Blog On the Project Telecom Churn Using Machine Learning.

Author Name- Ankur Kumar

Institute-DataTrained

Problem Statement:-

Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritise focused marketing efforts on that subset of their customer base.

Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

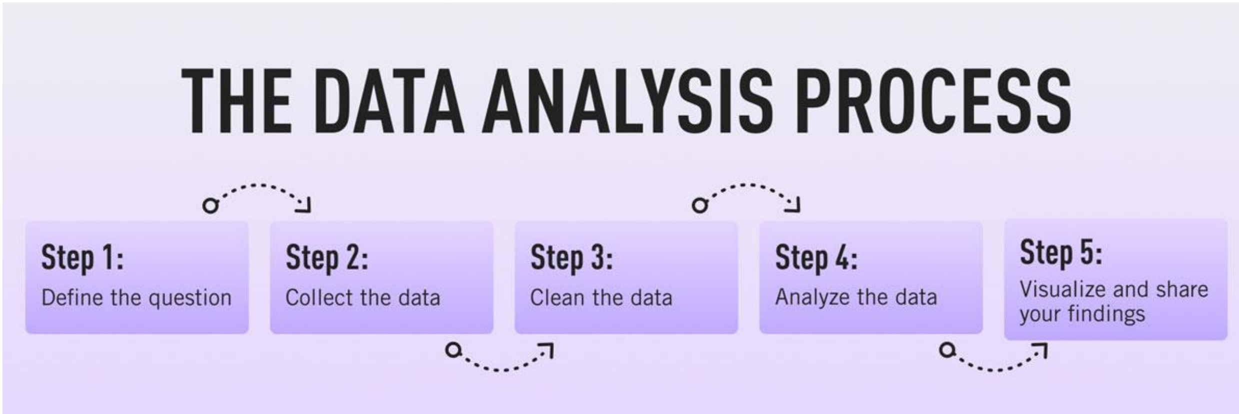
You will examine customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

In [5]:

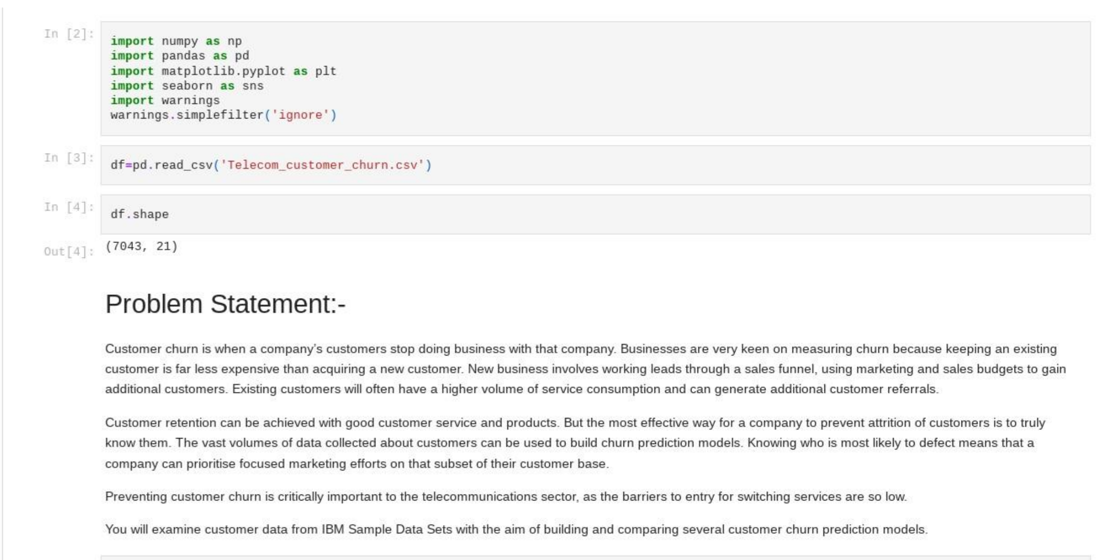
Data Analysis:-

Before Making predictive model lets do the data analysis part to get insights of the data.

Various Steps of Data Analysis are:-



1. Importing necessary libraries
2. Collection of Data.
3. Checking the dimension of data



In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter('ignore')
```

In [3]:
```
df=pd.read_csv('Telecom_customer_churn.csv')
```

In [4]:
```
df.shape
```

Out[4]:
```
(7043, 21)
```

Problem Statement:-

Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritise focused marketing efforts on that subset of their customer base.

Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

You will examine customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

In [5]:

The data contains 7043 rows and 21 columns

4. Checking data types.

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

```
In [8]: c=df.customerID
        df=df.drop('customerID',axis=1)
```

The data consist of 18 categorical column and 3 numerical columns.

5. Checking missing values in the data.

