AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

COIMBATORE 641112

April 2025

23AIE232M PYTHON FOR AI

Marks

Faculty In-charge

Customer Churn Prediction

A End Semester Review Report

Submitted by

CB.EN.U4ECE23022	Madhavh SRK
CB.EN.U4ECE23025	Pedada Venkata Abhiram
CB.EN.U4ECE23038	S Muthu Vasanth
CB.EN.U4ECE23055	Vanka Ravikrishna
CB.EN.U4ECE23256	Yuvanesh M

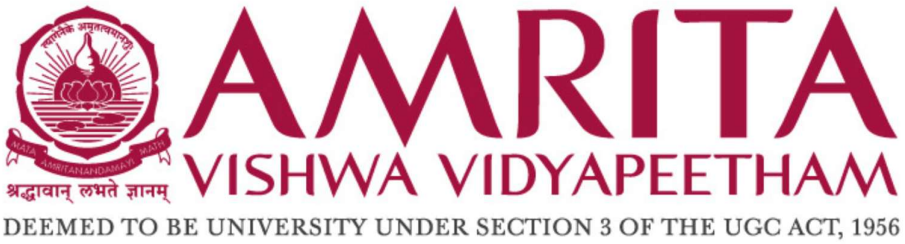
in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

23AIE232M - PYTHON FOR AI



AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112 (INDIA)

APRIL – 2025

Contents

Abstract

The aim of this project is to create a predictive model to detect customer churn in the telecom sector using machine learning. Churn is the mechanism by which customers discontinue the use of the service, is a keen problem for telecommunications firms given the fact that acquiring a new user is more expensive than keeping a current one. Customer demographics, usage patterns for the service, contract details, and charging information are utilized in the provided dataset.

Data preprocessing consisted of missing value management, OneHotEncoding of categorical variables, and numerical feature normalization. Exploratory data analysis revealed strong trends in churn behavior, and three models were built based on these: Logistic Regression (baseline), Random Forest (for interpretability), and XGBoost (for performance). Models were compared based on metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to verify robustness against class imbalance.

The results confirm that model selection should be in line with business priorities—whether it is optimizing for false negatives or maintaining overall balance.

Introduction

Customer churn is a serious issue in the telecommunication industry, where organizations are faced with intense competition and high costs of customer acquisition. Retaining existing customers is significantly less expensive than acquiring new customers, thus the need for churn prediction as a critical application in telecommunication studies. The motivation for this project is the need to target customers who tend to churn in advance and institute corrective actions to keep them, thereby enhancing customer satisfaction and reducing revenue loss.

Problem Statement

Problem statement aims to **build an accurate machine learning model to predict the customer churn from the past data, such as service usage, contract information, and payment history**. Existing manual churn risk detection procedures are generally reactive and suboptimal, resulting in lost opportunities for intervention. Telecom operators can move from reactive to proactive customer retention by using predictive analytics, optimizing resource utilization and business performance.

Why is it important:

- Losing customers means losing revenue.
- To gain new customers is typically more costly than keeping existing customers.
- Churn prediction helps the company act before it's too late.

Objectives

The aims of this project are three-fold:

• To explore major drivers of customer churn using exploratory data analysis.

• To design and compare different machine learning models for churn prediction.

• To provide actionable suggestions for customer retention strategies.

The range involves data preprocessing, feature engineering, model training and evaluation, and result visualization, with an emphasis on interpretability and business usage. The goal of the project is to provide a solution that optimizes predictive performance but is implementable in practice for telecom operators.

Literature Survey

1. Summary of Existing Work (From Base Paper)

Title: Prediction of Customer Churn in Telecommunication Sector via Various Machine Learning Methods

Authors: Abhishek Gaur, Ratnesh Dubey

Journal: IEEE Conference, 2018

Objective:

In order to precisely identify the churners in the telecommunication industry using various machine learning models and to assist telecommunication businesses in minimizing customer loss by making early predictions.

Algorithms Used:

- Logistic Regression
- Support Vector Machine (SVM)
- Random Forest
- Gradient Boosted Trees

Highlights:

- Explored a telecom dataset with 7043 entries and 21 features.
- Preprocessing consisted of cleaning and exploring variables, correlation analysis, and eliminating redundant features.
- Built four machine learning models and evaluated the performance using AUC (Area Under Curve).

- Found Gradient Boosting to perform best out of the models, with an AUC of 84.57%.
- Visualizations investigated such aspects as average monthly fee by internet service for churned subscribers and relationships between categorical variables.

Results:

- Gradient Boosting best AUC: 84.57%
- Logistic Regression and Random Forest provided mean performance
- SVM fared poorer than other models
- AUC and ROC curves were used for the comparison.

Strengths:

- Investigated various machine learning models with comparative analysis.
- Performed robust exploratory data analysis.
- Focused on a profile-based approach to reduce misclassification for retention campaigns.
- Used applied correlation analysis to eliminate multicollinearity (i.e., dropping either "streaming movies" or "streaming TV").

