# Introduction

The telecommunications sector has become one of the main industries in developed countries. The technical progress and the increasing number of operators raised the level of competition . Companies are working hard to survive in this competitive market depending on multiple strategies. Three main strategies have been proposed to generate more revenues : (1) acquire new customers, (2) upsell the existing customers, and (3) increase the retention period of customers. However, comparing these strategies taking the value of return on investment (RoI) of each into account has shown that the third strategy is the most profitable strategy , proves that retaining an existing customer costs much lower than acquiring a new one , in addition to being considered much easier than the upselling strategy . To apply the third strategy, companies have to decrease the potential of customer's churn, known as "the customer movement from one provider to another" .

## Overview :

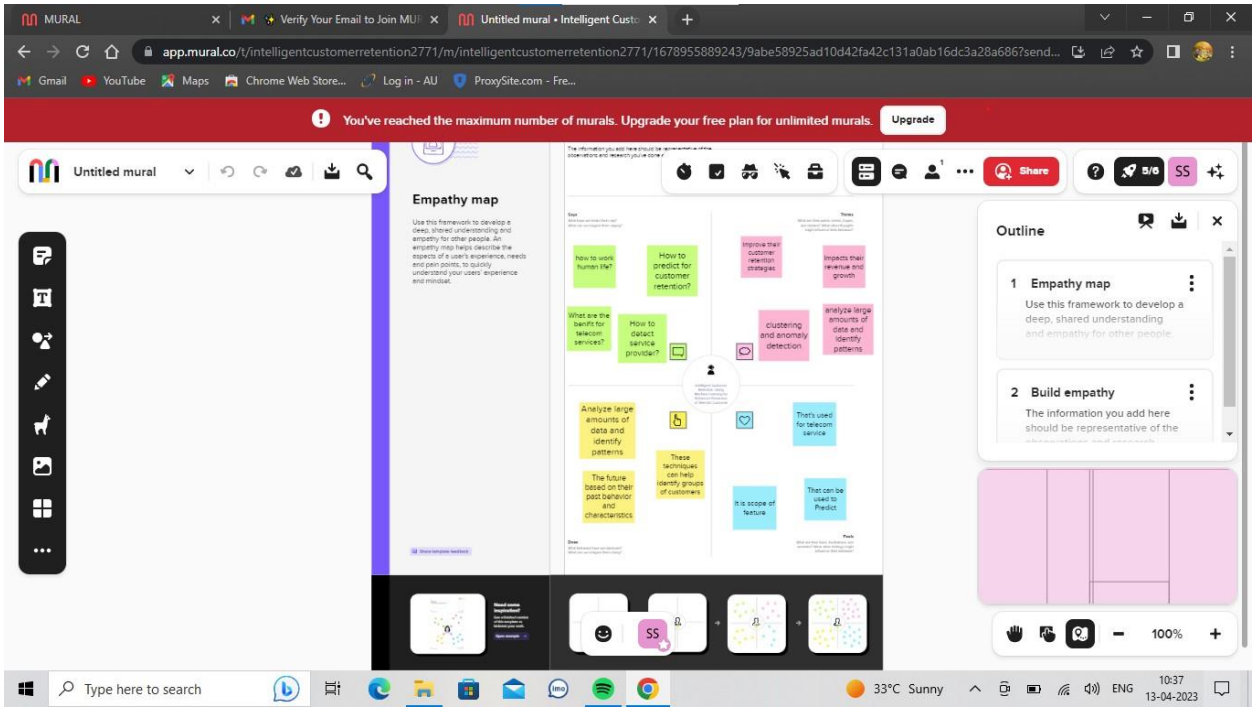
Telecommunications are the means of electronic transmission of information over distances. The information may be in the form of voice telephone calls, data, text, images, or video. Today, telecommunications are used to organize more or less remote computer systems into telecommunications networks.

