DELHI TECHNOLOGICAL UNIVERSITY

PREDICTING DIESEL PRICE IN DELHI USING MULTIPLE LINEAR REGRESSION

TABLE OF CONTENTS

INTRODUCTION

Today, a litre of diesel has been priced at ₹ 80.73 in Delhi. Fuel prices in Delhi are continuing to rise, so much so, that they have reached levels that have never been witnessed before. This is the outcome of boiling crude oil prices.

Here's a list of six factors that impact petrol and diesel prices:

- Cost of Crude Oil
- Increased Demand
- Mismatch of Supply & Demand
- Tax Rates
- Rupee to Dollar Exchange Rate
- Logistics

We considered two most fundamental parameters in order to predict the diesel price in delhi that are **Foreign exchange rate** & **Crude Oil Price**

What is Foreign exchange rate?

Foreign exchange (Forex or FX) is the conversion of one currency into another at a specific rate known as the foreign exchange rate. The conversion rates for almost all currencies are constantly floating as they are driven by the market forces of supply and demand.

For example, How many **Indian Rupee** does it take to get one **U.S. Dollar** ? As of April 24, 2021, the exchange rate is 74.93, meaning it takes **₹74.93** to get **$1.**

Many factors that can potentially influence the market forces behind foreign exchange rates. The factors include various economic, political, and even psychological conditions. The economic factors include a government's economic policies, trade balances, inflation, and economic growth outlook.

How does foreign exchange rate affect the rate of diesel in Delhi?

Because, we have to import crude oil from other countries out of which major are Iraq(Biggest oil exporter for India) ,Iran and Saudi Arabia ... but India pays them in terms of US Dollar. So, depreciation of Rupee(Say from 70 to 72) makes imports more costly.

For example,

1 Barrel of crude oil is equal to **159 liters**(approximately).
Suppose to buy one barrel of crude oil in the international market costs **$50.82** and to get $1 we have to pay **₹70**, so for $50.82 we end up paying **₹3557**.

Now if rupee depreciates from **₹70 to ₹74** for $1, so for the same $50.82 we have to pay **₹3760** ie, **₹203 more** per barrel only because of exchange rate change thus as an impact even oil price in Delhi will rise.

What is Crude Oil Price ?

Crude oil is a naturally occurring petroleum product composed of hydrocarbon deposits and other organic materials. A type of fossil fuel, **crude oil is refined to produce usable products including** gasoline, **diesel**, and various other forms of petrochemicals. It is a nonrenewable resource, which means that it can't be replaced naturally at the rate we consume it and is, therefore, a limited resource.

- Crude oil is the raw natural resource that is extracted from the earth and refined into products such as gasoline, jet fuel, and other petroleum products.
- Crude oil is a global commodity that trades in markets around the world, both as spot oil and via derivatives contracts.
- Many economists view crude oil as the single most important commodity in the world as it is currently the primary source of energy production.

How Crude Oil Affects the Price of Diesel ?

Diesel is also one of the products obtained from refining the crude oil. So the rate of crude oil directly affects the diesel price in any region of the world.

Petrol and diesel rates in the country have sharply increased after the latest round of fuel price hike by oil marketing companies (OMCs). Experts have said the continuous increase in fuel prices will have a widespread impact on citizens and the overall economy.

Petrol is now retailing above Rs 90 per litre in all major cities including Delhi and is on the verge of crossing Rs 100 in major cities. It has already hit a century in some circles. Diesel, too, has climbed to levels never seen before in the country.

PROBLEM DEFINITION

To make a model using multi linear regression which can predict the price of diesel in Delhi on the basis of two parameters foreign exchange rate and crude oil rate and then analyse our model by calculating various metrics.

METHODOLOGY

What Is Multiple Linear Regression (MLR) ?

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is

to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

- Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.
- Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable.
- MLR is used extensively in econometrics and financial inference.

Formula and Calculation of Multiple Linear Regression

Formula and Calculation of Multiple Linear Regression

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = explanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

What Multiple Linear Regression Can Tell You

Simple linear regression is a function that allows an analyst or statistician to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables—an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends to several explanatory variables.

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables
- The independent variables are not too highly correlated with each other
- yi observations are selected independently and randomly from the population
- Residuals should be normally distributed with a mean of 0 and variance σ

The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R2 always increases as more predictors are added to the MLR model, even though the predictors may not be related to the outcome variable.

