KINGSTON
ENGINEERING COLLEGE

COLLEGE CODE: 5113

APPLIED DATA SCIENCE

Project No.6 - STOCK PRICE PREDICTION

Batch Members:

1. ABHINANDAN A - (au511321104001) - abhiarjun0610@gmail.com
2. ABISHEK V P - (au511321104002) – abishekreddy7112003@gmail.com
3. MAHEDHAR V – (au511321104051) - vmahedhar7@gmail.com
4. BHAKTHULA CHANDU – (au511321104009)- chanchandu353@gmail.com

PHASE 5:

PROJECT DOCUMENTATION AND SUBMISSION

Problem Statement:

The problem at hand is to build a predictive model that can forecast stock prices based on historical market data. The primary objective is to create a tool that empowers investors to make informed decisions and optimize their investment strategies. This project encompasses multiple key tasks, including data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

DESIGN THINKING PROCESS:

Design thinking is a structured approach to problem-solving that emphasizes empathy for the end-users and a focus on innovative solutions. In the context of building a stock price prediction model, the design thinking process can be broken down into several phases:

Empathize:

The first step is to understand the needs and pain points of investors. What information do they require to make informed investment decisions, and what challenges do they face? This phase involves collecting feedback and gaining insights from potential users.

Define:

Once user needs are understood, a clear problem statement is defined. In this case, the problem is articulated as the need for a predictive model that can forecast stock prices accurately.

Ideate:

In this phase, creative ideas are generated for how to approach the problem. This could involve brainstorming potential features, data sources, and modeling techniques that can be used to predict stock prices.

Prototype:

A prototype or proof of concept is developed to demonstrate the feasibility of the solution. This might involve creating a basic version of the predictive model to test its functionality and performance.

Test:

The prototype is tested, and feedback is collected from users and stakeholders. This iterative testing and feedback process helps refine the model and ensure it meets user requirements.

Implement:

Once the prototype is refined and validated, it can be implemented as a functional predictive model. This may involve scaling up the system to handle real-time data and user interactions.

Evaluate:

Continuous evaluation is critical to ensure that the model performs as expected. Metrics for accuracy and performance are monitored to assess the model's effectiveness in assisting investors.

PHASES OF DEVELOPMENT:

The development process for building a stock price prediction model can be broken down into several phases:

Data Collection:

Historical stock market data is collected from various sources, including open and close prices, trading volumes, and relevant financial indicators. The quality and quantity of data are essential for building an accurate model.

Data Preprocessing:

The raw data is cleaned and preprocessed. This includes handling missing values, encoding categorical data, and standardizing or normalizing features to make them suitable for modeling.

Feature Engineering:

Additional features are created to enhance the model's predictive power. This might involve calculating moving averages, technical indicators, or lagged variables.

Model Selection:

The choice of modeling techniques is a critical decision. Depending on the nature of the problem, algorithms like linear regression, decision trees, random forests, or time series models such as ARIMA or LSTM may be selected.

Model Training:

The selected model is trained using the pre-processed data. The training phase involves learning the relationships between features and target variables from historical data.

Evaluation:

For the evaluation metric, we will use the ROC-AUC curve but why this is because instead of predicting the hard probability that is 0 or 1 we would like it to predict soft probabilities that are continuous values between 0 to 1. And with soft probabilities, the ROC-AUC curve is generally used to measure the accuracy of the predictions.

Deployment:

In a real-world scenario, the model is deployed to provide real-time or near-real-time stock price predictions. Considerations like data latency, model maintenance, and continuous retraining are vital in this phase.

By following this structured development process and incorporating the principles of design thinking, we can build a robust stock price prediction model that caters to the needs of investors and aids in making well-informed investment decisions.

DATASET DESCRIPTION:

The "Microsoft Lifetime Stocks Dataset" on Kaggle contains historical stock market data related to Microsoft Corporation (MSFT) over a significant time period. This dataset is a valuable resource for understanding and analysing the stock performance of one of the world's largest technology companies.

