**Student Name: Nidhidharshini.K**

**Register Number:** 712523121018

**Institution:** PPG Institute Of Technology **Department:** BIOMEDICAL ENGINEERING **Date of Submission:** 16-05-2025

**Github Repository Link: https://github.com/nidhidharshinik/predicting-customer-churn-using-machine-learning-to-uncover-hidden-pattern.git**

# 

# Problem Statement

Customer churn is a major business problem. This project uses machine learning classification techniques to predict churn based on demographic and behavioral data. The model helps businesses take proactive steps to retain customers by identifying key churn indicators, thus reducing revenue loss.

# Abstract

This project aims to build a predictive model to identify customers likely to churn from a bank. We use a publicly available dataset containing customer information and banking behavior. The dataset is preprocessed and explored using EDA techniques. Multiple machine learning models, including Random Forest, Logistic Regression, XGBoost, and SVM, are trained and evaluated. The best-performing model is interpreted using SHAP to understand feature importance. Finally, a Streamlit web application is built for interactive predictions and insights. This enables real-time decision-making and business retention strategies.

# System Requirements

**Hardware:**

* RAM: Minimum 4 GB
* CPU: Intel i3/i5 or equivalent

**Software:**

* Python 3.8 or above
* Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, shap, streamlit
* IDE: Jupyter Notebook / Google Colab / VS Code
* Deployment: Streamlit Cloud

# Objectives

 Predict whether a customer will churn based on behavioral and demographic attributes.

 Evaluate multiple machine learning models to find the best performer.

 Interpret model predictions using SHAP values.

 Deploy the entire solution as an interactive Streamlit app.

 Provide actionable insights for customer retention.

# Flowchart of Project Workflow

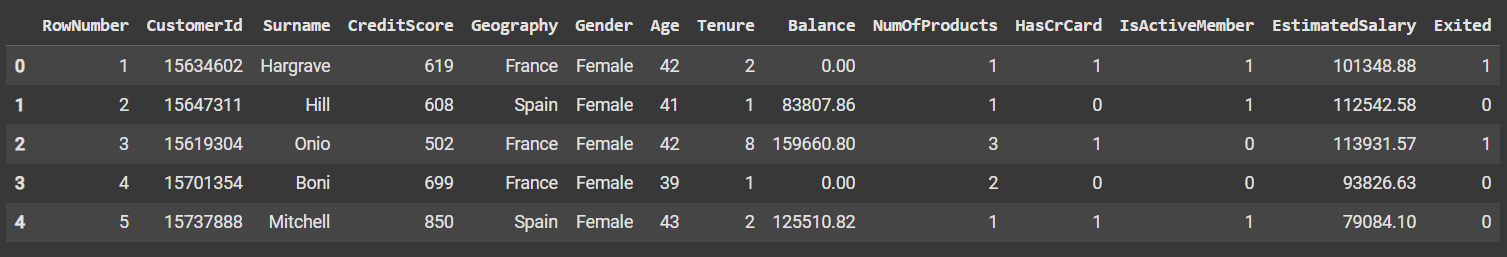


# Dataset Description

 **Source:** Kaggle (uploaded as churn.csv)

 **Type:** Public dataset

 **Size:** 10,000 rows × 14 columns

**Df.Head() :**

# Data Preprocessing

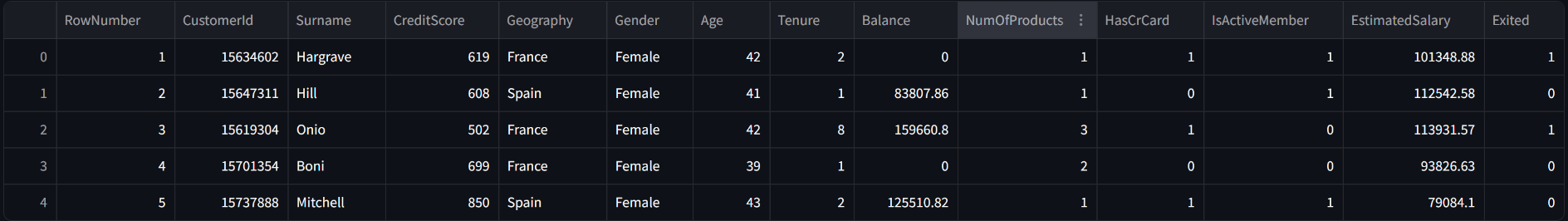
 **Dropped Columns:** RowNumber, CustomerId, Surname

