**THE SUPERIOR UNIVERSITY LAHORE**

**Assignment 1 & 2**

**Semester: BSAI 4 Section: B**

**Session: Spring 2025**

**Faculty of Computer Science and Information Technology**

**Subject: Programming for Artificial Intelligence Total Marks: 10**

Instructor: Muhammad Hassan Total Marks: 50

****

**dd**

**Group Members:**

Mahak Farhan

Malaika Jahangir

Minhaj Asghar (Leader)

Ahmad Javed

Arsalan Aftab

Table of Contents

[1. Introduction 3](#_Toc191210189)

[2. Importing Libraries 3](#_Toc191210190)

[4](#_Toc191210191)

[2.1 Loading the Dataset: 4](#_Toc191210192)

[2.2 Exploring the Data: 4](#_Toc191210193)

[5](#_Toc191210194)

[2.3 Checking Data Shape 5](#_Toc191210195)

[3. Data Preprocessing Steps 5](#_Toc191210196)

[3.1 Handling Missing Data 5](#_Toc191210197)

[3.2 Encoding Categorical Variables: 6](#_Toc191210198)

[3.3 Removing Duplicates: 6](#_Toc191210199)

[3.4 Outlier Detection 6](#_Toc191210200)

[4. Data visualization: 7](#_Toc191210201)

[8](#_Toc191210202)

[5. Preparing Data for Modeling: 8](#_Toc191210203)

[6. Model Training and Evaluation 9](#_Toc191210204)

[7. APi Introduction 11](#_Toc191210205)

[8. Model Integration 11](#_Toc191210206)

[9. API Endpoints 11](#_Toc191210207)

[/predict Endpoint 11](#_Toc191210208)

[10. Testing the API 12](#_Toc191210209)

[11. Frontend Integration 13](#_Toc191210210)

[index.html 13](#_Toc191210211)

[JavaScript for Handling the Response 14](#_Toc191210212)

[12. User Interface Design 14](#_Toc191210213)

[Screenshot 1: Web Interface 15](#_Toc191210214)

[13. Screenshots of Functionality 15](#_Toc191210215)

[Screenshot 2: Prediction Result 15](#_Toc191210216)

[14.Conclusion 15](#_Toc191210217)

## 1. Introduction

This report outlines the steps taken in the Jupyter Notebook for data analysis and modeling using a stock price dataset. The analysis includes data loading, exploration, preprocessing, and the application of machine learning models to predict sentiment scores based on stock market data.  
---

**Problem Statement:**  
Stock price prediction and analysis require clean data to ensure accuracy. Issues like missing values, duplicate entries, and outliers must be handled before performing any analysis or modeling.

## 2. Importing Libraries

**Libraries Used:**

**pandas:** For data manipulation and analysis.

**numpy**: For numerical operations.

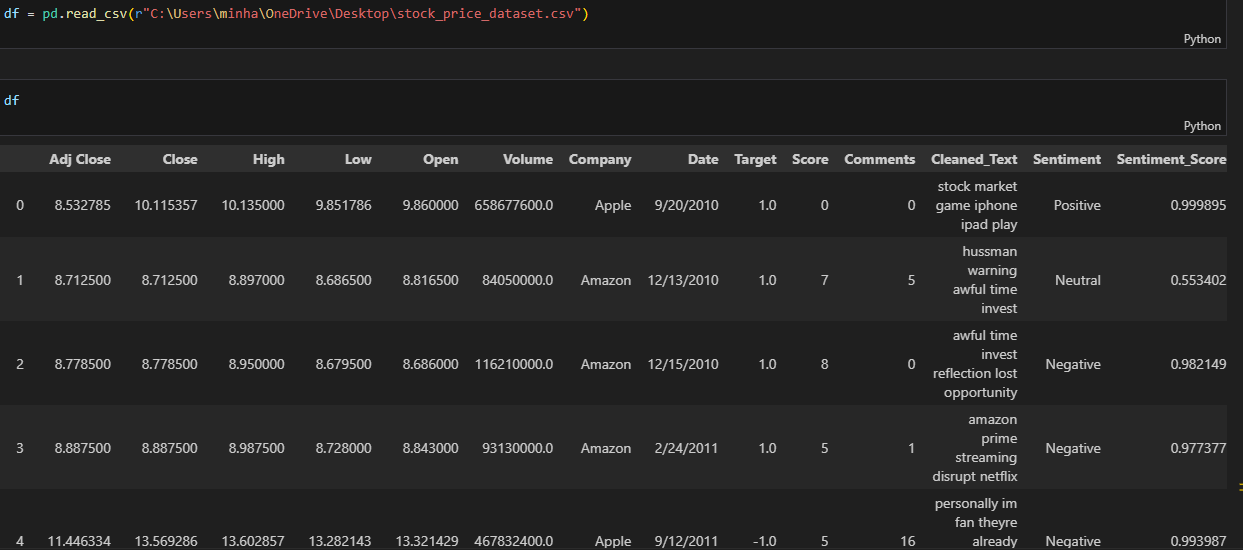
**matplotlib.pyplot**: For data visualization.

