Telecom Customer Churn Prediction

Project Description:

The project “telecom customer churn prediction” aims that in the competitive landscape of telecommunications, customer churn—defined as the rate at which customers switch from one service provider to another—is a critical metric that directly impacts business sustainability and growth. Understanding and predicting customer churn is essential for telecom companies to proactively retain their customer base, enhance service offerings, and maintain profitability.

Project Scenario:

Scenario 1: Company Background

Imagine you work for a leading telecom company that offers a comprehensive range of services, including mobile, internet, and television subscriptions. Operating in a highly competitive market, your company faces stiff competition from other providers offering similar services at varying price points and service levels.

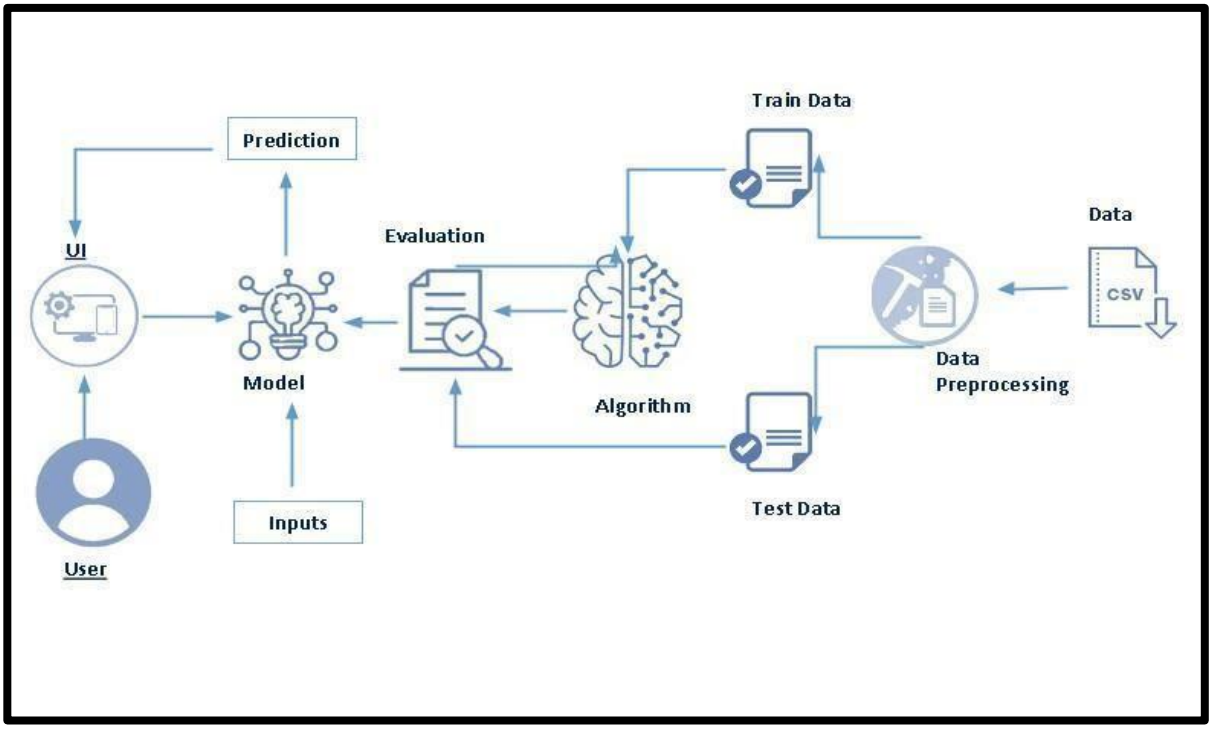
Scenario 2: Current Situation

Over the past few quarters, your company has observed a concerning trend of increasing customer churn rates. Despite implementing aggressive marketing campaigns and maintaining competitive pricing, a significant number of customers are opting to switch to rival telecom providers. This uptick in churn not only impacts revenue but also poses a threat to the company's market position and brand reputation.

Scenario 3: Customer Behavior and Usage Analytics

The system analyzes comprehensive customer data, including call records, data usage, billing information, and support interactions. Machine learning algorithms identify trends and anomalies that signal potential churn. For instance, a sudden drop in usage or increased complaints may trigger an alert. The system then suggests targeted actions such as upselling relevant services, providing usage incentives, or addressing billing concerns to improve customer retention.

