Name: Anish Kulkarni Roll No.: 29 Class: D15C AY: 2024-25

Experiment 5

Aim: Perform Regression Analysis using Scipy and Sci-kit learn.
a) Perform Logistic regression to find out relation between variables
b) Apply regression model technique to predict the data on the above dataset.

Performance:

- Prerequisite: Import essential libraries: pandas for data manipulation, numpy for numerical computations, matplotlib.pyplot and seaborn for data visualization, sklearn.model_selection for dataset splitting, sklearn.preprocessing for data scaling, sklearn.linear_model for logistic regression, and sklearn.metrics for model evaluation. Next, load the Electric Vehicle Population Dataset into a Pandas DataFrame using pd.read_csv(). Finally, explore the dataset by displaying the first few rows with df.head() and checking column names, data types, and missing values using df.info():

Command:
```
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())
```

```
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

   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  41.0       186450183
1        0.0                   1.0       478093654
2        0.0                  35.0       274800718
3        0.0                   2.0       260758165
4        0.0                  15.0       236581355

              Vehicle Location                               Electric Utility  \
0   POINT (-122.1621 47.64441)  PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
1  POINT (-122.20563 47.76144)  PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
2  POINT (-122.92333 47.03779)                         PUGET SOUND ENERGY INC
3  POINT (-122.81754 46.98876)                         PUGET SOUND ENERGY INC
4  POINT (-120.53145 46.65405)                                     PACIFICORP
```

```
   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
```

a) Perform Logistic regression to find out relation between variables:

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

```
array(['Battery Electric Vehicle (BEV)',
       'Plug-in Hybrid Electric Vehicle (PHEV)'], dtype=object)
```

First, df['Electric Vehicle Type'].unique() retrieves and displays the unique values in the Electric Vehicle Type column, helping to identify the different categories present in the dataset. Then, a new binary column, EV_Type_Binary, is created by mapping Battery Electric Vehicles (BEV) to 0 and Plug-in Hybrid Electric Vehicles (PHEV) to 1 using the map() function. This transformation converts categorical data into a numerical format, making it suitable for machine learning models.

Step 2: Select Features (X) and Target (y):-
Command: 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']
This step selects specific numerical columns—Model Year, Electric Range, Base MSRP, and Legislative District—from the dataset and stores them in df_selected. It then removes any rows with missing values using dropna() to ensure the data is clean for modeling. The feature matrix X is assigned the cleaned df_selected, while the target variable y is extracted from the EV_Type_Binary column, ensuring that both X and y have matching indices. This prepares the dataset for training a machine learning model.

Step 3: Train-Test Split:-
Command: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
This step splits the dataset into training and testing sets using the train_test_split() function. The feature matrix X and target variable y are divided into X_train, X_test, y_train, and y_test, where 30% of the data is allocated for testing (test_size=0.3), and 70% for training. Setting random_state=42 ensures reproducibility by making the split consistent across different runs. This step is essential for evaluating the model's performance on unseen data.

Step 4: Normalize the Features:-
Command: scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
This step standardizes the feature values using StandardScaler. First, an instance of StandardScaler is created. Then, fit_transform(X_train) computes the mean and standard deviation from the training data and scales it, ensuring all features have a mean of 0 and a standard deviation of 1. The same transformation is applied to X_test using transform(X_test), maintaining consistency. Standardization improves model performance by preventing features with larger ranges from dominating those with smaller ones.

Step 5: Train Logistic Regression Model:-
Command: logreg = LogisticRegression()
logreg.fit(X_train_scaled, y_train)

```
LogisticRegression
LogisticRegression()
```

This step initializes a Logistic Regression model using LogisticRegression(). The model is then trained on the standardized training data using fit(X_train_scaled, y_train), where it learns the relationship between the features and the target variable. This trained model can later be used to make predictions on new data.

Step 6: Make Predictions:-
Command: y_pred = logreg.predict(X_test_scaled)
This step uses the trained Logistic Regression model to make predictions on the standardized test data. The predict(X_test_scaled) function generates predicted labels (y_pred) based on the learned relationships from the training phase. These predictions will be compared with actual values to evaluate the model's performance.

