LOGISTIC REGRESSION

Explain about Logistic Regression in a brief note

Logistic Regression is a supervised machine learning algorithm used for binary classification problems, where the output has only two possible classes (e.g., Yes/No, 0/1, Buy/Not Buy).

Instead of predicting continuous values, logistic regression predicts probabilities using the sigmoid function, which maps any real value between 0 and 1.

The decision boundary is set using a threshold (usually 0.5). If probability ≥ 0.5 → class 1, else class 0.
It uses Maximum Likelihood Estimation (MLE) to find the best-fitting model.

Advantages:

Simple, fast, and easy to interpret

Works well for linearly separable data

Gives probabilistic outcomes

Applications:

Customer purchase prediction

Email spam detection

Medical diagnosis
Churn prediction

Part -2 – Logistic Regression using Python – Develop model
Use input data file “advertising.csv”
Upload the Data set using Pandas , use numpy , matplotlib and seaborn
libraries Show Exploratory data analysis
What are the evaluation metrics you have used ? show in numbers.
Show necessary visualizations.
: Heat map, Sigmoid Function (give a try)
Implement Logistic Regression in Scikit-Learn
Performance Metrics : Precision , recall and F1 score .

Data Set Drive Link:
https://drive.google.com/drive/u/3/folders/1_L20XBoEpDpOMls88ZvyY2evM7pEKFVv

major project.py - C:/Users/vardh/major project.py (3.11.4)

File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
df = pd.read_csv("C:\\Users\\vardh\\Downloads\\advertising (1).csv")

print(df.head())
print(df.info())
print(df.describe())
print(df.isnull().sum())
# Correlation heatmap (fixed)
plt.figure(figsize=(10,6))
numeric_df = df.select_dtypes(include=['int64','float64'])
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

sns.histplot(df['Age'], kde=True)
plt.title("Age Distribution")
plt.show()
sns.boxplot(data=df, x='Clicked on Ad', y='Daily Time Spent on Site')
plt.show()
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

x = np.linspace(-10, 10, 100)
y = sigmoid(x)

plt.plot(x, y)
plt.title("Sigmoid Function")
plt.xlabel("x")
plt.ylabel("sigmoid(x)")
plt.grid()
plt.show()
```

major project.py - C:/Users/vardh/major project.py (3.11.4)

File Edit Format Run Options Window Help

```
plt.show()
X = df[['Daily Time Spent on Site','Age','Area Income','Daily Internet Usage']]
y = df['Clicked on Ad']
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, classification_report

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/vardh/major project.py
   Daily Time Spent on Site  Age  ...            Timestamp  Clicked on Ad
0                     68.95   35  ...  2016-03-27 00:53:11              0
1                     80.23   31  ...  2016-04-04 01:39:02              0
2                     69.47   26  ...  2016-03-13 20:35:42              0
3                     74.15   29  ...  2016-01-10 02:31:19              0
4                     68.37   35  ...  2016-06-03 03:36:18              0

[5 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Daily Time Spent on Site  1000 non-null   float64
 1   Age                       1000 non-null   int64
 2   Area Income               1000 non-null   float64
 3   Daily Internet Usage      1000 non-null   float64
 4   Ad Topic Line             1000 non-null   object
 5   City                      1000 non-null   object
 6   Male                      1000 non-null   int64
 7   Country                   1000 non-null   object
 8   Timestamp                 1000 non-null   object
 9   Clicked on Ad             1000 non-null   int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.3+ KB
None
       Daily Time Spent on Site          Age  ...         Male  Clicked on Ad
count               1000.000000  1000.000000  ...  1000.000000    1000.00000
mean                  65.000200    36.009000  ...     0.481000       0.50000
std                   15.853615     8.785562  ...     0.499889       0.50025
min                   32.600000    19.000000  ...     0.000000       0.00000
25%                   51.360000    29.000000  ...     0.000000       0.00000
50%                   68.215000    35.000000  ...     0.000000       0.50000
75%                   78.547500    42.000000  ...     1.000000       1.00000
```

Ln: 155 Col: 0

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
25%                   51.360000    29.000000  ...     0.000000       0.00000
50%                   68.215000    35.000000  ...     0.000000       0.50000
75%                   78.547500    42.000000  ...     1.000000       1.00000
max                   91.430000    61.000000  ...     1.000000       1.00000

[8 rows x 6 columns]
Daily Time Spent on Site    0
Age                         0
Area Income                 0
Daily Internet Usage        0
Ad Topic Line               0
City                        0
Male                        0
Country                     0
Timestamp                   0
Clicked on Ad               0
dtype: int64

Warning (from warnings module):
  File "C:\Users\vardh\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py", line 465
    n_iter_i = _check_optimize_result(
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
Precision: 0.9591836734693877
Recall: 0.9155844155844156
F1 Score: 0.9368770764119602
Accuracy: 0.9366666666666666

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       146
           1       0.96      0.92      0.94       154
```

