STOCK PRICE PREDICTION

1.Abstract

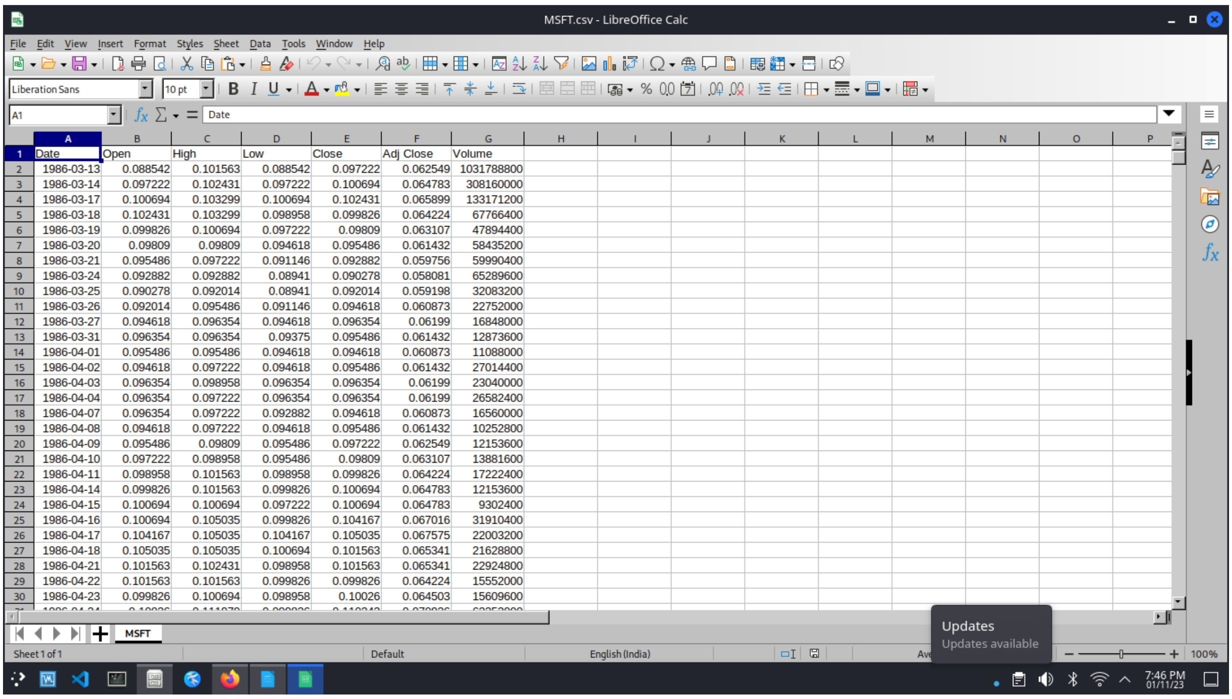
Problem Statement: Stock price prediction

The stock market is a dynamic and complex financial ecosystem where the prices of various assets are subject to constant fluctuations. Accurate prediction of stock prices is crucial for investors, traders, and financial analysts to make informed decisions. In this document, we will explore the process of stock price prediction using machine learning techniques.

Objectives

The primary objective of this project is to develop a predictive model that can forecast stock prices with reasonable accuracy. By leveraging historical stock price data and various machine learning algorithms, we aim to provide a valuable tool for financial decision-making.

