Project submission Phase-2-Innovation

**Stock Price Prediction**

**INTRODUCTION:**

A market where shares are publicly issued and traded is known as a share market.

Implementing the concept of algorithmic trading, which uses automated, pre-programmed trading strategies to predict stock prices. Time series forecasting(predicting future values based on historical values) applies well to stock forecasting. Developed a User Interface

**FUTURE WORK:**

Machine learning and data science is a game changer in this domain so there is a lot of data to find patterns in for predicting with high degree of accuracy.

In future we’ll try to predict the values based on multiple factors such as politics, globa economic conditions, unexpected events like covid, companies financial performance, and so on.

We are going to implement multiple types of algorithms because different types of data requires different types of techniques.

Decided to implement simple User Interface to operate this whole precoess for users so to make people engage in Stock Market.

**STEPS PERFORMED:**

1. Importing and Cleaning data
2. Split the Data into training/test sets
3. Creating and Training the Model
4. Making Predictions
5. Evaluating and Improving Predictions

**NEED OF PROJECT:**

* The stock market is known for being volatile, dynamic,& non linear
* Accurate stock price prediction is extremely challenging because of multiple factors.
* But, all of this also means that there is a lot of data to find patterns in.
* So, we keep exploring analytics techniques to detect stock market trends.
* So, they can be analysed as a sequence of discrete time data
* Despite the volatility, stock prices are not just randomly generated numbers.

**METHODOLOGY:**

PYTHON Language:

Python is a rich language for Data Science and AI

Libraries:

Pandas, Numpy, Sklearn, Tensorflow, etc

Algorithm:

Long Short Term Memory(LSTM)

Streamlit UI:

Provided User Interface using Streamlit

**CODE:**

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 metric

import warnings

warnings.filterwarnings('ignore')

df = pd.read\_csv('/content/MSFT.csv')

**OUTPUT:**

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 13-03-1986 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 |
| **1** | 14-03-1986 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 |
| **2** | 17-03-1986 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 |
| **3** | 18-03-1986 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 |
| **4** | 19-03-1986 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 |
|  |  |  |  |  |  |  |  |

**CODE:**

df.shape

**OUTPUT:**

(8525, 7)

**CODE:**

df.describe()

**OUTPUT:**

|  | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- |
| **count** | 8525.000000 | 8525.000000 | 8525.000000 | 8525.000000 | 8525.000000 | 8.525000e+03 |
| **mean** | 28.220247 | 28.514473 | 27.918967 | 28.224480 | 23.417934 | 6.045692e+07 |
| **std** | 28.626752 | 28.848988 | 28.370344 | 28.626571 | 28.195330 | 3.891225e+07 |
| **min** | 0.088542 | 0.092014 | 0.088542 | 0.090278 | 0.058081 | 2.304000e+06 |
| **25%** | 3.414063 | 3.460938 | 3.382813 | 3.414063 | 2.196463 | 3.667960e+07 |
| **50%** | 26.174999 | 26.500000 | 25.889999 | 26.160000 | 18.441576 | 5.370240e+07 |
| **75%** | 34.230000 | 34.669998 | 33.750000 | 34.230000 | 25.392508 | 7.412350e+07 |
| **max** | 159.449997 | 160.729996 | 158.330002 | 160.619995 | 160.619995 | 1.031789e+09 |

**CODE:**

df.info()

**OUTPUT:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8525 entries, 0 to 8524

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 8525 non-null object

1 Open 8525 non-null float64

2 High 8525 non-null float64

3 Low 8525 non-null float64

4 Close 8525 non-null float64

5 Volume 8525 non-null int64

6 open-close 8525 non-null float64

7 low-high 8525 non-null float64

8 target 8525 non-null int64

dtypes: float64(6), int64(2), object(1)

memory usage: 599.5+ KB

**CODE:**

plt.figure(figsize=(15,5))

