Telecom Customer Churn Prediction Using Artificial Neural Network (ANN)

Group Members:

Member 1: Name: Rupali Patole Roll No.: MCS21005 Email Id: mcs21005@iiitl.ac.in

Member 2: Name: Aman Mehta Roll no: MCS21011 Email Id: mcs21011@iiitl.ac.in

Member 3: Name: Shreya Goswami Roll No.: MCS21023 Email Id: mcs21023@iiitl.ac.in

---

Introduction:

Customer churn prediction is to measure why customers are leaving a business. It can be applied in different business sectors like banking, retail, telecom, etc. For this project we will be implementing customer churn prediction for the telecom business. We will build a deep learning model to predict the churn and use precision, recall, f1-score to measure performance of our model.

Dataset: https://www.kaggle.com/blastchar/telco-customer-churn

Algorithm: Building ANN using tensorflow / keras

Steps:

1. Data Loading
2. Data analysis
3. Data cleaning
4. Data splitting (Training set and Test set)
5. ANN Model Building
6. Evaluation Metrics (Accuracy, F1 score, Precision, Recall)
7. Hyperparameter Tuning
8. Performance Metrics (ROC Analysis)
9. Conclusion
10. Future Scope (Offers and Improving Churn Rate)

Code:

Imports

```
import numpy as np

import pandas as pd

import seaborn as sns

import plotly.express as px

import matplotlib.pyplot as plt

# %matplotlib inline
```

from sklearn import metrics

from sklearn.model_selection import train_test_split

import warnings

warnings.filterwarnings('ignore')

from keras.layers import Activation, Dense, Dropout

import tensorflow as tf

from keras.models import Sequential

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import plot_roc_curve, accuracy_score

Data Loading

df = pd.read_csv("customer_churn.csv")

df.sample(5)

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechS
6226	2097-YVPKN	Male	0	No	No	65	Yes	Yes	No	No internet service	...	No internet service	No i
4607	2853-CWQFQ	Male	0	No	Yes	1	Yes	No	DSL	No	...	No	
1010	0929-HYQEW	Male	0	No	No	3	Yes	No	DSL	Yes	...	No	
4932	5566-SOEZD	Male	0	Yes	Yes	27	Yes	No	Fiber optic	Yes	...	No	
6399	2101-RANCD	Female	0	No	No	55	Yes	Yes	Fiber optic	No	...	No	

5 rows × 21 columns

Data analysis and Data cleaning

Dropping customerID column as it is of no use

df.drop('customerID',axis='columns',inplace=True)

df.dtypes

```
Out[4]: gender               object
        SeniorCitizen         int64
        Partner              object
        Dependents           object
        tenure                int64
        PhoneService         object
        MultipleLines        object
        InternetService      object
        OnlineSecurity       object
        OnlineBackup         object
        DeviceProtection     object
        TechSupport          object
        StreamingTV          object
        StreamingMovies      object
        Contract             object
        PaperlessBilling     object
        PaymentMethod        object
        MonthlyCharges      float64
        TotalCharges         object
        Churn                object
        dtype: object
```

df.TotalCharges.values

```
Out[5]: array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
              dtype=object)
```

Finding out rows that are blank strings

pd.to_numeric(df.TotalCharges,errors='coerce').isnull()

```
Out[6]: 0       False
        1       False
        2       False
        3       False
        4       False
                ...
        7038    False
        7039    False
        7040    False
        7041    False
        7042    False
        Name: TotalCharges, Length: 7043, dtype: bool
```

df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]

Out[7]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSu
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	No	Yes	
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No in s
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Yes	Yes	
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No in s
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Yes	
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No in s
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No in s
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No in s
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No in s
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	Yes	
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	No	

df.shape

```
Out[8]: (7043, 20)
```

df.iloc[488].TotalCharges

```
Out[9]: ' '
```

df[df.TotalCharges!=' '].shape

```
Out[10]: (7032, 20)
```

Remove rows with space in TotalCharges

df1 = df[df.TotalCharges!=' ']

df1.shape

```
Out[11]: (7032, 20)
```

df1.dtypes

```
Out[12]: gender              object
         SeniorCitizen        int64
         Partner             object
         Dependents          object
         tenure               int64
         PhoneService        object
         MultipleLines       object
         InternetService     object
         OnlineSecurity      object
         OnlineBackup        object
         DeviceProtection    object
         TechSupport         object
         StreamingTV         object
         StreamingMovies     object
         Contract            object
         PaperlessBilling    object
         PaymentMethod       object
         MonthlyCharges     float64
         TotalCharges        object
         Churn               object
         dtype: object
```

df1.TotalCharges = pd.to_numeric(df1.TotalCharges)

df1.TotalCharges.values

```
Out[14]: array([  29.85, 1889.5 ,  108.15, ...,  346.45,  306.6 , 6844.5 ])
```

df1[df1.Churn=='No']

Out[15]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSu
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	
6	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	No	
7	Female	0	No	No	10	No	No phone service	DSL	Yes	No	No	
...	...	...	...	...	...	...	...	...	...	...	...	
7037	Female	0	No	No	72	Yes	No	No	No internet service	No internet service	No internet service	No in s
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No	Yes	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	Yes	Yes	
7040	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	No	No	
7042	Male	0	No	No	66	Yes	No	Fiber optic	Yes	No	Yes	

5163 rows × 20 columns

df1.Churn.value_counts()

Out[16]:
```
No     5163
Yes    1869
Name: Churn, dtype: int64
```

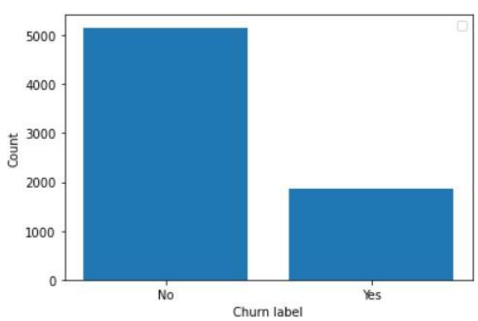
plt.bar(x = df1['Churn'].unique(), height = df1.Churn.value_counts())

plt.legend()

plt.xlabel("Churn label")

plt.ylabel("Count")

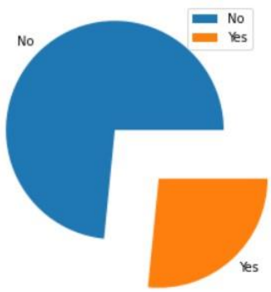
plt.show()



plt.pie(df1.Churn.value_counts(), labels = df1['Churn'].unique(), explode = [0.1, 0.5])

plt.legend()

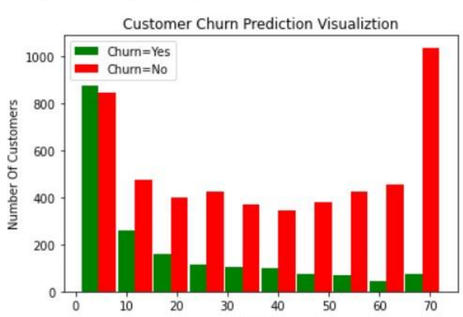
plt.show()