Dataset Link:

https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

Key attributes in the dataset typically include:

Date: The date of the stock market data.

Open Price: The opening price of MSFT stock on a given trading day.

High Price: The highest price reached during the trading day.

Low Price: The lowest price recorded on the same trading day.

Close Price: The closing price of MSFT stock on a given trading day.

Volume: The trading volume for the stock on a specific date.

Additional attributes may be available in the dataset, depending on its completeness. The dataset allows for time series analysis, predictive modelling, and understanding how various factors impact the stock's performance.

DATA PREPROCESSING STEPS:

Data preprocessing is a crucial stage in preparing the dataset for analysis and modelling. Several key steps are typically involved in this process:

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 xgboost import XGBClassifier

from sklearn.svm import SVC


from sklearn import metrics


import warnings

warnings.filterwarnings('ignore')
```

Loading the Data:

The dataset is loaded into a data structure suitable for analysis, such as a Pandas Data Frame in Python.

```
df = pd.read_csv("C:/Users/Abhinandan A/Downloads/MSFT.csv")
df.head()
df.shape
df.describe()
df.info()
```

Exploratory Data Analysis

EDA is an approach to analyzing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations.

While performing the EDA of the Tesla Stock Price data we will analyze how prices of the stock have moved over the period of time and how the end of the quarters affects the prices of the stock.

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

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

Handling Missing Values:

It's common for financial datasets to have missing values, which can impact the quality of analysis. These missing values are addressed through techniques like imputation or simply removing rows with missing data.

```
df.isnull().sum()
```

Distribution plot:

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

FEATURE ENGINEERING:

Feature Engineering helps to derive some valuable features from the existing ones. These extra features sometimes help in increasing the performance of the model significantly and certainly help to gain deeper insights into the data.

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


df['month'] = splitted[1].astype('int')

df['year'] = splitted[0].astype('int')

df['day'] = splitted[2].astype('int')


df.head()
```

Encoding Categorical Data:

If the dataset includes categorical variables (e.g., 'day of the week'), they are typically one-hot encoded to convert them into a numerical format that machine learning models can work with.

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

df.head()


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()


    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()



    plt.figure(figsize=(10, 10))
    sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
    plt.show()
```

From the above heatmap, we can say that there is a high correlation between OHLC that is pretty obvious, and the added features are not highly correlated with each other or previously provided features which means that we are good to go and build our model.

Data Splitting and Normalization:

To ensure that features are on a similar scale, standardization or normalization is applied.

After preprocessing, the dataset is split into features (X) and the target variable (YThis ensures that model performance is evaluated on unseen data, helping to assess generalization.

In this context, features are typically the historical price data (e.g., Open, High, Low) and engineered features. The target variable is often the Close Price.

```
features = df[['open-close', 'low-high', 'is_quarter_end']]
target = df['target']

scaler = StandardScaler()
features = scaler.fit_transform(features)

X_train, X_valid, Y_train, Y_valid = train_test_split(features, target,
    test_size=0.1, random_state=2022)
print(X_train.shape, X_valid.shape)
```

MODEL TRAINING PROCESS AND EVALUATION:

Train some state-of-the-art machine learning models(Logistic Regression, Support Vector Machine, XGBClassifier), and then based on their performance on the training and validation data we will choose which ML model is serving the purpose at hand better.

For the evaluation metric, we will use the ROC-AUC curve but why this is because instead of predicting the hard probability that is 0 or 1 we would like it to predict soft probabilities that are continuous values between 0 to 1. And with soft probabilities, the ROC-AUC curve is generally used to measure the accuracy of the predictions.

