Name - Aarya Thorat

D15A/63

EXPERIMENT 1

AIM: Implementation of Linear and Logistic Regression on Stock Market Dataset

---

THEORY:

1. Dataset Source

Dataset Name: **ADANIPORTS.csv**

Source: National Stock Exchange (NSE) Historical Dataset

Link:
https://www.kaggle.com/datasets/rohanrao/nifty50-stock-market-data/adaniports.csv

This dataset contains historical daily stock trading data of Adani Ports, which is part of the NIFTY 50 index. The data reflects real-world market behavior and includes price fluctuations and trading activity.

---

2. Dataset Description

The dataset represents financial time-series data collected daily from the stock exchange.

Dataset Characteristics

- Type: Structured tabular dataset
- Nature: Time-series (chronological order important)
- Domain: Financial analytics
- Records: Daily stock data across multiple years
- Data Type: Numerical

Financial datasets are highly volatile and influenced by market demand, supply, economic conditions, and investor sentiment.

Features (Independent Variables)

Feature	Description
Open	Price at which stock opens for trading
High	Maximum price during trading hours
Low	Minimum price during trading hours
Volume	Number of shares traded in that day

These features are strongly related to price movement and reflect daily trading dynamics.

Target Variables

For Linear Regression:

- Close (continuous variable representing final trading price)

For Logistic Regression:

- Movement (binary classification variable)

Feature Engineering:

Movement =

- 1 if Close > Open (Bullish Day)
- 0 if Close ≤ Open (Bearish Day)

This transformation enables us to convert a regression problem into a classification problem.

---

3. Mathematical Formulation of the Algorithm

3.1 Linear Regression

Linear regression models the dependent variable as a weighted sum of independent variables.

$Y=\beta 0+\beta 1X1+\beta 2X2+\beta 3X3+\beta 4X4$Y=β0+β1X1+β2X2+β3X3+β4X4

Objective:
Minimize squared prediction error.

$MSE=1n\sum(yi-yi$^)2MSE=n1∑(yi−yi^)2

Assumptions:

- Linearity
- Independence
- Homoscedasticity
- Normal distribution of errors

R² Score:

$R2=1-SSresSStot$R2=1−SStotSSres

R² measures proportion of variance explained by the model.

3.2 Logistic Regression

Logistic regression predicts probability using sigmoid function.

$P(Y=1)=11+e-z$P(Y=1)=1+e−z1

Where:

$z=\beta 0+\sum\beta iXi$z=β0+∑βiXi

The output ranges between 0 and 1.

The model minimizes Log Loss:

$Loss=-1n\sum[y\log(p)+(1-y)\log(1-p)]$Loss=−n1∑[ylog(p)+(1−y)log(1−p)]

Logistic regression provides probabilistic interpretation which is useful in financial decision-making.

---

4. Algorithm Limitations

Linear Regression

- Assumes linear relationship between variables
- Cannot capture nonlinear stock market behavior
- Sensitive to extreme outliers
- Performance degrades if multicollinearity exists

Stock markets often exhibit nonlinear and chaotic patterns, limiting predictive accuracy.

Logistic Regression

- Assumes linear decision boundary
- Poor performance when data is not linearly separable
- Cannot capture complex market patterns
- Performance sensitive to feature scaling

