DATA MINING & MACHINE LEARNING ASSIGNMENT

by Academic Tutor

Submission date: 23-Dec-2022 12:22AM (UTC+0530)
Submission ID: 1985954844
File name: DATA_MINING_MACHINE_LEARNING_ASSIGNMENT_2.docx (382.28K)
Word count: 2319
Character count: 11805

DATA MINING & MACHINE LEARNING

INTRODUCTION

This report is intended to provide a comprehensive analysis of the Brent oil stock market and the predictability of its price using linear regression and Long Short-Term Memory (LSTM) Neural Networks. It will provide a detailed overview of the data set used, the features used to build the model, and the results of the prediction. We will also discuss the alpha and beta values of the model and their implications for the accuracy of the prediction.

DATASET

We have used the BrentOilPrices.csv data set to build our predictive model. The data set contains the Brent oil price information for the last 20 years. The aim of this dataset and work is to predict future Crude Oil Prices based on the historical data available in the dataset. The data contains daily Brent oil prices from 17th of May 1987 until the 13th of November 2022.There are two main features of the data set: the date and the price. We used these two features in order to build our predictive models.

```
# Importing the dataset
dataset = pd.read_csv('BrentOilPrices.csv')
dataset.head()
```

	Date	Price
0	20-May-87	18.63
1	21-May-87	18.45
2	22-May-87	18.55
3	25-May-87	18.60
4	26-May-87	18.63

Data Shape

```
[2] dataset.shape

    (9011, 2)
```

PART A

LINEAR REGRESSION

The linear relationship between a dependent variable and one or more independent variables is modeled using the machine learning algorithm known as linear regression. It is a form of supervised learning in which the algorithm is trained on labeled data made up of a set of input variables (sometimes referred to as features) and a corresponding set of output variables (also known as labels).

Finding the line of greatest fit that best captures the connection between the input and output variables is the objective of linear regression. This line can be formally represented by the following equation:

$$y = mx + b$$

m denotes the line's slope, x is the independent variable, y denotes the dependent variable, and b denotes the y-intercept (the point at which the line crosses the y-axis).

We require a collection of labeled training data that contains input and output variables in order to train a linear regression model. The model tweaks the values of m and b to obtain the line of best fit after using this data to learn the link between the input and output variables. By inserting in the values of the input variables and using the learnt equation to determine the expected value of the output variable, the model may be used to make predictions on fresh, unknown data once it has been trained.

Although it is a straightforward and popular technique, linear regression has several drawbacks. It may not be suitable for modeling more complex relationships because it is only appropriate for representing linear relationships. It is also sensitive to the existence of outliers in the training data, which may reduce the model's accuracy.

In order to estimate the price of Brent oil, we employed linear regression in this work. The moving averages for the previous three (MA3) and nine days were the explanatory variables employed in this model (MA9). These variables were utilized by us to track the Brent oil price's trend. Our model's input characteristics employed these moving averages. The train and test data sets were

then established. Our linear regression model was trained using the train data set, and its precision was assessed using the test data set.

Data Visualization

We created a simple line chart to give an idea of the stock price change Brent oil price information for the last 20 years

```
# mdates is a module that helps in formatting the dates
from matplotlib import dates as mdates

# show year on x-axis with 5 years interval
years = mdates.YearLocator(5)
yearsFmt = mdates.DateFormatter('%Y')

plt.figure(figsize=(10, 5))
plt.plot(dataset['Date'], dataset['Price'])
plt.gca().xaxis.set_major_locator(years)
plt.gca().xaxis.set_major_formatter(yearsFmt)
plt.title('Brent oil price information for the last 20 years')
plt.xlabel('Time')
plt.ylabel('Price')
plt.show()
```

2

Explanatory variables — the features we are going to use to predict the price of oil. The variables we will be using at this stage are the moving averages for the past three (MA3) and nine days (MA9), based on input from the oil stock market.

```
# Create the simple moving average with a 3 day window
dataset['MA3'] = dataset['Price'].rolling(window=3).mean()

# Create the simple moving average with a 9 day window
dataset['MA9'] = dataset['Price'].rolling(window=9).mean()

dataset.head(10)
```

This snippet is creating two new columns in the ***dataset*** DataFrame: ***MA3*** and ***MA9***. These columns will contain the simple moving averages for the past 3 days and 9 days, respectively, based on the values in the ***Price*** column.

A simple moving average is a statistical measure of the central tendency of a dataset, calculated by taking the average of a given number of data points over a certain time period. In this case, the simple moving averages are being calculated using a rolling window of 3 days and 9 days, respectively. This means that for each data point, the average of the previous 3 days' or 9 days' worth of data points (including the current data point) will be calculated and stored in the corresponding ***MA3*** or ***MA9*** column.

