Intelligent Customer Retention: Using Machine Learning for Enhanced Prediction of Telecom Customer Churn

1. INTRODUCTION

1.1 Overview

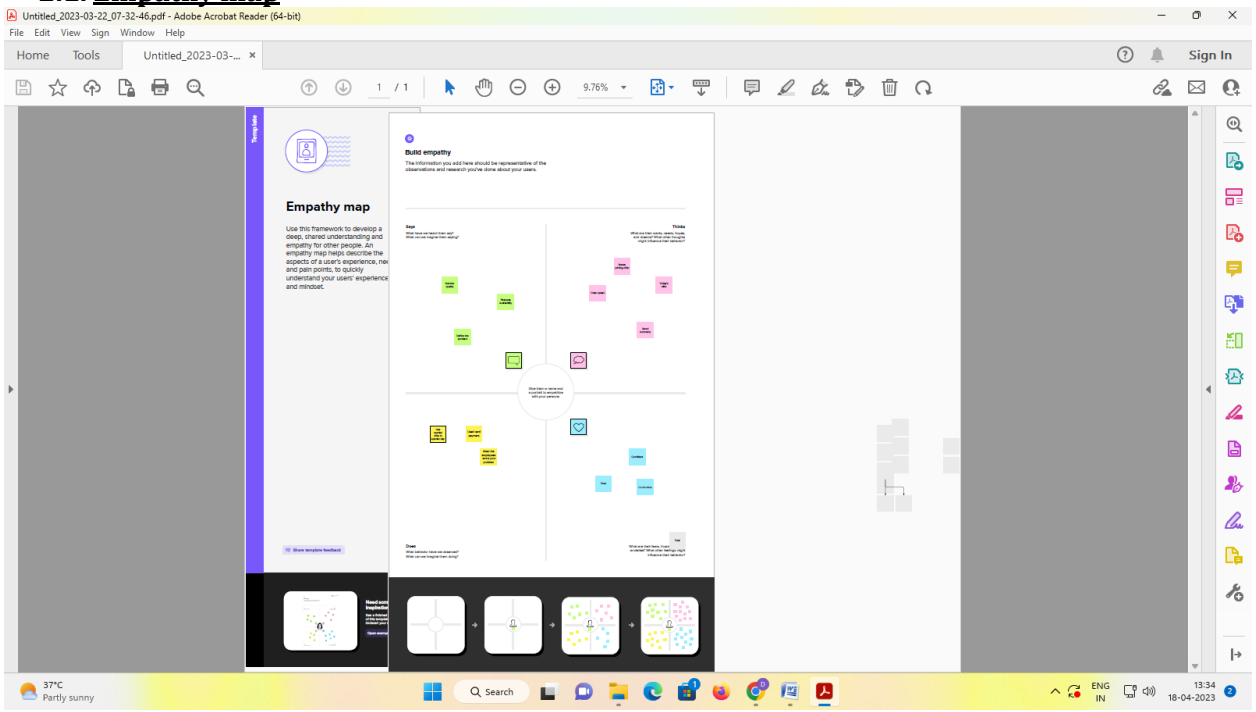
- **Customer churn is often referred to as customer attrition, or customer defection which is the rate at which the customers are lost. Customer churn is a major problem and one of the most important concerns for large companies. Due to the direct effect on the revenues of the companies, especially in the telecom field, companies are seeking to develop means to predict potential customer to churn. Looking at churn, different reasons trigger customers to terminate their contracts, for example better price offers, more interesting packages, bad service experiences or change of customers' personal situations.**

