Project Title:

Stock Price Prediction Using Machine Learning

Through the courses:

- Web Application Development
- Fundamentals of Data Science
- Data Engineering

Students’ Names:

- **Ahmed Nabil Hamza Khadr (Leader)**
- **Eman Ashraf El-Sayed**
- **Mirna Salah Saad**
- **Zeinab Ramadan Fathy**
- **Nada Ramadan Abdel-Maksoud**
- **Mohamed Ismail Emam**
- **Amr Khaled Mohamed**
- **Omar Ahmed Mohamady**
- **Ahmed Emad Sayed**
- **Ammar Yasser Shehat**
- **Noha Mohamed Mahmoud**
- **Belal Ali Lotfy**
- **Ahmed Essam Abu Zeid**

- **Mohamed Adel Mohamed**
- **Mohamed Sayed Mohamed**
- **Ahmed Mohamed El-Sayed Farah**
- **Fatma Ahmed Mahmoud**
- **Sondos Ahmed Mohamed**
- **Gana Allah Mahmoud Radwan**
- **Youssef Mohamed Hamed**
- **Abdallah Abu Taleb**

Instructor:
Dr. Sherif Ibrahim

Introduction:
This project, titled **"Stock Price Prediction Using Machine Learning,"** focuses on predicting the future stock prices of Apple Inc. (AAPL) using artificial intelligence techniques. The model learns from historical stock data to forecast future price values, similar to how investors analyze market trends.
The goal is to apply machine learning algorithms to real-world financial data and evaluate how accurately they can predict price movements.

The workflow included collecting data using the **yfinance** library, cleaning the dataset, performing **Exploratory Data Analysis (EDA)**, training ML models such as **RandomForestRegressor** and **XGBoost**, and finally building an interactive **Streamlit** web interface.

This system allows users to visualize real-time stock data, review model performance (MAE, RMSE, R^2), and generate predictions.
Although stock prices cannot be predicted with 100% accuracy due to external events, this project shows how data-driven models can support investment decisions.

Project Steps:

1. Data Collection
2. Data Cleaning & Preprocessing
3. Exploratory Data Analysis (EDA)
4. Model Building
5. Model Evaluation
6. Streamlit UI Development

Step 1: Collecting Apple Stock Data

In this step, real historical stock data for Apple (AAPL) was collected using the **yfinance** Python library. The dataset includes *Date, Open, High, Low, Close,* and *Volume*. After downloading the data, the index was reset and the dataset was saved as **stock_data.csv**.

Code Explanation:

• yf.download("AAPL", start="2020-01-01", end="2025-11-25") retrieves Apple's stock data from Yahoo Finance.

• data.reset_index(inplace=True) makes Date a separate column.

• The dataset was cleaned and saved using data.to_csv().

Output:

A CSV file (**stock_data.csv**) containing Date, Open, High, Low, Close, and Volume.

Step 2: Data Cleaning & Sentiment Collection

This step prepares and cleans the dataset so the machine learning model can process it correctly. Missing values were filled, dates were ordered, and all values were converted to numeric types. **News sentiment** was also added to the dataset.

Code Explanation:

• data.fillna(method="ffill") fills missing values.

• **GoogleNews** was used to collect Apple-related news headlines.

• Each headline was analyzed using **TextBlob** to calculate sentiment polarity.

• Daily average sentiment was merged with the stock dataset.

• The final dataset was saved as **clean_data.csv**.

Output:

A file (**clean_data.csv**) containing all stock price data plus an additional column: **Sentiment**.

Why This Step Matters:

Clean data improves model accuracy.

Adding sentiment helps the model capture emotional market reactions that influence stock movements.

Step 3: Exploratory Data Analysis (EDA)

In this step, trends, patterns, correlations, and anomalies in the data were analyzed. Multiple visualizations were generated to understand stock price behavior and the effect of sentiment.

Code Explanation:
• Line plot showing closing price over time → **price_plot.png**
• Histogram showing distribution of closing prices → **price_distribution.png**
• Scatter plot showing sentiment vs stock price → **sentiment_plot.png**
• Heatmap showing correlations between numerical features → **correlation_heatmap.png**

Why This Step Matters:
EDA helps understand data structure before model training, detects anomalies, and supports better model-building decisions.

