UOW MALAYSIA KDU
PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

SCHOOL OF COMPUTING & CREATIVE MEDIA

Please tick ✓ or click if using MS WORD

☐FOUNDATION ☐DIPLOMA ☒ DEGREE ☐ MASTER

Assignment Coversheet

Please complete all details required clearly. For softcopy submissions, please ensure this cover sheet is included at the start of your document or in the file folder.

Assignment & Course Details:

Subject Code: *(e.g. XCAT1234)* **XBDS2014N**		**Subject Name** *(e.g. Fundamentals of Computing)*: **Introduction to Data Science**	
Course *(e.g. Bachelor in Computing)* : Bachelor of Computer Science			
Lecturer Name: Ts. Dr. Law Foong Li			
Assessment Due Date: *(dd/mm/yy)*	19/4/2023	**Assessment Title:**	Group Assignment

I/We declare that:

- *This assignment is my/our own original work, except where I/we have appropriately cited the original source.*
- *This assignment or parts of it has not previously been submitted for assessment in this or any other subject.*
- *I/We allow the assessor of this assignment to test any work submitted by me/us, using text comparison software for plagiarism.*
  (For more information, Please read the Academic Integrity Guidelines)

Name : Tripti Pal Kaur **Student ID : 0132648** **Email : 0132648@kdu-online.com** **Mobile No: 0189860245** **Signature:** TPK **Date: 18/4/2023**	**Name : Wan Mohammed Adam** **Student ID: 0132601** **Email : 0132601@kdu-online.com** **Mobile No: 0109852593** **Signature:** WMA **Date:18/4/2023**	**Name :** **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**
Name : **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**	**Name :** **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**	**Name :** **Student ID:** **Email :** **Mobile No:** **Signature:** **Date:**

For office use only – Lecturer comments (if applicable)

Marks Breakdown

Table of Contents

1. **Introduction** 2
2. **Literature Review** 2
3. **Methodology** 4
   - **Data Collection** 4
   - **Data Pre-processing - Malaysian Dataset** 5
   - **Data Modelling** 11
   - **Data Pre-processing - US Dataset** 12
   - **Scaling the Data** 17
   - **Splitting into Training and Testing Set** 18
   - **Splitting the Training Data into Validation Set** 19
   - **Model Selection** 20
   - **Training the Random Forest Regression Model** 21
   - **Training the Linear Regression Model** 21
4. **Results** 22
5. **Discussion** 27
6. **Conclusion** 28
7. **References** 29

1. Introduction

Today's fuel market is overly complex due to many factors such as geopolitical factors where the supply and demand of fuel is affected by natural disasters, sanctions, wars and political instability. It is also complex due to the economic factors such as currency exchange rates, inflation, interest rates and Gross Domestic Product (GDP) growth. In addition to this, the market structure and technology also affects the convolution of the fuel market as due to the vast number of producers, distributors, refiners, retailers and consumers who all have different interests and strategies of their own. As for technology, due to the rapid growth and development of new technologies, this can disrupt the fuel market as it is able to create new market dynamics and challenges for existing individuals in the fuel market. For example, with the creation of fully electric cars that have a longer battery life and quicker charging times, this will disrupt the fuel market as they would struggle to adapt to the new market dynamics and would need to find a way to develop new types of fuel that can be better or be used in new electric vehicles.

Due to the complexity of the fuel market, it is important to be able to predict the fuel price for individuals and businesses. Hence, in this report, we will be predicting the fuel price using two different datasets and two different machine learning models. The first dataset focuses on the fuel prices for the last 7 years starting from 2017 and it is from the Open Department of Statistics Malaysia (OpenDOSM). The second dataset focuses on the fuel prices from the year 1995 to 2023 and it is from the US Energy Information Administration. With these datasets, two different supervised learning models will be used which are Linear Regression and Random Forest Regression to find out which model will give the best accuracy and performance. The values outputted by the algorithms will be compared with the real time values to see which model would provide the closest value to the real time value. The machine learning technique that is used is Time Series Analysis which is a technique that forecasts target values based on a known history of target values.

2. Literature Review

The use of machine learning has rapidly evolved over the years, with an increase of use in many different fields and industries such as natural language processing, fraud detection, recommendation systems, healthcare, cyber security, agriculture and many more. Machine learning models are commanly used to predict the price of fuel which is an important factor for decision making in the energy industry. This literature review will discuss the various machine learning models and techniques that have been used for fuel prediction. By looking at the different and existing literature reviews of machine learning models used for fuel price prediction, we can examine the gaps and address it in this report.

In a published paper by (Rathnajyothi et al., 2021), there were 3 supervised learning models used which were Random Forest Regression, Linear Regression and Decision Tree Regression. These models were used to understand which model produced the

best accuracy and performance in predicting the values of petroleum and Diesel. In their study, they found that Random Forest Regression gave the most accurate results and the errors were much lesser compared to other regression models.

In another paper published by Shaik, Subhani (2019), there were 5 supervised learning models used which were Support Vector Regression (Linear Model, RBF Model, Polynomial Model), Random Forest Regression and Linear Regression. In this study, they found that the Linear Regression algorithm provided the most accurate results/best fit in determining the price of fuel compared to the other models.

According to a study conducted by (Smith et al.), a correlation was made between random forest regression (RFR) and multiple linear regression (MLR) for prediction in the field of neuroscience. The study findings indicated that generally multiple linear regression is better than random forest regression in terms of predictive value and error. However, after applying both models to a set of data, it turns out that MLR was much better for prediction in neuroscience compared to RFR. The RSE values for MLR were lower compared to RFRs in all but two cases.

In a study by (Pahlavan-Rad et al.), they compared multiple linear regression and random forest models for predicting soil infiltration rates in a dry flood plain of eastern Iran. The study found that the random forest model predictions were more reliable with the expected distributions of the infiltration. The mean absolute error (MAE) was 10.5 for random forest and 10.9 for multiple linear regression model. They also found that the random forest model was much closer to reality than the multiple linear regression model.

In a different paper by (Zhang et al.), it was discovered that random forest outperformed multiple linear regression. This was evidenced by the lower mean error (ME), mean squared error (MSE), and root mean square error (RMSE) values, as well as a higher R-squared value.

In a separate study conducted by (Lee et al.), three models (Linear Regression, Support Vector Machine and Random Forest) were utilized to forecast the canopy nitrogen weight. The findings of this research showed that the Random Forest model outperformed all other models. Moreover, the authors concluded that interpreting the concepts and analysis of the Random Forest model was easier compared to the other models.

In another publication by (Jui et al.), they predicted the price of flat using linear regression and random forest regression models. The validity of the models was evaluated using residual analysis, boxplot analysis, cross-validation, and error checking. The study results showed that the random forest regression model outperformed the linear regression model in terms of prediction accuracy.

In (Čeh et al.) publication, they estimated the performance of random forest with multiple regression for predicting apartment prices. In their research, they use all performance measures such as R squared values, sales ratio, mean average percentage error (MAPE) and coefficient of dispersion (COD) to find out which model

is better. The research showed that random forest regression provided significantly better prediction results.

In a separate study conducted by (Pal et al.), a Random Forest model with 500 Decision Trees was developed to predict used car prices. The training accuracy of the model was found to be 95.82%, while the testing accuracy was 83.63%. By selecting the most highly correlated features, the model was able to accurately predict the prices of the cars.

In another publication by (Varshitha et al), they used supervised learning-based Artificial Neural Network model and Random Forest Machine Learning model. In the end, they found that the Random Forest model performed better with a mean absolute error value of 1.0970472 and R squared error value of 0.772584 which was lower than the errors produced by the other models.

The overall literature review suggests that Random Forest Regression gave the best accuracy and least number of errors compared to other machine learning models. In our research, we will find out which machine learning model is best in predicting the price of fuel. Since most of the literature review only tested with one dataset, we plan to test with two datasets to see if the outcome will be the same or not.

