

✓ EDA and Prediction


Churn is a one of the biggest problem in the telecom industry. Research has shown that the average monthly churn rate among the top 4 wireless carriers in the US is 1.9% - 2%.

```
1 import numpy as np # linear algebra
2 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
3 import seaborn as sns # For creating plots
4 import matplotlib.ticker as mtick # For specifying the axes tick format
5 import matplotlib.pyplot as plt
6
```

Let us read the data file in the python notebook

```
1 telecom_cust = pd.read_csv('/content/Telco-Customer-Churn.csv')
```

```
1 telecom_cust.head()
```



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

5 rows × 21 columns

```
1 telecom_cust.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')
```

Let's explore the data to see if there are any missing values.

```
1 # Checking the data types of all the columns
2 telecom_cust.dtypes
```

```
customerID      object
gender          object
SeniorCitizen    int64
Partner         object
Dependents      object
tenure          int64
PhoneService    object
MultipleLines    object
InternetService  object
OnlineSecurity   object
OnlineBackup     object
DeviceProtection object
TechSupport      object
StreamingTV      object
StreamingMovies  object
Contract         object
PaperlessBilling object
PaymentMethod    object
MonthlyCharges   float64
TotalCharges     object
Churn            object
dtype: object
```

```

1 # Converting Total Charges to a numerical data type.
2 telecom_cust.TotalCharges = pd.to_numeric(telecom_cust.TotalCharges, errors='coerce')
3 telecom_cust.isnull().sum()

```

```

customerID      0
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

```

After looking at the above output, we can say that there are 11 missing values for Total Charges. Let us replace remove these 11 rows from our data set

```

1 #Removing missing values
2 telecom_cust.dropna(inplace = True)
3 #Remove customer IDs from the data set
4 df2 = telecom_cust.iloc[:,1:]
5 #Convertin the predictor variable in a binary numeric variable
6 df2['Churn'].replace(to_replace='Yes', value=1, inplace=True)
7 df2['Churn'].replace(to_replace='No', value=0, inplace=True)
8
9 #Let's convert all the categorical variables into dummy variables
10 df_dummies = pd.get_dummies(df2)
11 df_dummies.head()

```

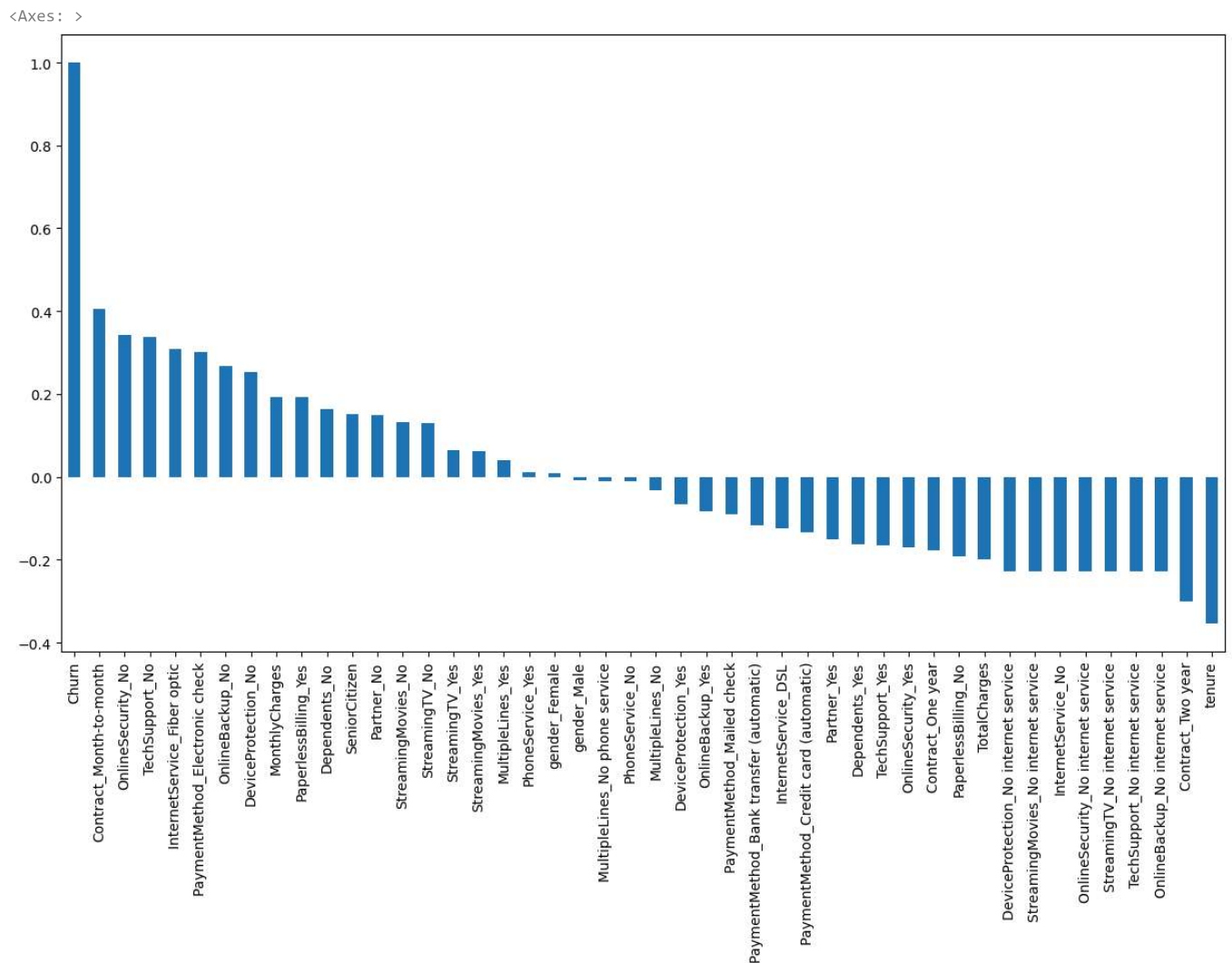
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	
0	0	1	29.85	29.85	0	True	False	False	True	True	
1	0	34	56.95	1889.50	0	False	True	True	False	True	
2	0	2	53.85	108.15	1	False	True	True	False	True	
3	0	45	42.30	1840.75	0	False	True	True	False	True	
4	0	2	70.70	151.65	1	True	False	True	False	True	

5 rows × 46 columns

```

1 #Get Correlation of "Churn" with other variables:
2 plt.figure(figsize=(15,8))
3 df_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')

```



Month to month contracts, absence of online security and tech support seem to be positively correlated with churn. While, tenure, two year contracts seem to be negatively correlated with churn.

