Customer Churn Analysis

Pratyush Raj
DATA TRAINED

Overview:

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. 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.

Note: You can find the dataset in the link below.

Goals

1. To Build churn prediction models using vast volumes of data collected
2. Knowing who is most likely to defect means that a company can prioritize focused marketing efforts on that subset of its customer base.
3. Customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

Specifications

What Is Customer Churn?

Customer churn is the percentage of customers that stopped using your company's product or service during a certain time frame. You can calculate the churn rate by dividing the number of customers you lost during that time period -- say a quarter -- by the number of customers you had at the beginning of that time period.

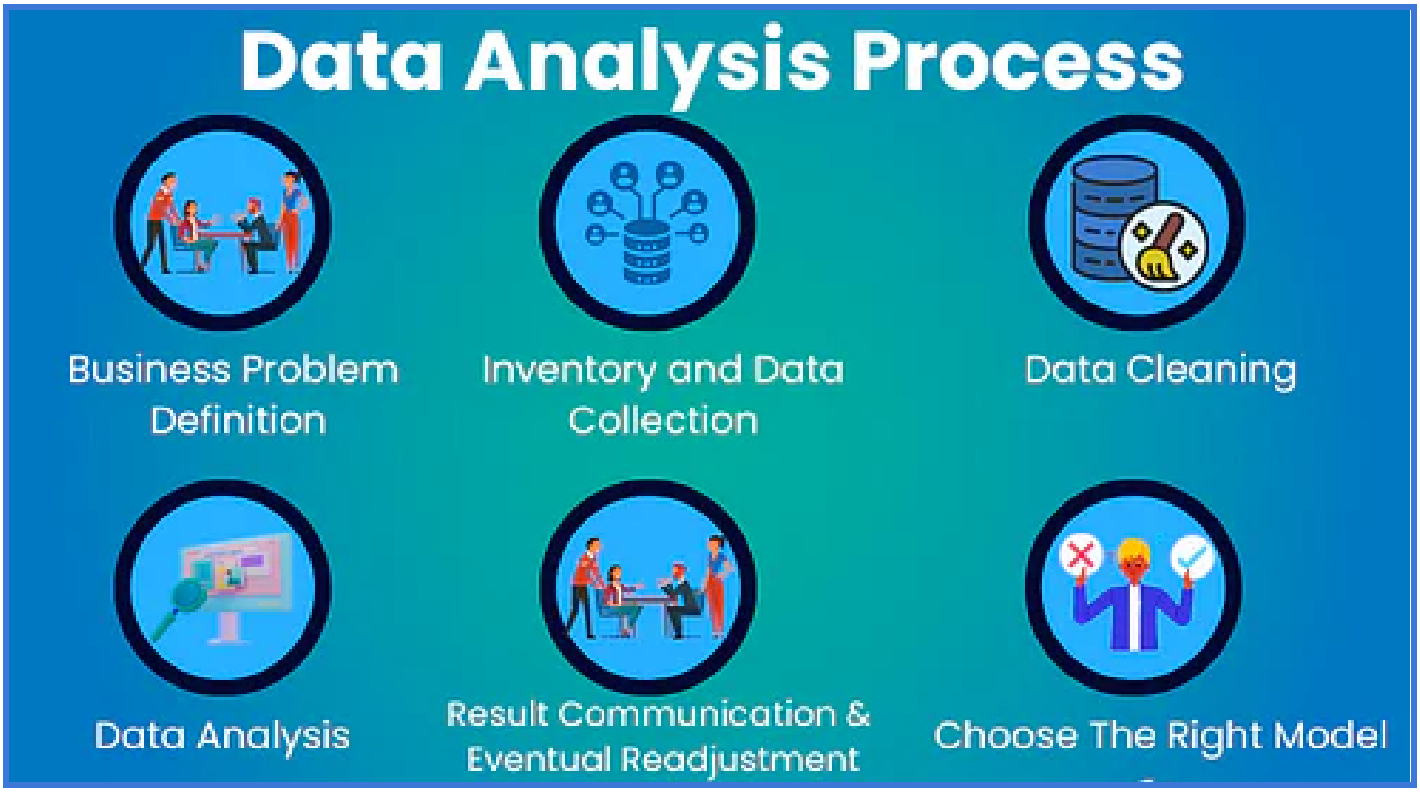
For example, if you start your quarter with 400 customers and end with 380, your churn rate is 5% because you lost 5% of your customers.

Obviously, your company should aim for a churn rate that is as close to 0% as possible. In order to do this, your company has to be on top of its churn rate at all times and treat it as a top priority.

In this example, I calculated the churn rate as the percentage of customers lost that quarter. However, you can calculate the churn rate in whatever way is best for your company. Some examples are:

1. The number of customers lost
2. The value of recurring business lost
3. The percentage of recurring value lost

Procedures:



Data Preparation: Load, clean, and format.

Importing important libraries:

In [24]:

```
#importing impotanat libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Importing Customer Churn Analysis dataset CSV file

In [25]:
```
#Importing Customer Churn Analysis dataset csv file.
df=pd.read_csv('Customer_churn.csv')
```

In [26]:
```
print('Number of rows',df.shape[0])
print('number of column',df.shape[1])
pd.set_option('display.max_columns',None) # # This will enable us to see truncated columns
df.head()
```

```
Number of rows 7043
number of column 21
```

Out[26]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	Tech
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	

Number of rows 7043, Number of columns 21

We looked into columns by their datatypes

In [29]:
```
#As we have 21 columns we will look into it by their datatypes
df.columns.to_series().groupby(df.dtypes).groups
```

Out[29]:
```
{int64: ['SeniorCitizen', 'tenure'], float64: ['MonthlyCharges'], object: ['customerID', 'gender', 'Partner', 'Dependents', 'PhoneServic
e', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovi
es', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges', 'Churn']}
```

Dropping unnecessary and splitting numerical and categorical data

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

```
#splitting numerical and categorical data
Categorical = [ 'gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService',
           'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
           'Contract', 'PaperlessBilling', 'PaymentMethod','SeniorCitizen', 'Churn']
Numerical = ['tenure','MonthlyCharges','TotalCharges']
```

By doing this we can make data use easily, we can see above there are very little Numerical data.

Data Integrity Test

Data Integrity Test

```
df.duplicated().sum()
```

```
22
```

We can see 22 duplicated data in datasets.So we will drop it

```
df.drop_duplicates(keep='last',inplace=True)
```

```
df.shape
```

```
(7021, 20)
```

```
df.isin([' ','NA','-']).sum()
```

```
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       11
Churn               0
dtype: int64
```

It is a process in which data is verified in the database whether it is accurate and functions as per requirements. If there is some null value or data is missing or some blank spaces.

Statical Analysis

Statical Analysis

```
df['TotalCharges'].isin([' ']).sum().any()
```

```
True
```

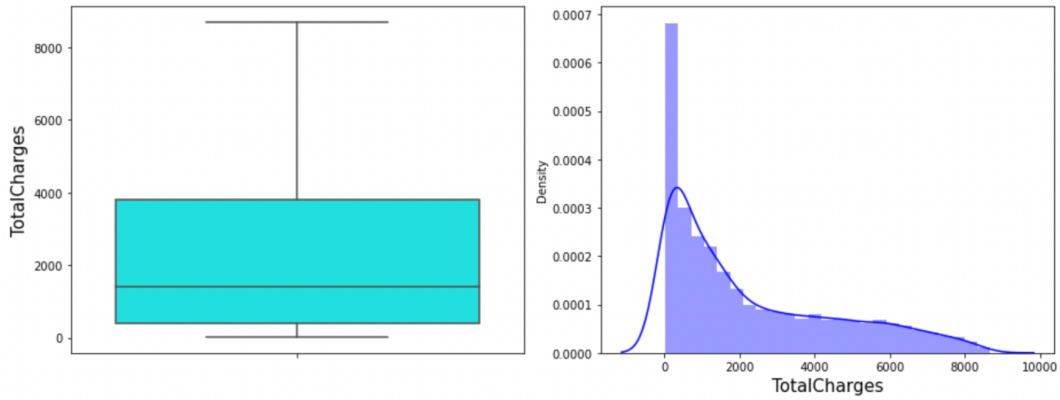
```
#replacing white space with null values
df['TotalCharges']=df['TotalCharges'].replace(' ',np.nan)
```

```
# Converting object datatype into float
df['TotalCharges']= df['TotalCharges'].astype(float)
```

```
missing_values=df.isnull().sum().sort_values(ascending=False)
percent_missing_values=(missing_values/len(df)*100)
print(pd.concat([missing_values,percent_missing_values],axis=1,keys=['missing_values','percent_missing_values']))
```

```
                  missing_values  percent_missing_values
TotalCharges                  11                0.156673
gender                         0                0.000000
SeniorCitizen                  0                0.000000
MonthlyCharges                 0                0.000000
PaymentMethod                  0                0.000000
PaperlessBilling               0                0.000000
Contract                       0                0.000000
StreamingMovies                0                0.000000
StreamingTV                    0                0.000000
TechSupport                    0                0.000000
DeviceProtection               0                0.000000
OnlineBackup                   0                0.000000
OnlineSecurity                 0                0.000000
InternetService                0                0.000000
MultipleLines                  0                0.000000
PhoneService                   0                0.000000
tenure                         0                0.000000
Dependents                     0                0.000000
Partner                        0                0.000000
Churn                          0                0.000000
```

```
plt.figure(figsize=(13,5))
plt.subplot(1,2,1)
sns.boxplot(y='TotalCharges', data=df,color='cyan')
plt.ylabel('TotalCharges',fontsize=15)
plt.subplot(1,2,2)
sns.distplot(df['TotalCharges'], color='b')
plt.xlabel('TotalCharges',fontsize=15)
plt.tight_layout()
plt.show()
```



Imputing missing values with mean

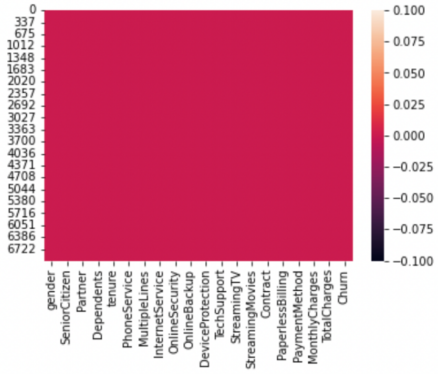
Imputing missing values with mean and checking null values.

