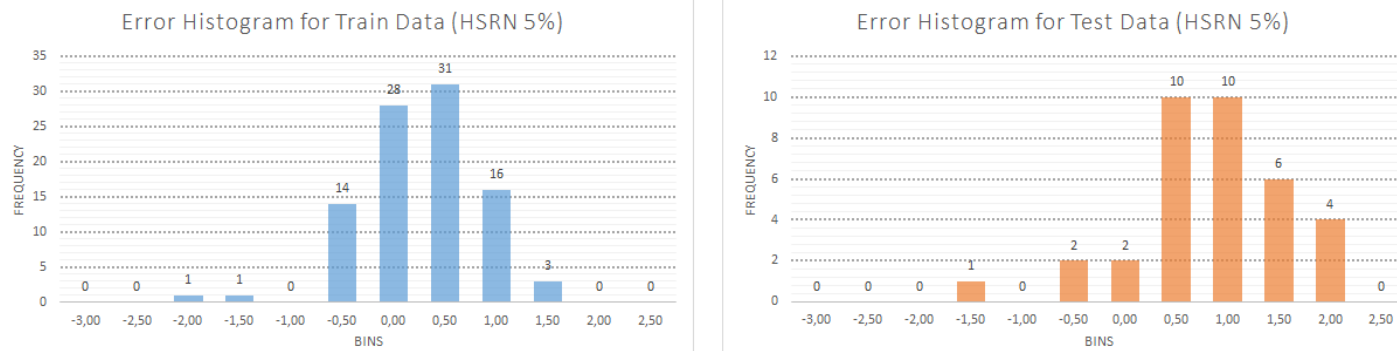
Model Label	$\hat{\epsilon}_6$	
Property	Diesel %5	
Model Type	Artificial Neural Network	
RMSE (Train)	4.87	
RMSE (Test)	3.21	
	Variable	Description
Predictor 1	X55	Simulation error with lag for diesel flash ($\epsilon_6(t-1)$)

All models chosen as final models to predict simulation errors are tabulated in the Table 7.11. After the selection of the models, the residual analysis is carried out to assess whether the observed errors is consistent with stochastic error. We should not be able to predict the error for any given observation. Residuals can be thought of as elements of variation unexplained by the fitted model. Since this is a form of error, the same general assumptions apply to the group of residuals that we typically use for errors in general: one expects them to be (roughly) normal and (approximately) independently distributed with a mean of 0 and some constant variance. The distribution of errors showing bell-shaped pattern for both data, meaning that the level of the error is independent of when the observation occurred in the study, or the size of the observation being predicted, or even the factor settings involved in making the prediction. There are several ways to judge the normality of the distribution of a group of residuals. One can also use normal probability plots, dot plots or histograms. In this thesis to check if there is a bell-shaped pattern in errors, error histograms for both training and testing data for each product quality are plotted and given in Figure 7.3.

After checking the normality of the errors, the percentage of the response variable variation that is explained by the final models are calculated by using Equation 6.6 and the R^2 values for training and testing data are given in Table 7.11.

Table 7.11 – Summary of final models for prediction simulation errors

ABEL	PREDICTED PROPERTY	MODEL	RMSE TRAIN	RMSE TEST	10-FOLD CV ERROR [RMSE]	R ² TRAIN	R ² TEST	SVM:EPSILON RF: MTRY LR: - ANN: # OF NEURON	SVM: C RF: NTREE LR: -	SVM : GAMMA RF: - LR: -	SVM : # OF SV RF: - LR: -	VAR 1	VAR 2	VAR 3	VAR 4	VAR 5
É1	HSRN %5	ANN	0,58	0,94	0,66	0,92	0,82	9				X34	X48	X62	X63	
É2	HSRN %95	ANN	1,56	1,88	1,53	0,60	0,65	17				X35	X24			
É3	KEROSENE %5	SVR-RBF	1,42	1,48	1,63	0,57	0,52	0,4	4,641588834	0,046415888	55	X40	X26	X36	X38	X46
É4	KEROSE. %95	ANN	1,96	2,24	2,20	0,52	0,50	3				X40	X37			
É5	KEROSE. Flash	ANN	1,26	1,17	1,23	0,56	0,46	3				X38	X21	X36	X46	
É6	DIESEL %5	ANN	4,87	3,21	3,70	0,10	0,10	10				X55				
É7	DIESEL %95	SVR-RBF	3,44	2,18	3,45	0,74	0,81	0	16,68100537	0,005994843	94	X40				
É8	DIESEL Flash	SVR-RBF	1,22	3,19	1,31	0,90	0,60	0,4	774,2636827	0,0001	67	X18	X10	X8	X17	X37
É9	Bottom %5	ANN	9,98	10,95	7,76	0,05	0,04	6				X42				
É10	HSRN API	ANN	0,26	0,40	0,27	0,87	0,81	7				X44	X17	X34	X20	X46
É11	KEROSENE API	ANN	0,35	0,29	0,31	0,65	0,79	19				X46	X37	X45		
É12	DIESEL API	ANN	0,38	0,34	0,33	0,78	0,79	3				X46				
É13	BOTTOM API	ANN	0,26	0,52	0,25	0,36	0,10	5				X47				

**Figure 7.3** – Error histograms of final simulation error models

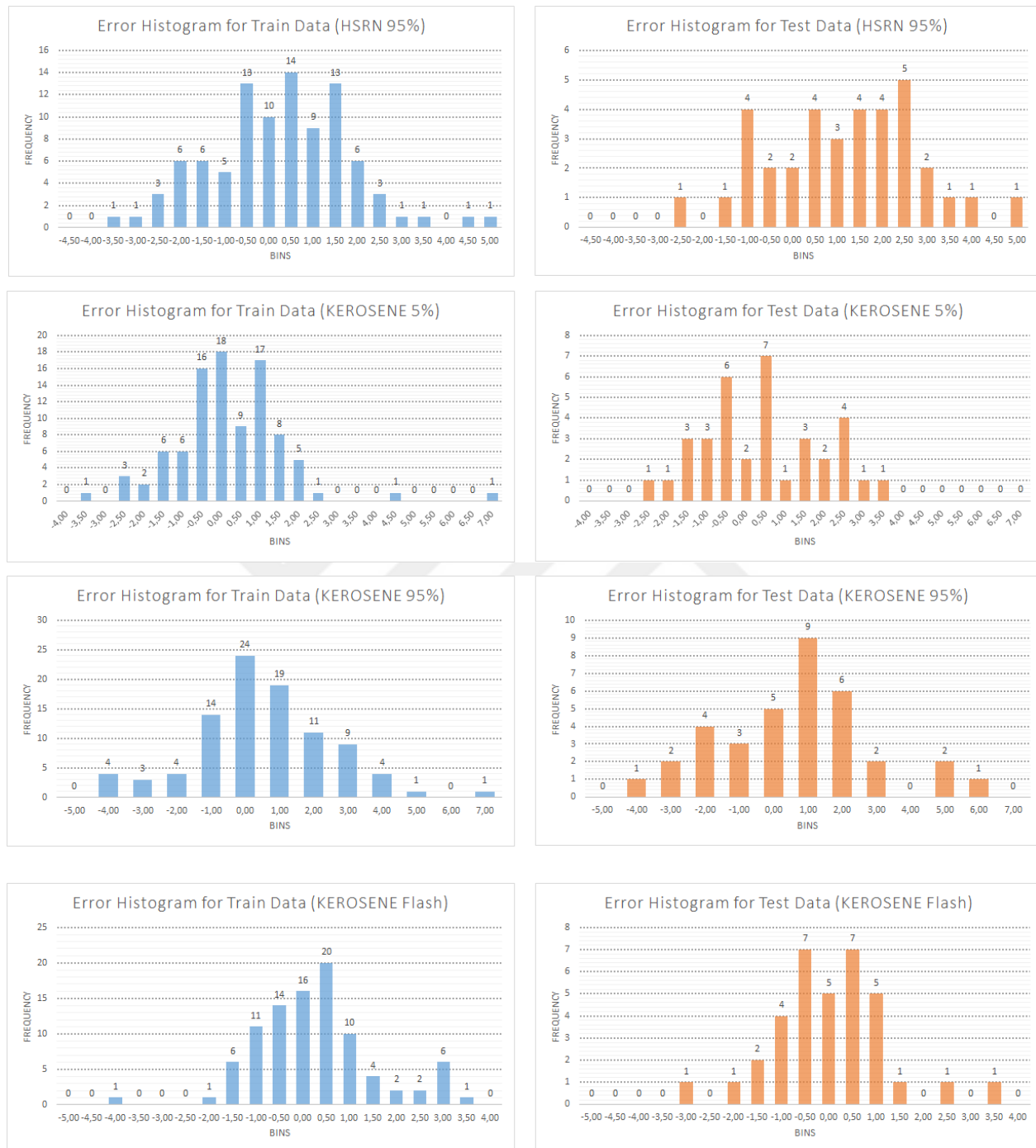


Figure 7.3 (Cont'd) – Error histograms of final simulation error models

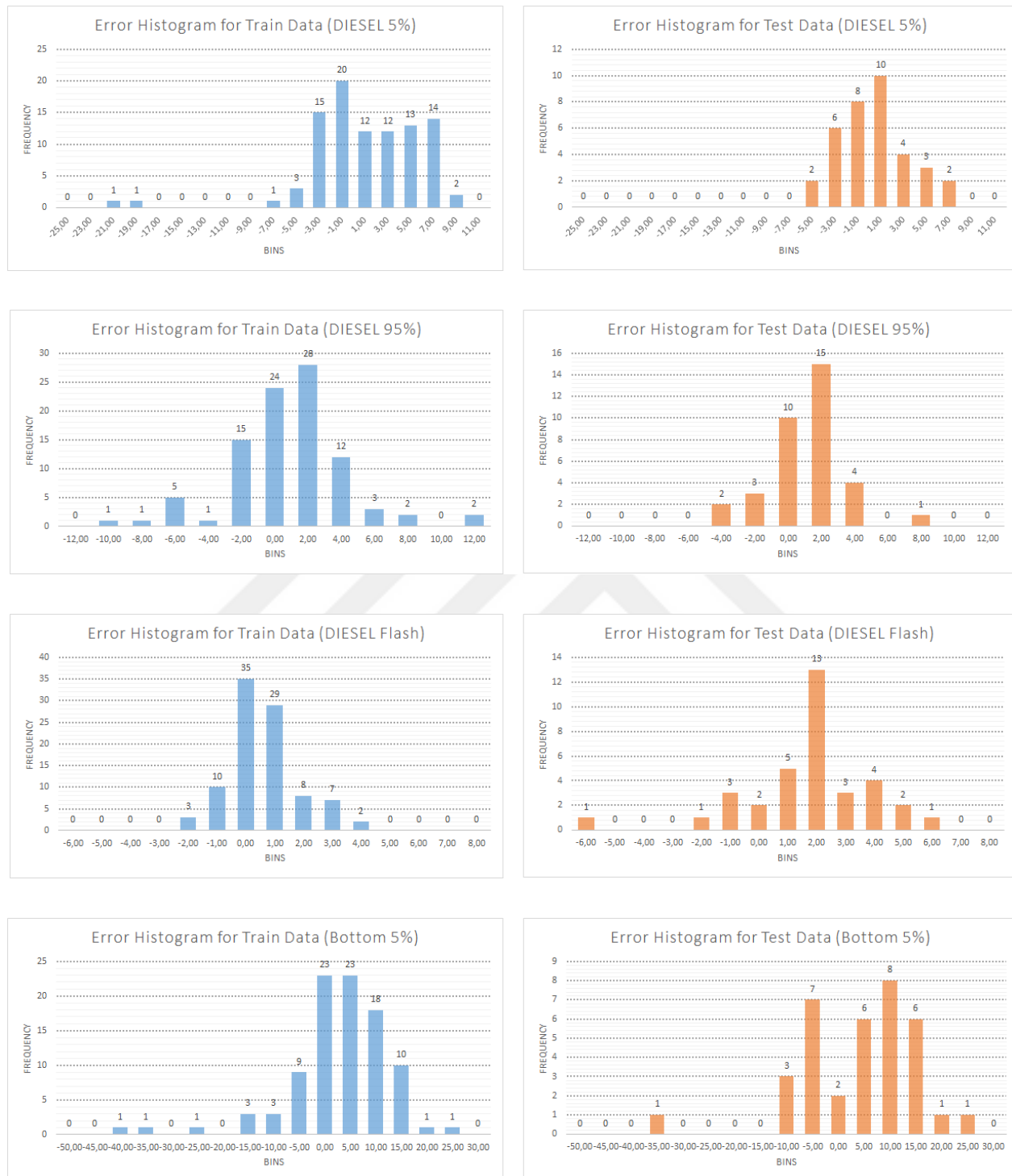


Figure 7.3 (Cont'd) – Error histograms of final simulation error models

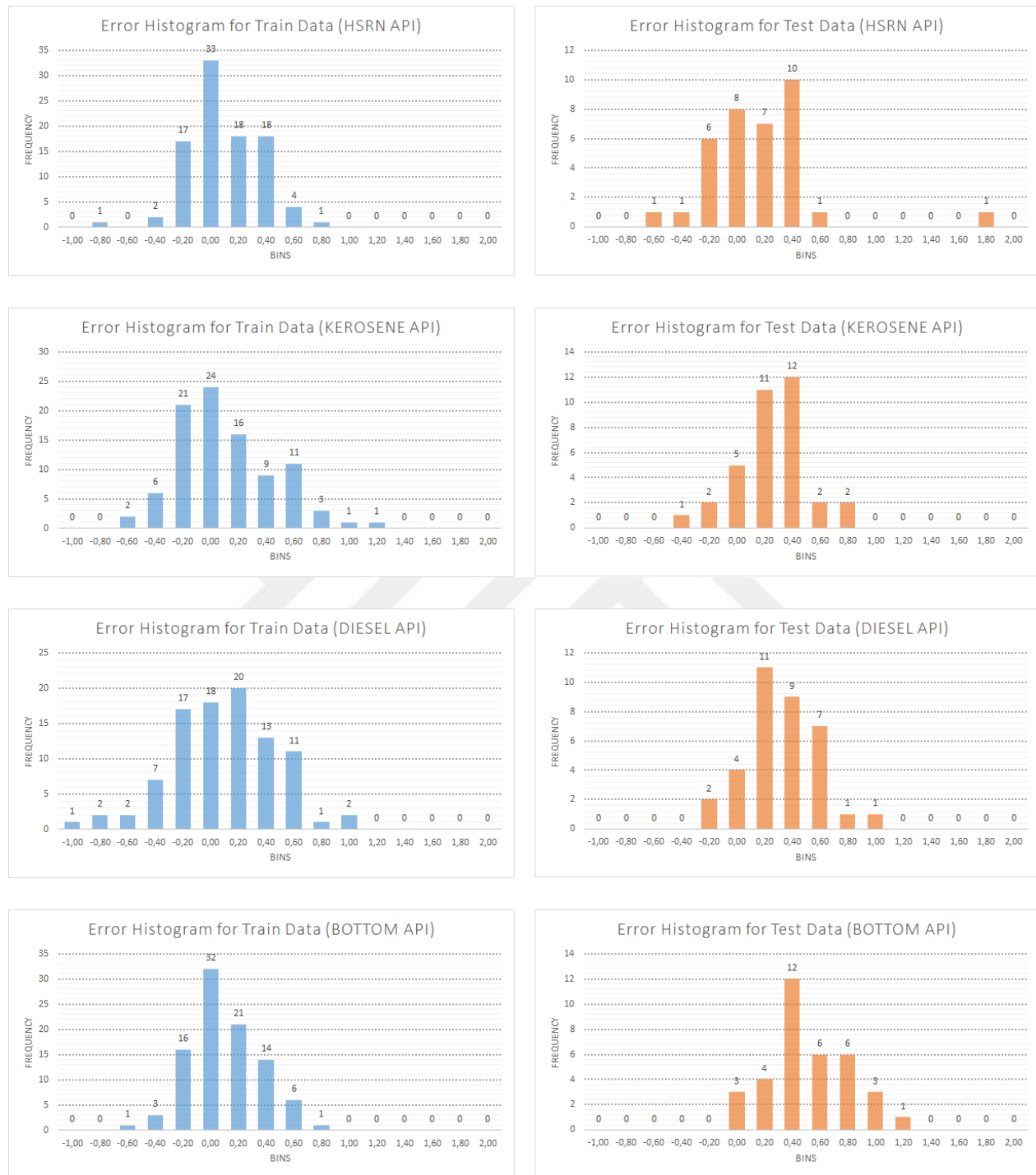


Figure 7.3 (Cont'd) – Error histograms of final simulation error models

In Figure 7.4, simulation error predictions are shown in the graphs located on the left side, whereas the product quality estimations calculated with the following equation are shown on the right side. In Figure 7.4, the blue curves of the graphs located left side denotes the actual values simulation errors E_i , and the orange curves in the same graphs shows the estimated simulation errors $\hat{E}_i$. Similarly, in the graphs located on the other side the blue curves shows the laboratory results, yellow ones show the estimated product qualities by using Equation 7.12 and the green curves shows the simulation calculation for the product quality of interest.

$$\hat{Y}_i = W_i + \hat{E}_i \quad (7.12)$$

Where, $\hat{Y}_i$ is the estimated product quality for product i, W_i is the simulation output for the same product and $\hat{E}_i$ is the estimated simulation error.

