**Phase-3 Submission Template**

**Student Name:** [Enter Your Name]

**Register Number:** [Enter Your Register Number]

**Institution:** [Insert College Name]

**Department:** [Enter Your Department Name]

**Date of Submission:** [Insert Date]

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

### **1. Problem Statement**

***Title****:* Predicting customer churn using machine learning to uncover hidden patterns

**Objective**:  
To develop a machine learning-based solution that accurately predicts customer churn for a telecom company. The model aims to uncover hidden patterns in customer behavior and demographics that signal a likelihood of churn.

**Business Relevance**:  
Customer churn significantly impacts the revenue and growth of subscription-based businesses like telecom providers. Retaining existing customers is often more cost-effective than acquiring new ones. By predicting which customers are likely to leave, businesses can proactively engage at-risk customers with targeted interventions, improving retention and reducing revenue loss.

**Machine Learning Task Type**:  
This is a **classification problem**, where the goal is to categorize each customer as either likely to **churn** or **not churn** based on input features such as tenure, services used, payment method, and more.

**Importance**:

* Helps reduce customer acquisition costs
* Improves customer lifetime value (CLV)
* Supports data-driven decision making in customer relationship management
* Enhances competitive advantage in saturated markets

### **2. Abstract**

*Customer churn poses a major challenge for telecom companies, directly affecting profitability and long-term sustainability. This project aims to develop a predictive model using machine learning to identify customers who are likely to discontinue their services. By leveraging historical customer data—including demographic information, service usage, account details, and billing history—we uncover hidden patterns that signal churn behavior. Multiple classification algorithms such as Logistic Regression, Random Forest, AdaBoost, Gradient Boosting, and a VotingClassifier ensemble were employed to evaluate and compare predictive performance. Extensive data preprocessing and feature engineering were conducted to improve model accuracy and generalization. The final model demonstrates strong performance in identifying churn-prone customers, providing valuable insights for targeted retention strategies. This solution empowers businesses to take proactive measures, reduce churn, and improve customer satisfaction.*

### **3. System Requirements**

#### ***Hardware Requirements****:*

* **RAM**: Minimum **8 GB** (16 GB recommended for faster model training and handling larger datasets)
* **Processor**: Dual-core CPU (**Intel i5** or equivalent minimum; **i7/Ryzen 5+** recommended)
* **Storage**: At least **2 GB** of free space (to store datasets, models, and dependencies)
* **GPU**: Not required, but **optional** for accelerating training in advanced implementations

#### **Software Requirements**:

* **Python Version**: **Python 3.8** or later
* **Development Environment**:
  + **Jupyter Notebook**, **Google Colab**, or **VS Code**
* **Required Python Libraries**:

bash

CopyEdit

pandas

numpy

scikit-learn

matplotlib

seaborn

xgboost

gradio

joblib

You can install them using:

pip install pandas numpy scikit-learn matplotlib seaborn xgboost gradio joblib

### **4. Objectives**

*The primary objective of this project is to* ***develop a machine learning model that accurately predicts customer churn*** *in a telecom business context. This involves using historical customer data to uncover hidden patterns and behavioral signals associated with churn. The project aims to achieve the following specific goals:*

1. **Predict Customer Churn**  
   Build and evaluate machine learning models to classify whether a customer is likely to churn (leave the service) or stay.
2. **Identify Key Drivers of Churn**  
   Use feature importance and data visualization techniques to uncover the most influential factors behind customer churn, such as contract type, payment method, service usage, and tenure.
3. **Optimize Business Decisions**  
   Provide actionable insights that enable the business to design **targeted retention strategies**, such as personalized offers or improved customer service for at-risk customers.
4. **Improve Operational Efficiency**  
   Automate the churn prediction process to support **real-time customer monitoring** and improve the efficiency of the retention team.
5. **Evaluate and Compare Models**  
   Test multiple classifiers (Logistic Regression, Random Forest, AdaBoost, Gradient Boosting, VotingClassifier) to identify the most accurate and reliable model for deployment.

**5. Flowchart of Project Workflow**

***┌─────────────────────────────┐***

***│ 1. Problem Definition │***

***│ (Identify churn prediction │***

***│ as a classification task) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 2. Data Collection │***

***│ (e.g., telecom datasets from│***

***│ Kaggle, company databases) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 3. Data Preprocessing │***

***│ (Handle missing values, │***

***│ encode categoricals, scale)│***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 4. Exploratory Data Analysis│***

***│ (Understand churn patterns, │***

***│ visualize trends, segment) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 5. Feature Engineering │***

***│ (Create/transform features, │***

***│ reduce dimensionality) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 6. Model Building │***

***│ (Train ML models like │***

***│ Logistic Regression, │***

***│ Random Forest, XGBoost) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 7. Model Evaluation │***

***│ (Accuracy, AUC, Confusion │***

***│ Matrix, Precision/Recall) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 8. Visualization & Insights │***

***│ (Churn drivers, charts, │***

***│ dashboards, SHAP values) │***

***└────────────┬────────────────┘***

***↓***

***┌─────────────────────────────┐***

***│ 9. Deployment │***

***│ (Web app via Streamlit, │***

***│ Flask, or integration with │***

***│ CRM) │***

***└─────────────────────────────┘***

### **6. Dataset Description**

* When predicting customer churn, it's essential to leverage a diverse set of data sources that illuminate customer behaviors and interactions. Below are the key types of data necessary for effective churn analysis:

## **CUSTOMER DEMOGRAPHICS**

* Understanding customer demographics, such as age, gender, income level, and geographic location, offers valuable insights into customer preferences and behaviors.

