Name: Shivam.R.Prajapati **Div: D15C** **Roll No: 41**

Experiment No: 5

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

THEORY:

1] Regression Analysis:

Regression analysis is **a statistical method used to understand the relationship between one dependent variable and one or more independent variables**. It **helps in predicting outcomes and identifying trends by analyzing past data.** For example, a business can use regression to predict future sales based on factors like advertising spend and customer reviews

2] Regression Model:

Regression Model for Prediction involves **training a model on historical data** to **identify patterns and relationships between variables**. The **trained model** is then used to **predict outcomes for new data**. Depending on the dataset, different regression techniques (such as logistic or linear regression) are applied to achieve accurate predictions.

3] Types of Regression Analysis:

- **Linear Regression** – The simplest form, where a straight line shows the relationship between one dependent and one independent variable.
- **Multiple Regression** – Similar to linear regression but involves multiple independent variables affecting the dependent variable.
- **Polynomial Regression** – Fits data into a curved line rather than a straight one, useful when relationships are non-linear.
- **Logistic Regression** – Used for classification problems where the output is binary (e.g., yes/no, pass/fail).
- **Ridge Regression** – A type of linear regression that prevents overfitting by adding a penalty to large coefficients.
- **Lasso Regression** – Similar to ridge regression but can shrink some coefficients to zero, helping with feature selection.

- **Stepwise Regression** – Automatically selects the most significant variables by adding or removing predictors step by step.
- **Elastic Net Regression** – A combination of ridge and lasso regression, useful when there are many correlated variables.
- **Quantile Regression** – Focuses on estimating medians or other percentiles instead of the mean, useful when data has outliers.
- **Poisson Regression** – Used for count data, such as predicting the number of customer visits in a store.

4] Logistic Regression:

Logistic Regression is a statistical method used for binary classification problems, **where the outcome is either 0 or 1** (e.g., success/failure, yes/no). It **estimates the probability of an event occurring based on independent variables using the sigmoid function**. Logistic Regression **uses the sigmoid function** (also called the logistic function) **to model the relationship between the independent variables and the probability of a binary outcome.**

The **formula** for Logistic Regression is:

$$P(Y=1) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_n X_n)}}$$

Where:

- **P(Y=1) i**s the probability that the output is 1 (positive class).
- **b0** is the intercept (bias term).
- **b1,b2,...,bn** are the coefficients of the independent variables **X1,X2,...,Xn.**
- **e** is Euler's number (approximately 2.718).

DATASET:

The **dataset contains information about electric vehicles**, including details such as VIN, county, city, state, model year, make, and model. It also includes attributes like electric vehicle type, eligibility for clean alternative fuel programs, electric range, base MSRP, legislative district, vehicle location, electric utility provider, and census tract. This dataset is useful for analyzing the distribution, characteristics, and adoption of electric vehicles across different regions.

Name: Shivam.R.Prajapati **Div: D15C** **Roll No: 41**

STEPS:

1] Load Dataset into the Google Colab and import necessary libraries

Code:

```
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.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

df = pd.read_csv('Electric_Vehicle_Population_Data.csv')

print(df.head())

print(df.info())
```

