```
#Name: Dimple Kundu
#ML Project TOPIC: Customer Churn Prediction


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")


from google.colab import drive
drive.mount('/content/drive')


     Mounted at /content/drive


# load dataset from google drive

df= pd.read_excel('/content/drive/MyDrive/Mw work/customer_churn_large_dataset.xlsx')


df.head()
```

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthl
0	1	Customer_1	63	Male	Los Angeles	17	
1	2	Customer_2	62	Female	New York	1	
2	3	Customer_3	24	Female	Los Angeles	5	
3	4	Customer_4	36	Female	Miami	3	
4	5	Customer_5	46	Female	Miami	19	

EDA

```
df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 100000 entries, 0 to 99999
     Data columns (total 9 columns):
      #   Column                      Non-Null Count   Dtype
     ---  ------                      --------------   -----
      0   CustomerID                  100000 non-null  int64
      1   Name                        100000 non-null  object
      2   Age                         100000 non-null  int64
      3   Gender                      100000 non-null  object
      4   Location                    100000 non-null  object
      5   Subscription_Length_Months  100000 non-null  int64
      6   Monthly_Bill                100000 non-null  float64
      7   Total_Usage_GB              100000 non-null  int64
```

```
 8   Churn                        100000 non-null  int64
dtypes: float64(1), int64(5), object(3)
memory usage: 6.9+ MB
```

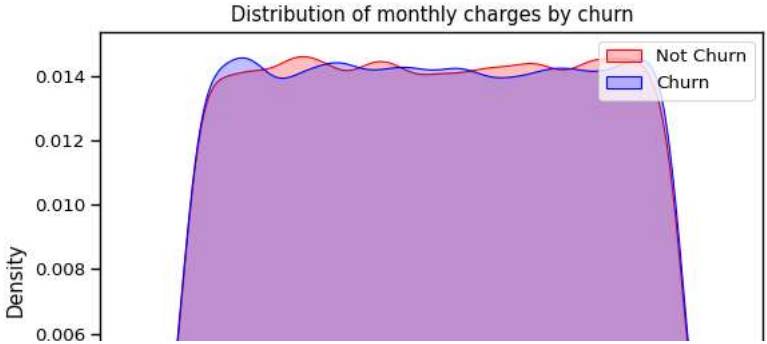
```
df.isnull().sum()#no null values found in the dataset
```

```
CustomerID                    0
Name                          0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
dtype: int64
```

```
df.describe()
```

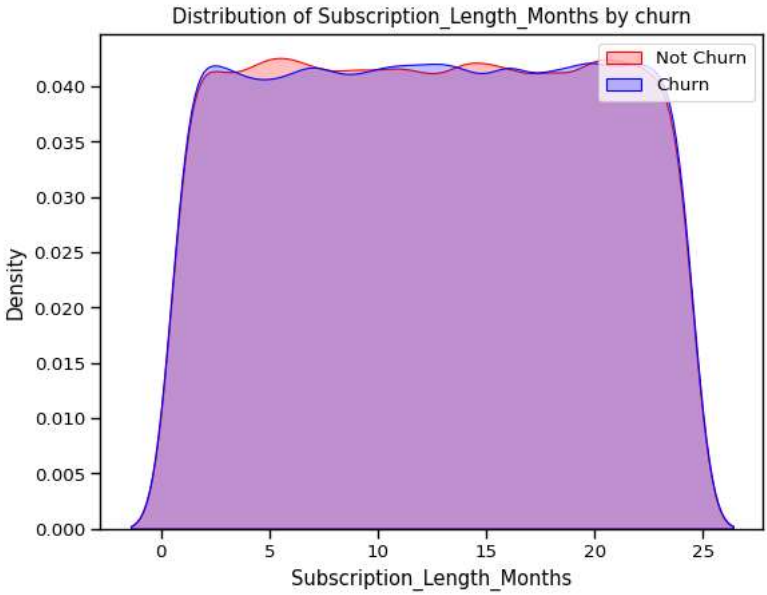
	CustomerID	Age	Subscription_Length_Months	Monthly_Bill	Total_U
count	100000.000000	100000.000000	100000.000000	100000.000000	10000(
mean	50000.500000	44.027020	12.490100	65.053197	27.
std	28867.657797	15.280283	6.926461	20.230696	13(
min	1.000000	18.000000	1.000000	30.000000	5(
25%	25000.750000	31.000000	6.000000	47.540000	16
50%	50000.500000	44.000000	12.000000	65.010000	27.
75%	75000.250000	57.000000	19.000000	82.640000	38
max	100000.000000	70.000000	24.000000	100.000000	50(

```
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(df.Monthly_Bill[(df["Churn"] == 0) ],
                 color="Red", shade = True);
ax = sns.kdeplot(df.Monthly_Bill[(df["Churn"] == 1) ],
                 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');
```



