Importing Libraries Dataset Information

```
#importing the common libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

To get the exact path and make it as executed.

```
#importing the dataset
df = pd.read_csv(r'C:\Users\Vijay\Desktop\WA_Fn_UseC_Telco-Customer-Churn.csv')
```

To show by using the head() check that all the columns is in a form.

```
df.head()
```

```
   customerID  gender  SeniorCitizen Partner Dependents  tenure
PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1
No
1  5575-GNVDE    Male              0      No         No      34
Yes
2  3668-QPYBK    Male              0      No         No       2
Yes
3  7795-CFOCW    Male              0      No         No      45
No
4  9237-HQITU  Female              0      No         No       2
Yes

      MultipleLines InternetService OnlineSecurity  ...
DeviceProtection  \
0  No phone service             DSL             No  ...
No
1                No             DSL            Yes  ...
Yes
2                No             DSL            Yes  ...
No
3  No phone service             DSL            Yes  ...
Yes
4                No     Fiber optic             No  ...
No

  TechSupport StreamingTV StreamingMovies        Contract
PaperlessBilling  \
0          No          No              No  Month-to-month
Yes
```

```
1          No          No              No        One year
No
2          No          No              No  Month-to-month
Yes
3         Yes          No              No        One year
No
4          No          No              No  Month-to-month
Yes

              PaymentMethod MonthlyCharges  TotalCharges Churn
0          Electronic check          29.85         29.85    No
1              Mailed check          56.95       1889.50    No
2              Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4          Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

Whether we check the Null value or not.

```
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
```

2. Exploratory Data Analysis

2.1 Automated EDA (Pandas Profiling)

```
from pandas_profiling import ProfileReport
profile = ProfileReport(df, title='Pandas Profiling Report')
df.profile_report()
```

{"version_major":2,"version_minor":0,"model_id":"aa323848d0f145339f018b060b075e0c"}

{"version_major":2,"version_minor":0,"model_id":"0409c404e33f4255a96ed2ff36552a6c"}

{"version_major":2,"version_minor":0,"model_id":"5fa413a3786f46dfb475d5b01527d90d"}

```
<IPython.core.display.HTML object>
```

using the shape function rows and columns.

2.2 Performing EDA

```
#checking how many rows and columns are present
df.shape
```

```
(7043, 21)
```

```
#getting to know the column names
df.columns
```

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner',
'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

```
for col in df.columns:
    print(col, ":", len(df[col].unique()),'labels')
```

```
customerID : 7043 labels
gender : 2 labels
SeniorCitizen : 2 labels
Partner : 2 labels
Dependents : 2 labels
tenure : 73 labels
PhoneService : 2 labels
```

```
MultipleLines : 3 labels
InternetService : 3 labels
OnlineSecurity : 3 labels
OnlineBackup : 3 labels
DeviceProtection : 3 labels
TechSupport : 3 labels
StreamingTV : 3 labels
StreamingMovies : 3 labels
Contract : 3 labels
PaperlessBilling : 2 labels
PaymentMethod : 4 labels
MonthlyCharges : 1585 labels
TotalCharges : 6531 labels
Churn : 2 labels
```

```python
#creating a function to create a table that has feature_name, dtype, missing values and the number of unique values
def insights_table(df):
    summary = pd.DataFrame(df.dtypes,columns=['dtypes'])
    summary = summary.reset_index()
    summary['Feature_name'] = summary['index']
    summary = summary[['Feature_name','dtypes']]
    summary['Missing_values'] = df.isnull().sum().values
    summary['No. Uniques_values'] = df.nunique().values
    return summary
insights_table(df)
```

	Feature_name	dtypes	Missing_values	No. Uniques_values
0	customerID	object	0	7043
1	gender	object	0	2
2	SeniorCitizen	int64	0	2
3	Partner	object	0	2
4	Dependents	object	0	2
5	tenure	int64	0	73
6	PhoneService	object	0	2
7	MultipleLines	object	0	3
8	InternetService	object	0	3
9	OnlineSecurity	object	0	3
10	OnlineBackup	object	0	3
11	DeviceProtection	object	0	3
12	TechSupport	object	0	3
13	StreamingTV	object	0	3
14	StreamingMovies	object	0	3
15	Contract	object	0	3
16	PaperlessBilling	object	0	2
17	PaymentMethod	object	0	4
18	MonthlyCharges	float64	0	1585
19	TotalCharges	object	0	6531
20	Churn	object	0	2

Observation: Missing Data - Initial Intuition

Here, we don't have any missing data.

```
df.describe(include="all")
```

```
        customerID gender  SeniorCitizen Partner Dependents  
tenure  \
count         7043   7043    7043.000000    7043       7043   
7043.000000   
unique        7043      2            NaN       2          2   
NaN   
top     7590-VHVEG   Male            NaN      No         No   
NaN   
freq             1   3555            NaN    3641       4933   
NaN   
mean           NaN    NaN       0.162147     NaN        NaN   
32.371149   
std            NaN    NaN       0.368612     NaN        NaN   
24.559481   
min            NaN    NaN       0.000000     NaN        NaN   
0.000000   
25%            NaN    NaN       0.000000     NaN        NaN   
9.000000   
50%            NaN    NaN       0.000000     NaN        NaN   
29.000000   
75%            NaN    NaN       0.000000     NaN        NaN   
55.000000   
max            NaN    NaN       1.000000     NaN        NaN   
72.000000   

       PhoneService MultipleLines InternetService OnlineSecurity  ...
\
count          7043          7043            7043           7043  ...

unique            2             3               3              3  ...

top             Yes            No     Fiber optic             No  ...

freq           6361          3390            3096           3498  ...

mean            NaN           NaN             NaN            NaN  ...

std             NaN           NaN             NaN            NaN  ...

min             NaN           NaN             NaN            NaN  ...

25%             NaN           NaN             NaN            NaN  ...

50%             NaN           NaN             NaN            NaN  ...
```

```
75%              NaN           NaN              NaN            NaN  ...

max              NaN           NaN              NaN            NaN  ...


       DeviceProtection TechSupport StreamingTV StreamingMovies  \
count              7043        7043        7043            7043
unique                3           3           3               3
top                  No          No          No              No
freq               3095        3473        2810            2785
mean                NaN         NaN         NaN             NaN
std                 NaN         NaN         NaN             NaN
min                 NaN         NaN         NaN             NaN
25%                 NaN         NaN         NaN             NaN
50%                 NaN         NaN         NaN             NaN
75%                 NaN         NaN         NaN             NaN
max                 NaN         NaN         NaN             NaN

              Contract PaperlessBilling     PaymentMethod
MonthlyCharges  \
count             7043             7043              7043
7043.000000
unique               3                2                 4
NaN
top     Month-to-month              Yes  Electronic check
NaN
freq              3875             4171              2365
NaN
mean               NaN              NaN               NaN
64.761692
std                NaN              NaN               NaN
30.090047
min                NaN              NaN               NaN
18.250000
25%                NaN              NaN               NaN
35.500000
50%                NaN              NaN               NaN
70.350000
75%                NaN              NaN               NaN
89.850000
max                NaN              NaN               NaN
118.750000

        TotalCharges Churn
count           7043  7043
unique          6531     2
top                     No
freq              11  5174
mean             NaN   NaN
```

```
std                NaN   NaN
min                NaN   NaN
25%                NaN   NaN
50%                NaN   NaN
75%                NaN   NaN
max                NaN   NaN

[11 rows x 21 columns]
```

Observation from the descriptive statistics

- Senior citizen column is the form of 0's and 1's, here the distribution is not proper
- Tenure : -->Average tenure is around less than 32 months -->25% customers have a tenure of less than 9 months -->50% customers have a tenure of less than 29 months -->75% customers have a tenure of less than 55months -->maximum customers have a tenure of less than 72months
- Monthly Charges : -->Average monthly charges is USD 64.76

The dataset has too many features with text data and are probably categorical features.

Removing customer ID, as it is unique to every record.

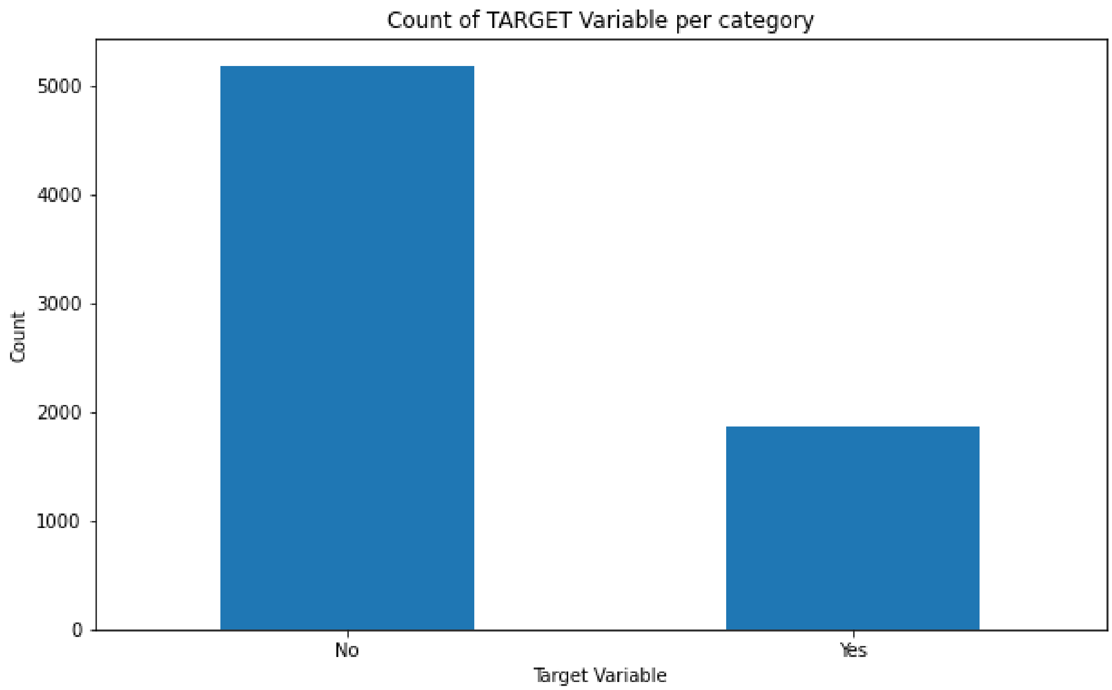
```
df.drop(columns = ['customerID'], inplace = True)
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
Input In [44], in <cell line: 1>()
----> 1 df.drop(columns = ['customerID'], inplace = True)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\
_decorators.py:311, in
deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*arg
s, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\
frame.py:4954, in DataFrame.drop(self, labels, axis, index, columns,
level, inplace, errors)
   4806 @deprecate_nonkeyword_arguments(version=None,
allowed_args=["self", "labels"])
   4807 def drop(
   4808     self,
   (...)
```

```
   4815     errors: str = "raise",
   4816 ):
   4817     """
   4818     Drop specified labels from rows or columns.
   4819 
   (...)
   4952             weight  1.0     0.8
   4953     """
-> 4954     return super().drop(
   4955         labels=labels,
   4956         axis=axis,
   4957         index=index,
   4958         columns=columns,
   4959         level=level,
   4960         inplace=inplace,
   4961         errors=errors,
   4962     )

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\
generic.py:4267, in NDFrame.drop(self, labels, axis, index, columns,
level, inplace, errors)
   4265 for axis, labels in axes.items():
   4266     if labels is not None:
-> 4267         obj = obj._drop_axis(labels, axis, level=level,
errors=errors)
   4269 if inplace:
   4270     self._update_inplace(obj)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\
generic.py:4311, in NDFrame._drop_axis(self, labels, axis, level,
errors, consolidate, only_slice)
   4309         new_axis = axis.drop(labels, level=level,
errors=errors)
   4310     else:
-> 4311         new_axis = axis.drop(labels, errors=errors)
   4312     indexer = axis.get_indexer(new_axis)
   4314 # Case for non-unique axis
   4315 else:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\
base.py:6644, in Index.drop(self, labels, errors)
   6642 if mask.any():
   6643     if errors != "ignore":
-> 6644         raise KeyError(f"{list(labels[mask])} not found in
axis")
   6645     indexer = indexer[~mask]
   6646 return self.delete(indexer)

KeyError: "['customerID'] not found in axis"
```

