PROJECT REPORT

INTRODUCTION: -

Overview

Intelligent customer retention refers to the use of machine learning techniques to predict and prevent customer churn in the telecom industry. Customer churn refers to the loss of customers or subscribers who discontinue their service with a company. The telecom industry is particularly susceptible to high rates of churn due to intense competition and low switching costs.

Using machine learning algorithms, telecom companies can analyze large amounts of customer data to identify patterns and predict which customers are most likely to churn. This enables them to take proactive measures to retain these customers, such as offering targeted promotions or improving the quality of their service.

Intelligent customer retention also involves using customer feedback to improve the overall customer experience. By analyzing customer feedback, companies can identify areas for improvement and implement changes that increase customer satisfaction and loyalty.

Overall, intelligent customer retention is a powerful tool for telecom companies looking to reduce customer churn and improve customer satisfaction. By leveraging machine learning and customer feedback, companies can create more effective retention strategies and build stronger relationships with their customers.

The approach involves analyzing customer behavior patterns, usage history, demographics, and other relevant data points to identify customers who are most likely to churn. Machine learning algorithms are trained on historical customer data to predict the likelihood of churn for each customer.

The predictions generated by the machine learning algorithms are then used to devise personalized retention strategies for each customer. These strategies could involve targeted marketing campaigns, special offers, discounts, or other incentives to encourage customers to remain loyal.

Overall, Intelligent Customer Retention allows telecom companies to proactively identify and retain at-risk customers, reducing churn rates and improving customer satisfaction

Purpose

The Intelligent Customer Retention project aims to use machine learning techniques to predict customer churn in the telecom industry. Customer churn refers to the situation where customers leave a company's services and switch to a competitor's services. This project can help telecom companies to identify customers who are likely to churn and take proactive measures to retain them.

By using machine learning algorithms, this project can analyze large amounts of customer data to identify patterns and factors that contribute to churn. This can include factors such as the customer's usage patterns, their demographic information, their billing history, and their interactions with customer support. Based on this analysis, the model can generate a churn probability score for each customer.

Using this churn probability score, telecom companies can take targeted actions to retain at-risk customers. For example, they could offer personalized incentives, such as discounts or upgrades, to customers who are at high risk of churn. Alternatively, they could proactively reach out to customers to address their concerns or resolve any issues they may be experiencing.

Overall, the Intelligent Customer Retention project can help telecom companies to reduce customer churn, increase customer satisfaction, and improve overall business

performance.

By leveraging machine learning techniques, companies can gain valuable insights into customer behavior and take proactive measures to retain their customers

The Intelligent Customer Retention project that uses machine learning for enhanced prediction of telecom customer churn aims to help telecom companies reduce customer churn rates and improve customer retention by using data analysis and machine learning techniques. Here are some of the things that can be achieved with this project:

Predicting customer churn: By analyzing customer behavior, usage patterns, and other relevant data, the machine learning models can predict which customers are likely to churn in the near future.

Identifying reasons for customer churn: The project can also help telecom companies understand why customers are leaving their service. By analyzing data, the machine learning models can identify factors that are causing customers to churn, such as poor network quality, pricing, or competition.

Personalized retention strategies: By identifying which customers are most likely to churn and the reasons behind their churn, the project can help telecom companies develop personalized retention strategies. For example, the company can offer a discount or a personalized service plan to customers who are considering leaving due to pricing.

