**Phase-2 Submission Template**

**Student Name:** [**k.shalini**]

**Register Number:** [**623323205027**]

**Institution:** [**vetri vinayaha college of engineering and technology**]

**Department:** [**B.tech-IT**]

**Date of Submission:** [**07.05.2025**]

**Github Repository Link:** [Update the project source code to your Github Repository]

**Predicting customer churn using machine learning to uncover hidden pattern**

**1.Problem Statemen****t**

**Customer churn remains one of the most pressing challenges for businesses across various industries, particularly in sectors like telecommunications, finance, and subscription-based services. Despite significant investments in customer acquisition, companies often struggle to retain their existing clients due to a lack of insight into churn behaviour.**

**This project aims to address this issue by developing a machine learning-based churn prediction model. By analyzing historical customer data—such as usage patterns, demographics, customer support interactions, and transaction history—the goal is to uncover underlying signals that indicate churn risk. The model will not only predict which customers are likely to churn but also provide actionable insights to help reduce churn rates and improve overall customer retention strategies.**

**2. Project Objective****s**

**1. Develop a Predictive Model: Build and evaluate machine learning models to accurately predict customer churn based on historical data.**

**2. Uncover Hidden Patterns: Use data exploration and feature engineering to identify key behavioral, transactional, and demographic indicators contributing to churn.**

**3. Improve Customer Retention: Provide actionable insights and recommendations to the business for targeted retention strategies.**

**4. Optimize Business Decisions: Enable data-driven decision-making by integrating churn predictions into customer relationship management processes.**

### **3. Flowchart of the Project Workflow**

### **Start**

**|**

**V**

**Problem Definition**

**|**

**V**

**Data Collection**

**|**

**V**

**Data Preprocessing**

**(Handling missing values,**

**Encoding, normalization)**

**|**

**V**

**Exploratory Data Analysis (EDA)**

**(Uncover hidden patterns,**

**Feature correlations)**

**|**

**V**

**Feature Engineering & Selection**

**|**

**V**

**Model Selection**

**(Logistic Regression,**

**Decision Tree, Random Forest,**

**XGBoost, etc.)**

**|**

**V**

**Model Training & Validation**

**(Cross-validation, hyperparameter tuning)**

**|**

**V**

**Model Evaluation**

**(Accuracy, Precision, Recall, ROC-AUC)**

**|**

**V**

**Insights & Interpretation**

**(Important features, churn drivers)**

**|**

**V**

**Deployment (optional)**

**(API integration, dashboard, etc.)**

**|**

**V**

**Monitoring & Maintenance**

**|**

**V**

**End**

### **4. Data Description**

**A typical customer churn dataset contains the following types of features:**

**1.Customer Demographics**

**CustomerID: Unique identifier**

**Gender: Male/Female**

**Age: Age of the customer**

**Geography: Region or country**

**Tenure: Number of years as a customer**

**2.Account Information**

**Balance: Account balance**

**NumOfProducts: Number of products/services subscribed**

**HasCrCard: Whether the customer has a credit card (0/1)**

**IsActiveMember: Whether the customer is active (0/1)**

**3.Service Usage**

**MonthlyCharges: Amount charged monthly**

**TotalCharges: Total charges accumulated**

**4.Target Variable**

**Churn: Whether the customer has left (Yes/No or 1/0)**

### **5. Data Preprocessing**

**Preprocessing ensures your data is clean, consistent, and suitable for model training. Below are the essential steps:**

