Topic:-Stock Market Prediction using Regression in Machine Learning

Introduction

- Stock Market Prediction: This involves forecasting the future value of company stock or other financial instruments traded on an exchange.
- Goal: To predict the closing price of a stock (using Apple - AAPL - as an example) based on historical data.
- Method: Utilizes Machine Learning, specifically Linear Regression, to model the relationship between historical stock data features and the closing price.
- Accurate stock predictions are crucial for investors making buy/sell decisions, traders developing strategies, and financial institutions managing risk.

Dataset Overview

Dataset: Apple Stock Dataset (AAPL.csv)

Features:

1. Open – Opening stock price on a given day
2. High-Highest price of the stock for the day
3. Low-Lowest price of the stock for the day
4. Volume -Total number of shares traded
5. MA5 – 5-day Moving Average of Closing Price
6. MA20 – 20-day Moving Average of Closing Price
7. MA50 – 50-day Moving Average of Closing Price
8. Daily_Return – Daily percentage return of the stock
9. RSI – Relative Strength Index (14-day)

Target: Close(t) – Closing stock price of the day (used for prediction)

Algorithm Used

1. Linear Regression
2. Supervised learning algorithm
3. Estimates relationship between dependent and independent variables
4. Predicts continuous values like stock prices
5. Simple, fast, and interpretable

Model Implementation

•**Load & Prepare Data:** Loads data, converts 'Date', selects relevant columns, and handles outliers by replacing them with the median.

•**Feature Engineering:** Calculates moving averages (MA5, MA20, MA50), daily return, and Relative Strength Index (RSI). Fills NaNs and drops the 'Date' column.

•**Scale & Split:** Separates features (X) and target (y), scales features using StandardScaler, and splits data into training, validation, and testing sets (70/15/15) without shuffling.

•**Train & Evaluate Regression:** Trains a Linear Regression model and evaluates its performance on the test set using MAE, RMSE, and R^2.

•**Visualize Regression:** Plots the actual vs. predicted closing prices on the test set.

•**Confusion Matrix (Threshold-based Classification):** Classifies actual and predicted values based on a threshold (mean of actual test values), calculates and visualizes the confusion matrix, and prints accuracy and classification report.

Results

Regression Metrics:
- **MAE** = 2.79
- **RMSE** = 15.99
- **R^2 Score** = 0.94 — Excellent model fit

