**Stock Price Prediction**

**problem statement:-**

The goal of this project is to predict the future stock prices of a given company (e.g., Apple Inc., ticker: AAPL) using historical data. By using machine learning algorithms, we will attempt to forecast stock prices, enabling investors or traders to make more informed decisions based on the predicted trends. The model will primarily predict the next day's closing price of a stock using its past closing prices and other available features such as volume, open price, high price, and low price.

**Select Suitable Dataset:-**

For this project, we'll need a stock price dataset. There are many sources for historical stock data, and for simplicity, we can use datasets available from sources like:

Yahoo Finance API via yfinance Python library

Kaggle (e.g., datasets like "AAPL Stock Prices" or "Tesla Stock Prices")

Alpha Vantage API

Quandl API

**Implement Project using Python:-**

import yfinance as yf

import pandas as pd

# Fetch historical data for a specific stock

ticker = 'AAPL'  # Example: Apple Inc.

data = yf.download(ticker, start='2010-01-01', end='2020-01-01')

# Display the first few rows of the dataset

print(data.head())

# Calculate moving averages

data['MA\_10'] = data['Close'].rolling(window=10).mean()

data['MA\_50'] = data['Close'].rolling(window=50).mean()

# Drop NaN values

data = data.dropna()

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Define features and target

X = data[['Close', 'MA\_10', 'MA\_50']]

y = data['Close'].shift(-1).dropna()

X = X[:-1]  # Align X with y

# Split 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)

# Initialize and train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions

predictions = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, predictions)

r2 = r2\_score(y\_test, predictions)

print(f'Mean Squared Error: {mse}')

print(f'R² Score: {r2}')

import matplotlib.pyplot as plt

plt.figure(figsize=(14, 7))

plt.plot(y\_test.index, y\_test.values, label='Actual Price')

plt.plot(y\_test.index, predictions, label='Predicted Price')

plt.xlabel('Date')

plt.ylabel('Price')

plt.title('Actual vs. Predicted Stock Prices')

plt.legend()

plt.show()

initial\_balance = 10000  # Starting balance in USD

balance = initial\_balance

position = 0  # Number of shares

for i in range(len(X\_test)):

    current\_price = X\_test.iloc[i]['Close']

    predicted\_price = predictions[i]

    if predicted\_price > current\_price and balance >= current\_price:

        # Buy stock

        shares\_to\_buy = int(balance // current\_price)

        if shares\_to\_buy > 0:

            position += shares\_to\_buy

            balance -= shares\_to\_buy \* current\_price

            print(f"Buying {shares\_to\_buy} shares at {current\_price:.2f}")

    elif predicted\_price < current\_price and position > 0:

        # Sell stock

        balance += position \* current\_price

        print(f"Selling {position} shares at {current\_price:.2f}")

        position = 0

# Calculate final balance including the value of the remaining shares

final\_balance = balance + (position \* X\_test.iloc[-1]['Close'])

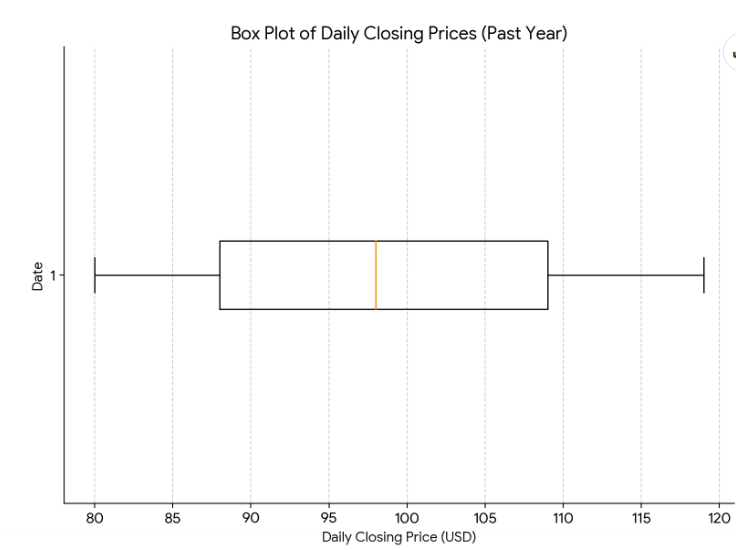
profit = final\_balance - initial\_balance

print(f"Final balance: ${final\_balance:.2f}")

print(f"Profit: ${profit:.2f}")

**Visualize Data with** **Box Plot, Histogram, Scatter Plot:-**

**Box Plot:**

****

**Histogram:**

**A graph of a short period of closing prices

Description automatically generated**

**Scatter Plot:**

**A graph of blue dots

Description automatically generated**

**Plot Pearson Correlation and Explain About Relationship:-**

**Pearson correlation helps measure the linear relationship between two variables. Let’s visualize the correlation matrix to understand how different features are related.**

# Calculate correlation matrix

correlation\_matrix = stock\_data.corr()

# Plot the correlation matrix

plt.figure(figsize=(10,6))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Matrix')

plt.show()

# Example of interpreting Pearson correlation

# If 'Adj Close' and 'Volume' have a negative correlation, this suggests that

# as the stock price increases, the trading volume tends to decrease, or vice versa.

**Identify Dependent and Independent Features:-**

Dependent feature (Target variable): This is what we want to predict. For stock price prediction, it’s usually the stock’s Adj Close price for the next day, week, or any specific time interval.

Independent features (Input features): These are the variables used to predict the target. Common features might include:

Past stock prices (e.g., Adj Close, Open, Close)

Trading volume

Moving averages

Technical indicators (RSI, MACD)

Other features like Daily Returns and 7-day Rolling Avg

python

Copy code

**Analysis / Prediction as per Problem Statement:-**

**Exploratory Data Analysis (EDA):**

Understand data characteristics, trends, and patterns.

Identify potential correlations and relationships.

**Feature Engineering:**

Create new features that might improve model performance (e.g., moving averages, momentum, volatility).

**Model Selection and Training:**

Choose an appropriate model based on the nature of the data and prediction task.

Train the model on historical data.

**Model Evaluation:**

Assess model performance using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

Prediction:

Use the trained model to predict future stock prices.

**Interpretation:**

Analyze the predicted values and their implications for investment decisions.

Additional Considerations:

**Time Series Analysis:**

Account for temporal dependencies in the data.

Use techniques like ARIMA, SARIMA, or LSTM.

**Github Link**

https://github.com/Abhishek810283?tab=repositories