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

1.INTRODUCTION:

1.1. OVERVIEW

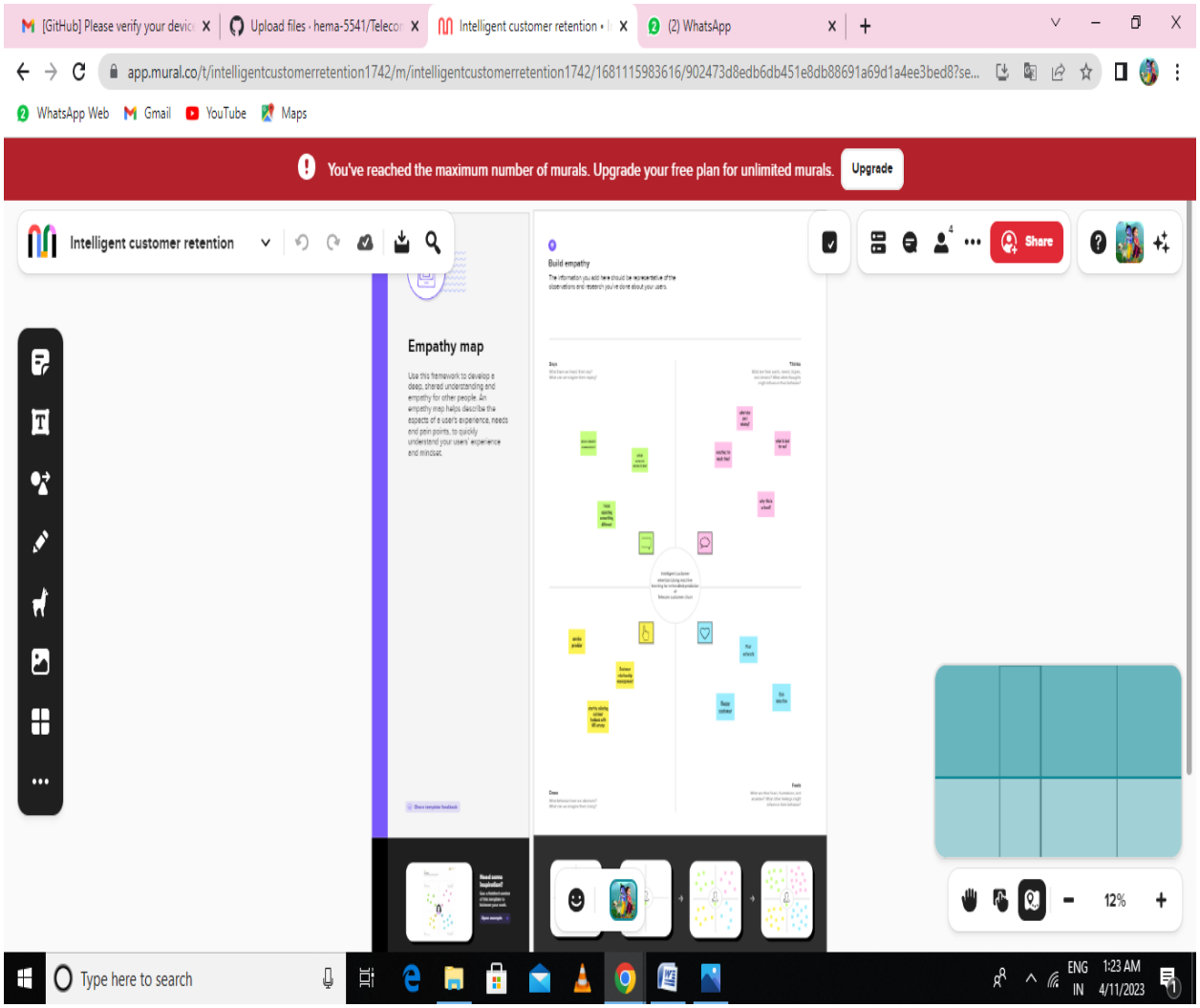
Customer churn is often referred to as customer attrition, or customer defection which is the rate at which the customers are lost. Customer churn is a major problem and one of the most important concerns for large companies. Due to the direct effect on the revenues of the companies, especially in the 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

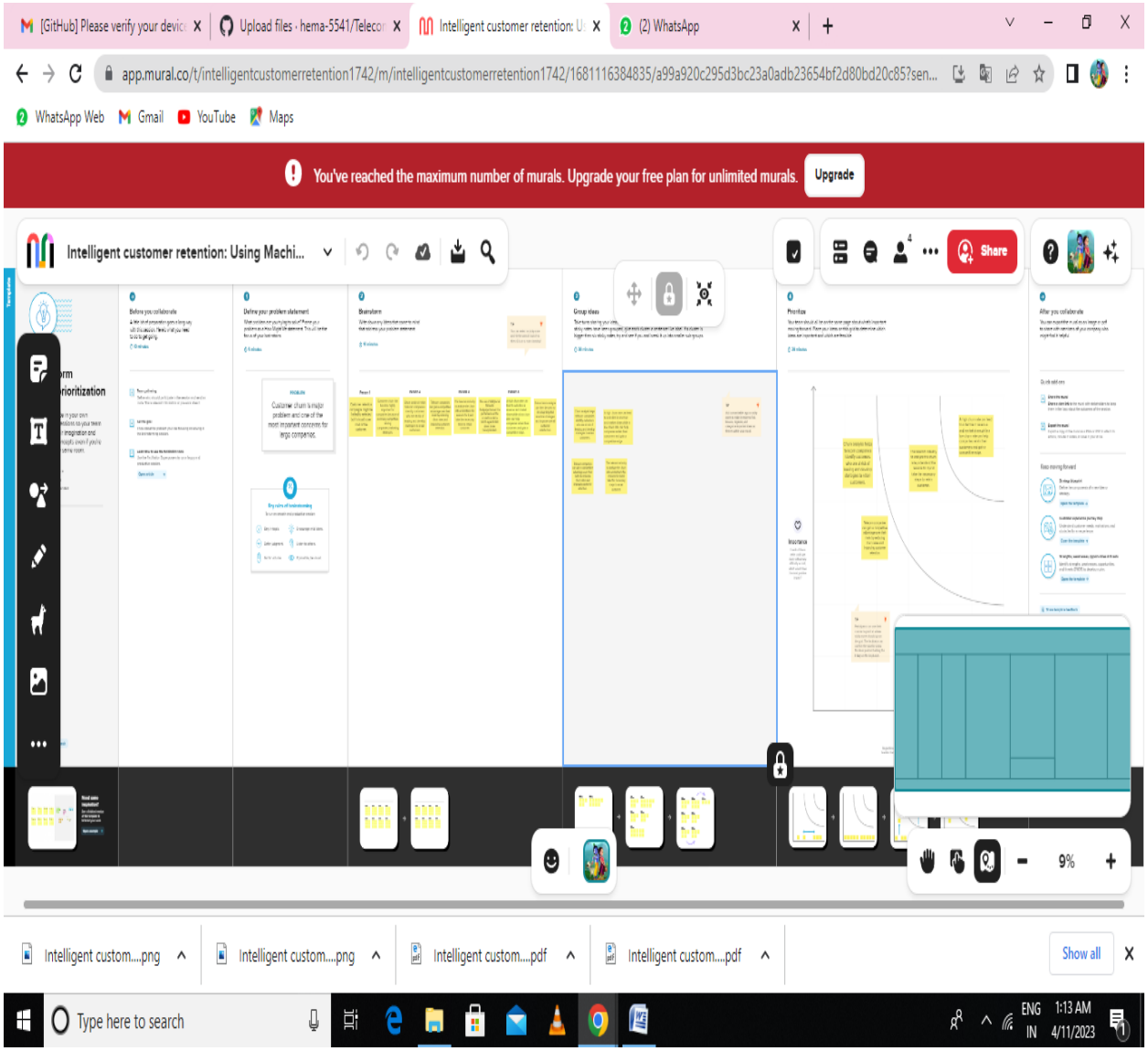
The purpose of this proposed work is to apply a novel retention technique called the targeted proactive and stop many customers from Leaving and help to not cancel our subscription to our service.