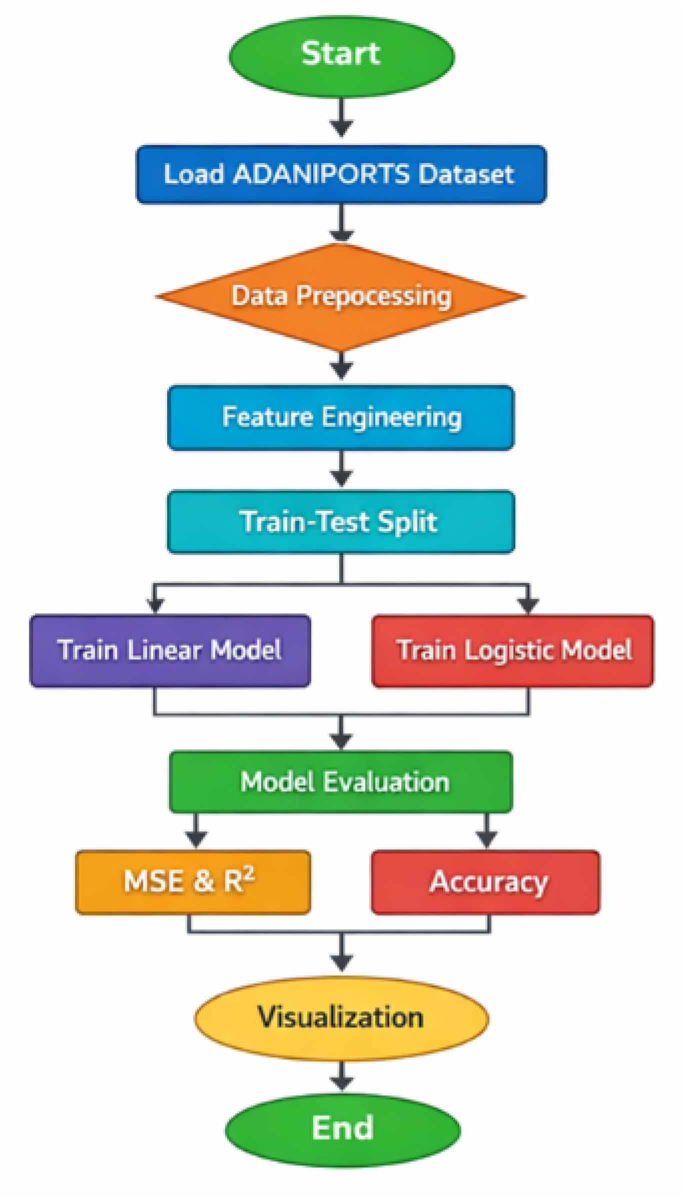
Thus, these models serve as baseline models rather than advanced financial forecasting systems.

---

5. Methodology / Workflow

Step 1: Load dataset using Pandas
Step 2: Inspect data using head(), info(), describe()
Step 3: Check for null values
Step 4: Select relevant financial features
Step 5: Create Movement column
Step 6: Split dataset (80% train, 20% test)
Step 7: Train Linear Regression
Step 8: Train Logistic Regression
Step 9: Evaluate using performance metrics
Step 10: Visualize results

Workflow:



---

6. Performance Analysis

Linear Regression

Observed Results:

- R^2 value close to 1 indicates strong dependency between price indicators.
- Low MSE indicates minimal prediction error.

- High and Low prices show strongest influence on Close price.

Interpretation:

Stock closing price is highly dependent on daily trading range. Since Close lies between High and Low, strong correlation exists.

However, sudden market events (news, policy changes) can cause prediction deviation.

Logistic Regression

Observed Results:

- Accuracy shows moderate performance.
- Confusion matrix reveals classification distribution.
- Model performs better when daily price movement is large.

Interpretation:

Logistic regression captures general upward or downward trends but cannot account for market shocks or sudden reversals.

---

7. Hyperparameter Tuning

For Logistic Regression:

Parameters tuned:

- C (regularization strength)
- max_iter

GridSearchCV used with cross-validation.

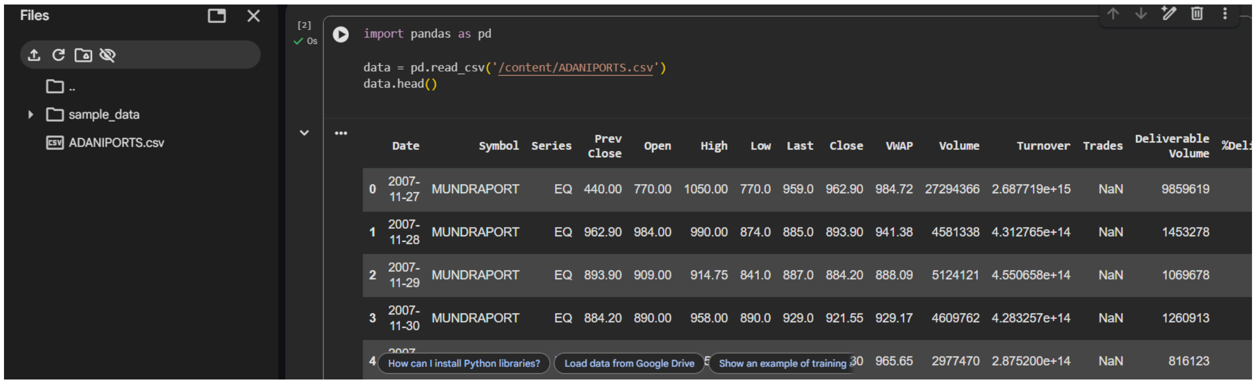
Impact of C:

- Small C → Strong regularization → Underfitting risk
- Large C → Weak regularization → Overfitting risk

Optimal C improved test accuracy and reduced variance.

Linear regression does not require heavy tuning.

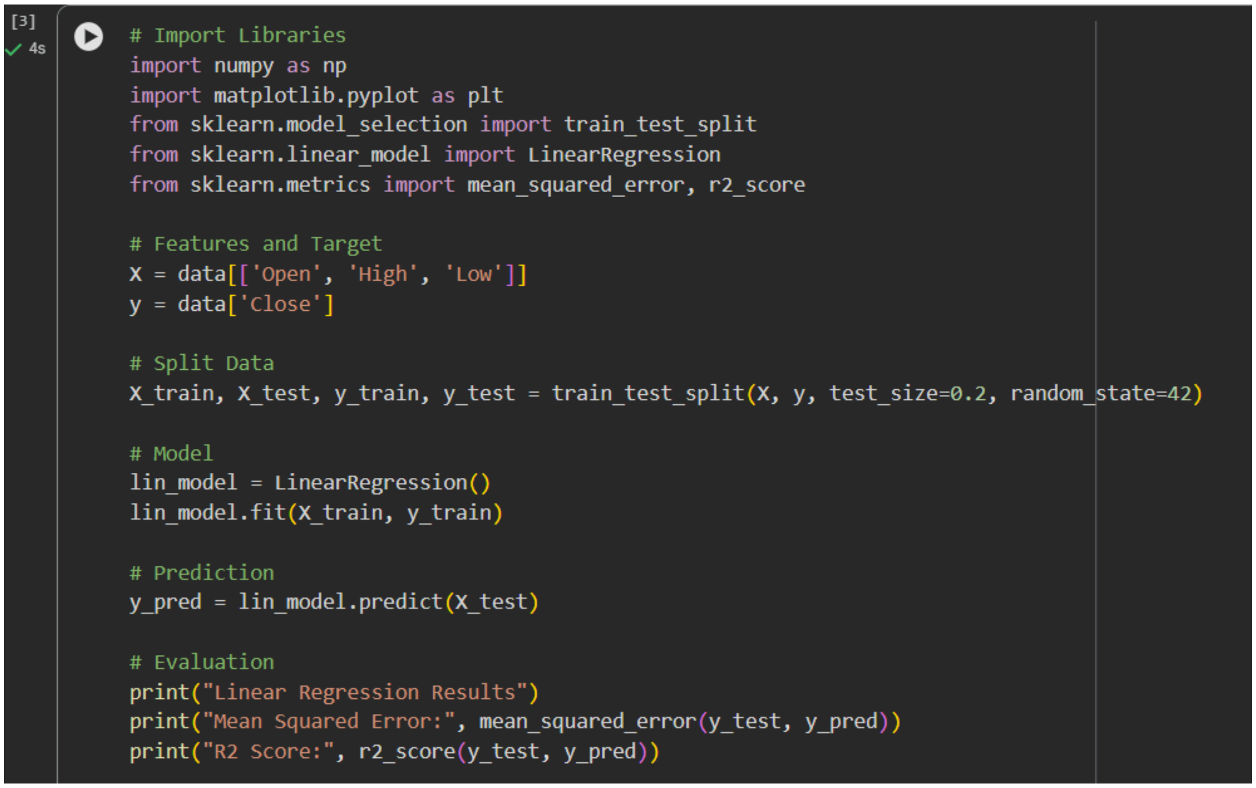
OUTPUT:


Files
..
sample_data
ADANIPORTS.csv
How can I install Python libraries?
Load data from Google Drive
Show an example of training

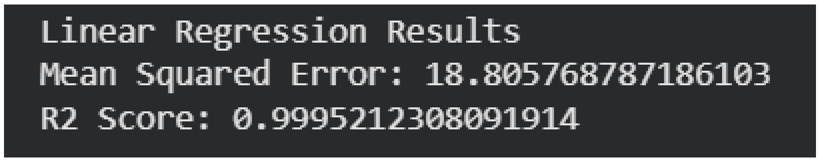
```
[2]
import pandas as pd

data = pd.read_csv('/content/ADANIPORTS.csv')
data.head()
```

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deli
0	2007-11-27	MUNDRAPORT	EQ	440.00	770.00	1050.00	770.0	959.0	962.90	984.72	27294366	2.687719e+15	NaN	9859619	
1	2007-11-28	MUNDRAPORT	EQ	962.90	984.00	990.00	874.0	885.0	893.90	941.38	4581338	4.312765e+14	NaN	1453278	
2	2007-11-29	MUNDRAPORT	EQ	893.90	909.00	914.75	841.0	887.0	884.20	888.09	5124121	4.550658e+14	NaN	1069678	
3	2007-11-30	MUNDRAPORT	EQ	884.20	890.00	958.00	890.0	929.0	921.55	929.17	4609762	4.283257e+14	NaN	1260913	
4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	965.65	2977470	2.875200e+14	NaN	816123	

