School of Computer Science and Engineering
Fall Semester-2024-25

Course Code : CBS3007

Course: Data Mining and Analytics

Alan Thomas

21BBS0115

Github link for the datasets and code-

https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA5

Aim

Consider the TATA MOTORS shares data from National stock exchange for the past 7 years. Implement the AutoRegressive Integrated Moving Average (ARIMA) model on the data and identify the 50 days moving average(MA), 200 days MA, 365 days MA and 500 days MA. Summarize the autocorrelations detected from the model.

LIBRARIES USED: Pandas, Numpy, Scikit Learn
Dataset : https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA5/Q1

SECTION 1

Sample Input

	A	B	C	D
1	Date	Price		
2	14-11-2024	774.3		
3	13-11-2024	786.25		
4	12-11-2024	784.85		
5	11-11-2024	804.7		
6	08-11-2024	805.45		
7	07-11-2024	819.75		
8	06-11-2024	839.7		
9	05-11-2024	835.65		
10	04-11-2024	824.1		
11	01-11-2024	843.45		
12	31-10-2024	834.05		
13	30-10-2024	840.2		
14	29-10-2024	842.75		
15	28-10-2024	878.45		
16	25-10-2024	864.3		
17	24-10-2024	880		
18	23-10-2024	877.65		
19	22-10-2024	879.5		
20	21-10-2024	903.3		
21	18-10-2024	910.15		
22	17-10-2024	891.6		
23	16-10-2024	907.45		

Code

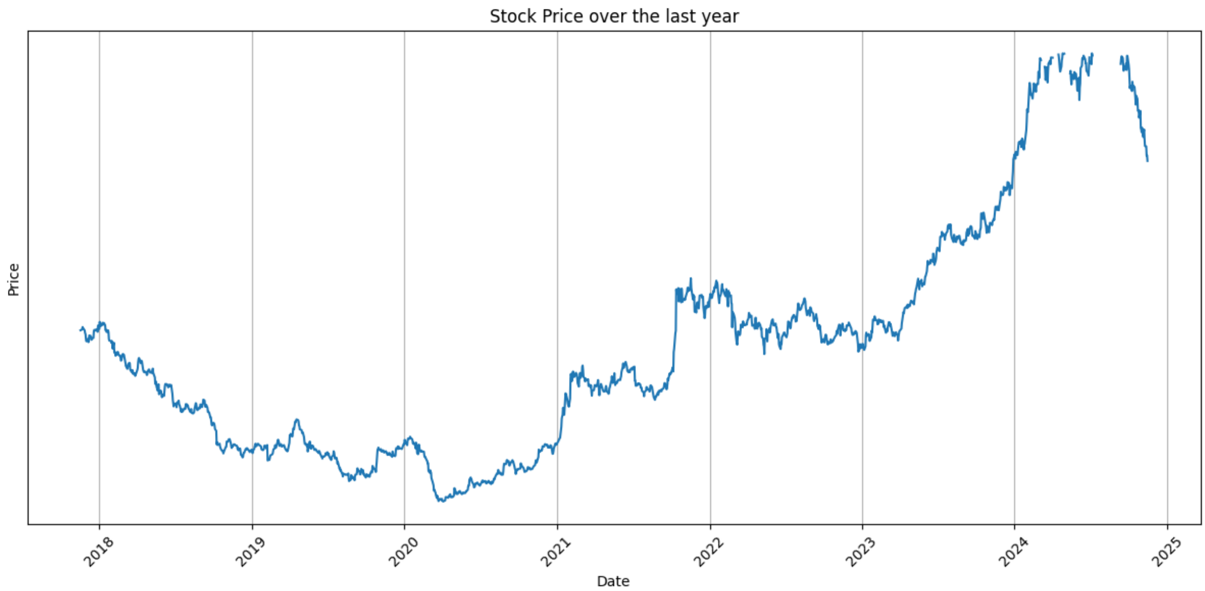
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

df = pd.read_csv('TAMO_stock.csv')
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y').dt.date
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')

# taking 100 past entries
df_last_year = df.head(50)
df_last_year = df_last_year.iloc[::-1]

# PRITING THE FIRST 50 ENTRIES
plt.figure(figsize=(12, 6))
plt.plot(df_last_year["Date"], df_last_year['Price'], marker='o', linestyle='-',
color='b')
plt.title('Stock Price over the last month')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid(True)
plt.xticks(rotation=45)
# plt.yticks([])
plt.tight_layout()
plt.show()