 **Encoding:** Label encoding for Geography, Gender

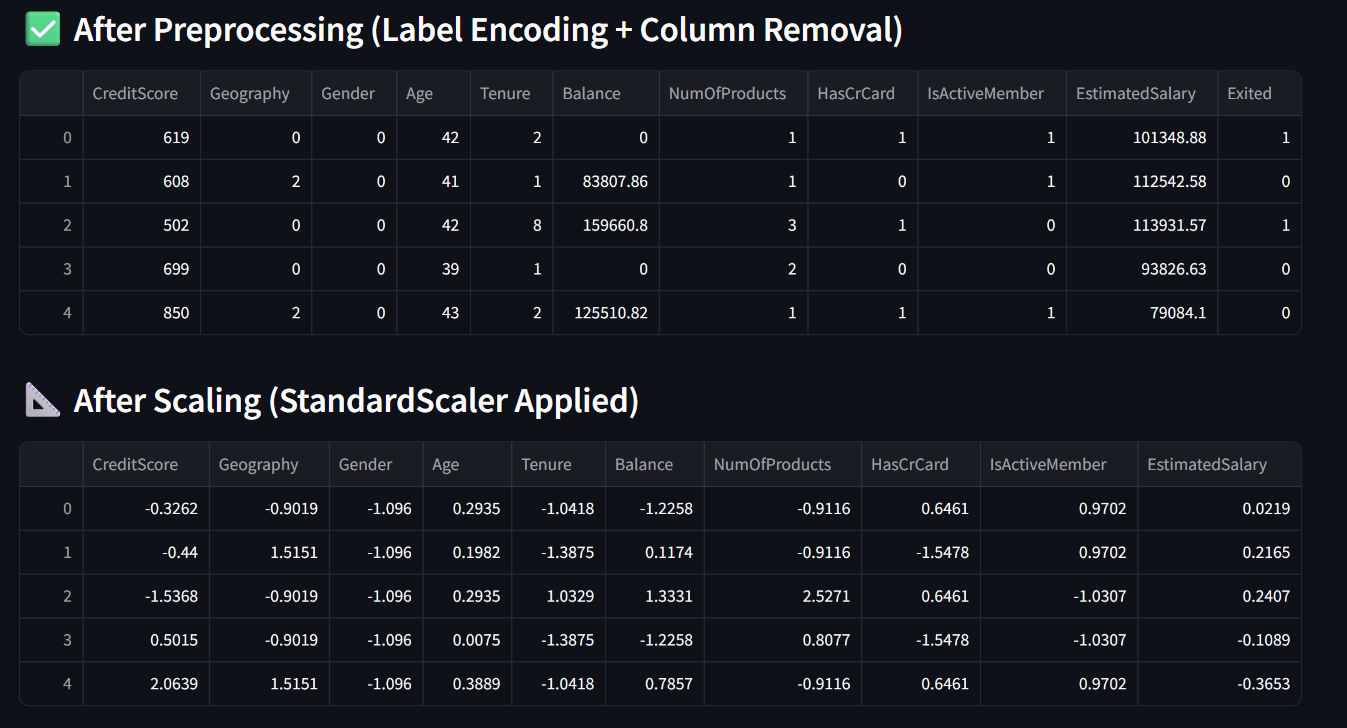
 **Scaling:** StandardScaler applied to numerical features