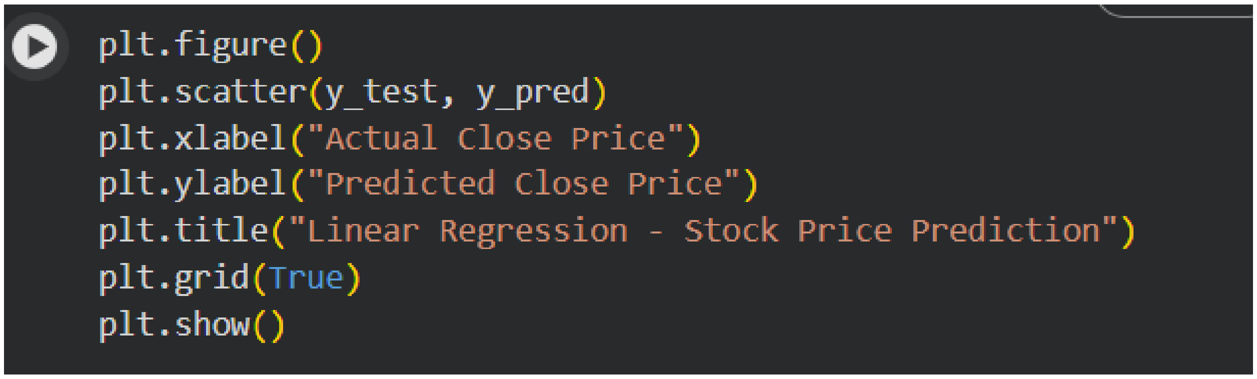
Linear regression



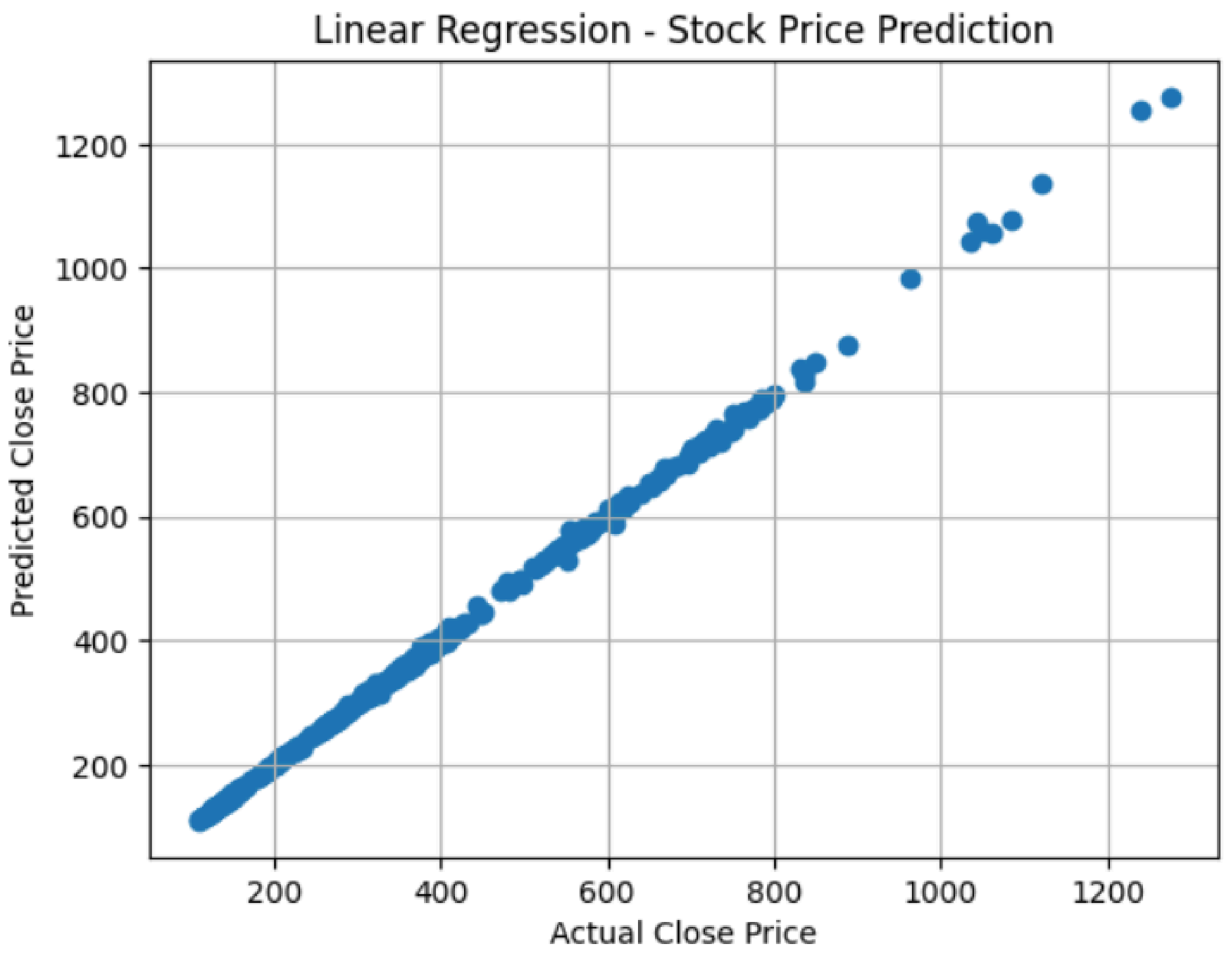
```
[3]
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Features and Target
X = data[['Open', 'High', 'Low']]
y = data['Close']

# Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)

# Prediction
y_pred = lin_model.predict(X_test)

# Evaluation
print("Linear Regression Results")
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
```