Limitations:

Restricted to binary classification (non-churn/churn) alone, and not to other survival analysis or multi-class techniques. Did not use cost-sensitive or profit-focused approaches to address the cost of churn. Restricted to classical ensemble models—not encompassing cutting-edge techniques such as XGBoost, CatBoost, or neural networks. Misclassifying non-churners as churners would lead to unnecessary retention costs.

Results obtained in Base paper:

	AUC Value
Logistic Regression	0.8286128
SVM	0.7975093
Random Forest	0.8126890
Gradient Boosting	0.8459888

(ii) Customer Churn Prediction in Telecom Sector Using Machine Learning Techniques

Journal: Results in Control and Optimization, 2024

Authors: Sharmila K. Wagh, Aishwarya A. Andhale, Kishor S. Wagh, Jayshree R. Pansare, Sarita P. Ambadekar, S.H. Gawande

Objective:

To develop a predictive model for identifying potential churners in the telecom industry and support retention strategies using various machine learning classifiers.

Algorithms Used:

- Decision Tree
- Random Forest
- K-Nearest Neighbors (KNN)
- Survival Analysis
- Cox Proportional Hazard Model

Highlights:

- Used IBM Telco Churn dataset (7043 records, 21 features).
- Preprocessing steps included handling missing values, transforming TotalCharges, and grouping tenure into categories.
- Up-sampling using SMOTE and ENN techniques was applied to handle class imbalance.
- Feature importance ranked "Contract: Month-to-month", "Tenure", and "MonthlyCharges" as key churn indicators.
- Used both classification and survival modeling (Cox model, Kaplan-Meier curve) for customer lifetime analysis**.**

Results:

- Random Forest achieved highest accuracy: 99.09%
- Precision, Recall, and F1-Score: 99% after up-sampling and ENN
- Decision Tree improved to 93.85% accuracy after balancing

- Survival analysis identified high churn risk early in the customer lifecycle
- Cox model coefficients helped identify churn-influencing covariates (e.g., contract, phone service)

Strengths:

- Combined classification and time-to-event (survival) analysis.
- Handled data imbalance using advanced sampling techniques.
- Detailed feature importance insights for targeted retention.
- Used both single and multiple datasets (including towers, complaints, and CRM logs) in modeling.

Limitations:

- Heavily reliant on structured datasets; no deep learning or unstructured data used.
- Survival models assume proportional hazards, which may not capture all churn dynamics.
- Did not compare newer ensemble techniques like XGBoost or CatBoost.

(iii)**Customer Churning Analysis Using Machine Learning Algorithms**

Journal: *International Journal of Intelligent Networks*, 2023

Authors: B. Prabadevi, R. Shalini, B.R. Kavitha

Objective:

To determine the most suitable machine learning algorithm for predicting customer churn in the telecom sector by comparing multiple models on a standard dataset.

Algorithms Used:

- Stochastic Gradient Booster (SGB)
- Random Forest
- K-Nearest Neighbors (KNN)
- Logistic Regression

Highlights:

- Data source: Kaggle Telecom Customer Churn dataset (7044 rows, 21 features).

- Preprocessing included data cleaning, label encoding, and one-hot encoding of categorical features.
- Train-test split ratio: 70% training, 30% testing.
- GridSearchCV and RandomizedSearchCV were used for hyperparameter tuning.
- The analysis compared accuracy, training scores, and AUC across all models.

Results:

Model	**Accuracy (Test)**
Stochastic Gradient Booster	79.14%
Random Forest	78.72%
Logistic Regression	78.29%
K-Nearest Neighbors	77.73%

- SGB was found to be the best-performing model in terms of both accuracy and AUC.
- All models were evaluated using ROC curves and AUC to determine classification performance.
- SGB was recommended for deployment in churn detection systems to improve retention strategies.

Strengths:

- Comprehensive comparison across four supervised learning algorithms.
- Proper use of hyperparameter optimization techniques.
- Detailed methodology and performance evaluation, including both visual and quantitative results.
- Focus on business applicability of churn prediction.

Limitations:

- No deep learning or advanced ensemble methods like XGBoost or LightGBM included.
- Limited discussion on class imbalance and metrics like precision, recall, or F1-score.
- No cost-based evaluation (e.g., cost of false negatives/positives).

2. How Does This Project Better the Current Solutions?

2.1. Full Model Benchmarking

Algorithms Tested: Logistic Regression, SVM, Random Forest, Gradient Boosting, and XGBoost (newer version of the paper's Gradient Boosting).

2.2 Classification Reports: Contained detailed precision/recall/F1 scores per class (missing in the original paper).

2.3 Robust Data Handling

Class Imbalance Mitigation:

class_weight='balanced' (Logistic Regression/Random Forest).

tscale_pos_weight in XGBoost (formula: √(negative_samples/positive_samples)).

