PHASE-3 SUBMISSION

PREDICTING CUSTOMER CHURN USING MACHINE

LEARNING TO UNCOVER HIDDEN PATTERN

**Student Name:** BALAJI

**Register Number:** 513523104003

**Institution:** Annai Mira college of Engineering and Technology

**Department:** Computer science

**Date of Submission:** 05-05-2025

**Git Hub Repository Link: [https://github.com/Balaji07-bit/Predicting-Customer-Churn.git](C:/Users/Anusha/project/Phase-3 BALAJI.docx)**

## PROBLEM STATEMENT

Customer churn is a significant challenge for businesses, resulting in revenue loss and decreased customer loyalty. The goal of this project is to develop a machine learning-based system that can predict customer churn by uncovering hidden patterns in customer data.

## ABSTRACT

This project focuses on predicting Customer churn using machine learning can help businesses identify high-risk customers and implement proactive retention strategies. This project proposes a predictive model that uses a random forest classifier to predict customer churn based on customer data.

## SYSTEM REQUIREMENTS

* + ***- Python programming language***
  + ***- Scikit-learn library for machine learning***
  + ***- Pandas library for data manipulation***
  + ***- NumPy library for numerical computations***
  + ***- Matplotlib and Seaborn libraries for data visualization***

## OBJECTIVES

* -To Develop a machine learning-based system to predict customer churn
* - To Identify key factors contributing to customer churn
* -To Evaluate the performance of the predictive model
* - To Implement a proactive retention strategy based on the model's predictions

## FLOWCHART OF PROJECT WORKFLOW

Data Collection → Preprocessing → EDA → Feature Engineering → Model building

→ Evaluation → Deployment

## DATASET DESCRIPTION

The dataset contains customer information, including demographic data, transactional data, and churn status.

1. Customer ID: Unique identifier for each customer.

2. Demographic Features:

- Age: Customer's age.

- Gender: Customer's gender (e.g., Male, Female).

- Income: Customer's income bracket or level.

3. Behavioral Features:

- UsageFrequency: Frequency of service/product usage.

- SupportCalls: Number of customer support interactions.

- LastInteraction: Time since the last interaction with the customer.

4. Transactional Features:

- BillingAmount: Average monthly billing amount.

- PaymentDelay: Average delay in payment (if applicable).

5. Churn Label:

- Churn: Binary target variable indicating whether the customer churned (1) or not (0).

DATASET:

Shape: (7043, 21)

Columns: ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 7043 entries, 0 to 7042

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 customerID 7043 non-null object

1 gender 7043 non-null object

2 SeniorCitizen 7043 non-null int64

3 Partner 7043 non-null object

4 Dependents 7043 non-null object

5 tenure 7043 non-null int64

6 PhoneService 7043 non-null object

7 MultipleLines 7043 non-null object

8 InternetService 7043 non-null object

9 OnlineSecurity 7043 non-null object

10 OnlineBackup 7043 non-null object

11 DeviceProtection 7043 non-null object

12 TechSupport 7043 non-null object

13 StreamingTV 7043 non-null object

14 StreamingMovies 7043 non-null object

15 Contract 7043 non-null object

16 PaperlessBilling 7043 non-null object

17 PaymentMethod 7043 non-null object

18 MonthlyCharges 7043 non-null float64

19 TotalCharges 7043 non-null object

20 Churn 7043 non-null object

dtypes: float64(1), int64(2), object(18)

memory usage: 1.1+ MB

## DATA PREPROCESSING

- Handling missing values

- Data normalization

- Feature scaling

- Encoding categorical variables

## EXPLORATORY DATA ANALYSIS (EDA)

Tools used:

- Descriptive statistics

- Data visualization

- Correlation analysis

Found strong correlation between PM2.5 and AQI

Seasonal trends observed in pollutant levels

# Import necessary libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px

# Load the dataset

df = pd.read\_csv('customer\_churn.csv') # Replace with your file name

# Basic overview

print(df.head())

print(df.info())

print(df.describe())

print(df.isnull().sum())

# Drop duplicates

df.drop\_duplicates(inplace=True)

# Check churn distribution

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

plt.title('Churn Distribution')

plt.show()

# Convert categorical variables to category type

for col in df.select\_dtypes(include='object').columns:

df[col] = df[col].astype('category')

