Table of Contents

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.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

2.1 Loading the Dataset:

```
df
```

Python

	Adj Close	Close	High	Low	Open	Volume	Company	Date	Target	Score	Comments	Cleaned_Text	Sentiment	Sentiment_Score
0	8.532785	10.115357	10.135000	9.851786	9.860000	658677600.0	Apple	9/20/2010	1.0	0	0	stock market game iphone ipad play	Positive	0.999895
1	8.712500	8.712500	8.897000	8.686500	8.816500	84050000.0	Amazon	12/13/2010	1.0	7	5	hussman warning awful time invest	Neutral	0.553402
2	8.778500	8.778500	8.950000	8.679500	8.686000	116210000.0	Amazon	12/15/2010	1.0	8	0	awful time invest reflection lost opportunity	Negative	0.982149
3	8.887500	8.887500	8.987500	8.728000	8.843000	93130000.0	Amazon	2/24/2011	1.0	5	1	amazon prime streaming disrupt netflix	Negative	0.977377
4	11.446334	13.569286	13.602857	13.282143	13.321429	467832400.0	Apple	9/12/2011	-1.0	5	16	personally im fan theyre already	Negative	0.993987

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.

```python
df.head()
```

Python

	Adj Close	Close	High	Low	Open	Volume	Company	Date	Target	Score	Comments	Cleaned_Text	Sentiment	Sentiment_Score
0	8.532785	10.115357	10.135000	9.851786	9.860000	658677600.0	Apple	9/20/2010	1.0	0	0	stock market game iphone ipad play	Positive	0.999895
1	8.712500	8.712500	8.897000	8.686500	8.816500	84050000.0	Amazon	12/13/2010	1.0	7	5	hussman warning awful time invest	Neutral	0.553402
2	8.778500	8.778500	8.950000	8.679500	8.686000	116210000.0	Amazon	12/15/2010	1.0	8	0	awful time invest reflection lost opportunity	Negative	0.982149
3	8.887500	8.887500	8.987500	8.728000	8.843000	93130000.0	Amazon	2/24/2011	1.0	5	1	amazon prime streaming disrupt netflix	Negative	0.977377
4	11.446334	13.569286	13.602857	13.282143	13.321429	467832400.0	Apple	9/12/2011	-1.0	5	16	personally im fan theyre already beatendown pr	Negative	0.993987

```python
df.tail()
```

Python

	Adj Close	Close	High	Low	Open	Volume	Company	Date	Target	Score	Comments	Cleaned_Text	Sentiment	Sentiment_Score
12148	170.000000	170.000000	171.600006	169.781494	170.042496	66316000.0	Amazon	4/13/2021	1.0	44	43	thought viacom need recap margin call fiasco k...	Neutral	0.931414
12149	254.106674	254.106674	254.333328	236.886673	237.566666	133958400.0	Tesla	4/13/2021	-1.0	1324	922	ride brief overview lordstown motor corp ride ...	Positive	0.999995
12150	254.106674	254.106674	254.333328	236.886673	237.566666	133958400.0	Tesla	4/13/2021	-1.0	1	8	bought 13 may 21 2021 put f thought taking spr...	Positive	0.998987
12151	244.076660	244.076660	260.263336	242.676666	256.899994	147052200.0	Tesla	4/14/2021	-1.0	56	905	daily discussion run monday friday including t...	Positive	0.999992
12152	244.076660	244.076660	260.263336	242.676666	256.899994	147052200.0	Tesla	4/14/2021	-1.0	26	76	ive tried estimate 10year return	Positive	0.998830

2.3 Checking Data Shape:

```python
print(f"Number of Rows: {df.shape[0]} \nNumber of Columns: {df.shape[1]}")
```

```
Number of Rows: 12153
Number of Columns: 14
```

---

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.

```python
for i in df.select_dtypes(include="number").columns:
    df[i] = df[i].fillna(df[i].mean())
```

```python
for i in df.select_dtypes(include="object").columns:
    df[i] = df[i].fillna(df[i].mode()[0])
```

3.2 Encoding Categorical Variables:

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

```python
le = LabelEncoder()

for i in df.select_dtypes(include="object").columns:
    df[i] = le.fit_transform(df[i])
```

3.3 Removing Duplicates:

- Identified and removed duplicate records using pandas to prevent redundancy.

```python
df.drop_duplicates(inplace = True)
```

3.4 Outlier Detection

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

```
columns = ["Target"]
for i in columns:
    q1 = df[i].quantile(0.25)
    q3 = df[i].quantile(0.75)

    iqr = q3 - q1

    lower_limit = q1 - 1.5*iqr
    upper_limit = q3 + 1.5*iqr

    df = df[(df[i]>=lower_limit) & (df[i]<=upper_limit)]
```

```
df
```