R2 by itself can't thus be used to identify which predictors should be included in a model and which should be excluded. R2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables and 1 indicates that the outcome can be predicted without error from the independent variables.

When interpreting the results of multiple regression, beta coefficients are valid while holding all other variables constant ("all else equal"). The output from a multiple regression can be displayed horizontally as an equation, or vertically in table form.

The Difference Between Linear and Multiple Regression

Ordinary linear squares (OLS) regression compares the response of a dependent variable given a change in some explanatory variables. However, it is rare that a dependent variable is explained by only one variable. In this case, an analyst uses multiple regression, which attempts to explain a dependent variable using more than one independent variable. Multiple regressions can be linear and nonlinear.

Multiple regressions are based on the assumption that there is a linear relationship between both the dependent and independent variables. It also assumes no major correlation between the independent variables.

DATASET DESCRIPTION

We have taken our dataset from Kaggle. The link to the dataset has been given below:

https://www.kaggle.com/adityaghodgaonkar/delhidieselpriceprediction?select=Diesel_Price_Train.csv

It is a Delhi Diesel Price Prediction dataset, which has 5300 records. It has recorded the values of Diesel Prices in Delhi over the period of year 2003 - 2017, having the dependency on Foreign Exchange Rate and Crude Oil Prices.

EXPLANATION OF THE MODEL

Multiple Linear Regression

Step 1: Importing the Libraries

We have used three libraries Numpy, Pandas and Matplotlib.
NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices. NumPy stands for Numerical Python.

Pandas is a Python library which is used to analyze data.

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Multiple Linear Regression

Importing the libraries

```
[1]  #importing numpy library for numerical calculation
     import numpy as np

     #importing pandas library for accessing the dataset and performing manipulations
     import pandas as pd

     #importing matplotlib for plotting functions
     import matplotlib.pyplot as plt
```

Step 2: Importing the Dataset

Next step is to import the dataset. For the dataset, we have imported the Delhi Diesel Price Prediction dataset from Kaggle. It consists of records of Diesel Price starting from the year 2003 upto the year 2017.

Importing the dataset

```
[2]  #importing the dataset using pandas library
     dataset = pd.read_csv('Diesel_Price_Train.csv')

     #printing the initial values of our dataset
     dataset.head()
```

	Date	Delhi Diesel Price	Foreign Exchange Rate	Crude Oil Price
0	01-01-2003	19.47	48.01	29.59
1	02-01-2003	19.47	48.05	29.59
2	03-01-2003	19.07	48.00	29.59
3	04-01-2003	19.07	48.00	29.59
4	05-01-2003	19.07	48.00	29.59

```
[3] #printing the whole dataset consisting of approx 5300 records
    dataset
```

	Date	Delhi Diesel Price	Foreign Exchange Rate	Crude Oil Price
0	01-01-2003	19.47	48.01	29.59
1	02-01-2003	19.47	48.05	29.59
2	03-01-2003	19.07	48.00	29.59
3	04-01-2003	19.07	48.00	29.59
4	05-01-2003	19.07	48.00	29.59
...	...	...	...	...
5294	30-06-2017	53.46	64.73	46.56
5295	01-07-2017	53.33	64.73	47.86
5296	02-07-2017	53.36	64.73	47.86
5297	03-07-2017	53.47	64.66	47.86
5298	04-07-2017	53.44	64.81	47.86

5299 rows × 4 columns

```
[4] # Printing the information of the dataset
    dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5299 entries, 0 to 5298
Data columns (total 4 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Date                   5299 non-null   object 
 1   Delhi Diesel Price     5299 non-null   float64
 2   Foreign Exchange Rate  5299 non-null   float64
 3   Crude Oil Price        5299 non-null   float64
dtypes: float64(3), object(1)
memory usage: 165.7+ KB
```

```
[5]  # describe the dataset
     dataset.describe()
```

	Delhi Diesel Price	Foreign Exchange Rate	Crude Oil Price
count	5299.000000	5299.000000	5299.000000
mean	38.082433	51.257207	70.719891
std	11.191295	8.732785	28.862465
min	19.070000	39.040000	24.210000
25%	30.450000	44.800000	46.590000
50%	37.750000	46.890000	66.900000
75%	48.010000	60.330000	101.570000
max	59.020000	68.810000	132.470000

Step 3 : Check for the Outliers

The next step is to check for the outliers present in the dataset.