Data Visualization

```
tenure_churn_no = df1[df1.Churn=='No'].tenure

tenure_churn_yes = df1[df1.Churn=='Yes'].tenure


plt.xlabel("tenure")

plt.ylabel("Number Of Customers")

plt.title("Customer Churn Prediction Visualiztion")


blood_sugar_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77, 82, 129]

blood_sugar_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120, 112, 100]


plt.hist([tenure_churn_yes, tenure_churn_no], rwidth=0.95,
color=['green','red'],label=['Churn=Yes','Churn=No'])

plt.legend()
```

```
Out[19]: <matplotlib.legend.Legend at 0x212e377e5e0>
```



mc_churn_no = df1[df1.Churn=='No'].MonthlyCharges

mc_churn_yes = df1[df1.Churn=='Yes'].MonthlyCharges

plt.xlabel("Monthly Charges")

plt.ylabel("Number Of Customers")

plt.title("Customer Churn Prediction Visualiztion")

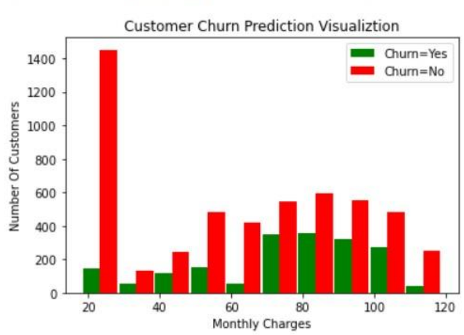
blood_sugar_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77, 82, 129]

blood_sugar_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120, 112, 100]

plt.hist([mc_churn_yes, mc_churn_no], rwidth=0.95, color=['green','red'],label=['Churn=Yes','Churn=No'])

plt.legend()

```
Out[20]: <matplotlib.legend.Legend at 0x2128a616fa0>
```



df1.groupby(['gender', 'Partner', 'Dependents', 'Churn']).size()

```
Out[21]: gender  Partner  Dependents  Churn
         Female  No       No          No       1068
                                      Yes       587
                          Yes         No        112
                                      Yes        33
                 Yes      No          No        618
                                      Yes       187
                          Yes         No        746
                                      Yes       132
         Male    No       No          No       1089
                                      Yes       536
                          Yes         No        170
                                      Yes        44
                 Yes      No          No        615
                                      Yes       233
                          Yes         No        745
                                      Yes       117
         dtype: int64
```

df1.groupby(['gender', 'Partner', 'Dependents', 'Churn']).size().reset_index(name='Count')

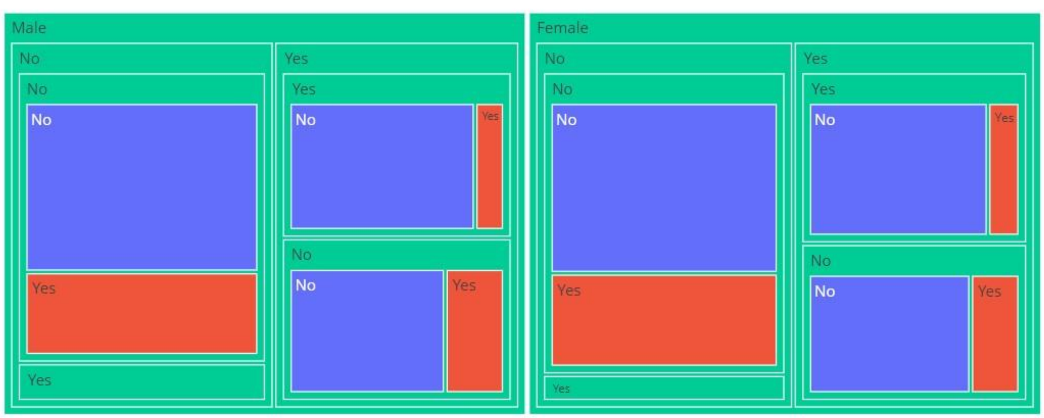
Out[22]:

	gender	Partner	Dependents	Churn	Count
0	Female	No	No	No	1068
1	Female	No	No	Yes	587
2	Female	No	Yes	No	112
3	Female	No	Yes	Yes	33
4	Female	Yes	No	No	618
5	Female	Yes	No	Yes	187
6	Female	Yes	Yes	No	746
7	Female	Yes	Yes	Yes	132
8	Male	No	No	No	1089
9	Male	No	No	Yes	536
10	Male	No	Yes	No	170
11	Male	No	Yes	Yes	44
12	Male	Yes	No	No	615
13	Male	Yes	No	Yes	233
14	Male	Yes	Yes	No	745
15	Male	Yes	Yes	Yes	117

1. How gender, partner and dependents are related to churn?

```
fig = px.treemap(df1.groupby(['gender', 'Partner', 'Dependents', 'Churn']).size().reset_index(name='Count'),
            path=['gender', 'Partner', 'Dependents', 'Churn'], values='Count', color='Churn',
            title='1. How gender, partner and dependents are related to churn?')

fig.show()
```

1. How gender, partner and dependents are related to churn?



Observation: Whether male or female, if they do not have partner or dependents, they are more likely to churn! 2. Does tenure has any impact on churn?

df1.groupby(['tenure', 'Churn']).size().reset_index(name='count')

Out[24]:

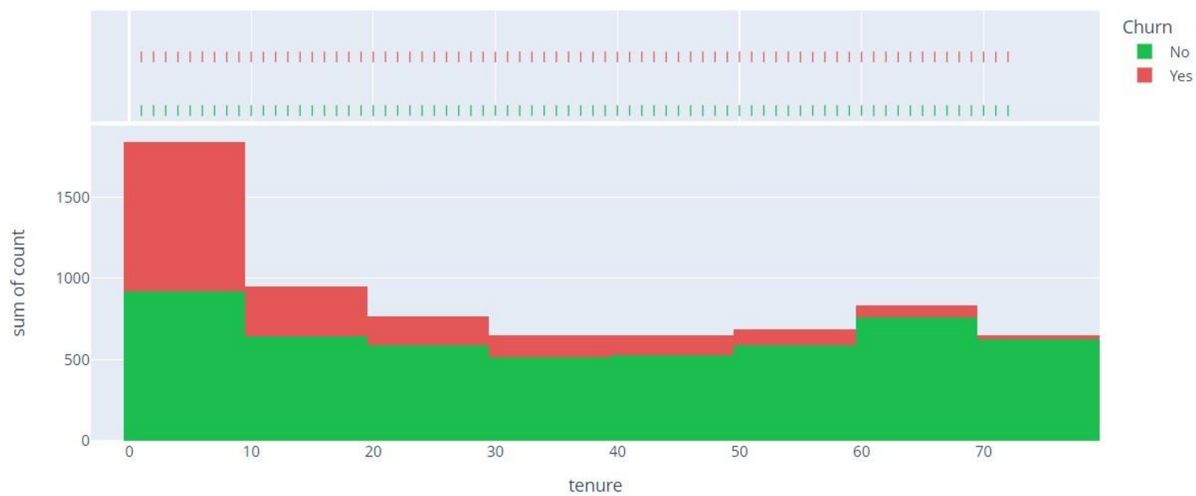
	tenure	Churn	count
0	1	No	233
1	1	Yes	380
2	2	No	115
3	2	Yes	123
4	3	No	106
...	...	...	...
139	70	Yes	11
140	71	No	164
141	71	Yes	6
142	72	No	356
143	72	Yes	6