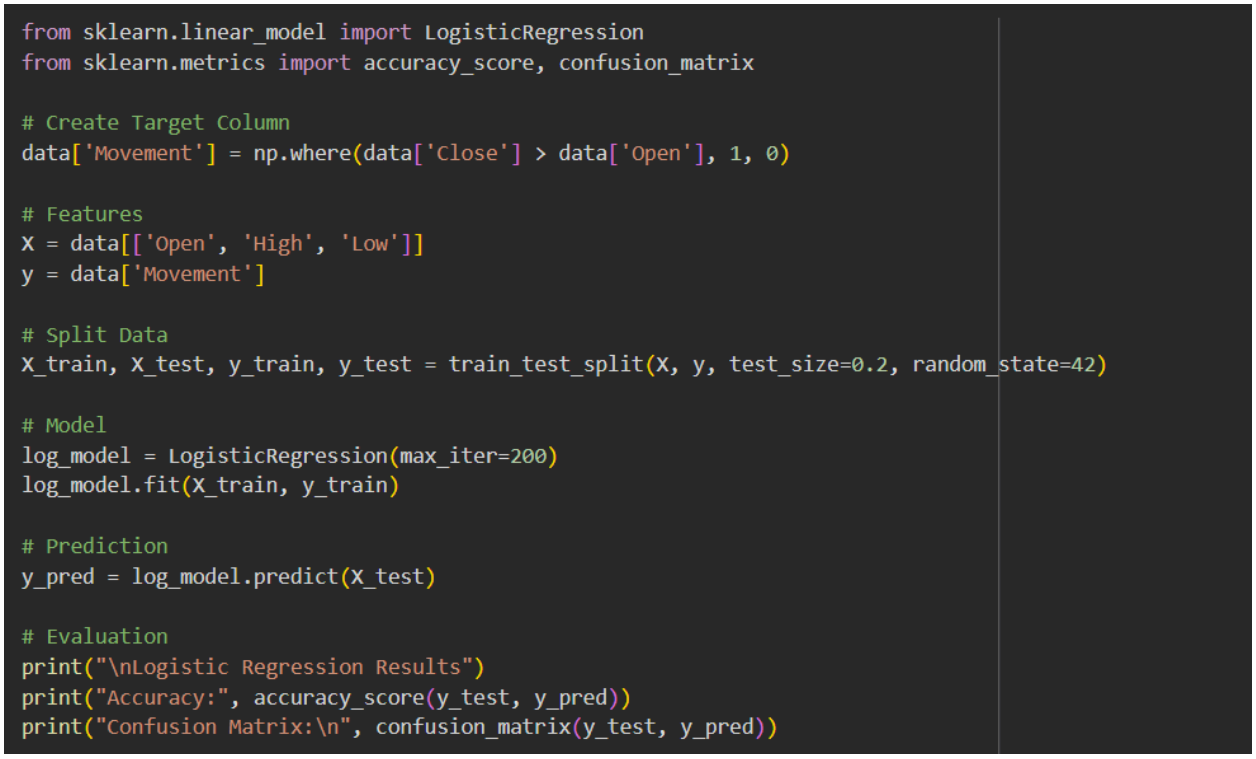
```
Linear Regression Results
Mean Squared Error: 18.805768787186103
R2 Score: 0.9995212308091914
```



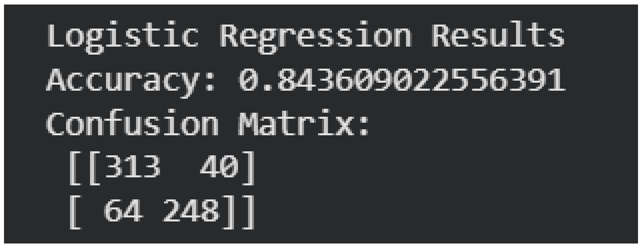
```
plt.figure()
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Close Price")
plt.ylabel("Predicted Close Price")
plt.title("Linear Regression - Stock Price Prediction")
plt.grid(True)
plt.show()
```