Interestingly, services such as Online security, streaming TV, online backup, tech support, etc. without internet connection seem to be negatively related to churn.

We will explore the patterns for the above correlations below before we delve into modelling and identifying the important variables.

▼ Data Exploration

Let us first start with exploring our data set, to better understand the patterns in the data and potentially form some hypothesis. First we will look at the distribution of individual variables and then slice and dice our data for any interesting trends.

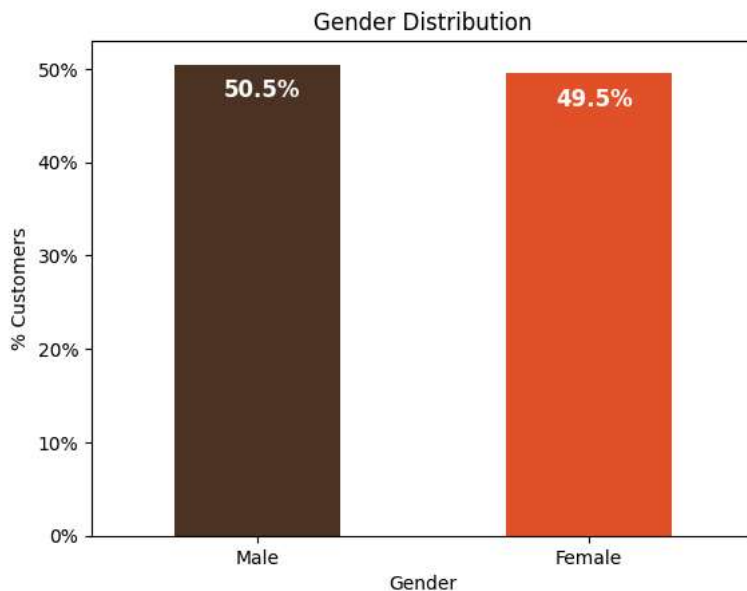
A.) Demographics - Let us first understand the gender, age range, patner and dependent status of the customers

1. **Gender Distribution** - About half of the customers in our data set are male while the other half are female

```

1 colors = ['#4D3425', '#E4512B']
2 ax = (telecom_cust['gender'].value_counts()*100.0 /len(telecom_cust)).plot(kind='bar',
3                                     stacked = True,
4                                     rot = 0,
5                                     color = colors)
6 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
7 ax.set_ylabel('% Customers')
8 ax.set_xlabel('Gender')
9 ax.set_ylabel('% Customers')
10 ax.set_title('Gender Distribution')
11
12 # create a list to collect the plt.patches data
13 totals = []
14
15 # find the values and append to list
16 for i in ax.patches:
17     totals.append(i.get_width())
18
19 # set individual bar labels using above list
20 total = sum(totals)
21
22 for i in ax.patches:
23     # get_width pulls left or right; get_y pushes up or down
24     ax.text(i.get_x()+.15, i.get_height()-3.5, \
25             str(round((i.get_height()/total), 1))+'%',
26             fontsize=12,
27             color='white',
28             weight = 'bold')

```



2. **% Senior Citizens** - There are only 16% of the customers who are senior citizens. Thus most of our customers in the data are younger people.

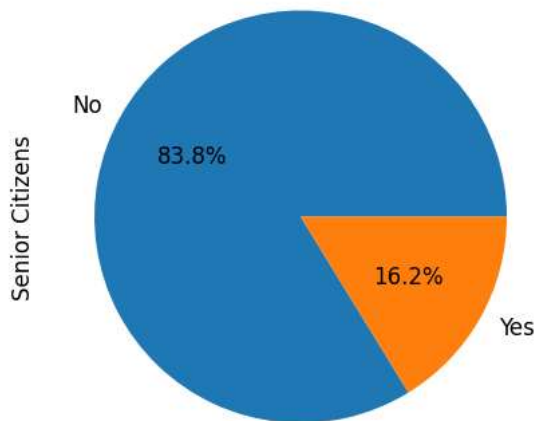
```

1 ax = (telecom_cust['SeniorCitizen'].value_counts()*100.0 /len(telecom_cust))\
2 .plot.pie(autopct='%1f%%', labels = ['No', 'Yes'],figsize =(5,5), fontsize = 12 )
3 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
4 ax.set_ylabel('Senior Citizens',fontsize = 12)
5 ax.set_title('% of Senior Citizens', fontsize = 12)

```

Text(0.5, 1.0, '% of Senior Citizens')

