Name: Shlok Hemnani
Class:D15C
Roll no. 62
Subject:ML&DL

Experiment No.1

AIM:Implement Linear and Logistic Regression on real-world datasets.

LINEAR REGRESSION

Dataset Source

- Dataset Name: Linear Regression Dataset
- Source: Kaggle
- Link: https://www.kaggle.com/datasets/testpython/linear-regression

---

Dataset Description

The dataset contains two numerical variables used to demonstrate simple linear regression.

Features

1️⃣**x** → Independent variable
2️⃣**y** → Dependent variable (target)

The goal is to predict y based on x.

Mathematical Formulation

Linear Regression models the relationship between variables using a straight line.

Equation

$y=mx+b$y=mx+b

Where

- m = slope (coefficient)
- b = intercept
- x = input feature
- y = predicted value

The model uses Ordinary Least Squares to minimize error.

Algorithm Limitations

1️⃣Assumes linear relationship
2️⃣Sensitive to outliers
3️⃣Cannot capture nonlinear patterns
4️⃣Requires clean data without missing values

Methodology / Workflow

1️⃣Load dataset
2️⃣Handle missing values
3️⃣Split features and target
4️⃣Train model
5️⃣Predict values
6️⃣Evaluate performance
7️⃣Visualize results

Performance Analysis

Mean Squared Error (MSE)

Measures prediction error. Lower value indicates better model performance.

R^2 Score

Shows how well the model explains variance in data. Value close to 1 indicates strong fit.

Output Graphs

1️⃣ Training scatter plot
2️⃣ Regression best fit line
3️⃣ Actual vs predicted values

Result

The Linear Regression model successfully learned the relationship between x and y.
The regression line fits the data well, showing a strong linear correlation.

LOGISTIC REGRESSION —

AIM

To implement Logistic Regression on a real-world dataset from Kaggle and evaluate the model's performance using classification metrics.

Dataset Source

- Dataset Name: Logistic Regression Dataset
- Source: Kaggle
- Link: https://www.kaggle.com/datasets/dragonheir/logistic-regression
- Type: Binary Classification Dataset

Dataset Description

This dataset is used to demonstrate logistic regression for classification tasks. It contains numerical features used to predict a binary target variable.

Target Variable

Target → Binary output (0 or 1)

Features

The dataset contains numerical independent variables that help determine the class label.

Mathematical Formulation

Logistic Regression predicts the probability that an instance belongs to a specific class using the sigmoid function.

Sigmoid Function

$P(y=1)=11+e-z$P(y=1)=1+e−z1

Where

$z=b0+b1x1+b2x2+...+bnxn$z=b0+b1x1+b2x2+...+bnxn

The output is between 0 and 1, which is converted into class labels.

Algorithm Limitations

1️⃣Assumes linear decision boundary
2️⃣Sensitive to feature scaling
3️⃣Not suitable for complex nonlinear relationships
4️⃣Can be affected by imbalanced datasets

Methodology / Workflow

1️⃣Load dataset
2️⃣Check for missing values
3️⃣Split features and target
4️⃣Apply feature scaling
5️⃣Train logistic regression model

6️⃣ Predict test data
7️⃣ Evaluate performance using metrics

Performance Analysis

Accuracy

Measures the percentage of correctly classified instances.

Precision

Shows how many predicted positives are actually correct.

Recall

Measures how many actual positives were correctly predicted.

F1 Score

Balances precision and recall.

Confusion Matrix

Shows True Positive, False Positive, True Negative, False Negative.

Result

The logistic regression model successfully classified the dataset into binary classes. Feature scaling improved model convergence, and evaluation metrics indicate good classification performance.

