NAME: Sahil Motiramani DIV:D15C ROLL NO: 35

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: Bank Churn Modelling

The dataset used in this experiment is related to bank churn prediction, where the goal is to analyze factors affecting whether a customer will churn (leave the bank) or not. The dataset contains variables such as:

- Customer ID (Unique Identifier)
- Credit Score
- Geography
- Gender
- Age
- Tenure
- Balance
- Number of Products
- Has Credit Card
- Is Active Member
- Estimated Salary
- Exited (Target variable: 1 if customer churned, 0 otherwise)

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
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
# Load Dataset (Modify path if needed)
file_path = "Churn_Modelling.csv"
df = pd.read_csv(file_path)

# Display first few rows
print("Dataset Preview:")
df.head()
```

```
Dataset Preview:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1.0	1.0	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41.0	1	83807.86	1	0.0	1.0	112542.58	0
2	3	15619304	Onio	502	France	Female	42.0	8	159660.80	3	1.0	0.0	113931.57	1
3	4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0.0	0.0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43.0	2	125510.82	1	NaN	1.0	79084.10	0

Step 2: Data Preprocessing

```
# Drop irrelevant columns
df.drop(columns=["RowNumber", "CustomerId", "Surname"], inplace=True)

# Handle missing values
imputer = SimpleImputer(strategy="most_frequent")
df["HasCrCard"] = imputer.fit_transform(df[["HasCrCard"]])

# Encode categorical variables
df["Gender"] = LabelEncoder().fit_transform(df["Gender"])  # Female=0, Male=1
df = pd.get_dummies(df, columns=["Geography"], drop_first=True)  # One-hot encoding for Geography

# Fill any remaining missing values
df["Age"].fillna(df["Age"].median(), inplace=True)
df["IsActiveMember"].fillna(df["IsActiveMember"].mode()[0], inplace=True)
```

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

```
# Separate features and target variable
X = df.drop(columns=["Exited"])
y = df["Exited"]

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Standardize numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- 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.

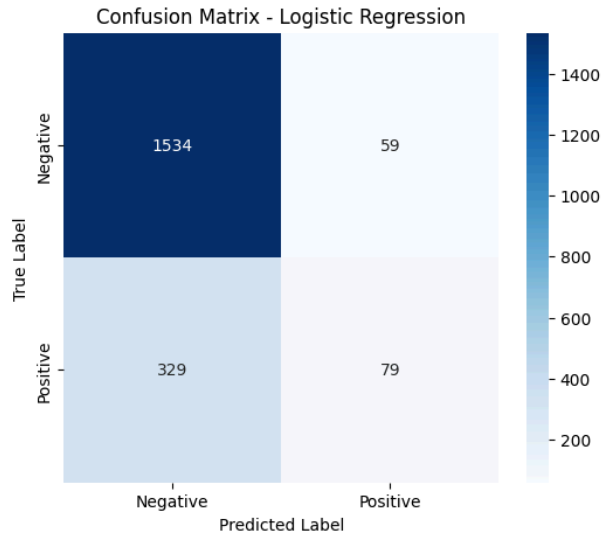
Step 4: Logistic Regression Model

```
# Train Logistic Regression Model
log_model = LogisticRegression(random_state=42)
log_model.fit(X_train, y_train)

# Make predictions
y_pred_log = log_model.predict(X_test)

# Evaluate Logistic Regression
accuracy_log = accuracy_score(y_test, y_pred_log)
print("\nLogistic Regression Accuracy:", accuracy_log)
print("\nClassification Report:\n", classification_report(y_test, y_pred_log))
cm = confusion_matrix(y_test, y_pred_log)

# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```

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.

```
Logistic Regression Accuracy: 0.8060969515242379

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.96      0.89      1593
           1       0.57      0.19      0.29       408

    accuracy                           0.81      2001
   macro avg       0.70      0.58      0.59      2001
weighted avg       0.77      0.81      0.77      2001
```



Evaluation

```python
from sklearn.metrics import r2_score

# Train Linear Regression Model
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)

# Predict using Linear Regression
y_pred_linear = lin_model.predict(X_test)

# Calculate R-squared Score
r2 = r2_score(y_test, y_pred_linear)
print(f"R-squared Score: {r2:.4f}")
```

```
R-squared Score: 0.1492
```

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred_linear)
print(f"Mean Absolute Error (MAE): {mae}")

# Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_linear)
print(f"Mean Squared Error (MSE): {mse}")

# Root Mean Square Error (RMSE)
rmse = np.sqrt(mse)
print(f"Root Mean Square Error (RMSE): {rmse}")
```

```
Mean Absolute Error (MAE): 0.28625508254666526
Mean Squared Error (MSE): 0.13809721804330075
Root Mean Square Error (RMSE): 0.37161434047046776
```

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:

$$y = mx + c$$

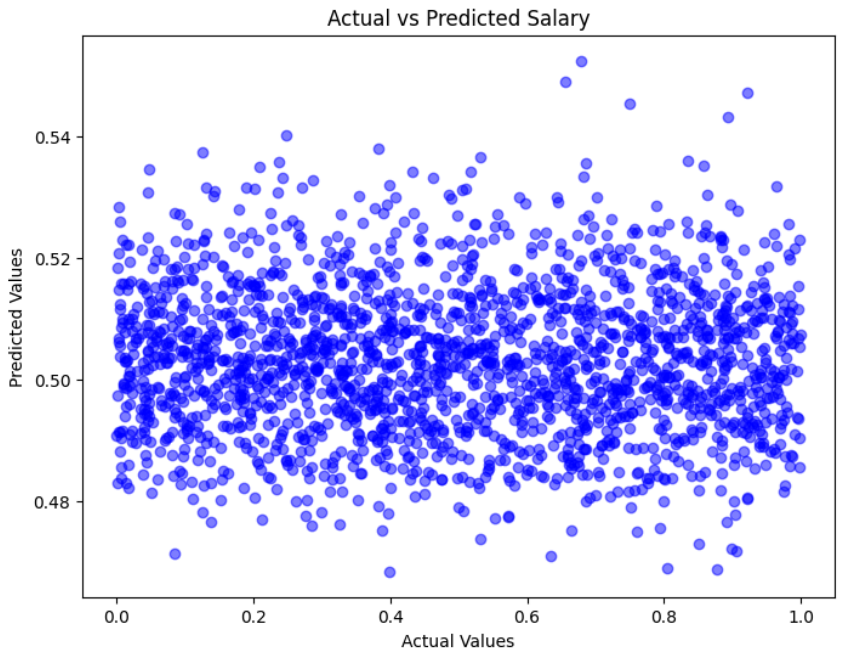
```
lin_reg = LinearRegression()
lin_reg.fit(X_train_reg, y_train_reg)

# Predict
y_pred_reg = lin_reg.predict(X_test_reg)

# Evaluate Model
print("R-squared Score:", lin_reg.score(X_test_reg, y_test_reg))

# Plot Predicted vs Actual values
plt.figure(figsize=(8, 6))
plt.scatter(y_test_reg, y_pred_reg, alpha=0.5, color='blue')
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted Salary")
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.