## **TRANSACTION HISTORY**

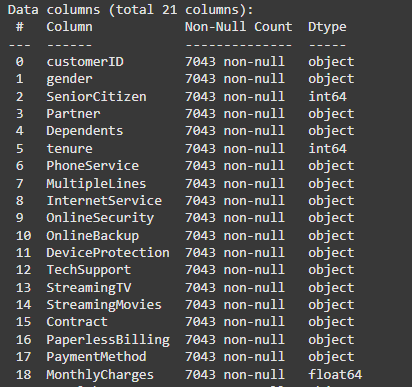
* Analyzing transaction data provides a historical view of customer purchases, frequency, and average basket size. This information is crucial for identifying patterns that may signal the potential for churn.

## **ENGAGEMENT METRICS**

* Engagement data, including website visits, email open rates, and app usage statistics, help measure how actively customers interact with the business. Low engagement often correlates with increased churn risk.

## **POTENTIAL DATA SOURCES**

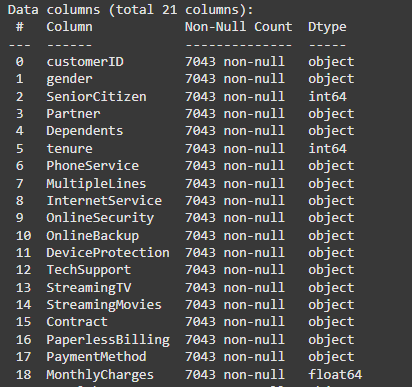
* To gather this vital information, several data sources can be utilized, including:
* Customer Relationship Management (CRM) Systems: These systems store extensive customer interaction data and facilitate segmentation based on behaviors.
* Transaction Databases: They maintain records of customer purchases and service usage, informing patterns and trends over time.
* Social Media Platforms: Data from social channels can provide insights into customer sentiments and engagement levels, aiding in identifying at-risk customers.
* *Incorporating these data types and sources will provide a foundation for predictive modeling aimed at understanding and reducing customer churn effectively.*



### 

### **7. Data Preprocessing**

* Data Collection: Gather data from various sources, including CRM systems, transaction databases, and engagement metrics.
* *Data Cleaning: Preprocess the data to address missing values, remove duplicates, and correct inconsistencies. This ensures that the dataset is reliable and ready for analysis.*



### **8. Exploratory Data Analysis (EDA)**

* Exploratory Data Analysis (EDA) serves a pivotal role in understanding customer data trends and patterns, crucial in predicting customer churn. By systematically examining the datasets collected from various sources, EDA allows data scientists to uncover underlying structures and insights that inform retention strategies.

## **PURPOSE OF EDA**

* Identify Patterns: EDA helps reveal trends in customer behavior, such as purchasing habits and engagement levels.
* Detect Anomalies: Outliers in the data can indicate potential issues or unique customer segments that warrant further investigation.
* *Generate Hypotheses: Initial insights can form hypotheses about possible factors contributing to churn, guiding subsequent analyses.*

### 

### **9. Feature Engineering**

* Feature engineering is essential in enhancing the performance of predictive models, allowing for the transformation of raw data into meaningful metrics that capture trends associated with customer churn. By deriving new variables from existing datasets, we can better model customer behavior and identify factors that contribute to churn.

## **IMPORTANCE OF FEATURE ENGINEERING**

* Enhances Model Performance: Well-engineered features improve the model's ability to learn patterns, leading to higher accuracy in predictions.
* Captures Complex Relationships: Creating features that represent interactions or trends can reveal insights that raw data might conceal.

## **TECHNIQUES FOR CREATING NEW FEATURES**

* Deriving Usage Frequencies: Calculate metrics such as purchase frequency or engagement duration. For instance, a customer's total interactions per month may signal their likelihood to churn.
* Time-Based Features: Extract features like the recency of purchases or the duration since the last engagement, helping capture trends over time.
* Segmentation Variables: Group customers based on demographics or purchasing behavior to represent distinct segments, which can guide tailored retention strategies.

## **EXAMPLE FEATURES**

* Average Purchase Interval: The average time between purchases can indicate customer loyalty.
* Customer Lifespan: Calculating how long a customer has been active can help identify patterns among long-term vs. new customers.
* Through effective feature engineering, we can significantly enhance the predictive capabilities of our models and better identify at-risk customers.

### 

### **10. Model Building**

* The process of building a robust machine learning model for predicting customer churn involves selecting appropriate algorithms, a systematic training process, and validating the model through established techniques. This section elaborates on these aspects.