Technical Architecture:



Project Flow: Telecom Customer Churn Prediction

1. Project Initiatio: Clearly define the project's goals, such as predicting customer churn and identifying factors contributing to churn. Identify and engage stakeholders including marketing, customer service, IT, and data science teams**.**

2. Data Collection and Integration: Identify and gather data from various sources including:Customer demographics (CRM),Service usage patterns (Call records, Data usage),Billing information,Customer support interactions,Network performance metrics, Use ETL (Extract, Transform, Load) processes to integrate data into a unified data warehouse.

3. Data Preprocessing: Handle missing values, outliers, and inconsistencies. Create new features that might be useful for the prediction model (e.g., tenure, average monthly charges).Normalize or standardize features, encode categorical variables, and split data into training and test sets.

4. Exploratory Data Analysis (EDA): Analyze data distributions, correlations, and summary statistics.Use visual tools to understand data patterns and relationships (e.g., histograms, scatter plots, heatmaps).

5. Model Development:Choose appropriate machine learning algorithms (e.g., logistic regression, decision trees, random forests, gradient boosting, neural network. Train models using the training dataset. Optimize hyperparameters using techniques such as grid search or random search. Evaluate models using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

6. Model Validation: Perform cross-validation to ensure model robustness.Compare performance of different models to select the best one.

7. Model Deployment: Deploy the selected model into the telecom company's existing systems.Develop APIs to integrate the model with CRM and other operational system.Enable real-time churn prediction and recommendations.

8. Intervention Strategy Design: Design personalized retention strategies based on churn prediction results.Plan and execute retention campaigns (e.g., targeted discounts, loyalty programs, service improvements).

9. Monitoring and Feedback: Continuously monitor the model's performance and operational impact. Collect feedback from retention campaigns to refine and improve the model and strategies.

10. Reporting and Insights: Develop dashboards to visualize key metrics and insights.Provide regular reports to stakeholders on project outcomes and business impact.

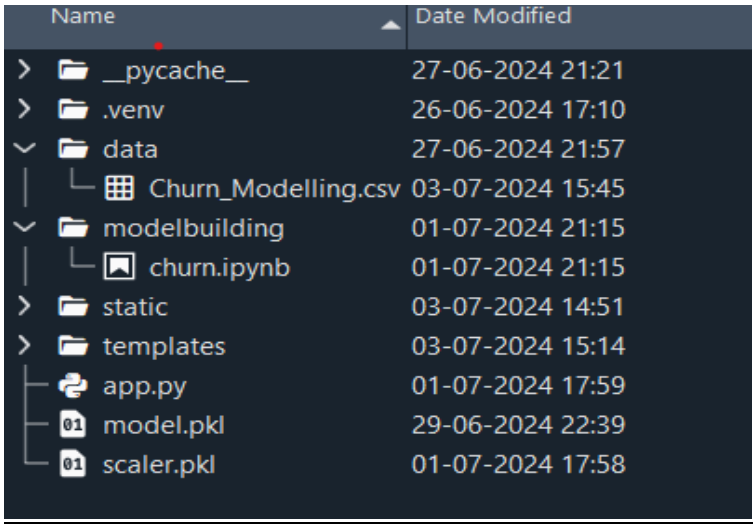
prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

- ML Concepts
- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- Hyper parameter Tuning : https://www.geeksforgeeks.org/hyperparameter-tuning/
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a Flask application that needs HTML pages stored in the templates folder and a Python script app.py for scripting.
- model.pkl is our saved model. Further, we will use this model for flask integration.
- churn.ipynb contains a model training file.

Milestone 1: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link:https://www.kaggle.com/datasets/shrutimechlearn/churn modelling?select=Churn_Modelling.csv

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 2: Importing the libraries

Import the necessary libraries as shown in the image.

```
[1]: #importing the libraries
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.metrics import accuracy_score,confusion_matrix
```

```
[23]: #labelEncoding
```

```
[24]: from sklearn.preprocessing import LabelEncoder
```

```
[42]: #training and testing the data
      from sklearn.model_selection import train_test_split
```

```
[54]: #logistic Regression
      from sklearn.linear_model import LogisticRegression
```

```
[43]: #Feature Scaling
      from sklearn.preprocessing import StandardScaler
```

```
[56]: #Decision Tree classifier]
      from sklearn.tree import DecisionTreeClassifier
```

```
[58]: #random forest classifier
      from sklearn.ensemble import RandomForestClassifier
```

```
: from sklearn.metrics import precision_score,accuracy_score
```

```
7]: #kNeighborsClassifier
    from sklearn.neighbors import KNeighborsClassifier
```

```
: from sklearn.metrics import mean_squared_error,r2_score
```

```
: #naive bayes classifier
  from sklearn.naive_bayes import GaussianNB
  gnb = GaussianNB()
```

```
4]: import  pickle
```