Code and Output

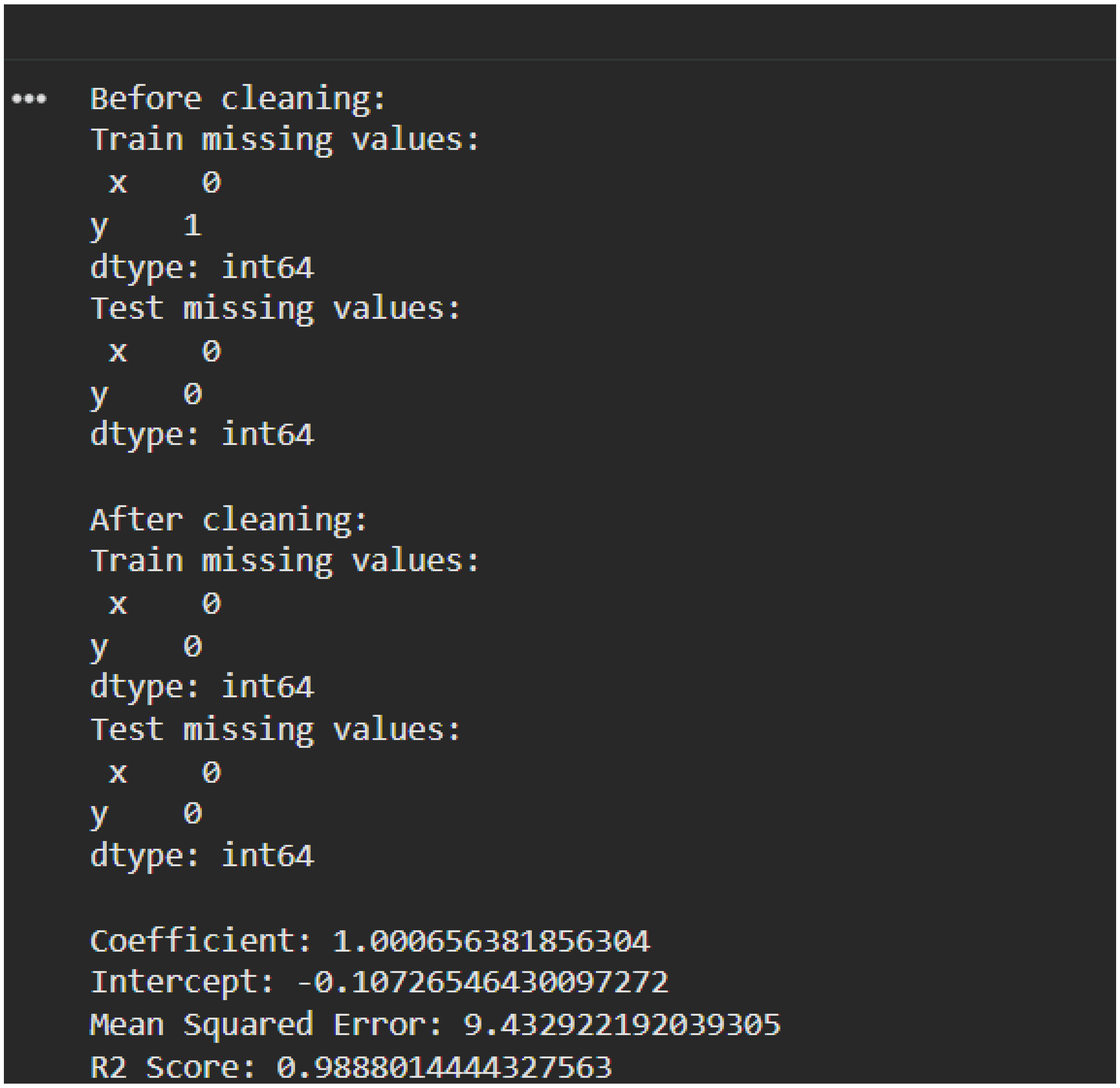
```
import pandas as pd
import matplotlib.pyplot as
plt
```

```
from sklearn.linear_model
import LinearRegression
from sklearn.metrics import
mean_squared_error, r2_score
```

