Lab Assignment : 1

Name: Siddharth Vikram Maral

Roll No: CS3172

T.Y. – C.S. (A)

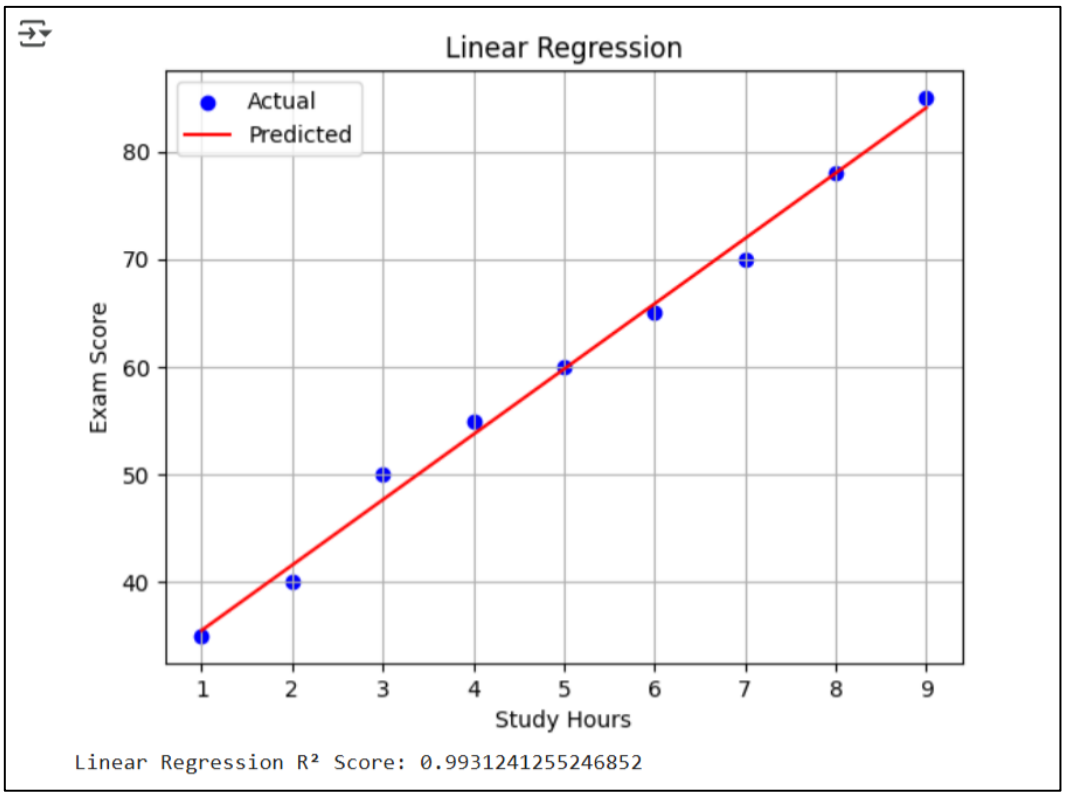
Code Implementation :

1) Linear Regression :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Dataset: Study Hours vs Exam Scores
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9]).reshape(-1, 1)
y = np.array([35, 40, 50, 55, 60, 65, 70, 78, 85])

# Model training
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

# Visualization
plt.scatter(X, y, color='blue', label='Actual')
plt.plot(X, y_pred, color='red', label='Predicted')
plt.title('Linear Regression')
plt.xlabel('Study Hours')
plt.ylabel('Exam Scores')
plt.legend()
plt.grid(True)
```

```
plt.show()