Generated Outputs:

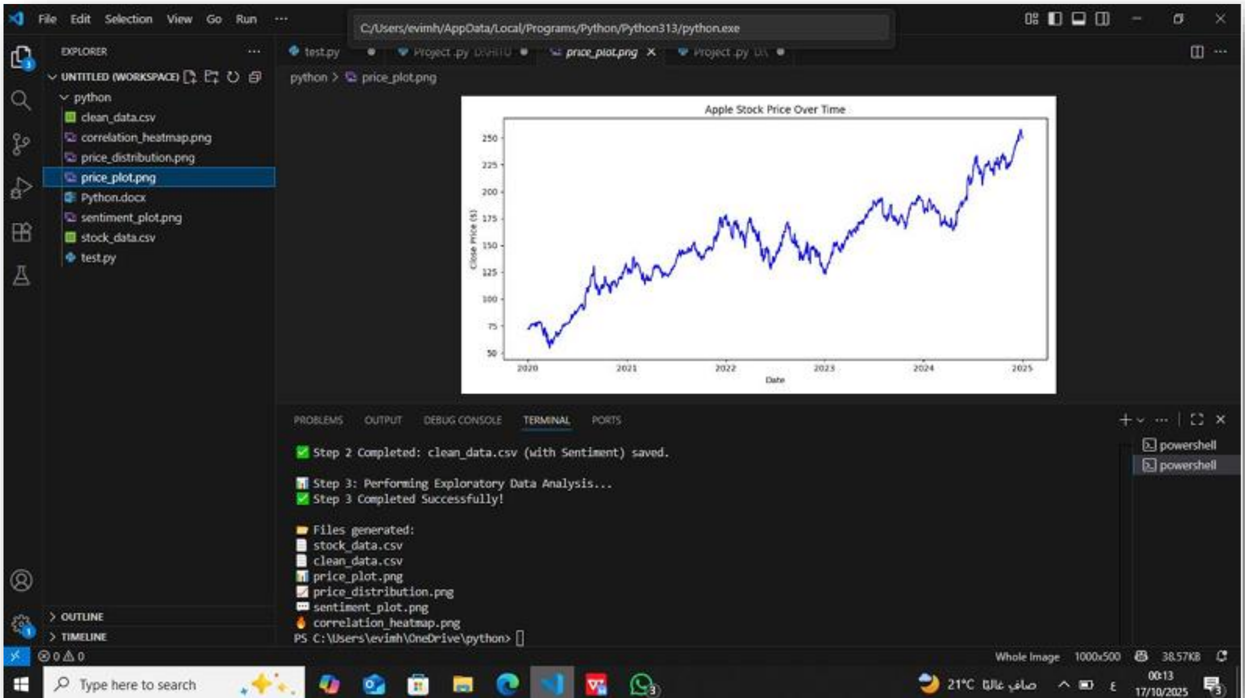
1. **price_plot.png** — Apple stock price trend
2. **price_distribution.png** — distribution of closing prices
3. **sentiment_plot.png** — relationship between sentiment and stock price

4. **correlation_heatmap.png** — correlations between numerical features such as Open, High, Low, Close, Volume, and Sentiment → Output: **correlation_heatmap.png**

This Step Matters:
Exploratory Data Analysis helps us understand whether the data is stable, shows an upward or downward trend, or contains anomalies. This understanding allows us to prepare the data more effectively and make better modeling decisions later.

Output:
The following visualizations were generated and saved:

1. **price_plot.png → Apple stock price trend over time**
   Description: Shows a time series of the stock price (likely Apple) from 2020 to 2025.
   X-axis: Date
   Y-axis: Closing Price
   Meaning: Shows how the stock price changes over time—i.e., the market trend (upward, downward, or stable).



