MACHINE LEARNING WITH PYTHON

TITLE:PREDICTION OF TELECOM CUSTOMER CHURN

BY

T.ASHWINI

S.MADHANKUMAR

P.SATHISH

H.SWETHA

CONTENTS

1.INRODUCTION

1.1 OVERVIEW

Telecom Customer Retention Using Machine Learning

- 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.
- 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 beused. 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 acompany losing profits because of the discounts offered to continuous subscribers.
- Telecommunication industry always suffers from a very high churn rates when one industry offers a better plan than the previous there is a high possibility of the customer churning from the present due to a better plan in such a scenario it is very difficult to avoid losses but through prediction we can keep it to a minimal level.
- Telecom companies often use customer churn as a key business metrics to predict the number of customers that will leave a telecom service provider. A machine learning model can be used to identity the probable churn customers and then makes the necessarybusiness decisions.

1.2 Purpose

- Intelligent customer retention is a crucial aspect of customer relationship management for telecom companies. One of the most significant challenges in this domain is the prediction of customer churn, which refers to the phenomenon where customers decide to discontinue their service subscription with a telecom company. To address this challenge,machine learning techniques can be used to enhance the accuracy of churn predictionmodels.

- To build a machine learning model for churn prediction, a historical dataset of customer behavior, usage patterns, demographics, and other relevant variables should be collected. This dataset can then be used to train a model that can identify patterns in the data and make predictions about which customers are likely to churn in the future.

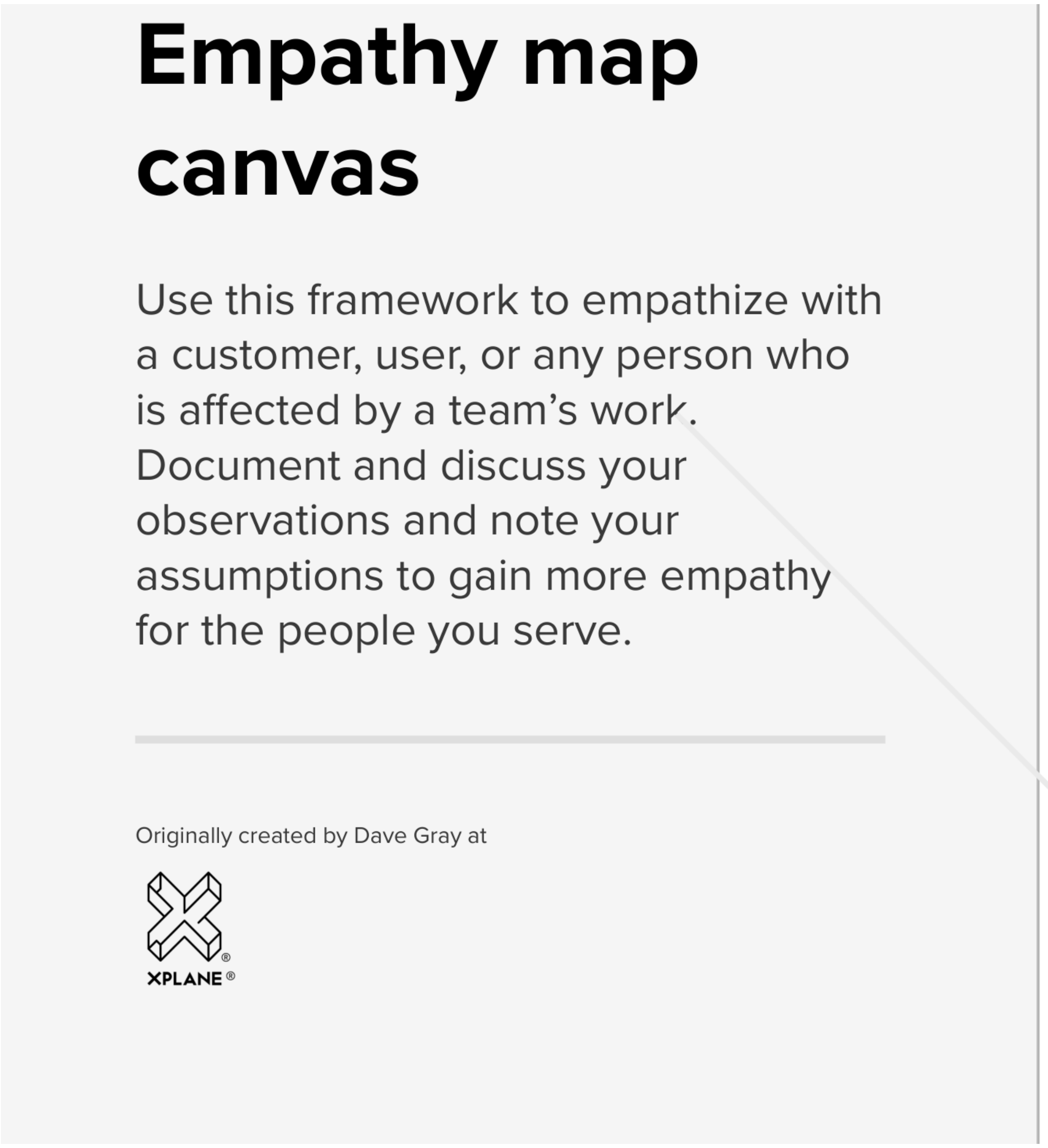
- Several machine learning algorithms can be used for churn prediction, including logistic regression, decision trees, random forests, and neural networks. These models can be further enhanced through the use of techniques such as ensemble learning, feature selection, and hyperparameter optimization.

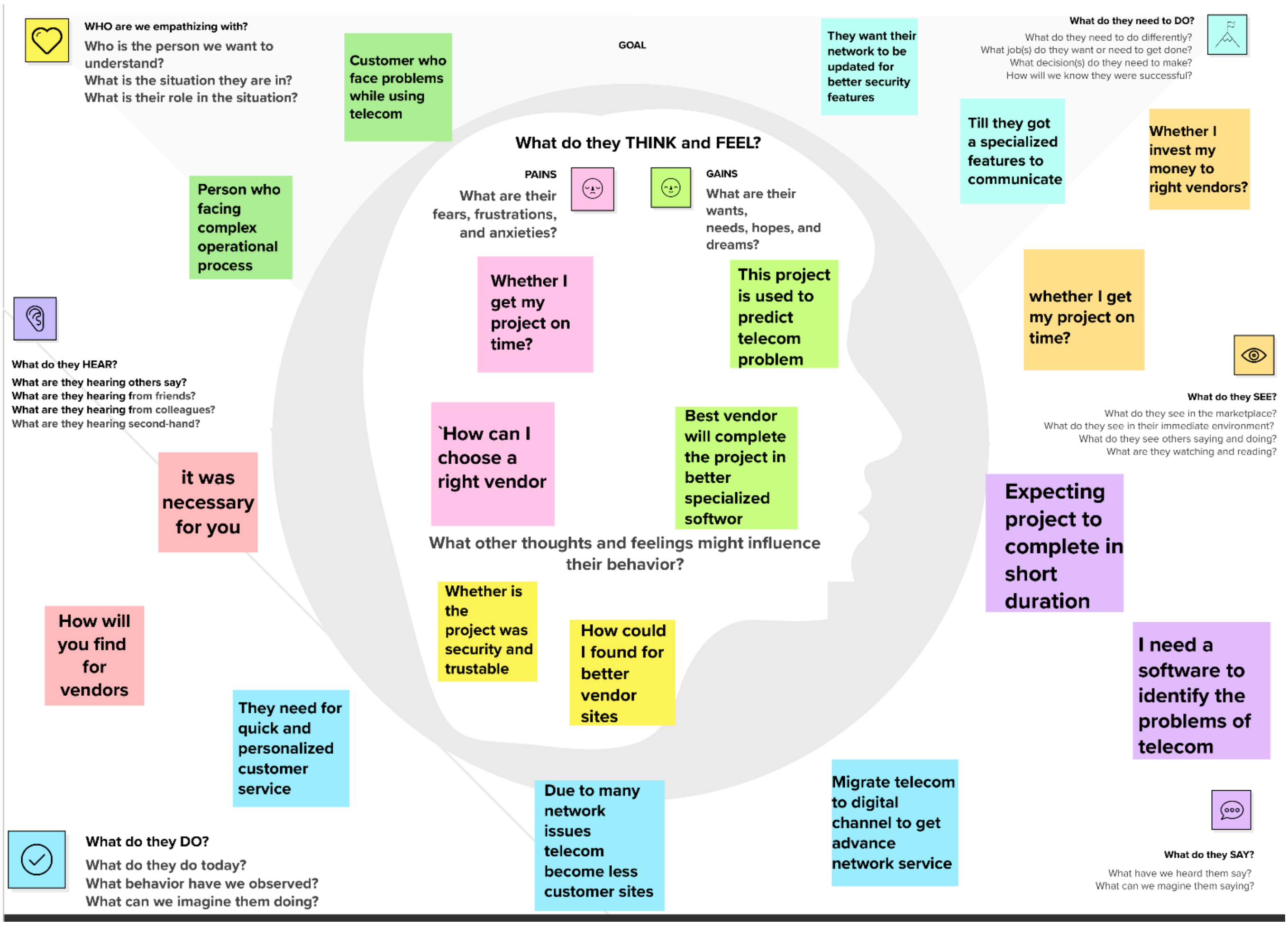
- Once a churn prediction model has been developed, it can be integrated into a customer retention strategy. This strategy may involve targeted marketing campaigns, personalized offers, loyalty programs, and proactive customer service. By leveraging machine learning for intelligent customer retention, telecom companies can reduce churn rates, improve customer satisfaction, and increase revenue.

1. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map

In the ideation phase we have empathized as our client Optimizing spam filtering with machine learning and we have acquired the details which are represented in the Empathy Map given below.



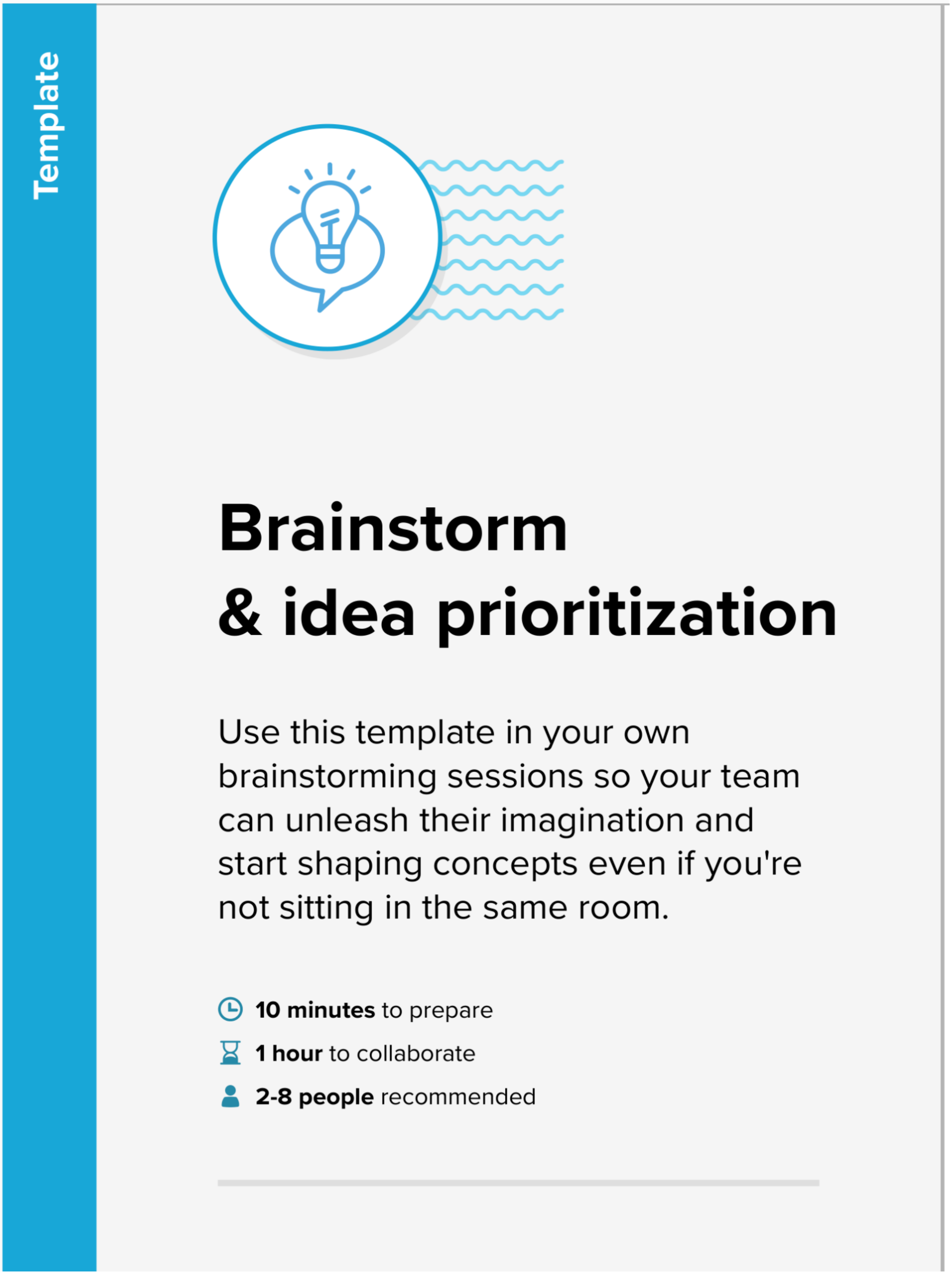


WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
Customer who face problems while using telecom
Person who facing complex operational process
GOAL
They want their network to be updated for better security features
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Till they got a specialized features to communicate
Whether I invest my money to right vendors?
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
Whether I get my project on time?
This project is used to predict telecom problem
whether I get my project on time?
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
`How can I choose a right vendor
Best vendor will complete the project in better specialized softwor
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
it was necessary for you
What other thoughts and feelings might influence their behavior?
Expecting project to complete in short duration
How will you find for vendors
Whether is the project was security and trustable
How could I found for better vendor sites
I need a software to identify the problems of telecom
They need for quick and personalized customer service
Due to many network issues telecom become less customer sites
Migrate telecom to digital channel to get advance network service
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
What do they SAY?
What have we heard them say?
What can we magine them saying?

2.2 IDEATION 7 BRAINSTORMING MAP

Under this activity our team members have gathered and discussed various idea to solve our project problem. Each member contributed 6 to 10 ideas after gathering all ideas we have assessed the impact and feasibility of each point. Finally, we have assign the priority for each point based on the impact value.

STEP-1: Team Gathering, Collaboration and Select the Problem



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**

Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**

Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

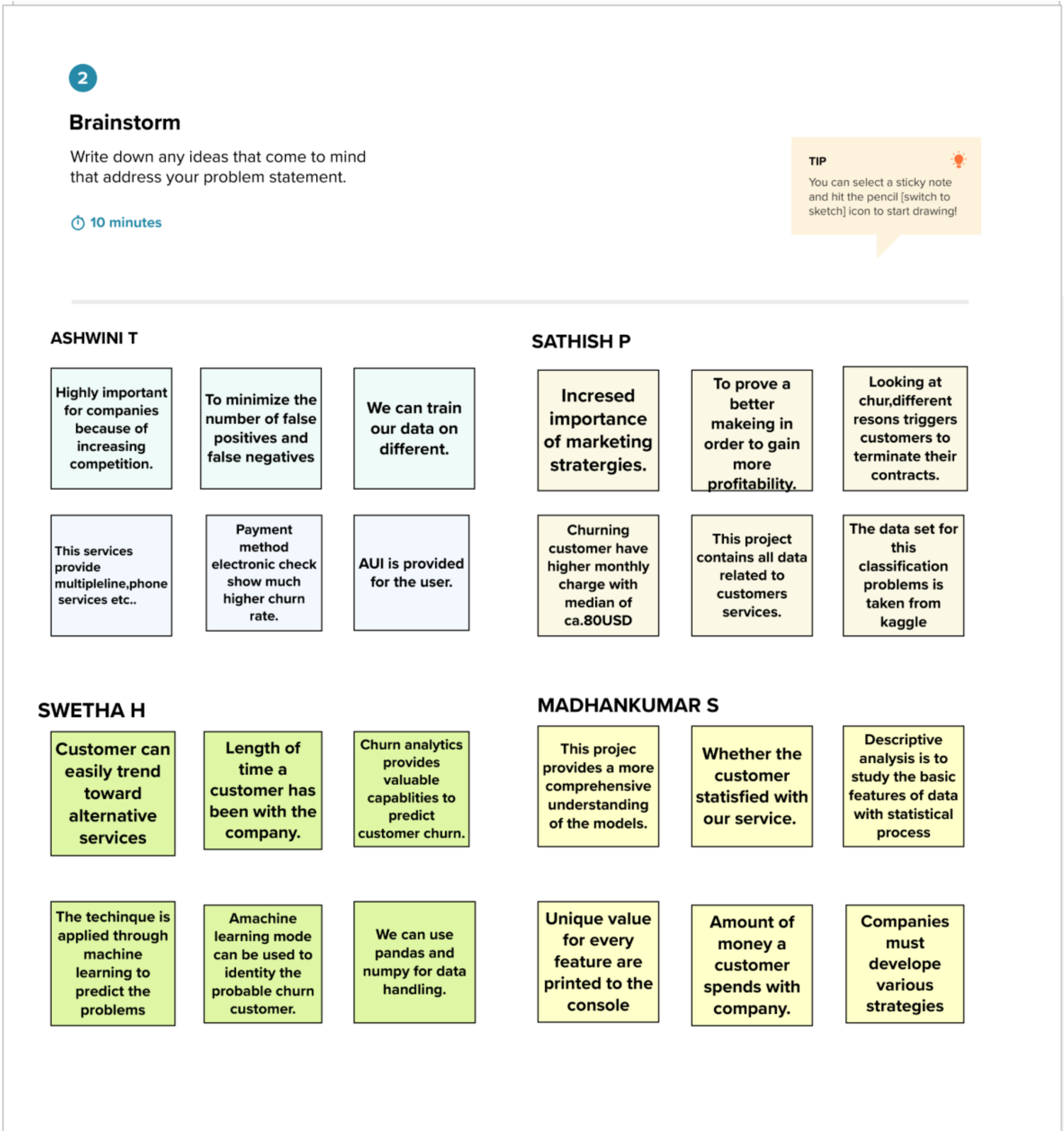
How might we [Using Machine Learning For Enhanced Prediction Of Telecom Customer Churn]