Activity 3: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas, we have a function called read_csv() to read the dataset. As a parameter, we have to give the directory of the CSV file.

```
2]: #Reading the dataset
    dt=pd.read_csv(r"Churn_Modelling.csv")
```

Activity 4: Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data

- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 5: Handling missing values

Let's find the shape of our dataset first. To find the shape of our data, the data. shape method is used. To find the data type, the data.info() function is used.

```
[4]: #Handling missing values
     dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

For checking the null values, data.isnull() function is used. To sum those null values we use .sum() function. From the below image, we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
dt.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

As we can see our dataset has no missing values.

We can proceed with it.

Milestone 2: Exploratory Data Analysis

Activity 1: Descriptive statistics

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max, and percentile values of continuous features.

```
22]: data.describe()
```

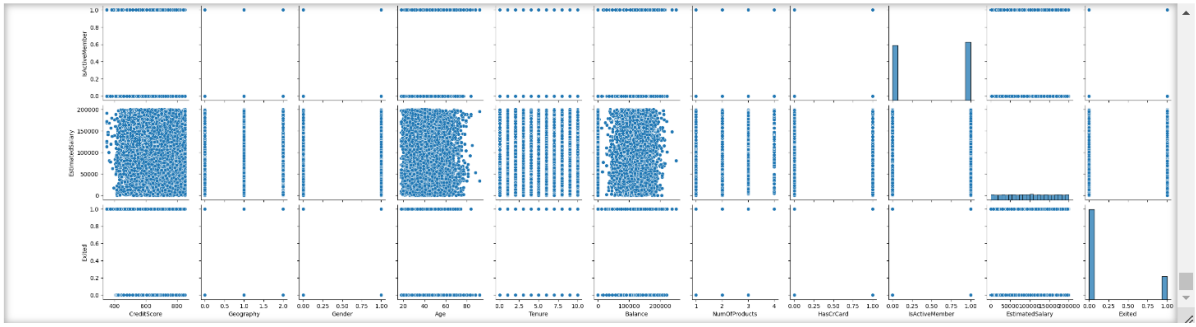
22]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Activity 2: Visual analysis

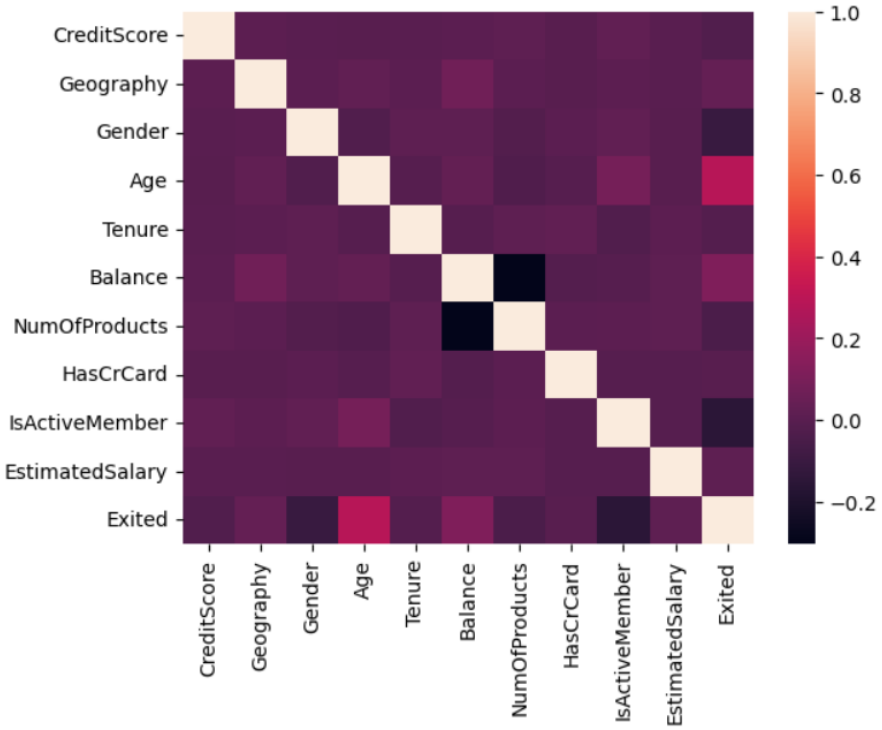
Visual analysis is using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