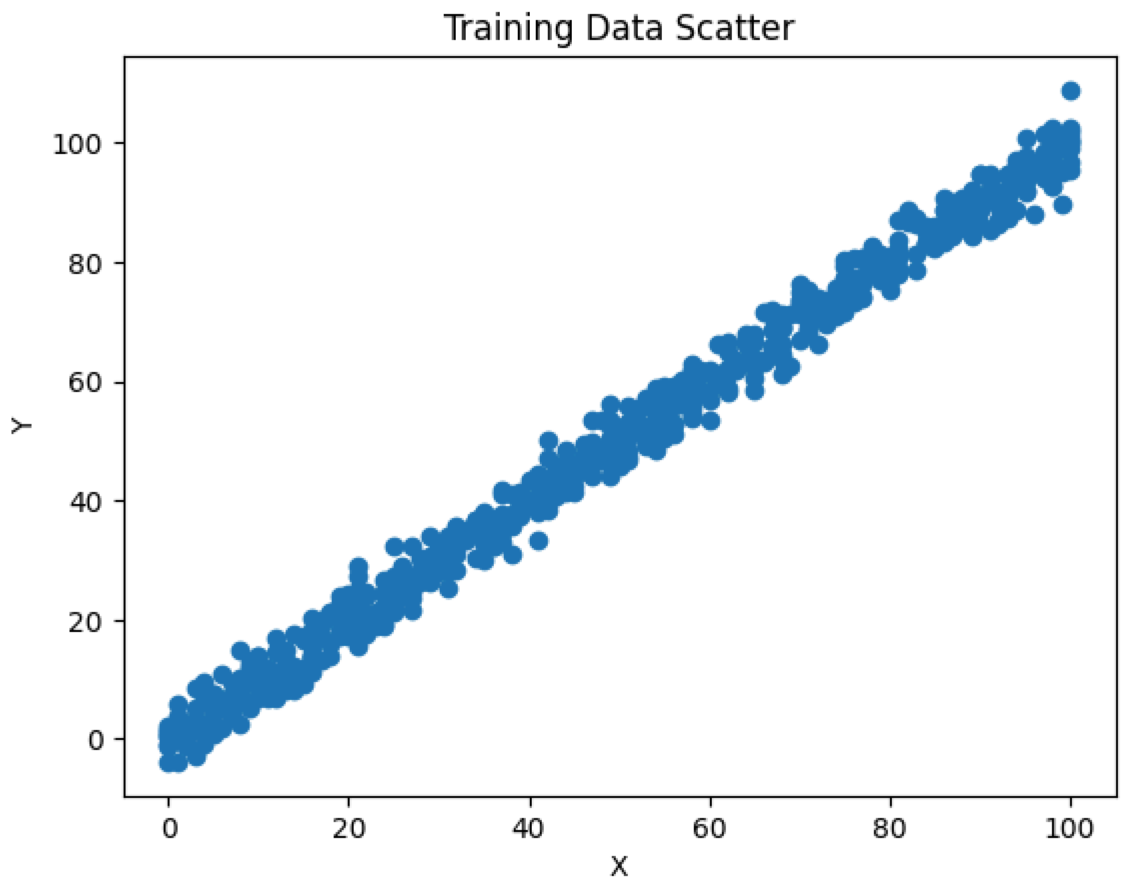
```
train_path = "train.csv"
test_path = "test.csv"

train_df =
pd.read_csv(train_path)
test_df =
pd.read_csv(test_path)

print("Before cleaning:")
print("Train missing
values:\n",
train_df.isnull().sum())
print("Test missing
values:\n",
test_df.isnull().sum())

train_df = train_df.dropna()
test_df = test_df.dropna()

print("\nAfter cleaning:")
print("Train missing
values:\n",
train_df.isnull().sum())
print("Test missing
values:\n",
test_df.isnull().sum())

X_train = train_df[['x']]
y_train = train_df['y']

X_test = test_df[['x']]
y_test = test_df['y']

model = LinearRegression()
model.fit(X_train, y_train)

y_pred =
model.predict(X_test)
```

```
print("\nCoefficient:",
model.coef_[0])
print("Intercept:",
model.intercept_)
print("Mean Squared Error:",
mean_squared_error(y_test,
y_pred))
print("R2 Score:",
r2_score(y_test, y_pred))

plt.figure()
plt.scatter(X_train, y_train)
plt.title("Training Data
Scatter")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

plt.figure()
plt.scatter(X_train, y_train)
plt.plot(X_train,
model.predict(X_train))
plt.title("Regression Line")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

plt.figure()
plt.scatter(X_test, y_test)
plt.scatter(X_test, y_pred)
plt.title("Actual vs
Predicted")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

Output