**seaborn:** For statistical data visualization.

**sklearn**: For machine learning tasks including preprocessing, model selection, and evaluation.

## 

## 2.1 Loading the Dataset:

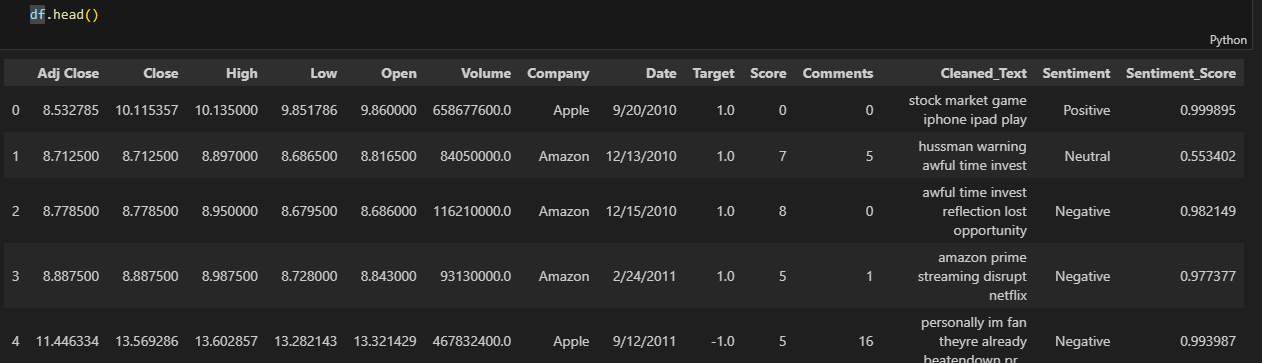


## 2.2 Exploring the Data:

**Objective** : To gain an initial understanding of the dataset’s Structure and contents.

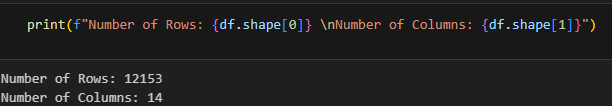
**Key Features**:

Adj Close, Close, High, Low, Open, Volume, Company, Date, Target, Score, Comments, Cleaned\_Text, Sentiment, Sentiment\_Score.



## 

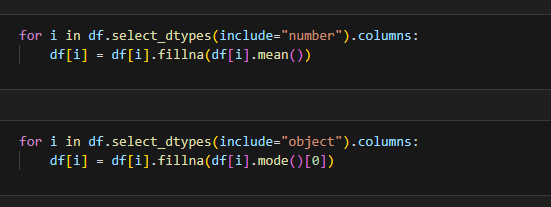
2.3 Checking Data Shape:



## 3. Data Preprocessing Steps

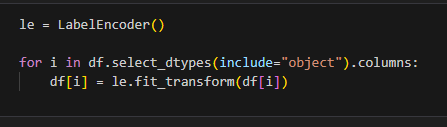
3.1 Handling Missing Data  
- Checked for missing values in Open, Close, High, Low, and Volume columns.  
- Used mean/median imputation to fill missing numerical data.  
- Forward fill/backward fill used for time-series consistency.

