#### **What is Customer Churn?**

Customer churn is defined as when customers or subscribers discontinue doing business with a firm or service. Customers in the telecom industry can choose from a variety of service providers and actively switch from one to the next. The telecommunications business has an annual churn rate of 15-25 percent in this highly competitive market.

To reduce customer churn, telecom companies need to predict which customers are at high risk of churn

#### **Objectives:**

I will explore the data and try to answer some questions like:

* What's the % of Churn Customers and customers that keep in with the active services?
* There any patterns in Churn Customers based on the gender?
* There any patterns/preference in Churn Customers based on the type of service provided?
* What's the most profitable service types?
* Which features and services are most profitable?
* Many more questions that will arise during the analysis

# **Loading libraries and data**

import pandas as pd

import numpy as np

import missingno as msno

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import plotly.graph\_objects as go

from plotly.subplots import make\_subplots

import warnings

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from xgboost import XGBClassifier

from catboost import CatBoostClassifier

from sklearn import metrics

from sklearn.metrics import roc\_curve

from sklearn.metrics import recall\_score, confusion\_matrix, precision\_score, f1\_score, accuracy\_score, classification\_report

Using this libraries we can deal with Data manipulation and Data visualization

*#loading data*

df = pd.read\_csv('Telecom\_customer\_churn\_prediction.csv')

While using the above code we can able to load data from

the dataset

**DATASET LINK:**

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

UNDERSTANDING DATA:

We can understand data by making them visible to us like performing some operation on, we can see them in the following Each row represents a customer, each column contains customer’s attributes described on the column Metadata

**The data set includes information about:**

* **Customers who left within the last month** – the column is called Churn
* **Services that each customer has signed up for** – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
* **Customer account information** - how long they’ve been a customer, contract, payment method, paperless billing, monthly charges, and total charges
* **Demographic info about customers** – gender, age range, and if they have partners and dependents

# to print columns

df.columns.values

array (['customer ID', 'gender', 'Senior Citizen', 'Partner', 'Dependents’

,'tenure', 'Phone Service', 'Multiple Lines', 'Internet Service',

'Online Security', 'Online Backup', 'Device Protection',

'Tech Support', 'Streaming TV', 'Streaming Movies', 'Contract',

'Paperless Billing', 'Payment Method', 'Monthly Charges',

'Total Charges', 'Churn']

are the columns present in the dataset



Understanding data is one of the major task in this prediction

**VISUALIZE MISSING VALUES:**

def missing\_values\_tables(dataframe,na\_name=False):

na\_columns=[col for col in dataframe.columns if dataframe[col].isnull().sum()>0]

n\_miss=dataframe[na\_columns].isnull().sum().sort\_values(ascending=False)

ratio=(dataframe[na\_columns].isnull().sum()/dataframe.shape[0]\*100).sort\_values(ascending=False)

missing\_df=pd.concat([n\_miss,np.round(ratio,2)],axis=1,keys=['n\_miss','ratio'])

print(missing\_df,end="\n")

if na\_name:

return na\_columns

missing\_values\_tables(df)

