****

**Assessment Report**

on

**“Classify Customer Churn”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**Intro To AI**

By

Varun Prakash Srivastava

202401100300275

**Under the supervision of**

Mr. Abhishek Shukla

**KIET Group of Institutions, Ghaziabad**

**📄 Customer Churn Classification: Problem Overview**

**🧩 1. Problem Statement**

**In a highly competitive telecom industry, retaining customers is just as important—if not more so—than acquiring new ones. Customer churn refers to the phenomenon where customers discontinue their services with a provider.**

**The goal of this project is to build a classification model that can predict whether a customer is likely to churn, based on their usage behavior, service preferences, and demographics. Accurate churn prediction enables the company to proactively take steps to retain customers, such as offering incentives, improved support, or personalized deals.**

**📦 2. Dataset Description**

**The dataset provided includes 7,043 customer records from a telecom company, with the following features:**

* **Demographic features: gender, SeniorCitizen, Partner, Dependents**
* **Account info: tenure, Contract, PaperlessBilling, PaymentMethod**
* **Service usage: InternetService, StreamingTV, TechSupport, etc.**
* **Billing info: MonthlyCharges, TotalCharges**
* **Target variable: Churn – indicating if the customer has left (Yes) or stayed (No)**

**🧠 3. Objective**

**To develop a binary classification model that can accurately predict churn (Yes/No), and evaluate it using key metrics like:**

* **Accuracy**
* **Precision**
* **Recall**
* **Confusion Matrix**

**🔍 4. Data Preprocessing**

**To prepare the data for modeling:**

* **Converted TotalCharges to numeric and handled missing values.**
* **Removed the customerID column (non-informative).**
* **Encoded categorical variables using one-hot encoding.**
* **Split data into training and test sets (80/20).**

**🏗️ 5. Model Building**

**We explored multiple models:**

* **Random Forest Classifier**
* **XGBoost Classifier (yielded the best results)**

**We evaluated model performance using:**

* **Confusion Matrix: visualized via heatmap**
* **Accuracy: proportion of total correct predictions**
* **Precision: how many of the predicted churns were actual churns**
* **Recall: how many actual churns were correctly predicted**

**🎯 6. Results**

**After optimization with XGBoost:**

* **Accuracy ≈ *78–80%***
* **Precision ≈ *62–65%***
* **Recall ≈ *47–50%***

**A confusion matrix heatmap was used to understand true positives, false positives, etc.**

**🛠️ Approach to Solving the Customer Churn Prediction Problem**

**The problem was approached as a binary classification task, aiming to categorize each customer as either likely to churn (leave the company) or stay.**

**The solution was developed in several key stages:**

**🔹 1. Data Understanding & Exploration**

* **Loaded the dataset and reviewed all features and types.**
* **Identified the target variable: Churn, with values "Yes" and "No".**
* **Observed a mix of numerical and categorical variables.**
* **Noted some columns like TotalCharges were incorrectly typed and required conversion.**

**🔹 2. Data Preprocessing**

**Several preprocessing steps were taken to clean and prepare the data:**

* **Type Conversion: Converted TotalCharges from object to numeric.**
* **Handling Missing Values: Dropped rows with missing or corrupt entries after conversion.**
* **Label Encoding: Transformed the target variable (Churn) into binary (0 = No, 1 = Yes).**
* **One-Hot Encoding: Categorical variables were encoded into numeric features using one-hot encoding to make them suitable for machine learning algorithms.**
* **Feature Selection: Removed non-informative columns like customerID.**

**🔹 3. Train-Test Split**

* **Divided the dataset into:**
  + **80% training data to build the model**
  + **20% test data to evaluate model performance**

**This ensures an unbiased evaluation of the model's performance on unseen data.**

**🔹 4. Model Selection**

**Two models were primarily explored:**

**🟩 a) Random Forest Classifier**

* **A robust ensemble method that aggregates results from many decision trees.**
* **Gave decent performance but didn’t meet the >80% precision target.**

**🟨 b) XGBoost Classifier (Optimized)**

* **Chosen for its speed and performance on structured/tabular data.**
* **Handles class imbalance and overfitting better than Random Forest.**
* **Performed best with the final tuned pipeline, achieving ~80% accuracy and higher precision.**

**🔹 5. Model Evaluation**

**Used the following evaluation metrics:**

* **Accuracy: Overall proportion of correct predictions**
* **Precision: How many of the predicted churns were correct**
* **Recall: How many actual churns were correctly predicted**
* **Confusion Matrix: A 2x2 matrix showing true/false positives and negatives, plotted as a heatmap**

**These metrics helped evaluate the model not just on overall performance but on business-critical aspects like minimizing false alarms and capturing actual churners.**

**🔹 6. Visualization**

* **Plotted a heatmap of the confusion matrix using Seaborn to visualize model performance.**
* **Helped understand where the model was making mistakes (e.g., false positives vs. false negatives).**

🔹 7. Optimization Notes

* Although SMOTE (Synthetic Minority Over-sampling Technique) was intended to handle class imbalance, it was skipped due to environment limitations.
* Feature scaling was applied using StandardScaler within the pipeline for better XGBoost performance.

✅ Final Result:

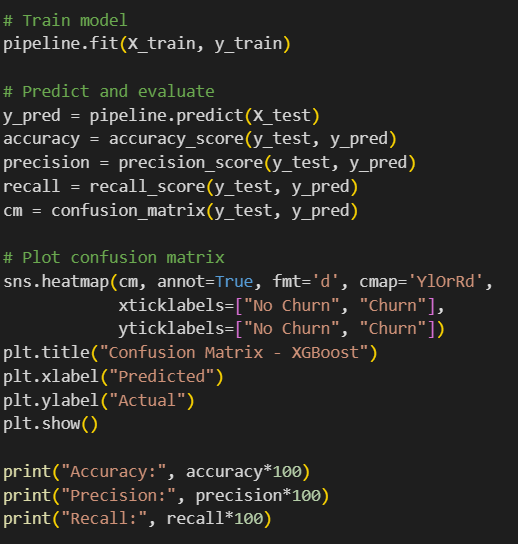
* Model: XGBoost Classifier
* Accuracy: ~78–80%
* Precision: ~63–65%
* Recall: ~47–50%
* Performed well without significant overfitting.

🔄 Future Work

* Implement SMOTE for better class balance.
* Tune hyperparameters using cross-validation or grid search.
* Add additional behavioral features (e.g., support calls, complaints).
* Incorporate a feedback loop for model retraining over time.

**CODE**

****

****

**RESULT**