```
Before cleaning:
Train missing values:
 x    0
y    1
dtype: int64
Test missing values:
 x    0
y    0
dtype: int64

After cleaning:
Train missing values:
 x    0
y    0
dtype: int64
Test missing values:
 x    0
y    0
dtype: int64

Coefficient: 1.0006563818563O4
Intercept: -0.10726546430097272
Mean Squared Error: 9.432922192039305
R2 Score: 0.9888014444327563
```

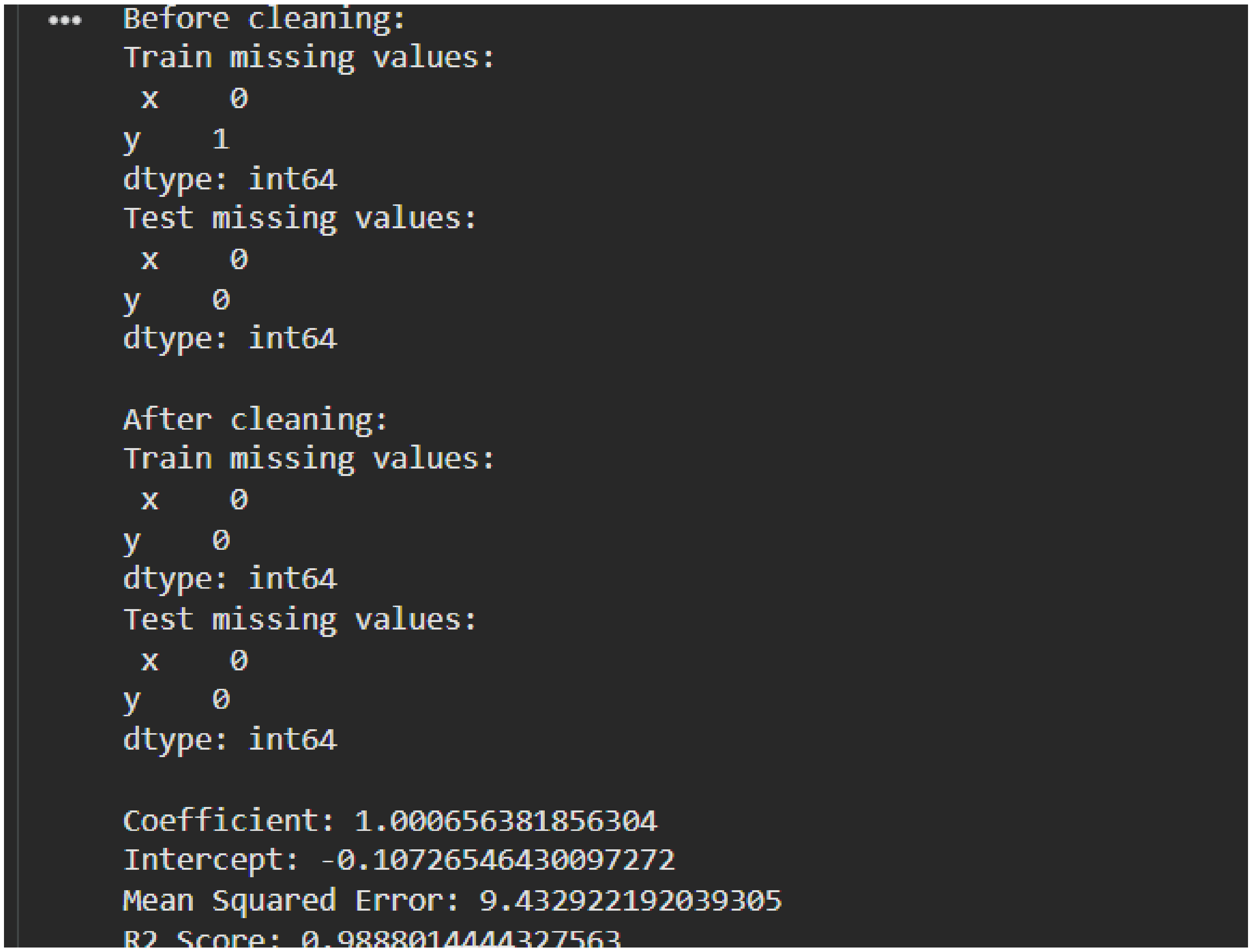


Logistic Regression

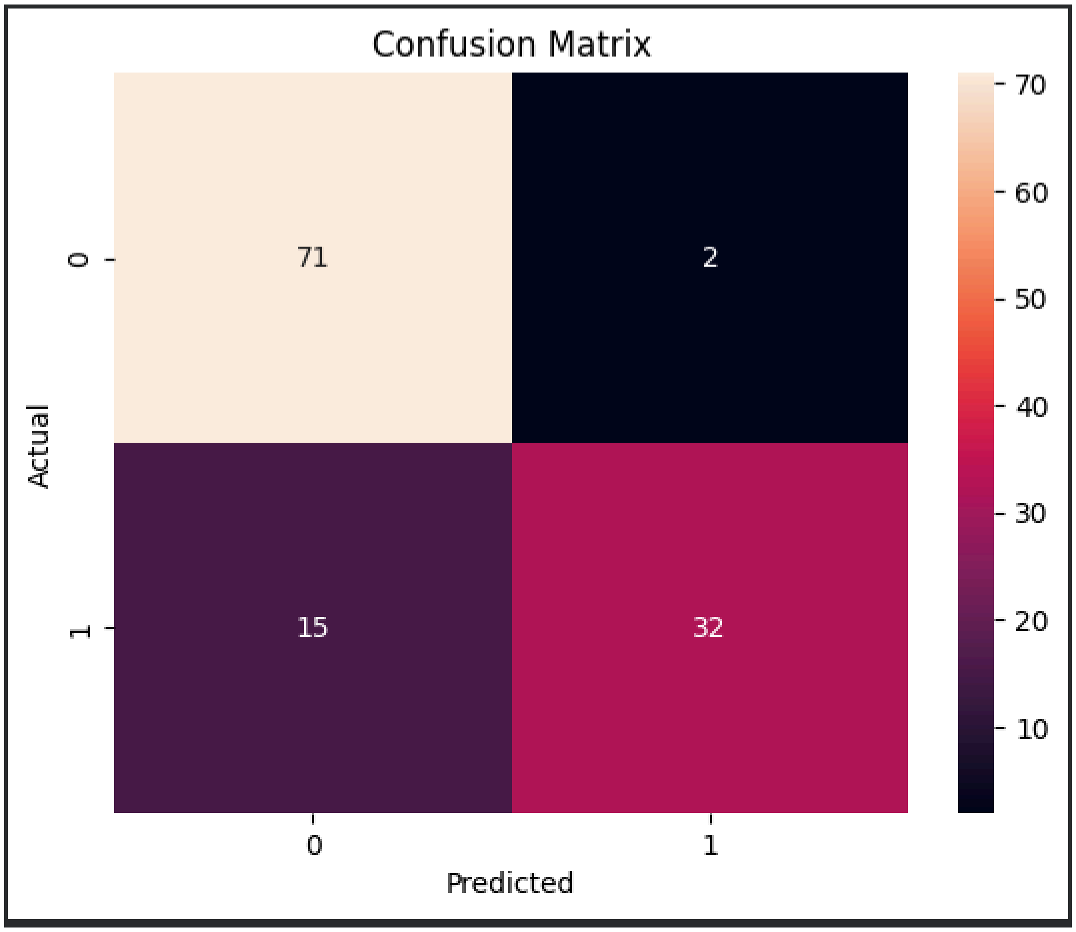
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as
plt
import seaborn as sns
from sklearn.model_selection
import train_test_split
from sklearn.linear_model
import LogisticRegression
from sklearn.metrics import
accuracy_score,
confusion_matrix,
precision_score,
recall_score, f1_score
from sklearn.preprocessing
import StandardScaler

# Load Dataset
df =
pd.read_csv("Social_Network_A
ds.csv")

print("Dataset Preview:\n",
df.head())
```