144 rows × 3 columns

2. Does tenure has any impact on churn?

```
fig = px.histogram(df1.groupby(['tenure', 'Churn']).size().reset_index(name='count'),
                   x="tenure", y="count", color="Churn", marginal="rug", color_discrete_map={"Yes":"#E45756", "No":"#1CBE4F"},
                   title="2. Does tenure has any impact on churn?")

fig.show()
```

2. Does tenure has any impact on churn?



Observation: During 0-10 years of tenure, we can see maximum churning. As the customer turns old, they might get habituated using same telecom service

3. As the dataset is about telecom industry, we need some insights on phone and internet services!

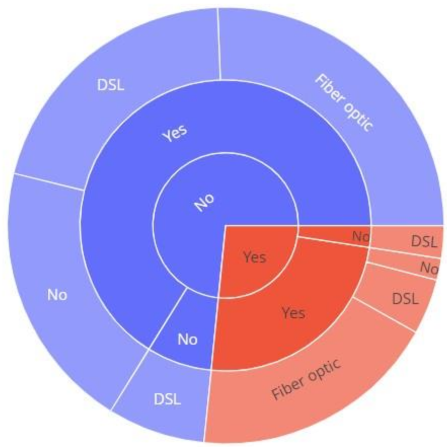
df1.groupby(['Churn', 'PhoneService', 'InternetService']).size()

```
Out[26]: Churn  PhoneService  InternetService
         No     No            DSL                 510
                Yes           DSL                1447
                              Fiber optic        1799
                              No                 1407
         Yes    No            DSL                 170
                Yes           DSL                 289
                              Fiber optic        1297
                              No                  113
         dtype: int64
```

3. As the dataset is about telecom industry, we need some insights on phone and internet services!

```
fig = px.sunburst(df1.groupby(['Churn', 'PhoneService', 'InternetService']).size().reset_index(name='count'),
            path = ['Churn', 'PhoneService', 'InternetService'], values = 'count',
            title='3. As the dataset is about telecom industry, we need some insights on phone and internet services!')

fig.show()
```

3. As the dataset is about telecom industry, we need some insights on phone and internet services!



np.unique(df1.TechSupport)

```
Out[28]: array(['No', 'No internet service', 'Yes'], dtype=object)
```

data_techSupport_yes = df1[df1['TechSupport'] == 'Yes']

data_techSupport_no = df1[df1['TechSupport'] == 'No']

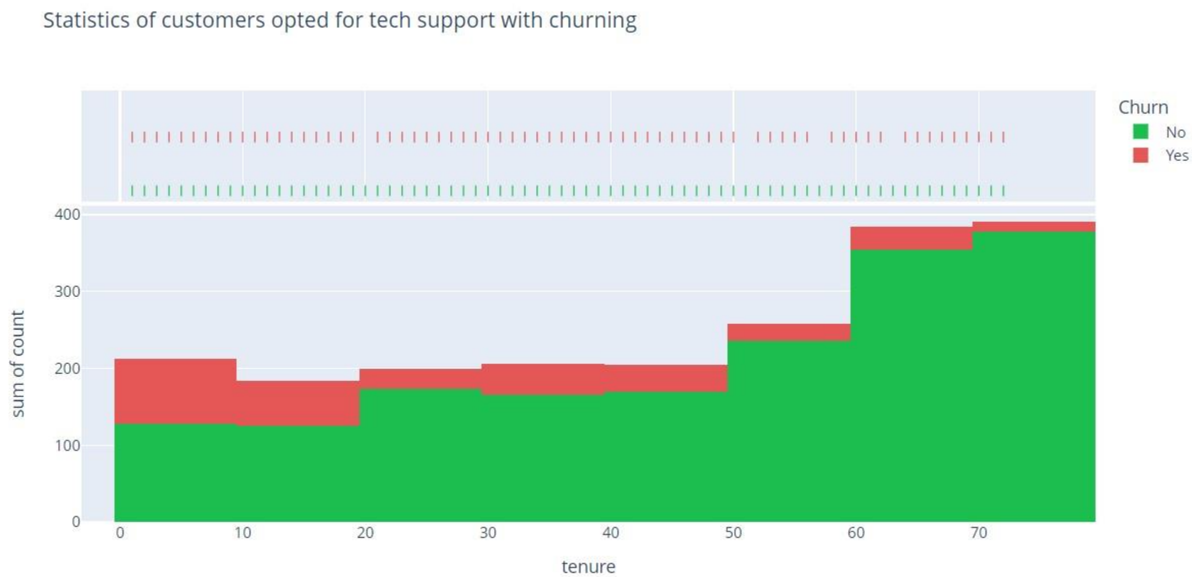
Customers who took tech support

data_techSupport_yes.groupby(['tenure', 'Churn']).size()

```
Out[30]: tenure  Churn
         1       No        19
                 Yes       14
         2       No         8
                 Yes       11
         3       No        20
                          ...
         70      Yes        8
         71      No        89
                 Yes        2
         72      No       221
                 Yes        3
         Length: 140, dtype: int64
```

```
fig = px.histogram(data_techSupport_yes.groupby(['tenure', 'Churn']).size().reset_index(name='count'),
              x="tenure", y="count", marginal="rug", color="Churn", color_discrete_map={"Yes":"#E45756", "No":"#1CBE4F"},
              title="Statistics of customers opted for tech support with churning")
```

fig.show()



Customers who didn't took tech support

data_techSupport_no.groupby(['tenure', 'Churn']).size()

