Name : Shreyash Kamat **Div : D15C** **Roll No : 22**

Experiment No. - 5 :

Aim : Perform Regression Analysis using Scipy and Sci-kit learn.

Problem Statement:

a) Perform Logistic regression to find out relation between variables
b) Apply regression model technique to predict the data on the above dataset.

Introduction :

Logistic Regression : Logistic regression is a widely used statistical method for analyzing and modeling relationships between a dependent variable and one or more independent variables. Unlike linear regression, logistic regression is applied when the target variable is categorical, typically binary (0 or 1). It uses the logistic function to estimate probabilities, making it suitable for classification tasks.

In this implementation, we explore the relationship between variables using logistic regression to understand their influence on the target variable. Additionally, we apply this model to predict outcomes based on the dataset, leveraging techniques like model fitting, evaluation metrics, and performance assessment to validate the predictions.

For this dataset , we will be working on an AQI dataset that contains values of various pollutants contributing to the overall AQI value.

Implementation Process :

Step 1 **:** We start with performing pre-processing operations on the dataset by eliminating all of the duplicate values and missing values.

```
[ ]  # Preprocessing the Data
     import pandas as pd

     df = pd.read_csv("city_hour.csv")
     df = df.drop_duplicates()
     df = df.dropna()

     print("Number of rows after cleaning:", len(df))

     df.to_csv("aqidata.csv", index=False)
     df.info()
     df.head(10)
```

```
Number of rows after cleaning: 129277
<class 'pandas.core.frame.DataFrame'>
Index: 129277 entries, 50888 to 707867
Data columns (total 16 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   City        129277 non-null  object
 1   Datetime    129277 non-null  object
 2   PM2.5       129277 non-null  float64
 3   PM10        129277 non-null  float64
 4   NO          129277 non-null  float64
 5   NO2         129277 non-null  float64
 6   NOx         129277 non-null  float64
 7   NH3         129277 non-null  float64
 8   CO          129277 non-null  float64
 9   SO2         129277 non-null  float64
 10  O3          129277 non-null  float64
 11  Benzene     129277 non-null  float64
 12  Toluene     129277 non-null  float64
 13  Xylene      129277 non-null  float64
 14  AQI         129277 non-null  float64
 15  AQI_Bucket  129277 non-null  object
dtypes: float64(13), object(3)
```

Step 2 : The categorical values of the AQI_Bucket column that categorizes the AQI value are converted into binary target variables.

Mapping categories to binary labels:

- 0 for 'Good', 'Satisfactory', 'Moderate' (considered as good/acceptable air quality). ● 1 for 'Poor', 'Very Poor', 'Severe' (considered as bad/unacceptable air quality).

This creates a new column 'AQI_Binary' with binary values (0 or 1) for classification.

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("aqidata.csv")
print(df['AQI_Bucket'].unique())

df['AQI_Binary'] = df['AQI_Bucket'].map({
    'Good': 0,        # Define "Good" as 0
    'Satisfactory': 0,
    'Moderate': 0,
    'Poor': 1,        # Define "Bad" as 1
    'Very Poor': 1,
    'Severe': 1
})

df = df.dropna(subset=['AQI_Binary'])
df.info()
```

```
['Moderate' 'Poor' 'Very Poor' 'Satisfactory' 'Good' 'Severe']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129277 entries, 0 to 129276
Data columns (total 17 columns):
```

Step 3 : The numerical values in the dataset are selected for the purposes of prediction, the dataset is then split into a 70 : 30 ratio, the features selected are standardized and then the logistic regression model is trained.

```
[ ] # Select numerical features for prediction
    df_selected = df[['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene']]

    df_selected = df_selected.dropna()

    # Define X (features) and y (target)
    X = df_selected
    y = df.loc[df_selected.index, 'AQI_Binary']

[ ] # Split into training and testing sets (70% train, 30% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

[ ] # Standardize features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

[ ] # Train logistic regression model
    logreg = LogisticRegression()
    logreg.fit(X_train_scaled, y_train)
```

Step 4 : The model is then used to make predictions for the classification on the basis of AQI into a score of 0 for low AQI and 1 for high and dangerous levels of AQI and compare it with the actual values for the first 5 rows.

```
# Predict using trained model
y_pred = logreg.predict(X_test_scaled)
import pandas as pd
df = pd.DataFrame({"Actual": y_test, "Predicted": y_pred})
print(df.head(5))
```

```
        Actual  Predicted
22160        0          0
115833       0          0
124772       0          0
51509        0          0
5464         0          0
```

Step 5 : The model is then evaluated to check for accuracy along with the confusion matrix.

```
# Print evaluation metrics
print("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))
```

```
Accuracy: 0.9171823432343235
Confusion Matrix:
 [[29531   914]
 [ 2298  6041]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     30445
           1       0.87      0.72      0.79      8339

