INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Customer Relationship Management & Change Management with Business Intelligence Coursework 02 Task A: Multiple Linear Regression

Student Name: Ann Moira Alfred

IIT Identification: 20211486

Westminster Identification: w1911178

Module: 6BUIS017C CRM & CM with Business Intelligence

Contents

Introduction

The report intends to compartmentalize and discuss a Multiple Regression model built with purpose of predicting the price for natural gas (Brent Oil Prices). The model utilizes a dataset that contains the price of Brent oil prices from 1987 to 2019. Although the Brent Oil prices are influenced by various factors such as geopolitical and macroeconomic aspects, this Multiple Regression model will only proceed to employ two independent variables to conduct the predictions – MA3 (3 day Moving Average on the actual price) and MA9 (9 day Moving Average on the actual price).

Objectives

- Build a predictive model using Multiple Linear Regression to forecast Brent oil prices.
- Evaluate the model's performance using accuracy measures such as Mean Squared Error (MSE) and R-Squared (R_2).
- Visualize the model's predictions to depict the model's effectiveness.
- Construct the Multiple Linear Regression Equation.

Data

The dataset comprises of 8216 records with only two attributes – Date and Price. The data is loaded and pre-processed to retrieve any null values and duplicates.

Originally, the attribute Date consists of the data type "object" and the attribute Price of the data type "float64". However, for further analysis, the attribute Date should be converted to "datetime" datatype for the Python to handle data more effectively.

```
brentoilpricesdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8216 entries, 0 to 8215
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    8216 non-null   datetime64[ns]
 1   Price   8216 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 128.5 KB
```

Figure 1: Pre-processed dataset information

Part (a): Data Visualization

- **Data Visualization**

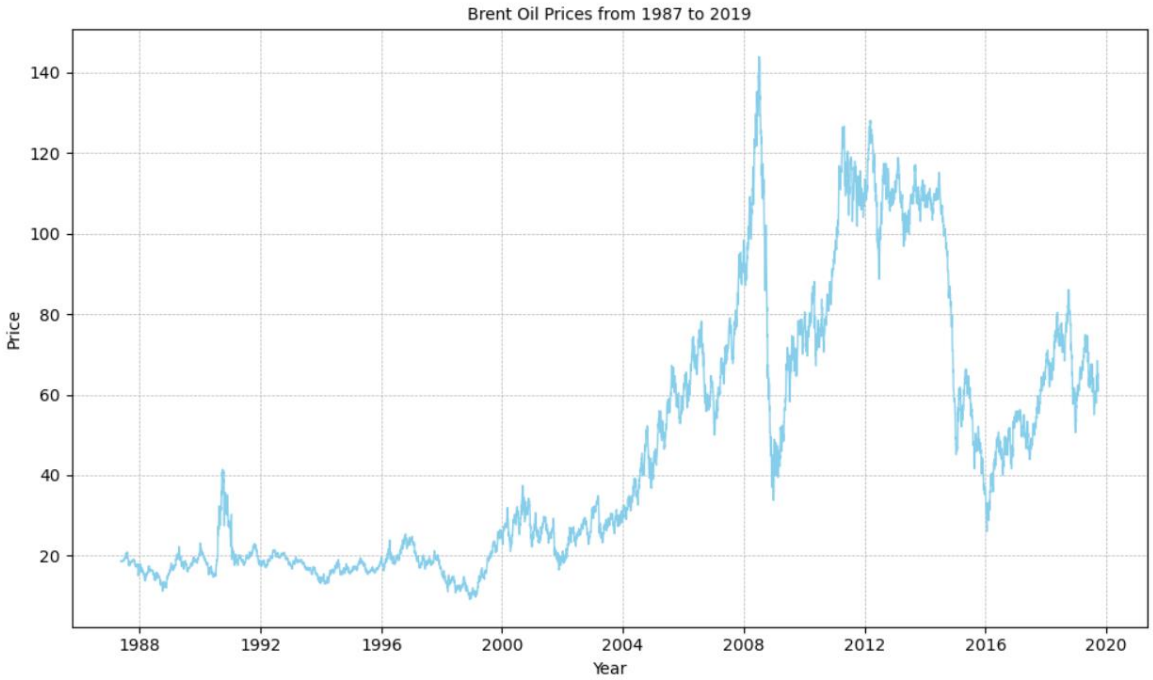

Figure 2: Brent oil price visualization

- **Line Chart Behavior**

1987 – 2000: Brent oil prices during this period appear to be relatively low and stable, fluctuating only between $10 and $30 for the most part except for the sharp spike that occurs in the early 1990s but returns to a stable position again. This could be the result of the first Gulf War which restricted the supply of the resource to the rest of the world (Sun, 2022).