Step 7: Evaluate the Model:-
Command: 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))

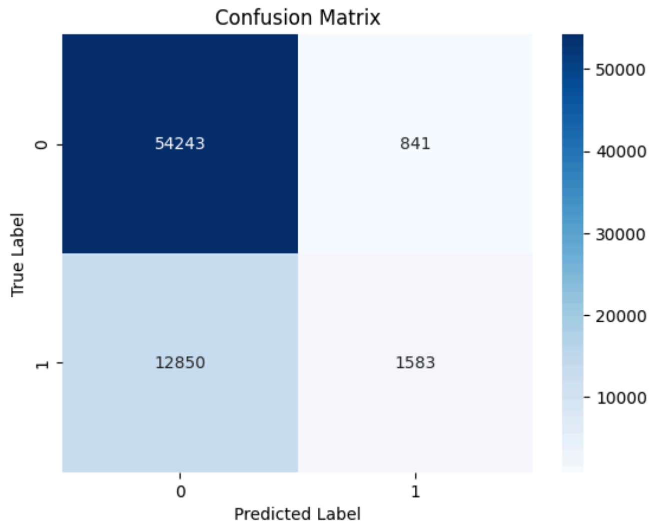
```
Accuracy: 0.803055367751773
Confusion Matrix:
 [[54243   841]
 [12850  1583]]
Classification Report:
               precision    recall  f1-score   support

           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
```

This step evaluates the Logistic Regression model's performance by calculating key metrics. The accuracy score measures the percentage of correct predictions, while the confusion matrix displays the distribution of true positives, true negatives, false positives, and false negatives. Finally, the classification report provides detailed metrics such as precision, recall, and F1-score for each class, offering insights into the model's effectiveness.

Step 8: Visualize Confusion Matrix for better understanding:-
Command: 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()



This step visualizes the confusion matrix using Seaborn's heatmap() function. The confusion matrix is plotted with numerical values (annot=True) in a blue color scheme (cmap='Blues'). Labels for the x-axis (Predicted Label) and y-axis (True Label) are added for clarity, along with a title (Confusion Matrix). Finally, plt.show() displays the heatmap, making it easier to interpret the model's classification performance.

b) Apply regression model technique to predict the data on the above dataset:

Step 1: Change Target Variable (y) to "Electric Range":-
Command: y_reg = df_selected['Electric Range']
X_reg = df_selected.drop(['Electric Range'], axis=1)
This step prepares the dataset for regression analysis by defining the target variable (y_reg) and feature matrix (X_reg). The Electric Range column is selected as the target variable (y_reg) since the goal is to predict it. The remaining columns in df_selected are assigned to X_reg by dropping Electric Range using drop(axis=1), ensuring that only independent variables are used as input features for the regression model.

Step 2: Train a Linear Regression Model:-

Command:
```
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)
```

This step applies Linear Regression to predict Electric Range using the selected features. The dataset is split into training (70%) and testing (30%) sets, ensuring reproducibility with random_state=42. The features are standardized using StandardScaler() to prevent numerical imbalances. A Linear Regression model is then initialized and trained on the scaled training data with fit(). Finally, predictions are made on the test set using predict(), generating y_pred_reg, which contains the predicted electric range values.

Step 3: Evaluate Regression Model:-

Command:
```
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))
```

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

This step evaluates the Linear Regression model's performance using three key metrics. Mean Absolute Error (MAE) measures the average absolute difference between actual and predicted values, while Mean Squared Error (MSE) penalizes larger errors more heavily. The R^2 Score indicates how well the model explains the variance in Electric Range, with values closer to 1 signifying better performance. These metrics provide insights into the model’s accuracy and effectiveness.

Conclusion:

1. In this experiment, we learned to perform Regression Analysis using Scipy and Sci-kit learn.
2. The dataset was preprocessed by handling missing values, encoding categorical variables, and selecting relevant numerical features.
3. A Logistic Regression model was trained to classify electric vehicle types based on selected features.
4. The classification model achieved an accuracy of 80.3%, indicating a relatively strong ability to distinguish between Battery Electric Vehicles (BEVs) and Plug-in Hybrid Electric Vehicles (PHEVs).

5. The confusion matrix showed that the model correctly classified 54,243 BEVs but misclassified 841 as PHEVs, while it correctly identified only 1,583 PHEVs and misclassified 12,850 as BEVs.
6. The classification report revealed high precision (81%) and recall (98%) for BEVs, but low recall (11%) for PHEVs, indicating the model struggles to correctly identify PHEVs.
7. A Linear Regression model was applied to predict Electric Range using features like Model Year, Base MSRP, and Legislative District.
8. The regression model resulted in a Mean Absolute Error (MAE) of 48.34 and a Mean Squared Error (MSE) of 5150.48, indicating notable prediction variations.
9. The R^2 score of 0.277 suggests that the regression model explains only 27.7% of the variance in Electric Range, meaning other influential factors are missing from the dataset.
10. The classification model performed well overall but had difficulty identifying PHEVs, while the regression model had limited predictive power for Electric Range.