Understanding the Target variable

```
df['Churn'].value_counts().plot.bar(figsize=(10, 6), rot = 0)
plt.ylabel("Count")
plt.xlabel("Target Variable")
plt.title("Count of TARGET Variable per category");
```




```
(df['Churn'].value_counts()/len(df['Churn']))*100
```

```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

Observation:

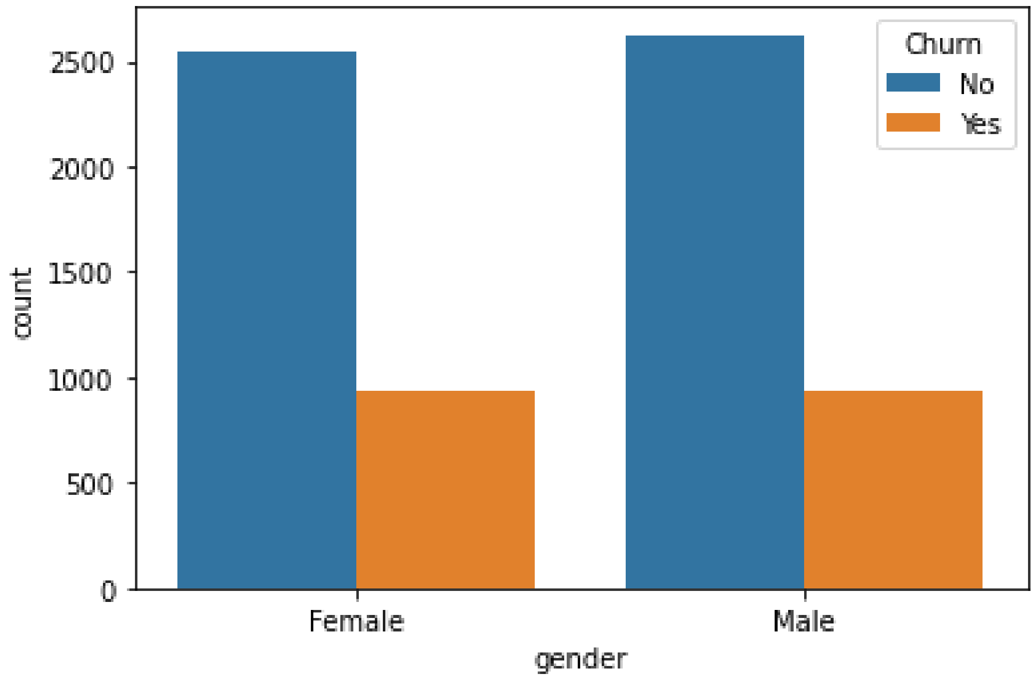
We see that our target variables are imbalanced(73%:27% ratio)

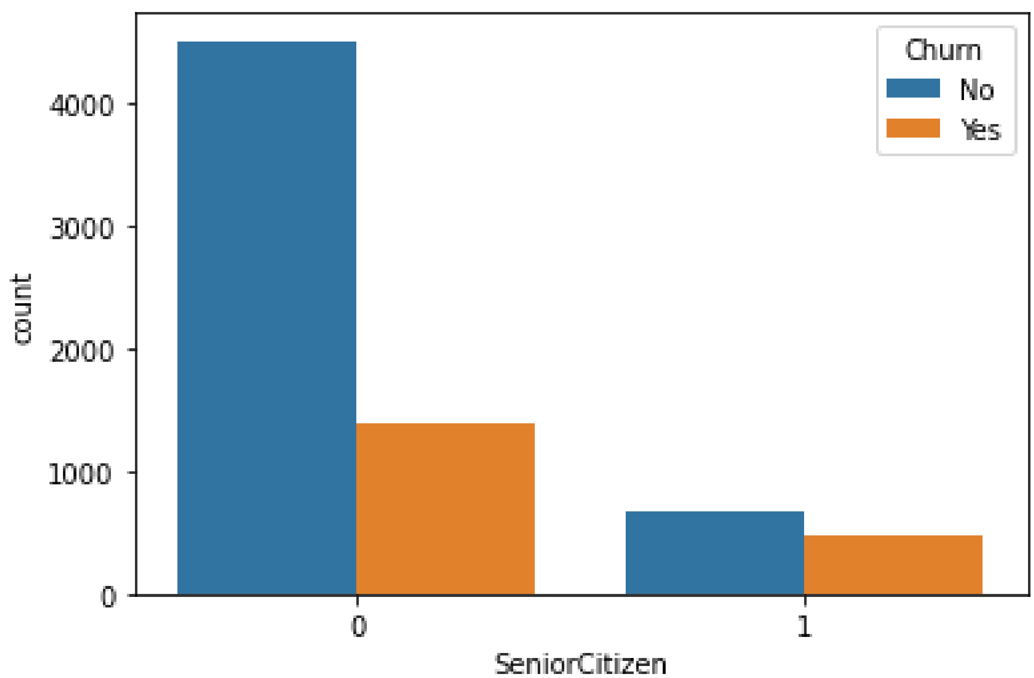
Univariate Analysis with Churn

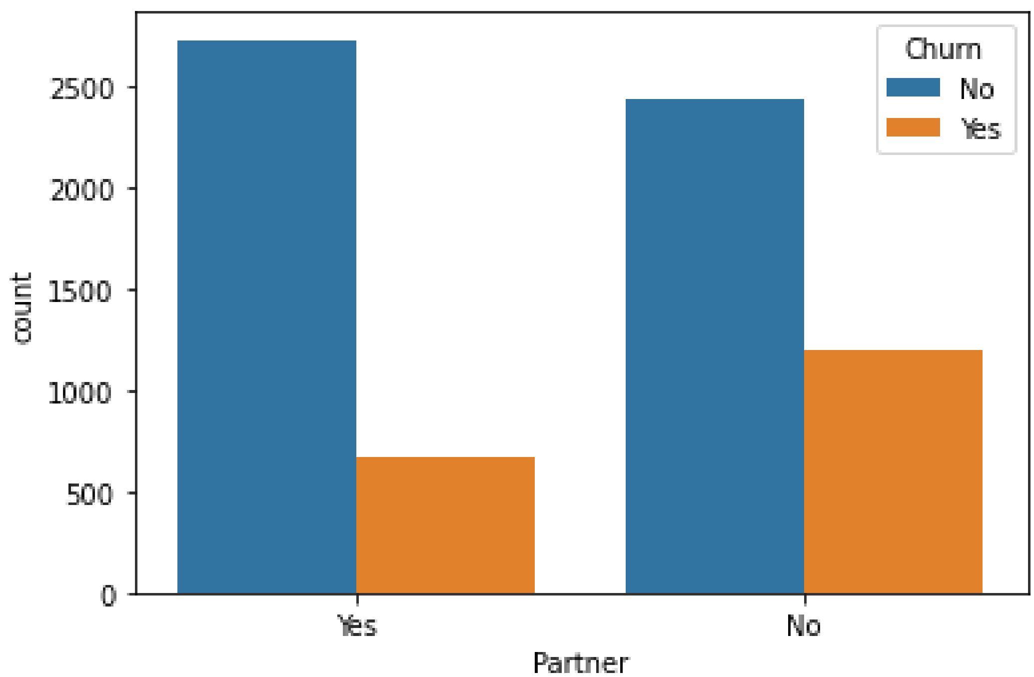
```
df1 = df.drop(columns = ['Churn','TotalCharges', 'MonthlyCharges'])
df1.columns
```

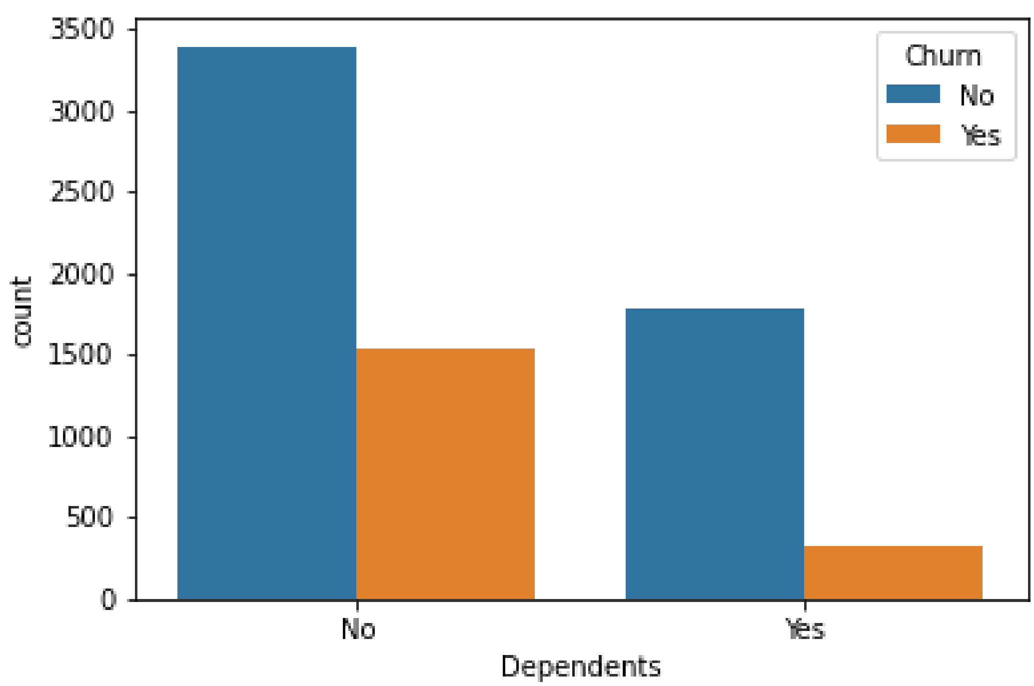
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling',
'PaymentMethod'],
      dtype='object')
```