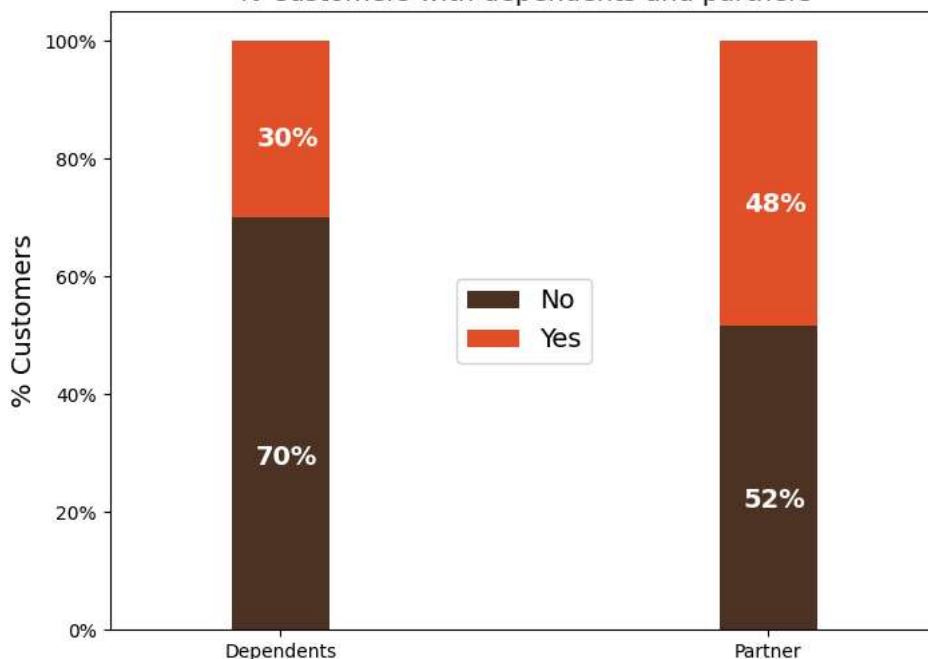
% of Senior Citizens



3. **Partner and dependent status** - About 50% of the customers have a partner, while only 30% of the total customers have dependents.

```
1 df2 = pd.melt(telecom_cust, id_vars=['customerID'], value_vars=['Dependents', 'Partner'])
2 df3 = df2.groupby(['variable', 'value']).count().unstack()
3 df3 = df3*100/len(telecom_cust)
4 colors = ['#4D3425', '#E4512B']
5 ax = df3.loc[:, 'customerID'].plot.bar(stacked=True, color=colors,
6                                       figsize=(8,6), rot = 0,
7                                       width = 0.2)
8
9 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
10 ax.set_ylabel('% Customers', size = 14)
11 ax.set_xlabel('')
12 ax.set_title('% Customers with dependents and partners', size = 14)
13 ax.legend(loc = 'center', prop={'size':14})
14
15 for p in ax.patches:
16     width, height = p.get_width(), p.get_height()
17     x, y = p.get_xy()
18     ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),
19               color = 'white',
20               weight = 'bold',
21               size = 14)
```

% Customers with dependents and partners

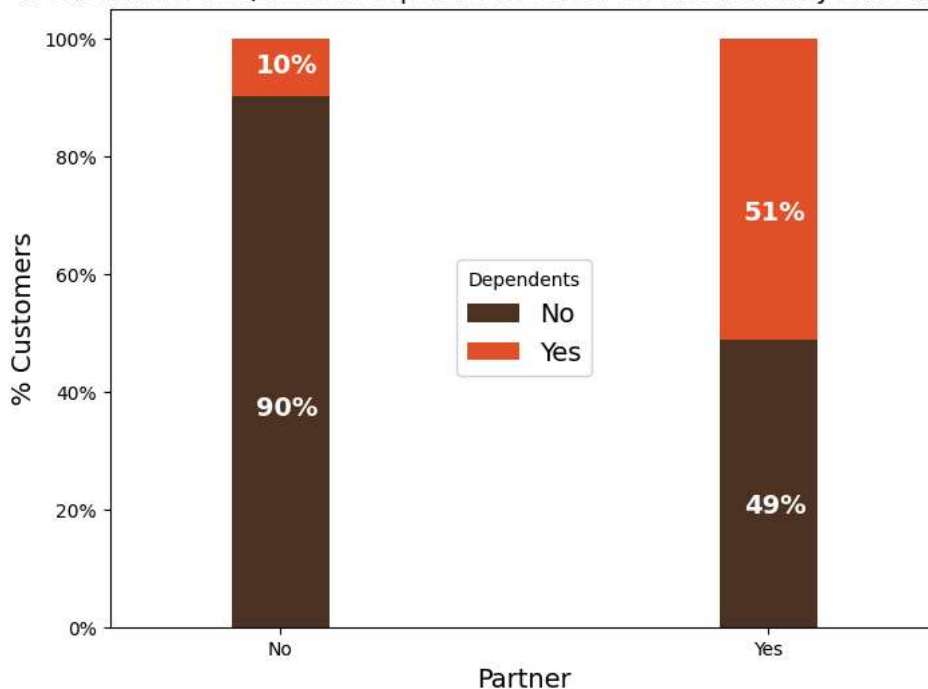


*What would be interesting is to look at the % of customers, who have partners, also have dependents. We will explore this next. *

Interestingly, among the customers who have a partner, only about half of them also have a dependent, while other half do not have any dependents. Additionally, as expected, among the customers who do not have any partner, a majority (80%) of them do not have any dependents.

```
1 colors = ['#4D3425', '#E4512B']
2 partner_dependents = telecom_cust.groupby(['Partner', 'Dependents']).size().unstack()
3
4 ax = (partner_dependents.T*100.0 / partner_dependents.T.sum()).T.plot(kind='bar',
5                                     width = 0.2,
6                                     stacked = True,
7                                     rot = 0,
8                                     figsize = (8,6),
9                                     color = colors)
10 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
11 ax.legend(loc='center',prop={'size':14},title = 'Dependents',fontsize =14)
12 ax.set_ylabel('% Customers',size = 14)
13 ax.set_title('% Customers with/without dependents based on whether they have a partner',size = 14)
14 ax.xaxis.label.set_size(14)
15
16 # Code to add the data labels on the stacked bar chart
17 for p in ax.patches:
18     width, height = p.get_width(), p.get_height()
19     x, y = p.get_xy()
20     ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),
21               color = 'white',
22               weight = 'bold',
23               size = 14)
```

% Customers with/without dependents based on whether they have a partner



I also looked at any differences between the % of customers with/without dependents and partners by gender. There is no difference in their distribution by gender. Additionally, there is no difference in senior citizen status by gender.