 **Missing Values & Duplicates:** None found

**Before Transformation:**



**After Transformation:**



# Exploratory Data Analysis (EDA)

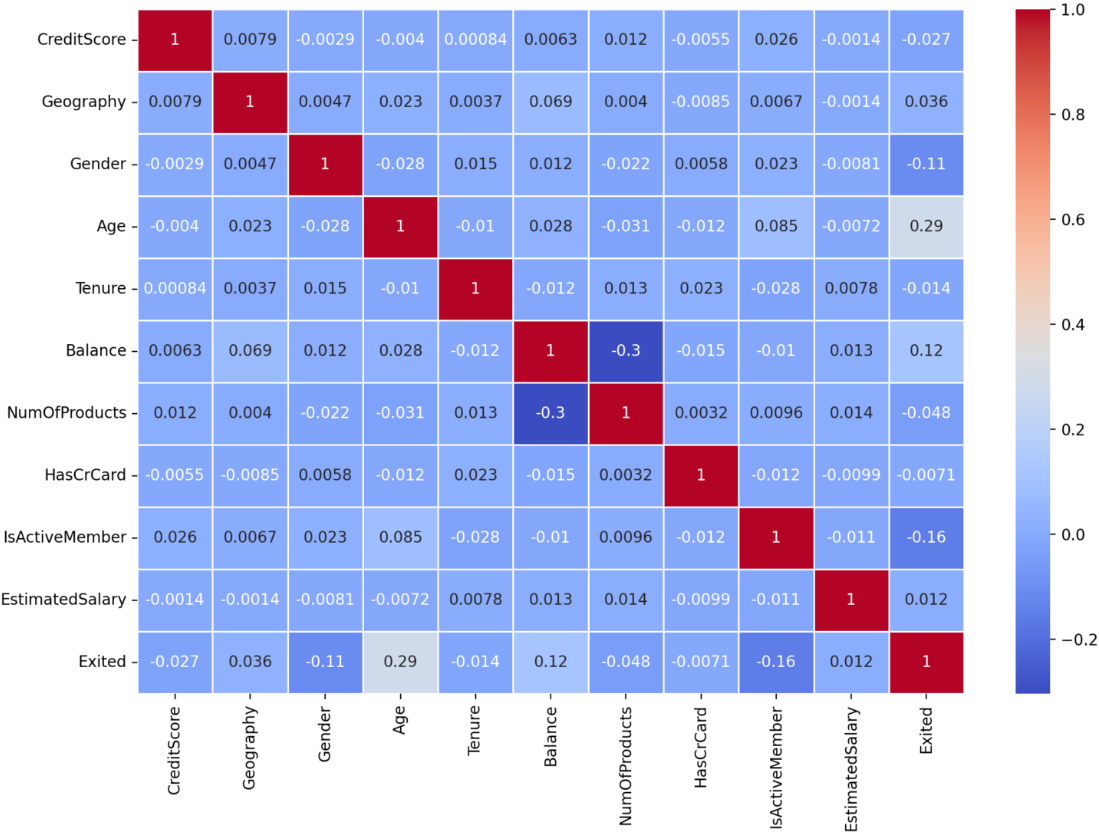
**Visualizations Used:**

* + Correlation Heatmap
  + Countplot of target variable
  + Histogram of features

**Key Insights:**

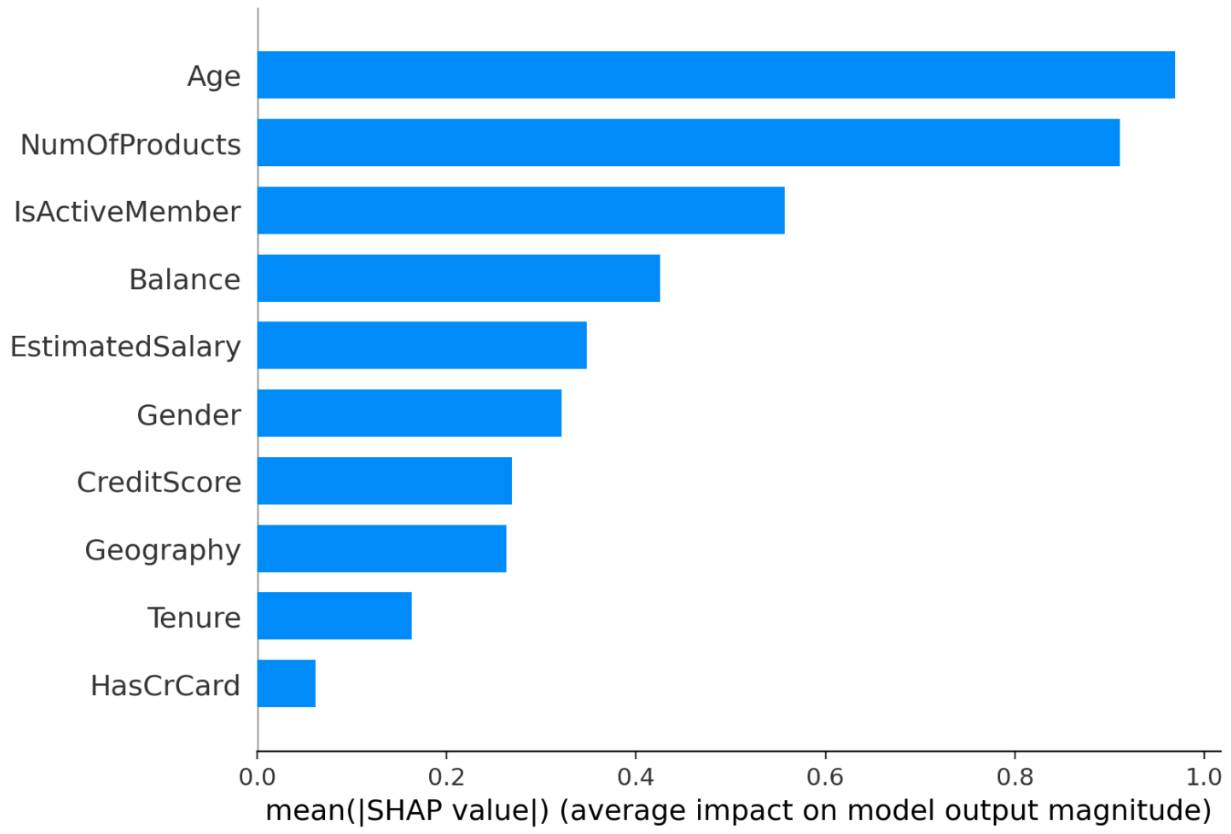
* Balance and age show strong correlation with churn.
* Churn rate is about 20%.
* Senior citizens and inactive members are more likely to churn.