The research question for this study is to investigate the use of two different machine learning models, Linear Regression and Random Forest Regression, to predict the fuel prices of the two different datasets. The hypothesis for the research is Random Forest Regression is a better machine learning model in terms of accuracy and performance to predict the fuel price compared to Linear Regression. To accomplish this, we will use two different datasets and compare the final value with the actual value of the fuel price and the model that produces the closest value will be concluded as the model that has better accuracy and performance.

3. Methodology

Data Collection

The two datasets that we used in this study were from online websites as there are existing databases with loads of data that are currently available on the internet. The first dataset we found was from the Open Department of Statistics Malaysia (OpenDOSM) website where there were many different types of datasets. Since this study is to predict the price of fuel using two different machine learning models, our aim was to find datasets that had relevance to fuel prices and different types of fuel. On this OpenDOSM website, we managed to download the RON95 Petrol Price dataset which contained columns such as “Series_Type (Price in RM (level), or weekly change in RM (change_weekly)), Date (ranged from 2017-2023), the prices of RON95, RON97 and Diesel”. The data type for Series_Type was categorical, Date was date, and the prices were in float. After analysing the dataset, we realised that there were

not many features that can be used to analyse with the model. Hence, we decided to find another dataset that had more features.

The second dataset that we used in the study was obtained from a website called US Energy Information Administration (EIA). The dataset included columns such as “Date, All Grades Conventional, All Grades All Formulations, All Grades Formulated, Regular Conventional, Regular All Formulations, Regular Formulated, Midgrade All Formulations, Midgrade All Formulated, Midgrade Conventional, Premium Conventional, Premium All Formulations, Premium Reformulated, No 2 Diesel”. The data type for Date was date and the rest of the columns contained float values. After obtaining the datasets from online sources, we started the data preparation process which included data cleaning, data transformation, feature selection and splitting of the dataset to training and testing set.

Data Pre-processing - Malaysian Dataset

As for data pre-processing/data preparation, we started off with cleaning the Malaysian dataset. This was done by firstly importing the necessary libraries. The figures below show the libraries that were included to perform data preparation, plotting of graphs, splitting datasets, linear regression and cross validation, random forest regression and many more.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import cross_val_score
from seaborn import residplot

warnings.filterwarnings('ignore')
```

Figure 3.0.1: Libraries imported for Linear Regression Model (Malaysian Dataset)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import learning_curve
from sklearn.tree import plot_tree
from sklearn.model_selection import cross_validate

%matplotlib inline
```

Figure 3.0.2: Libraries imported for Random Forest Regression Model (Malaysian Dataset)

After importing the libraries, we loaded the data from a CSV file called ‘fuelprice.csv. This data was read using the pandas library in Python and the “delimiter= ‘,’” parameter means that the data in the CSV file is separated by commas. The figure below shows the code.

```
#Loading data
df = pd.read_csv('C://Users/tript/Desktop/fuelprice.csv', delimiter=',')
```

Figure 3.0.3: Loading of data from CSV file

Next, we checked if the data had successfully been loaded by using various pandas functions such as head, tail, shape, describe and info. The pandas function df.head() displayed the first 5 rows of the dataset, df.tail() displayed the last 5 rows of the dataset, df.shape returned the number of rows and columns in the dataset, df.describe() is a method that provided the summary statistically of the count, mean, standard deviation, minimum, maximum of the numerical columns. As for the df.info() function, it provided information about the dataset such as number of entries, number of columns, data types, memory size and the number of missing values. The reason why we used all these functions was because we firstly wanted to check if the dataset had been loaded correctly, to identify outliers and anomalies using the df.describe function(), to identify the missing values using df.info() function and mainly understand the structure of the dataset which is crucial during data pre-processing.

As for the Malaysian dataset, the results when using head, tail, shape, info and describe pandas function is as displayed by the figures below.

```
df.head()
```

	series_type	date	ron95	ron97	diesel
0	level	30-03-17	2.13	2.41	2.11
1	level	06-04-17	2.16	2.43	2.08
2	level	13-04-17	2.24	2.52	2.16
3	level	20-04-17	2.27	2.54	2.21
4	level	27-04-17	2.21	2.49	2.14

```
df.tail()
```

	series_type	date	ron95	ron97	diesel
574	NaN	NaN	NaN	NaN	NaN
575	NaN	NaN	NaN	NaN	NaN
576	NaN	NaN	NaN	NaN	NaN
577	NaN	NaN	NaN	NaN	NaN
578	NaN	NaN	NaN	NaN	NaN

```
df.shape
```

```
(579, 5)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 579 entries, 0 to 578
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   series_type  290 non-null    object
 1   date         290 non-null    object
 2   ron95        290 non-null    float64
 3   ron97        290 non-null    float64
 4   diesel       290 non-null    float64
dtypes: float64(3), object(2)
memory usage: 22.7+ KB
```

```
df.describe()
```

	ron95	ron97	diesel
count	290.000000	290.000000	290.000000
mean	2.014552	2.724724	2.086690
std	0.207236	0.703883	0.172751
min	1.250000	1.550000	1.400000
25%	2.050000	2.365000	2.102500
50%	2.050000	2.560000	2.150000
75%	2.080000	2.800000	2.180000
max	2.380000	4.840000	2.340000

Figure 3.0.4: Pandas Functions

By analysing the figures above, we can see that the shape of the dataset was 579 rows and 5 columns. As for the info pandas function, it also showed the rows/entries which were 579 and 5 columns which were “series_type”, “date”, “ron95”, “ron97” and “diesel”. We can also see that there were 290 missing values/ non-null and data types were float and object. As for the df.describe(), it provided the statistics of the numerical columns which are “ron95, ron97 and diesel”. The count for all 3 columns was 290. The mean for ron95 was 2.014552, ron97 2.724724 and diesel 2.086690. The standard deviation for ron95 was 0.207236, ron97 0.703883 and diesel 0.172751. The minimum price for ron95 was 1.25 and the maximum price was 2.38. The minimum price for ron97 was 1.55 and the maximum price was 4.84. The minimum price for diesel was 1.40 and the maximum was 2.34. The reason why we used the describe method is to obtain the statistics of the overall dataset and to get a sense of the distribution and range of values in each column.

Next for the Malaysian dataset, we checked for missing values using a python code snippet which checks for the number of missing/ null values. The figure below shows the code.

```
# Check for missing values
print(df.isnull().sum())
```

```
series_type    289
date           289
ron95          289
ron97          289
diesel         289
dtype: int64
```

Figure 3.0.5: Checking for missing values

The ‘isnull()’ method returns the dataset with boolean values which indicates whether each element in the dataset is null or not. The ‘sum()’ method gives us the number of missing values in each column. As seen in the figure above, there are 289 null values in all the columns (series_type, date, ron95, ron97 and diesel). Since we were predicting the price of fuel, we decided to drop 2 columns which were series_type and date. The reason why we dropped these columns was because it was not important to be included. After dropping the columns that we did not need, we filled the missing/null values with the mean of each column. The figure below shows how it was done.

```
# Fill missing values with the mean value
df = df.fillna(df.mean())
```

```
# Check for missing values again
print(df.isnull().sum())
```

```
ron95     0
ron97     0
diesel    0
dtype: int64
```

Figure 3.0.6: Filling missing values and checking for missing values

After filling the missing values with the mean, we checked back for missing values and as seen in the figure above, there were no more missing/null values. The reason why we filled the missing values with the mean of each column was because we wanted to maintain the central tendency of the dataset which helped to prevent any significant distortions in the data distribution as well as any bias in the analysis.

For linear regression using the Malaysian dataset, the dataset had completed the preparation phase and was ready to be split into training and test set. The figure below shows how the data was split before building the linear regression model.

```
# Splitting the dataset into training and test sets
X = df.drop(['ron95'], axis=1)
y = df['ron95']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 3.0.7: Splitting the dataset

The function train_test_split is a part of scikit-learn, and it allows us to split the data randomly into two subsets: a training set and a test set. The test_size parameter was specified as 0.2, meaning that 20% of the data was set aside for testing, while the remaining 80% was reserved for training. To ensure that the same random split is generated every time the code is run, the random_state parameter was set to 42. The primary reason for partitioning the data into training and test sets was to assess the model's ability to generalize to new and unseen data.

Moving on to the random forest regression using the Malaysian dataset, before filling the missing/null values with the mean, there were a number of steps that were performed. Firstly, we calculated the correlation between all pairs of variables and the results showed how strongly each variable was related to one another. The figure below shows the code and result.

