Ex.No – 7 Roll No – 210701149

Implement Linear and Logistic Regression

AIM:

To implement linear and logistic regression techniques in machine learning.

PROCEDURES:

1. Collect and load the dataset from sources like CSV files or databases.
2. Clean and preprocess the data, including handling missing values and encoding categorical variables.
3. Split the dataset into training and testing sets to evaluate model performance.
4. Normalize or standardize the features to ensure consistent scaling.
5. Choose the appropriate model: Linear Regression for continuous outcomes, Logistic Regression for binary outcomes.
6. Train the model on the training data using the `fit` method.
7. Make predictions on the testing data using the `predict` method.
8. Evaluate the model using metrics like Mean Squared Error (MSE) for Linear Regression or accuracy and confusion matrix for Logistic Regression.
9. Visualize the results with plots, such as scatter plots for Linear Regression or decision boundaries for Logistic Regression.
10. Fine-tune the model by adjusting hyperparameters or applying regularization techniques.

CODE:

LinearRegression.py

```
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

# Generating sample data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Creating and training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)

# Evaluating the model
```

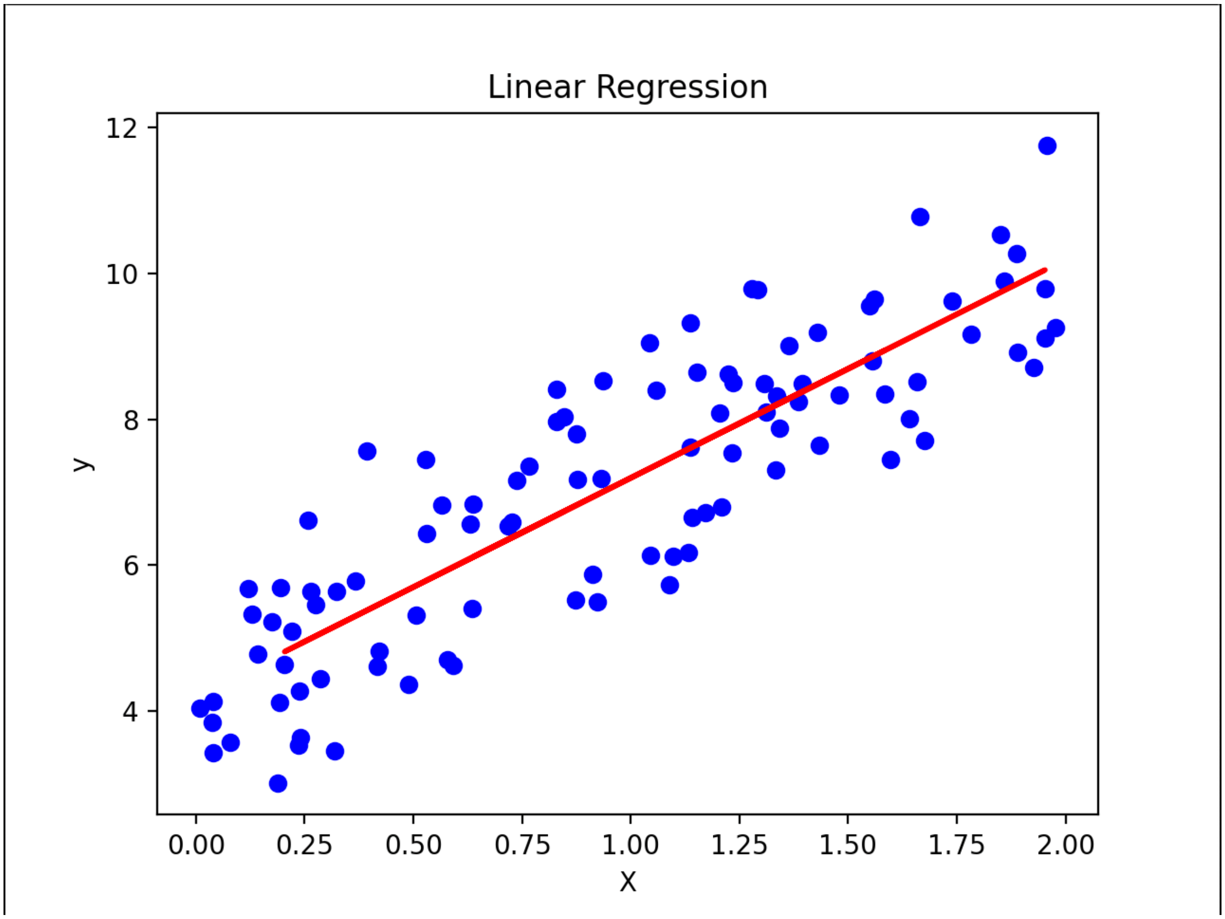
```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Plotting the results
plt.scatter(X, y, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.title('Linear Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```

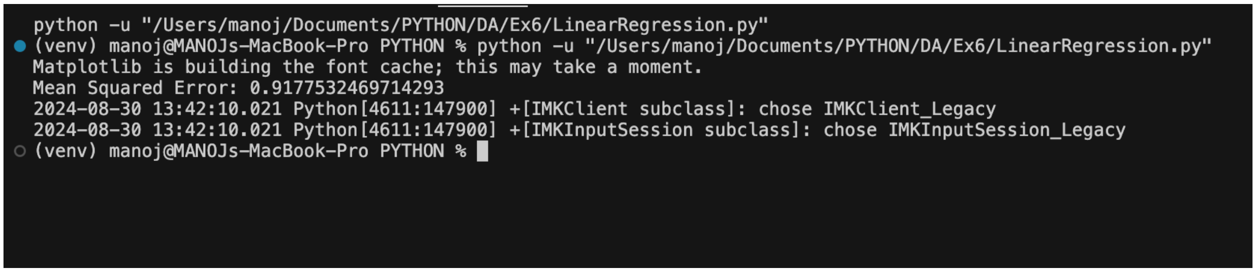
LogisticRegression.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Generating sample data
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = (X > 5).astype(int).ravel()  # Classify as 1 if X > 5, else 0

# Splitting the data into training and testing sets
```