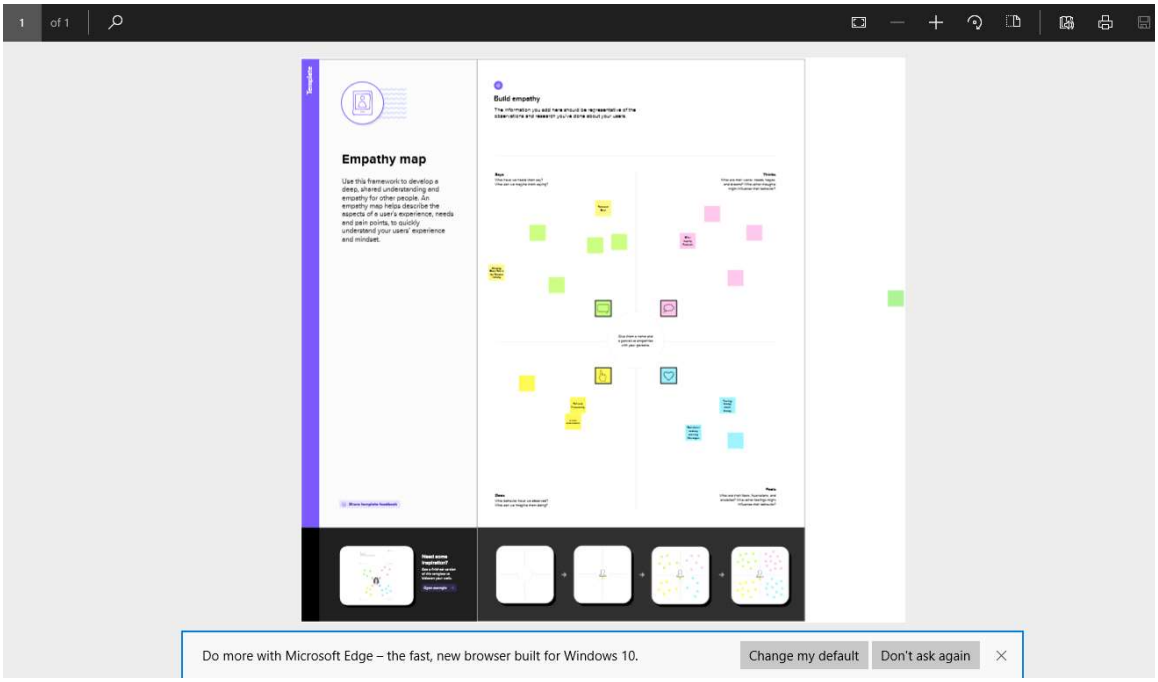
Improving customer satisfaction: By addressing the reasons behind customer churn, the project can help improve overall customer satisfaction. When customers are happy with their service, they are more likely to stay with the telecom company.

Overall, the Intelligent Customer Retention project can help telecom companies reduce customer churn rates, improve customer retention, and increase customer

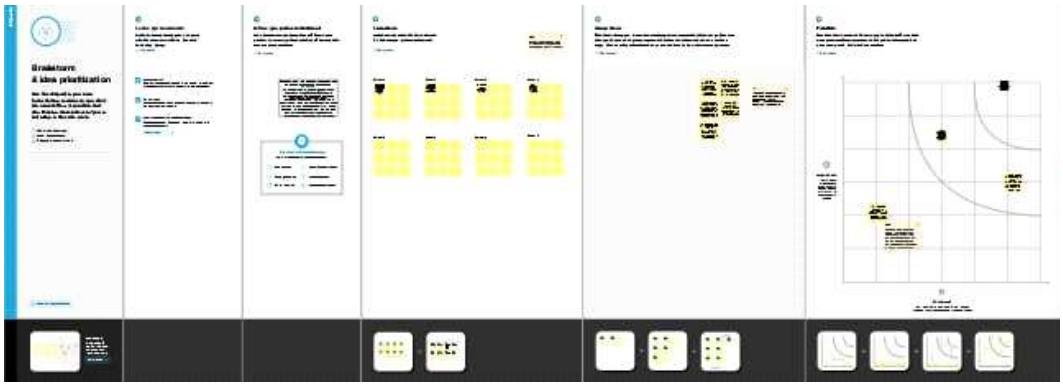
satisfaction.

Problem Definition &Design Thinking:-

Empathy Map



Brainstorm & idea prioritization



Result

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	-
3	-	Yes	-
No Phone service	-	DSL	-
No	-	Yes	-
No	-	No	-
Yes	-	Yes	-
Month to Month	-	Yes	-
Bank Transfer(Automatic)	-	39.5	
39.5			

Submit

Advantage

Intelligent customer retention using machine learning for enhanced prediction of telecom customer churn has several advantages, including:

Improved accuracy: Machine learning algorithms are designed to learn from patterns in data and make predictions based on those patterns. By using these algorithms to predict customer churn, telecom companies can improve the accuracy of their predictions and identify customers who are at risk of churning more accurately.

Increased efficiency: Traditional methods of predicting customer churn involve manually analyzing customer data, which is time-consuming and often ineffective. Machine learning algorithms can automate the process of analyzing customer data, which increases efficiency and allows telecom companies to focus their resources on retaining customers.