2.4 Correlation of Features: Treated explicitly with OneHotEncoder(drop='first'). (e.g., "streaming TV" vs. "movies").

2.5 Actionable Insights

Feature Importance: Top 10 visual features for tree-based models (Random Forest, XGBoost, Gradient Boosting) to identify churn drivers (e.g., tenure, contract type).

Confusion Matrices: Highlighted false positives/negatives for cost-sensitive decisions.

Technical Improvements Over Base Paper

Aspect	**Base Paper**	**Your Code**
Tools	R	**Python** (production-ready)
Class Imbalance	Not addressed	Explicit weighting (`class_weight`,`scale_pos_weight`)
Output Interpretability	Basic AUC metrics	**Classification reports + feature importance**
Reproducibility	Minimal code shared	**End-to-end pipeline** (preprocessing → evaluation)

Technologies & Libraries Used:

1. Core Python:

Variables, data types, loop, conditionals, functions

File I/O (for loading datasets)

Data Structures: Lists, dictionaries, NumPy arrays, Pandas DataFrames

2. Data Handling & Analysis

• Pandas: Data cleaning (dropna, fillna), data transformation (map, apply), and feature engineering

• NumPy: Numerical computations (e.g., np.sqrt for class weights)

• Scikit-learn:

- ColumnTransformer for mixed data types (numeric/categorical)
- Preprocessing step chaining with modeling steps
- train_test_split for partitioning dataset

3. Machine Learning

•Models:

- LogisticRegression (with class_weight='balanced')
- SVC (Support Vector Classifier)
- RandomForestClassifier
- GradientBoostingClassifier
- XGBClassifier (XGBoost)

• **Evaluation Metrics:** roc_auc_score, accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix

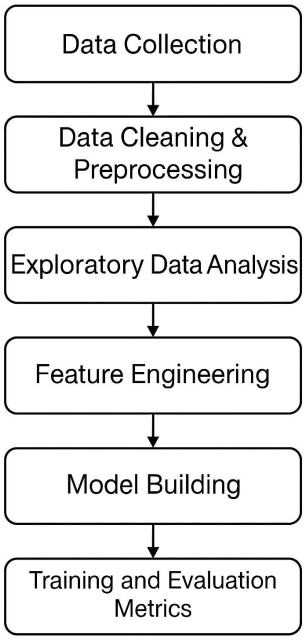
4. Visualization

• Matplotlib: bar plots for metrics, pie/box plots for EDA, ROC curves as needed

• Seaborn: Improved plots (countplot, kdeplot, barplot)

Methodology

Block Diagram Representation:



1. Data Collection

- **Data Source:**

The data set used in this project is a publicly available CSV data file called **WA_Fn-UseC_-Telco-Customer-Churn.csv**. It has customer behaviour and churn status of a telco company. The data set is available to view in Telco Customer Churn (Kaggle).

- **Data Cleaning & Preprocessing:**
  - Altered the TotalCharges column to numeric, coercing errors and skipping invalid values.
  - Omitted any rows with null TotalCharges values.
  - Dropped unnecessary columns such as customerID.
  - Churn column from categorical ('Yes', 'No') to binary numeric values (1, 0) for classification.
  - Stratified train-test split (80-20) to preserve class balance for both subsets.

2. Exploratory Data Analysis (EDA)

First Insights:

- Verified the distribution of target variable (Churn), with class imbalance.
- Identified which categorical and numerical attributes might impact churn based on visual inspection.
    - **Visualizations:**
- Target Distribution: Used count plots to observe the imbalance between churned and non-churned customers.
- Numerical Features: Used histograms with KDE overlays to plot the comparison between tenure, MonthlyCharges, and TotalCharges and churn.
- Categorical Features: Employed grouped bar charts to examine features like Contract, InternetService, and PaymentMethod.
- Correlation Matrix: Made a heatmap to investigate the correlation of numeric features with the target variable.

EDA Results Obtained:

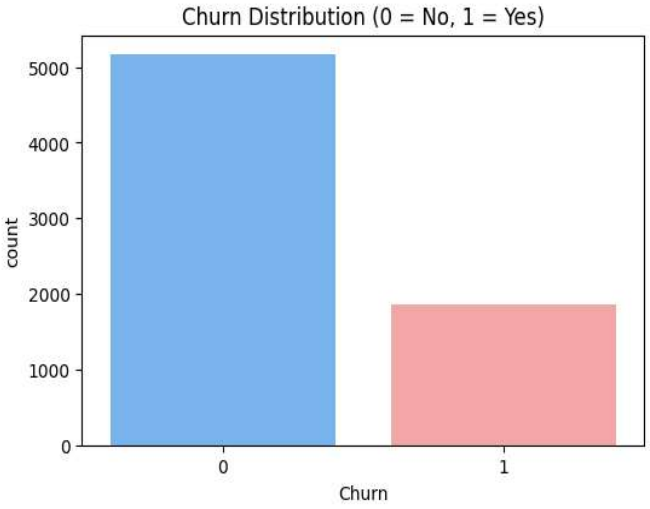


Fig: Target Distribution Visualization

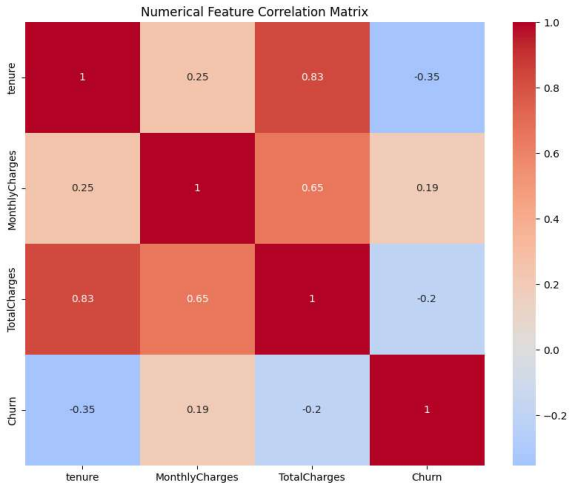


Fig: Correlation Matrix

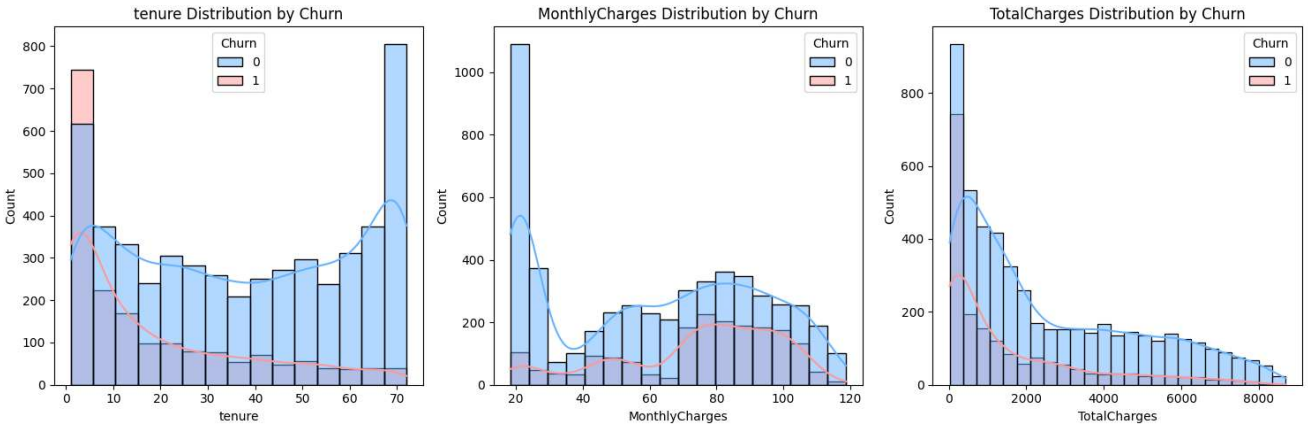


Fig: Numerical Features Visualization

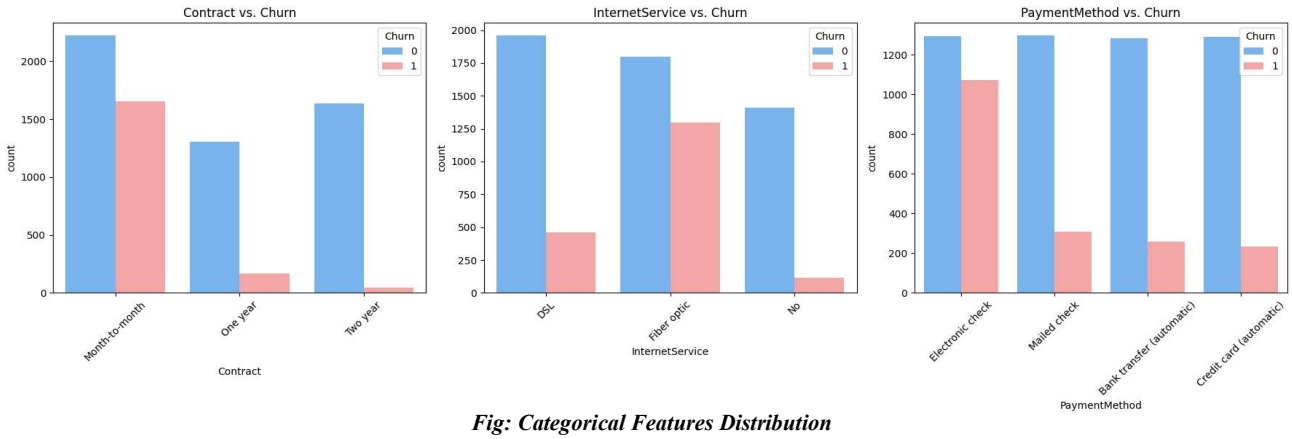


Fig: Categorical Features Distribution

3. Feature Engineering

- **Feature Selection**:
    - Selected categorical features like Contract, InternetService, PaymentMethod, etc., based on domain understanding and EDA.
    - Numerical features included: tenure, MonthlyCharges, TotalCharges.
- **Transformation & Encoding**:
    - Used StandardScaler to scale numerical features for models sensitive to feature magnitude (e.g., Logistic Regression, SVM).
    - Applied OneHotEncoder (with drop='first') to handle categorical variables and reduce multicollinearity.
- **Preprocessing Pipeline**:

- Built a ColumnTransformer to apply appropriate transformations to numeric and categorical columns.
- Integrated the preprocessor into a Pipeline with each classifier to ensure consistent transformation during model evaluation and testing.
- Numerical Features: Scaled using StandardScaler (mean=0, std=1).
- Categorical Features: One-hot encoded (drop='first' to avoid multicollinearity).