```
[32]: sns.pairplot(data=data,markers=["^","V"],palette="inferno")
```



```
[31]: sns.heatmap(data.corr(),annot=False)
```

```
[31]: <Axes: >
```



Activity 3: Splitting data into train and test

Now let's split the Dataset into train and test sets. First, split the dataset into x and y and then split the data set, here x and y variables are created. On the x variable, data is passed by dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
[36]: #splitting the data
      x=data.iloc[:,0:10].values
      y=data.iloc[:,10:11].values
```

```
[42]: #training and testing the data
      from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
[43]: #Feature Scaling
      from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      x_train=sc.fit_transform(x_train)
      x_test = sc.transform(x_test)
```

The code imports the StandardScaler module for feature scaling. It then scales the features in the input data 'x' using standardization and splits the dataset into training and testing sets with a test size of 20% and a random state of 0.

Milestone 3: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying classification algorithms. The best model is saved based on its performance.

Model 1 :

Logistic Regression

```
[54]: #logistic Regression
      from sklearn.linear_model import LogisticRegression

      model=LogisticRegression()
      model.fit(x_train,y_train)
      accuracy_score(model.predict(x_test),y_test)
```

```
[54]: 0.807
```

```
[55]: model.predict([[619,0,0,42,2,0,1,1,1,101348.88]])
```

```
[55]: array([1], dtype=int64)
```

Logistic regression is a statistical technique used to predict binary outcomes, such as whether a customer will churn or not. It calculates the probability of an event occurring based on input variables, fitting the data to a sigmoidal (S-shaped) curve. This method is valued for its simplicity, interpretability, and efficiency in handling large datasets, making it a popular choice for predicting customer behavior in various industries, including telecom.

Model 2:

Decision Tree

[56]:
```
#Decision Tree classifier]
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=42)
classifier.fit(x_train, y_train)
pred=classifier.predict(x_test)
dtc_acc=accuracy_score(pred,y_test)
dtc_acc
```

[56]: 0.7835

Decision trees are a machine learning algorithm used for both classification and regression tasks. They work by recursively partitioning the data into subsets based on the values of input features. At each step, the algorithm selects the feature that best splits the data, aiming to minimize impurity or maximize information gain. Decision trees are interpretable, versatile, and can handle both numerical and categorical data, making them popular in various domains such as finance, healthcare, and marketing.

Model 3:
Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and combines their predictions to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the training data and a random subset of features. The final prediction is determined by averaging or taking a majority vote of the predictions made by individual trees.

[58]:
```
#random forest classifier
from sklearn.ensemble import RandomForestClassifier
rc=RandomForestClassifier(random_state=42)
rc.fit(x_train,y_train)
pred=rc.predict(x_test)
rfc_acc=accuracy_score(y_test,pred)
rfc_acc
```

[58]:
```
0.864
```

[59]:
```
d=[[758,0,0,34,3,0,2,1,1,124226.16]]
```

[60]:
```
input_data_scaled = sc.transform(d)
rc.predict(input_data_scaled)
```

[60]:
```
array([0], dtype=int64)
```

[61]:
```
rfc_con=confusion_matrix(pred,y_test)



rfc_con
```

[61]:
```
array([[1528,  205],
       [  67,  200]], dtype=int64)
```

2]:
```
from sklearn.metrics import precision_score,accuracy_score
precision_score(y_test,pred)
```

2]:
```
0.7490636704119851
```

]:

3]:
```
from sklearn.metrics import mean_squared_error,r2_score
```

4]:
```
mean_squared_error(y_test,pred)
```

4]:
```
0.136
```

5]:
```
r2_score(y_test,pred)
```

5]:
```
0.15786214636789353
```

Model 4:
svm

```
[50]: #Model Building

[51]: #support Vector Machine
      from sklearn.svm import SVC
      svm=SVC(kernel="linear")

[52]: svm.fit(x_train,y_train)

[52]: SVC
      SVC(kernel='linear')

[53]: svm_pred = svm.predict(x_test)
      svm_acc = accuracy_score(svm_pred,y_test)
      svm_acc

[53]: 0.7975
```

Support Vector Machines (SVM) are supervised learning models used for classification and regression tasks. In classification, SVM finds the optimal hyperplane that best separates data points of different classes with the maximum margin. This method is effective in high-dimensional spaces and can handle non-linear relationships using kernel functions. SVMs are widely used in various applications for their ability to generalize well to new data and handle complex datasets efficiently.

