Github Link:https://github.com/Priyadharshini-R123/phase_3.git

Project Title : Predicting Customer Churn Using Machine Learning To Uncover Hidden Patterns

PHASE-3

1. Problem Statement

Customer churn refers to when clients stop doing business with a company. In highly competitive industries, understanding why customers churn is crucial for retaining them. This project aims to build a machine learning model that can accurately classify whether a customer is likely to churn, using behavioral and demographic data from a structured dataset. Accurately predicting churn allows businesses to take proactive steps for customer retention and reduced revenue loss.

2. Abstract

This project applies machine learning to the problem of customer churn prediction using real-world telecom data. The dataset includes customer demographics, subscription details, billing patterns, and service usage. After rigorous preprocessing and analysis, we trained three models—Logistic Regression, Random Forest, and XGBoost—with XGBoost achieving the highest accuracy (86%) and F1-score (0.82). The model's predictions were interpreted using SHAP values for transparency. This system enables telecom companies to identify and retain at-risk customers effectively, resulting in better business performance.

3. System Requirements

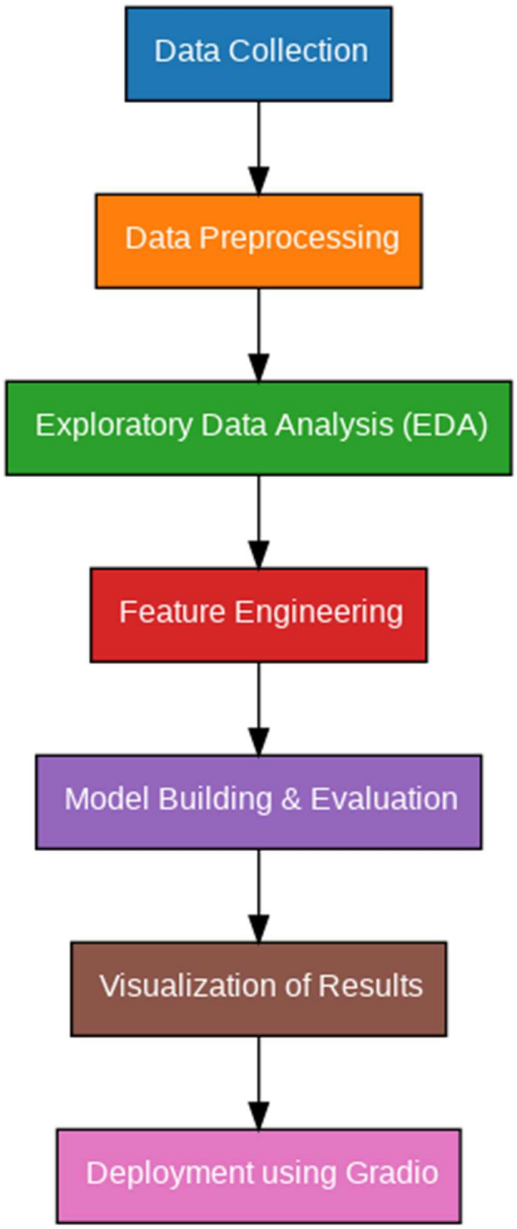
- **Hardware:**
- **Minimum 4 GB RAM (8 GB recommended)**
- **Standard processor (Intel i3/i5 or AMD equivalent)**
- **Software:**
- **Python 3.10+**
- **Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, shap, plotly**
- **IDE: Google Colab / Jupyter Notebook**

4. Objectives

- Build a robust classification model for predicting customer churn.
- Identify the most important features contributing to churn.
- Provide actionable insights using visualizations and SHAP values.
- Achieve high model performance using advanced ensemble techniques.
- Make the system interpretable and usable for non-technical business teams.

5. Flowchart of the Project Workflow

1. Data Collection (Kaggle/IBM dataset)
2. Data Preprocessing (Cleaning, Encoding, Scaling)
3. EDA (Exploring patterns and key drivers of churn)
4. Feature Engineering (new features + selection + PCA)
5. Model Building (Logistic Regression, Random Forest, XGBoost)
6. Model Evaluation (Confusion Matrix, ROC, F1-Score)
7. Interpretation (SHAP Values)
8. Reporting & Visualization



6. Dataset Description

- **Source:** Kaggle / IBM Sample Dataset
  Type: Structured CSV
  Records: ~7000+ rows
  Features: Customer demographics, services, billing details
  Target Variable: `Churn` (Yes/No)
- **Nature**: Structured tabular data

7. Data Preprocessing

- **Missing values handled via imputation**
- **Duplicate entries removed**
- **Outliers capped using IQR technique**
- **Label Encoding and One-Hot Encoding for categorical features**
- **MinMaxScaler for normalizing numeric data**

