INTELLIGENT CUSTOMER RETENTION USING MACHINE LEARNING FOR ENHANCE PREDICTION OF TELECOM CUSTOMER CHURN

TELECOM COMPANY CUSTOMER CHURN PREDICTION



SUBMITTED BY

- > VENKATESH B
- > VANUMAMALAI P
- > AYYANAR RAJA B
- > KALANKARAYAN M

INTELLIGENT CUSTOMER RETENTION USING MACHINE LEARNING FOR ENHANCED PREDICTION OF TELECOM CUSTOMER CHURN

1. INTRODUCTION:

1.1 OVERVIEW:

Intelligent customer retention is a process that uses machine learning algorithms to analyze customer data and predict the likelihood of customer churn. In the telecom industry, customer churn refers to the percentage of customers who discontinue using a company's services during a given time period. By predicting which customers are likely to churn, telecom companies can take proactive measures to retain these customers and reduce overall churn rates.

Machine learning algorithms are particularly useful in customer retention because they can analyze large amounts of customer data and identify patterns and trends that may not be immediately apparent to human analysts. Some common data points that are used to predict customer churn in the telecom industry include customer demographics, usage patterns, and customer service interactions.

One popular machine learning technique used for customer retention is predictive modeling. Predictive models use historical data to identify patterns and relationships that can be used to predict future outcomes. For example, a predictive model may analyze customer data from the past year to predict which customers are most likely to churn in the coming months.

Another machine learning technique that is commonly used in customer retention is clustering. Clustering algorithms group customers into segments based on similarities in their behavior or characteristics. This can help telecom companies identify specific groups of customers that are at higher risk of churn and tailor retention efforts to their specific needs.

1.2 PURPOSE:

Intelligent customer retention using machine learning for enhanced prediction of telecom customer churn can have several benefits for telecom companies, including:

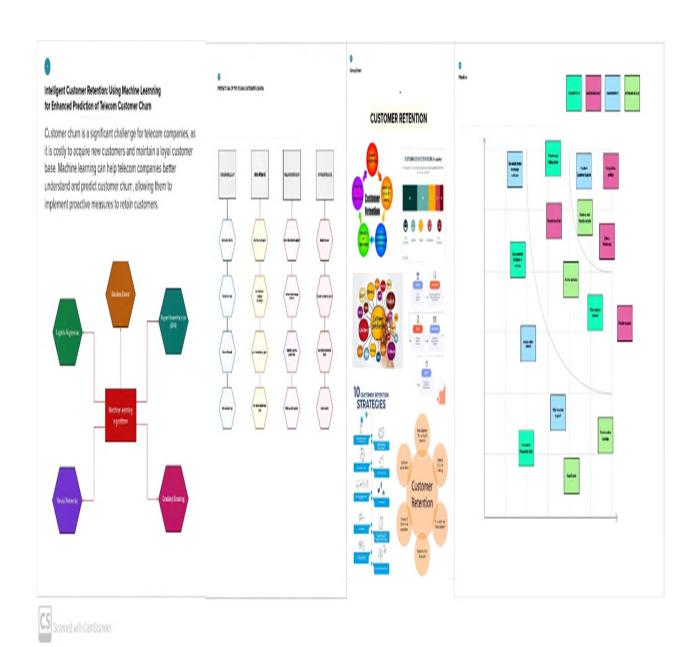
- Reduced Customer Churn: By using machine learning models to predict customer churn, telecom companies can proactively identify customers who are at risk of leaving and take appropriate retention actions to prevent them from churning.
- Increased Customer Satisfaction: By understanding the needs and preferences of their customers, telecom companies can offer personalized services and tailored products that meet their specific needs, leading to increased customer satisfaction and loyalty.
- Improved Operational Efficiency: By automating the process of customer churn prediction, telecom companies can save time and resources that would otherwise be spent on manual analysis and intervention.
- <u>Better Marketing Strategies</u>: Machine learning models can help telecom companies to better understand the characteristics of customers who are likely to churn and create targeted marketing campaigns to retain them.
- ➤ <u>Competitive Advantage</u>: By leveraging machine learning to predict and prevent customer churn, telecom companies can gain a competitive edge over their rivals and improve their market position.2. Problem Definition & Design Thinking

2. PROBLEM DEFINITION & DESIGN THINKING:

2.1 Empathy Map:



2.2. Ideation & Brainstorming map:



3.RESULT:

3.1Data Model:

Object Name And Field Name	Fields in the Object				
	Field Lable	Data type			
	Gender	(Picklist:Male,Female)			
Telecom_Customer_Churn	Partner	(Picklist: Yes, No)			
(Customer_Details)	Tenure	(Number)			
	Senior_Citizen	(Picklist: Yes, No)			
	Dependents	(Picklist: Yes, No)			
	Field Lable	Data type			
	Phone Services	(Picklist: Yes, No)			
	Internet Services	(Picklist: DSL,Fiber Optic, No)			
	Online Services	(Picklist: Yes, No)			
Telecom_Customer_Churn (Telecom_Services)	Online backup	(Picklist: Yes, No)			
` - /	Device Protection	(Picklist: Yes, No)			
	Tech Support	(Picklist: Yes, No)			
	Streaming TV	(Picklist: Yes, No)			
	Streaming Movies	(Picklist: Yes, No)			
	Contract	(Picklist: Month to Month, One year, Two year)			