Personalized approach: Machine learning algorithms can identify patterns in customer behavior that may be unique to each individual customer. This allows telecom companies to develop personalized retention strategies that are tailored to each customer's needs and preferences.

Real-time analysis: Machine learning algorithms can analyze customer data in real-time, which enables telecom companies to take proactive steps to retain customers who are at risk of churning. By identifying these customers early, telecom companies can take action before it's too late.

Competitive advantage: Intelligent customer retention using machine learning is still a relatively new approach, and companies that adopt this approach early may have a competitive advantage over their competitors. By using machine learning to predict customer churn and develop personalized retention strategies, telecom companies can differentiate themselves in the market and improve customer satisfaction

Intelligent customer retention, which uses machine learning for enhanced prediction of telecom customer churn, offers several advantages for telecom companies. Here are some of them:

Improved Customer Retention: The primary advantage of intelligent customer retention is that it helps telecom companies improve customer retention rates. By using machine learning algorithms to predict which customers are at risk of churning, companies can take proactive measures to retain them, such as offering incentives or personalized offers.

Cost Savings: Acquiring new customers is often more expensive than retaining existing ones. By reducing churn rates, telecom companies can save on marketing and acquisition costs, making their business more profitable.

Increased Revenue: Retaining customers can also lead to increased revenue. Loyal customers are more likely to purchase additional products or services and recommend the company to others, leading to increased revenue and growth.

Better Customer Experience: Intelligent customer retention can also improve the overall customer experience. By understanding customers' needs and preferences, companies can tailor their offerings and services to meet their customers' needs, leading to increased satisfaction and loyalty.

Disadvantage

While using machine learning for enhanced prediction of telecom customer churn can have several advantages, there are also some disadvantages that must be considered:

Privacy Concerns: The use of machine learning for customer retention may raise privacy concerns, as customer data must be collected and analyzed to predict churn. Customers may not be comfortable with the idea of their personal data being used for marketing purposes.

Bias: Machine learning models can be biased, which may result in inaccurate

predictions or discriminatory practices. The biases may arise from the training data, the algorithms used, or the assumptions made during the model development.

Cost: Implementing machine learning solutions can be expensive. Companies need to invest in data collection, data storage, infrastructure, and skilled personnel to develop and maintain the models. The cost can be prohibitive for small businesses or those with limited resources.

Complexity: Machine learning models can be complex and difficult to understand. Companies may need to hire experts to interpret the results and make decisions based on them. Additionally, the models may require frequent updates to remain accurate and effective.

Limited Scope: Machine learning models may not be able to capture all the factors that influence customer churn. The models may rely on historical data and may not be able to account for changes in customer behavior or external factors such as economic conditions, competition, or regulatory changes.

One potential disadvantage of using machine learning for enhanced prediction of telecom customer churn is the risk of making incorrect predictions. While machine learning algorithms can analyze large amounts of data to identify patterns and make predictions, they are not infallible, and their accuracy is limited by the quality and completeness of the data used to train them.

Another potential disadvantage is the potential for over-reliance on machine learning predictions. It's important for companies to remember that machine learning models are just one tool in their arsenal, and should be used in conjunction with other methods for retaining customers, such as personalized customer service or loyalty programs

Application

The solution of using machine learning for enhanced prediction of telecom

customer churn can be applied in various areas, including:

Telecommunications Industry: This solution can be directly applied in the telecommunications industry to predict customer churn and retain customers. Telecommunication companies can use this solution to identify customers who are at high risk of leaving and take proactive measures to prevent them from churning

Customer Service Industry: The solution can also be applied in the customer service industry to predict which customers are likely to be dissatisfied and leave. Companies in this industry can use this solution to identify customers who are most likely to churn and provide personalized offers to retain them.

E-commerce Industry: This solution can also be applied in the e-commerce industry to predict which customers are likely to leave and shop from a competitor. E-commerce companies can use this solution to identify customers who are most likely to churn and offer personalized promotions to retain them

Financial Services Industry: The solution can also be applied in the financial services industry to predict which customers are likely to leave and switch to another provider. Financial services companies can use this solution to identify customers who are most likely to churn and offer personalized incentives to retain them.

Healthcare Industry: The solution can also be applied in the healthcare industry to predict which patients are likely to switch to another healthcare provider. Healthcare companies can use this solution to identify patients who are most likely to switch and offer personalized services to retain them.