Logistic Regression

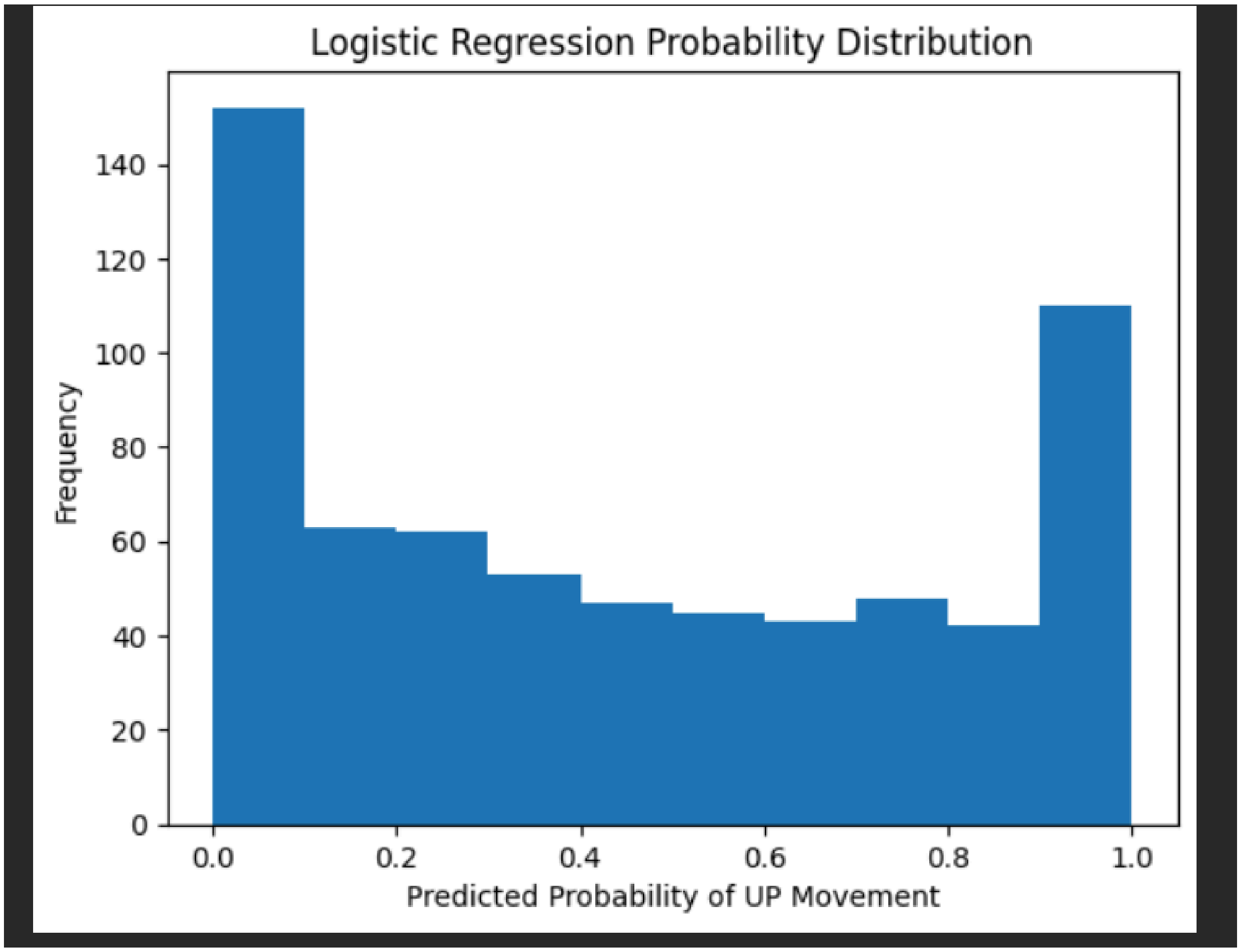


```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# Create Target Column
data['Movement'] = np.where(data['Close'] > data['Open'], 1, 0)

# Features
X = data[['Open', 'High', 'Low']]
y = data['Movement']

# Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model
log_model = LogisticRegression(max_iter=200)
log_model.fit(X_train, y_train)

# Prediction
y_pred = log_model.predict(X_test)

# Evaluation
print("\nLogistic Regression Results")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```



```
Logistic Regression Results
Accuracy: 0.843609022556391
Confusion Matrix:
 [[313  40]
 [ 64 248]]
```

```
# Get probabilities
probs = log_model.predict_proba(X_test)[:, 1]

plt.figure()
plt.hist(probs)
plt.xlabel("Predicted Probability of UP Movement")
plt.ylabel("Frequency")
plt.title("Logistic Regression Probability Distribution")
plt.show()
```



CONCLUSION:

This experiment implemented Linear Regression to predict stock closing prices and Logistic Regression to classify stock movement using real-world financial data. Linear Regression showed strong predictive capability with high R^2 and low MSE, indicating a strong relationship between daily price indicators. Logistic Regression achieved moderate accuracy in classifying upward and downward trends. The experiment demonstrates the practical application of regression techniques in financial analytics while highlighting their limitations in handling market volatility.