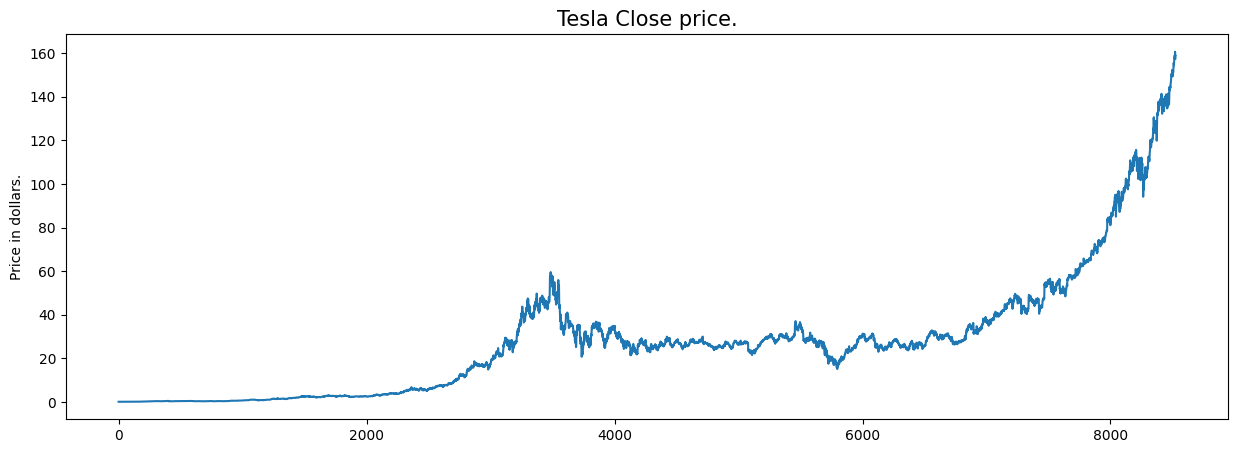
plt.plot(df['Close'])

plt.title('Tesla Close price.', fontsize=15)

plt.ylabel('Price in dollars.')

plt.show()

**OUTPUT:**



**CODE:**

df.head()

**OUTPUT:**

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 13-03-1986 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 |
| **1** | 14-03-1986 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 |
| **2** | 17-03-1986 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 |
| **3** | 18-03-1986 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 |
| **4** | 19-03-1986 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 |

**CODE:**

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

**OUTPUT:**

(32, 7)

**CODE:**

df.isnull().sum()

**OUTPUT:**

Date 0

Open 0

High 0

Low 0

Close 0

Volume 0

dtype: int64

**CODE:**

features = ['Open', 'High', 'Low', 'Close', 'Volume']

plt.subplots(figsize=(20,10))

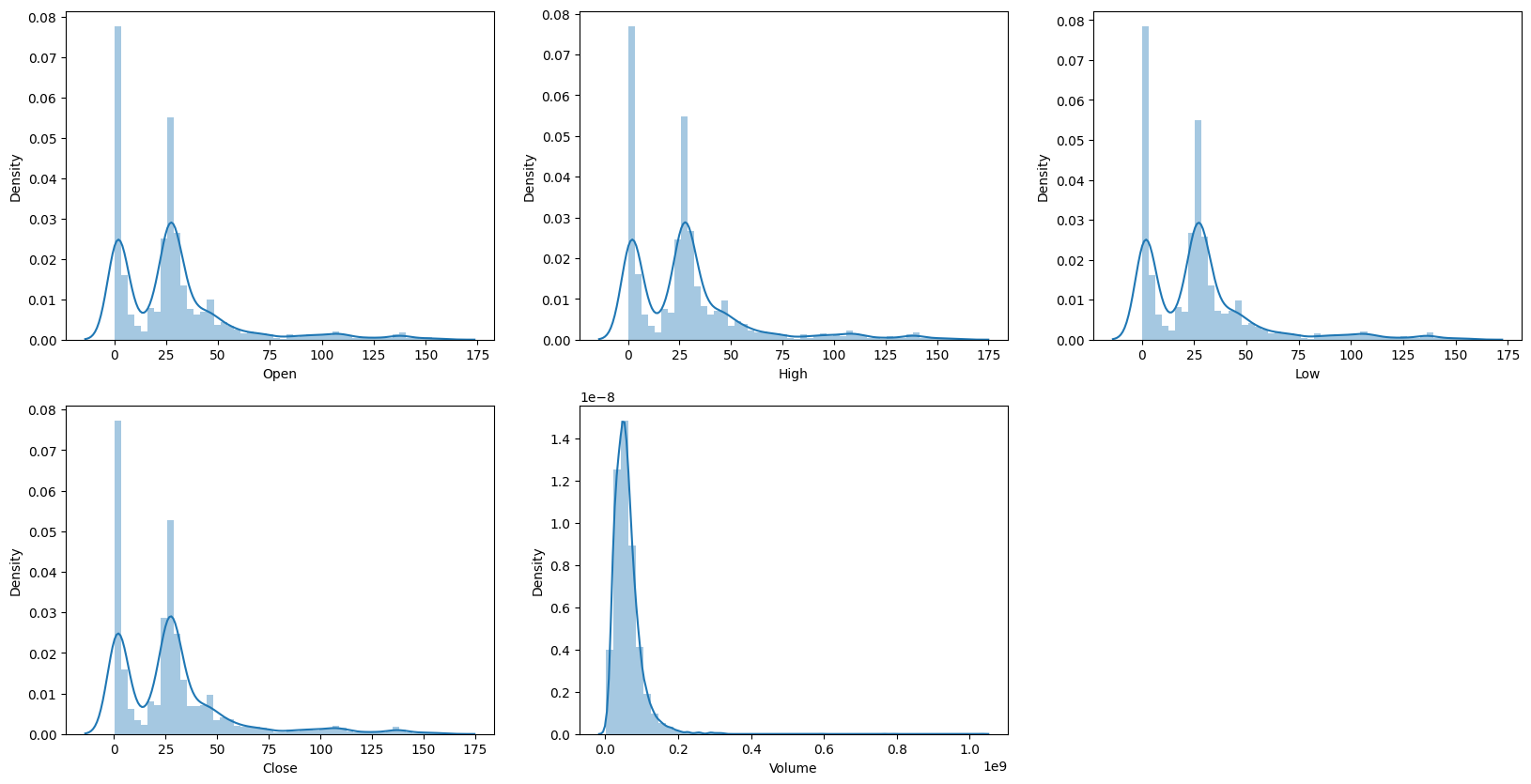
for i, col in enumerate(features):