```
models = [LogisticRegression(), SVC(kernel='poly', probability=True),
XGBClassifier()]
for i in range(3):models[i].fit(X_train, Y_train)
    print(f'{models[i]} : ')
    print('Training Accuracy : ', metrics.roc_auc_score(
        Y_train, models[i].predict_proba(X_train)[:,1]))
    print('Validation Accuracy : ', metrics.roc_auc_score(
        Y_valid, models[i].predict_proba(X_valid)[:,1]))
    print()

metrics.plot_confusion_matrix(models[0], X_valid, Y_valid)
plt.show()
```

PROGRAM:

```
#import packages
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 xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')

#importing dataset
df=pd.read_csv("C:/Users/Abhinandan A/Downloads/MSFT.csv")
df.head()
df.shape
df.describe()
df.info()
```

```
# exploratory data analysis
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('MSFT Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()
df.head()

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

df.isnull().sum()

#dist plot
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()
#BOX PLOT
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
```

```
  sb.boxplot(df[col])
plt.show()

#feature enginearing
splitted = df['Date'].str.split('-', expand=True)

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

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

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()

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()

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

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

plt.show()

#splitting and normalization

features = df[['open-close', 'low-high', 'is_quarter_end']]

target = df['target']


scaler = StandardScaler()

features = scaler.fit_transform(features)

X_train, X_valid, Y_train, Y_valid = train_test_split(features, target,
    test_size=0.1, random_state=2022)

print(X_train.shape, X_valid.shape)


#model training and evaluation

models = [LogisticRegression(), SVC(kernel='poly', probability=True),
XGBClassifier()]

for i in range(3):models[i].fit(X_train, Y_train)

    print(f'{models[i]} : ')

    print('Training Accuracy : ', metrics.roc_auc_score(

        Y_train, models[i].predict_proba(X_train)[:,1]))

print('Validation Accuracy : ', metrics.roc_auc_score(
```

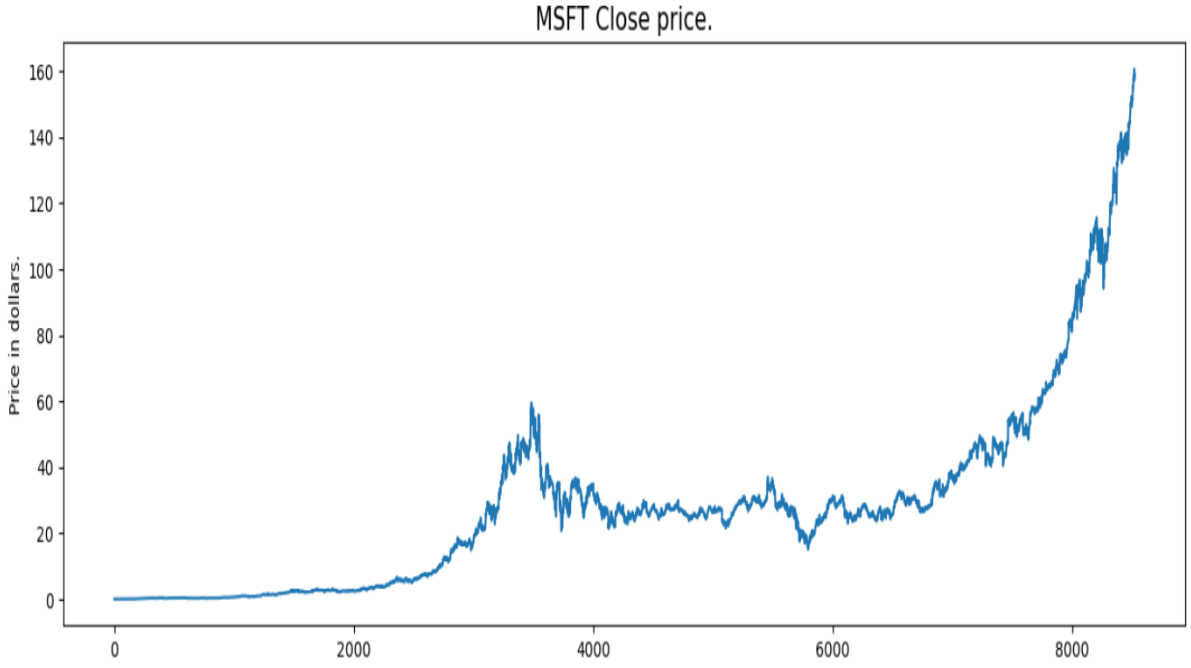
```
Y_valid, models[i].predict_proba(X_valid)[:,1]))
print()

