STOCK PRICE PREDICTION - Phase 3

Madras Institute of Technology

2021506061 - Pavithra S

2021506045 - Mahamudha Begam A

2021506025 - Gowri R

2021506037 - Kavisri S

INTRODUCTION

This phase aims to clean, transform, and engineer features in a way that maximizes the model's ability to capture patterns and make accurate predictions. Through careful data preparation and feature engineering, we enhance the quality of input fed into the models. Effective preprocessing lays the foundation for improved predictive models. **The given dataset has been pre-processed and the outputs are attached with snap shots.**

1.IMPORTING LIBRARIES

Importing libraries in pre-processing is a crucial step in data analysis and machine learning. These libraries, such as NumPy, Pandas, and scikit-learn in Python, provide essential tools and functions for tasks like data manipulation, cleaning, and feature engineering before applying machine learning algorithms.

```
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

2.UPLOADING FILE

Uploading files in pre-processing enables subsequent data cleaning, transformation, and analysis as part of a broader data processing pipeline.

```
from google.colab import files
uploadf = files.upload()
```

3.LOADING THE DATA

Loading data involves reading raw data from various sources such as files or databases and converting it into a format suitable for further analysis.

```
df = pd.read_csv('MSFT.csv')
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.063107	47894400
...	...	...	...	...	...	...	...
8520	2019-12-31	156.770004	157.770004	156.449997	157.699997	157.699997	18369400
8521	2020-01-02	158.779999	160.729996	158.330002	160.619995	160.619995	22622100
8522	2020-01-03	158.320007	159.949997	158.059998	158.619995	158.619995	21116200
8523	2020-01-06	157.080002	159.100006	156.509995	159.029999	159.029999	20813700
8524	2020-01-07	159.320007	159.669998	157.330002	157.580002	157.580002	18017762

8525 rows × 7 columns

4.DATA OUTLIERS

Detecting and handling outliers is essential to ensure the quality and reliability of data analysis and machine learning models.

```
thresholds ={'Date': ("1986-03-19","2020-01-02")}
for col, (lower,upper) in thresholds.items():
  df = df[(df[col] >= lower) & (df[col] <= upper)]
print(df)
```

```
            Date        Open        High         Low       Close   Adj Close  \
4     1986-03-19    0.099826    0.100694    0.097222    0.098090    0.063107
5     1986-03-20    0.098090    0.098090    0.094618    0.095486    0.061432
6     1986-03-21    0.095486    0.097222    0.091146    0.092882    0.059756
7     1986-03-24    0.092882    0.092882    0.089410    0.090278    0.058081
8     1986-03-25    0.090278    0.092014    0.089410    0.092014    0.059198
...          ...         ...         ...         ...         ...         ...
8517  2019-12-26  157.559998  158.729996  157.399994  158.669998  158.669998
8518  2019-12-27  159.449997  159.550003  158.220001  158.960007  158.960007
8519  2019-12-30  158.990005  159.020004  156.729996  157.589996  157.589996
8520  2019-12-31  156.770004  157.770004  156.449997  157.699997  157.699997
8521  2020-01-02  158.779999  160.729996  158.330002  160.619995  160.619995

        Volume
4     47894400
5     58435200
6     59990400
7     65289600
8     32083200
...        ...
8517  14520600
8518  18412800
8519  16348400
8520  18369400
8521  22622100

[8518 rows x 7 columns]
```

5.CHECKING MISSING VALUES

The `isnull()` function is a common method used in data preprocessing to identify missing values in a dataset. It returns a Boolean value (True or False) for each data point, indicating whether the value is missing (True) or not (False), allowing data analysts to effectively handle or impute missing data during data cleaning and preparation.

```
missing_values = df.isnull().sum()
print(missing_values)
```

```
Date          0
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
dtype: int64
```

6.FILTERING THE DATA

Filtering data in pre-processing involves selecting and retaining specific information or removing unwanted elements from a dataset to improve its quality and relevance for subsequent analysis. This process helps in reducing noise, enhancing signal-to-noise ratios, and focusing on the most important aspects of the data.

```
#create a new dataframe with only the 'Close column
data = df.filter(['Close'])
#Convert the dataframe to a numpy array
dataset = data.values
#Get the number of rows to train model on
training_data_len = math.ceil(len(dataset) * .8 )

training_data_len
```

```
6815
```