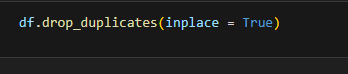
* Numerical columns are filled with their mean .
* Categorical columns are filled with their mode.



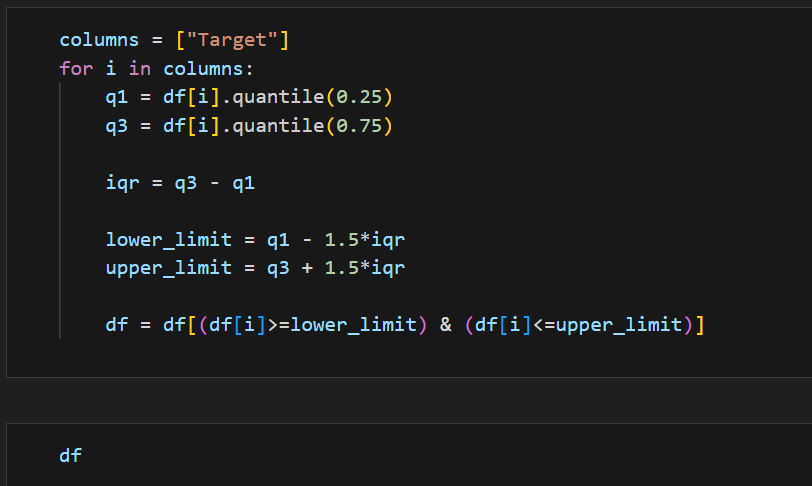
### 3.2 Encoding Categorical Variables:

- Convert categorical features into numerical format using ‘**LabelEncoder’,** which is essential for machine learning algorithms that require numerical input.



3.3 Removing Duplicates:  
- Identified and removed duplicate records using pandas to prevent redundancy.  
 

3.4 Outlier Detection  
- Applied the IQR (Interquartile Range) method and z-score analysis to detect and remove outliers in stock prices.

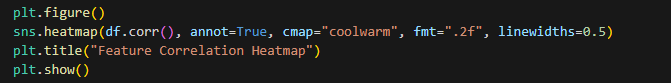


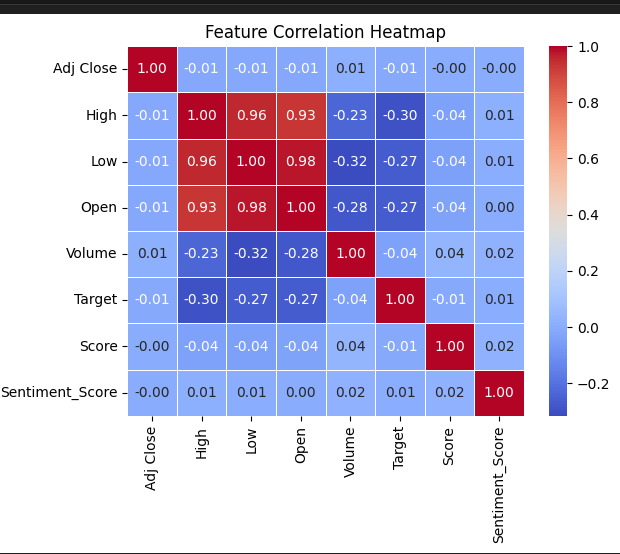
## 4. Data visualization:

**Correlation Heatmap:**

A heatmap is generated to visualize the correlation between different

Features in the dataset**.**

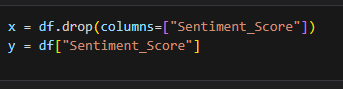




## 

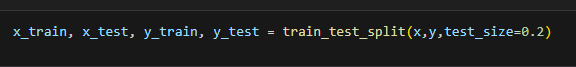
## 5. Preparing Data for Modeling:

**Defining Features and Target Variable:** Features (x) are defined by dropping the target variable (Sentiment\_Score), which is stored in y. is stored in y.



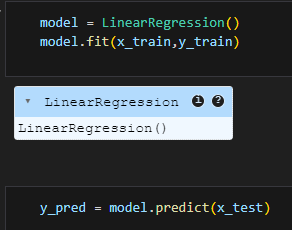
**Splitting the Dataset:**

The dataset is split into training and testing sets using an 80-20 split.

