Stock Price Prediction - Data Preprocessing

O.Importing the required libraries:

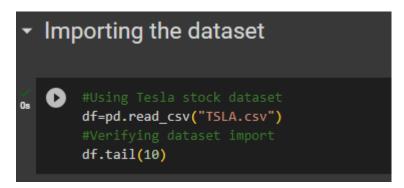
```
importing Necessary Libraries

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn import model_selection
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

All the above libraries are to be used to perform stock price predictions on the given dataset and hence they are imported to be used in this notebook

1. Importing the Dataset

In this step we are going to import the dataset that is going to be used across this project



Using the panadas library we are able to read data from the dataset with the title "TSLA.csv" and this is then saved to the "df" variable for further use, which yields the following output.

∃		Date	0pen	High	Low	Close	Adj Close	Volume	
	1500	2020-12-16	209.410004	210.833328	201.666672	207.589996	207.589996	126287400	11.
	1501	2020-12-17	209.396667	219.606674	206.500000	218.633331	218.633331	168810300	
	1502	2020-12-18	222.966660	231.666672	209.513336	231.666672	231.666672	666378600	
	1503	2020-12-21	222.080002	222.833328	215.356674	216.619995	216.619995	174135900	
	1504	2020-12-22	216.000000	216.626663	204.743332	213.446671	213.446671	155148000	
	1505	2020-12-23	210.733337	217.166672	207.523331	215.326660	215.326660	99519000	
	1506	2020-12-24	214.330002	222.029999	213.666672	220.589996	220.589996	68596800	
	1507	2020-12-28	224.836670	227.133331	220.266663	221.229996	221.229996	96835800	
	1508	2020-12-29	220.333328	223.300003	218.333328	221.996674	221.996674	68732400	
	1509	2020-12-30	224.000000	232.199997	222.786667	231.593338	231.593338	128538000	

2. Dataset Summary

This step provides a quick statistical summary of the dataset, including count, mean, standard deviation, minimum, and maximum values for each column. Following is the code snippet of this section.

```
#Preprocessing Data
print(df.describe())
print("---"*10)
```

With the following output:

	0pen	High	Low	Close	Adj Close	Volume				
count	1510.000000	1510.000000	1510.000000	1510.000000	1510.000000	1.510000e+03				
mean	30.921814	31.621175	30.198765	30.976507	30.976507	1.203075e+08				
std	36.948396	37.993719	35.848150	37.139818	37.139818	8.891357e+07				
min	9.488000	10.331333	9.403333	9.578000	9.578000	1.062000e+07				
25%	15.179500	15.391334	14.938667	15.139667	15.139667	6.385538e+07				
50%	18.877334	19.214666	18.497333	18.944000	18.944000	9.240750e+07				
75%	23.163501	23.485835	22.814500	23.162333	23.162333	1.432407e+08				
max	224.836670	232.199997	222.786667	231.666672	231.666672	9.140820e+08				

3. Date Column Data Type Conversion

This step converts the 'Date' column to a datetime datatype, which is important for time series analysis. Following is the code snippet of this section.

```
#Changing the values in the Date Column to datetime datatype
df['Date'] = pd.to_datetime(df['Date'])
print(df.info())
```

With the following output:

4. Checking for Missing Values

This step helps identify and handle missing values in the dataset. Depending on the amount and nature of missing data, you may choose to remove or impute missing values. Following is the code snippet of this section:

```
#Checking for missing values
missing_vals = df.isnull().sum()
print(missing_vals)
```

With the following output:

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

5. Data Splitting

This step splits the dataset into training and testing sets. The exact splitting ratio (here, 800 data points for training) can be adjusted to suit your needs. Following is the code snippet of this section:

```
#Splitting the dataset into training and test
training = df.iloc[:800, 1:2].values
testing = df.iloc[800:, 1:2].values
```

6. Data Scaling with Min-Max Scaling

This step scales the data to a range between 0 and 1. LSTM models often perform better when data is scaled to a specific range, as they are sensitive to input scale. Following is the code snippet of this section:

```
#Since LSTM is senstivite to Scale, We use MinMax to standardize the scale of the dataset sc = MinMaxScaler(feature_range = (0, 1)) training_scaled = sc.fit_transform(training)
```

These are the preprocessing steps applied to the dataset to prepare it for stock price prediction using an LSTM model. Additional steps related to model development, training, and evaluation can be performed after these preprocessing steps.