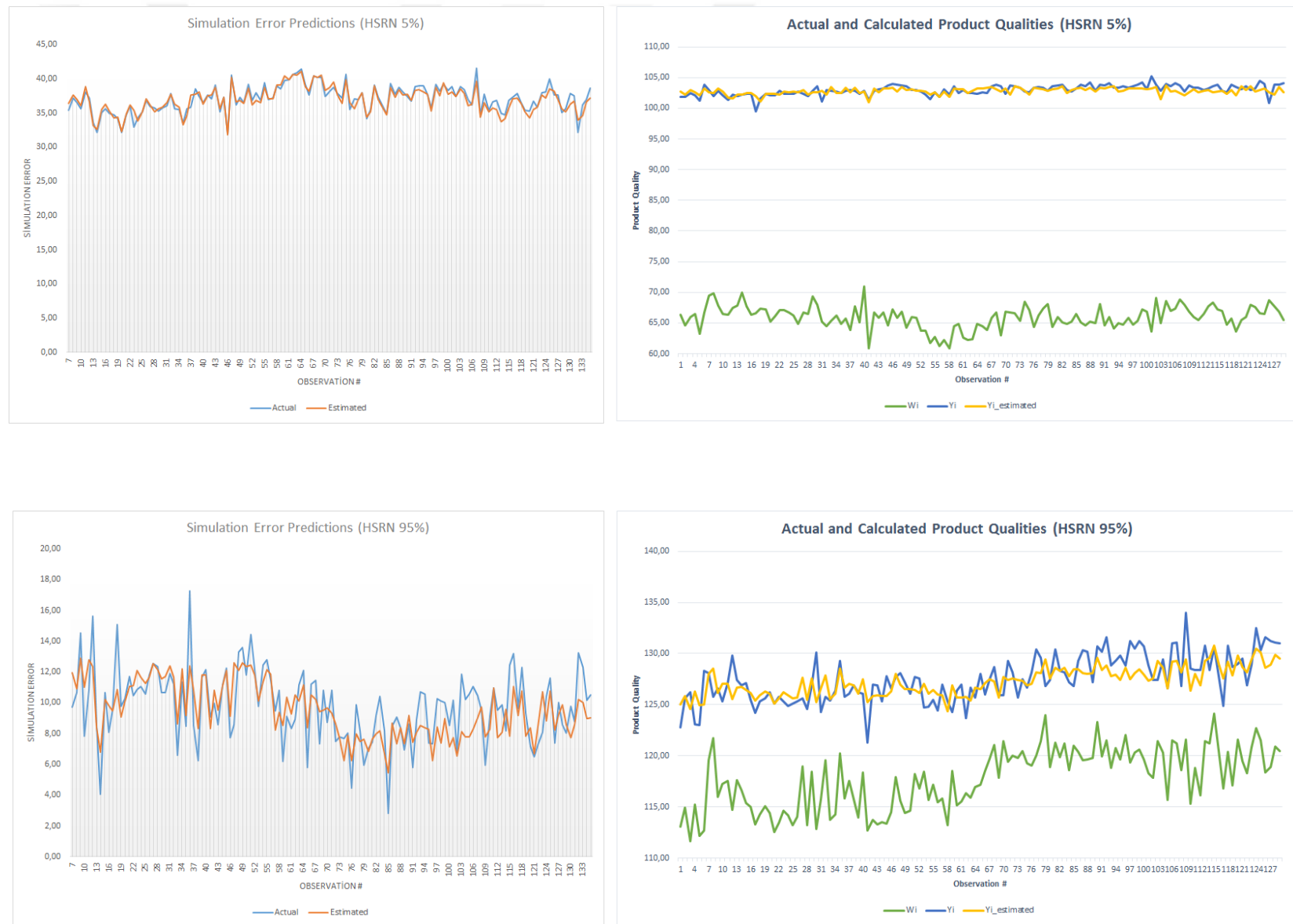


Figure 7.4 – Simulation error and product quality predictions

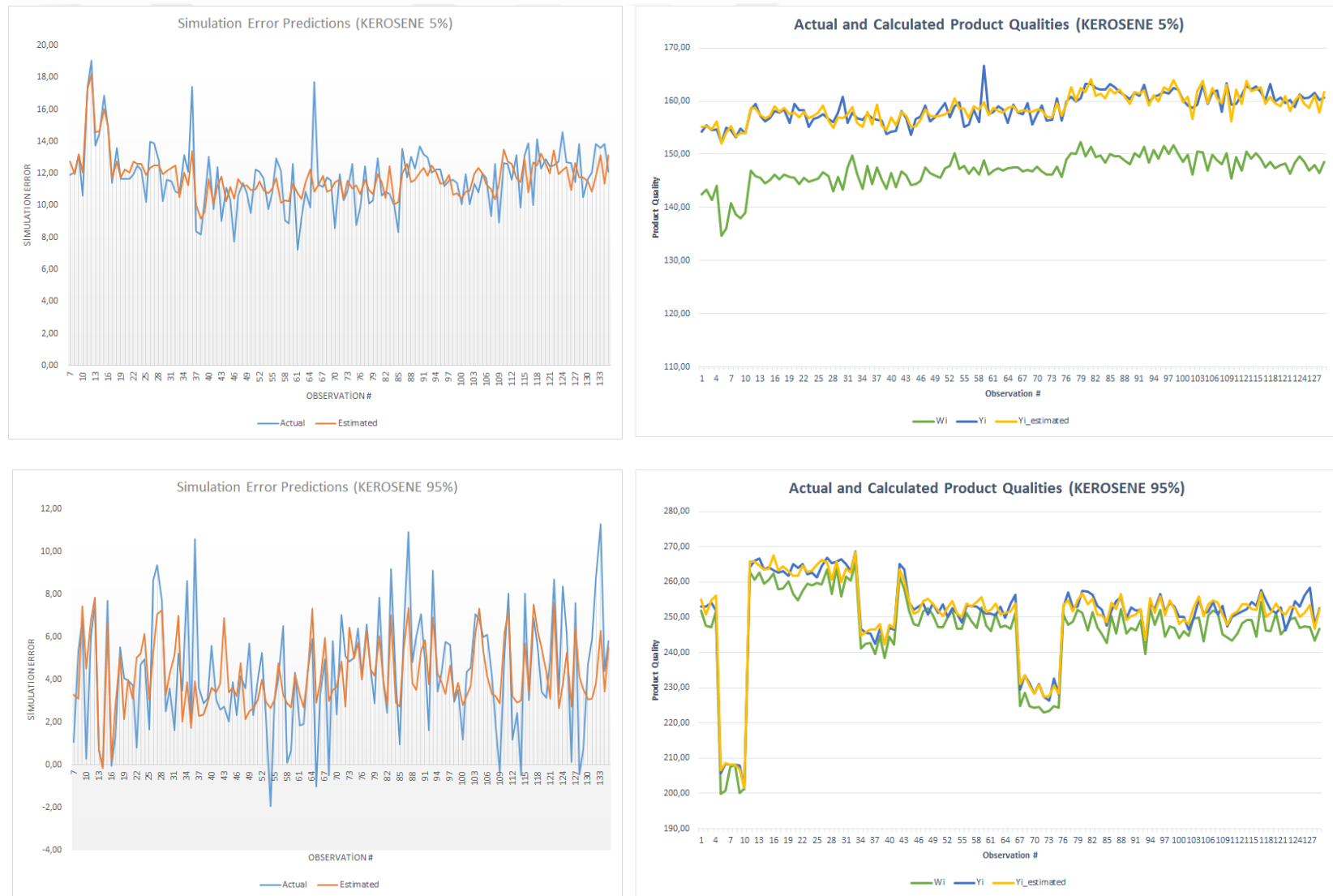


Figure 7.4 (Cont'd) – Simulation error and product quality predictions

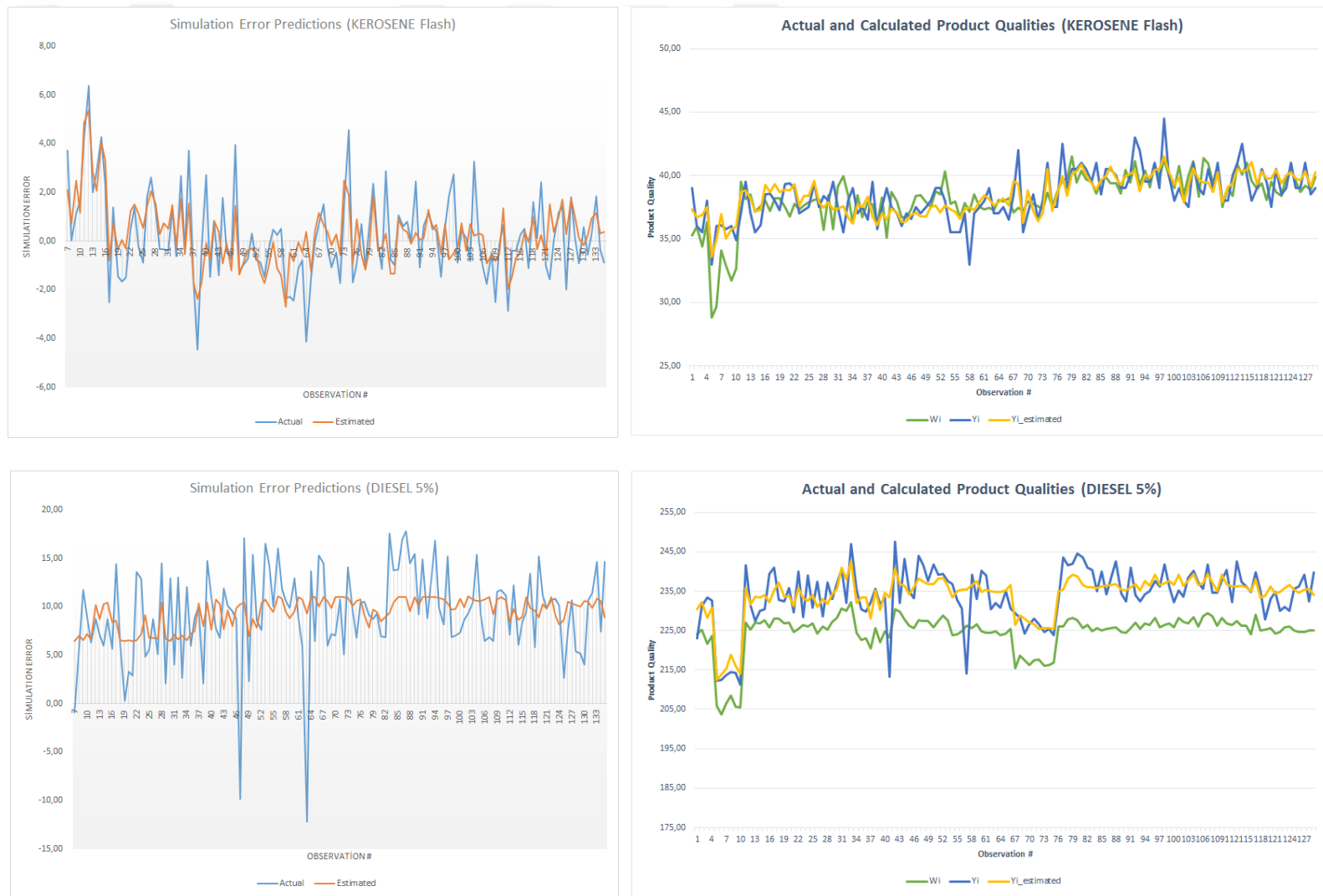


Figure 7.4 (Cont'd) – Simulation error and product quality predictions

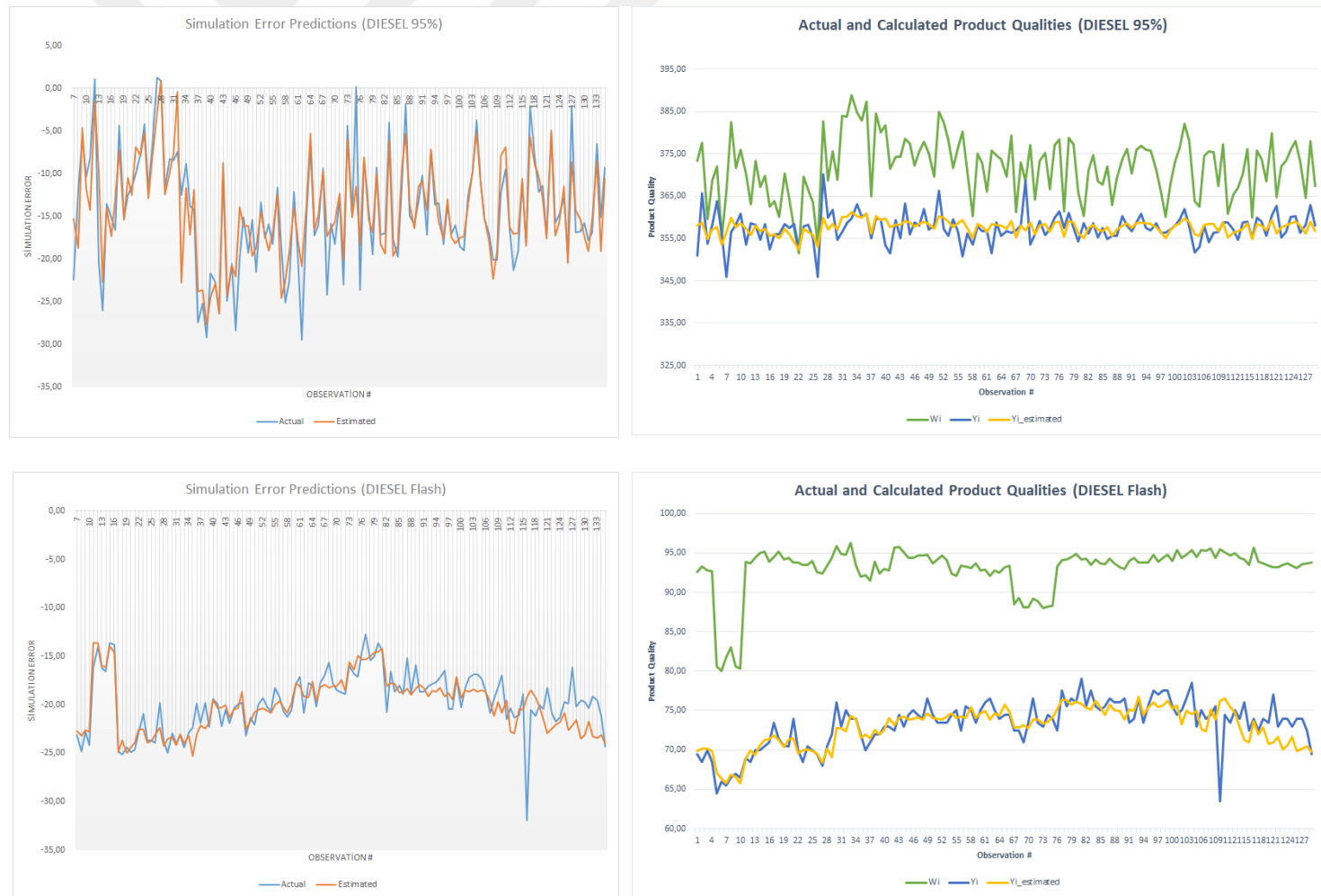


Figure 7.4 (Cont'd) – Simulation error and product quality predictions

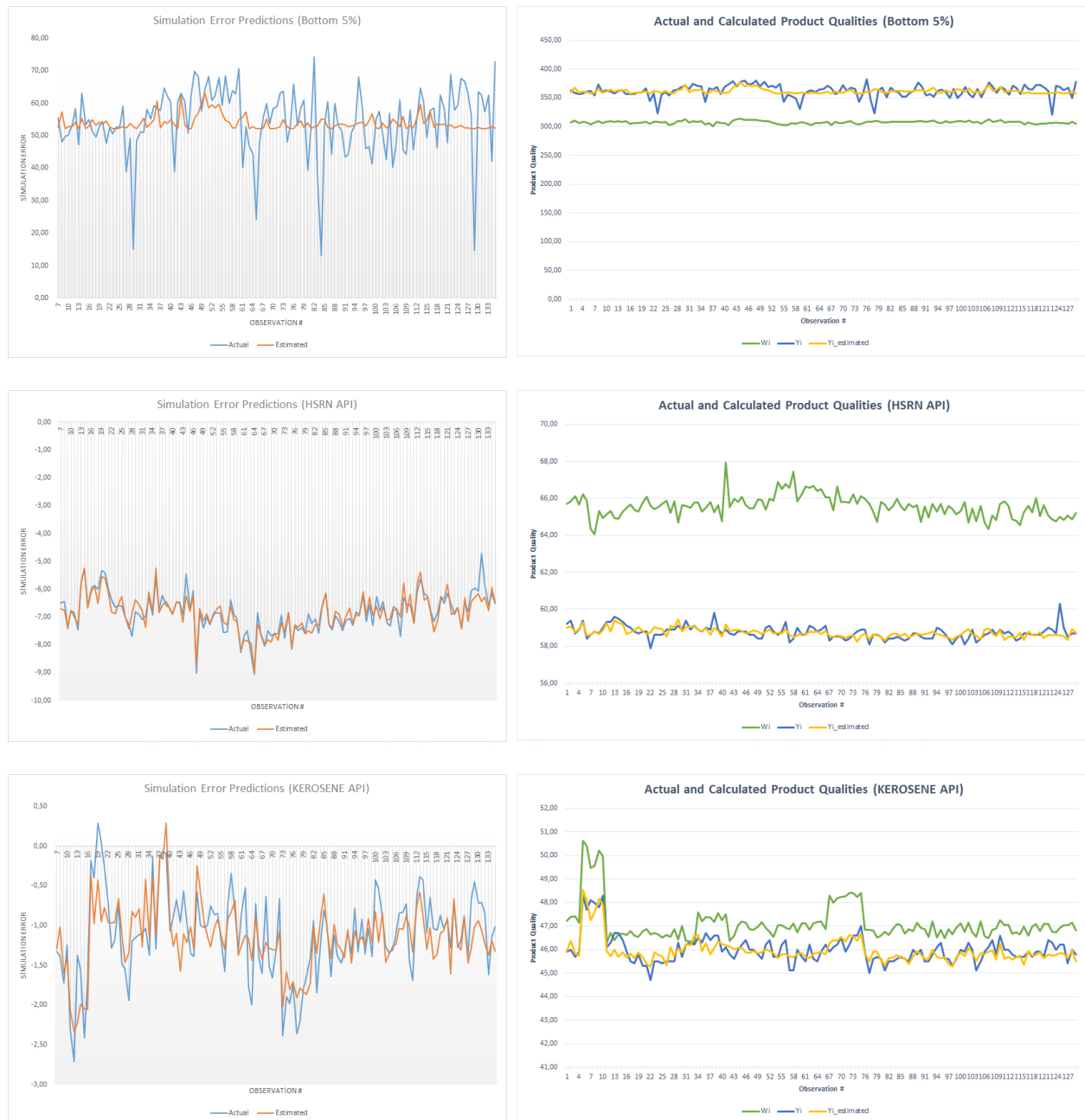


Figure 7.4 (Cont'd) – Simulation error and product quality predictions

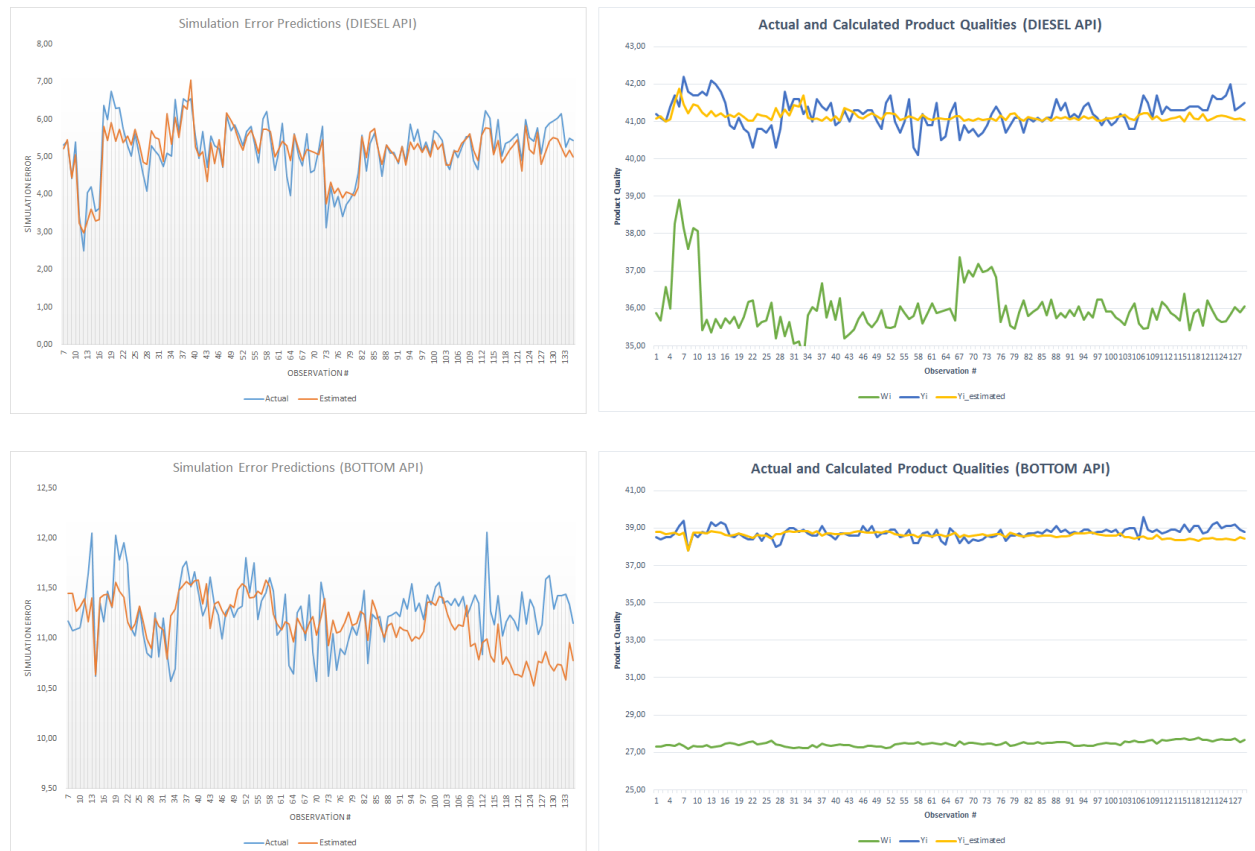


Figure 7.4 (Cont'd) – Simulation error and product quality predictions

7.3.2. Model Accuracy Assessment

In this section the errors are compared with the repeatability and reproducibility in measurements to discuss whether it is possible to enhance the model performance or not by using the following factor:

$$B = \frac{(RMSE)^2}{(R\&R)^2} \quad (7.13)$$

where, RMSE refers to root mean squared error and R&R denotes repeatability and reproducibility in measurements. It is important to note that if the ratio, B , is close to 1, then the residuals can be explained by the measurement error; hence there is no point in struggling to enhance the model. If B factor is less 4, the model error are acceptable since measurement errors are the main cause for these errors.

Table 7.9 shows the B -factors for all simulation error models for training, testing and CV errors as well as the R&R values. In Figure 7.5, Table 7.9 is summarized in a graph. As can be seen from the graph,

except diesel 5% boiling product quality, all B-factors are less than 4, meaning that the errors in the models can be explained by the measurement errors.

Table 7.12 – Repeatability and reproducibility in measurements and B-factors

		R&R	B _{TRAIN}	B _{TEST}	B _{CV}
$\hat{R}_1$	HSRN %5	0.5	1,3	3,5	1,7
$\hat{R}_2$	HSRN %95	1.0	2,4	3,5	2,3
$\hat{R}_3$	KEROSENE %5	1.5	0,9	1,0	1,2
$\hat{R}_4$	KEROSENE %95	2.0	1,0	1,3	1,2
$\hat{R}_5$	KEROSENE Flash	1.0	1,6	1,4	1,5
$\hat{R}_6$	DIESEL %5	2.5	3,8	1,6	2,2
$\hat{R}_7$	DIESEL %95	3.0	1,3	0,5	1,3
$\hat{R}_8$	DIESEL Flash	1.5	0,7	4,5	0,8
$\hat{R}_9$	Bottom %5	6.0	2,8	3,3	1,7
$\hat{R}_{10}$	HSRN API	0.3	0,8	1,8	0,8
$\hat{R}_{11}$	KEROSENE API	0.3	1,4	0,9	0,8
$\hat{R}_{12}$	DIESEL API	0.3	1,6	1,3	1,1
$\hat{R}_{13}$	BOTTOM API	0.3	0,8	3,0	1,2

One can see that R&R values increases as the product gets heavier and the volume distilled percent get higher. This is because of the fact that when the product gets heavier, the number of molecules in the mixture gets more and more and in such a case the uncertainty in measurements increases.

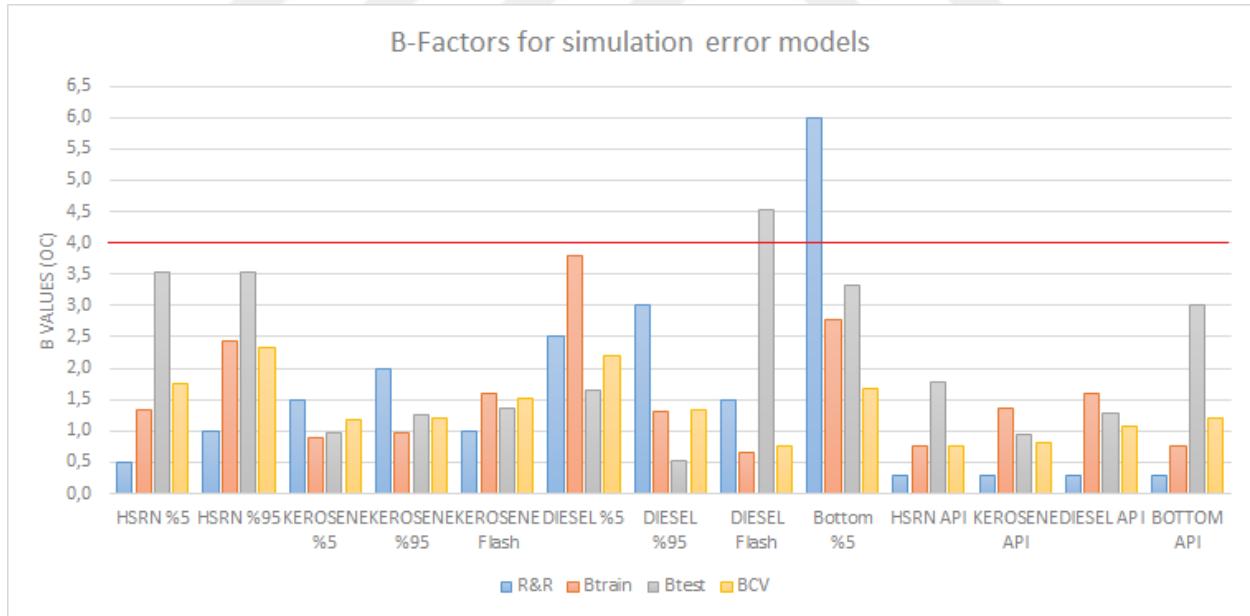


Figure 7.5 – B-factors for simulation error models

Chapter 8

Discussion of Model Results

8.1. Discussion of Process Simulation Outputs

The simulation outputs for the product qualities are given in Figure 5.4 and they are compared with the actual laboratory results. As can be seen the difference between the simulation error and laboratory result is not constant; meaning that there is no constant bias between those. These differences could exist due to several reasons listed below:

- TBP distillation is the single-most important information for the feed property estimation. In case of inaccurate boiling point curve for the feed, the simulator may fail to estimate the feed physical properties. Feed characterization model outputs, which are the different boiling points of the feeds, are trained with the TBP curve obtained by back-blending method. In this method, the feed TBP is obtained from the ASTM D86 distillation points of the column products except the bottom product. For bottom product, ASTM D1160 distillation points are used for back-blending purposes. Since the boiling points of some fractions, such as %70, %80 boiling points of the bottom product can be so high and at these high temperatures a thermal cracking could happen; hence, there could be no data for such higher fractions. This case could contribute the difference between the simulation outputs and the actual outputs.
- The tray efficiencies are quite important for the simulator. The vapor leaving each tray is in equilibrium with the liquid leaving the same tray. This is the case if the trays are operating at 100% efficiency. However, in practice, the trays never operate perfect due to insufficient time of contact and insufficient degree of mixing. Therefore, tray efficiency values for each tray are needed to be considered very carefully. In the simulation process building stage, we use the tray efficiency values from the literature. However, tray efficiency values are specific to the distillation columns itself. Fine tuning of the tray efficiencies could minimize the difference between the actual and simulator outputs of product qualities.

8.2. Discussion of Feed Characterization Model Results

Data Preprocessing

There are many factors affecting the success of machine learning on a given task. The representation and quality of the data is first and foremost. The knowledge discovery during the training process of a model is more difficult with irrelevant or noisy unreliable data. The modelers should consider the data preparation and filtering steps in machine learning process and these steps include data cleaning, normalization, transformation, feature extraction and selection, etc and the final product of these steps is the final training set. Thus, all the inductive learning algorithms rely heavily on the product of this stage, which is the final training set.