2.PROBLEM DEFINITION & DESIGN THINKING:

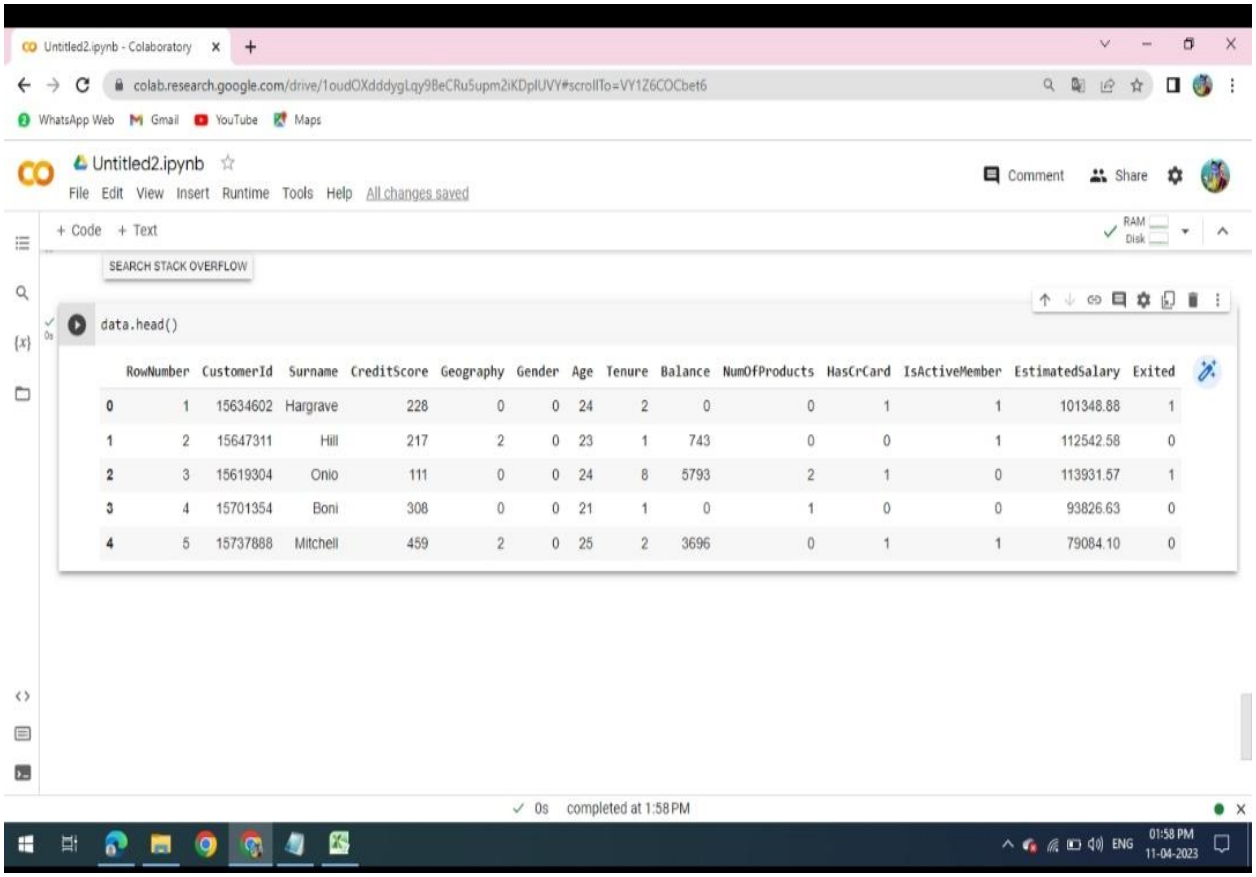
2.1.EMPATHY MAP:



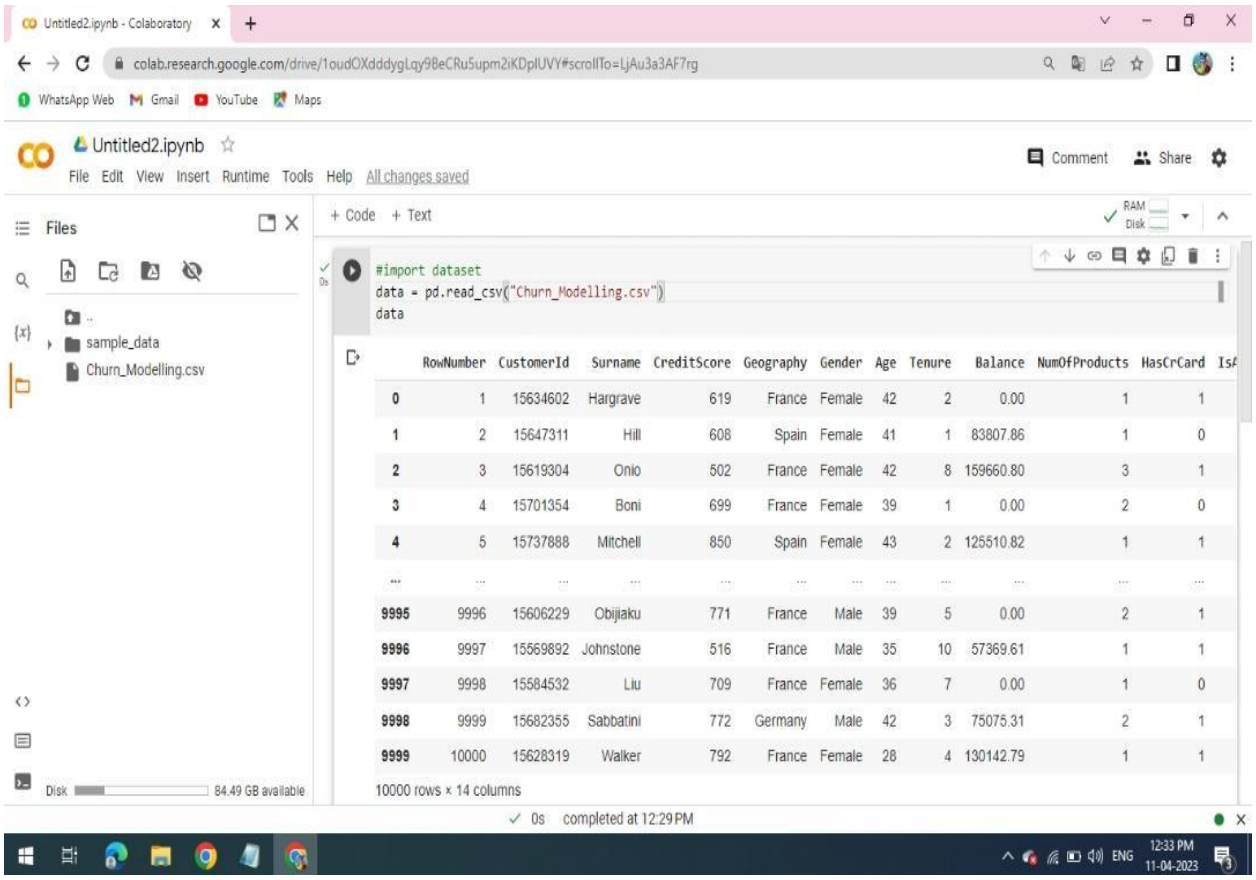
2.2.IDEATION & BRAINSTORMING MAP :



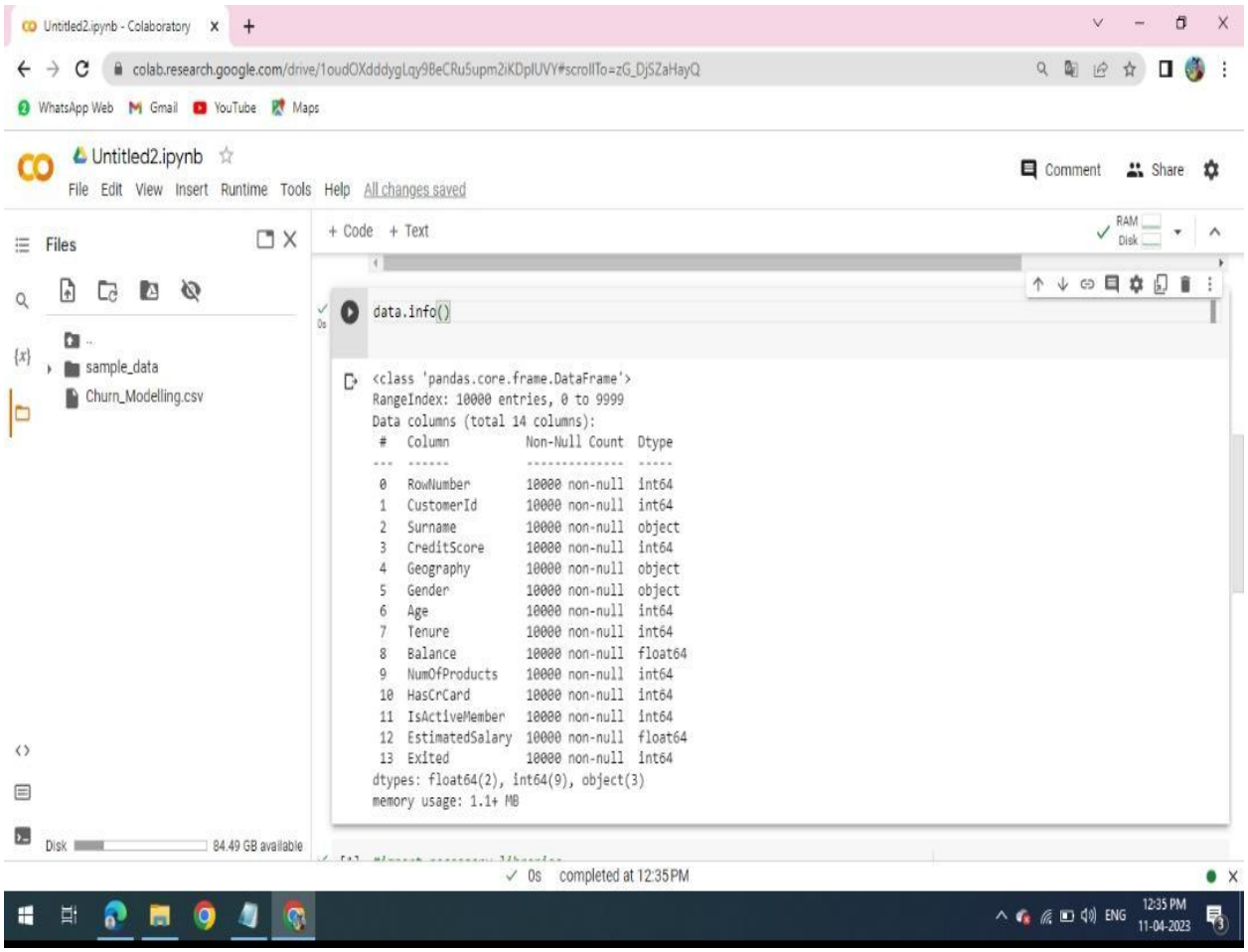
3.RESULT:

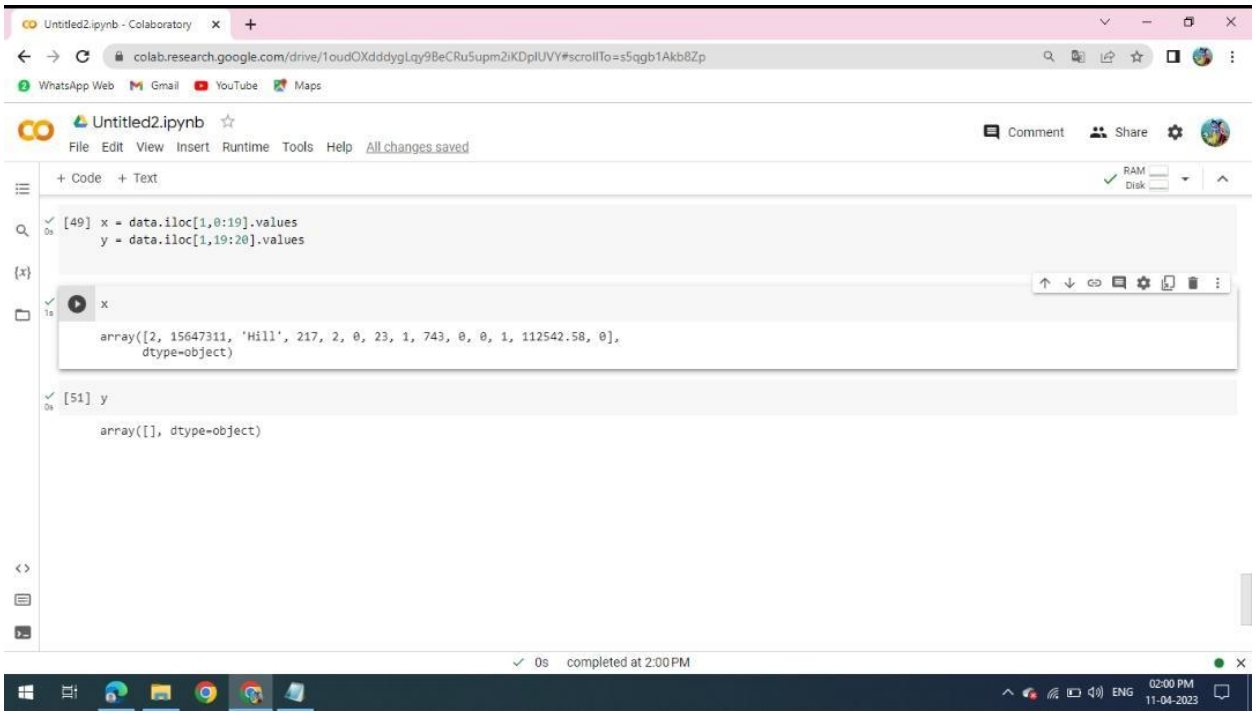


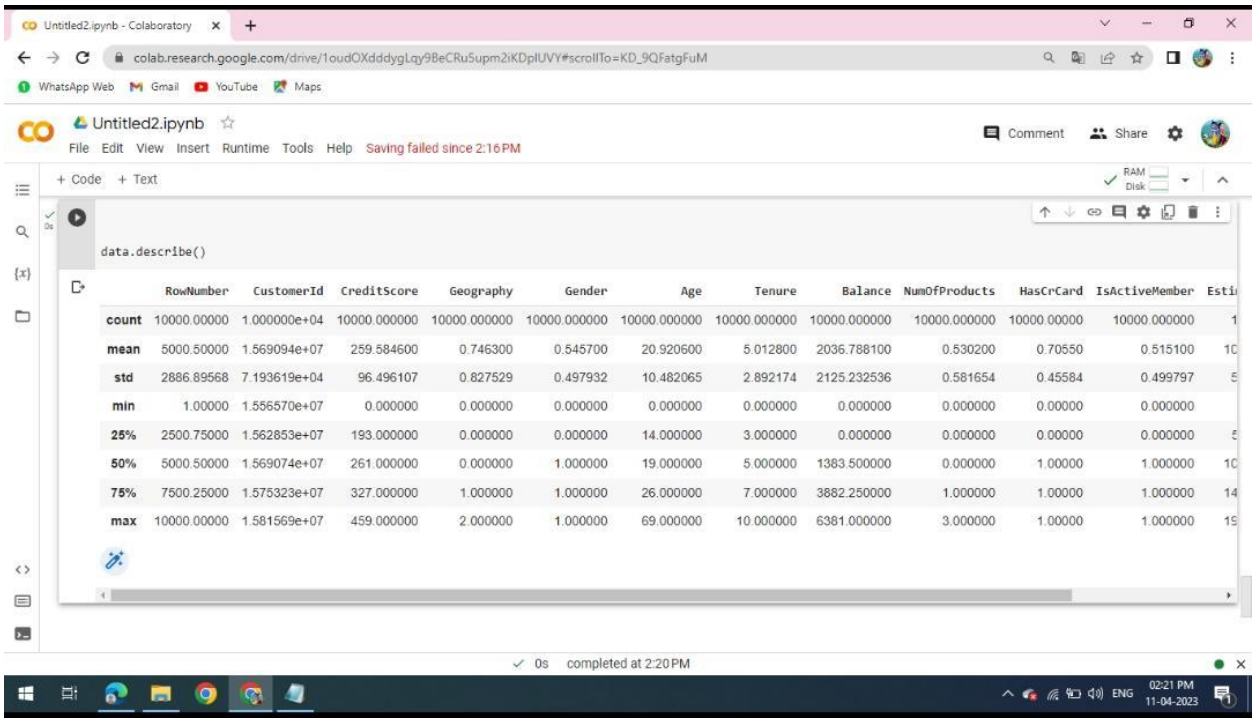
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	228	0	0	24	2	0	0	1	1	101348.88	1
1	2	15647311	Hill	217	2	0	23	1	743	0	0	1	112542.58	0
2	3	15619304	Onio	111	0	0	24	8	5793	2	1	0	113931.57	1
3	4	15701354	Boni	308	0	0	21	1	0	1	0	0	93826.63	0
4	5	15737888	Mitchell	459	2	0	25	2	3696	0	1	1	79084.10	0



