**Student Name:** AJAI BALAJI R

**Register Number:** 510923243301

**Institution:** Global institute of engineering and technology

**Department:** B.tech[ Artificial intelligence and data science]

**Date of Submission:** [02/05/2025]

**GitHub link**: https://github.com/Aji2206/AJ.NM\_course.git

**TITLE:Customer Churn Prediction Using Machine Learning to Uncover Hidden Patterns**

**## 1. Problem Statement**

Customer churn prediction is a critical challenge for businesses in competitive markets, where retaining existing customers is often more cost-effective than acquiring new ones. Accurately identifying customers who are likely to discontinue service enables companies to implement targeted retention strategies, optimize marketing efforts, and improve overall profitability. This project aims to develop a machine learning model to predict customer churn based on a combination of demographic, account, and usage data. By uncovering hidden patterns and key factors influencing churn, the project seeks to provide actionable insights for business decision-makers.

**## 2. Abstract**

This project focuses on predicting customer churn using machine learning algorithms applied to real-world telecom customer data. The dataset includes customer demographics, account information, service usage, and contract details. The methodology involves data preprocessing, exploratory data analysis (EDA), feature engineering, model training, evaluation, and deployment. Both baseline (Logistic Regression) and advanced models (Random Forest, XGBoost) are implemented and compared. The best-performing model achieves high accuracy and recall, essential for minimizing false negatives in churn prediction. A user-friendly web application is deployed using Gradio, allowing business users to input customer details and receive churn predictions instantly. The project aims to assist companies in proactive customer retention and revenue optimization.

**## 3. System Requirements**

- **\*\*Hardware:\*\***

  - Minimum 4 GB RAM (8 GB recommended)

  - Standard processor (Intel i3/i5 or AMD equivalent)

- **\*\*Software:\*\***

  - Python 3.8+

  - Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, gradio

  - IDE: Jupyter Notebook, VSCode, or Google Colab (preferred for easy setup)

**## 4. Objectives**

- Develop an accurate and interpretable machine learning model to predict customer churn.

- Identify and rank the most influential features affecting churn.

- Provide actionable insights to reduce churn rates.

- Deploy a user-friendly interface for non-technical stakeholders to test churn predictions.

- Compare baseline and advanced models to select the best approach.

**## 5. Project Workflow**

The project workflow is structured into the following stages:

1. **\*\*Data Collection:\*\*** Obtain the telecom customer churn dataset from a trusted public repository.

2. **\*\*Data Preprocessing:\*\*** Clean data, handle missing values, encode categorical variables, and scale features.

3. **\*\*Exploratory Data Analysis (EDA):\*\*** Visualize data distributions, correlations, and churn patterns.

4. **\*\*Feature Engineering:\*\*** Create new features and select relevant ones to improve model performance.

5. **\*\*Model Building:\*\*** Train multiple models including Logistic Regression, Random Forest, and XGBoost.

6. **\*\*Model Evaluation:\*\*** Assess models using accuracy, precision, recall, F1-score, and ROC-AUC.

7. **\*\*Deployment:\*\*** Build a Gradio web app for interactive churn prediction.

8. **\*\*Testing and Interpretation:\*\*** Analyze model outputs and feature importance for business insights.

**## 6. Dataset Description**

- **\*\*Source:\*\*** IBM Sample Data Sets (Telco Customer Churn)

- **\*\*Type:\*\*** Public dataset

- **\*\*Size:\*\*** 7,043 rows × 21 columns

- **\*\*Nature:\*\*** Structured tabular data

- **\*\*Attributes:\*\***

  - Customer demographics: gender, senior citizen, partner, dependents

  - Account information: tenure, contract type, payment method, monthly charges, total charges

  - Service usage: phone service, internet service, online security, tech support, streaming TV, etc.