1.2 Purpose

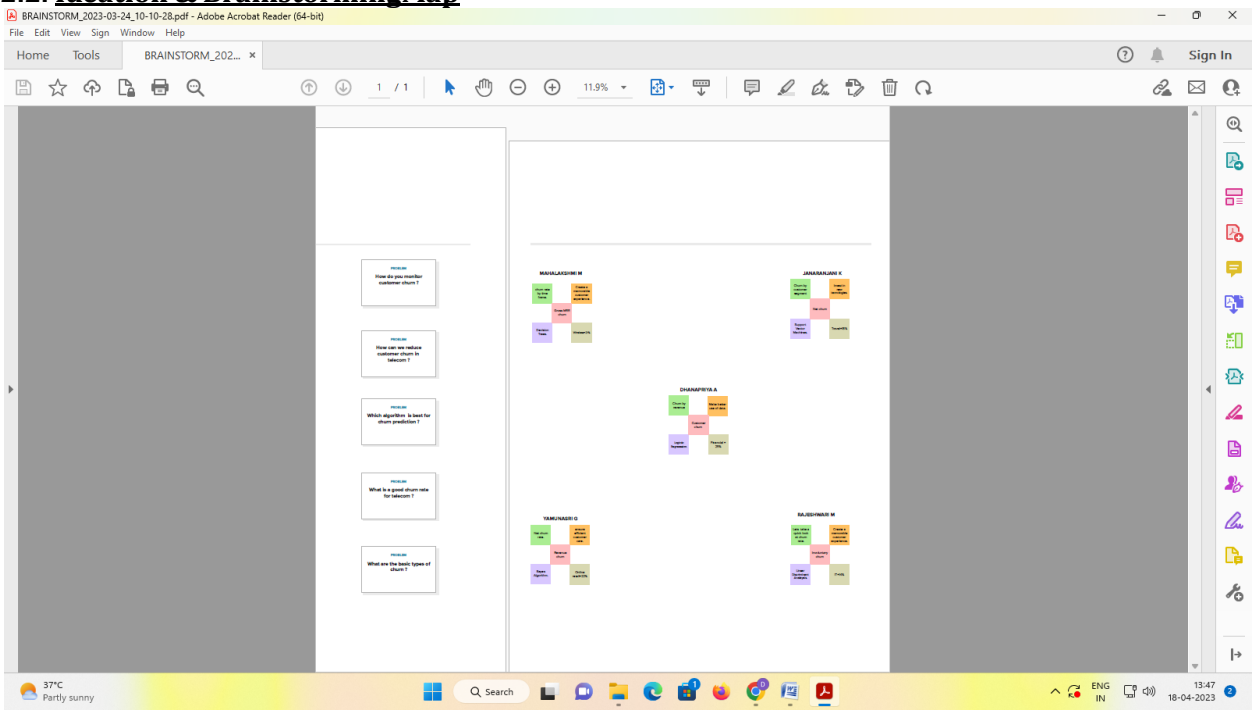
- **Companies can classify potential customers who leave the services using advanced machine learning (ML) technology. Then, using existing data, the company can identify potential churn customers. This knowledge would allow the company to target those customers and recover them.**
- **Churn analysis helps you identify pain points throughout the entire customer journey. Understanding those pain points then opens up avenues to improve your products, services, and communication. Sure, customer churn is inevitable.**