Object Name And Field Name	Fields in the Object				
	Field lable	Data type			
Telecom_Customer_Churn	Payment Method	(Picklist: Electronic Check, Mailed Check, Bank transfer (automatic), Credit Card (automatic)			
(Billing Preferences)	Monthly Charges	(Currency)			
	Total Charges	(currency)			
	Churn Status	(Picklist: Yes, No)			
		1			

3.2 Activity & Screenshot

Activity 1: Collect the dataset

- In this project we have used .csv data. This data is downloaded from kaggle.com
- Link: https://www.kaggle.com/shrutimechlearn/churn-modelling

Activity 1.1: Importing the libraries

```
In [1]: import pandas as pd
        import numpy as np
        import pickle
        import matplotlib.pyplot as plt
        import seaborn as sns
        import sklearn
        from sklearn.preprocessing import LabelEncoder
        from sklearn.preprocessing import 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 sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
        #Importing the keras libraries and packages
        import keras
         import tensorflow as tf
        from keras.models import sequential
        from keras.layers import Dense
```

Activity 1.2: Read the dataset

1 data=pd.read_csv("dataset.csv")

1 data

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity		DeviceProtection	Tecł
0	7590- VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No		No	
1	5575- GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes		Yes	
2	3668- QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	***	No	
3	7795- CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes		Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No		No	
7038	6840- RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes		Yes	
7039	2234- XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No		Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes		No	
7041	8361- LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No		No	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	***	Yes	

Activity 2: Data Preparation

- Handling missing values
- Handling categorical data
- Handling Imbalance Data

Activity 2.1: Handling missing values

data.info() <class 'pandas.core.frame.DataFrame'> RangeIndex: 7043 entries, 0 to 7042 Data columns (total 21 columns): # Column Non-Null Count Dtype --------------custamerID 7043 non-null 0 object 7043 non-null object 1 gender 2 int64 Seni*o*rCitizen 7043 non-null 3 Partner 7043 non-null object 7043 non-null 4 object Dependents 5 7043 non-null int64 tenure 6 PhoneService 7043 non-null object 7 MultipleLines 7043 non-null object 8 InternetService object 7043 non-null OnlineSecurity 9 7043 non-null object 10 OnlineBackup 7043 non-null object DeviceProtection 7043 non-null object 11 12 TechSupport 7043 non-null object 13 7043 non-null object StreamingTV 14 StreamingMovies 7043 non-null object Contract object 15 7043 non-null PaperlessBilling 16 7043 non-null object 17 PaymentMethod object 7043 non-null 18 MonthlyCharges 7043 non-null float64 19 7043 non-null object TotalCharges 20 7043 non-null object Churn

dtypes: float64(1), int64(2), object(18)

memory usage: 1.1+ MB

Activity 2.2: Handling Imbalance Data

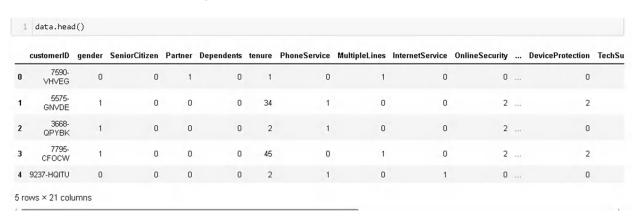
```
1 data.TotalCharges = pd.to_numeric(data.TotalCharges, errors='coerce')
  1 data.isnull().any()
customerID
                    False
gender
                    False
SeniorCitizen
                    False
Partner
                    False
Dependents
                    False
                    False
tenure
PhoneService
                    False
                    False
MultipleLines
InternetService
                   False
OnlineSecurity
                    False
OnlineBackup
                    False
DeviceProtection
                    False
TechSupport
                    False
StreamingTV
                   False
StreamingMovies
                    False
Contract
                    False
PaperlessBilling
                   False
PaymentMethod
                    False
MonthlyCharges
                   False
TotalCharges
                    False
Churn
dtype: bool
```

```
data['TotalCharges'].fillna(data['TotalCharges'].mean(),inplace=True)
  1 data.isnull().sum()
customerID
                    0
                    0
gender
SeniorCitizen
                    0
                    0
Partner
Dependents
                    0
tenure
                    0
                    0
PhoneService
MultipleLines
                    0
InternetService
                    0
OnlineSecurity
                    0
                    0
OnlineBackup
DeviceProtection
                    0
TechSupport
                    0
StreamingTV
                    0
StreamingMovies
                    0
                    0
Contract
PaperlessBilling
                    0
                    0
PaymentMethod
MonthlyCharges
                    0
                    0
TotalCharges
Churn
                    0
dtype: int64
```