Equal Distribution of higher mothly bill w.r.t. Churn

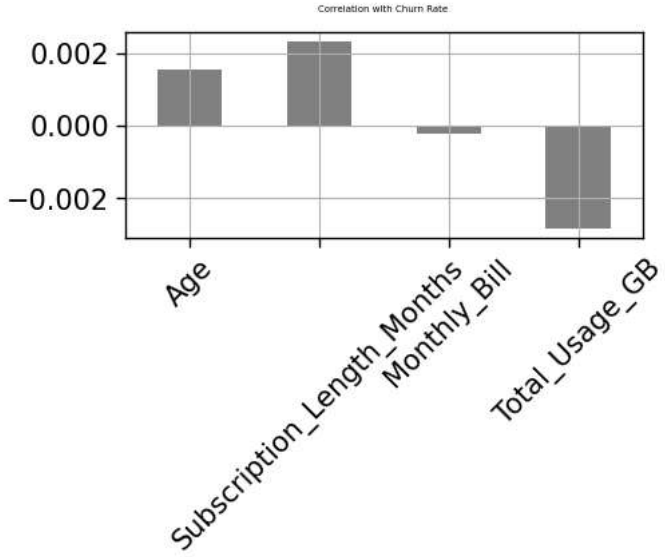
```
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(df.Subscription_Length_Months[(df["Churn"] == 0) ],
                 color="Red", shade = True);
ax = sns.kdeplot(df.Subscription_Length_Months[(df["Churn"] == 1) ],
                 ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Subscription_Length_Months');
ax.set_title('Distribution of Subscription_Length_Months by churn');
```



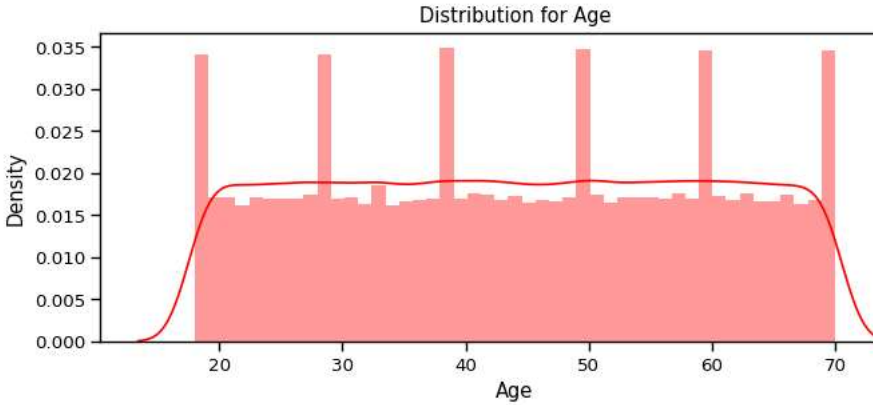
Equal Distribution of higher Subscription_Length_Months w.r.t. Churn

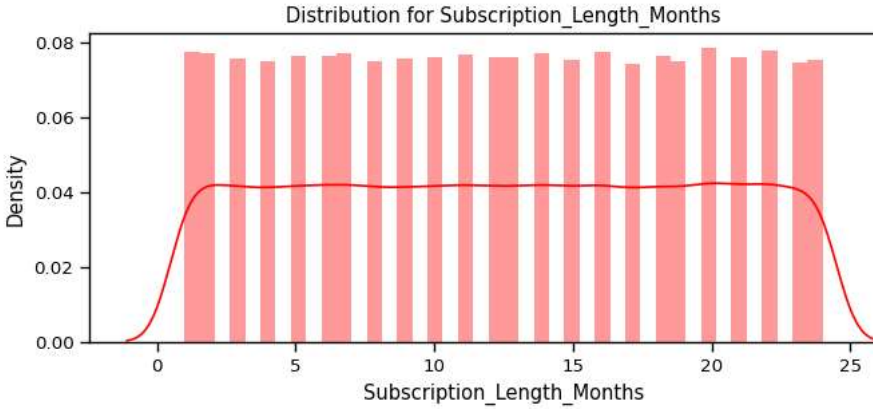
```
data2 = df[['Age',  'Subscription_Length_Months',   'Monthly_Bill', 'Total_Usage_GB'    ]]

correlations = data2.corrwith(df.Churn)
correlations = correlations[correlations!=1]
positive_correlations = correlations[correlations >0].sort_values(ascending = False)
negative_correlations =correlations[correlations<0].sort_values(ascending = False)

correlations.plot.bar(
        figsize = (5, 2),
        fontsize = 15,
        color = 'grey',
        rot = 45, grid = True)
plt.title('Correlation with Churn Rate \n',
horizontalalignment="center", fontstyle = "normal",
fontsize = "5", fontfamily = "sans-serif")
```