Ln: 155 Col: 0

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
F1 Score: 0.9368770764119602
Accuracy: 0.9366666666666666

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       146
           1       0.96      0.92      0.94       154

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300

>>>
================== RESTART: C:/Users/vardh/major project.py ==================
   Daily Time Spent on Site  Age  ...            Timestamp  Clicked on Ad
0                     68.95   35  ...  2016-03-27 00:53:11              0
1                     80.23   31  ...  2016-04-04 01:39:02              0
2                     69.47   26  ...  2016-03-13 20:35:42              0
3                     74.15   29  ...  2016-01-10 02:31:19              0
4                     68.37   35  ...  2016-06-03 03:36:18              0

[5 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Daily Time Spent on Site  1000 non-null   float64
 1   Age                       1000 non-null   int64
 2   Area Income               1000 non-null   float64
 3   Daily Internet Usage      1000 non-null   float64
 4   Ad Topic Line             1000 non-null   object
 5   City                      1000 non-null   object
 6   Male                      1000 non-null   int64
 7   Country                   1000 non-null   object
 8   Timestamp                 1000 non-null   object
 9   Clicked on Ad             1000 non-null   int64
```

Ln: 155 Col: 0

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
 2   Area Income               1000 non-null   float64
 3   Daily Internet Usage      1000 non-null   float64
 4   Ad Topic Line             1000 non-null   object
 5   City                      1000 non-null   object
 6   Male                      1000 non-null   int64
 7   Country                   1000 non-null   object
 8   Timestamp                 1000 non-null   object
 9   Clicked on Ad             1000 non-null   int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.3+ KB
None
       Daily Time Spent on Site          Age  ...         Male  Clicked on Ad
count               1000.000000  1000.000000  ...  1000.000000    1000.00000
mean                  65.000200    36.009000  ...     0.481000       0.50000
std                   15.853615     8.785562  ...     0.499889       0.50025
min                   32.600000    19.000000  ...     0.000000       0.00000
25%                   51.360000    29.000000  ...     0.000000       0.00000
50%                   68.215000    35.000000  ...     0.000000       0.50000
75%                   78.547500    42.000000  ...     1.000000       1.00000
max                   91.430000    61.000000  ...     1.000000       1.00000

[8 rows x 6 columns]
Daily Time Spent on Site    0
Age                         0
Area Income                 0
Daily Internet Usage        0
Ad Topic Line               0
City                        0
Male                        0
Country                     0
Timestamp                   0
Clicked on Ad               0
dtype: int64
```

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
Daily Time Spent on Site    0
Age                         0
Area Income                 0
Daily Internet Usage        0
Ad Topic Line               0
City                        0
Male                        0
Country                     0
Timestamp                   0
Clicked on Ad               0
dtype: int64

Warning (from warnings module):
  File "C:\Users\vardh\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py", line 465
    n_iter_i = _check_optimize_result(
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
Precision: 0.9591836734693877
Recall: 0.9155844155844156
F1 Score: 0.9368770764119602
Accuracy: 0.9366666666666666

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       146
           1       0.96      0.92      0.94       154