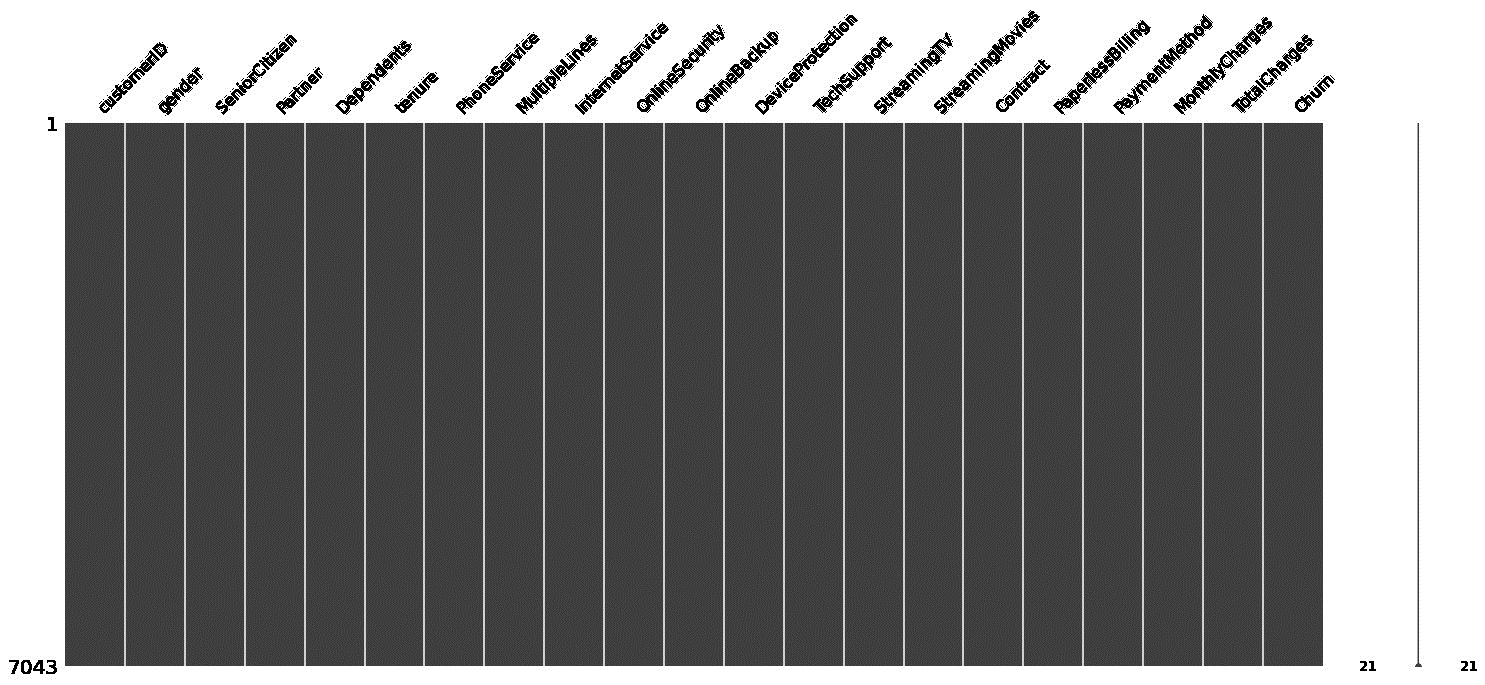
output:

Empty DataFrame

Columns: [n\_miss, ratio]

Index: []

*# Visualize missing values as a matrix*

msno.matrix(df);.

# **Data Manipulation**

df = df.drop(['customerID'], axis = 1)

df.head()

On deep analysis, we can find some indirect missing ness in our data (which can be in form of blank spaces). Let's see that!

df['TotalCharges'] = pd.to\_numeric(df.TotalCharges, errors='coerce')

df.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

Paperless Billing 0

Payment Method 0

Monthly Charges 0

Total Charges 11

Churn 0

d type: int64

Here we see that the Total Charges has 11 missing values. Let's check this data

It can also be noted that the Tenure column is 0 for these entries even though the Monthly Charges column is not empty

d[df['tenure'] == 0]. index

Int64 Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754], d type='int64')

* There are no additional missing values in the Tenure column.

Let's delete the rows with missing values in Tenure columns since there are only 11 rows and deleting them will not affect the data.

df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)

df[df['tenure'] == 0].index

Int64Index([], dtype='int64')

To solve the problem of missing values in Total Charges column, I decided to fill it with the mean of Total Charges values.

numerical\_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']

df[numerical\_cols].describe()

|  | tenure | Monthly Charges | Total Charges |
| --- | --- | --- | --- |
| count | 7032.000000 | 7032.000000 | 7032.000000 |
| mean | 32.421786 | 64.798208 | 2283.300441 |
| std | 24.545260 | 30.085974 | 2266.771362 |
| min | 1.000000 | 18.250000 | 18.800000 |
| 25% | 9.000000 | 35.587500 | 401.450000 |
| 50% | 29.000000 | 70.350000 | 1397.475000 |
| 75% | 55.000000 | 89.862500 | 3794.737500 |
| max | 72.000000 | 118.750000 | 8684.800000 |

df["InternetService"].describe(include=['object', 'bool'])

count 7032

unique 3

top Fiber optic

freq 3096

Name: Internet Service, dtype: object

**Data Visualization**

g\_labels = ['Male', 'Female']

c\_labels = ['No', 'Yes']