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



Date	Open	High	Low	Close	Adj Close	Volume
1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
1986-03-19	0.099826	0.100694	0.097222	0.09809	0.063107	47894400
1986-03-20	0.09809	0.09809	0.094618	0.095486	0.061432	58435200
1986-03-21	0.095486	0.097222	0.091146	0.092882	0.059756	59990400
1986-03-24	0.092882	0.092882	0.08941	0.090278	0.058081	65289600
1986-03-25	0.090278	0.092014	0.08941	0.092014	0.059198	32083200
1986-03-26	0.092014	0.095486	0.091146	0.094618	0.060873	22752000
1986-03-27	0.094618	0.096354	0.094618	0.096354	0.06199	16848000
1986-03-31	0.096354	0.096354	0.09375	0.095486	0.061432	12873600
1986-04-01	0.095486	0.095486	0.094618	0.094618	0.060873	11088000
1986-04-02	0.094618	0.097222	0.094618	0.095486	0.061432	27014400
1986-04-03	0.096354	0.098958	0.096354	0.096354	0.06199	23040000
1986-04-04	0.096354	0.097222	0.096354	0.096354	0.06199	26582400
1986-04-07	0.096354	0.097222	0.092882	0.094618	0.060873	16560000
1986-04-08	0.094618	0.097222	0.094618	0.095486	0.061432	10252800
1986-04-09	0.095486	0.09809	0.095486	0.097222	0.062549	12153600
1986-04-10	0.097222	0.098958	0.095486	0.09809	0.063107	13881600
1986-04-11	0.098958	0.101563	0.098958	0.099826	0.064224	17222400
1986-04-14	0.099826	0.101563	0.099826	0.100694	0.064783	12153600
1986-04-15	0.100694	0.100694	0.097222	0.100694	0.064783	9302400
1986-04-16	0.100694	0.105035	0.099826	0.104167	0.067016	31910400
1986-04-17	0.104167	0.105035	0.104167	0.105035	0.067575	22003200
1986-04-18	0.105035	0.105035	0.100694	0.101563	0.065341	21628800
1986-04-21	0.101563	0.102431	0.098958	0.101563	0.065341	22924800
1986-04-22	0.101563	0.101563	0.099826	0.099826	0.064224	15552000
1986-04-23	0.099826	0.100694	0.098958	0.10026	0.064503	15609600

2. Approches

This project focuses on the development of a predictive model aimed at forecasting stock prices by leveraging historical market data. The primary objective is to provide investors with a valuable tool to aid in making well-informed decisions and optimizing their investment strategies. The project encompasses a comprehensive workflow, including data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

Data Collection: We gather historical stock market data, encompassing crucial attributes such as date, open price, close price, volume, and other relevant indicators. This dataset forms the foundation for our predictive modeling efforts.

Data Preprocessing: A critical step involves cleaning and preprocessing the collected data. This includes addressing missing values, handling outliers, and converting categorical features into numerical representations, ensuring data quality and consistency.

Feature Engineering: To enhance the predictive capabilities of our model, we employ feature engineering techniques. This includes creating new features, such as moving averages, technical indicators, and lagged variables, which can capture underlying patterns and trends in the stock market.

Model Selection: We carefully choose appropriate algorithms for time series forecasting, such as ARIMA (AutoRegressive Integrated Moving Average) and LSTM (Long Short-Term Memory), to predict stock prices effectively. Model selection is a critical decision in ensuring accurate and reliable predictions.

Model Training: With a selected model in place, we proceed to train it using the preprocessed dataset. The training phase involves learning from historical data, enabling the model to capture patterns and relationships that influence stock price movements.

Evaluation: The performance of our predictive model is rigorously assessed using suitable time series forecasting metrics, such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). This evaluation ensures that the model's predictions align with actual stock price movements and provides an indication of its reliability.

Ultimately, this project strives to empower investors with a robust tool for making informed decisions in the dynamic and complex world of stock market investing. By leveraging historical data, preprocessing techniques, feature engineering, and advanced forecasting models, we aim to create a valuable asset for optimizing investment strategies and enhancing financial decision-making.

Program for stock price prediction.

Data Preprocessing:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

file_path = '/content/drive/MyDrive/Projects/Stock-price-prediction/MSFT.csv'
df = pd.read_csv(file_path)

# Display basic information about the dataset
print(df.head())  # Display the first few rows
print(df.info())  # Summary of the dataset, data types, and missing values

# Check for missing values
print(df.isnull().sum())
```

 # Descriptive statistics

```
print(df.describe())

# Visualize the stock price over time
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], label='Stock Price')
plt.title('Stock Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()