```
df['TotalCharges']=df['TotalCharges'].fillna(df['TotalCharges'].mean())
```

Checking Null Values

```
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```



Look more into data with description.

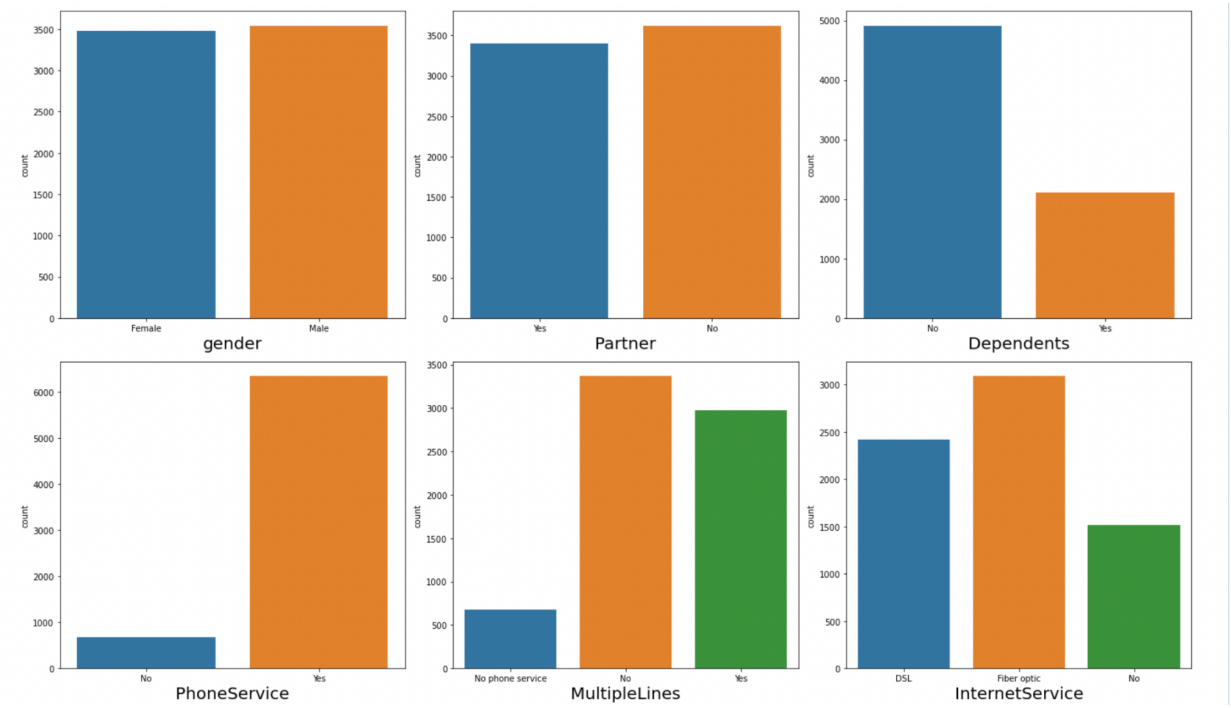
```
df.describe()
```

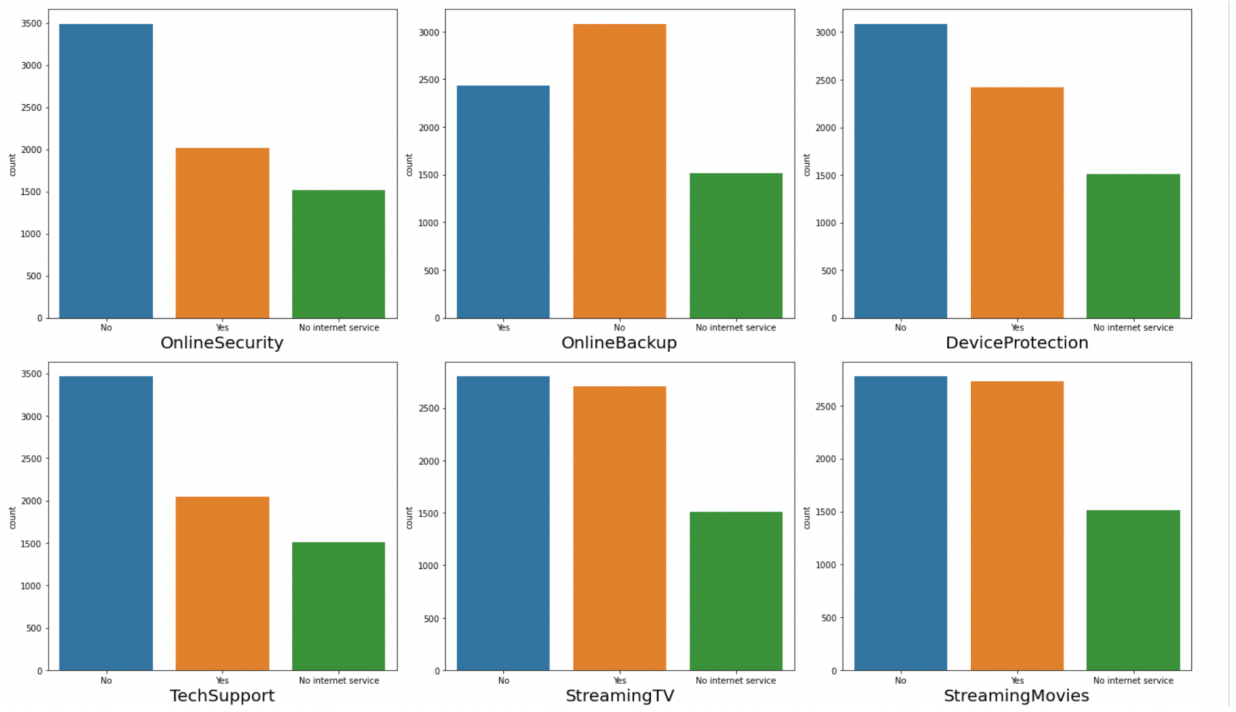
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7021.000000	7021.000000	7021.000000	7021.000000
mean	0.162512	32.469449	64.851894	2290.353388
std	0.368947	24.534965	30.069001	2265.044136
min	0.000000	0.000000	18.250000	18.800000
25%	0.000000	9.000000	35.750000	411.150000
50%	0.000000	29.000000	70.400000	1410.250000
75%	0.000000	55.000000	89.900000	3801.700000
max	1.000000	72.000000	118.750000	8684.800000