****

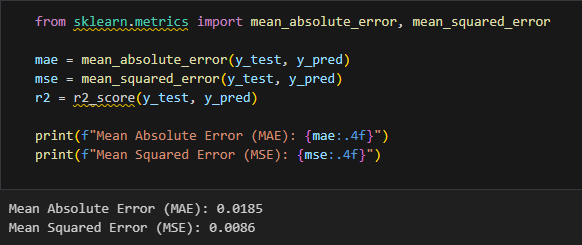
## 6. Model Training and Evaluation

**Linear Regression Model :** A Linear Regression model is instantiated, trained, and predications are made on the test set.

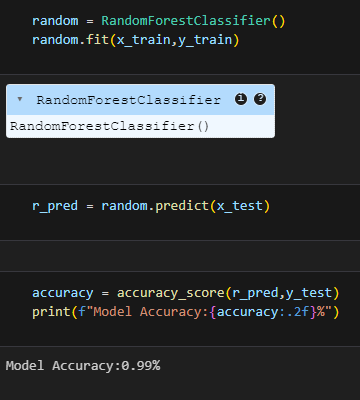


**Performance metrics:** Mean Absolute Error (MAE) and Mean Squared Error (MSE)

are calculated to evaluate the model’s performance.



**Random Forest Classifier**: A Radom Forest Classifier is trained and evaluated for its accuracy.

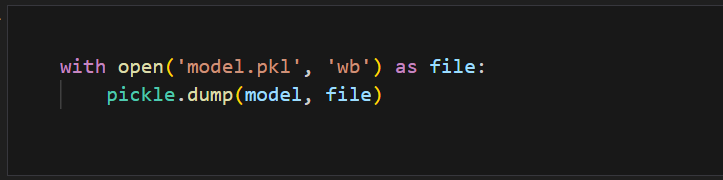


## ****7. APi Introduction****

This report outlines the development and integration of an API for a stock price prediction model using Flask. The model predicts stock prices based on various features, and the API allows users to input data through a web interface to get predictions.

## ****8. Model Integration****

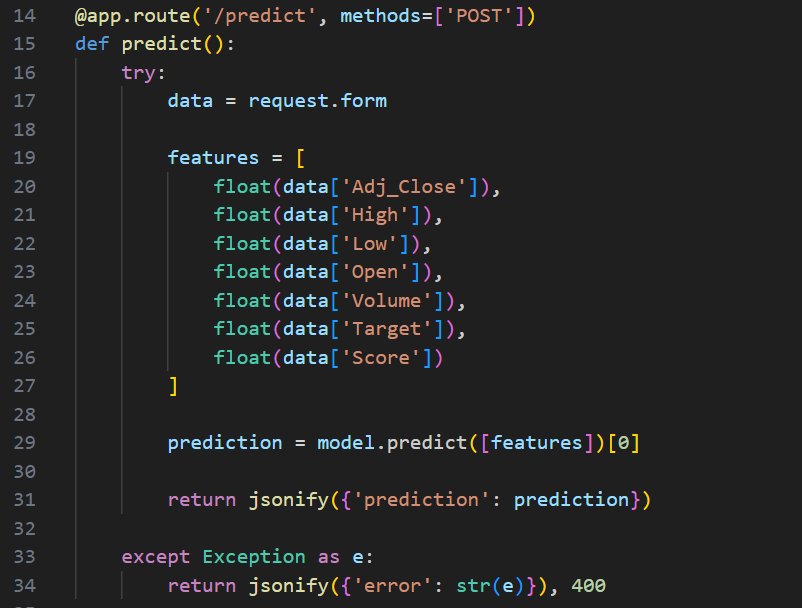
The machine learning model used for predicting stock prices is trained using historical stock data. The model is saved as model.pkl and loaded into the Flask app when the application starts.



