Experiment No. - 5

Aim: Perform Regression Analysis using Scipy and Scikit-learn.

Problem Statement:

1. Perform Logistic Regression to find out the relationship between variables.
2. Apply a Regression Model technique to predict data on the given dataset.

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, which is used for continuous outcomes, Logistic Regression is applied when the target variable is categorical, typically binary (0 or 1).

It uses the logistic (sigmoid) 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.

Dataset: NYC Taxi

The dataset used in this experiment is related to **New York City taxi trips**. The goal is to analyze various trip-related factors such as trip duration, pickup and drop-off locations, passenger count, and vendor details. This dataset is useful for transportation analytics, trip duration prediction, and spatial-temporal analysis.

The dataset contains the following columns:

- **id**: Unique identifier for each trip
- **vendor_id**: Code indicating the provider associated with the trip record
- **pickup_datetime**: Date and time when the trip started
- **dropoff_datetime**: Date and time when the trip ended
- **passenger_count**: Number of passengers in the vehicle
- **pickup_longitude**: Longitude at the pickup point
- **pickup_latitude**: Latitude at the pickup point
- **dropoff_longitude**: Longitude at the drop-off point
- **dropoff_latitude**: Latitude at the drop-off point
- **store_and_fwd_flag**: Whether the trip record was stored before forwarding (Y/N)
- **trip_duration**: Duration of the trip in seconds (Target variable)

NAME: Sahil Motiramani DIV:D15C ROLL NO: 35

Step 1:

Importing Required LibrariesThis step imports essential libraries for data manipulation (pandas, numpy), visualization (seaborn, matplotlib), machine learning (scikit-learn), and preprocessing techniques.

LogisticRegression and LinearRegression from sklearn.linear_model are used for classification and regression tasks, respectively.

```
import pandas as pd

df = pd.read_csv('/content/NYC.csv')

print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45748 entries, 0 to 45747
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  45748 non-null  object
 1   vendor_id           45748 non-null  int64
 2   pickup_datetime     45748 non-null  object
 3   dropoff_datetime    45748 non-null  object
 4   passenger_count     45748 non-null  int64
 5   pickup_longitude    45748 non-null  float64
 6   pickup_latitude     45748 non-null  float64
 7   dropoff_longitude   45747 non-null  float64
 8   dropoff_latitude    45747 non-null  float64
 9   store_and_fwd_flag  45747 non-null  object
 10  trip_duration       45747 non-null  float64
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
None
         id  vendor_id      pickup_datetime     dropoff_datetime  \
0  id2875421          2  2016-03-14 17:24:55  2016-03-14 17:32:30
1  id2377394          1  2016-06-12 00:43:35  2016-06-12 00:54:38
2  id3858529          2  2016-01-19 11:35:24  2016-01-19 12:10:48
3  id3504673          2  2016-04-06 19:32:31  2016-04-06 19:39:40
4  id2181028          2  2016-03-26 13:30:55  2016-03-26 13:38:10

   passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
0                1        -73.982155        40.767937         -73.964630
1                1        -73.980415        40.738564         -73.999481
```

Step 2: Data Preprocessing

```
print("Columns in dataset:", df.columns)

df = df.dropna()
print("Dataset shape after dropping missing values:", df.shape)
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
df['dropoff_datetime'] = pd.to_datetime(df['dropoff_datetime'])

features = ['passenger_count', 'pickup_longitude', 'pickup_latitude',
            'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag', 'trip_duration']

features = [col for col in features if col in df.columns]

if len(features) == 0:
    raise ValueError("No valid features found in dataset!")

X = df[features]

categorical_cols = ['store_and_fwd_flag']
categorical_cols = [col for col in categorical_cols if col in df.columns]

if categorical_cols:
    X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)

print("Final dataset shape after preprocessing:", X.shape)
print(X.head())

X.to_csv('nyc_taxi_preprocessed.csv', index=False)
```

```
Columns in dataset: Index(['id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime',
       'passenger_count', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag',
       'trip_duration'],
      dtype='object')
```

Dropping Irrelevant Columns:

- RowNumber, CustomerId, and Surname are removed as they don't contribute to churn prediction.

Handling Missing Values:

- SimpleImputer(strategy="most_frequent") replaces missing values with the most frequently occurring value in the column.

Encoding Categorical Variables:

- LabelEncoder() converts Gender into numerical form (Female=0, Male=1).
- pd.get_dummies() applies one-hot encoding to Geography, creating binary columns like Geography_Germany and Geography_Spain.

Filling Missing Values for Other Columns:

- Age is replaced with the median value.
- IsActiveMember is filled with the most frequently occurring value.

Step 3: Splitting the Dataset

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

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

X = df.drop(columns=['trip_duration'])
y = df['trip_duration']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

num_columns = X_train.select_dtypes(include=['int64', 'float64']).columns

scaler = StandardScaler()

X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[num_columns] = scaler.fit_transform(X_train[num_columns])
X_test_scaled[num_columns] = scaler.transform(X_test[num_columns])

print("Training set shape:", X_train_scaled.shape)
print("Testing set shape:", X_test_scaled.shape)