# PRITING THE FIRST 50 ENTRIES
plt.figure(figsize=(12, 6))
plt.plot(df["Date"], df['Price'])
plt.title('Stock Price over the past 7 years')
plt.xlabel('Date')
plt.ylabel('Price')
```

```python
plt.grid(True)
plt.xticks(rotation=45)
plt.yticks([])
plt.tight_layout()
plt.show()


# Calculate the Moving average
df['MA_50'] = df['Price'].rolling(window=50).mean()
df['MA_200'] = df['Price'].rolling(window=200).mean()
df['MA_365'] = df['Price'].rolling(window=365).mean()
df['MA_500'] = df['Price'].rolling(window=500).mean()

# Plotting moving averages
plt.figure(figsize=(12, 8))
plt.plot(df['Price'], label='Original Data')
plt.plot(df['MA_50'], label='50-Day MA', linestyle='dashed')
plt.plot(df['MA_200'], label='200-Day MA', linestyle='dashed')
plt.plot(df['MA_365'], label='365-Day MA', linestyle='dashed')
plt.plot(df['MA_500'], label='500-Day MA', linestyle='dashed')
plt.legend()
plt.title('Moving Averages')
plt.show()

result = adfuller(df['Price'].dropna())
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")
if result[1] > 0.05:
   print("Data is non-stationary. Differencing may be needed.")
else:
   print("Data is stationary.")

# FITTING THE ARIMA MODEL
model = ARIMA(df['Price'], order=(2, 1, 2))
arima_result = model.fit()

print(arima_result.summary())
```

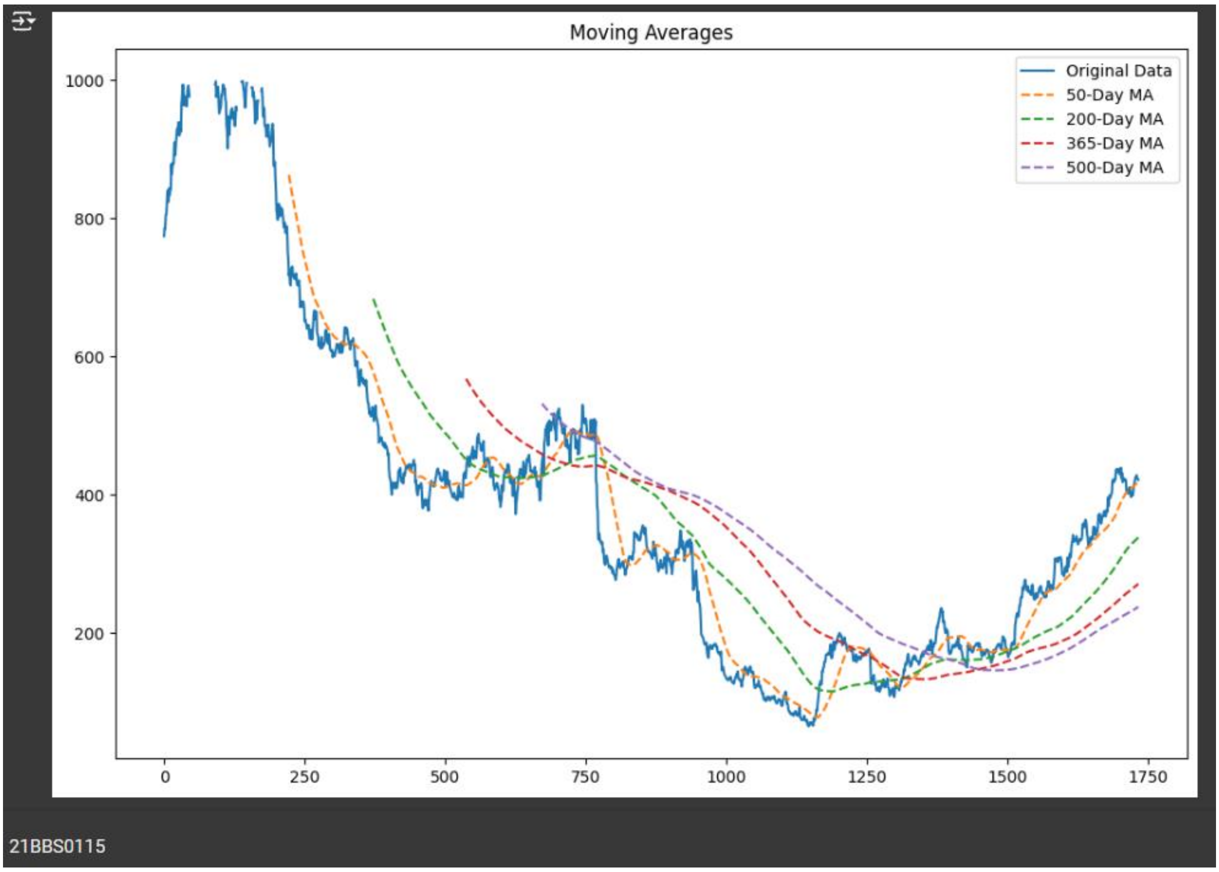
```
df_test = df.iloc[1:]