**Heatmap:**



**SHAP Explainability For XGBoost:**





# Feature Engineering

* Encoded categorical variables
* Scaled features to normalize ranges
* Selected features based on correlation and model performance

**Impact:**

Proper encoding and scaling enhanced model accuracy, especially for algorithms sensitive to feature scale like SVM and Logistic Regression.

# Model Building

**Models Tried:**

* + Logistic Regression
  + Random Forest
  + XGBoost
  + Support Vector Machine (SVM)
  + K-Nearest Neighbors (KNN)

**Training Details:**

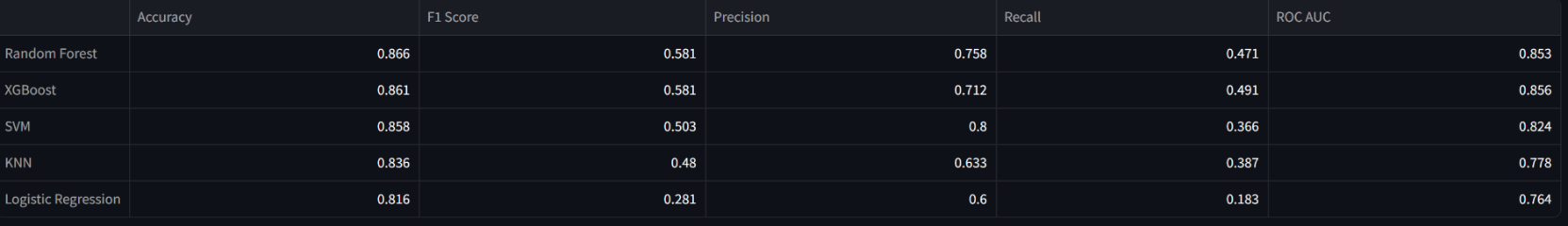
* Train/test split = 80/20
* Hyperparameters tuned minimally for speed