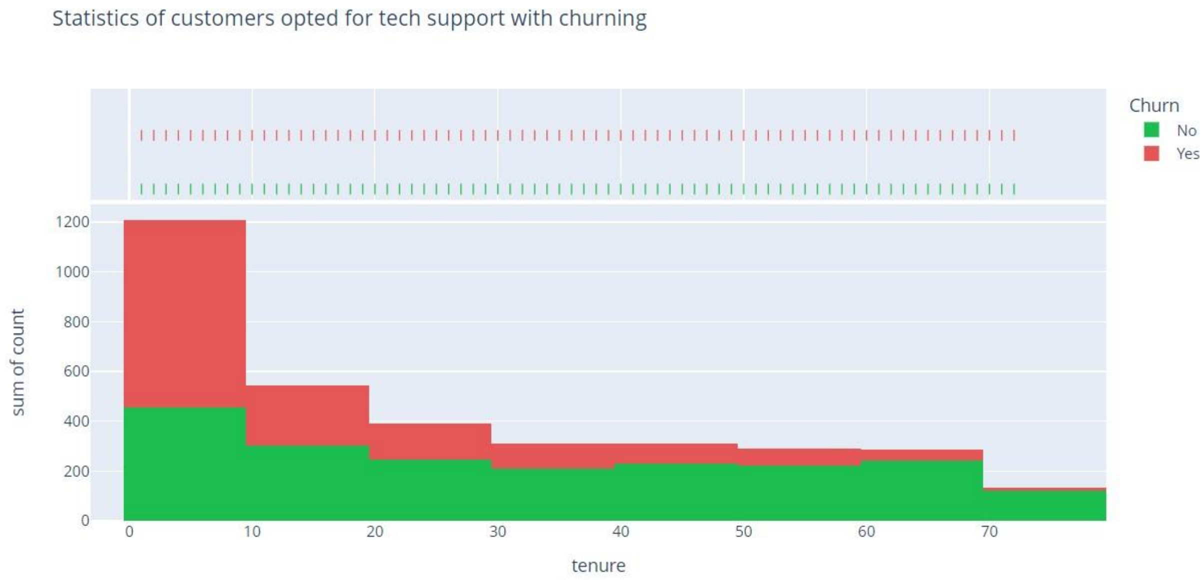
```
Out[32]: tenure  Churn
         1       No       106
                 Yes      308
         2       No        66
                 Yes      106
         3       No        53
                          ...
         70      Yes        3
         71      No        34
                 Yes        4
         72      No        69
                 Yes        3
         Length: 144, dtype: int64
```

fig = px.histogram(data_techSupport_no.groupby(['tenure', 'Churn']).size().reset_index(name='count'),

x='tenure', y='count',color='Churn', marginal='rug', color_discrete_map={"Yes":"#E45756", "No":"#1CBE4F"},

title="Statistics of customers opted for tech support with churning")

fig.show()

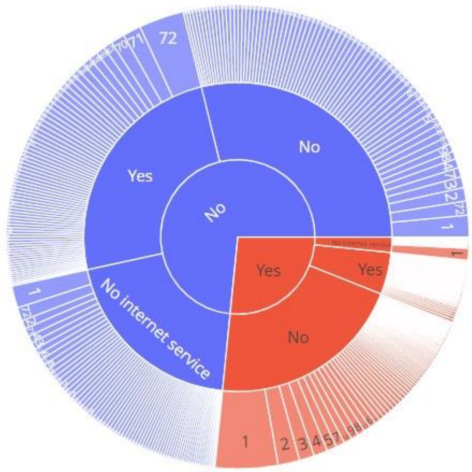


Observations: People with Phone services (yes) and 'Fiber optic' Internet Service are churning more

4. Does customers opted for tech support stayed for longer tenure with less churn?

```
fig = px.sunburst(df1.groupby(['Churn', 'TechSupport', 'tenure']).size().reset_index(name='count'),
                  path=['Churn', 'TechSupport', 'tenure'], values='count',
                  title='4. Does customers opted for tech support stayed for longer tenure with less churn?')

fig.show()
```

4. Does customers opted for tech support stayed for longer tenure with less churn?



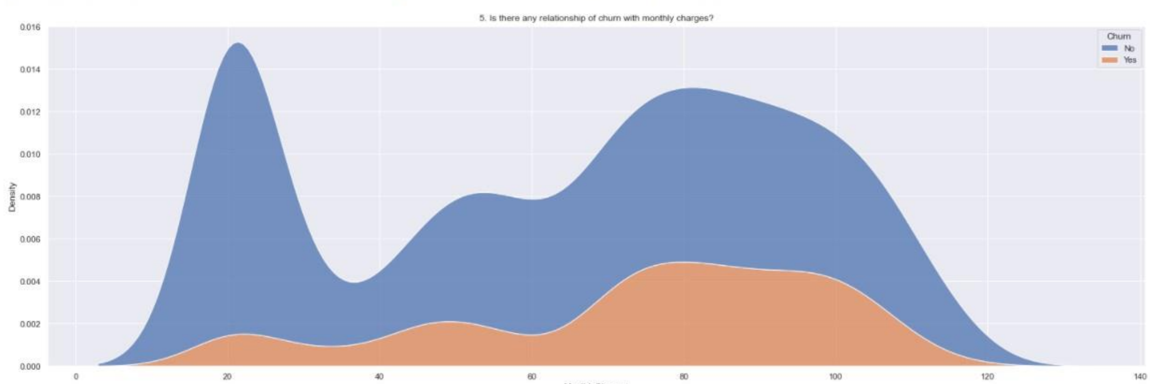
Observations: more churning takes place in first 10 yrs (max in first year itself), for customers with or without tech support. But Churning is more in case of "without tech support" customers

5. Is there any relationship of churn with monthly charges or total charges?

sns.set(rc={'figure.figsize':(26, 8.27)}) # rc - seems row, column

sns.kdeplot(data = df1, x="MonthlyCharges", hue="Churn", multiple="stack").set(title= "5. Is there any relationship of churn with monthly charges?")

```
Out[35]: [Text(0.5, 1.0, '5. Is there any relationship of churn with monthly charges?')]
```

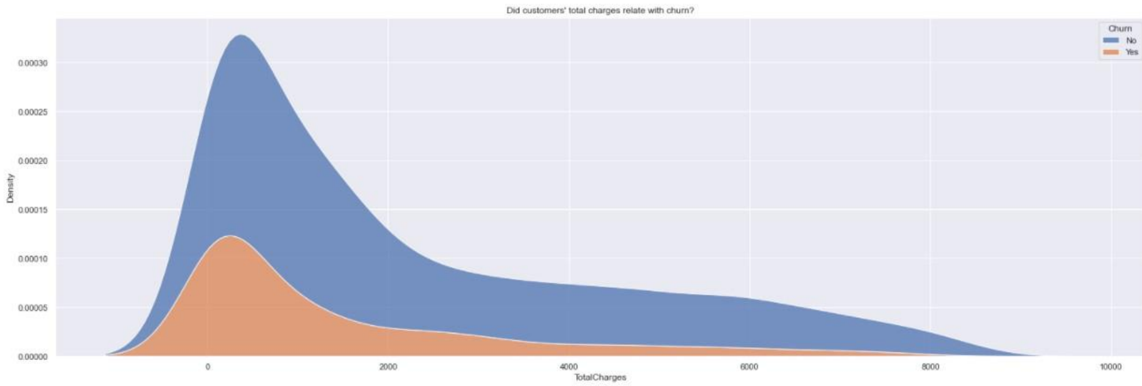


Observations: As the monthy charges are incresing, we can see the density increasing too (60-120), which means more churning with increasing monthly charges

sns.set(rc={'figure.figsize':(26,8.27)})

sns.kdeplot(data=df1, x="TotalCharges", hue="Churn", multiple="stack").set(title="Did customers' total charges relate with churn?")

```
Out[36]: [Text(0.5, 1.0, "Did customers' total charges relate with churn?")]
```