2000 – 2008: A slow and steady price increment is visible in the beginning of the year 2000 corresponding to aggregated global demand for Brent oil with other emerging markets and economies partaking in the aggregation of demand for this product.

2008: A steep decline is evident consequent to the 2008 global financial crisis that evidently led to the collective demise of global markets crashing any demand that exists for produce such a Brent oil (EuropeanCentralBank, 2012). A significant price change is visible from over $140 to $40 in less than a couple of months as global demand collapses.

2009 – 2014: A quick stabilisation is evident from 2009 after the global financial crisis, steadying the price between $80 to $120 per barrel.

2014 – 2016: Yet another steep price drop is made evident on the visualization reaching to extreme lows as almost $30 by early 2016. This may correspond to the oil glut that occurred in the US due to an oversupply of Brent oil driven by the US shale production and OPEC's (Organization of the Petroleum Exporting Countries) decision to not bring down manufacture numbers. (WorldBank, 2018)

2016 – 2019: Brent oil prices emerge to increment over a steady pace fluctuating between $40 and $80 per barrel.

Part (b): Explanatory Variables

- **Explanatory Variables (MA3 & MA9)**

Two attributes are added to the data frame **"brentoilpricesdf"** – MA3 and MA9, representing a 3-day Moving Average and a 9-day Moving Average. These attributes are calculated using the **Rolling Window method** which operates by taking the average of the last **N** number of values instructed, where **N** is the window size. For example, a 3-day Moving Average is calculated by using the average of the past 3 days' values present of the data frame.

```
brentoilpricesdf = brentoilpricesdf.sort_values(by='Date')

# Calculate the 3-day moving average (MA3) and 9-day moving average (MA9)
brentoilpricesdf['MA3'] = brentoilpricesdf['Price'].rolling(window=3).mean()
brentoilpricesdf['MA9'] = brentoilpricesdf['Price'].rolling(window=9).mean()

# Display the dataframe with the new moving averages
print(brentoilpricesdf[['Date', 'Price', 'MA3', 'MA9']].head(10))
```

```
        Date  Price        MA3        MA9
0 1987-05-20  18.63        NaN        NaN
1 1987-05-21  18.45        NaN        NaN
2 1987-05-22  18.55  18.543333        NaN
3 1987-05-25  18.60  18.533333        NaN
4 1987-05-26  18.63  18.593333        NaN
5 1987-05-27  18.60  18.610000        NaN
6 1987-05-28  18.60  18.610000        NaN
7 1987-05-29  18.58  18.593333        NaN
8 1987-06-01  18.65  18.610000  18.587778
9 1987-06-02  18.68  18.636667  18.593333
```

Figure 3: Building Explanatory Variables

The Rolling Window method was decided due its ability to smooth out fluctuations in time series data analysis, aiding the user to interpret clearer data trends. This ability makes it particularly useful for forecasting purposes since it provides a rather stable depiction of recent data patterns. The Lag method however is dependent on fixed intervals for its analysis unlike the Rolling Window method. It's dynamic and flexible nature allows models to capture short term

patterns without being constrained to rigid tame intervals allowing the model to respond effectively to recent fluctuations, making it the better choice for Time Series Forecasting.

The attributes **MA3 and MA9** will aid as **the independent variables** for the Multiple Linear Regression model. These Moving Average values reflect on flattened patterns of Brent oil prices during the past 3 and 9 days which are presumed to exhibit a predictive correlation with the model's dependant variable that is the Brent oil Price.

- **Removal for NaN Values**

NaN valued records are immediately eliminated after the building of explanatory variables due to their incomplete nature that performs negative purpose in the prediction model. It would only distort the analysis and predictions performed by the model.

```
# Drop rows with NaN values
brentoilpricesdf = brentoilpricesdf.dropna()

# Verify the dataset
print(brentoilpricesdf.head())
```

```
         Date  Price        MA3        MA9
8  1987-06-01  18.65  18.610000  18.587778
9  1987-06-02  18.68  18.636667  18.593333
10 1987-06-03  18.75  18.693333  18.626667
11 1987-06-04  18.78  18.736667  18.652222
12 1987-06-05  18.65  18.726667  18.657778
```

Figure 4: Cleaned Brent oil prices data frame ready for prediction

Part (c): Train and Test Data

- **Train and Test Dataset**

```
from sklearn.model_selection import train_test_split

X = brentoilpricesdf[['MA3', 'MA9']]
Y = brentoilpricesdf[['Price']]


# Split the data into training (80%) and testing (20%) sets
Xtrain, Xtest, Ytrain,Ytest = train_test_split(X, Y, test_size=0.2, random_state=42)

# Check the shape of the split data
print("Training set:", Xtrain.shape, Ytrain.shape)
print("Testing set:", Xtest.shape, Ytest.shape)
```

```
Training set: (6566, 2) (6566, 1)
Testing set: (1642, 2) (1642, 1)
```

Figure 5: Splitting the Brent oil prices dataset.