metrics.plot_confusion_matrix(models[0], X_valid, Y_valid)
plt.show()
#end
```

OUTPUT:

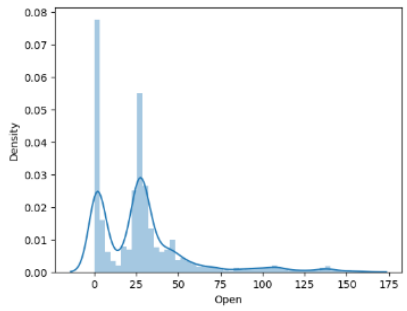
	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224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

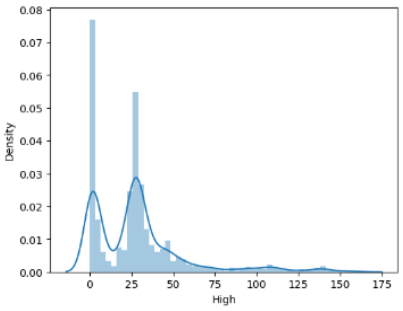
	Open	High	Low	Close	Adj Close	Volume
count	8525.000000	8525.000000	8525.000000	8525.000000	8525.000000	8.525000e+03
mean	28.220247	28.514473	27.918967	28.224480	23.417934	6.045692e+07
std	28.626752	28.848988	28.370344	28.626571	28.195330	3.891225e+07
min	0.088542	0.092014	0.088542	0.090278	0.058081	2.304000e+06
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

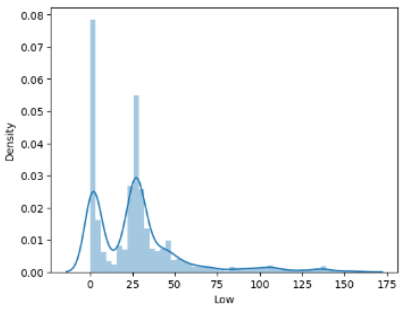
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       8525 non-null   object
 1   Open       8525 non-null   float64
 2   High       8525 non-null   float64
 3   Low        8525 non-null   float64
 4   Close      8525 non-null   float64
 5   Adj Close  8525 non-null   float64
 6   Volume     8525 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 466.3+ KB
```