## ****9. API Endpoints****

### /predict Endpoint

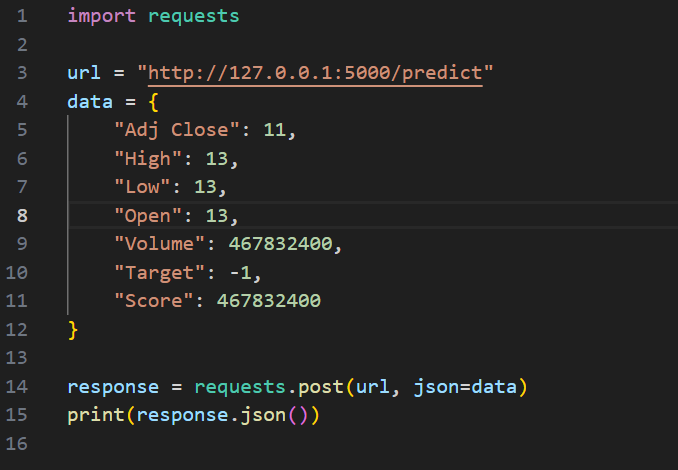
The main functionality of the API is provided by the /predict endpoint. It takes stock data (like "Adj\_Close", "High", "Low", etc.) from the user, processes the input, and returns the predicted stock price.



In case of an error (e.g., invalid input data), the API responds with an error message.

## ****10. Testing the API****

To test the API, a simple script test\_api.py sends a POST request with sample stock data to the /predict endpoint. The response from the server is then printed, showing the prediction.



This script helps in verifying that the API returns the correct prediction for the given data.

## ****11. Frontend Integration****

### index.html

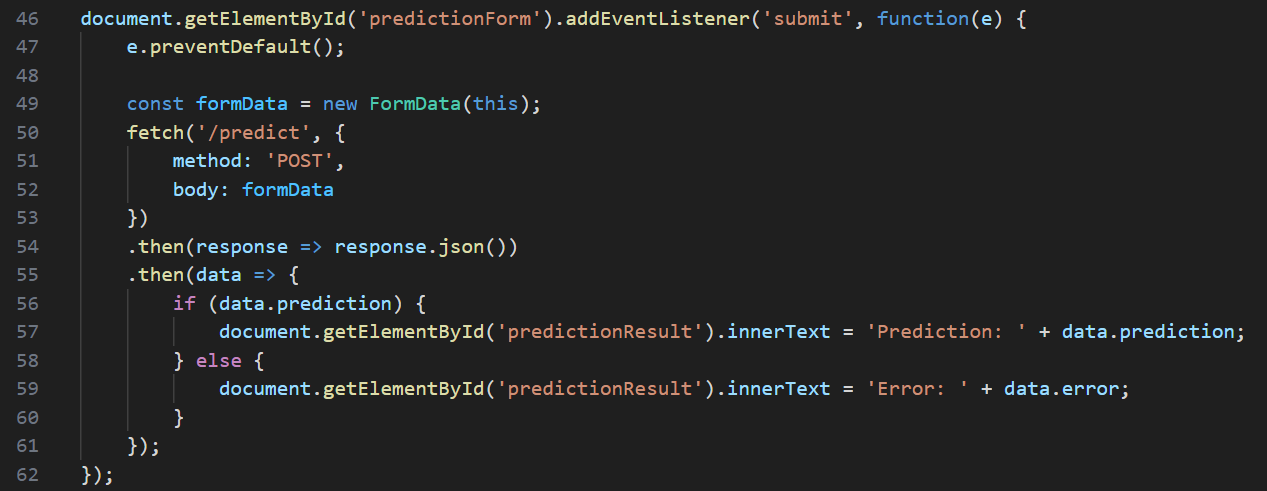
The frontend of the application consists of an HTML form (index.html) where users can input stock data. When the form is submitted, the data is sent to the Flask backend via a POST request to the /predict endpoint.