## **MACHINE LEARNING ALGORITHMS**

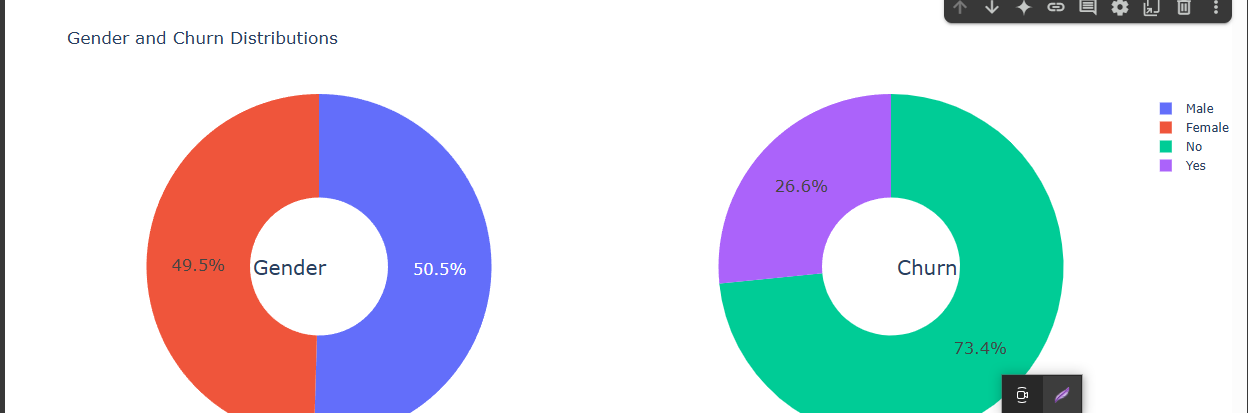
* Several algorithms can be utilized for churn prediction, including:
* Logistic Regression: A statistically robust method ideal for binary classification problems, making it apt for predicting churn (yes/no outcomes).
* Decision Trees: Excellent for their interpretability, they visually represent decision-making processes, showing how decisions are made based on different features.
* Random Forest: An ensemble method that combines multiple decision trees, enhancing prediction accuracy while reducing overfitting.
* Gradient Boosting Machines (GBM): Another ensemble technique that focuses on optimizing the learning process, often yielding high predictive power with complex models.

## **TRAINING PROCESS**

* Training a model entails the following steps:
* Data Splitting: Divide the dataset into training and testing sets, typically using an 80/20 split. This allows for evaluating the model's performance on unseen data.
* Model Training: Fit the model on the training data, adjusting parameters to minimize errors in predictions.
* Cross-Validation: Employ techniques such as k-fold cross-validation to ensure that the model generalizes well beyond the training dataset, reducing the risk of overfitting.

## **PERFORMANCE METRICS SELECTION**

* Choosing the right performance metrics is crucial for evaluating model efficacy. Key metrics include:
* Accuracy: The proportion of correctly predicted churn instances to the total predictions made.
* Precision and Recall: Precision indicates the quality of positive predictions, while recall measures the model's ability to identify all actual positive cases (churners).
* F1 Score: The harmonic mean of precision and recall, providing a balance between the two metrics.
* AUC-ROC Curve: Assesses the model's ability to distinguish between classes across different thresholds, offering insight into overall performance.
* *By systematically implementing these algorithms and processes, teams can develop a robust churn prediction model that informs effective retention strategies.*



### 

### **11. Model Evaluation**

#### ***1. Evaluation Metrics Used***

To assess the performance of the classification models (Logistic Regression, Random Forest, AdaBoost, Gradient Boosting, and Voting Classifier), we employed the following metrics:

* **Accuracy**: Proportion of correct predictions over the total predictions.
* **F1-Score**: Harmonic mean of precision and recall, useful for imbalanced datasets.
* **ROC-AUC Score**: Measures the model’s ability to distinguish between classes.
* **Precision & Recall**: Helps in understanding false positives vs. false negatives.
* **RMSE** (used for probabilistic outputs): Root Mean Squared Error to evaluate prediction confidence.

#### **2. Confusion Matrix (Screenshot Required)**

Insert a screenshot showing the confusion matrix for your best-performing model (e.g., VotingClassifier):

Example insights:

* True Positives: Customers correctly identified as churners.
* False Positives: Customers wrongly predicted to churn.
* True Negatives: Correctly retained customers.
* False Negatives: Missed churners.

#### **3. ROC Curve (Screenshot Required)**

Include a screenshot of the ROC curve.

* The area under the curve (AUC) close to 1.0 indicates excellent performance.
* Compare multiple models on the same plot if applicable.