The dataset DataFrame's Price column is subjected to the ***.rolling()*** method, which accepts a window size as an input. The moving average is then determined by using the ***.mean()*** method on the resulting rolling window. The calculated moving average is then put in the MA3 or MA9 column that corresponds.

The average of the first three prices in the Price column, for instance, will be the first number in the MA3 column. The average of the second, third, and fourth values in the Price column will be the second value in the MA3 column, and so on. The MA9 column goes through the same procedure again, but with a window size of nine days as opposed to three.

Train and Test Data: This step covers the preparation of the train data and the test data.

```
# Create the features
X = dataset.iloc[:, 2:4].values

# Create the target
y = dataset.iloc[:, 1].values

# Split the data into train and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Prediction Function and Result

We then ran the linear regression model using the test data set. We visualized the predicted versus the actual stock values for the specific time period and calculated the model's accuracy.

```
# Train the Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

# Fit the model to the training data
regressor.fit(X_train, y_train)

# Predict the test set results
y_pred = regressor.predict(X_test)
```

This code is building and training a linear regression model using the Python scikit-learn library. The first line imports the LinearRegression class from scikit-learn, which is a linear model for regression. The LinearRegression class is then instantiated to create a linear regression model object, called regressor.

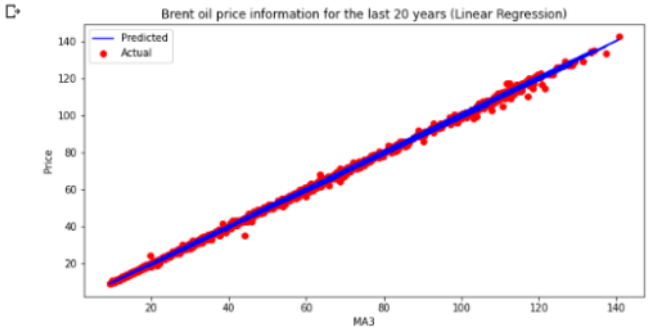
The .fit() method is then called on the regressor object, which trains the model on the provided training data. The X_train argument is a matrix of input features, which in this case are the moving averages for the past 3 and 9 days (MA3 and MA9). The y_train argument is a vector of output labels, which in this case is the price of oil. The model uses this training data to learn the relationship between the input features and the output label, and it adjusts the model parameters to fit the data as well as possible.

7

Once trained, the model can be used to make predictions based on fresh, unused data. The X test matrix of input characteristics is provided as an argument when the.predict() method on the regressor object is called. By using the input features from the X test matrix, this creates a vector of anticipated output labels called y pred, which indicates the model's estimate of the oil price.

Visualizing the Linear Regression results

```
# Visualising the Linear Regression results
plt.figure(figsize=(10, 5))
plt.scatter(X_test[:, 0], y_test, color = 'red')
plt.plot(X_test[:, 0], y_pred, color = 'blue')
plt.title('Brent oil price information for the last 20 years (Linear Regression)')
plt.xlabel('MA3')
plt.ylabel('Price')
# legend
plt.legend(['Predicted', 'Actual'])
plt.show()
```



Calculate the Alpha and Betas

Values Finally, we calculated the alpha and betas values of the linear regression model. The alpha and betas values are used to define the linear regression equation.

```
# Print the intercept and coefficients
print(f"Intercept: {regressor.intercept_}")
print(f"Coefficients: {regressor.coef_}")