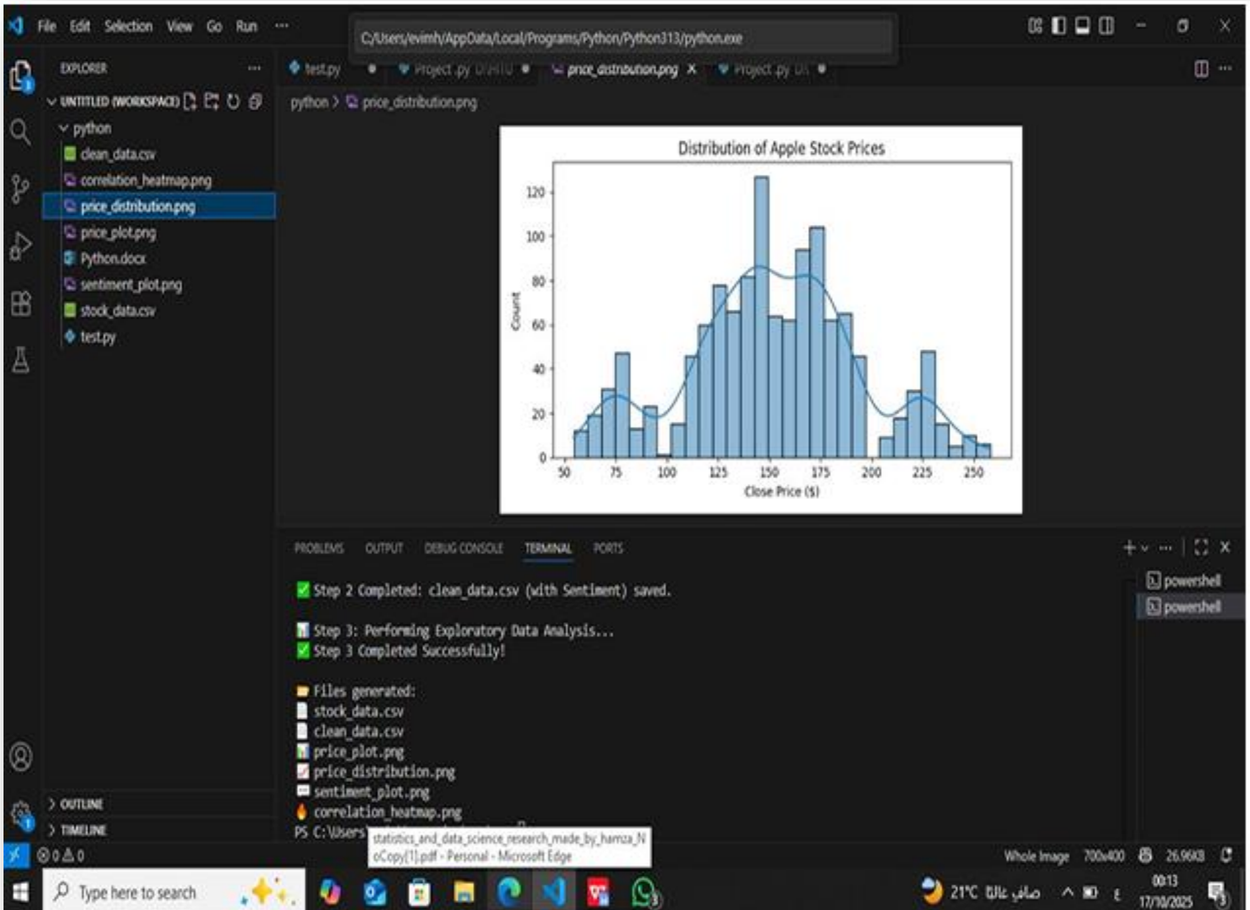
2. price_distribution.png → Distribution of closing prices

Description:

A distribution plot showing how the closing prices are spread over time.

Meaning:

Helps understand whether the prices are centered around a certain value or have high variability.