# Distribution of stock prices
plt.figure(figsize=(8, 4))
sns.histplot(df['Close'], bins=30, kde=True)
plt.title('Distribution of Stock Prices')
plt.show()


df['Returns'] = df['Close'].pct_change()  # Calculate daily returns
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Returns'], label='Daily Returns', color='orange')
plt.title('Daily Returns Over Time')
plt.xlabel('Date')
plt.ylabel('Returns')
plt.legend()
plt.show()

# Correlation matrix
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Rolling statistics (e.g., 50-day moving average)
df['50_Day_MA'] = df['Close'].rolling(window=50).mean()
df['200_Day_MA'] = df['Close'].rolling(window=200).mean()

plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], label='Stock Price')
plt.plot(df['Date'], df['50_Day_MA'], label='50-Day MA', color='orange')
plt.plot(df['Date'], df['200_Day_MA'], label='200-Day MA', color='red')
plt.title('Stock Price and Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()

# Visualize trading volume
plt.figure(figsize=(12, 6))
```

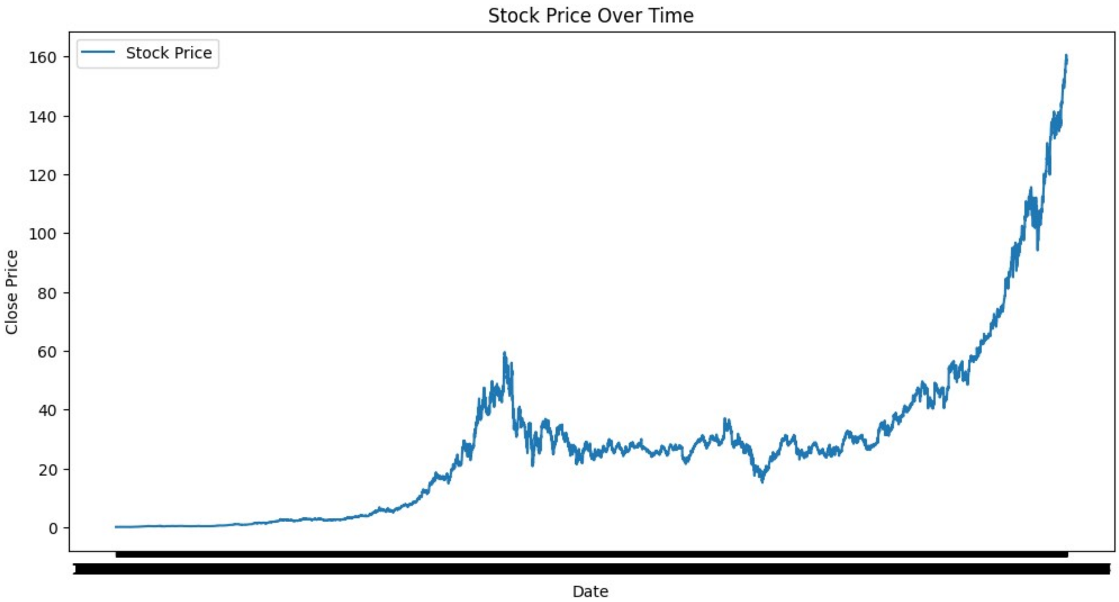
```
plt.plot(df['Date'], df['Volume'], label='Trading Volume', color='purple')
plt.title('Trading Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.legend()