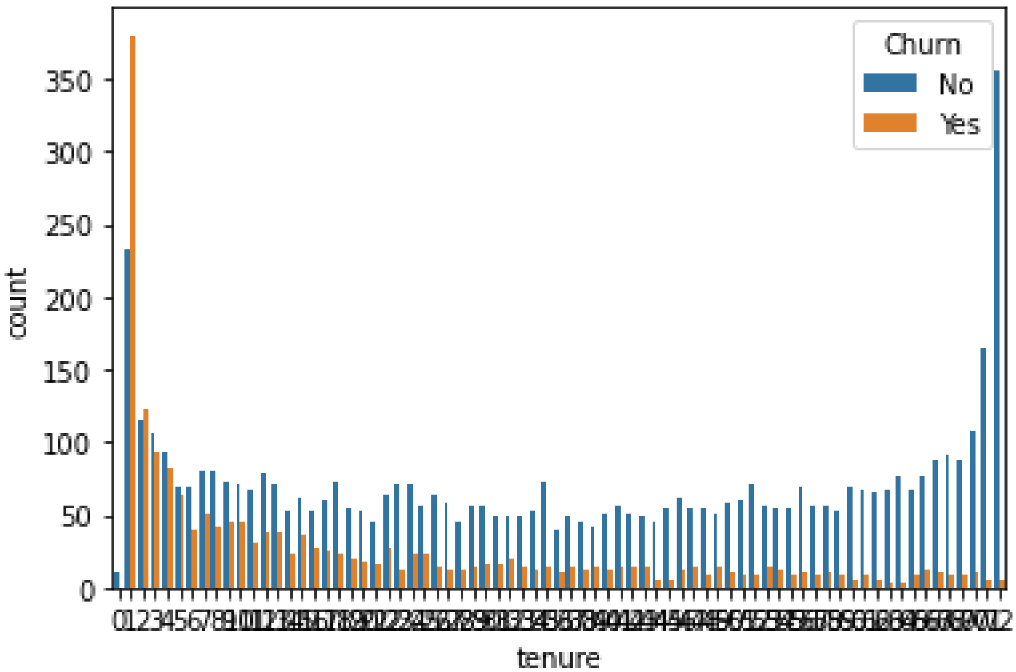
```python
for i in df1:
    plt.figure(i)
    sns.countplot(data=df, x = i, hue='Churn')
```



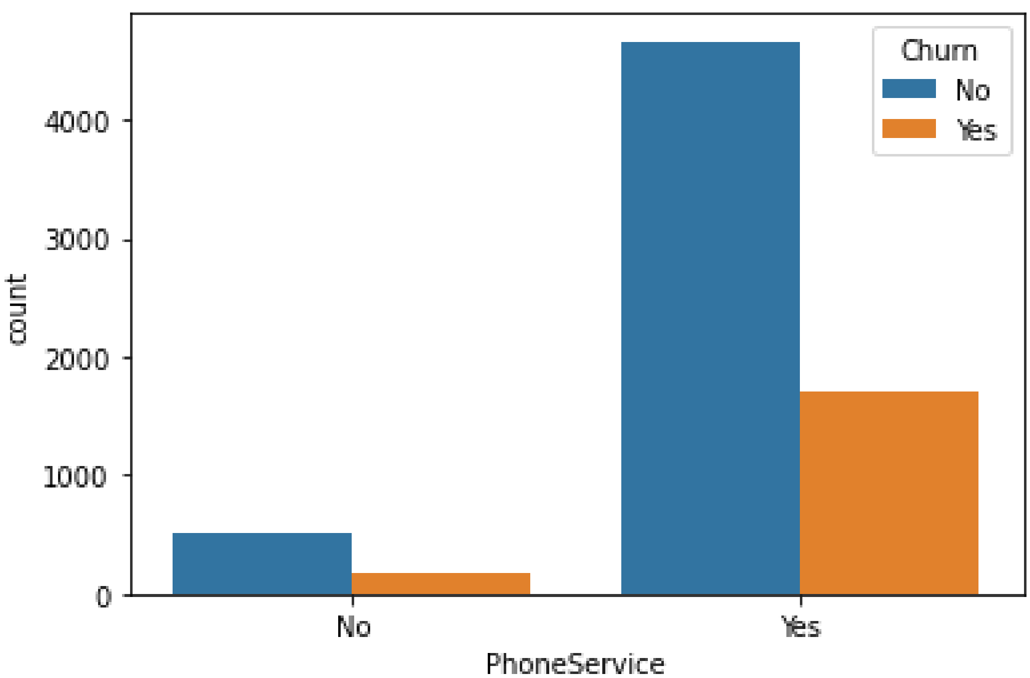



Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
Partner

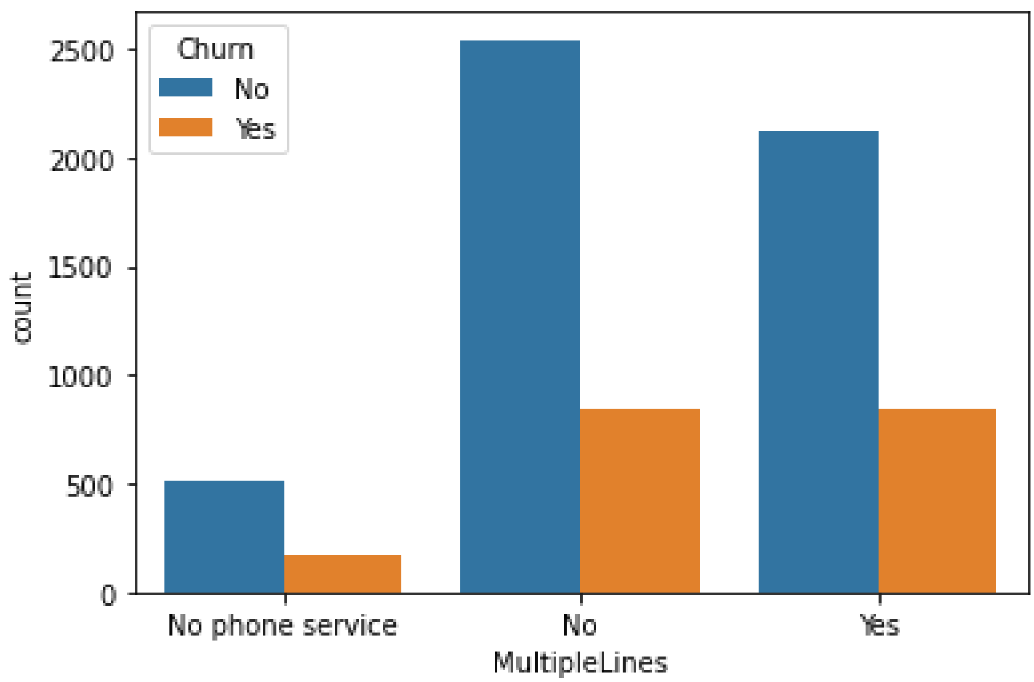

Churn
No
Yes
3500
3000
2500
2000
1500
1000
500
0
count
No
Yes
Dependents



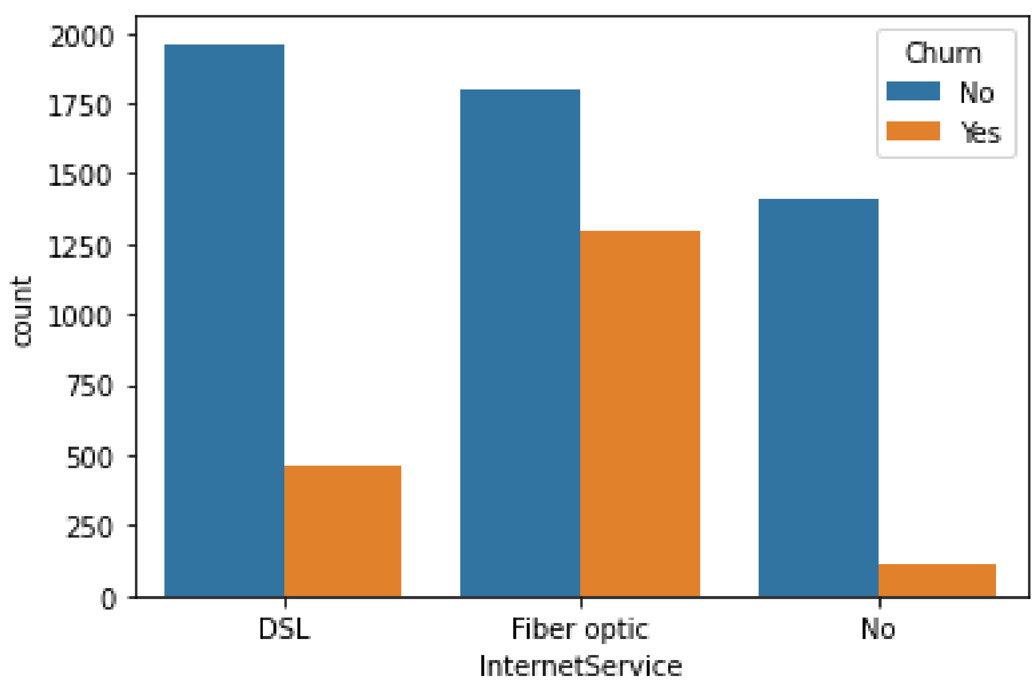
Churn
No
Yes
count
tenure



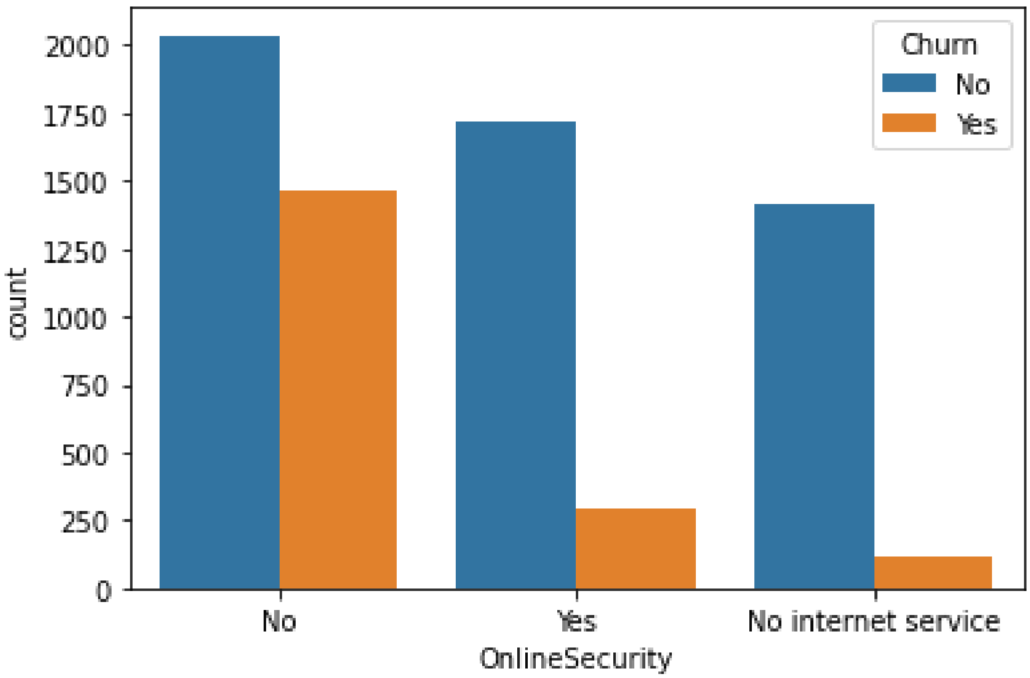
Churn
No
Yes
count
0
1000
2000
3000
4000
No
Yes
PhoneService

