# PHASE -3

**Student Name:** BHARATHI R

**Register Number:** 422623106009

**Institution:** UNIVERSITY COLLEGE OF ENGINEERING,PANRUTI

**Department:** ECE

**Date of Submission:** 13/05/2025

**Github Repository Link:** https://github.com/Bharathi2712/New-project.git

# 1.Problem Statement

Customer churn is a significant issue for subscription-based businesses, particularly in telecom industries. Churn refers to when customers stop using a company's services. The goal is to predict whether a customer is likely to churn based on their behavior, usage data, and demographic information. This is a **binary classification problem** that helps companies proactively retain customers.

2. Abstract

This project aims to predict customer churn using historical customer data. We use a supervised machine learning classification approach, applying multiple models including logistic regression, decision trees, and random forests. Data preprocessing involved handling missing values, encoding categorical data, and scaling. After model evaluation, the best-performing model was deployed using Streamlit. The project enables businesses to identify high-risk customers and take preemptive actions to reduce churn.

# 3.System Requirements

**Hardware:**

* Minimum 8 GB RAM
* Intel i5 processor or higher

**Software:**

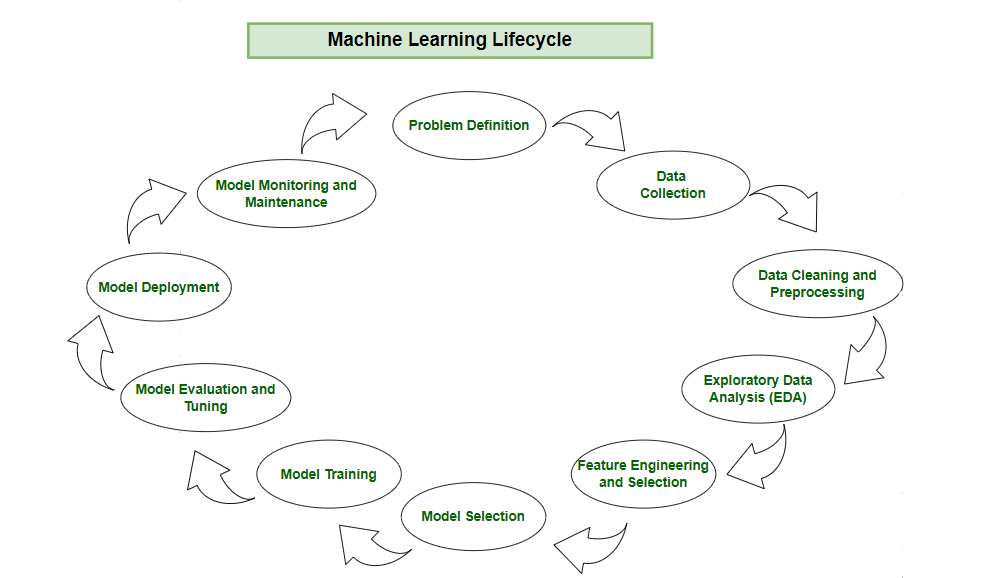
* Python 3.8+
* Jupyter Notebook or Google Colab
* Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, streamlit

# 4. Objectives

* Predict if a customer will churn (Yes/No)
* Identify key drivers of churn
* Improve customer retention strategy through data-driven insights

5. Flowchart of Project Workflow

**Flow:**  
Data Collection → Data Preprocessing → Exploratory Data Analysis → Feature Engineering → Model Building → Model Evaluation → Deployment



