PROJECT REPORT

INTRODUCTION

OVERVIEW:

Customer retention is a measure of how many customers stay with your business for the long term. It's what demonstrates your business's ability to stimulate customers to make repeat purchases and spend more money on your products and services over time.

PURPOSE:

The goal of customer retention programs is to help companies retain as many customers as possible, often through customer loyalty and brand loyalty initiatives

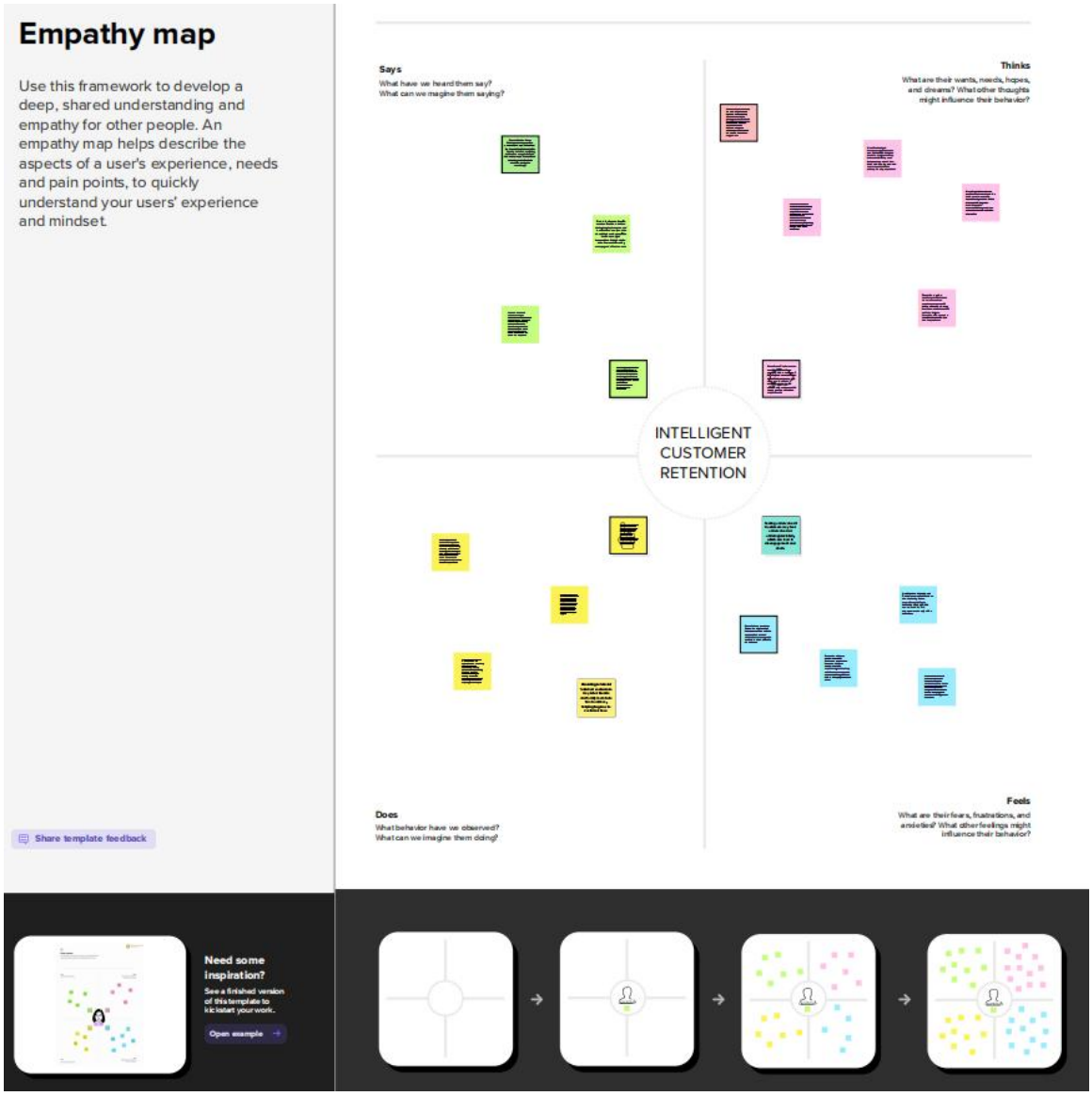
PROBLEM DEFINITION AND DESING THINKING :

Customer retention refers to the rate at which customers stay with a business in a given period of

time. This is often referred to as churn rate and is a key metric for practically all B2B and B2C businesses.