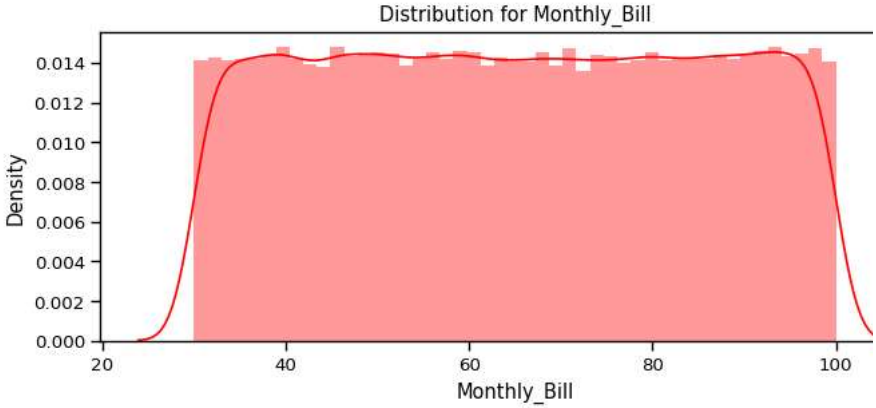
```
Text(0.5, 1.0, 'Correlation with Churn Rate \n')
```

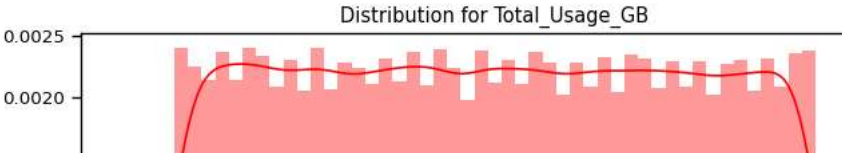


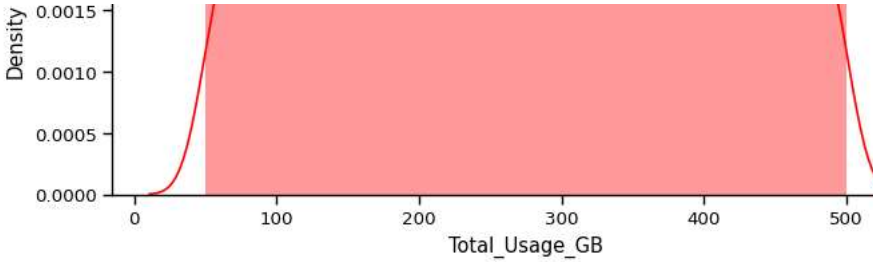

```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)


col =  ["Age", 'Subscription_Length_Months', 'Monthly_Bill','Total_Usage_GB']
for features in col :distplot(features, df)
```


Distribution for Age
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
30
40
50
60
70
Age


Distribution for Subscription_Length_Months
Density
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
Subscription_Length_Months

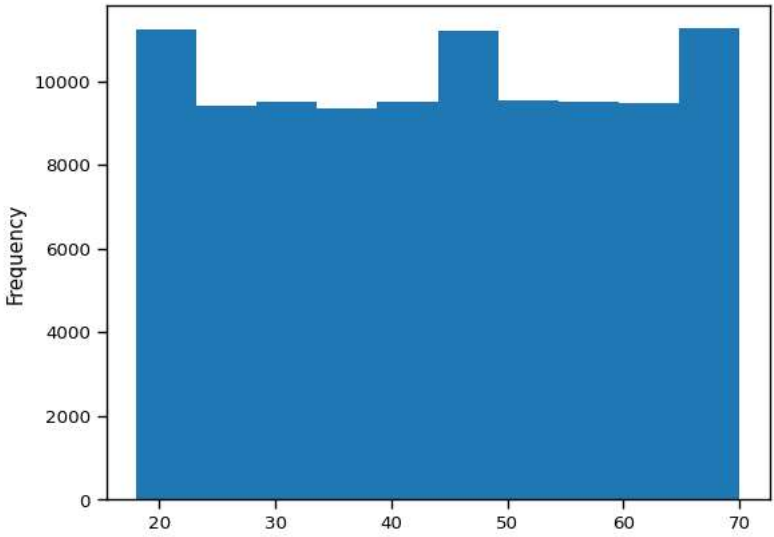

Distribution for Monthly_Bill
Density
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
20
40
60
80
100
Monthly_Bill


Distribution for Total_Usage_GB
0.0025
0.0020




```
#Histogram: frequency count chart
#I approach
df['Age'].plot(kind='hist',bins=10)
```

```
<Axes: ylabel='Frequency'>
```