```
MonthlyCharges      1585
TotalCharges        6531
Churn                  2
dtype: int64
```

```
In [10]: df.isna().sum()
```

```
Out[10]: gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
In [11]: for col in df:
             if df[col].dtypes=='object':
                 print(df[col].value_counts())
                 print(col)
                 print('^'*100)
```

```
Male      3555
Female    3488
```

Visualization:-

Performing Exploratory Data Analysis with the target i.e customer churn to get insight which factor is responsible for attrition of the customer much,and we the company need to work to retain those customers.

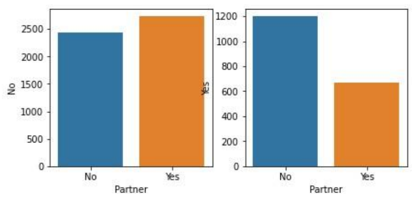
Partner vs Churn- ***Clearly the churn of customers who is not partner is more. The churn can be decrease by making them parter.***

In [52]:

```
sns.set_style()
plt.figure(figsize=(7,7))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[52]: <AxesSubplot:xlabel='Partner', ylabel='Yes'>



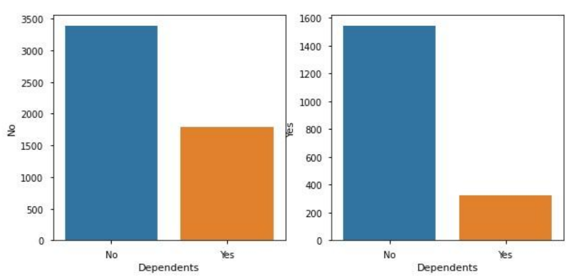
Observation:- Clearly the churn of customer who not partner is more.

*Churn Vs Dependents-**Those who don't have dependents the churn rate is high***

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[54]: <AxesSubplot:xlabel='Dependents', ylabel='Yes'>



Observation:- Those who dont have dependents the churn rate is high

In [55]:

```
cr=pd.crosstab(index=df['PhoneService'],columns=df['Churn'])
```

Churn vs PhoneServices- Most of the customer is using phone services and it does not show any relation for churn rate.

Out[55]:

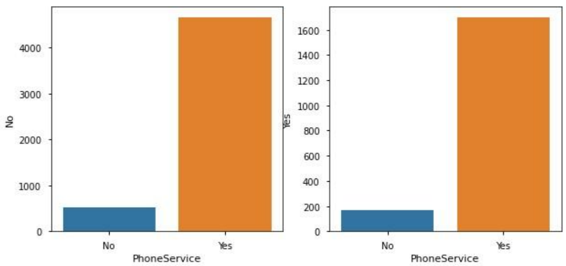
Churn	No	Yes
PhoneService		
No	512	170
Yes	4662	1699

In [56]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[56]: <AxesSubplot:xlabel='PhoneService', ylabel='Yes'>



*Churn Vs MultipleLines-**Those who are using multiple line the churn rate is also high***

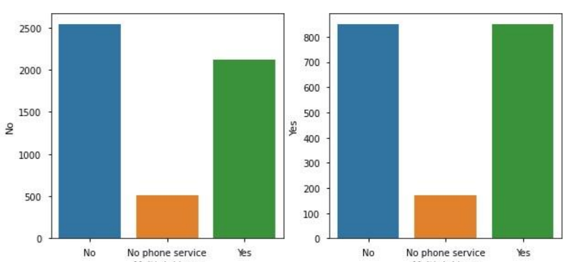
No phone service	512	170
Yes	2121	850

In [58]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[58]: <AxesSubplot:xlabel='MultipleLines', ylabel='Yes'>



Observation:- Those who are using multiple line the churn rate is also high

*Churn vs Internet Services-**The customer with fiber optic has high rate of churn***

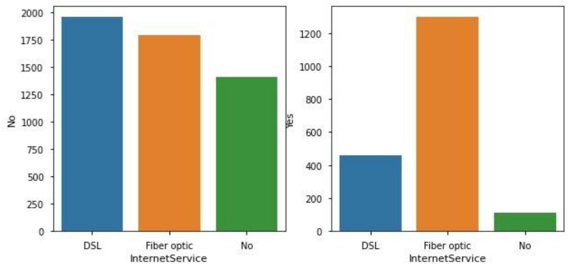
Fiber optic	1799	1297
No	1413	113

In [60]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[60]: <AxesSubplot:xlabel='InternetService', ylabel='Yes'>