df['Fitted'] = arima_result.fittedvalues
plt.figure(figsize=(12, 6))
plt.plot(df_test["Date"] , df_test['Price'], label='Original Data')
plt.plot(df_test["Date"] , df_test['Fitted'], label='ARIMA Fitted Values',
linestyle='dashed')
plt.legend()
plt.title('ARIMA Fitted Values')
plt.show()

df['Fitted'] = arima_result.fittedvalues
plt.figure(figsize=(12, 6))
plt.plot(df_test["Date"].head(30) , df_test['Price'].head(30), label='Original
Data')
plt.plot(df_test["Date"].head(30) , df_test['Fitted'].head(30), label='ARIMA
Fitted Values', linestyle='dashed')
plt.legend()
plt.title('ARIMA Fitted Values over 30 entries (A closer look at the graph)')
plt.show()

fig, ax = plt.subplots(2, 1, figsize=(12, 8))
plot_acf(df['Price'].dropna(), lags=40, ax=ax[0])
plot_pacf(df['Price'].dropna(), lags=40, ax=ax[1])
plt.show()
```

Output

Data for the first 50 entries



Data for all 7 years



Moving Averages

```
ADF Statistic: -1.7055612961522493
p-value: 0.4282494317288183
Data is non-stationary. Differencing may be needed.
```

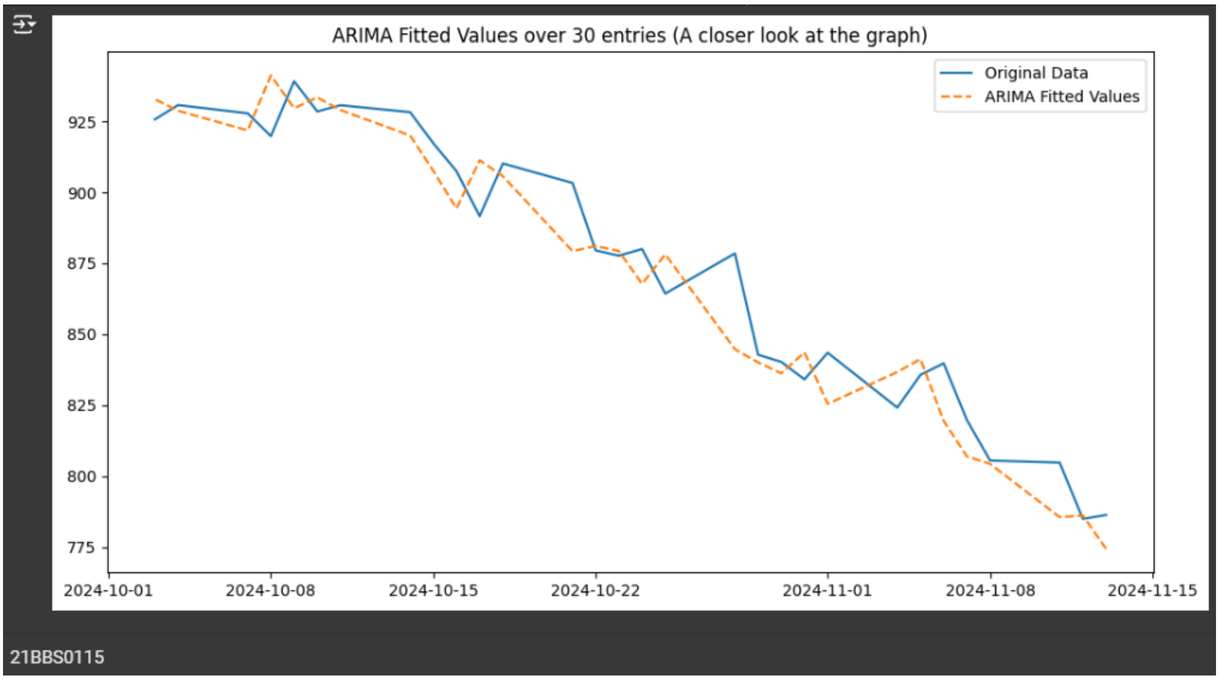
21BBS0115

Stability

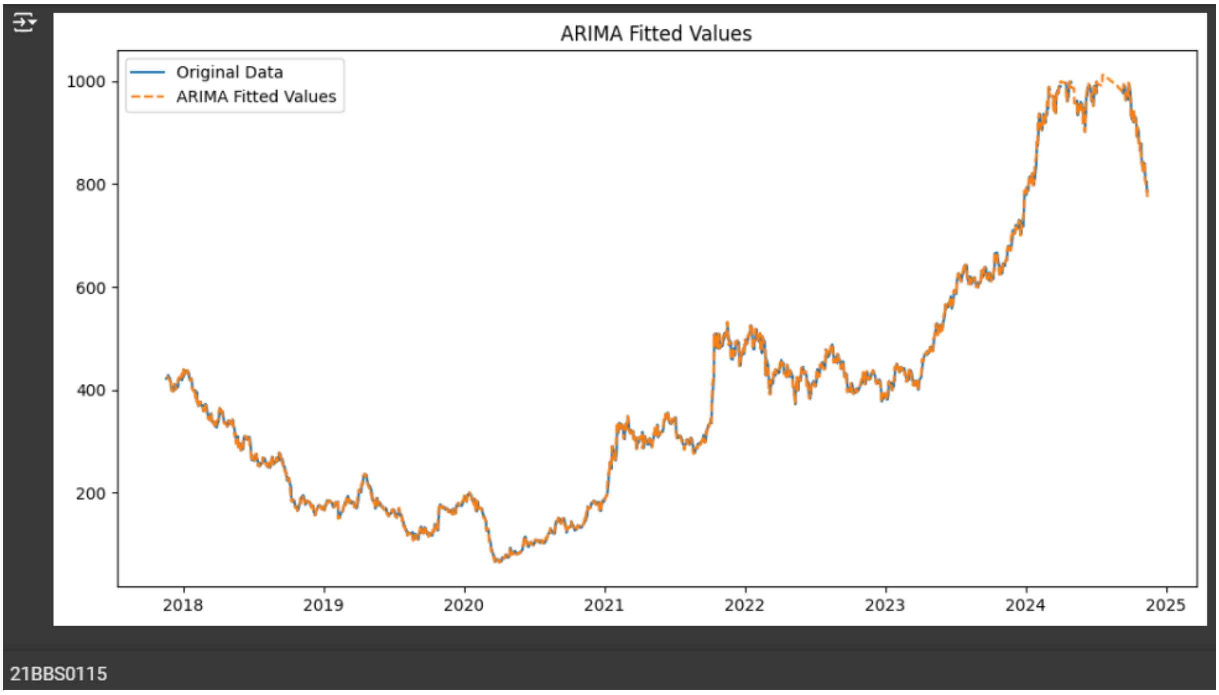
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Price   No. Observations:                 1733
Model:                 ARIMA(2, 1, 2)   Log Likelihood               -5975.619
Date:                Sun, 17 Nov 2024   AIC                          11961.238
Time:                        16:54:05   BIC                          11988.523
Sample:                             0   HQIC                         11971.330
                               - 1733
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1206      0.102      1.181      0.238      -0.080       0.321
ar.L2          0.8443      0.100      8.409      0.000       0.648       1.041