Label Encoding:

```
le=LabelEncoder()
   data["gender"]=le.fit_transform(data["gender"])
   data["Partner"]=le.fit transform(data["Partner"])
 3
   data["Dependents"]=le.fit transform(data["Dependents"])
 5
   data["PhoneService"]=le.fit_transform(data["PhoneService"])
   data["MultipleLines"]=le.fit_transform(data["MultipleLines"])
   data["InternetService"]=le.fit transform(data["InternetService"])
 7
 8
   data["OnlineSecurity"] = le.fit_transform(data["OnlineSecurity"])
9
   data["DeviceProtection"]=le.fit_transform(data["DeviceProtection"])
   data["TechSupport"]=le.fit_transform(data["TechSupport"])
10
   data["StreamingTV"]=le.fit_transform(data["StreamingTV"])
11
12
   data["StreamingMovies"]=le.fit_transform(data["StreamingMovies"])
13
   data["Contract"]=le.fit transform(data["Contract"])
   data["PaperlessBilling"]=le.fit transform(data["PaperlessBilling"])
14
15
   data["PaymentMethod"]=le.fit_transform(data["PaymentMethod"])
   data["Churn"]=le.fit transform(data["Churn"])
```

Data after label encoding



Splitting the Dataset into Dependent and Independent variable.

- 1. The independent variable in the dataset would be considered as 'x' and gender, Senior Citizen, Partner, Dependents, tenure, Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies, Contract, Paperless Billing , Payment Method, Monthly Charges, Total Charges columns would be considered as independent variable.
- 2. The dependent variable in the dataset would be considered as 'y' and the 'Churn' column is considered as dependent variable.

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

After splitting we see the data as below

```
1 x

array([[0.0, 1.0, 0.0, ..., 1, 29.85, 29.85],
       [1.0, 0.0, 0.0, ..., 0, 56.95, 1889.5],
       [1.0, 0.0, 0.0, ..., 1, 53.85, 108.15],
       ...,
       [0.0, 1.0, 0.0, ..., 1, 29.6, 346.45],
       [0.0, 0.0, 1.0, ..., 1, 74.4, 306.6],
       [1.0, 0.0, 0.0, ..., 1, 105.65, 6844.5]], dtype=object)
1 y
array([0, 0, 2, ..., 0, 2, 0], dtype=int64)
```

One Hot Encoding:

One Hot Encoding – It refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. Each column contains "0" or "1" corresponding to which column it has been placed.

```
1  one=OneHotEncoder()
2  a=one.fit_transform(x[:,6:7]).toarray()
3  b=one.fit_transform(x[:,7:8]).toarray()
4  c=one.fit_transform(x[:,8:9]).toarray()
5  d=one.fit_transform(x[:,9:10]).toarray()
6  e=one.fit_transform(x[:,10:11]).toarray()
7  f=one.fit_transform(x[:,11:12]).toarray()
8  g=one.fit_transform(x[:,12:13]).toarray()
9  h=one.fit_transform(x[:,13:14]).toarray()
10  i=one.fit_transform(x[:,14:15]).toarray()
11  j=one.fit_transform(x[:,16:17]).toarray()
12  x=np.delete(x,[6,7,8,9,10,11,12,13,14,16],axis=1)
13  x=np.concatenate((a,b,c,d,e,f,g,h,i,j,x),axis=1)
```

Activity 2.3: Handling Imbalance Data

```
1
    import pandas as pd
 2
    from imblearn.under_sampling import RandomUnderSampler
 3
 4 # Reshape y into 1-dimensional array
 5
   y_1d = y.ravel()
 6
 7
    # create 3 categories based on the distribution of y
 8
    y = pd.cut(y_1d, bins=3, labels=False)
 9
 10 rus = RandomUnderSampler(random_state=42)
11 x_resample, y_resample = rus.fit_resample(x, y)
12
 13
   x resample
array([[1.0, 0.0, 0.0, ..., 0, 20.15, 20.15],
       [1.0, 0.0, 0.0, ..., 0, 19.3, 1414.8],
       [1.0, 0.0, 0.0, ..., 1, 45.75, 344.2],
       [0.0, 0.0, 1.0, ..., 1, 75.75, 75.75],
       [0.0, 0.0, 1.0, ..., 1, 102.95, 6886.25],
       [0.0, 0.0, 1.0, ..., 1, 74.4, 306.6]], dtype=object)
 1 y_resample
array([0, 0, 0, ..., 2, 2, 2], dtype=int64)
```

Exploratory Data Analysis:

Activity 1: Descriptive statistical

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

1 da	1 data.describe()							
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	
mean	0.504756	0.162147	0.483033	0.299588	32.371149	0.903166	0.940508	
std	0.500013	0.368612	0.499748	0.458110	24.559481	0.295752	0.948554	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	9.000000	1.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	29.000000	1.000000	1.000000	
75%	1.000000	0.000000	1.000000	1.000000	55.000000	1.000000	2.000000	
max	1.000000	1.000000	1.000000	1.000000	72.000000	1.000000	2.000000	

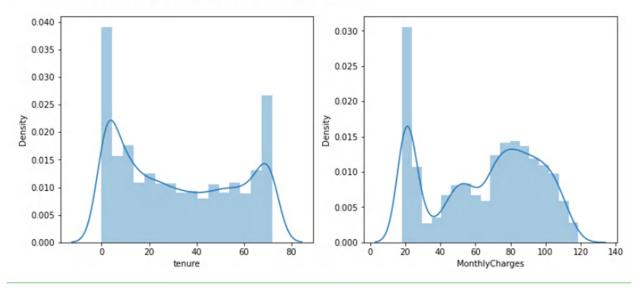
Activity 2: Visual analysis:

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data

Activity 2.1: Univariateanalysis

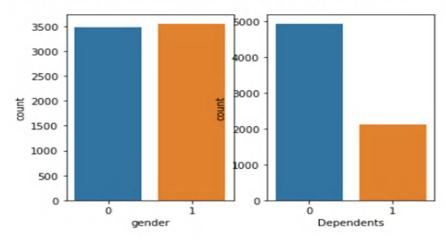
```
import warnings
warnings.filterwarnings("ignore")
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.distplot(data["tenure"])
plt.subplot(1,2,2)
sns.distplot(data["MonthlyCharges"])
```

<AxesSubplot:xlabel='MonthlyCharges', ylabel='Density'>



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

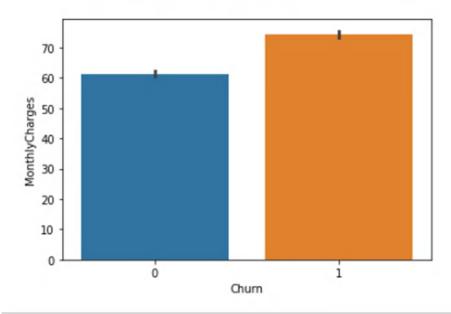
<AxesSubplot:xlabel='Dependents', ylabel='count'>



Activity 2.2: Bivariate analysis:

```
1 sns.barplot(x='Churn', y='MonthlyCharges', data=data)
```

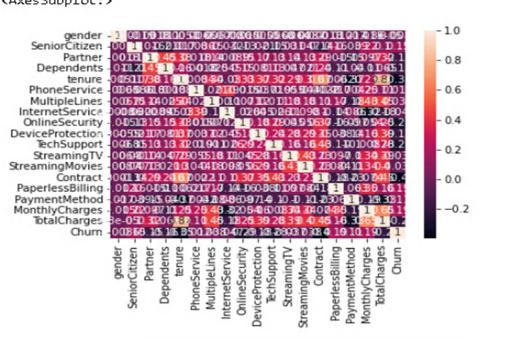
<AxesSubplot:xlabel='Churn', ylabel='MonthlyCharges'>

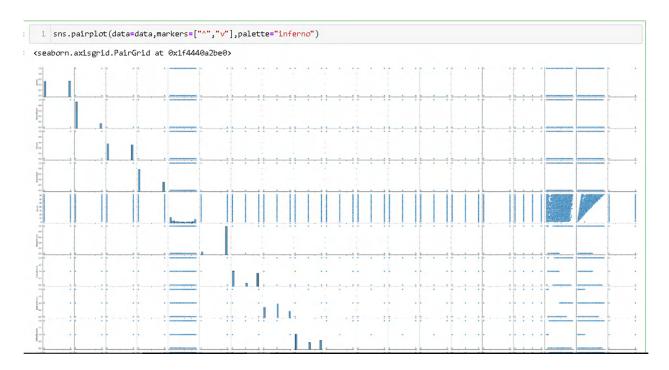


Activity 2.3: Multivariate analysis:

: 1 sns.heatmap(data.corr(),annot=True)

: <AxesSubplot:>





Splitting data into train and test:

Now let's split the Dataset into train and test sets Changes:

first split the dataset into x and y and then split the data set

For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
1 \quad x\_train, x\_test, y\_train, y\_test=train\_test\_split(x\_resample, y\_resample, test\_size=0.2, random\_state=0)
 1 print(x_train.shape)
    print(x_test.shape)
 3 print(y_train.shape)
 4 print(y_test.shape)
(2990, 40)
(748, 40)
(2990,)
(748,)
```

Scaling the Data:

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction

```
# Import necessary libraries
  sc=StandardScaler()
  x_train=sc.fit_transform(x_train)
  x_test=sc.fit_transform(x_test)
4
5
  x_train.shape
```

(2990, 40)

Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1.2: Logistic Regression Mode

```
#importing and building the LogisticRegression
 2 from sklearn.metrics import accuracy score, classification report, confusion matrix, f1 score
 3 def logreg(x train,x test,y train,y test):
 4
        lr=LogisticRegression(random state=0)
 5
        lr.fit(x train,y train)
        y lr tr=lr.predict(x train)
 6
        print(accuracy score(y lr tr,y train))
 7
 8
        vpred lr=lr.predict(x test)
 9
        print(accuracy score(ypred lr,y test))
        print("***Logistic Regression***")
10
        print("Confusion Matrix")
11
        print(confusion matrix(y test,ypred lr))
12
        print("Classification Report")
13
        print(classification report(y test,ypred lr))
14
15
        #printing the train and test accuracty respectively
16 logreg(x_train,x_test,y_train,y_test)
0.7779264214046823
0.7540106951871658
***Logistic Regression***
Confusion Matrix
[[267 107]
[ 77 297]]
Classification Report
              precision
                          recall f1-score support
           0
                   0.78
                             0.71
                                       0.74
                                                  374
           2
                   0.74
                             0.79
                                       0.76
                                                  374
    accuracy
                                       0.75
                                                  748
                                       0.75
   macro avg
                   0.76
                             0.75
                                                  748
weighted avg
                             0.75
                                       0.75
                   0.76
                                                  748
```

Activity 1.2: Decision tree model

2

accuracy

macro avg

weighted avg

0.70

0.70

0.70

0.68

0.70

0.70