Model 5:
Knn

```
[67]: #kNeighborsClassifier
      from sklearn.neighbors import KNeighborsClassifier
      knn=KNeighborsClassifier()

[68]: knn.fit(x_train,y_train)

[68]: ▾ KNeighborsClassifier
      KNeighborsClassifier()

[69]: knn_acc=accuracy_score(knn.predict(x_test),y_test)
      knn_acc

[69]: 0.8345

[70]: knn_con=confusion_matrix(knn.predict(x_test),y_test)
      knn_con

[70]: array([[1496,  232],
             [  99,  173]], dtype=int64)
```

K-Nearest Neighbors (KNN) is a simple and effective supervised learning algorithm used for classification and regression tasks. In KNN classification, for a given new data point, the algorithm identifies the k nearest neighbors based on a distance metric (e.g., Euclidean distance) from the training data. The class of the majority of its nearest neighbors determines the class of the new data point.

Model 6:
Naives bayes

Naive Bayes is a probabilistic classifier that assumes independence between features. It calculates the probability of each class based on the given features and selects the most likely class as the prediction. Naive Bayes is simple, efficient, and effective for text classification tasks like spam detection and sentiment analysis.

[71]:
```
#naive bayes classifier
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train, y_train)
nb_acc=accuracy_score(gnb.predict(x_test),y_test)
nb_acc
```

[71]:
```
0.8275
```

[72]:
```
nb_con=confusion_matrix(gnb.predict(x_test),y_test)
nb_con
```

[72]:
```
array([[1548,  298],
       [  47,  107]], dtype=int64)
```

Milestone 5: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

[73]:
```
#model deployment
```

[74]:
```
import  pickle
filename="model.pkl"
pickle.dump(rc,open(filename,"wb"))
```

[75]:
```
model=pickle.load(open(filename,'rb'))
```

[76]:
```
model.score(x_test,y_test)*100
```

[76]:
```
86.4
```

[77]:
```
filename="scaler.pkl"
pickle.dump(sc,open(filename,"wb"))
```

This code exports the trained machine learning model 'model' using pickle serialization and saves it as a file named "model. pkl". This allows for the

model to be easily stored, shared, and loaded for future use without needing to retrain it.

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 3: Building Html Pages:

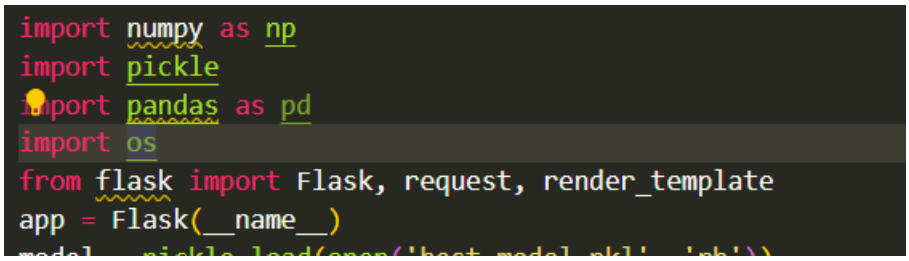
For this project create two HTML files namely

- index.html
- base.html
- predictYes.html
- predictNo.html

and save them in the templates folder.

Activity 4: Build Python code:

Import the libraries



Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (__name__) as argument.

Here we will be using a declared constructor to route to the HTML page that we have created earlier.

In the below example, the '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the HTML page will be rendered.

Whenever you enter the values from the html page the values can be retrieved using POST Method.

- Retrieves the value from UI:

```
@app.route('/')
def base():
    return render_template('base.html')
@app.route('/predict')
def predict():
    return render_template('index.html')
```

- Here we are routing our app to predict the () function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. submit() function. This function returns the prediction. This prediction value will be rendered to the text that we have mentioned in the index.html page earlier.

```
@app.route('/submit',methods=["POST"])
def submit():
    CreditScore=int(request.form["credit_score"])
    Geography=request.form["geography"]
    if(Geography=="France"):
        Geography=0
    elif(Geography=="Germany"):
        Geography=1
    else:
        Geography=2
    Gender=request.form["gender"]
    if(Gender=='female'):
       Gender=0
    else:
       Gender=1
    Age=int(request.form["age"])
    Tenure=int(request.form["tenure"])
    Balance=float(request.form["balance"])
    NumberOfProducts=int(request.form["number_of_products"])
    HasCrCard=request.form["has_cr_card"]
    if(HasCrCard=='yes'):
       HasCrCard=1
    else:
       HasCrCard=0
    IsActiveMember=request.form["is_active_member"]
    if(IsActiveMember=='yes'):
        IsActiveMember=1
    else:
        IsActiveMember=0
    EstimatedSalary=float(request.form["estimated_salary"])
    t=[[CreditScore,Geography,Gender,Age,Tenure,Balance,NumberOfProducts,HasCrCard,IsActiveMember,EstimatedSalary]]
    i= sc.transform(t)

    x=model.predict(i)
    if(x[0]==0):
        return render_template("predNo.html")
    else:
        return render_template("predYes.html")
```