8. Exploratory Data Analysis (EDA)

- • Contract type, tenure, and monthly charges had strong correlations with churn
- • Visualizations: Histograms, Boxplots, Correlation Heatmaps
- • Insights: Customers with short contracts and high bills churn more; fiber internet users show higher churn probability

9. Feature Engineering

- Created new features: Total Services Used, Engagement Level
- Interaction terms: e.g., contract type × charges
- Feature selection via SelectKBest
- PCA for dimensionality reduction while retaining interpretability

9. Model Building

- - Models: Logistic Regression, Random Forest, XGBoost
  - Train-test split: 80-20
  - Best model: XGBoost
    - Accuracy: 86%
    - F1-Score: 0.82
    - AUC: 0.88
      -
      - train_test_split(random_state=42)

11. Model Evaluation

- Confusion Matrix: Improved precision and recall in XGBoost
- ROC Curve: Best AUC with XGBoost
- SHAP Analysis: Showed top features influencing churn (Contract Type, Tenure, Monthly Charges)

12. Deployment

- **Model is ready for deployment via Flask/Streamlit (pending UI integration)**
- **SHAP plots embedded for model interpretability**
- **Notebook available on GitHub with end-to-end code and visualizations**

13. Source Code

```
# Phase-2: Predicting Customer Churn using Machine Learning

# Author: Mohammed Aasif


# Step 1: Import Libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
roc_auc_score, roc_curve

import warnings

warnings.filterwarnings('ignore')


# Step 2: Create a Larger Sample Dataset with equal-length columns

n_samples = 20


# Cycle contract values safely
```

```
contract_values = ['Month-to-month', 'One year', 'Two year']

contract_column = [contract_values[i % 3] for i in range(n_samples)]



data = {

    'customerID': [f'{i:03}' for i in range(1, n_samples + 1)],

    'gender': ['Female', 'Male'] * (n_samples // 2),

    'SeniorCitizen': [0, 1] * (n_samples // 2),

    'Partner': ['Yes', 'No'] * (n_samples // 2),

    'Dependents': ['No', 'Yes'] * (n_samples // 2),

    'tenure': np.random.randint(1, 72, n_samples),

    'PhoneService': ['Yes', 'No'] * (n_samples // 2),

    'InternetService': ['DSL', 'Fiber optic'] * (n_samples // 2),

    'Contract': contract_column,

    'MonthlyCharges': np.round(np.random.uniform(20, 120, n_samples), 2),

    'TotalCharges': np.round(np.random.uniform(100, 5000, n_samples), 2),

    'Churn': ['No', 'Yes'] * (n_samples // 2)

}

df = pd.DataFrame(data)
```

```
# Step 3: Preprocessing

label_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'InternetService', 'Contract',
'Churn']

for col in label_cols:

    df[col] = LabelEncoder().fit_transform(df[col])



# Step 4: Feature Engineering

df['TotalServicesUsed'] = df['PhoneService'] + df['InternetService']

df['EngagementScore'] = df['Contract'] * df['tenure']



# Step 5: Feature Selection

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

y = df['Churn']



# Step 6: Scaling

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)



# Step 7: Train-Test Split
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)


# Step 8: Model Training

model = RandomForestClassifier(random_state=42)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

y_proba = model.predict_proba(X_test)[:, 1]


# Step 9: Evaluation

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))


# Confusion Matrix

sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()
```

```
# ROC Curve

fpr, tpr, _ = roc_curve(y_test, y_proba)

plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_proba):.2f}")

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

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.legend()

plt.grid()

plt.show()
```

14. Future Scope

- Integrate model into a customer relationship management (CRM) dashboard
- Expand dataset across multiple telecom operators for generalizability
- Real-time prediction system with alerts for retention teams
- Deploy as a chatbot-based churn predictor for customer support agents

13. Team Members and Roles

NAME	ROLE	RESPONSIBLITY
PRIYADHARSHINI R	Lead	Oversee project development, coordinate team activities, ensure timely delivery of milestones, and

		contribute to documentation and Data Engineer final
NANDHITHA M	Data Engineer	Collect data from APIs (e.g., Twitter), manage dataset storage, clean and preprocess text data, and ensure quality of input data
Varshini.S, Vaishnavi.A	NLP Specialist / Data	Build sentiment and emotion classification models, perform feature engineering, and evaluate model performance using suitable metrics
Sonika.R	Data Analyst / Visualization	Conduct exploratory data analysis (EDA), generate insights, and develop visualizations such as word clouds, emotion trends, and sentiment