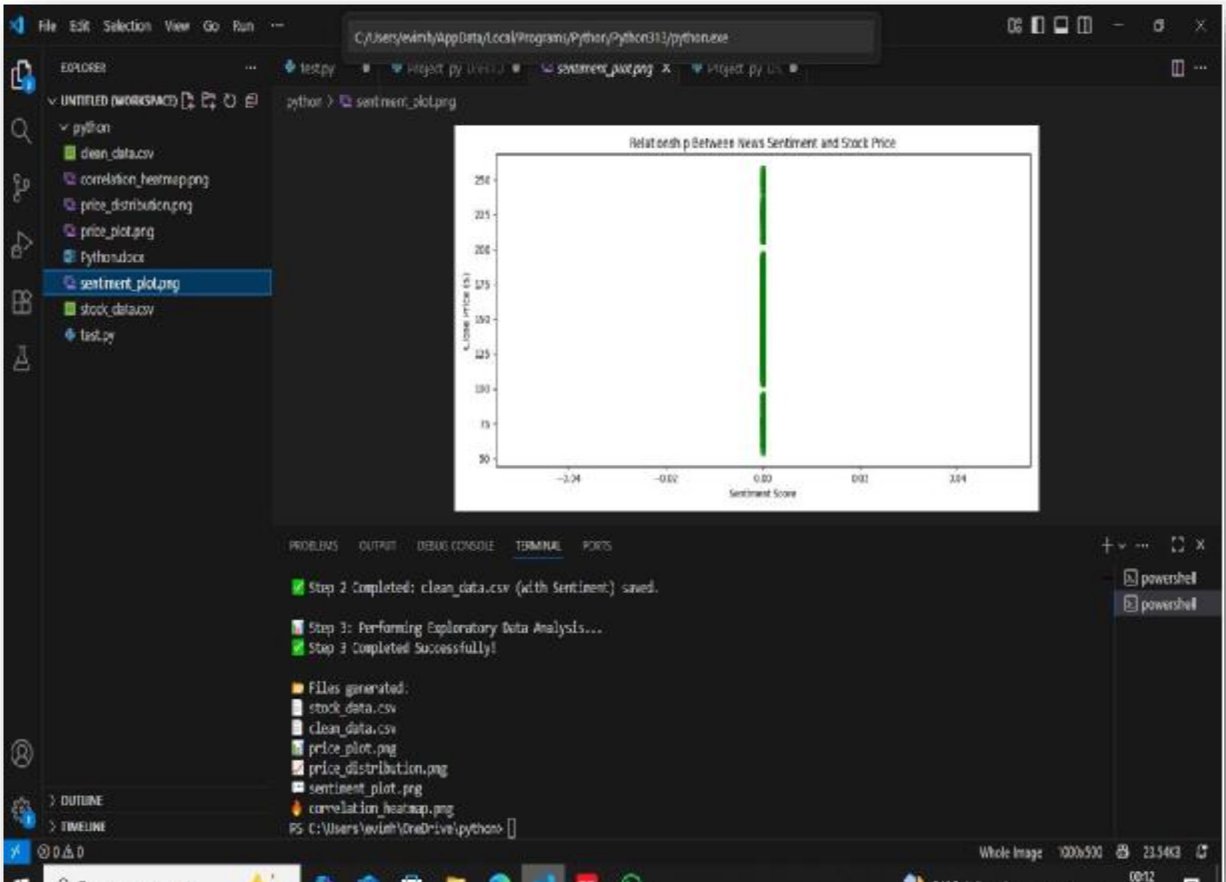
3. sentiment_plot.png → Relationship between sentiment and price

Description:
Shows the relationship between sentiment analysis (Sentiment Score) from news or tweets and the stock price.

X-axis: Sentiment score (from negative to positive)
Y-axis: Closing Price

Meaning:

If the points are clustered around zero, it means sentiment analysis has little effect on the price, or the data is biased toward neutral values.



4. correlation_heatmap.png → Correlation between all numerical variables

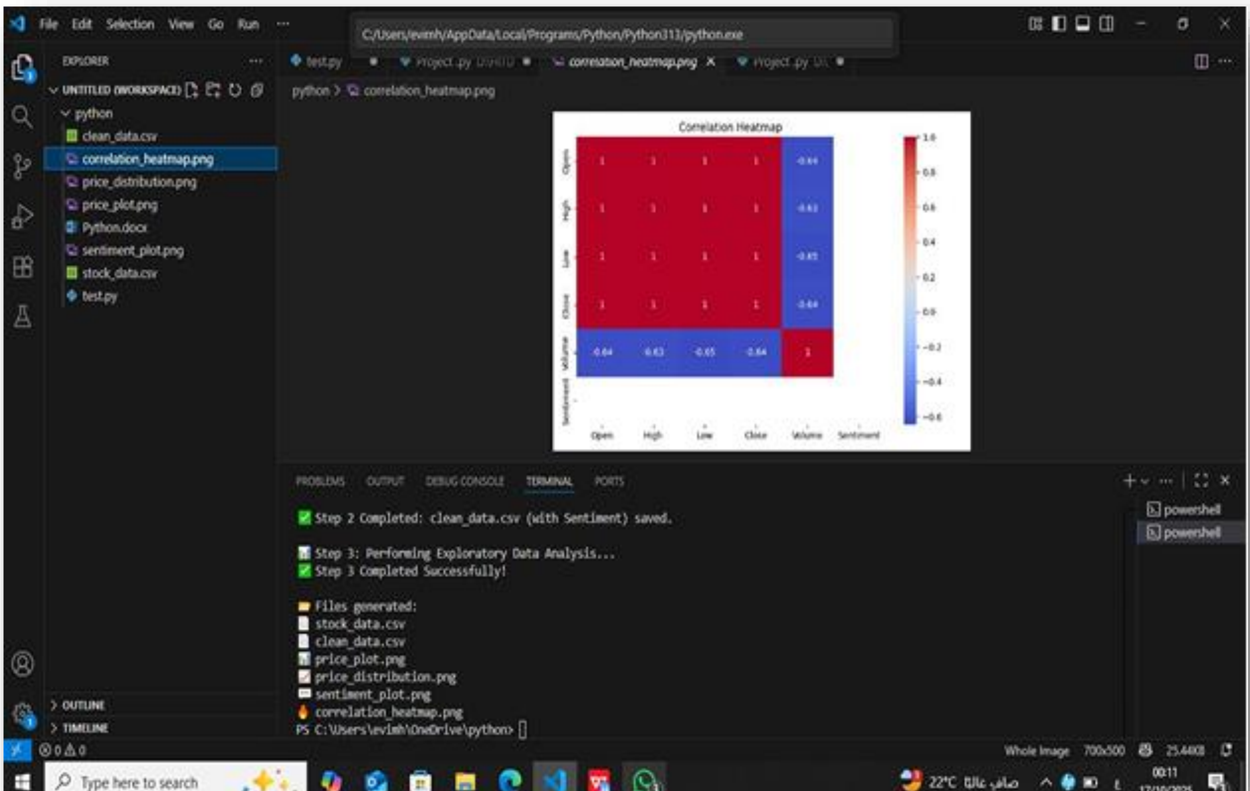
Description:

A heatmap showing the correlations between numerical variables (Open, High, Low, Close, Volume, Sentiment).

Colors:

- **Red** = strong positive correlation (as one increases, the other increases).
- **Blue** = negative correlation (as one increases, the other decreases).

Meaning:
From the image, the stock prices (Open, High, Low, Close) are highly correlated (close to 1), while Sentiment shows a slight negative correlation.



Model Building

The model development process included the following steps:

1. Data Preparation

- Loading historical stock data using **yfinance**.
- Extracting the **closing prices** and cleaning the dataset, including:
    - Handling missing values.
    - Correcting formatting and ensuring consistency.
- These steps ensure that the data is ready for model training.

2. Feature Engineering

- Creating a supervised learning structure by shifting the closing price values:
    - **X = current day's closing price**
    - **y = next day's closing price**
- Reshaping and preparing the data to match the input requirements of the model.
- This allows the model to learn the relationship between consecutive days.

3. Model Selection and Training

- **Linear Regression** was chosen as the baseline model due to its simplicity, speed, and interpretability.
- The dataset was split into **training and testing sets** to evaluate performance.
- The model was trained on historical closing price data to predict future values.
- This step enables the model to capture basic trends in the stock price.

Model Evaluation

To ensure reliability and accuracy, the following steps were applied:

1. Performance Metrics

- Calculating **Mean Absolute Error (MAE)** and **R^2 Score** to measure prediction accuracy.
- Comparing **predicted values** with **actual closing prices** to quantify performance.

2. Validation

- Visualizing **predicted vs. actual prices** to ensure that the model follows real price trends.
- Checking for **overfitting** to confirm that the model generalizes well to unseen data.

3. Output Formatting

- Producing a **clean, tabular output** showing predicted closing prices for upcoming days.
- Making the results easy to **interpret and use** for further analysis or decision-making.

💡 **Additional notes for report:**

- You can include a **small plot** of actual vs predicted prices under this section to make it more visual.

- Mention that this is a **baseline model**, and more advanced models (like LSTM) can improve predictions in future work.

➢ **Streamlit UI Development**

The Streamlit UI Development step focuses on creating a **simple and interactive user interface** for the stock price prediction model.

1. Purpose

- Provide an **easy-to-use interface** for users to input stock tickers and prediction parameters.
- Display **predicted stock prices**, tables, and charts directly in a web page.
- Allow users to interact with the model **without coding knowledge**.

2. Implementation

- Streamlit is a **Python library** that enables rapid development of data apps.

- Users can select a **stock ticker** and the **number of days** for prediction.
- The app dynamically **runs the model** and shows the results, including visualizations and tables.

3. Advantages

- **Easy and fast** to develop without HTML/JS knowledge.
- Supports **interactive inputs** (text, sliders, checkboxes).
- Provides **visual feedback** for better understanding of model predictions.

4. Example Code Snippet

```
import streamlit as st

import pandas as pd


st.title("Stock Price Prediction App")


ticker = st.text_input("Enter Stock Ticker", "AAPL")

days = st.slider("Days to Predict", 1, 30, 5)


st.write("Predicted prices will appear here...")
```

5. Summary

- Streamlit allows presenting the model **professionally** for demonstration purposes.

- It enhances the project by providing a **user-friendly interface** for non-technical users.
- This step is **optional** but strongly recommended for showcasing the project.