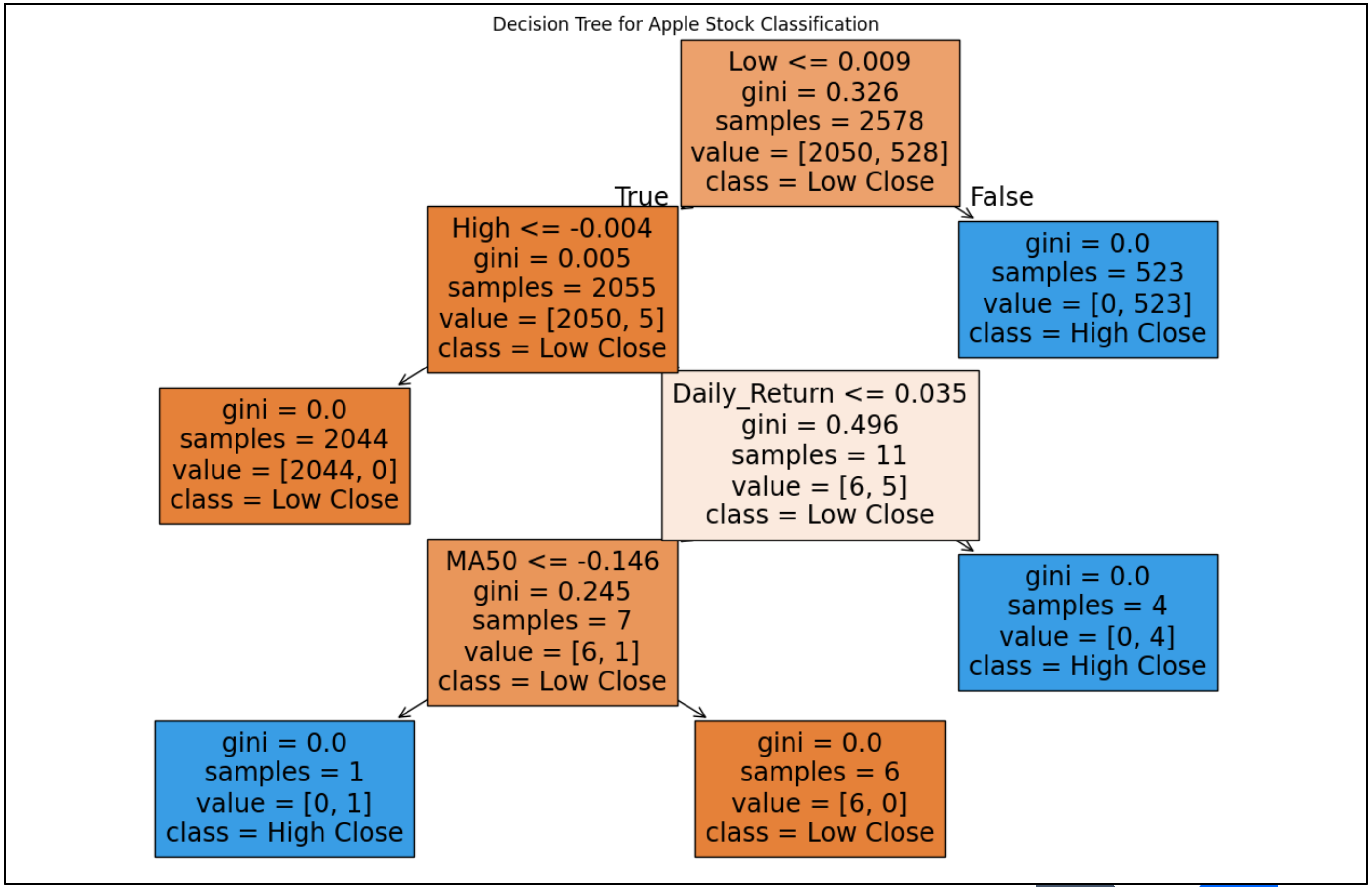
Trend Direction Classification:
- **Accuracy** = **98.73%**
- **Precision**:
  - – Class 0 (Price ↓): 98%
  - – Class 1 (Price ↑): 99%
- **Recall**:
  - – Class 0 (Price ↓): 98%
  - – Class 1 (Price ↑): 99%
- **F1-Score**: Balanced at 0.98–0.99 across both classes
- **Confusion Matrix:**

Model accurately predicts **up/down trends** in stock prices with very high precision

Regression Model Visualization

- Linear Regression achieved high stock prediction accuracy ($R^2 = 0.94$).
- Feature engineering (Moving Averages, RSI) improved model performance.
- Outlier treatment and feature scaling enhanced stability.
- Achieved low MAE (2.79), RMSE (15.99), and 98.7% trend classification accuracy.
- Visualizations validated predictions and trend behavior.
- Some limitations exist due to market volatility and unseen factors.

Decision Tree Visualization



Output

```
✅ Step 1: Data loaded
        Date  Open  High   Low  Close(t)     Volume      SD20  Upper_Band  \
0 2005-10-17  6.66  6.69  6.50      6.60  154208600  0.169237    6.827473
1 2005-10-18  6.57  6.66  6.44      6.45  152397000  0.168339    6.819677
2 2005-10-19  6.43  6.78  6.32      6.78  252170800  0.180306    6.861112
3 2005-10-20  6.72  6.97  6.71      6.93  339440500  0.202674    6.931847
4 2005-10-21  7.02  7.03  6.83      6.87  199181500  0.216680    6.974860

   Lower_Band  S_Close(t-1)  ...  QQQ_MA10  QQQ_MA20  QQQ_MA50  SnP_Close  \
0    6.150527          6.67  ...    33.692   33.9970   34.2690    1190.10
1    6.146323          6.60  ...    33.570   33.9525   34.2466    1178.14
2    6.139888          6.45  ...    33.562   33.9600   34.2330    1195.76
3    6.121153          6.78  ...    33.567   33.9455   34.2190    1177.80
4    6.108140          6.93  ...    33.586   33.9365   34.2034    1179.59

   SnP(t-1))  SnP(t-5)  DJIA_Close  DJIA(t-1))  DJIA(t-5)  Close_forcast
0    1186.57   1187.33    10348.10    10287.34   10238.76           6.45
1    1190.10   1184.87    10285.26    10348.10   10253.17           6.78
2    1178.14   1177.68    10414.13    10285.26   10216.91           6.93
3    1195.76   1176.84    10281.10    10414.13   10216.59           6.87
4    1177.80   1186.57    10215.22    10281.10   10287.34           7.01

[5 rows x 64 columns]
```