*# Create subplots: use 'domain' type for Pie subplot*

fig = make\_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])

fig.add\_trace(go.Pie(labels=g\_labels, values=df['gender'].value\_counts(), name="Gender"),

1, 1)

fig.add\_trace(go.Pie(labels=c\_labels, values=df['Churn'].value\_counts(), name="Churn"),

1, 2)

*# Use `hole` to create a donut-like pie chart*

fig.update\_traces(hole=.4, hoverinfo="label+percent+name", textfont\_size=16)

fig.update\_layout(

title\_text="Gender and Churn Distributions",

*# Add annotations in the center of the donut pies.*

annotations=[dict(text='Gender', x=0.16, y=0.5, font\_size=20, showarrow=False),

dict(text='Churn', x=0.84, y=0.5, font\_size=20, showarrow=False)])

fig.show()

As a result

* 26.6 % of customers switched to another firm.
* Customers are 49.5 % female and 50.5 % male

df["Churn"][df["Churn"]=="No"].groupby(by=df["gender"]).count()

gender

Female 2544

Male 2619

Name: Churn, dtype: int64

df["Churn"][df["Churn"]=="Yes"].groupby(by=df["gender"]).count()

gender

Female 939

Male 930

Name: Churn, dtype: int64

plt.figure(figsize=(6, 6))

labels =["Churn: Yes","Churn:No"]

values = [1869,5163]

labels\_gender = ["F","M","F","M"]

sizes\_gender = [939,930 , 2544,2619]

colors = ['#ff6666', '#66b3ff']

colors\_gender = ['#c2c2f0','#ffb3e6', '#c2c2f0','#ffb3e6']

explode = (0.3,0.3)

explode\_gender = (0.1,0.1,0.1,0.1)

textprops = {"fontsize":15}

*#Plot*

plt.pie(values, labels=labels,autopct='**%1.1f%%**',pctdistance=1.08, labeldistance=0.8,colors=colors, startangle=90,frame=True, explode=explode,radius=10, textprops =textprops, counterclock = True, )

plt.pie(sizes\_gender,labels=labels\_gender,colors=colors\_gender,startangle=90, explode=explode\_gender,radius=7, textprops =textprops, counterclock = True, )

*#Draw circle*

centre\_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)

fig = plt.gcf()

fig.gca().add\_artist(centre\_circle)

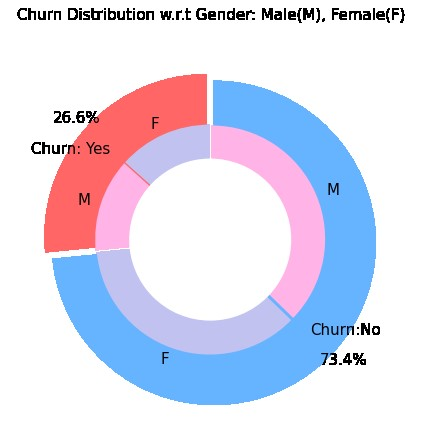
plt.title('Churn Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15, y=1.1)

*# show plot*

plt.axis('equal')

plt.tight\_layout()

plt.show()



There is negligible difference in customer percentage/ count who changed the service provider. Both genders behaved in similar fashion when it comes to migrating to another service provider/firm

fig = px.histogram(df, x="Churn", color="Contract", barmode="group", title="<b>Customer contract distribution<b>")

fig.update\_layout(width=700, height=500, bargap=0.1)

fig.show()

As a result we get

About 75% of customer with Month-to-Month Contract opted to move out as compared to13% of customers with One Year Contract and 3% with Two Year Contract

labels = df['PaymentMethod'].unique()

values = df['PaymentMethod'].value\_counts()

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])

fig.update\_layout(title\_text="<b>Payment Method Distribution</b>")

fig.show()

As a result we get

* Major customers who moved out were having Electronic Check as Payment Method.
* Customers who opted for Credit-Card automatic transfer or Bank Automatic Transfer and Mailed Check as Payment Method were less likely to move out.