## Purpose :

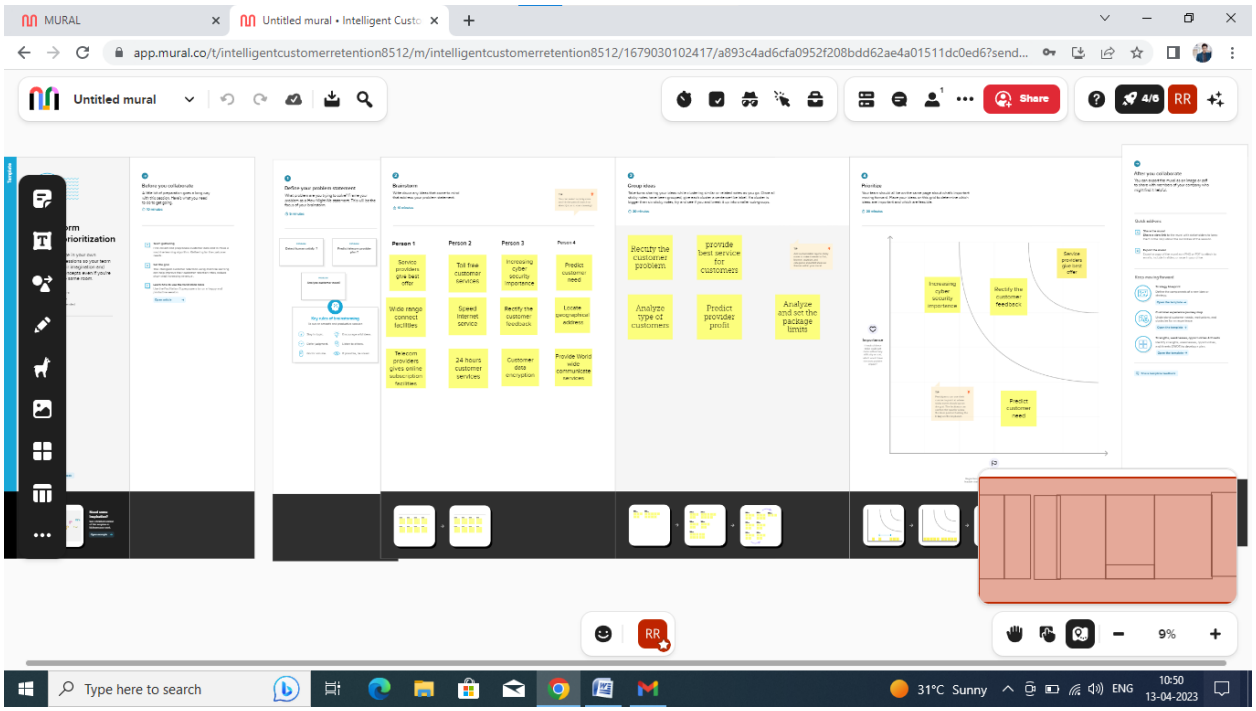
The purpose of a telecommunication system is to exchange information among users of the system. This information exchange can take place in a variety of ways, for example, multiparty voice communications, television, electronic mail, and electronic message exchange.

# Problem Definition & Design Thinking :

## Empathy map :




# Ideation & Brainstorming map :



## Related work

Many approaches were applied to predict churn in telecom companies. Most of these approaches have used machine learning and data mining. The majority of related work focused on applying only one method of data mining to extract knowledge, and the others focused on comparing several strategies to predict churn.

He et al. A proposed a model for prediction based on the Neural Network algorithm in order to solve the problem of customer churn in a large Chinese telecom company which contains about 5.23 million customers. The prediction accuracy standard was the overall accuracy rate, and reached 91.1%.

Idris. The proposed an approach based on genetic programming with AdaBoost to model the churn problem in telecommunications. The model was tested on two standard data sets. One by Orange Telecom and the other by cell2cell, with 89% accuracy for the cell2cell dataset and 63% for the other one.

## Data set

There are many types of data in SyriaTel used to build the churn model. These types are classified as follow:

1. 1.
   *Customer data* It contains all data related to customer's services and contract information. In addition to all offers, packages, and services subscribed to by the customer. Furthermore, it also contains information generated from CRM system like (all customer GSMs, Type of subscription, birthday, gender, the location of living and more ...).
2. 2.
   *Towers and complaints database* The information of action location is represented as digits. Mapping these digits with towers' database provides the location of this transaction, giving the longitude and latitude, sub-area, area, city, and state. Complaints' database provides all complaints submitted and statistics inquiries related to coverage, problems in offers and packages, and any problem related to the telecom business.
3. 3.
   *Network logs data* Contains the internal sessions related to internet, calls, and SMS for each transaction in Telecom operator, like the time needed to open a session for the internet and call ending status. It could indicate if the session dropped due to an error in the internal network.
4. 4.

*Call details records "CDRs"* Contain all charging information about calls, SMS, MMS, and internet transaction made by customers. This data source is generated as text files.

5. 5.
*Mobile IMEI information* It contains the brand, model, type of the mobile phone and if it's dual or mono SIM device.
This data has a large size and there is a lot of detailed information about it. We spent a lot of time to understand it and to know its sources and storing format. In addition to these records, the data must be linked to the detailed data stored in relational databases that contain detailed information about the customer. The nine months of data sets contained about ten million customers. The total number of columns is about ten thousand columns.

# Data exploration and challenges with SyriaTel dataset

Spark engine is used to explore the structure of this dataset, it was necessary to make the exploration phase and make the necessary pre-preparation so that the dataset becomes suitable for classification algorithms. After exploring the data, we found that about 50% of all numeric variables contain one or two discrete values, and nearly 80% of all the categorical variables have Less than 10 categories, 15% of the numerical variables and 33% of the categorical variables have only one value. Most of some variables' values are around zero.

### Data volume

Since we don't know the features that could be useful to predict the churn, we had to work on all the data that reflect the customer behavior in general.

### Data variety

The data used in this research is collected from multiple systems and databases. Each source generates the data in a different type of files as structured, semi-structured (XML-JSON) or unstructured (CSV-Text).

### Unbalanced dataset

The generated dataset was unbalanced since it is a special case of the classification problem where the distribution of a class is not usually homogeneous with other classes. The dominant class is called the basic class, and the other is called the secondary class.