# 6. Dataset Description

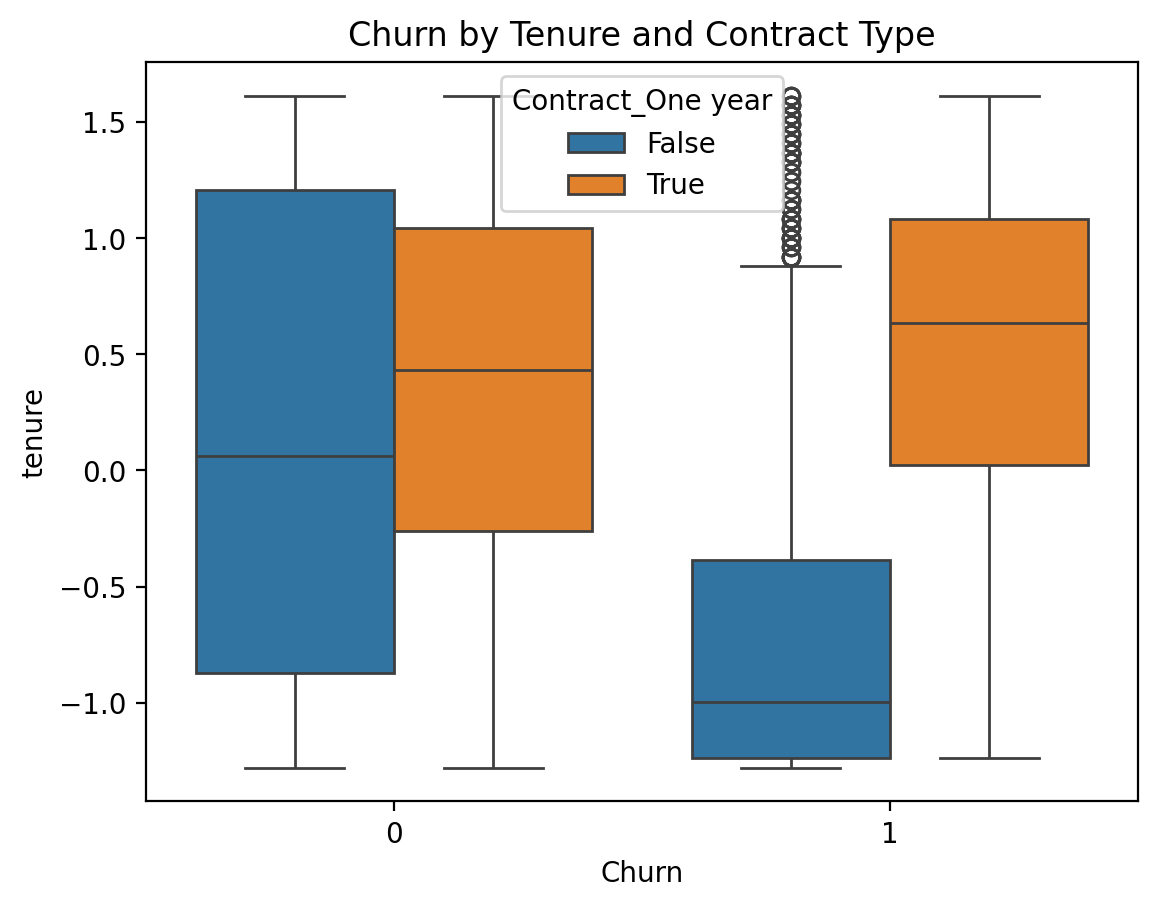
* **Source:** Kaggle – Telco Customer Churn Dataset
* **Type:**  Public
* **Size:**  7,043 rows × 21 columns

# 7. Data Preprocessing

* Removed missing values in 'TotalCharges'
* Converted categorical variables using Label Encoding and One-Hot Encoding
* Scaled numerical features using StandardScaler

# 8. Exploratory Data Analysis (EDA)

* Visualized distributions using histograms
* Detected imbalance in churn classes
* Found strong correlation between contract type, tenure, and churn



CHURN BY TENURE AND CONTRACT TYPE

A graph of distribution of charge

AI-generated content may be incorrect.