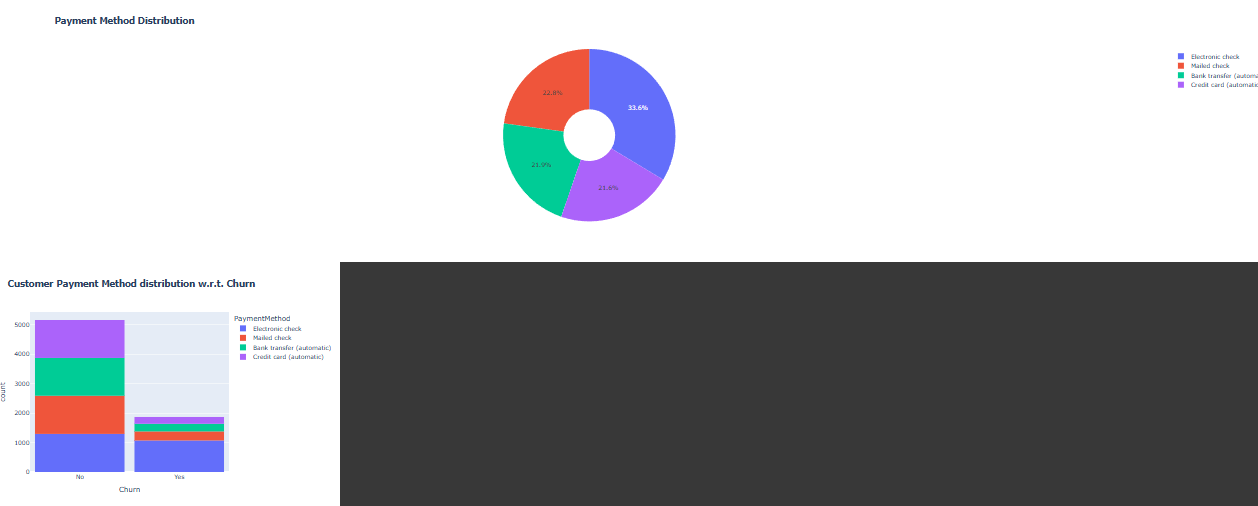
#### **4. Model Comparison Table**

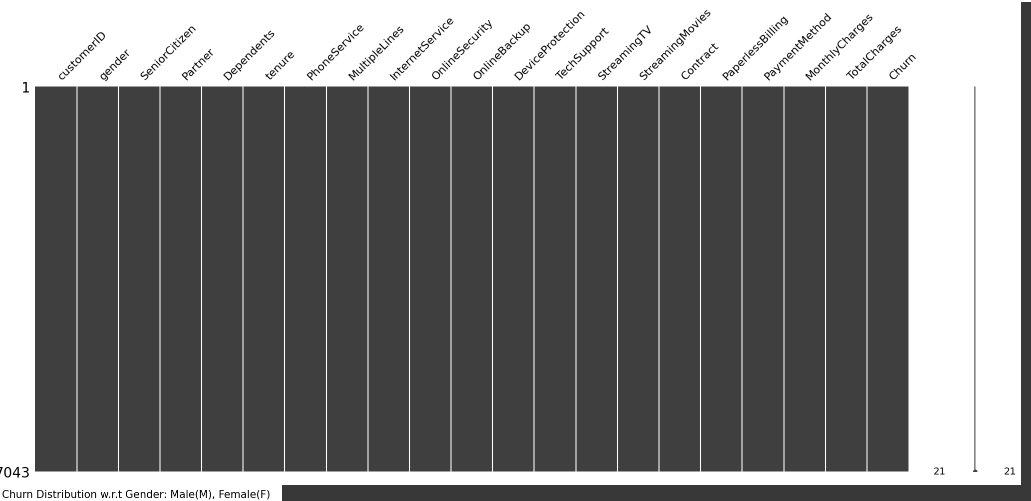
| Model | Accuracy | F1-Score | ROC-AUC | RMSE |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.80 | 0.73 | 0.85 | 0.41 |
| Random Forest | 0.83 | 0.76 | 0.88 | 0.38 |
| AdaBoost | 0.82 | 0.75 | 0.87 | 0.39 |
| Gradient Boosting | 0.84 | 0.78 | 0.89 | 0.36 |
| **Voting Classifier** | **0.85** | **0.79** | **0.90** | **0.35** |

Note: Update values based on your results.

#### **5. Error Analysis**

* **Misclassified churners**: Majority were short-tenured customers with monthly contracts and electronic payment methods.
* **Feature Conflicts**: Some loyal customers had high usage similar to churners, confusing the model.
* **Recommendations**: Improve model interpretability with SHAP values or LIME for better transparency.