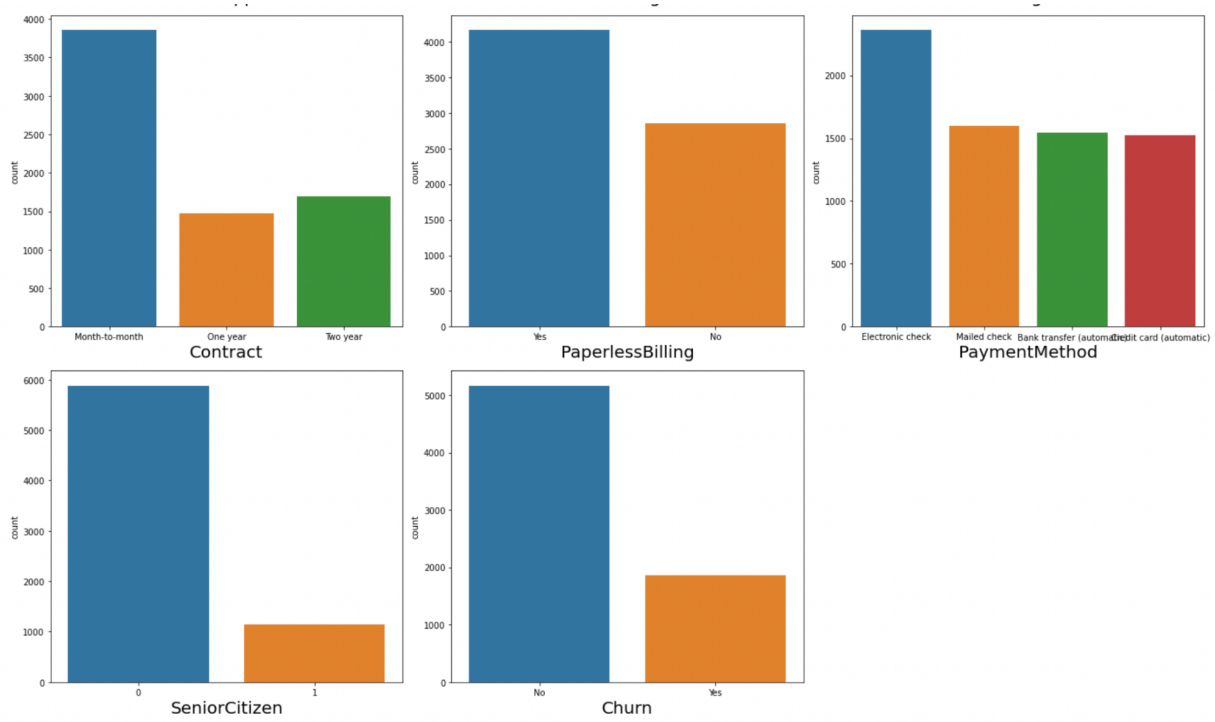
```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
SeniorCitizen	7021.0	0.162512	0.368947	0.00	0.00	0.00	0.0	1.00
tenure	7021.0	32.469449	24.534965	0.00	9.00	29.00	55.0	72.00
MonthlyCharges	7021.0	64.851894	30.069001	18.25	35.75	70.40	89.9	118.75
TotalCharges	7021.0	2290.353388	2265.044136	18.80	411.15	1410.25	3801.7	8684.80

Start with Enlisting Value counts & Sub-categories of different categorial features available.

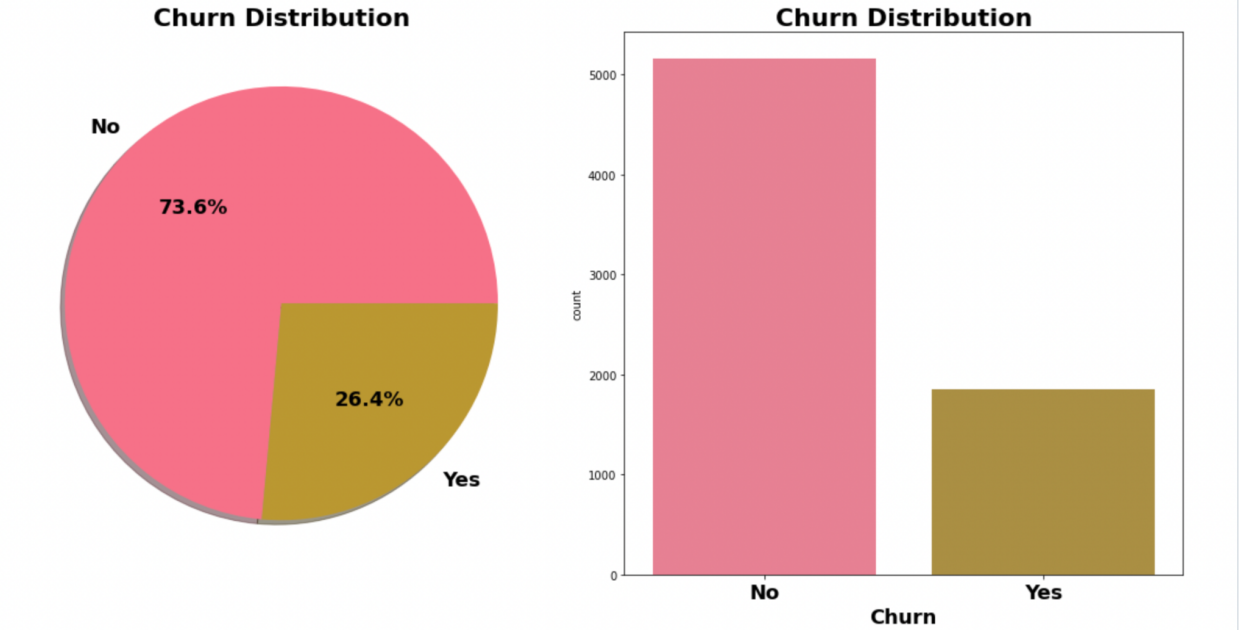





count
Month-to-month
One year
Two year
Contract
Yes
No
PaperlessBilling
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod
0
1
SeniorCitizen
No
Yes
Churn

Churn Distribution:

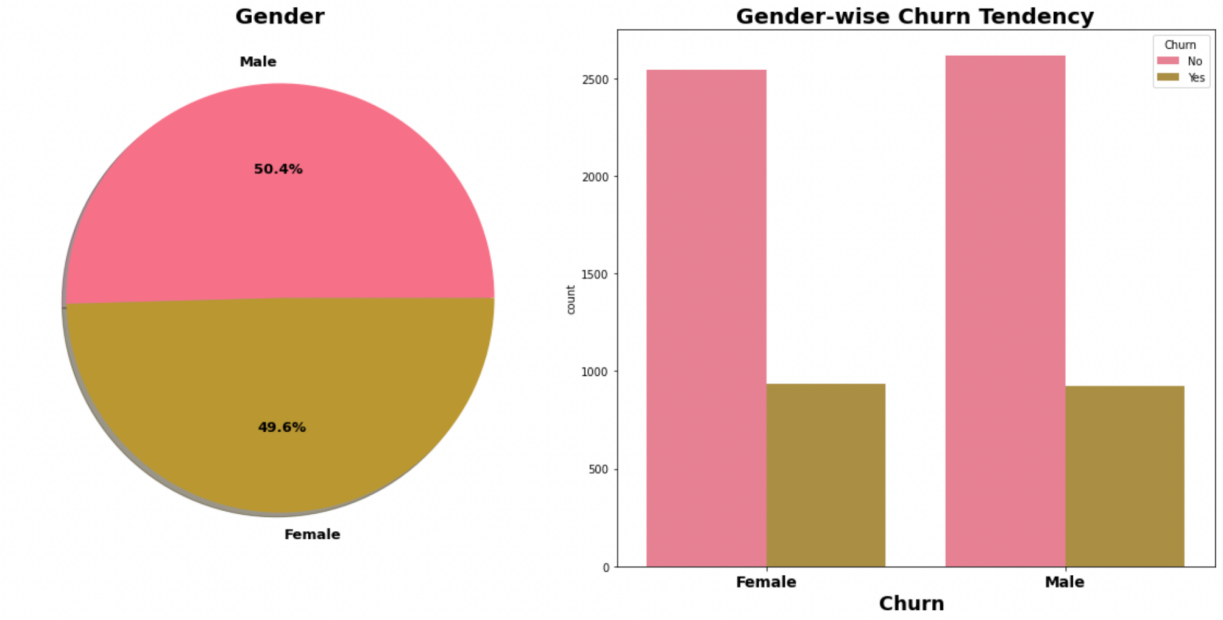
```
plt.rcParams["figure.autolayout"] = True
sns.set_palette('husl')
f,ax=plt.subplots(1,2,figsize=(15,8))
df['Churn'].value_counts().plot.pie(autopct='%3.1f%%',
                                    textprops ={ 'fontweight': 'bold','fontsize':18}, ax=ax[0],shadow=True)
ax[0].set_title('Churn Distribution', fontsize=22,fontweight ='bold')
ax[0].set_ylabel('')
sns.countplot('Churn',data=df,ax=ax[1])
ax[1].set_title('Churn Distribution',fontsize=22,fontweight ='bold')
ax[1].set_xlabel("Churn",fontsize=18,fontweight ='bold')
plt.xticks(fontsize=18,fontweight ='bold')
plt.show()
```



26.4 % of Customers choose to churn service in the last month. Which is quite a high number. This all leads to an imbalanced data case as churn is our target variable.

Gender-wise Churn tendency

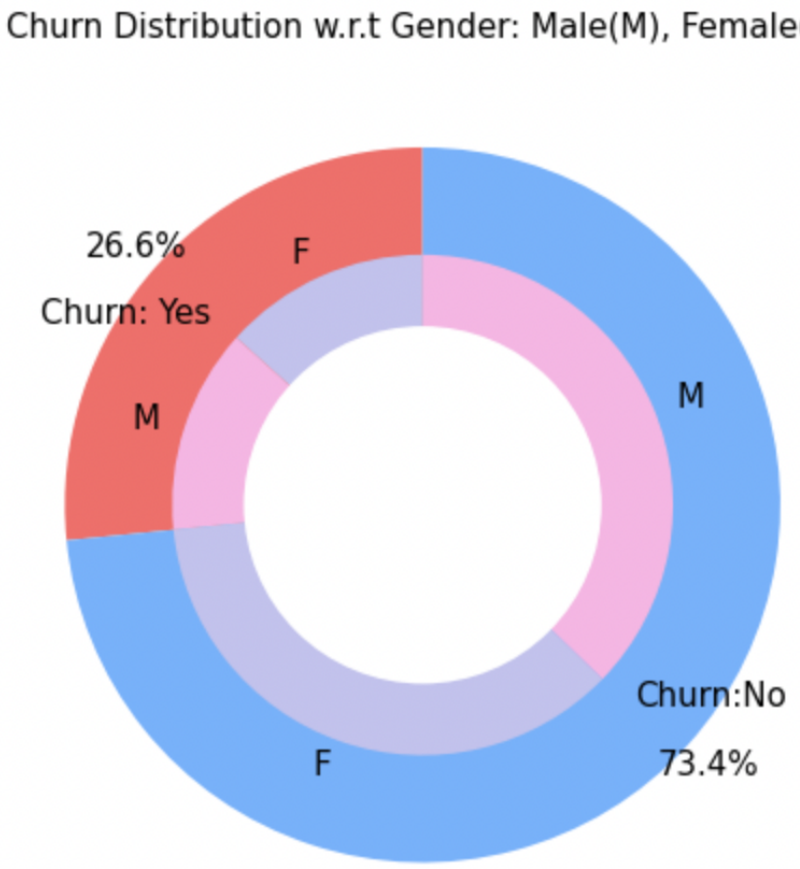
```
plt.rcParams["figure.autolayout"] = True
sns.set_palette('husl')
fig,ax=plt.subplots(1,2,figsize=(16,8))
df['gender'].value_counts().plot.pie(autopct='%2.1f%%',
                                     textprops ={ 'fontweight': 'bold','fontsize':13}, ax=ax[0],shadow=True)
ax[0].set_title('Gender', fontsize=20,fontweight ='bold')
ax[0].set_ylabel('')
sns.countplot('gender',hue="Churn",data=df,ax=ax[1])
ax[1].set_title('Gender-wise Churn Tendency',fontsize=20,fontweight ='bold')
ax[1].set_xlabel("Churn ",fontsize=18,fontweight ='bold')
plt.xticks(fontsize=14,fontweight ='bold')
plt.tight_layout()
plt.show()
```