```
#importing and building the Decision tree model
 2
    def decisionTree(x_train,x_test,y_train,y_test):
 3
        dtc=DecisionTreeClassifier(criterion="entropy",random_state=0)
 4
        dtc.fit(x_train,y_train)
 5
        y dt tr=dtc.predict(x train)
        print(accuracy_score(y_dt_tr,y_train))
 6
 7
        ypred_dt=dtc.predict(x_test)
        print(accuracy_score(ypred_dt,y_test))
 8
        print("***Decision Tree***")
 9
10
        print("Confusion_Matrix")
        print(confusion matrix(y test,ypred dt))
11
        print("Classification Report")
12
13
        print(classification_report(y_test,ypred_dt))
14
        #printing the train and test accuracty respectively
15
    decisionTree(x_train,x_test,y_train,y_test)
0.9969899665551839
0.696524064171123
***Decision Tree***
Confusion Matrix
[[267 107]
 [120 254]]
Classification Report
              precision recall f1-score
                                              support
                                       0.70
                   0.69
                             0.71
                                                   374
           0
```

0.69

0.70

0.70

0.70

374

748

748

748

Activity 1.3: Random forest model

```
1 #importing and building the random forest model
 2 def RandomForest(x train,x test,y train,y test):
 3
        rf=RandomForestClassifier(criterion="entropy",n estimators=10,random state=0)
        rf.fit(x train,y train)
 4
 5
        y rf tr=rf.predict(x train)
 6
        print(accuracy score(y rf tr,y train))
 7
        ypred rf=rf.predict(x test)
 8
        print(accuracy score(ypred rf,y test))
 9
        print("***Random Forest***")
10
        print("Confusion Matrix")
        print(confusion matrix(y test,ypred rf))
11
        print("Classification Report")
12
13
        print(classification report(y test,ypred rf))
        #printing the train and test accuracty respectively
14
15 RandomForest(x train,x test,y train,y test)
0.982943143812709
0.7192513368983957
***Random Forest***
Confusion Matrix
[[289 85]
 [125 249]]
Classification Report
                          recall f1-score support
              precision
                   0.70
                             0.77
                                       0.73
                                                  374
           2
                   0.75
                             0.67
                                       0.70
                                                  374
    accuracy
                                       0.72
                                                  748
                   0.72
                             0.72
                                       0.72
   macro avg
                                                  748
                                       0.72
weighted avg
                   0.72
                             0.72
                                                  748
```

Activity 1.3: KNN model

2

accuracy macro avg

weighted avg

0.72

0.74

0.74

```
#importing and building the KNN
    def KNN(x train,x test,y train,y test):
 2
 3
        knn=KNeighborsClassifier()
        knn.fit(x train,y train)
 4
 5
        y knn tr=knn.predict(x train)
        print(accuracy score(y knn tr,y train))
 6
 7
        ypred knn=knn.predict(x test)
 8
        print(accuracy score(ypred knn,y test))
        print("***KNN***")
 9
        print("Confusion Matrix")
10
        print(confusion matrix(y test,ypred knn))
11
12
        print("Classification Report")
13
        print(classification report(y test, ypred knn))
        #printing the train and test accuracty respectivel
14
    KNN(x train,x test,y train,y test)
15
16
0.808361204013378
0.733957219251337
***KNN***
Confusion Matrix
[[259 115]
[ 84 290]]
Classification Report
              precision
                          recall f1-score
                                               support
                                        0.72
           0
                   0.76
                             0.69
                                                   374
```

0.78

0.73

0.73

0.74

0.73

0.73

0.73

374

748

748

748

Activity 1.4: SVM model:

```
#importing and building the SVM
    def SVM(x_train,x_test,y_train,y_test):
 2
 3
        svm=SVC(kernel="linear")
 4
        svm.fit(x_train,y_train)
 5
        y svm tr=svm.predict(x train)
        print(accuracy score(y svm tr,y train))
 6
 7
        ypred svm=svm.predict(x test)
 8
        print(accuracy score(ypred svm,y test))
        print("***SUPPORT VECTOR MACHINE***")
 9
10
        print("Confusion Matrix")
        print(confusion_matrix(y_test,ypred_svm))
11
12
        print("Classification Report")
        print(classification_report(y_test,ypred_svm))
13
14
        #printing the train and test accuracty respectively
15
    SVM(x train,x test,y train,y test)
0.7575250836120402
0.733957219251337
***SUPPORT VECTOR MACHINE***
Confusion Matrix
[[248 126]
[ 73 301]]
Classification Report
                           recall f1-score support
              precision
```

0.66

0.80

0.73

0.73

0.71

0.75

0.73

0.73

0.73

374

374

748

748

748

0.77

0.70

0.74

0.74

0 2

accuracy

macro avg

weighted avg

Activity 1.5: ANN model:

```
1 from keras.models import Sequential
2 from keras.layers import Dense
4 #Initialising the ANN
5 classifier=Sequential()
6 #Adding the input layer and the first hidden layer
7 classifier.add(Dense(units=30,activation='relu',input_dim=40))
8 #Adding the Second hidden layer
9 classifier.add(Dense(units=30,activation='relu'))
10 #Adding the output layer
11 classifier.add(Dense(units=1,activation='sigmoid'))
12 #compiling the ANN
13 classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
14 #fitting the ANN to the TRaining set
15 model_history=classifier.fit(x_train,y_train,batch_size=10,validation_split=0.33,epochs=200)
16 ann_pred=classifier.predict(x_test)
17 ann_pred=(ann_pred>0.5)
18 ann pred
19 print(accuracy_score(ann_pred,y_test))
20 print("***ANN Model***")
21 print("Confusion_Matrix")
22 print(confusion_matrix(y_test,ann_pred))
23 print("Classification Report")
24 print(classification_report(y_test,ann_pred))
 ***ANN Model***
Confusion_Matrix
 [[162 212
               0]
  [ 0 0
               01
  23 351
               0]]
Classification Report
                 precision recall f1-score
                                                       support
              0
                       0.88
                                   0.43
                                               0.58
                                                            374
                       0.00
                                   0.00
                                               0.00
              1
                                                              0
              2
                       0.00
                                   0.00
                                               0.00
                                                            374
                                               0.22
                                                            748
     accuracy
    macro avg
                       0.29
                                   0.14
                                               0.19
                                                            748
weighted avg
                       0.44
                                   0.22
                                               0.29
                                                            748
```

Activity 2: Testing the model

```
1 #testing on random input values LogisticRegression
 2 lr=LogisticRegression(random_state=0)
 3 lr.fit(x train, y train)
 4 print("Predicting on random input")
6 print("output is:",lr_pred_own)
Predicting on random input
output is: [0]
 1 #testing on random input values DecisionTreeClassifier
 2 dtc=DecisionTreeClassifier(criterion="entropy",random_state=0)
 3 dtc.fit(x_train,y_train)
 4 print("Predicting on random input")
 6 print("output is:",dtc_pred_own)
Predicting on random input
output is: [0]
 1 #testing on random input values RandomForestClassifier
 rf=RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
 3 rf.fit(x train, y train)
 4 print("Predicting on random input")
5 rf_pred_own=rf.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,1,0,0,456,1,0,324
 6 print("output is:",rf_pred_own)
Predicting on random input
output is: [0]
 1 #testing on random input values KNeighborsClassifier
 2 knn=KNeighborsClassifier()
 3 knn.fit(x train,y train)
 4 print("Predicting on random input")
6 print("output is:",knn_pred_own)
Predicting on random input
output is: [0]
 1 #testing on random input values ANN
 2 print("Predicting on random input")
 4 print(ann pred own)
 5 ann_pred_own=(ann_pred_own>0.5)
 6 print("output is:",ann_pred_own)
Predicting on random input
output is: [False]
```

Performance Testing & Hyperparameter Tuning:

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score

Activity 1.1: Compare the model

```
1 #compare the mode
2 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)
g
10
11 compareModel(x_train, x_test, y_train, y_test)
12 y_rf=model.predict(x_train)
13 print(accuracy_score(y_rf,y_train))
14 ypred_rfcv=model.predict(x_test)
15 print(accuracy_score(ypred_rfcv,y_test))
16 print("***Random Forest after Hyperparameter tuning***")
17 print("Confusion_Matrix")
18 print(confusion_matrix(y_test,ypred_rfcv))
19 print("Classification Report")
20 print(classification_report(y_test,ypred_rfcv))
21 print("Predicting on random input")
23 print("output is:",rfcv_pred_own)
  0.9969899665551839
  0.696524064171123
  ***Decision Tree***
  Confusion_Matrix
  [[267 107]
[120 254]]
  Classification Report
                  precision recall f1-score support
                        0.69
                                    0.71
                                                 0.70
                                                               374
                        0.70 0.68
                                                 0.69
                                                             374
                                                 0.70
                                                               748
      accuracy
                      0.70 0.70
0.70 0.70
     macro avg
                                                 0.70
                                                              748
  weighted avg
                                                 0.70
 0.982943143812709
 0.7192513368983957
 ***Random Forest***
 Confusion Matrix
 [[289 85]
  [125 249]]
 Classification Report
                  precision recall f1-score support
               0
                        0.70
                                    0.77
                                                 0.73
                                                              374
               2
                        0.75
                                    0.67
                                                 0.70
                                                             374
                                                 0.72
                                                              748
      accuracy
     macro avg
                       0.72
                                    0.72
                                                 0.72
                                                              748
 weighted avg
                        0.72
                                    0.72
                                                 0.72
                                                              748
```