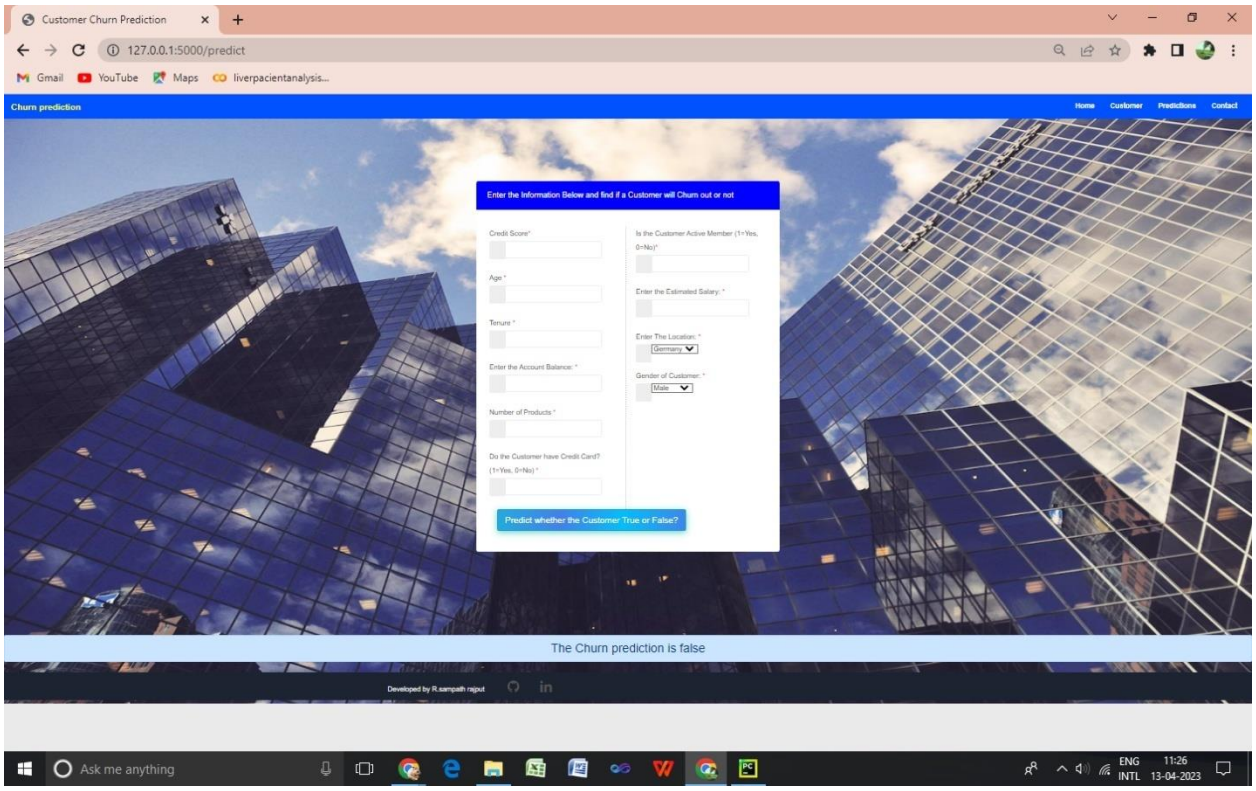
### Extensive features

The collected data was full of columns, since there is a column for each service, product, and offer related to calls, SMS, MMS, and internet, in addition to columns related to personnel and demographic information.

### Missing values

There is a representation of each service and product for each customer. Missing values may occur because not all customers have the same subscription.

# Result :



# Advantages & Disadvantages :

## Advantage:

- Improve efficiency
- Bring flexibility for workplace
- Save time
- Inspire collaboration
- Lack of travel time
- Do not have to drive distance
- Easy to access to the people so you need to contact with different people
- Worldwide access
- Next, bring a thing to being there
- Quick and accessible communication
- Lack of travel time
- Easy to exchange ideas and also basic information via phone and using fax
- Communication must be less effort in using transportation just to meet an individual personally
- Developed new product and invention
- Enable end user to communicate electronically and share software, hardware and data resources
- You just stay home and use a telephone or a cellphone if you want to talk to someone