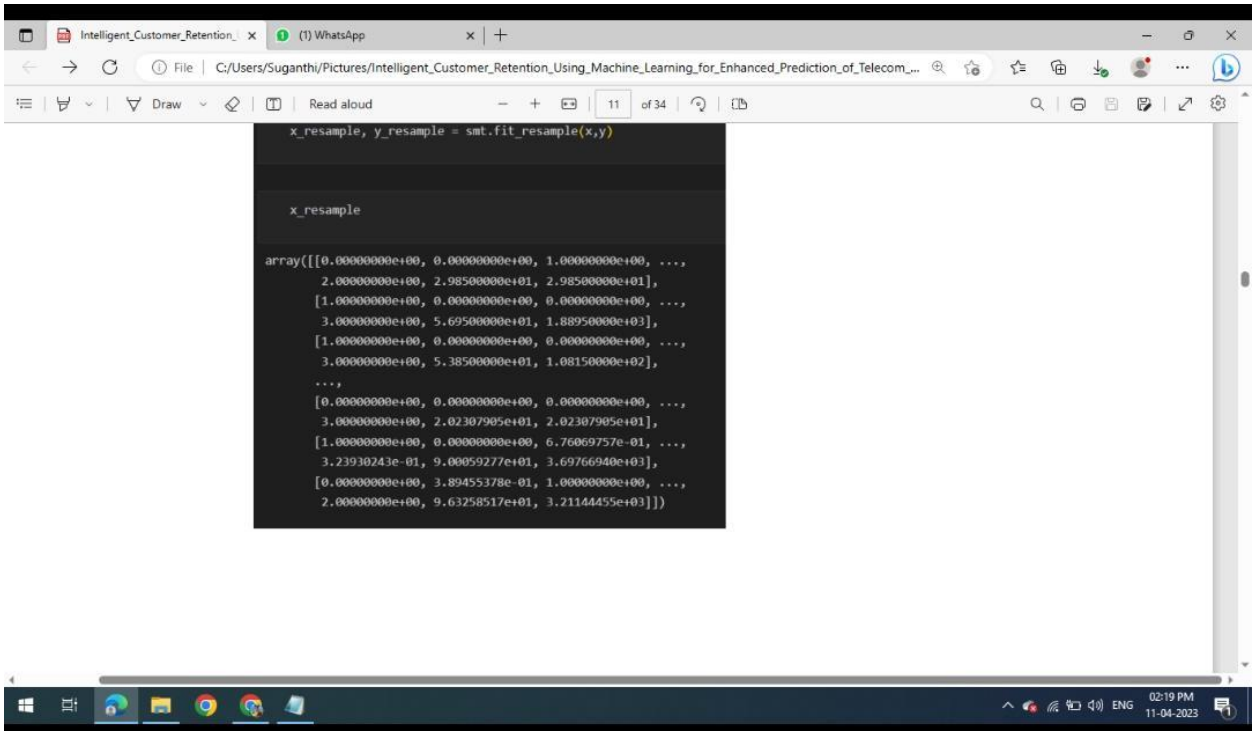
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	

