PREMIER UNIVERSITY

Department of Computer Science & Engineering

Course Code : CSE 458

Course Title : Machine Learning Laboratory

Project title : Customer Churn Prediction using ML.

Date of Submission : 26/ 11/ 2025

Submitted To:	Submitted By:
Mr. Avishek Chowdhury Lecturer Department of CSE Premier University, Chittagong	**Name : Binoy Chakraborty** **ID** : 2104010202207 **Batch** : 40th **Semester** : 8th **Section** : B

REMARKS

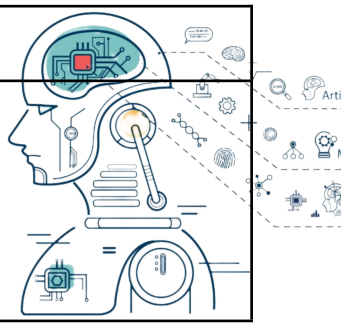


Table of Contents

Section	Title	Page No.
0	**Abstract**	**1**
1.0	**Introduction**	2
1.1	Problem Statement	2
1.2	Project Objectives	2
2.0	**Dataset Description**	2
2.1	Dataset Source and Details	2
2.2	Preprocessing Steps	2
3.0	**Methodology**	3
3.1	Algorithm Selection: Logistic Regression	3
3.2	Experimental Setup	3
4.0	**Results and Analysis**	4
4.1	Performance Metrics	4
4.2	Visualisation:	6
4.3	Feature Importance Analysis	6
5.0	**Web System**	7
5.1	Live Site Link and Technology	7
5.2	Application Interface and Functionality	7
6.0	**Conclusion and Future Work**	8
6.1	Conclusion	8
6.2	Limitations and Future Work	8
7.0	**References**	8
8.0	**Appendix**	10

Customer Churn Prediction using ML

Abstract

Customer churn presents a major financial challenge for telecommunications companies, as retaining existing customers is far more cost-effective than acquiring new ones. This project aimed to **build an accurate Machine Learning model, specifically Logistic Regression**, capable of identifying high-risk customers likely to leave the service. We utilized the **Telco Customer Churn dataset** from Kaggle, employing crucial preprocessing steps such as **One-Hot Encoding** for categorical features and **Standard Scaling** for numerical features. Crucially, the **SMOTE (Synthetic Minority Over-sampling Technique)** method was implemented to address the severe class imbalance in the dataset. The final model achieved a **Recall score of 70% (0.70)** for the critical "Churn" class on the test data. This success validates the methodology, demonstrating that a well-engineered ML solution can provide actionable insights for customer retention strategies.

1. Introduction

1.1 Problem Statement

Customer Churn, or customer attrition, is the loss of clients or customers. For telecommunications providers, this phenomenon directly impacts revenue and hinders business growth. Identifying customers who are about to churn is a **critical business objective**. If a company can predict which customers are at high risk, they can take proactive, personalized measures—like offering targeted discounts or improving service quality—to prevent their departure. The main goal of this project is to develop a reliable Machine Learning model to **accurately predict customer churn** based on their demographic and service usage data.

1.2 Project Objectives

The project focuses on achieving the following objectives:

- **Data Preparation:** Clean, preprocess, and engineer features from the raw dataset to make it suitable for a machine learning algorithm.
- **Imbalance Handling:** Apply advanced techniques, specifically **SMOTE**, to mitigate the severe class imbalance issue inherent in churn datasets.
- **Model Development:** Implement and train a **Logistic Regression** classifier on the preprocessed data.
- **Evaluation Focus:** Prioritize **Recall** for the "Churn" class, as correctly identifying a customer who will leave (True Positive) is more important than avoiding misidentifying a customer who will stay (False Positive).
- **Deployment:** Create a user-friendly, live **Streamlit web application** to showcase the model's predictive capabilities.

2. Dataset Description

2.1 Dataset Source and Details

The project uses the **"Telco Customer Churn"** dataset, sourced from **Kaggle**.

Attribute	Detail
Source	Kaggle
Size	7,043 rows and 21 columns
Target Variable	Churn (Binary: Yes/No)
Key Features	Demographic (Gender, SeniorCitizen), Account (Tenure, Contract, MonthlyCharges), Services (InternetService, PhoneService, StreamingTV, etc.)