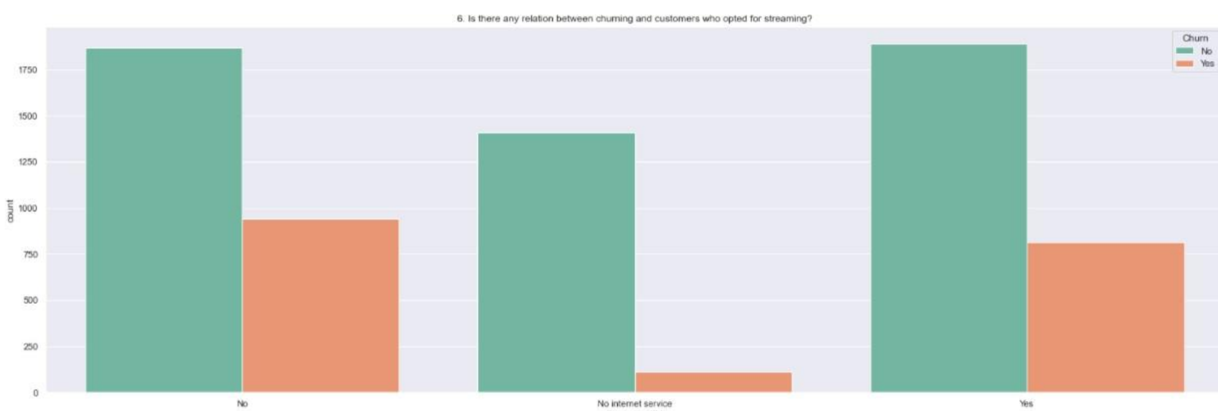
Observation: It is quite opposite of what has been seen for monthly charges. Here high churning occurs when total charges is less, 0-2000 total charges have maximum churning

df1.groupby(['Churn', 'StreamingTV']).size()

```
Out[37]: Churn  StreamingTV
         No     No                     1867
                No internet service    1407
                Yes                    1889
         Yes    No                      942
                No internet service     113
                Yes                     814
         dtype: int64
```

ax = sns.barplot(x="StreamingTV", y="count", hue="Churn", data = df1.groupby(['Churn', 'StreamingTV']).size().reset_index(name='count'),

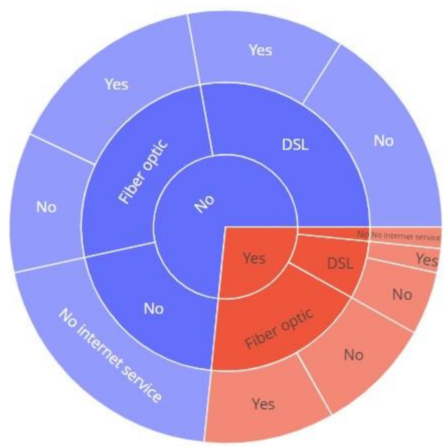
palette="Set2").set(title = "6. Is there any relation between churning and customers who opted for streaming?")



6. Is there any relation between churning and customers who opted for streaming?

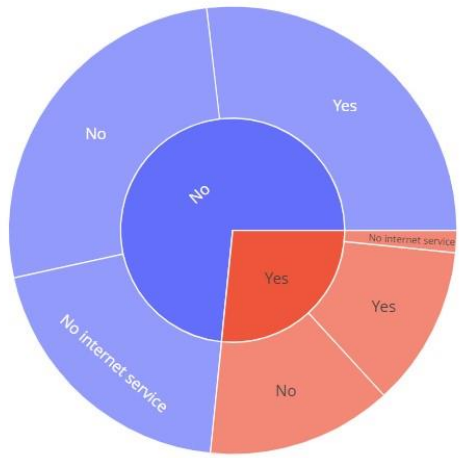
```
fig = px.sunburst(df1.groupby(['Churn', 'InternetService', 'StreamingTV']).size().reset_index(name='count'),
                  path=['Churn', 'InternetService', 'StreamingTV'], values='count',
                  title='6. Is there any relation between churning and customers who opted for streaming?')

fig.show()
```

6. Is there any relation between churning and customers who opted for streaming?




```
fig = px.sunburst(df1.groupby(['Churn', 'StreamingTV']).size().reset_index(name='count'),
                  path=['Churn', 'StreamingTV'], values='count',
                  title='Do customers opted for streaming, faced issue with the service?')

fig.show()
```

Do customers opted for streaming, faced issue with the service?



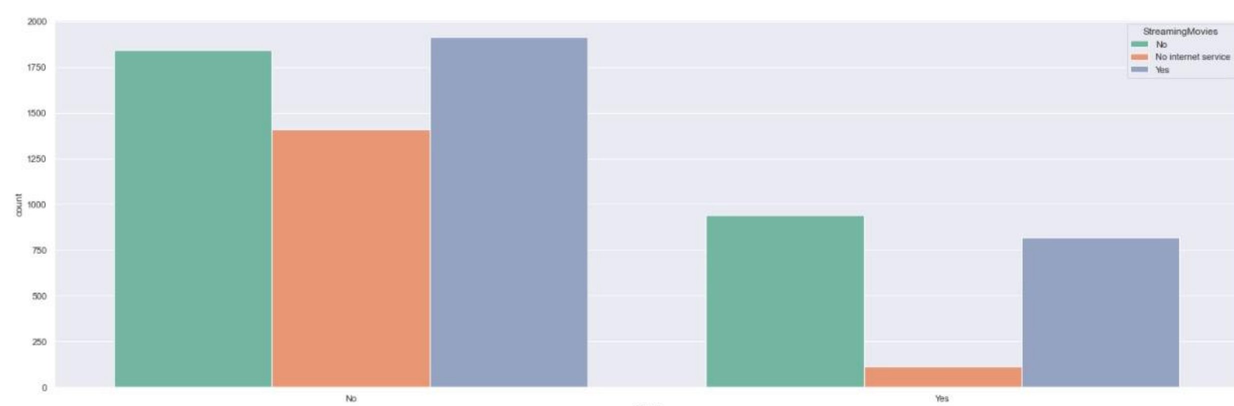
Observation: Churning is being observed equally for the 'Yes', 'No' group of whether connected StreamingTv or not!