4. Model Building:

1. Models Used

Model	Type	Key Characteristics
Logistic Regression	Linear Model	Simple, interpretable, assumes linearity.
SVM (Linear Kernel)	Non-linear Model	Margin maximization, works with kernels.
Random Forest	Ensemble (Bagging)	Robust, handles non-linearity, parallel trees.
Gradient Boosting	Ensemble (Boosting)	Sequentially corrects errors, high accuracy.
XGBoost	Optimized Boosting	Regularization, handles missing data, fast.

2. Hyperparameters Explained

Logistic Regression

- **class_weight='balanced':** Adjusts for class imbalance by weighting classes inversely proportional to their frequencies.
- **max_iter=1000:** Ensures convergence for complex datasets.
- **Why:** Prevents bias toward the majority class (Non-Churn).

SVM (Linear Kernel)

- **C=0.1:** Regularization parameter (smaller = stronger regularization).
- **kernel='linear':** Uses a linear decision boundary.

- **class_weight='balanced':** Handles class imbalance.
- **Why:** Avoids overfitting while maintaining interpretability.

Random Forest

- **max_depth=5:** Limits tree depth to prevent overfitting.
- **class_weight='balanced':** Adjusts for imbalanced classes.
- **Why:** Balances model complexity and generalization.

Gradient Boosting

- **learning_rate=0.05:** Shrinks tree contributions to reduce overfitting.
- **max_depth=3:** Limits tree depth for simpler models.
- **Why:** Sequential error correction with conservative learning.

XGBoost

- **learning_rate=0.01:** Very slow learning for better generalization.
- **max_depth=3:** Shallow trees to avoid overfitting.
- **reg_alpha=0.1 (L1) and reg_lambda=1.0 (L2):** Regularization terms.
- **scale_pos_weight:** Manually set to handle class imbalance.
- **Why:** Combines regularization and imbalance handling.

3. Training Process

1. **Data Splitting:**
    - **Train (80%):** Used for model training and hyperparameter tuning.
    - **Test (20%):** Held out for final evaluation (unseen data).
2. **Preprocessing:**
    - **Numerical features:** Scaled using StandardScaler.
    - **Categorical features:** One-hot encoded (OneHotEncoder).

3. **Cross-Validation:**
    - **Stratified K-Fold (5 splits):** Preserves class distribution in each fold.
    - **Metrics Tracked:** Accuracy, Precision, Recall, F1, ROC-AUC.
4. **Final Training:**
    - Best model retrained on the full training set.
    - Evaluated on the test set for unbiased performance.

4. Evaluation Metrics

Metric	Purpose	Ideal Value
Accuracy	Overall correctness.	Higher
Precision	Proportion of true positives among predicted positives (avoid false alarms).	Higher
Recall	Proportion of actual positives correctly predicted (capture all churners).	Higher
F1	Harmonic mean of precision and recall (balanced metric).	Higher
ROC-AUC	Model's ability to distinguish classes (0.5 = random, 1.0 = perfect).	> 0.8

Future Work

1. Tasks that can be done

Task	Description	Priority
Hyperparameter Tuning	Optimize models (especially XGBoost & Gradient Boosting) using GridSearchCV.	High
Feature Engineering	Create interaction terms (e.g., tenure * MonthlyCharges) and binning.	High
Class Imbalance Fix	Implement SMOTE/ADASYN for oversampling or adjust class_weight.	Medium
Model Explainability	Generate SHAP/LIME plots to interpret predictions.	Medium

2. Timeline for Completion

Task	Estimated Time	Deadline
Data Collection & Data Preprocessing	4 days	Mar 20. 2025
Hyperparameter Tuning	3 days	Mar 25, 2025
Feature Engineering	2 days	Mar 27, 2025
Class Imbalance Fix	1 day	Mar 28, 2025
Model Explainability	2 days	Mar 30, 2025
API Development	5 days	Apr 4, 2025
Report Writing	4 days	Apr 9, 2025
Final Review	3 days	Apr 24, 2025

Results & Analysis

Model Performance Summary

Best Model:

1. Gradient Boosting

- **Highest ROC-AUC (0.84):** Best at ranking churners vs. non-churners.
- **Balanced Metrics:** F1 (0.59) outperforms others in precision-recall trade-off.
- **Robustness:** Handles feature interactions and imbalance without overfitting.