plt.show()
```

Output of Data Preprocessing :

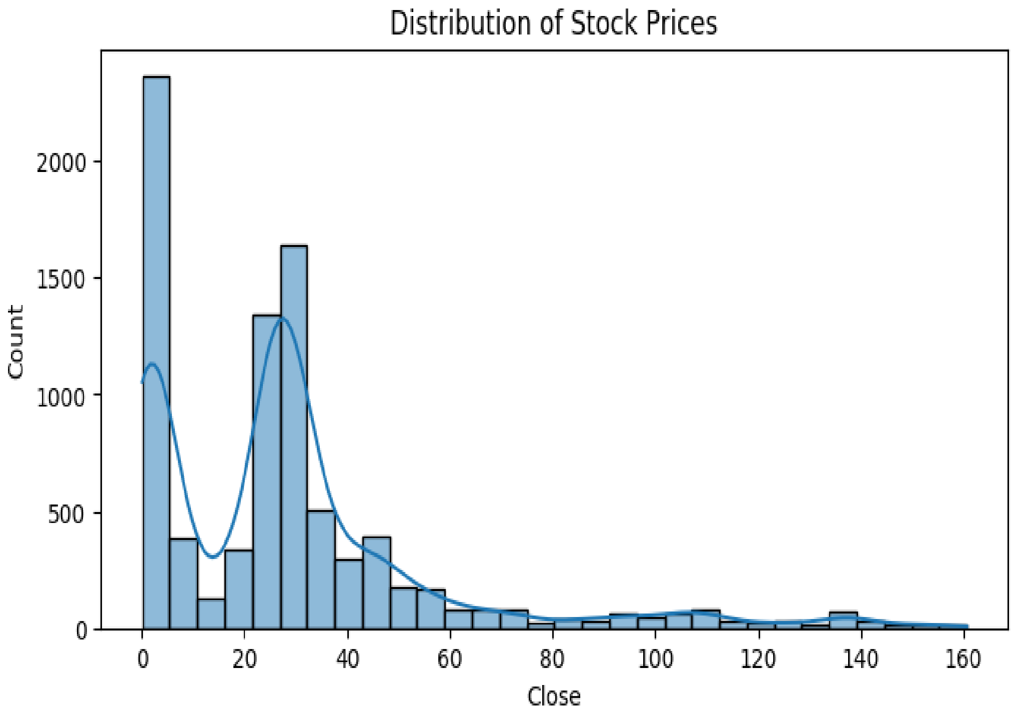
```
         Date      Open      High       Low     Close  Adj Close
Volume
0  1986-03-13  0.088542  0.101563  0.088542  0.097222   0.062549  1031788800
1  1986-03-14  0.097222  0.102431  0.097222  0.100694   0.064783   308160000
2  1986-03-17  0.100694  0.103299  0.100694  0.102431   0.065899   133171200
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
<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
None
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
              Open         High          Low        Close    Adj Close  \
count  8525.000000  8525.000000  8525.000000  8525.000000  8525.000000
mean     28.220247    28.514473    27.918967    28.224480    23.417934
std      28.626752    28.848988    28.370344    28.626571    28.195330
min       0.088542     0.092014     0.088542     0.090278     0.058081
25%       3.414063     3.460938     3.382813     3.414063     2.196463
50%      26.174999    26.500000    25.889999    26.160000    18.441576
75%      34.230000    34.669998    33.750000    34.230000    25.392508
max     159.449997   160.729996   158.330002   160.619995   160.619995