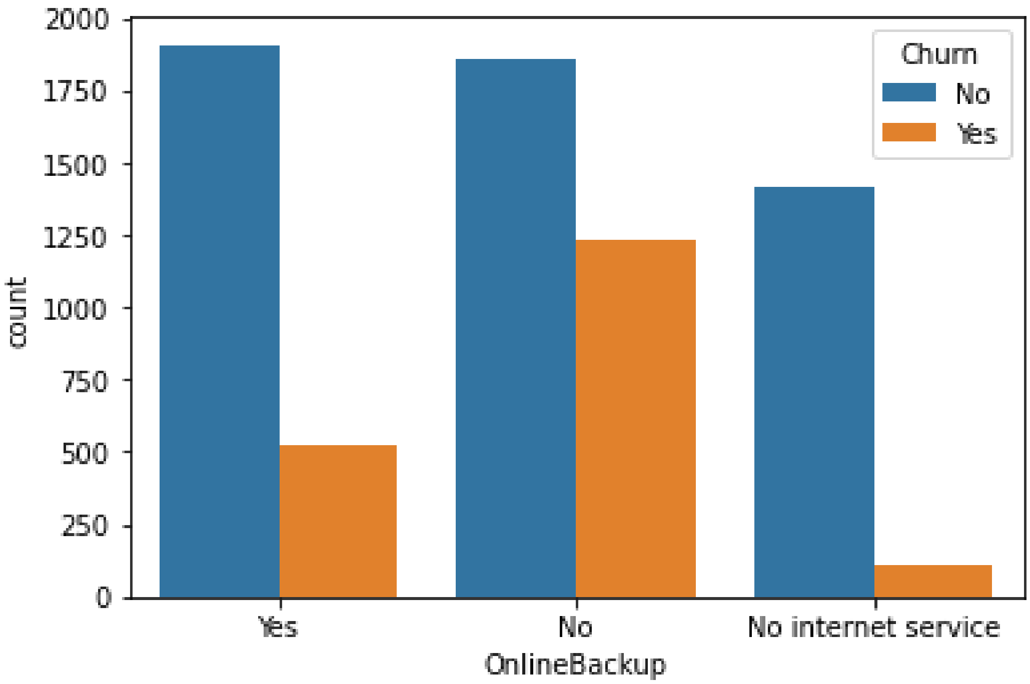


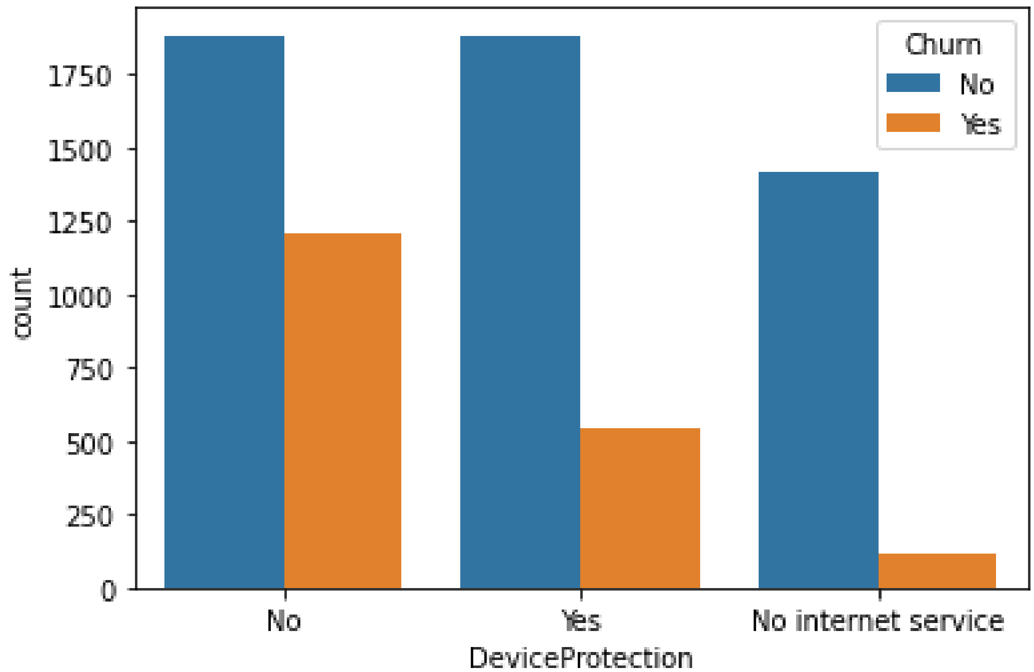
Churn
No
Yes
count
MultipleLines
No phone service
No
Yes

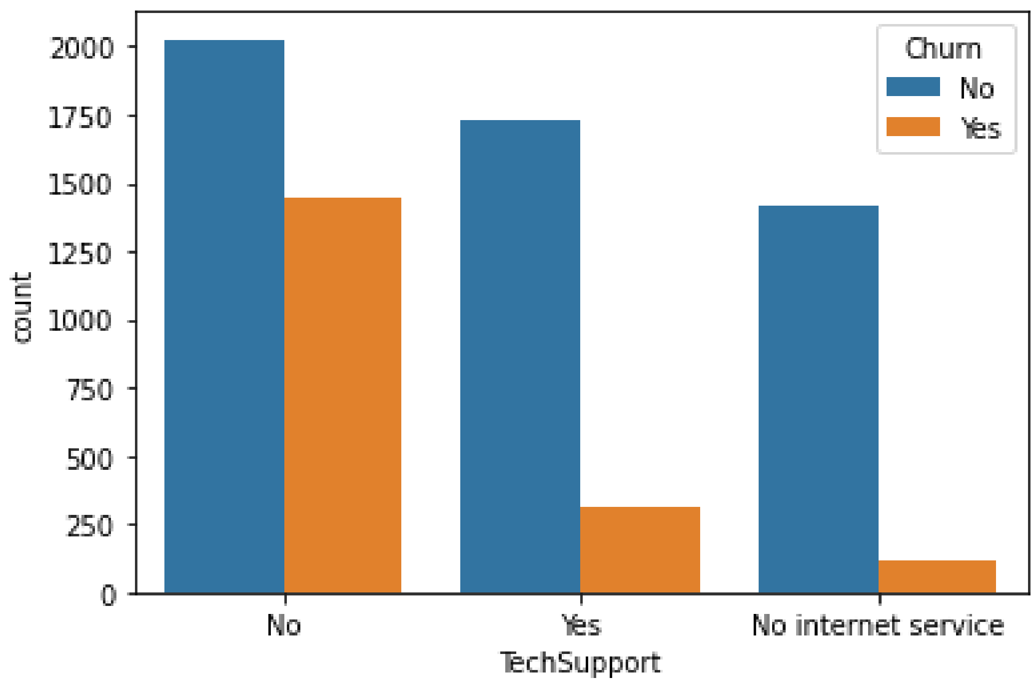


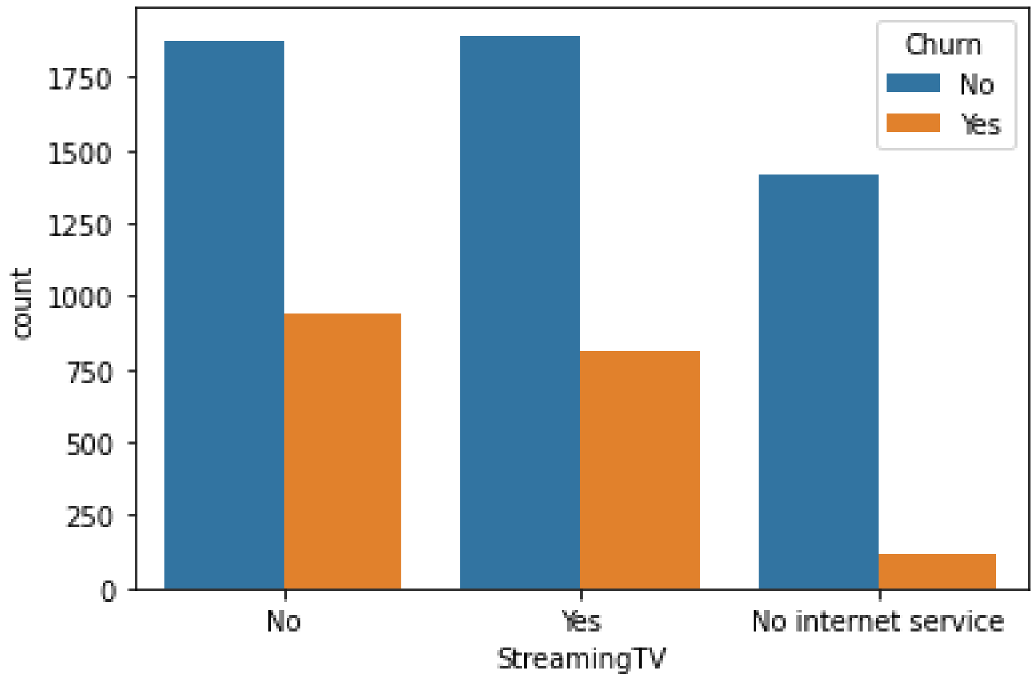
Churn
No
Yes
count
InternetService
DSL
Fiber optic
No