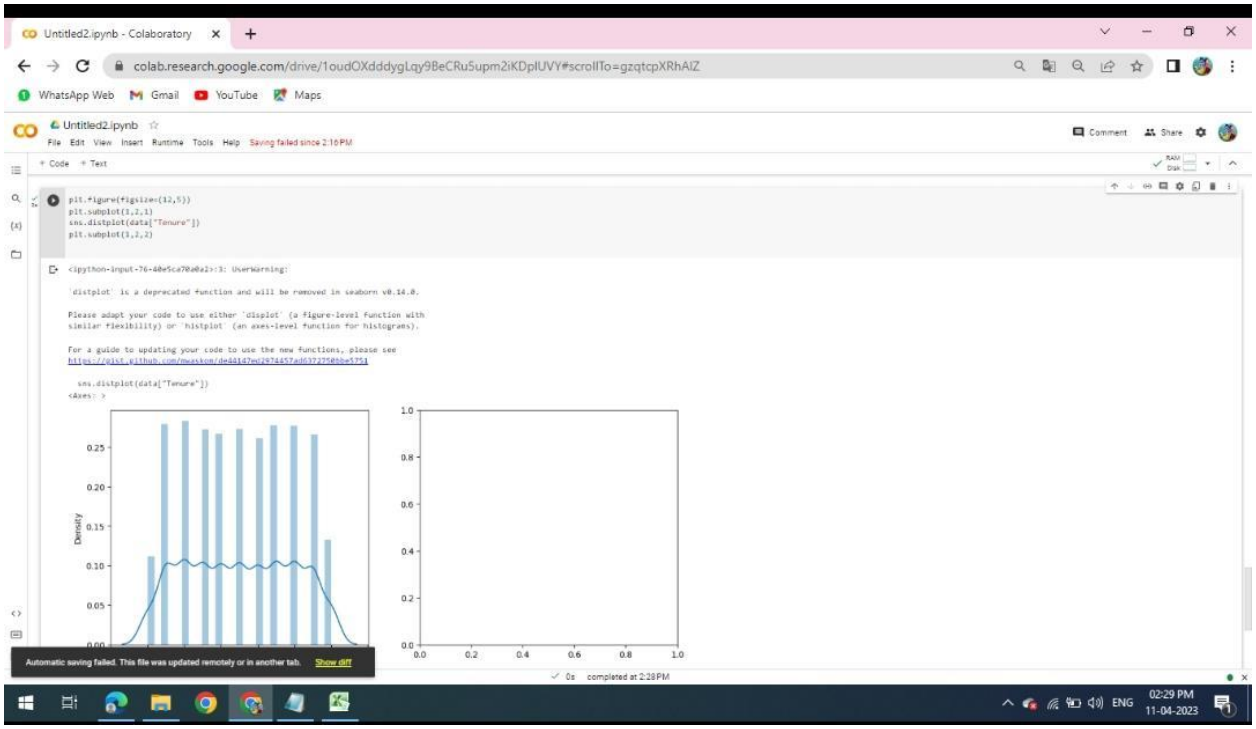

Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1oudOXdddygLqy9BeCRu5upm2iKDpIUVY#scrollTo=zG_DjSZaHayQ
WhatsApp Web
Gmail
YouTube
Maps
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
sample_data
Churn_Modelling.csv
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
Column Non-Null Count Dtype
0 RowNumber 10000 non-null int64
1 CustomerId 10000 non-null int64
2 Surname 10000 non-null object
3 CreditScore 10000 non-null int64
4 Geography 10000 non-null object
5 Gender 10000 non-null object
6 Age 10000 non-null int64
7 Tenure 10000 non-null int64
8 Balance 10000 non-null float64
9 NumOfProducts 10000 non-null int64
10 HasCrCard 10000 non-null int64
11 IsActiveMember 10000 non-null int64
12 EstimatedSalary 10000 non-null float64
13 Exited 10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
Disk
84.49 GB available
0s completed at 12:35PM
ENG
12:35 PM
11-04-2023

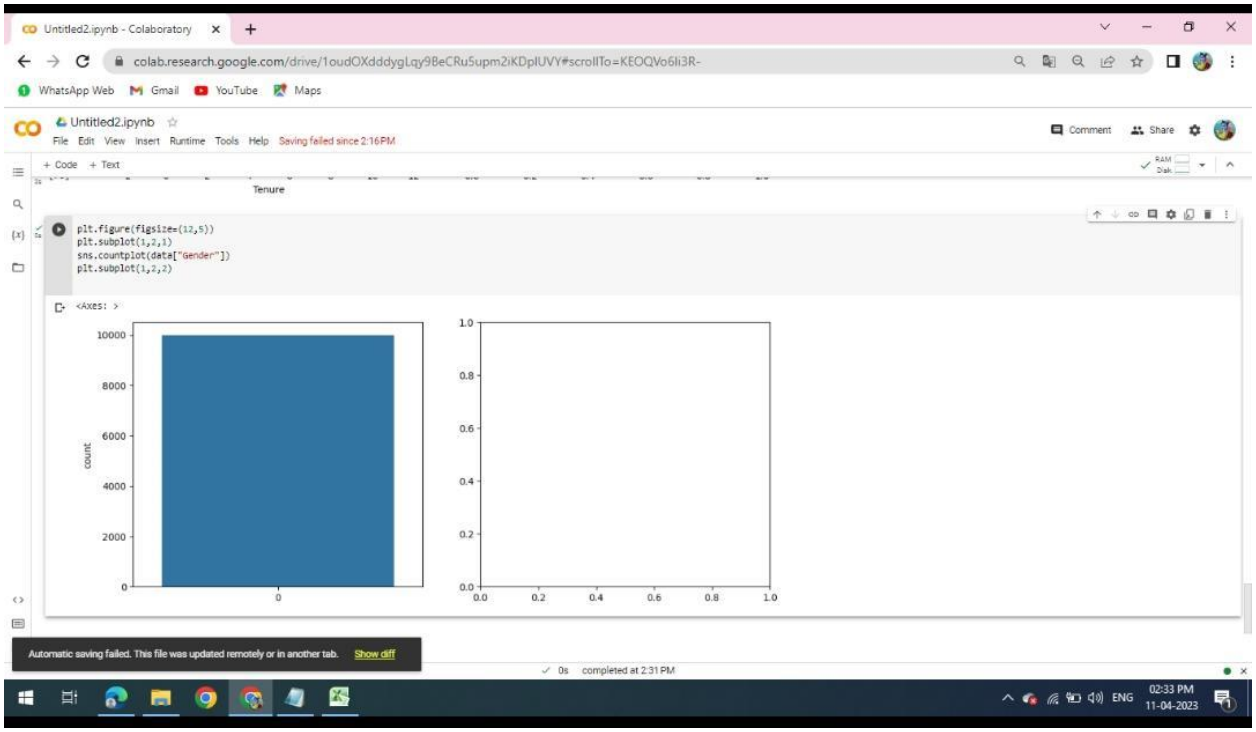

Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1oudOXdddygLqy9BeCRu5upm2iKDpIUVY#scrollTo=s5qgb1Akb8Zp
WhatsApp Web
Gmail
YouTube
Maps
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[49] x = data.iloc[1,0:19].values
y = data.iloc[1,19:20].values
x
array([2, 15647311, 'Hill', 217, 2, 0, 23, 1, 743, 0, 0, 1, 112542.58, 0],
dtype=object)
[51] y
array([], dtype=object)
0s completed at 2:00 PM
ENG
02:00 PM
11-04-2023