Considering the mentioned things above, the input data is firstly analyzed so as to detect if there is outliers or unreliable observation/measurements for the variables. The followings are some examples of unreliable observations in the training data set:

- The laboratory results for the product qualities are measured with laboratory instruments and logged to Laboratory Information Management Systems (LIMS), which is a software-based laboratory and information management system with features supporting a modern laboratory's operations, such as data tracking support. The results obtained from the laboratory instruments can be entered to LIMS by the laboratory technician or automatically. In other words, LIMS may import instrument results to extract data for product quality or not. In Turkish petroleum Refineries Corporation, the results are entered by the laboratory technician and in such a case, there could be some errors. For instance, product quality for Diesel 5% is recorded as 2390 °C on 18th September 2016. However, comparing the other observations, it should be 239 °C. Therefore, this value is changed from 2390 °C to 239°C.
- Process variables needs to be measured by a sensor which converts the measurement and then transmitted to a controller for a process to be controlled and manipulated. Signal transmission is done by the transmitters. However, each transmitter have a range value and the measured value is not within this range, the measurement is incorrect. Initially, there were 143 observations starting from 16th February 2015 to 18th March 2016. However, in March of 2015 (from 7th March 2015 to 28th March of 2015), the transmitters for reactor inlet pressures are out-of-range so that the 8 observations corresponding to these dates are eliminated from the training data.

Even though variable-by-variable data cleaning is straightforward filter approach, this can help on detecting on a number of possible data quality problems. The previous unreliable observations examples are detected by using *single-step outlier detection method*, which detects outlier by using the following

rule: $outlier(\alpha_n, \hat{\mu}_n, \hat{\sigma}_n^2) = \{x: |x - \hat{\mu}_n| > g(n, \alpha_n)\}$, where n is the size of the sample, $\hat{\mu}_n$ and $\hat{\sigma}_n^2$ are the estimated mean and standard deviation of the target distribution based on the sample; α_n denotes the confidence coefficient following the correction for multiple comparisons tests, and $g(n, \alpha_n)$ is the critical number of standard deviations of outlier regions. If these 8 observations, where pressure transmitters are out-of-range and incorrectly typed laboratory results are used in the models, the obtained result will be inevitably unreliable.

Parameter Optimization Approaches

In this thesis, one of our aims is to find the best model parameters for a given regression problem. To achieve this target, optimization methods are implemented to seek the best model parameters. Optimization methods for multi-variable, non-linear programming problems fall naturally into three classes, which are: direct search methods, gradient methods and intelligent optimization methods. Direct search methods require only the evaluation of the objective function; in other words, the use of partial derivatives of the objective function is not required. These methods are iterative and they start with an initial guess of the solution and then progress by generating a sequence of new estimates. Random search method, grid search method, simplex search method are some examples for these methods. In gradient methods, in addition to the objective function, the derivatives of the objective function is required. Some examples for these techniques are steepest descent method, newton's method, and Fletcher-Reeves method. Intelligent optimization(or heuristic) methods are also used to solve non-linear programming problems and genetic algorithm, particle swarm algorithm, tabu search are samples of these methods. Genetic Algorithm is one of the popular heuristic optimization method, which mimics the process of natural selection where a population of candidate solutions evolve through generations to a better solution.

In this study, grid search algorithm is used to find model parameters of support vector regression, random forest, and artificial neural networks models. Even though previous studies show that another optimization algorithms like Genetic Algorithm (Lahiri, et al., 2009) and Particle Swarm Optimizer (Yao, et al., 2012) are used to search for the best parameters of SVR models in modeling chemical processes, the grid search algorithm performs well for our aim since the required maximum is known to be within a finite area defined by upper and lower bounds of each of the independent model parameters. This method systematically search all of the possible states located on the grid nodes. The advantage of grid-search algorithm is that it can search independent parameters in parallel, and consume less time when limited parameters are available. This method is very simple to program; on the other hand, it can be very inefficient because of it may require excessive computational time, especially when more parameter needed and difficulty to estimate the range of parameters and penalty factor. Fortunately, in our problems (finding the

best model parameters), we have limited parameters and the lower and upper bounds for the parameters are well-known so that this approach does not require long computational times.