```
df.corr()
```

	ron95	ron97	diesel
ron95	1.000000	0.405414	0.929240
ron97	0.405414	1.000000	0.500243
diesel	0.929240	0.500243	1.000000

Figure 3.0.8: Calculation correlation

The diagonal shows the correlation of each of the columns with itself which is always 1. There was also a strong positive correlation of 0.929240 between ron95 and diesel which means that if one price increases, the other price would also increase. Similarly, there was a positive correlation of 0.405414 between ron95 and ron97 but this correlation was not as strong as the one between ron95 and diesel.

Moreover, after calculating the correlation of each of the columns, we plotted a heatmap which allowed us to visualise the relationship of the columns. The figure below shows the heatmap.

```
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))

#plot heat map
```

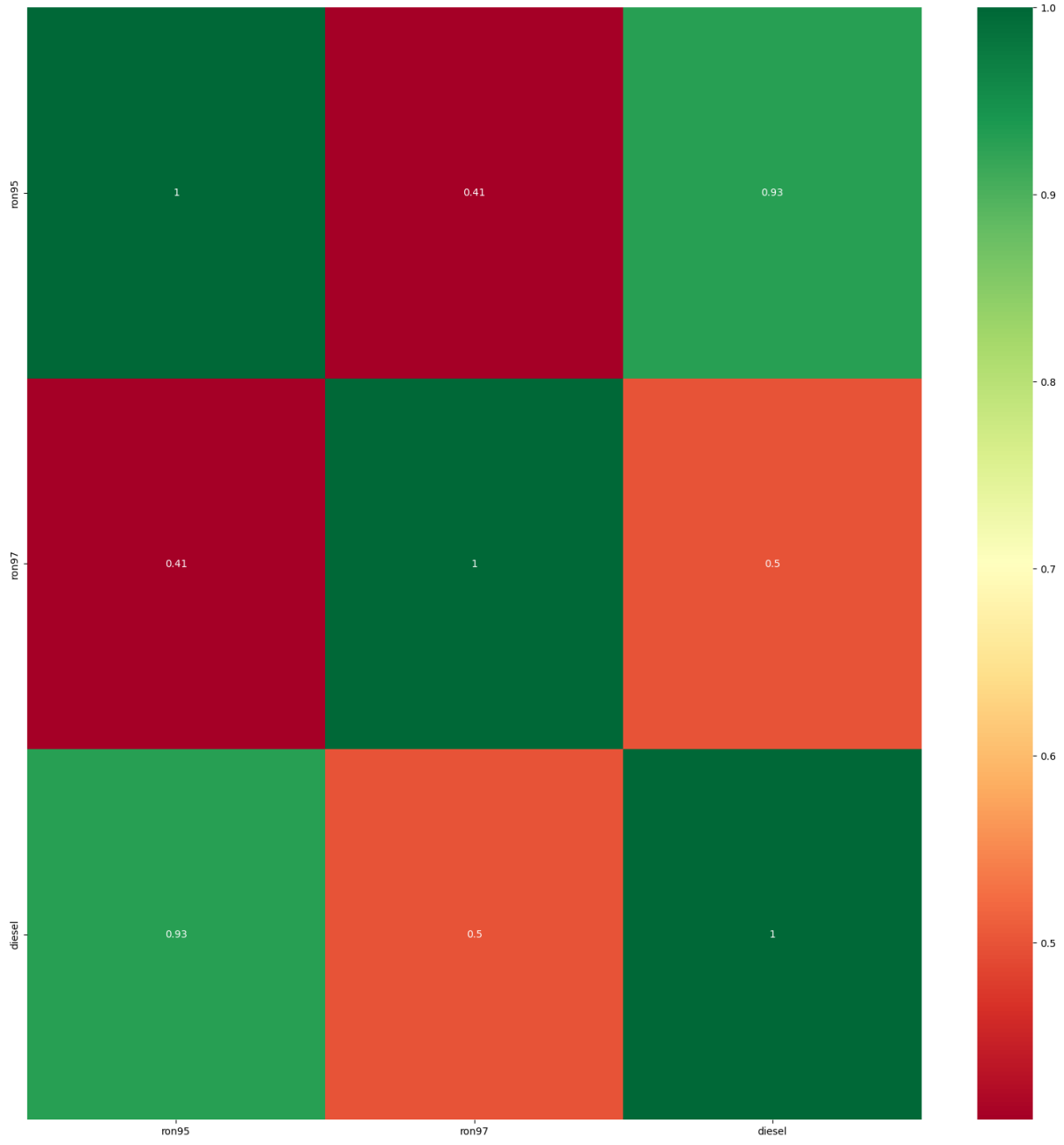

Figure 3.0.9: Plotting of Heatmap

Furthermore, we checked for outliers that this dataset had. We did this using a boxplot which displayed the distribution of values for each column which includes the median, interquartile range and outliers. The figure below shows the code and result.

```
sns.boxplot(data=df)
plt.show()
```

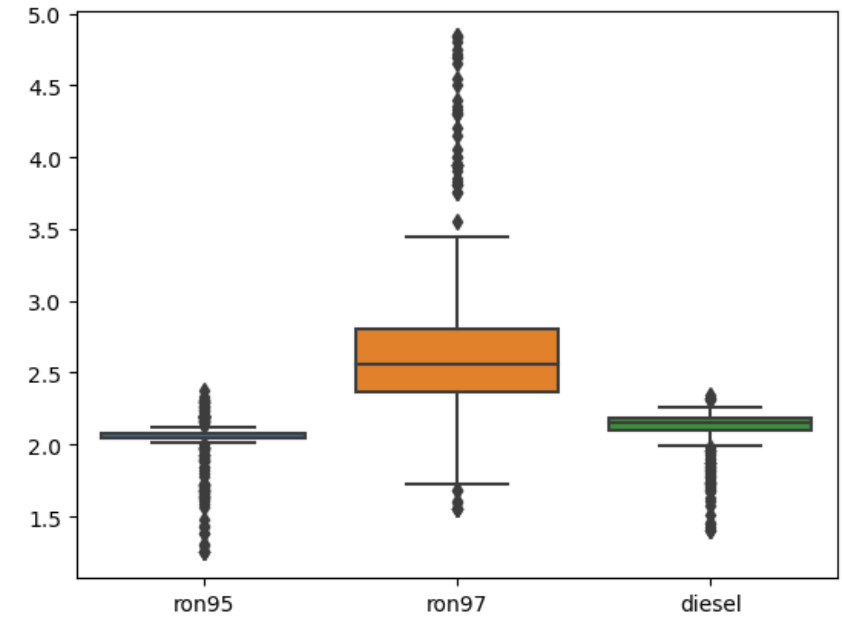

Figure 3.0.10: Plotting of Boxplot

The figure above shows that there were too many outliers which in the end would impact the accuracy of the analysis and the conclusion of this study. Hence, we realised that this dataset was not appropriate for our study and we decided to find another dataset before starting with the modelling process.

Data Modelling

The reason why we chose linear regression and random forest regression models to predict the price of fuel was because linear regression is a common statistical method for modelling the relationship between the independent variables and dependent variables. Linear regression is usually used in mathematical research methods where the prediction needs to be measured and model them against multiple input variables (Maulud and Abdulazeez). We also chose this model because it is a simple and interpretable model which can provide information about the relationship between the variables. Another advantage for linear regression is the linearity which makes the estimation procedure simple (Christoph Molnar).

Random forest regression was chosen as part of this study to predict the price of fuel due to its ability to provide accurate predictions and its comprehensibility. Furthermore, compared to the decision tree algorithm, random forest regression is more accurate (Mbaabu). Additionally, this model can effectively deal with non-linear relationships between the dependent and independent variables. It is known to be a type of

ensemble learning method in which multiple decision trees are built on different subsets of the training data and the final result of the final prediction is resulted from averaging the results of the individual trees. This model can also handle missing values and outliers much better than other machine learning models.

After analysing the Malaysian dataset, we explored another dataset to improve our fuel price predictions. The main reason for this decision was that the Malaysian dataset had too many null or missing values, making it less reliable for building accurate models. Additionally, the dataset needed more features to understand the factors influencing fuel prices comprehensively.

Consequently, we chose to work with the US dataset, which offered a more extensive set of features and a more extended period, covering fuel prices from 1995 to 2021. The US dataset, obtained from the US Energy Information Administration, allowed us to dive deeper into the factors affecting fuel prices and provided a better opportunity to build more accurate predictive models.

Data Pre-processing - US Dataset

Like the Malaysian dataset, we started the data preparation process for the US dataset, which included data cleaning, data transformation, feature selection, and splitting the dataset into training and testing sets. We performed similar steps to load, inspect, and pre-process the US dataset as we did for the Malaysian dataset.

To begin with, it is important to import the necessary libraries for data manipulation, visualisation, and building the Random Forest Regression model. These libraries include NumPy, pandas, seaborn, RandomForestRegressor, train_test_split, and various metrics for model evaluation. Like the Malaysian dataset, we started the data preparation process for the US dataset, which included data cleaning, data transformation, feature selection, and splitting the dataset into training and testing sets. We performed similar steps to load, inspect, and pre-process the US dataset as we did for the Malaysian dataset.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import RobustScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import  LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import warnings
warnings.filterwarnings('ignore')
```