Key rules of brainstorming

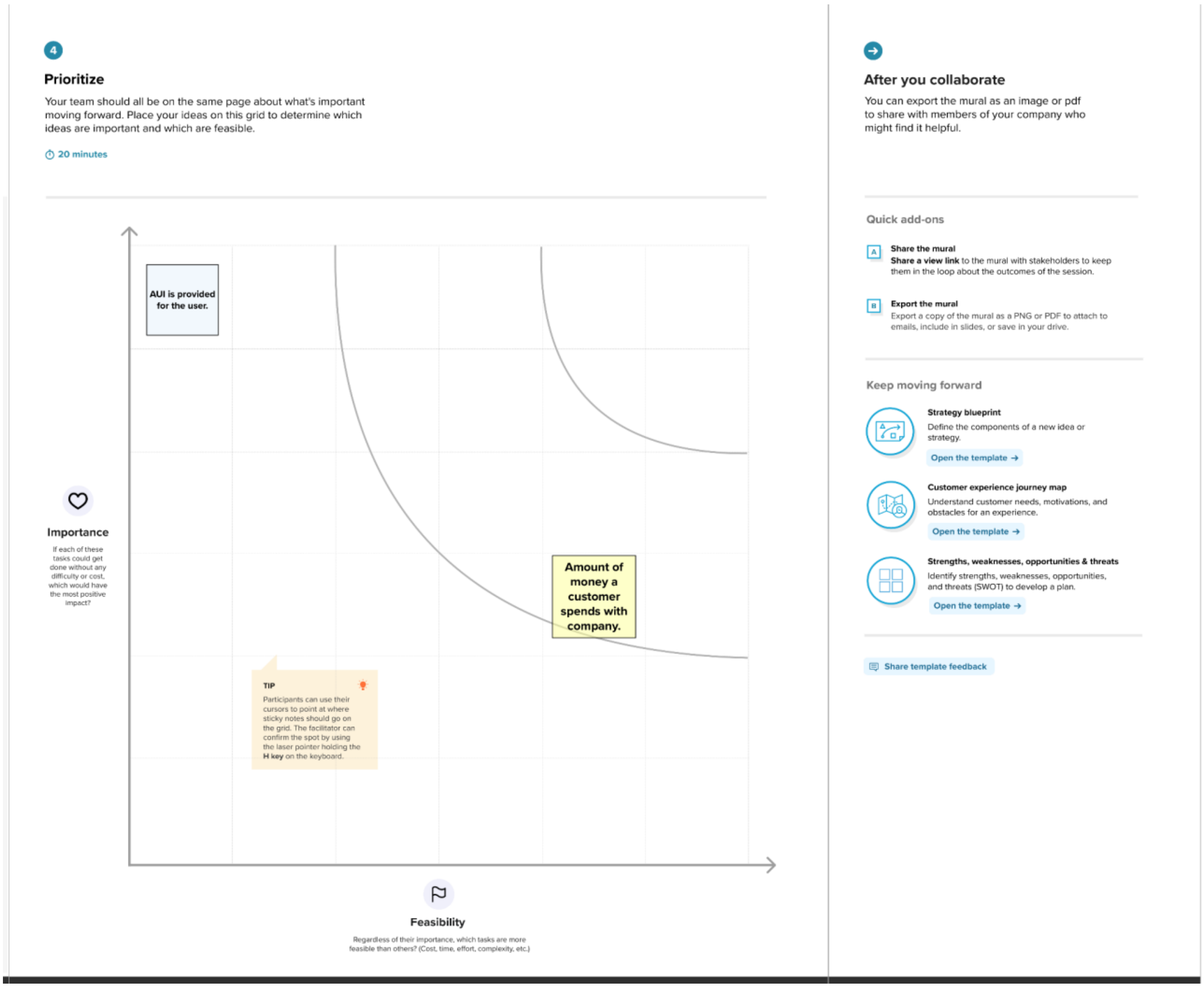
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

STEP-2: Brainstorm, Idea Listing and Grouping



STEP-3: Idea Prioritization



2. RESULT

Read the datasets

```
data = pd.read_csv("//content/Telecom Churn Rate Dataset.csv")
data
```

	Gender	Senior_Citizen	Partner	Dependents	Tenure	Phone_Service	Multiple_Lines	Internet_Service	Online_Security	Online_Backup	...	Streaming_TV	Streaming_Movies	Contract	Paper_less_Billing	Payment_Method	Monthly_Charges	Yearly_Charge	Admin_Tickets	T
0	Male	Yes	No	No	1	No	No phone service	DSL	No	No	...	No	Yes	Month-to-month	Yes	Electronic check	396.5	4758	0	
1	Female	Yes	Yes	No	71	Yes	Yes	Fiber optic	Yes	Yes	...	No	No	Two year	Yes	Credit card (automatic)	963.5	11562	0	
2	Male	Yes	Yes	No	2	Yes	No	Fiber optic	No	No	...	Yes	Yes	Month-to-month	Yes	Credit card (automatic)	955.0	11460	0	
3	Male	Yes	No	No	1	Yes	No	DSL	No	No	...	No	No	Month-to-month	No	Bank transfer (automatic)	452.5	5430	0	
4	Female	Yes	No	No	43	Yes	Yes	Fiber optic	No	Yes	...	Yes	No	Month-to-month	Yes	Electronic check	902.5	10830	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1137	Female	Yes	Yes	No	63	Yes	Yes	Fiber optic	No	Yes	...	Yes	Yes	Month-to-month	Yes	Electronic check	1035.0	12420	0	
1138	Female	Yes	No	No	6	No	No phone service	DSL	No	No	...	Yes	Yes	Month-to-month	Yes	Electronic check	444.0	5328	0	
1139	Male	Yes	Yes	No	55	Yes	Yes	DSL	Yes	Yes	...	No	No	One year	No	Credit card (automatic)	600.0	7200	0	
1140	Male	Yes	No	No	1	Yes	Yes	Fiber optic	No	No	...	No	No	Month-to-month	Yes	Electronic check	757.5	9090	0	
1141	Male	Yes	Yes	No	4	Yes	Yes	Fiber optic	No	No	...	No	No	Month-to-month	Yes	Mailed check	744.0	8928	0	

1142 rows × 22 columns

Handling missing values

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1142 entries, 0 to 1141
Data columns (total 22 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             1142 non-null   object
 1   Senior_Citizen     1142 non-null   object
 2   Partner            1142 non-null   object
 3   Dependents         1142 non-null   object
 4   Tenure             1142 non-null   int64
 5   Phone_Service      1142 non-null   object
 6   Multiple_Lines     1142 non-null   object
 7   Internet_Service   1142 non-null   object
 8   Online_Security    1142 non-null   object
 9   Online_Backup      1142 non-null   object
 10  Device_Protection  1142 non-null   object
 11  Tech_Support       1142 non-null   object
 12  Streaming_TV       1142 non-null   object
 13  Streaming_Movies   1142 non-null   object
 14  Contract           1142 non-null   object
 15  Paper_less_Billing 1142 non-null   object
 16  Payment_Method     1142 non-null   object
 17  Monthly_Charges    1142 non-null   float64
 18  Yearly_Charge      1142 non-null   int64
 19  Admin_Tickets      1142 non-null   int64
 20  Tech_Tickets       1142 non-null   int64
 21  Churn              1142 non-null   object
dtypes: float64(1), int64(4), object(17)
memory usage: 196.4+ KB
```

```
#checking for null values
data.isnull().any()
```

```
Gender               False
Senior_Citizen       False
Partner              False
Dependents           False
Tenure               False
Phone_Service        False
Multiple_Lines       False
Internet_Service     False
Online_Security      False
Online_Backup        False
Device_Protection    False
Tech_Support         False
Streaming_TV         False
Streaming_Movies     False
Contract             False
Paper_less_Billing   False
Payment_Method       False
Monthly_Charges      False
Yearly_Charge        False
Admin_Tickets        False
Tech_Tickets         False
Churn                False
dtype: bool
```

```
data.isnull().sum()
```

```
Gender                0
Senior_Citizen        0
Partner               0
Dependents            0
Tenure                0
Phone_Service         0
Multiple_Lines        0
Internet_Service      0
Online_Security       0
Online_Backup         0
Device_Protection     0
Tech_Support          0
Streaming_TV          0
Streaming_Movies      0
Contract              0
Paper_less_Billing    0
Payment_Method        0
Monthly_Charges       0
Yearly_Charge         0
Admin_Tickets         0
Tech_Tickets          0
Churn                 0
```