Intelligent algorithms like genetic algorithm or particle swarm algorithms are most beneficial when the search space is very complex, large, and poorly understood. They also perform well, especially when domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space. Unfortunately, these algorithms do not always come with global optimum all time especially when overall solution has various population and they are very complex to be able to apply problem or to be understood. Even though they can handle arbitrary kinds of constraints and objectives, they should be used if the traditional search methods fail.

Considering all of these, grid search is considered for model parameter optimization despite of its high computational cost.

Discussion on Final Models for Feed Characterization

In the stage of building models for feed characterization, input variables are chosen by using the process knowledge. The following table (Table 8.1) is the correlation matrix for feed characterization model inputs, which are previously listed in Table 6.1.

Table 8.1 – Correlation matrix for feed characterization model inputs

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}
F_1											
F_2	0.34										
F_3	0.01	0.24									
F_4	0.29	0.52	0.58								
F_5	0.28	0.41	-0.09	0.29							
F_6	-0.10	-0.22	0.67	0.34	-0.12						
F_7	0.05	-0.27	-0.36	0.05	0.15	0.35					
F_8	0.05	-0.23	-0.20	0.07	0.11	0.47	0.96				
F_9	0.04	-0.24	-0.27	0.04	0.08	0.42	0.97	0.97			
F_{10}	-0.03	-0.01	-0.36	-0.42	0.52	-0.16	0.08	0.02	0.07		
F_{11}	-0.04	-0.02	-0.39	-0.43	0.51	-0.18	0.09	0.03	0.08	1.00	

A simple way to detect collinearity is to look at the correlation matrix of the predictors. In Table 8.1, there is a strong correlation between F_7 (WABT of 2nd bed of C-5 reactor) and F_8 (WABT of 1st bed of C-6 reactor). Similarly, F_9 is highly correlated F_7 (WABT of 2nd bed of C-5 reactor) and F_8 (WABT of 1st bed of C-6 reactor). Moreover, the correlation coefficient is very high for F_{10} (Inlet pressure of C-5 reactor)- F_{11} (Inlet pressure of C-6 reactor) relation. These results show us that *collinearity*, which is a phenomenon in which two or more predictor variables in a regression model are highly correlated, appears in these model. If let's say, X_1 and X_2 are highly correlated with each other, then we say that they are *collinear*. The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the

individual effects of collinear variables on the response. In other words, it can be very difficult to determine how each predictors separately is associated with the response. For linear regression problems, collinearity reduces the accuracy of the estimates of the regression coefficients, which leads to grow in the standard error for predictor coefficient and reduce in the probability of correctly detecting a non-zero coefficient.

Unfortunately, not all collinearity problems can be detected by inspection of the correlation matrix: it is possible for collinearity exist between three or more variables even if no pair of variables has a particularly high correlation. This situation is called as multicollinearity and a better way of assess it is to compute the *variance inflation factor* (VIF), which is the ratio of the variance of predictor coefficient estimation obtained from linear regression analysis when full model divided by the variance of this coefficient estimation if fit on its own. The smallest value for VIF is 1 and as a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity. The VIF for each variable can be computed using the formula

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R_{X_j|X_{-j}}^2}$$

Where $R_{X_j|X_{-j}}^2$ is the R^2 from a regression of X_j onto all of the other predictors. If all of the 135 observations are used, the predictors have VIF values of 1.3, 3.2, 89.9, 12.3, 3.8, 48.8, 133.6, 50.3, 34.9, 3726.3, 3863.4, which indicates there is strong multicollinearity in the data.

Multicollinearity does not reduce the predictive power of reliability of the model as a whole. In case of linear regression, it only affects calculations regarding individual predictors. In case of neural networks, the models are extremely flexible, almost nonparametric tool less prone to the curse of dimensionality. Neural network-based analysis methods need not assume that independent variables are not correlated with each other. In other words, neural networks do not assume that residuals of independent observations are independent of each other. This solves the problem of multicollinearity that arises in multiple linear regression. In Fabian Jaimes et al (Jaimes, et al., 2005), the study was conducted to predict the mortality within the first 28 days from admission. The study population comprised patients with suspected bacterial infection as their main diagnosis for admission to the emergency room at two University-based hospitals. The prediction models are built by using logistic regression and artificial neural networks with an adaptive genetic algorithm. Initially, all input variables are used in the neural network models and then for the comparison purposes, a new neural network model is built by using the limited number of predictor variables to the same set of predictor variables used in the logistic regression. Nevertheless, the predictive ability was almost the same with the both approaches. Only the network structure and the network weights are different. In case of support vector regression, the advantage of using support vector is that it builds a

highly accurate model through an engineering problem-oriented kernel. Also, it makes use of the regularization term to avoid over-fitting. It also does not suffer from multicollinearity (Yu-Wei, et al., 2015).

In short, Multilayer Perceptron (MLP) and Support vector machine (SVM) are not severely affected by multicollinearity, but the model interpretability is very low. Both MLP and SVM are not affected by multicollinearity since they first perform a nonlinear transformation from the space of the predictors to a new space, which is the projection step, and then a linear transformation from this new space, which is the modelling step. Considering the Table 6.6, SVM with RBF kernel model is chosen as a final model for modelling the feed characterization, since it has the lowest 10-fold CV error comparing to the other models. Table 6.10 shows the final models and their corresponding model parameters and errors for training, testing and 10-fold cross validation stages. Cost parameter ,C, is treated as a tuning parameter and chosen via cross-validation. When C is small, we seek narrow margins that are rarely violated leading to low bias but high variance. On the other hand, when C is larger, the margin is wider and more violations are allowed so that it is more possible to have more bias but lower variance. As can be seen from the table, when C is large, then the margin is wide and many observation violate the margin, and so there are many support vectors. One should consider that it is possible to misevaluate the relationship between C and the number of support vectors from Table 6.10, since the label variables are not the same. For instance, for 70% and %80 boiling point models, C tuning parameter is the same as 7; however, the former has 115 support vectors, whereas the latter has 35 support vectors. From Table 8.2, we can conclude that the parameter C doesn't depend on the boiling point of interest. In other words, there is a random pattern between the boiling point of the feed and C tuning parameter. As we go from 5% boiling point to that of 95%, the best tuning parameter C randomly changes. Similar random pattern is also observed for epsilon parameter, which controls the width of the ϵ -insensitive zone. Similar to parameter C, epsilon can also affect the number of support vectors used to construct the regression function. Thus, both C and epsilon values affect the model complexity but in a different way.

8.3.Discussion of Simulation Error Model Results

Final Models to predict the simulation errors for the product qualities are given in previous section in Table 7.11. As can be seen from this table, even though the best modelling technique is varying according to the product quality of interest, 10 of 13 models are artificial neural networks. The other models are support vector regression models with RBF kernel. Both of these techniques can capture the non-linear relationship between the input and the output. Even though, random forest models are not the final models, the variable importance ranking information was quite beneficial to select the variables to be used in ANN, linear regression and SVR modelling in approach 6. Different from the feed characterization models, support vector regression is not the best modelling technique for simulation error estimation. Random forest models

and linear models are not chosen as final model due to having lower 10-fold CV errors compared to those of SVR and ANN models.

Simulation error for HSRN 5% boiling point is estimated with X_{34} (Simulation output for HSRN 5%), X_{48} (Simulation error with lag for HSRN 5%, $E_1(t-1)$), X_{62} (Derived variable 1 from the simulation errors with lags), and X_{63} (Derived variable 2 from the simulation errors with lags). For training data, the average error between the simulation output and the actual product quality is 37.06 °C and it has a standard deviation of 2.06 °C. It is possible to say that this gap does not change a lot and it has a small variance, and so it is possible to capture the simulation output error variations with the previous simulation errors. The inputs suggested by using the random forest variable rankings is supporting this conclusion. Simulation error for HSRN 95% boiling point is estimated with X_{35} (Simulation output for HSRN 95%) and X_{24} (The amount of hydrogen supplied to the 1st of hydrocracker reactor). Figure 6.5 shows the relationship between the hydrogen consumed in the reactor and the yield of light components like pentane, i-butane. As can be seen from this figure, after certain point the yield for the components boiling at between 180 Fahrenheit (=82 °C) and 400 Fahrenheit (=204 °C) starts to decrease. The components with these boiling range directly affect the 95% boiling point of HSRN. Hence it is possible to say that after a certain amount of the hydrogen, the simulation error may increase or decrease due to the uncertainty increased in the feed TBP. Simulation error for Kerosene flash point is predicted with X_{38} (Simulation output for Kerosene flash point), X_{21} (Bottom product flow to reboiler of kerosene stripper), X_{36} (Simulation output for kerosene %5), and X_{46} (Diesel API gravity). In distillation columns with side-strippers, the flash point is manipulated with the amount of steam sent to the stripper in case of the existence of steam or with the heating duty of the stripper. In kerosene stripper, the amount of this duty is manipulated with the amount of heating medium, which is directly related with the amount of bottom product flow, X_{21} . Hence, it is possible to say that there is a relationship between kerosene flash point and the amount of bottom product sent to the reboiler of kerosene stripper. Hence, it will also affect simulation output for HSRN 5% boiling point. The gap-overlap is a measure of separation between two adjacent products from the distillation columns like fractionator column. The heavy cut is the lower product and has the higher molecular weight. The lighter product is the upper product and has the lower molecular weight. In that sense, there is a gap or overlap between the adjacent product like kerosene and diesel so that the API gravity (measure of density) of the diesel product, X_{46} , may affect the lightest components of kerosene and this may also affect the simulator output for kerosene flash point. Since X_{21} and X_{46} have a considerable impact on the simulation output for kerosene flash point, these are ranked as an important variables by random forest variable scoring method.

In distillation process, many operational variables affects the others. In such a system, input variables for the product quality models are correlated. From these examples given above, it can be seen

that it is quite difficult to explain the relationship between the inputs and the outputs since there are many variable dependent on the other variables in. Considering that the linear regression coefficient can be misleading with such a interacting system. For ANN, SVR, and Random forest models the most important target is the accuracies of the models, not the interpretability of the models. All simulation error models are evaluated in Section 7.7.2 by considering the uncertainties in laboratory measurements and it is shown that all B-factors are less than 4, meaning that the errors in the models can be explained by the measurement errors. Moreover, severe multicollinearity is a problem for the linear models because it can increase the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model. The result is that the coefficient estimates are unstable and difficult to interpret. Considering these facts, it is better to use the modelling techniques that can handle multicollinearity problem, especially when the accuracy of the model is the main target, as in the case of our work.

Finally, we should note that in simulation error model building stage, the training data includes the simulator outputs for product qualities, which uses the estimated feed properties as label variables. Therefore, it is possible to say that the errors in the feed property estimations are also propagated to the errors in the simulation error models. This is because of the fact that the simulation outputs with the estimated feeds, W_i 's, are the input variables for the simulation error models.

8.4.Discussion on Model Selection

In this thesis in order to estimate parameters for predictive models in order to improve predictions, 10-fold cross validation error is used. The model parameter set which give the lowest 10-Fold CV error is chosen as a final model parameters and the decision among the models obtained with different modelling algorithms, like ANN and SVR, is again done by seeking the lowest 10-Fold CV error. In the ideal situation it is assumed that we would have enough data to train and validation the models and have separate data for assessing the quality for our model. Moreover, for both training and testing samples, we assume that they are sufficiently large and diverse in order to be representative. We are aware of that cross-validation pitfalls could happen when selecting and assessing regression models for the selection of model based on performance of a single cross-validation and reporting a cross-validation errors as an estimate of the test error. In order to eliminate the danger of k-fold CV pitfall, it is possible to use another techniques ;such as, repeated cross-validation, double cross-validation, repeated double cross-validation and nested cross-validation. On the other hand,these techniques are not used since the model results for feed characterization and simulator output calibrator results are quite satisfactory, especially when the laboratory uncertainties are considered, and the number of observations is sufficient. If the number of observation is much more scarce, these methods could be used to estimate the generalization performance of a model.

Chapter 9

Conclusion

In this study, Support Vector Regression with radial basis function kernel, Artificial Neural Network with back-transient algorithm, Random Forest regression and linear regression were employed for estimating the simulation errors for product qualities of industrial hydrocracker fractionator column in İzmit Refinery of Turkish Petroleum Refineries Corporation. 6 different approaches are used to estimate the simulation errors. In these approaches, even though the label variable is the same, input variables and the model types differ. The model parameters were optimized by embedding k-fold cross validation in conjunction with grid search optimization. The model parameters were optimized in a time period ranging from 1 minutes to 1 hour, but once the model parameters are decided, the models are trained with these parameters in less than 1 minute.

In addition to modelling of simulation errors, feed distillation properties (5%, 10%, 20%, ..., 90%, 95% boiling points) are estimated with operational data to run a process simulator at any desired time. Input variables for feed characterization models are selected by considering the process knowledge. Linear regression, ANN with back-propagation, and support vector regression with three kernels (Linear, RBF, and polynomial) were employed as learning techniques.

The 10-fold cross validation results show that Support Vector Regression with RBF kernel is the best modelling technique for feed distillation property estimation even though 10-fold CV errors are close to each other for all modelling techniques. 10-fold CV error in RMSE for all product qualities vary between 1.10 °C and 4.7 °C and the average 10-fold CV error in RMSE for all distillation points is 3.11 °C. The average of test errors in RMSE for all boiling points is 2.69 °C and the maximum test result in RMSE is 5.33 °C. The standart deviation for testing errors for all qualities is 1.15 °C.

Simulation error model results show that test errors in RMSE for distillation points of products vary between 0.94 °C and 10.95 °C with an average of 3.03 °C and standard deviation of 3.08 °C. Similarly, the test errors in RMSE for densities of products in API vary between 0.29 and 0.52 with an average of 0.39 and standard deviation of 0.10. Considering the measurement uncertainties, the model performances are quite acceptable to be used for estimating simulation errors.

The results shows that estimation of the feed stream TBP curve of the hydrocracker unit from the operational data of the reactor system can be used to characterize the feed properties so that process engineers can run the process simulator for any desired time once they have the feed characterization models. The study also provide that the process simulator outputs can be calibrated with a high accuracy by implementing data-driven modelling approaches.

This work can be extended by making the models adaptive, i.e. with an addition of a new data, the model parameters can be updated for a certain number of new samples or for a certain threshold of the cumulative sum of errors. This could enhance the generalization performance of the models, since there is an aging of instruments, catalyst, and equipments in the field and by the passage of time it could be needed to update the model parameters to adopt the aging.