```
pd.crosstab(df['gender'],df["Churn"],margins=True).style.background_gradient(cmap='summer_r')
```

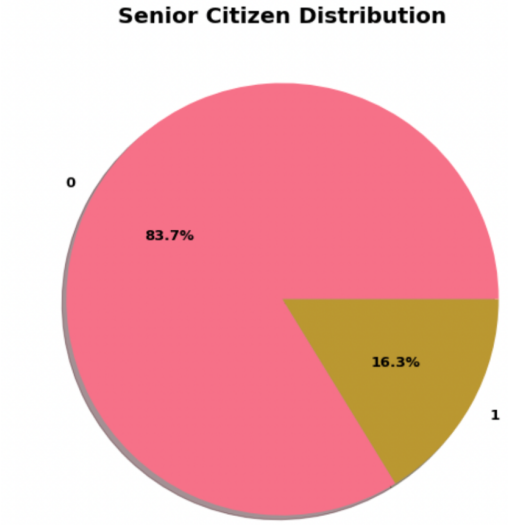
Churn	No	Yes	All
gender			
Female	2546	934	3480
Male	2618	923	3541
All	5164	1857	7021

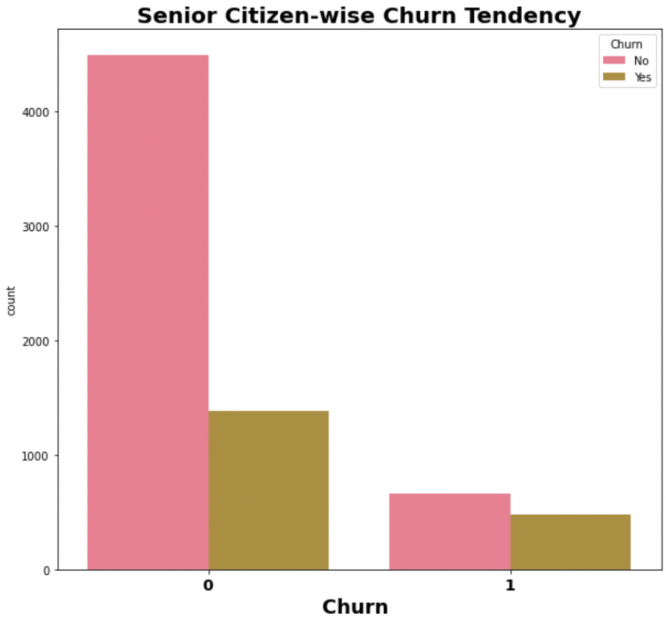
Churn Distribution w.r.t Gender



Data contain both genders almost in the same proportion with a minor difference. Both genders have a tendency to attrition in the same percentage.

Senior Citizen Distribution:





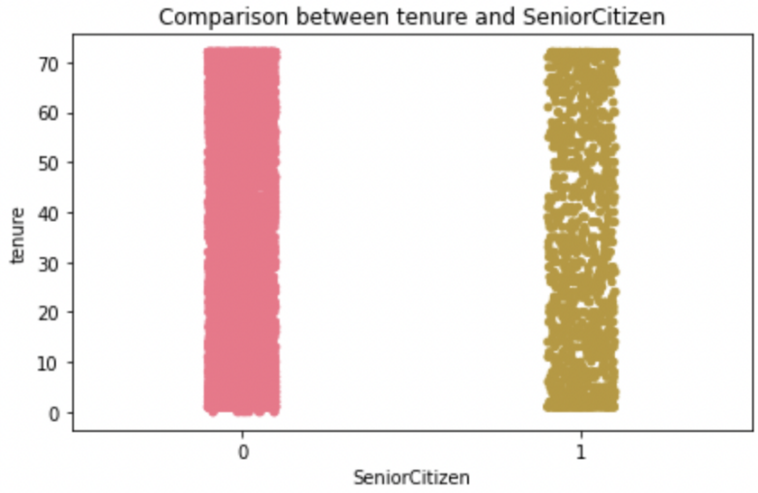