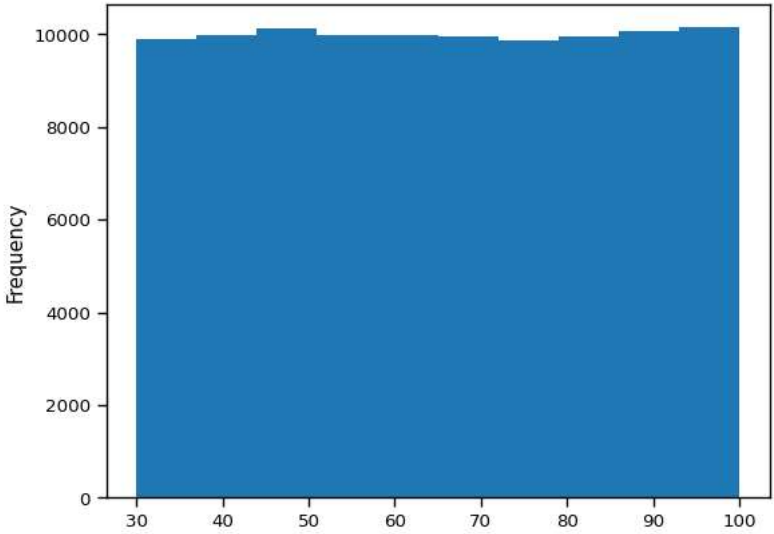



```
df['Subscription_Length_Months'].plot(kind='hist',bins=5)
```

```
<Axes: ylabel='Frequency'>
```


```
df['Monthly_Bill'].plot(kind='hist',bins=10)
```

```
<Axes: ylabel='Frequency'>
```




```
df['Total_Usage_GB'].plot(kind='hist',bins=10)
```

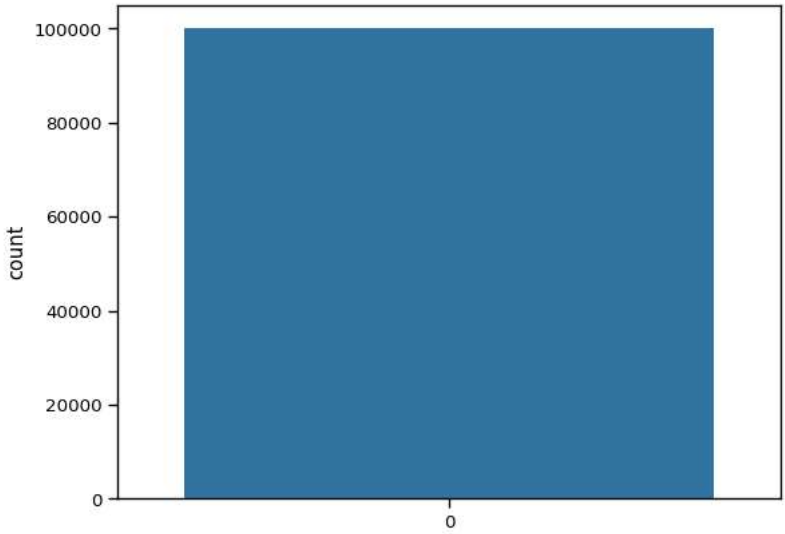
```
<Axes: ylabel='Frequency'>
```


```
df['Churn'].value_counts()
```

```
0    50221
1    49779
Name: Churn, dtype: int64
```

```
sns.countplot(df['Churn'])
```

```
<Axes: ylabel='count'>
```