df1.groupby(['Churn', 'StreamingMovies']).size()

```
Out[41]: Churn  StreamingMovies
         No     No                     1843
                No internet service    1407
                Yes                    1913
         Yes    No                      938
                No internet service     113
                Yes                     818
         dtype: int64
```

ax = sns.barplot(x="Churn", y="count", hue="StreamingMovies",

data = df1.groupby(['Churn', 'StreamingMovies']).size().reset_index(name="count"), palette="Set2").set(title="")



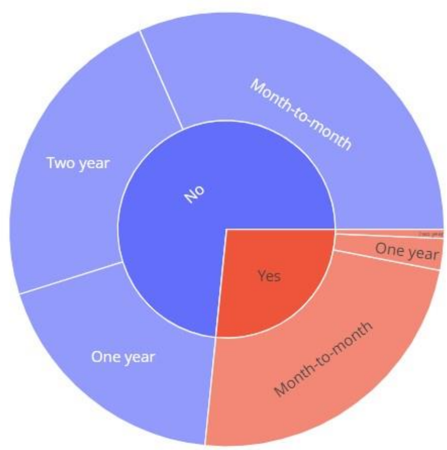
Observation: Churning is being observed equally for both the 'Yes', 'No' group of StreamingMovies df1.groupby(['Churn', 'Contract']).size()

```
Out[43]: Churn  Contract
         No     Month-to-month    2220
                One year          1306
                Two year          1637
         Yes    Month-to-month    1655
                One year           166
                Two year            48
         dtype: int64
```

7. How contract is impacting business?

```
fig = px.sunburst(df1.groupby(['Churn', 'Contract']).size().reset_index(name='count'),
                  path=['Churn', 'Contract'], values='count',
                  title='7. How contract is impacting business?')

fig.show()
```

7. How contract is impacting business?



Observations: clearly visible that customers with month-to-month contract are the highest churners

Senior Citizen vs Churning

```
fig = px.sunburst(df1.groupby(['Churn', 'SeniorCitizen']).size().reset_index(name='count'),
                  path=['Churn', 'SeniorCitizen'], values='count',
                  title='How being or non being SeniorCitizen is impacting Churning?')
```

fig.show()

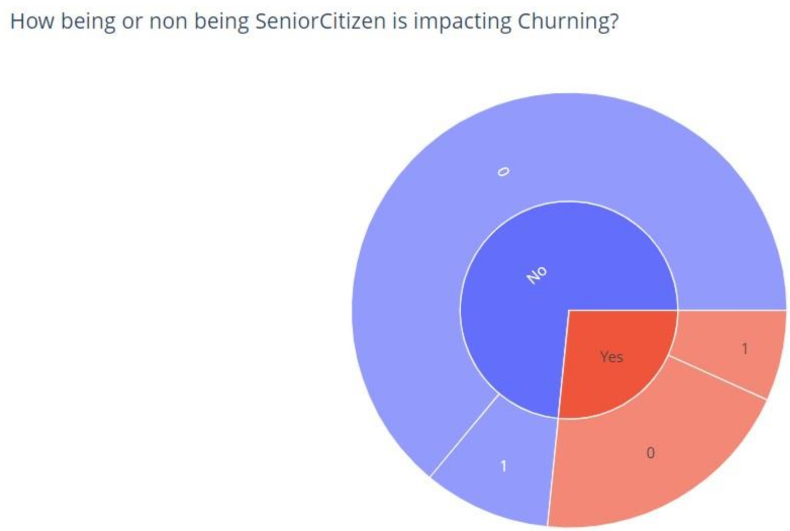
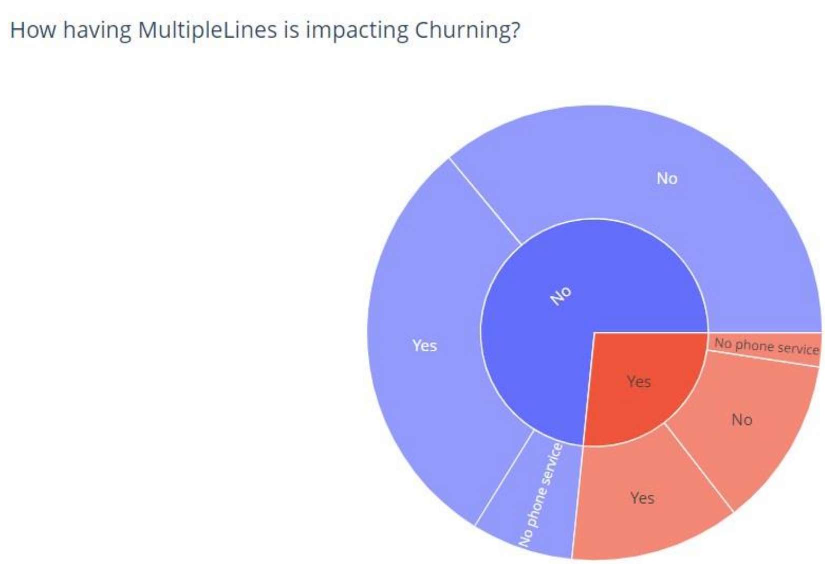


fig = px.sunburst(df1.groupby(['Churn', 'MultipleLines']).size().reset_index(name='count'),

path=['Churn', 'MultipleLines'], values='count',

title='How having MultipleLines is impacting Churning?')

fig.show()



Observation: Having (yes) multiple lines have almost equal impact as not having (No) multiple lines

Many of the columns are yes, no etc. Let's print unique values in object columns to see data values

```
def print_unique_col_values(df):
    for column in df:
        if df[column].dtypes=='object':
            print(f'{column}: {df[column].unique()}')
print_unique_col_values(df1)
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

Some of the columns have no internet service or no phone service, that can be replaced with a simple No

```
df1.replace('No internet service','No',inplace=True)
df1.replace('No phone service','No',inplace=True)
```

```
print_unique_col_values(df1)
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes']
OnlineBackup: ['Yes' 'No']
DeviceProtection: ['No' 'Yes']
TechSupport: ['No' 'Yes']
StreamingTV: ['No' 'Yes']
StreamingMovies: ['No' 'Yes']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

Convert Yes and No to 1 or 0

```
yes_no_columns = ['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup',
            'DeviceProtection','TechSupport','StreamingTV','StreamingMovies','PaperlessBilling','Churn']
for col in yes_no_columns:
  df1[col].replace({'Yes': 1,'No': 0},inplace=True)
```

```
for col in df1:
  print(f'{col}: {df1[col].unique()}')
```

```
gender: ['Female' 'Male']
SeniorCitizen: [0 1]
Partner: [1 0]
Dependents: [0 1]
tenure: [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService: [0 1]
MultipleLines: [0 1]
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: [0 1]
OnlineBackup: [1 0]
DeviceProtection: [0 1]
TechSupport: [0 1]
StreamingTV: [0 1]
StreamingMovies: [0 1]
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: [1 0]
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges: [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges: [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn: [0 1]
```

```
df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