**1.Import Required Libraries**

**Import pandas as pd**

**Import numpy as np**

**From sklearn.preprocessing import LabelEncoder, StandardScaler**

**2.Load the Dataset**

**Df = pd.read\_csv(‘customer\_churn.csv’)**

**3.Handle Missing Values**

**# Check for nulls**

**Print(df.isnull().sum())**

**# Example handling**

**Df[‘TotalCharges’] = pd.to\_numeric(df[‘TotalCharges’], errors=’coerce’)**

**Df[‘TotalCharges’].fillna(df[‘TotalCharges’].median(), inplace=True)**

**4.Encode Categorical Variables**

**# Label Encoding for binary categories**

**Le = LabelEncoder()**

**Df[‘Gender’] = le.fit\_transform(df[‘Gender’]) # Male:1, Female:0**

**Df[‘HasCrCard’] = le.fit\_transform(df[‘HasCrCard’])**

**Df[‘IsActiveMember’] = le.fit\_transform(df[‘IsActiveMember’])**

**# One-hot encoding for multi-category features**

**Df = pd.get\_dummies(df, columns=[‘Geography’], drop\_first=True)**

**5.Feature Scaling**

**Scaler = StandardScaler()**

**Numerical\_features = [‘Age’, ‘Balance’, ‘Tenure’, ‘MonthlyCharges’, ‘TotalCharges’]**

**Df[numerical\_features] = scaler.fit\_transform(df[numerical\_features])**

**6.Split Features and Target**

**X= df.drop([‘CustomerID’, ‘Churn’], axis=1)**

**Y = df[‘Churn’].apply(lambda x: 1 if x == ‘Yes’ else 0)**

### **6.Exploratory data analysis**

**Univariate analysis**

**1. Univariate Analysis: Categorical Features (Count Plots)**

**Categorical\_features = [‘gender’, ‘SeniorCitizen’, ‘Partner’, ‘Dependents’,**

**‘PhoneService’, ‘InternetService’, ‘Contract’, ‘PaymentMethod’]**

**For col in categorical\_features:**

**Plt.figure(figsize=(6, 4))**

**Sns.countplot(x=col, data=df)**

**Plt.title(f’{col} Distribution’)**

**Plt.xticks(rotation=45)**

**Plt.tight\_layout()**

**Plt.show()**

**2. Univariate Analysis: Numerical Features (Histograms)**

**Numerical\_features = [‘tenure’, ‘MonthlyCharges’, ‘TotalCharges’]**

**For col in numerical\_features:**

**Plt.figure(figsize=(6, 4))**

**Sns.histplot(df[col].dropna(), kde=True, bins=30)**

**Plt.title(f’{col} Distribution’)**

**Plt.xlabel(col)**

**Plt.ylabel(‘Frequency’)**

**Plt.tight\_layout()**

**Plt.show()**

**Bivariate analysis**

* **Correlation Matrix (Numerical Features)**

**Import seaborn as sns**

**Import matplotlib.pyplot as plt**

**Plt.figure(figsize=(8, 6))**

**Numerical\_cols = [‘tenure’, ‘MonthlyCharges’, ‘TotalCharges’]**

**Corr = df[numerical\_cols].corr()**

**Sns.heatmap(corr, annot=True, cmap=’coolwarm’, fmt=”.2f”)**

**Plt.title(“Correlation Matrix of Numerical Features”)**

**Plt.show()**

* **Scatter Plots (Numerical Relationships)**

**# Scatter plot: MonthlyCharges vs TotalCharges**

**Sns.scatterplot(x=’MonthlyCharges’, y=’TotalCharges’, hue=’Churn’, data=df)**

**Plt.title(‘Monthly Charges vs Total Charges by Churn’)**

**Plt.show()**

**# Scatter plot: Tenure vs MonthlyCharges**

**Sns.scatterplot(x=’tenure’, y=’MonthlyCharges’, hue=’Churn’, data=df)**

**Plt.title(‘Tenure vs Monthly Charges by Churn’)**

**Plt.show()**

* **Bar Plots (Categorical Features vs Churn)**

**Categorical\_cols = [‘Contract’, ‘InternetService’, ‘PaymentMethod’, ‘SeniorCitizen’]**

**For col in categorical\_cols:**

**Plt.figure(figsize=(6, 4))**

**Sns.barplot(x=col, y=’Churn’, data=df.replace({‘Churn’: {‘Yes’: 1, ‘No’: 0}}), ci=None)**

**Plt.title(f’Churn Rate by {col}’)**

**Plt.xticks(rotation=45)**

**Plt.tight\_layout()**

**Plt.show()**

### **7. Feature Engineering**

**Key engineered features might include:**

1. **Behavioral Features**

**Average monthly spend: TotalCharges / Tenure**

**Tenure bucket: group tenure into bins (e.g., 0–12 months, 13–24, etc.)**

**Service count: count the number of services a customer uses**

**Customer lifetime value estimate: based on revenue and tenure**

**Interaction frequency: number of support calls per tenure month**

1. **Contract Risk Features**