Ultimately, customer retention is about building relationships with your existing customers, providing value in every interaction, and giving them memorable experiences. It's about meeting customer expectations and building loyalty that encourages them to return to purchase your products or services over and over.

EMPATHY MAP :


Empathy map
Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.
Share template feedback
Says
What have we heard them say?
What can we imagine them saying?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
INTELLIGENT CUSTOMER RETENTION
Does
What behavior have we observed?
What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

BRAINSTORM:

IMPORT LIBRARIES:

```
#import necessary libraries
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import  RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

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.

```
#import dataset
data = pd.read_csv(r"C:\Users\Shivani_SB\OneDrive\Desktop\Telecom churn modelling-updated\data\DataSet.csv")
data
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	No	Yes
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	No	No

Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe 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.

```
data.describe()
```

	SeniorCitizen	tenure	MonthlyCharges
ount	7043.000000	7043.000000	7043.000000
ean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Visual analysis :

Visual analysis is the process of 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

LOGISTIC REGRESSION:

Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

```
#importing and building the Decision tree model
def logreg(x_train,x_test,y_train,y_test):
    lr = LogisticRegression(random_state=0)
    lr.fit(x_train,y_train)
    y_lr_tr = lr.predict(x_train)
    print(accuracy_score(y_lr_tr,y_train))
    yPred_lr = lr.predict(x_test)
    print(accuracy_score(yPred_lr,y_test))
    print("***Logistic Regression***")
    print("Confusion Matrix")
    print(confusion_matrix(y_test,yPred_lr))
    print("Classification Report")
    print(classification_report(y_test,yPred_lr))
```

```
#printing the train accuracy and test accuracy respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.7734960135298381
0.7734299516908213
***Logistic Regression***
Confusion Matrix
[[754 279]
 [190 847]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.73      0.76      1033
           1       0.75      0.82      0.78      1037

    accuracy                           0.77      2070
   macro avg       0.78      0.77      0.77      2070
weighted avg       0.78      0.77      0.77      2070
```

Decision tree model :

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
#importing and building the Decision tree model
def decisionTree(x_train,x_test,y_train,y_test):
    dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
    dtc.fit(x_train,y_train)
    y_dt_tr = dtc.predict(x_train)
    print(accuracy_score(y_dt_tr,y_train))
    yPred_dt = dtc.predict(x_test)
    print(accuracy_score(yPred_dt,y_test))
    print("***Decision Tree***")
    print("Confusion Matrix")
    print(confusion_matrix(y_test,yPred_dt))
    print("Classification Report")
    print(classification_report(y_test,yPred_dt))
```

```
#printing the train accuracy and test accuracy respectively
decisionTree(x_train,x_test,y_train,y_test)
```

```
0.9981879681082387
0.6067632850241546
***Decision Tree***
Confusion Matrix
[[ 242  791]
 [  23 1014]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.23      0.37      1033
           1       0.56      0.98      0.71      1037

    accuracy                           0.61      2070
   macro avg       0.74      0.61      0.54      2070
```

Random forest model:

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For

evaluating the model, a confusion matrix and classification report is done.

```python
#importing and building the random forest model
def RandomForest(x_tarin,x_test,y_train,y_test):
    rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
    rf.fit(x_train,y_train)
    y_rf_tr = rf.predict(x_train)
    print(accuracy_score(y_rf_tr,y_train))
    yPred_rf = rf.predict(x_test)
    print(accuracy_score(yPred_rf,y_test))
    print("***Random Forest***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_rf))
    print("Classification Report")
    print(classification_report(y_test,yPred_rf))
```

```python
#printing the train accuracy and test accuracy respectively
RandomForest(x_train,x_test,y_train,y_test)
```

```
0.9886446001449626
0.7536231884057971
***Random Forest***
Confusion_Matrix
[[563 470]
 [ 40 997]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.55      0.69      1033
           1       0.68      0.96      0.80      1037

    accuracy                           0.75      2070
   macro avg       0.81      0.75      0.74      2070
weighted avg       0.81      0.75      0.74      2070
```

KNN model

A function named KNN is created and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done

```
#importing and building the KNN model
def KNN(x_train,x_test,y_train,y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    y_knn_tr = knn.predict(x_train)
    print(accuracy_score(y_knn_tr,y_train))
    yPred_knn = knn.predict(x_test)
    print(accuracy_score(yPred_knn,y_test))
    print("***KNN***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_knn))
    print("Classification Report")
    print(classification_report(y_test,yPred_knn))
```

```
#printing the train accuracy and test accuracy respectively
KNN(x_train,x_test,y_train,y_test)
```

```
0.8570910848030925
0.7913043478260869
***KNN***
Confusion_Matrix
[[730 303]
 [129 908]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.71      0.77      1033
           1       0.75      0.88      0.81      1037

    accuracy                           0.79      2070
   macro avg       0.80      0.79      0.79      2070
weighted avg       0.80      0.79      0.79      2070
```

SVM model

“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate.