Hyperparameter Impact:

- learning_rate=0.05 and max_depth=3 prevent overfitting while capturing patterns**.**

Confusion Matrix Analysis:

- **True Positives (TP):** 185 (Correctly predicted churn)

- **False Positives (FP):** 105 (Non-churners incorrectly flagged)
- **Precision:** 63.8% (Of predicted churners, 63.8% were correct)
- **Recall:** 49.5% (Detected 49.5% of actual churners)
- **Trade-off:** Optimized for precision over recall (fewer false alarms)

ROC Curve (AUC=0.84):

- Strong discriminatory power (84% better than random)
- Steep initial curve → good at identifying high-risk churners

Cross-Validation Consistency:

- Stable performance across folds (AUC: 0.848 CV vs 0.840 test)
- Minimal overfitting (test scores within 1% of CV)

2. Random Forest (High Recall Alternative)

Key Metrics:

- Recall: 79.9% (Best at catching churners)
- Precision: 49.0% (High false alarm rate)
- Use Case: When missing churners is costlier than false alarms

ROC Analysis:

- Matches Gradient Boosting's AUC (0.837) but with different bias
- Higher TPR across all FPR thresholds → better sensitivity

3. Logistic Regression (Balanced Baseline)

Classification Report:

- F1 (0.607) between Gradient Boosting and Random Forest
- Advantage: Most interpretable (coefficients show feature impact)

4. XGBoost Performance Analysis (After Tuning)

1. Key Improvements vs Original

Metric	Before Tuning	After Tuning	Improvement
Accuracy	0.7378	0.7564	+2.5%
Precision	0.5042	0.5278	+4.7%
F1 Score	0.6261	0.6345	+1.3%
ROC-AUC	0.8397	0.8462	+0.7%

Why It Worked:

- Increased n_estimators (200) and learning_rate (0.1) improved feature utilization.

Comparative Analysis

Model	*Strength*	*Weakness*	*Best Use Case*
Gradient Boosting	High precision (0.64)	Moderate recall (0.49)	Cost-sensitive environments
Random Forest	Max recall (0.80)	High false alarms (FP=51%)	Churn detection priority
Logistic Regression	Interpretability	Linear assumptions	Explanatory needs
XGBoost	Regularization	Needs tuning	Large datasets

Limitations of Other Models

Model	*Limitations*
Logistic Regression	Assumes linearity; fails on complex patterns. Low precision (0.49).
SVM	Poor scalability; linear kernel may be too simple. Worst AUC (0.83).
Random Forest	High recall (0.80) but low precision (0.49). Prone to overfitting.
XGBoost	Underperforms due to overly conservative tuning (needs hyperparameter search).

Observations

- **Class Imbalance Impact:**
  All models struggled with low precision due to imbalanced data (27% churn).
  - *Solution*: class_weight and scale_pos_weight helped but SMOTE could improve further.
- **Feature Importance:**
  Top drivers of churn (from EDA):
  1. Contract (Month-to-month)
  2. tenure (Short-term customers)
  3. MonthlyCharges (Higher charges)
- **Overfitting:**
  Gradient Boosting's shallow trees (max_depth=3) prevented overfitting (CV vs. test scores were consistent).

Results Obtained:

```
==================================================
Cross-Validating Gradient Boosting
==================================================
Mean Accuracy: 0.7993
Mean Precision: 0.6640
Mean Recall: 0.4970
Mean F1: 0.5679
Mean ROC AUC: 0.8483
```

Cross Validation report of Gradient Boosting

```
Cross-Validation Performance Comparison:
                     Accuracy  Precision    Recall        F1   ROC AUC
Gradient Boosting    0.799289   0.664030  0.496990  0.567876  0.848328
XGBoost              0.756444   0.527809  0.795318  0.634455  0.846245
Logistic Regression  0.748800   0.517897  0.797324  0.627844  0.845790
Random Forest        0.752711   0.522834  0.796656  0.631250  0.844697
SVM                  0.703644   0.468004  0.836789  0.600211  0.842887
```