Output:

```
   VIN (1-10)    County      City State  Postal Code  Model Year    Make  \
0  2T3YL4DV0E      King  Bellevue    WA      98005.0        2014  TOYOTA
1  5YJ3E1EB6K      King   Bothell    WA      98011.0        2019   TESLA
2  5UX43EU02S  Thurston   Olympia    WA      98502.0        2025     BMW
3  JTMAB3FV5R  Thurston   Olympia    WA      98513.0        2024  TOYOTA
4  5YJYGDEE8M    Yakima     Selah    WA      98942.0        2021   TESLA

        Model                   Electric Vehicle Type  \
0        RAV4          Battery Electric Vehicle (BEV)
1     MODEL 3          Battery Electric Vehicle (BEV)
2          X5  Plug-in Hybrid Electric Vehicle (PHEV)
3  RAV4 PRIME  Plug-in Hybrid Electric Vehicle (PHEV)
4     MODEL Y          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric Range  \
0            Clean Alternative Fuel Vehicle Eligible           103.0
1            Clean Alternative Fuel Vehicle Eligible           220.0
2            Clean Alternative Fuel Vehicle Eligible            40.0
3            Clean Alternative Fuel Vehicle Eligible            42.0
4  Eligibility unknown as battery range has not b...             0.0
```

```
   2020 Census Tract
0       5.303302e+10
1       5.303302e+10
2       5.306701e+10
3       5.306701e+10
4       5.307700e+10
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232230 entries, 0 to 232229
Data columns (total 17 columns):
 #   Column                                             Non-Null Count   Dtype  
---  ------                                             --------------   -----  
 0   VIN (1-10)                                         232230 non-null  object 
 1   County                                             232226 non-null  object 
 2   City                                               232226 non-null  object 
 3   State                                              232230 non-null  object 
 4   Postal Code                                        232226 non-null  float64
 5   Model Year                                         232230 non-null  int64  
 6   Make                                               232230 non-null  object 
 7   Model                                              232230 non-null  object 
 8   Electric Vehicle Type                              232230 non-null  object 
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  232230 non-null  object 
 10  Electric Range                                     232203 non-null  float64
 11  Base MSRP                                          232203 non-null  float64
 12  Legislative District                               231749 non-null  float64
 13  DOL Vehicle ID                                     232230 non-null  int64  
 14  Vehicle Location                                   232219 non-null  object 
 15  Electric Utility                                   232226 non-null  object 
 16  2020 Census Tract                                  232226 non-null  float64
dtypes: float64(5), int64(2), object(10)
memory usage: 30.1+ MB
None
```

The output of **df.head() displays the first five rows of the dataset**, showing key details such as VIN, county, city, state, model year, make, model, electric vehicle type, eligibility for clean fuel programs, electric range, base MSRP, legislative district, DOL vehicle ID, vehicle location, electric utility, and census tract.

The output of **df.info()** provides an overview of the dataset structure, showing that it contains **232,320 entries** and **17 columns**. It l**ists column names, data types (object, float64, int64), and the number of non-null values for each column**, indicating that most columns have complete data except for a few missing values in Base MSRP, Legislative District, DOL Vehicle ID, and Vehicle Location. The dataset's memory usage is around **30.1 MB**.

2] Perform Logistic regression to find out relation between variables

Step 1: Select Target Column ("Electric Vehicle Type")

Code:

```
df['Electric Vehicle Type'].unique()

df['EV_Type_Binary'] = df['Electric Vehicle Type'].map({

'Battery Electric Vehicle (BEV)': 0,

'Plug-in Hybrid Electric Vehicle (PHEV)': 1

})
```

The command **df['Electric Vehicle Type'].unique() retrieves unique values in the "Electric Vehicle Type" column**, likely including "Battery Electric Vehicle (BEV)" and "Plug-in Hybrid Electric Vehicle (PHEV)". **The next command creates a new column**, "**EV_Type_Binary**", **mapping "Battery Electric Vehicle (BEV)" to 0 and "Plug-in Hybrid Electric Vehicle (PHEV)" to 1, converting categorical data into numerical form** for machine learning models like Logistic Regression.

Step 2: Select Features (X) and Target (y)

Code:

```
df_selected = df[['Model Year', 'Electric Range', 'Base MSRP', 'Legislative District']]

df_selected = df_selected.dropna()

X = df_selected

y = df.loc[df_selected.index, 'EV_Type_Binary']
```

The **code selects specific columns** ("Model Year", "Electric Range", "Base MSRP", and "Legislative District") **from the dataset and stores them in df_selected**. Then, it **removes any rows with missing values** using **dropna()**. The **variable X** is assigned the cleaned dataset containing the selected features, while **y** is assigned the corresponding "EV_Type_Binary" values from the original dataset, ensuring both X and y have matching indices for model training.