```
[ ] #data.head(10)
    data
```

	Gender	Senior_Citizen	Partner	Dependents	Tenure	Phone_Service	Multiple_Lines	Internet_Service	Online_Security	Online_Backup	...	Streaming_TV	Stre
0	0	0	0	0	1	0	2	0	0	0	...	1	
1	1	0	1	0	71	1	1	1	1	1	...	1	
2	0	0	1	0	2	1	0	1	0	0	...	0	
3	0	0	0	0	1	1	0	0	0	0	...	1	
4	1	0	0	0	43	1	1	1	0	1	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1137	1	0	1	0	63	1	1	1	0	1	...	0	
1138	1	0	0	0	6	0	2	0	0	0	...	0	
1139	0	0	1	0	55	1	1	0	1	1	...	1	
1140	0	0	0	0	1	1	1	1	0	0	...	1	
1141	0	0	1	0	4	1	1	1	0	0	...	1	

1142 rows × 22 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1142 entries, 0 to 1141
Data columns (total 22 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             1142 non-null   int64
 1   Senior_Citizen     1142 non-null   int64
 2   Partner            1142 non-null   int64
 3   Dependents         1142 non-null   int64
 4   Tenure             1142 non-null   int64
 5   Phone_Service      1142 non-null   int64
 6   Multiple_Lines     1142 non-null   int64
 7   Internet_Service   1142 non-null   int64
 8   Online_Security    1142 non-null   int64
 9   Online_Backup      1142 non-null   int64
 10  Device_Protection  1142 non-null   int64
 11  Tech_Support       1142 non-null   int64
 12  Streaming_TV       1142 non-null   int64
 13  Streaming_Movies   1142 non-null   int64
 14  Contract           1142 non-null   int64
 15  Paper_less_Billing 1142 non-null   int64
 16  Payment_Method     1142 non-null   int64
 17  Monthly_Charges    1142 non-null   float64
 18  Yearly_Charge      1142 non-null   int64
 19  Admin_Tickets      1142 non-null   int64
 20  Tech_Tickets       1142 non-null   int64
 21  Churn              1142 non-null   int64
dtypes: float64(1), int64(21)
memory usage: 196.4 KB
```

```
[9] x= data.iloc[:,0:20].values
    y= data.iloc[:,21:].values
```

```
x
```

```
array([[0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.9650e+02, 4.7580e+03,
        0.0000e+00],
       [1.0000e+00, 0.0000e+00, 1.0000e+00, ..., 9.6350e+02, 1.1562e+04,
        0.0000e+00],
       [0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 9.5500e+02, 1.1460e+04,
        0.0000e+00],
       ...,
       [0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 6.0000e+02, 7.2000e+03,
        0.0000e+00],
       [0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 7.5750e+02, 9.0900e+03,
        0.0000e+00],
       [0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 7.4400e+02, 8.9280e+03,
        0.0000e+00]])
```

```
[11] y
```

```
array([[0],
       [1],
       [1],
       ...,
       [1],
       [0],
       [0]])
```

```
x_resample
```

```
array([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, ...,
        3.96500000e+02, 4.75800000e+03, 0.00000000e+00],
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, ...,
        9.63500000e+02, 1.15620000e+04, 0.00000000e+00],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        9.55000000e+02, 1.14600000e+04, 0.00000000e+00],
       ...,
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, ...,
        8.98183314e+02, 1.07781998e+04, 0.00000000e+00],
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, ...,
        9.51024538e+02, 1.14122945e+04, 0.00000000e+00],
       [4.05751036e-01, 5.94248964e-01, 0.00000000e+00, ...,
        7.98702876e+02, 9.58443451e+03, 0.00000000e+00]])
```

```
[16]
    y_resample
```

```
array([0, 1, 1, ..., 0, 0, 0])
```

```
[17] data.describe()
```

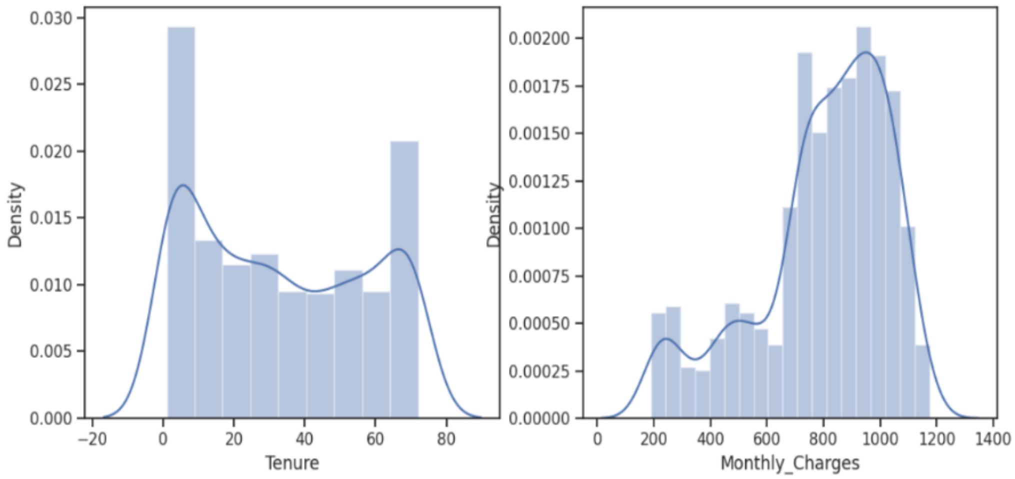
	Gender	Senior_Citizen	Partner	Dependents	Tenure	Phone_Service	Multiple_Lines	Internet_Service	Online_Security	Onli
count	1142.000000	1142.0	1142.000000	1142.000000	1142.000000	1142.000000	1142.000000	1142.000000	1142.000000	11
mean	0.497373	0.0	0.501751	0.079685	33.295972	0.908932	0.764448	0.818739	0.338004	
std	0.500212	0.0	0.500216	0.270923	24.188530	0.287832	0.602097	0.489575	0.561339	
min	0.000000	0.0	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.0	0.000000	0.000000	10.000000	1.000000	0.000000	1.000000	0.000000	
50%	0.000000	0.0	1.000000	0.000000	31.000000	1.000000	1.000000	1.000000	0.000000	
75%	1.000000	0.0	1.000000	0.000000	56.000000	1.000000	1.000000	1.000000	1.000000	
max	1.000000	0.0	1.000000	1.000000	72.000000	1.000000	2.000000	2.000000	2.000000	

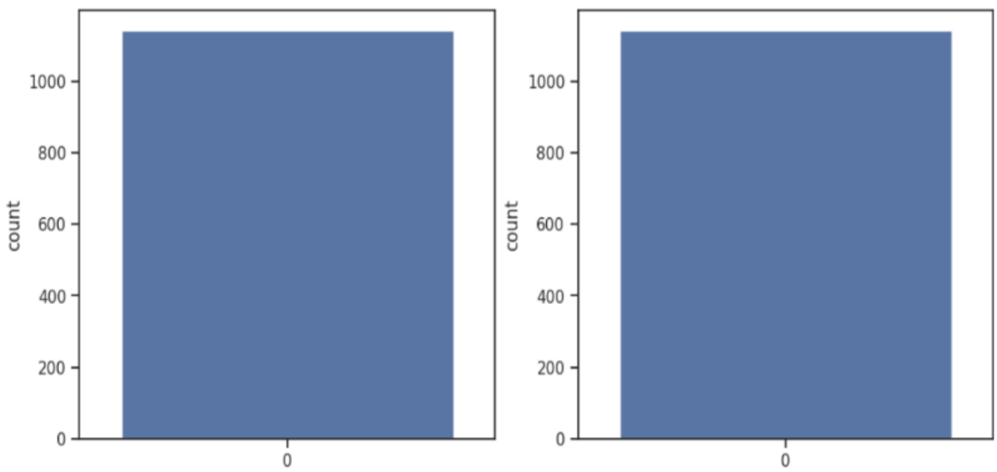
8 rows × 22 columns

Exploratary Data Analysis:

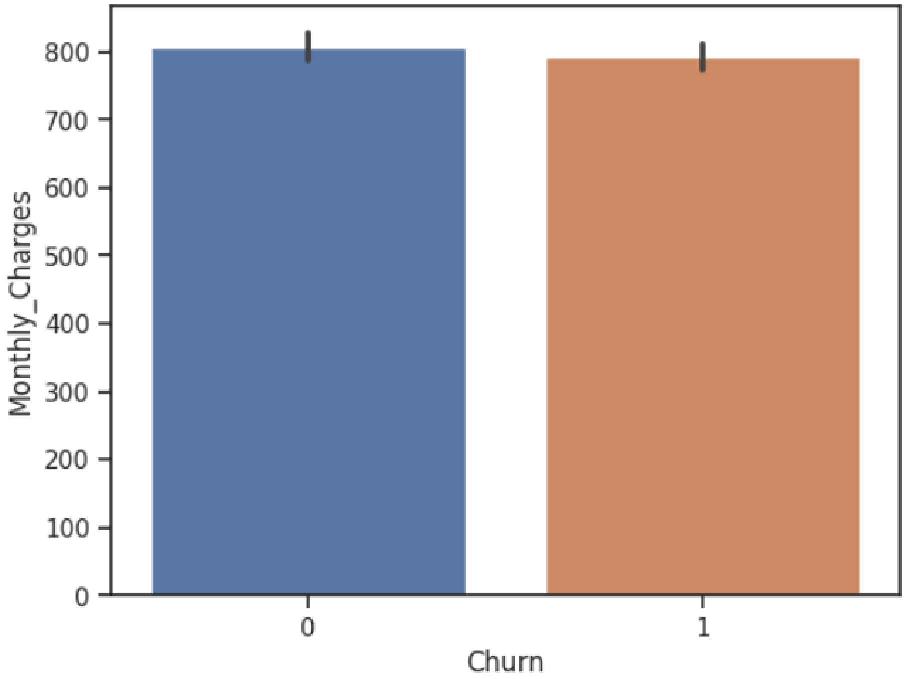
Visual Analysis

Univariate Analysis

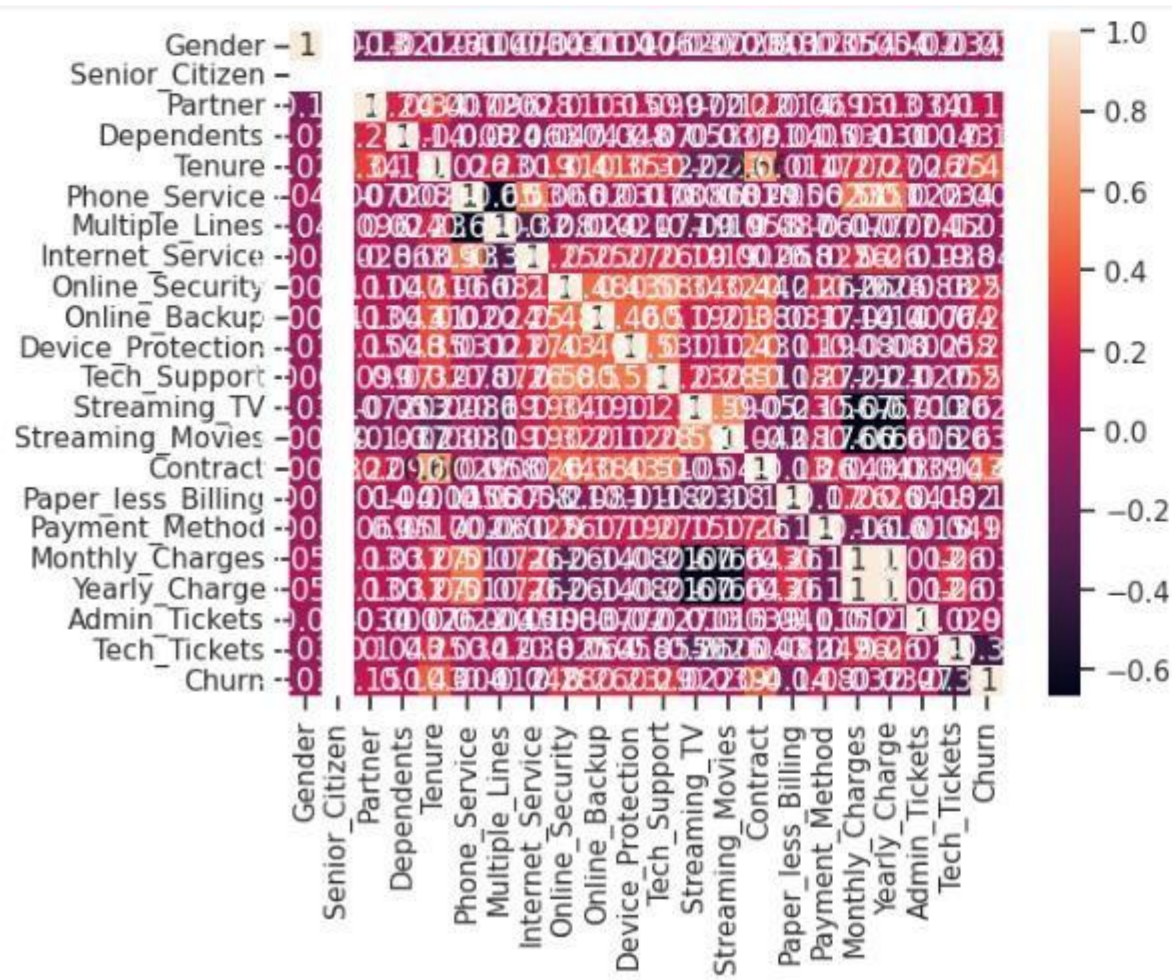


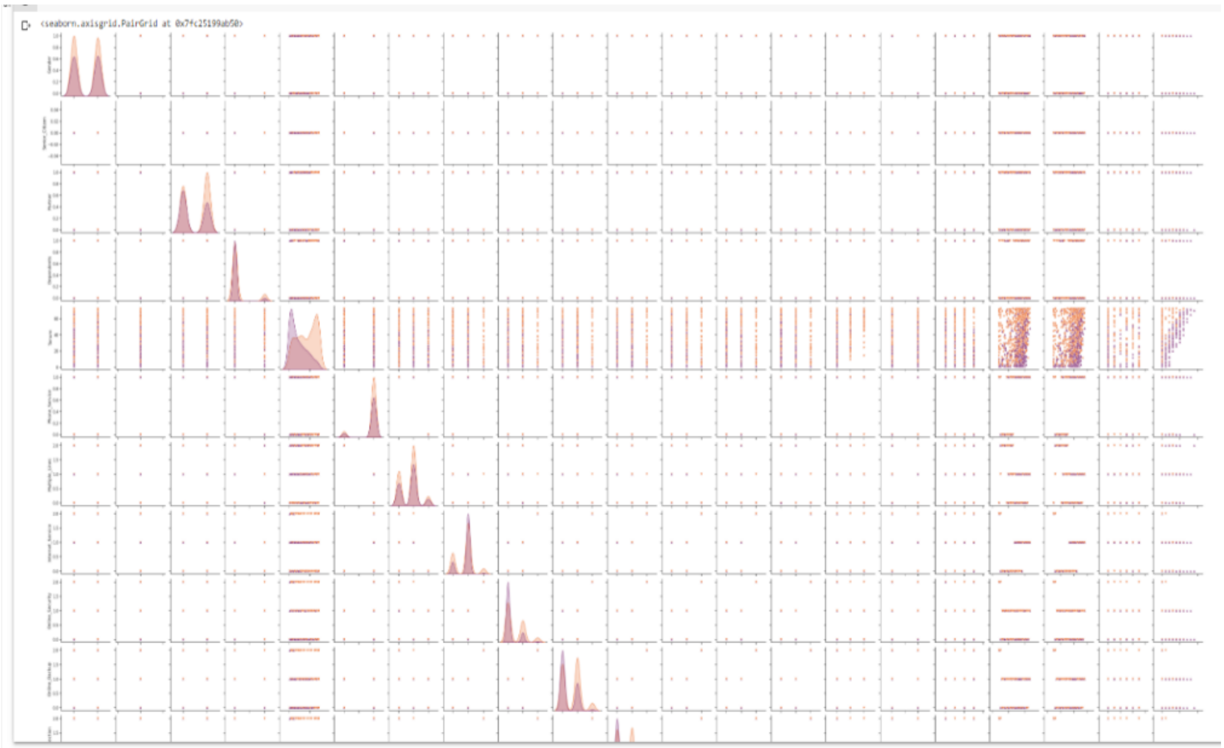


Bivariate Analysis



Multivariate Analysis





Model Building

Scaling The Data

```
[26] x_train.shape

(1065, 39)
```

```
y_train.shape

(1065,)
```

Training the model in multiple algorithms:

Logistic Regression Model

```
#printing the train accuracy and test accuracy respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.7305164319248826
0.7153558052434457
***Logistic Regression***
Confusion_Matrix
[[100  32]
 [ 44  91]]
Classification_Report
              precision    recall  f1-score   support

           0       0.69      0.76      0.72       132
           1       0.74      0.67      0.71       135

    accuracy                           0.72       267
   macro avg       0.72      0.72      0.72       267
weighted avg       0.72      0.72      0.71       267
```

Decision Tree Model

```
[30] #printing the train accuracy and test accuracy respectively
     decisionTree(x_train,x_test,y_train,y_test)
```

```
0.9990610328638497
0.6891385767790262
***Decision Tree***
Confusion_Matrix
[[90 42]
 [41 94]]
Classification_Report
              precision    recall  f1-score   support

           0       0.69      0.68      0.68       132
           1       0.69      0.70      0.69       135

    accuracy                           0.69       267
   macro avg       0.69      0.69      0.69       267
weighted avg       0.69      0.69      0.69       267
```

Random Forest Model

```
[32] #printing the train accuracy and test accuracy respectively
     RandomForest(x_train,x_test,y_train,y_test)
```

```
0.9784037558685446
0.7265917602996255
***Random Forest***
Confusion_Matrix
[[112  20]
 [ 53  82]]
Classification_Report
              precision    recall  f1-score   support

           0       0.68      0.85      0.75       132
           1       0.80      0.61      0.69       135

    accuracy                           0.73       267
   macro avg       0.74      0.73      0.72       267
weighted avg       0.74      0.73      0.72       267
```