```
pd.crosstab([df.gender,df.SeniorCitizen],df["Churn"],margins=True).style.background_gradient(cmap='summer_r')
```

gender	SeniorCitizen	Churn	No	Yes	All
Female	0		2218	695	2913
	1		328	239	567
Male	0		2280	687	2967
	1		338	236	574
All			5164	1857	7021

There are only 16.3 % of the customers are senior citizens. Thus most of our customers in the data are younger people.

Comparing tenure and SeniorCitizen:

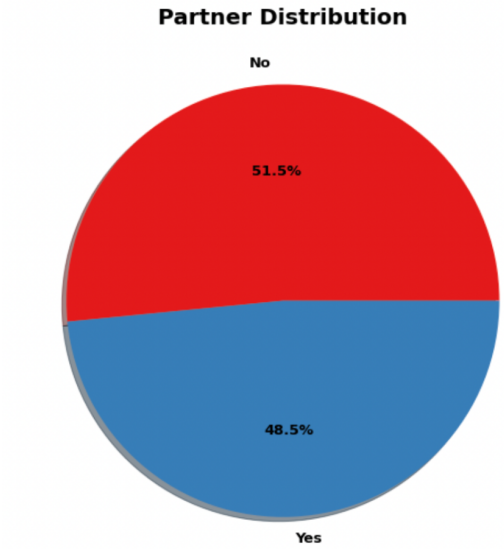
```
# Comparing tenure and SeniorCitizen
plt.title("Comparison between tenure and SeniorCitizen")
sns.stripplot(x = "SeniorCitizen",y="tenure",data = df)
plt.show()
```

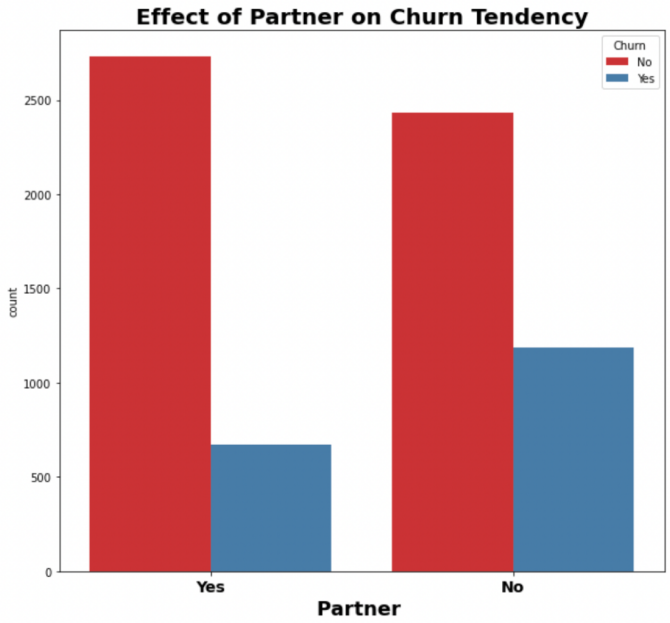


Around 16% of customers are Senior citizens and from the count plot, we can see they have more tendency to churn.

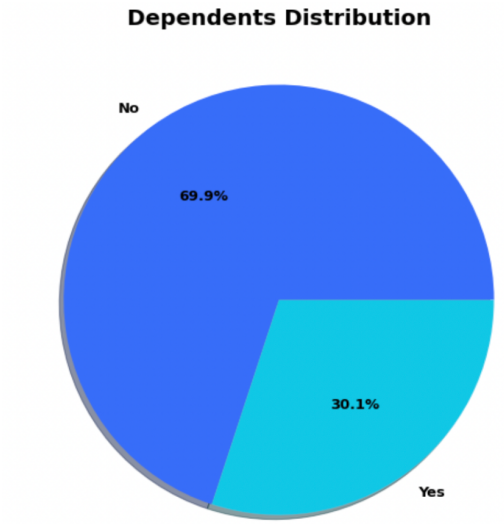
There is no significant relation between Senior Citizens and Tenure.

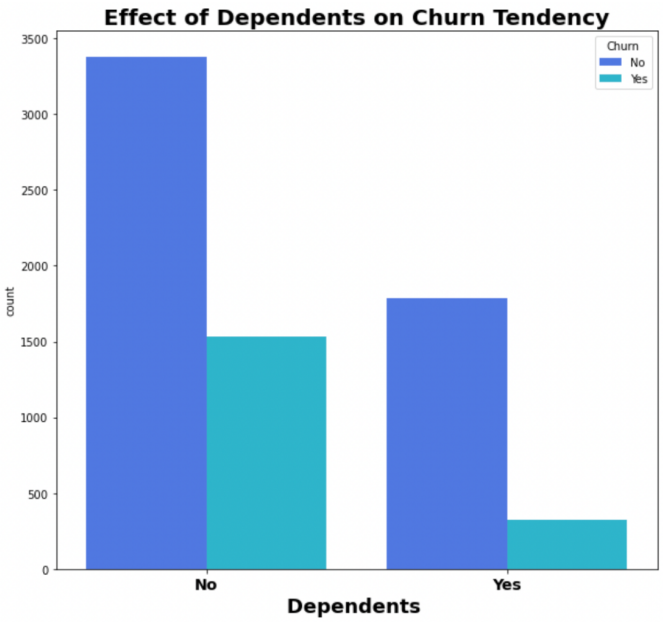
Partner Distribution:





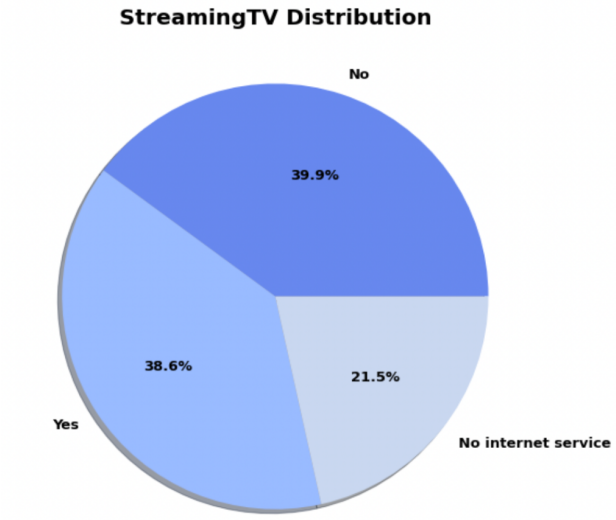
Dependents Distribution:

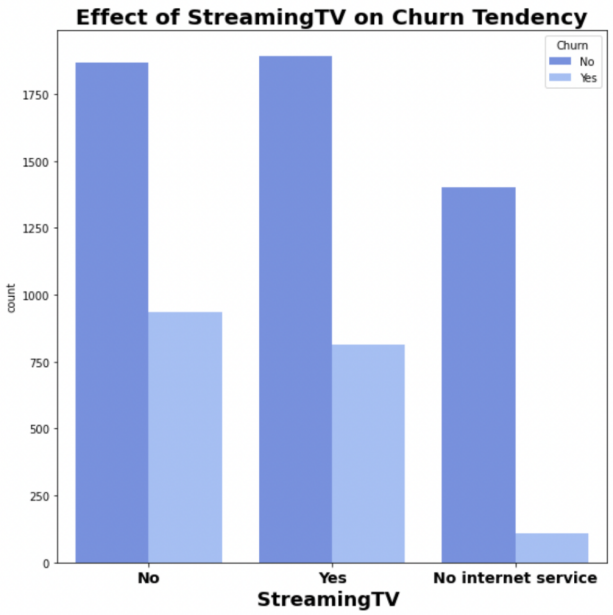




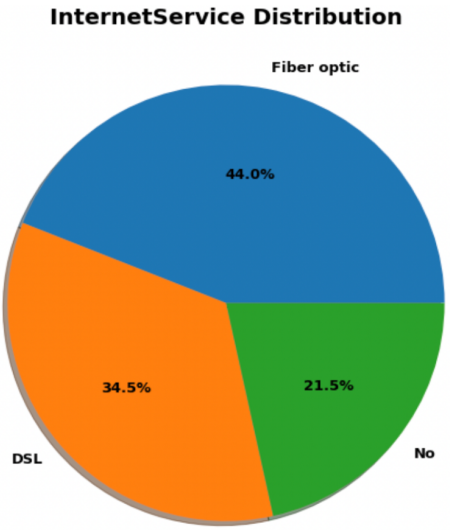
Customers having Partners have less tendency to Churn. Almost 30% of Customers have dependents on them and they also have less tendency to churn compared to the remaining 70%

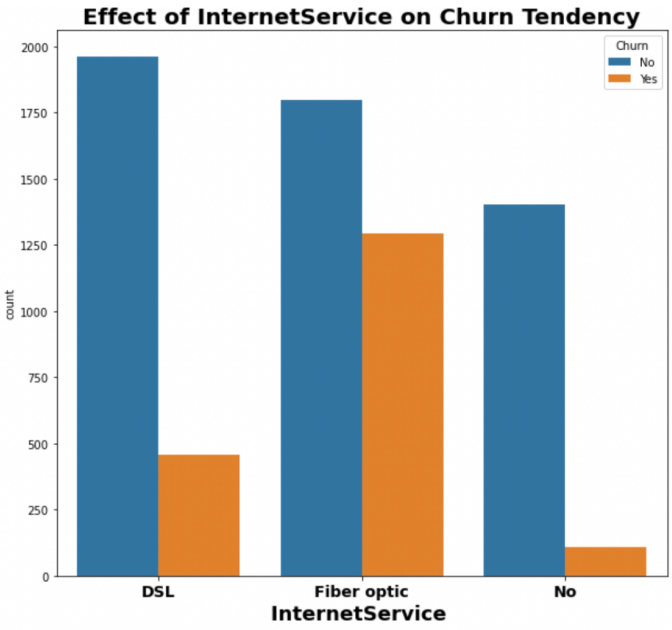
Streaming TV Distribution:



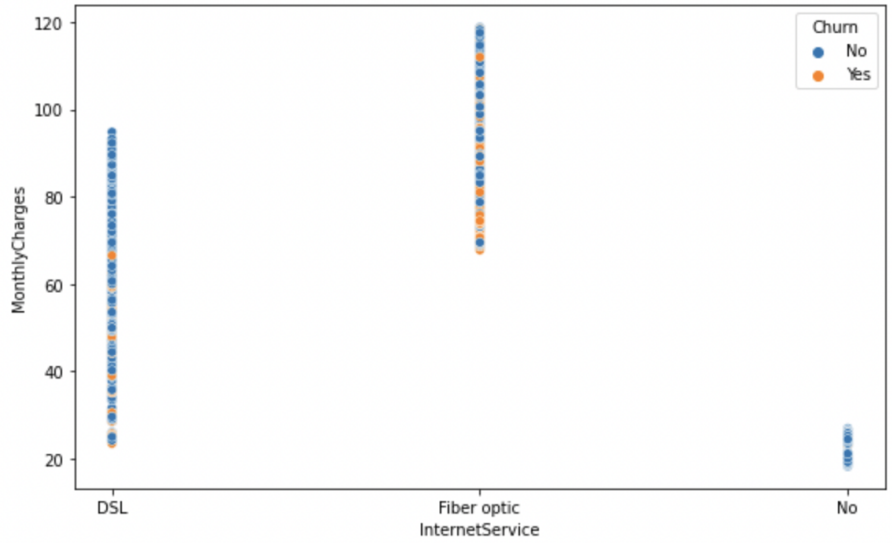


Internet service Distribution:





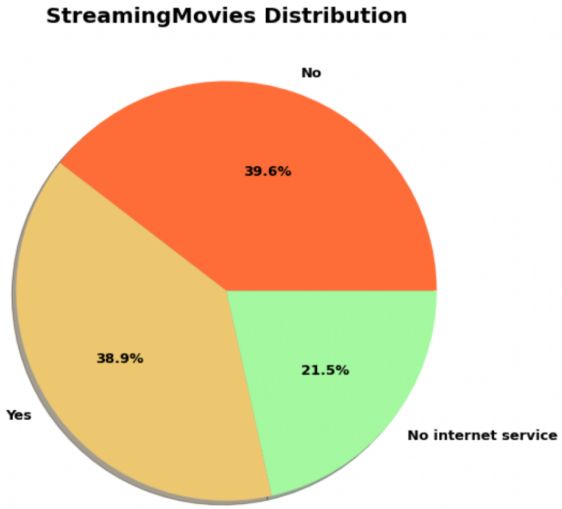
```
plt.figure(figsize=(8,5))
sns.scatterplot(x="InternetService", y='MonthlyCharges',data=df,hue="Churn")