The "brentoilpricesdf" data frame is divided into a training dataset and a testing dataset using an **80 - 20 ratio**, meaning that the data that is utilized to train this model will comprise of only 80% (6566 records) of the "brentoilpricesdf" and the remaining 20% (1642 records) will be exploited in testing the accuracy of its predictions.

- **Training Dataset (Xtrain, Ytrain)**

Xtrain (6566,2): The training dataset features in the Xtrain Dataframe containing 6566 records / instances and 2 features / attributes which are MA3 ad MA9.

Ytrain (6566,1): The training dataset target variable Ytrain contains 6566 records / instances and only 1 feature / attribute which is Brent oil Price.

- **Testing Dataset (Xtest, Ytest)**

Xtest (1642,2): The testing dataset assigned variable Xtest contains 1642 records / instances and 2 features /attributes (MA3 and MA9) ready for evaluating the model's performance.

Ytest (1642,1): The testing target dataset only comprises of 1642 records / instances and only 1 feature / attribute which is Brent oil Price.

Figure 6: Graphical depiction of data split

- **Reproducible Seed**

A Reproducible Seed, also known as the **Random Seed**, is settled value and a parameter in this Regression model that is employed as a random number generator, confirming the outcomes of random processes such as data splitting so that it remains unswerving over multiple runs. The purpose of the Random seed is to make the entire process reproducible. This means that any user who is running the model can expect the same split and model behaviour. The choice of number **42** is completely arbitrary however is commonly used in mathematical and computational practices as a default value.

```
# Split the data into training (80%) and testing (20%) sets
Xtrain, Xtest, Ytrain,Ytest = train_test_split(X, Y, test_size=0.2, random_state=42)
```

Figure 7: Random / Reproducible Seed

Part (d): Building a Linear Regression Model

- **Defining and Training the Linear Regression Model**

The Linear Regression class is retrieved from the **"sklearn.linear_model"** module to create a Linear Regression Model. The model is then trained using the **"fit( )"** method, which uses the parameters **Xtrain** and target values dataset **Ytrain.** During this computation, the model will learn the relationship between the independent variables (MA3 and MA9) and the dependent variable (Price) by calculating the optimum values for the X coefficients and the Y intercept. The model is now ready to make predictions on new and unseen data.

```
from sklearn.linear_model import LinearRegression

# Create the linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(Xtrain, Ytrain)
```

```
LinearRegression
LinearRegression()
```