             Volume
count  8.525000e+03
mean   6.045692e+07
std    3.891225e+07
min    2.304000e+06
25%    3.667960e+07
50%    5.370240e+07
75%    7.412350e+07
max    1.031789e+09
```

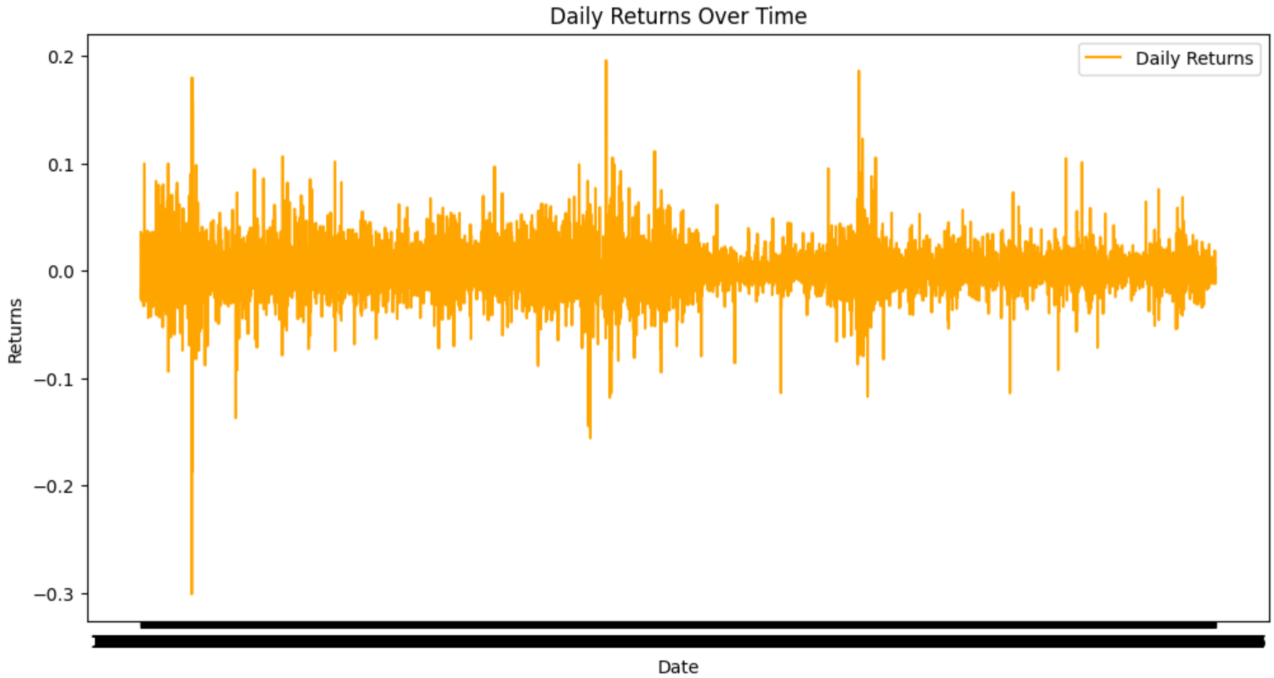
Price over time :



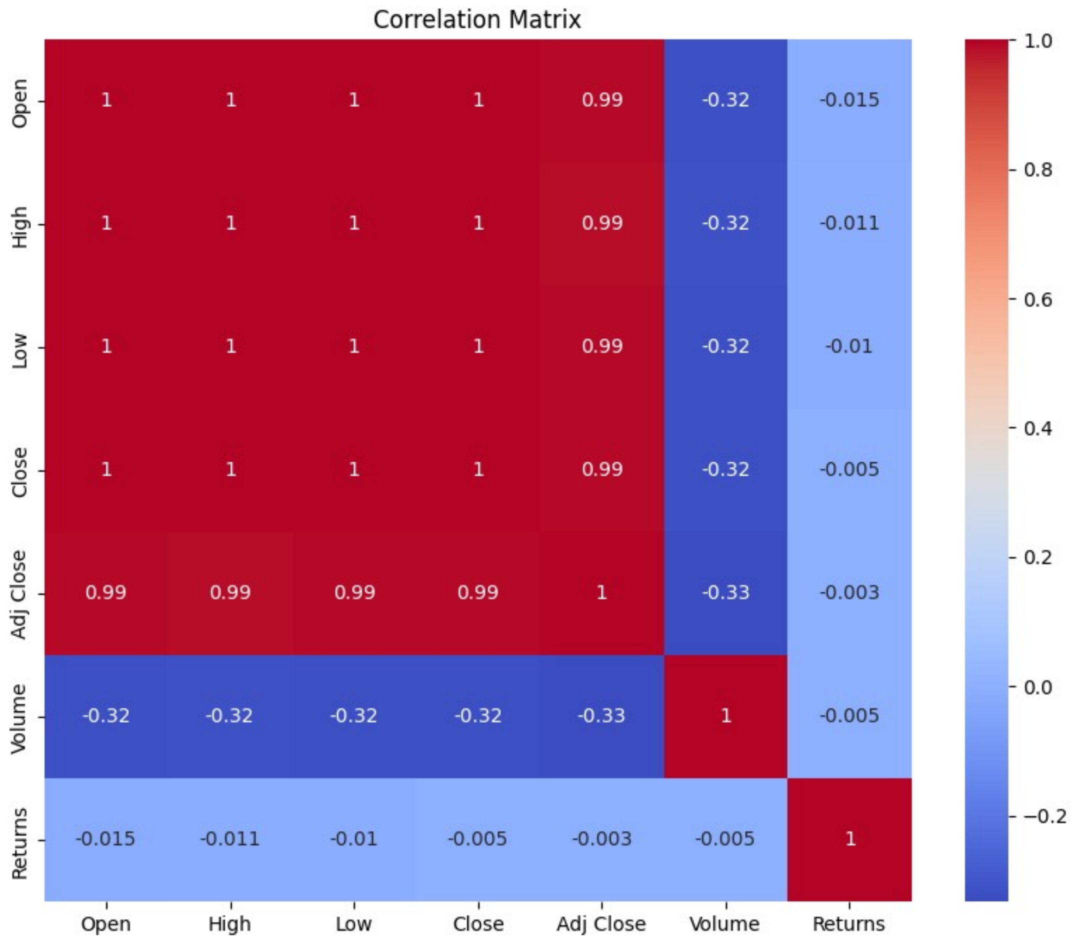
Distriution of stocks:



Daily returns overt time:

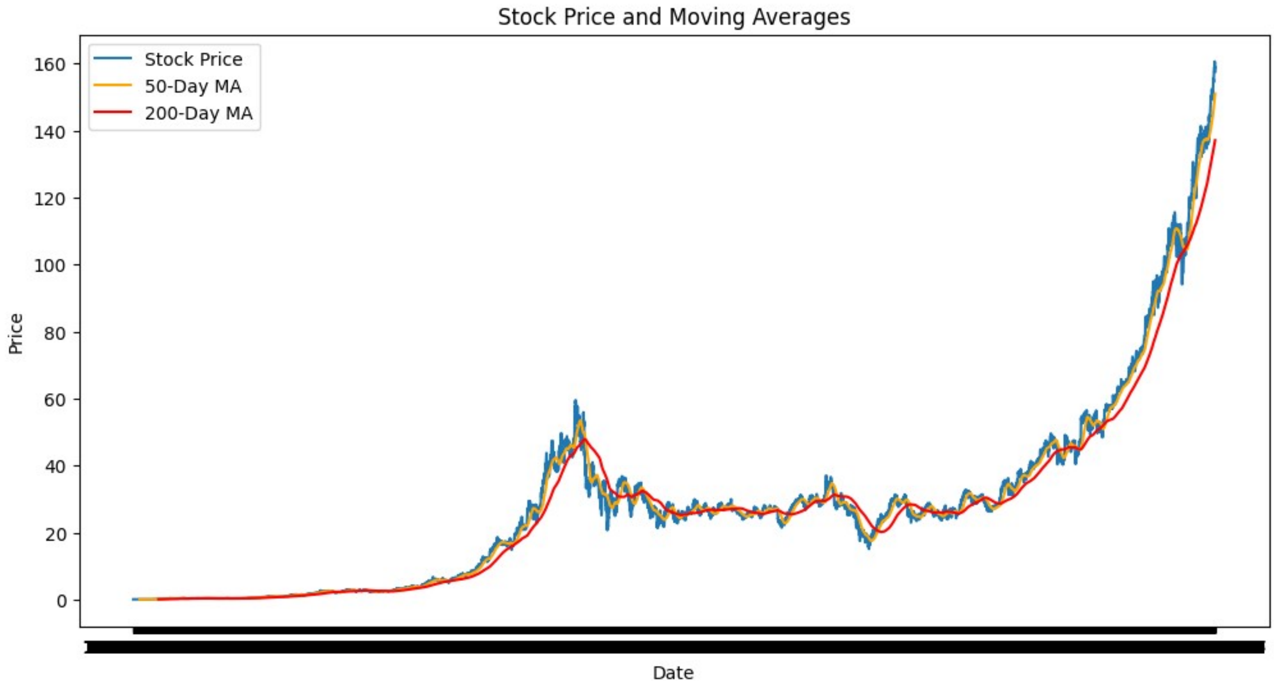


Correlation matrix:

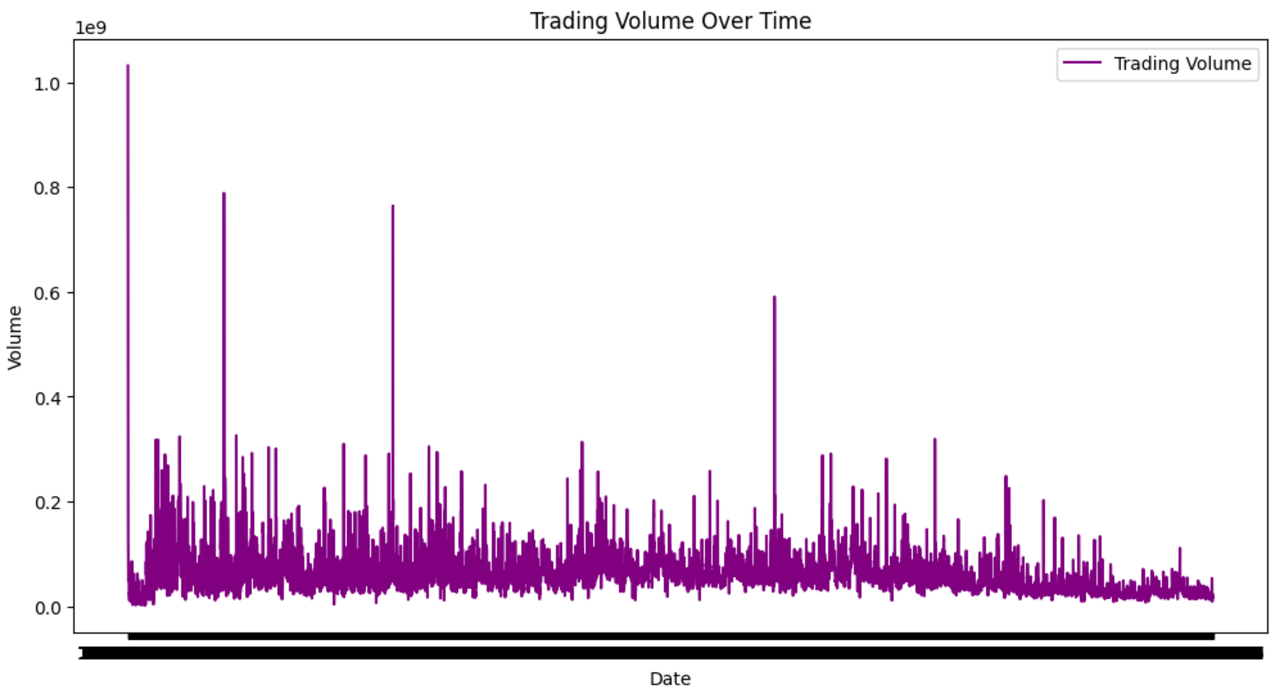


	Open	High	Low	Close	Adj Close	Volume	Returns
Open	1	1	1	1	0.99	-0.32	-0.015
High	1	1	1	1	0.99	-0.32	-0.011
Low	1	1	1	1	0.99	-0.32	-0.01
Close	1	1	1	1	0.99	-0.32	-0.005