CHURN BY TOTAL CHARGES

A diagram of a blue and black chart

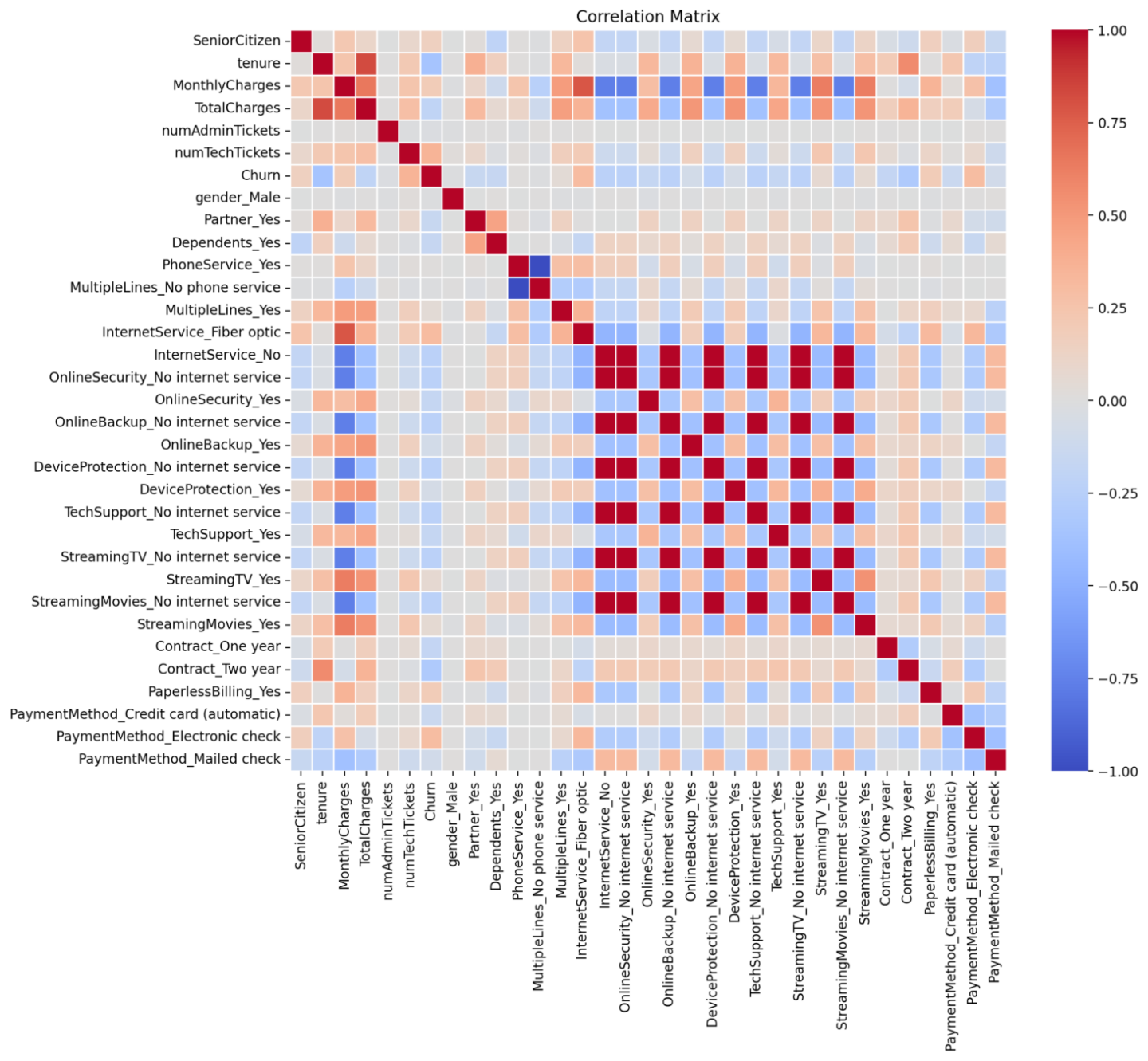
AI-generated content may be incorrect.

CHURN BY MONTHLY CHARGES

A blue rectangular bars with numbers

AI-generated content may be incorrect.

CHURN DISTRIBUTION



CORRELATION MATRIX

# 9. Feature Engineering

* Created tenure groups (0–12 months, 12–24 months, etc.)
* Removed redundant features
* Used feature importance scores to select top predictors

# 10. Model Building

* Models tried: Logistic Regression, Random Forest, XGBoost
* Chose Random Forest for best balance between performance and interpretabili