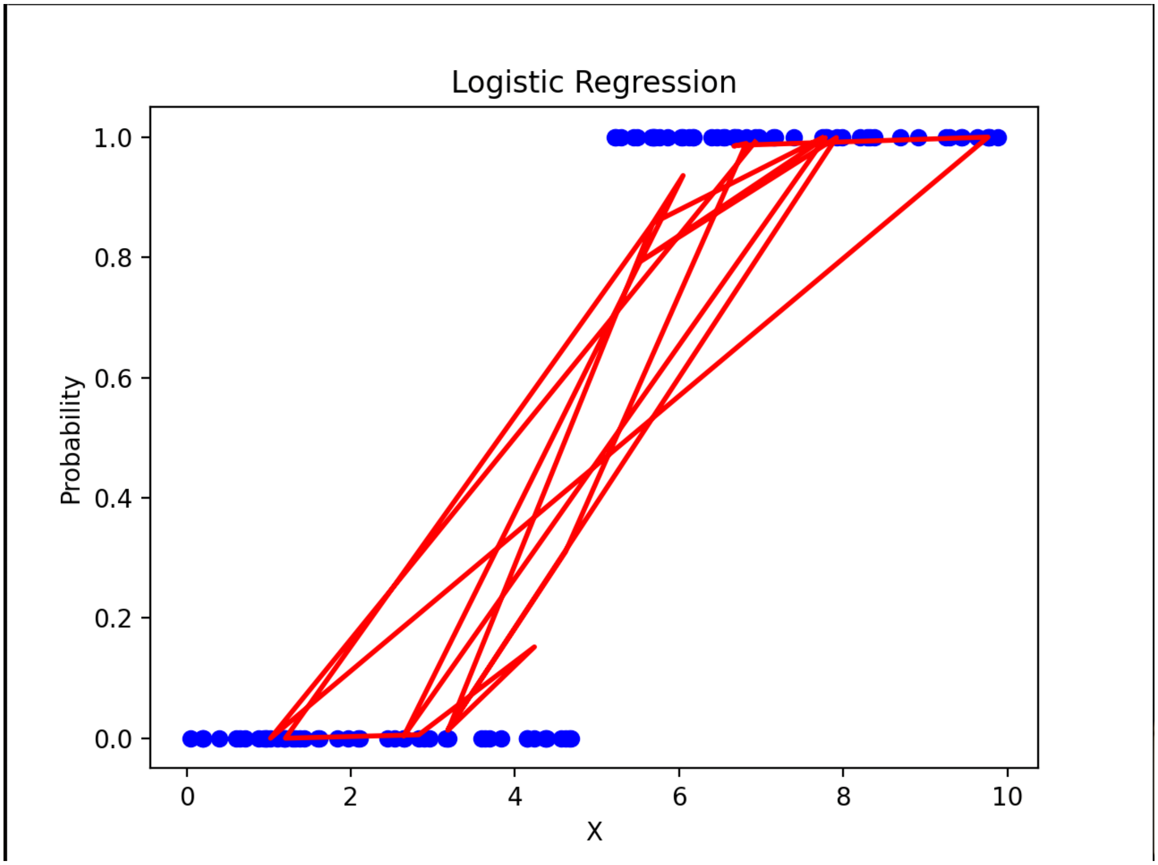
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Creating and training the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