  plt.subplot(2,3,i+1)

  sb.distplot(df[col])

plt.show()

**OUTPUT:**



**CODE:**

plt.subplots(figsize=(20,10))

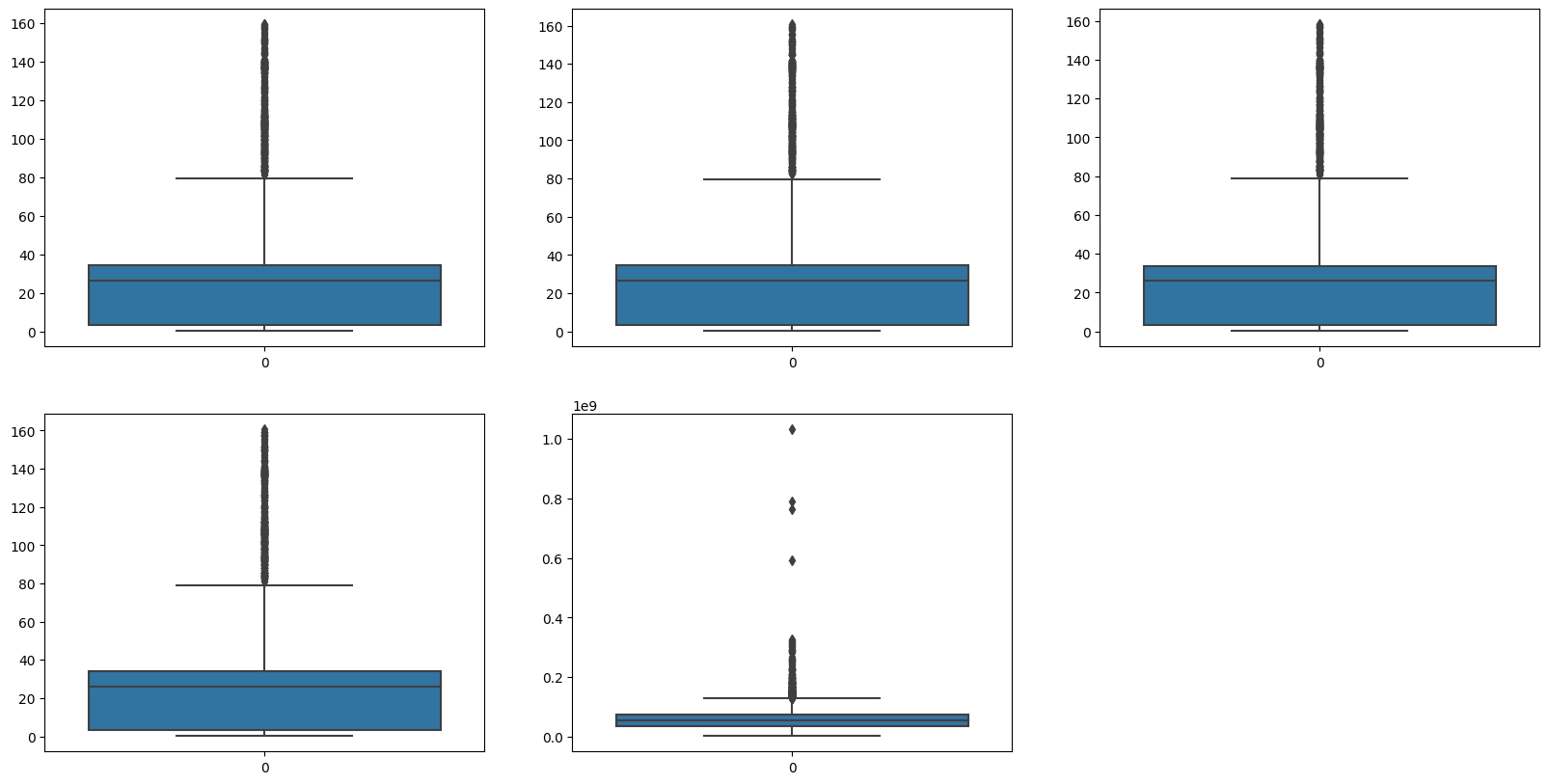
for i, col in enumerate(features):