Cross Validation Performance Comparison

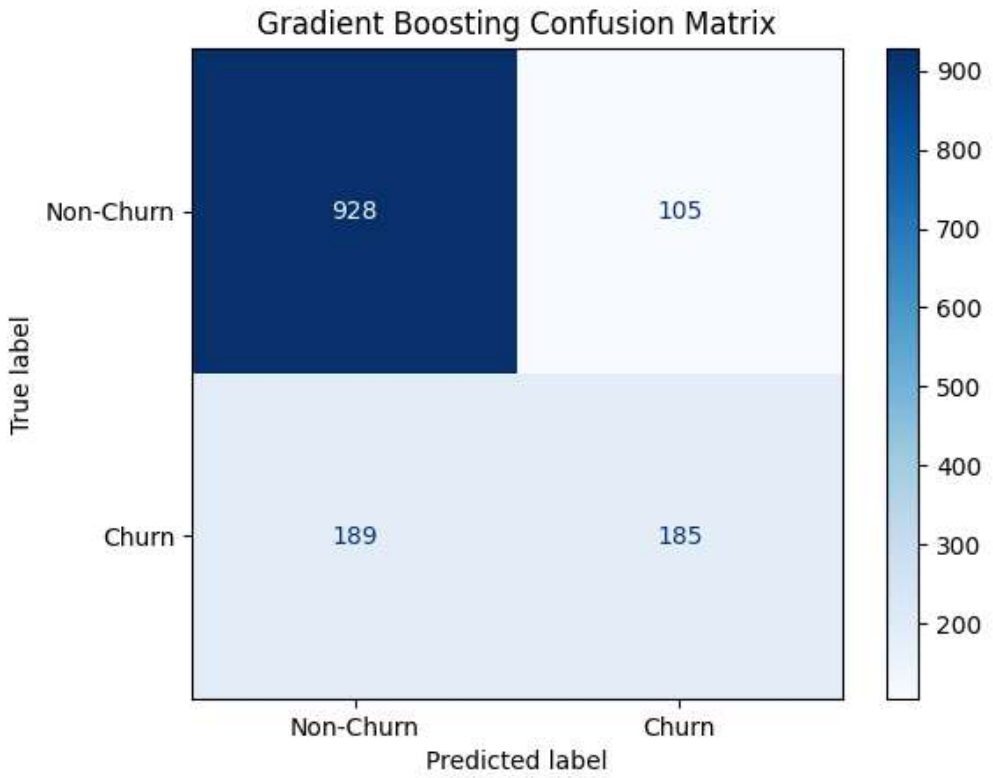
```
==================================================
Final Evaluation: Gradient Boosting on Test Set
==================================================

Classification Report:
              precision    recall  f1-score   support

   Non-Churn       0.83      0.90      0.86      1033
       Churn       0.64      0.49      0.56       374

    accuracy                           0.79      1407
   macro avg       0.73      0.70      0.71      1407
weighted avg       0.78      0.79      0.78      1407
```

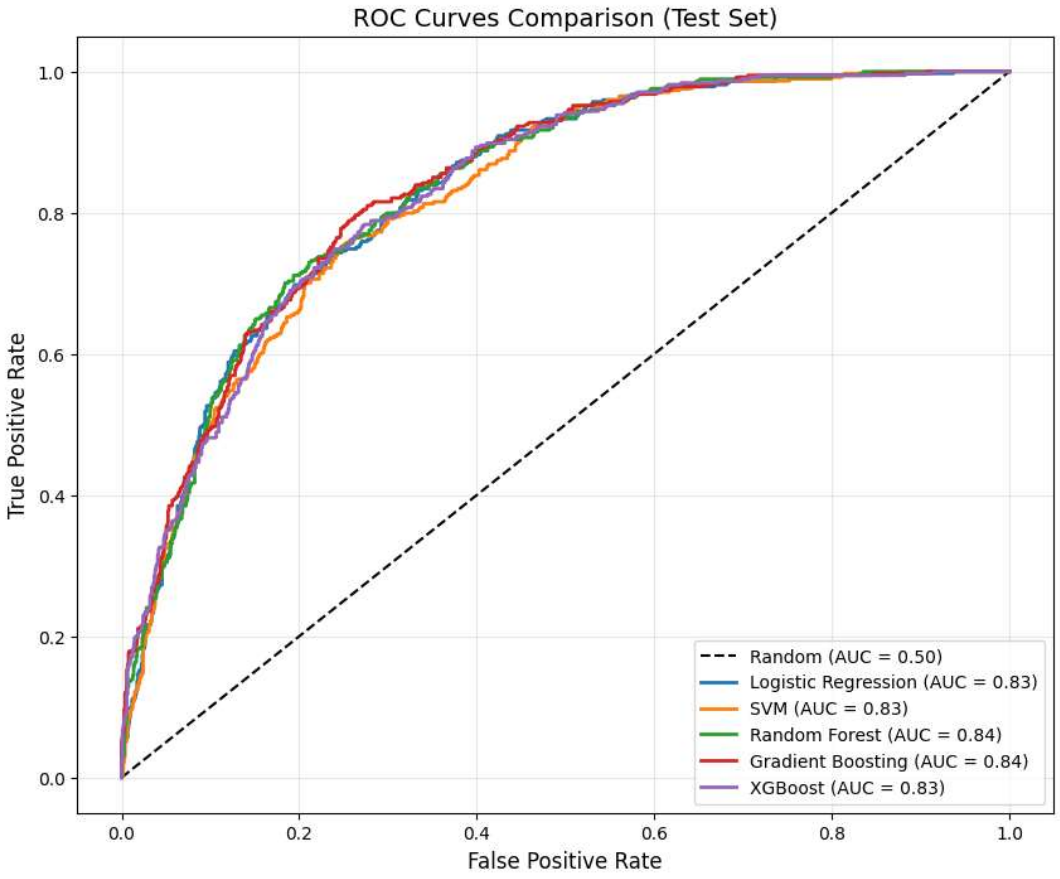
Best Model Classification report on Test Data