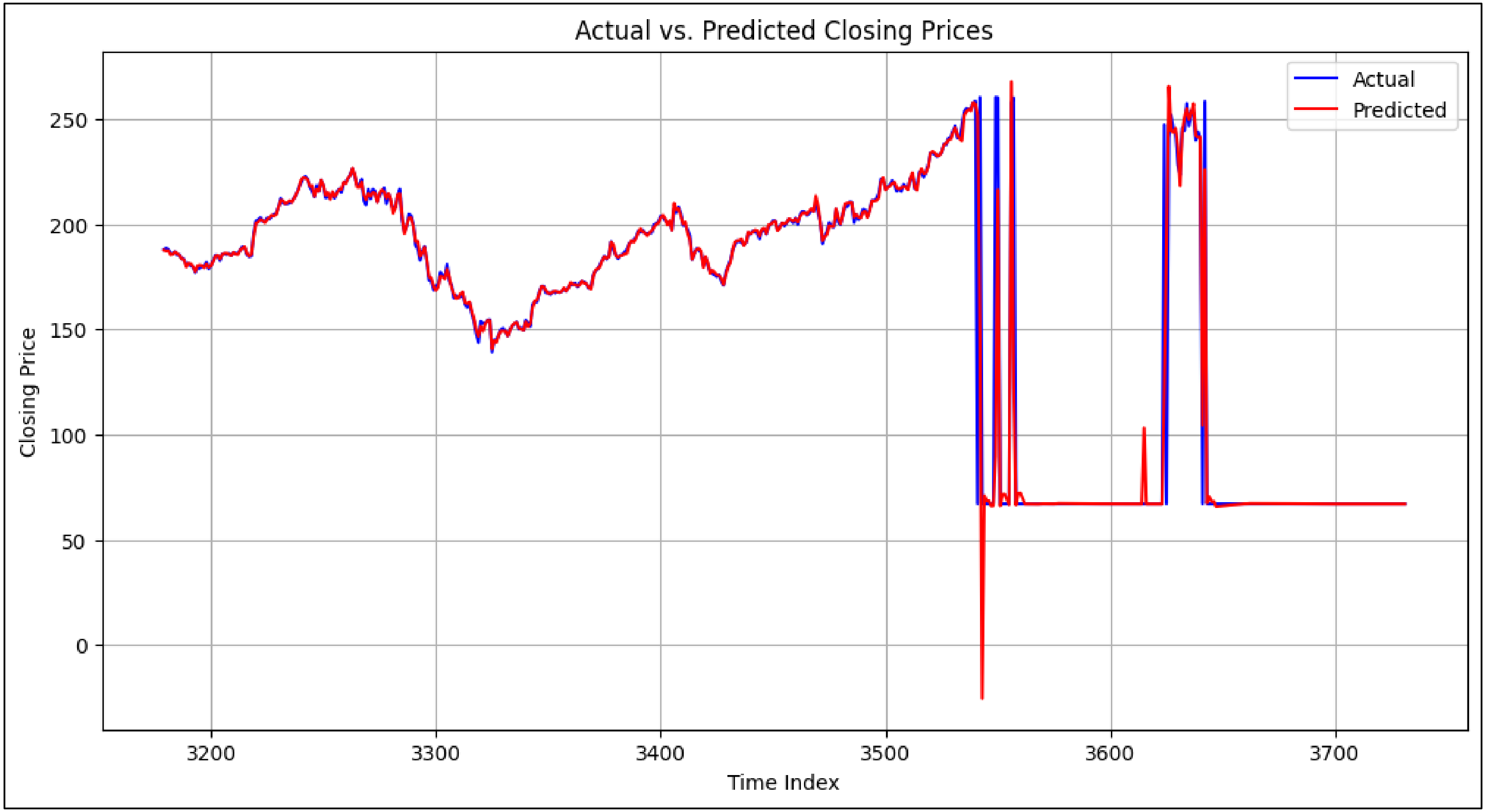
```
✅ Step 2: Kept necessary columns
        Date  Open  High   Low  Close(t)     Volume
0 2005-10-17  6.66  6.69  6.50      6.60  154208600
1 2005-10-18  6.57  6.66  6.44      6.45  152397000
2 2005-10-19  6.43  6.78  6.32      6.78  252170800
3 2005-10-20  6.72  6.97  6.71      6.93  339440500
4 2005-10-21  7.02  7.03  6.83      6.87  199181500
✅ Step 3: Outliers replaced with median
                                Date         Open         High          Low  \
count                           3732  3732.000000  3732.000000  3732.000000
mean   2013-03-16 07:53:49.581993472    75.868867    76.433650    75.073207
min              2005-10-17 00:00:00     6.390000     6.530000     6.190000
25%              2009-07-01 18:00:00    22.575000    22.895000    22.160000
50%              2013-03-18 12:00:00    67.135000    67.635000    66.480000
75%              2016-11-28 06:00:00   107.757500   108.690000   106.835000
max              2020-08-13 00:00:00   259.760000   261.510000   257.840000
std                              NaN    60.133889    60.487775    59.542028

          Close(t)        Volume
count  3732.000000  3.732000e+03
mean     75.824412  1.022502e+08
min       6.260000  1.136200e+07
25%      22.532500  3.714862e+07
50%      67.062500  8.543010e+07
75%     107.600000  1.487201e+08
max     260.210000  3.425807e+08
std      60.107917  7.636387e+07
aNs
```