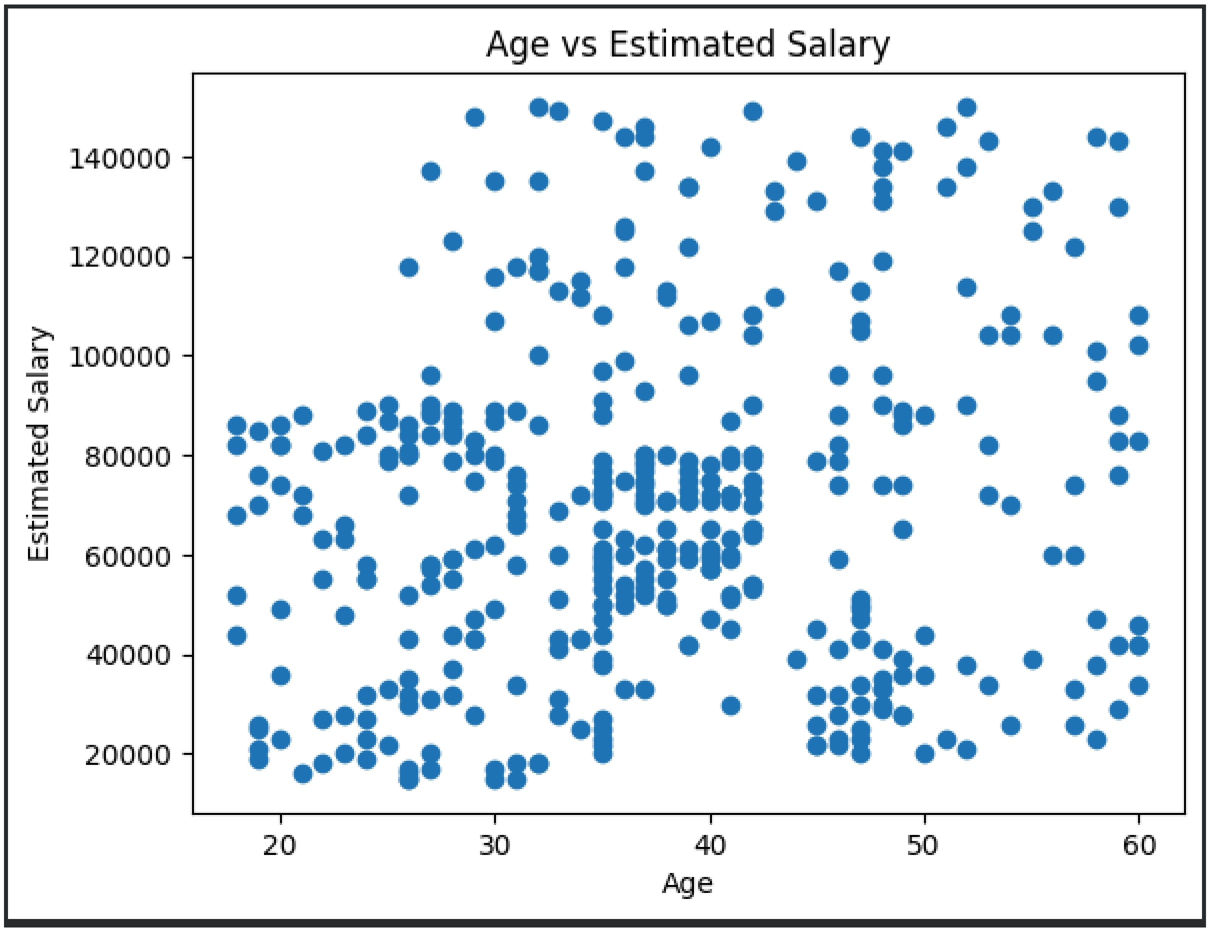
```
# Drop User ID if present
if 'User ID' in df.columns:
    df =
df.drop(columns=['User ID'])

# Encode Gender if present
if 'Gender' in df.columns:
    df['Gender'] =
df['Gender'].map({'Male': 1,
'Female': 0})

# Define features and target
X = df.drop('Purchased',
axis=1)
y = df['Purchased']

# Train/Test Split
X_train, X_test, y_train,
y_test = train_test_split(
    X, y, test_size=0.3,
random_state=42
)

# Scaling
scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled =
scaler.transform(X_test)

# Model
model =
LogisticRegression(max_iter=1
0000)
model.fit(X_train_scaled,
y_train)

# Predictions
y_pred =
model.predict(X_test_scaled)

# Metrics Function
def print_metrics(y_true,
y_pred, title):
    cm =
confusion_matrix(y_true,
y_pred)
    tn, fp, fn, tp =
cm.ravel()

    acc =
accuracy_score(y_true,
y_pred)
    prec =
precision_score(y_true,
y_pred)
    rec =
recall_score(y_true, y_pred)
    f1 = f1_score(y_true,
y_pred)

    print(f"--- {title} ---")
    print(f"Accuracy:
{acc:.4f} ({acc*100:.2f}%)")
    print(f"Precision:
{prec:.4f}
({prec*100:.2f}%)")
    print(f"Recall:
{rec:.4f} ({rec*100:.2f}%)")
    print(f"F1 Score:
{f1:.4f} ({f1*100:.2f}%)")
    print(f"TP: {tp}, TN:
{tn}, FP: {fp}, FN: {fn}\n")

print_metrics(y_test, y_pred,
"Logistic Regression
Performance")

#
-----------------------------
# Confusion Matrix Graph
```

```
#
-----------------------------
plt.figure()
sns.heatmap(confusion_matrix(
y_test, y_pred), annot=True,