Adj Close	0.99	0.99	0.99	0.99	1	-0.33	-0.003
Volume	-0.32	-0.32	-0.32	-0.32	-0.33	1	-0.005
Returns	-0.015	-0.011	-0.01	-0.005	-0.003	-0.005	1

Stock price and average moving:



Trading volume overtime :



Code for Feature Engineering:

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Projects/Stock-price-prediction/MSFT.csv')
def calculate_moving_averages(df, window=10):
    df['SMA'] = df['Close'].rolling(window=window).mean()  # Simple Moving Average
    df['EMA'] = df['Close'].ewm(span=window, adjust=False).mean()  # Exponential Moving Average
def calculate_rsi(df, window=14):
    delta = df['Close'].diff()
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)

    avg_gain = gain.rolling(window=window, min_periods=1).mean()
    avg_loss = loss.rolling(window=window, min_periods=1).mean()

    rs = avg_gain / avg_loss
    df['RSI'] = 100 - (100 / (1 + rs))
def calculate_macd(df, short_window=12, long_window=26):
    short_ema = df['Close'].ewm(span=short_window, adjust=False).mean()
    long_ema = df['Close'].ewm(span=long_window, adjust=False).mean()
    df['MACD'] = short_ema - long_ema
calculate_moving_averages(df)
calculate_rsi(df)
calculate_macd(df)
df.dropna(inplace=True)
```