✓ B.) Customer Account Information: Let u now look at the tenure, contract

1. Tenure: After looking at the below histogram we can see that a lot of customers have been with the telecom company for just a month, while quite a many are there for about 72 months. This could be potentially because different customers have different contracts. Thus based on the contract they are into it could be more/less easier for the customers to stay/leave the telecom company.

```
1 ax = sns.distplot(telecom_cust['tenure'], hist=True, kde=False,
2                   bins=int(180/5), color = 'darkblue',
3                   hist_kws={'edgecolor':'black'},
4                   kde_kws={'linewidth': 4})
5 ax.set_ylabel('# of Customers')
6 ax.set_xlabel('Tenure (months)')
7 ax.set_title('# of Customers by their tenure')
```

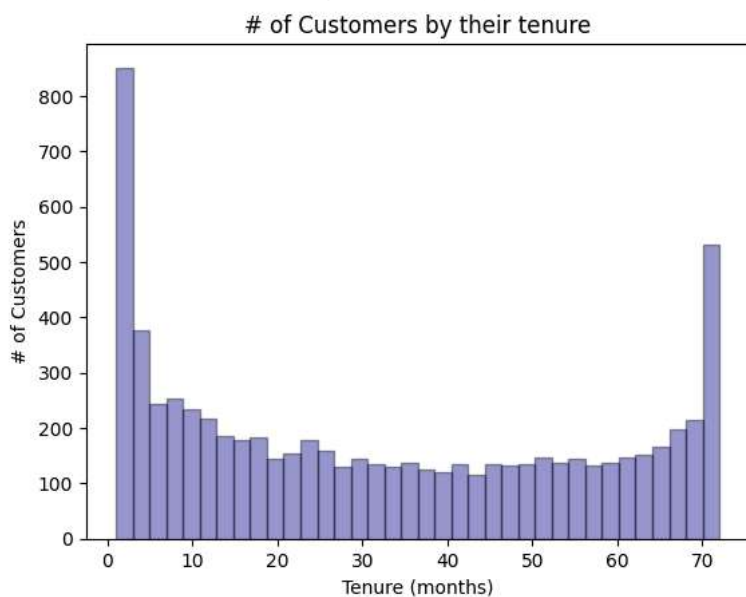
<ipython-input-14-22c5657c788f>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

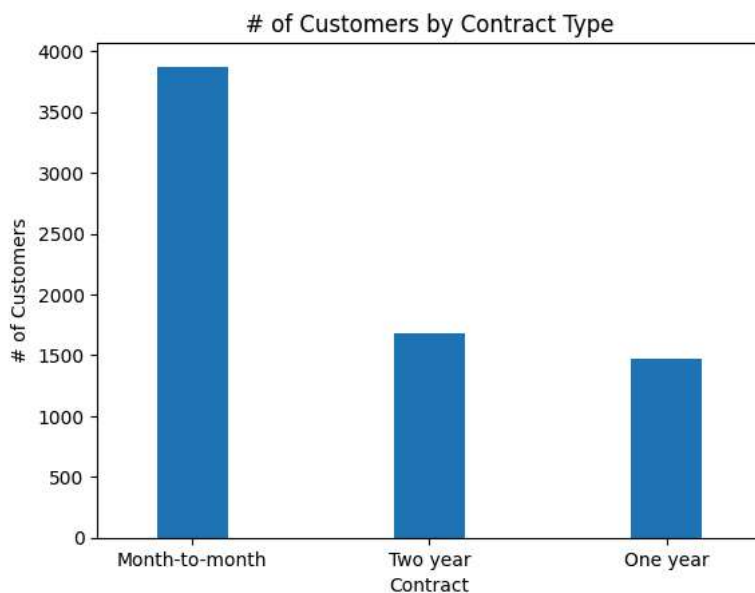
```
ax = sns.distplot(telecom_cust['tenure'], hist=True, kde=False,
Text(0.5, 1.0, '# of Customers by their tenure')
```



2. Contracts: To understand the above graph, lets first look at the # of customers by different contracts.

```
1 ax = telecom_cust['Contract'].value_counts().plot(kind = 'bar',rot = 0, width = 0.3)
2 ax.set_ylabel('# of Customers')
3 ax.set_title('# of Customers by Contract Type')
```

```
Text(0.5, 1.0, '# of Customers by Contract Type')
```



As we can see from this graph most of the customers are in the month to month contract. While there are equal number of customers in the 1 year and 2 year contracts.

Below we will understand the tenure of customers based on their contract type.

```

1 fig, (ax1,ax2,ax3) = plt.subplots(nrows=1, ncols=3, sharey = True, figsize = (20,6))
2
3 ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='Month-to-month']['tenure'],
4                   hist=True, kde=False,
5                   bins=int(180/5), color = 'turquoise',
6                   hist_kws={'edgecolor':'black'},
7                   kde_kws={'linewidth': 4},
8                   ax=ax1)
9 ax.set_ylabel('# of Customers')
10 ax.set_xlabel('Tenure (months)')
11 ax.set_title('Month to Month Contract')
12
13 ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='One year']['tenure'],
14                   hist=True, kde=False,
15                   bins=int(180/5), color = 'steelblue',
16                   hist_kws={'edgecolor':'black'},
17                   kde_kws={'linewidth': 4},
18                   ax=ax2)
19 ax.set_xlabel('Tenure (months)',size = 14)
20 ax.set_title('One Year Contract',size = 14)
21
22 ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='Two year']['tenure'],
23                   hist=True, kde=False,
24                   bins=int(180/5), color = 'darkblue',
25                   hist_kws={'edgecolor':'black'},
26                   kde_kws={'linewidth': 4},
27                   ax=ax3)
28
29 ax.set_xlabel('Tenure (months)')
30 ax.set_title('Two Year Contract')