KNN Model

```
[34] #printing the train accuracy and test accuracy respectively
     KNN(x_train,x_test,y_train,y_test)
```

```
0.7868544600938967
0.6966292134831461
***KNN***
Confusion_Matrix
[[104  28]
 [ 53  82]]
Classification_Report
              precision    recall  f1-score   support

           0       0.66      0.79      0.72       132
           1       0.75      0.61      0.67       135

    accuracy                           0.70       267
   macro avg       0.70      0.70      0.69       267
weighted avg       0.70      0.70      0.69       267
```

SVN Model

```
[36] #printing the train accuracy and test accuracy respectively
     svm(x_train,x_test,y_train,y_test)
```

```
0.7211267605633803
0.7228464419475655
***Support Vector Machine***
Confusion_Matrix
[[101  31]
 [ 43  92]]
Classification_Report
              precision    recall  f1-score   support

           0       0.70      0.77      0.73       132
           1       0.75      0.68      0.71       135

    accuracy                           0.72       267
   macro avg       0.72      0.72      0.72       267
weighted avg       0.72      0.72      0.72       267
```

ANN Model

```
array([[-0.68839327, -1.23588017,  3.2995671 , ..., -1.31695241,
        -1.31695241,  3.50355727],
       [-0.68839327,  0.82606646, -0.30577376, ...,  0.17796227,
         0.17796227, -0.40067279],
       [-0.68839327,  0.82606646, -0.30577376, ...,  0.09074248,
         0.09074248, -0.40067279],
       ...,
       [-0.41776866,  0.57006738, -0.30577376, ...,  0.67430052,
         0.67430052,  1.6510296 ],
       [-0.68839327, -1.23588017,  3.2995671 , ..., -0.9085471 ,
        -0.9085471 , -0.40067279],
       [-0.68839327,  0.82606646, -0.30577376, ...,  0.90755309,
         0.90755309, -0.40067279]])
```

```
array([1, 0, 0, ..., 0, 1, 0])
```

```
75/75 [==============================] - 0s 3ms/step - loss: 0.1678 - accuracy: 0.9342 - val_loss: 1.1180 - val_accuracy: 0.6875
Epoch 189/200
75/75 [==============================] - 0s 3ms/step - loss: 0.1641 - accuracy: 0.9356 - val_loss: 1.1015 - val_accuracy: 0.6969
Epoch 190/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1713 - accuracy: 0.9383 - val_loss: 1.1342 - val_accuracy: 0.6812
Epoch 191/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1719 - accuracy: 0.9315 - val_loss: 1.1843 - val_accuracy: 0.6906
Epoch 192/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1776 - accuracy: 0.9248 - val_loss: 1.2114 - val_accuracy: 0.6969
Epoch 193/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1767 - accuracy: 0.9356 - val_loss: 1.1321 - val_accuracy: 0.6875
Epoch 194/200
75/75 [==============================] - 0s 5ms/step - loss: 0.1683 - accuracy: 0.9329 - val_loss: 1.1887 - val_accuracy: 0.6938
Epoch 195/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1647 - accuracy: 0.9409 - val_loss: 1.1904 - val_accuracy: 0.6938
Epoch 196/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1732 - accuracy: 0.9329 - val_loss: 1.1847 - val_accuracy: 0.6844
Epoch 197/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1575 - accuracy: 0.9396 - val_loss: 1.1442 - val_accuracy: 0.6781
Epoch 198/200
75/75 [==============================] - 0s 4ms/step - loss: 0.1612 - accuracy: 0.9423 - val_loss: 1.1283 - val_accuracy: 0.6844
Epoch 199/200
75/75 [==============================] - 0s 3ms/step - loss: 0.1769 - accuracy: 0.9315 - val_loss: 1.2077 - val_accuracy: 0.6875
Epoch 200/200
75/75 [==============================] - 0s 3ms/step - loss: 0.1564 - accuracy: 0.9383 - val_loss: 1.2231 - val_accuracy: 0.6781
```

```
array([[False],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [False],
```

Testing The Model

```
Predicting on random input
output is: [1]
```

```
Predicting on random input
output is: [1]
```

```
Predicting on random input
output is: [1]
```

```
Predicting on random input
output is: [1]
```

```
Predicting on random input
output is: [1]
```

```
Predicting on random input
[[0.9573193]]
output is:  [[ True]]
```

```
Predicting on random input
[[0.9573193]]
output is:  [[ True]]
```

Hyperparameter Tuning

```
0.7427230046948357
0.7453183520599251
***Logistic Regression***
Confusion_Matrix
[[102  30]
 [ 38  97]]
Classification_Report
              precision    recall  f1-score   support

           0       0.73      0.77      0.75       132
           1       0.76      0.72      0.74       135

    accuracy                           0.75       267
   macro avg       0.75      0.75      0.75       267
weighted avg       0.75      0.75      0.75       267
```

```
______________________________________________________
0.9990610328638497
0.6966292134831461
***Decision Tree***
Confusion_Matrix
[[101  31]
 [ 50  85]]
Classification_Report
              precision    recall  f1-score   support

           0       0.67      0.77      0.71       132
           1       0.73      0.63      0.68       135

    accuracy                           0.70       267
   macro avg       0.70      0.70      0.70       267
weighted avg       0.70      0.70      0.70       267
```

```
______________________________________________________
0.9821596244131455
0.7228464419475655
***Random Forest***
Confusion_Matrix
[[120  12]
 [ 62  73]]
Classification_Report
              precision    recall  f1-score   support

           0       0.66      0.91      0.76       132
           1       0.86      0.54      0.66       135

    accuracy                           0.72       267
   macro avg       0.76      0.72      0.71       267
weighted avg       0.76      0.72      0.71       267
```

```
______________________________________________________
0.651685393258427
**ANN Model**
Confusion_Matrix
[[88 44]
 [49 86]]
Classification Report
              precision    recall  f1-score   support

           0       0.64      0.67      0.65       132
           1       0.66      0.64      0.65       135

    accuracy                           0.65       267
   macro avg       0.65      0.65      0.65       267
weighted avg       0.65      0.65      0.65       267
```

Comparing model accuracy before & after applying hyperparameter tuning

```
0.9784037558685446
0.7265917602996255
***Random Forest after Hyperparameter tuning***
Confusion_Matrix
[[112  20]
 [ 53  82]]
Classification Report
              precision    recall  f1-score   support

           0       0.68      0.85      0.75       132
           1       0.80      0.61      0.69       135

    accuracy                           0.73       267
   macro avg       0.74      0.73      0.72       267
weighted avg       0.74      0.73      0.72       267

Predicting on random input
output is:  [1]
```

Integrate With Web Frame Work

Building HTML Pages

```
<!DOCTYPE html>

<html>

<head>

<title>Prediction Form</title>

</head>

<body background="wo.jpg" style="background-repeat:no-repeat; background-size:100%
100%" text='black'>

<h1>

<b>

<i>

<font size=15>

<center>Prediction Form</center>

</font>

</i>

</b>

</h1>

<div style="background-color:white">

<hr>

<hr></div>

<h2> Enter the details to check whether Loan is eligible ot not!</h2>

<h4>

<form action="{{url_for('predict')}}" method="post">

<center>

<table>

<tr>
```

```
<td>Gender:

<input type="radio" name="gender" id="male">

<label for="male">Male</label>

<input type="radio" name="gender" id="female">

<label for="female">Female</label><br>

<tr>

<td>Senior_Citizen<td>:   <input type='text' name='Senior_Citizen' placeholder='Enter 1 for no 0 for yes' required='required'/><br>

</tr>

<tr>

<td>Dependents<td>:   <input type='text' name='Dependents' placeholder='Enter 0 for no 1 for yes' required='required' /><br>

</tr>

<tr>

<td>Partner<td>:   <input type='text' name='Partner' placeholder='Enter 0 for no 1 for yes' required='required' /><br>

</tr>

<tr>

<td>Tenure<td>:   <input type='text' name='Tenure' placeholder='Enter 0 for 1 for 71' required='required' /><br>

</tr>

<tr>

<td>Phone_Service<td>:   <input type='text' name='Phone_Service' placeholder='Enter 0 for no 1 for yes' required='required' /><br>

</tr>

<tr>

<td>Multiple_Lines<td>:   <input type='text' name='Multiple_Lines' placeholder='Enter 0 for no 1 for yes 2 for No phone service' required='required' /><br>

</tr>
```

```
<tr>

<td>Online_Security<td>:   <input type='text' name='Online_Security'
placeholder='Enter 0 for no 1 for yes 2 for No internet service' required='required'/><br>

</tr>

<tr>

<td>Online_Backup<td>:   <input type='text' name='Online_Backup'
placeholder=' Enter 0 for no 1 for yes 2 for No internet service ' required='required' /><br>

</tr>

<tr>

<td>Streaming_TV<td>:   <input type='text' name='Streaming_TV'
placeholder='Enter 0 for no 1 for yes 2 for No internet service  ' required='required' /><br>

</tr>

<tr>

<td>Streaming_Movies<td>:   <input type='text'
name='Streaming_Movies' placeholder=' Enter 0 for no 1 for yes 2 for No internet service '
required='required'/><br>

</tr>

<tr>

<td>Paper_less_Billing<td>:   <input type='text' name='Paper_less_Billing'
placeholder=' Enter 0 for no 1 for yes ' required='required' /><br>

</tr>

<tr>

<td>Churn<td>:   <input type='text' name='Churn' placeholder=' Enter 0
for no 1 for yes ' required='required' /><br>

</tr>

<tr>

<td>Online Backup<td>:   <input type='text' name='Online Backup'
placeholder=' Enter 0 for yes 1 for no ' required='required'/><br>

</tr>

<tr>
```

```
<td>Contract<td>:   <input type='text' name='Contract' placeholder=' Enter 0 for Month-to-month 1 for One year 2 for Two year ' required='required' /><br>

</tr>

<tr>

<td>Internet_Service<td>:   <input type='text' name='Internet_Service' placeholder=' Enter 0 for DSL 1 for Fiber optic 2 for No' required='required' /><br>

</tr>

<tr>

<td>Payment_Method<td>:   <input type='text' name='Payment_Method' placeholder=' Enter 0 for Electronic check 1 for Credit card (automatic) 2 for Bank transfer (automatic) 3 for Mailed check ' required='required' /><br>

</tr>

<tr>

<td>Device_Protection<td>:   <input type='text' name='Device_Protection' placeholder=' Enter 0 for no 1 for yes 2 for No internet service' required='required' /><br>

</tr>

<tr>

<td>Tech_Support<td>:   <input type='text' name='Tech_Support' placeholder=' Enter 0 for no 1 for yes 2 for No internet service' required='required' /><br>

</tr>

<tr>

<td><br><br><center>               <button type="submit" class="btn btn-primary btn-block btn-large">Predict</center></button>

</tr>

</form>

</h4>

</table>

</center>

<h2>
```