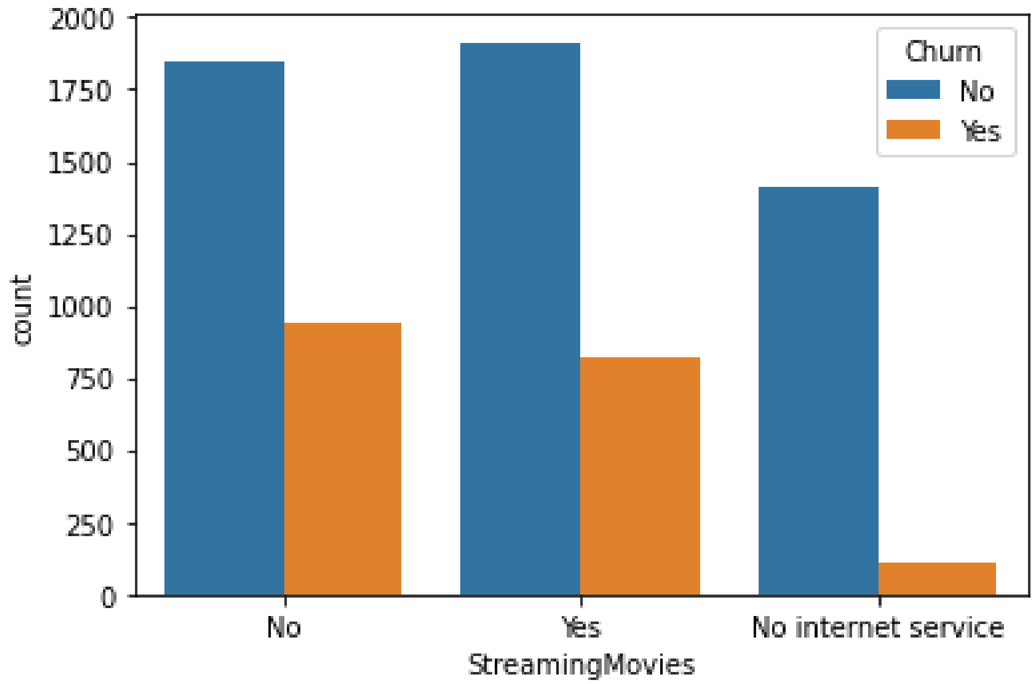

Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
OnlineSecurity

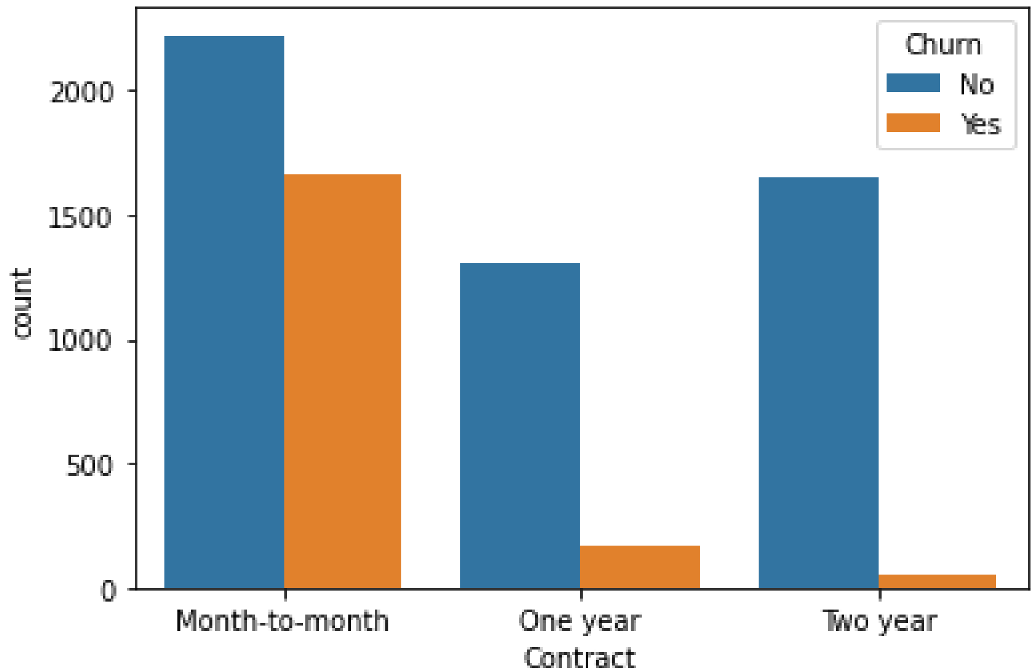

Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
Yes
No
No internet service
OnlineBackup

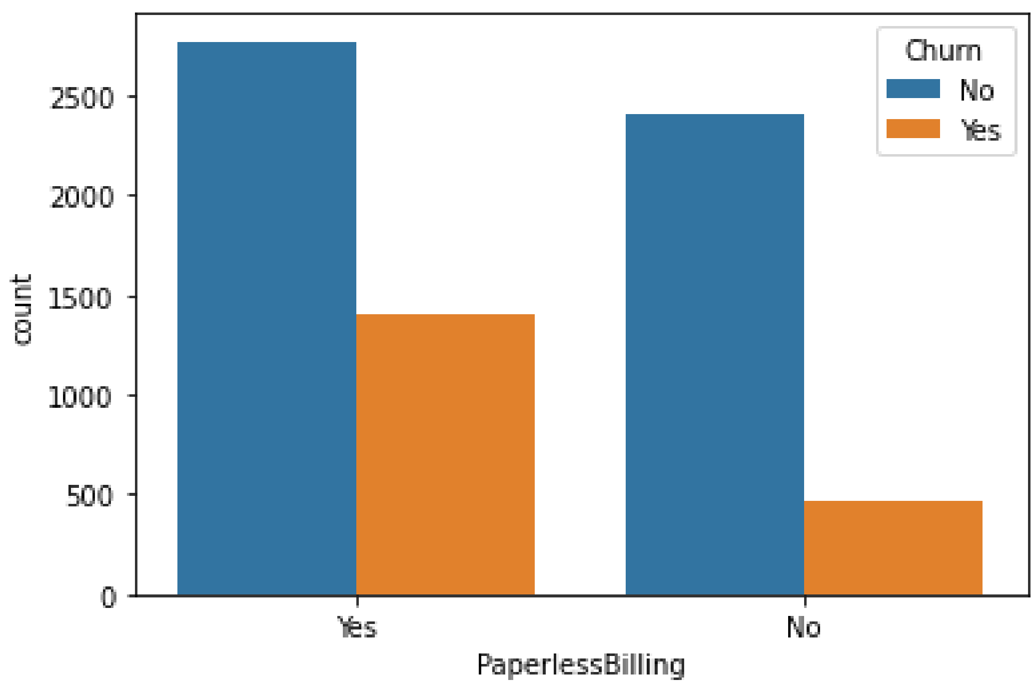

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
DeviceProtection

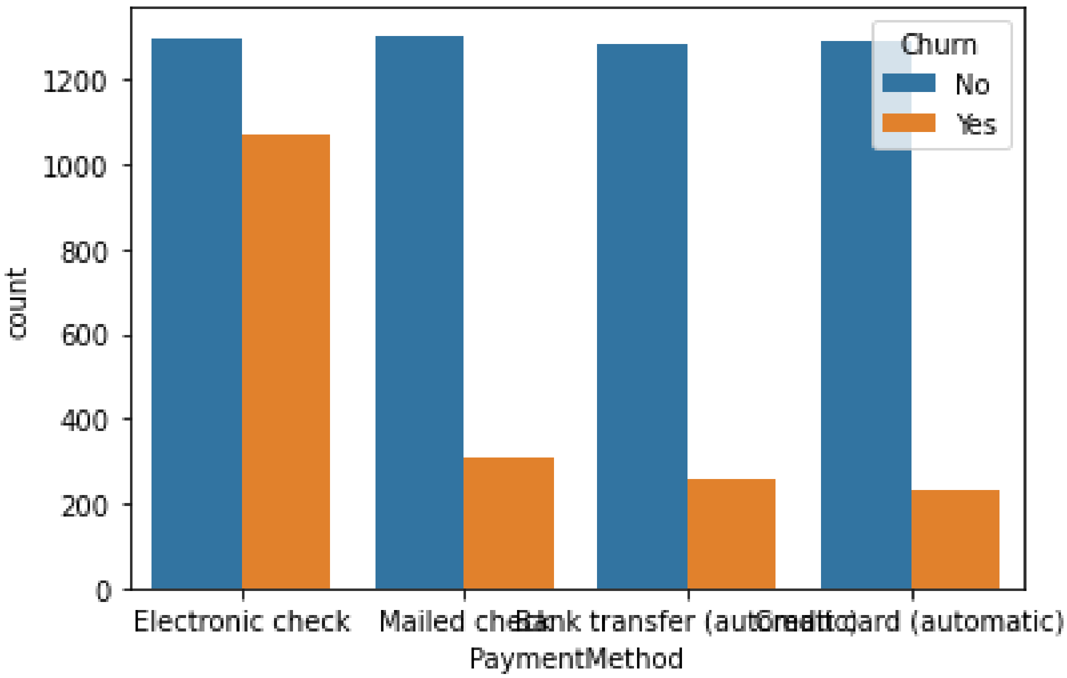

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
TechSupport


Churn
No
Yes
count
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
StreamingTV


Churn
No
Yes
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
StreamingMovies


Churn
No
Yes
count
2000
1500
1000
500
0
Month-to-month
One year
Two year
Contract


Churn
No
Yes
count
2500
2000
1500
1000
500
0
Yes
No
PaperlessBilling



3. Feature Engineering (Data Preprocessing)

```
new_df = df.copy(deep = True)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

new_df = df.copy(deep = True)
text_data_features = [i for i in list(df.columns) if i not in
list(df.describe())]
text_data_features.remove('TotalCharges')
# removing TotalCharges here because it is a float varible but given
as object, so we shouldnt do label encoding for it

print('Label Encoder Transformation\n')
for i in text_data_features :
    new_df[i] = le.fit_transform(new_df[i])
    print(i,' : ',new_df[i].unique(),' =
',le.inverse_transform(new_df[i].unique()))
```

```
Label Encoder Transformation

gender  :  [0 1]  =  ['Female' 'Male']
Partner  :  [1 0]  =  ['Yes' 'No']
Dependents  :  [0 1]  =  ['No' 'Yes']
PhoneService  :  [0 1]  =  ['No' 'Yes']
MultipleLines  :  [1 0 2]  =  ['No phone service' 'No' 'Yes']
InternetService  :  [0 1 2]  =  ['DSL' 'Fiber optic' 'No']
OnlineSecurity  :  [0 2 1]  =  ['No' 'Yes' 'No internet service']
OnlineBackup  :  [2 0 1]  =  ['Yes' 'No' 'No internet service']
```

```
DeviceProtection  :  [0 2 1]  =  ['No' 'Yes' 'No internet service']
TechSupport  :  [0 2 1]  =  ['No' 'Yes' 'No internet service']
StreamingTV  :  [0 2 1]  =  ['No' 'Yes' 'No internet service']
StreamingMovies  :  [0 2 1]  =  ['No' 'Yes' 'No internet service']
Contract  :  [0 1 2]  =  ['Month-to-month' 'One year' 'Two year']
PaperlessBilling  :  [1 0]  =  ['Yes' 'No']
PaymentMethod  :  [2 3 0 1]  =  ['Electronic check' 'Mailed check'
'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn  :  [0 1]  =  ['No' 'Yes']
```

```
#viewing the new dataframe after label encoding
new_df.head()
```

```
   gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0       0              0        1           0       1             0
1       1              0        0           0      34             1
2       1              0        0           0       2             1
3       1              0        0           0      45             0
4       0              0        0           0       2             1

   MultipleLines  InternetService  OnlineSecurity  OnlineBackup  \
0              1                0               0             2
1              0                0               2             0
2              0                0               2             2
3              1                0               2             0
4              0                1               0             0

   DeviceProtection  TechSupport  StreamingTV  StreamingMovies
Contract  \
0                 0            0            0                0
0
1                 2            0            0                0
1
2                 0            0            0                0
0
3                 2            2            0                0
1
4                 0            0            0                0
0

   PaperlessBilling  PaymentMethod  MonthlyCharges TotalCharges  Churn

0                 1              2           29.85        29.85      0

1                 0              3           56.95      1889.50      0

2                 1              3           53.85       108.15      1

3                 0              0           42.30      1840.75      0
```

```
4                  1              2            70.70        151.65      1
```

Total Charges should be numeric. So converting it to numerical data type

```
new_df.TotalCharges = pd.to_numeric(new_df.TotalCharges,
errors='coerce')
new_df.TotalCharges.isnull().sum()
```

```
11
```

Using errors='coerce'. It will replace all non-numeric values with NaN.