  plt.subplot(2,3,i+1)

  sb.boxplot(df[col])

plt.show()

**OUTPUT:**



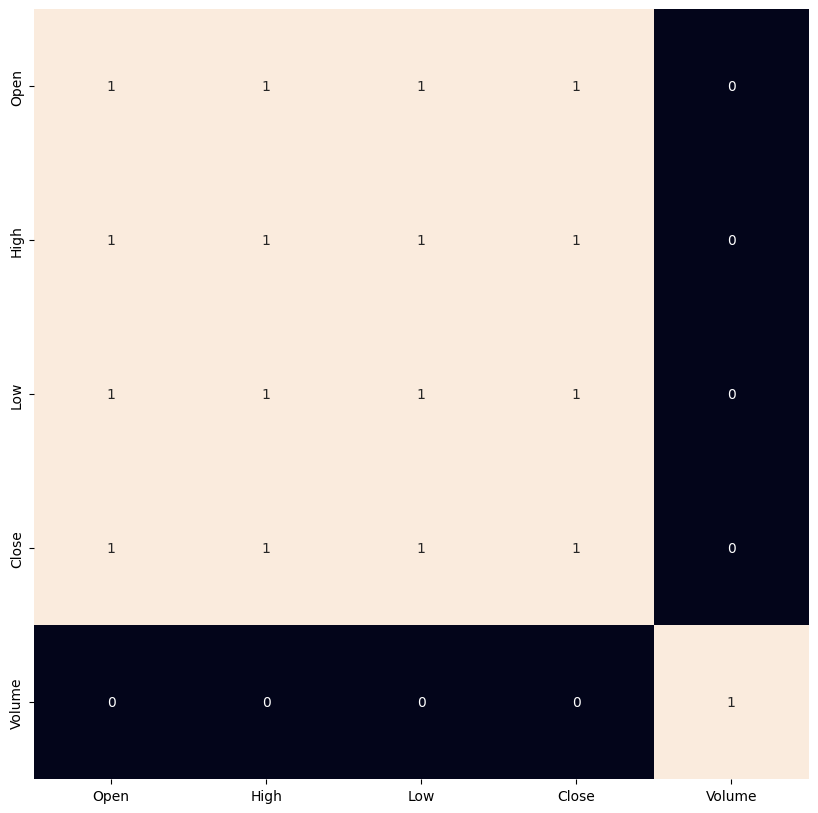
**CODE:**

plt.figure(figsize=(10, 10))

sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)

plt.show()

**OUTPUT:**



**CODE:**

df['open-close'] = df['Open'] - df['Close']

df['low-high'] = df['Low'] - df['High']

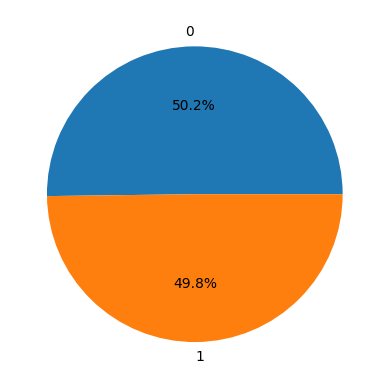
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)

plt.pie(df['target'].value\_counts().values,

    labels=[0, 1], autopct='%1.1f%%')

plt.show()

**OUTPUT:**



**CODE:**

plt.figure(figsize=(10, 10))

sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)

plt.show()

**OUTPUT:**