plt.show()
```

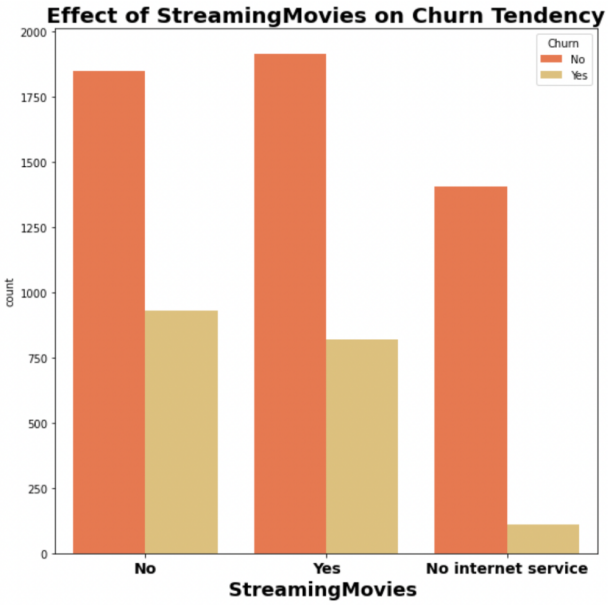


44% of Customers prefer Fibre optic as an Internet service and surprisingly we can find a high churn rate among them.

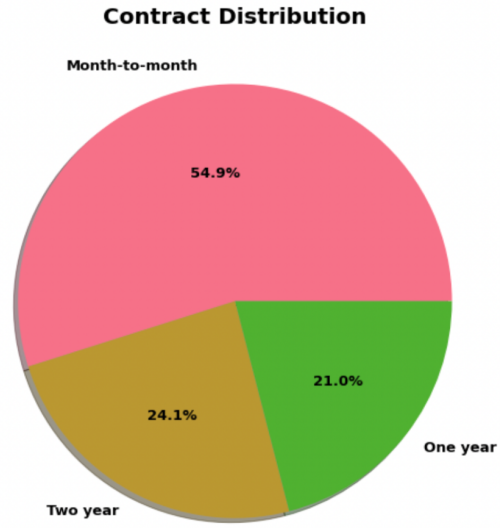
We can find high monthly charges among customers using fiber optic compared to DSL. We can conclude that High charges are the reason for customer churn.

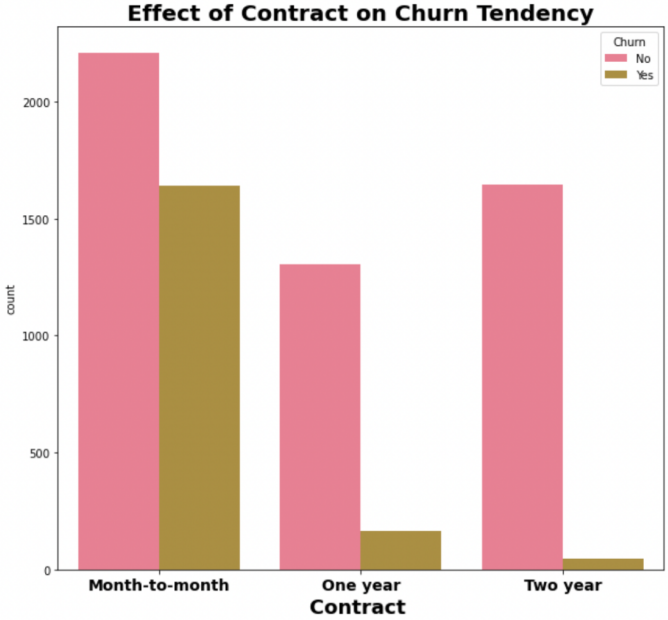
StreamingMovies Distribution:



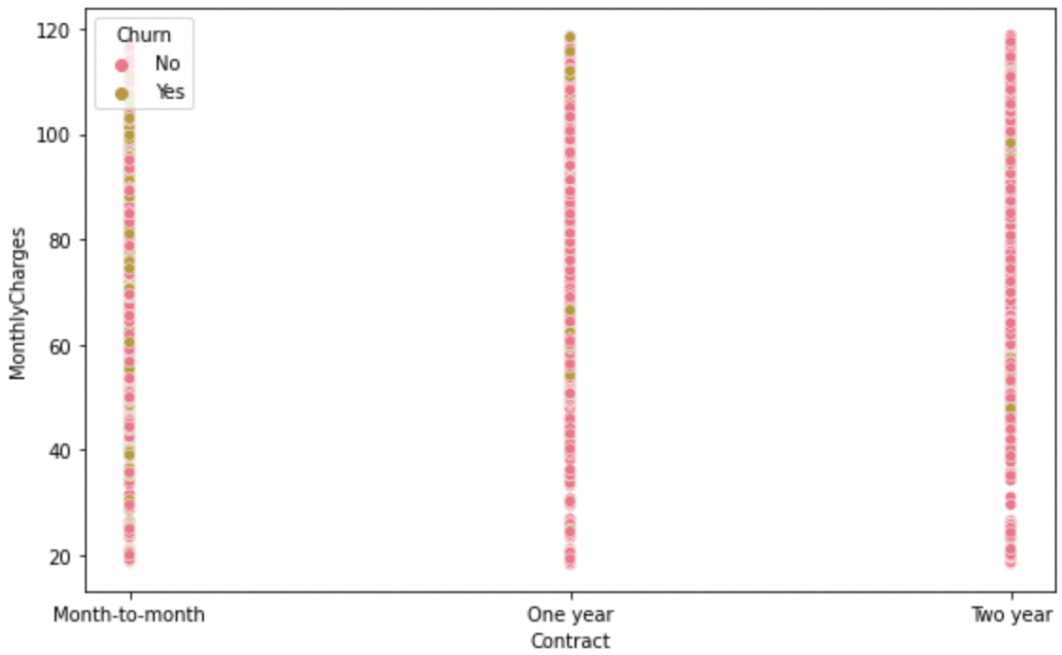


Contract Distribution:





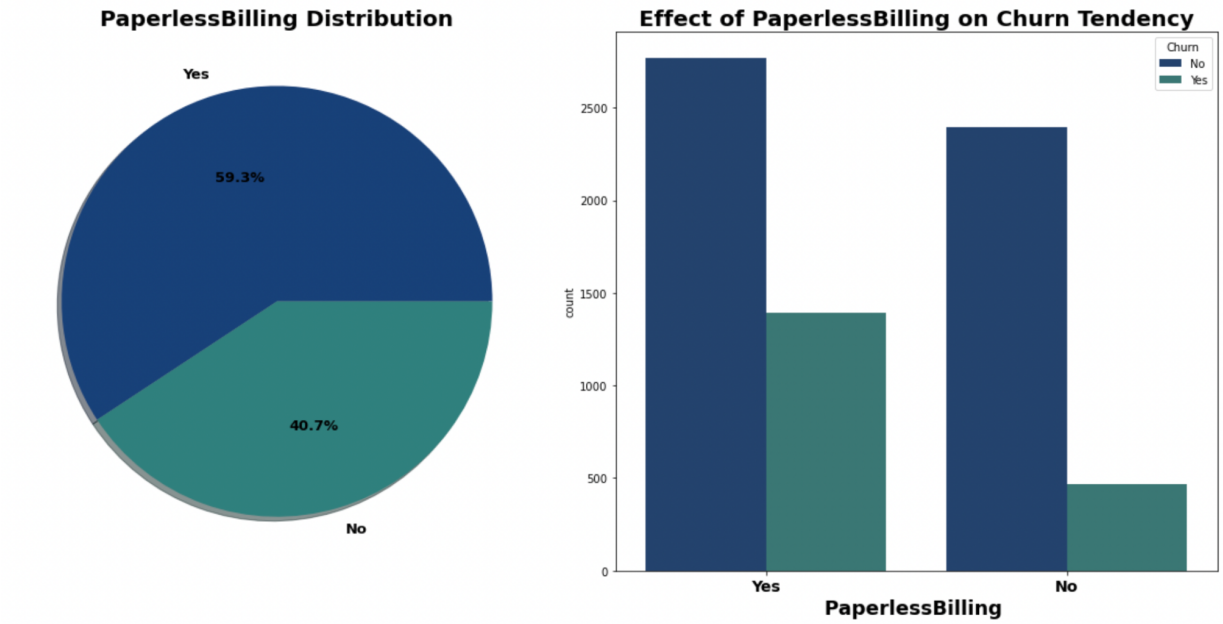
```
plt.figure(figsize=(8,5))
sns.scatterplot(x="Contract", y='MonthlyCharges',data=df,hue="Churn")
plt.show()
```



Almost 55% of customers prefer month-to-month contracts compared to others. We also find a high churn rate in these customers.

We did not find any relation between Monthly charges and contract tenure.

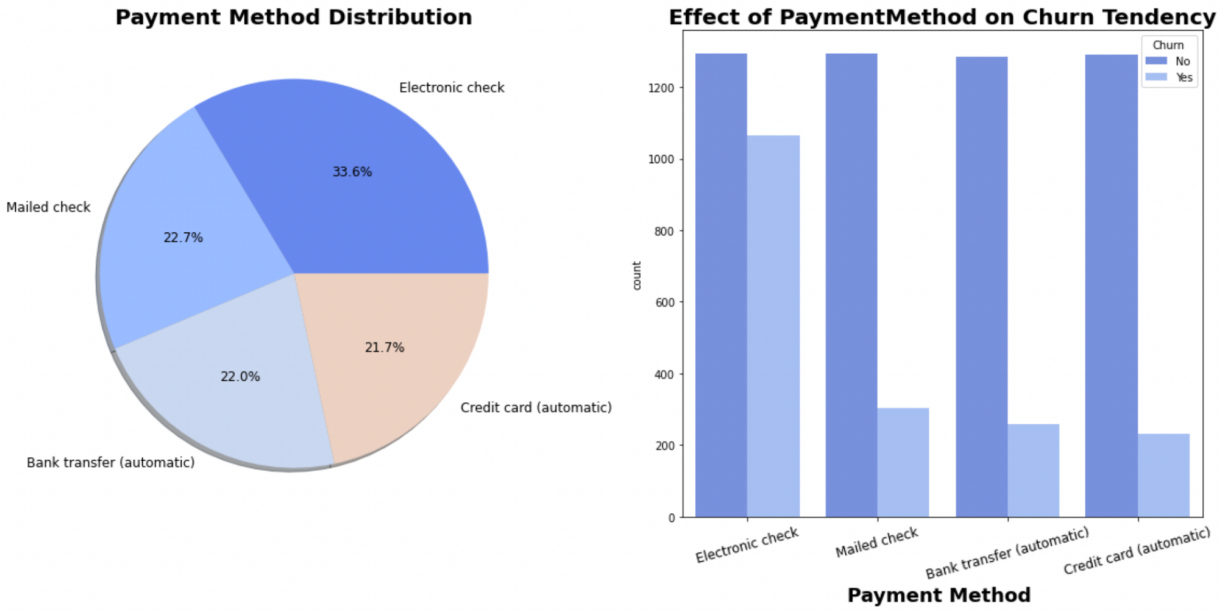
Paperless Billing Distribution:



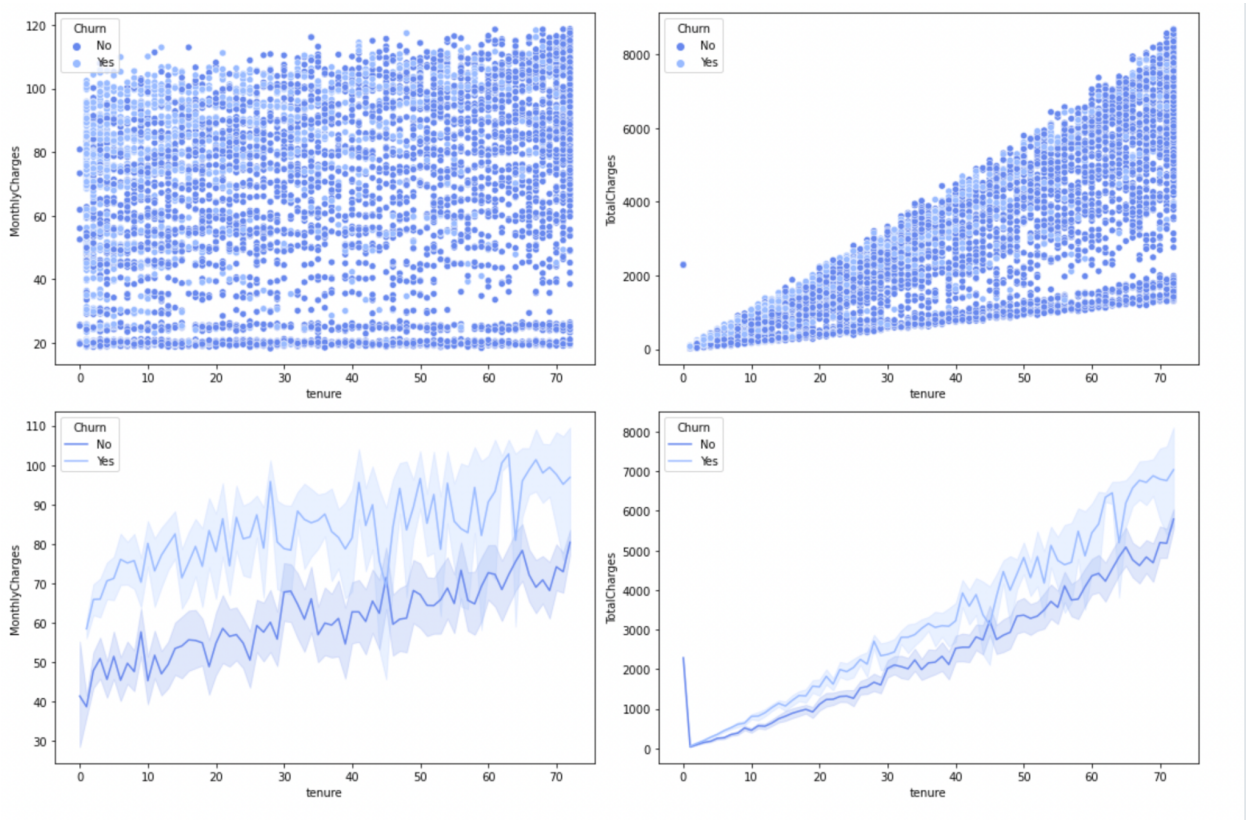
60% of Customers prefer paperless billing.

Customers who prefer paperless billing have a high churn rate.

Payment Method Distribution:

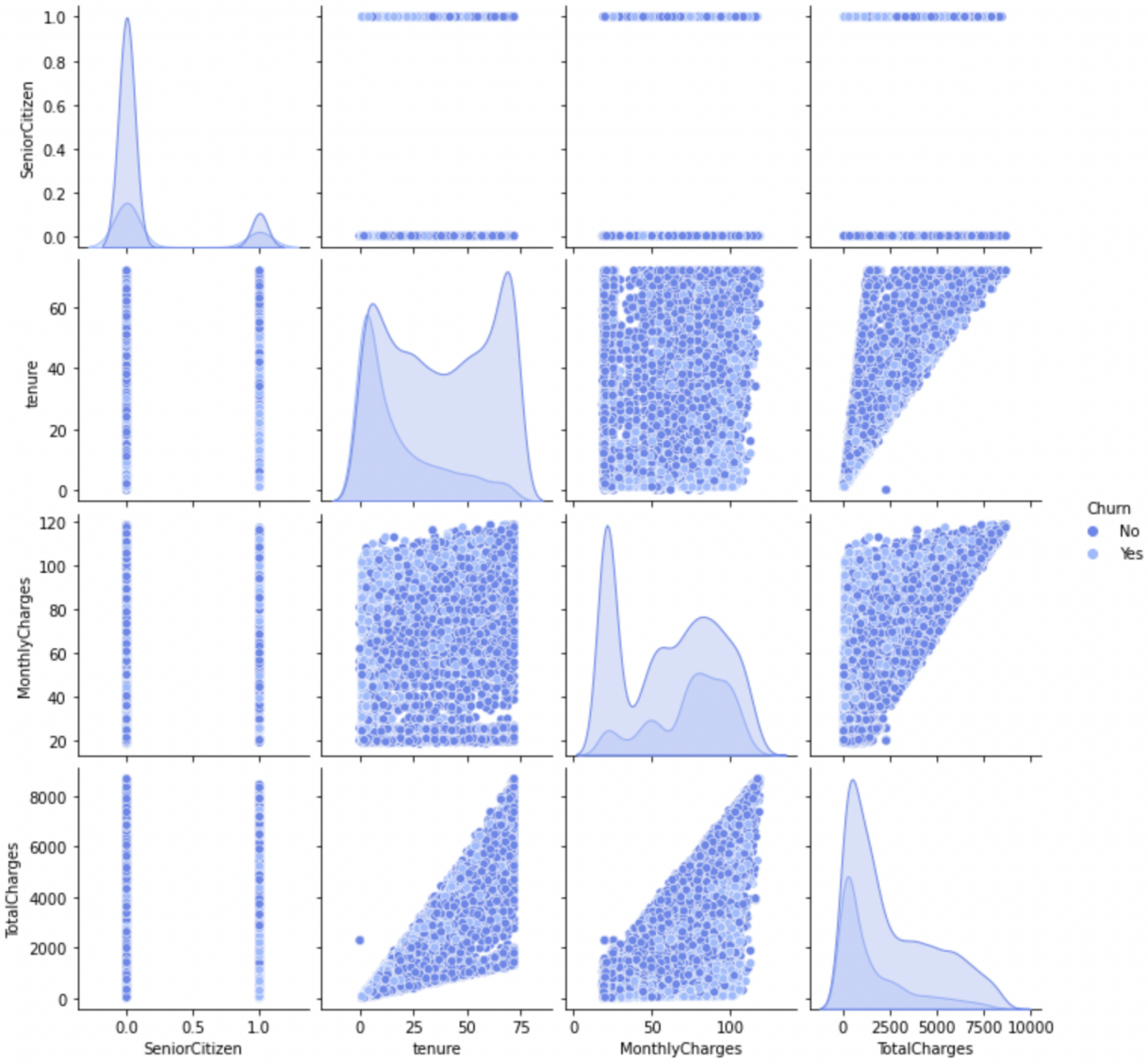


Some plots for the comparison of monthly and total charges.



High Monthly Charges for customers who choose churn compared to the rest. The same goes for High Total Charges in customers who choose churn compared to the rest.

Pair plot:


SeniorCitizen
tenure
MonthlyCharges
TotalCharges
Churn
No
Yes

Encoding categorical data

Encoding categorical data

