Stock Price Prediction using Machine Learning in Python

Importing Libraries

Python libraries make it very easy for us to handle the data and perform typical and complex tasks with a single line of code.

Pandas – This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.

Numpy – Numpy arrays are very fast and can perform large computations in a very short time.

Matplotlib/Seaborn – This library is used to draw visualizations.

Sklearn – This module contains multiple libraries having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

XGBoost – This contains the eXtreme Gradient Boosting machine learning algorithm which is one of the algorithms which helps us to achieve high accuracy on predictions.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

https://www.kaggle.com/datasets/timoboz/tesla-stock-data-from-2010-to-2020

df = pd.read_csv('/content/Tesla.csv')
df.head()

Output:

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001

df.shape

Output:

(1692,7)

df.describe()

Output:

	Open	High	Low	Close	Volume	Adj Close
count	1692.000000	1692.000000	1692.000000	1692.000000	1.692000e+03	1692.000000
mean	132.441572	134.769698	129.996223	132.428658	4.270741e+06	132.428658
std	94.309923	95.694914	92.855227	94.313187	4.295971e+06	94.313187
min	16.139999	16.629999	14.980000	15.800000	1.185000e+05	15.800000
25%	30.000000	30.650000	29.215000	29.884999	1.194350e+06	29.884999
50%	156.334999	162.370002	153.150002	158.160004	3.180700e+06	158.160004
75%	220.557495	224.099999	217.119999	220.022503	5.662100e+06	220.022503
max	287.670013	291.420013	280.399994	286.040009	3.716390e+07	286.040009

df.info()

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1692 entries, 0 to 1691
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1692 non-null   object
 1   Open       1692 non-null   float64
 2   High       1692 non-null   float64
 3   Low        1692 non-null   float64
 4   Close      1692 non-null   float64
 5   Volume     1692 non-null   int64
 6   Adj Close  1692 non-null   float64
dtypes: float64(5), int64(1), object(1)
memory usage: 92.7+ KB
```

Exploratory Data Analysis

plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Tesla Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()

Output:

df.head()

Output:

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001

df[df['Close'] == df['Adj Close']].shape

Output:

(1692,7)

df = df.drop(['Adj Close'], axis=1)

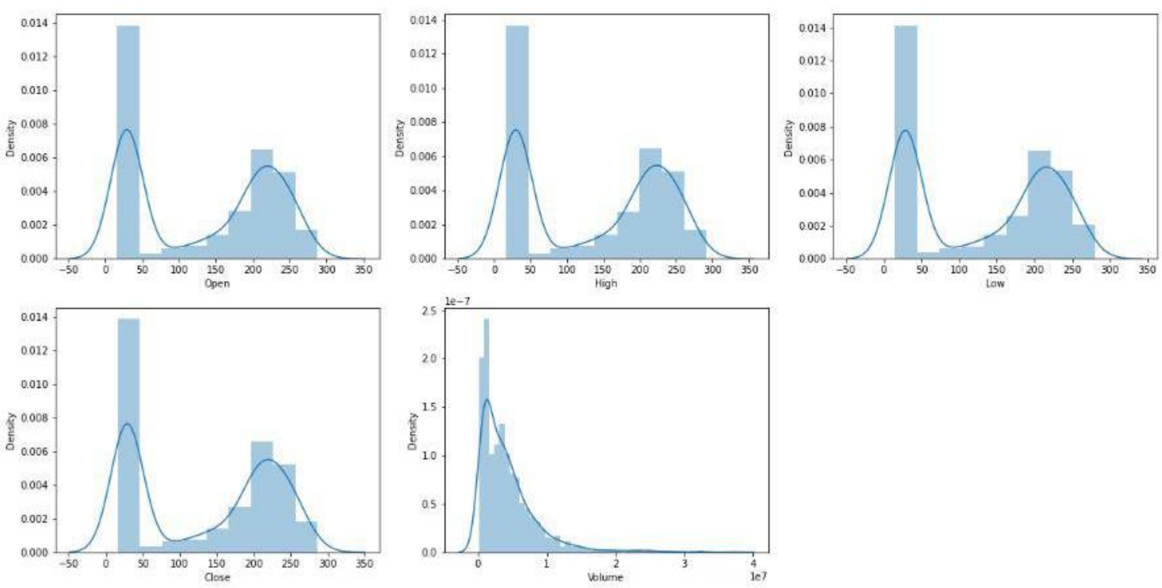
df.isnull().sum()

Output:

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

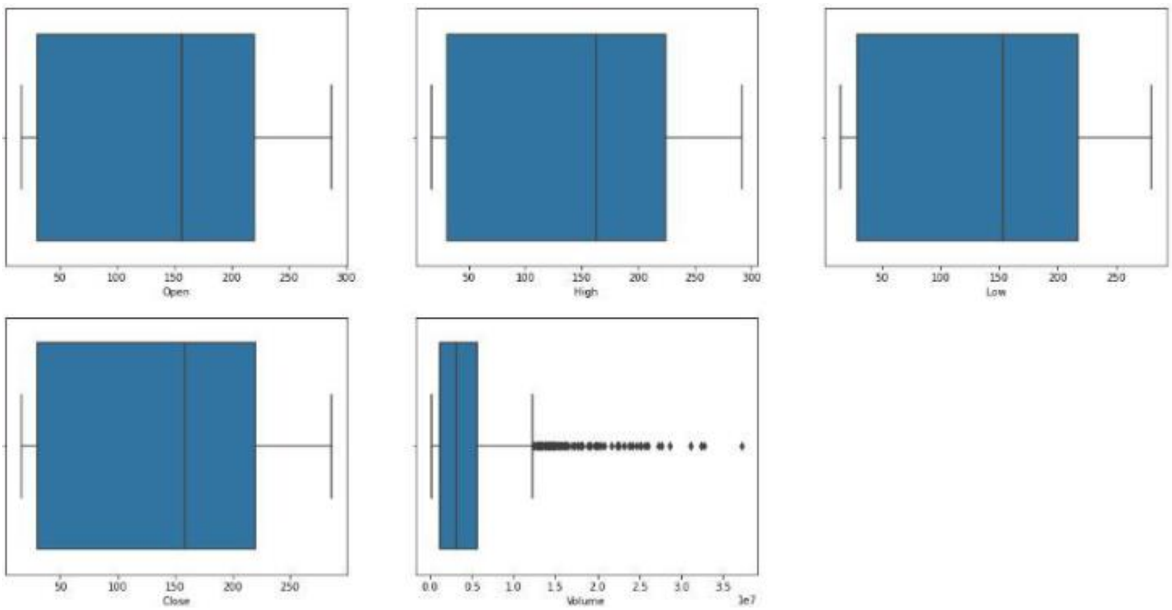
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
plt.subplot(2,3,i+1)
sb.distplot(df[col])
plt.show()
```