## Disadvantages :

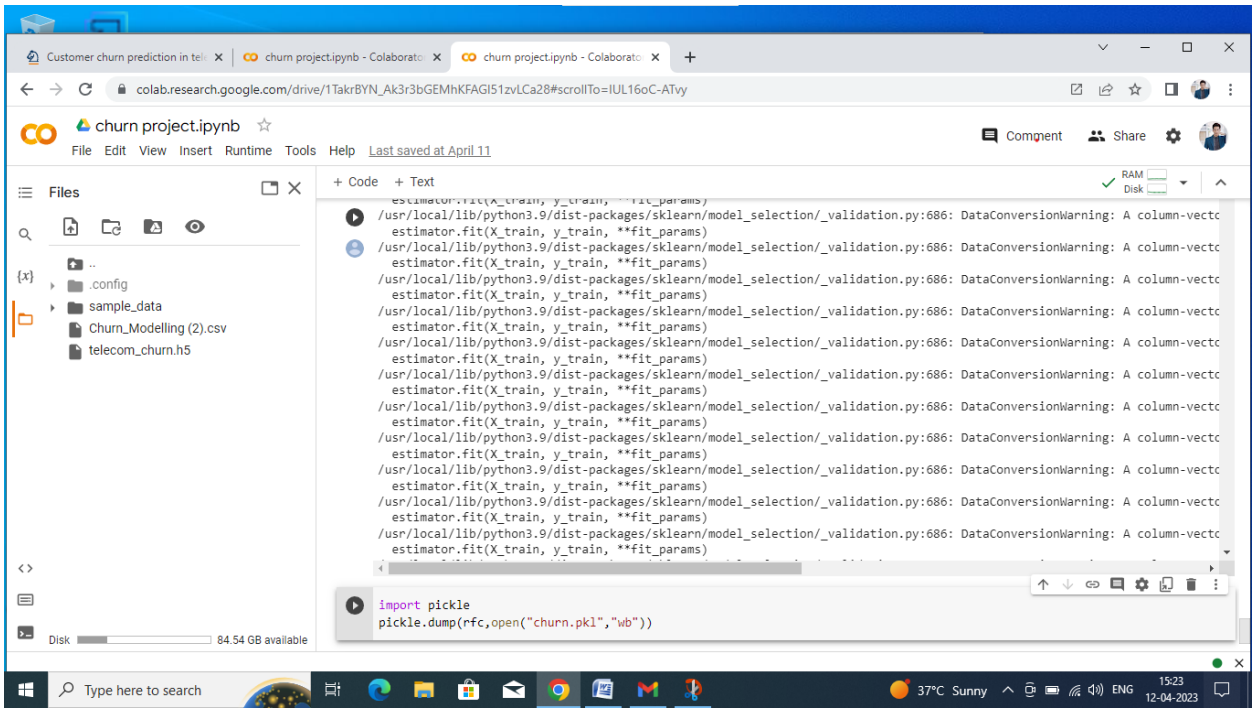
- Cultural barrier
- Prank calls
- High electric bills
- Sometimes expensive
- It can also be a source of low grades if abused
- Misunderstanding
- Remote areas do not have access
- Poor connection
- Cannot see whom you speaking with
- Eliminate face to face communication
- Increase vulnerability to information hacking and attacks

# Conclusion

The importance of this type of research in the telecom market is to help companies make more profit. It has become known that predicting churn is one of the most important sources of income to telecom companies. Hence, this research aimed to build a system that predicts the churn of customers in SyriaTel telecom company. These prediction models need to achieve high AUC values. To test and train the model, the sample data is divided into 70% for training and 30% for testing. We chose to perform cross-validation with 10-folds for validation and hyperparameter optimization. We have applied feature engineering, effective feature transformation and selection approach to make the features ready for machine learning algorithms. In addition, we encountered another problem: the data was not balanced. Only about 5% of the entries represent customers' churn. This problem was solved by undersampling or using trees algorithms not affected by this problem. Four tree based algorithms were chosen because of their diversity and applicability in this type of prediction.

## Output :

This is Final 5th Task Output In Google Colab.



# Future Scope :

**Improved accuracy in predicting customer churn:** Machine learning algorithms can analyze a large amount of data and identify patterns that may be missed by humans. By leveraging this data, telecom companies can accurately predict which customers are most likely to churn and take steps to retain them.

**Personalized customer retention strategies:** Machine learning algorithms can identify customer preferences and behaviors, allowing telecom companies to personalize their retention strategies. For example, a customer who frequently uses data may be offered a discount on data plans to encourage them to stay with the company.

**Reduced churn rates:** By using machine learning algorithms to identify and retain customers who are likely to churn, telecom companies can reduce their churn rates, leading to increased revenue and improved customer satisfaction.

**Enhanced customer experience:** By using machine learning algorithms to personalize retention strategies, telecom companies can improve the overall customer experience. This can lead to increased customer loyalty and improved brand reputation.