Main Function:

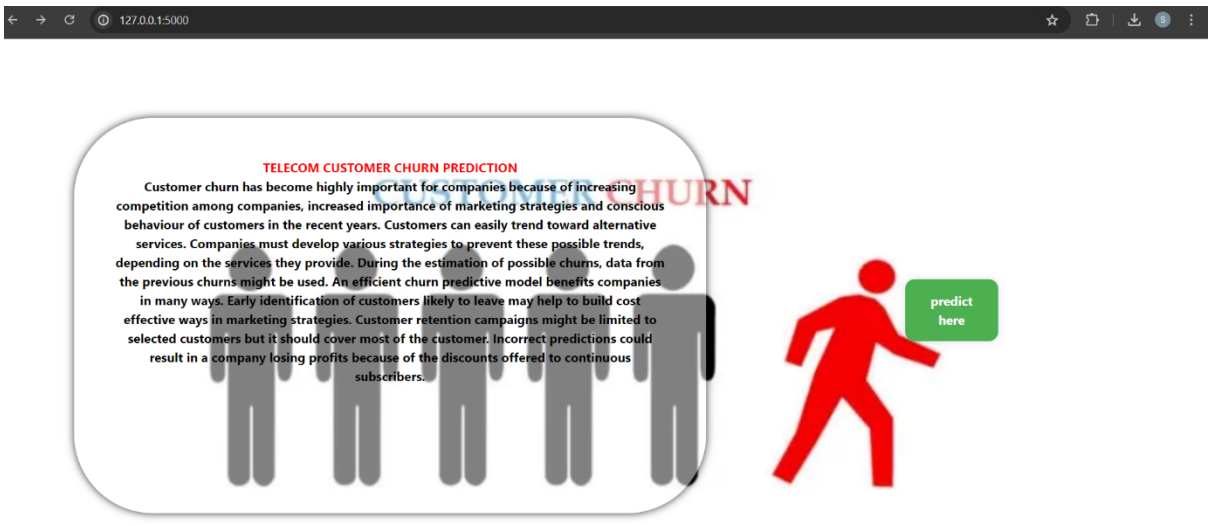
```
if __name__=="__main__":
    app.run( debug=True, port = 5000)
```

Activity 5: Run the web application

- Open the anaconda prompt from the start menu
- Navigate to the folder where your Python script is.
- Now type the “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

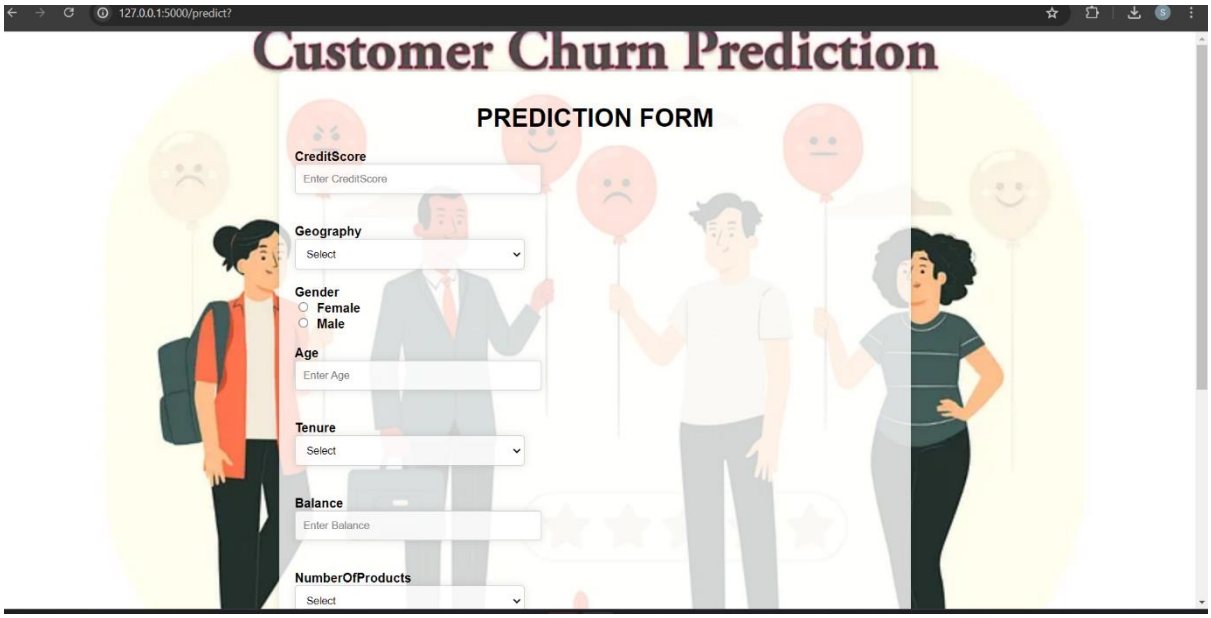
```
 * Debug mode: on
WARNING: This is a development server. D
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
```