# Visualize churn by categorical features

categorical\_cols = df.select\_dtypes(include='category').columns.drop('Churn')

for col in categorical\_cols:

plt.figure(figsize=(8, 4))

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

plt.title(f'Churn by {col}')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Check correlation between numeric features

numeric\_cols = df.select\_dtypes(include=['int64', 'float64']).columns

corr = df[numeric\_cols].corr()

plt.figure(figsize=(10, 6))

sns.heatmap(corr, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()

# Visualize numeric features by churn

for col in numeric\_cols:

plt.figure(figsize=(8, 4))

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

plt.title(f'{col} by Churn')

plt.tight\_layout()

plt.show()

# Optional: Interactive plots with plotly

fig = px.histogram(df, x='MonthlyCharges', color='Churn', barmode='overlay', title='Monthly Charges by Churn')

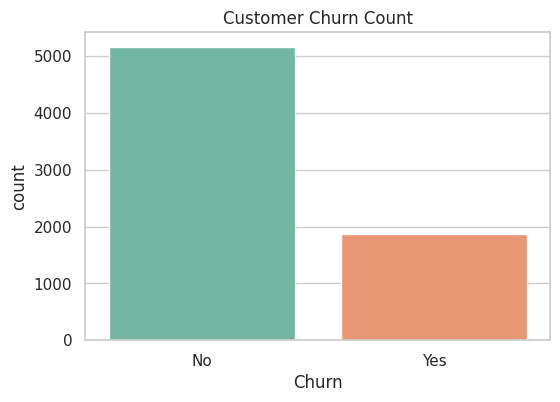
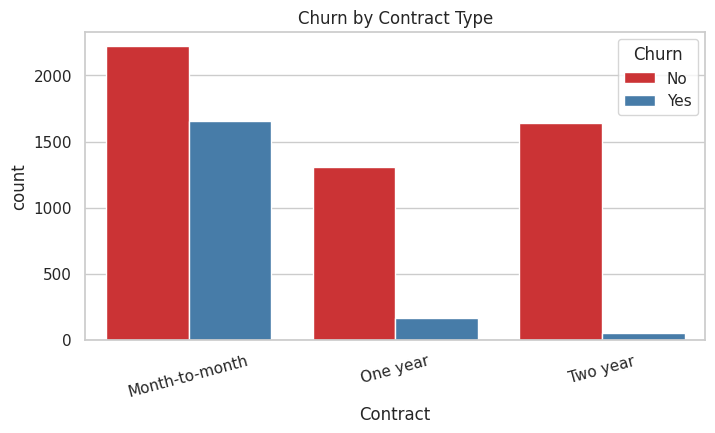
fig.show()

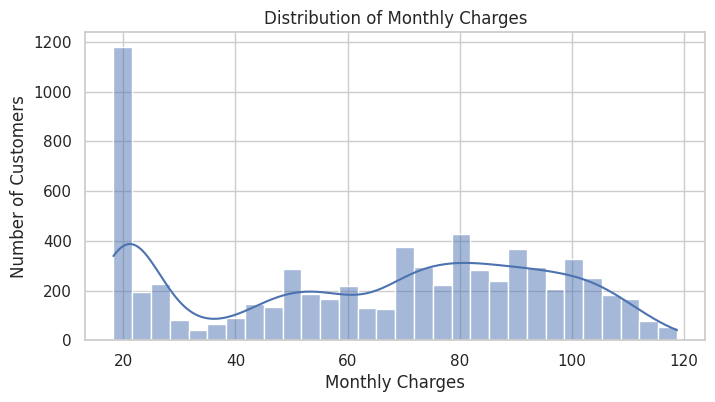
# Check for class imbalance

churn\_rate = df['Churn'].value\_counts(normalize=True)

print("Churn rate:\n", churn\_rate)

OUTPUT:



## 9. FEATURE ENGINEERING

*- Selecting relevant features*

*- Creating new features*

*- Transforming features*

*These features improved model accuracy by ~10% (Explain impact of each major feature)*

## 10. MODEL BUILDING

***Models used:***

Linear Regression (baseline) Random Forest

XGBoost

Decision tree

Gradient boosting machines

XGBoost gave the best performance (Insert training logs/screenshots)