ma.L1         -0.1390      0.112     -1.242      0.214      -0.358       0.080
ma.L2         -0.7989      0.109     -7.296      0.000      -1.013      -0.584
sigma2        75.4123      1.207     62.484      0.000      73.047      77.778
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              6410.40
Prob(Q):                              1.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.37   Skew:                            -0.70
Prob(H) (two-sided):                  0.00   Kurtosis:                        12.32
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

21BBS0115

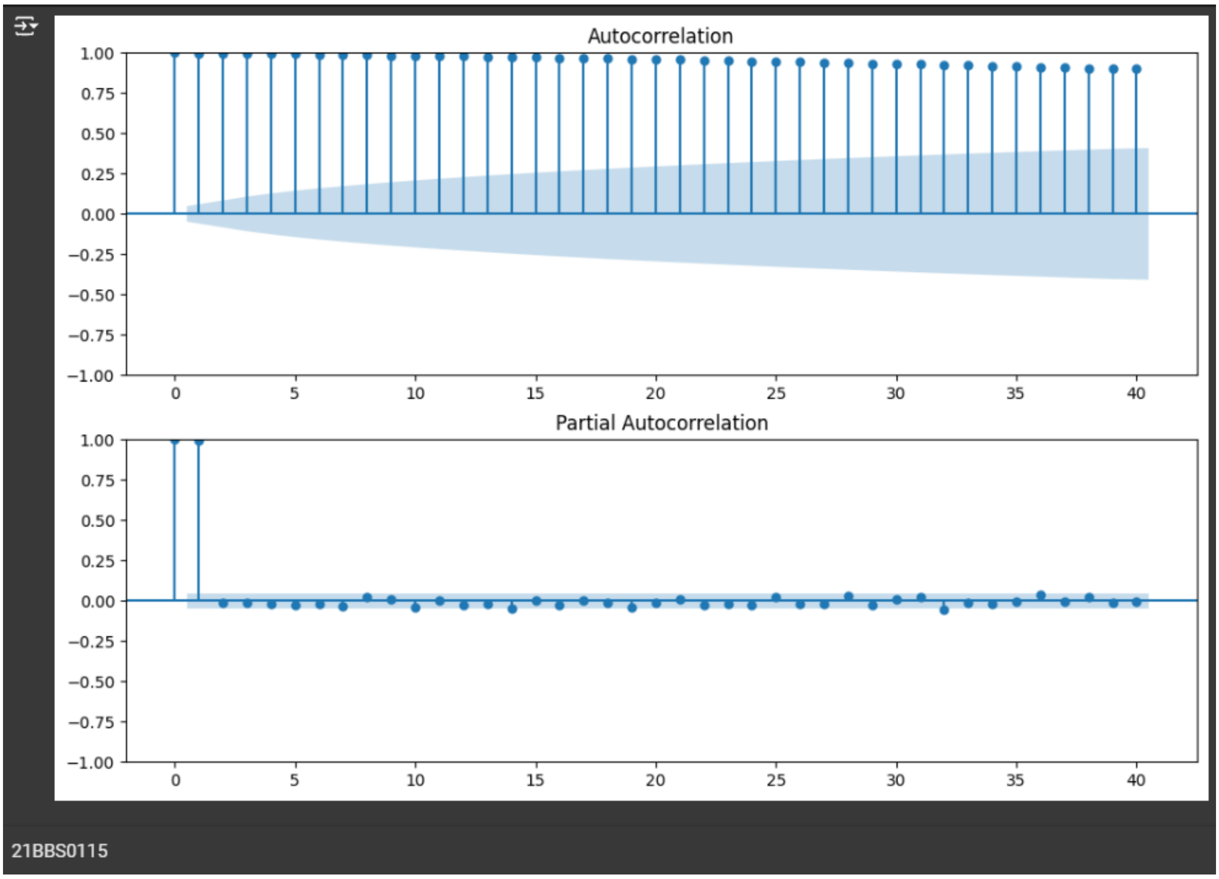
Report



ARIMA Caclulated values(small entries)



ARIMA Caclulated values(all entries)



Result

Result

- The model explains some variation in the Price data, particularly with the AR(2) and MA(2) terms.
- However, the residuals exhibit non-normality and some heteroskedasticity (variance inconsistency), which suggests the model might not fully capture all patterns or may require adjustments (e.g., using a different model or transforming the data).
- Despite this, the model shows reasonable significance for the AR(2) and MA(2) components and appears to handle autocorrelation well, given the Ljung-Box test result.

SECTION 2

Aim : Implement the Logistic regression for predicting the Possibility of enrolling into a university. The dataset can determine the probability of a student getting accepted to a particular university or a degree course in a college by studying the relationship between the estimator variables

Libraries : Numpy, Pandas, sklearn, seaborn

Dataset : https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA5/Q2

Sample Input

	A	B	C	D	E	F	G	H	I
1	CGPA	GRE	GMAT	TOEFL	Research_	Mini_Proje	Internship_	Enrolled	