Observation:- The customer with fiber optic has high rate of churn

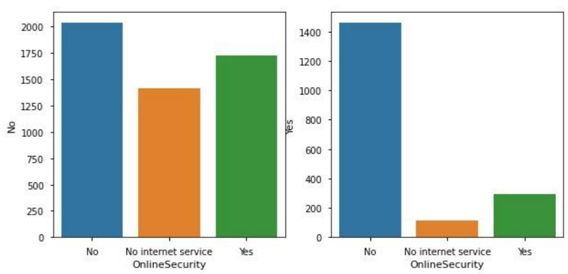
*Churn vs Online Security-**The customer who is not using the online security having high churn rate***

In [62]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[62]: <AxesSubplot:xlabel='OnlineSecurity', ylabel='Yes'>



Observation:- The customer who is not using the online security having high churn rate

In [63]:

```
cr=pd.crosstab(index=df['OnlineBackup'],columns=df['Churn'])
cr
```

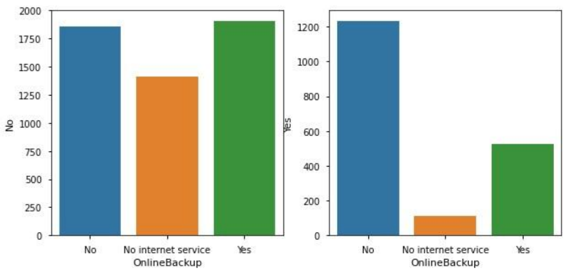
*Churn vs Online Backup:- **The churn rate is high for those customer who is not using online backup.***

In [64]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[64]: <AxesSubplot:xlabel='OnlineBackup', ylabel='Yes'>



Observation:- The churn rate is high for those customer who is not using online backup

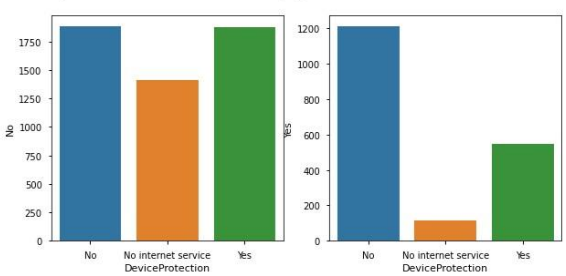
In [65]:

```
cr=pd.crosstab(index=df['DeviceProtection'],columns=df['Churn'])
```

*Churn vs Device Protection:- **Those who dont uses device protection the churn rate is also high.***

```
ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[66]: <AxesSubplot:xlabel='DeviceProtection', ylabel='Yes'>



Observation:- Those who dont uses device protection the churn rate is also high

In [67]:

```
cr=pd.crosstab(index=df['TechSupport'],columns=df['Churn'])
cr
```

Out[67]:

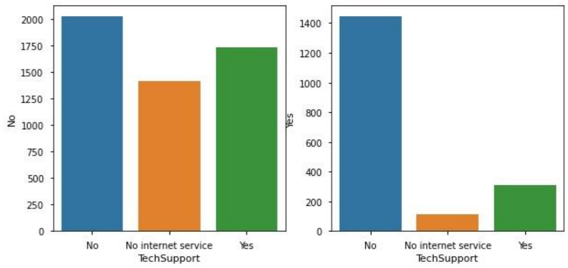
Churn	No	Yes
TechSupport		

Churn vs Tech Support:-Those who are not using Tech support has higher chance of churn

```
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[68]: `<AxesSubplot:xlabel='TechSupport', ylabel='Yes'>`



Observation:- Those who are not using Tech support has higher chance of churn

In [69]:
```
cr=pd.crosstab(index=df['StreamingTV'],columns=df['Churn'])
cr
```

Out[69]:

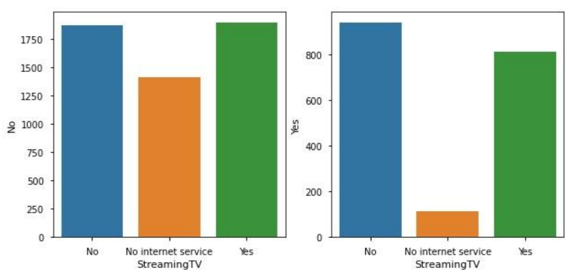
Churn	No	Yes
StreamingTV		

Churn vs Streaming TV:- ***Those who are not streaming movies and TV has higher churn rate***