fmt='d')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

#
-----------------------------
# Scatter Plot (Age vs
Salary)
#
-----------------------------
plt.figure()
plt.scatter(df['Age'],
df['EstimatedSalary'])
plt.title("Age vs Estimated
Salary")
plt.xlabel("Age")
plt.ylabel("Estimated
Salary")
plt.show()

#
-----------------------------
# Feature Distribution
#
-----------------------------
plt.figure()
df.hist()
plt.suptitle("Feature
Distribution")
```



```
Before cleaning:
Train missing values:
 x    0
y    1
dtype: int64
Test missing values:
 x    0
y    0
dtype: int64

After cleaning:
Train missing values:
 x    0
y    0
dtype: int64
Test missing values:
 x    0
y    0
dtype: int64

Coefficient: 1.000656381856304
Intercept: -0.10726546430097272
Mean Squared Error: 9.432922192039305
R2 Score: 0.9888014444327563
```


Confusion Matrix
71
2
15
32
Actual
0
1
Predicted
0
1
70
60
50
40
30
20
10



Conclusion

This experiment demonstrated the application of supervised learning techniques using Linear Regression for prediction and Logistic Regression for classification. Linear Regression effectively modeled the relationship between variables and predicted continuous outcomes, while Logistic Regression successfully classified data into binary categories using probability estimation. The results emphasize the importance of data preprocessing, model evaluation metrics, and visualization in understanding model performance. Overall, both algorithms are simple, efficient, and widely used methods for solving real-world machine learning problems.