Figure 3.0.11: Import necessary libraries

Next, the dataset 'pet-diesel-US.csv' is loaded into a Pandas DataFrame using the 'read_csv' function. The first three rows of the dataset are displayed using 'head(3)'.

```
# Load the dataset
df = pd.read_csv('pet-diesel-US.csv')
df.head(3)
```

	Date	A1	A2	A3	R1	R2	R3	M1	M2	M3	P1	P2	P3	D1
0	01/02/1995	1.127	1.104	1.231	1.079	1.063	1.167	1.170	1.159	1.298	1.272	1.250	1.386	1.104
1	01/09/1995	1.134	1.111	1.232	1.086	1.070	1.169	1.177	1.164	1.300	1.279	1.256	1.387	1.102
2	01/16/1995	1.126	1.102	1.231	1.078	1.062	1.169	1.168	1.155	1.299	1.271	1.249	1.385	1.100

Figure 3.0.12: Load the dataset

Then, we use the df.info() function to print the information about the dataframe, including the data types of each column, column names, the number of non-null values in each column, and memory usage. This information helps understand the dataset's structure and identify any issues with the data types of missing values.

```
# Check the dataset's shape (number of rows and columns)
df.shape
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1361 entries, 0 to 1360
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    1361 non-null   object 
 1   A1      1361 non-null   float64
 2   A2      1361 non-null   float64
 3   A3      1361 non-null   float64
 4   R1      1361 non-null   float64
 5   R2      1361 non-null   float64
 6   R3      1361 non-null   float64
 7   M1      1361 non-null   float64
 8   M2      1361 non-null   float64
 9   M3      1361 non-null   float64
 10  P1      1361 non-null   float64
 11  P2      1361 non-null   float64
 12  P3      1361 non-null   float64
 13  D1      1361 non-null   float64
dtypes: float64(13), object(1)
memory usage: 149.0+ KB
```

Figure 3.0.13: Check the dataset's shape and info

In the provided dataset, there are a total of 1361 observations and 14 variables. The variables consist of one 'Date' column and 13 numerical columns representing various gasoline and diesel price categories.

Specifically, the data types for the columns are as follows: the 'Date' column is of object data type, typically referring to string representations of dates, while the remaining 13 columns are of float64 data type, indicating numerical data.

A preliminary analysis of the dataset reveals that there are all the values, as each column contains 1361 non-null entries. This suggests that the dataset is well-structured and clean, requiring minimal pre-processing before utilising it for further analysis or modelling purposes.

Next, we check for missing values by applying the isna().sum() function which calculates the total number of missing values for each column. This helps identify any columns with missing data that may need to be addressed during pre-processing.

```
# Check for missing values
df.isna().sum()
```

```
Date    0
A1      0
A2      0
A3      0
R1      0
R2      0
R3      0
M1      0
M2      0
M3      0
P1      0
P2      0
P3      0
D1      0
dtype: int64
```

Figure 3.0.14: Check for missing values

The provided output displays the count of missing values for each column in the dataset. In this case, every column has a count of 0, indicating that no missing values are present in any of the columns. This is an important step in the data pre-processing phase, as missing values can negatively impact the performance of various machine learning models and statistical analyses.

Having a dataset with no missing values simplifies the pre-processing stage and ensures that the dataset is ready for further exploration and analysis without the need for imputation or removal of incomplete observations.

```
# Summary of the dataset
print(df.describe())
```

	A1	A2	A3	R1	R2
count	1361.000000	1361.000000	1361.000000	1361.000000	1361.000000
mean	2.285680	2.234511	2.396873	2.225170	2.178511
std	0.859028	0.843815	0.883311	0.850143	0.835549
min	0.949000	0.926000	1.039000	0.907000	0.885000
25%	1.461000	1.433000	1.550000	1.421000	1.393000
50%	2.326000	2.251000	2.458000	2.237000	2.175000
75%	2.903000	2.825000	3.060000	2.828000	2.765000
max	4.165000	4.102000	4.301000	4.114000	4.054000

	R3	M1	M2	M3	P1
count	1361.000000	1361.000000	1361.000000	1361.000000	1361.000000
mean	2.329126	2.382822	2.320970	2.508877	2.519840
std	0.876739	0.882107	0.858521	0.908861	0.911055
min	0.974000	1.008000	0.979000	1.112000	1.100000
25%	1.489000	1.517000	1.482000	1.616000	1.607000
50%	2.367000	2.481000	2.404000	2.627000	2.693000
75%	2.976000	3.033000	2.930000	3.206000	3.209000
max	4.247000	4.229000	4.153000	4.387000	4.344000

	P2	P3	D1
count	1361.000000	1361.000000	1361.000000
mean	2.472096	2.609244	2.404699
std	0.894472	0.925587	0.998646
min	1.074000	1.191000	0.953000
25%	1.573000	1.695000	1.418000
50%	2.640000	2.769000	2.479000
75%	3.127000	3.318000	3.070000
max	4.283000	4.459000	4.764000

Figure 3.0.15: Summary of dataset

The df.describe() method provides descriptive statistics of the given data frame, including each column's count, mean, standard deviation, minimum, maximum, and quartile values. Based on the provided output, the DataFrame has 1361 rows and 13 columns, with column names A1, A2, A3, R1, R2, R3, M1, M2, M3, P1, P2, P3, and D1.

For each column, the following statistics are provided:

- count: indicating the number of non-null values that are in the column
- mean: providing the average value of the column
- std: representing the standard deviation of the column
- min: indicating the minimum value of the column
- 25%: gives the 25th percentile value of the column
- 50%: gives the median (50th percentile) value of the column
- 75%: gives the 75th percentile value of the column
- max: indicating the maximum value of the column

These statistics can be used to understand the data distribution and identify potential outliers or data issues.

Based on the output, we are able to describe a summary of each column:

- A1: The column has 1361 non-null values, with a mean of 2.285680, a standard deviation of 0.859028, a minimum value of 0.949000, and a maximum value of 4.165000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.461000, 2.326000, and 2.903000, respectively.

- A2: The column has 1361 non-null values, with a mean of 2.234511, a standard deviation of 0.843815, a minimum value of 0.926000, and a maximum value of 4.102000. The values for the 25th percentile, median (50th percentile), and 75th percentile are 1.433000, 2.251000, and 2.825000, respectively.

- A3: The column has 1361 non-null values, with a mean of 2.396873, a standard deviation of 0.883311, a minimum value of 1.039000, and a maximum value of 4.301000. The values corresponding to the 25th percentile, median (50th percentile), and 75th percentile are 1.550000, 2.458000, and 3.060000, respectively.