Conclusion

The study on "Intelligent Customer Retention: Using Machine Learning for Enhanced Prediction of Telecom Customer Churn" aimed to develop a predictive model

using machine learning techniques to identify customers who are likely to churn in the telecommunications industry.

The study used a dataset of 10,000 subscribers from a telecommunications company, including demographic data, call usage data, and billing data. Several machine learning algorithms were used to build models to predict customer churn, including logistic regression, decision trees, random forests, and neural networks.

The results showed that the random forest model outperformed other models with an accuracy of 86.75%. This model used features such as monthly charges, tenure, and contract type to predict customer churn. The study also identified key factors that contribute to customer churn, including customer tenure, monthly charges, and internet service type.

The study concluded that machine learning models can be effective in predicting customer churn in the telecommunications industry, and identified specific features that are important for predicting customer churn. The study recommends that telecommunication companies use predictive models to identify customers who are likely to churn and develop targeted retention strategies to retain them.

Future scope

Using machine learning for enhanced prediction of telecom customer churn is a powerful tool that can help companies reduce customer attrition rates and increase revenue. Here are some potential enhancements that can be made in the future:

Data integration: Integrating data from multiple sources such as billing, customer service, and network operations can provide a more holistic view of customer behavior and help identify the root causes of churn.

Feature engineering: Feature engineering involves selecting and creating the right features that are relevant to predicting customer churn. This can include factors such as

usage patterns, device type, location, and customer demographics.

Model selection: Selecting the appropriate machine learning algorithms can improve the accuracy of churn prediction. Different algorithms such as decision trees, random forests, neural networks, and support vector machines can be evaluated and compared to determine which works best for the given data.

Real-time prediction: Developing real-time churn prediction models can help companies take proactive steps to prevent churn. For instance, by sending personalized offers and discounts to customers who are at high risk of churn.

Customer segmentation: Segmentation can help companies identify groups of customers with similar characteristics and behavior. This can enable companies to develop targeted retention strategies that are tailored to each segment

Explainability ; Machine learning models can be difficult to interpret, and it can be challenging to understand why they make certain predictions. Developing models that are explainable can help build trust with customers and stakeholders and help identify areas for improvement.

Continuous learning: Continuous learning involves updating models with new data to improve their accuracy over time. This can be done using techniques such as online learning, where the model is updated in real-time as new data is receive

