**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:**



*df.shape*

**Output:**

(1692,7)

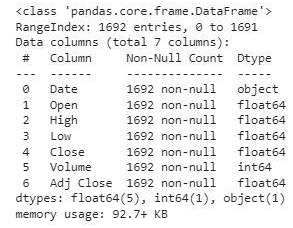
*df.describe()*

**Output:**



*df.info()*

**Output:**

****

**Exploratory Data Analysis**

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

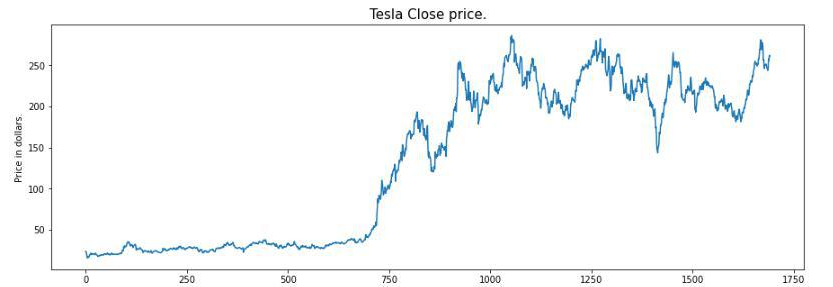
*plt.plot(df['Close'])*

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

*plt.ylabel('Price in dollars.')*

*plt.show()*

**Output:**



*df.head()*

**Output:**



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

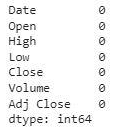
**Output:**

(1692,7)

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

*df.isnull().sum()*

**Output:**



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

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

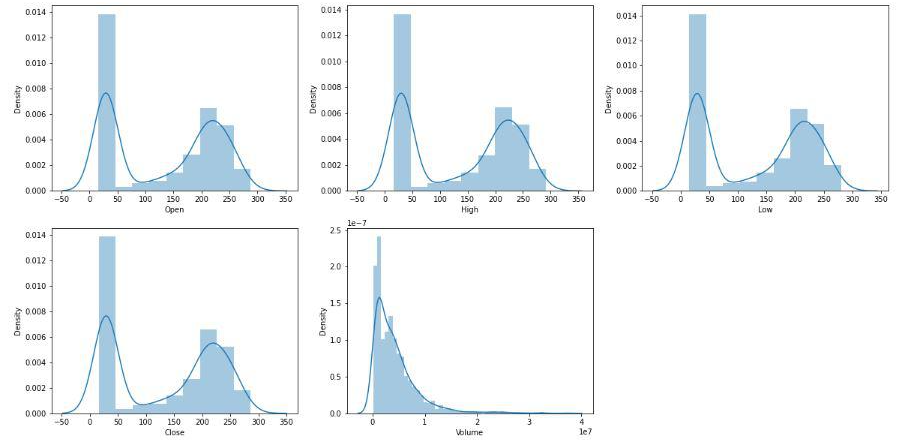
*for i, col in enumerate(features):*

*plt.subplot(2,3,i+1)*

*sb.distplot(df[col])*

*plt.show()*

**Output:**

****

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

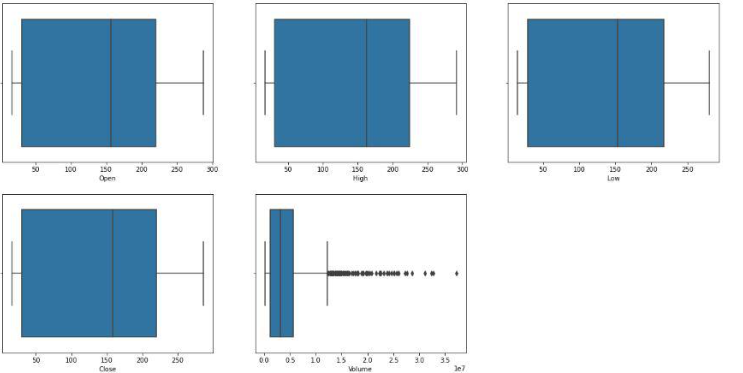
*for i, col in enumerate(features):*

*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:**



df['is\_quarter\_end'] = np.where(df['month']%3==0,1,0)

df.head()

**Output:**

****

*data\_grouped = df.groupby('year').mean()*

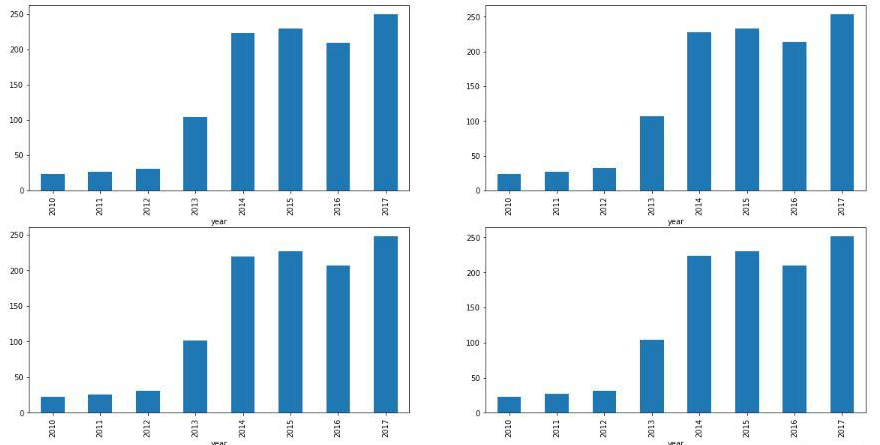
*plt.subplots(figsize=(20,10))*

*for i, col in enumerate(['Open', 'High', 'Low', 'Close']):*

*plt.subplot(2,2,i+1)*

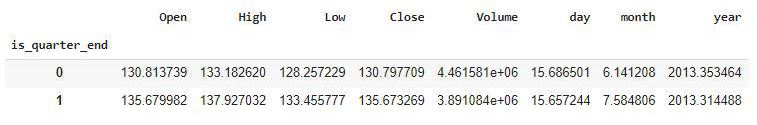
*data\_grouped[col].plot.bar()*

*plt.show()*



*df.groupby('is\_quarter\_end').mean()*

**Output:**

****

*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()*