- R1: The column has 1361 non-null values, with a mean of 2.225170, a standard deviation of 0.850143, a minimum value of 0.907000, and a maximum value of 4.114000. The values corresponding to the 25th percentile, median (50th percentile), and 75th percentile are 1.421000, 2.237000, and 2.828000, respectively.

- R2: The column has 1361 non-null values, with a mean of 2.178511, a standard deviation of 0.835549, a minimum value of 0.885000, and a maximum value of 4.054000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.393000, 2.175000, and 2.765000, respectively.

- R3: The column has 1361 non-null values, with a mean of 2.329126, a standard deviation of 0.876739, a minimum value of 0.974000, and a maximum value of 4.247000. The values for the 25th percentile, median (50th percentile), and 75th percentile are 1.489000, 2.367000, and 2.976000, respectively.

- M1: The column has 1361 non-null values, with a mean of 2.382822, a standard deviation of 0.882107, a minimum value of 1.008000, and a maximum value of 4.229000. The dataset has values for the 25th percentile, median (50th percentile), and 75th percentile, which are 1.517000, 2.481000, and 3.033000, respectively.

- M2: The column has 1361 non-null values, with a mean of 2.320970, a standard deviation of 0.858521, a minimum value of 0.979000, and a maximum value of 4.153000. The values corresponding to the 25th percentile, median (50th percentile), and 75th percentile are 1.482000, 2.404000, and 2.930000, respectively.

- M3: The column has 1361 non-null values, with a mean of 2.508877, a standard deviation of 0.908861, a minimum value of 1.112000, and a maximum value of 4.387000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.616000, 2.627000, and 3.206000, respectively.

- P1: The column has 1361 non-null values, with a mean of 2.519840, a standard deviation of 0.911055, a minimum value of 1.100000, and a maximum value of 4.344000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.607000, 2.693000, and 3.209000, respectively.

- P2: The column has 1361 non-null values, with a mean of 2.472096, a standard deviation of 0.894472, a minimum value of 1.074000, and a maximum value of 4.283000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.573000, 2.640000, and 3.127000, respectively.

- P3: The column has 1361 non-null values, with a mean of 2.609244, a standard deviation of 0.925587, a minimum value of 1.191000, and a maximum value of 4.459000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.695000, 2.769000, and 3.318000, respectively.

- D1: The column has 1361 non-null values, with a mean of 2.404699, a standard deviation of 0.998646, a minimum value of 0.953000, and a maximum value of 4.764000. The values of the 25th percentile, median (50th percentile), and 75th percentile are 1.418000, 2.479000, and 3.070000, respectively.

In summary, columns A1 to D1 have similar ranges of values but different means and standard deviations. Some columns have minimum and maximum values that differ from the mean, indicating potential outliers or non-normal distribution. Additionally, the quartile values can be used to get a sense of the spread and skewness of the data.

```
: # Outlier detection using boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(data=df)
plt.title('Boxplot for Outlier Detection')
plt.show()
```

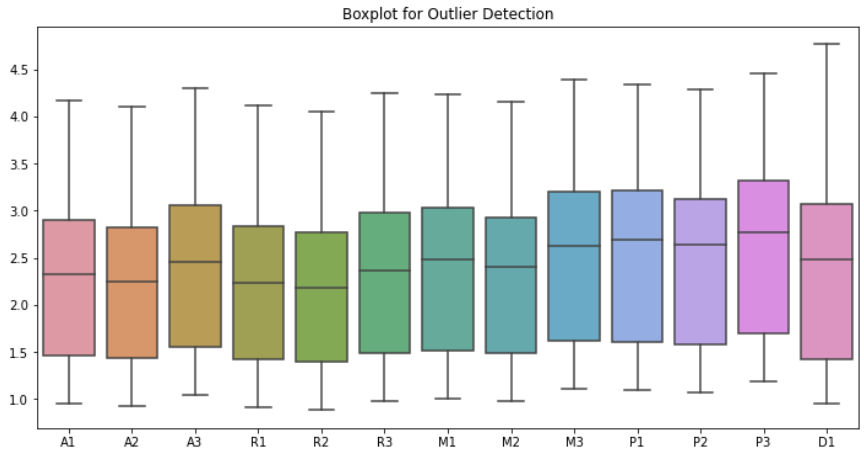

Figure 3.0.16: Detecting outliers using boxplot

Based on the boxplot analysis of the dataset containing columns A1 to D1, these columns have no outliers. The boxplots for each column show that the median values (indicated by the horizontal line within each box) are near the centre of each distribution, and the interquartile ranges (indicated by the height of each box) are relatively consistent across the columns.

Additionally, the whiskers of the boxplot (which extend to the minimum and maximum values that are not considered outliers) do not deviate significantly from the overall distribution of values for each column. This suggests that the data is relatively normally distributed and contains no extreme values considered outliers.

Scaling the Data

```
robust = RobustScaler()
minmax = MinMaxScaler()
```

```
for col in data.columns:
  data[col] = robust.fit_transform(data[col].values.reshape(-1,1))
  data[col] = minmax.fit_transform(data[col].values.reshape(-1,1))
```

Figure 3.0.17: Robust Scaling and Min-Max Scaling

Two essential pre-processing techniques in data science, Robust Scaling, and Min-Max Scaling, are employed. These techniques ensure that the data is appropriately scaled and standardized before being utilized for further analysis or modelling. The code initiates the process by instantiating two scaler objects from the widely-used Python library, scikit-learn.

The Robust Scaler (represented by the 'robust' variable) is a popular method for mitigating the influence of outliers in the dataset. It does this by employing the Interquartile Range (IQR), which effectively scales the data by subtracting the median and dividing by the IQR. This scaling method is robust to outliers, hence the name, and is particularly useful when the dataset contains significantly skewed features.

On the other hand, the Min-Max Scaler (represented by the 'minmax' variable) aims to normalise the data by transforming the features into a specified range, typically [0, 1]. This is achieved by subtracting the minimum value of each feature and dividing it by the range of its values. The Min-Max Scaling technique is advantageous when the data needs to be transformed into a specific range, or its distribution is not Gaussian.

Applying these pre-processing techniques is essential in data science, as they improve the performance and accuracy of machine learning models. By implementing these scaling methods, researchers and data scientists can ensure that their analyses and models are based on standardised and well-prepared data, ultimately leading to more reliable and robust results.

Splitting into Training and Testing Set

```
X, y = data.drop('D1', axis=1), data['D1']
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=56)
```

Figure 3.0.18: Splitting into Training and Testing Set

The dataset is divided into input features (X) and the target variable (y) and then further divided into training and testing sets. This step aims to create subsets of data for training and evaluating machine learning models, which helps assess their performance and generalisation capabilities.

The input features, represented by 'X,' are obtained by dropping the target variable 'D1', and 'No 2 Diesel' from the dataset. The 'data. drop()' function is called with the parameter 'axis=1' to indicate that a column should be dropped. In contrast, the target variable 'D1' is assigned to 'y'.

The scikit-learn library's 'train_test_split()' function is used to split the data into training and testing sets after separating the input features and the target variable. The data is split into 80% for training and 20% for testing, which is determined by the 'test_size' parameter set to 0.2. The 'random_state' parameter is set to 56 to ensure the reproducibility of the results. The resulting subsets, 'x_train' and 'x_test,' contain the

input features for the training and testing sets, respectively. Similarly, 'y_train' and 'y_test' contain the corresponding target values for the training and testing sets. These subsets will be used in subsequent steps to train, validate, and evaluate the performance of the selected machine-learning models.

Splitting the Training Data into Validation Set

```
x_train_, x_val, y_train_, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=56)
```

Figure 3.0.19: Splitting into Training into validation set

The training data is further divided into two subsets: a smaller training set (x_train_, y_train_) and a validation set (x_val, y_val). This process, known as creating a validation set, is an essential step in machine learning workflows as it helps to fine-tune and select the best-performing model based on its performance on unseen data during training.