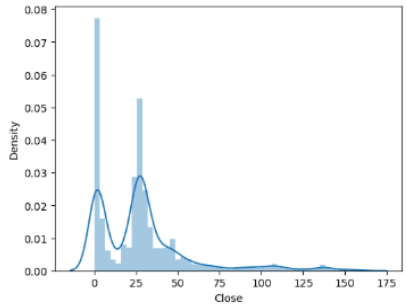


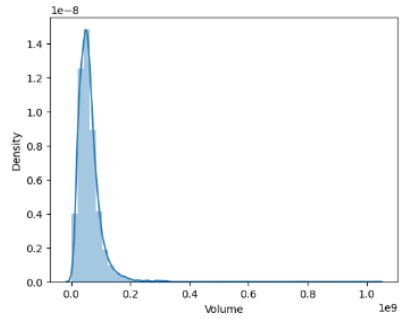

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

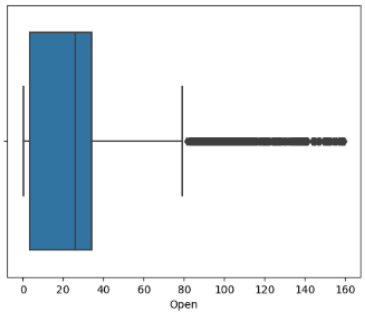


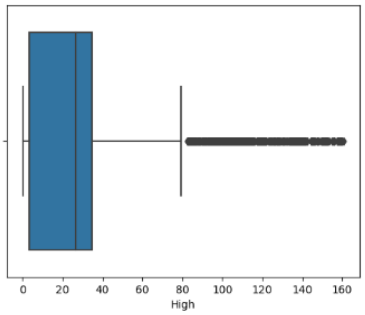


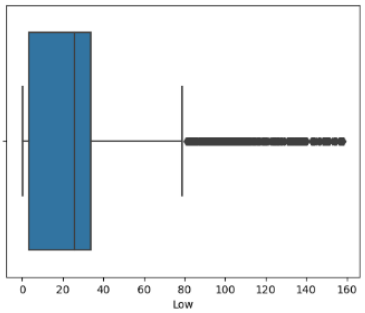


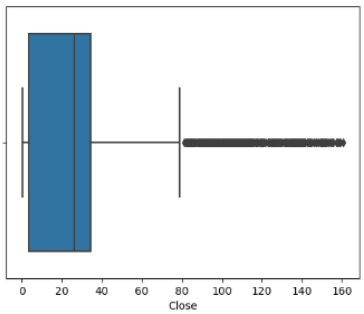