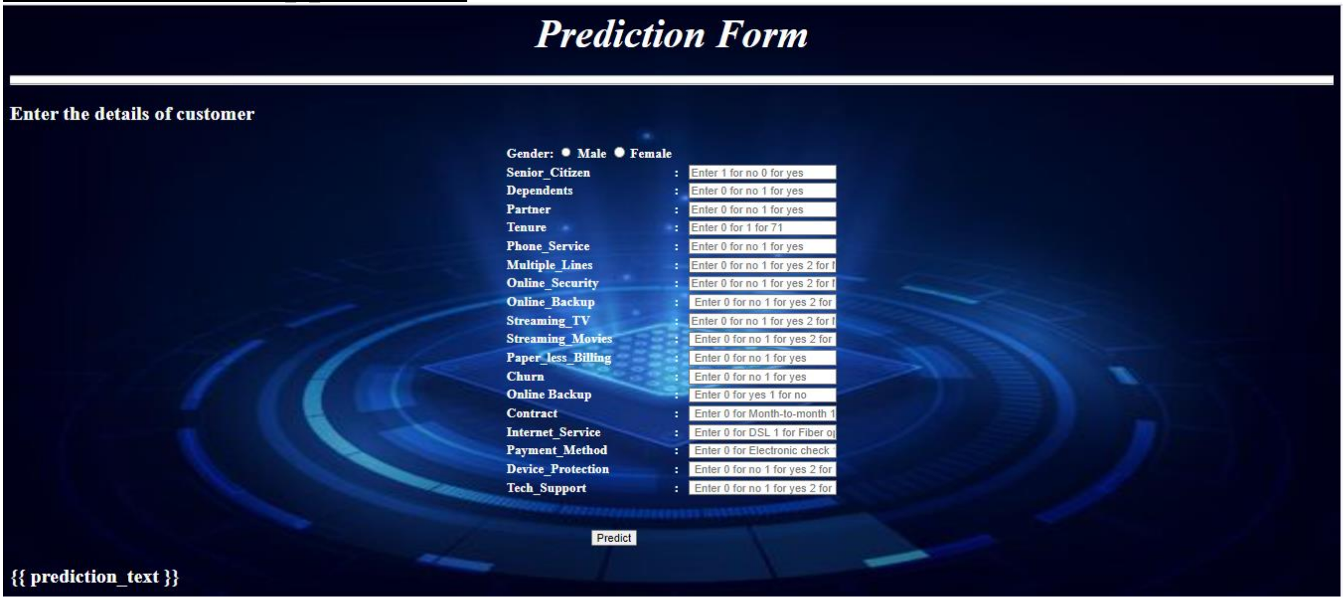
```
<b>
{{ prediction_text }}
</b>
</h2>
</body>
</html>
```

Building Python Code

```
import flask
from flask import Flask, render_template, request
import pickle
import numpy as np
import sklearn
from flask_ngrok import run_with_ngrok
import warnings

warnings.filterwarnings('ignore')

app = Flask(_name_)
run_with_ngrok(app)

model = pickle.load(open('ssam.pkl', 'wb'))



@app.route('/', methods=['GET'])
def home():
return render_template('rename.html')
```

```
@app.route('/', methods=['GET', "POST"])

def predict():

input_values = [float(x) for x in request.form.values()]

inp_features = [input_values]

print(inp_features )

prediction = model.predict(inp_features)

if prediction == 1:

return render_template('index.html', prediction_text='Eligible to loan, Loan will be sanctioned')

else:

return render_template('index.html', prediction_text='Not eligible to loan')


app.run()
```

Run the Web Application



3. ADVANTAGES & DISADVANTAGES

Advantages:

- By using machine learning algorithms to analyze customer data, telecom companies can identify customers who are likely to churn and take proactive measures to retain them. This can lead to improved customer retention rates and increased revenue for the company.
- Intelligent customer retention can also help improve the overall customer experience by identifying areas where customers may be dissatisfied and addressing these issues before they become reasons for churn.
- It is generally more cost-effective to retain existing customers than to acquire new ones. By using machine learning to predict customer churn and taking proactive measures to retain these customers, telecom companies can save money on customer acquisition costs.
- Telecom companies that are able to effectively use machine learning for customer retention may have a competitive advantage over those that do not. By retaining more customers and providing a better customer experience, these companies may be able to differentiate themselves in a crowded market.
- Intelligent customer retention allows telecom companies to make data-driven decisions based on customer behavior and preferences. This can lead to more effective marketing strategies, product development, and customer service initiatives.

Disadvantage:

- The use of machine learning algorithms to analyze customer data raises concerns about data privacy and security. Telecom companies must ensure that they are following all applicable regulations and taking steps to protect customer data.
- Machine learning algorithms may be biased if the data used to train them is not representative of the entire customer population. This could lead to incorrect predictions and ineffective retention strategies.
- While machine learning can be a valuable tool for customer retention, it should not be the only strategy used. Telecom companies must also focus on building strong relationships with customers through effective communication and customer service.
- Implementing machine learning algorithms for customer retention can be expensive, and smaller telecom companies may not have the resources to do so.
- Machine learning algorithms can be complex and difficult to interpret, which may make it challenging for telecom companies to understand why certain customers are likely to churn and how best to retain them.

4. APPLICATION

- Telecom customer churn is a common problem faced by service providers. In order to reduce churn and retain customers, telecom companies can leverage machine learning to enhance their prediction capabilities.
- One approach to intelligent customer retention is to use machine learning algorithms to analyze large amounts of customer data, such as call logs, billing information, and usage patterns. By analyzing this data, algorithms can identify patterns that are predictive of customer churn, such as a decrease in usage, frequent complaints, or missed payments.
- Once these patterns are identified, telecom companies can use this information to proactively engage with customers who are at risk of churning. This could include targeted marketing campaigns, personalized offers, or proactive customer service outreach.
- Machine learning algorithms can also be used to predict the likelihood of churn for individual customers. This information can be used to prioritize retention efforts and allocate resources more effectively. For example, a customer who is predicted to have a high likelihood of churn may be offered a more personalized retention offer than a customer who is predicted to have a lower likelihood of churn.
- Overall, by leveraging machine learning to enhance their customer retention capabilities, telecom companies can reduce churn and improve customer satisfaction.
- Customer retention is a critical factor in the telecommunications industry, where companies face fierce competition and high customer churn rates. Machine learning can help telecom companies enhance their customer retention strategies by predicting customer churn and enabling them to take proactive measures to prevent it.
- To develop a machine learning model for predicting customer churn, telecom companies can use historical customer data such as demographic information, usage patterns, billing history, and customer service interactions.
- The algorithm can then generate a risk score for each customer, which can be used to prioritize retention efforts. For example, customers with high risk scores could receive personalized retention offers, such as discounts or special promotions, to encourage them to stay with the company.
- Another approach is to use machine learning to identify patterns that are associated with customer churn. This could involve analyzing customer interactions with the company's website or mobile app, as well as their usage patterns and behavior. The algorithm can then generate insights into the factors that are driving churn, allowing the company to take targeted action to improve retention.