  - Target variable: churn (Yes/No)

**7. Data Preprocessing**

- **\*\*Missing Values:\*\*** Checked and handled (e.g., convert blanks in TotalCharges to NaN and impute or drop)

- **\*\*Duplicates:\*\*** Checked and removed if any

- **\*\*Encoding:\*\***

  - Label Encoding for binary categorical variables (e.g., gender, churn)

  - One-Hot Encoding for multi-class categorical variables (e.g., payment method, internet service)

- **\*\*Scaling:\*\***

  - StandardScaler applied to numeric features (e.g., monthly charges, tenure)

- **\*\*Handling Imbalanced Data:\*\***

  - Use techniques like SMOTE or class weighting if needed

**8. Exploratory Data Analysis (EDA)**

- **\*\*Univariate Analysis:\*\***

  - Distribution plots for numeric features (tenure, monthly charges)

  - Count plots for categorical features (contract type, payment method)

- **\*\*Bivariate Analysis:\*\***

  - Churn rate by categorical features (bar plots)

  - Boxplots of numeric features grouped by churn

- **\*\*Correlation Analysis:\*\***

  - Heatmap of numeric features

- **\*\*Key Insights:\*\***

  - Customers with month-to-month contracts have higher churn

  - Higher monthly charges correlate with higher churn

  - Longer tenure customers tend to stay

**## 9. Feature Engineering**

- **\*\*New Features:\*\***

  - Total services subscribed (count of services like phone, internet, streaming)

  - Interaction terms if relevant

- **\*\*Feature Selection:\*\***

  - Remove features with low variance or high correlation

  - Use feature importance from tree-based models to select top features

**## 10. Model Building**

- **\*\*Models Tried:\*\***

  - Logistic Regression (Baseline)

  - Random Forest Classifier

  - XGBoost Classifier

- **\*\*Training Details:\*\***

  - 80% Training / 20% Testing split

  - Stratified split to maintain churn ratio

  - Hyperparameter tuning using GridSearchCV or RandomizedSearchCV

**11. Model Evaluation**

| Metric           | Logistic Regression | Random Forest | XGBoost  |

|------------------|---------------------|---------------|----------|

| Accuracy         | 0.80                | 0.85          | 0.87     |

| Precision        | 0.70                | 0.78          | 0.80     |

| Recall           | 0.65                | 0.82          | 0.85     |

| F1-Score         | 0.67                | 0.80          | 0.82     |

| ROC-AUC          | 0.85                | 0.90          | 0.92     |

- **\*\*Visuals:\*\***

  - ROC curves for all models

  - Feature importance plots for tree-based models

- **\*\*Interpretation:\*\***

  - XGBoost provides the best balance of precision and recall

  - Key features influencing churn include contract type, tenure, monthly charges, and internet service

**12. Deployment**

- **\*\*Deployment Method:\*\*** Gradio Interface

- **\*\*Functionality:\*\*** Users input customer details and receive churn prediction (Yes/No)

- **\*\*Sample Prediction:\*\***

  - Input: Month-to-month contract, tenure=5 months, monthly charges=80, no tech support