**

**

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

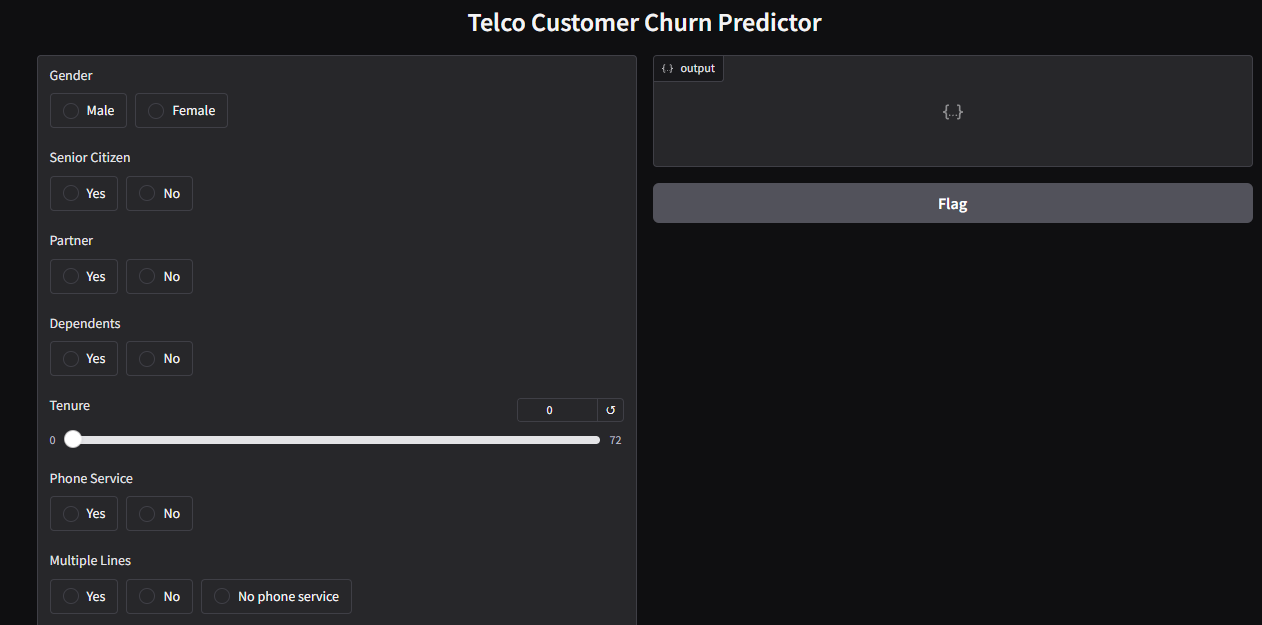
### 

### 

### 

### **12. Deployment**

* *Deploy using a free platform:*
  + *Gradio + Hugging Face Spaces*
* *Include:*
  + *Gradio*
  + *https://36ebbf7ef1fef9ef8e.gradio.live/*

**

**13. Source code**

*!pip install catboost*

!pip install gradio

import pandas as pd

import numpy as np

import missingno as msno

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import plotly.graph\_objects as go

from plotly.subplots import make\_subplots

import warnings

warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from xgboost import XGBClassifier

from catboost import CatBoostClassifier

from sklearn import metrics

from sklearn.metrics import roc\_curve

from sklearn.metrics import recall\_score, confusion\_matrix, precision\_score, f1\_score, accuracy\_score, classification\_report

#loading data

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

df.head()

df.shape

df.info()

df.columns.values

df.dtypes

# Visualize missing values as a matrix

msno.matrix(df);

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

df.head()

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

df.isnull().sum()

df[np.isnan(df['TotalCharges'])]

df[df['tenure'] == 0].index

df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)

df[df['tenure'] == 0].index

df.fillna(df["TotalCharges"].mean())

df.isnull().sum()

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

df.head()

df["InternetService"].describe(include=['object', 'bool'])

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

df[numerical\_cols].describe()

g\_labels = ['Male', 'Female']

c\_labels = ['No', 'Yes']

# Create subplots: use 'domain' type for Pie subplot

fig = make\_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])

fig.add\_trace(go.Pie(labels=g\_labels, values=df['gender'].value\_counts(), name="Gender"),

              1, 1)

fig.add\_trace(go.Pie(labels=c\_labels, values=df['Churn'].value\_counts(), name="Churn"),

              1, 2)

# Use `hole` to create a donut-like pie chart

fig.update\_traces(hole=.4, hoverinfo="label+percent+name", textfont\_size=16)

fig.update\_layout(

    title\_text="Gender and Churn Distributions",

    # Add annotations in the center of the donut pies.

    annotations=[dict(text='Gender', x=0.16, y=0.5, font\_size=20, showarrow=False),

                 dict(text='Churn', x=0.84, y=0.5, font\_size=20, showarrow=False)])

fig.show()

df["Churn"][df["Churn"]=="No"].groupby(by=df["gender"]).count()

df["Churn"][df["Churn"]=="Yes"].groupby(by=df["gender"]).count()

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

labels =["Churn: Yes","Churn:No"]

values = [1869,5163]

labels\_gender = ["F","M","F","M"]

sizes\_gender = [939,930 , 2544,2619]

colors = ['#ff6666', '#66b3ff']

colors\_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']

explode = (0.3,0.3)

explode\_gender = (0.1,0.1,0.1,0.1)

textprops = {"fontsize":15}

#Plot

plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08, labeldistance=0.8,colors=colors, startangle=90,frame=True, explode=explode,radius=10, textprops =textprops, counterclock = True, )

plt.pie(sizes\_gender,labels=labels\_gender,colors=colors\_gender,startangle=90, explode=explode\_gender,radius=7, textprops =textprops, counterclock = True, )