APPENDIX I – SIMULATION OUTPUTS

Table 10.1 – Simulation Outputs

	Simulation Outputs														Column Temperatures						
	Product Qualities																				
	HSRN %5	HSRN %95	KEROSENE %5	KEROSENE %95	KEROSENE FLASH	DIESEL %5	DIESEL %95	DIESEL FLASH	BOTTOM %5	BOTTOM %95	HSRN API	KEROSENE API	DIESEL API	BOTTOM API	Top T	5th Tray T	14th Tray T	19th Tray T	24th Tray T	40th Tray T	Reboiler Duty
1	66.67	117.6	147.42	254.49	39.85	225.41	360.6	93.91	310.45	443.25	65.13	46.55	36.17	27.35	119.5	145.5	199.37	258.42	289.69	351.76	796818.924
2	66.8	114.1	144.81	247.29	37.22	223.8	356.76	92.68	307.84	442.07	65.7	47.24	36.28	27.52	118.1	143.7	195.22	256.93	289.85	351.32	888439.762
3	66.77	112.7	139.63	230.76	32.95	220.1	373.08	90.78	308.19	459.51	65.75	48.35	36.51	27.39	117.7	142.5	191.54	247.48	286.01	352.02	648128.7
4	66.25	116.2	142.57	233.87	35.45	217.57	364.15	89.69	306.72	442.85	65.43	48	36.91	27.45	121.2	147.4	195.04	249.25	284.86	351.73	767869.42
5	66.94	118.7	145.88	254.73	38.88	224.38	370.38	92.08	307.09	442.77	64.95	46.78	35.76	27.43	121.1	147	200.1	258.79	291.97	351.54	616441.287
6	64.77	112.4	141.47	246.98	34.48	223.61	377.59	92.65	307.44	444.32	65.96	47.53	35.93	27.3	117.1	142.6	195.14	255.71	290.32	352.26	604994.093
7	66.41	113.1	142.38	251.83	35.27	224.02	373.38	92.61	307.41	442.98	65.69	47.22	35.88	27.33	117.7	142.9	196.17	257.36	290.67	351.97	622286.609
8	64.62	114.9	143.4	247.69	35.99	225.2	377.66	93.32	311.04	442.74	65.84	47.38	35.69	27.32	116.5	142.6	194.7	254.78	290.46	351.34	630732.675
9	66.04	111.7	141.46	247.27	34.38	221.67	359.55	92.78	306.2	441.57	66.11	47.42	36.57	27.41	117.2	142	194.22	256.45	287.97	351.64	829397.033
10	66.53	115.3	144.1	251.63	36.37	223.66	368.72	92.66	307.76	442.26	65.67	47.14	36	27.39	118.1	144.9	197.12	257.63	290.43	351.3	649220.804
11	63.18	112.2	134.72	199.77	28.78	205.9	372.02	80.64	307.69	443.59	66.23	50.61	38.27	27.35	117.2	142.8	180.44	223.45	268.53	352.1	719001.732
12	66.72	112.7	135.97	200.75	29.64	203.75	355.11	80.02	303.67	459.1	65.86	50.41	38.9	27.46	118.7	144.9	181	225.73	267.82	352.26	1044673.91
13	69.45	119.6	140.76	207.41	34.12	206.59	366.32	81.76	306.76	459.72	64.34	49.47	38.15	27.34	123.6	149.8	186.5	228.33	270.63	352.05	865943.533
14	69.88	121.7	138.71	208.23	32.94	208.57	382.61	83.03	309.96	460.49	64.05	49.56	37.6	27.17	122.9	149.1	186.85	227.41	271.5	351.5	443919.613

Table 10.1 (Cont'd) – Simulation Outputs

15	67.79	116	137.94	200.18	31.74	205.62	371.71	80.59	306.61	459.96	65.33	50.21	38.15	27.34	119	146.7	182.78	223.83	269.08	352.12	820527.102
16	66.45	117.2	139.06	201.36	32.66	205.58	375.93	80.32	307.98	460.58	64.93	49.96	38.07	27.33	122	148.4	184.59	223.96	269.18	351.82	835117.123
17	66.41	117.5	146.99	262.97	39.51	227.1	370.13	93.84	309.63	442.81	65.15	46.28	35.42	27.33	120.5	146.5	202.08	263.43	294.21	352.13	655788.215
18	67.43	114.7	145.88	260.6	38.13	225.16	363.03	93.66	308.31	443.73	65.3	46.7	35.7	27.39	119.2	144.8	197.41	261.52	292.89	352.28	959951.009
19	67.89	117.6	145.64	262.63	38.55	227.1	373.46	94.41	309.13	441.76	64.93	46.41	35.36	27.27	120.3	145.7	200.77	262.28	293.78	351.96	546207.277
20	70.02	116.7	144.55	259.56	37.17	226.8	367.19	94.93	308.64	442.35	64.89	46.65	35.71	27.31	120.7	145.3	199.53	262.72	293.06	351.75	577165.771
21	67.7	115.4	145.13	260.55	37.6	227.58	369.88	95.19	309.4	441.55	65.29	46.67	35.48	27.34	115.9	143.5	197.83	261.68	293.27	351.51	628440.159
22	66.3	115	146.24	262.51	38.04	225.74	362.56	93.89	304.63	442.54	65.47	46.63	35.74	27.46	115.9	144.1	197.62	262.61	292.86	351.02	864806.71
23	66.58	113.3	145.29	258.01	37.22	228.01	363.83	94.45	306.39	443.53	65.65	46.78	35.61	27.5	115.5	143.2	197.56	264.22	295.19	351.71	811483.104
24	67.36	114.2	146.18	258.25	38.22	227.98	360.21	95.19	306.49	442.27	65.37	46.59	35.79	27.47	119	143.4	198.38	264.02	294.38	352.35	822978.406
25	67.19	115	145.68	260.13	38.18	226.87	370.34	94.18	307.3	441.91	65.29	46.53	35.49	27.39	119.7	145.4	200.34	263.11	294.29	351.41	680447.545
26	65.21	114.4	145.52	256.52	37.5	226.97	364.27	94.42	307.78	441.9	65.78	46.77	35.78	27.46	117.8	143.9	198.24	262.04	293.5	351.87	788235.247
27	66.14	112.6	144.4	254.74	36.8	224.63	357.25	93.82	305.2	441.07	66.07	46.85	36.17	27.54	118.7	145.1	198.9	263.12	292.9	351.68	873965.631
28	67.07	113.4	145.51	257.39	37.79	225.44	351.47	93.77	308.81	440.6	65.6	46.64	36.22	27.59	118.2	143.4	198.13	262.16	291.02	350.77	887363.716
29	67.14	114.7	144.84	259.58	37.35	226.44	369.69	93.47	307.92	442.64	65.41	46.69	35.51	27.44	118.1	143.5	198.07	261.18	293.04	351.62	676215.628
30	66.73	114.2	145.1	259.03	37.64	226.09	366.54	93.45	306.67	441.21	65.51	46.65	35.64	27.48	118.5	143.7	198.86	262.04	293.25	351.72	690129.799
31	66.23	113.2	145.47	259.7	37.89	226.82	363.45	94	308.78	440.79	65.7	46.51	35.67	27.5	118.6	143.8	199.4	263.13	293.28	351.2	788206.126
32	64.87	114.1	146.65	259.19	38.05	224.28	353.43	92.61	303.06	442.66	65.87	46.61	36.16	27.64	119.4	146	199.15	264.04	292.77	351.72	1078625.77
33	66.66	119	145.85	263.52	38.13	225.98	382.74	92.4	304.95	443.2	65.21	46.53	35.2	27.43	120.7	146.9	199.84	258.92	291.88	346.69	572373.459
34	66.47	113.2	142.97	256.69	35.74	225.16	368.81	93.39	308.88	441.01	65.83	46.88	35.78	27.4	118.1	143.2	197.46	259.22	290.7	349.71	650881.827
35	69.32	118.4	145.69	264.07	38.44	227.12	375.61	94.41	309.71	441.65	64.68	46.43	35.27	27.31	123.1	148.1	201.72	262.5	293.91	349.29	564699.196

Table 10.1 (Cont'd) – Simulation Outputs

36	67.99	112.8	143.38	255.93	35.77	228.13	368.98	95.91	312.71	441.59	65.63	46.99	35.64	27.29	115.2	139.2	192.31	258.74	291.31	351.38	860709.639
37	65.28	115.8	147.54	261.57	39.14	230.57	384.02	94.92	307.35	441.79	65.6	46.37	35.05	27.23	115.5	143.1	198.86	263.11	295.66	352.79	613116.496
38	64.49	119.5	149.74	260.32	39.95	229.97	383.79	94.83	309.27	440.98	65.47	46.36	35.12	27.29	116.6	144.8	198.96	261.13	294.44	352.33	618299.963
39	65.38	113.7	146.31	265.6	38.43	232.19	388.96	96.27	308.87	442.05	65.76	46.3	34.65	27.23	114.9	141	197.83	264.33	296.79	351.01	623178.578
40	66.24	114.3	143.45	241.1	36.3	224.36	384.79	93.4	309.66	442.39	65.78	47.57	35.82	27.24	116	142.8	192.88	251.38	288.36	350.85	669013.815
41	64.9	120.2	147.81	242.53	38.47	222.68	382.86	91.95	304.18	442.24	65.28	47.19	36.04	27.38	118.3	147	196.49	252.27	288.36	351.77	729475.614
42	65.7	115.9	144.42	242.72	36.68	223.03	387.33	92.19	306.63	442.92	65.47	47.38	35.93	27.28	117.4	144.4	196.24	252.18	288.32	351.67	602909.711
43	63.91	117.5	147.49	239.66	37.92	220.41	365.01	91.53	300.92	441.45	65.81	47.36	36.67	27.49	118.5	147.6	196.57	253.9	287.85	352.58	962640.389
44	67.74	115.8	145.21	244.79	37.72	225.53	384.64	93.88	308.49	440.33	65.24	47.16	35.75	27.37	117.6	144.3	197	255.85	290.68	351.34	579363.185
45	65.17	114	143.43	238.52	36.23	222.11	380.2	92.35	306.17	440.99	65.61	47.56	36.19	27.36	116.5	142.7	193.71	251.16	287.35	351.39	731825.94
46	70.96	118.3	146.55	244.57	38.26	224.77	381.78	92.97	306.19	442.2	64.74	47.25	35.69	27.41	119.3	147.4	195.71	255.45	291.32	351.92	851249.943
47	60.84	112.7	143.79	242.22	35.06	223.18	371.53	92.76	303.1	441.64	67.91	47.49	36.28	27.43	112.9	141.1	192.7	254.62	288.76	351.81	884614.043
48	66.66	113.7	146.71	261.5	38.73	230.39	374.38	95.68	310.89	440.63	65.53	46.38	35.19	27.38	117.3	142.9	198.72	264.31	295.84	351.59	766728.59
49	65.83	113.3	146.01	257.9	37.99	229.81	374.32	95.79	312.43	440.29	65.99	46.59	35.31	27.39	116.1	142.9	197.85	262.88	294.94	351.6	753434.254
50	66.74	113.5	144.19	251.96	36.71	227.76	378.72	95.04	313.82	441.62	65.79	47.03	35.45	27.31	117.4	143.6	196.45	259.03	293.1	351.49	699894.333
51	64.55	113.4	144.34	248.03	36.7	226.3	377.55	94.41	311.73	441.64	66.07	47.2	35.73	27.28	117.6	144.4	196.48	258.27	292.2	351.73	731095.021
52	67.21	114.5	144.99	247.66	37.35	225.61	372.25	94.34	312.16	441.68	65.64	47.15	35.9	27.29	118.3	145.4	196.22	257.49	291	351.68	791207.106
53	65.88	117.9	147.5	251.7	38.4	227.53	375.62	94.72	311.73	442.2	65.46	46.86	35.63	27.35	118.2	146.3	197.88	258.87	292.44	351.21	712685.558
54	66.88	115.6	146.44	252.65	38.47	227.4	377.93	94.71	312.22	441.68	65.46	46.85	35.5	27.34	117.9	145.5	197.9	258.97	292.77	351.2	672697.264
55	64.24	114.4	146.09	250.76	37.87	227.46	375.33	94.74	310.76	441.37	65.96	46.97	35.68	27.31	116.8	144.7	196.97	259.11	292.44	351.32	721832.149
56	66.03	114.6	145.55	247.21	37.54	225.75	369.71	93.72	309.6	441.18	65.92	47.17	35.96	27.33	118.4	146.4	196.87	258.51	291.77	351.52	817906.94

Table 10.1 (Cont'd) – Simulation Outputs

57	65.81	118.2	147.44	247.09	38.74	227.35	384.91	94.19	309.13	460.05	65.38	46.92	35.49	27.24	117.2	146.1	197.93	256.61	292.11	350.86	638829.772
58	63.75	116.8	147.82	249.89	38.49	228.85	382.51	94.71	307.08	443.9	65.98	46.75	35.49	27.29	116.3	144.7	198.27	259.51	293.44	351.02	691249.174
59	63.7	118.5	150.22	251.85	40.34	227.63	378.19	94.09	305.17	441.85	65.86	46.54	35.52	27.42	118.3	148	200.92	261.62	294.57	350.88	753359.739
60	61.76	115.7	147.18	246.66	37.77	223.75	371.69	92.38	303.28	441.76	66.87	47.02	36.05	27.47	115.7	146.2	197.83	258.13	291.37	351.49	858915.562
61	62.7	117.2	147.87	246.76	37.93	224.01	376.44	92.12	302.87	441.23	66.5	47.04	35.89	27.5	117.6	146.7	197.95	257.98	291.9	351.44	807880.585
62	61.24	115.5	146.32	251.29	36.63	224.74	380.28	93.34	302.27	440.65	66.78	46.92	35.71	27.46	116.5	145.9	199.3	261.23	293.68	351.68	685874.386
63	62.2	115.8	147.55	248.93	37.8	226.2	371.04	93.29	306.11	441.53	66.56	46.87	35.81	27.47	116.6	145.7	197.05	259.15	292.37	351.18	954272.838
64	60.81	113.2	146.22	246.98	37.13	225.54	360.37	93.04	304.68	442.07	67.45	47.1	36.13	27.56	114.2	144.4	193.8	259.24	291.77	351.55	1343705.38
65	64.43	118.5	148.9	252.82	38.54	226.63	375	93.7	306.75	441.72	65.84	46.72	35.59	27.44	118.9	148.2	199.65	261.81	294.29	351.06	827786.419
66	64.89	115.2	146.17	247.6	37.53	224.78	372.79	92.78	306.88	440.38	66.23	47.1	35.88	27.48	116.9	146.4	198.56	259.95	292.92	350.74	731924.554
67	62.63	115.5	146.86	246.15	37.31	224.48	366.17	92.89	305.28	440.23	66.64	47.1	36.13	27.52	115.4	145.9	197.86	259.65	292.05	350.73	809338.928
68	62.3	116.3	147.35	250.99	37.48	224.43	375.84	92.12	303.17	440.8	66.58	46.84	35.88	27.47	117.2	147.3	200.82	261.6	293.45	351.27	685462.032
69	62.31	115.9	146.86	247.21	37.27	224.9	374.75	92.76	306.3	439.72	66.68	47.11	35.92	27.44	115.4	145.9	198.42	259.59	292.51	350.76	692171.205
70	64.9	117	147.32	247.56	38.07	223.76	373.84	92.52	305.72	438.48	66.39	47.16	35.95	27.52	117.1	148.3	200.32	261.02	293.56	350.96	644715.933
71	64.43	117.2	147.47	246.76	37.96	224.18	369.62	93.23	307.01	440.81	66.5	47.2	36	27.44	116.9	147.6	197.58	258.73	291.85	350.65	771451.39
72	63.8	118.5	147.57	251.19	38.22	225.5	379.37	93.43	307.85	440.7	66.04	46.86	35.68	27.35	117.4	146.8	199.26	259.23	292.16	350.2	549996.527
73	65.8	119.7	146.71	224.64	37.08	215.44	361.25	88.49	303.27	440.51	66.06	48.28	37.38	27.58	117.1	150	193.91	245.27	282.07	350.89	831015.326
74	66.74	121	147.01	228.47	37.46	218.67	373.09	89.25	307.91	442.04	65.35	47.99	36.69	27.45	119.2	149.3	194.03	245.4	283.84	350.67	739632.867
75	62.95	117.9	146.79	224.8	37.21	217.38	368.57	88.09	306.58	441.37	66.65	48.18	37.02	27.51	115.4	147.5	192.21	243.72	282.08	350.88	936728.557
76	66.81	121.5	147.74	224.26	37.93	216.3	377.18	88.07	306.14	441.09	65.82	48.24	36.85	27.5	118.5	151.7	195.54	244.96	283.68	351.1	711063.465
77	66.67	119.5	146.73	224.5	37.79	217.53	364.16	89.21	308.82	441.75	65.79	48.26	37.18	27.46	117.5	148.8	191.97	243.54	280.89	350.02	879462.061

Table 10.1 (Cont'd) – Simulation Outputs

78	66.61	120	146.2	222.99	37.59	217.61	373.46	88.91	309.41	441.24	65.77	48.4	36.97	27.41	118.2	149.2	192.15	242.42	281.33	350.13	732406.3
79	65.37	119.8	146.18	223.5	37.36	215.96	375.35	87.98	305.04	441.08	66.21	48.4	37.02	27.48	117.2	149.6	193.31	243.81	282.3	351.01	732406.3
80	68.51	120.5	147.66	224.65	38.63	216.18	366.67	88.15	303.74	442.92	65.69	48.24	37.11	27.47	119.5	151.6	193.97	245.22	283.11	351.17	966787.396
81	67.16	119.2	145.8	224.24	37.6	216.94	377.08	88.35	305.58	442.87	66.13	48.39	36.84	27.41	118.3	149.9	193.26	244.65	283.65	351.23	815875.625
82	64.31	119	149.01	250.68	38.66	226.02	378.37	93.24	307.79	442.33	65.96	46.83	35.63	27.42	116.2	146.7	198.78	258.57	292.36	350.75	646159.302
83	66.25	120	150.18	247.84	39.64	226.06	361.47	94.07	308.03	440.5	65.68	46.84	36.07	27.55	117.5	148.3	198.21	259.23	291.58	350.08	847995.603
84	67.33	121.4	150.03	248.75	39.75	227.81	378.79	94.19	309.87	441.8	65.27	46.81	35.54	27.36	117.6	147.6	198.27	257.75	292.37	350	626154.539
85	68.1	124	152.29	252.15	41.48	228.21	377.32	94.53	309.67	440.93	64.74	46.5	35.47	27.41	119.4	150	200.54	259.7	293.09	349.86	594271.222
86	64.33	118.9	149.67	251.31	39.44	227.68	365.98	94.88	307.56	439.74	65.79	46.6	35.9	27.48	115.8	145.6	198.05	260.01	291.65	349.96	766116.605
87	65.98	121.3	151.47	246.37	40.4	225.66	360.39	94.19	306.82	440.91	65.66	46.74	36.22	27.53	118.5	149.8	199.23	258.93	290.98	349.79	917659.544
88	65.11	119.9	149.45	251.48	39.71	226.5	371.14	94.28	308.07	439.7	65.35	46.62	35.79	27.48	120.2	149	200.88	260.69	292.86	349.65	689340.925
89	64.89	121.2	149.79	247.17	39.51	224.92	374.69	93.44	308.56	439.7	65.59	46.89	35.91	27.47	119.5	150.9	201.51	259.32	292.36	350.11	582704.232
90	65.25	118.6	148.49	245.14	38.56	225.67	368.62	94.13	308.56	439.81	65.99	47.07	35.99	27.53	117.5	147.7	198.41	258.88	291.93	350.24	692095.317
91	66.42	121	150.01	242.72	39.57	225.08	367.7	93.67	308.46	440.33	65.55	47.04	36.17	27.48	117.6	148.3	197.87	255.92	289.61	349.86	714068.765
92	65.06	120.4	149.6	250.29	39.86	225.43	372.02	93.58	307.84	440.2	65.35	46.68	35.82	27.5	119.8	149.4	201.37	260.23	292.58	349.96	622728.561
93	64.65	119.6	149.67	245.41	39.39	225.63	362.88	94.32	307.94	440.44	65.69	46.86	36.23	27.5	116.8	146.7	197.43	257.02	289.56	349.97	784605.366
94	65.27	119.6	148.92	252.37	39.37	225.78	369.19	93.66	308.49	439.29	65.52	46.78	35.73	27.55	119.2	149.2	200.42	261.55	293.75	350.58	679321.833
95	65.02	119.8	148.14	245.39	38.61	224.54	373.85	93.21	308.97	438.88	65.62	47.13	35.88	27.53	119	149.3	199.3	257.69	291.84	350.08	621196.45
96	68.13	123.3	150.27	246.94	40.47	224.38	376.24	92.95	309.13	438.18	64.72	46.92	35.76	27.54	120.8	151.6	200.67	257.61	292.01	350.45	585535.359
97	64.55	120	149.45	246.19	39.43	225.69	370.42	93.98	308.04	440.26	65.57	46.86	35.96	27.51	117.3	146.5	197.9	256.56	290.3	350.1	657881.349
98	65.94	121.5	151.38	249.16	41.15	227.08	375.99	94.43	309.45	441.26	64.95	46.53	35.8	27.37	119.2	148.3	200.07	257	290.25	349.68	605707.492