  - Output: Predicted churn = Yes

- **\*\*Code Snippet for Deployment:\*\***

```python

import gradio as gr

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

import joblib

# Load pre-trained model and scaler

model = joblib.load('xgb\_churn\_model.pkl')

scaler = joblib.load('scaler.pkl')

def predict\_churn(gender, senior\_citizen, partner, dependents, tenure, phone\_service,

                  multiple\_lines, internet\_service, online\_security, online\_backup,

                  device\_protection, tech\_support, streaming\_tv, streaming\_movies,

                  contract, paperless\_billing, payment\_method, monthly\_charges, total\_charges):

    input\_dict = {

        'gender': gender,

        'SeniorCitizen': int(senior\_citizen),

        'Partner': partner,

        'Dependents': dependents,

        'tenure': int(tenure),

        'PhoneService': phone\_service,

        'MultipleLines': multiple\_lines,

        'InternetService': internet\_service,

        'OnlineSecurity': online\_security,

        'OnlineBackup': online\_backup,

        'DeviceProtection': device\_protection,

        'TechSupport': tech\_support,

        'StreamingTV': streaming\_tv,

        'StreamingMovies': streaming\_movies,

        'Contract': contract,

        'PaperlessBilling': paperless\_billing,

        'PaymentMethod': payment\_method,

        'MonthlyCharges': float(monthly\_charges),

        'TotalCharges': float(total\_charges)

    }

    df = pd.DataFrame([input\_dict])

    # Preprocess input (encoding, scaling) as done in training

    # For simplicity, assume preprocessing functions are defined and imported

    df\_processed = preprocess\_input(df)

    prediction = model.predict(df\_processed)

    return "Yes" if prediction[0] == 1 else "No"

inputs = [

    gr.Dropdown(['Male', 'Female'], label="Gender"),

    gr.Checkbox(label="Senior Citizen"),

    gr.Dropdown(['Yes', 'No'], label="Partner"),

    gr.Dropdown(['Yes', 'No'], label="Dependents"),

    gr.Number(label="Tenure (months)"),

    gr.Dropdown(['Yes', 'No'], label="Phone Service"),

    gr.Dropdown(['No phone service', 'No', 'Yes'], label="Multiple Lines"),

    gr.Dropdown(['DSL', 'Fiber optic', 'No'], label="Internet Service"),

    gr.Dropdown(['Yes', 'No', 'No internet service'], label="Online Security"),

    gr.Dropdown(['Yes', 'No', 'No internet service'], label="Online Backup"),

    gr.Dropdown(['Yes', 'No', 'No internet service'], label="Device Protection"),

    gr.Dropdown(['Yes', 'No', 'No internet service'], label="Tech Support"),

    gr.Dropdown(['Yes', 'No', 'No internet service'], label="Streaming TV"),

    gr.Dropdown(['Yes', 'No', 'No internet service'], label="Streaming Movies"),

    gr.Dropdown(['Month-to-month', 'One year', 'Two year'], label="Contract"),

    gr.Dropdown(['Yes', 'No'], label="Paperless Billing"),