```



```
<ipython-input-16-5c4ebb8bfebf>:3: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='Month-to-month']['tenure'],
<ipython-input-16-5c4ebb8bfebf>:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

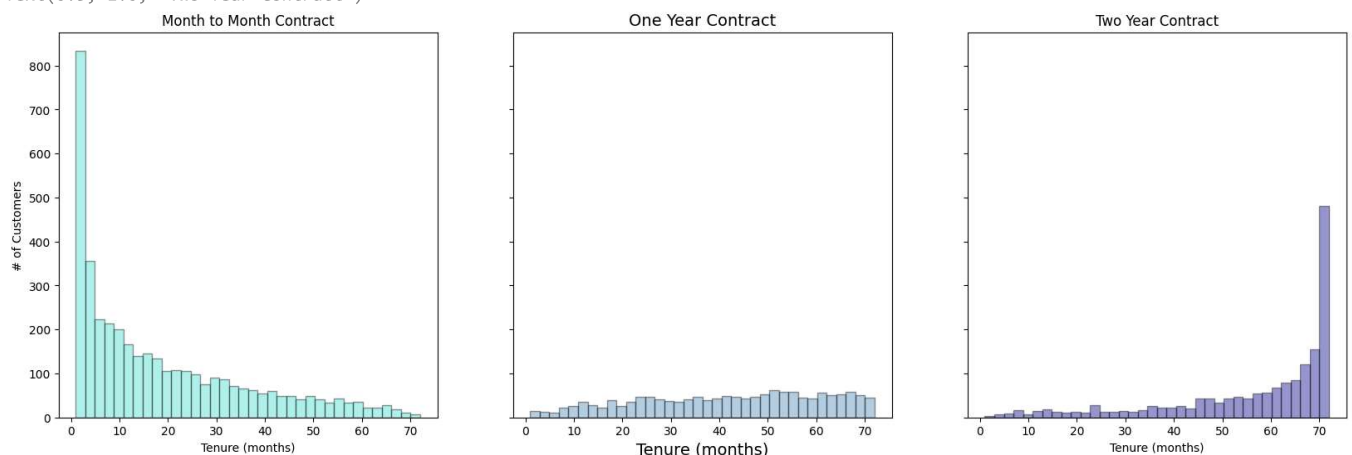
ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='One year']['tenure'],
<ipython-input-16-5c4ebb8bfebf>:22: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='Two year']['tenure'],
Text(0.5, 1.0, 'Two Year Contract')
```



Interestingly most of the monthly contracts last for 1-2 months, while the 2 year contracts tend to last for about 70 months. This shows that the customers taking a longer contract are more loyal to the company and tend to stay with it for a longer period of time.

This is also what we saw in the earlier chart on correlation with the churn rate.

✓ C. Let us now look at the distribution of various services used by customers

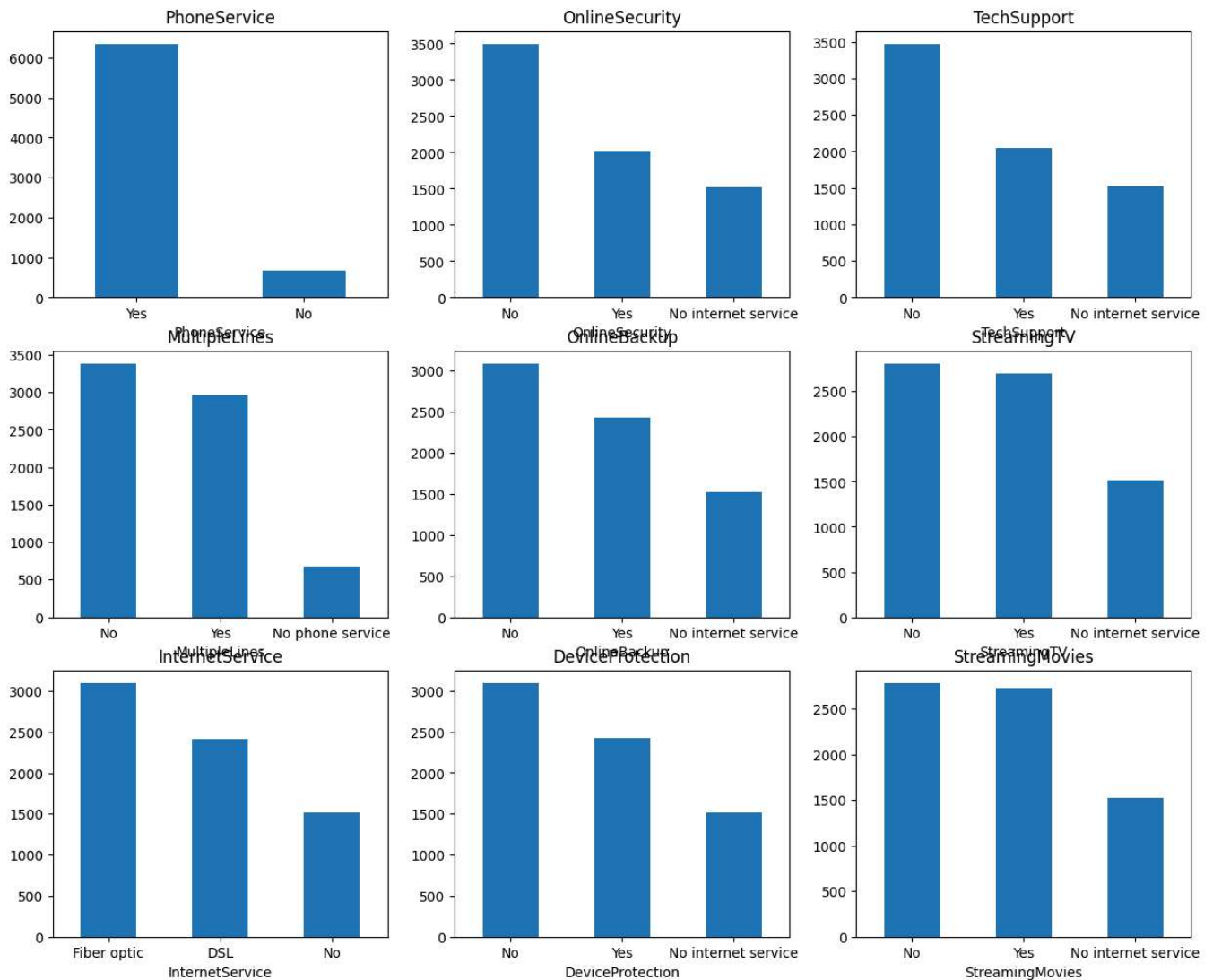
```
1
2 telecom_cust.columns.values

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