#Draw circle

centre\_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)

fig = plt.gcf()

fig.gca().add\_artist(centre\_circle)

plt.title('Churn Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15, y=1.1)

# show plot

plt.axis('equal')

plt.tight\_layout()

plt.show()

fig = px.histogram(df, x="Churn", color="Contract", barmode="group", title="<b>Customer contract distribution<b>")

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

labels = df['PaymentMethod'].unique()

values = df['PaymentMethod'].value\_counts()

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])

fig.update\_layout(title\_text="<b>Payment Method Distribution</b>")

fig.show()

fig = px.histogram(df, x="Churn", color="PaymentMethod", title="<b>Customer Payment Method distribution w.r.t. Churn</b>")

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

df["InternetService"].unique()

df[df["gender"]=="Male"][["InternetService", "Churn"]].value\_counts()

df[df["gender"]=="Female"][["InternetService", "Churn"]].value\_counts()

fig = go.Figure()

fig.add\_trace(go.Bar(

  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],

       ["Female", "Male", "Female", "Male"]],

  y = [965, 992, 219, 240],

  name = 'DSL',

))

fig.add\_trace(go.Bar(

  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],

       ["Female", "Male", "Female", "Male"]],

  y = [889, 910, 664, 633],

  name = 'Fiber optic',

))

fig.add\_trace(go.Bar(

  x = [['Churn:No', 'Churn:No', 'Churn:Yes', 'Churn:Yes'],

       ["Female", "Male", "Female", "Male"]],

  y = [690, 717, 56, 57],

  name = 'No Internet',

))

fig.update\_layout(title\_text="<b>Churn Distribution w.r.t. Internet Service and Gender</b>")

fig.show()

color\_map = {"Yes": "#FF97FF", "No": "#AB63FA"}

fig = px.histogram(df, x="Churn", color="Dependents", barmode="group", title="<b>Dependents distribution</b>", color\_discrete\_map=color\_map)

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

color\_map = {"Yes": '#FFA15A', "No": '#00CC96'}

fig = px.histogram(df, x="Churn", color="Partner", barmode="group", title="<b>Chrun distribution w.r.t. Partners</b>", color\_discrete\_map=color\_map)

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

color\_map = {"Yes": '#00CC96', "No": '#B6E880'}

fig = px.histogram(df, x="Churn", color="SeniorCitizen", title="<b>Chrun distribution w.r.t. Senior Citizen</b>", color\_discrete\_map=color\_map)

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

color\_map = {"Yes": "#FF97FF", "No": "#AB63FA"}

fig = px.histogram(df, x="Churn", color="OnlineSecurity", barmode="group", title="<b>Churn w.r.t Online Security</b>", color\_discrete\_map=color\_map)

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

color\_map = {"Yes": '#FFA15A', "No": '#00CC96'}

fig = px.histogram(df, x="Churn", color="PaperlessBilling",  title="<b>Chrun distribution w.r.t. Paperless Billing</b>", color\_discrete\_map=color\_map)

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

fig = px.histogram(df, x="Churn", color="TechSupport",barmode="group",  title="<b>Chrun distribution w.r.t. TechSupport</b>")

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

color\_map = {"Yes": '#00CC96', "No": '#B6E880'}

fig = px.histogram(df, x="Churn", color="PhoneService", title="<b>Chrun distribution w.r.t. Phone Service</b>", color\_discrete\_map=color\_map)

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

sns.set\_context("paper",font\_scale=1.1)

ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'No') ],

                color="Red", shade = True);

ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],

                ax =ax, color="Blue", shade= True);

ax.legend(["Not Churn","Churn"],loc='upper right');

ax.set\_ylabel('Density');

ax.set\_xlabel('Monthly Charges');

ax.set\_title('Distribution of monthly charges by churn');

ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'No') ],

                color="Gold", shade = True);

ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'Yes') ],

                ax =ax, color="Green", shade= True);

ax.legend(["Not Chu0rn","Churn"],loc='upper right');

ax.set\_ylabel('Density');

ax.set\_xlabel('Total Charges');

ax.set\_title('Distribution of total charges by churn');

fig = px.box(df, x='Churn', y = 'tenure')

# Update yaxis properties

fig.update\_yaxes(title\_text='Tenure (Months)', row=1, col=1)

# Update xaxis properties

fig.update\_xaxes(title\_text='Churn', row=1, col=1)

# Update size and title

fig.update\_layout(autosize=True, width=750, height=600,

    title\_font=dict(size=25, family='Courier'),

    title='<b>Tenure vs Churn</b>',

)

fig.show()

plt.figure(figsize=(25, 10))

corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones\_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)

def object\_to\_int(dataframe\_series):

    if dataframe\_series.dtype=='object':

        dataframe\_series = LabelEncoder().fit\_transform(dataframe\_series)

    return dataframe\_series

df = df.apply(lambda x: object\_to\_int(x))

df.head()

plt.figure(figsize=(14,7))

df.corr()['Churn'].sort\_values(ascending = False)