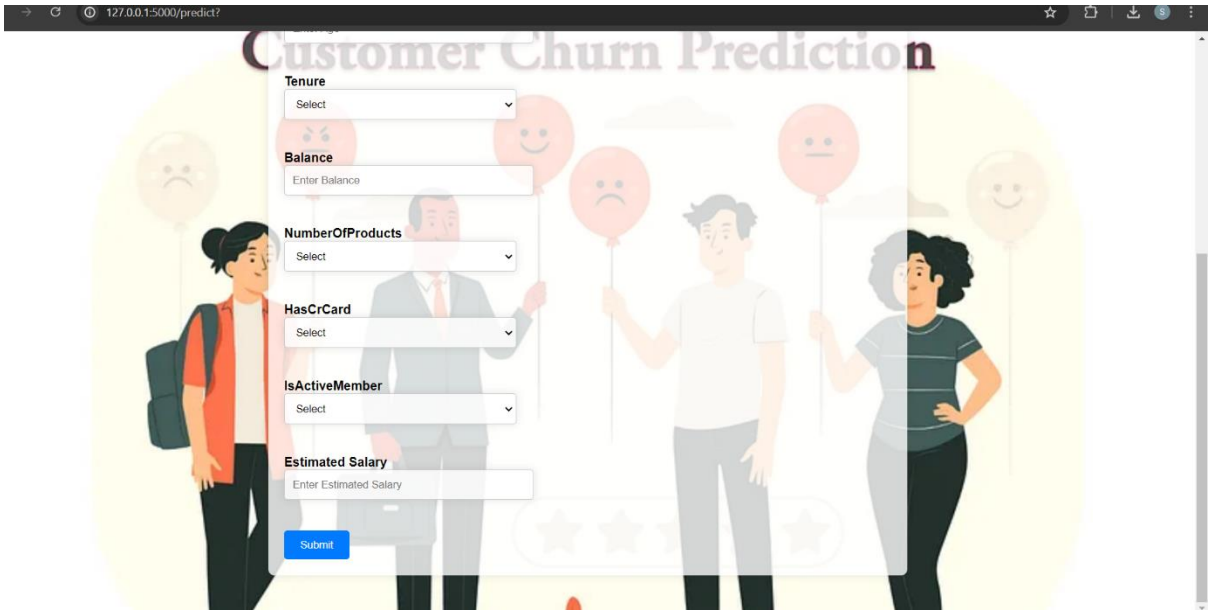
Now, Go to the web browser and write the localhost URL (http://127.0.0.1:5000) to get the below result



The above UI page is the index page. In the index page, it contains home, predict

On the page we have Get predict the button click on it. So, that it will be redirected to the inner page.