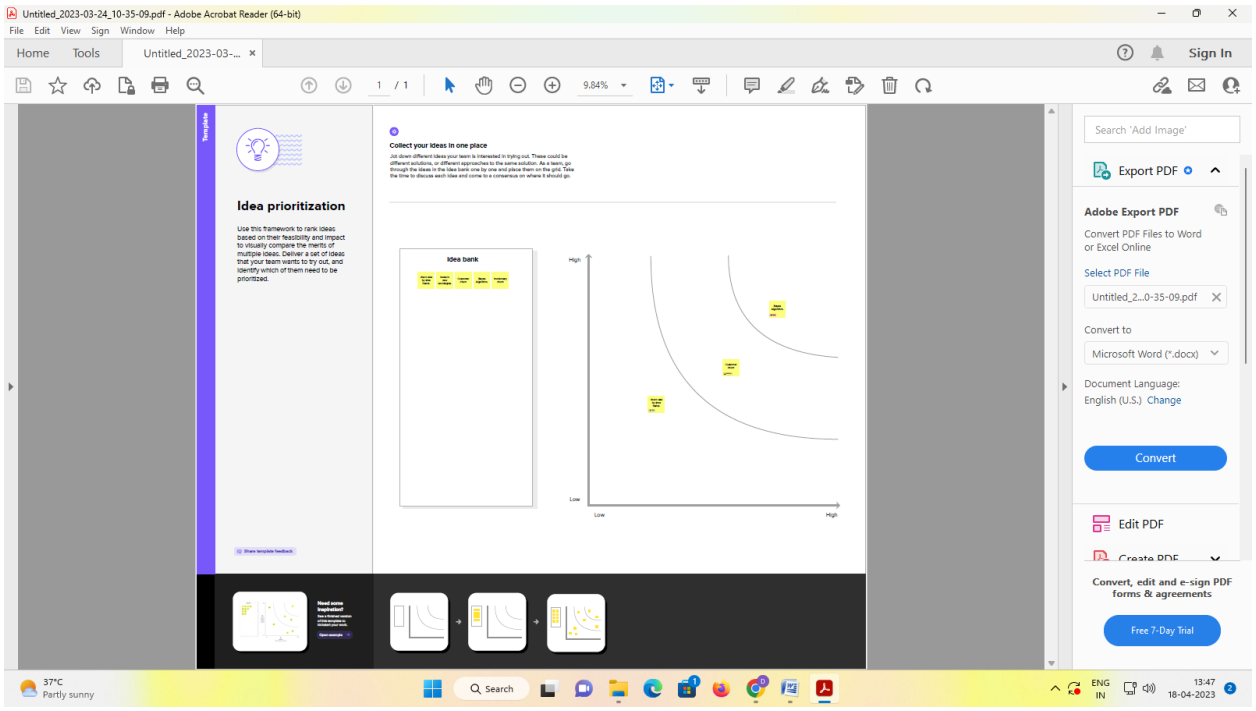
2.PROBLEM DEFINITION & DESIGN THINKING

2.1. Empathy map



2.2. Ideation & BrainstormingMap




Untitled_2023-03-24_10-35-09.pdf - Adobe Acrobat Reader (64-bit)
File Edit View Sign Window Help
Home Tools
Sign In
Idea prioritization
Use this framework to rank ideas based on their feasibility and impact to visually compare the merits of multiple ideas. Deliver a set of ideas that your team wants to try out, and identify which of them need to be prioritized.
Collect your ideas in one place
Idea bank
High
Low
Low
High
Export PDF
Adobe Export PDF
Convert PDF Files to Word or Excel Online
Select PDF File
Convert to
Microsoft Word (*.docx)
Document Language:
English (U.S.) Change
Convert
Edit PDF
Convert, edit and e-sign PDF forms & agreements
Free 7-Day Trial
37°C
Partly sunny
Search
ENG
IN
13:47
18-04-2023