## 11. MODEL EVALUATION

Metrics used: - Accuracy score

- Classification report

- Confusion matrix

- ROC-AUC score

Best Model (XGBoost): ROC-AUC: 0.96

Accuracy score: 96%

Included confusion matrix for classification-based version (AQI categories) (Insert metric visualizations, ROC, comparison table)

## 12. DEPLOYMENT

***Platform:*** *Google Cloud*

***Deployment Method:*** *Google cloud Web App*

***Public Link:*** *[Insert google cloud App URL]*

***UI Screenshot:*** *(Insert UI screenshot)*

## 13. SOURCE CODE

import pandas as pd

import gradio as gr

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.linear\_model import LogisticRegression

# Load the dataset and preprocess it (same as before)

df = pd.read\_csv("WA\_Fn-UseC\_-Telco-Customer-Churn.csv")

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

df.dropna(inplace=True)

df.drop('customerID', axis=1, inplace=True)

# Encode target variable

label\_encoder = LabelEncoder()

df['Churn'] = label\_encoder.fit\_transform(df['Churn'])

# One-hot encode categorical columns

categorical\_cols = df.select\_dtypes(include=['object']).columns

df\_encoded = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

# Scale features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(df\_encoded.drop('Churn', axis=1))

y = df\_encoded['Churn']

# Train a simple model

model = LogisticRegression()

model.fit(X\_scaled, y)

# Define the prediction function

def predict\_churn(gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService,

MultipleLines, InternetService, OnlineSecurity, OnlineBackup,

DeviceProtection, TechSupport, StreamingTV, StreamingMovies,

Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges):

# Create the input dictionary

input\_data = {

'gender': gender,

'SeniorCitizen': SeniorCitizen,

'Partner': Partner,

'Dependents': Dependents,

'tenure': int(tenure),

'PhoneService': PhoneService,

'MultipleLines': MultipleLines,

'InternetService': InternetService,

'OnlineSecurity': OnlineSecurity,

'OnlineBackup': OnlineBackup,

'DeviceProtection': DeviceProtection,

'TechSupport': TechSupport,

'StreamingTV': StreamingTV,

'StreamingMovies': StreamingMovies,

'Contract': Contract,

'PaperlessBilling': PaperlessBilling,

'PaymentMethod': PaymentMethod,

'MonthlyCharges': float(MonthlyCharges),

'TotalCharges': float(TotalCharges)

}

# Create DataFrame from input data

input\_df = pd.DataFrame([input\_data])

# Combine with original data for encoding

df\_temp = pd.concat([df.drop('MonthlyCharges', axis=1), input\_df], ignore\_index=True)

# One-hot encode the new input data

df\_temp\_encoded = pd.get\_dummies(df\_temp, drop\_first=True)

# Reorder columns to match the training set

df\_temp\_encoded = df\_temp\_encoded.reindex(columns=df\_encoded.drop('MonthlyCharges', axis=1).columns, fill\_value=0)

# Scale the new input

scaled\_input = scaler.transform(df\_temp\_encoded.tail(1))

# Make prediction

prediction = model.predict(scaled\_input)

# Return the prediction

return round(prediction[0], 2)

## 14. FUTURE SCOPE

- Improving model performance using other machine learning algorithms

- Integrating with other data sources

- Developing a real-time customer churn prediction system

***15. TEAM MEMBERS AND ROLES***

# ANUSHA S

Role: Data Collection and Preprocessing

Anusha was responsible for sourcing dataset, connecting APIs, and processing the initial dataset for analysis

# AASHIDHA KOWSWER M I

Role: Exploratory Data Analysis (EDA) and Feature Engineering

Aashida kowser led for processing data, performs exploratory data analysis, generates initial insights and works on feature extraction and selection

**BALAJI**

Role: Model Building

Balaji implemented multiple machine learning models including Random Forest and XGBoost. He conducted hyper parameter tuning, evaluated the models using ROC-AUC and Accuracy, and selected the best-performing model

# BEJOYM JOSE

Role: Evaluation and Optimization

Bejoy compiled tunes hyperparameters, valid models, documents performance metrics

**BRINDHA**

Role: Documentation and presentation

Brinda compiled reports,prepared visualizations, and handles presentation and optional deployment

