Telecom Customer Churn Prediction

1. Problem Statement

Customer churn is a critical business challenge for telecom companies. Understanding which customers are likely to leave enables proactive retention strategies. This project develops and compares six machine learning models to predict customer churn based on demographic and service usage data. The goal is to identify the most effective model for accurately predicting churn, enabling targeted intervention efforts to improve customer retention and reduce revenue loss.

2. Dataset Description

Dataset Overview

• **File Name**: `Churn.csv`

• **Total Records**: 7,043 customer records

• **Features**: 20 customer attributes

• **Target Variable**: Churn (Binary: Yes/No)

• **Churn Distribution**: Imbalanced dataset with approximately 26% churn rate

Dataset Characteristics

• **Customer Demographics**: Age-related information through tenure, contract types

• **Service Information**: Internet service type, online security, tech support, phone service

• **Billing Information**: Monthly charges, total charges, contract duration, billing method

• **Data Type**: Mix of categorical and numerical features

Data Preprocessing Steps

1. **Type Conversion**: Converted `TotalCharges` from string to numeric

2. **Missing Values Handling**: Identified NaN values (18 records) and filled with median values

3. **Feature Engineering**:

• Binary encoding: Converted Yes/No columns to 0/1 (Partner, Dependents, PhoneService, PaperlessBilling, Churn)

• Categorical encoding: Applied one-hot encoding to categorical variables

• Feature removal: Dropped non-predictive `customerID` column

4. **Train-Test Split**: 80% training (5,634 samples), 20% testing (1,409 samples)

• Random state: 42 (for reproducibility)

• Stratified split to maintain class balance

Final Feature Set

• **Total Features After Preprocessing**: 46 (after one-hot encoding)

• **Target**: Churn (0 = No Churn, 1 = Churn)

Project Structure

Code:

```
project-folder/ ■ ■■■ app.py # Streamlit web application ■■■ requirements.txt # Python dependencies ■■■ README.md # This file - Project documentation ■■■ Churn.csv # Input dataset ■■■ train_model.ipynb # Jupyter notebook with model training ■ ■■■ model/ # Saved trained models ■■■ logistic_regression_model.pkl ■■■ decision_tree_model.pkl ■■■ knn_model.pkl ■■■ nb_model.pkl ■■■ random_forest_model.pkl ■■■ xgboost_model.pkl
```

3. Models Used and Performance Comparison

3.1 Model Selection Rationale

Six machine learning models were implemented to predict customer churn:

1. **Logistic Regression**: Fast linear classifier, good baseline

2. **Decision Tree**: Interpretable tree-based model with max_depth=5

3. **K-Nearest Neighbors (kNN)**: Instance-based learning with k=5

4. **Naive Bayes (Gaussian)**: Probabilistic classifier

5. **Random Forest**: Ensemble of 100 decision trees

6. **XGBoost**: Gradient boosting ensemble with 100 estimators

3.2 Evaluation Metrics Comparison Table

ML Model Name	Accuracy	AUC	Precision	Recall	F1 Score	MCC
---	---	---	---	---	---	---
Logistic Regression	0.8211	0.7497	0.6862	0.5979	0.6390	0.5230
Decision Tree	0.7999	0.8400	0.6188	0.6354	0.6270	0.4903
K-Nearest Neighbors (kNN)	0.7728	0.7753	0.5942	0.4906	0.5374	0.3949
Naive Bayes	0.6977	0.8376	0.4623	0.8713	0.6041	0.4469
Random Forest (Ensemble)	0.7913	0.8364	0.6513	0.4558	0.5363	0.4178
XGBoost (Ensemble)	0.7928	0.8400	0.6294	0.5282	0.5744	0.4417

3.3 Metric Definitions

• **Accuracy**: Overall correctness = (TP + TN) / Total

• **AUC (Area Under Curve)**: Probability of correctly ranking a random churner higher than a random non-churner

• **Precision**: Of predicted churners, how many actually churned = TP / (TP + FP)

• **Recall**: Of actual churners, how many were identified = TP / (TP + FN)

• **F1 Score**: Harmonic mean of Precision and Recall = 2 × (Precision × Recall) / (Precision + Recall)

• **MCC (Matthews Correlation Coefficient)**: Correlation between predicted and observed values, ranges from -1 to 1

4. Model Performance Analysis and Observations

4.1 Individual Model Performance

	Observation about Model Performance
---	
ighest accuracy (0.8211). Good balance between precision (0.6862) and recall (0.5979). Linear model works well for this dataset. Solid MCC score (0
y (0.7999). Moderate precision (0.6188) and good recall (0.6354), indicating balanced detection of both classes. Simple, interpretable model but lower
d lowest MCC (0.3949). Very low recall (0.4906) means it misses many actual churners. Poor precision-recall trade-off. Instance-based approach less
ng most churners (false negatives minimized). Very low precision (0.4623) means many false positives. Excellent AUC (0.8376) shows good probabilis
t lower recall (0.4558), missing many actual churners. Good AUC (0.8364). Moderate MCC (0.4178). Ensemble method provides robustness. Better th
0.8400). Balanced precision (0.6294) and recall (0.5282). Best ensemble model with good generalization. Highest complexity but justified by performa