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

We have checked for the outliers in the dataset using the boxplot method implemented with the help of seaborn library. We plotted the boxplot of all the attributes i.e Diesel Price, Forex Rate and Crude Oil Price and found that the dataset contained no outliers as show in the boxplots given below:

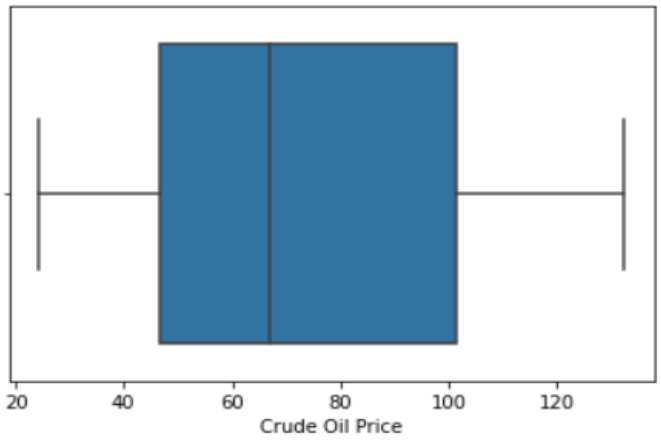
BoxPlot for Crude Oil Price

Checking for Outliers using BoxPlot Method

```
[6]  #importing the seaborn library
     import seaborn as sns

     #Plotting the boxplot for Crude Oil Price
     sns.boxplot(x = dataset['Crude Oil Price'])
```

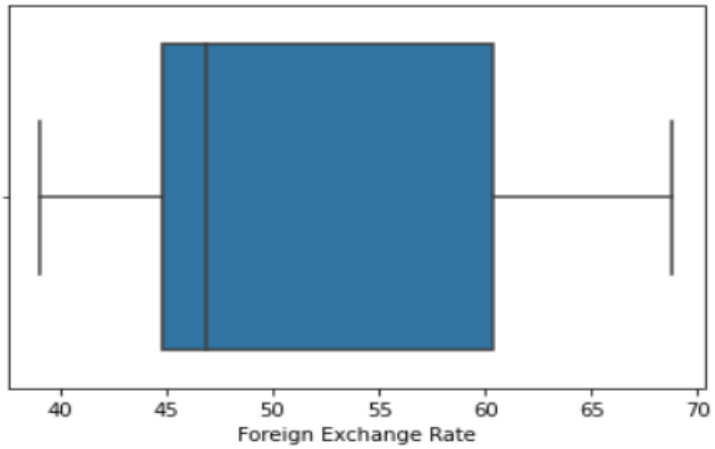
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42ac6bab90>
```



BoxPlot for Foreign Exchange Rate

```
[7]  #Plotting the boxplot for Forex Rate
     sns.boxplot(x = dataset['Foreign Exchange Rate'])
```

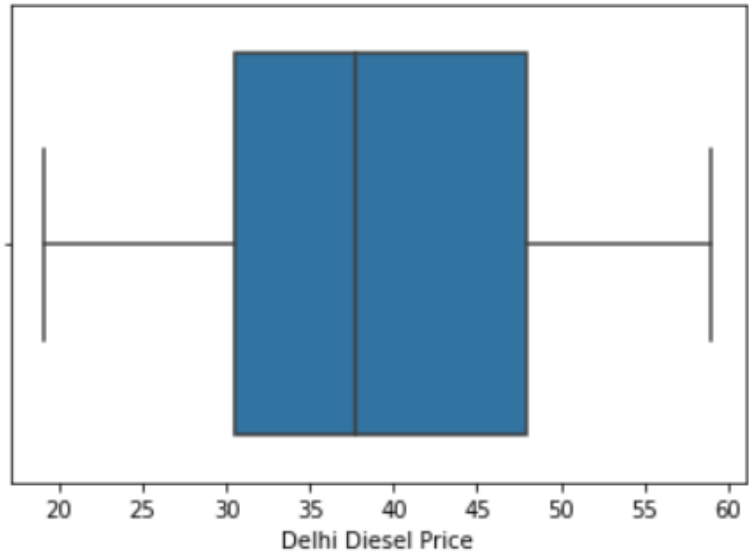
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42ac612a10>
```



BoxPlot for Diesel Price in Delhi

```
[8]  #Plotting the boxplot for Delhi Diesel Price
     sns.boxplot(x = dataset['Delhi Diesel Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42ac142950>
```



Step 4 : Splitting the dataset into Independent and Dependent Variables

The next most important for making the multi linear model is to divide the dataset into independent and dependent variables.