In [70]:
```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[70]: `<AxesSubplot:xlabel='StreamingTV', ylabel='Yes'>`



In [71]:
```
cr=pd.crosstab(index=df['StreamingMovies'],columns=df['Churn'])
cr
```

Out[71]:

Churn	No	Yes
StreamingMovies		

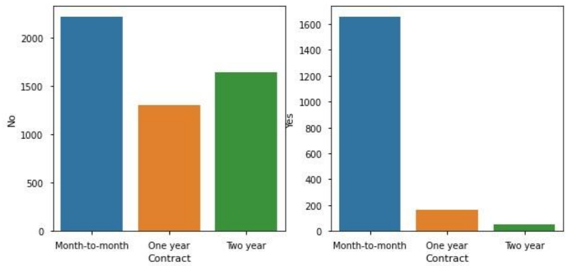
Churn Vs Contract:- ***Month to month contract user has higher churn rate***

In [74]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

Out[74]: `<AxesSubplot:xlabel='Contract', ylabel='Yes'>`



Observation:- Month to month contract user has higher churn rate

In [75]:

```
cr=pd.crosstab(index=df['PaperlessBilling'],columns=df['Churn'])
cr
```

*Churn Vs Paperless billing:- **Those who using paperless billing has higher churn rate***

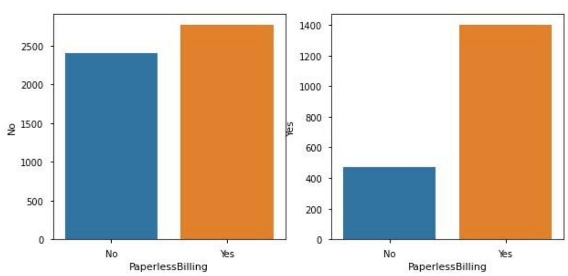
PaperlessBilling		
No	2403	469
Yes	2771	1400

In [76]:

```
plt.figure(figsize=(10,10))
ax=plt.subplot(2,2,1)
sns.barplot(x=cr.index,y=cr.No)


ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
```

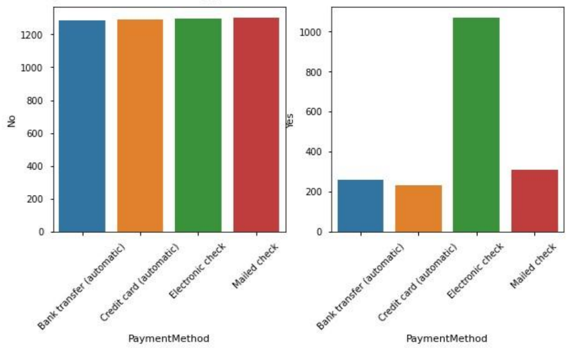
Out[76]: `<AxesSubplot:xlabel='PaperlessBilling', ylabel='Yes'>`



*Churn vs Payment Method:-**Electronic check payment method has high churn rate***

```
ax=plt.subplot(2,2,2)
sns.barplot(x=cr.index,y=cr.Yes)
plt.xticks(rotation=45)
```

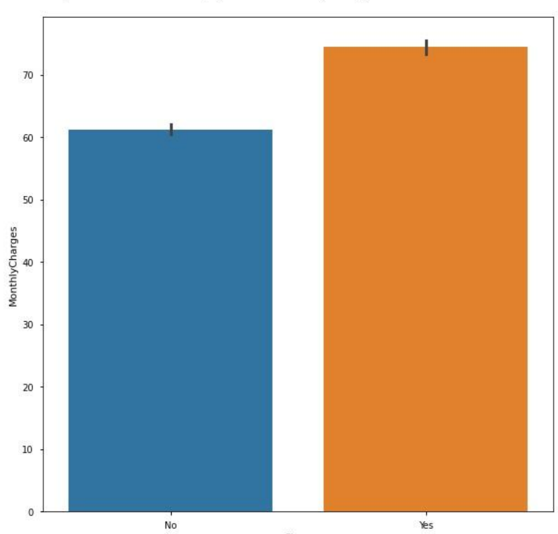
Out[78]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Bank transfer (automatic)'),
  Text(1, 0, 'Credit card (automatic)'),
  Text(2, 0, 'Electronic check'),
  Text(3, 0, 'Mailed check')])
```



Observation:- Electronic check payment method has high churn rate

Churn Vs Monthly Charges:- ***As Monthly charges increases the churn rate increases but Totalcharges is opposite as it increase the churn rate decreases***

Out[79]:
```
<AxesSubplot:xlabel='Churn', ylabel='MonthlyCharges'>
```



Encoding Categorical Variable:-Encoding categorical variable into numbers to proceed for predictive model.Here Label Encoder is used for encoding purpose.