```
df.columns.to_series().groupby(df.dtypes).groups
```

```
{int64: ['SeniorCitizen', 'tenure'], float64: ['MonthlyCharges', 'TotalCharges'], object: ['gender', 'Partner', 'Dependents', 'PhoneServic
e', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovi
es', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'Churn']}
```

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

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

```
# Using Label Encoder on categorical variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Category:
    df[i] = le.fit_transform(df[i])
df.head()
```

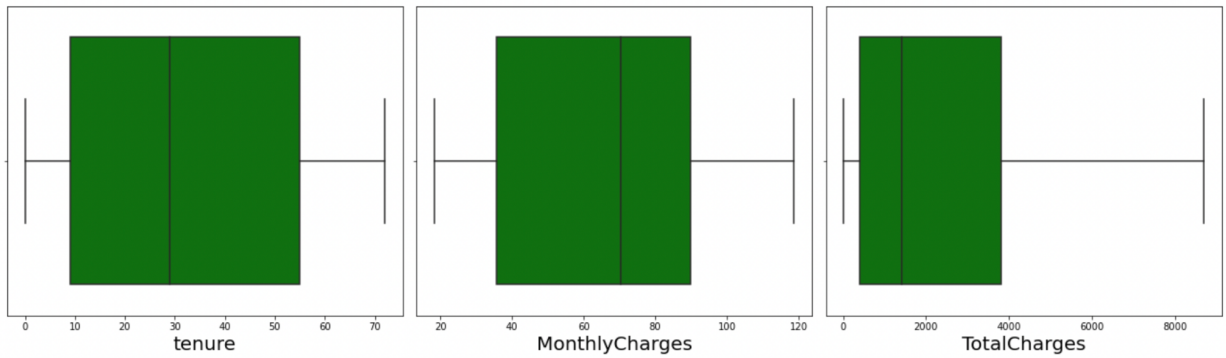
	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Str
0	0	0	1	0	1	0	1	0	0	2	0	0	
1	1	0	0	0	34	1	0	0	2	0	2	0	
2	1	0	0	0	2	1	0	0	2	2	0	0	
3	1	0	0	0	45	0	1	0	2	0	2	2	
4	0	0	0	0	2	1	0	1	0	0	0	0	

Feature selection and Engineering

Feature selection and Engineering

Outliers Detection and Removal

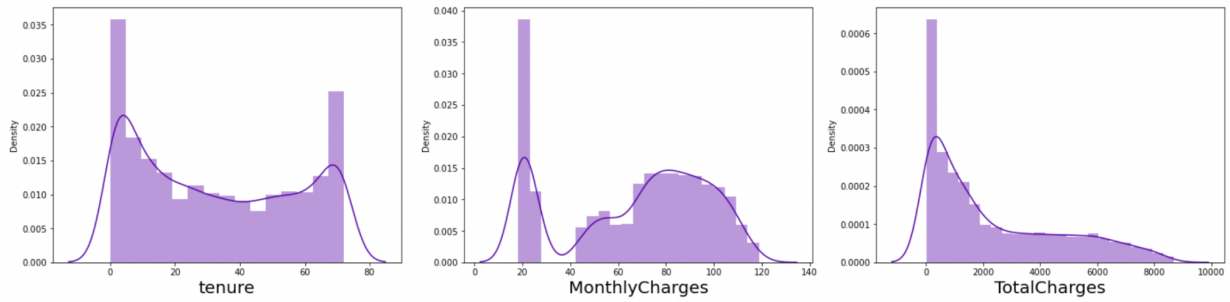
```
plt.figure(figsize=(18,10),facecolor='white')
plotnumber=1

for column in Numerical:
    if plotnumber<=6:
        ax=plt.subplot(2,3,plotnumber)
        sns.boxplot(df[column],color='g')
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.show()
```



Skewness of features

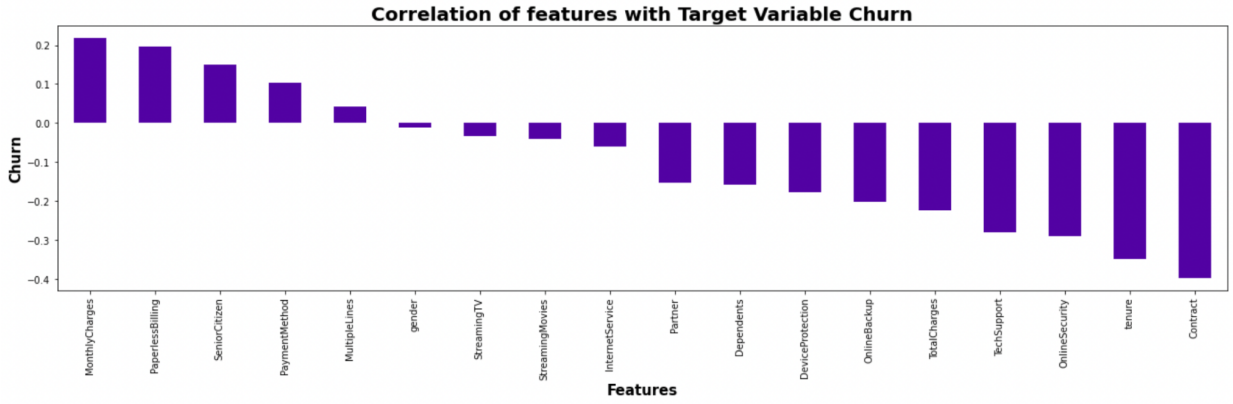
Skewness of features

```
plt.figure(figsize=(20,5),facecolor='white')
sns.set_palette('plasma')
plotnum=1
for col in Numerical:
    if plotnum<=3:
        plt.subplot(1,3,plotnum)
        sns.distplot(df[col])
        plt.xlabel(col,fontsize=20)
    plotnum+=1
plt.show()
```



Correlation of features with Target Variable Churn

```
plt.figure(figsize = (18,6))
df.corr()['Churn'].drop(['Churn']).sort_values(ascending=False).plot(kind='bar')
plt.xlabel('Features',fontsize=15,fontweight='bold')
plt.ylabel('Churn',fontsize=15,fontweight='bold')
plt.title('Correlation of features with Target Variable Churn',fontsize = 20,fontweight='bold')
plt.show()
```



Balancing Imbalanced target feature

Balanceing Imbalanced target feature

```
df.Churn.value_counts()
```

```
0    4652
1    1687
Name: Churn, dtype: int64
```

As Target variable data is Imbalanced in nature we will need to balance target variable.

Balancing using SMOTE

```
from imblearn.over_sampling import SMOTE
```

```
# Splitting data in target and dependent feature
X = df.drop(['Churn'], axis =1)
Y = df['Churn']
```

```
# Oversampleing using SMOTE Techniques
oversample = SMOTE()
X, Y = oversample.fit_resample(X, Y)
```