```
#importing and building the random forest model
def svm(x_tarin,x_test,y_train,y_test):
    svm = SVC(kernel = "linear")
    svm.fit(x_train,y_train)
    y_svm_tr = svm.predict(x_train)
    print(accuracy_score(y_svm_tr,y_train))
    yPred_svm = svm.predict(x_test)
    print(accuracy_score(yPred_svm,y_test))
    print("***Support Vector Machine***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_svm))
    print("Classification Report")
    print(classification_report(y_test,yPred_svm))
```

```
#printing the train accuracy and test accuracy respectively
svm(x_train,x_test,y_train,y_test)
```

```
0.7628654264315052
0.7555555555555555
***Support Vector Machine***
Confusion_Matrix
[[719 314]
 [192 845]]
Classification Report
              precision    recall  f1-score   support

           0       0.79      0.70      0.74      1033
           1       0.73      0.81      0.77      1037

    accuracy                           0.76      2070
   macro avg       0.76      0.76      0.75      2070
weighted avg       0.76      0.76      0.75      2070
```

ANN model

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epoch

ANN Model

```
[ ] # Importing the Keras libraries and packages
    import keras
    from keras.models import Sequential
    from keras.layers import Dense
```

```
[ ] # Initialising the ANN
    classifier = Sequential()
```

```
[ ] # Adding the input layer and the first hidden layer
    classifier.add(Dense(units=30, activation='relu', input_dim=40))
```

```
[ ] # Adding the second hidden layer
    classifier.add(Dense(units=30, activation='relu'))
```

```
[ ] # Adding the output layer
    classifier.add(Dense(units=1, activation='sigmoid'))
```

```
[ ] # Compiling the ANN
    classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Fitting the ANN to the Training set
model_history = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.33, epochs=200)
```

```
Epoch 1/200
555/555 [==============================] - 4s 3ms/step - loss: 0.5017 - accuracy: 0.7494 - val_loss: 0.4688 - val_accuracy: 0.7756
Epoch 2/200
555/555 [==============================] - 2s 3ms/step - loss: 0.4535 - accuracy: 0.7815 - val_loss: 0.4627 - val_accuracy: 0.7782
Epoch 3/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4424 - accuracy: 0.7865 - val_loss: 0.4691 - val_accuracy: 0.7778
Epoch 4/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4325 - accuracy: 0.7950 - val_loss: 0.4541 - val_accuracy: 0.7917
Epoch 5/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4239 - accuracy: 0.8002 - val_loss: 0.4536 - val_accuracy: 0.7892
Epoch 6/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4146 - accuracy: 0.8078 - val_loss: 0.4564 - val_accuracy: 0.7936
Epoch 7/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4058 - accuracy: 0.8100 - val_loss: 0.4551 - val_accuracy: 0.7921
Epoch 8/200
555/555 [==============================] - 1s 2ms/step - loss: 0.3999 - accuracy: 0.8150 - val_loss: 0.4510 - val_accuracy: 0.7943
```

```
Epoch 195/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1564 - accuracy: 0.9335 - val_loss: 0.7783 - val_accuracy: 0.8093
Epoch 196/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1514 - accuracy: 0.9347 - val_loss: 0.7982 - val_accuracy: 0.7994
Epoch 197/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1549 - accuracy: 0.9327 - val_loss: 0.8319 - val_accuracy: 0.7917
Epoch 198/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1593 - accuracy: 0.9320 - val_loss: 0.7693 - val_accuracy: 0.8130
Epoch 199/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1535 - accuracy: 0.9362 - val_loss: 0.7646 - val_accuracy: 0.8089
Epoch 200/200
555/555 [==============================] - 1s 3ms/step - loss: 0.1544 - accuracy: 0.9356 - val_loss: 0.7744 - val_accuracy: 0.8115
```

Building Html Pages:

- base.html
- index.html
- predyes.html
- predno.html

Build Python code:

Import the libraries