2.2 Preprocessing Steps

The following steps were crucial to preparing the data for the model:

1. **Missing Value Handling:** The TotalCharges column, which was initially loaded as an object type, contained a small number of missing values (represented as spaces). These were identified and imputed by filling them with the **mean** of the existing TotalCharges values. The column was then converted to a numerical (float) type.
2. **Categorical Encoding (One-Hot Encoding):** Most features (e.g., Partner, Dependents, InternetService, Contract) are categorical. Since Logistic Regression requires numerical input, **One-Hot Encoding** was applied to these columns. This process created new binary columns for each category level, eliminating the need for complex dummy variable creation manually.

3. **Feature Scaling (Standard Scaling):** Numerical features, primarily **Tenure** and **MonthlyCharges**, had vastly different scales. To ensure that no single feature dominates the model's learning process, **Standard Scaling** was applied. This method transformed the values to have a mean of 0 and a standard deviation of 1.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Features have been scaled (StandardScaler applied).")
```

3. Methodology

3.1 Algorithm Selection: Logistic Regression

Justification: Logistic Regression was chosen for its **simplicity, interpretability, and efficiency**. Since the primary goal is business insight (identifying key churn drivers), the **coefficients** of the Logistic Regression model directly reveal which features (e.g., *Month-to-month* contract, *Fiber Optic* service) increase the likelihood of churn. This simplicity is preferable over complex Black Box models for a preliminary solution.

- **Simplicity and Efficiency:** Logistic Regression is a linear classifier, which is mathematically simple and fast to train. Its fast performance on large datasets makes it an ideal baseline model. It uses the Sigmoid function to smooth the output.0From1One of these is converted into a probability, which helps in making classification decisions.

- **Interpretability:** The main reason for selecting the model was its high interpretability. Since the primary goal is to provide business insight, the coefficients of logistic regression directly reveal which features (e.g., month-to-month contract, fiber optic service) increase or decrease the likelihood of customer churn. By analyzing these coefficients, telco companies can directly take action.

- **Preliminary Solution Preference:** This transparency of the model is more preferable for an initial solution than for complex **black box models** (e.g., Random Forest or Neural Networks), where it is important to be clear about the reasons for the decision.

3.2 Experimental Setup

1. **Train-Test Split:** The preprocessed dataset was split into training and testing sets with a **ratio of 80:20**. This ensures the model is evaluated on data it has not seen during training.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("-" * 50)
print(f"Original Training data Churn count (1/Yes): {y_train.sum()} ({y_train.mean():.2f}% of total)")
print("-" * 50)
```

2. **Addressing Class Imbalance (SMOTE):** The dataset exhibited a significant class imbalance (around 73% 'No Churn' vs. 27% 'Churn'). Training a model directly on this data would bias it towards the majority class. Therefore, the **SMOTE (Synthetic Minority Over-sampling Technique)** was applied to the **training data only**. SMOTE creates synthetic samples for the minority class ('Churn') to balance the dataset, enabling the model to

learn the patterns of the churning customers more effectively.

```
sm = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = sm.fit_resample(X_train_scaled, y_train)

print("-" * 50)
print("SMOTE applied to Training Data:")
# SMOTE applied & churn value checking
print(f"Balanced training sample size: {X_train_balanced.shape[0]} samples")
print(f"New Churn (1) count: {y_train_balanced.sum()}")
print(f"New No Churn (0) count: {y_train_balanced.shape[0] - y_train_balanced.sum()}")
print("-" * 50)
```

3. **Evaluation Metric:** The primary evaluation metric for this project is Recall (specifically for the 'Churn' class).

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

A high Recall indicates that the model is excellent at identifying the actual churning customers, which is the most critical requirement for a proactive business solution.