50% stayed 50% left

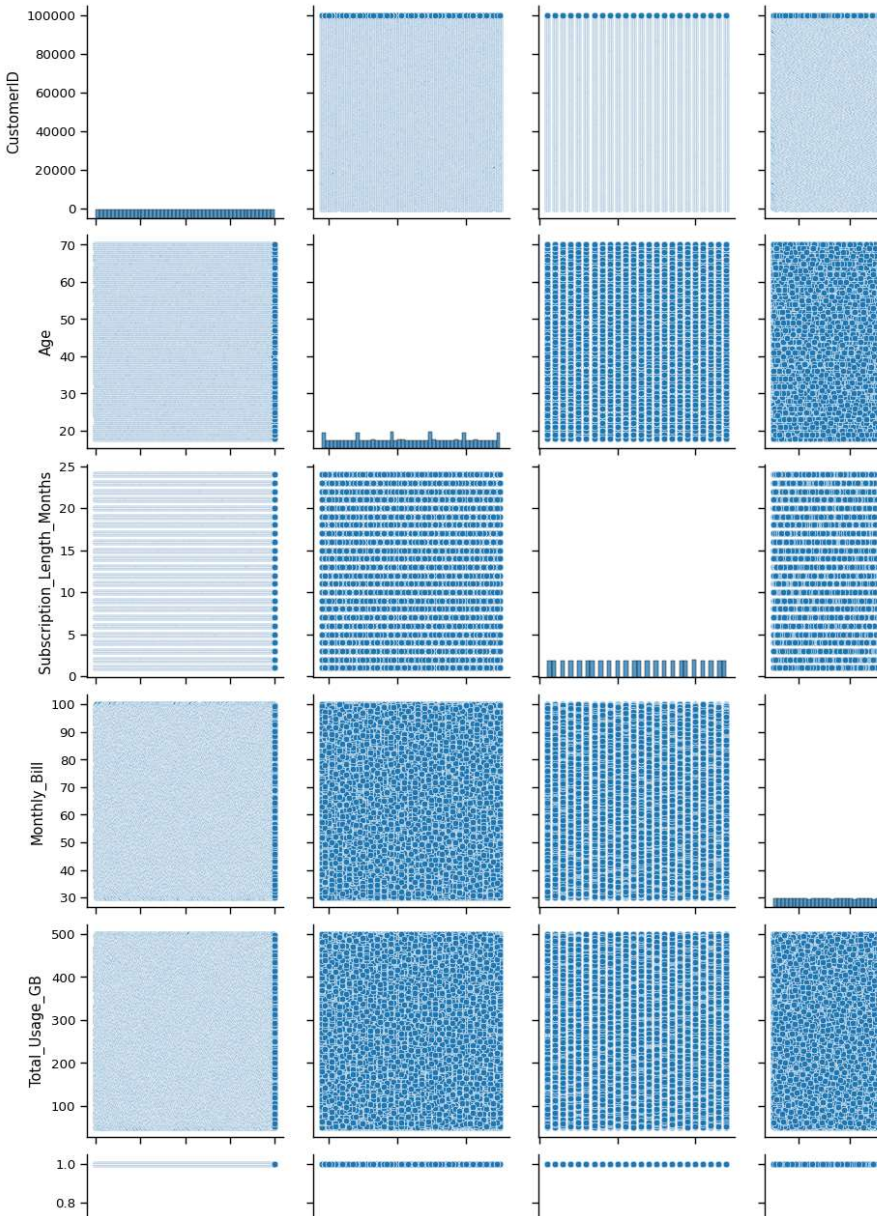
```
col=df.columns
```

```
col
```

```
Index(['CustomerID', 'Name', 'Age', 'Gender', 'Location',
       'Subscription_Length_Months', 'Monthly_Bill', 'Total_Usage_GB',
       'Churn'],
      dtype='object')
```

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7dbd75452d10>
```



1.CLEAN THE DATA | REPLACE| IMPUTE

DATA PREPROCESSING:

1.Encoding categorical data

> 2.Spliting dataset into features and outcomes

> 3.feature scaling

ENCODING- LabelEncoder

```
df.dtypes
```

```
CustomerID                     int64
Name                          object
Age                            int64
Gender                        object
Location                      object
Subscription_Length_Months     int64
Monthly_Bill                 float64
Total_Usage_GB                 int64
Churn                          int64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
le=LabelEncoder()
sc=StandardScaler()
```

```
# fit and transform the data
# we have to read data from columns and overwrite on same column

df.Gender= le.fit_transform(df.Gender)

df.Location= le.fit_transform(df.Location)
```

CHECK: df.dtypes while Label Encoding

```
df.dtypes#TotalCharges converts from object to int 64
```

```
CustomerID                     int64
Name                          object
Age                            int64
Gender                         int64
Location                       int64
Subscription_Length_Months     int64
Monthly_Bill                 float64
Total_Usage_GB                 int64
Churn                          int64
dtype: object
```

```
df.head(10)
```

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthly
0	1	Customer_1	63	1	2	17	
1	2	Customer_2	62	0	4	1	
2	3	Customer_3	24	0	2	5	
3	4	Customer_4	36	0	3	3	
4	5	Customer_5	46	0	3	19	
5	6	Customer_6	67	1	4	15	
6	7	Customer_7	30	0	0	3	
7	8	Customer_8	67	0	3	1	
8	9	Customer_9	20	0	3	10	
9	10	Customer_10	53	0	2	12	

Perform feature engineering

```
df['Age_Group'] = pd.cut(df['Age'], bins=[0, 30, 50, float('inf')], labels=['Young', 'Middl

# location_dummies = pd.get_dummies(df['Location'], prefix='Location')
# df = pd.concat([df, location_dummies], axis=1)

df['Subscription_Cost'] = df['Monthly_Bill'] / df['Subscription_Length_Months']
```