We see there are 11 missing values in TotalCharges column.

```
new_df[new_df.TotalCharges.isna()]
```

```
      gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
488        0              0        1           1       0             0

753        1              0        0           1       0             1

936        0              0        1           1       0             1

1082       1              0        1           1       0             1

1340       0              0        1           1       0             0

3331       1              0        1           1       0             1

3826       1              0        1           1       0             1

4380       0              0        1           1       0             1

5218       1              0        1           1       0             1

6670       0              0        1           1       0             1

6754       1              0        0           1       0             1


      MultipleLines  InternetService  OnlineSecurity  OnlineBackup  \
488               1                0               2             0
753               0                2               1             1
936               0                0               2             2
1082              2                2               1             1
1340              1                0               2             2
3331              0                2               1             1
3826              2                2               1             1
```

```
4380                0                 2                1              1
5218                0                 2                1              1
6670                2                 0                0              2
6754                2                 0                2              2

      DeviceProtection  TechSupport  StreamingTV  StreamingMovies
Contract  \
488                  2            2            2                0
2
753                  1            1            1                1
2
936                  2            0            2                2
2
1082                 1            1            1                1
2
1340                 2            2            2                0
2
3331                 1            1            1                1
2
3826                 1            1            1                1
2
4380                 1            1            1                1
2
5218                 1            1            1                1
1
6670                 2            2            2                0
2
6754                 0            2            0                0
2

      PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges
Churn
488                  1              0           52.55           NaN
0
753                  0              3           20.25           NaN
0
936                  0              3           80.85           NaN
0
1082                 0              3           25.75           NaN
0
1340                 0              1           56.05           NaN
0
3331                 0              3           19.85           NaN
0
3826                 0              3           25.35           NaN
0
4380                 0              3           20.00           NaN
0
5218                 1              3           19.70           NaN
0
```

```
6670                    0               3           73.35           NaN
0
6754                    1               0           61.90           NaN
0
```

Observation:

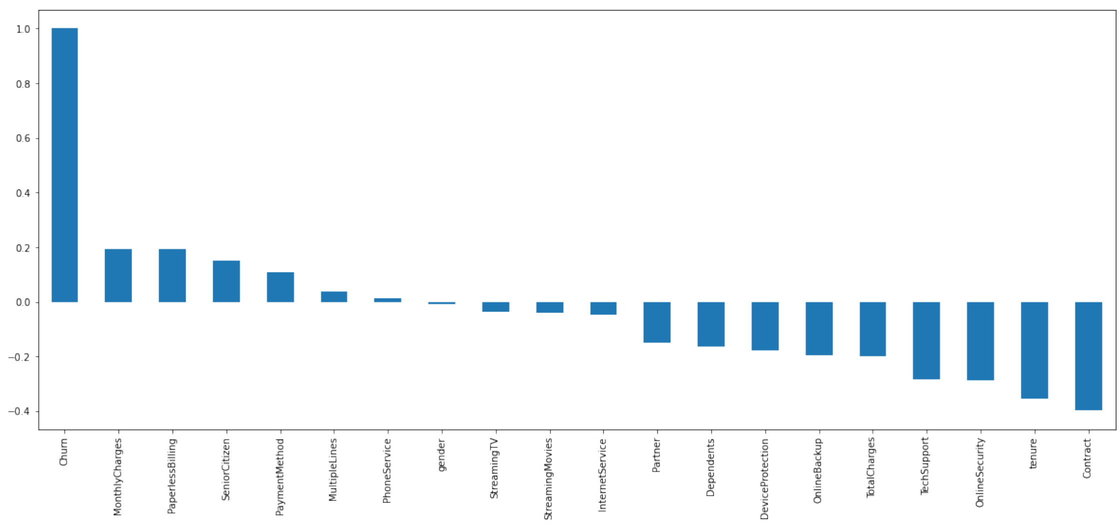
All the new customers(i.e. tenure=0 months) have no total charges data available. Since the number of these records compared to total dataset is very low, so it is safe to ignore them from further processing.

We can drop these rows here as all of them have been in the system for less than a month and Customers are marked as churned if they have left the system in the last one month.

```
#Removing missing values
new_df.dropna(inplace = True)
```

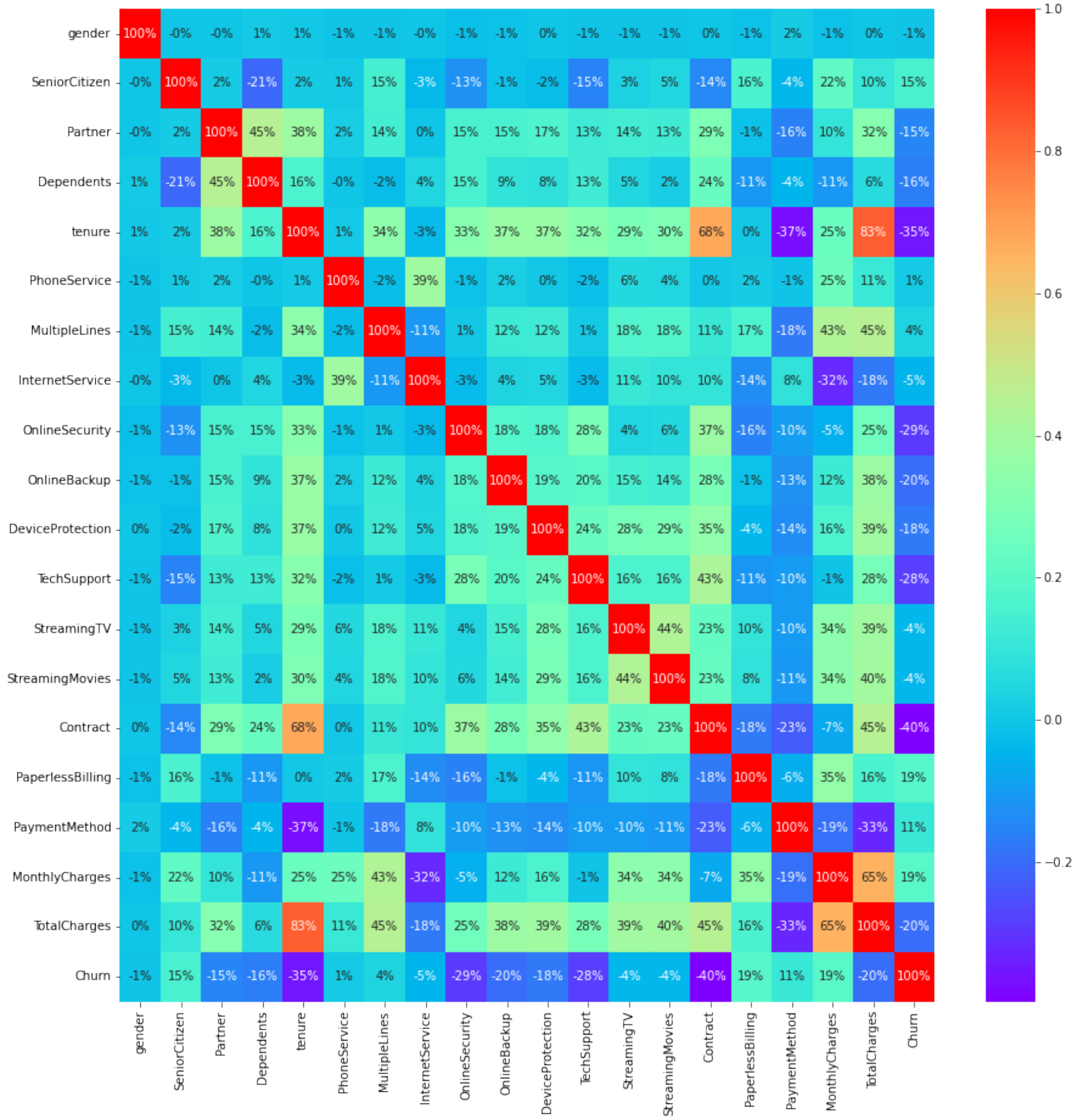
```
plt.figure(figsize=(20,8))
new_df.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

```
<AxesSubplot:>
```



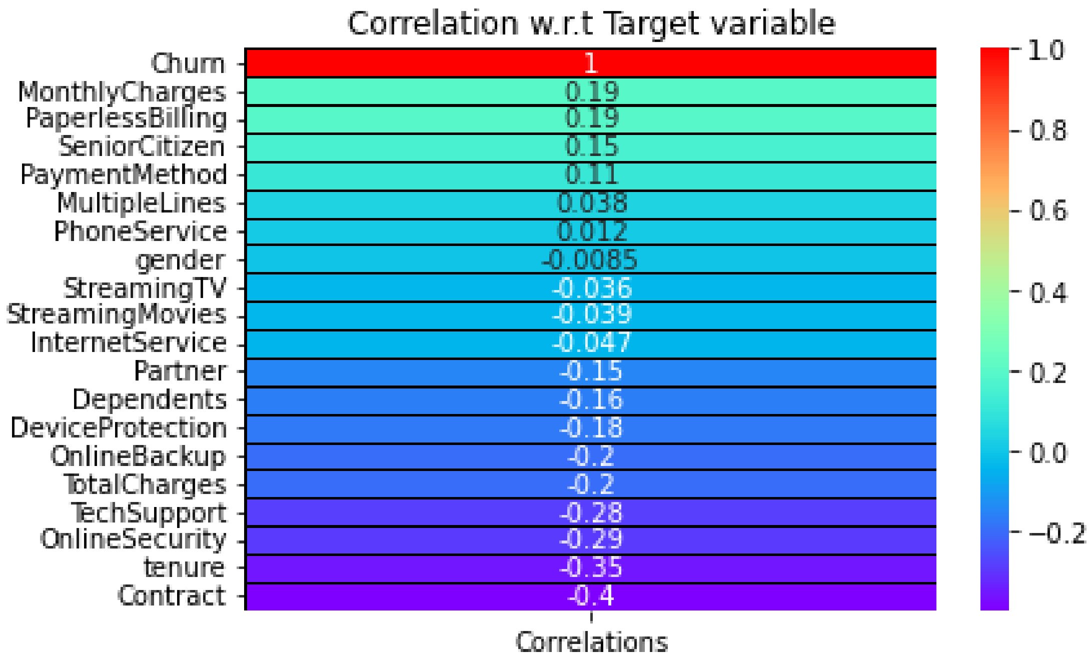

```
plt.figure(figsize=(15,15))
sns.heatmap(new_df.corr(), cmap="rainbow", annot = True, fmt = '.0%')
```

```
<AxesSubplot:>
```



It is a huge matrix with too many features. So, we will check the correlation only with respect to Churn.