# Model Evaluation

**Metrics:**

* Accuracy
* F1 Score
* Precision
* Recall
* ROC AUC

**Comparison Table:**



# Deployment

 **Platform:** Streamlit Cloud

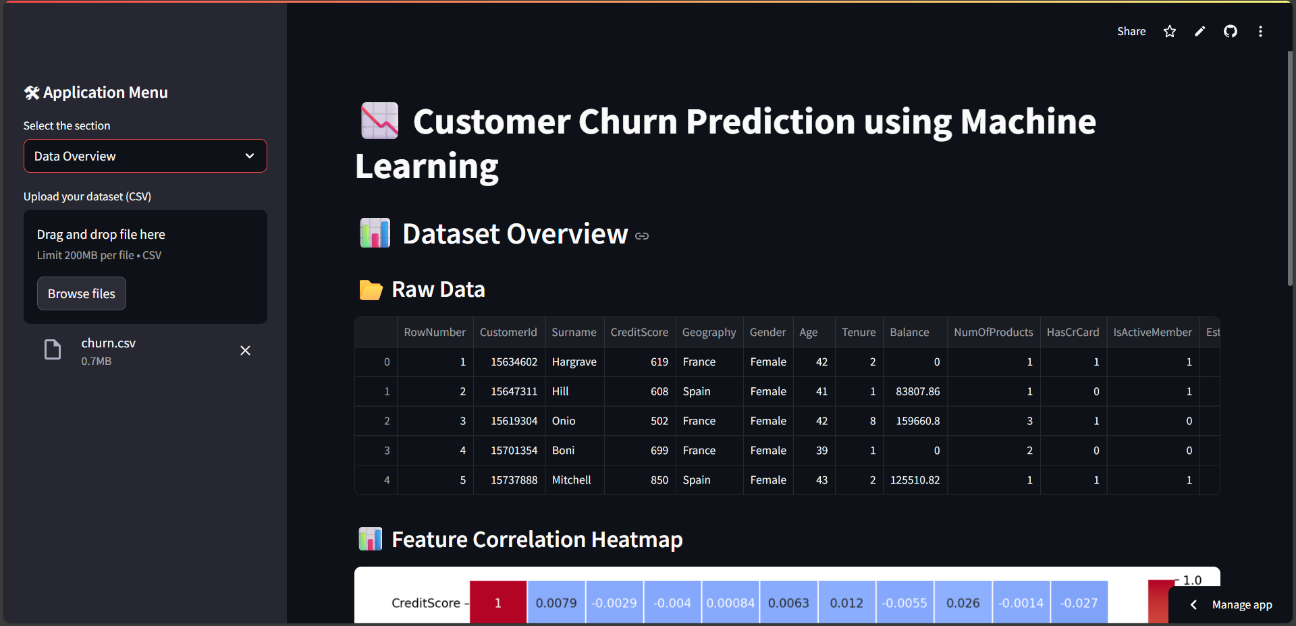
 **Method:** Streamlit Python script

 **Link:** [Customer Churn Predictor · Streamlit](https://nanmudhalvan-project2-adfcuugvxxgel5vd22q6o9.streamlit.app/)

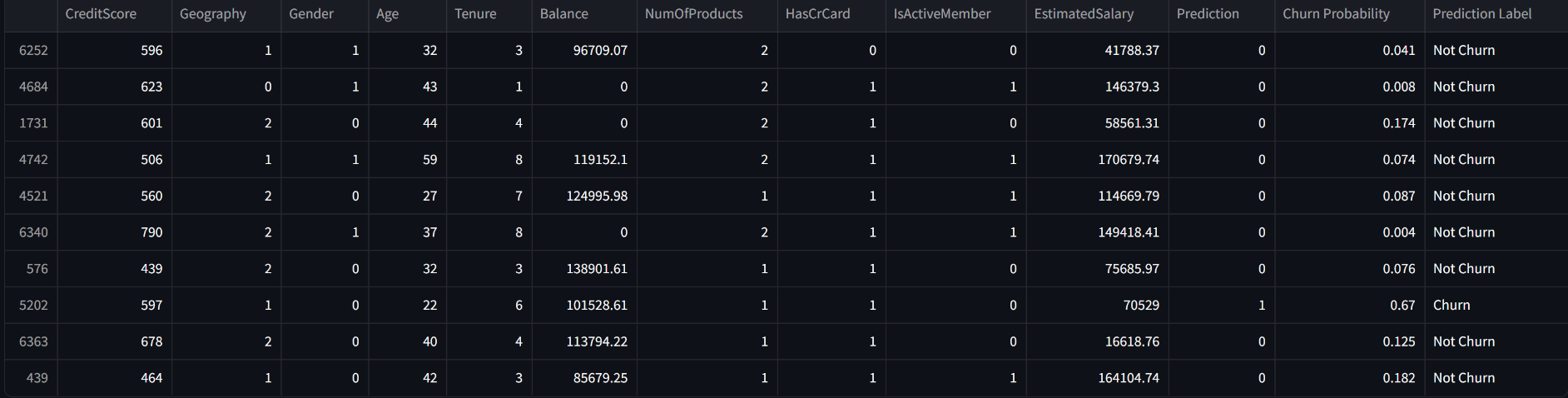
 **UI Features:**

* Dataset Upload
* Model Comparison
* SHAP Explainability

**UI SCREENSHOT:**



**PREDICTION OUTPUT** :



# Source code

import streamlit as st

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score, recall\_score, roc\_auc\_score