#df["InternetService"].unique()

output:

array(['DSL', 'Fiber optic', 'No'], dtype=object)

#df[df["gender"]=="Male"][["InternetService", "Churn"]].value\_counts()

output:

InternetService Churn

DSL No 992

Fiber optic No 910

No No 717

Fiber optic Yes 633

DSL Yes 240

No Yes 57

dtype: int64

#sns.set\_context("paper",font\_scale=1.1)

ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'No') ],

color="Red", shade = True);

ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],

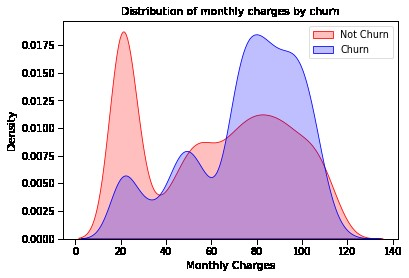
ax =ax, color="Blue", shade= True);

ax.legend(["Not Churn","Churn"],loc='upper right');

ax.set\_ylabel('Density');

ax.set\_xlabel('Monthly Charges');

ax.set\_title('Distribution of monthly charges by churn');



ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'No') ],

color="Gold", shade = True);

ax = sns.kdeplot(df.TotalCharges[(df["Churn"] == 'Yes') ],

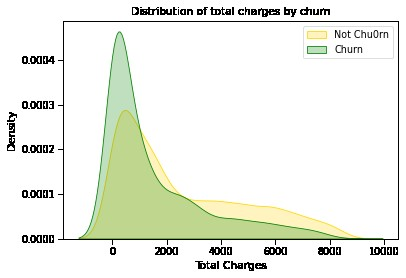
ax =ax, color="Green", shade= True);

ax.legend(["Not Chu0rn","Churn"],loc='upper right');

ax.set\_ylabel('Density');

ax.set\_xlabel('Total Charges');

ax.set\_title('Distribution of total charges by churn');