```
corr = new_df.corrwith(new_df['Churn']).sort_values(ascending =
False).to_frame()
corr.columns = ['Correlations']
sns.heatmap(corr,annot = True,cmap = 'rainbow',linewidths =
0.6,linecolor = 'black');
plt.title('Correlation w.r.t Target variable');
```



DataFrame.corrwith()

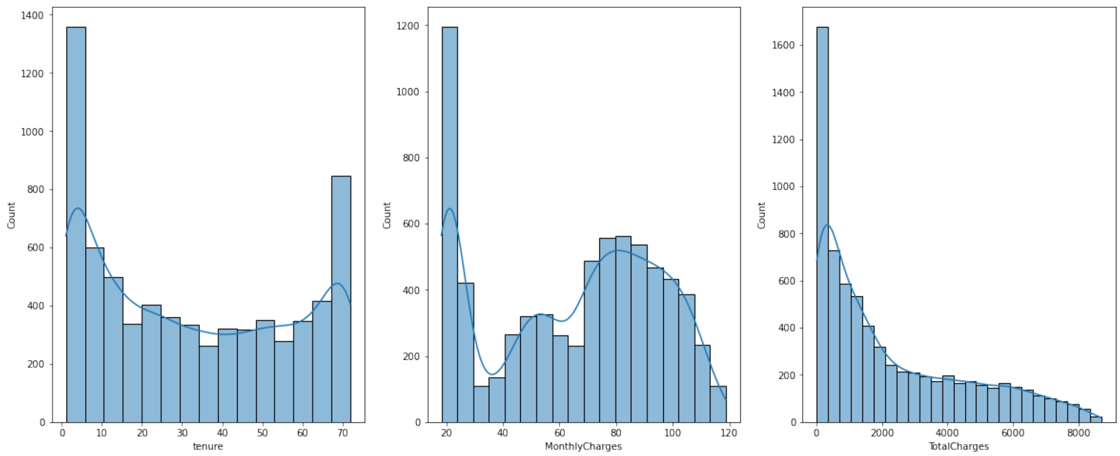
Compute pairwise correlation.Pairwise correlation is computed between rows or columns of DataFrame with rows or columns of Series or DataFrame.

Observation from correlation:

- MulipleLines, PhoneService, gender, StreamingTV, StreamingMovies and InternetService does not display any kind of correlation. We drop the features with correlation coefficient between (-0.1,0.1).
- Remaining features either display a significant positive or negative correlation.
- Services such as Online security, Online backup, Tech support and others without internet connection seem to be negatively related to churn.

Analyzing the Numerical Variables

```
plot , ax = plt.subplots(1 , 3 , figsize = (20 , 8))
g = sns.histplot(new_df['tenure'] , kde = True , ax = ax[0])
g = sns.histplot(new_df['MonthlyCharges'] , kde = True , ax = ax[1])
g = sns.histplot(new_df['TotalCharges'] , kde = True , ax = ax[2])
```



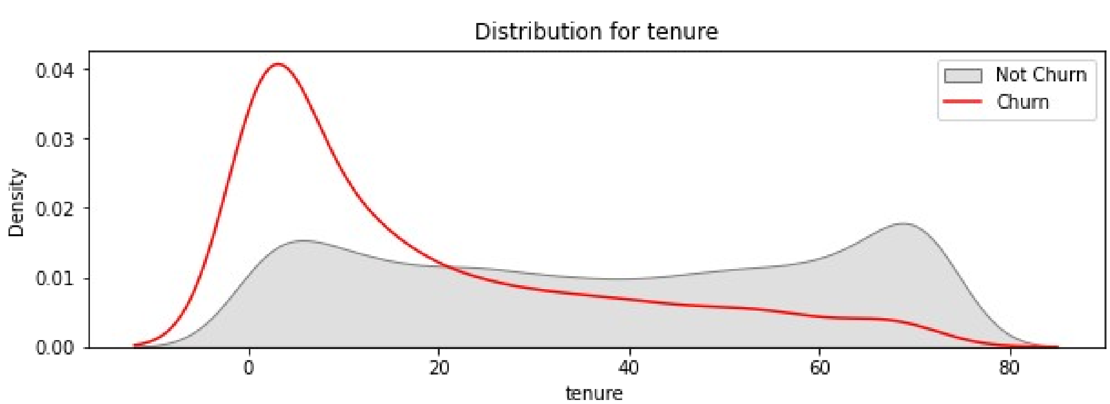
KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable.

Observation:

- The numerical variables are not following a normal distribution. These distributions indicate there are different data distributions present in population data with separate and independent peaks.
- TotalCharges is following Right Skewed distribution

Visualizing the numerical variables wrt churn

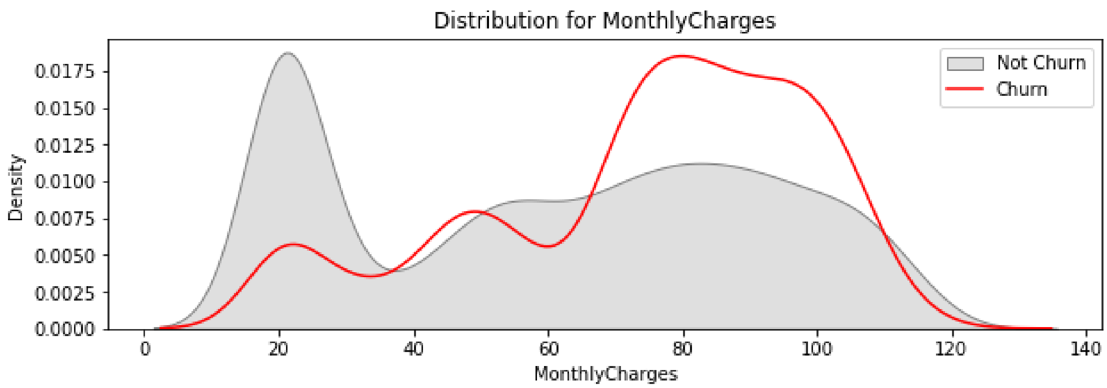
```
def kdeplot(feature):
    plt.figure(figsize=(10, 3))
    plt.title("Distribution for {}".format(feature))
    plot1 = sns.kdeplot(new_df[new_df['Churn'] == 0]
[feature].dropna(), color= 'grey', label= 'Churn: No', shade = True)
    plot2 = sns.kdeplot(new_df[new_df['Churn'] == 1]
[feature].dropna(), color= 'Red', label= 'Churn: Yes')
    plt.legend(["Not Churn","Churn"],loc='upper right')

kdeplot('tenure')
```



Observation:

- Churn is higher at lower tenure values
- Recent customers are more likely to churn.

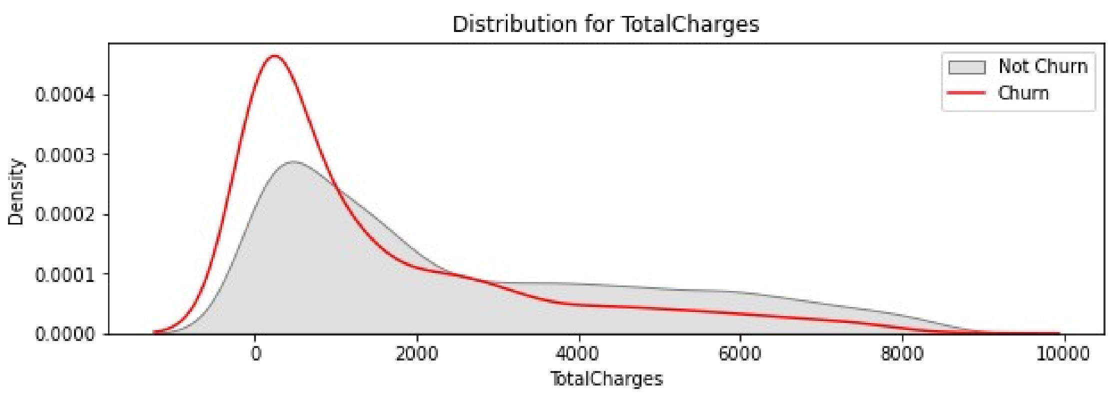
```
kdeplot('MonthlyCharges')
```



Observation:

- Churn is high when Monthly Charges are high
- Customers with higher MonthlyCharges are more likely to churn.

```
kdeplot('TotalCharges')
```



Observation:

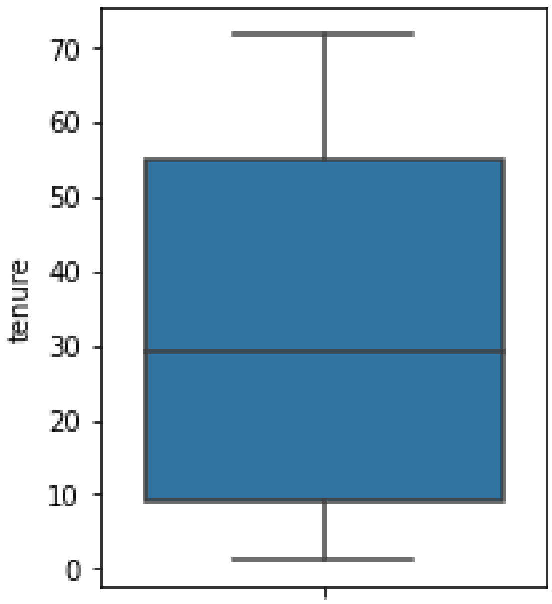
Churn is higher at lower Total Charges

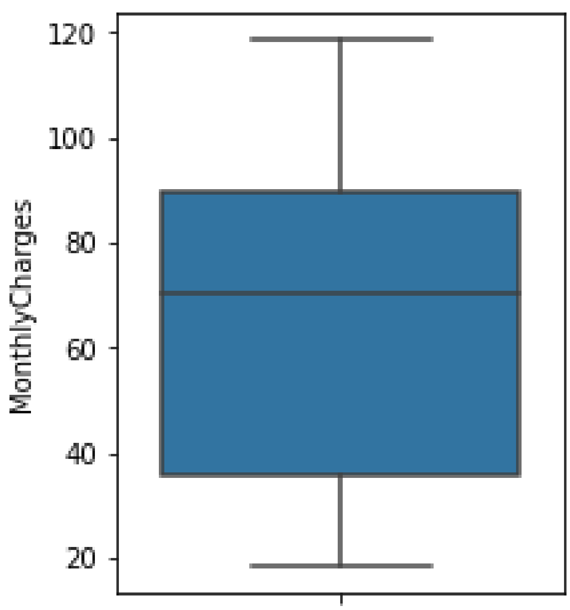
However, all 3 numerical features Monthly Charges, tenure and Total Charges are linked to High Churn.

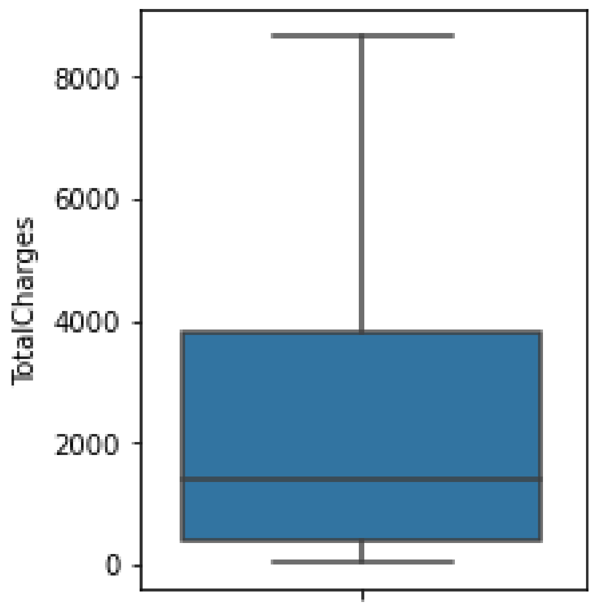
Checking for outliers