Table 10.1 (Cont'd) – Simulation Outputs

99	64.17	118.8	148.38	239.58	39.26	225.5	376.92	93.81	310.74	442.01	65.71	47.19	36.05	27.37	115.6	145.2	195.51	250.97	287.16	349.44	571943.301
100	65.04	120.8	150.75	252.61	40.37	226.83	376.11	93.79	307.03	442	65.26	46.52	35.7	27.38	119.3	148.3	200.39	259.38	292.04	350.39	662594.508
101	64.78	119.7	149.12	247.92	39.06	226.37	375.88	93.83	306.47	441.73	65.68	46.84	35.89	27.34	116.8	146.1	197.58	257.18	290.53	350.68	684060.766
102	65.89	122.1	151.63	252.17	40.88	228.19	371.43	94.73	308.94	441.82	65.13	46.49	35.76	27.35	117.5	147.8	198.89	258.8	291.13	349.9	693717.704
103	64.71	119.3	150.05	244.44	39.83	225.74	366.12	93.88	307.34	441.59	65.6	46.86	36.23	27.42	117.3	146.7	197.33	255.94	289	350.23	805556.486
104	65.41	120.3	151.67	247.38	41.25	226.45	360.11	94.34	308.1	441.31	65.42	46.65	36.23	27.47	117.8	148	198.54	257.92	289.79	349.95	856275.84
105	67.2	120.6	150.08	247.02	39.99	226.92	367.85	94.75	309.8	440.87	65.14	46.91	35.92	27.5	116.7	147	197.07	256.83	290.5	349.82	691751.154
106	66.9	119.7	148.51	244	39.04	225.73	373.26	93.99	309.06	441.55	65.32	47.1	35.92	27.47	117.4	147.3	197.18	255.67	290.14	349.64	648218.801
107	63.64	118.3	149.86	246.07	40.75	228.23	376.66	95.37	307.92	440.85	65.79	46.75	35.75	27.48	116.7	144.8	197.77	256.45	291.09	349.42	646651.199
108	69.13	117.9	146.19	244.78	38.48	227.15	382.1	94.39	310.29	441.01	64.69	47.13	35.67	27.38	119.6	146.7	197.88	256.34	291.95	350.07	532898.691
109	65.03	121.5	150.48	249.73	40	226.77	378.05	94.74	306.71	439.37	65.47	46.72	35.57	27.58	121.4	152.4	202.99	262.97	295.9	350.7	675538.648
110	68.6	120.3	150.33	249.82	41.13	228.38	363.9	95.42	308.08	441.5	64.75	46.54	35.91	27.57	119.8	148.5	200.65	261.42	293	349.61	716434.065
111	66.98	115.7	146.89	243.07	38.34	226.04	362.56	94.53	305.32	441.01	65.6	47.19	36.13	27.65	119.8	146.6	197.23	259.6	292.94	350.27	894716.871
112	67.3	121.5	149.89	250.52	41.37	228.53	374.55	95.39	309.84	439.47	64.68	46.56	35.59	27.56	120.2	148.3	200.88	260.19	293.37	349.09	615450.368
113	68.79	121.2	148.84	251.87	40.91	229.5	375.55	95.31	312.28	438.07	64.33	46.49	35.46	27.54	120.3	146.4	199.73	259.06	292.78	349.74	573537.567
114	67.96	118.6	148.11	251.18	39.39	228.53	375.4	95.6	308.75	436.82	65.07	46.83	35.47	27.63	118.9	146.1	197.92	260.8	294.41	350.28	672643.981
115	66.86	121.6	150.18	245.18	40.72	226.13	367.38	94.38	309.55	437.24	64.83	46.89	36	27.66	119.8	148.1	198	256.52	290.54	349.84	745361.264
116	65.99	115.3	145.43	244.36	37.5	228.26	377.31	95.47	311.2	438.65	65.7	47.24	35.7	27.47	115.4	142.5	194.77	256.45	291.55	349.84	577067.421
117	65.48	118.8	149.49	243.33	39.1	226.84	360.88	95.05	307.38	438.94	65.85	47.03	36.18	27.67	117	147	196.32	258.09	291.57	350.36	985328.388
118	66.47	116.1	146.94	245.35	38.38	226.51	365.28	94.67	308.68	438.81	65.61	47.05	36.05	27.63	117.3	145	196.39	257.95	291.18	350.1	783537.579
119	67.78	121.5	150.59	248.25	40.88	227.41	366.99	95.01	308.48	437.73	64.86	46.67	35.88	27.67	120.6	149.4	200.04	259.98	292.57	349.86	747222.657

Table 10.1 (Cont'd) – Simulation Outputs

120	68.34	121.2	149.22	249.25	40.09	226.13	370.27	94.4	308.7	434.92	64.8	46.73	35.79	27.73	121.5	149.7	200.28	259.9	292.76	349.67	694766.106
121	67.18	124.1	150.25	249.3	41.01	226.26	376.24	94.23	308.08	434.81	64.54	46.63	35.68	27.73	120.1	149.7	200.37	257.82	291.86	349.9	528234.02
122	66.99	120.5	149.21	244.6	39.56	224.02	359.97	93.45	304.2	438.83	65.26	46.98	36.39	27.74	116.7	146.5	196.6	256.38	289.39	350.53	835278.196
123	64.69	116.8	147.53	254.37	38.98	228.94	375.81	95.72	307.11	438.28	65.6	46.6	35.41	27.66	118.4	145.9	200.69	263.83	296.22	350.36	607153.978
124	65.72	120.4	148.59	246.35	39.36	225.13	373.86	93.86	304.7	438.37	65.25	46.98	35.88	27.71	117.4	146.8	197.72	257.13	291.48	350.31	686728.637
125	63.59	117.1	147.32	246.05	38.07	225.15	368.49	93.73	304.28	437.58	66.01	47.09	35.98	27.79	116.6	145.2	196.94	258.44	292.2	350.91	753860.926
126	65.47	121.6	148.04	251.15	39.49	225.62	379.87	93.38	305.36	438.95	65.04	46.78	35.53	27.66	120	147.9	199.69	259.3	293.68	350.66	590996.747
127	65.98	119.5	148.26	245.12	38.71	224.13	364.85	93.16	305.14	439.07	65.61	47.09	36.21	27.66	117.7	147.7	197.66	257.63	291.17	350.96	789656.804
128	67.91	118.3	146.35	246.67	38.38	224.68	372.1	93.2	305.63	440.15	65.06	47.08	35.93	27.61	118.4	146	197.12	257.32	291.41	350.73	653822.952
129	67.59	120.9	148.4	249.47	39.91	225.83	373.41	93.53	306.38	438.91	64.86	46.75	35.71	27.67	119.3	147.1	198.46	258.01	292.22	350.57	720652.438
130	66.64	122.7	149.64	249.77	40.44	225.98	375.93	93.68	307.42	436.39	64.77	46.72	35.65	27.71	119.5	148.4	199.22	257.67	292.27	350.83	649861.333
131	66.46	121.5	148.56	247.01	39.56	224.95	378.11	93.38	306.48	438.57	65	46.91	35.67	27.67	121.3	150	199.85	258.37	293.17	351.13	708800.953
132	68.67	118.4	146.88	247.33	38.72	224.67	373.27	93.13	305.58	439.69	64.83	47.04	35.84	27.67	120.6	146.8	197.48	257.12	291.96	351.08	767832.258
133	67.67	118.9	147.93	247.1	39.17	224.59	364.66	93.55	305.4	439.89	65.09	47.02	36.04	27.76	120.2	148.5	198.04	258.5	292.37	350.94	882315.353
134	66.87	120.9	146.44	243.31	38.86	224.96	378.04	93.65	307.88	441.22	64.85	47.15	35.9	27.56	118.6	147.2	197.92	254.83	290.67	350.36	491918.347
135	65.43	120.5	148.61	246.8	39.88	225.12	367.37	93.8	305.26	440.76	65.21	46.82	36.05	27.65	117.3	145.8	197.93	257.16	290.77	350.36	684916.544

APPENDIX II – Modelling Codes in R

Feed TBP Modelling

```
# Set directory and load functions and libraries
setwd("H:/THESIS - FEED TBP MODELS")
rm(list=ls())
source("calculate.RMSE.R")
source("norm.input.R")
library(e1071)      # install.packages(e1071)
library(beepR)      # install.packages("beepR")
library("sfsmisc")  # install.packages("emdbook")
library(neuralnet)  # install.packages("neuralnet")
library(MASS)       # install.packages("MASS")

----- Generate RData file from CSV
# alldata = read.csv("DATA.CSV",sep=";",header=TRUE)
# n=nrow(alldata) # Number of samples
# ntrain <- round(n*0.85) # Prepare train and test data
# tindex <- sample(n,ntrain)
# save.image("DATA.RData")

load("DATA.RData")

alldata=subset(alldata,select=c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,Y2))  #Choose the desired output from Y1,Y2,...,Y13
#y1: IKN y2: %5 y3: %10 y4: %20 y5: %30 y6: %40 y7: %50 y8: %60 y9: %70 y10:%80 y11:%90 y12:%95 y13:SKN
colnames(alldata)[12] <- "Y"

#alldata=norm.input(alldata)
datatrain <- alldata[tindex,]
datatest <- alldata[-tindex,]

#####
#####
#####
# GRID SEARCH AREA -----

## PARAMETERS FOR SVM
# cost.range=lseq(0.1,10000,20)
# gamma.range=lseq(0.0001,10000,20)
```

```

# eps=seq(0,2,length=11)

cost.range=lseq(0.1,10000,20)
gamma.range=lseq(0.0001,10000,20)
eps=seq(0,2,length=11)

gridarea=data.frame(cost.range,gamma.range,eps)

## PARAMETERS FOR ANN
k=50 # Max number of neurons in hidden layer 1

option=2    # SVM Kernel Options >>>>  1:Linear 2:Gaussian 3:Polynomial 4:Sigmoid
          # Others >>>>                5:Linear regression 6: ANN

tc <- tune.control(cross = 10)

print.lin.results=2 # If =1, then show linear regression results on trend plots, if=0, do not show !!
#####
#####
#####
start= Sys.time()    # or use ptm <- proc.time()
start

if(option==1){
  tuneResult <- tune(svm, Y ~ ., data = datatrain,kernel="linear",scale=TRUE, ranges = list(epsilon=eps,cost= cost.range))
  tunedModel <- tuneResult$best.model
}

if(option==2){
  #tuneResult <- tune(svm, Y ~ ., data = datatrain,kernel="radial",scale=TRUE, ranges = list(epsilon=eps,cost= cost.range,
gamma=gamma.range),tunecontrol=tune.control(sampling="cross", cross = 10,performances = TRUE))
  #tunedModel <- tuneResult$best.model

# libsvm ile Cross-val deneme
tuneResult=tune(svm, Y ~ ., data = datatrain,kernel="radial",scale=TRUE, ranges = list(epsilon=eps,cost= cost.range, gamma=gamma.range),tunecontrol=tc)
tunedModel <- tuneResult$best.model

# Kernlab toolbox ile Cross-val deneme
#tunedModel <- ksvm(Y~.,data=datatrain,kernel="rbfdot", type="eps-svr", kpar=list(C=cost.range,sigma=gamma.range,epsilon=eps),,cross=10)

```

```

}

if(option==3){
  tuneResult <- tune(svm, Y ~ ., data = datatrain, kernel="polynomial", scale=TRUE, degree=3, coef0=1, ranges = list(epsilon=eps, cost= cost.range,
gamma=gamma.range))
  tunedModel <- tuneResult$best.model
}

if(option==4){
  tuneResult <- tune(svm, Y ~ ., data = datatrain, kernel="sigmoid", scale=TRUE, coef0=0, ranges = list(epsilon=eps, cost= cost.range, gamma=gamma.range))
  tunedModel <- tuneResult$best.model
}

if(option==5){
  tunedModel <- glm(Y~., data=datatrain)
  summary(tunedModel)

  pr.lm.train <- predict(tunedModel, datatrain)
  RMSE.lm.train <- sqrt(sum((pr.lm.train - datatrain$Y)^2)/nrow(datatrain))
  pr.lm.test <- predict(tunedModel, datatest)
  RMSE.lm.test <- sqrt(sum((pr.lm.test - datatest$Y)^2)/nrow(datatest))
}

if(option==6){
  temp=1
  if(temp==1){
    maxs <- apply(alldata, 2, max) #Storing the max and min values for 0-1 normalization
    mins <- apply(alldata, 2, min)
    scaled.alldata <- as.data.frame(scale(alldata, center = mins, scale = maxs - mins))
    datatrain.scaled <- scaled.alldata[tindex,]
    datatest.scaled <- scaled.alldata[-tindex,]
    names <- names(datatrain.scaled)
    formula <- as.formula(paste("Y ~", paste(names[!names %in% "Y"], collapse = " + "))) #For some reason the formula y~. is not accepted in the neuralnet()
function. You need to first write the formula and then pass it as an argument in the fitting function.
    no.of.inputs=ncol(scaled.alldata)-1
    RMSE.results=data.frame(matrix(0, ncol = 2, nrow = k))
  }
}

```

```

for(i in 1:k){
  structure=c(i)
  tunedModel <- neuralnet(formula,data=datatrain.scaled,hidden=structure, algorithm="backprop",learningrate=0.01, err.fct="sse",linear.output=FALSE)

  y.est.nnet.scaled.train <- neuralnet::compute(tunedModel, datatrain.scaled[,1:no.of.inputs]) # Normalized y values for train data
  y.est.nnet.train <- y.est.nnet.scaled.train$net.result*(max(alldata$Y)-min(alldata$Y))+min(alldata$Y) # y values for train data

  y.est.nnet.scaled.test <- neuralnet::compute(tunedModel, datatest.scaled[,1:no.of.inputs]) # Normalized y values for test data
  y.est.nnet.test <- y.est.nnet.scaled.test$net.result*(max(alldata$Y)-min(alldata$Y))+min(alldata$Y) # y values for test data

  RMSE.train <- sqrt(sum((y.est.nnet.train - datatrain$Y)^2)/nrow(datatrain))
  RMSE.test <- sqrt(sum((y.est.nnet.test - datatest$Y)^2)/nrow(datatest))

  RMSE.results[i,1]=RMSE.train
  RMSE.results[i,2]=RMSE.test
}

if(option==7){
  n_folds <- 10
  folds_i <- sample(rep(1:n_folds, length.out = nrow(datatrain)))
  cv_tmp <- matrix(NA, nrow = n_folds, ncol = 1)
  k=1
  for (k in 1:n_folds) {
    test_i <- which(folds_i == k)
    train_cv <- datatrain[-test_i, ]
    test_cv <- datatrain[test_i, ]
    model = lm(Y ~ .,data = train_cv)
    y_test_est=predict(model,test_cv)
    cv_tmp[k, ]= sum((test_cv$Y -y_test_est )^2)/nrow(test_cv) #MSE
  }
  cv_mse = colMeans(cv_tmp)