Figure 8: Multiple Linear Regression model construction

Part (e): Prediction Function and Result

- **Visualize the Linear Regression Model**

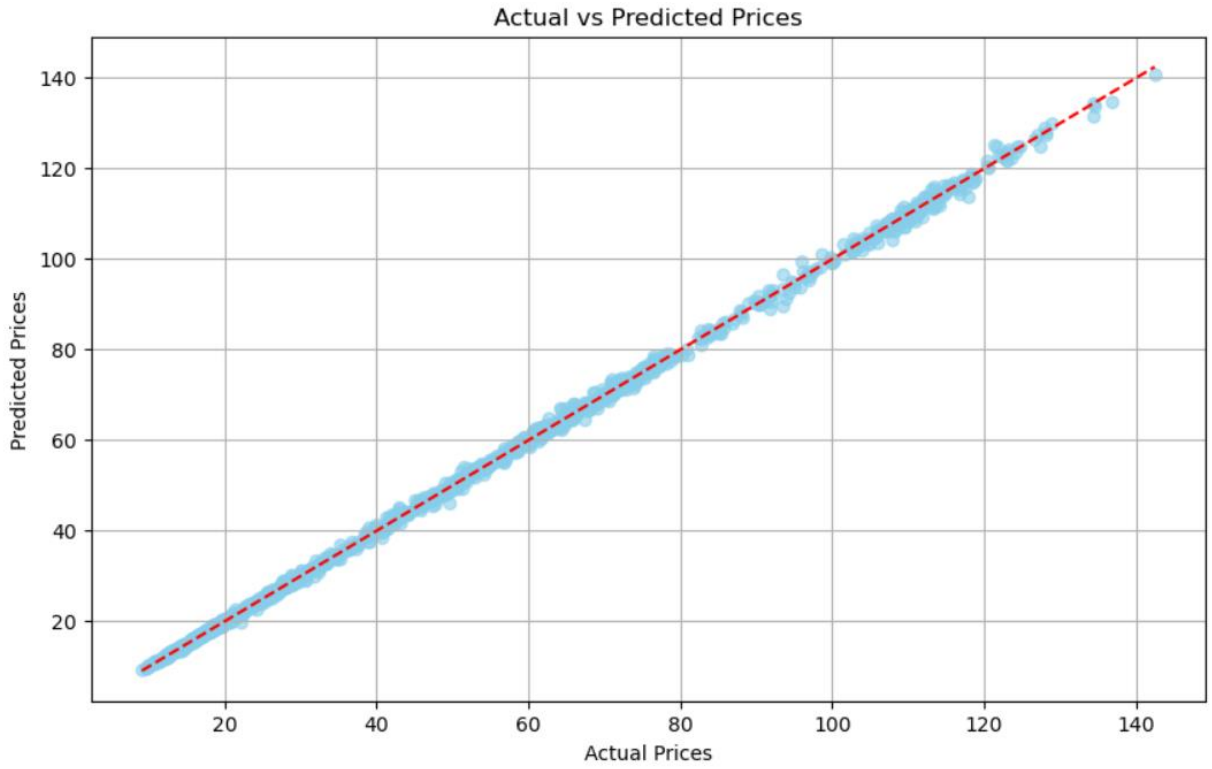

Figure 9: Graphical representation of the Actual and Predicted Brent oil prices

The scatterplot illustrates the relationship between the actual and the predicted Brent oil prices of the priorly processed Multiple Linear Regression model. There is a red dashed line that runs through the datapoints that clearly illustrate the users a concise **Trend Line**. The **close alignment** of the data points along this trend line demonstrates the high performance of the Regression model. Whilst the visualizations exist, there are also other methods to evaluate the performance of the Regression model that is further explored below.

- **Test the Linear Regression Model**

Multiple evaluation metrics are employed to assess the performance of the Multiple Linear Regression Model. These metrics include **Mean Squared Error (MAE)**, **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)** and $\mathbf{R^2}$ Score for both the training and testing datasets. These metrics compare the predicted prices against the actual prices and provide a comprehensive assessment of the model's performance.

Evaluation Metrics	Training	Testing
Mean Squared Error (MAE)	0.5977	0.5190
Root Mean Squared Error (RMSE)	0.7731	0.7204
Mean Absolute Error (MAE)	0.4938	0.4673
R_2	0.9994	0.9995

Figure 10: Evaluation Metrics Values

01. Mean Squared Error (MSE)

Mean Squared Error measures the average squared difference between the actual and predicted Brent oil prices. A low Mean Squared Error is output, implying that the model's prediction values are extremely close to the actual values suggesting theta the model's performance is efficient and with minimal prediction errors.

02. Root Mean Squared Error (RMSE)

Root Mean Squared Error is the square root of Mean squared Error and provides the error in the same unit as the target variable which is the Brent oil Price. The average prediction error is between 0.72 and 0.77 which is insignificant when compared to oil prices, thus leaving very little room for error. This again suggests that the model is performing quite well when it comes to accuracy.

03. Mean Absolute Error (MAE)

Mean Absolute Error measures the average between the actual and the predicted Brent oil prices. In this case the Mean Absolute Error appears to be less than 0.5 units indicating a highly precise set predictions with minimal deviations from the actual Brent oil prices.

04. R_2

The R2 Score measures how well the model performs in terms of variance in target variable. With the score being 0.9995 and extremely close to 1.0, the model appears to be extremely promising in terms of performance and accuracy in both training and testing datasets.

Part (f): The Alpha and Beta Values

- **The Alpha and Beta Values**

The **Slope / Coefficients** and the **Intercept** signifies important parameters in a regression model. The Slope values (1.22549343 -0.22610689) indicate how changes in each input feature will affect the predicted price and the Intercept (0.0274723) depicts the predicted price when both input features are zero. These values reflect on the relationship that the input features have with the oil prices. Below are the alpha and beta values presented,

```
# Print the model coefficients
print("Slope:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Slope: [[ 1.22549343 -0.22610689]]
Intercept: [0.0274723]
```

Figure 11: Alpha and Beta Value Calculation

α = 0.0274723 **(Intercept)**

β = (1.22549343 -0.22610689) **(Slope / Coefficients)**

- **The Multiple Linear Regression Formula**

$$Y = 0.0274723 + 1.22549343\ X_1 - 0.22610689\ X_2$$

Once the alpha and beta values are calculated, it is only a matter of replacing them in the Multiple Linear Regression Equation next to the Input features **(X1 & X2 / MA3 & MA9)** to get the complete formula to the model.

References

EuropeanCentralBank, 2012. The Development of Prices and Costs During the 2008-09 Recession. *European Central Bank Monthly Bulletin*, pp. 71-85.

Sun, Y., 2022. *The Impacts of Wars on Oil Prices.* s.l., Atlantis Press.

WorldBank, 2018. With the Benefit of Hindsight: The Impact of the 2014-16 Oil Price Collapse. *Global Economic Prospects: Broad-Based Upturn, but for How Long?*, pp. 51-72.

The End.