Step 3: Train-Test Split

Code:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

random_state=42)
```

The **code splits the dataset into training and testing sets** using **train_test_split()** from scikit-learn. Here, **30% of the data is allocated for testing** (X_test, y_test), **while 70% is used for training** (X_train, y_train). The **parameter random_state=42 ensures reproducibility** by keeping the split consistent across different runs.

Step 4: Normalize the Features

Code:

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

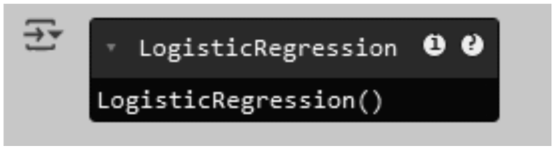
The **code** uses **StandardScaler()** from scikit-learn to **standardize the features in X_train and X_test**. It first fits the scalar to X_train and transforms it, ensuring the data has a mean of 0 and a standard deviation of 1. Then, the same transformation is applied to X_test using the previously computed scaling parameters. This helps improve the performance of machine learning models by normalizing the feature values.

Step 5: Train Logistic Regression Model

Code:

```
logreg = LogisticRegression()

logreg.fit(X_train_scaled, y_train)
```

Output:



The code initializes a **LogisticRegression()** model and trains it using the **fit()** method with the standardized training data (X_train_scaled) and corresponding labels (y_train). This allows the logistic regression model to learn the relationship between the input features and the target variable (EV_Type_Binary), enabling it to make predictions on new data.

Step 6: Make Predictions

Code:

```
y_pred = logreg.predict(X_test_scaled)
```

The code uses the **trained LogisticRegression model** to **predict the target variable** for the test dataset. The **predict() function takes the standardized test data (X_test_scaled) and generates y_pred**, which contains the predicted binary values (0 or 1) for the "EV_Type_Binary" classification. **These predictions can later be compared with actual values (y_test) to evaluate the model's performance**.

Step 7: Evaluate the Model

Code:

```
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))
```

Output:

```
Accuracy: 0.803055367751773
Confusion Matrix:
 [[54243   841]
 [12850  1583]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.98      0.89     55084
           1       0.65      0.11      0.19     14433

    accuracy                           0.80     69517
   macro avg       0.73      0.55      0.54     69517
weighted avg       0.78      0.80      0.74     69517
```

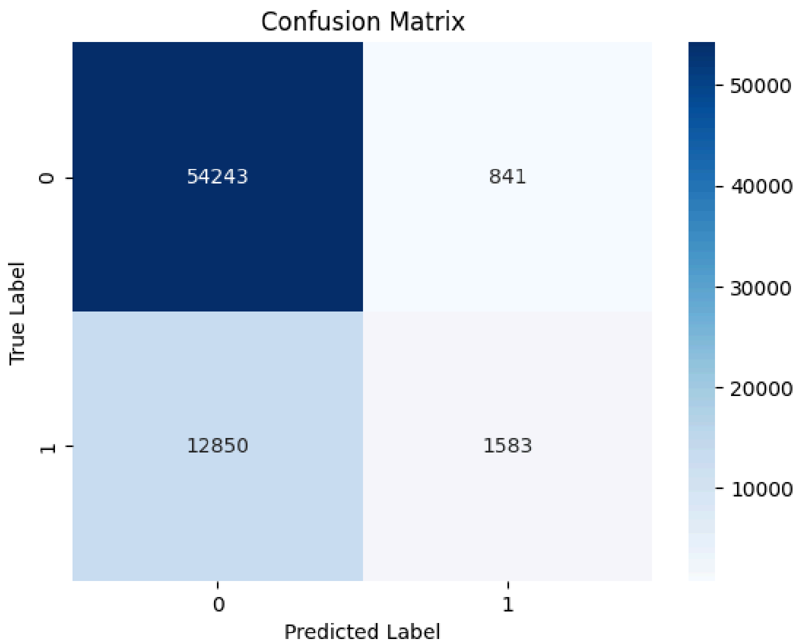
The output shows the model's performance metrics. The **accuracy is 80.3%,** indicating that the logistic regression model correctly predicts the electric vehicle type in most cases. The **confusion matrix shows the number of correct and incorrect predictions for each class**. The classification report provides precision, recall, and F1-score for both classes. Class 0 (BEV) has high precision (0.81) and recall (0.98), meaning it is well predicted. However, Class 1 (PHEV) has a low recall (0.11), indicating many false negatives. The weighted average F1-score is 0.74, summarizing the overall model performance.

Step 8: Visualize Confusion Matrix for better understanding:

Code:

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

plt.show()
```