# Appendix :

## A. Source Code :

### Main code

```
import numpy as np
import pandas as pd
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn import preprocessing
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, GridSearchCV
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, confusio
n_matrix, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

df = pd.read_csv('/content/Churn_Modelling (2).csv')
df

df.info()
df.isnull().any()

df.isnull().sum()

le=LabelEncoder()
df["RowNumber"]=le.fit_transform(df["RowNumber"])
df["CustomerId"]=le.fit_transform(df["CustomerId"])
df["Surname"]=le.fit_transform(df["Surname"])
df["CreditScore"]=le.fit_transform(df["CreditScore"])
df["Geography"]=le.fit_transform(df["Geography"])
df["Gender"]=le.fit_transform(df["Gender"])
df["Age"]=le.fit_transform(df["Age"])
df["Tenure"]=le.fit_transform(df["Tenure"])
df["Balance"]=le.fit_transform(df["Balance"])
df["NumOfProducts"]=le.fit_transform(df["NumOfProducts"])
df["HasCrCard"]=le.fit_transform(df["HasCrCard"])
df["IsActiveMember"]=le.fit_transform(df["IsActiveMember"])
df["EstimatedSalary"]=le.fit_transform(df["EstimatedSalary"])
df["Exited"]=le.fit_transform(df["Exited"])

df

x=df.iloc[:,0:13].values
y=df.iloc[:,13:14].values

x,y

one=OneHotEncoder()
a=one.fit_transform(x[:,1:2]).toarray()
b=one.fit_transform(x[:,2:3]).toarray()
c=one.fit_transform(x[:,3:4]).toarray()
d=one.fit_transform(x[:,4:5]).toarray()
e=one.fit_transform(x[:,5:6]).toarray()
```

```
f=one.fit_transform(x[:,7:8]).toarray()
g=one.fit_transform(x[:,8:9]).toarray()
h=one.fit_transform(x[:,9:10]).toarray()
i=one.fit_transform(x[:,10:11]).toarray()
j=one.fit_transform(x[:,11:12]).toarray()
k=one.fit_transform(x[:,12:13]).toarray()
x=np.delete(x,[1,2,3,4,5,7,8,9,10,11,12],axis=1)
x=np.concatenate((a,b,c,d,e,f,g,h,i,k,x),axis=1)

x

smt=SMOTE()
x_resample,y_resample=smt.fit_resample(x,y)

x_resample,y_resample

x.shape,x_resample.shape

y.shape,y_resample.shape

df.describe()

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sns.distplot(df["Tenure"])
plt.subplot(1,2,2)
sns.distplot(df["CreditScore"])

sns.countplot(data=df, x="Gender")

sns.barplot(data=df, x="Tenure", y="Balance", color="b")
plt.title("Tenure VS Balance")

sns.heatmap(df.corr(),annot=True)

sns.pairplot(data=df, markers=["^","v"], palette="inferno")

x = df.drop(columns=["Exited"])
y = df["Balance"]
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,rand
om_state=0)
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

```
x_train.shape
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)

#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))


logreg(x_train, x_test, y_train, y_test)

#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))
decisionTree(x_train,x_test,y_train,y_test)
#importing and building the random forest model
def RandomForest(x_train,x_test,y_train,y_test):
  rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_s
tate=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))


RandomForest(x_train,x_test,y_train,y_test)
#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))


KNN(x_train,x_test,y_train,y_test)
#importing and building the ramdom forest mode1
def svm(x_train, 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))

# Call the function with appropriate arguments
svm(x_train, x_test, y_train, y_test)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam

# load data
x = np.random.rand(200, 40)  # example input data with 200 samples and 40
features
y = np.random.randint(0, 2, size=(200, 1))  # example target labels (binar
y)

# split data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
random_state=42)

# define the model
num_features = x_train.shape[1]
classifier = Sequential()
classifier.add(Dense(units=32, activation='relu', input_shape=(num_feature
s,)))
classifier.add(Dense(units=16, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))

# compile the model
optimizer = Adam(learning_rate=0.001)
classifier.compile(optimizer=optimizer, loss='binary_crossentropy', metric
s=['accuracy'])

# train the model
history = classifier.fit(x_train, y_train, batch_size=32, validation_split
=0.2, epochs=50)

model_history = classifier.fit(x_train, y_train, batch_size=10, validation
_split=0.33, epochs=200)

ann_pred = classifier.predict(x_test)
ann_pred = (ann_pred > 0.5)
print(ann_pred)
accuracy = accuracy_score(y_test, ann_pred)
conf_matrix = confusion_matrix(y_test, ann_pred)
class_report = classification_report(y_test, ann_pred)