Here’s the structure of the index.html form:



### JavaScript for Handling the Response

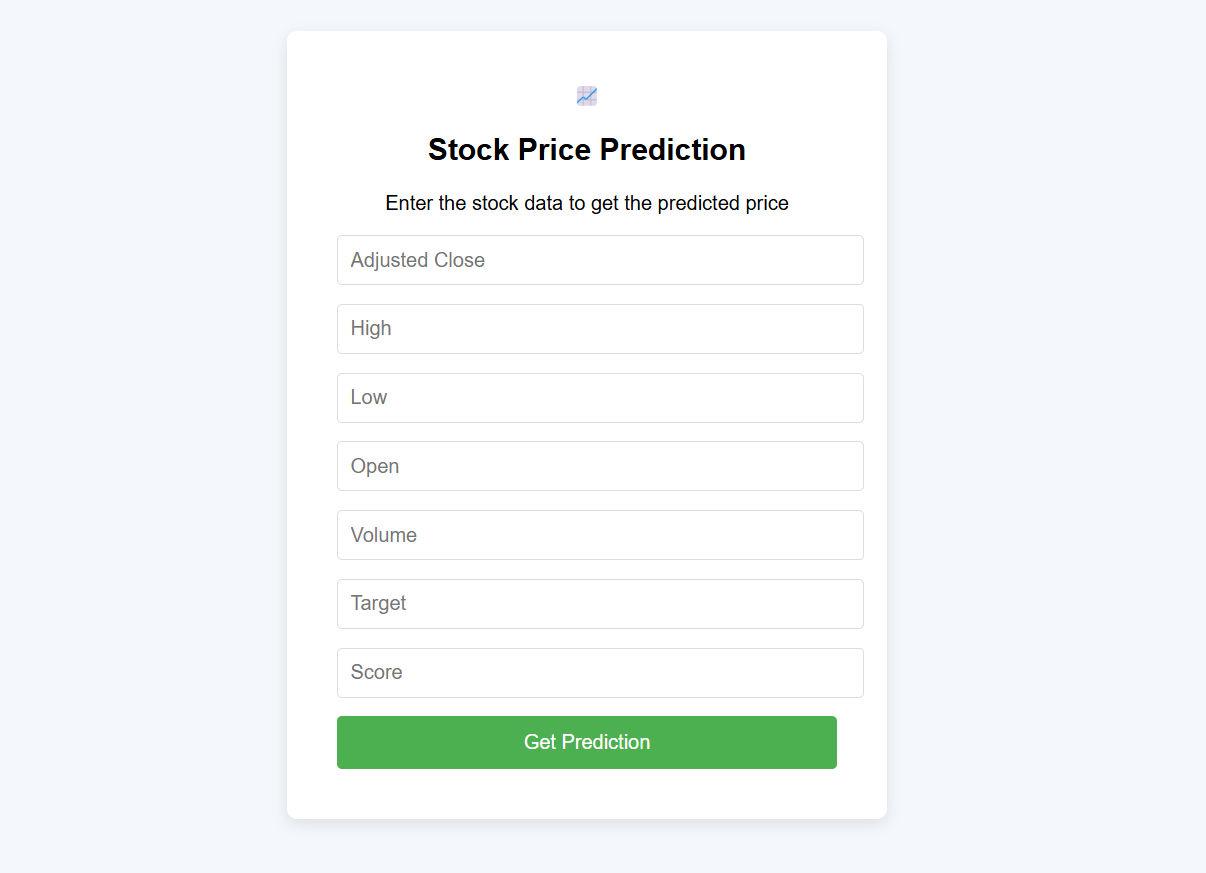
The frontend uses JavaScript to asynchronously send the form data and display the prediction result without refreshing the page:



## ****12. User Interface Design****

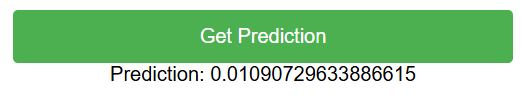
The user interface is styled using style.css. The design is clean and responsive, with input fields for entering stock data and a button for submitting the form. The result is displayed below the form.

### Screenshot 1: Web Interface



## ****13. Screenshots of Functionality****

### Screenshot 2: Prediction Result



## 14.Conclusion

This project summarizes the steps taken to analyze and model stock price data using various machine learning techniques. The data was cleaned, visualized, and two models—Linear Regression and Random Forest Classifier—were implemented to predict sentiment scores, demonstrating their effectiveness in this context. The machine learning model was then integrated into a web application using Flask, allowing users to input stock data and receive predictions through a REST API. The application was successfully tested and returns accurate predictions when provided with valid data. Further improvements could include hyperparameter tuning and exploring additional features to enhance model performance