5. CONCLUSION

- Telecom companies often use customer churn as a key business metrics to predict the number of customers that will leave a telecom service provider. A machine learning model can be used to identity the probable churn customers and then makes the necessary business decisions.
- We have developed the machine leaning project using python programming language an the reports are shown the above.
-

6. FUTURE SCOPE

- Intelligent customer retention is a concept that refers to using advanced technologies like machine learning to predict customer churn and take proactive measures to retain them. In the telecom industry, customer churn is a major concern, and companies are always looking for ways to reduce it. By using machine learning algorithms, telecom companies can predict customer churn with higher accuracy and take steps to prevent it.
- Telecom companies can integrate IoT devices with their machine learning algorithms to improve customer retention. IoT devices like smart homes, wearables, and connected cars can provide valuable data that can be analyzed to predict customer behavior and preferences.
- Machine learning algorithms can be used to analyze customer data and create personalized engagement strategies. Telecom companies can use this information to tailor their marketing and communication efforts to each customer, increasing the chances of retaining them.
- Machine learning algorithms can be used to analyze customer service interactions and identify patterns in customer behavior that indicate dissatisfaction. This information can then be used to proactively address issues and prevent customer churn.
- Telecom companies can use machine learning algorithms to predict when equipment and devices are likely to fail. This can help companies proactively replace or repair equipment before it fails, reducing the likelihood of service interruptions and customer churn.
- Machine learning algorithms can be used to analyze customer data and identify patterns of fraudulent behavior. This can help telecom companies proactively detect and prevent fraud, reducing the risk of financial loss and increasing customer trust.
- Overall, machine learning has the potential to revolutionize the telecom industry by improving customer retention, reducing churn, and enhancing the overall customer experience. By investing in advanced technologies like machine learning, telecom companies can stay ahead of the competition and continue to provide high-quality services to their customers.

7. APPENDIX

Source Code:

Importing the 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, confusion_matrix, f1_score
```

Read the dataset

```
data = pd.read_csv("/content/Telecom Churn Rate Dataset.csv")
```

data

Handling missing values:

data.info()

data.isnull().any()

data.isnull().sum()

Handling Categorial Values

Label Encoding.

data['Gender']=data['Gender'].replace({'Male': 0,'Female':1})

data['Senior_Citizen']=data['Senior_Citizen'].replace({'Yes': 0,'No':1})

data['Dependents']=data['Dependents'].replace({'No': 0,'Yes':1})

data['Partner']=data['Partner'].replace({'No': 0,'Yes':1})

data['Tenure']=data['Tenure'].replace({'1': 0,'71':1})

data['Phone_Service']=data['Phone_Service'].replace({'No': 0,'Yes':1})

data['Multiple_Lines']=data['Multiple_Lines'].replace({'No': 0,'Yes':1,'No phone service':2})

data['Online_Security']=data['Online_Security'].replace({'No': 0,'Yes':1,'No internet service':2})

data['Online_Backup']=data['Online_Backup'].replace({'No': 0,'Yes':1,'No internet service':2})

data['Streaming_TV']=data['Streaming_TV'].replace({'Yes': 0,'No':1,'No internet service':2})

data['Streaming_Movies']=data['Streaming_Movies'].replace({'Yes': 0,'No':1,'No internet service':2})

data['Paper_less_Billing']=data['Paper_less_Billing'].replace({'No': 0,'Yes':1})

data['Churn']=data['Churn'].replace({'Yes': 0,'No':1})

data['Contract']=data['Contract'].replace({'Month-to-month': 0,'Two year':2,'One year':1})

data['Internet_Service']=data['Internet_Service'].replace({'DSL': 0,'Fiber optic':1,'No':2}) data['Payment_Method']=data['Payment_Method'].replace({'Electronic check': 0,'Credit card (automatic)':1,'Bank transfer (automatic)':2,'Mailed check':3})

data['Device_Protection']=data['Device_Protection'].replace({'No': 0,'Yes':1,'No internet service':2})

data['Tech_Support']=data['Tech_Support'].replace({'No': 0,'Yes':1,'No internet service':2

Data after label encoding

data.head()

data.info()

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

y= data.iloc[:,21:].values

x

y

One Encoding

from sklearn.preprocessing import OneHotEncoder

one = OneHotEncoder()

a= one.fit_transform(x[:,6:7]).toarray()

b= one.fit_transform(x[:,7:8]).toarray()

c= one.fit_transform(x[:,8:9]).toarray()

d= one.fit_transform(x[:,9:10]).toarray()

e= one.fit_transform(x[:,10:11]).toarray()

f= one.fit_transform(x[:,11:12]).toarray()

g= one.fit_transform(x[:,12:13]).toarray()

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

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

j= one.fit_transform(x[:,15:16]).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)

Handling Imbalance Data

```
from imblearn.over_sampling import SMOTE

smt = SMOTE()

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

x_resample

y_resample
```

EXPLORATORY DATA ANALYSIS

```
data.descirbe()
```

Visual analysis

Univariate analysis

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

plt.subplot(1,2,1)

sns.distplot(data["Tenure"])

plt.subplot(1,2,2)

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

Countplot :-

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

plt.subplot(1,2,1)

sns.countplot(data["Gender"])

plt.subplot(1,2,2)

sns.countplot(data["Dependents"])
```

Bivariate analysis

```
sns.barplot(x="Churn",y="Monthly_Charges",data=data)
```

Multivariate analysis

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

sns.pairplot(data=data, markers=["^","v"],  hue='Churn',palette="inferno")
```

Splitting data into train and test

```
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)

y_train.shape

x_train.shape
```

PERFORMANCE TESTING

Training the model in multiple algorithmsLogistic

Regression Model

```
#importing and building the LogisticRegression 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)

Decision tree model

#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)

Random forest model

#importing and buliding the random forest model

def RandomForest(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("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

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

KNN model

```
#importing and buliding 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)
```

SVM model

```
#importing and buliding the SVM model
def svm(x_train,x_test,y_train,y_test):
svm = SVC(kernel = 'linear',gamma = 'scale', shrinking = False,)
svm.fit(x_train,y_train)
y_svm_tr = svm.predict(x_train)
print(accuracy_score(y_svm_tr,y_train))
yPred_svm = svm.predict(x_test)
print(accuracy_score(yPred_svm,y_test))
print("**Support Vector Machine**")
print("Confusion_Matrix")
print(confusion_matrix(y_test,yPred_svm))
print("Classification_Report")
print(classification_report(y_test,yPred_svm))

#printing the train accuracy and test accuracy respectively
"""svm(x_train,x_test,y_train,y_test)"""
```

ANN model

```
import tensorflow as tf
from tensorflow.python import keras
from keras import layers
from keras.layers import Activation,Dense

classifier = keras.Sequential()
classifier.add(Dense(units=30, activation='relu', input_dim=40))
classifier.add(Dense (units=30, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))
```

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

classifier.add(Dense(units=30, activation= 'relu', input_dim=40))

classifier.add(Dense(units=30, activation= 'relu'))

classifier.add(Dense(units=1, activation= 'sigmoid'))

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

x_train

y_train

# Fitting the AVM to the Training set

model_history = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.3, 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 the model

```
lr = LogisticRegression(random_state=0)

lr.fit(x_train,y_train)

print("Predicting on random input")
```

```
lr_pred_own =
lr.predict(sc.transform([[0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,
1,0,3245,4567]]))
print("output is:",lr_pred_own)
dtc = DecisionTreeClassifier(criterion="entropy", random_state=0)
dtc.fit(x_train,y_train)
print("Predicting on random input")
dtc_pred_own =
dtc.predict(sc.transform([[0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456
,1,0,3245,4567]]))
print("output is:",dtc_pred_own)
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([[0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,
1,0,3245,4567]]))
print("output is:",rf_pred_own)
from sklearn.svm import SVC # "Support vector classifier"
svm = SVC(kernel='linear', random_state=0)
svm.fit(x_train, y_train)
#svm = RandomForestClassifier(criterion="entropy",n_estimators=10, random_state=0)
#svm.fit(x_train,y_train)
print("Predicting on random input")
svm_pred_own =
svm.predict(sc.transform([[0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,45
6,1,0,3245,4567]]))
print("output is:",svm_pred_own)
```

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

knn.fit(x_train,y_train)

print("Predicting on random input")

knn_pred_own = rf.predict(sc.transform([[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)
```

ANFor N

```
print("Predicting on random input")

ann_pred_own = classifier.predict(sc.transform([[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)
```

TUNNING THE MODEL

Compare the model

```
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)

KNN(x_train,x_test,y_train,y_test)

print('_'*100)

#svm(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))
```

Comparing model accuracy before & afte applying hyperparameter tuning

```
from sklearn import model_selection
models= ['dt',DecisionTreeClassifier(),
'rf',RandomForestClassifier(),'svm',SVC(),'knn',KNeighborsClassifier()]
rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
rf.fit(x_train,y_train)
y_rf = rf.predict(x_train)
print(accuracy_score(y_rf,y_train))
yPred_rfcv = rf.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 =
rf.predict(sc.transform([[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)
```