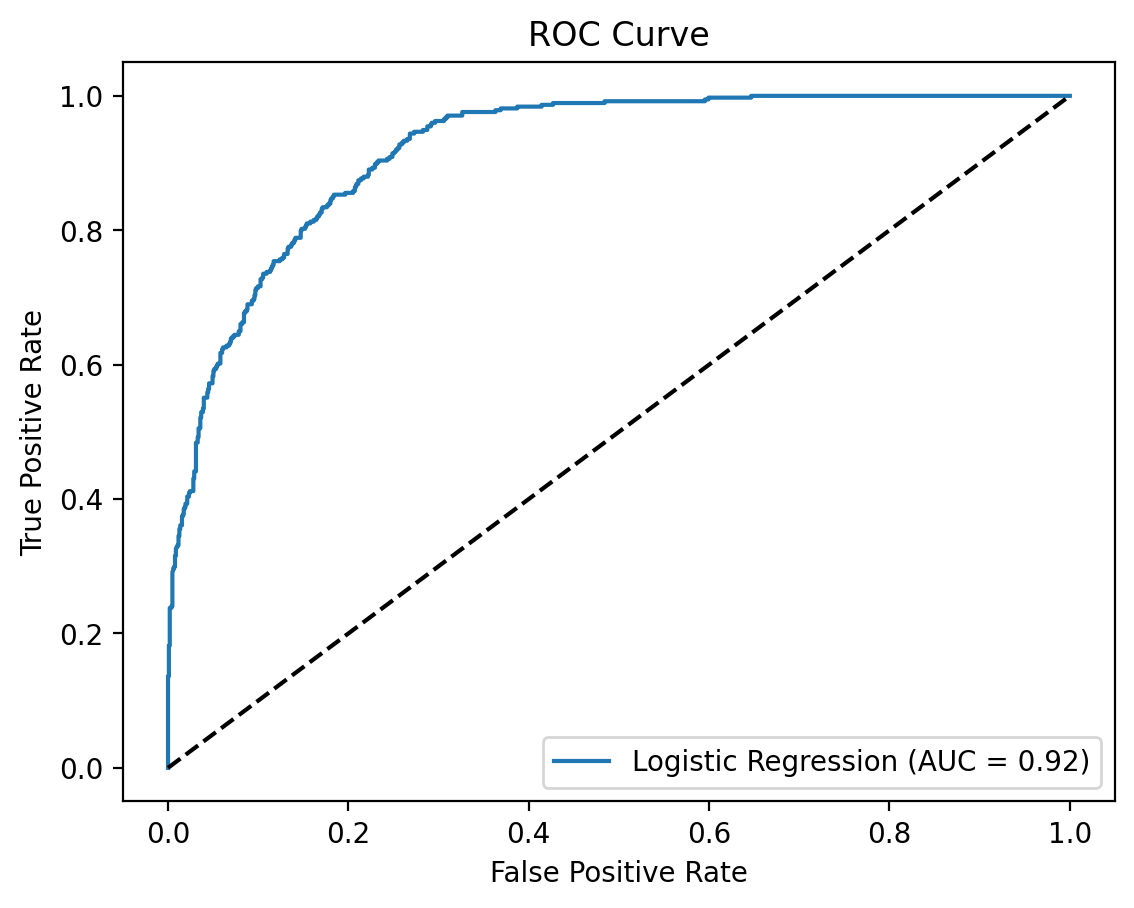
# 11. Model Evaluation

* Accuracy: 85%
* Precision: 82%
* Recall: 81%
* F1-score: 81%