2	8.4	286	667	91	1	9	4	0	
3	7.89	306	678	94	0	5	0	0	
4	8.52	308	704	107	0	7	3	0	
5	9.22	298	702	106	2	6	4	1	
6	7.81	311	581	99	2	12	0	1	
7	7.81	323	603	101	3	8	4	1	
8	9.26	340	675	112	3	5	3	1	
9	8.61	318	675	94	1	4	2	0	
10	7.62	320	675	105	1	2	3	0	
11	8.43	313	780	97	2	6	6	1	
12	7.63	276	678	97	2	4	3	0	
13	7.63	314	706	110	0	3	3	1	
14	8.19	316	697	108	1	4	2	1	
15	6.47	340	682	108	0	3	1	0	
16	6.62	311	634	113	2	9	4	1	
17	7.55	321	687	100	1	7	1	1	
18	7.19	314	611	106	0	5	3	0	
19	8.25	291	638	96	1	8	2	1	
20	7.27	337	625	103	1	6	3	0	
21	6.87	330	654	98	1	4	3	1	
22	9.17	330	765	100	0	9	4	1	
23	7.82	296	556	105	1	7	2	0	
24	8.05	340	684	91	1	3	2	1	
25	6.86	286	569	120	1	5	1	0	

Code

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


df = pd.read_csv('university_enrollment_data.csv')


X = df.drop('Enrolled', axis=1)

y = df['Enrolled']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

logreg = LogisticRegression()


# Fit the model

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)


print(y_pred)

print("\n\n")

print(y_test)


print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

print("Classification Report:\n", classification_report(y_test, y_pred))

# Define a new student's data
new_entry = {
    'CGPA': 8.5,
    'GRE': 320,
    'GMAT': 650,
    'TOEFL': 105,
    'Research_Articles': 2,
    'Mini_Project_Exp': 3,
    'Internship_Completed': 1
}

# Convert to DataFrame
new_entry_df = pd.DataFrame([new_entry])

new_entry_df.columns = ['CGPA', 'GRE', 'GMAT', 'TOEFL', 'Research_Papers',
      'Mini_Project_Months', 'Internship_Months']

# Prediction
enrollment_prediction = logreg.predict(new_entry_df)
print("Enrollment Prediction:", "Enrolled" if enrollment_prediction[0] == 1 else
"Not Enrolled")
```