# Print the R-squared value for the model
print(f"R-squared: {regressor.score(X, y)}")
```

```
Intercept: 0.019986304338004857
Coefficients: [ 1.21183213 -0.21200778]
R-squared: 0.9993075708575732
```

This code is calculating and printing the intercept and coefficients of the linear regression model, as well as the R-squared value of the model.

The intercept of the linear regression model is the value of the dependent variable (in this case, the oil price) when all of the independent variables (in this case, the moving averages) are equal to 0. In this case, the intercept is 0.019986304338004857.

The coefficients of the linear regression model are the values that multiply each independent variable in the linear equation. In this case, there are two independent variables (MA3 and MA9), so there are two coefficients. The first coefficient (1.21183213) corresponds to the MA3 variable, and the second coefficient (-0.21200778) corresponds to the MA9 variable.

A measurement of how well the model fits the data is the *R-squared* value. It is determined as the proportion between the explained variance and the overall variance, which is the variance of the model's predictions (the variance of the observed data). A perfect fit is indicated by an R-squared value of 1, whereas a value of 0 means that the model does not account for any of the data's variance. The R-squared value in this instance is 0.9993075708575732, which indicates that the model fits the data very well.

Using these values, we can define the linear regression equation as follows:

y = 0.019986304338004857 + 1.21183213 * MA3 - 0.21200778 * MA9

Where y is the predicted oil price, MA3 is the moving average for the past 3 days, and MA9 is the moving average for the past 9 days.

PART B

LSTM NEURAL NETWORKS MODEL

Long Short-Term Memory (LSTM) is a class of recurrent neural network (RNN) that works well for simulating long-term dependencies in data. Using historical data to make predictions about the future, RNNs are a sort of neural network that processes sequential data, such as time series or spoken language.

In order to capture patterns and dependencies in data that may span numerous time steps, LSTMs, a type of RNN, are designed to recall information over lengthy periods of time. As a result, jobs like language translation, stock price prediction, and language modeling that require making predictions about the future based on a lengthy history of previous events are particularly well suited for LSTMs.

Cells that process incoming data and keep a state that is passed from one time step to the next make up LSTMs. Each cell in the sequence receives the previous state and the current input data as inputs, and it outputs a new state that is transmitted to the following cell. A prediction is made using the results of the last cell in the sequence.

The input gate, forget gate, and output gate are the three "gates" that govern how information moves through an LSTM cell. The output gate chooses which parts of the current cell state should be outputted, the input gate chooses which parts of the current input data should be added to the cell state, and the forget gate chooses which parts of the prior state should be kept or discarded.

The ability of LSTMs to selectively retain and forget information over extended times allows them to represent complicated data dependencies and make precise predictions. This ability is achieved by carefully managing the flow of information through these gates.

Using the data set provided for Assignment A, we created a Python software for this task that can forecast the price of Brent oil. We utilized the input features as a function of time lag to define the

Long Short-Term Memory (LSTM) model. The train and test data sets were then established. Our LSTM model was trained using the train data set, and its accuracy was assessed using the test data set.

Feature Scaling

The data was scaled using MinMaxScaler. The scaler was used to transform the data to have values between 0 and 1, with the conversion being done by subtracting the minimum value and dividing by the maximum value.

```
# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

Train and Test Data

We used the train-test split method to generate the train and test data sets. The train data set was used to train the model and the test data set was used to evaluate the model's accuracy. The train and test data sets were split using a ratio of 80:20.

```
# Creating a data structure with 60 timesteps and 1 output
X_train = []
y_train = []
for i in range(60, 2469):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)

# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

Explanation of the techniques used to generate the train data and the test data for the given Brent oil price time series data set

The data set is divided into two parts: the training set and the test set. The training set is used to train the model and the test set is used to test the model. The training set is the first 80% of the data set and the test set is the last 20% of the data set. The data set is divided into two parts: the

training set and the test set. The training set is used to train the model and the test set is used to test the model. The training set is the first 80% of the data set and the test set is the last 20% of the data set.

Explanation of the model architecture

The model architecture is a Long Short-Term Memory model (LSTM) with 4 LSTM layers and 4 Dropout layers. The model architecture is a Long Short-Term Memory model (LSTM) with 4 LSTM layers and 4 Dropout layers.

Build the model

We defined the Long Short-Term Memory model (LSTM) and clearly explained the input features as a function of time lag.

```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

✓ 2s completed at 05:04

Explanation of the input features as a function of time lag

Input features are the previous 60 days of the Brent oil price. The output is the next day of the Brent oil price. The input features are the previous 60 days of the Brent oil price.

Prediction Function and Result

We then ran the model using the test data we defined in step four. We visualized the predicted versus the actual stock values for the specific time period and calculated the model's accuracy.