```
column_name = ['tenure','MonthlyCharges','TotalCharges']

def identify_outliers(give_df_name, give_column_name):
    for i in column_name:
        fig = plt.figure(figsize=(3,4))
        sns.boxplot(data = new_df, y = i)
```

```
plt.show()

identify_outliers(new_df, column_name)
```







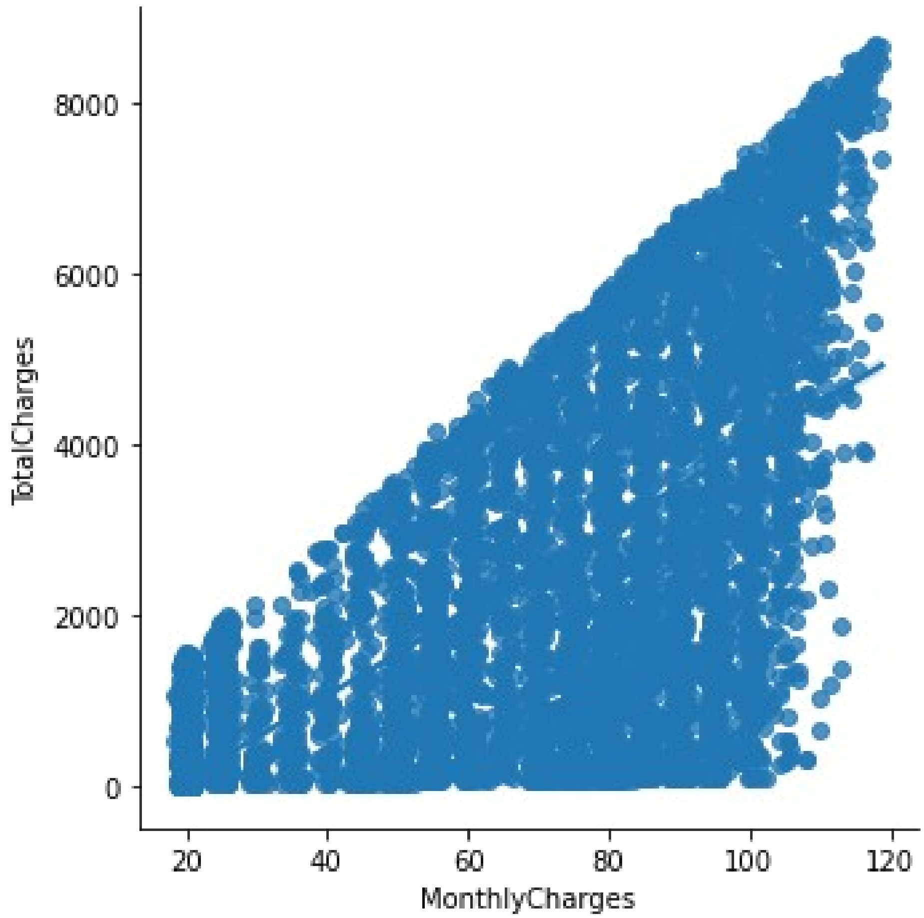
Observation:

There are no values present beyond the upper and lower extremes of the Box plots

Relationship between Monthly Charges and Total Charges

```
sns.lmplot(x ='MonthlyCharges', y ='TotalCharges', data = new_df)
```

```
<seaborn.axisgrid.FacetGrid at 0x285b2917640>
```



Observation:

Total Charges increase as Monthly Charges increase

```
new_df.drop(columns = ['PhoneService',
'gender','StreamingTV','StreamingMovies','MultipleLines','InternetServ
ice'],inplace = True)
new_df.head()
```

```
   SeniorCitizen  Partner  Dependents  tenure  OnlineSecurity
OnlineBackup  \
0              0        1           0       1               0
2
1              0        0           0      34               2
0
2              0        0           0       2               2
2
3              0        0           0      45               2
0
4              0        0           0       2               0
0
```

```
   DeviceProtection  TechSupport  Contract  PaperlessBilling
PaymentMethod  \
0                 0            0         0                 1
2
1                 2            0         1                 0
3
2                 0            0         0                 1
3
3                 2            2         1                 0
0
4                 0            0         0                 1
2

   MonthlyCharges  TotalCharges  Churn
0           29.85         29.85      0
1           56.95       1889.50      0
2           53.85        108.15      1
3           42.30       1840.75      0
4           70.70        151.65      1
```

Normalization

```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
mms = MinMaxScaler() # Normalization
ss = StandardScaler() # Standardization

new_df['tenure'] = mms.fit_transform(new_df[['tenure']])
new_df['MonthlyCharges'] =
mms.fit_transform(new_df[['MonthlyCharges']])
new_df['TotalCharges'] = mms.fit_transform(new_df[['TotalCharges']])
new_df.head()
```

```
   SeniorCitizen  Partner  Dependents    tenure  OnlineSecurity
OnlineBackup  \
0              0        1           0  0.000000               0
2
1              0        0           0  0.464789               2
0
2              0        0           0  0.014085               2
2
3              0        0           0  0.619718               2
0
4              0        0           0  0.014085               0
0

   DeviceProtection  TechSupport  Contract  PaperlessBilling
PaymentMethod  \
0                 0            0         0                 1
2
1                 2            0         1                 0
3
```

```
2                  0            0         0                 1
3
3                  2            2         1                 0
0
4                  0            0         0                 1
2

   MonthlyCharges  TotalCharges  Churn
0        0.115423      0.001275      0
1        0.385075      0.215867      0
2        0.354229      0.010310      1
3        0.239303      0.210241      0
4        0.521891      0.015330      1
```

- Machine learning model does not understand the units of the values of the features. It treats the input just as a simple number but does not understand the true meaning of that value. Thus, it becomes necessary to scale the data. Eg : Age = Years; FastingBS = mg / dl; Charges = Currency
- We have 2 options for data scaling : 1) Normalization 2) Standardization. As most of the algorithms assume the data to be normally (Gaussian) distributed, Normalization is done for features whose data does not display normal distribution and standardization is carried out for features that are normally distributed where their values are huge or very small as compared to other features.
- Normalization : tenure, MonthlyCharges and TotalCharges features are normalized
- Standardization : None of the features are standardized for the above data.

Data Balancing using SMOTE :

In order to cope with imbalanced data, there are 2 options :

- Undersampling : Trim down the majority samples of the target variable.
- Oversampling : Increase the minority samples of the target variable to the majority samples.
- After doing trial-error with undersampling & oversampling, we have decided to go with oversampling!
- For data balancing, we will use imblearn.
- pip statement : !pip install imbalanced-learn

```
import imblearn
from collections import Counter
from imblearn.over_sampling import SMOTE

----------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
Input In [68], in <cell line: 1>()
```

```
----> 1 import imblearn
      2 from collections import Counter
      3 from imblearn.over_sampling import SMOTE

ModuleNotFoundError: No module named 'imblearn'
```

```
x1 = new_df.iloc[:,:13]
y1 = new_df.iloc[:,13]
```

```
y1
```

```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7032, dtype: int32
```

4. Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

Selecting the features from the above conducted tests and splitting the data into 80 - 20 train - test groups.

```
x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1,
test_size = 0.20, random_state = 0)
print('Classes and number of values in trainset before
SMOTE:',Counter(y1_train),'\n')
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Input In [72], in <cell line: 2>()
      1 x1_train, x1_test, y1_train, y1_test = train_test_split(x1,
y1, test_size = 0.20, random_state = 0)
----> 2 print('Classes and number of values in trainset before
SMOTE:',Counter(y1_train),'\n')

NameError: name 'Counter' is not defined
```

```
#smote
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
x1_train,y1_train = oversample.fit_resample(x1_train,y1_train)
print('Classes and number of values in trainset after
SMOTE:',Counter(y1_train),'\n')
```

```
----------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
Input In [73], in <cell line: 2>()
      1 #smote
----> 2 from imblearn.over_sampling import SMOTE
      3 oversample = SMOTE()
      4 x1_train,y1_train = oversample.fit_resample(x1_train,y1_train)

ModuleNotFoundError: No module named 'imblearn'
```

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 42)

model = log_reg.fit(x1_train, y1_train)
y1_pred = model.predict(x1_test)
print('model.predict :',y1_pred)
print('model.score :', model.score(x1_train, y1_train),'\n')

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y1_test, y1_pred)
print('Accuracy : ',accuracy,'\n')

from sklearn.model_selection import cross_val_score
scores = cross_val_score(log_reg, x1, y1, cv=5)
print('Cross Validation scores :', scores,'\n')

mean_accuracy_log_reg = (np.mean(scores))*100
print('Mean Accuracy :',mean_accuracy_log_reg)
```

```
model.predict : [0 0 0 ... 1 0 1]
model.score : 0.8005333333333333

Accuracy :  0.7995735607675906

Cross Validation scores : [0.81023454 0.7960199  0.78520626 0.80440967
0.79871977]

Mean Accuracy : 79.8918029240103
```

```
#importing Decision Trees
from sklearn.tree import DecisionTreeClassifier
```

```
dtc = DecisionTreeClassifier(random_state=42,max_depth=4)

model = dtc.fit(x1_train, y1_train)
y1_pred = model.predict(x1_test)
print('model.predict :',y1_pred)
print('model.score :', model.score(x1_train, y1_train))

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y1_test, y1_pred)
print('Accuracy : ',accuracy,'\n')

from sklearn.model_selection import cross_val_score
scores = cross_val_score(dtc, x1, y1, cv=5)
print('Cross Validation scores :', scores,'\n')

mean_accuracy_dtc = (np.mean(scores))*100
print('Mean Accuracy :',mean_accuracy_dtc,'\n')
```

```
model.predict : [0 0 0 ... 1 0 0]
model.score : 0.7916444444444445
Accuracy :  0.7945984363894811

Cross Validation scores : [0.78322672 0.77256574 0.77027027 0.78378378
0.7859175 ]

Mean Accuracy : 77.91528033476187
```