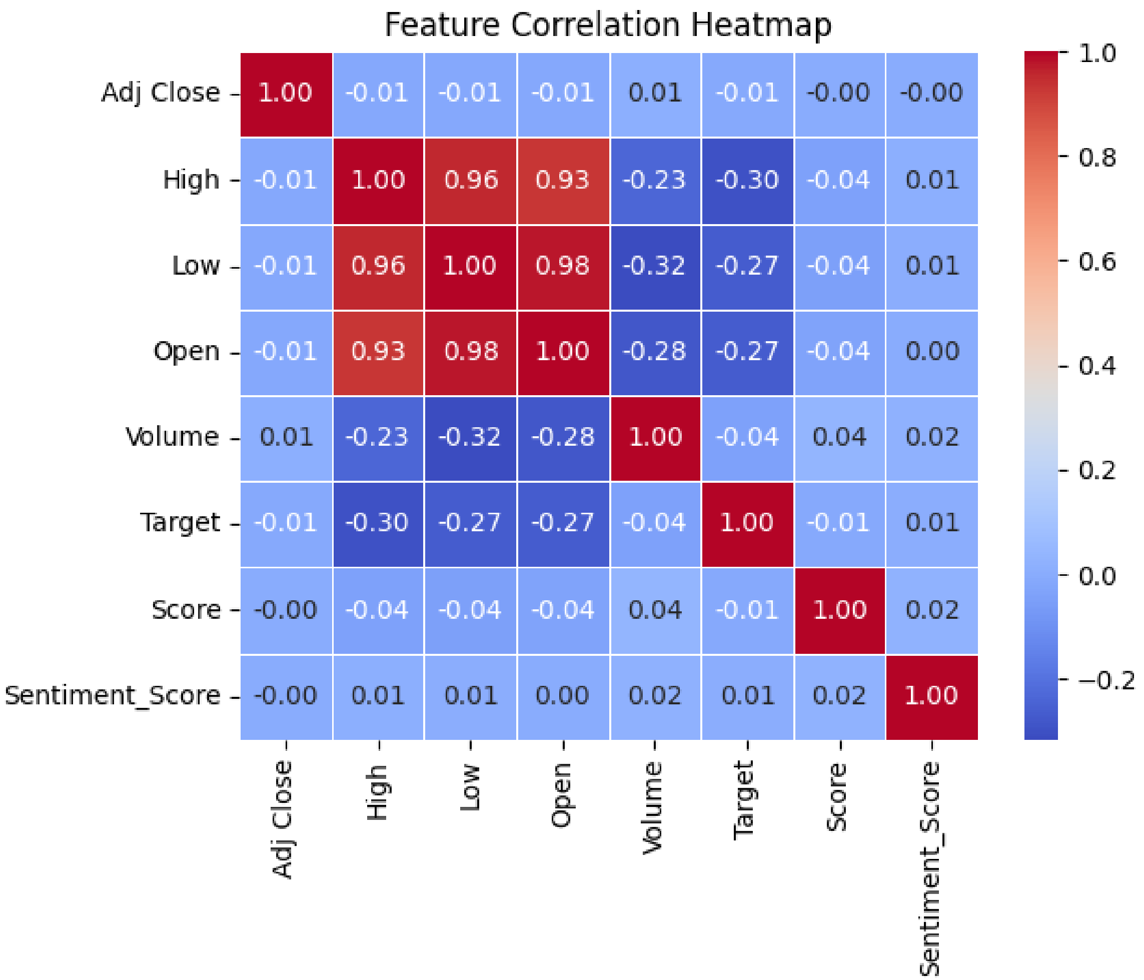
---

4. Data visualization:

Correlation Heatmap:

A heatmap is generated to visualize the correlation between different Features in the dataset.

```
plt.figure()
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```



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.

```
x = df.drop(columns=["Sentiment_Score"])
y = df["Sentiment_Score"]
```

Splitting the Dataset:

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

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

---

6. Model Training and Evaluation

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

```
model = LinearRegression()
model.fit(x_train,y_train)
```

```
LinearRegression
LinearRegression()
```

```
y_pred = model.predict(x_test)
```

Performance metrics: Mean Absolute Error (MAE) and Mean Squared Error (MSE) are calculated to evaluate the model's performance.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
```

```
Mean Absolute Error (MAE): 0.0185
Mean Squared Error (MSE): 0.0086
```

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

```
random = RandomForestClassifier()
random.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
r_pred = random.predict(x_test)
```

```
accuracy = accuracy_score(r_pred,y_test)
print(f"Model Accuracy:{accuracy:.2f}%")
```

```
Model Accuracy:0.99%
```

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.

```python
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

---

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.

```python
@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.form

        features = [
            float(data['Adj_Close']),
            float(data['High']),
            float(data['Low']),
            float(data['Open']),
            float(data['Volume']),
            float(data['Target']),
            float(data['Score'])
        ]

        prediction = model.predict([features])[0]

        return jsonify({'prediction': prediction})

    except Exception as e:
        return jsonify({'error': str(e)}), 400
```

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.