3. RESULT

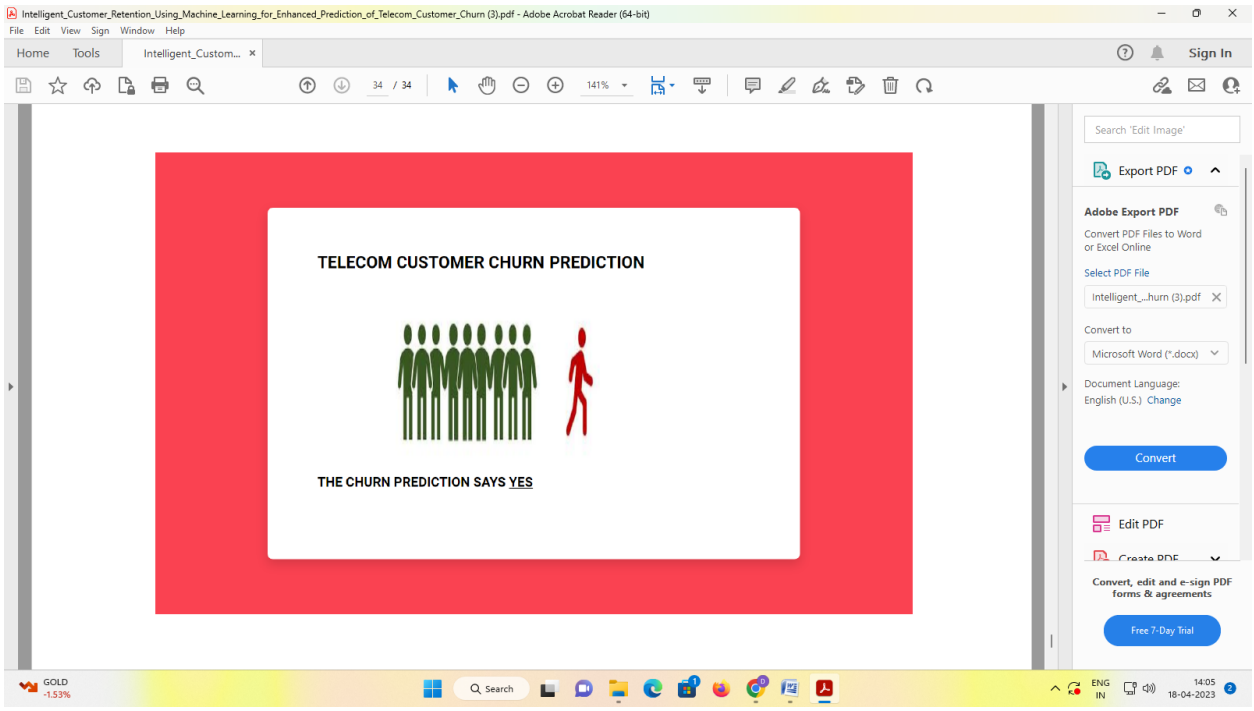
PREDICTION FORM

Gender	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



4. ADVANTAGES & DISADVANTAGES

➢ ADVANTAGES

- Customer churn analysis helps businesses understand why customers don't return for repeat business. Churn rate tells you what portion of your customers leave over a period of time. It's often useful to look at churn by product, region or other granular factors.

- Loyal customers are essential for a business to succeed and a low customer churn rate helps the business gain a competitive advantage. You can calculate this metric by looking at how many customers are supporting the business.
- The benefits of preventing churn are pretty obvious: keep more customers, unlock more growth, continue to scale your business. Awesome. But it goes even further than this. Reducing your churn can have staggering impacts all across the board; from employee retention to customer loyalty, all the way down to LTV and CAC.

➢ DISADVANTAGES

- Reduced Monthly or Annually Recurring Revenue. Lost customers equal lost revenue. ...
- Lower Customer Lifetime Value. ...
- Shrinking Customer Base. ...
- Added Win-Back Expenses

- higher staff wages from hiring employees who are experts in customer service.
- paying for staff training.
- the extra services offered, such as refreshments.
- higher wage costs from the extra time staff take to provide post-sales service.

5. APPLICATION

- Churn prediction means detecting which customers are likely to leave a service or to cancel a subscription to a service. It is a critical prediction for many businesses because acquiring new clients often costs more than retaining existing ones.

- Determining an effective predictive model helps detect early warning signals when churn occurs and assigns to each customer a score called "churn score" that indicates the likelihood that the individual might migrate to another network over a predefined time period.

6. CONCLUSION

- With machine learning and deep analysis for customer churn prediction, your company will be able to increase customer retention rate, save up on retention costs and even protect the future revenue from churning users.

7. FUTURE SCOPE

- Predicting Customer Churn. Churn prediction means detecting which customers are likely to leave a service or to cancel a subscription to a service**. It is a** critical prediction for many businesses because acquiring new clients often costs more than retaining existing ones.