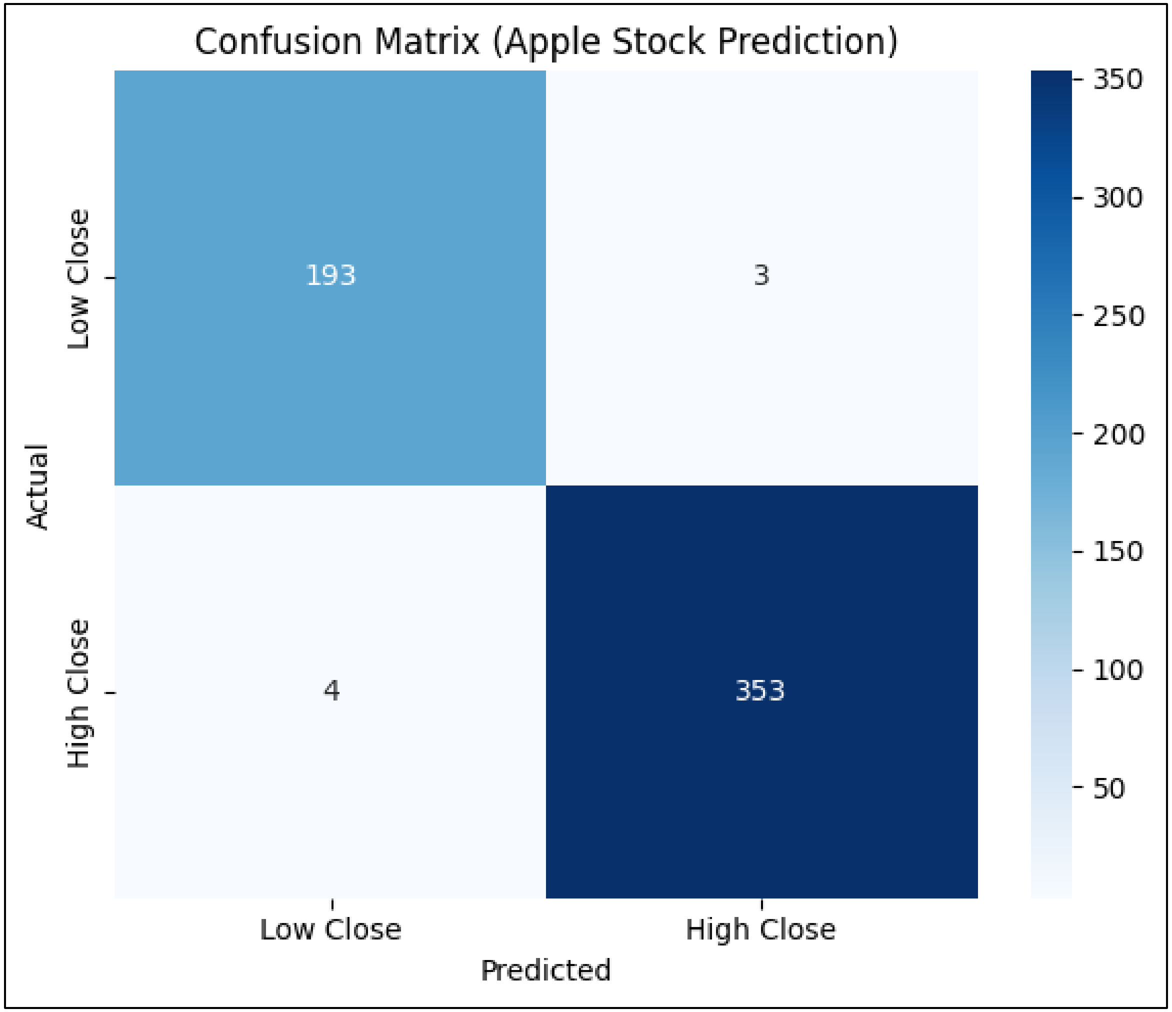
```
✅ Step 4: Feature engineering done
         Date  Open  High   Low  Close(t)       Volume    MA5    MA20
MA50  \
49 2005-12-27  9.14  9.28  9.13      9.16  147647500.0  9.064  8.9640
8.0510
50 2005-12-28  9.19  9.23  9.05      9.08   99528800.0  9.100  8.9975
8.1006
51 2005-12-29  9.11  9.11  8.82      8.82  122506300.0  9.050  9.0200
8.1480
52 2005-12-30  8.75  8.94  8.68      8.87  156065700.0  8.996  9.0215
8.1898
53 2006-01-03  8.94  9.23  8.92      9.23  201808600.0  9.032  9.0345
8.2358
    Daily_Return        RSI
49      1.215470  50.819672
50     -0.873362  48.062016
51     -2.863436  39.215686
52      0.566893  40.000000
53      4.058625  49.456522
✅ Step 5: Date column dropped
    Open  High   Low  Close(t)       Volume    MA5    MA20    MA50  \
49  9.14  9.28  9.13      9.16  147647500.0  9.064  8.9640  8.0510
50  9.19  9.23  9.05      9.08   99528800.0  9.100  8.9975  8.1006
51  9.11  9.11  8.82      8.82  122506300.0  9.050  9.0200  8.1480
52  8.75  8.94  8.68      8.87  156065700.0  8.996  9.0215  8.1898