```

```
cv_rmse = sqrt(cv_mse)
cv_rmse
}
}

#summary(tuneResult)

end=Sys.time() #proc.time() - ptm
Tuning.duration=end-start
cat("Training duration : ",Tuning.duration)

print(tunedModel)

if(option!=5 | option!=6){
  plot(tuneResult,main="Parameter Search - Mean Squared Error")
}

#####
#####
#####
# Step 2 - Loading and Saving Models ----

# save(tuneResult,file="model%95-3tune.rda")
# save(tunedModel,file="model%95-3.rda")

gecici=1
if(gecici==2){
  gridsearch.results=tuneResult$performances
  write.table(gridsearch.results, file = "gridresults.csv", sep = ",")
}

# load(file="model%95-1tune.rda")      # It brings tuneResult # DIKKAT !!!!!!!!!!!!!!!
# load(file="model%95-1.rda")         # It brings tunedModel # DIKKAT !!!!!!!!!!!!!!!

#####
#####
##### RMSE , Trends, Histograms, Error Analysis Section#####
```

```
#####
#####
```

RMSE of Training

```
if(option!=6){
tunedModelY <- predict(tunedModel, datatrain)
}else{
  tunedModelY = y.est.nnet.train
}
error.train <- datatrain$Y - tunedModelY
tunedModelRMSE <- calculate.RMSE(error.train)
RMSE.training=calculate.RMSE(error.train)
cat("Training error(RMSE): ", RMSE.training)
```

RMSE of Testing :

```
if(option!=6){
  Y.test.est <- predict(tunedModel,datatest)
}else{
  Y.test.est = y.est.nnet.test
}
result.test=data.frame(datatest$Y,Y.test.est)
error.test= datatest$Y-Y.test.est
RMSE.test=calculate.RMSE(error.test)
cat("Test error(RMSE): ", RMSE.test)
```

TREND PLOTS ! GÜNCELLE - Sadece PLOT YLABEL!

```
par(mfrow = c(1,2))
# Testing Trend :
plot(datatest$Y,ylim=c(min(alldata$Y)*0.75,max(alldata$Y))*1.2,panel.first = grid(nx=20,ny=30),ylab = "Feed TBP%5(Celcius)",main = "Test Performance" )
points(Y.test.est,col="red",pch=16)
if(print.lin.results==1){
  linearmodel = glm(Y~., data=datatrain)
  lin.y.train <- predict(linearmodel, datatrain)
  lin.y.test <- predict(linearmodel, datatest)
  points(lin.y.test,col="blue",pch=15,cex=0.5) #Linear regression results
}
legend('topright', c('Real Value','Model Estimation') , pch=c(1,1), col=c('black', 'red'), bty='y', cex=1.0)
```

```
# Training Trend :
plot(datatrain$Y,ylim=c(min(alldata$Y)*0.75,max(alldata$Y)*1.2),panel.first = grid(nx=20,ny=30),ylab = "Feed TBP%5(Celcius)",main = "Training Performance" )
points(tunedModelY,col="red",pch=16)
if(print.lin.results==1){
points(lin.y.train,col="blue",pch=15,cex=0.5) #Linear regression results
}
legend('topright', c('Real Value','Model Estimation') , pch=c(1,1), col=c('black', 'red'), bty='y', cex=1.0)

### Histograms ###
sonuc.test=data.frame(datatest$Y,Y.test.est,error.test)
names(sonuc.test)=c("Real","Estimated","Error")
sonuc.train=data.frame(datatrain$Y,tunedModelY,error.train)
names(sonuc.train)=c("Real","Estimated","Error")
par(mfrow = c(1,2))
hist(error.train)
#legend('topright', legend = mylabel, bty = 'n')
hist(error.test)
#legend('topright', legend = mylabel, bty = 'n')

### R2 plots #####
par(mfrow = c(1,2))
# Train Plot
par(pty="s")
a=min(alldata$Y)
b=max(alldata$Y)
plot(datatrain$Y,tunedModelY,ylim=c(a,b),xlim=c(a,b),ylab='Predicted Y',xlab='Actual Y',main="Actual Y vs Estimated Y - TRAINING")
abline(0,1)
fit.lm = lm(tunedModelY ~ datatrain$Y)
r2.test=summary(fit.lm)$r.squared
mylabel = bquote(italic(R)^2 == .(format(r2.test, digits = 3)))
legend('topleft', legend = mylabel, bty = 'n')
# Test Plot
par(pty="s")
a=min(alldata$Y)
b=max(alldata$Y)
plot(datatest$Y,Y.test.est,ylim=c(a,b),xlim=c(a,b),ylab='Predicted Y',xlab='Actual Y',main="Actual Y vs Estimated Y - TEST")
abline(0,1)
```

```
fit.lm = lm(Y.test.est ~ datatest$Y)
r2.test=summary(fit.lm)$r.squared
mylabel = bquote(italic(R)^2 == .(format(r2.test, digits = 3)))
legend('topleft', legend = mylabel, bty = 'n')
```

```
## FINAL SUMMARIES ##
summary(tunedModel)
print(tunedModel)
cat("Training error(RMSE): ", RMSE.training)
cat("Test error(RMSE): ", RMSE.test)
cat("Training duration : ", Tuning.duration)
start
end
#beep(3)

plot(error.train)
plot(error.test)
```

Simulation Error Modelling Code

```
# Set directory and load functions and libraries
setwd("E:/NEW_THESIS- SIMULATION ERROR MODELLING")
rm(list=ls())
```

```
library(e1071)      # install.packages(e1071)
library(beepr)      # install.packages("beepr")
library("sfsmisc")  # install.packages("emdbook")
library(neuralnet)  # install.packages("neuralnet")
library(MASS)       # install.packages("MASS")
library(rpart)      # install.packages("rpart")
library(randomForest) # install.packages("randomForest")
library(mlbench)    # install.packages("mlbench")
library(caret)      # install.packages("caret")
library(rJava)      # install.packages("rJava")
library(xlsx)       # install.packages("xlsx")
library(beepr)      # install.packages("beepr")
library(kernlab)    # install.packages("kernlab")
```

```

library(progress) # install.packages("progress")
library(plyr)
source("norm.input.R")

##### USER INPUTS #####
literature=0 # If literature==1, then only F and X variables will be used. 0 or 1
TrainSubset=2 # S1: 7-100,S2:51-100,S3:76-100,S4:91-100
option=2 # 1: Linear Regression 2: Random Forest 3: SVR (RBF or Linear Kernel) 4:ANN
PARAMETER_OPTIMIZATION= 0 # Only for Random Forest

myvars_all = c("X35","X49","X62","X63","Y") #CHANGE
if(option==2){
  if(literature==0){myvars_all=c("X1" , "X2" , "X3" , "X4" , "X5" , "X6" , "X7" , "X8" , "X9" , "X10" , "X11" , "X12" , "X13" , "X14" , "X15" , "X16" , "X17"
, "X18" , "X19" , "X20" , "X21" , "X22" , "X23" , "X24" , "X25" , "X26" , "X27" , "X28" , "X29" , "X30" , "X31" , "X32" , "X33" , "X34" , "X35" , "X36" , "X37" , "X38"
, "X39" , "X40" , "X41" , "X42" , "X43" , "X44" , "X45" , "X46" , "X47" , "X48" , "X49" , "X50" , "X51" , "X52" , "X53" , "X54" , "X55" , "X56" , "X57" , "X58" , "X59"
, "X60" , "X61" , "X62" , "X63" , "Y")}
  if(literature==1){myvars_all=c("X1" , "X2" , "X3" , "X4" , "X5" , "X6" , "X7" , "X8" , "X9" , "X10" , "X11" , "X12" , "X13" , "X14" , "X15" , "X16" , "X17"
, "X18" , "X19" , "X20" , "X21" , "X22" , "X23" , "X24" , "X25" , "X26" , "X27" , "X28" , "X29" , "X30" , "X31" , "X32" , "X33" , "Y")}
}

a="Y5"

# string_temp=paste(a,"_",b,"_", "_Random Forest Parameter Search","_",cix,".jpg",sep="")
# jpeg(string_temp,width = 1080, height = 780, units = "px",quality = 100)
# plot(custom)
# dev.off()

# PARAMETERS FOR RANDOM FOREST
if(literature==0){mtry.range=c(1,3,6,10,20,30,40,50,60,63)}
if(literature==1){mtry.range=c(1,3,6,10,15,20,25,30,33)}
ntree.range=c(50,100,500,1000)
cross_rf=5
tree_number= 100 # Change
Candidate_variables= 1 # Change (Number of variables randomly sampled as candidates at each split.)

# PARAMETERS FOR SVR
cost.range=lseq(0.1,10000,10)

```

```

gamma.range=lseq(0.0001,10000,10)
eps=seq(0,2,length=11)
tc <- tune.control(cross = 5)
cekirdek="radial"           # CHANGE "radial" or "linear"
scaling=FALSE               # CHANGE

# PARAMETERS FOR ANN
max_neuron=20
ANN_cross_validation=5

##### VARIABLES #####
if(a=="Y1"){b='HSRN 5'}
if(a=="Y2"){b='HSRN 95'}
if(a=="Y3"){b='KEROSENE 5'}
if(a=="Y4"){b='KEROSENE 95'}
if(a=="Y5"){b='KEROSENE FLASH'}
if(a=="Y6"){b='DIESEL 5'}
if(a=="Y7"){b='DIESEL 95'}
if(a=="Y8"){b='DIESEL FLASH'}
if(a=="Y9"){b='BOTTOM 5'}
if(a=="Y11"){b='HSRN API'}
if(a=="Y12"){b='KEROSENE API'}
if(a=="Y13"){b='DIESEL API'}
if(a=="Y14"){b='BOTTOM API'}

i=1
if(option==2){
  if(literature==0){i=63}
  if(literature==1){i=33}
}

# Case 1(4) ve Case 2(33) için deęiřtir
tot_no_of_vars=length(myvars_all)
i_end=tot_no_of_vars-1

#####
for(i in 1:i_end){

```

```

gecici=1
if(gecici==1){
  if(a=="Y1"){load("Data_Feature_Selected_Models_SIMERR_Y1.RData")}
  if(a=="Y2"){load("Data_Feature_Selected_Models_SIMERR_Y2.RData")}
  if(a=="Y3"){load("Data_Feature_Selected_Models_SIMERR_Y3.RData")}
  if(a=="Y4"){load("Data_Feature_Selected_Models_SIMERR_Y4.RData")}
  if(a=="Y5"){load("Data_Feature_Selected_Models_SIMERR_Y5.RData")}
  if(a=="Y6"){load("Data_Feature_Selected_Models_SIMERR_Y6.RData")}
  if(a=="Y7"){load("Data_Feature_Selected_Models_SIMERR_Y7.RData")}
  if(a=="Y8"){load("Data_Feature_Selected_Models_SIMERR_Y8.RData")}
  if(a=="Y9"){load("Data_Feature_Selected_Models_SIMERR_Y9.RData")}
  if(a=="Y11"){load("Data_Feature_Selected_Models_SIMERR_Y11.RData")}
  if(a=="Y12"){load("Data_Feature_Selected_Models_SIMERR_Y12.RData")}
  if(a=="Y13"){load("Data_Feature_Selected_Models_SIMERR_Y13.RData")}
  if(a=="Y14"){load("Data_Feature_Selected_Models_SIMERR_Y14.RData")}
}
if(!option==2){no_of_vars=i} # Number of input vars
if(option==2){no_of_vars=i_end}
var=paste(no_of_vars,"_variable",sep="")

myvars = myvars_all
myvars = myvars[c(1:no_of_vars,tot_no_of_vars)]
alldata=alldata[myvars]
datatrain=datatrain[myvars]
datatest=datatest[myvars]

if(literature==1){
  alldata=subset(alldata,select=
c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16,X17,X18,X19,X20,X21,X22,X23,X24,X25,X26,X27,X28,X29,X30,X31,X32,X33,Y))
  datatest=subset(datatest,select=
c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16,X17,X18,X19,X20,X21,X22,X23,X24,X25,X26,X27,X28,X29,X30,X31,X32,X33,Y))
  datatrain=subset(datatrain,select=
c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16,X17,X18,X19,X20,X21,X22,X23,X24,X25,X26,X27,X28,X29,X30,X31,X32,X33,Y))
}
if(TrainSubset==1){
  datatrain=alldata[7:100,]
  datatest=alldata[101:135,]
}

```

```

if(literature==0){ cix="SI=(0,1)" }
if(literature==1){ cix="SI=(1,1)" }
}
if(TrainSubset==2){
  datatrain=alldata[91:100,]
  datatest=alldata[101:135,]
  if(literature==0){ cix="SI=(0,2)" }
  if(literature==1){ cix="SI=(1,2)" }
}

##### MODEL BUILDING #####

start= Sys.time()    # or use ptm <- proc.time()
start

# Linear Regression
if(option==1){
  model = lm(Y ~ .,data = datatrain) }

# Random Forest
if(option==2){

  # model=tuneRF(Y~., data=datatrain, ntreeTry=50, stepFactor=2, improve=0.05,trace=TRUE, plot=TRUE, doBest=FALSE)

  if(PARAMETER_OPTIMIZATION==1){
    temp=1
    if(temp==1){
      customRF <- list(type = "Regression", library = "randomForest", loop = NULL) # Or "Regression"
      customRF$parameters <- data.frame(parameter = c("mtry", "ntree"), class = rep("numeric", 2), label = c("mtry", "ntree"))
      customRF$grid <- function(x, y, len = NULL, search = "grid") {}
      customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
        randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
      }
      customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
        predict(modelFit, newdata)
      customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
        predict(modelFit, newdata, type = "prob")
      customRF$sort <- function(x) x[order(x[,1]),]
    }
  }
}

```

```

start= Sys.time()    # or use ptm <- proc.time()
start

control <- trainControl(method="repeatedcv", number=cross_rf, repeats=3)
tunegrid <- expand.grid(.mtry=mtry.range, .ntree=ntree.range)
custom <- train(Y~, data=datatrain, method=customRF, metric="RMSE", tuneGrid=tunegrid, trControl=control)
summary(custom)

end=Sys.time() #proc.time() - ptm
Tuning.duration=end-start

cat("Training duration : ",Tuning.duration)

#####

string_temp=paste(a,"_",b,"_", "_Random Forest Parameter Search","_",cix,".jpg",sep="")
jpeg(string_temp,width = 1080, height = 780, units = "px",quality = 100)
plot(custom)
dev.off()

#####

file <- paste(getwd(), "/",a,"_",b,"_", "_Random Forest Tuning ",cix,".xlsx", sep="")
write.xlsx(custom$results, file,sheetName="Tuning Results",append=TRUE)
write.xlsx(custom$bestTune, file,sheetName="Best Tune",append=TRUE)

}

if(PARAMETER_OPTIMIZATION==0){
  model <- randomForest(Y~, data=datatrain, ntree=tree_number,mtry=Candidate_variables,importance=T) }

if(PARAMETER_OPTIMIZATION==0 ){
  string_temp=paste(a,"_",b,"_", "_Variable Importance Plot","_",cix,".jpg",sep="")
  jpeg(string_temp,width = 1080, height = 780, units = "px",quality = 100)
  varImpPlot(model,sort = T,main="Variable Importance",n.var=10)
  dev.off()
}

```

```
#View(importance(model))
}
}

# SVR (RBF Kernel)
if(option==3){
  if(cekirdek=="radial"){tuneResult= tune(svm, Y ~ ., data = datatrain,kernel="radial",scale=scaling, ranges = list(epsilon=eps,cost= cost.range,
gamma=gamma.range),tunecontrol=tc)}
  if(cekirdek=="linear"){tuneResult <- tune(svm, Y ~ ., data = datatrain,kernel="linear",scale=scaling, ranges = list(epsilon=eps,cost=
cost.range),tunecontrol=tc)}

  #model <- ksvm(Y~,data=datatrain,kernel="rbfdot", type="eps-svr", kpar=list(C=cost.range,sigma=gamma.range,epsilon=eps),,cross=10)

  model <- tuneResult$best.model
  print(model)
  #plot(tuneResult,main="Parameter Search - Mean Squared Error")
}

# ANN
if(option==4){

  alldata=alldata[7:135,]
  maxs <- apply(alldata, 2, max)
  mins <- apply(alldata, 2, min)

  scaled <- as.data.frame(scale(alldata, center = mins, scale = maxs - mins)) # T-normalization Note: Y included.
  cv.error <- NULL
  k = ANN_cross_validation

  ANN_Par_Optim_Results=data.frame(matrix(0, ncol = 1, nrow = max_neuron))
  names(ANN_Par_Optim_Results)[1]<-"RMSE_Cross_Validation"

  n <- names(alldata)
  f <- as.formula(paste("Y ~", paste(n[!n %in% "Y"], collapse = " + ")))

  t=1 # Index for neuron number
  for(t in 1:max_neuron){
```

```
j=1 # Index for cross validation
for(j in 1:k){
  index <- sample(1:nrow(alldata),round(0.9*nrow(alldata))) # index for %90 of data
  train_cv <- scaled[index,]
  test_cv <- scaled[-index,]

  nn <- neuralnet(f,data=train_cv, algorithm = "rprop+",hidden=c(t),linear.output=T)

  pr_nn <- compute(nn,test_cv[,1:no_of_vars])
  pr_nn <- pr_nn$net.result*(max(alldata$Y)-min(alldata$Y))+min(alldata$Y)

  test_cv_r <- (test_cv$Y)*(max(alldata$Y)-min(alldata$Y))+min(alldata$Y)

  cv.error[j] <- sum((test_cv_r - pr_nn)^2)/nrow(test_cv) #MSE
}

ANN_Par_Optim_Results[t,1]=sqrt(mean(cv.error)) #RMSE

}

# Model building
opt_no_of_neurons=order(ANN_Par_Optim_Results[,1],decreasing=F)[1]

if(TrainSubset==1){
  datatrain_scaled=scaled[1:94,]
  datatest_scaled=scaled[95:129,]
}

if(TrainSubset==2){
  datatrain_scaled=scaled[85:94,]
  datatest_scaled=scaled[95:129,]
}

model=neuralnet(f,data=datatrain_scaled, algorithm = "rprop+",hidden=c(t),linear.output=T)

}
```