```
df1.gender.unique()
```

```
Out[54]: array([1, 0], dtype=int64)
```

One hot encoding for categorical columns

```
df2 = pd.get_dummies(data=df1, columns=['InternetService','Contract','PaymentMethod'])
df2.columns
```

```
Out[55]: Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'InternetService_DSL', 'InternetService_Fiber optic',
       'InternetService_No', 'Contract_Month-to-month', 'Contract_One year',
       'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)',
       'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check'],
      dtype='object')
```

df2.sample(5)

Out[56]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	...	InternetService_DSL
2814	1	0	1	1	62	0	0	1	1	0	...	1
4933	1	0	0	1	4	0	0	1	1	1	...	1
2040	0	0	1	0	71	1	1	1	1	1	...	1
4772	1	0	1	1	69	1	0	0	1	1	...	1
1674	1	0	1	1	23	1	1	0	0	0	...	0

5 rows × 27 columns

df2.dtypes

```
Out[57]: gender                                       int64
SeniorCitizen                                int64
Partner                                      int64
Dependents                                   int64
tenure                                       int64
PhoneService                                 int64
MultipleLines                                int64
OnlineSecurity                               int64
OnlineBackup                                 int64
DeviceProtection                             int64
TechSupport                                  int64
StreamingTV                                  int64
StreamingMovies                              int64
PaperlessBilling                             int64
MonthlyCharges                             float64
TotalCharges                               float64
Churn                                        int64
InternetService_DSL                          uint8
InternetService_Fiber optic                  uint8
InternetService_No                           uint8
Contract_Month-to-month                      uint8
Contract_One year                            uint8
Contract_Two year                            uint8
PaymentMethod_Bank transfer (automatic)      uint8
PaymentMethod_Credit card (automatic)        uint8
PaymentMethod_Electronic check               uint8
PaymentMethod_Mailed check                   uint8
dtype: object
```

cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df2[cols_to_scale] = scaler.fit_transform(df2[cols_to_scale])

for col in df2:

print(f'{col}: {df2[col].unique()}')

```
gender: [1 0]
SeniorCitizen: [0 1]
Partner: [1 0]
Dependents: [0 1]
tenure: [0.         0.46478873 0.01408451 0.61971831 0.09859155 0.29577465
 0.12676056 0.38028169 0.85915493 0.16901408 0.21126761 0.8028169
 0.67605634 0.33802817 0.95774648 0.71830986 0.98591549 0.28169014
 0.15492958 0.4084507  0.64788732 1.         0.22535211 0.36619718
 0.05633803 0.63380282 0.14084507 0.97183099 0.87323944 0.5915493
 0.1971831  0.83098592 0.23943662 0.91549296 0.11267606 0.02816901
 0.42253521 0.69014085 0.88732394 0.77464789 0.08450704 0.57746479
 0.47887324 0.66197183 0.3943662  0.90140845 0.52112676 0.94366197
 0.43661972 0.76056338 0.50704225 0.49295775 0.56338028 0.07042254
 0.04225352 0.45070423 0.92957746 0.30985915 0.78873239 0.84507042
 0.18309859 0.26760563 0.73239437 0.54929577 0.81690141 0.32394366
 0.6056338  0.25352113 0.74647887 0.70422535 0.35211268 0.53521127]
PhoneService: [0 1]
MultipleLines: [0 1]
OnlineSecurity: [0 1]
OnlineBackup: [1 0]
DeviceProtection: [0 1]
TechSupport: [0 1]
StreamingTV: [0 1]
StreamingMovies: [0 1]
PaperlessBilling: [1 0]
MonthlyCharges: [0.11542289 0.38507463 0.35422886 ... 0.44626866 0.25820896 0.60149254]
TotalCharges: [0.0012751  0.21586661 0.01031041 ... 0.03780868 0.03321025 0.78764136]
Churn: [0 1]
InternetService_DSL: [1 0]
InternetService_Fiber optic: [0 1]
InternetService_No: [0 1]
Contract_Month-to-month: [1 0]
```

Data Splitting

Train Test Split

X = df2.drop('Churn',axis='columns')

y = df2['Churn']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=5)

X_train.shape

```
Out[61]: (5625, 26)
```

X_test.shape

```
Out[62]: (1407, 26)
```

X_train[:10]

Out[63]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	...	InternetService_D!
5664	1	1	0	0	0.126761	1	0	0	0	1	...	
101	1	0	1	1	0.000000	1	0	0	0	0	...	
2621	0	0	1	0	0.985915	1	0	0	1	1	...	
392	1	1	0	0	0.014085	1	0	0	0	0	...	
1327	0	0	1	0	0.816901	1	1	0	0	1	...	
3607	1	0	0	0	0.169014	1	0	1	0	0	...	
2773	0	0	1	0	0.323944	0	0	0	0	1	...	
1936	1	0	1	0	0.704225	1	0	1	1	0	...	
5387	0	0	0	0	0.042254	0	0	0	0	0	...	
4331	0	0	0	0	0.985915	1	1	0	0	0	...	

10 rows × 26 columns

```
len(X_train.columns)
```

Out[64]: 26

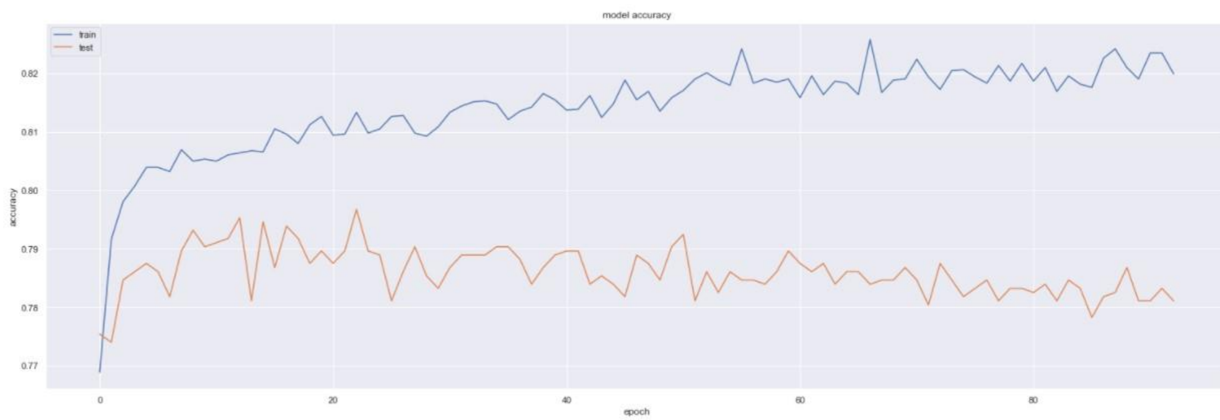
ANN Model Building