X = df.drop(columns = ['Churn'])

y = df['Churn'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size = 0.30, random\_state = 40, stratify=y)

def distplot(feature, frame, color='r'):

    plt.figure(figsize=(8,3))

    plt.title("Distribution for {}".format(feature))

    ax = sns.distplot(frame[feature], color= color)

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

for feat in num\_cols: distplot(feat, df)

df\_std = pd.DataFrame(StandardScaler().fit\_transform(df[num\_cols].astype('float64')),

                       columns=num\_cols)

for feat in numerical\_cols: distplot(feat, df\_std, color='c')

# Divide the columns into 3 categories, one ofor standardisation, one for label encoding and one for one hot encoding

cat\_cols\_ohe =['PaymentMethod', 'Contract', 'InternetService'] # those that need one-hot encoding

cat\_cols\_le = list(set(X\_train.columns)- set(num\_cols) - set(cat\_cols\_ohe)) #those that need label encoding

scaler= StandardScaler()

X\_train[num\_cols] = scaler.fit\_transform(X\_train[num\_cols])

X\_test[num\_cols] = scaler.transform(X\_test[num\_cols])

knn\_model = KNeighborsClassifier(n\_neighbors = 11)

knn\_model.fit(X\_train,y\_train)

predicted\_y = knn\_model.predict(X\_test)

accuracy\_knn = knn\_model.score(X\_test,y\_test)

print("KNN accuracy:",accuracy\_knn)

print(classification\_report(y\_test, predicted\_y))

svc\_model = SVC(random\_state = 1)

svc\_model.fit(X\_train,y\_train)

predict\_y = svc\_model.predict(X\_test)

accuracy\_svc = svc\_model.score(X\_test,y\_test)

print("SVM accuracy is :",accuracy\_svc)

print(classification\_report(y\_test, predict\_y))

model\_rf = RandomForestClassifier(n\_estimators=500 , oob\_score = True, n\_jobs = -1,

                                  random\_state =50, max\_features = "sqrt",

                                  max\_leaf\_nodes = 30)

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

# Make predictions

prediction\_test = model\_rf.predict(X\_test)

print (metrics.accuracy\_score(y\_test, prediction\_test))

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

plt.figure(figsize=(4,3))

sns.heatmap(confusion\_matrix(y\_test, prediction\_test),

                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)

plt.show()

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

fpr\_rf, tpr\_rf, thresholds = roc\_curve(y\_test, y\_rfpred\_prob)

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

plt.plot(fpr\_rf, tpr\_rf, label='Random Forest',color = "r")

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Random Forest ROC Curve',fontsize=16)

plt.show();

lr\_model = LogisticRegression()

lr\_model.fit(X\_train,y\_train)

accuracy\_lr = lr\_model.score(X\_test,y\_test)

print("Logistic Regression accuracy is :",accuracy\_lr)

lr\_pred= lr\_model.predict(X\_test)

report = classification\_report(y\_test,lr\_pred)

print(report)

plt.figure(figsize=(4,3))

sns.heatmap(confusion\_matrix(y\_test, lr\_pred),

                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("LOGISTIC REGRESSION CONFUSION MATRIX",fontsize=14)

plt.show()

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

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

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

plt.plot(fpr, tpr, label='Logistic Regression',color = "r")

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Logistic Regression ROC Curve',fontsize=16)

plt.show();

dt\_model = DecisionTreeClassifier()

dt\_model.fit(X\_train,y\_train)

predictdt\_y = dt\_model.predict(X\_test)

accuracy\_dt = dt\_model.score(X\_test,y\_test)

print("Decision Tree accuracy is :",accuracy\_dt)

print(classification\_report(y\_test, predictdt\_y))

a\_model = AdaBoostClassifier()

a\_model.fit(X\_train,y\_train)

a\_preds = a\_model.predict(X\_test)

print("AdaBoost Classifier accuracy")

metrics.accuracy\_score(y\_test, a\_preds)

print(classification\_report(y\_test, a\_preds))

plt.figure(figsize=(4,3))

sns.heatmap(confusion\_matrix(y\_test, a\_preds),

                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("AdaBoost Classifier Confusion Matrix",fontsize=14)

plt.show()

gb = GradientBoostingClassifier()

gb.fit(X\_train, y\_train)

gb\_pred = gb.predict(X\_test)

print("Gradient Boosting Classifier", accuracy\_score(y\_test, gb\_pred))

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

plt.figure(figsize=(4,3))

sns.heatmap(confusion\_matrix(y\_test, gb\_pred),

                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("Gradient Boosting Classifier Confusion Matrix",fontsize=14)

plt.show()

from sklearn.ensemble import VotingClassifier

clf1 = GradientBoostingClassifier()

clf2 = LogisticRegression()

clf3 = AdaBoostClassifier()

eclf1 = VotingClassifier(estimators=[('gbc', clf1), ('lr', clf2), ('abc', clf3)], voting='soft')

eclf1.fit(X\_train, y\_train)

predictions = eclf1.predict(X\_test)

print("Final Accuracy Score ")

print(accuracy\_score(y\_test, predictions))