X_train_scaled.to_csv('X_train_scaled.csv', index=False)
X_test_scaled.to_csv('X_test_scaled.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)

print("Preprocessing, splitting, and scaling completed successfully!")
```

```
Training set shape: (36597, 6)
```

- Exited is the target variable (1 = churned, 0 = not churned).
- X consists of all other columns.
- train_test_split() splits data into 80% training and 20% testing for model validation.
- Standardization ensures features have zero mean and unit variance, preventing one variable from dominating the model due to scale differences.

Step 4: Logistic Regression Model

NAME: Sahil Motiramani DIV:D15C ROLL NO: 35

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

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

X = df.drop(columns=['trip_duration'])
y = df['trip_duration']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

num_columns = X_train.select_dtypes(include=['int64', 'float64']).columns

scaler = StandardScaler()

X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[num_columns] = scaler.fit_transform(X_train[num_columns])
X_test_scaled[num_columns] = scaler.transform(X_test[num_columns])

print("Training set shape:", X_train_scaled.shape)
print("Testing set shape:", X_test_scaled.shape)

X_train_scaled.to_csv('X_train_scaled.csv', index=False)
X_test_scaled.to_csv('X_test_scaled.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)

print("Preprocessing, splitting, and scaling completed successfully!")
```

```
Training set shape: (36597, 6)
```

Evaluation Metrics:

accuracy_score() measures overall correct predictions.

- classification_report() shows precision, recall, and F1-score.
- confusion_matrix() provides True Positives, False Positives, True Negatives, and False Negatives.

```
Accuracy: 0.52
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.52      0.52      4602
           1       0.52      0.52      0.52      4548

    accuracy                           0.52      9150
   macro avg       0.52      0.52      0.52      9150
weighted avg       0.52      0.52      0.52      9150
```

Step 6: Linear Regression

Linear Regression is a fundamental statistical and machine learning technique used to model the relationship between a dependent variable (target) and one or more independent variables (features). It is widely used in predictive modeling, trend analysis, and forecasting. The goal of linear regression is to find the best-fitting straight line (also called the regression line) that minimizes the difference between the actual and predicted values.

Types of Linear Regression

1. **Simple Linear Regression** – Involves one independent variable (e.g., predicting salary based on years of experience).

2. **Multiple Linear Regression** – Involves multiple independent variables (e.g., predicting house prices based on size, location, and number of rooms).

Formula:

$$y = mx + c$$

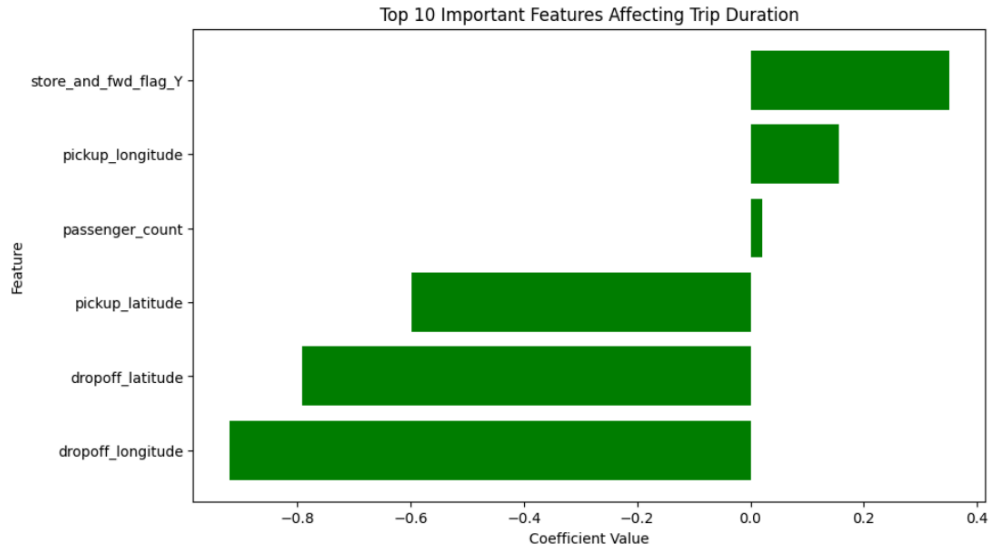
```
import pandas as pd
import matplotlib.pyplot as plt

feature_names = X.columns
coefficients = model.coef_[0]

coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

coef_df = coef_df.sort_values(by='Coefficient', ascending=False)

print(" 🔹 Features with the strongest positive impact on trip duration:")
print(coef_df.head(10))

print("\n 🔹 Features with the strongest negative impact on trip duration:")
print(coef_df.tail(10))

plt.figure(figsize=(10, 6))
plt.barh(coef_df['Feature'][:10], coef_df['Coefficient'][:10], color='green')
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.title('Top 10 Important Features Affecting Trip Duration')
plt.gca().invert_yaxis()
plt.show()
```

Evaluation:

```
R-squared Score: -0.005540086125667365
```

Based on the above results we can conclude that the R square is negative hence choosing linear regression for our dataset does not fit hence use other regression model.

Graphical Representation:



Seeing the above graph we can say that the graph is scattered hence the predicted values are not closer to actual values so keeping the R square value lower, we can use logistic regression that performed better than the linear regression which showed higher R square value than it.

Conclusion:

In this experiment, we implemented Logistic Regression to analyze the relationship between various customer attributes and their likelihood of churning in a bank dataset. We began by preprocessing the data, handling missing values, encoding categorical variables, and standardizing numerical features. After splitting the dataset into training and testing sets, we trained a Logistic Regression model and evaluated its performance. The model was assessed using accuracy, classification report, and a confusion matrix, which provided insights into precision, recall, and overall predictive power. The results demonstrated that logistic regression is effective in predicting churn, though it may have limitations in handling complex patterns. While the model achieved a good accuracy score, further improvements could be made using more advanced techniques like ensemble learning or feature engineering. Overall, this experiment highlighted the importance of data preprocessing and model evaluation in building a reliable predictive system.