8.APPENDIX

```
from ast import increment_lineno
import pandas as pdSSSSS
import numpy as np
import pickle
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
data = pd.read_csv(r"/content/Churn_Modelling.csv")
data
data.info()
data.Total Charges=pd.to_numeric(data.Total Charges, errors='coerce')
data.isnull().sum()
data["Total Charges"].fillna(data["Total Charges"].median(), inplace =True)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["Partner"]=le.fit_transform(data["Partner"])
data["Depandents"]=le.fit_transform(data["Dependents"])
data["PhoneService"]=le.fit_transform(data["PhoneService"])
data["MultipleLines"]=le.fit_transform(data["Multiplelines"])
data["InternetService"]=le.fit_transform(data["InternetService"])
data["OnlineSecurity"]=le.fit_transform(data["OnlineSecurity"])
data["OnlineBackup"]=le.fit_transform(data["onlineBackup"])
data["DeviceProtection"]=le.fit_transform(data["DeviceProtection"])
data["TechSupport"]=le.fit_transform(data["TechSupport"])
data["StreamingTV"]=le.fit_transform(data["StreamingTv"])
data["StreamingMovies"]=le.fit_transform(data["StreamingMovies"])
data["Contract"]=le.fit_transform(data["Contract"])
data["PaperLessingBilling"]=le.fit_transform(data["PaperLessingBilling"])
data["PaymentMethod"]=le.fit_transform(data["PaymentMethod"])
data["Churn"]=le.fit_transform(data["Churn"])
data.head()
x=data.iloc[:,0:19].values
y=data.iloc[:,19:20].values
x
y
from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
a=one.fit_transform(x[:,6:7]).toarray()
b=one.fit_transform(x[:,7:8]).toarray()
c=one.fit_transform(x[:,8:9]).toarray()
d=one.fit_transform(x[:,9:10]).toarray()
e=one.fit_transform(x[:,10:11]).toarray()
f=one.fit_transform(x[:,11:12]).toarray()
g=one.fit_transform(x[:,12:13]).toarray()
i=one.fit_transform(x[:,14:15]).toarray()
j=one.fit_transform(x[:,16:17]).toarray()
```

```
x=np.delete(x,[6,7,8,9,10,11,12,13],axis=1)
x=np.concatenate((a,b,c,d,e,f,g,h),axis=1)
from imblearn.over_sampling import SMOTE
smt=SMOTE()
x_resample,y_resample=smt.fit_resample(x,y)
x_resample
y_resampledata.describe()
data.describe()
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.displot(data["tenure"])
sns.displot(data["MonthCharges"])
plt.figure(figsize=12,5))
pltsns.countplot(data["{gender"])
plt.subplot(1,2,2)
sns.countplot(data["Dependants"])
sns.barplot(x="churn",y="MonthlyCharges",data=data)
sns.heatmap(data.corr(),annot=True)
sns.pairplot(data=data,markers=["^","v"],palette="inferno")
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test_spilit(x_resample,y_resample,test_size=0.2,random_
state=0)
from sklearn.preprocessing import StandardScaler
sc=Standardscaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
x_train.shape
#importing ana 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("***Random Forest***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,ypred_rf))
print("Classification Report")
print(classfication_report(y_test,ypred_rf))
#printing the train accuracy and test accuracy respectively
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_train)
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)
#importing and building the random forest model
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))
print("***Support Vector Machine***")
print("Confusion_Matrix")
print(confusion_matrixity(y_test,ypred_svm))
print(classification_report(y_test,ypred_svm))
#printing the train accuracy and test accuracy respectively
svm(x_train,x_test,y_train,y_test)
#importing the Keras liabraries and packages
import Keras
from kears.models import Sequential
from Keras.layers import Dense
#Initialising the ANN
classfier=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',input-dim=40))
classifier .add(Dense(units=30,activation='relu'))
#adding the output layer
classfier.add(Dense(units=1,activation='sigmoid'))
#compling the ANN
classfier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accurac
y'])
#filting the ANN to the Training set
model_history=classifier.fit(x_train,y_train,batch_size=10,validation_spilit=0.33,e
pochs=200)
ann_pred=classfier.predict(x_test)
ann_pred=(ann_pred>0.5)
```

```
ann_pred
print(accuracy_score(ann_pred,y_test))
print("***ANN Model***")
print("Confusion_Matrix(t_test,ann_pred"))
print("Classification Report")
print(classification_repor(y_test,ann_pred))
#testing on random input values
lr=LogisticRegrassion(random_state=0)
lr.fit(x_train,y_train)
print("Prediciting on random 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("output is:",lr_pred-own)