```

1 services = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
2             'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']
3
4 fig, axes = plt.subplots(nrows = 3,ncols = 3,figsize = (15,12))
5 for i, item in enumerate(services):
6     if i < 3:
7         ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i,0],rot = 0)
8
9     elif i >=3 and i < 6:
10         ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i-3,1],rot = 0)
11
12     elif i < 9:
13         ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i-6,2],rot = 0)
14     ax.set_title(item)

```

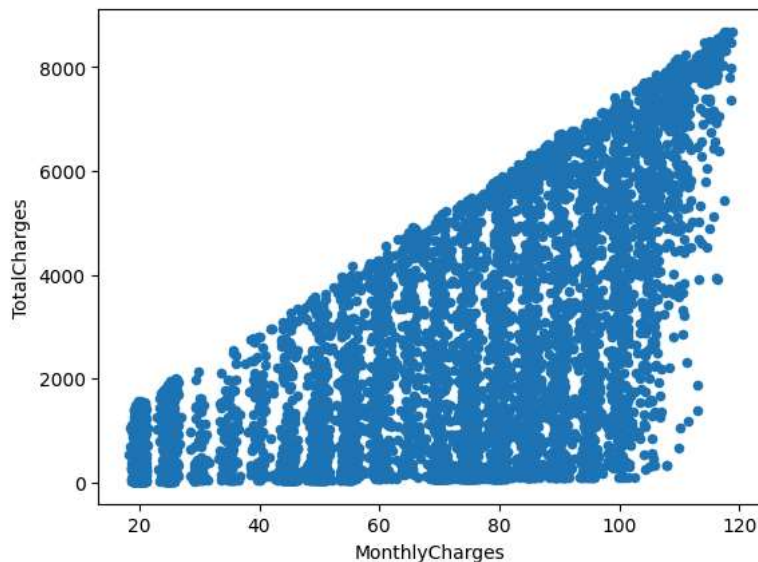


- ✓ D.) Now let's take a quick look at the relation between monthly and total charges

We will observe that the total charges increases as the monthly bill for a customer increases.

[illegible]

<Axes: xlabel='MonthlyCharges', ylabel='TotalCharges'>



- ✓ E.) Finally, let's take a look at our predictor variable (Churn) and understand its interaction with other important variables as was found out in the correlation plot.

1. Let's first look at the churn rate in our data

```
1 colors = ['#4D3425', '#E4512B']
2 ax = (telecom_cust['Churn'].value_counts()*100.0 / len(telecom_cust)).plot(kind='bar',
3                                     stacked = True,
4                                     rot = 0,
5                                     color = colors,
6                                     figsize = (8,6))
7 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
8 ax.set_ylabel('% Customers', size = 14)
9 ax.set_xlabel('Churn', size = 14)
10 ax.set_title('Churn Rate', size = 14)
11
12 # create a list to collect the plt.patches data
13 totals = []
14
15 # find the values and append to list
16 for i in ax.patches:
17     totals.append(i.get_width())
18
19 # set individual bar labels using above list
20 total = sum(totals)
21
22 for i in ax.patches:
23     # get_width pulls left or right; get_y pushes up or down
24     ax.text(i.get_x()+.15, i.get_height()-4.0, \
25             str(round((i.get_height()/total), 1))+'%',
26             fontsize=12,
27             color='white',
28             weight = 'bold',
29             size = 14)
```

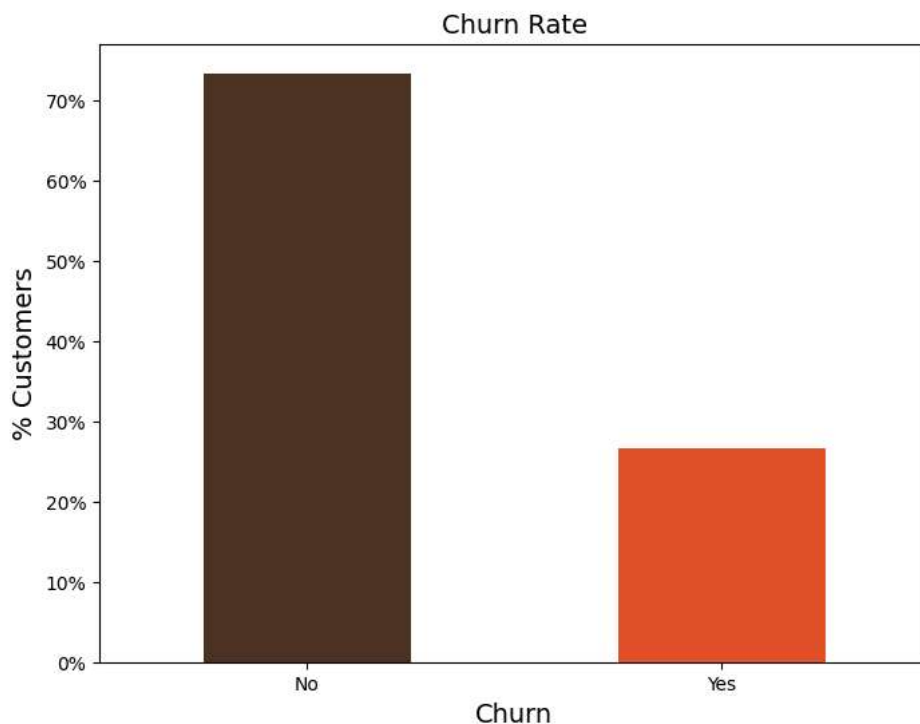
```

-----
TypeError                                Traceback (most recent call last)
<ipython-input-20-f3a5e1e0626c> in <cell line: 22>()
    22 for i in ax.patches:
    23     # get_width pulls left or right; get_y pushes up or down
--> 24     ax.text(i.get_x()+.15, i.get_height()-4.0, \
    25             str(round((i.get_height()/total), 1))+ '%',
    26             fontsize=12,