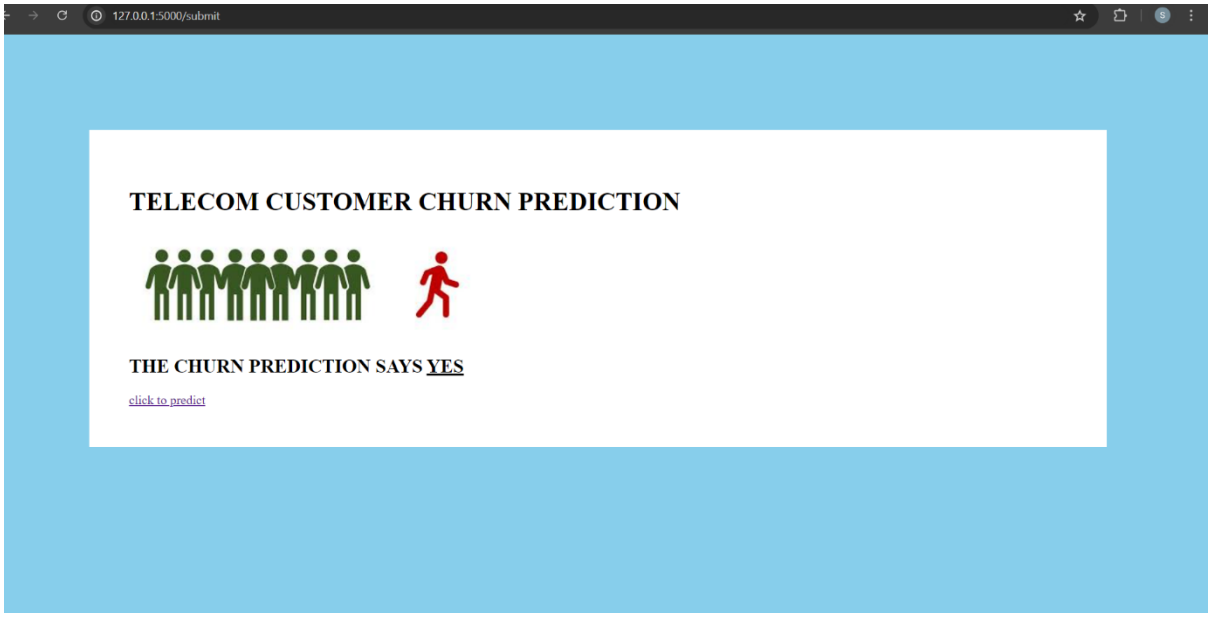




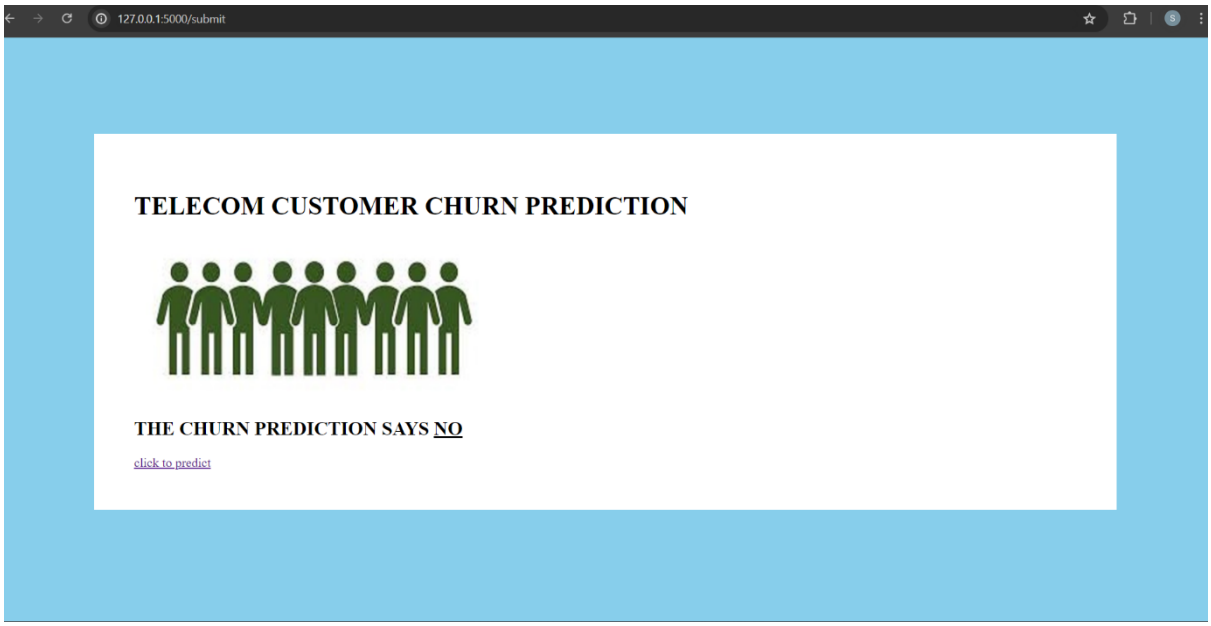
In this page gives the respected values for the respected fields .

Output :

In this final page the output that is customer churned or not.



If the customer has Churned.It is showing YES.



If the customer has not churned.It is showing NO.