print("***ANN Model***")
print("Accuracy Score:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

```
# create a StandardScaler object and fit it on the training data
sc = StandardScaler()
sc.fit(x_train)

# transform the training and test data with the scaler object
x_train_scaled = sc.transform(x_train)
x_test_scaled = sc.transform(x_test)

# create and train the logistic regression model
lr = LogisticRegression(random_state=0)
lr.fit(x_train_scaled, y_train)

# predict on test data and evaluate the model
lr_pred = lr.predict(x_test_scaled)
print("***Logistic Regression Model***")
print("Accuracy Score")
print(accuracy_score(lr_pred, y_test))
print("Confusion Matrix")
print(confusion_matrix(lr_pred, y_test))
print("Classification Report")
print(classification_report(lr_pred, y_test))

# predicting on randon input
lr_pred_own = lr.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,
1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("Predicted output is:", lr_pred_own)


#testing on random input values
dt = DecisionTreeClassifier(random_state=0)
dt.fit(x_train, y_train)

# predict on test data and evaluate the model
dt_pred = dt.predict(x_test)
print("***Decision Tree Classifier Model***")
print("Accuracy Score")
print(accuracy_score(dt_pred, y_test))
print("Confusion Matrix")
print(confusion_matrix(dt_pred, y_test))
print("Classification Report")
print(classification_report(dt_pred, y_test))

# predicting on randon input
```

```
dt_pred_own = dt.predict([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1
,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]])
print("Predicted output is:", dt_pred_own)
x_train_scaled = sc.transform(x_train)
x_test_scaled = sc.transform(x_test)

# create and train the random forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, criterion='gini',
 random_state=0)
rf_classifier.fit(x_train_scaled, y_train)

# predict on test data and evaluate the model
rf_pred = rf_classifier.predict(x_test_scaled)
print("***Random Forest Classifier Model***")
print("Accuracy Score")
print(accuracy_score(rf_pred, y_test))
print("Confusion Matrix")
print(confusion_matrix(rf_pred, y_test))
print("Classification Report")
print(classification_report(rf_pred, y_test))

# predict on a random input
random_input = sc.transform([[0]*40])
print("Predicting on random input")
rf_pred_own = rf_classifier.predict(random_input)
print("Output is:", rf_pred_own)
x_train_scaled = sc.transform(x_train)
x_test_scaled = sc.transform(x_test)

# create and train the SVM classifier
svm_classifier = SVC(kernel='linear', random_state=0)
svm_classifier.fit(x_train_scaled, y_train)

# predict on test data and evaluate the model
svm_pred = svm_classifier.predict(x_test_scaled)
print("***SVM Classifier Model***")
print("Accuracy Score")
print(accuracy_score(svm_pred, y_test))
print("Confusion Matrix")
print(confusion_matrix(svm_pred, y_test))
print("Classification Report")
print(classification_report(svm_pred, y_test))

# predict on a random input
random_input = sc.transform([[0]*40])
```

```
print("Predicting on random input")
svm_pred_own = svm_classifier.predict(random_input)
print("Output is:", svm_pred_own)
knn_classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p
=2)
knn_classifier.fit(x_train_scaled, y_train)

# predict on test data and evaluate the model
knn_pred = knn_classifier.predict(x_test_scaled)
print("***KNN Classifier Model***")
print("Accuracy Score")
print(accuracy_score(knn_pred, y_test))
print("Confusion Matrix")
print(confusion_matrix(knn_pred, y_test))
print("Classification Report")
print(classification_report(knn_pred, y_test))

# predict on a random input
random_input = sc.transform([[0]*40])
print("Predicting on random input")
knn_pred_own = knn_classifier.predict(random_input)
print("Output is:", knn_pred_own)
classifier = Sequential()
classifier.add(Dense(units=30, activation='relu', input_dim=40))
classifier.add(Dense(units=30, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))
# Random Forest Classifier without hyperparameter tuning
rfc = RandomForestClassifier(random_state=42)
rfc.fit(x_train, y_train)
rfc_pred = rfc.predict(x_test)

print("***RFC Model without Hyperparameter Tuning***")
print("Accuracy:", accuracy_score(y_test, rfc_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, rfc_pred))
print("Classification Report:\n", classification_report(y_test, rfc_pred))

# Random Forest Classifier with hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30],
    'min_samples_leaf': [1, 2, 4]
}

rfc_tuned = RandomForestClassifier(random_state=42)
rfc_cv = GridSearchCV(rfc_tuned, param_grid, cv=5)
```

```
rfc_cv.fit(x_train, y_train)
rfc_tuned_pred = rfc_cv.predict(x_test)

print("***RFC Model with Hyperparameter Tuning***")
print("Best Parameters:", rfc_cv.best_params_)
print("Accuracy:", accuracy_score(y_test, rfc_tuned_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, rfc_tuned_pred))
print("Classification Report:\n", classification_report(y_test, rfc_tuned_
pred))

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

# train the model
model_history = classifier.fit(x_train_scaled, y_train, batch_size=10, val
idation_split=0.33, epochs=200)

# predict on test data and evaluate the model
ann_pred = classifier.predict(x_test_scaled)
ann_pred = (ann_pred>0.5)
print("***ANN Classifier Model***")
print("Accuracy Score")
print(accuracy_score(ann_pred, y_test))
print("Confusion Matrix")
print(confusion_matrix(ann_pred, y_test))
print("Classification Report")
print(classification_report(ann_pred, y_test))

# predict on a random input
random_input = sc.transform([[0]*40])
print("Predicting on random input")
ann_pred_own = classifier.predict(random_input)
ann_pred_own = (ann_pred_own>0.5)
print("Output is:", ann_pred_own)
import pickle
pickle.dump(rfc,open("churn.pkl","wb"))
```

# App.py

```
import pickle

from flask import Flask, render_template, request
from sklearn.preprocessing import StandardScaler

app = Flask(__name__)
model = pickle.load(open('Customer_Churn_Prediction.pkl', 'rb'))
@app.route('/', methods=['GET'])
def Home():
    return render_template('index.html')

standard_to = StandardScaler()
@app.route('/predict',methods=['POST'])
def predict():
    if request.method == 'POST':
        CreditScore = int(request.form['CreditScore'])
        Age = int(request.form['Age'])
        Tenure = int(request.form['Tenure'])
        Balance = float(request.form['Balance'])
        NumOfProducts = int(request.form['NumOfProducts'])
        HasCrCard = int(request.form['HasCrCard'])
        IsActiveMember = int(request.form['IsActiveMember'])
        EstimatedSalary = float(request.form['EstimatedSalary'])
        Geography_Germany = request.form['Geography_Germany']
        if(Geography_Germany == 'Germany'):
            Geography_Germany = 1
            Geography_Spain= 0
            Geography_France = 0

        elif(Geography_Germany == 'Spain'):
            Geography_Germany = 0
            Geography_Spain= 1
            Geography_France = 0

        else:
            Geography_Germany = 0
            Geography_Spain= 0
            Geography_France = 1
        Gender_Male = request.form['Gender_Male']
        if(Gender_Male == 'Male'):
            Gender_Male = 1
            Gender_Female = 0
        else:
            Gender_Male = 0
```

```
        Gender_Female = 1
    prediction =
model.predict([[CreditScore,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,Estimated
Salary,Geography_Germany,Geography_Spain,Gender_Male]])
    if prediction==1:
        return render_template('index.html',prediction_text="The Churn prediction is True")
    else:
        return render_template('index.html',prediction_text="The Churn prediction is false")

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

## HTML CODE:

```
<!DOCTYPE html>

<html lang="en">



<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Customer Churn Prediction</title>

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css" integrity="sha512-
SfTiTlX6kk+qitfevl/7LibUOeJWlt9rbyDn92a1DqWOw9vWG2MFoays0sgObmWazO5BQPiFuc
nnEAjpAB+/Sw==" crossorigin="anonymous" />

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min
.css" integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">

</head>



<body>
```

```
    <!-- Navbar Starts -->

    <section id="nav-bar">

    <nav class="navbar navbar-expand-lg navbar-light bg-light">

    <a class="navbar-brand font-weight-bold" href="#" style="color:
yellow;">Churn prediction</a>

    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarNav">

        <ul class="navbar-nav ml-auto">

             <li class="nav-item active">

                <a class="nav-link" href="#">Home <span class="sr-
only">(current)</span></a>

             </li>

             <li class="nav-item">

                <a class="nav-link" href="#">Customer</a>

             </li>

             <li class="nav-item">

                <a class="nav-link" href="#">Predictions</a>

             </li>

              <li class="nav-item">

                <a class="nav-link" href="#">Contact</a>
```

```
            </li>


        </ul>
    </div>
    </nav>
    </section>
    <!-- Navbar Ends -->




        <form action="{{ url_for('predict')}}" method="post">
          <h1>Enter the Information Below and find if a Customer will
Churn out or not</h1>


    <div class="contentform">
      <div id="sendmessage"> Your message has been sent successfully.
Thank you. </div>



      <div class="leftcontact">
                        <div class="form-group">
                          <p>Credit Score<span>*</span></p>
                          <span class="icon-case"></span>
                               <input name="CreditScore"
name="CreditScore" type="number" >
                <div class="validation"></div>
```

```
</div>

      <div class="form-group">
      <p>Age <span>*</span></p>
      <span class="icon-case"></span>
                  <input name="Age" required="required">
          <div class="validation"></div>
           </div>

           <div class="form-group">
           <p>Tenure <span>*</span></p>
           <span class="icon-case"></span>
          <input name="Tenure" required="required">
          <div class="validation"></div>
           </div>

           <div class="form-group">
           <p>Enter the Account Balance:  <span>*</span></p>
           <span class="icon-case"></span>
                 <input name="Balance" required="required">
          <div class="validation"></div>
           </div>

           <div class="form-group">
           <p>Number of Products <span>*</span></p>
```

```
                <span class="icon-case"></span>

                      <input name="NumOfProducts" required="required">

               <div class="validation"></div>

                </div>


                <div class="form-group">

                <p>Do the Customer have Credit Card? (1=Yes, 0=No)
<span>*</span></p>

                <span class="icon-case"></span>

                      <input name="HasCrCard" required="required">

               <div class="validation"></div>

                </div>




      </div>


      <div class="rightcontact">


                <div class="form-group">

                <p>Is the Customer Active Member (1=Yes,
0=No)<span>*</span></p>

                <span class="icon-case"></span>

                      <input name="IsActiveMember" required="required">

               <div class="validation"></div>

                </div>
```

```
      <div class="form-group">
      <p>Enter the Estimated Salary: <span>*</span></p>
      <span class="icon-case"></span>
            <input name="EstimatedSalary" required="required">
     <div class="validation"></div>
      </div>

      <div class="form-group">
      <p>Enter The Location: <span>*</span></p>
      <span class="icon-case"></span>
     <select name="Geography_Germany" required="required">
     <option value="Germany">Germany</option>
     <option value="Spain">Spain</option>
     <option value="France">France</option>
</select>
<div class="validation"></div>
      </div>

      <div class="form-group">
      <p>Gender of Customer:  <span>*</span></p>
      <span class="icon-case"></span>
     <select name="Gender_Male" id="resea" required="required">
     <option value="Male">Male</option>
     <option value="Female">Female</option>
```

```
            </select>

                <div class="validation"></div>

                 </div>




      </div>

      </div>

<button  type="submit " class="btn btn-lg btn-info mb-5 ml-5" style="box-shadow: 0 4px 15px 0 rgba(49, 196, 190, 0.75); background-image: linear-gradient(to right, #25aae1, #4481eb, #04befe, #3f86ed);">Predict whether the Customer True or False?</button>



</form>









        <br><br><h3 class="alert alert-primary text-center" role="alert">{{ prediction_text }}<h3>