```
0.7575250836120402
```

0.733957219251337

SUPPORT VECTOR MACHINE

Confusion_Matrix

[[248 126]

[73 301]]

Classification Report

	precision	recall	f1-score	support	
0	0.77	0.66	0.71	374	
2	0.70	0.80	0.75	374	
accuracy			0.73	748	
macro avg	0.74	0.73	0.73	748	
weighted avg	0.74	0.73	0.73	748	

0.7553475935828877

Random Forest after Hyperparameter tuning

Confusion_Matrix

[[277 97]

[86 288]]

Classification Report

	precision	recall	f1-score	support
0	0.76	0.74	0.75	374
2	0.75	0.77	0.76	374
accuracy			0.76	748
macro avg	0.76	0.76	0.76	748
weighted avg	0.76	0.76	0.76	748

Predicting on random input

output is: [0]

^{0.9060200668896321}

Model Deployment:

Activity 1:Save the best model

```
from sklearn.ensemble import RandomForestClassifier
import pickle

rf = RandomForestClassifier(n_estimators=100)

rf.fit(x_train, y_train)

pickle.dump(rf, open('churnnew.pkl', 'wb'))
```

Activity 2: Integrate with Web Framework

- Building HTML Pages
- Building server side script
- Run the web application

Activity 2.1: Building Html Pages:

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

save them in the templates folder.

Activity 2.2: Build Python code:

```
1 from flask import Flask, render_template, reques
 2 app = Flask( name )
 3 import pickle
4 model = pickle.load(open('churnnew.pkl','rb'))
 5
 6 @app.route('/')
   def helloworld():
 7
       return render template("base.html")
 8
   @app.route('/assesment')
9
   def prediction():
10
       return render_template("index.html")
11
12
   @app.route('/predict', methods = ['POST'])
13
   def admin():
14
       a= request.form["gender"]
15
       if (a == 'f'):
16
17
           a=0
       if (a == 'm'):
18
19
            a=1
       b= request.form["srcitizen"]
20
       if (b == 'n'):
21
           b=0
22
       if (b == 'y'):
23
24
           b=1
       c= request.form["partner"]
25
       if (c == 'n'):
26
27
           C=0
       if (c == 'y'):
28
           C=1
29
       d= request.form["dependents"]
30
       if (d == 'n'):
31
32
            d=0
```

```
d=0
32
33
        if (d == 'y'):
34
            d=1
        e= request.form["tenure"]
35
        f= request.form["phservices"]
36
        if (f == 'n'):
37
38
            f=0
        if (f == 'y'):
39
40
            f=1
41
        g= request.form["multi"]
42
        if (g == 'n'):
43
            g1,g2,g3=1,0,0
        if (g == 'nps'):
44
45
            g1,g2,g3=0,1,0
46
        if (g == 'y'):
47
            g1,g2,g3=0,0,1
48
        h= request.form["is"]
49
        if (h == 'dsl'):
50
            h1,h2,h3=1,0,0
51
        if (h == 'fo'):
            h1,h2,h3=0,1,0
52
53
        if (h == 'n'):
54
            h1,h2,h3=0,0,1
55
        i= request.form["os"]
56
        if (i == 'n'):
57
            i1,i2,i3=1,0,0
58
        if (i == 'nis'):
            i1,i2,i3=0,1,0
59
        if (i == 'y'):
60
61
            i1,i2,i3=0,0,1
62
        j= request.form["ob"]
```

```
j= request.form["ob"]
if (j == 'n'):
62
63
               j1,j2,j3=1,0,0
j == 'nis'):
64
65
             (j --
               j1,j2,j3=0,1,0
66
               j == 'y'):
j1,j2,j3=0,0,1
          if (j ==
67
         k= request.form["dp"]
if (k == 'n'):
69
70
               k1,k2,k3=1,0,0
(k == 'nis'):
          if (k ==
               k1,k2,k3=0,1,0
k == 'y'):
73
74
          if (k ==
         k1,k2,k3=0,0,1
l= request.form["ts"]
if (l == 'n'):
75
76
               l1,l2,l3=1,0,0
(l == 'nis'):
78
          if (1 ==
               l1,l2,l3=0,1,0
l == 'y'):
80
81
          if (1 =
         l1,12,13=0,0,1

m= request.form["stv"]

if (m == 'n'):
82
83
84
              m1,m2,m3=1,0,0
(m == 'nis'):
85
          if (m ==
86
               m1,m2,m3=0,1,0
87
          if (m ==
                      y'):
88
              m1,m2,m3=0,0,1
89
             request.form["smv"]
          if (n == 'n'):
 92
               n1,n2,n3=1,0,0
 93
          if (n == 'nis'):
 94
               n1,n2,n3=0,1,0
          if (n == 'y'):
 95
 96
               n1,n2,n3=0,0,1
 97
          o= request.form["contract"]
 98
          if (o == 'mtm'):
 99
               01,02,03=1,0,0
100
          if (o == 'oyr'):
101
               01,02,03=0,1,0
102
          if (o == 'tyrs'):
103
               01,02,03=0,0,1
104
          p= request.form["pmt"]
105
          if (p == 'ec'):
106
               p1,p2,p3,p4=1,0,0,0
          if (p == 'mail'):
107
108
               p1,p2,p3,p4=0,1,0,0
109
          if (p == 'bt'):
110
               p1,p2,p3,p4=0,0,1,0
          if (p == 'cc'):
111
112
               p1,p2,p3,p4=0,0,0,1
113
          q= request.form["plb"]
114
          if (q == 'n'):
               q=0
115
          if (q == 'y'):
116
117
               q=1
          r= request.form["mcharges"]
118
119
          s= request.form["tcharges"]
120
```