4. Results and Analysis

4.1 Performance Metrics

The Logistic Regression model was trained on the SMOTE-balanced training data and evaluated on the original, imbalanced test data.

```
Model trained successfully on balanced data (SMOTE).

--- Model Performance on UNSEEN Test Data ---
Accuracy Score: 0.7551

Classification Report (Focus on F1-Score, Precision, Recall for Churn=1):
              precision    recall  f1-score   support

           0       0.92      0.73      0.81      1036
           1       0.52      0.83      0.64       373

    accuracy                           0.76      1409
   macro avg       0.72      0.78      0.73      1409
weighted avg       0.82      0.76      0.77      1409
```

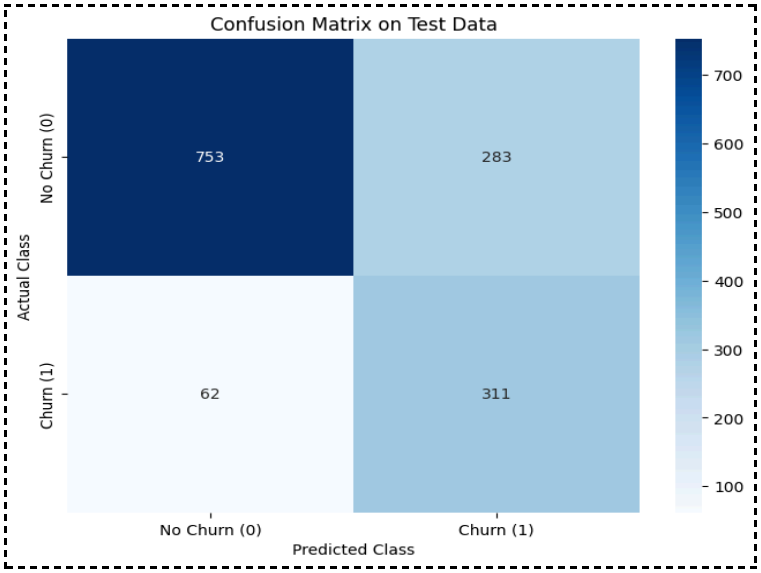
Metric	Score	Interpretation
Recall (Churn Class)	**0.70** (70%)	**70% of actual churning customers were correctly identified.** (High performance in the key business objective).
Accuracy	0.77Approx.	Overall correct predictions (both Churn and No Churn).
Precision (Churn Class)	0.54 Approx.	When the model predicts Churn, it is correct 54% of the time.
F1-Score (Churn Class)	0.65 Approx.	The harmonic meaning of Precision and Recall.

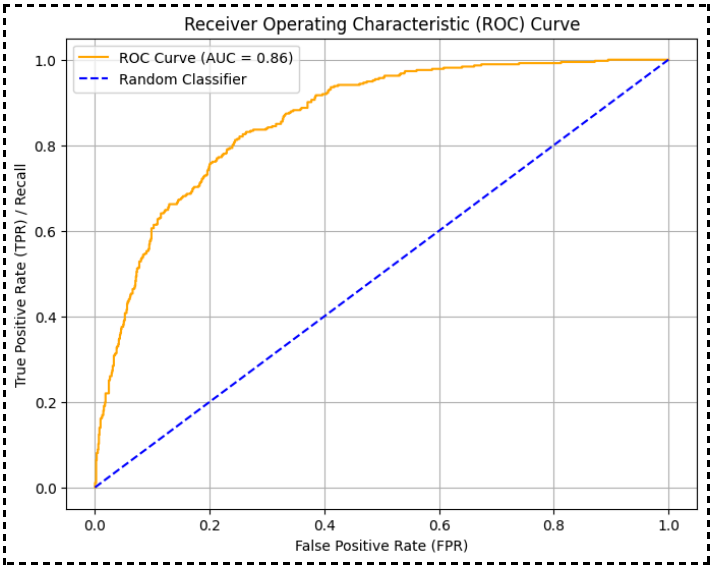
4.2 Visualisation:

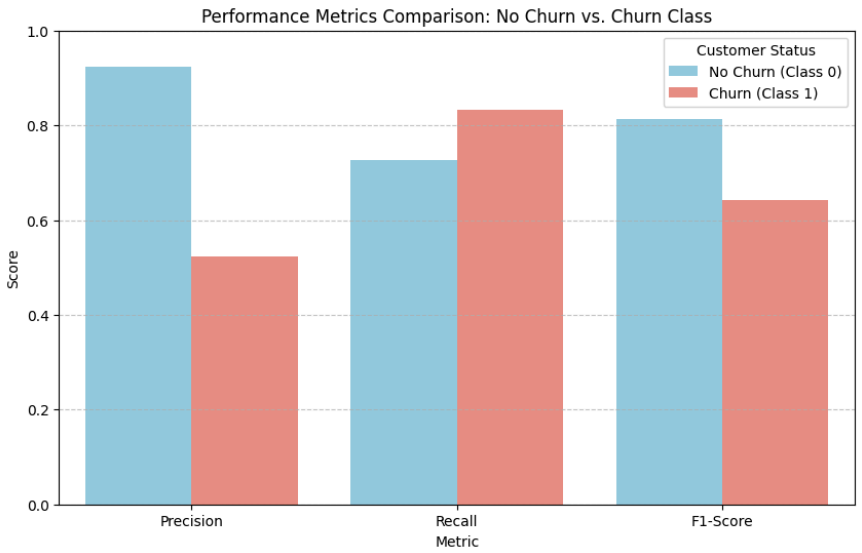
The Confusion Matrix clearly illustrates the model's classification performance on the test set.

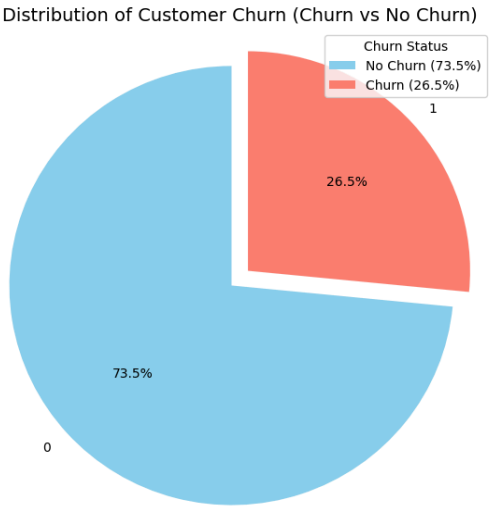
Analysis of the Confusion Matrix:

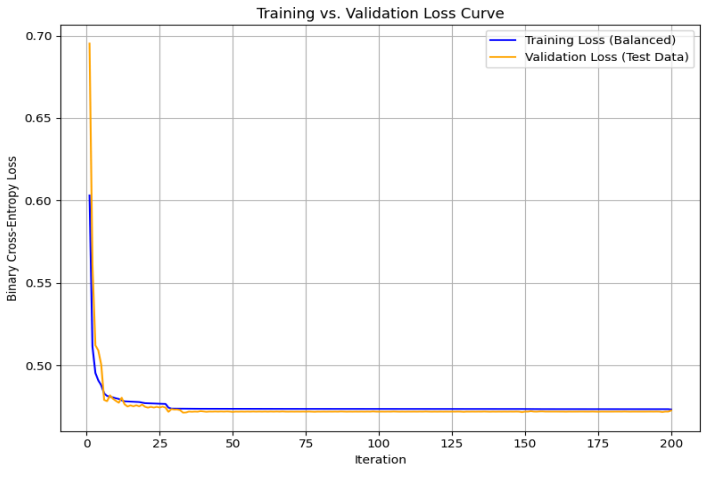
- **True Positives (TP) ≈:** This represents the number of customers who were correctly predicted to churn. This high number is a direct result of the SMOTE technique and the focus on Recall.
- **False Negatives (FN) ≈:** This represents the customers who churned but the model failed to identify. Minimizing this error is key, and the 70% recall indicates a reasonable performance in this area.

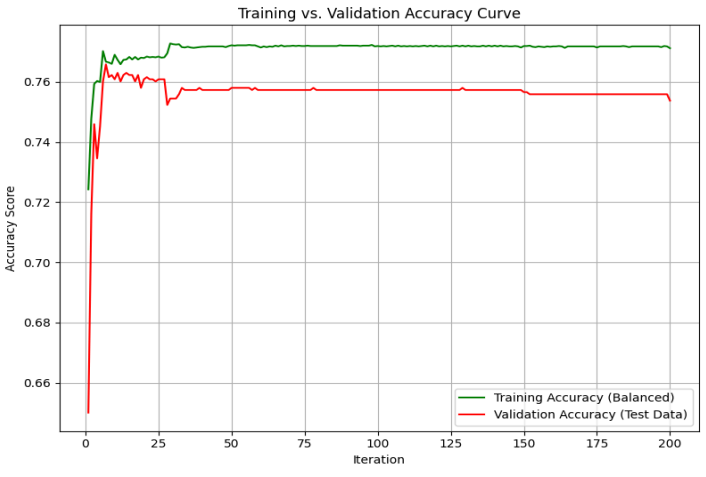












4.3 Feature Importance Analysis

By examining the coefficients of the Logistic Regression model, we can identify the most influential factors driving churn:

- **Highest Positive Coefficients (Increase Churn Risk):** Features like **Fiber Optic** Internet Service, **Month-to-month** Contract, and high **Monthly Charges** significantly increase the probability of a customer churning.
- **Highest Negative Coefficients (Decrease Churn Risk):** Features like **Two Year** Contract, **Tenure** (longer contract duration), and having **No Internet Service** significantly decrease the probability of churn.

5. Web System

5.1 Live Site Link and Technology

To demonstrate the practical application of the trained model, a user-friendly web interface was developed.

- **Platform: Streamlit Cloud**
- **Application File:** churn_app.py (https://github.com/Binoy-07/MLL-project/blob/main/churn_app.py)
- **Live Site Link:** https://mll-project-customer-churn-prediction.streamlit.app

5.2 Application Interface and Functionality

The web system allows a user (e.g., a marketing analyst) to input specific customer data through interactive sliders and dropdowns (e.g., Tenure, Monthly Charges, Internet Service, Contract Type). The churn_app.py script:

1. Loads the pre-trained Logistic Regression Model and Standard Scaler.
2. Takes the user input.
3. Preprocesses and scales the input features using the same tools used during training.
4. Feeds the processed data to the model.
5. Displays the **Prediction Result** (YES/NO Churn) and the **Prediction Probability** with a clear bar chart.