In [83]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [84]:
```python
cat=[]
for i in df.columns:
    if df[i].dtypes == 'object':
        cat.append(i)

print(cat)
```

```
['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProte
ction', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn']
```

In [85]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df[cat]=df[cat].apply(le.fit_transform)
```

In [86]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   int64
 1   SeniorCitizen     7043 non-null   int64
 2   Partner           7043 non-null   int64
 3   Dependents        7043 non-null   int64
 4   tenure            7043 non-null   int64
 5   PhoneService      7043 non-null   int64
 6   MultipleLines     7043 non-null   int64
 7   InternetService   7043 non-null   int64
 8   OnlineSecurity    7043 non-null   int64
```

Checking Duplicates in data and removing those rows.

6499	1	0	0	0	1	1	0	2	1	1	1	1
6518	1	0	0	0	1	1	0	0	0	0	0	0
6609	1	0	0	0	1	1	0	2	1	1	1	1
6706	0	0	0	0	1	1	0	2	1	1	1	1
6764	0	0	0	0	1	1	0	1	0	0	0	0
6774	0	0	0	0	1	1	0	2	1	1	1	1
6924	1	0	0	0	1	1	0	1	0	0	0	0

In [88]:
```python
df.duplicated().sum()
```

Out[88]: 22

In [89]:
```python
df.drop_duplicates(keep="first",inplace=True)
```

In [90]:
```python
df[df.duplicated()]
```

Out[90]:

gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV

In [91]:
```python
df.shape
```

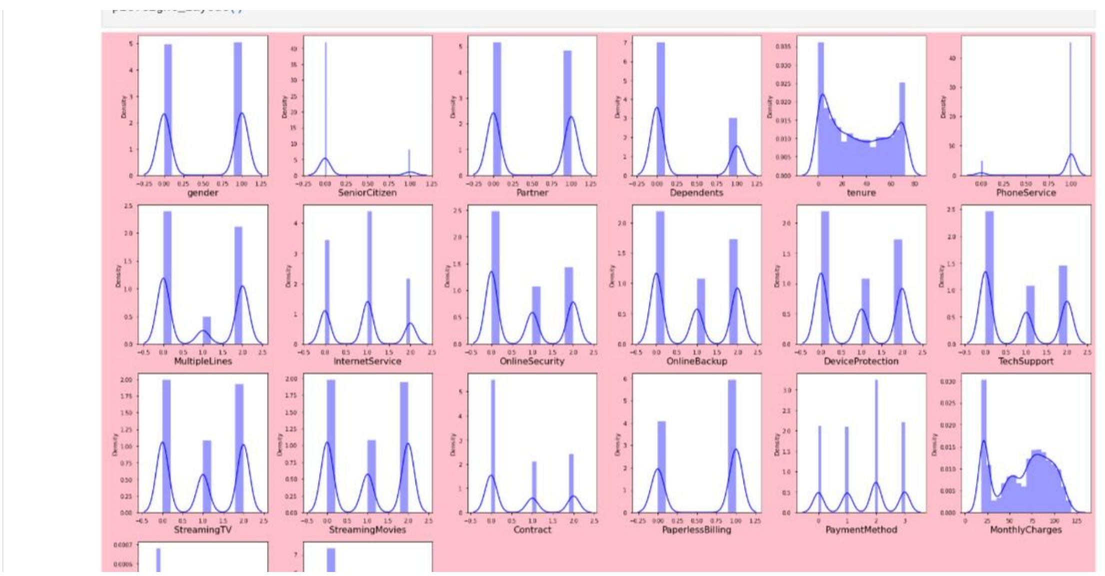
Out[91]: (7021, 20)

In [92]:
```python
plt.figure(figsize=(30,30))
sns.heatmap(df.corr(),annot=True,annot_kws={'size':10},fmt='.1g')
```

```
<AxesSubplot:>
```

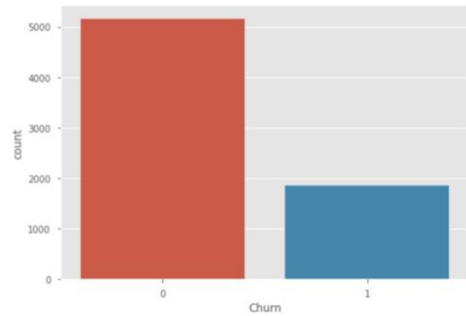
Finding Multicollinearity problem in the data.