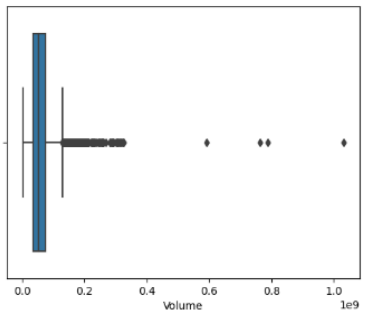






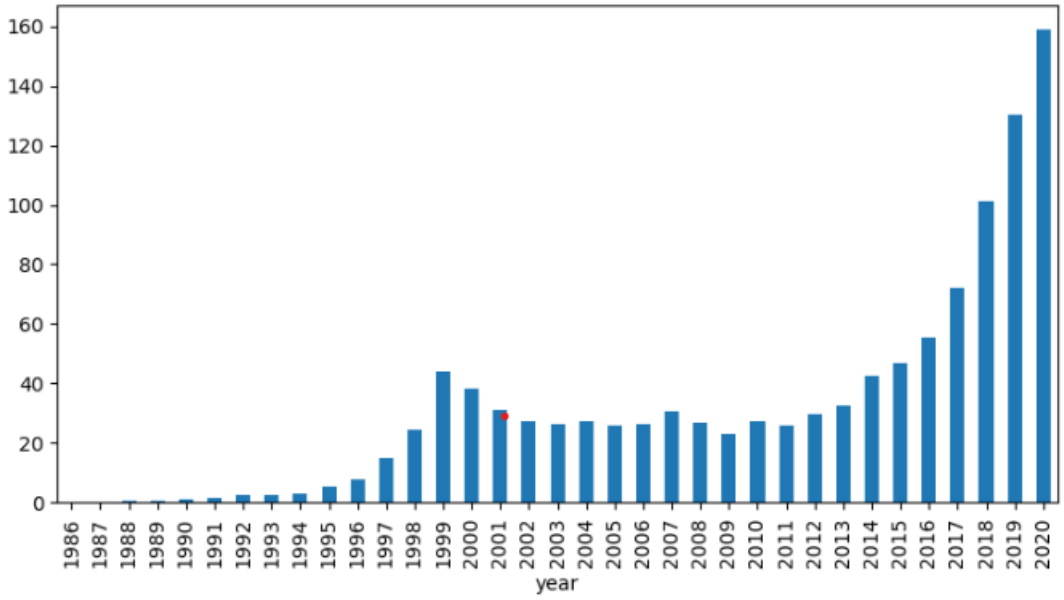




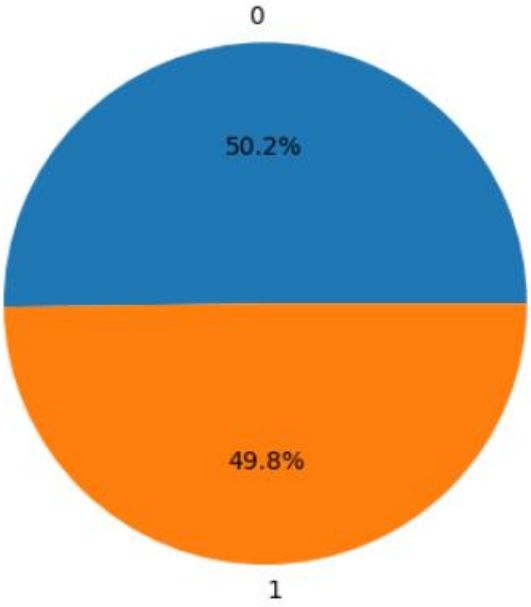


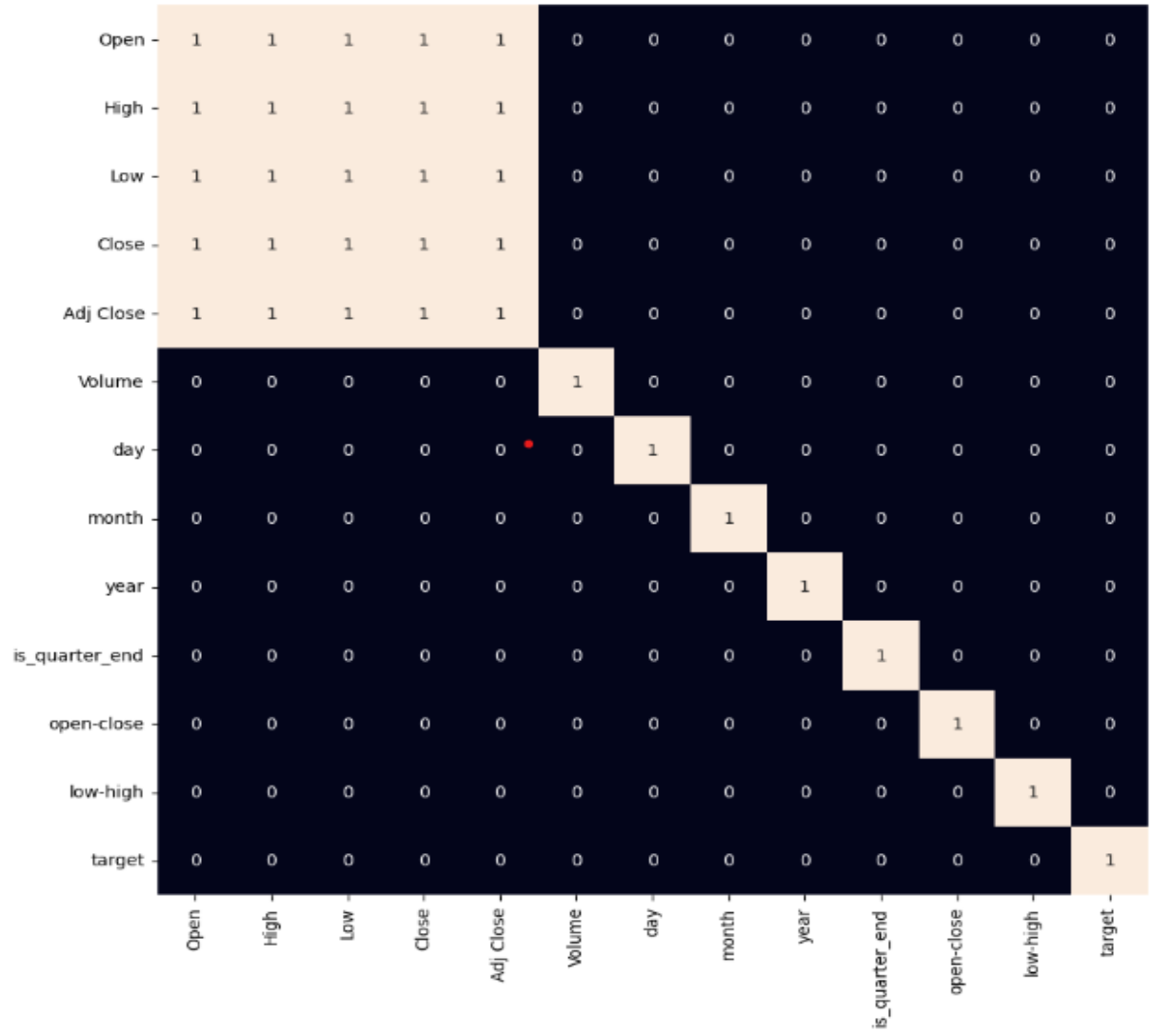
	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224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