4 frames
/usr/local/lib/python3.10/dist-packages/matplotlib/cbook/__init__.py in normalize_kwargs(kw, alias_mapping)
    1772     canonical = to_canonical.get(k, k)
    1773     if canonical in canonical_to_seen:
-> 1774         raise TypeError(f"Got both {canonical_to_seen[canonical]!r} and "
    1775                         f"{k!r}, which are aliases of one another")
    1776     canonical_to_seen[canonical] = k

TypeError: Got both 'fontsize' and 'size', which are aliases of one another

```



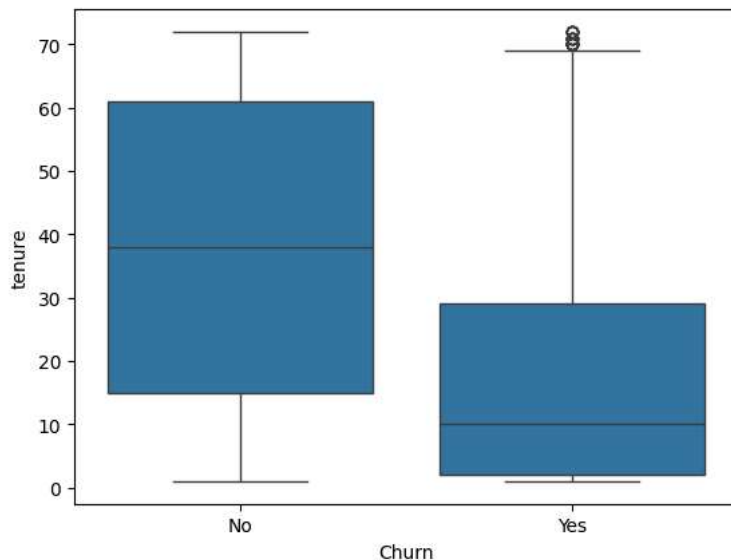
In our data, 74% of the customers do not churn. Clearly the data is skewed as we would expect a large majority of the customers to not churn. This is important to keep in mind for our modelling as skewness could lead to a lot of false negatives. We will see in the modelling section on how to avoid skewness in the data.

2. Lets now explore the churn rate by tenure, seniority, contract type, monthly charges and total charges to see how it varies by these variables.

i.) **Churn vs Tenure:** As we can see form the below plot, the customers who do not churn, they tend to stay for a longer tenure with the telecom company.

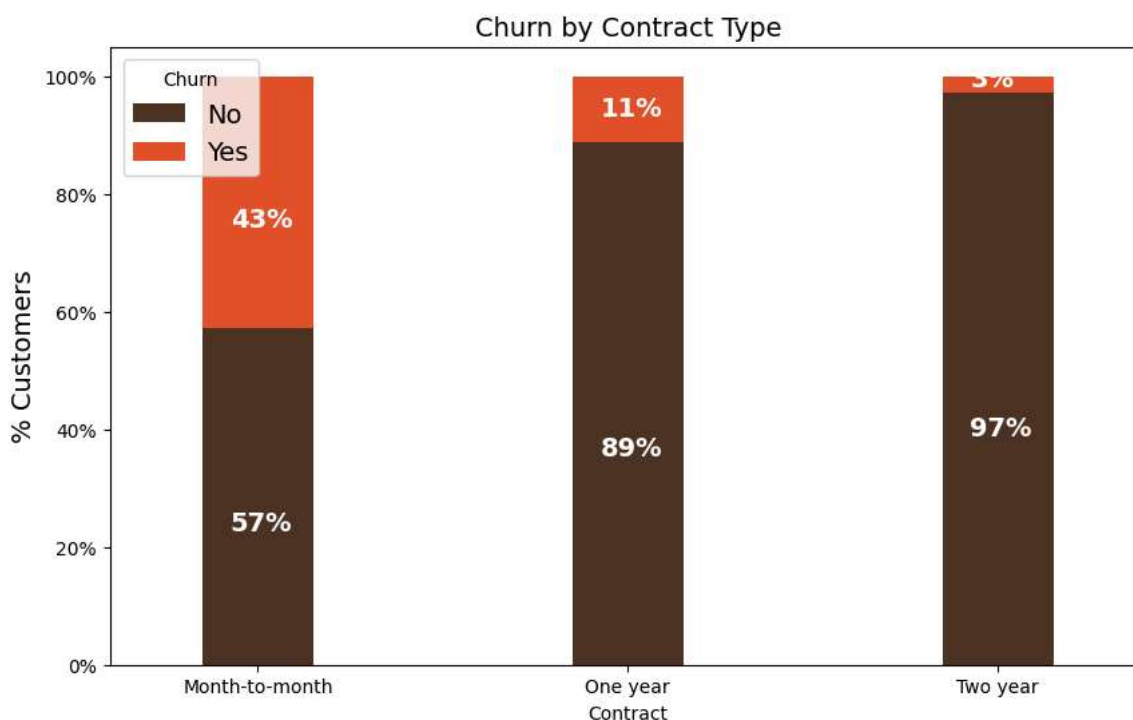
```
1 sns.boxplot(x = telecom_cust.Churn, y = telecom_cust.tenure)
```

<Axes: xlabel='Churn', ylabel='tenure'>



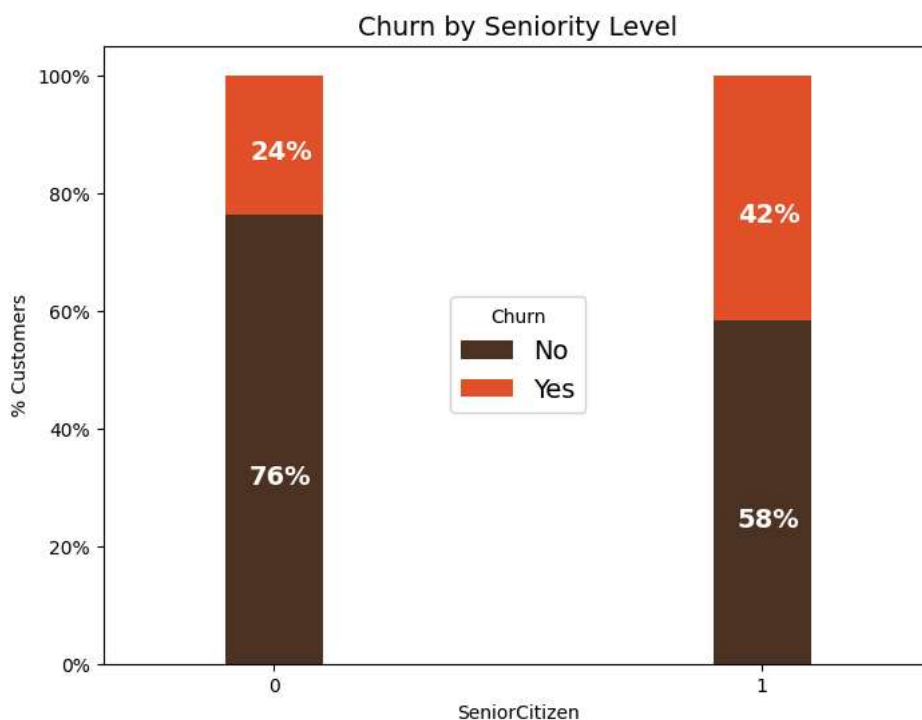
ii.) **Churn by Contract Type:** Similar to what we saw in the correlation plot, the customers who have a month to month contract have a very high churn rate.