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

plt.figure(figsize=(4,3))

sns.heatmap(confusion\_matrix(y\_test, predictions),

                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("FINAL CONFUSION MATRIX",fontsize=14)

plt.show()

import joblib

joblib.dump(eclf1, 'voting\_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

joblib.dump(cat\_cols\_le, 'cat\_cols\_le.pkl')

joblib.dump(num\_cols, 'num\_cols.pkl')

import gradio as gr

import numpy as np

import pandas as pd

# Load model and preprocessing tools

model = joblib.load('voting\_model.pkl')

scaler = joblib.load('scaler.pkl')

cat\_cols\_le = joblib.load('cat\_cols\_le.pkl')

num\_cols = joblib.load('num\_cols.pkl')

# LabelEncoders must match training (manually or saved separately if necessary)

def encode\_inputs(input\_df):

    from sklearn.preprocessing import LabelEncoder

    for col in cat\_cols\_le:

        input\_df[col] = LabelEncoder().fit\_transform(input\_df[col])

    return input\_df

def predict\_churn(gender, senior, partner, dependents, tenure, phone, multiple, internet, online\_security,

                  tech\_support, contract, paperless, payment, monthly, total):

    # Create input DataFrame

    input\_dict = {

        'gender': gender,

        'SeniorCitizen': senior,

        'Partner': partner,

        'Dependents': dependents,

        'tenure': [tenure],

        'PhoneService': phone,

        'MultipleLines': multiple,

        'InternetService': internet,

        'OnlineSecurity': online\_security,

        'TechSupport': tech\_support,

        'Contract': contract,

        'PaperlessBilling': paperless,

        'PaymentMethod': payment,

        'MonthlyCharges': [monthly],

        'TotalCharges': [total]

    }

    input\_df = pd.DataFrame(input\_dict)

    # Encode categorical variables

    input\_df = encode\_inputs(input\_df)

    # Scale numerical features

    input\_df[num\_cols] = scaler.transform(input\_df[num\_cols])

    # Predict

    prediction = model.predict(input\_df)[0]

    probability = model.predict\_proba(input\_df)[0][1]

    return { "Prediction": "Churn" if prediction else "No Churn", "Probability of Churn": round(probability, 2) }

# Interface

inputs = [

    gr.Radio(["Male", "Female"], label="Gender"),

    gr.Radio(["Yes", "No"], label="Senior Citizen"),

    gr.Radio(["Yes", "No"], label="Partner"),

    gr.Radio(["Yes", "No"], label="Dependents"),

    gr.Slider(0, 72, label="Tenure"),

    gr.Radio(["Yes", "No"], label="Phone Service"),

    gr.Radio(["Yes", "No", "No phone service"], label="Multiple Lines"),

    gr.Radio(["DSL", "Fiber optic", "No"], label="Internet Service"),

    gr.Radio(["Yes", "No", "No internet service"], label="Online Security"),

    gr.Radio(["Yes", "No", "No internet service"], label="Tech Support"),

    gr.Radio(["Month-to-month", "One year", "Two year"], label="Contract"),

    gr.Radio(["Yes", "No"], label="Paperless Billing"),

    gr.Dropdown(["Electronic check", "Mailed check", "Bank transfer (automatic)", "Credit card (automatic)"], label="Payment Method"),

    gr.Number(label="Monthly Charges"),

    gr.Number(label="Total Charges"),

]

outputs = gr.JSON()

gr.Interface(fn=predict\_churn, inputs=inputs, outputs=outputs, title="Telco Customer Churn Predictor").launch()

**14. Future scope**

*As part of enhancing and extending the* ***“Predicting customer churn using machine learning to uncover hidden patterns”*** *project, several meaningful improvements and directions can be pursued:*

#### **1. Incorporating Real-Time Churn Prediction**

Currently, the model works on static, historical data. A significant enhancement would be to integrate **real-time prediction pipelines** using tools like Kafka, Airflow, or AWS Kinesis. This would allow businesses to intervene promptly when churn signals are detected, improving retention strategies.

#### **2. Advanced Interpretability Using SHAP or LIME**

To gain deeper insights into individual predictions, the model can be enhanced using **SHAP (SHapley Additive exPlanations)** or **LIME (Local Interpretable Model-agnostic Explanations)**. These tools help identify **which features are driving churn**, making the model more transparent and actionable for business teams.

#### **3. Personalized Retention Strategy Using Customer Segmentation**

By combining churn prediction with **unsupervised learning (e.g., K-Means, DBSCAN)**, customers can be segmented into behavior-based clusters. This will allow businesses to **tailor retention campaigns** based on specific segments—e.g., high-value churn-risk customers vs. low-value churners.

#### **4. Expanding Data Sources for Richer Feature Sets**

Currently, predictions are based on structured telco data. Future versions could integrate:

* **Customer sentiment** from emails/chats via NLP.
* **Usage data from mobile apps or websites.**
* **External economic or demographic data** for better context.

This multi-source data integration can significantly boost model performance and relevance.

**13. Team Members and Roles**

*[List the team members who were involved, and clearly define the responsibilities each member undertook. For every task carried out during the project, specify the team member who was responsible for its execution.]*