```python
import requests

url = "http://127.0.0.1:5000/predict"
data = {
    "Adj Close": 11,
    "High": 13,
    "Low": 13,
    "Open": 13,
    "Volume": 467832400,
    "Target": -1,
    "Score": 467832400
}

response = requests.post(url, json=data)
print(response.json())
```

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:

```html
    <form action="/predict" method="POST" id="predictionForm">
        <div class="input-group">
            <input type="number" id="Adj_Close" name="Adj_Close" placeholder="Adjusted Close" required>
        </div>
        <div class="input-group">
            <input type="number" id="High" name="High" placeholder="High" required>
        </div>
        <div class="input-group">
            <input type="number" id="Low" name="Low" placeholder="Low" required>
        </div>
        <div class="input-group">
            <input type="number" id="Open" name="Open" placeholder="Open" required>
        </div>
        <div class="input-group">
            <input type="number" id="Volume" name="Volume" placeholder="Volume" required>
        </div>
        <div class="input-group">
            <input type="number" id="Target" name="Target" placeholder="Target" required>
        </div>
        <div class="input-group">
            <input type="number" id="Score" name="Score" placeholder="Score" required>
        </div>
        <button type="submit" class="submit-btn">Get Prediction</button>
    </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:

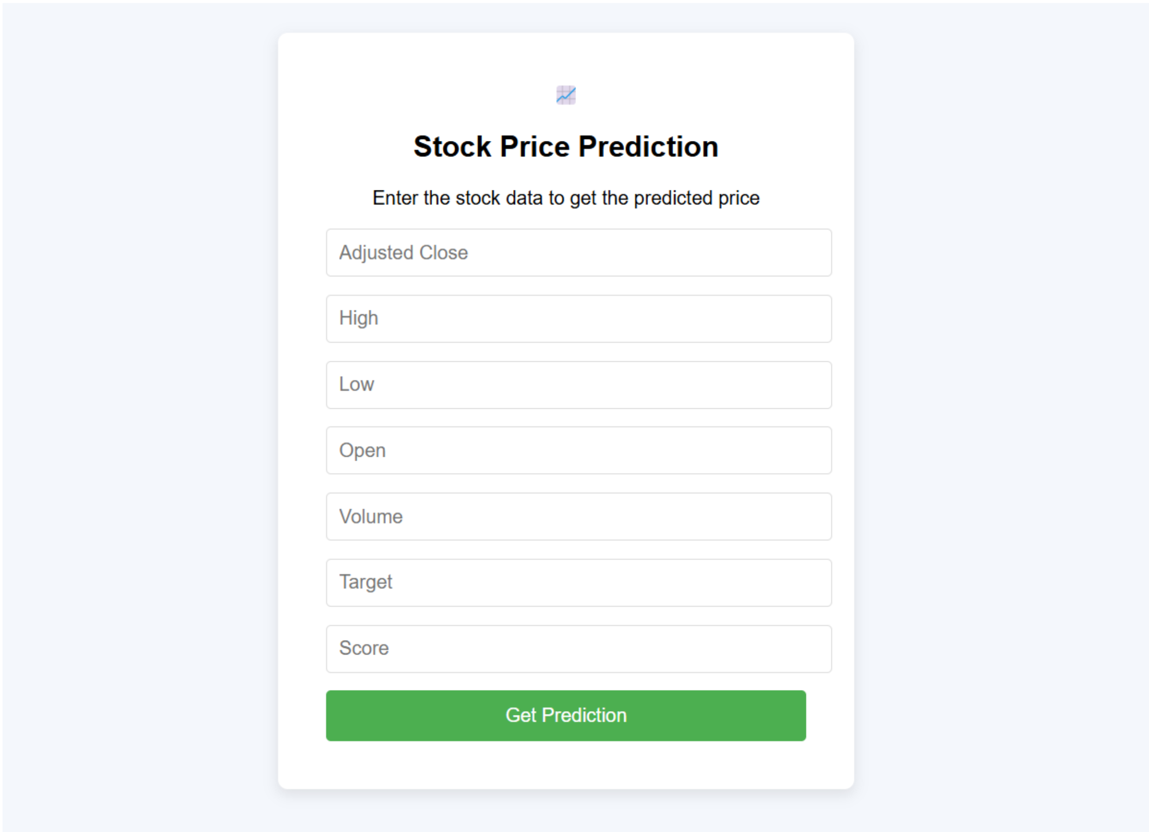
```javascript
document.getElementById('predictionForm').addEventListener('submit', function(e) {
    e.preventDefault();

    const formData = new FormData(this);
    fetch('/predict', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        if (data.prediction) {
            document.getElementById('predictionResult').innerText = 'Prediction: ' + data.prediction;
        } else {
            document.getElementById('predictionResult').innerText = 'Error: ' + data.error;
        }
    });
});
```

---

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