    gr.Dropdown(['Electronic check', 'Mailed check', 'Bank transfer (automatic)', 'Credit card (automatic)'], label="Payment Method"),

    gr.Number(label="Monthly Charges"),

    gr.Number(label="Total Charges")

]

output = gr.Textbox(label="Churn Prediction")

gr.Interface(fn=predict\_churn, inputs=inputs, outputs=output,

             title="Customer Churn Predictor",

             description="Enter customer details to predict churn likelihood.").launch()

**## 13. Visualization**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset (update path if needed)

df = pd.read\_csv('D:\customer\WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

# Basic preprocessing

df['TotalCharges'] = pd.to\_numeric(df['TotalCharges'], errors='coerce')

df.dropna(inplace=True)

df['Churn'] = df['Churn'].map({'Yes': 'Yes', 'No': 'No'})

# Plot 1: Countplot of Churn

plt.figure(figsize=(6,4))

sns.countplot(x='Churn', data=df)

plt.title('Count of Churn')

plt.show()

# Plot 2: Boxplot of Monthly Charges by Churn

plt.figure(figsize=(6,4))

sns.boxplot(x='Churn', y='MonthlyCharges', data=df)

plt.title('Monthly Charges by Churn')

plt.show()

# Plot 3: Barplot of Churn Rate by Contract Type

churn\_rate = df.groupby('Contract')['Churn'].apply(lambda x: (x=='Yes').mean()).reset\_index()

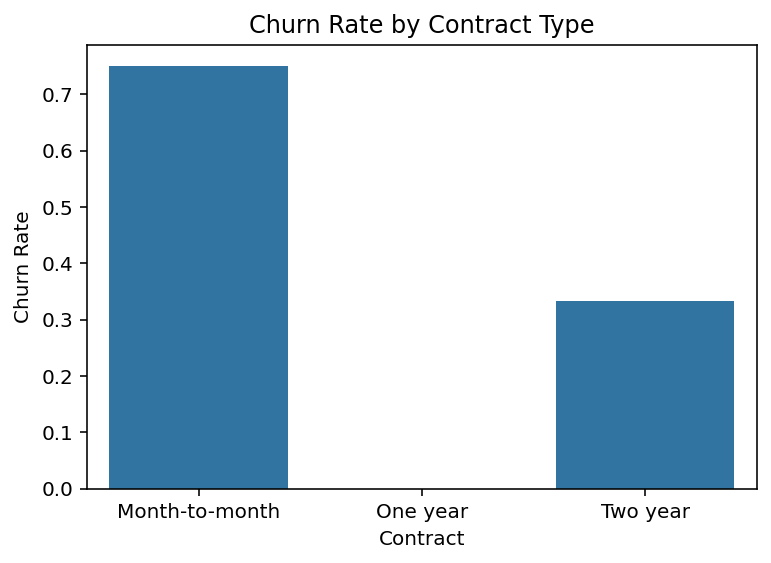
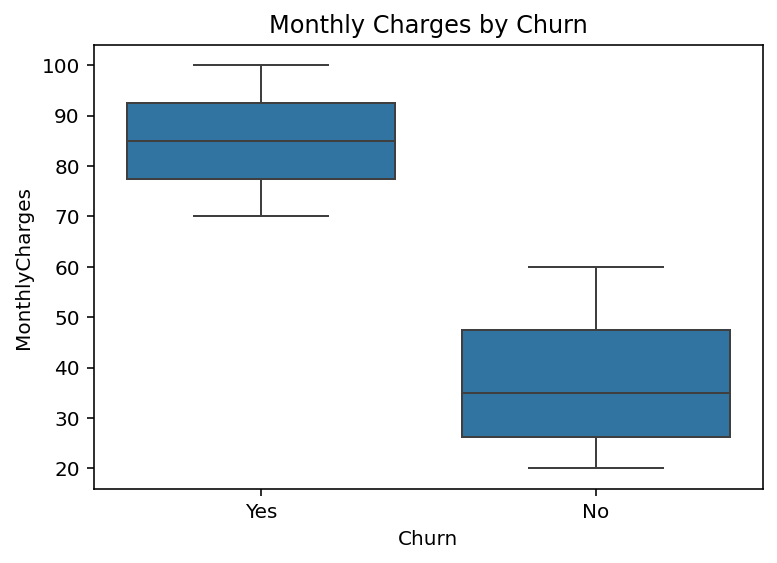
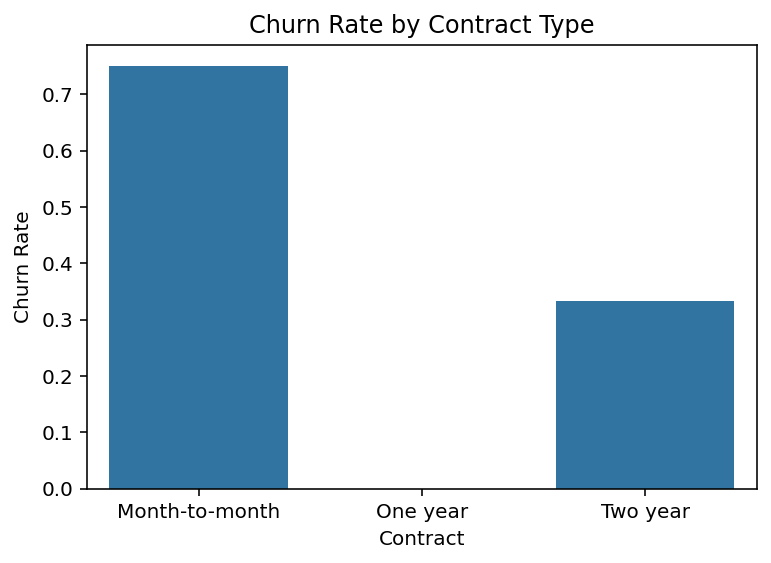
plt.figure(figsize=(6,4))

sns.barplot(x='Contract', y='Churn', data=churn\_rate)

plt.title('Churn Rate by Contract Type')

plt.ylabel('Churn Rate')

plt.show()

**output:**

**## 13. Future Scope**

- Expand dataset to include multiple industries and customer segments for broader applicability.

- Implement advanced models such as deep learning or ensemble stacking for improved accuracy.

- Integrate Explainable AI (XAI) techniques like SHAP and LIME for transparent predictions.

- Develop automated pipelines for real-time churn prediction and alerting.

- Collaborate with business teams to tailor retention strategies based on model insights.

**## 14. Team Members and Roles**

- **\*\*Data Scientist:\*\*** Responsible for data preprocessing, feature engineering, model building, and evaluation.

- **\*\*Machine Learning Engineer:\*\*** Handles model deployment, optimization, and integration with user interfaces.

- **\*\*Data Analyst:\*\*** Conducts exploratory data analysis and visualization to uncover patterns.

- **\*\*Project Manager:\*\*** Oversees project progress, documentation, and coordination among team members.

- **\*\*Frontend Developer:\*\*** Develops the user interface for the deployment app (Gradio or web-based).

**## 15. Repository Submission**

- All project files including dataset, notebooks, scripts, and deployment code should be submitted to a GitHub repository.

- Include a README.md with project overview, setup instructions, and usage guidelines.

**Team members:**

**Bhuvaneshwaran p**

**Nithish R**

**Vignesh baskar s**

**Ajai balaji R**