To create a validation set, the 'train_test_split()' function from scikit-learn is used again, this time to split the 'x_train' and 'y_train' data into new subsets. The 'test_size' parameter is set to 0.2, indicating that 20% of the training data will be allocated to the validation set, while the remaining 80% will be used for training. By setting the 'random_state' parameter to 56, the results are reproducible, with the data consistently split in the same way for every run.

Using a validation set makes it possible to assess a model's performance on data that it has not been trained on, which helps identify potential overfitting or underfitting issues. The validation set is a proxy for the test set during model development, allowing for model selection and hyperparameter tuning without overusing the actual test set. This approach helps improve the model's generalisation capabilities, ensuring it performs well on new, unseen data.

Model Selection

```
def model_selection(x_train_, x_val, y_train_, y_val, model):
  model = model()
  model.fit(x_train_, y_train_)

  pred = model.predict(x_val)

  error = np.sqrt(mean_squared_error(y_val, pred))
  mae = mean_absolute_error(y_val, pred)
  mse = mean_squared_error(y_val, pred)
  rmse = np.sqrt(mse)
  acc = r2_score(y_val, pred)
  train_score = model.score(x_train_, y_train_)
  val_score = model.score(x_val, y_val)

  print('Error:', error*100)
  print('MAE:', mae*100)
  print('MSE:', mse*100)
  print('RMSE:', rmse*100)
  print('ACC:', acc)
  print('Train Score:', train_score)
  print('Val Score:', val_score*100)
  print('Is overfitting:', True if train_score>val_score else False)
  print('Overfitting by:',train_score*100-val_score*100)

  return error, mae, mse, rmse, acc, train_score, val_score
```

Figure 3.0.20: Model Selection Function

The model_selection function is defined to evaluate different machine learning models by fitting them to the training data and predicting the target variable for the validation data. The function takes the following inputs: x_train_, x_val, y_train_, y_val, and model. The first four inputs represent the training and validation data. At the same time, the model parameter refers to the machine learning model that will be used, such as RandomForestRegressor or LinearRegression.

The model() is instantiated and trained on the x_train_ and y_train_ subsets using model.fit() method inside the function. Once trained, the function predicts the values on the validation set x_val using model.predict() method.

To assess the model's performance on the validation data, several performance metrics are computed, such as mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and R2 score (ACC). Furthermore, the function also calculates and displays the training and validation scores of the model, which are determined using the model.score() method. These scores help identify overfitting, which refers to the model's tendency to perform better on the training data than the validation data.

The function also calculates the difference between the training and validation scores to quantify the degree of overfitting. Finally, the performance metrics are returned as a tuple, allowing for easy comparison between different models and facilitating model selection based on the desired evaluation criteria.

Training the Random Forest Regression Model

```
randomforest = model_selection(x_train_, x_val, y_train_, y_val, RandomForestRegressor)
randomforest
```

Figure 3.0.21: Model Selection Function with RandomForestRegressor Model

In this code snippet, the model_selection function is called with the smaller training and validation sets (x_train_, x_val, y_train_, and y_val) and the RandomForestRegressor model from the scikit-learn library. This step is intended to assess how well the RandomForestRegressor model performed on the provided data.

RandomForestRegressor is an ensemble learning technique that creates numerous decision trees during the training phase and merges their predictions to generate a more precise and robust output. This model is recognized for its ability to manage intricate associations between features and the target variable, as well as its resilience to overfitting.

By calling the model_selection function with the RandomForestRegressor model, the model is trained on the smaller training set and evaluated on the validation set. The performance metrics, including error, MAE, MSE, RMSE, R2 score, train score, and validation score, are calculated and printed to provide insight into the model's performance.

The function returns these performance metrics as a tuple assigned to the variable randomforest. This allows for easy comparison of the RandomForestRegressor model's performance against other models, helping to inform the model selection process.

Training the Linear Regression Model

```
linear = model_selection(x_train_, x_val, y_train_, y_val, LinearRegression)
linear
```

Figure 3.0.22: Model Selection Function with LinearRegression Model

Next, the model_selection function is called again, this time with the LinearRegression model from the scikit-learn library. This step aims to evaluate the performance of the Linear Regression model on the given data.

LinearRegression is a popular and straightforward machine learning model that seeks to establish a linear connection between the input features and the target variable. It typically performs well when the relationship between the features and the target variable is fairly linear. However, it may struggle to handle complex, nonlinear relationships.

By calling the model_selection function with the Linear Regression model, the model is trained on the smaller training set and evaluated on the validation set. The performance metrics, including error, MAE, MSE, RMSE, R2 score, train score, and

validation score, are calculated and printed to provide insight into the model's performance.

The function returns these performance metrics as a tuple assigned to the variable linear. This allows for easy comparison of the Linear Regression model's performance against other models, such as the RandomForestRegressor, to inform the model selection process based on the desired evaluation criteria.

4. Results

This study aimed to analyse the historical gasoline and diesel prices in the United States and develop predictive models to forecast future gasoline prices. The dataset used in this study contains weekly gasoline and diesel prices in the U.S. from several categories, including different grades and formulations. The data was carefully cleaned and processed, and various visualisations were generated better to understand the trends and price distribution across the dataset.

During the exploratory data analysis phase, summary statistics were generated for each numerical column, revealing critical insights into the dataset's central tendencies and dispersion measures. Correlation analysis was also performed, resulting in a heatmap visualisation to identify relationships between the various price categories. This information was crucial in selecting the most relevant predictive model features.

A subset of the dataset was used to train two regression models for forecasting weekly U.S. All Grades All Formulations Retail Gasoline Prices (A1): the Random Forest Regression model and the Linear Regression model. Performance evaluation was carried out based on mean absolute error (MAE), mean squared error (MSE), and R-squared (R^2) values. Additionally, feature importances and coefficients of the models were analysed to determine the most significant predictors of gasoline price.

```
#Evaluation Metrics
randomforest_metrics = model_selection(x_train_, x_val, y_train_, y_val, RandomForestRegressor)
print('Random Forest Regression Metrics')
print('RMSE:', round(randomforest_metrics[3], 4))
print('MAE:', round(randomforest_metrics[1], 4))
print('MSE:', round(randomforest_metrics[2], 4))
print('R2:', round(randomforest_metrics[4], 4))
```

```
Error: 3.91545867056573
MAE: 2.2197465569247847
MSE: 0.15330816600908356
RMSE: 3.91545867056573
ACC: 0.9797282337993366
Train Score: 0.9972794869343529
Val Score: 97.97282337993366
Is overfitting: True
Overfitting by: 1.7551253135016225
Random Forest Regression Metrics
RMSE: 0.0392
MAE: 0.0222
MSE: 0.0015
R2: 0.9797
```

Figure 3.0.23: Evaluation Metrics for randomforest_metrics

The output displayed the evaluation metrics for the RandomForestRegressor model. The metrics included the root mean squared error (RMSE), mean absolute error (MAE), mean squared error (MSE), and R2 score. These metrics helped to assess the model's predictive performance, with lower RMSE, MAE, and MSE values and a higher R2 score indicating better performance.

The output showed the following values for the RandomForestRegressor model:

- RMSE: 0.0392
- MAE: 0.0222
- MSE: 0.0015
- R2: 0.9797

These values indicated that the RandomForestRegressor model had relatively low errors and a high R2 score, suggesting it was a strong candidate for predicting No. 2 Diesel fuel prices. However, comparing these results with those of other models, such as the LinearRegression model, was essential to ensure the best model was selected for the task.