Checking skewness in the variable:-



Treating Imbalanced Dataset for prediction:- Here from sklearn library i used SMOTE to balance the target variable.

Out[97]: <AxesSubplot:xlabel='Churn', ylabel='count'>

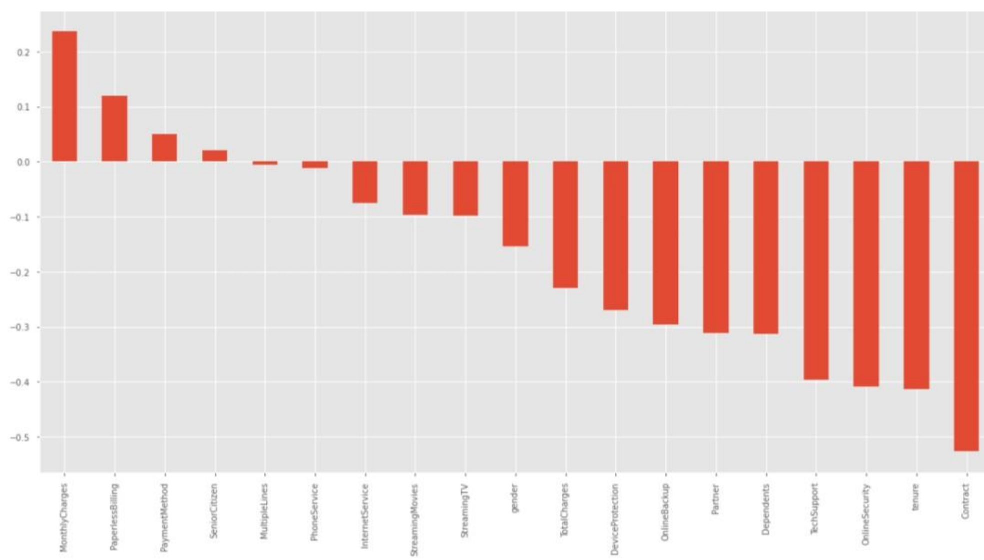


In [105…
```
from imblearn.over_sampling import SMOTE
sm=SMOTE()
```

Observation:- Clearly the sample is imbalanced so we to balance first

In [106…
```
X=df.drop('Churn',axis=1)
y=df.Churn
```

Correlation With target variable:-



Treating Multicollinearity problem:-Through variance inflation Factor the multicollinearity problem is treated if score is more than 5 the column is removed.

```
vif['vif']=[variance_inflation_factor(x_scaled,i) for i in range(x_scaled.shape[1])]
vif['features']=X.columns
vif
```

Out[118…

	vif	features
0	1.022897	gender
1	1.091300	SeniorCitizen
2	1.542007	Partner
3	1.428689	Dependents
4	8.371725	tenure
5	1.669033	PhoneService
6	1.396016	MultipleLines
7	1.682516	InternetService
8	1.366945	OnlineSecurity
9	1.294980	OnlineBackup
10	1.355070	DeviceProtection
11	1.427509	TechSupport
12	1.488824	StreamingTV
13	1.472980	StreamingMovies
14	2.523190	Contract
15	1.148787	PaperlessBilling
16	1.180027	PaymentMethod
17	4.275973	MonthlyCharges
18	10.587510	TotalCharges

Model Building:-

10328 rows × 17 columns

In [123…

```
## Model building

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,plot_roc_curve
```

In [124…

```
# Logistic Regression
lr=LogisticRegression()
```

In [126…

```
for i in range(0,1000):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=i)
    lr.fit(x_train,y_train)
    ypred=lr.predict(x_train)
    y_pred=lr.predict(x_test)
    if round(accuracy_score(y_train,ypred)*100)==round(accuracy_score(y_test,y_pred)*100):
        print('At random state',i,'model performs well')
        print('At random state ',i)
        print(round(accuracy_score(y_test,y_pred)*100))
```

```
At random state 1 model performs well
At random state  1
80
At random state 4 model performs well
At random state  4
80
At random state 6 model performs well
```

1.**Logistic Regression:- It has given the accuracy of 77%**

In [141…
```
model(dt,x_train,x_test,y_train,y_test,train=False)
```

```
76.96030977734753
              precision    recall  f1-score   support

           0       0.78      0.76      0.77      1577
           1       0.76      0.78      0.77      1522

    accuracy                           0.77      3099
   macro avg       0.77      0.77      0.77      3099
weighted avg       0.77      0.77      0.77      3099

****************************************************************************************************
cross_val_score 0.7548514048505846
```