    </div>

    <!-- Footer -->

        <!-- Footer -->

    <div class="unique-color-dark mt-4 white-text " style="background-color: #1c2331 !important;">

      <div class="blue-gradient">

        <div class="container">
```

```
          <!-- Grid row-->

          <div class="row py-3 d-flex align-items-center">

            <!-- Grid column -->

            <div

              class="col-md-6 col-lg-5 text-center text-md-left"

            >

              <h6 class="mb-0 text-white">Developed by R.sampath rajput
<a href="samsampath2003@gmail.com"><i class="ml-5 fa fa-github fa-2x"
aria-hidden="true"></i></a><i class="ml-5 fa fa-linkedin fa-2x" aria-
hidden="true"></i></a></h6>

            </div>

            <!-- Grid column -->




            <!-- Grid column -->

          </div>

          <!-- Grid row-->

        </div>

      </div>







    <style>

        * {

    margin: 0;
```

```
    padding: 0;

}

body {

    font-family: sans-serif;

    background-image: url(https://images.unsplash.com/photo-1462206092226-
f46025ffe607?ixlib=rb-
1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1353&q=80)
!important;

    background-repeat: no-repeat !important;

    background-size: 100% 100%!important;

}

#nav-bar {

    position: sticky;

    top: 0;

    z-index: 10;

}

.navbar {

    background: rgb(0, 81, 255) !important;

}

.navbar-nav li {

    padding: 0 10px;

}

.navbar-nav li a {

    color: #fff !important;

    font-weight: 600;

}
```

```
.navbar-toggler {
    outline: none !important;
}
 #banner {
         background: rgb(0, 81, 255) !important;
         color: #fff;
         padding-top: 5%;
      }
      .promo-title {
        font-size: 40px;
        font-weight: 600;
        margin-top: 100px;
      }
      body {
  margin: auto;
  background: #eaeaea;
  font-family: 'Open Sans', sans-serif;
}
 .info p {
  text-align:center;
  color: #999;
  text-transform:none;
  font-weight:600;
  font-size:15px;
  margin-top:2px
```

```
}


.info i {
  color:#F6AA93;
}
form h1 {
  font-size: 18px;
  background:blue none repeat scroll 0% 0%;
  color: rgb(255, 255, 255);
  padding: 22px 25px;
  border-radius: 5px 5px 0px 0px;
  margin: auto;
  text-shadow: none;
  text-align:left
}


form {
  border-radius: 5px;
  max-width:700px;
  width:100%;
  margin: 5% auto;
  background-color: #FFFFFF;
  overflow: hidden;
}
```

```
p span {
  color: #F00;
}

p {
  margin: 0px;
  font-weight: 500;
  line-height: 2;
  color:#333;
}

h1 {
  text-align:center;
  color: #666;
  text-shadow: 1px 1px 0px #FFF;
  margin:50px 0px 0px 0px
}

input {
  border-radius: 0px 5px 5px 0px;
  border: 1px solid #eee;
  margin-bottom: 15px;
  width: 75%;
  height: 40px;
  float: left;
```

```
  padding: 0px 15px;
}

a {
  text-decoration:inherit
}

textarea {
  border-radius: 0px 5px 5px 0px;
  border: 1px solid #EEE;
  margin: 0;
  width: 75%;
  height: 130px;
  float: left;
  padding: 0px 15px;
}

.form-group {
  overflow: hidden;
  clear: both;
}

.icon-case {
  width: 35px;
  float: left;
```

```
  border-radius: 5px 0px 0px 5px;
  background:#eeeeee;
  height:42px;
  position: relative;
  text-align: center;
  line-height:40px;
}

i {
  color:#555;
}

.contentform {
  padding: 40px 30px;
}

.bouton-contact{
  background-color: #81BDA4;
  color: #FFF;
  text-align: center;
  width: 100%;
  border:0;
  padding: 17px 25px;
  border-radius: 0px 0px 5px 5px;
  cursor: pointer;
```

```
  margin-top: 40px;
  font-size: 18px;
}

.leftcontact {
  width:49.5%;
  float:left;
  border-right: 1px dotted #CCC;
  box-sizing: border-box;
  padding: 0px 15px 0px 0px;
}

.rightcontact {
  width:49.5%;
  float:right;
  box-sizing: border-box;
  padding: 0px 0px 0px 15px;
}

.validation {
  display:none;
  margin: 0 0 10px;
  font-weight:400;
  font-size:13px;
  color: #DE5959;
```

```
}

#sendmessage {
  border:1px solid #fff;
  display:none;
  text-align:center;
  margin:10px 0;
  font-weight:600;
  margin-bottom:30px;
  background-color: #EBF6E0;
  color: #5F9025;
  border: 1px solid #B3DC82;
  padding: 13px 40px 13px 18px;
  border-radius: 3px;
  box-shadow: 0px 1px 1px 0px rgba(0, 0, 0, 0.03);
}

#sendmessage.show,.show  {
  display:block;
}




    </style>
```

```
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>

    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.j
s" integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
crossorigin="anonymous"></script>

</body>



</html>
```