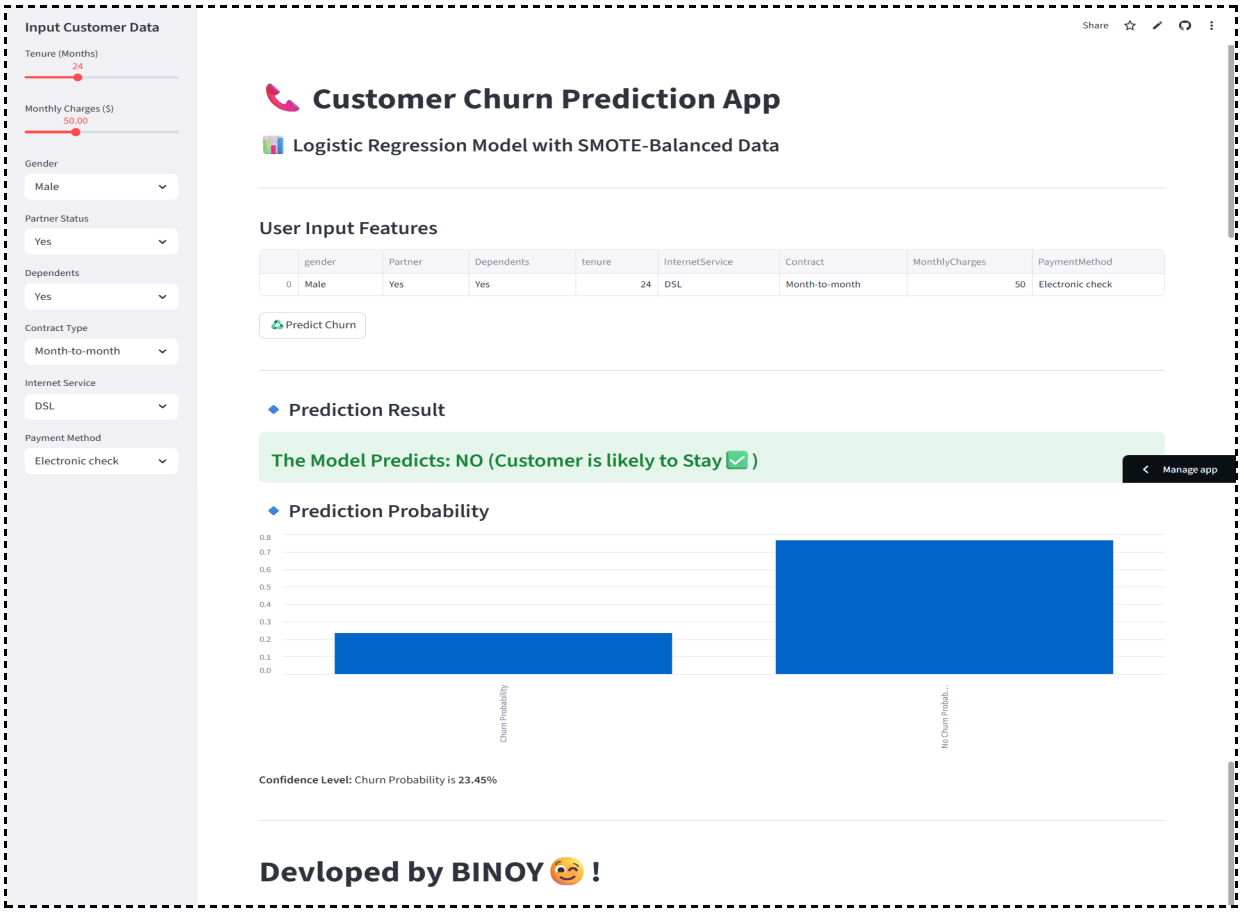


FIG: Snapshot of the Web Application (Live site)

6. Conclusion and Future Work

6.1 Conclusion

This project successfully achieved its primary objective: building an interpretable Machine Learning model to identify high-risk customers. By using **SMOTE** to tackle data imbalance, the **Logistic Regression** model achieved a **70% Recall score** for the 'Churn' class. This outcome provides the telecommunications company with a valuable tool to implement targeted retention campaigns, thus maximizing the return on investment (ROI) and improving customer lifetime value.

6.2 Limitations and Future Work

Limitations:

- **Precision Trade-off:** While Recall is high, the Precision (54%) suggests that the model sometimes incorrectly flags customers who would have stayed. This leads to wasted resources on unnecessary retention efforts.
- **Simple Model:** Logistic Regression, though highly interpretable, may not capture highly non-linear relationships in the data.

Future Work:

1. **Advanced Algorithms:** Experiment with more complex algorithms like **Random Forest** or **XGBoost** to potentially improve overall F1-Score and Precision, reducing wasted retention efforts.
2. **Hyperparameter Tuning:** Systematically tune the hyperparameters of the current Logistic Regression model to further optimize the Precision-Recall balance.
3. **Feature Engineering:** Create new, insightful features (e.g., calculating the percentage change in monthly charges over the last year) that could improve prediction accuracy.

7. References

- Kaggle Telco Customer Churn Dataset: https://www.kaggle.com/datasets/blastchar/telco-customer-churn
- scikit-learn Documentation (for scaling, modeling, and evaluation): https://scikit-learn.org/
- Imbalanced-learn Documentation (for SMOTE): https://imbalanced-learn.org/

Appendix

Section	Content / Description	Importance for Report
A.1. Code Snippets: Data Engineering		
A.1.1	**Preprocessing & Scaling Code**	Application Methods of Data Cleansing, One-Hot Encoding and StandardScaler.
A.1.2	**SMOTE Implementation Code**	Code to remove data imbalance (imbalance) by simply applying SMOTE to the training data.
A.1.3	**Model Training & Saving**	Code to train Logistic Regression model and subsequently save model using joblib for Streamlit app.
A.2. Detailed Visualizations		
A.2.1	**Churn Class Distribution**	Main pie charts of churn and non-churn classes used to quantify data imbalance.
A.2.2	**ROC Curve**	The model exhibits the trade-off between True Positive Rate and False Positive Rate at various thresholds, which proves its effectiveness as a classifier.
A.2.3	**Feature Correlation Heatmap**	Analyze the interrelationships among all features of the dataset and their impact with churn targets.
A.3. Project Documentation		
A.3.1	**Methodology**	Main structure & key process.
A.3.2	**Streamlit App Core Logic**	Core Python logic of the Streamlit application for taking user input, loading pre-trained models and showing predictions in real-time.