	day	month	year	is_quarter_end	open-close	low-high	target
0	13	3	1986	1	-0.008680	-0.013021	1
1	14	3	1986	1	-0.003472	-0.005209	1
2	17	3	1986	1	-0.001737	-0.002605	0
3	18	3	1986	1	0.002605	-0.004341	0
4	19	3	1986	1	0.001736	-0.003472	0

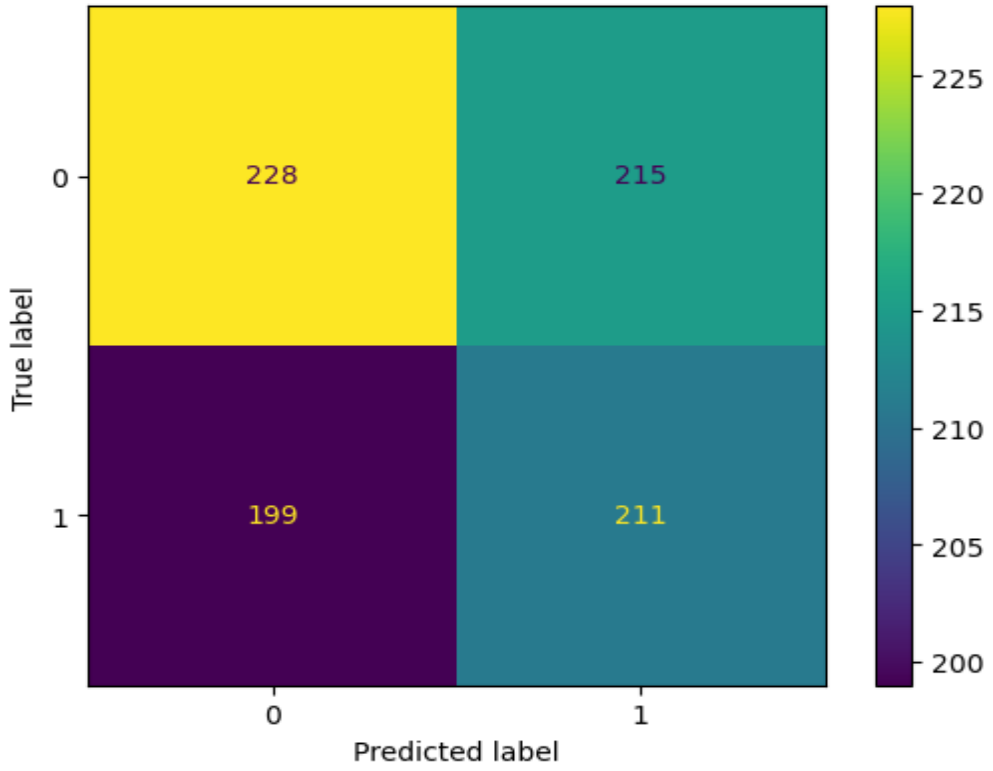


is_quarter_end	Open	High	Low	Close	Adj Close	Volume	day	month	year
0	26.717413	27.010233	26.412092	26.712410	21.781026	6.217010e+07	6.533321	2002.489373	15.721362
1	31.026750	31.323604	30.733016	31.048233	26.474818	5.725761e+07	6.618567	2002.759839	15.771611






```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  0.8171678308846017
Validation Accuracy :  0.48974838958321865
```