```
Y.value_counts()
```

```
0    4652
1    4652
Name: Churn, dtype: int64
```

Standard Scaling

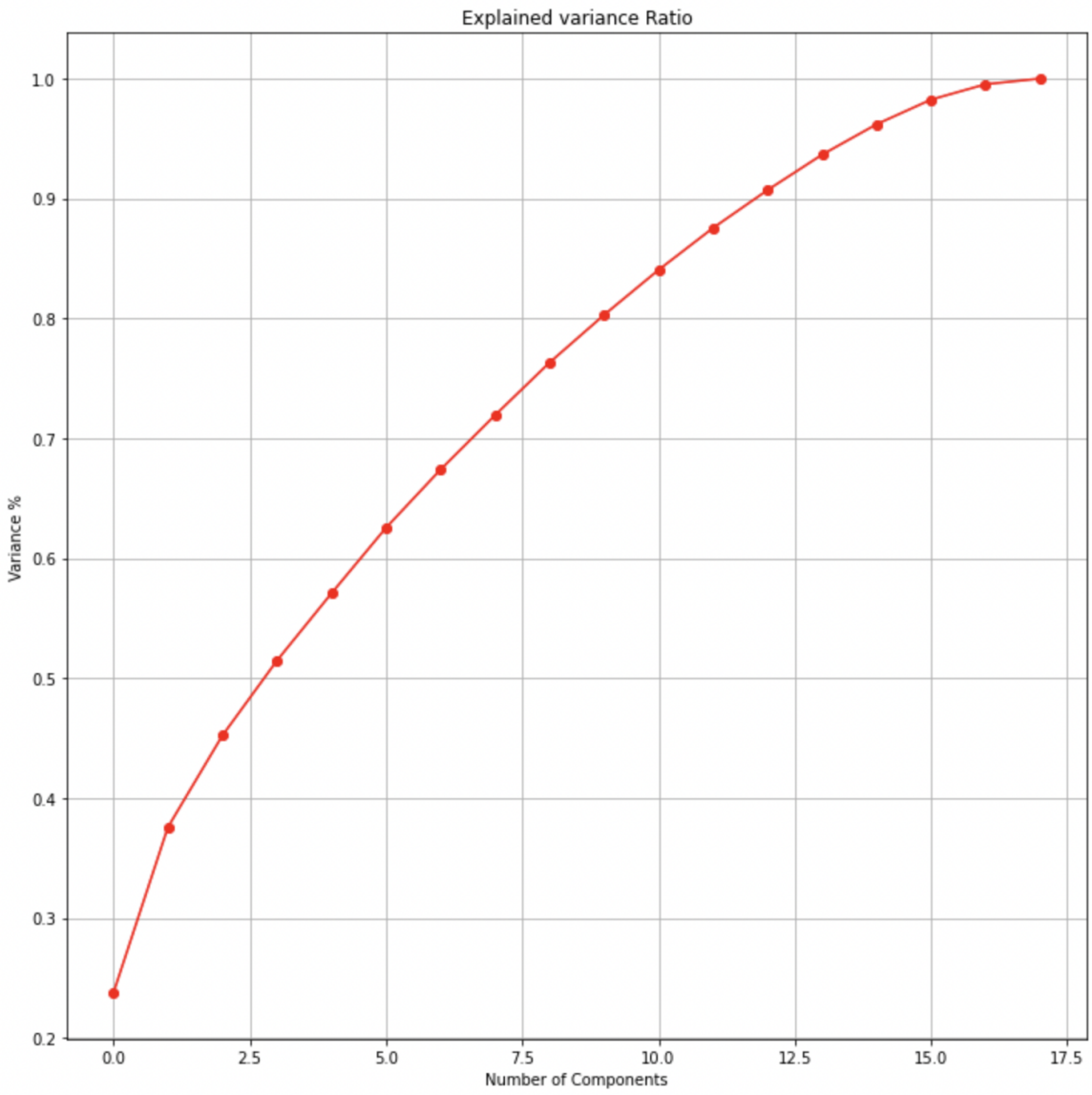
```
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()
X_scale = scaler.fit_transform(X)
```

Checking Multicollinearity between features using variance_inflation_factor

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif["VIF values"] = [variance_inflation_factor(X_scale,i) for i in range(len(X.columns))]
vif["Features"] = X.columns
vif
```

	VIF values	Features
0	1.013287	gender
1	1.100394	SeniorCitizen
2	1.550225	Partner
3	1.436131	Dependents
4	6.585733	tenure
5	1.435334	MultipleLines
6	1.471549	InternetService
7	1.336005	OnlineSecurity
8	1.242156	OnlineBackup
9	1.332055	DeviceProtection
10	1.398875	TechSupport
11	1.525723	StreamingTV
12	1.515101	StreamingMovies
13	2.545665	Contract
14	1.178300	PaperlessBilling
15	1.178373	PaymentMethod
16	3.229273	MonthlyCharges
17	6.056129	TotalCharges

PCA


Explained variance Ratio
Variance %
Number of Components
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Machine Learning Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score
```

```
X_train, X_test, Y_train, Y_test = train_test_split(principle_x, Y, random_state=101, test_size=.28)
print('Training feature matrix size:',X_train.shape)
print('Training target vector size:',Y_train.shape)
print('Test feature matrix size:',X_test.shape)
print('Test target vector size:',Y_test.shape)
```

```
Training feature matrix size: (6698, 15)
Training target vector size: (6698,)
Test feature matrix size: (2606, 15)
Test target vector size: (2606,)
```

Finding best Random state

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,f1_score
maxAccu=0
maxRS=0
for i in range(1,250):
    X_train,X_test,Y_train,Y_test = train_test_split(X_scale,Y,test_size = 0.28, random_state=i)
    log_reg=LogisticRegression()
    log_reg.fit(X_train,Y_train)
    y_pred=log_reg.predict(X_test)
    acc=accuracy_score(Y_test,y_pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print('Best accuracy is', maxAccu ,'on Random_state', maxRS)
```

```
Best accuracy is 0.8188795088257866 on Random_state 11
```

ML Algorithm	Accuracy Score	CV Mean Score
Matrix LogisticRegression	0.818879508825787	0.77966857915446
Evaluation Matrix SVC	0.826937835763623	0.79665193299937
GaussianNB	0.787413660782809	0.767738988750412
DecisionTreeClassifier	0.781657712970069	0.752905652528124
KNeighborsClassifier(n_neighbors=18)	0.792018419033001	0.785472084033905
RandomForestClassifier	0.852264006139	0.803746396029421
ExtraTreesClassifier	0.841519570222563	0.799124184592629

So as per the data from the above table Random forest is best performing model.

Hyper Parameter Tuning: GridSearchCV

```
GCV.best_params_
```

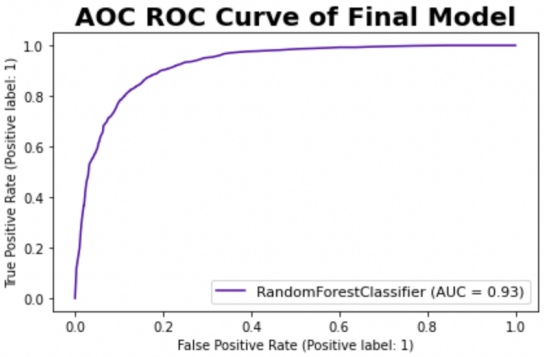
```
{'bootstrap': True,
 'criterion': 'gini',
 'max_depth': 50,
 'max_features': 'auto',
 'n_estimators': 60}
```

Final Model

```
Final_mod = RandomForestClassifier(bootstrap=True,criterion='gini',n_estimators= 60, max_depth=50 ,max_features='auto')
Final_mod.fit(X_train,Y_train)
y_pred=Final_mod.predict(X_test)
print('Accuracy Score :','\n', accuracy_score(Y_test, y_pred))
```

```
Accuracy Score :
 0.8549501151189562
```

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import plot_roc_curve
disp = plot_roc_curve(Final_mod,X_test,Y_test)
plt.legend(prop={'size':11}, loc='lower right')
plt.title('AOC ROC Curve of Final Model',fontsize=20,fontweight='bold')
plt.show()
auc_score = roc_auc_score(Y_test, Final_mod.predict(X_test))
print('\033[1m'+'Auc Score :'+'\033[0m\n',auc_score)
```




```
Auc Score :
 0.8545231128280837
```

Saving model

```
import joblib
joblib.dump(Final_mod,'Customer_Churn.pkl')
```

```
['Customer_Churn.pkl']
```

Predicting the Final Model

```
# Prediction
prediction = Final_mod.predict(X_test)
```

```
Actual = np.array(Y_test)
df_Pred = pd.DataFrame()
df_Pred["Predicted Values"] = prediction
df_Pred["Actual Values"] = Actual
df_Pred.head()
```

	Predicted Values	Actual Values
0	1	1
1	1	1
2	0	0
3	1	1
4	0	0

Conclusion:

You may be wondering why it's necessary to calculate the churn rate. Naturally, you're going to lose some customers here and there, and 5% doesn't sound too bad, right?

Well, it's important because it costs more to acquire new customers than it does to retain existing customers. In fact, an increase in customer retention of just 5% can create at least a 25% increase in profit. This is because returning customers will likely spend 67% more on your company's products and services. As a result, your company can spend less on the operating costs of having to acquire new customers. You don't need to spend time and money on convincing an existing customer to select your company over competitors because they've already made that decision.

Again, it might seem like a 5% churn rate is solid and healthy. You can still make a vast revenue with that churn rate. However, consider the example below when thinking about the impact of your churn rate.

In this example, simply decreasing your churn rate by 10% could add an extra $100,000 in revenue to your company. Dropping from 3% to 2.7% doesn't seem like a lot, but it actually adds large benefits to your company

How to Reduce Customer Churn

1. Focus your attention on your best customers.

Rather than simply focusing on offering incentives to customers who are considering churning, it could be even more beneficial to pool your resources into your loyal, profitable customers.

2. Analyze churn as it occurs.

Use your churned customers as a means of understanding why customers are leaving. Analyze how and when churn occurs in a customer's lifetime with your company, and use that data to put into place preemptive measures.

3. Show your customers that you care.

Instead of waiting to connect with your customers until they reach out to you, try a more proactive approach. Communicate with them all the perks you offer and show them you care about their experience, and they'll be sure to stick around.