```
df['SeniorCitizen'] =( df['Age'] >=60)
```

```
df.SeniorCitizen=le.fit_transform(df['SeniorCitizen'])
df.SeniorCitizen=sc.fit_transform(df[['SeniorCitizen']])
```

```
df.head()
```

me	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Chu
_1	63	1	2	0.651115	0.410606	-0.294289	
_2	62	0	4	-1.658879	-0.805374	-0.784852	
_3	24	0	2	-1.081380	1.009204	1.422681	
_4	36	0	3	-1.370129	1.625597	0.173279	
_5	46	0	3	0.939864	-0.341720	-0.064338	

Divide the dataset into features and outcome

```
Cost'],'Monthly_Bill': df['Monthly_Bill'],'Total_Usage_GB': df['Total_Usage_GB'],'SeniorCiti


for features in x :distplot(features, df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

```
df.head()
```

me	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Chu
_1	63	1	2	0.651115	0.410606	-0.294289	
_2	62	0	4	-1.658879	-0.805374	-0.784852	
_3	24	0	2	-1.081380	1.009204	1.422681	
_4	36	0	3	-1.370129	1.625597	0.173279	
_5	46	0	3	0.939864	-0.341720	-0.064338	

```
x.head()
```

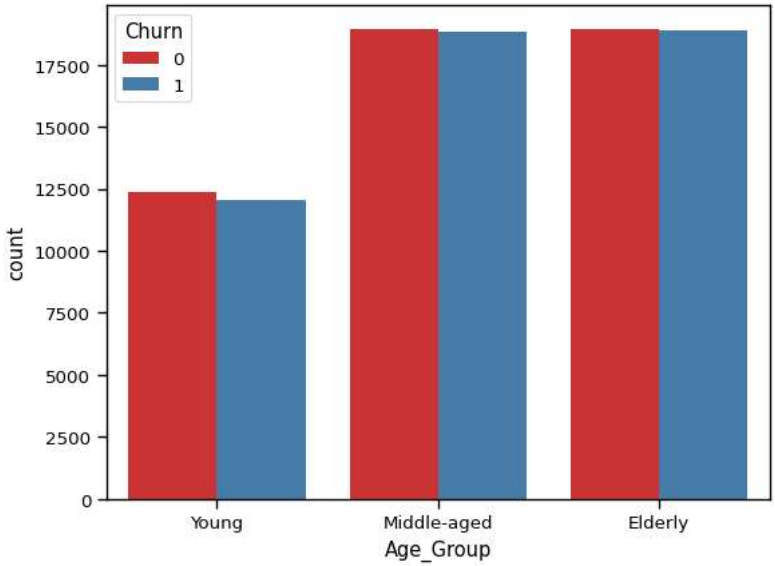
	Gender	Age_Group	Subscription_Cost	Monthly_Bill	Total_Usage_GB	SeniorCitizen
0	1	0	-0.413328	0.410606	-0.294289	1.956141
1	0	0	2.650731	-0.805374	-0.784852	1.956141
2	0	2	0.467648	1.009204	1.422681	-0.511211
3	0	1	1.539863	1.625597	0.173279	-0.511211
4	0	1	-0.499869	-0.341720	-0.064338	-0.511211


```
sns.countplot(data=df, x='Age_Group', hue='Churn', palette='Set1')
```

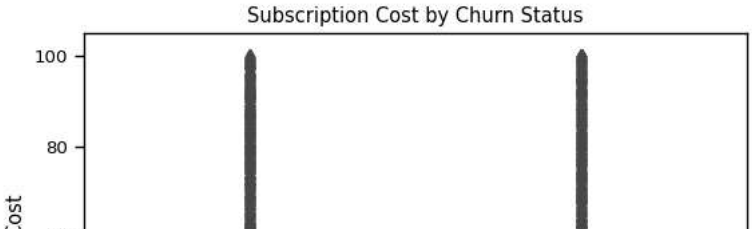
```
<Axes: xlabel='Age_Group', ylabel='count'>
```

```
sns.countplot(data=df, x='Age_Group', hue='Churn', palette='Set1')
```