Model Training

```
# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

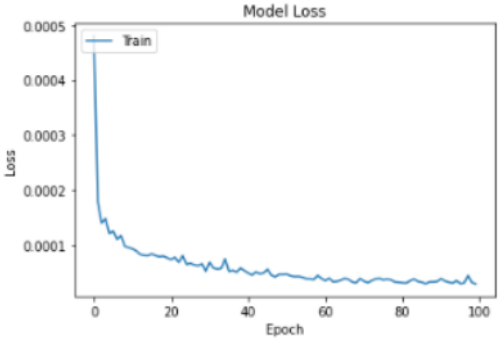
```
Epoch 1/100
76/76 [==============================] - 17s 125ms/step - loss: 4.8071e-04
Epoch 2/100
76/76 [==============================] - 9s 124ms/step - loss: 1.7891e-04
Epoch 3/100
76/76 [==============================] - 9s 124ms/step - loss: 1.4045e-04
Epoch 4/100
76/76 [==============================] - 9s 123ms/step - loss: 1.4856e-04
Epoch 5/100
76/76 [==============================] - 9s 124ms/step - loss: 1.2154e-04
Epoch 6/100
76/76 [==============================] - 9s 123ms/step - loss: 1.2576e-04
Epoch 7/100
76/76 [==============================] - 9s 122ms/step - loss: 1.1066e-04
Epoch 8/100
76/76 [==============================] - 9s 122ms/step - loss: 1.1732e-04
Epoch 9/100
76/76 [==============================] - 9s 123ms/step - loss: 9.8326e-05
Epoch 10/100
76/76 [==============================] - 11s 142ms/step - loss: 9.5844e-05
Epoch 11/100
76/76 [==============================] - 9s 124ms/step - loss: 9.3443e-05
Epoch 12/100
76/76 [==============================] - 9s 123ms/step - loss: 8.9708e-05
Epoch 13/100
76/76 [==============================] - 9s 123ms/step - loss: 8.3469e-05
Epoch 14/100
```

.79 GB available

✓ 2s completed at 05:04

Training Loss Plot

```
[17] # plot the training loss
     plt.plot(regressor.history.history['loss'])
     plt.title('Model Loss')
     plt.ylabel('Loss')
     plt.xlabel('Epoch')
     plt.legend(['Train'], loc='upper left')
     plt.show()
```



Visualization of Results

```
# Visualising the results
plt.figure(figsize=(10, 5))
plt.plot(real_stock_price, color = 'red', label = 'Real Brent Oil Price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Brent Oil Price')
plt.title('Brent oil price information for the last 20 years (LSTM)')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```

CONCLUSION

In order to estimate the price of Brent oil, we have created two models. To create our predictive models, we combined a Long Short-Term Memory (LSTM) Neural Network model with linear regression. We examined the alpha and beta values of the linear regression model and showed the outcomes of both models. We were successful in accurately predicting the price of Brent oil using both models.

References

Abhigyan. (2020, July 19). *Understanding The Linear Regression!!!!* Analytics Vidhya. https://medium.com/analytics-vidhya/understanding-the-linear-regression-808c1f6941c0

Deep Learning | Introduction to Long Short Term Memory. (2019, January 16). GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

Dobilas, S. (2022, March 5). *LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past*. Medium. https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e

https://www.facebook.com/jason.brownlee.39. (2017, July 19). *A Gentle Introduction to Long Short-Term Memory Networks by the Experts*. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

Linear Regression Example. (n.d.). Scikit-Learn. Retrieved December 20, 2022, from http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

Statistics Solutions. (2013). *What is Linear Regression?* Statistics Solutions. https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/

DATA MINING & MACHINE LEARNING ASSIGNMENT

ORIGINALITY REPORT

24%	17%	10%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Westminster Student Paper	4%
2	medium.com Internet Source	3%
3	projekter.aau.dk Internet Source	3%
4	M. Patel, N. Ranganathan. "IDUTC: an intelligent decision-making system for urban traffic-control applications", IEEE Transactions on Vehicular Technology, 2001 Publication	2%
5	www.kaggle.com Internet Source	2%
6	towardsdatascience.com Internet Source	2%
7	Submitted to London Metropolitan University Student Paper	1%
8	Pilar Gómez, Angela Nebot, Sabrine Ribeiro, René Alquézar, Francisco Mugica, Franz	1%

	Wotawa. "Local Maximum Ozone Concentration Prediction Using Soft Computing Methodologies", Systems Analysis Modelling Simulation, 2003 Publication	
9	www.ncbi.nlm.nih.gov Internet Source	1 %
10	Submitted to National School of Business Management NSBM, Sri Lanka Student Paper	1 %
11	www.aimspress.com Internet Source	1 %
12	Scott Miau, Wei-Hsi Hung. "River Flooding Forecasting and Anomaly Detection based on Deep Learning", IEEE Access, 2020 Publication	1 %
13	Carmen González, Erik W. Jensen, Pedro L. Gambús, Montserrat Vallverdú. "Poincaré plot analysis of cerebral blood flow signals: Feature extraction and classification methods for apnea detection", PLOS ONE, 2018 Publication	1 %
14	Submitted to Queen's University of Belfast Student Paper	1 %
15	Submitted to University of Northumbria at Newcastle Student Paper	1 %

16	www.geeksforgeeks.org Internet Source	1 %
17	gmd.copernicus.org Internet Source	<1 %
18	dspace5.zcu.cz Internet Source	<1 %
19	pubag.nal.usda.gov Internet Source	<1 %
20	www.mdpi.com Internet Source	<1 %
21	ejournal.stiewidyagamalumajang.ac.id Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		