Predict probabilities as well

enrollment_probabilities = logreg.predict_proba(new_entry_df)

print("Enrollment Probabilities (Not Enrolled, Enrolled):", enrollment_probabilities[0])

Output

```
df.head()
```

	CGPA	GRE	GMAT	TOEFL	Research_Papers	Mini_Project_Months	Internship_Months	Enrolled
0	8.40	286	667	91	1	9	4	0
1	7.89	306	678	94	0	5	0	0
2	8.52	308	704	107	0	7	3	0
3	9.22	298	702	106	2	6	4	1
4	7.81	311	581	99	2	12	0	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

21BBS0115

First 5 rows

```
# Fit the model
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

print(y_pred)
print("\n\n")
print(y_test)
print("21BBS0115")
```

```
[1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1]



99    1
45    1
4     1
33    1
30    0
92    0
16    0
65    1
37    0
47    0
69    0
15    1
44    0
39    1
23    0
86    1
6     1
50    1
24    1
48    1
Name: Enrolled, dtype: int64
21BBS0115
```

Logistics regression predicted and test datasets

```
Accuracy Score: 0.75
Confusion Matrix:
 [[ 4  4]
 [ 1 11]]

21BBS0115
```

Metric scores

```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.50      0.62         8
           1       0.73      0.92      0.81        12

    accuracy                           0.75        20
   macro avg       0.77      0.71      0.72        20
weighted avg       0.76      0.75      0.74        20
```

21BBS0115

Classification Report

```
Enrollment Prediction: Enrolled
Enrollment Probabilities (Not Enrolled, Enrolled): [0.23799867 0.76200133]
```

21BBS0115

Prediction for new data point

Result

We have created the logistics regression and tested the model on a new entry as well. We have used sklearn libraries.

XXXXXXXXXXXXXXX