Model Training code :

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# Prepare the target variable and features
target_column = 'Close'
X = df.drop(columns=['Date', target_column])
y = df[target_column]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Training
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared (R2) Score: {r2}")
```

Ouptut for model training:

```
Mean Squared Error: 0.049454099596285155
R-squared (R2) Score: 0.9999419068878214
```

Model Evaluation code :

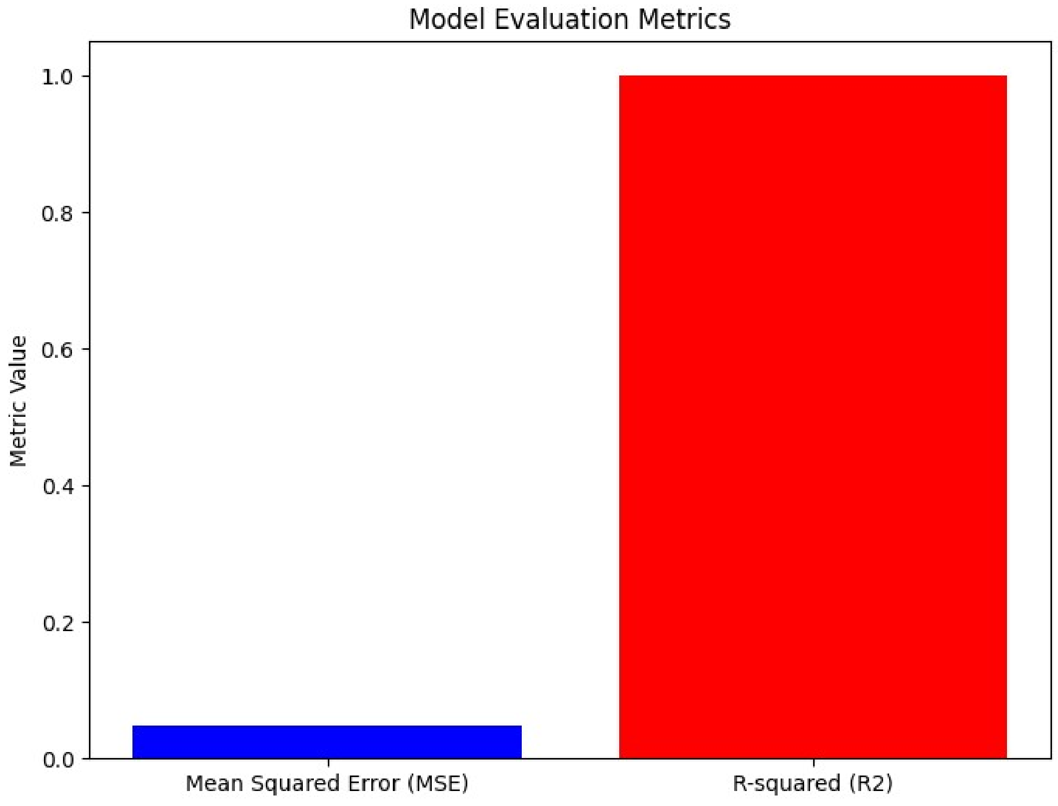
```
import matplotlib.pyplot as plt

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Create a bar chart for evaluation metrics
metrics = ['Mean Squared Error (MSE)', 'R-squared (R2)']
values = [mse, r2]

plt.figure(figsize=(8, 6))
plt.bar(metrics, values, color=['blue', 'red'])
plt.ylabel('Metric Value')
plt.title('Model Evaluation Metrics')
plt.show()
```

Model evaluation output:



Conclusion:

In this project, we undertook the challenging task of predicting stock prices using various machine learning and data analysis techniques. We collected historical stock price data, performed feature engineering, and implemented different models to make predictions. Our project aimed to understand the dynamics of stock prices and to develop a predictive model that could assist investors and traders in making informed decisions.