4.2 Key Performance Insights

• **Logistic Regression** achieves highest accuracy (82.11%) but this doesn't guarantee best performance for imbalanced churn data

• Accuracy can be misleading; AUC and F1 Score provide better evaluation for imbalanced datasets

• **Naive Bayes**: Maximizes recall (87.13%) but precision only 46.23% - many false alarms

• **Random Forest**: High precision (65.13%) but low recall (45.58%) - misses half of actual churners

• **Logistic Regression & XGBoost**: Better balance between precision and recall

• All models achieve AUC > 0.74, indicating reasonable discriminative ability

• **Decision Tree, Naive Bayes, Random Forest, XGBoost** all achieve AUC ≈ 0.84

• High AUC suggests models can effectively rank churners vs non-churners

• **Logistic Regression** has highest MCC (0.5230), indicating best overall classification quality

• **kNN** has lowest MCC (0.3949), confirming poor performance

• MCC > 0.4 for most models indicates reasonable balance

4.3 Business Impact Considerations

For Churn Prediction Use Case:

1. **False Negatives (Missing Churners)** are costly - lost revenue

2. **False Positives (Unnecessary Offers)** cost resources but prevent churn

Recommended Model Selection:

• **XGBoost** best for balanced approach with AUC 0.8400, accuracy 0.7928

• **Logistic Regression** good for interpretability and highest accuracy

• **Naive Bayes** if capturing all possible churners is priority (high recall requirement)

4.4 Comparative Summary

Aspect	Winner	Score
---	---	---
Highest Accuracy	Logistic Regression	0.8211
Highest AUC	Decision Tree, XGBoost	0.8400
Best Precision	Random Forest	0.6513
Best Recall	Naive Bayes	0.8713
Best F1 Score	Logistic Regression	0.6390
Best MCC	Logistic Regression	0.5230
Overall Best	**XGBoost/Logistic Regression**	**Balanced Performance**

5. Installation & Setup

Prerequisites

• Python 3.8+

• pip or conda

Installation Steps

1. **Clone the repository**

Code:
```
git clone cd Assignment_2
```

2. **Create virtual environment (optional but recommended)**

Code:
```
python -m venv venv source venv/bin/activate # On Windows: venv\Scripts\activate
```

3. **Install dependencies**

Code:
```
pip install -r requirements.txt
```

6. Usage

Training Models (Jupyter Notebook)

Code:
```
jupyter notebook train_model.ipynb
```

• Loads and preprocesses the Churn.csv dataset

• Trains all 6 models

• Saves trained models to the `model/` directory

• Generates evaluation metrics for each model

Running the Streamlit App

Code:

```
streamlit run app.py
```

The web application provides:

• **Home**: Overview and dataset statistics

• **Single Prediction**: Predict churn for individual customers

• **Batch Prediction**: Process multiple customers via CSV upload

• **Model Evaluation**: Upload test data, evaluate models with metrics and confusion matrix

• **Model Comparison**: Compare performance metrics of all models

7. Dependencies

Code:
```
pandas # Data manipulation numpy # Numerical computations scikit-learn # Machine learning models matplotlib # Data visualization seaborn # Advanced visualization streamlit # Web application framework joblib # Model serialization XGBoost # Gradient boosting
```

For exact versions, see `requirements.txt`

8. Recommendations

Best Model for Production

XGBoost is recommended for production deployment because:

• Strong accuracy (0.7928) and AUC (0.8400)

• Balanced precision (0.6294) and recall (0.5282)

• Good generalization with ensemble approach

• Handles feature interactions effectively

Alternative Recommendations

• **Logistic Regression** for maximum interpretability and fastest inference

• **Naive Bayes** if recall (catching all churners) is critical

• **Random Forest** for balance between interpretability and performance

Future Enhancements

1. Hyperparameter tuning with GridSearchCV/RandomizedSearchCV
2. Feature importance analysis and visualization
3. SHAP values for model explainability
4. Class imbalance handling (SMOTE, class weights)
5. Cross-validation analysis
6. Model stacking for improved performance
7. API deployment with Docker
8. Real-time prediction logging

9. Assignment Details

• **Assignment**: 2025aa05444
• **Course**: BITS
• **Date**: January 2026
• **Topic**: Customer Churn Prediction using Machine Learning

Last Updated: January 26, 2026