53  8.94  9.23  8.92      9.23  201808600.0  9.032  9.0345  8.2358
    Daily_Return        RSI
49      1.215470  50.819672
50     -0.873362  48.062016
51     -2.863436  39.215686
52      0.566893  40.000000
53      4.058625  49.456522
```

```
✅ Step 6: Feature scaling done
       Open      High       Low    Volume       MA5      MA20      MA50  \
0 -1.127025 -1.127582 -1.124839  0.605071 -1.136439 -1.145197 -1.167990
1 -1.126191 -1.128410 -1.126185 -0.025443 -1.135834 -1.144630 -1.167142
2 -1.127524 -1.130398 -1.130056  0.275638 -1.136674 -1.144248 -1.166331
3 -1.133524 -1.133215 -1.132412  0.715378 -1.137582 -1.144223 -1.165616
4 -1.130357 -1.128410 -1.128373  1.314762 -1.136977 -1.144003 -1.164829
   Daily_Return       RSI
0      0.066610 -0.146156
1     -0.109380 -0.280652
2     -0.277048 -0.712101
3      0.011966 -0.673849
4      0.306153 -0.212639
✅ Step 7: Data split into train, validation, test
X_train shape: (2578, 9)
X_val shape: (552, 9)
X_test shape: (553, 9)
✅ Step 8: Linear Regression model trained
✅ Step 9: Evaluation
MAE = 2.79, RMSE = 15.99, R² = 0.94
```