2. **Random Forest:- It has given the accuracy of 80%**

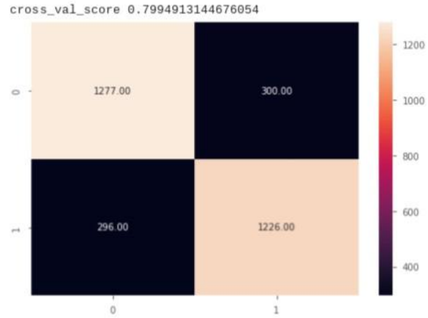
In [144…
```
model(rf,x_train,x_test,y_train,y_test,train=False)
```

```
80.76798967408841
              precision    recall  f1-score   support

           0       0.81      0.81      0.81      1577
           1       0.80      0.81      0.80      1522

    accuracy                           0.81      3099
   macro avg       0.81      0.81      0.81      3099
weighted avg       0.81      0.81      0.81      3099

****************************************************************************************************
cross_val_score 0.7994913144676054
```



GradientBoosting Classifier

3. **GradientBoosting Classifier:- By using various model sequentially we got better accuracy. And the Accuracy is 82%**

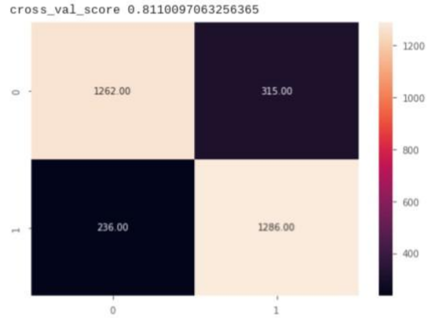
```
In [147…  model(gb,x_train,x_test,y_train,y_test,train=False)

82.22007099064214
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1577
           1       0.80      0.84      0.82      1522

    accuracy                           0.82      3099
   macro avg       0.82      0.82      0.82      3099
weighted avg       0.82      0.82      0.82      3099

****************************************************************************************************
cross_val_score 0.8110097063256365
```



Hyperparameter Tunning:-A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.

However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

GridSearchCV

In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

```
In [154…  # Hyperparameter Tunning
          from sklearn.model_selection import GridSearchCV

In [155…  # GradientBoosting Classifier
          params={'n_estimators':[100],
                 'learning_rate':[0.1,0.01,0.001,0.2,1,0.002],
                 'max_depth':[4,5,6,7,8,9],
                 'min_samples_split':[3,4,5,6,7],
                 'min_samples_leaf':[3,4,5,7,9]}

In [156…  gs=GridSearchCV(gb,param_grid=params,n_jobs=-1)
          gs.fit(x_train,y_train)

Out[156… GridSearchCV(estimator=GradientBoostingClassifier(), n_jobs=-1,
                      param_grid={'learning_rate': [0.1, 0.01, 0.001, 0.2, 1, 0.002],
                                  'max_depth': [4, 5, 6, 7, 8, 9],
                                  'min_samples_leaf': [3, 4, 5, 7, 9],
                                  'min_samples_split': [3, 4, 5, 6, 7],
                                  'n_estimators': [100]})

In [158…  gs.best_params_

Out[158… {'learning_rate': 0.2,
          'max_depth': 4,
          'min_samples_leaf': 3,
          'min_samples_split': 4,
          'n_estimators': 100}

In [191…  gb=GradientBoostingClassifier(learning_rate=0.2,max_depth=4,min_samples_leaf=3,min_samples_split=4,n_estimators=100)
```

GradientBoostingClassifier:- By applying GridSearchCV the model is less overfitted and we found that accuracy also increased from 82% to 83%


```
In [154…  # Hyperparameter Tunning
          from sklearn.model_selection import GridSearchCV

In [155…  # GradientBoosting Classifier
          params={'n_estimators':[100],
                 'learning_rate':[0.1,0.01,0.001,0.2,1,0.002],
                 'max_depth':[4,5,6,7,8,9],
                 'min_samples_split':[3,4,5,6,7],
                 'min_samples_leaf':[3,4,5,7,9]}

In [156…  gs=GridSearchCV(gb,param_grid=params,n_jobs=-1)
          gs.fit(x_train,y_train)

Out[156… GridSearchCV(estimator=GradientBoostingClassifier(), n_jobs=-1,
                      param_grid={'learning_rate': [0.1, 0.01, 0.001, 0.2, 1, 0.002],
                                  'max_depth': [4, 5, 6, 7, 8, 9],
                                  'min_samples_leaf': [3, 4, 5, 7, 9],
                                  'min_samples_split': [3, 4, 5, 6, 7],
                                  'n_estimators': [100]})

In [158…  gs.best_params_

Out[158… {'learning_rate': 0.2,
          'max_depth': 4,
          'min_samples_leaf': 3,
          'min_samples_split': 4,
          'n_estimators': 100}

In [191…  gb=GradientBoostingClassifier(learning_rate=0.2,max_depth=4,min_samples_leaf=3,min_samples_split=4,n_estimators=100)
```