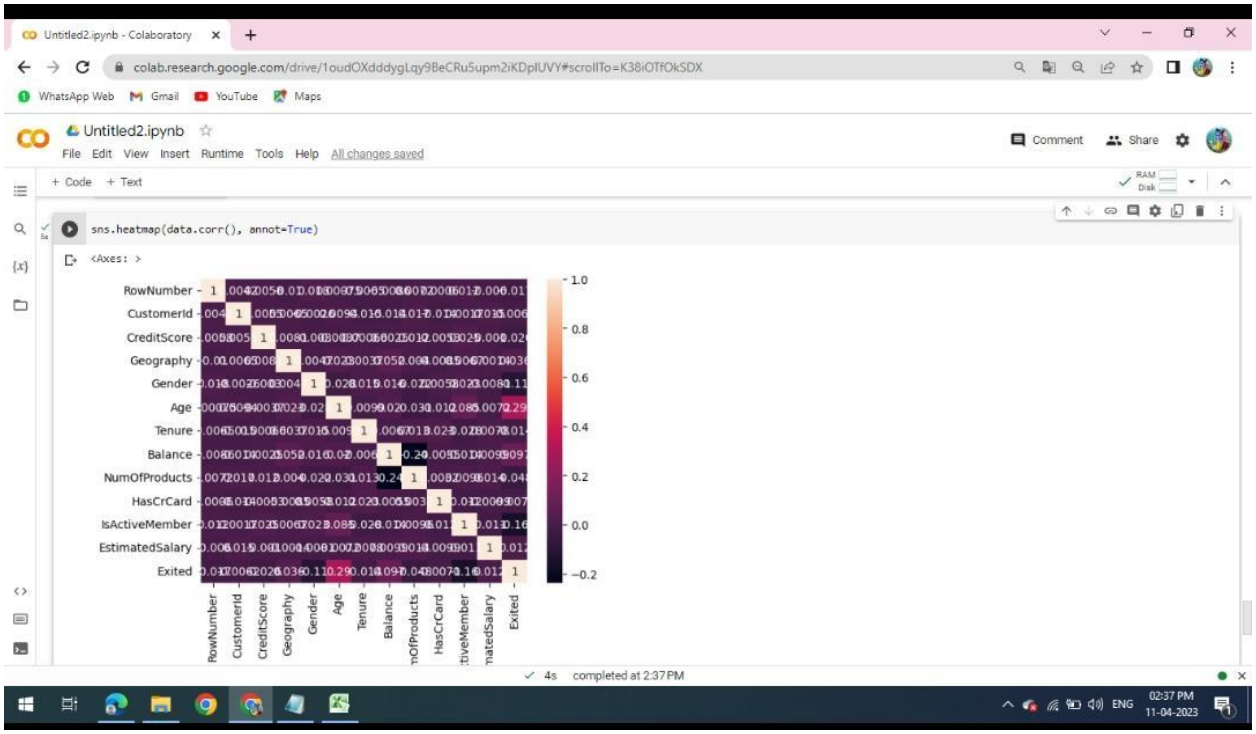


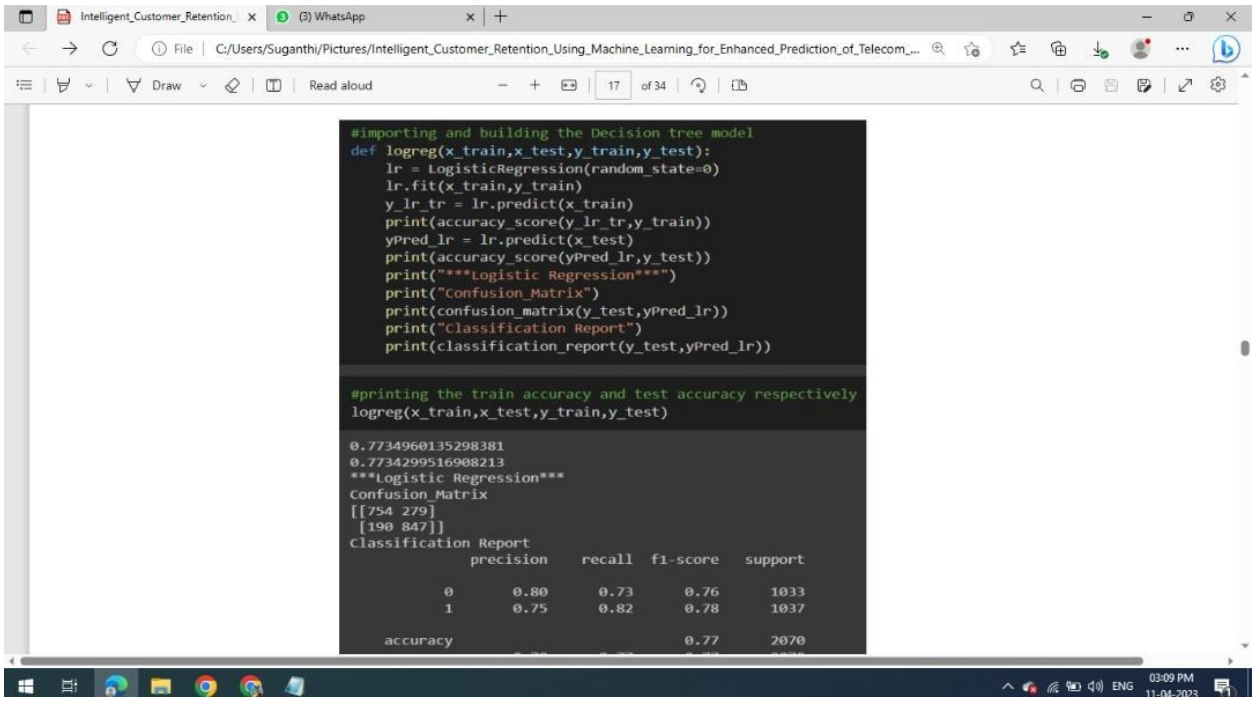
	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Esti
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	1
mean	5000.50000	1.569094e+07	259.584600	0.746300	0.545700	20.920600	5.012800	2036.788100	0.530200	0.70550	0.515100	10
std	2886.89568	7.193619e+04	96.496107	0.827529	0.497932	10.482065	2.892174	2125.232536	0.581654	0.45584	0.499797	5
min	1.00000	1.556570e+07	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	
25%	2500.75000	1.562853e+07	193.000000	0.000000	0.000000	14.000000	3.000000	0.000000	0.000000	0.00000	0.000000	5
50%	5000.50000	1.569074e+07	261.000000	0.000000	1.000000	19.000000	5.000000	1383.500000	0.000000	1.00000	1.000000	10
75%	7500.25000	1.575323e+07	327.000000	1.000000	1.000000	26.000000	7.000000	3882.250000	1.000000	1.00000	1.000000	14
max	10000.00000	1.581569e+07	459.000000	2.000000	1.000000	69.000000	10.000000	6381.000000	3.000000	1.00000	1.000000	19


Intelligent_Customer_Retention_
(1) WhatsApp
File | C:/Users/Suganthi/Pictures/Intelligent_Customer_Retention_Using_Machine_Learning_for_Enhanced_Prediction_of_Telecom_...
Draw
Read aloud
11
of 34
x_resample, y_resample = smt.fit_resample(x,y)
x_resample
array([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, ...,
2.00000000e+00, 2.98500000e+01, 2.98500000e+01],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
3.00000000e+00, 5.69500000e+01, 1.88950000e+03],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
3.00000000e+00, 5.38500000e+01, 1.08150000e+02],
...,
[0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
3.00000000e+00, 2.02307905e+01, 2.02307905e+01],
[1.00000000e+00, 0.00000000e+00, 6.76069757e-01, ...,
3.23930243e-01, 9.00059277e+01, 3.69766940e+03],
[0.00000000e+00, 3.89455378e-01, 1.00000000e+00, ...,
2.00000000e+00, 9.63258517e+01, 3.21144455e+03]])
ENG
02:19 PM
11-04-2023


Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1oudOXdddygLqy9BeCRu5upm2iKDpIUVY#scrollTo=gzqtcpXRhAIZ
WhatsApp Web
Gmail
YouTube
Maps
Untitled2.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 2:16 PM
Comment
Share
+ Code + Text
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.distplot(data["Tenure"])
plt.subplot(1,2,2)
<ipython-input-76-40e5ca70e0a2>:3: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
sns.distplot(data["Tenure"])
<Axes: >
Density
0.25
0.20
0.15
0.10
0.05
0.00
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Automatic saving failed. This file was updated remotely or in another tab. Show diff
0s completed at 2:28 PM
ENG
02:29 PM
11-04-2023


Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1oudOXdddygLqy9BeCRu5upm2iKDplUVY#scrollTo=KEOQVo6li3R-
WhatsApp Web
Gmail
YouTube
Maps
Untitled2.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 2:16PM
Comment
Share
+ Code + Text
Tenure
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.countplot(data["Gender"])
plt.subplot(1,2,2)
<Axes: >
count
Automatic saving failed. This file was updated remotely or in another tab.
Show diff
0s completed at 2:31 PM
ENG
02:33 PM
11-04-2023


Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1oudOXdddygLqy9BeCRu5upm2iKDplUVY#scrollTo=K38iOTfOkSDX
WhatsApp Web
Gmail
YouTube
Maps
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
sns.heatmap(data.corr(), annot=True)
<Axes: >
RowNumber
CustomerId
CreditScore
Geography
Gender
Age
Tenure
Balance
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
Exited
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
4s completed at 2:37 PM
ENG
02:37 PM
11-04-2023


Intelligent_Customer_Retention_
(3) WhatsApp
File | C:/Users/Suganthi/Pictures/Intelligent_Customer_Retention_Using_Machine_Learning_for_Enhanced_Prediction_of_Telecom_...
Draw
Read aloud
17
of 34
#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
ENG
03:09 PM
11-04-2023

4.ADVANTAGES & DISADVANTAGE:

ADVANTAGE:

- Cheaper than acquisition
- Loyal customers yield higher profits
- More word of mouth referrals
- Easy up-selling and cross-selling
- Loyal customers are more forgiving
- Better communication with customers

DISADVANTAGES:

- Large investment in terms of price and time.
- Require concerted commitment and business culture.
- There are a multitude of issues that can lead customers to leave a business, but there are a few that are considered to be the leading causes of customer.
- The first is poor customer service.

5.APPLICATION:

Machine learning algorithms assist telecom network engineers with detecting instances of illegal access, fake caller profiles and cloning. To achieve this, the algorithms perform behavioral monitoring of the global telecom networks of CSPs. Accordingly , the network traffic along such networks is closely monitored.

6.CONCLUSION:

Implementing customer churn models in your business, you will stop many consumers from leaving, and by that – make more money. If you have any questions or need an explanation about customer churn prediction using machine learning, ping us a message.

7.FUTURE SCOPE:

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:

A. 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 SMOTE

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

#from sklearn.metrics import accuracy_score,classification_report,confustion_matrix,fi_score


#import dataset
```

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

data
```

```
data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex:7043 entries,0 to 7042

Data columns (total 20 columns):
```

```
#checking for null values

data.TotalCharges=pd.to_numeric(data.totalCharges,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_transfrm(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["StreamimgTV"] = le.fit_transform(data["StreamingTV"])
data["StreamingMovies"] = le.fit_transform(data["StreamingMovies"])
data["Contract"] = le.fit_transform(data["Contract"])
data["PaperlessBilling"] = le.fit_transform(data["PaperlessBiing"])
data["PaymentMethod"] = le.fit_transform(data["Paymentmethod"])
data["Churn"] = le.fit_transform(data["Churn"])

data.head()

x = data.iloc[1,0:19].values
y = data.iloc[1,19:20].values

from sklearn.preprocessing import OnellotEncoder
one = OnellotEncoder()
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()

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),axis=1)
```

```
from imblearn.over_sampling import SMOT

snt = SMOT()

x_resample,y_resample = snt.fit_resample(x,y)
```

```
data.describe()
```

```
plt.figure(figsize-(12,5))
plt.subplot(1,2,1)
sns.distplot(data["Encure"])
plt.subplot(1,2,2)
sns.distplot(dats["MonthlyCharges"])
```

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.countplot(data["gender"])
plt.sublot(1,2,2)
sns.countplot(data["Dependents"])
```

```
sns.barplot(x-"Churn", y-"MonthlyCharges",data-data)
```

```
<AxesSubplot:Xlabel='Churn', ylabel='MonthlyCharges'>
```

```
sns.heatnap(data.corr(), annot=True)
```

```
sns.pairplot(data=data,markers=["x","y"],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,text_size=0.2,random_sta
te=0)


from sklearn.preprocessing import StandardScaler

sc - StandardScaler(0

x_train - sc.fit_transfrom(x_train)

x_test - sc.fit_transform(x_test_)


x_train.shape



#importing and buliding 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(accurac_score(y_lr_tr,y_train))

yPred_lr = lr.predict(x_test)

print(accurary_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)



#importing and building the Decision tree model
det 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_reort(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 RandomFrest(x_train,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("Classification Reprt")
    print(classification_report(y_test,yPred_rf))

#printing the train accuracy and test accurac 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_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)



#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,y_test))
```

```
    print("***Support Vector Machine***")
    print("Confusion_Matrix")
    print(confusion_matrix(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 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', metrix=['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)


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 =
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)
```

```
#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_estimation=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 = "inear")

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 = KNeighborsCassifier()

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)




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(classification_report(y_test,ann_pred))



y_rf = model.predict(x_train)
print(accuracy_score(y_rf,y_train))
yPred_rfcv = model.predict(x_test)
print(accurcy_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([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,
456,1,0,3245,4567]]))

print("output is: ",rfcv_pred_own)




classifier.save("telcom_churn.h5")



from flask import Flask, render_template, request

import keras

from keras.models import load_model



app = Flask(__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'):
   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,i2,i3=1,0,0
if (i =='nis'):
    n1,n2,n3=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=0,1,0
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= 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),int(k1),int(k2),int(k3),int(l1),int(l2),int(l3),int(m1),int(m2),int(m3),int(n1),int(n2),int(n3),int(o1),int(o2),int(o3),int(p1),int(p2),int(p3),int(p4),int(q1)]]

    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)
```