```
from flask import Flask, render_template, request
import keras
from keras.models import  load_model
```

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

```
app = Flask(__name__)
model = load_model("telcom_churn.h5")
```

Render HTML page:

```
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

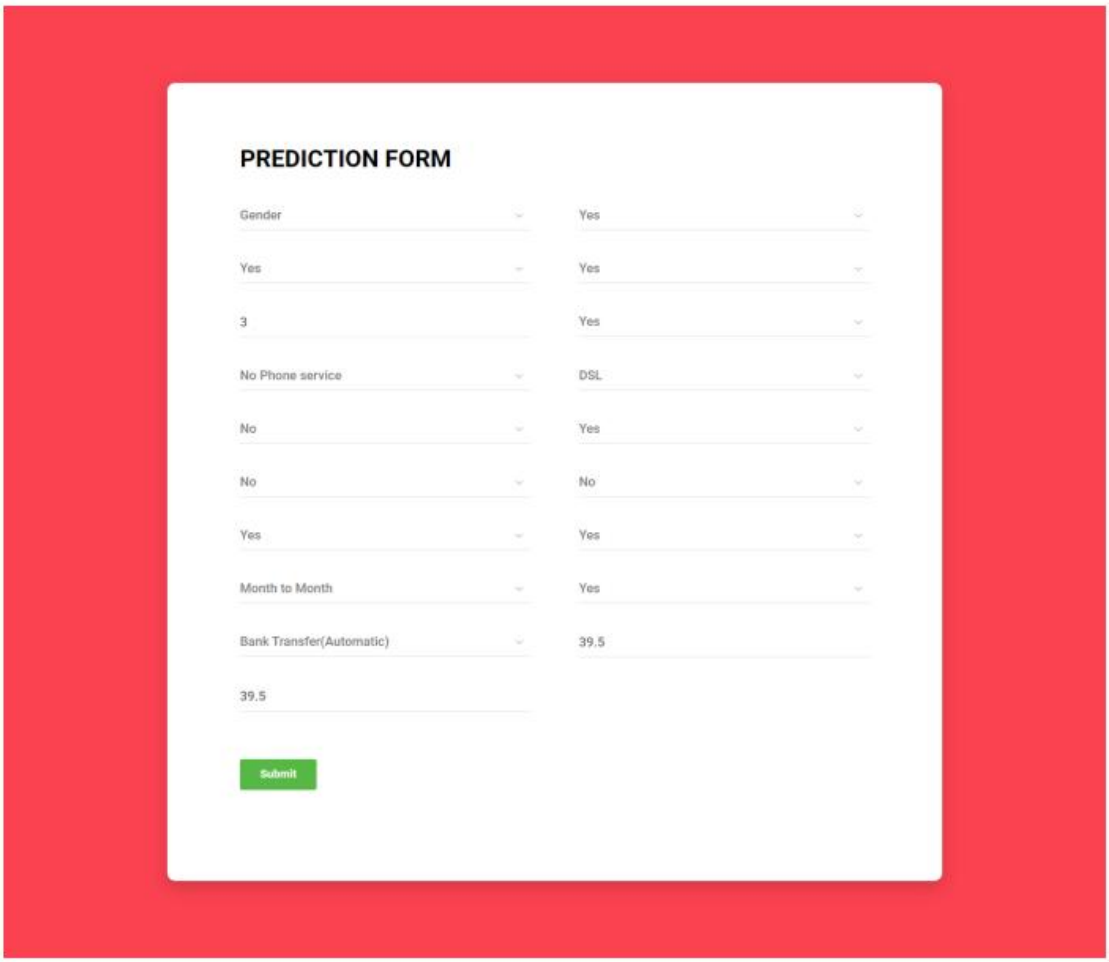
Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' 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: Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

TELECOM CUSTOMER CHURN PREDICTION

Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behaviour of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.

Click me to continue with prediction


PREDICTION FORM
Gender
Yes
Yes
Yes
Yes
3
Yes
No Phone service
DSL
No
Yes
No
No
Yes
Yes
Month to Month
Yes
Bank Transfer(Automatic)
39.5
39.5
Submit

TELECOM CUSTOMER CHURN PREDICTION

THE CHURN PREDICTION SAYS NO

TELECOM CUSTOMER CHURN PREDICTION

THE CHURN PREDICTION SAYS YES

END