Predicting Actual Vs Predicted:-

In [195…

```
y_pred=gb.predict(x_test)
```

In [196…

```
Pred=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
```

In [197…

```
Pred
```

Out[197…

	Actual	Predicted
8636	1	1
3092	0	0
310	0	0
7259	1	1
2205	0	0
...	...	...
442	0	0
7571	1	1
935	0	1
5764	0	1
5499	0	0

3099 rows × 2 columns

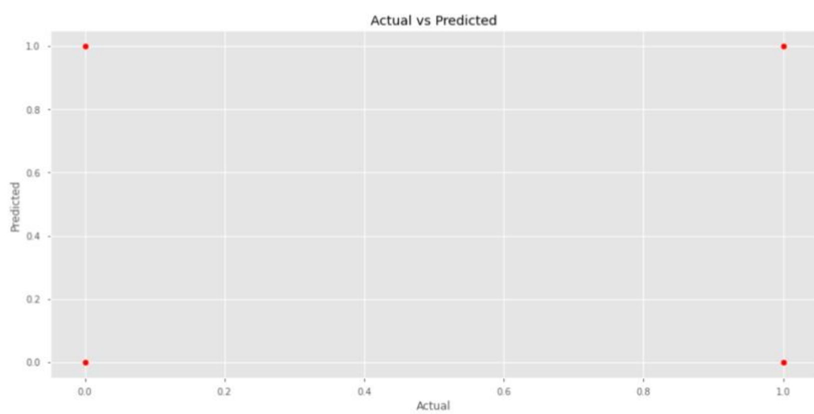
In [198…

```
plt.figure(figsize=(15,7))
sns.scatterplot(y_test,y_pred,color='red')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
```

In [198…

```
plt.figure(figsize=(15,7))
sns.scatterplot(y_test,y_pred,color='red')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
```

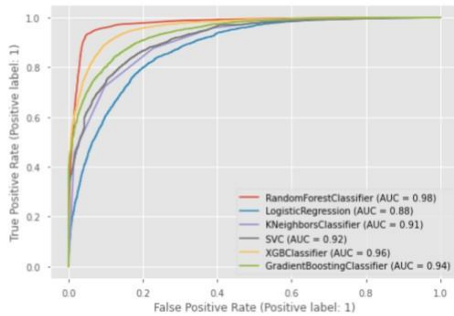
Out[198… Text(0.5, 1.0, 'Actual vs Predicted')



AUC-ROC Curve:-

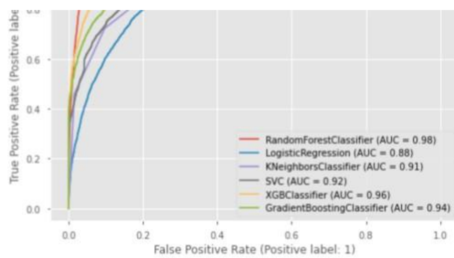
```
In [199…
disp=plot_roc_curve(rf,x,y)
plot_roc_curve(lr,x,y,ax=disp.ax_)
plot_roc_curve(knn,x,y,ax=disp.ax_)
plot_roc_curve(svc,x,y,ax=disp.ax_)
plot_roc_curve(xgb,x,y,ax=disp.ax_)
plot_roc_curve(gb,x,y,ax=disp.ax_)

Out[199… <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x76d37da198>
```




```
In [200…
# Saving the model
import pickle
filename='Telecom_churn'
pickle.dump(rf,open(filename,'wb'))
```

Saving The Model:-




```
In [200…
# Saving the model
import pickle
filename='Telecom_churn'
pickle.dump(rf,open(filename,'wb'))

In [ ]:
```


Conclusion:-

Machine learning is quickly growing field in computer science. It has applications in nearly every other field of study and is already being implemented commercially because machine learning can solve problems too difficult or time consuming for humans to solve. To describe machine learning in general terms, a variety models are used to learn patterns in data and make accurate predictions based on the patterns it observes.