Objective: To provide hands-on experience to students in implementing and applying linear regression models and evaluate the performance.

1. Download the following dataset: Wine Quality Dataset: Predicting wine quality based on various features. URL: https://archive.ics.uci.edu/dataset/186/wine+quality

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv"
df = pd.read_csv(url, sep=";")

```

```
print("Statistical values of the dataset:")
print(df.describe())
print("\nShape of the data:", df.shape)
```

```
Statistical values of the dataset:
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4898.000000       4898.000000  4898.000000     4898.000000
mean        6.854788          0.278241     0.334192        6.391415
std         0.843868          0.100795     0.121020        5.072058
min         3.800000          0.080000     0.000000        0.600000
25%         6.300000          0.210000     0.270000        1.700000
50%         6.800000          0.260000     0.320000        5.200000
75%         7.300000          0.320000     0.390000        9.900000
max        14.200000          1.100000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  4898.000000          4898.000000           4898.000000  4898.000000
mean      0.045772            35.308085            138.360657     0.994027
std       0.021848            17.007137             42.498065     0.002991
min       0.009000             2.000000              9.000000     0.987110
25%       0.036000            23.000000            108.000000     0.991723
50%       0.043000            34.000000            134.000000     0.993740
75%       0.050000            46.000000            167.000000     0.996100
max       0.346000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  4898.000000  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267     5.877909
std       0.151001     0.114126     1.230621     0.885639
min       2.720000     0.220000     8.000000     3.000000
```

```
25%       3.090000     0.410000     9.500000     5.000000
50%       3.180000     0.470000    10.400000     6.000000
75%       3.280000     0.550000    11.400000     6.000000
max       3.820000     1.080000    14.200000     9.000000

Shape of the data: (4898, 12)
```

```
X = df.drop('quality', axis=1)
Y = df['quality']
```

```
(4898, 11)
```

```
print("\nShape of X (features):", X.shape)
print("Shape of Y (target variable):", Y.shape)
```

```
Shape of X (features): (4898, 11)
Shape of Y (target variable): (4898,)
```

```

df_train, df_test = train_test_split(df, test_size=0.2, random_state=42)


```

```
lr_model = LinearRegression()
lr_model.fit(df_train.drop('quality', axis=1), df_train['quality'])
```

▾ LinearRegression
LinearRegression()

```
# Polynomial Regression model
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(df_train.drop('quality', axis=1))
poly_model = LinearRegression()
poly_model.fit(X_poly, df_train['quality'])

```

▾ LinearRegression
LinearRegression()

```
def evaluate_model(model, X, y_true):
    y_pred = model.predict(X)
    mse = mean_squared_error(y_true, y_pred)
```

```
    rmse = mean_squared_error(y_true, y_pred, squared=False)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    return mse, rmse, mae, r2
```

```
mse_lr, rmse_lr, mae_lr, r2_lr = evaluate_model(lr_model, df_test.drop('quality', axis=1), df_test['quality'])
print("\nLinear Regression Model Evaluation:")
print("MSE:", mse_lr)
print("RMSE:", rmse_lr)
print("MAE:", mae_lr)
print("R2 Score:", r2_lr)

```

```
Linear Regression Model Evaluation:
MSE: 0.5690247717229278
RMSE: 0.754337306331145
MAE: 0.5862665383250473
R2 Score: 0.2652750042179125
```

```
X_poly_test = poly.transform(df_test.drop('quality', axis=1))
mse_poly, rmse_poly, mae_poly, r2_poly = evaluate_model(poly_model, X_poly_test, df_test['quality'])
print("\nPolynomial Regression Model Evaluation:")
print("MSE:", mse_poly)
print("RMSE:", rmse_poly)
print("MAE:", mae_poly)
print("R2 Score:", r2_poly)
```

```
Polynomial Regression Model Evaluation:
MSE: 0.6193710931852203
RMSE: 0.7870013298497153
MAE: 0.5669837736822492
R2 Score: 0.20026781531816773
```

IRIS DATASET

```
import pandas as pd


column_names = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]
df_iris = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header=None, names=column_names)

print(df_iris.describe())
```

```
print(df_iris.shape)
```

```
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
(150, 5)
```

```
from sklearn.model_selection import train_test_split

X_iris = df_iris.drop('class', axis=1)
Y_iris = df_iris['class']

X_iris_train, X_iris_test, Y_iris_train, Y_iris_test = train_test_split(X_iris, Y_iris, test_size=0.2, random_state=42)

```

```
from sklearn.linear_model import LogisticRegression

lr_iris = LogisticRegression(max_iter=200).fit(X_iris_train, Y_iris_train)
predictions_iris = lr_iris.predict(X_iris_test)

```

```
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, roc_auc_score, roc_curve, auc
import numpy as np

accuracy = accuracy_score(Y_iris_test, predictions_iris)
precision, recall, fscore, _ = precision_recall_fscore_support(Y_iris_test, predictions_iris, average='macro')

from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier

Y_iris_test_bin = label_binarize(Y_iris_test, classes=["Iris-setosa", "Iris-versicolor", "Iris-virginica"])
n_classes = Y_iris_test_bin.shape[1]

classifier = OneVsRestClassifier(LogisticRegression(max_iter=200))
score = classifier.fit(X_iris_train, label_binarize(Y_iris_train, classes=["Iris-setosa", "Iris-versicolor", "Iris-virginica"])).decision_function(X_iris_test)

fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(n_classes):
```

```
    fpr[i], tpr[i], _ = roc_curve(Y_iris_test_bin[:, i], score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

fpr["micro"], tpr["micro"], _ = roc_curve(Y_iris_test_bin.ravel(), score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F-score:", fscore)
print("AUC (micro-averaged):", roc_auc["micro"])
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F-score: 1.0
AUC (micro-averaged): 0.9794444444444446
```