**Is Month-to-Month: flag customers with flexible contracts**

**Auto-payment: customers with auto-pay may be less likely to churn**

1. **Derived Interaction Features**

**Payment consistency: detect fluctuations in payments**

**Internet dependency: ratio of online service use to total services**

1. **Customer Profile Complexity**

**Complexity score: weighted score of number of services, tenure, and monthly charges to reflect complexity of customer needs**

### **8. Model Building**

**Start with baseline models, then progress to advanced ones:**

1. **Baseline Model: Logistic Regression**

**Easy to interpret**

**Good for understanding linear relationships**

**From sklearn.linear\_model import LogisticRegression**

**Model = LogisticRegression()**

**Model.fit(X\_train, y\_train)**

1. **Tree-Based Models**

**Handle non-linear data well**

**Provide feature importance for insights**

**Random Forest:**

**From sklearn.ensemble import RandomForestClassifier**

**Model = RandomForestClassifier()**

**Model.fit(X\_train, y\_train)**

**Gradient Boosting (e.g., XGBoost):**

**From xgboost import XGBClassifier**

**Model = XGBClassifier()**

**Model.fit(X\_train, y\_train)**

1. **Other Options**

**SVM for smaller datasets**

**KNN for similarity-based churn detection**

**Neural Networks for complex, high-dimensional data**

### **9. Visualization of Results & Model Insights**

**Import seaborn as sns**

**Importance = model.feature\_importances\_**

**Features = X\_train.columns**

**Feat\_imp\_df = pd.DataFrame({‘Feature’: features, ‘Importance’: importance})**

**Feat\_imp\_df = feat\_imp\_df.sort\_values(by=’Importance’, ascending=False)**

**Plt.figure(figsize=(10, 6))**

**Sns.barplot(x=’Importance’, y=’Feature’, data=feat\_imp\_df.head(10))**

**Plt.title(“Top 10 Important Features”)**

**Plt.tight\_layout()**

**Plt.show()**

### **10. Tools and Technologies Used**

**1.Data Collection**

**Goal: Gather relevant customer data (demographics, transactions, support interactions, etc.)**

**Tools/Technologies:**

**SQL, Python**

**APIs (REST, SOAP)**

**Data lakes (Amazon S3, Google Cloud Storage)**

**ETL tools (Apache NiFi, Talend)**

**2.Data Preprocessing & Cleaning**

**Goal: Prepare data for analysis (handle nulls, normalize, encode, etc.)**

**Techniques: Missing value imputation, feature encoding, scaling, outlier detection**

**Tools: Python (pandas, NumPy, Scikit-learn), Apache Spark**

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

**Goal: Understand trends, distributions, correlations, and potential predictors**

**Tools:**

**Visualization: Matplotlib, Seaborn, Plotly**

**Profiling: pandas-profiling, Sweetviz**

**4.Feature Engineering**

**Goal: Create new meaningful features to enhance model performance**

**Examples: Tenure, RFM scores, average support call duration, usage frequency**

**Tools: Scikit-learn Pipelines, Featuretools, SQL**

**5.Model Selection & Training**

**Goal: Train machine learning models to predict churn**

**Algorithms:**

**Logistic Regression**

**Random Forest, Gradient Boosting (XGBoost, LightGBM)**

**Neural Networks (Keras/TensorFlow)**

**SVM**

**Tools: Scikit-learn, XGBoost, TensorFlow, PyTorch, MLflow (for tracking)**

**6.Model Evaluation**

**Goal: Measure model performance**

**Metrics: Precision, Recall, F1-score, ROC-AUC, Confusion Matrix**

**Tools: Scikit-learn, SHAP (for explainability), LIME**

### **11. Team Members and Contributions**

* **K.Shalini :**
* **Malathi :**
* **J.Jaiselvan :**