Independent variables are represented by 'X' and are the items which are not dependent on other variables. In our dataset, Forex Rate and Crude Oil Price are the independent variables.

Dependent variables are represented by 'y' and are the items which are dependent on the independent variables. In our dataset, Diesel Price is the dependent variable which depends on the Crude Oil Price and Forex Rate.

Splitting the independent (X) and dependent variables (y)

```
[9]  # X consists of independent variable (Forex Rate and Crude Oil Price)
     # y is the dependent variable (diesel price)
     X = dataset.iloc[:, 2: ].values
     y = dataset.iloc[:, dataset.columns == 'Delhi Diesel Price'].values
```

```
[10] #printing the independent variables (Forex Rate and Crude Oil Price)
     print(X)
```

```
[[48.01 29.59]
 [48.05 29.59]
 [48.   29.59]
 ...
 [64.73 47.86]
 [64.66 47.86]
 [64.81 47.86]]
```

```
[11] #printing the dependent variable (diesel price)
     print(y)
```

```
[[19.47]
 [19.47]
 [19.07]
 ...
 [53.36]
 [53.47]
 [53.44]]
```

We have printed the independent variables and dependent variables as shown above.

Step 5 : Splitting the independent and dependent variables into training set and testing set

The fifth step is to divide the variables into training set and testing set.

As the name suggests, the training set will be used for training our multi regression model and the testing set will be used to check the accuracy of our trained model.

We have divided the dataset into training and testing in the ratio of 80 : 20.

Splitting the dataset into the Training set and Test set

```
[12] #importing the sklearn libary for perfroming train test split
     from sklearn.model_selection import train_test_split

     #splitting the dataset into training set and testing set
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Then we have printed the dimensions of the training set and testing set.

X_train : Training Set consisting of Independent Variables

y_train : Training set consisting of Dependent variable

X_test : Testing Set consisting of Independent Variables

y_test : Testing set consisting of Dependent variable

```
[13] #Printing the dimensions of training and testing set
     print(X_train.shape)
     print(y_train.shape)
     print(X_test.shape)
     print(y_test.shape)
```

```
(4239, 2)
(4239, 1)
(1060, 2)
(1060, 1)
```

Step 6 : Plotting the correlation matrix

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables.

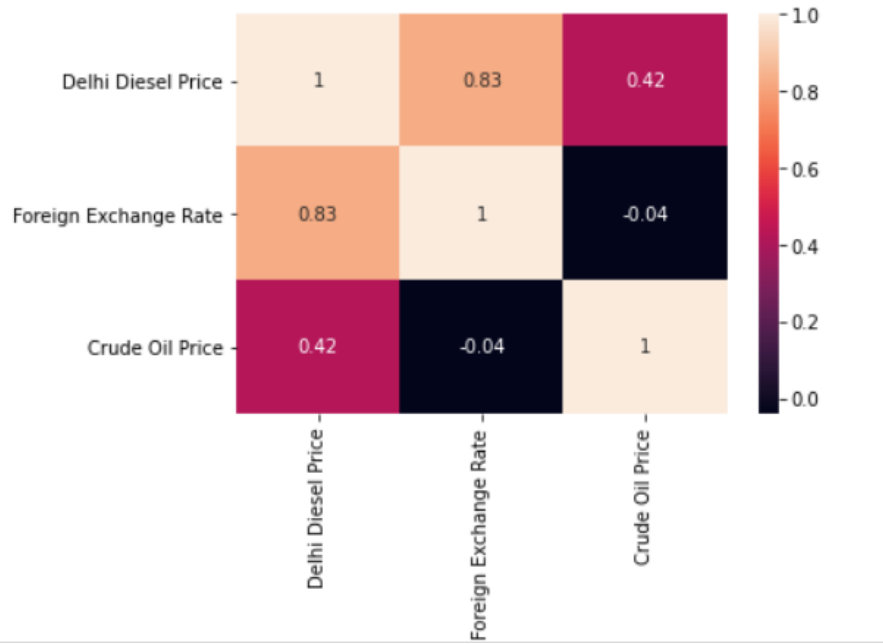
We have plotted the correlation matrix heatmap for the variables using the seaborn.

Plotting the correlation heatmap for the dataset