Actual vs. Predicted Closing Prices



Confusion Matrix



Challenges

1)Handling Outliers
•Stock data is highly volatile. Sudden spikes or crashes in prices/volume can distort model learning.
•Replacing outliers using the **Interquartile Range (IQR) method** and substituting with the median helped stabilize training but might also hide important rare events.

2)Feature Engineering Complexity

•Derived indicators like **moving averages (MA5, MA20, MA50)** and **RSI (Relative Strength Index)**

introduced NaN values at the start due to rolling calculations.

•Managing missing values with forward filling was necessary but imperfect.

3)Time-Series Nature
•Stock prices are sequential. We couldn't shuffle data during splitting, which made **chronological train-test-validation splits** crucial to prevent future data leaking into training.

4)Feature-Target Correlation
•Not all engineered features had strong linear correlation with the target (Close price). This challenged the performance of a **simple linear regression** model.

Model Evaluation

Accuracy: **98.73%**

Class 0 (Price Down): Precision = 0.98, Recall = 0.98, F1-score = 0.98, Support = 196

Class 1 (Price Up): Precision = 0.99, Recall = 0.99, F1-score = 0.99, Support = 357

Total Samples: **553**

Threshold Sensitivity

1. To convert predicted prices into "up/down" labels, a threshold (mean close price) was used.
2. This threshold directly affects accuracy (here, 98.73%) — a different cutoff might lower it.

No Info About How Far Predictions Are Off

1. While precision and recall are high (>0.98 for both classes), they don’t show how close predicted prices are to actual values.
2. We still need regression metrics like MAE or RMSE to measure exact price accuracy.

More 'Up' Days in Data

1. Out of 553 points, 357 were labeled as price up (class 1) and 196 as price down (class 0).
2. The model may lean towards predicting “up” more often due to this imbalance.

Linear Regression is Not Time-Aware

1. It doesn't handle sequential patterns like trends or volatility well, which limits performance during sudden market shifts.

High Accuracy ≠ Good TradingEven

1. with 98.73% accuracy, if the model makes wrong predictions on key days, it can lead to losses in actual trading.

Conclusion

- Linear Regression achieved strong stock price prediction with an R^2 score of 0.94.
- Feature engineering (Moving Averages, RSI) significantly improved model accuracy.
- Outlier handling and feature scaling enhanced model stability and performance.
- The model achieved low MAE (2.79), low RMSE (15.99), and 98.7% trend classification accuracy.
- Visualization helped in validating model predictions and understanding trend-following behavior.
- Some limitations remain due to market volatility and external factors not included in the dataset.

future scope

- Use **time-series models** like LSTM or ARIMA for better trend learning.
- Combine **regression for price** and **classification for direction**.
- Perform **backtesting** to test strategy performance on past data.
- Experiment with **dynamic thresholds** instead of fixed averages.

References

- **Pandas:** Data manipulation and analysis library. https://pandas.pydata.org/
- **NumPy:** Fundamental package for numerical computation. https://numpy.org/
- **Scikit-learn:** Machine learning library for regression, scaling, metrics, etc. https://scikit-learn.org/
- **Matplotlib:** Plotting library for visualizations. https://matplotlib.org/
- **Seaborn:** Statistical data visualization library based on Matplotlib. https://seaborn.pydata.org/
- **Data Source:** AAPL.csv (https://www.kaggle.com/code/pandeyharsh407/stock-prediction-using-linear-regression/input?select=AAPL.csv)

Thank you!!!