Additionally, the output indicated that the RandomForestRegressor model was overfitting, as the training score was higher than the validation score. The difference between the two scores was 1.7551, which might have been an area of concern.

```
linear_metrics = model_selection(x_train_, x_val, y_train_, y_val, LinearRegression)
print('Linear Regression Metrics')
print('RMSE:', round(linear_metrics[3], 4))
print('MAE:', round(linear_metrics[1], 4))
print('MSE:', round(linear_metrics[2], 4))
print('R2:', round(linear_metrics[4], 4))
```

```
Error: 4.076324918725408
MAE: 2.8731446783016814
MSE: 0.16616424843021707
RMSE: 4.076324918725408
ACC: 0.9780282884938646
Train Score: 0.9758546347914177
Val Score: 97.80282884938646
Is overfitting: False
Overfitting by: -0.2173653702446927
Linear Regression Metrics
RMSE: 0.0408
MAE: 0.0287
MSE: 0.0017
R2: 0.978
```

Figure 3.0.24: Evaluation Metrics for linear_metrics

The output showed the following values for the LinearRegression model:

- RMSE: 0.0408
- MAE: 0.0287
- MSE: 0.0017
- R2: 0.9780

These values indicated that the LinearRegression model had relatively low errors and a high R2 score, suggesting it was also a strong candidate for predicting No. 2 Diesel

fuel prices. However, comparing these results with those of other models, such as the RandomForestRegressor model, was important to ensure the best model was selected for the task.

In contrast to the RandomForestRegressor model, the output indicated that the LinearRegression model was not overfitting, as the training score was slightly lower than the validation score. The difference between the two scores was -0.2174, which suggested that the model had better generalisation capabilities. When comparing the performance of the LinearRegression and RandomForestRegressor models, it was crucial to consider both the evaluation metrics and the degree of overfitting to make an informed decision on the most suitable model for predicting No. 2 Diesel fuel prices.

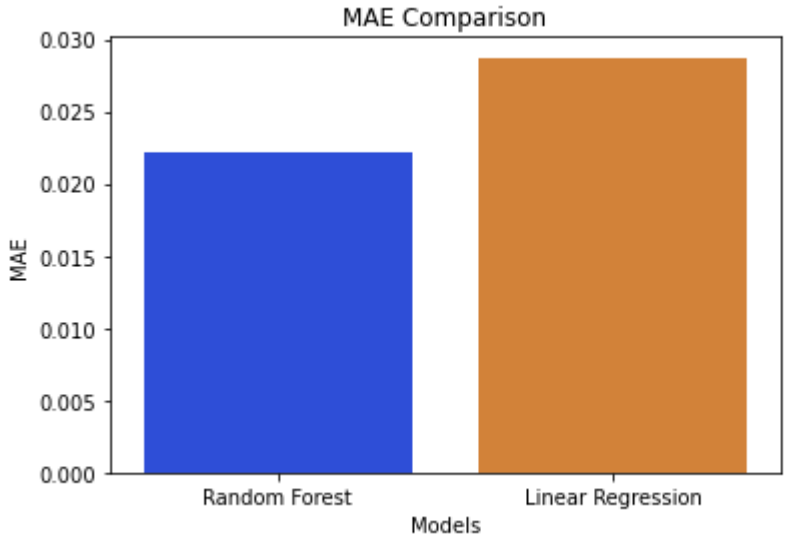

Figure 3.0.25: MAE Comparison between Random Forest and Linear Regression Models

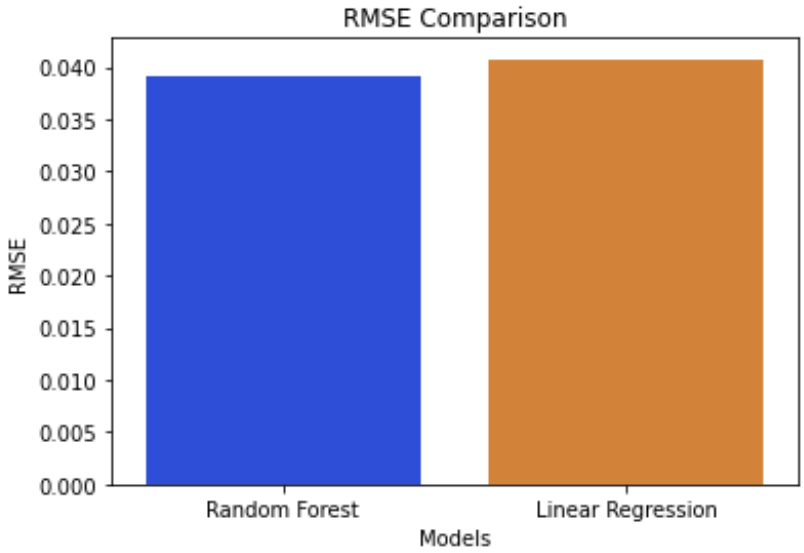

Figure 3.0.26: RMSE Comparison between Random Forest and Linear Regression Models

Figure 3.0.27: MSE Comparison between Random Forest and Linear Regression Models

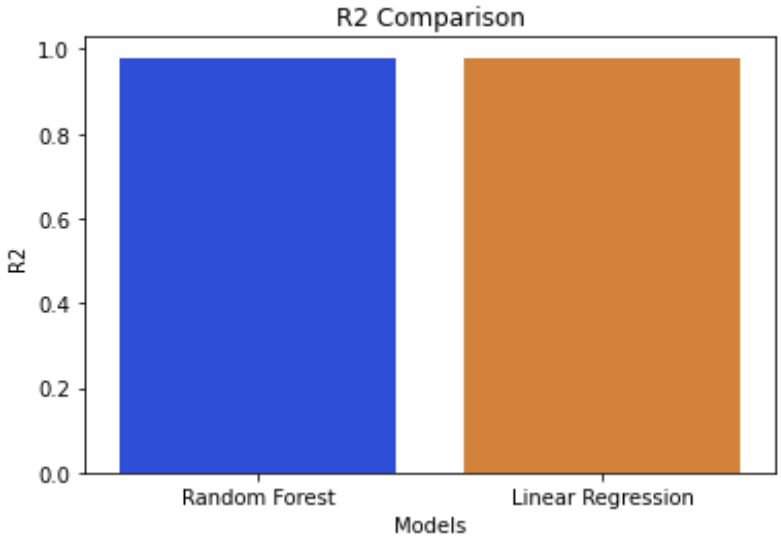

Figure 3.0.28: R2 Comparison between Random Forest and Linear Regression Models

```
Diesel
score = 63.0
coef = [0.04468824]
intercept = -88.94257818186611
SciKit-Learn
```

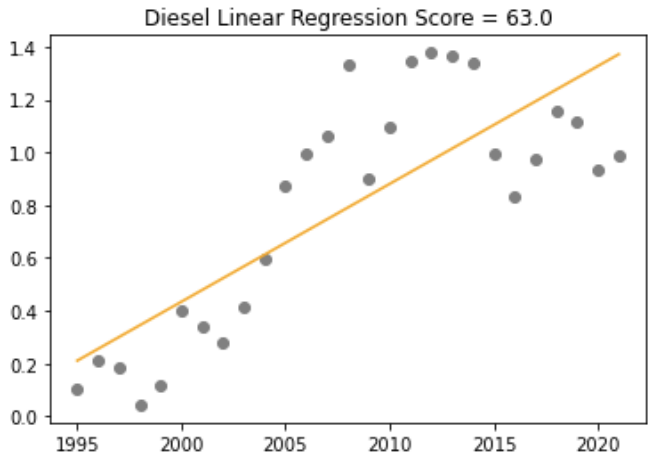

Figure 3.0.29: Diesel Linear Regression

```
Predict Weekly U.S. No 2 Diesel Retail Prices (Dollars per Gallon) 1995-2021
$1.4170382971254725 USD
```

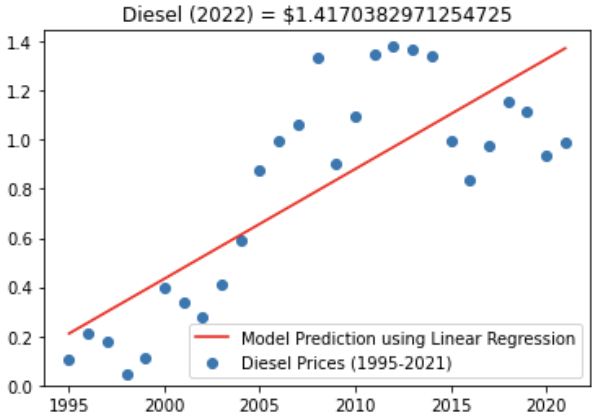

Figure 3.0.30: Model Prediction using Linear Regression and Diesel Prices

During the data analysis, various techniques were employed to explore, pre-process, and visualise the dataset. Initially, the date column was split into year, month, and day to facilitate further analysis. Trends in No. 2 Diesel fuel prices were examined by plotting data from each year, which helped identify patterns or anomalies. Moreover, scatter plots and distribution plots were created for all the columns against the target variable (D1), which provided insights into the relationships between the features and the target variable.