```
[14] #Finding the correlation between our dataset
     corr = dataset.corr()

     #Importing the seaborn library for plotting the correlation martix
     import seaborn as sns
     sns.heatmap(corr, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42ac62a3d0>
```



Step 7 : Training the multi-linear regression model

First work is to import the Linear Regression module from sklearn model and then train the model using X_train and y_train.

▸ Training the Simple Linear Regression model on the Training set

```
[15] #importing the linear regression module from sklearn libary
     from sklearn.linear_model import LinearRegression

     #getting the linear regression model function for our regressor
     regressor = LinearRegression()

     #training the model using linear regression on training set
     regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Step 8 : Predicting the results

Once we have trained the model, the next step is to predict the values on the test set i.e. X_test.

The predicted values have been shown on the left hand side and actual values of X_test have been shown on the right hand side.

Predicting the Diesel Prices on the Test set

```
[16] #predicting the values(diesel prices) on our test dataset
     #using the multiple linear regression model
     y_pred = regressor.predict(X_test)

     #set the precision upto 2 decimal places
     np.set_printoptions(precision = 2)

     #Print the predicted values and actual values of the test dataset
     print(np.concatenate((y_pred.reshape(len(y_pred), 1), y_test.reshape(len(y_test),1)), 1))
```

```
[[28.25 30.76]
 [47.73 48.16]
 [53.56 59.02]
 ...
 [52.69 50.84]
 [38.52 41.12]
 [39.22 34.8 ]]
```

Step 9 : Printing the coefficients and the intercept

Once we have predicted the values on the test dataset, then, we have to find the coefficients associated with independent variables and the intercept for getting the optimal equation of our model.

Printing the Coefficients of Multilinear Model

```
[17] #Printing the coefficients related with our model
     print('Coefficients: \n', regressor.coef_)
```

```
Coefficients:
 [[1.09 0.18]]
```

```
[18] #Printing the intercept of the model
     print('Intercepts : \n', regressor.intercept_)
```

```
Intercepts :
 [-29.9]
```

Then after finding the coefficients and intercept, then equation came out to be:

Relationship between dependent and independent variables

Diesel Prices = 1.09 x (Forex Rate) + 0.18 x (Crude Oil Price) - 29.9

Step 10 : Printing the Variance score

After finding the coefficients and intercept, we find the variance associated with the model.

Variance tells the goodness of the model if it tends to 1.

```
[19] #Print the variance score of our model
     print('Variance score: %.2f' % regressor.score(X_test, y_test))

     Variance score: 0.90
```

```
[20] #Print the Mean Squared Error of our Model
     print("Mean squared error: %.2f" % np.mean((regressor.predict(X_test) - y_test) ** 2))

     Mean squared error: 13.14
```

Step 11 : Finding the Individual Relationship of Diesel Price on Crude and Forex Rate

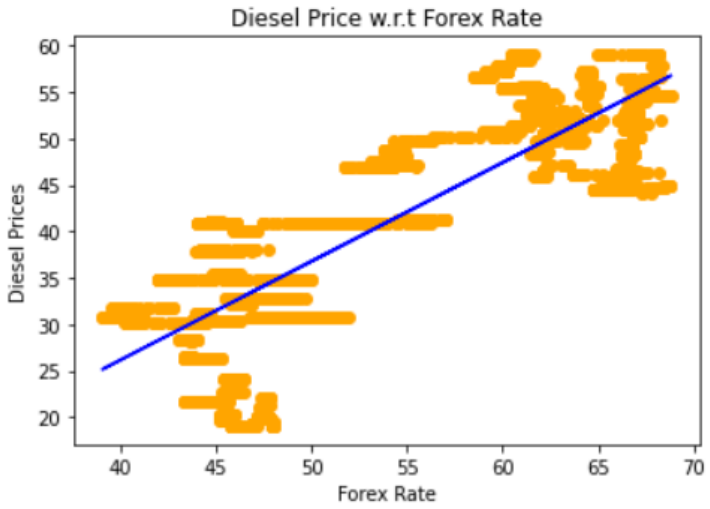
Relationship between Diesel Price and Forex Rate

```
[21] # Training the new model by considering only Forex Rate as independent variable
     # and Diesel as dependent variable
     Forex_train = X_train[:, 0:1]
     regressor.fit(Forex_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Plotting the graph for diesel price w.r.t Foreign Exchange Rate