```
nn_model = Sequential()
nn_model.add(Dense(64,kernel_regularizer=tf.keras.regularizers.l2(0.001), input_dim=26, activation='relu' ))
nn_model.add(Dropout(rate=0.2))
nn_model.add(Dense(8,kernel_regularizer=tf.keras.regularizers.l2(0.001),activation='relu'))
nn_model.add(Dropout(rate=0.1))
nn_model.add(Dense(1, activation='sigmoid'))
lr_schedule = tf.keras.optimizers.schedules.InverseTimeDecay( 0.001,
   decay_steps=(X_train.shape[0]/32)*50,
   decay_rate=1,
   staircase=False)
```

This time decay means for every 50 epochs the learning rate will be half of 0.001 value

```
def get_optimizer():
  return tf.keras.optimizers.Adam(lr_schedule)
def get_callbacks():
  return [tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',patience=70,restore_best_weights=True)]
```

```
nn_model.compile(loss = "binary_crossentropy",
            optimizer = get_optimizer(),
            metrics=['accuracy'])

history = nn_model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=150, batch_size=32, callbacks=get_callbacks(),verbose=0)

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Evaluation Metrics

```
yprednn=nn_model.predict(X_test)
yprednn=yprednn.round()
print('Neural Network:\n {}\n'.format(
  metrics.classification_report(yprednn, y_test)))
nn_conf_matrix=metrics.confusion_matrix(yprednn,y_test)
conf_mat_nn = pd.DataFrame(nn_conf_matrix,
```

```
    columns=["Predicted NO", "Predicted YES"],
    index=["Actual NO", "Actual YES"])
print(conf_mat_nn)
```

Accuracy with ANN Model: 80.0%

```
Neural Network:
              precision    recall  f1-score   support

         0.0       0.89      0.83      0.86      1067
         1.0       0.57      0.68      0.62       340

    accuracy                           0.80      1407
   macro avg       0.73      0.76      0.74      1407
weighted avg       0.81      0.80      0.80      1407

            Predicted NO  Predicted YES
Actual NO            890            177
Actual YES           109            231

Accuracy with ANN Model: 80.0%
```

Hyperparameter Tuning

```
gb = GradientBoostingClassifier()

parameters = {
    'loss': ['deviance', 'exponential'],
    'learning_rate': [0.001, 0.1, 1, 10],
    'n_estimators': [100, 150, 180, 200]
}

grid_search = GridSearchCV(gb, parameters, cv = 5, n_jobs = -1, verbose = 1)
grid_search.fit(X_train, y_train)
```

```
        Fitting 5 folds for each of 32 candidates, totalling 160 fits
Out[76]: GridSearchCV(cv=5, estimator=GradientBoostingClassifier(), n_jobs=-1,
                     param_grid={'learning_rate': [0.001, 0.1, 1, 10],
                                 'loss': ['deviance', 'exponential'],
                                 'n_estimators': [100, 150, 180, 200]},
                     verbose=1)
```

```
print(grid_search.best_params_)
```

```
print(grid_search.best_score_)
```

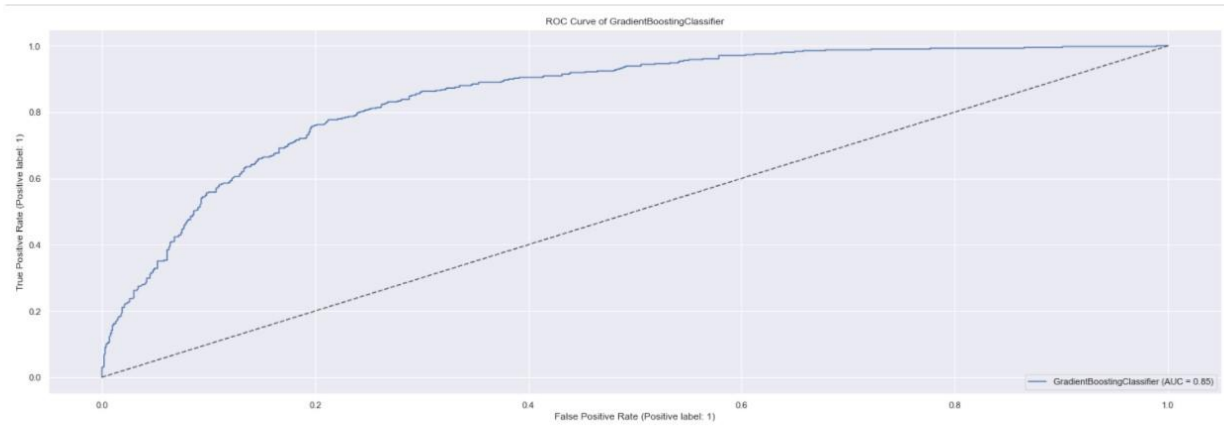
```
{'learning_rate': 0.1, 'loss': 'exponential', 'n_estimators': 100}
0.8071111111111111
```

```
gb = GradientBoostingClassifier(learning_rate = 0.1, loss = 'exponential', n_estimators = 80)
gb.fit(X_train,y_train)
y_pred = gb.predict(X_test)
print("Accuracy with Gradient Booting Classifier: {0:.5}%".format(accuracy_score(y_test, y_pred)*100))
```

```
Accuracy with Gradient Booting Classifier: 80.028%
```

Performance Metrics

```
gbc_disp = plot_roc_curve(gb, X_test, y_test)
plt.title("ROC Curve of {}".format(type(gb).__name__))
plt.plot([0,1],[0,1],"--",color="k",alpha=0.7)
plt.show()
```



Conclusion:

1. When we built a Model with Artificial Neural Network using Tensorflow/Keras, we found that the accuracy was 80%.

2. Hyperparameter Tuning using Gradient Boosting Classifier yeilded an accuracy score of 80.028% and an AUC value of 0.85.

Thus, we can conclude that our ANN Model has performed appropriately.

Future Scope:

Offers and Improving Churn Rate:

1. Discounts: As the most important feature is total charges, followed by monthly charges, potential churners identified through the modelling should be offered huge discount on next month or months contract. This covers 80 % of the reasons identified for churning. For this modelling, the False Negative Rate should be minimized or Recall should be maximized so that the discounts are sent to maximum of the potential churners.

2. New contract: A six month or four month contract should be implemented. This will encourage the reserved customers who want shorter contracts and will increase their tenure on the service thus making them less likely to churn.

3. Online Security: This service should be promoted more and offered complimentary/free for trial periods depending on cost to company. The customers who do not have online security are more likely to churn and thus this offer could be combined with the first one mentioned and discount could only be offered on this.

4. Fiber optic: The fiber optic internet is costly and thus should either be promoted to appropriate target audience or better technology can be implemented to cut cost on this service. Ultimately the market research team has to decide the break even point for this service, whether it is profiting as much as the loss of customers it is causing.

5. Another method to quantify the offers to be made is using manually generated features and their effect on the model.