dtc=DecisionTreeClassfier(criterion="entropy",random_state=0)
print("predicting on random input")
dtc_pred_own=dtc.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
,1,0,0,1,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567))print("output is:",dtc_pred_own))]]
rf =RandomForestclassicfier(criterion="entropy",n-estimators=10,random-
state=10)
rf.fit(x-train,y-train)
rf-pred-own
=rf.predict(sc.tranform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,o,o,1,o,o,1,o,o,1,o,1,o,o,1,
1,0,0,456,1,0,3245,4567]])
print("output is:",rf-pred-own)
svc=svc(kernal="liner")
print("output is:",rf-pred-own)
svc=svc(kernal="liner")
svc.fit(x-train,y-train)
print("output is:",svm-pred own)
knn=KNeighborsclassifier()
knn.fit(x-train,y-train)
print("predicting on random input")
knn-pred-
own="knn.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("output is:",knn_pred_own)
print("predicting on random input")
ann_pred_own=
classifier.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,
1,0,0,1,0,0,0,456,1,0,3245,4567]]))
print(ann_pred_own)
ann_pred_own=(ann_pred_own>0.5)
print("output is:",ann_pred_own)
def compareModel(x_train,x_test,y_train,y_test):
logreg(x_train,x_test,y_train,y_test)
print('_'*100)
```

```
decisionTree(x_train,x_test,y_train,y_test)
print('_'*100)
RandomForest(X_train,X_test,y_train,y_test)
print('-'*100)
svm(X_train,X_test,y_train,y_test)
print('-'100)
KNN(X_train,X_test,y_train,y_test)
print('-'100)
compareMOdel(X_train,X_test,y_train,y_test)
print(accuracy_score(ann_Pred,y_test))
print("***ANN Model***")
print("Confusion_Matrix")
Print(confusion_matrix(y_test,ann_pred))
print("classification Report")
print(classfication_report(y_test,ann_pred))
y_rf=model.predict(x_train)
print(accuracy_score(y_pred_rfcv,y_train))
ypred_rfcv=model.predict(x_test)
print(accuracy_score(ypred_rfcv,y_test))
print("***Random Forest after Hyperparameter tuning***")
print("confusion_Matrix")
print(confusion_matrixity_test,ypred_rfcv))
print("classication_report(y_test,ypred_rfcv"))
print("Classification Report")
print(classification_report(y_test,ypred_rfcv))
print("Predicating on random input")
rfcv_pred_own=model.predicat(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("output is:",rfcv_pred_own)
from  flask import Flask,render_template,request
import keras
from keras.models import load_model
app=Flask)_name_)
model=load_model("telecom_churn.h5)")
@app.route('/')
def helloworld():
return render_template("base.html")
@app.route('/assesment')
def predication():
return render_template("index.html")
@app.route('/predict',methods=['POST'])
def admin():
a=request.form["gender"]
if(a== 'f'):
a=0
if(a== 'm'):
```

```
a=1
b=request.form["srcitizen"]
if(b== 'n'):
b=0
if(b== 'y'):
b=1
c=request.form["partner"]
if(c== 'n')
c=0
if(c=='y'):
c=1
d=request.form["dependants"]
d=0
if(d== 'n'):
if(d== 'y'):
d=1
e=request.form["tenure"]
if (f== 'n'):
f=0
if(f== 'y'):
f=1
g=request.form["multi"]
if(g== 'n'):
g1,g2,g3=1,0,0
if(g== 'y'):
g1,g2,g3=1,0,0
if(g == 'nps'):
  g1,g2,g3=0,1,0
if(g== 'y'):
  g1,g2,g3=0,0,1
h= request.form["is"]
if(h== 'ds1'):
  h1,h2,h3=1,0,0
if(h== 'fo'):
  h1,h2,h3=0,0,1
i=request.form["os"]
if(i =='n'):
  i1,i2,i3=1,0,0
if(i =='nis'):
   i1,i2,i3=0,1,0
if(i =='y'):
  i1,i2,i3=0,0,1
j= request.form['ob']
if(j =='n'):
   j1,j2,j3=1,0,0
if(j== 'nis'):
```

```
  j1,j2,j3=0,1,0
if(j=='y'):
  j1,j2,j3=0,0,1
k=request.form['dp']
if(k== 'n'):
  k1,k2,k3=1,0,0
if(k=='nis'):
  k1,k2,k3=0,0,1
if(k== 'y'):
  k1,k2,k3=0,0,1
l=request.form["ts"]
if(l== 'n')
 l1,l2,l3=1,0,0
 if(l=='nis'):
   l1,l2,l3=0,1,0
if(l== 'y')
l1,l2,l3=0,0,1
m=request.form["stv"]
if(m== 'n'):
  m1,m2,m3=1,0,0
if(m== 'nis'):
  m1,m2,m3=0,1,0
if(m== 'y')
 m1,m2,m3=0,0,1
n=request.form["smv"]
if(n== 'n'):
 n1,n2,n3=1,0,0
if(n== 'nis'):
 n1,n2,n3=0,1,0
if(n== 'y'):
 n1,n2,n3=0,0,1
o=request.form["contract"]
if(o== 'mtm'):
 o1,o2,o3=1,0,0
if(o== 'oyr'):
 o1,o2,o3=0,1,0
if(o== 'tyrs'):
 o1,o2,o3=0,0,1
p=request.form["pmt"]
if(p== 'ec'):
 p1,p2,p3,p4=1,0,0,0
if(p== 'mail'):
 p1,p2,p3,p4=0,1,0,0
if(p== 'bt'):
  p1,p2,p3,p4=0,0,1,0
if(p== 'cc'):
```

```
  p1,p2,p3,p4=0,0,0,1
q=request.form["plb"]
if(q== 'n'):
  q=0
if(q== 'y'):
  q=1
r=request.form["mcharges"]
s=request.form["tcharges"]
t=[[int(g1),int(g2),int(g3),int(h1),int(h2),int(h3),int(i1),int(i2),int(i3),int(j1),int(j2),int(j3
)]]
print(t)
x=model.predict(t)
print(x[0])
if(x[[0]]<=0.5):
  y="No"
  return render_template("predno.html",z=y)
if(x[[0]]>=0.5):
y="Yes"
return render_template("predyes.html",z=y)
```