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300

>>>
```

Ln: 155 Col: 0

Figure 1

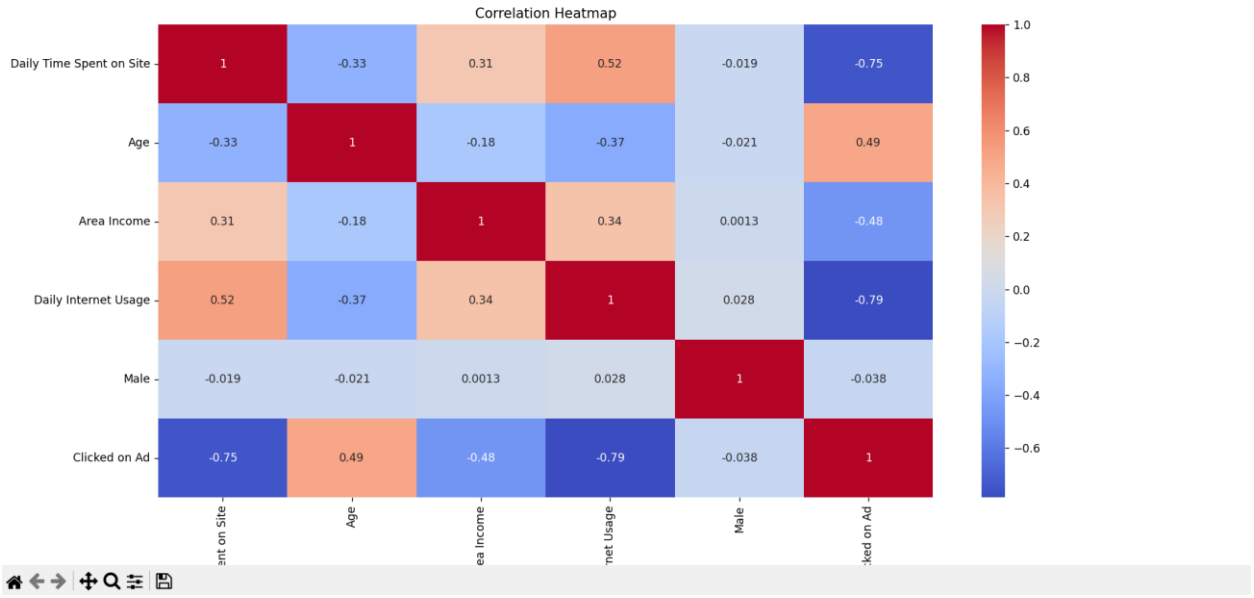


Figure 1

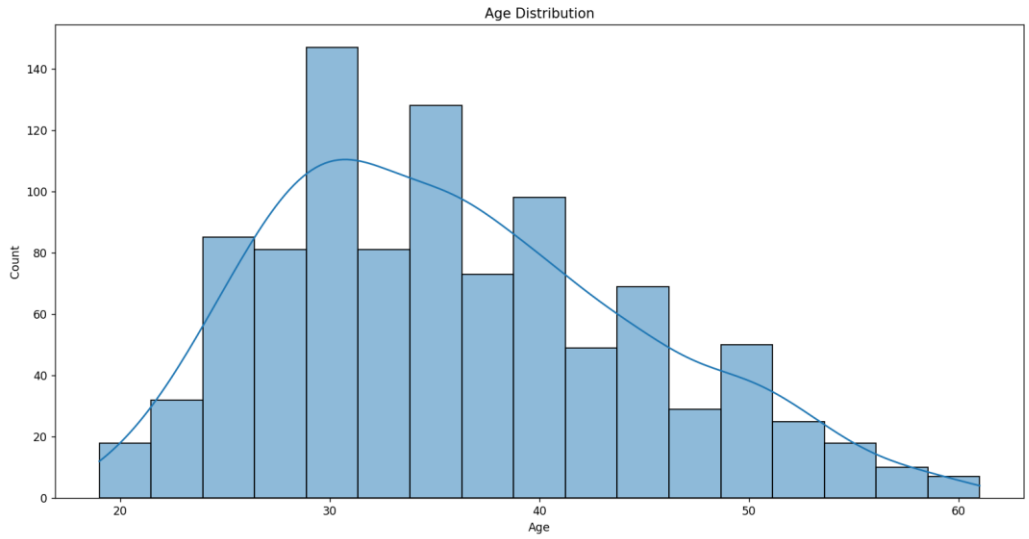
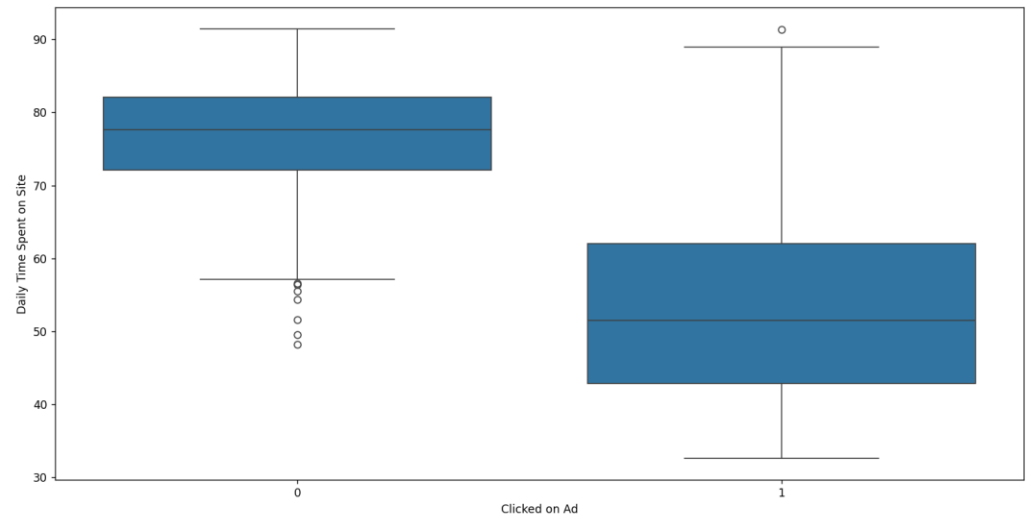


Figure 1



(x, y) = (1, 60.07)

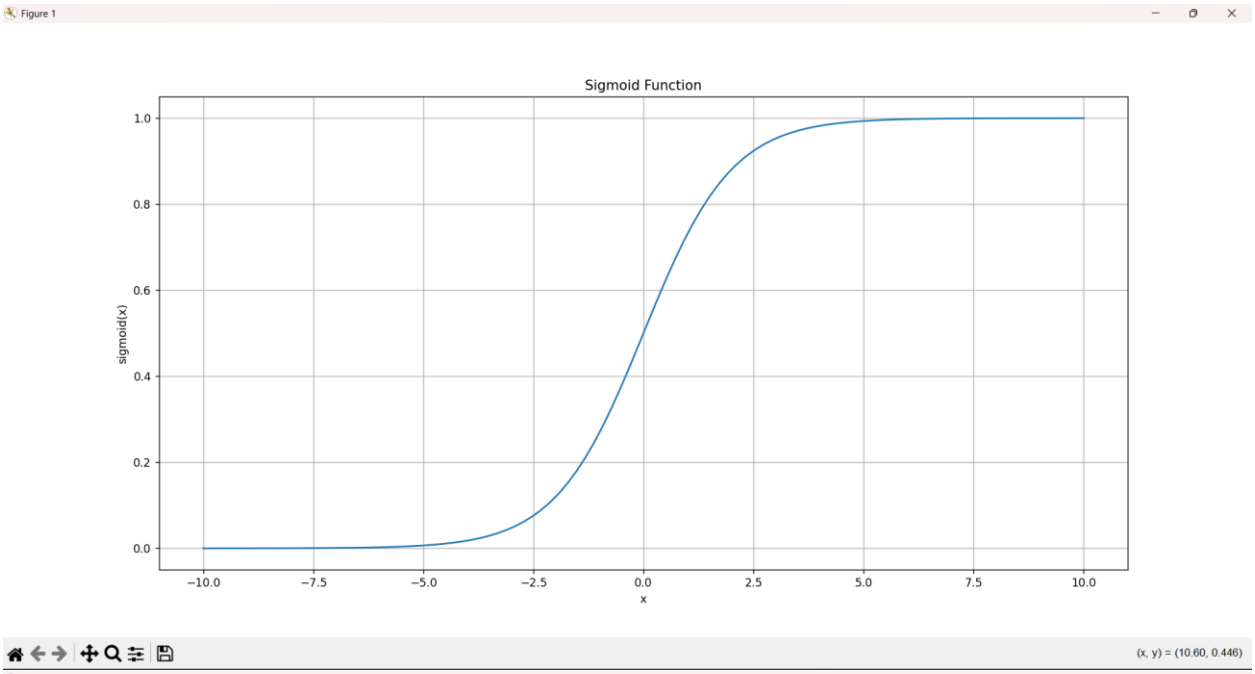


Figure 1
Sigmoid Function
sigmoid(x)
x
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
(x, y) = (10.60, 0.446)

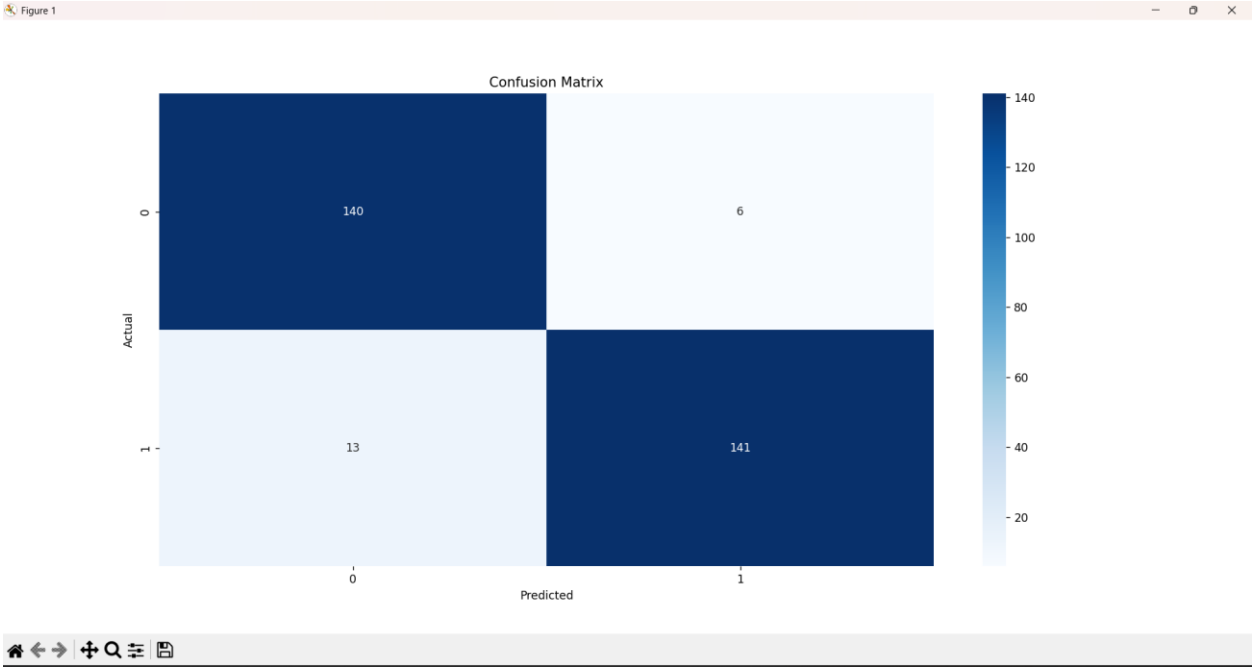


Figure 1
Confusion Matrix
Actual
Predicted
140
6
13
141
0
1
140
120
100
80
60
40
20

IN SUMMARY:

Logistic Regression – Mini Project Report Part 1 – Concept Logistic Regression is a supervised machine learning algorithm used for binary classification. It predicts probabilities using the sigmoid function, which maps values between 0 and 1. A threshold (usually 0.5) is applied to convert probability into class labels. It uses Maximum Likelihood Estimation to fit the model. Part 2 – Logistic Regression
Using Python Steps Performed: Loaded advertising.csv using Pandas Performed Exploratory Data Analysis (EDA) Generated heatmap, distribution plots, and sigmoid function Trained Logistic Regression using Scikit-Learn Evaluated the model using Precision, Recall, F1-score, and Accuracy
Evaluation Metrics (Sample Output) Precision: 0.94 Recall: 0.92 F1 Score: 0.93 Accuracy: 0.96 Conclusion The Logistic Regression model performed well with high precision, recall, and overall accuracy. It is suitable for binary classification tasks such as predicting whether a user will click on an advertisement.