import shap

st.set\_page\_config(page\_title="Customer Churn Predictor", layout="wide")

st.title("📉 Customer Churn Prediction using Machine Learning")

st.sidebar.header("🛠 Application Menu")

option = st.sidebar.selectbox("Select the section", [

"Data Overview",

"Preprocessing Overview",

"Model Evaluation",

"SHAP Explainability",

"Customer Prediction"

])

uploaded\_file = st.sidebar.file\_uploader("Upload your dataset (CSV)", type=["csv"])

if uploaded\_file is not None:

df = pd.read\_csv(uploaded\_file)

else:

st.sidebar.warning("Please upload a dataset to get started!")

def preprocess\_data(df):

df\_cleaned = df.drop(columns=[col for col in ['RowNumber', 'CustomerId', 'Surname'] if col in df.columns])

le = LabelEncoder()

for col in ['Geography', 'Gender']:

if col in df\_cleaned.columns:

df\_cleaned[col] = le.fit\_transform(df\_cleaned[col])

return df\_cleaned

@st.cache\_resource

def train\_all\_models(X\_train\_scaled, y\_train):

models = {

'Random Forest': RandomForestClassifier(n\_estimators=50, random\_state=42),

'Logistic Regression': LogisticRegression(max\_iter=500),

'XGBoost': XGBClassifier(eval\_metric='logloss', n\_estimators=50),

'SVM': SVC(probability=True, random\_state=42),

'KNN': KNeighborsClassifier()

}

for model in models.values():

model.fit(X\_train\_scaled, y\_train)

return models

@st.cache\_resource

def train\_xgboost\_model(X\_train\_scaled, y\_train):

model = XGBClassifier(eval\_metric='logloss')

model.fit(X\_train\_scaled, y\_train)

return model

@st.cache\_resource

def compute\_shap\_values(\_model, X\_train\_scaled, X\_test\_scaled):

explainer = shap.Explainer(\_model)

shap\_values = explainer(X\_test\_scaled[:50])

return shap\_values

if option == "Data Overview" and uploaded\_file is not None:

st.header("📊 Dataset Overview")

st.subheader("📂 Raw Data ")

st.write(df.head())

st.subheader("📊 Feature Correlation Heatmap")

df\_cleaned = preprocess\_data(df)

fig, ax = plt.subplots(figsize=(12, 8))

sns.heatmap(df\_cleaned.corr(), annot=True, cmap='coolwarm', linewidths=0.5, ax=ax)

st.pyplot(fig)

elif option == "Preprocessing Overview" and uploaded\_file is not None:

st.header("🧹 Data Preprocessing Overview")

st.subheader("📂 Before Transformation (Raw Data)")

st.write(df.head())

df\_cleaned = preprocess\_data(df)

st.subheader("✅ After Preprocessing (Label Encoding + Column Removal)")

st.write(df\_cleaned.head())

feature\_names = df\_cleaned.drop('Exited', axis=1).columns

scaler = StandardScaler()

X\_scaled = pd.DataFrame(scaler.fit\_transform(df\_cleaned.drop('Exited', axis=1)), columns=feature\_names)

st.subheader("📐 After Scaling (StandardScaler Applied)")

st.write(X\_scaled.head())

elif option == "Model Evaluation" and uploaded\_file is not None:

st.header("🏆 Model Performance Comparison")

df\_cleaned = preprocess\_data(df)

X = df\_cleaned.drop('Exited', axis=1)

y = df\_cleaned['Exited']

feature\_names = X.columns

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

scaler = StandardScaler()

X\_train\_scaled = pd.DataFrame(scaler.fit\_transform(X\_train), columns=feature\_names)

