TECH SAKSHAM

FINAL CASE STUDY

Name: Raksha U

Department: MCA (BCU)

Semester: Third

USN: P18BR23S126017

---

1. Title

Predicting Customer Churn Using Logistic Regression

---

2. Objective:

The objective of this project is to build a machine learning model using logistic regression to predict customer churn for a given dataset. By identifying customers who are likely to churn, businesses can take proactive measures to retain them and improve customer satisfaction.

---

3. Problem Statement:

Customer churn is a critical issue for businesses, as losing customers directly impacts revenue and growth. It is essential to identify patterns in customer behaviour that may indicate a higher likelihood of churn. The dataset contains customer information, and the task is to use this data to predict whether a customer will churn (Yes) or not (No).

---

4. Solution:

The solution involves the following steps:

- Loading the Dataset: Importing and exploring the customer churn dataset.
- Data Pre-processing: Cleaning and pre-processing the data by handling missing values, encoding categorical variables, and addressing any class imbalance.
- Feature Splitting: Splitting the data into training and testing subsets to evaluate the model's performance.
- Model Training: Training a logistic regression model to predict customer churn.
- Evaluation: Evaluating the model using metrics such as accuracy, classification report, and confusion matrix to assess its performance.

5. Code Implementation:

```
# Import required libraries
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.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Step 1: Load the dataset
file_path = "customer_churn.csv"  # Update this to your file path
data = pd.read_csv(file_path)

# Step 2: Data Preprocessing
# Drop unnecessary columns
data = data.drop(['customerID'], axis=1)

# Convert 'TotalCharges' to numeric
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')

# Handle missing values in 'TotalCharges'
data['TotalCharges'] = data['TotalCharges'].fillna(data['TotalCharges'].median())

# Encode categorical variables
data = pd.get_dummies(data, drop_first=True)

# Check for imbalance in target variable
print("\nClass Distribution in Target Variable:")
print(data['Churn_Yes'].value_counts())

# Step 3: Splitting the dataset
X = data.drop('Churn_Yes', axis=1)
y = data['Churn_Yes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Step 4: Model Training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Step 5: Predictions and Evaluation
y_pred = model.predict(X_test)

# Evaluation Metrics
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=1))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

6. Output:

```
Class Distribution in Target Variable:
Churn_Yes
False    6
True     4
Name: count, dtype: int64

Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

       False       1.00      1.00      1.00         1
        True       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```