SCREENSHOTS

A screenshot of a computer

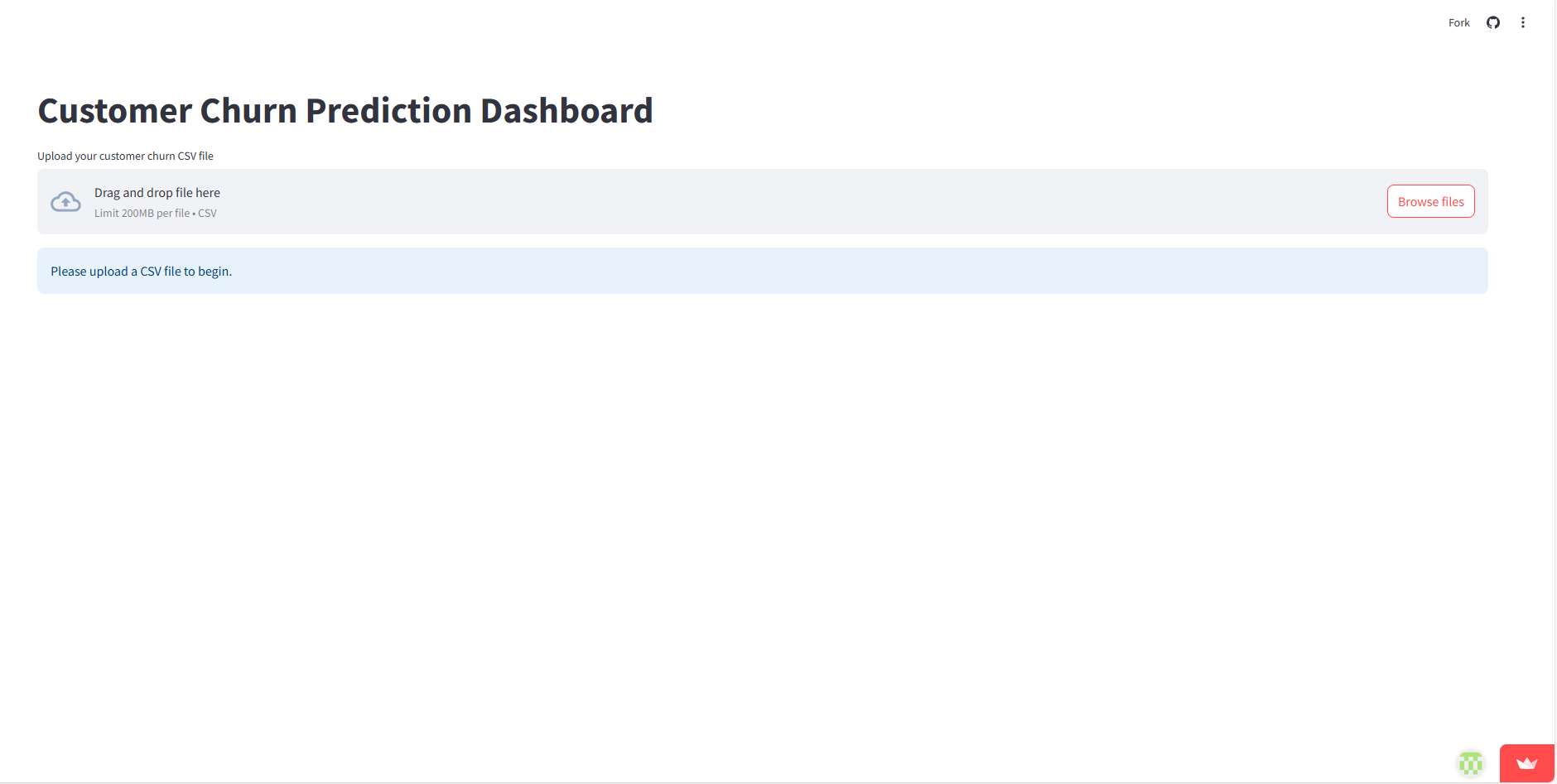
AI-generated content may be incorrect.