```
[22] #Plot of Diesel Price w.r.t Forex Rate
     plt.scatter(Forex_train, y_train, color = 'orange')
     plt.plot(Forex_train, regressor.predict(Forex_train), color = 'blue')
     plt.title('Diesel Price w.r.t Forex Rate')
     plt.xlabel('Forex Rate')
     plt.ylabel('Diesel Prices')
     plt.show()
```



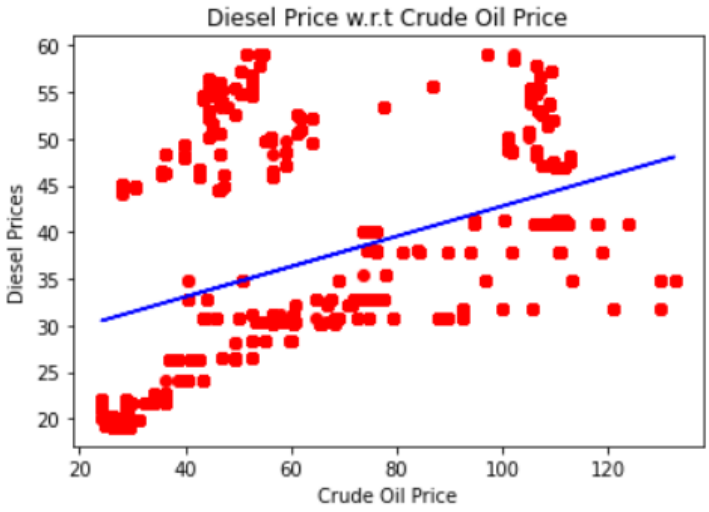
Relationship between Diesel Price and Crude Oil Price

```
[23] # Training the new model by considering only Crude Oil as independent variable
     # and Diesel as dependent variable
     Crude_train = X_train[:, 1:]
     regressor.fit(Crude_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

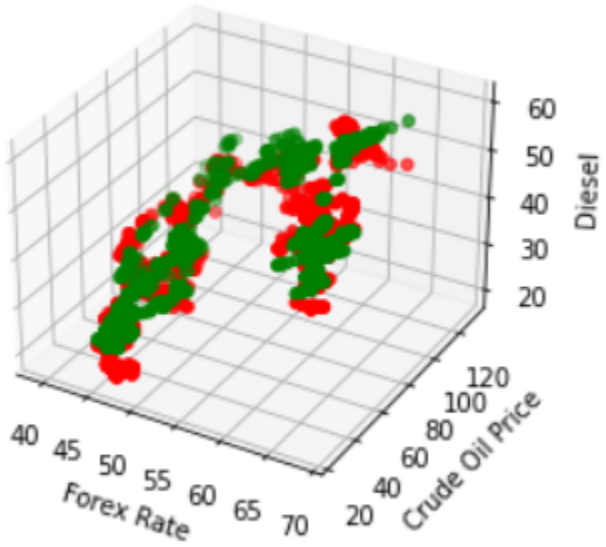
Plotting the graph for diesel price w.r.t Crude Oil Price

```
[24] #Plot of Diesel Price w.r.t Forex Rate
     plt.scatter(Crude_train, y_train, color = 'red')
     plt.plot(Crude_train, regressor.predict(Crude_train), color = 'blue')
     plt.title('Diesel Price w.r.t Crude Oil Price')
     plt.xlabel('Crude Oil Price')
     plt.ylabel('Diesel Prices')
     plt.show()
```



Step 12 : Visualising Our Multiple Linear Regression using 3D Plot

```
[25] fig = plt.figure()
     ax = fig.add_subplot(111, projection = '3d')
     ax.scatter(X_test[:, 0], X_test[:, 1], y_test, c = 'r', marker = 'o')
     ax.scatter(X_test[:, 0], X_test[:, 1], y_pred, c = 'g', marker = 'o')
     ax.set_xlabel('FOREX')
     ax.set_ylabel('CRD')
     ax.set_zlabel('Diesel')
     plt.draw()
```



The above 3D plot shows the Diesel dependence on Forex Rate and Crude Oil Price. The **red color** shows the **actual results** of the test dataset and the **green color** shows the **predicted results** on the test dataset. Since the **green color overlaps** most part of the region covered by **red color**, it implies that our model is **well-trained** and **precise**.