RESULTS AND PERFORMANCES

```

if(!PARAMETER_OPTIMIZATION==1){
  if(!option==4){y_train_est=predict(model,datatrain)}
  if(!option==4){y_test_est=predict(model,datatest)}

  if(option==4){
    y_train_est=compute(model,datatrain_scaled[,1:no_of_vars])
    y_train_est=y_train_est$net.result*(max(alldata$Y)-min(alldata$Y))+min(alldata$Y)

    y_test_est=compute(model,datatest_scaled[,1:no_of_vars])
    y_test_est=y_test_est$net.result*(max(alldata$Y)-min(alldata$Y))+min(alldata$Y)
  }

  error_train=datatrain$Y-y_train_est
  error_test=datatest$Y-y_test_est

  result_train=data.frame(datatrain$Y,y_train_est,error_train)
  result_test=data.frame(datatest$Y,y_test_est,error_test)
  names(result_train)=c("Real","Estimated","Error")
  names(result_test)=c("Real","Estimated","Error")

  # Strings
  string1=paste(a,"_",b,"_",var,"_Train Trend_",cix,".jpg",sep="")      # Train trend jpeg dosyasının adı
  string2=b                                                             # Train + Test trend y_label
  string3=paste(a,"_",b,"_",var,"_Test Trend_",cix,".jpg",sep="")      # Test trend jpeg dosyasının adı
  string4=paste(a,"_",b,"_",var,"_Train Error Histogram_",cix,".jpg",sep="") # Error histogram_Train dosyasının adı
  string5=paste(a,"_",b,"_",var,"_Test Error Histogram_",cix,".jpg",sep="") # Error histogram_Test dosyasının adı

  # Training Performance Plots
  jpeg(string1,quality = 100)
  plot(datatrain$Y,ylim=c(min(datatrain$Y)*0.75,max(datatrain$Y))*1.25,panel.first = grid(nx=20,ny=30),ylab = string2,main = "Train Performance" )
  points(y_train_est,col="red",pch=16)
  legend('bottomright', c('Real Value','Model Estimation') , pch=c(1,1), col=c('black', 'red'), bty='y', cex=1.0)
  dev.off()

```

```

# Test Performance Plots
jpeg(string3,quality = 100)
plot(datatest$Y,ylim=c(min(datatest$Y)*0.75,max(datatest$Y))*1.25,panel.first = grid(nx=20,ny=30),ylab = string2,main = "Test Performance" )
points(y_test_est,col="red",pch=16)
legend('bottomright', c('Real Value','Model Estimation') , pch=c(1,1), col=c('black', 'red'), bty='y', cex=1.0)
dev.off()

# Error Histogram Plots
jpeg(string4,quality = 100)
hist(error_train,main = string4)
dev.off()

jpeg(string5,quality = 100)
hist(error_test,main = string5)
dev.off()

# Performance Evaluations

RMSE_train <- sqrt(sum((y_train_est - datatrain$Y)^2)/nrow(datatrain))
RMSE_test <- sqrt(sum((y_test_est - datatest$Y)^2)/nrow(datatest))

MAPE_Train=mean(abs( (datatrain$Y - y_train_est)/datatrain$Y)*100 )

MAPE_Test=mean(abs( (datatest$Y - y_test_est)/datatest$Y)*100 )
if(!option==4){performance=data.frame(RMSE_train,RMSE_test,MAPE_Train,MAPE_Test)}
if(option==4){performance=data.frame(RMSE_train,RMSE_test,MAPE_Train,MAPE_Test,opt_no_of_neurons,min(ANN_Par_Optim_Results))}
#View(performance)

file <- paste(getwd(), "/",a,"_",b,"_",var,"_Model Results ",cix,".xlsx", sep="")
write.xlsx(result_train, file,sheetName="Train Results",append=TRUE)
write.xlsx(result_test, file,sheetName="Test Results",append=TRUE)
write.xlsx(performance, file,sheetName="Performance",append=TRUE)

if(option==4){write.xlsx(ANN_Par_Optim_Results, file,sheetName="ANN_Tuning Results",append=TRUE)}

if(option==2){
  write.xlsx(model$importance, file,sheetName="VarImp",append=TRUE)
}

if(option==3){
  gridsearch.results=tuneResult$performances

```

```
SVR.best.parameters=cbind.data.frame(tuneResult$best.parameters,model$tot.nSV,tuneResult$best.performance)

write.xlsx(gridsearch.results, file,sheetName="Parameter Search",append=TRUE)
write.xlsx(SVR.best.parameters, file,sheetName="Model",append=TRUE)
write.xlsx(myvars, file,sheetName="Variables",append=TRUE)
}

}
# beep()

} # For Loop sonu
#####

end=Sys.time() #proc.time() - ptm
Tuning.duration=end-start
cat("Training duration : ",Tuning.duration)

beep()
```

BIBLIOGRAPHY

A.J.Smola Regression estimation with support vector learning machines, Master's thesis [Report]. - [s.l.] : Technische Universitat München, 1996.

Aronszajn N. Theory of reproducing kernels [Journal] // Trans.Amer.Math.. - 1950.

Askarian Mahdiah [et al.] Data-based fault detection in chemical processes: Managing records with operator intervention and uncertain labels [Journal]. - [s.l.] : Elsevier, 2016.

ASTM Report on the development, construction, calculation and preparation of the ASTM [Report]. - Philadelphia : Institute of Petroleum, 1960.

Azadeh A., Ebrahimipour V. and Bavar P. A Fuzzy inference system for pump failure diagnosis to improve maintenance process [Journal]. - [s.l.] : Elsevier, 2010.

Bhartiya S. and Whiteley J.R. Development of inferential measurements using neural networks [Journal] // ISA Trans.. - 2001. - Vol. 40. - pp. 307-323.

Breiman Leo Random Forest [Journal] // Machine Learning. - 2001. - 45. - pp. 5-32.

Chang Ai-Fu, Pashikanti Kiran and Liu Y.A. Refinery Engineering: Integrated process modelling and optimization [Book]. - [s.l.] : Wiley, 2012.

Chen F.Z. and Wang X.Z. Software sensor design using Bayesian automatic classification and back-propagation neural networks [Journal] // Ind.Eng.Chem.Res.. - 1998. - pp. 3985-3991.

Dam M. and Saraf D.N. Design of neural networks using genetic algorithm for on-line property estimation of crude fractionator products [Journal] // Computer and Chemical Engineering. - 2006. - 30. - pp. 722-729.

Detienne Kristen Bell, Detienne David and A.Joshi Neural Networks as Statistical Tools for Business Researchers [Journal]. - 2003.

Enrique A.M. and Paz-Paredes Jose I. Artificial Neural Network Modeling Techniques Applied to the Hydrodesulfurization Process [Journal] // Journal of Mathematical and Computer Modelling. - 2009.

European Union emission regulations [Online]. - 2009. - <http://www.dieselnet.com/standards/eu/ld.html>.

Fletcher R. Practical Methods of Optimization (2nd Ed.) [Book]. - New York : Wiley-Interscience, 1987.

Fortuna L. [et al.] Soft analyzers for a sulfur recovery unit [Journal] // Contr.Eng.Pract.. - 2003. - Vol. 11. - pp. 1491-1500.

Gary J.H. and Handwerk G.E. Petroleum Refining Technology and Economics (2nd ed.) [Book]. - [s.l.] : Marcel Dekker Inc., 1984.

- Graf A.B.A., Smola A.J. and Borer S.** Classification in a Normalized Feature Space using Support Vector Machines [Journal] // IEEE Transactions on Neural Networks. - 2003. - 14. - pp. 597-605.
- Gunn. S.R.** Support vector machines for classification and regression [Report]. - [s.l.] : ISIS Technical Report, 1998.
- Hemptinne Jean-Charles de [et al.]** Select Thermodynamic Models for process simulation [Book]. - [s.l.] : Energies Nouvelles, 2011.
- Herbrich R. and Graepel T.** "A PAC-Bayesian margin bound for linear classifiers: Why SVMs work [Journal] // Advances in Neural Information Processing Systems. - 2001. - pp. 224-230.
- Hornik K., Stinchcombe M. and White H.** Multilayer Feedforward Neural Networks are Universal Approximators [Journal] // Neural Networks. - 1989. - pp. 359-366.
- Huber Peter** Robust estimation of a location parameter [Journal] // Annals of Mathematical Statistics. - 1964. - pp. 73-101.
- Jaimes Fabian [et al.]** Comparison between logistic regression and neural networks to predict death in patients with suspected sepsis in the emergency room [Journal] // Critical Care. - 2005.
- James Gareth [et al.]** An Introduction to Statistical Learning (2nd ed.) [Book]. - [s.l.] : Springer, 2009.
- Jechura John** Hydroprocessing: Hydrotreating & Hydrocracking - Chapters 7&9 [Online]. - November 14, 2016. - http://inside.mines.edu/~jjechura/Refining/08_Hydroprocessing.pdf.
- Jechura John** inside.mines.edu [Online]. - Colorado School of Mines, 2016. - http://inside.mines.edu/~jjechura/Refining/03_Crude_Units.pdf.
- Kaastra I. and Boyd M.** Designing a neural network for forecasting financial and economic time series [Journal] // Neurocomputing. - 1996. - 10. - pp. 215-236.
- Kadlec Petr, Gabrys Bogdan and Strandt Sibylle** Data-driven Soft Sensors in the Process Industry [Journal] // Computers and Chemical Engineering. - 2008.
- Kaneko H., Arakawa M. and Funatsu K.** Development of a new soft sensor method using independent component analysis and partial least squares [Journal] // AIChE. - 2009. - 55. - pp. 87-98.
- Karush W.** Minima of Functions of Several Variables with inequalities as Side conditions. - Chicago : Department of Mathematics, University of Chicago, 1939.
- Kresta J.V., Marlin T.E. and MacGregor J.F.** Development of Inferential process models using PLS [Journal] // Computer and Chemical Engineering. - 1994. - pp. 597-611.
- Lahiri S.K. and Ghanta K.C.** Hybrid Support Vector Regression and Genetic Algorithm Technique-A Novel Approach in Process Modeling [Journal] // Chemical Product and Process Modeling. - 2009.
- Leffler W.L.** Petroleum Refining for the Nontechnical Person(2nd Ed.) [Book]. - [s.l.] : PennWell Books., 1985.
- Li Yafen, Li Qi and Wang Huijuan** Soft Sensing Based on LS-SVM and Its Application to a Distillation Column [Journal] // Intelligent Systems Design and Applications. ISDA '06. - 2006.
- More R.K. [et al.]** Optimization of crude distillation system using aspen plus: Effect of binary feed selection on grass-root design [Journal] // Chemical Engineering Research and Design. - 2010.

Park S. and Han C. A nonlinear soft sensor based on multivariate smoothing procedure for quality estimation in distillation columns [Journal] // Computer and Chemical Engineering . - 2000. - Vol. 24. - pp. 871-877.

Parkash Surinder Refining Processes Handbook [Book]. - 2003.

Perry Robert H. and Green Don W. Perry's Chemical Engineers' Handbook(6th Ed.) [Book]. - [s.l.] : McGraw-Hill, 1984.

Pouladi B., Hassankiadeh M.N. and Behroozshad F. Dynamic simulation and optimization of an industrial-scale absorption tower for CO₂ capturing from ethane gas [Journal] // Energy Reports. - 2016.

R.J.Vanderbei LOQO User Manual Technical Report [Report]. - [s.l.] : Princeton University, 1997.

Raji O.Y. [et al.] Modelling and Optimization of Fluid Catalytic Cracking Unit (FCCU) Using Hysys [Journal] // International Journal of Emerging trends in Engineering and Development. - 2012.

Ramli N.M. [et al.] Composition Prediction of a Debutanizer Column using Equation Based ANN Model [Journal] // Neurocomputing. - 2014.

Ramli N.M. [et al.] Composition Prediction of a Debutanizer Column using Equation Based Artificial Neural Network Model [Journal] // Neurocomputing . - 2014. - 131. - pp. 59-76.

Riedmiller M. and Braun H. A Direct Adaptive Method for Faster Back-Propagation Learning [Journal] // Neural Networks. - 1993.

Sadighi Sepehr and Mohaddecy Reza Seif Predictive Modeling for an Industrial Naptha Reforming Plant using Artificial Neural Network with Recurrent Layers [Journal] // International Journal of Technology. - 2013. - 2 : Vol. Vol 4.

Samborskaya M.A. [et al.] Optimal fractionation of products of refining straight-run gasoline on zeolite catalyst with account of its deactivation [Journal] // Chemistry and Chemical Engineering in XXI century. - 2014.

Speight James G. The Chemistry and Technology of Petroleum (4th edition) [Book]. - CRC Press : [s.n.], 2006.

Tippett T.W. and Ward J.W. National Petroleum Refiners Association (NPRA) Annual Meeting. - 1978.

UKpia Statistical Data on the UK downstream industry [Online]. - http://www.ukpia.com/industry_information/industry-data.aspx.

Vapnik Vladimir and Cortes C. Support-Vector Networks [Journal] // Machine Learning. - 1995. - pp. 273-297.

Vapnik Vladimir The Nature of Statistical Learning Theory (2nd Ed.) [Book]. - New York : Springer-Verlag, 2000.

Vapnik Vladimir, Golowich S. and Smola A. SV method for multivariate density estimation [Journal] // Advances in neural information processing systems. - 1999. - Vol. Vol. 12. - pp. 659-665.

Wang D. and Feng X. Simulation and multi-objective optimization of an integrated process for hydrogen production from refinery off-gas [Journal] // International Journal of Hydrogen Energy. - 2013.

Wang D., Lui J. and Srinivasan R. Data-driven soft sensor approach for quality prediction in a refining process [Journal] // IEEE Trans.Ind.Inf.. - 2010. - 6. - pp. 11-17.

Yafen Li and Wang Huiqin RBF Neural Network Based Soft Measurement Model for Dryness of Wet Steam [Journal] // Process Automation Instrumentation. - 2003. - p. 24(9).

Yafen Li, Want Huihuan and Ningsheng M. Soft Sensing Based on LS-SVM and Its application to a distillation column [Conference] // Sixth International Conference on Intelligent Systems Design and Applications. - 2006.

Yan W., Shao X. and Wang X. Soft sensing modeling based on support vector machine and Bayesian model selection [Journal] // Computer and Chemical Engineering. - 2004. - 28. - pp. 1489-1498.

Yan X. Modified nonlinear generalized ridge regression and its application to develop naphtha cut point soft sensor [Journal] // Comput. Chem. Eng.. - 2008. - 32. - pp. 608-621.

Yao H. and Chu J. Operational optimization of a simulated atmospheric distillation column using support vector regression models and information analysis [Journal] // Chemical Engineering Research and Design. - 2012. - pp. 2247-2261.

Yu-Wei and Chiu Machine Learning with R Cookbook [Book]. - 2015.

Z.Kister Henry Can we believe the simulation results? [Article] // Cep Magazine. - 2002.

Zhou H. [et al.] Modeling and optimization of an industrial hydrocracking unit to improve the yield of diesel or kerosene [Journal] // Journal of fuel. - 2011.

VITA

Sadık Ödemiş graduated from Kadıköy İntaş Lisesi as highest ranked student in 2008. Then he obtained BS degree in Chemical Engineering at Boğaziçi University in 2013. He worked as a Process Modelling & Optimization engineer at Turkish Petroleum Refineries Corporation (Tüpraş). He lead or supported many projects regarding simulation, engineering design and optimization of refinery units. In 2015 he joined Process Control group at the same company and supported many project regarding advanced process control applications, alarm management studies, HMI Graphic Development, and so on. While working as in Tüpraş, he joined Koç University to pursue MSc. Degree in Industrial Engineering. He is currently a Process Control Supervisor at Tüpraş. He has co-authored a conference paper in IFAC'17 (under submission).