# Evaluation
print("R² Score (Linear Regression):", r2_score(y, y_pred))
```

Output :

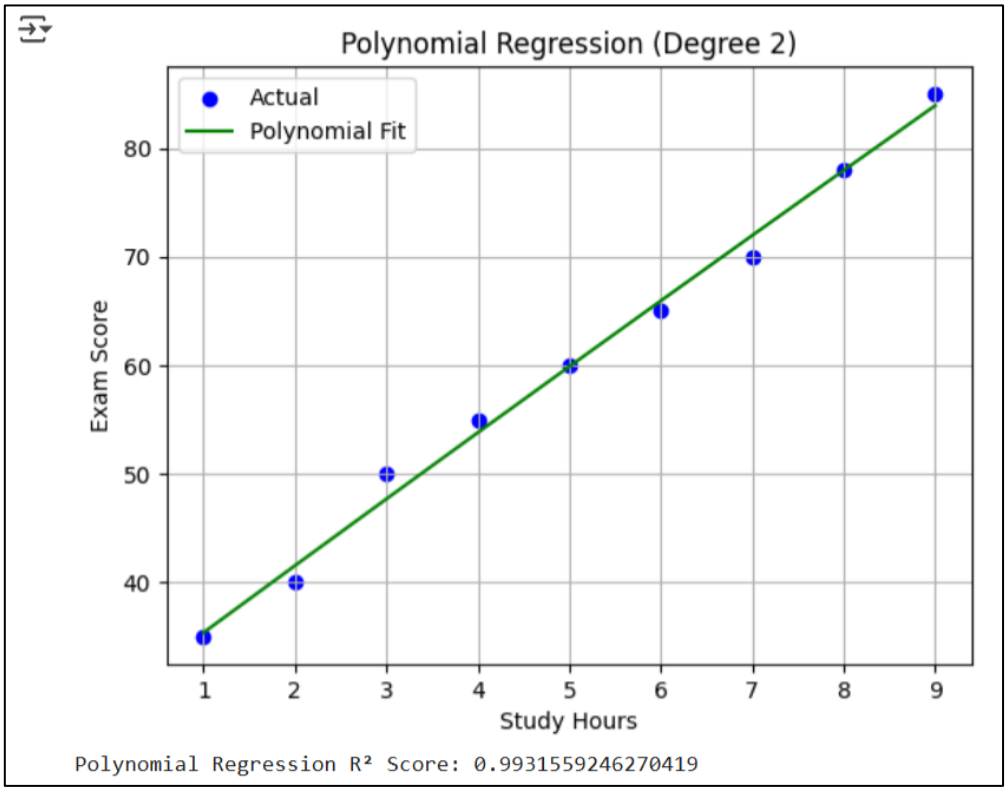


2) Polynomial Regression :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures

# Dataset: Study Hours vs Exam Scores
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9]).reshape(-1, 1)
y = np.array([35, 40, 50, 55, 60, 65, 70, 78, 85])
```

```
# Transform to polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
poly_model = LinearRegression()
poly_model.fit(X_poly, y)
y_poly_pred = poly_model.predict(X_poly)
plt.scatter(X, y, color='blue', label='Actual')
plt.plot(X, y_poly_pred, color='green', label='Polynomial Fit')
plt.title('Polynomial Regression (Degree 2)')
plt.xlabel('Study Hours')
plt.ylabel('Exam Score')
plt.legend()
plt.grid(True)
plt.show()
print("Polynomial Regression R² Score:", r2_score(y, y_poly_pred))
```

Output :