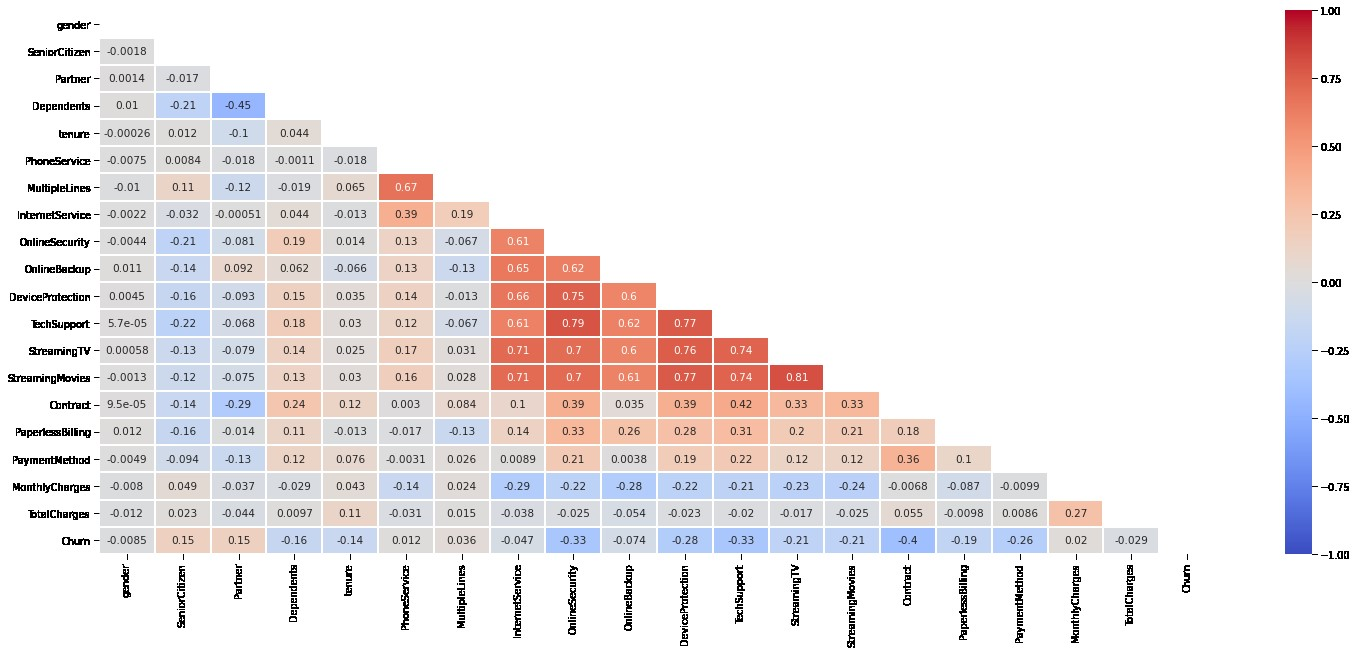


plt.figure(figsize=(25, 10))

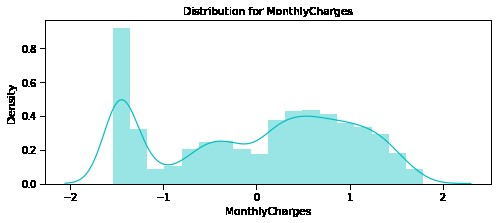
corr = df.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones\_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)



#Distribution of monthly charges



# **Data Pre-processing:**

## What is data pre-processing?

Data pre-processing, a component of [data preparation](https://searchbusinessanalytics.techtarget.com/definition/data-preparation), describes any type of processing performed on [raw data](https://searchdatamanagement.techtarget.com/definition/raw-data) to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the [data mining](https://searchbusinessanalytics.techtarget.com/definition/data-mining) process. More recently, data pre-processing techniques have been adapted for training machine learning models and AI models and for running inferences against them.

Data pre-processing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML) and AI development pipeline to ensure accurate results.

There are several different tools and methods used for pre-processing data, including the following:

* sampling, which selects a representative subset from a large population of data;
* transformation, which manipulates raw data to produce a single input;
* denoising, which removes [noise](https://whatis.techtarget.com/definition/noise) from data;
* imputation, which synthesizes statistically relevant data for missing values;
* [normalization](https://searchsqlserver.techtarget.com/definition/normalization), which organizes data for more efficient access; and
* feature extraction, which pulls out a relevant feature subset that is significant in a particular context.

These tools and methods can be used on a variety of data sources, including data stored in files or databases and streaming data.

## Why is data pre-processing important?

Virtually any type of data analysis, [data science](https://www.techtarget.com/searchenterpriseai/definition/data-science) or AI development requires some type of data pre-processing to provide reliable, precise and robust results for enterprise applications.

Real-world data is messy and is often created, processed and stored by a variety of humans, business processes and applications. As a result, a data set may be missing individual fields, contain manual input errors, or have duplicate data or different names to describe the same thing. Humans can often identify and rectify these problems in the data they use in the line of business, but [data used to train machine learning](https://searchbusinessanalytics.techtarget.com/feature/Data-preparation-in-machine-learning-6-key-steps) or deep learning algorithms needs to be automatically pre-processed.

# Replace missing values with a specific value (e.g., 0)

df.fillna(0, inplace=True)

# Remove rows with missing values

df.dropna(inplace=True)

# Removing Duplicates

df.drop\_duplicates(inplace=True)

**What are the steps involved in data pre-processing?**

* The steps involved in data preprocessing are:
* Data collection, Data cleaning, Data integration, Data transformation, Data reduction, Data discretization, Data normalization or Data standardization, Feature selection, and Data representation.

The insights you gain from your churn prediction models can also be used to enlighten your product teams. The on boarding experience can be fine-tuned and optimized with these key insights.