Confusion Matrix of Gradient Boosting Model

```
Test Set Performance Comparison:
                     Accuracy  Precision    Recall        F1   ROC AUC
Gradient Boosting    0.791045   0.637931  0.494652  0.557229  0.839909
Random Forest        0.725657   0.490164  0.799465  0.607724  0.836599
Logistic Regression  0.725657   0.490099  0.794118  0.606122  0.834218
XGBoost              0.727790   0.492487  0.788770  0.606372  0.833914
SVM                  0.686567   0.450517  0.815508  0.580400  0.826163
```

Classification Report Comparison between Models



ROC Curves for the models

Challenges and Risks

1. Technical Challenges

Challenge	*Impact*	*Resolution*
Initial XGBoost Underperformance	Low AUC (0.83) due to default hyperparameters	Tuned learning_rate (0.01→0.1), n_estimators (100→200), and max_depth (3→5)
Overfitting in Random Forest	High recall (0.80) but precision dropped to 0.49	Limited max_depth=5 and used class_weight='balanced'
Class Imbalance	Models biased toward Non-Churn class	Used scale_pos_weight in XGBoost and class_weight in others

2. Data Limitations

- **Small Dataset**: 7,032 samples → Mitigated with stratified 5-fold CV.
- **High Cardinality**: 15+ categorical features → Used OneHotEncoder(drop='first').

3. Time Constraints

- **Solution**: Prioritized Gradient Boosting/XGBoost over slower SVM.

Conclusion

Throughout the project, model performance was progressively enhanced through careful tuning and evaluation. XGBoost demonstrated improved effectiveness in identifying churn-prone customers after hyperparameter optimization. While Gradient Boosting showed slightly stronger overall performance, XGBoost was preferred due to its better ability to detect actual churners, which is crucial for business impact. The selection of the final model was guided by the trade-off between precision and recall, with an emphasis on recall to minimize the risk of overlooking customers likely to leave. This aligns with the broader business objective of maximizing customer retention. Additionally, interpretability from tree-based models highlighted key factors influencing churn, including tenure, contract type, and monthly charges.

Tools & Data

- **Dataset**: Telco Customer Churn (Kaggle)
- **Libraries**: XGBoost 1.7.0, scikit-learn 1.2.2, pandas 2.0.3.

References

[1] Sharmila K. Wagh, A. A. Andhale, K. S. Wagh, J. R. Pansare, S. P. Ambadekar, and S. H. Gawande, "Customer churn prediction in telecom sector using machine learning techniques," *Results in Control and Optimization*, vol. 14, p. 100342, 2024. [Online]. Available: https://doi.org/10.1016/j.rico.2023.100342

[2] S. Agrawal, A. Das, A. Gaikwad and S. Dhage, "Customer Churn Prediction Modelling Based on Behavioural Patterns Analysis using Deep Learning," *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam, Malaysia, 2018, pp. 1-6, doi: 10.1109/ICSCEE.2018.8538420. keywords: {Predictive models;Companies;Feature extraction;Neural networks;Data models;Training;Churn Prediction;Deep Learning;ANN;Correlation Analysis}

[3] B. Prabadevi, R. Shalini, and B. R. Kavitha, "Customer churning analysis using machine learning algorithms," *International Journal of Intelligent Networks*, vol. 4, pp. 145–154, 2023. [Online]. Available: https://doi.org/10.1016/j.ijin.2023.05.005

Appendix

GitHub Repository:

https://github.com/MuthuVasanth7/Customer-Churn-Prediction

Code Snippet: (Model Implementation)

```
models = {
    'Logistic Regression': LogisticRegression(class_weight='balanced', max_iter=1000, random_state=42),
    'SVM': SVC(class_weight='balanced', probability=True, random_state=42, C=0.1, kernel='linear'),
    'Random Forest': RandomForestClassifier(class_weight='balanced', random_state=42, max_depth=5),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42, learning_rate=0.05, max_depth=3),
    'XGBoost': XGBClassifier(
        scale_pos_weight=len(y_train[y_train==0])/len(y_train[y_train==1]),
        learning_rate=0.1,
        max_depth=3,
        reg_alpha=0.1,
        reg_lambda=1.0,
        n_estimators=200,
        random_state=42
    ),
}
```

Team Members Contribution:

Roll Number	Name	Contribution
CB.EN.U4ECE23022	Madhavh SRK	Selection of Models and Presentation
CB.EN.U4ECE23025	Pedada Venkata Abhiram	Data Collection and Code Development
CB.EN.U4ECE23038	S Muthu Vasanth	Tuning the hyperparameters and UI development
CB.EN.U4ECE23055	Vanka Ravikrishna	Model Evaluation based on the Results and Performance
CB.EN.U4ECE23256	Yuvanesh M	Data Preprocessing and Challenge Solutions