Step 13 : Calculating the metrics for determining the performance of the model

Regression Metrics for Model Peformance

```
[26] #printing the maximum predicted diesel price on test set
     print(max(y_pred))

     #printing the maximum actual diesel price on test set
     print(max(y_test))
```

```
[60.9]
[59.02]
```

```
[27] #Printing the Mean Absolute Error using sklearn library
     from sklearn.metrics import mean_absolute_error
     print("Mean Absolute Error : ", mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error :  2.9960182186261273
```

```
[28] #Printing the Mean Squared Error using sklearn library
     from sklearn.metrics import mean_squared_error
     print("Mean Squared Error : ", mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error :  13.142888688530956
```

```
[29] #Printing the Root Mean Squared Error using sklearn library
     from sklearn.metrics import mean_squared_error
     print("Root Mean Squared Error : ", mean_squared_error(y_test, y_pred, squared = False))
```

```
Root Mean Squared Error :  3.625312219455168
```

```
[30] #Printing the R2 value using sklearn library
     from sklearn.metrics import r2_score
     print ("R2 Score value: {:.4f}".format(r2_score(y_test, y_pred)))
```

```
R2 Score value: 0.8974
```

Mean Absolute Error (MAE):

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

MAE for our model came out to be **2.996**

Mean Squared Error (MSE):

Mean squared error (MSE) or mean squared deviation (MSD) of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

MSE for our model came out to be **13.142**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Root mean squared error (RMSE):

RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

RMSE for our model came out to be **3.625**

R2 Score:

The R^2 score is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It gives information about the goodness of a trained model.

R^2 for our model came to be 0.897 which is near to 1. It specifies our model is very fit for getting predictions.

RESULT & CONCLUSION

R^2Score is a metric to evaluate performance of a regression model. It is also called the coefficient of determination. It gives us an idea of goodness of fit for the linear regression models. It indicates the percentage of variance that is explained by the model.

Mathematically, R^2Score = Explained Variation/Total Variation

In general, the higher the R^2Score value, the better the model fits the data. Usually, its value ranges from 0 to 1. So, we want its value to be as close to 1. Its value can become negative if our model is wrong.

From our model we achieved a R^2**Score = 0.8974** which is assumed to be as accurate as it is close to 1. It implies our multi-linear model is perfectly fit for predictions and is accurate and precise.

From the model, we derived the following:

Coefficient of Foreign exchange rate = 1.09

Coefficient of Crude Oil rate = 0.18

Intercept value = -29.9

Thus we can say that the regression model that we formed adequately predicts the price of diesel from the values of foreign exchange rate and crude oil price.

Our final equation for the regression was:

$$\text{Diesel Price} = 1.09 \times \text{Forex Rate} + 0.18 \times \text{Crude Oil} - 29.9$$

IMPLEMENTATION CODE

We have implemented the code for our project at the link given below:

https://colab.research.google.com/drive/1Tqz4D8tyh_n72Fty5vlPtgrXngF-zBhE#scrollTo=qlHVNpX3wJ01

REFERENCES

The concepts and ideas in this project have been taken from the following websites and books :-

i. Machine learning notes by Andrew Ng

ii. https://en.wikipedia.org/wiki/Linear_regression

iii.https://en.wikipedia.org/wiki/Simple_linear_regression

iv. https://en.wikipedia.org/wiki/Ordinary_least_squares

v. https://en.wikipedia.org/wiki/Root-mean-square_deviation

vi. https://en.wikipedia.org/wiki/Coefficient_of_determination

vii.https://www.statisticssolutions.com/assumptions-of-linear-regression/

viii.Python Data Science Handbook by Jake VanderPlas

ix. Hands-On Machine Learning with Scikit Learn and Tensorflow by Aurilien Geron

x. Introduction to Machine Learning with Python by Andreas C Muller and Sarah Guido