X\_test\_scaled = pd.DataFrame(scaler.transform(X\_test), columns=feature\_names)

models = train\_all\_models(X\_train\_scaled, y\_train)

model\_results = {}

for name, model in models.items():

y\_pred = model.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

roc\_auc = roc\_auc\_score(y\_test, model.predict\_proba(X\_test\_scaled)[:, 1])

model\_results[name] = {

'Accuracy': accuracy,

'F1 Score': f1,

'Precision': precision,

'Recall': recall,

'ROC AUC': roc\_auc

}

result\_df = pd.DataFrame(model\_results).T.round(3).sort\_values(by="Accuracy", ascending=False)

st.dataframe(result\_df)

elif option == "SHAP Explainability" and uploaded\_file is not None:

st.header("🔍 SHAP Explainability for XGBoost")

df\_cleaned = preprocess\_data(df)

X = df\_cleaned.drop('Exited', axis=1)

y = df\_cleaned['Exited']

feature\_names = X.columns

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

scaler = StandardScaler()

X\_train\_scaled = pd.DataFrame(scaler.fit\_transform(X\_train), columns=feature\_names)

X\_test\_scaled = pd.DataFrame(scaler.transform(X\_test), columns=feature\_names)

xgb\_model = train\_xgboost\_model(X\_train\_scaled, y\_train)

shap\_values = compute\_shap\_values(xgb\_model, X\_train\_scaled, X\_test\_scaled)

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

shap.summary\_plot(shap\_values, features=X\_test\_scaled[:50], feature\_names=feature\_names, show=False)

st.pyplot(fig)

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

shap.summary\_plot(shap\_values, features=X\_test\_scaled[:50], feature\_names=feature\_names, plot\_type="bar", show=False)

st.pyplot(fig)

elif option == "Customer Prediction" and uploaded\_file is not None:

st.header("🔸 Customer Prediction For Random Customers")

df\_cleaned = preprocess\_data(df)

X = df\_cleaned.drop('Exited', axis=1)

y = df\_cleaned['Exited']

feature\_names = X.columns

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

xgb\_model = train\_xgboost\_model(X\_scaled, y)

sample\_df = X.sample(n=10, random\_state=42)

sample\_scaled = scaler.transform(sample\_df)

sample\_preds = xgb\_model.predict(sample\_scaled)

sample\_probs = xgb\_model.predict\_proba(sample\_scaled)[:, 1]

sample\_results = sample\_df.copy()

sample\_results['Prediction'] = sample\_preds

sample\_results['Churn Probability'] = sample\_probs.round(3)

sample\_results['Prediction Label'] = sample\_results['Prediction'].map({0: 'Not Churn', 1: 'Churn'})

st.dataframe(sample\_results)

selected\_index = st.selectbox("Select a customer index to see top factors influencing prediction", sample\_results.index)

explainer = shap.Explainer(xgb\_model)

shap\_values = explainer(sample\_scaled)

st.subheader(f"🔍 Top Factors Influencing Prediction for Customer Index: {selected\_index}")

top\_shap\_vals = shap\_values[sample\_results.index.get\_loc(selected\_index)].values

top\_feature\_indices = np.argsort(np.abs(top\_shap\_vals))[-5:][::-1]

top\_features = sample\_df.columns[top\_feature\_indices]

top\_values = top\_shap\_vals[top\_feature\_indices]

st.markdown("### Top 5 Factors Influencing Prediction:")

for feat, val in zip(top\_features, top\_values):

impact = "increases" if val > 0 else "decreases"

st.write(f"\*\*{feat}\*\*: {impact} the chance of churn by {abs(val):.4f}")

# Future scope

 Add live database/API integration to stream customer data.

 Automate hyperparameter tuning using GridSearchCV or Optuna.

 Build a feedback system to improve model based on user input.

 Implement model drift detection and retraining pipeline

# 15. Team Members and Roles

|  |  |
| --- | --- |
| Name | Contributions |
| Nidhidharshini.K | Team Leader; coordinated the project workflow, managed deadlines, performed data cleaning and preprocessing, and led exploratory data analysis (EDA). |
| Janani.V | Conducted in-depth exploratory data analysis, created visualizations to identify trends and patterns, and helped interpret insights to guide modeling. |
| Mohana.A | Responsible for feature engineering, including feature creation, transformation, and selection to enhance model accuracy and efficiency. |
| Charumathi | Carried out model evaluation by training multiple algorithms, comparing their performance, tuning hyperparameters, and selecting the best model. |
| Iniyavarshini.S | Led deployment efforts by developing and deploying the web application using Streamlit, ensuring a smooth user interface and real-time predictions. |