    accuracy                           0.92     38784
   macro avg       0.90      0.85      0.87     38784
weighted avg       0.92      0.92      0.91     38784
```

Linear Regression : Linear regression is a fundamental statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the input features and the target variable, aiming to find the best-fitting line that minimizes the difference between predicted and actual values. Linear regression is widely used for predictive analysis in fields like finance, healthcare, and social sciences to forecast continuous outcomes like prices, scores, or measurements. The model's simplicity, interpretability, and efficiency make it a popular choice for regression tasks in data science.

For our dataset,we make use of linear regression to make predictions about the AQI values

Step 1 :
We prepare the dataset for applying linear regression to predict the Air Quality Index (AQI) based on various pollutants. It begins by cleaning column names, ensuring the presence of the target variable AQI, and verifying that all required feature columns (like PM2.5, NO2, CO, etc.) are present. Missing values in the target or feature columns are removed to maintain data consistency. The feature matrix (X_reg) and target variable (y_reg) are then finalized for modeling. This setup ensures accurate and reliable predictions in the subsequent regression analysis.

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

df.columns = df.columns.str.strip()

if 'AQI' not in df.columns:
    raise KeyError("Column 'AQI' not found. Check dataset structure.")

y_reg = df['AQI']  # AQI is the target variable
feature_columns = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene']

missing_features = [col for col in feature_columns if col not in df.columns]
if missing_features:
    raise KeyError(f"Missing feature columns: {missing_features}")

X_reg = df[feature_columns]

df_selected = df.dropna(subset=['AQI'] + feature_columns)

X_reg = df_selected[feature_columns]
y_reg = df_selected['AQI']
```

Step 2 : The dataset is split into a 70 : 30 ratio for training and testing.The linear regression model is applied and then used to predict values and compare them with the actual values from the dataset.

```python
[ ]  # Split data into training (70%) and testing (30%) sets
    X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size=0.3, random_state=42)

    scaler_reg = StandardScaler()
    X_train_reg_scaled = scaler_reg.fit_transform(X_train_reg)
    X_test_reg_scaled = scaler_reg.transform(X_test_reg)

    linreg = LinearRegression()
    linreg.fit(X_train_reg_scaled, y_train_reg)

    # Predict AQI values
    y_pred_reg = linreg.predict(X_test_reg_scaled)

    import pandas as pd
    df_results = pd.DataFrame({'Actual AQI': y_test_reg[:10], 'Predicted AQI': y_pred_reg[:10]})
```

	Actual AQI	Predicted AQI
22160	44.0	88.561925
115833	69.0	69.859090
124772	101.0	89.095485
51509	134.0	165.851721
5464	69.0	83.331704
2571	44.0	49.108907
125801	117.0	153.129551
101139	62.0	60.028689
75492	81.0	120.020093
93810	48.0	47.966254

Step 3 : The prepared model is evaluated for its performance on the basis of evaluation metrics like Mean Absolute Error,Mean Squared Error and the R2 Error.

Mean Absolute Error (MAE):
Measures the average absolute difference between actual and predicted values. It is simple and interpretable but doesn't emphasize large errors.

Mean Squared Error (MSE):
Computes the average of squared differences between actual and predicted values. Squaring emphasizes larger errors, making it sensitive to outliers.

R^2 Score (Coefficient of Determination):
Indicates the proportion of variance in the target variable explained by the model. Ranges from 0 to 1 (or negative for poor models). Higher R^2 suggests better fit.

```
[ ]  print("Mean Absolute Error:", mean_absolute_error(y_test_reg, y_pred_reg))
     print("Mean Squared Error:", mean_squared_error(y_test_reg, y_pred_reg))
     print("R² Score:", r2_score(y_test_reg, y_pred_reg))
```

```
Mean Absolute Error: 32.60513742490674
Mean Squared Error: 2270.1515445187156
R² Score: 0.7620454246256327
```

Low metric values in predicting **AQI** using linear regression are likely due to the complex, non-linear nature of air quality data, which a simple linear model may struggle to capture. Additionally, the dataset may lack crucial features like **weather conditions** or **traffic patterns**, leading to incomplete modeling.

Step 4 :

Root Mean Squared Error (RMSE):
The square root of MSE, maintaining the same unit as the target variable. It balances sensitivity to large errors with interpretability.

Baseline MSE:
The MSE is calculated by predicting the mean of the target variable for all inputs. It serves as a benchmark; if a model's MSE is significantly lower than the baseline, it's considered effective.

```
print(df['AQI'].min(), df['AQI'].max())

import numpy as np
rmse = np.sqrt(2270.15)  # Square root of MSE
print("RMSE:", rmse)
print(df['AQI'].std())  # Standard deviation of AQI

y_baseline = df['AQI'].mean()  # Predicting the mean AQI for all values
mse_baseline = np.mean((df['AQI'] - y_baseline) ** 2)
print("Baseline MSE:", mse_baseline)
```

```
18.0 760.0
RMSE: 47.64609113033303
97.01102086065393
Baseline MSE: 9411.065370185535
```

The minimum and maximum AQI values (18.0 and 760.0) show a wide range of air quality data. The Root Mean Squared Error (RMSE) of 47.65 is relatively small compared to the AQI range, suggesting that while the model makes errors, they are not extreme. However, the standard deviation of 97.01 indicates significant variability in AQI, implying that the model might struggle with accurately predicting all values. The Baseline MSE of 9411.07, based on predicting the mean AQI, is substantially higher than the model's MSE of 2270.15, indicating that the linear regression model does perform better than a naive prediction. Despite this, the high variability and potential data complexities suggest that a more advanced model may better capture the relationships within the dataset.

Conclusion :

In this analysis, we applied both **logistic regression** and **linear regression** to understand the relationships within the air quality dataset and make predictions. Logistic regression was effective in classifying air quality as **"Good"** or **"Bad"** based on pollutant levels, demonstrating the model's capability in binary classification tasks. Linear regression aimed to predict the **AQI** numerically but faced challenges due to the data's high variability and complex non-linear patterns. Despite achieving better-than-baseline performance, the model's low **R^2 score** and relatively high **error metrics** suggest that more advanced techniques or additional features may be necessary for improved predictions. The analysis showcased the practical applications of regression techniques in environmental data analysis while highlighting their limitations.