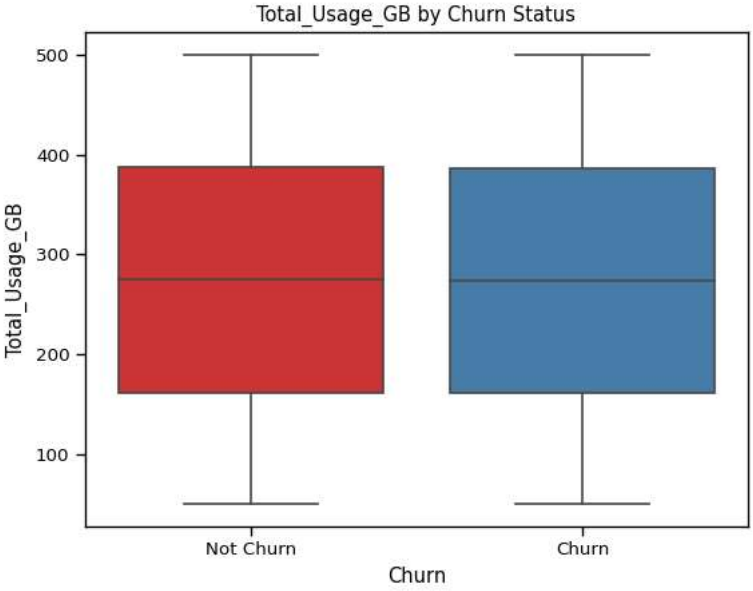
```
<Axes: xlabel='Age_Group', ylabel='count'>
```




```
sns.boxplot(data=df, x='Churn', y='Subscription_Cost', palette='Set1')
# Set plot labels and title
plt.xlabel('Churn')
plt.ylabel('Subscription Cost')
plt.title('Subscription Cost by Churn Status')

# Show the plot
plt.xticks([0, 1], ['Not Churn', 'Churn'])
plt.show()
```

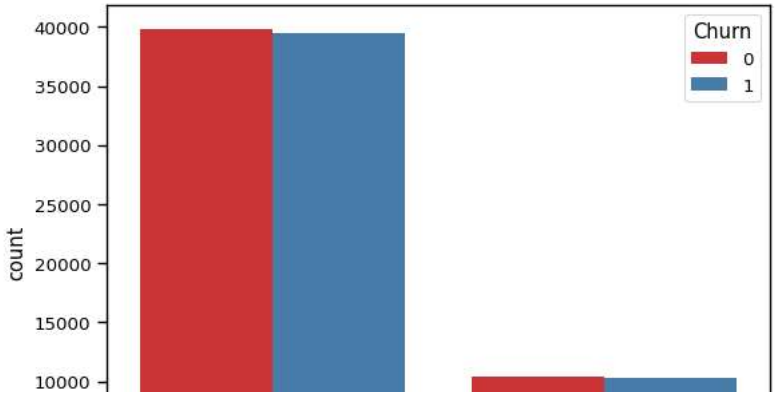



```
sns.boxplot(data=df, x='Churn', y='Total_Usage_GB', palette='Set1')
# Set plot labels and title
plt.xlabel('Churn')
plt.ylabel('Total_Usage_GB')
plt.title('Total_Usage_GB by Churn Status')

# Show the plot
plt.xticks([0, 1], ['Not Churn', 'Churn'])
plt.show()
```




```
sns.countplot(data=df, x='SeniorCitizen', hue='Churn', palette='Set1')
```

```
<Axes: xlabel='SeniorCitizen', ylabel='count'>
```




```
y.head()
```

```
0    0
1    0
2    0
3    1
4    0
Name: Churn, dtype: int64
```

```
df.head()
```

Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Churn	Age_Group
2	0.651115	0.410606	-0.294289	0	0
4	-1.658879	-0.805374	-0.784852	0	0
2	-1.081380	1.009204	1.422681	0	2
3	-1.370129	1.625597	0.173279	1	1
3	0.939864	-0.341720	-0.064338	0	1

Data Normalization- MinMaxScaler StandardScaler

```
from sklearn.preprocessing import LabelEncoder,MinMaxScaler,StandardScaler
le=LabelEncoder()
ms=MinMaxScaler()
ss=StandardScaler()
```

```
df.Gender=le.fit_transform(df['Gender'])
df.Location=le.fit_transform(df['Location'])
df.Age_Group=le.fit_transform(df['Age_Group'])
```

```
df.Subscription_Cost=ss.fit_transform(df[['Subscription_Cost']])
df.Subscription_Length_Months=ss.fit_transform(df[['Subscription_Length_Months']])
df.Monthly_Bill=ss.fit_transform(df[['Monthly_Bill']])
df.Total_Usage_GB=ss.fit_transform(df[['Total_Usage_GB']])


# df.Age=ms.fit_transform(df[['Age']])
# df.Subscription_Length_Months=ss.fit_transform(df[['Subscription_Length_Months']])
# df.Monthly_Bill=ss.fit_transform(df[['Monthly_Bill']])
# df.Total_Usage_GB=ss.fit_transform(df[['Total_Usage_GB']])



# df.Subscription_Length_Months = pd.DataFrame(A, columns=['Age'])
# df.Subscription_Length_Months = pd.DataFrame(SLM, columns=['Subscription_Length_Months'])
# df.Monthly_Bill = pd.DataFrame(MB, columns=['Monthly_Bill'])
# df.Total_Usage_GB = pd.DataFrame(TUGB, columns=['Total_Usage_GB'])


x = pd.DataFrame({ 'Subscription_Cost':df['Subscription_Cost'],'Subscription_Length_Months':
= pd.DataFrame({ 'Subscription_Cost':df['Subscription_Cost'],'Total_Usage_GB': df['Total_Usa


x.head()
```