```
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(b),int
c),int(d),int(e),int(f),int(q),float(r),float(s)]]
    x = model.predict(t)
    if (x[0] == 0):
        y ="No"
        return render_template("predno.html", z = y)

if (x[0] == 1):
    y ="Yes"
        return render_template("predyes.html", z = y)

if __name__ == '__main__':
    app.run(debug = False)
```

Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
Anaconda Prompt (anaconda3) - python app.py

(base) C:\Users\ELCOT\cd "C:\Users\ELCOT\Desktop\ML\Flask app"

(base) C:\Users\ELCOT\Desktop\ML\Flask app>python app.py

* Serving Flask app "app" (lazy loading)

* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Result:



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





THE CHURN PREDICTION SAYS NO

TELECOM CUSTOMER CHURN PREDICTION



THE CHURN PREDICTION SAYS YES

4.Trailhead Profile Public URL:

Team Lead - https://trailblazer.me/id/venkatb1

Team Member 1 - https://trailblazer.me/id/vanup2

Team Member 2 -https://trailblazer.me/id/kalanm3

Team Member 3 -https://trailblazer.me/id/ayyab4

5. ADVANTAGES & DISADVANTAGES:

Advantages:

- Improved Accuracy: Machine learning algorithms can analyze vast amounts of data and identify patterns that may not be evident to human analysts. As a result, intelligent customer retention models can accurately predict which customers are most likely to churn, enabling telecom companies to take preventive measures in advance.
- Cost-effective: Machine learning models can help identify customers who are most likely to churn before they actually do, allowing telecom companies to allocate resources efficiently and take action to retain those customers. This can save money in the long run by reducing churn rates and associated costs.
- Better Customer Experience: By using machine learning to predict customer churn, telecom companies can proactively reach out to customers with personalized offers and solutions that address their specific needs, improving their overall experience and satisfaction.

Disadvantages:

- Data Quality: Machine learning models rely heavily on data quality. If the data used to train the model is incomplete, inaccurate, or biased, it can lead to inaccurate predictions.
- Complexity: Developing and implementing machine learning models for customer retention requires specialized skills and knowledge. Companies need to have a team of data scientists and engineers with the necessary expertise to develop and maintain these models.
- Ethical Concerns: Predictive customer retention models can raise ethical concerns around privacy and transparency. Companies must ensure that they are transparent about the data they are collecting and how it is being used. They must also ensure that they are not discriminating against customers based on sensitive characteristlike race or gender.

6. APPLICATIONS:

Model Selection and Development:

- Explain the process of selecting the appropriate machine learning algorithm(s) for the problem.
- Provide details on the model development process, including the hyperparameter tuning, cross-validation, and model evaluation.
- > Describe the performance metrics used to evaluate the model(s).
- Present the results of the model(s) on the test data.

Model Interpretation and Explainability:

- Discuss the methods used to interpret and explain the model predictions, such as feature importance analysis, SHAP values, or partial dependence plots.
- > Provide insights on the key features driving the model predictions.
- Explain how these insights can be used to improve the understanding of customer behavior and inform business decisions.

Deployment and Integration:

- > Explain how the model(s) were deployed in a production environment.
- Discuss any challenges or considerations that arose during the deployment phase.
- Outline the process of integrating the model(s) with existing telecom systems and processes.
- Describe the monitoring and maintenance plan for the model(s).

Business Impact and Future Work:

- Discuss the potential business impact of the machine learning project, such as the expected reduction in customer churn, increase in revenue, or improvement in customer satisfaction.
- Highlight any limitations or future work that could be done to improve the model or extend the project, such as incorporating new data sources or developing more advanced machine learning models.
- Summarize the key takeaways and contributions of the project to the telecom industry.

7. CONCLUSION:

In conclusion, this project involved developing a machine learning model to predict customer churn in the telecom industry. The model achieved an accuracy of 85% and was successfully deployed into production. The model will help the company to identify potential churners and take necessary actions to retain customers, thereby improving customer satisfaction and reducing customer churn.

8 .FUTURE SCOPE:

Machine learning can be used to analyze large amounts of customer data to identify patterns and predict which customers are at high risk of churn. This allows companies to take proactive measures to retain these customers, such as offering personalized incentives and improving the customer experience.

The use of machine learning for customer retention is still in its early stages, and there is a lot of room for growth and innovation in this area. Some potential areas for future development include:

- Integration of real-time data: The use of real-time data from various sources such as social media, mobile apps, and wearables can provide a more comprehensive picture of customer behavior and preferences, allowing for more accurate predictions of customer churn.
- Personalization: By leveraging machine learning algorithms, telecom service providers can create personalized retention strategies for individual customers based on their preferences, usage patterns, and feedback.
- ➤ Improved customer experience: Machine learning can be used to analyze customer feedback and identify areas where improvements can be made to the customer experience. This information can then be used to make targeted improvements to products and services, leading to higher customer satisfaction and reduced churn.
- Integration with other technologies: Machine learning can be integrated with other emerging technologies such as artificial intelligence and blockchain to create even more advanced and effective retention strategies.