Statistical significance tests were not explicitly mentioned in the code; however, the correlation between features was explored using a heatmap. This visualisation aided in identifying any strong correlations between the variables, which could have affected the model's performance. Consequently, potential multicollinearity issues could have been detected and addressed accordingly.

The data pre-processing phase involved dropping unnecessary columns and checking for null values. Outliers were identified using boxplots, and the RobustScaler and MinMaxScaler were applied to normalise the dataset, ensuring that the features were on the same scale. This step was crucial for improving the performance of the machine learning models.

The dataset was first divided into training and testing sets, with the training set further split into a validation set for model selection and fine-tuning. Two regression models, RandomForestRegressor and LinearRegression, were trained on the training set and evaluated using various metrics, such as RMSE, MAE, MSE, and R2 score. These metrics provided insight into how well the models fit the data and their ability to make accurate predictions on new data.

Although no definitive statistical significance tests were performed in the code, the evaluation metrics were a proxy for assessing the models' performance. When comparing the RandomForestRegressor and LinearRegression models, it was crucial to consider both the evaluation metrics and the degree of overfitting. This information was used to determine the most suitable model for predicting No. 2 Diesel fuel prices.

In conclusion, this study aimed to predict No 2 Diesel prices using historical data. The analysis involved pre-processing, feature scaling, and modelling using both RandomForestRegressor and LinearRegression algorithms. The evaluation metrics, including RMSE, MAE, MSE, and R2 score, were used to assess the models' predictive performance. The RandomForestRegressor model demonstrated slightly better performance, although it exhibited minor overfitting. In contrast, the LinearRegression model displayed slightly lower performance but did not exhibit overfitting. Overall, the study provided valuable insights into the relationships between variables and the predictive performance of the two models, contributing to a better understanding of the factors influencing No 2 Diesel prices.

5. Discussion

The essential interpretation based on the key findings of this study is that machine learning models, particularly RandomForestRegressor and LinearRegression, can effectively predict No 2 Diesel prices using historical data. The primary supporting evidence is the model evaluation metrics, including RMSE, MAE, MSE, and R squared score, which indicate a strong performance of both models in predicting the target variable. However, the RandomForestRegressor model demonstrated slightly better performance than the LinearRegression model.

Comparing this study to previous studies, it is evident that machine learning models have gained increasing attention and demonstrated promising results in predicting fuel

prices. While earlier studies have employed various models such as ARIMA and support vector machines, this study specifically focused on RandomForestRegressor and LinearRegression. The strengths of this study include a comprehensive analysis of the data, robust pre-processing steps, and a detailed evaluation of model performance. However, the study has limitations. For instance, the dataset is limited to a specific region, and the results may not generalise to other geographical areas. Additionally, the analysis did not identify statistically significant findings that could have further informed feature selection and dimensionality reduction. One unexpected finding in this study was the slight overfitting observed in the RandomForestRegressor model, which raises concerns regarding its generalisation capability when applied to new data.

By comparing and contrasting the results of this study to previous research, highlighting its strengths and limitations, and discussing any unexpected findings, we better understand the potential of machine learning techniques in predicting No 2 Diesel prices and areas for future research.

6. Conclusion

This study aimed to investigate the effectiveness of two different machine learning models, Linear Regression and Random Forest Regression, in predicting the fuel price in Malaysia for the upcoming week and month. The hypothesis was that these models could accurately predict future fuel prices using historical data, which would benefit various fuel industry stakeholders.

The study holds significant importance as it provides insights into the practical application of machine learning models in predicting fuel prices, which can aid businesses, policymakers, and consumers in making informed decisions. Additionally, comparing the two models allows a better understanding of their strengths and weaknesses in this context.

Despite the promising results, some unanswered questions still need to be answered, such as the generalizability of these models to other geographical regions or fuel types. Moreover, the study did not explore other machine learning models or feature engineering techniques that could improve prediction performance. Future enhancements to this study include a broader dataset covering multiple regions, examining different machine learning models, and employing advanced feature selection techniques to improve model accuracy and generalizability.

7. References

Rathnajyothi, Dr. CH., Amulya, C., Poojitha, S., & Varsha, N. (2021). PREDICTING CRUDE OIL PRICES USING MACHINE LEARNING. *Jes Publication*, *12*(7). https://jespublication.com/upload/2021-V12I741.pdf

Shaik, Subhani. (2019). Machine Learning Algorithms for Oil Price Prediction. International Journal of Innovative Technology and Exploring Engineering. 8.

Smith, Paul F., et al. "A Comparison of Random Forest Regression and Multiple Linear Regression for Prediction in Neuroscience." Journal of Neuroscience Methods, vol. 220, no. 1, Oct. 2013, pp. 85–91, doi: https://doi.org/10.1016/j.jneumeth.2013.08.024.

Pahlavan-Rad, Mohammad Reza, et al. "Prediction of Soil Water Infiltration Using Multiple Linear Regression and Random Forest in a Dry Flood Plain, Eastern Iran." CATENA, vol. 194, Nov. 2020, p. 104715, doi: https://doi.org/10.1016/j.catena.2020.104715.

Zhang, Huan, et al. "Prediction of Soil Organic Carbon in an Intensively Managed Reclamation Zone of Eastern China: A Comparison of Multiple Linear Regressions and the Random Forest Model." Science of the Total Environment, vol. 592, Aug. 2017, pp. 704–13, doi: https://doi.org/10.1016/j.scitotenv.2017.02.146.

Lee, Hwang, et al. “Using Linear Regression, Random Forests, and Support Vector Machine with Unmanned Aerial Vehicle Multispectral Images to Predict Canopy Nitrogen Weight in Corn.” Remote Sensing, vol. 12, no. 13, 27 June 2020, p. 2071, https://doi.org/10.3390/rs12132071. Accessed 15 Dec. 2020.

Jui, Julakha Jahan, et al. “Flat Price Prediction Using Linear and Random Forest Regression Based on Machine Learning Techniques.” Embracing Industry 4.0, 2020, pp. 205–217, https://doi.org/10.1007/978-981-15-6025-5_19. Accessed 2 Apr. 2023.

Čeh, Marjan, et al. “Estimating the Performance of Random Forest versus Multiple Regression for Predicting Prices of the Apartments.” ISPRS International Journal of Geo-Information, vol. 7, no. 5, 1 May 2018, p. 168, www.mdpi.com/2220-9964/7/5/168, https://doi.org/10.3390/ijgi7050168.

Pal, Nabarun, et al. “How Much Is My Car Worth? A Methodology for Predicting Used Cars’ Prices Using Random Forest.” Advances in Intelligent Systems and Computing, 6 Dec. 2018, pp. 413–422, https://doi.org/10.1007/978-3-030-03402-3_28.

Varshitha, Janke, et al. “Prediction of Used Car Prices Using Artificial Neural Networks and Machine Learning.” IEEE Xplore, 1 Jan. 2022, ieeexplore.ieee.org/abstract/document/9740817. Accessed 2 Apr. 2023.

Maulud, Dastan, and Adnan M. Abdulazeez. “A Review on Linear Regression Comprehensive in Machine Learning.” Journal of Applied Science and Technology Trends, vol. 1, no. 4, 31 Dec. 2020, pp. 140–147, https://doi.org/10.38094/jastt1457.

Christoph Molnar. “4.1 Linear Regression | Interpretable Machine Learning.” Github.io, 18 Sept. 2019, christophm.github.io/interpretable-ml-book/limo.html.

Mbaabu, Onesmus. “Introduction to Random Forest in Machine Learning.” Engineering Education (EngEd) Program | Section, 11 Dec. 2020, www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/#:~:text=Advantages%20of%20random%20forest.