	Subscription_Cost	Total_Usage_GB	SeniorCitizen
0	-0.413328	-0.294289	1.956141
1	2.650731	-0.784852	1.956141
2	0.467648	1.422681	-0.511211
3	1.539863	0.173279	-0.511211
4	-0.499869	-0.064338	-0.511211

```
ynew=pd.DataFrame({'Churn':df['Churn']})
```

Double-click (or enter) to edit

DIVIDE THE DATA INTO train test split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,ynew,test_size=.05,random_state=10)


x_test
```

	Subscription_Cost	Total_Usage_GB	SeniorCitizen
33226	-0.445233	-0.769522	1.956141
64804	2.677618	0.817142	-0.511211
39763	-0.366506	-1.405720	-0.511211
51270	-0.533230	0.863133	-0.511211
9698	-0.299251	0.219269	1.956141
...	...	...	...
83522	0.744860	-0.815512	-0.511211
94118	-0.434754	-1.543691	-0.511211
34725	0.148176	1.445676	-0.511211

y_test

	Churn
33226	0
64804	0
39763	1
51270	1
9698	0
...	...
83522	1
94118	0
34725	1
48336	0
80166	0

5000 rows × 1 columns

x_train

	Subscription_Cost	Total_Usage_GB	SeniorCitizen
37703	-0.469075	-0.094998	-0.511211
1749	-0.467204	0.127288	-0.511211
35211	0.852752	0.801812	1.956141
68305	-0.213817	-0.148653	-0.511211
92097	-0.304723	0.571861	-0.511211

```
y_train.head()
```

	Churn
37703	1
1749	0
35211	0
68305	0
92097	1

```
# from sklearn.linear_model import LogisticRegression
# model=LogisticRegression()
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100, random_state=42)
```

```
model.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
pred=model.predict(x_test)
pred
```

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

```
#  NOW  to do **side by side comparision**
# for this **create a copy of testing feature **
```

```
result=x_test.copy()
result
```

	Subscription_Cost	Total_Usage_GB	SeniorCitizen
33226	-0.445233	-0.769522	1.956141
64804	2.677618	0.817142	-0.511211
39763	-0.366506	-1.405720	-0.511211
51270	-0.533230	0.863133	-0.511211
9698	-0.299251	0.219269	1.956141
...	...	...	...
83522	0.744860	-0.815512	-0.511211
94118	-0.434754	-1.543691	-0.511211

```
# we need to add 2 columns || Actual || Prediction
result['Actual'] = y_test
result['Prediction'] = pred
```

```
result['Prediction'].head(15)
```

```
33226    0
64804    0
39763    1
51270    0
9698     1
5948     0
27955    0
55001    0
50875    1
47755    0
29430    0
32953    1
6808     1
32927    1
64298    0
Name: Prediction, dtype: int64
```

```
result.head(20)
```

	Subscription_Cost	Total_Usage_GB	SeniorCitizen	Actual	Prediction
33226	-0.445233	-0.769522	1.956141	0	0
64804	2.677618	0.817142	-0.511211	0	0
39763	-0.366506	-1.405720	-0.511211	1	1
51270	-0.533230	0.863133	-0.511211	1	0
9698	-0.299251	0.219269	1.956141	0	1
5948	2.990610	-0.301954	-0.511211	0	0
27955	-0.592699	1.131409	-0.511211	1	0
55001	-0.338706	-0.309619	-0.511211	1	0
50875	-0.396162	-0.999473	-0.511211	1	1
47755	-0.410092	-0.049008	1.956141	0	0
29430	-0.521612	-0.240634	-0.511211	0	0

MODEL PREDICTION

confusion_matrix,precision_score,recall_score,f1_score,accuracy_score

```
from sklearn.metrics import confusion_matrix,precision_score,recall_score,f1_score,accuracy_

confusion_matrix(y_test,pred)
```

```
array([[1314, 1256],
       [1204, 1226]])
```

```
print("precision_score: ",precision_score(y_test,pred))
```

```
precision_score:  0.4939564867042707
```

```
print("recall_score: ",recall_score(y_test,pred))
```

```
recall_score:  0.5045267489711934
```

✓ 0s completed at 7:33 PM