KEY FINDINGSS:

Data Quality and Dataset Overview:

The dataset is well-maintained with no missing values. It provides a comprehensive historical record of Microsoft's stock performance, including key attributes such as opening and closing prices, high and low prices, and trading volume.

Long-Term Growth:

The time series plot of Microsoft's closing stock prices illustrates a consistent upward trend. This long-term growth suggests that MSFT is a compelling choice for long-term investors, offering a positive outlook for capital appreciation.

Quarterly Patterns:

The identification of quarter-end dates in the dataset implies that investors should pay close attention to the stock's performance during these periods. Quarterly financial reports and announcements may influence stock price movements, making it a strategic consideration for investors.

Outlier Analysis:

The presence of outliers in certain features, as indicated by box plots, is notable. Investigating the reasons behind these outliers can provide insights into exceptional market events or anomalies that may affect stock prices.

INSIGHTS:

Investment Strategy:

Long-term investors can benefit from Microsoft's consistent growth. A buy-and-hold strategy aligns well with the long-term upward trend in MSFT stock.

Quarterly Earnings Strategy:

Investors might consider adjusting their positions around quarter-end dates when financial results are reported. These periods often lead to significant price movements, which can be leveraged for strategic investment decisions.

Outlier Management:

Understanding and monitoring outliers is essential, especially for features like trading volume. Anomalies may be tied to significant news or market events, necessitating a thorough investigation for informed investment strategies.

RECOMMENDATIONS:

Long-Term Investment:

Long-term investors can take advantage of Microsoft's consistent growth trajectory. Holding onto MSFT shares with a focus on the company's long-term prospects can lead to capital appreciation.

Earnings Season Strategy:

Given the quarterly patterns, investors should consider portfolio adjustments around quarter-end dates. This strategic approach can help capture price movements associated with earnings reports.

Outlier Investigation:

Analyze and understand the reasons behind outliers in the dataset. Abnormal trading volumes or price fluctuations may signal crucial events that can impact stock prices. An anomaly detection strategy can help manage these outliers.

CONCLUSION:

In summary, our data analysis of the Microsoft Lifetime Stocks Dataset has yielded valuable insights into the stock's behavior and potential investment strategies. We began by ensuring data quality, eliminating missing values, and delving into the dataset's characteristics.

The most striking discovery was the long-term upward trend in Microsoft's stock price, reflecting its consistent growth over time. This finding underscores the allure of long-term investment in a stable and successful company.

Furthermore, the identification of quarterly patterns tied to earnings announcements highlights the potential for strategic actions during these periods. Investors can use this knowledge to make

timely adjustments to their portfolios, optimizing returns around earnings seasons.

The investigation of outliers in trading volume signifies the relevance of unusual market events. These outliers can serve as valuable indicators of significant news or developments that might affect stock prices. By proactively monitoring and responding to such outliers, investors can make informed decisions.

The correlations between features, particularly the strong positive correlation between the opening and closing prices, emphasize the importance of feature selection when developing predictive models.

This analysis reaffirms the pivotal role of data science in financial decision-making. It empowers investors to navigate the complexities of the stock market, extract actionable insights from data, and enhance investment strategies. The future lies in the fusion of traditional financial wisdom with data-driven precision, enabling investors to thrive in the ever-evolving landscape of finance.Depending on the model's performance, hyperparameter tuning and optimization may be necessary to improve predictive accuracy.