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

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

# Plotting the decision boundary
plt.scatter(X, y, color='blue')
plt.plot(X_test, model.predict_proba(X_test)[:, 1], color='red', linewidth=2)
plt.title('Logistic Regression')
plt.xlabel('X')
plt.ylabel('Probability')
plt.show()
```

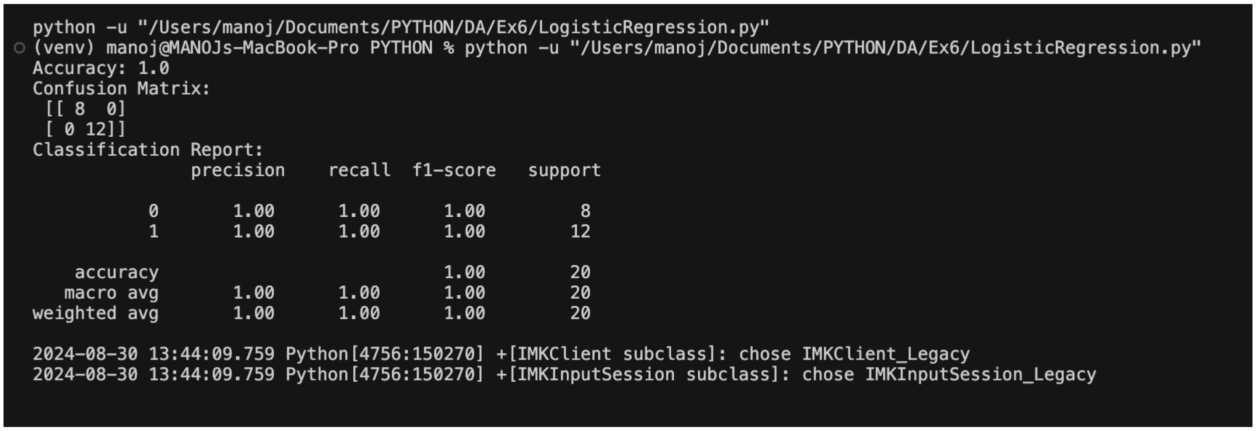
OUTPUT:





```
  python -u "/Users/manoj/Documents/PYTHON/DA/Ex6/LinearRegression.py"
● (venv) manoj@MANOJs-MacBook-Pro PYTHON % python -u "/Users/manoj/Documents/PYTHON/DA/Ex6/LinearRegression.py"
  Matplotlib is building the font cache; this may take a moment.
  Mean Squared Error: 0.9177532469714293
  2024-08-30 13:42:10.021 Python[4611:147900] +[IMKClient subclass]: chose IMKClient_Legacy
  2024-08-30 13:42:10.021 Python[4611:147900] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
○ (venv) manoj@MANOJs-MacBook-Pro PYTHON %
```





```
python -u "/Users/manoj/Documents/PYTHON/DA/Ex6/LogisticRegression.py"
○ (venv) manoj@MANOJs-MacBook-Pro PYTHON % python -u "/Users/manoj/Documents/PYTHON/DA/Ex6/LogisticRegression.py"
Accuracy: 1.0
Confusion Matrix:
 [[ 8  0]
 [ 0 12]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00        12

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

2024-08-30 13:44:09.759 Python[4756:150270] +[IMKClient subclass]: chose IMKClient_Legacy
2024-08-30 13:44:09.759 Python[4756:150270] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
```

RESULT:

Thus, to implement linear and logistic regression using machine learning is completed successfully.