**Machine Learning Model Evaluations and Predictions:**

* [KNN](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#101)
* [SVC](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#102)
* [Random Forest](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#103)
* [Logistic Regression](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#104)
* [Decision Tree Classifier](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#105)
* [Ada Boost Classifier](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#106)
* [Gradient Boosting Classifier](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#107)
* [Voting Classifier](https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction#108)

#### **KNN**

knn\_model = KNeighborsClassifier(n\_neighbors = 11)

knn\_model.fit(X\_train,y\_train)

predicted\_y = knn\_model.predict(X\_test)

accuracy\_knn = knn\_model.score(X\_test,y\_test)

print("KNN accuracy:",accuracy\_knn)

KNN accuracy: 0.7753554502369668

print(classification\_report(y\_test, predicted\_y))

precision recall f1-score support

0 0.83 0.87 0.85 1549

1 0.59 0.52 0.55 561

accuracy 0.78 2110

macro avg 0.71 0.69 0.70 2110

weighted avg 0.77 0.78 0.77 2110

like this we can use different **Machine Learning Model Evaluations and to Predict the churn**

**ACCURACY:**

y=df['Churn']

X=df.drop(["customerID", "Churn"],axis=1)

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.30,random\_state=17)

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import SGDClassifier, LogisticRegression

from sklearn.metrics import accuracy\_score,precision\_score, recall\_score,f1\_score,roc\_auc\_score

from sklearn.preprocessing import StandardScaler

rf\_model=LogisticRegression()

rf\_model.fit(X\_train,y\_train)

y\_pred=rf\_model.predict(X\_test)

accuracy\_score(y\_pred,y\_test)

**OUTPUT:**

0.7834123222748816

**Missing Values:**

def missing\_values\_tables(dataframe,na\_name=False):

na\_columns=[col for col in dataframe.columns if dataframe[col].isnull().sum()>0]

n\_miss=dataframe[na\_columns].isnull().sum().sort\_values(ascending=False)

ratio=(dataframe[na\_columns].isnull().sum()/dataframe.shape[0]\*100).sort\_values(ascending=False)

missing\_df=pd.concat([n\_miss,np.round(ratio,2)],axis=1,keys=['n\_miss','ratio'])

print(missing\_df,end="\n")

if na\_name:

return na\_columns

missing\_values\_tables(df)

**Output:**

Empty DataFrame

Columns: [n\_miss, ratio]

Index: []

**Excited:**

df["Exited"].value\_counts()

Out[9]:

0 7963

1 2037

Name: Exited, dtype: int64

**Tenure:**

not\_churn["Tenure"].value\_counts().sort\_values()

Out[12]:

0 318

10 389

6 771

9 771

4 786

3 796

5 803

1 803

8 828

2 847

7 851

Name: Tenure, dtype: int64

**Geography:**

not\_churn.Geography.value\_counts().sort\_values()

Out[20]:

Germany 1695

Spain 2064

France 4204

Name: Geography, dtype: int64

**Estimated Salary:**

not\_churn["EstimatedSalary"].describe([0.05,0.25,0.50,0.75,0.90,0.95,0.99])

Out[39]:

count 7963.000000

mean 99738.391772

std 57405.586966

min 90.070000

5% 9773.542000

25% 50783.490000

50% 99645.040000

75% 148609.955000

90% 179453.212000

95% 190107.557000

99% 198131.465200

max 199992.480000

Name: EstimatedSalary, dtype: float64

**Outliers:**

def outlier\_thresholds(dataframe, variable, low\_quantile=0.05, up\_quantile=0.95):

quantile\_one = dataframe[variable].quantile(low\_quantile)

quantile\_three = dataframe[variable].quantile(up\_quantile)

interquantile\_range = quantile\_three - quantile\_one

up\_limit = quantile\_three + 1.5 \* interquantile\_range

low\_limit = quantile\_one - 1.5 \* interquantile\_range

return low\_limit, up\_limit

CreditScore has None Outliers

**CONCLUSION:**

At the end we can predict the churn by using the above given methods and the innovation is to build the complete customer churn prediction model in telecom industry .so use many ML algorithms and using various python libraries we predict the churn and overcome this.