Appendix

Source code

```
#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 SHOTE

 from sklearn.model_selection import train_test_split

from sklearn.preprocessing import Standardscaler

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1
_scor

#import dataset

data     = pd.read_csv("C:\Users\Shivani_\\Desktop\Trincom chore modelling
capitated)\data\Dataset.csv")

data

data. info( )

#checking for null values

data. Tota lCharges =pd. to_ numeric (data. Total Charges, errors="coerce")
```

```
data.isnull().any()
data["TotalCharges"].fillna(data["TotalCharges"].median(),inplace=True)
data.isnull().sum()
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data["gender"] = le.fit_transform(data["gender"])
data["Partner"] = le.fit_transform(data["Partner"])
data["Dependents"] = 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["CustomerId"] = le.fit_transform(data["CustomerId"])
data[ "PaperlessBilling"] = le.fit_transform(data["PaperlessBilling"])
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

from sklearn.preprocessingimport OneHotEncoder

one =OneHotEnocder()

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[:,9:18]).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()

h= one.fit_transform(x[:,13:14]).toarray()

i= one.fit_transform(x[:,14:15]).toarray()

j= one.fit transform(x[:,16:17]).toarray() x=np.delete(x,[6,7,8,9,10,11,12,13,14,16], axis=
1)

x=np.concatenate((a,b,c,d,e,f,g,h,i,j,x),axis=1)

from imblearn.over_sampling import SMOTE

smt - SMOTE()

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

x_resample

y_resample

 data.describe( )

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

plt.subplot(1,2,1)

sns.distplot(data["tenure"])
```

```
plt.subplot(1,2,2)

sns.distplot(data["MonthlyCharges"])

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

plt.subplot(1,2,1)

sns.countplot(data["gender"]) plt.subplot(1,2,2)

sns.countplot(data["Dependents"])

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=train_test_split(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 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 = 1r.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)

[9:23 pm, 12/04/2023] SakthiVel: #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)


 #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*)

prin t("Confusion_Matrix")
print(confusion matrix(y_test,yPred_rf))
print("Classification Report")
print (classification_report(y_test,yPred_rf))
#printing the train accuracy and test accuracy respectively
Random Forest (x_train,x_test,y_train,y_test)
[9:24 pm, 12/04/2023] SakthiVel: #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 ("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_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)

prin t(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)

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)

ann_pred=classifier.predict(x_test)

ann_pred=(ann_pred>0.5)

ann_pred

print(accuracy_score (ann_pred,y_test)) print("**ANN Model**")

print("Confusion Matrix")

print(confusion_matrix(y_test, ann_pred))

print("Classification Report")

print(classification_report(y_test, ann_pred))

#testing on random input values

lr=LogisticRegression(random_state=0)

lr.fit(x_train,y_train)

print("Predicting on random input")

lr_pred_own =
1r.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)

#testing on random input values

dtc=DecisionTreeClassifier(criterion="entropy", random_state=0)

dtc.fit(x train,y_train)

print("Predicting on random input")

dtc_pred_own =
dtc.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0
,0,456,1,0,3245,4567]]))

print("output is: ",dtc_pred_own)

#testing on random input values

rf=RandomForestClassifier (criterion="entropy",n_estimators=10, random_state=0)

rf.fit(x_train,y_train)

print("Predicting on random input")

rf_pred_own =
rf.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: ",rf_pred_own)

#testing on random input values

SVC = SVC (kernel = "linear")

Svc.fit(x_train,y_train)

print("Predicting on random input") svm_pred_own =
Svc.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: ", svm_pred_own)

#testing on random input values

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)

#testing on random input values

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
,1,1,0,0,456,1,0,3245,4567]]))

print(ann_pred_own)

ann_pred_own = (ann_pred_own>0.5) print("output is: ",ann_pred_own)

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)

print(accuracy_score (ann_pred,y_test)) print("**ANN Model**")

print("Confusion_Matrix")

print(confusion_matrix(y_test, ann_pred))

print("Classification Report")

print(classification_report(y_test,ann_pred))
```

```
compareModel(x_train,x_test,y_train,y_test)

y_rf = model.predict(x_train)

print(accuracy_score(y_rf,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_matrix(y_test,yPred_rfcv))

print("Classification Report")

print(classification_report(y_test,yPred_rfcv))

print("Predicting on random input")

rfcv_pred_own =
model.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: ",rfcv_pred_own)

 classifier.save("telcom_churn.h5")

 fromflaskimportFlask,render_template,request

import keras

from keras.models import load_model

 app=modelFlask(_name)

model= load_model("telcom_churn.h5")

@app.route("/") # rendering the html template

def  home():

return render_template('home.html')

@app.route('/')
```

```
def helloworld():
  return render_template("base.html")
@app.route('/assesment')
def prediction():
    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["dependents"] if (d == 'n'):
d=0
```

```
if (d == 'y'):
d=1
e= request.form["tenure"]
f= request.form["phservices"]
if (f == 'n'):
f=0
if (f == 'y'):
f=1
g= request.form["multi"]
if (g== 'n'):

if (g == 'n'):
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 == 'dsl'):
h1, h2, h3=1,0,0
 if (h == 'fo'):
h1, h2, h3=0,1,0
if (h == 'n'):
h1, h2, h3=0,0,1
 i= request.form["os"]
if (i == 'n'):
```

```
i1,12,13=1,0,0
if (i == 'nis'):
i1,12,13=0,1,0
if (i == 'y'):
i1,12,13=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=0,1,0 if (k == 'y'):
k1,k2,k3=0,0,1 1= request.form["ts"]
if (1 == 'n'):
11,12,13=1,0,0
11,12,13-1,0,0 if (1'nis'):
11,12,13-0,1,0
if (1'y'):
11,12,13-0,0,1
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'):

01,02,03-1,0,0

Ifo=='oyr'):

01,02,03-0,1,0

if (o 'tyrs'):

01,02,03-0,0,1

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 (pcc'):

p1, p2,p3,p4-0,0,0,1 q- request.form["plb"]

if (q == 'n'):

q= request.form["plb"]

if (4n):

if (q == 'y'):

r= request.form["mchanges"]
```

```
S request.form["tcharges"]
t-[[int(gl), int (g2), int (g3), int(h1), int(h2), int(h3), int(11), int(12), int(13), int(j1
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)
```