Output:



The **confusion matrix heatmap visually represents the model's classification performance**. The **dark blue cell (top-left) indicates 54,243 correct predictions** for **class 0 (BEV)**, while the **light blue cell (bottom-right) shows 1,583 correct predictions** for **class 1 (PHEV)**. The **841 false positives (top-right) mean BEVs were misclassified as PHEVs**, whereas the 12,850 false negatives (bottom-left) indicate a significant number of PHEVs were incorrectly classified as BEVs. The imbalance in false negatives suggests the model struggles to correctly identify PHEVs

3] Apply regression model technique to predict the data on the above dataset.

Step 1: Change Target Variable (y) to "Electric Range"

Code:

```
y_reg = df_selected['Electric Range']

X_reg = df_selected.drop(['Electric Range'], axis=1)
```

y_reg is assigned the 'Electric Range' column, which will be the target variable for regression. Meanwhile, **X_reg** contains the remaining features ('Model Year', 'Base MSRP', and 'Legislative District') after dropping 'Electric Range', meaning these will be the independent variables used to predict the electric range of vehicles. This setup prepares the dataset for regression analysis.

Step 2: Train a Linear Regression Model

Code:

```
from sklearn.linear_model import LinearRegression

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)

y_pred_reg = linreg.predict(X_test_reg_scaled)
```

In this process, a **Linear Regression model is trained to predict the electric range of vehicles based on features like Model Year, Base MSRP, and Legislative District.** The **dataset is first split into training (70%) and testing (30%) sets**. Then, **feature scaling is applied using StandardScaler to normalize the data**. After scaling, the LinearRegression model is fitted to the training data, and predictions (y_pred_reg) are generated on the test set. This setup allows us to evaluate how well the model predicts the electric range of vehicles.

Step 3: Evaluate **Regression Model**

Code:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

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))
```

Output:

```
Mean Absolute Error: 48.33783743198793
Mean Squared Error: 5150.480320071757
R² Score: 0.2767198568472038
```

The Linear Regression model shows a **Mean Absolute Error (MAE) of ~48.34,** meaning **predictions deviate by around 48 miles on average**. The **Mean Squared Error (MSE) of ~5150.48** suggests **larger errors are present.** The **R^2 Score of 0.28** indicates the **model explains only 27.67% of the variance in electric range**, implying a weak fit and suggesting room for improvement, possibly by including more relevant features.

CONCLUSION:

In this experiment, we analyzed electric vehicle (EV) data to classify vehicle types and predict electric range using Logistic Regression and Linear Regression models.First, we explored the dataset, cleaned it, and selected relevant features like Model Year, Electric Range, Base MSRP, and Legislative District. We then created a new binary column to classify EVs into Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs).

For classification, we trained a Logistic Regression model to predict EV type. The model achieved 80.3% accuracy, with better performance in identifying BEVs than PHEVs. The confusion matrix showed a significant number of false negatives for PHEVs, indicating the model struggled to classify them correctly.

For regression, we trained a Linear Regression model to predict the electric range of vehicles. The results showed an MAE of 48.34, meaning predictions were off by about 48 miles on average. The R^2 score of 0.28 indicates that our model explains only 27.67% of the variation in electric range, suggesting it is not very accurate and could be improved by adding more relevant features.