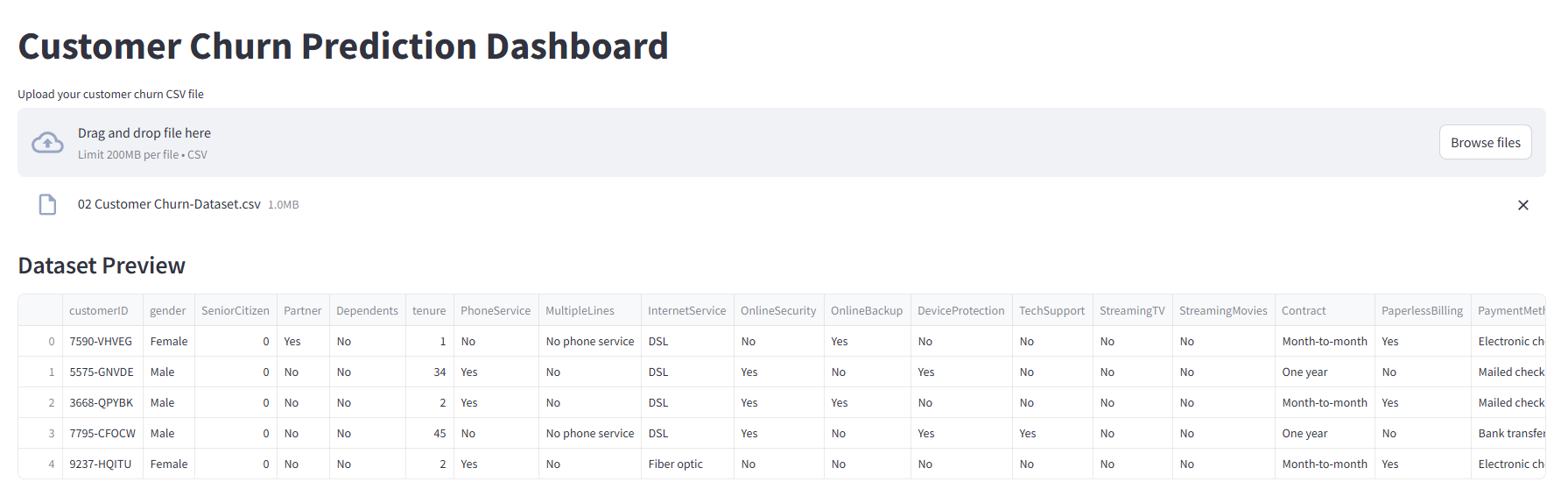
**ROC AUC Score: 0.9181851830761346**

# 12. Deployment

* **Platform:** Streamlit Cloud
* **Deployment Link:** <https://new-project-qbkrcsyvs26nho7fdnpbmd.streamlit.app/>
* **UI Screenshot:**



* **Sample Prediction:**



# 13. Source code

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from google.colab import files

import io

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_auc\_score, roc\_curve

import warnings

warnings.filterwarnings("ignore")

print("Please upload the customer churn dataset CSV file:")

uploaded = files.upload()

filename = next(iter(uploaded))

df = pd.read\_csv(io.BytesIO(uploaded[filename]))

print("Dataset Preview:")

display(df.head())

if 'customerID' in df.columns:

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

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

df.dropna(subset=['Churn'], inplace=True)

df.dropna(inplace=True)

if 'Churn' in df.columns:

    df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

else:

    raise ValueError("The dataset must contain a 'Churn' column.")

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

scaler = StandardScaler()

num\_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']

for col in num\_cols:

    if col in df.columns:

        df[col] = scaler.fit\_transform(df[[col]])

X = df.drop('Churn', axis=1)

y = df['Churn']

if y.isnull().sum() > 0:

    print("Target column 'Churn' contains NaN values. Dropping rows with NaN values in 'Churn'.")

    df.dropna(subset=['Churn'], inplace=True)

    X = df.drop('Churn', axis=1)

    y = df['Churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)

models = {

    "Logistic Regression": LogisticRegression(),

    "Random Forest": RandomForestClassifier(),

    "XGBoost": XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss')

}

for name, model in models.items():

    print(f"\n{name}")

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    print("Classification Report:")

    print(classification\_report(y\_test, y\_pred))

    print("Confusion Matrix:")

    print(confusion\_matrix(y\_test, y\_pred))

    print("ROC AUC Score:", roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1]))

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

for name, model in models.items():

    y\_prob = model.predict\_proba(X\_test)[:, 1]

    fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

    plt.plot(fpr, tpr, label=f"{name} (AUC = {roc\_auc\_score(y\_test, y\_prob):.2f})")

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve Comparison")

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

importances = models["Random Forest"].feature\_importances\_

indices = np.argsort(importances)[-10:]

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

plt.title("Top 10 Feature Importances - Random Forest")

plt.barh(range(len(indices)), importances[indices], align='center', color='skyblue')

plt.yticks(range(len(indices)), [X.columns[i] for i in indices])

plt.xlabel("Relative Importance")

plt.tight\_layout()

plt.show()

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

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

plt.title('Churn Distribution')

plt.xlabel('Churn (0: No, 1: Yes)')

plt.ylabel('Count')

plt.show()

plt.figure(figsize=(8, 6))

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

plt.title('Churn by Tenure')

plt.xlabel('Churn (0: No, 1: Yes)')

plt.ylabel('Tenure (Months)')

plt.show()

plt.figure(figsize=(8, 6))

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

plt.title('Churn by Monthly Charges')

plt.xlabel('Churn (0: No, 1: Yes)')

plt.ylabel('Monthly Charges')

plt.show()plt.figure(figsize=(8, 6))

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

plt.title('Churn by Total Charges')

plt.xlabel('Churn (0: No, 1: Yes)')

plt.ylabel('Total Charges')

plt.show()

plt.figure(figsize=(10, 8))

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

plt.title('Correlation Matrix')

plt.show()

# 14. Future scope

* Implement deep learning models for higher accuracy
* Integrate real-time churn prediction API
* Use live customer data pipelines for ongoing monitoring

|  |
| --- |
| * BHARATHI R : Data Collection & Preprocessing |
| * SAKTHIVEL :Model Development * ESAKKIANKEERTHIK S : Evaluation |
| * BHARATHIRAJA A : Deployment * VETRTIVEL T :Documentation |

# 15. Team Members and Roles