7.SCALING AND NORMALIZING THE DATA

Scaling ensures that all features have the same range, preventing some features from dominating others, while normalization transforms the data to have a consistent mean and standard deviation, making it suitable for algorithms that rely on feature similarity or distance measures.

```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

```
array([[4.86638869e-05],
       [3.24425913e-05],
       [1.62212956e-05],
       ...,
       [9.81124996e-01],
       [9.81810234e-01],
       [1.00000000e+00]])
```

8.CREATING TRAINING DATASET

Creating a trained dataset involves collecting and preparing a structured collection of data, often by cleaning, formatting, and labeling it, to serve as the input for machine learning algorithms. This dataset should reflect the problem domain and be sufficiently representative to enable effective model training and evaluation.

```
#Create the traning dataset
#Create the scaled training dataset
train_data = scaled_data[0:training_data_len , :]
#Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
  x_train.append(train_data[i-60:i, 0])
  y_train.append(train_data[i, 0])
  if i<= 60:
    print(x_train)
    print(y_train)
    print()
```

```
[array([4.86638869e-05, 3.24425913e-05, 1.62212956e-05, 0.00000000e+00,
       1.08141971e-05, 2.70354927e-05, 3.78496898e-05, 3.24425913e-05,
       2.70354927e-05, 3.24425913e-05, 3.78496898e-05, 3.78496898e-05,
       2.70354927e-05, 3.24425913e-05, 4.32567884e-05, 4.86638869e-05,
       5.94780840e-05, 6.48851826e-05, 6.48851826e-05, 8.65198062e-05,
       9.19269047e-05, 7.02985105e-05, 7.02985105e-05, 5.94780840e-05,
       6.21816333e-05, 1.24369496e-04, 1.67632514e-04, 1.73039612e-04,
       1.51404989e-04, 1.35183693e-04, 1.24369496e-04, 1.24369496e-04,
       1.18962397e-04, 1.24369496e-04, 1.24369496e-04, 1.29776595e-04,
       1.24369496e-04, 1.29776595e-04, 1.35183693e-04, 1.29776595e-04,
       1.29776595e-04, 1.35183693e-04, 1.24369496e-04, 1.18962397e-04,
       1.08148200e-04, 1.08148200e-04, 1.08148200e-04, 1.29776595e-04,
       1.51404989e-04, 1.67632514e-04, 1.94668007e-04, 1.73039612e-04,
       1.73039612e-04, 1.67632514e-04, 1.78446711e-04, 1.78446711e-04,
       1.45997890e-04, 1.45997890e-04, 1.45997890e-04, 1.18962397e-04])]
[0.00011896239747311097]
```

9.CONVERTING AS NUMPY ARRAYS

Converting a trained dataset into numpy arrays during preprocessing involves transforming the raw data into a structured format for machine learning. Numpy arrays provide efficient data storage and manipulation capabilities, making them ideal for tasks like feature extraction and data preparation.

```
# Convert x_train and y_train to numpy arrays
x_train = np.array(x_train)
y_train = np.array(y_train)

# Reshape x_train for use in an LSTM model
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))


# Print the shape to confirm
print("x_train shape:", x_train.shape)
print("y_train shape:", y_train.shape)
```

```
x_train shape: (6755, 60, 1)
y_train shape: (6755,)
```

10.CREATING A TESTING DATASET

Creating a testing dataset during pre-processing is essential for evaluating the performance of a machine learning model. Typically, a portion of the original dataset is set aside, ensuring it's not used for training, to assess the model's generalization and accuracy.

```
#create testing dataset
#create a new containing scaled values
test_data = scaled_data[training_data_len - 60: , :]
#create the datasets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
  x_test.append(test_data[i-60:i, 0])
```

11.RESHAPING THE DATA

Reshaping data involves transforming the raw input into a format suitable for analysis or modeling.

```
#reshape the data
x_test = np.array(x_train)

# Reshape x_train for use in an LSTM model
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

CONCLUSION

In the third phase, the dataset has been preprocessed, which is fundamental to building accurate and reliable predictive models. This involved handling missing values, scaling, encoding categorical features, and possibly applying other transformations like feature engineering or selection. The preprocessed dataset is now ready for the subsequent phases, where it will be utilized to train and validate models.