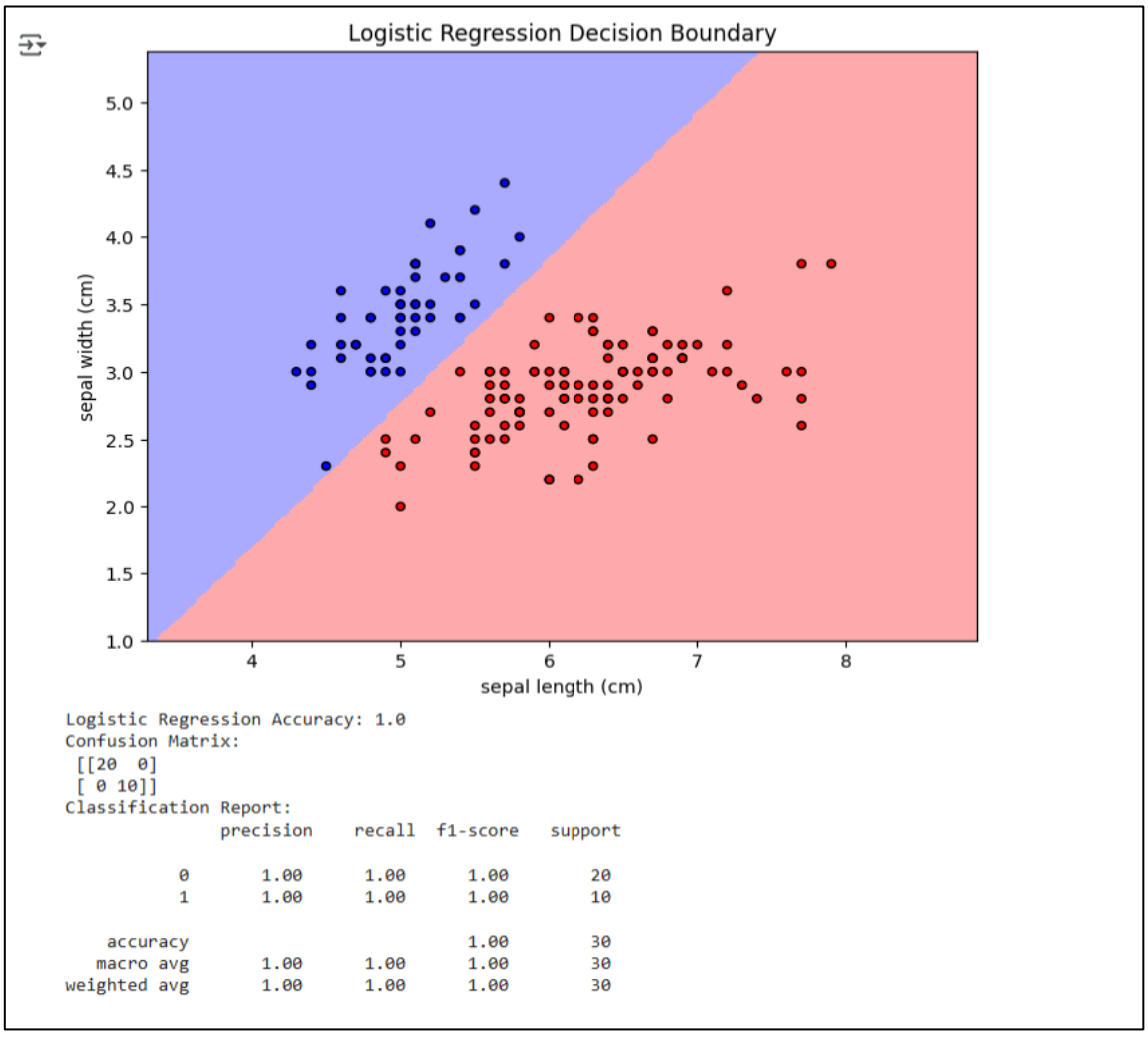


3) Logistic Regression :

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

# Load dataset
iris = load_iris()
X = iris.data[:, :2]  # Use first 2 features for simplicity
y = (iris.target == 0).astype(int)  # Setosa = 1, Others = 0
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train logistic regression
log_model = LogisticRegression()
log_model.fit(X_train, y_train)

cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
np.arange(y_min, y_max, .02))
Z = log_model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolor='k', s=20)
plt.title('Logistic Regression Decision Boundary')
```

```
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.show()

# Output evaluation metrics
y_pred = log_model.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Output :




```
Logistic Regression Accuracy: 1.0
Confusion Matrix:
 [[20  0]
 [ 0 10]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```