Output:




```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
plt.subplot(2,3,i+1)
sb.boxplot(df[col])
plt.show()
```

Output:



Feature Engineering

splitted = df['Date'].str.split('/', expand=True)

df['day'] = splitted[1].astype('int')
df['month'] = splitted[0].astype('int')
df['year'] = splitted[2].astype('int')

df.head()

Output:

	Date	Open	High	Low	Close	Volume	day	month	year
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	29	6	2010
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	30	6	2010
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	1	7	2010
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	2	7	2010
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	6	7	2010

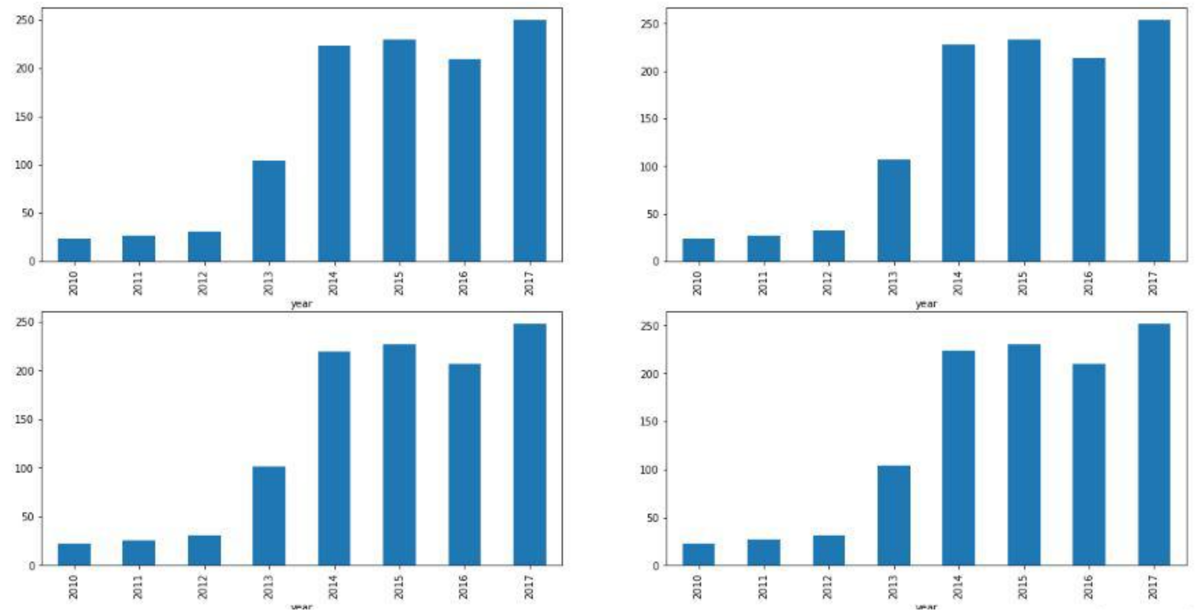
df['is_quarter_end'] = np.where(df['month']%3==0,1,0)
df.head()

Output:

	Date	Open	High	Low	Close	Volume	day	month	year	is_quarter_end
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	29	6	2010	1
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	30	6	2010	1
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	1	7	2010	0
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	2	7	2010	0
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	6	7	2010	0

```
data_grouped = df.groupby('year').mean()
plt.subplots(figsize=(20,10))

for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
plt.subplot(2,2,i+1)
data_grouped[col].plot.bar()
plt.show()
```

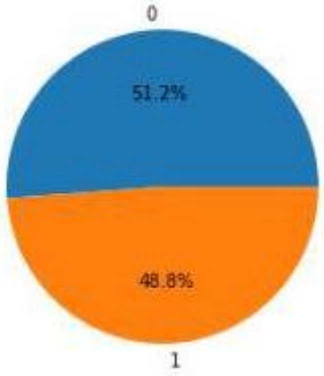



```
df.groupby('is_quarter_end').mean()
```

Output:

	Open	High	Low	Close	Volume	day	month	year
is_quarter_end								
0	130.813739	133.182620	128.257229	130.797709	4.461581e+06	15.686501	6.141208	2013.353464
1	135.679982	137.927032	133.455777	135.673269	3.891084e+06	15.657244	7.584806	2013.314488

```
df['open-close'] = df['Open'] - df['Close']
df['low-high'] = df['Low'] - df['High']
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
plt.pie(df['target'].value_counts().values,
                    labels=[0, 1], autopct='%1.1f%%')
plt.show()
```


0
51.2%
48.8%
1