```
1 colors = ['#4D3425', '#E4512B']
2 contract_churn = telecom_cust.groupby(['Contract', 'Churn']).size().unstack()
3
4 ax = (contract_churn.T*100.0 / contract_churn.T.sum()).T.plot(kind='bar',
5                                     width = 0.3,
6                                     stacked = True,
7                                     rot = 0,
8                                     figsize = (10,6),
9                                     color = colors)
10 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
11 ax.legend(loc='best',prop={'size':14},title = 'Churn')
12 ax.set_ylabel('% Customers',size = 14)
13 ax.set_title('Churn by Contract Type',size = 14)
14
15 # Code to add the data labels on the stacked bar chart
16 for p in ax.patches:
17     width, height = p.get_width(), p.get_height()
18     x, y = p.get_xy()
19     ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),
20             color = 'white',
21             weight = 'bold',
22             size = 14)
```



iii.) **Churn by Seniority:** Senior Citizens have almost double the churn rate than younger population.

```
1 colors = ['#4D3425', '#E4512B']
2 seniority_churn = telecom_cust.groupby(['SeniorCitizen', 'Churn']).size().unstack()
3
4 ax = (seniority_churn.T*100.0 / seniority_churn.T.sum()).T.plot(kind='bar',
5                                     width = 0.2,
6                                     stacked = True,
7                                     rot = 0,
8                                     figsize = (8,6),
9                                     color = colors)
10 ax.yaxis.set_major_formatter(mtick.PercentFormatter())
11 ax.legend(loc='center',prop={'size':14},title = 'Churn')
12 ax.set_ylabel('% Customers')
13 ax.set_title('Churn by Seniority Level',size = 14)
14
15 # Code to add the data labels on the stacked bar chart
16 for p in ax.patches:
17     width, height = p.get_width(), p.get_height()
18     x, y = p.get_xy()
19     ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),
20               color = 'white',
21               weight = 'bold',size =14)
```



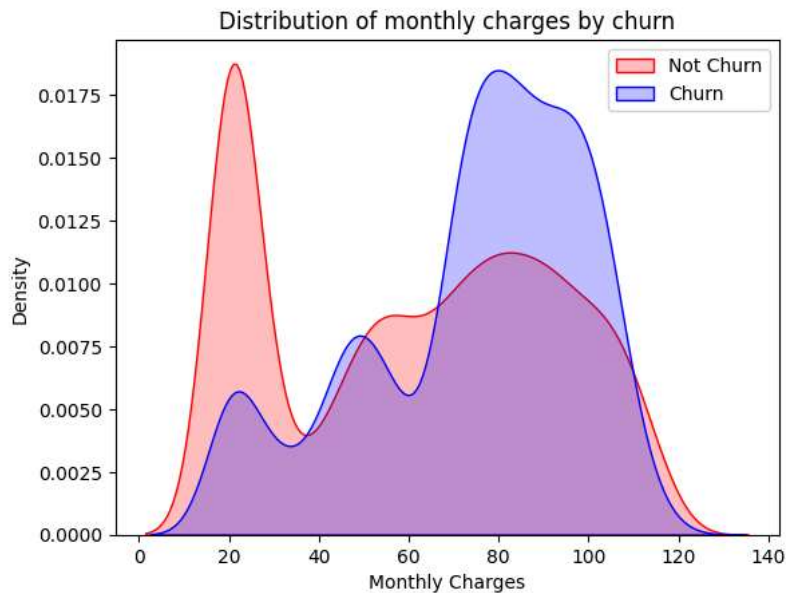
iv.) **Churn by Monthly Charges:** Higher % of customers churn when the monthly charges are high.

```
1 ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'No') ],
2                 color="Red", shade = True)
3 ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'Yes') ],
4                 ax =ax, color="Blue", shade= True)
5 ax.legend(["Not Churn", "Churn"],loc='upper right')
6 ax.set_ylabel('Density')
7 ax.set_xlabel('Monthly Charges')
8 ax.set_title('Distribution of monthly charges by churn')
```

```
<ipython-input-24-546dea8a96f3>:1: FutureWarning:
`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'No') ],
<ipython-input-24-546dea8a96f3>:3: FutureWarning:
`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'Yes') ],
Text(0.5, 1.0, 'Distribution of monthly charges by churn')
```



v.) **Churn by Total Charges:** It seems that there is higher churn when the total charges are lower.

```
1 ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust["Churn"] == 'No') ],
2     color="Red", shade = True)
3 ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust["Churn"] == 'Yes') ],
4     ax=ax, color="Blue", shade= True)
5 ax.legend(["Not Churn", "Churn"], loc='upper right')
6 ax.set_ylabel('Density')
7 ax.set_xlabel('Total Charges')
8 ax.set_title('Distribution of total charges by churn')
```

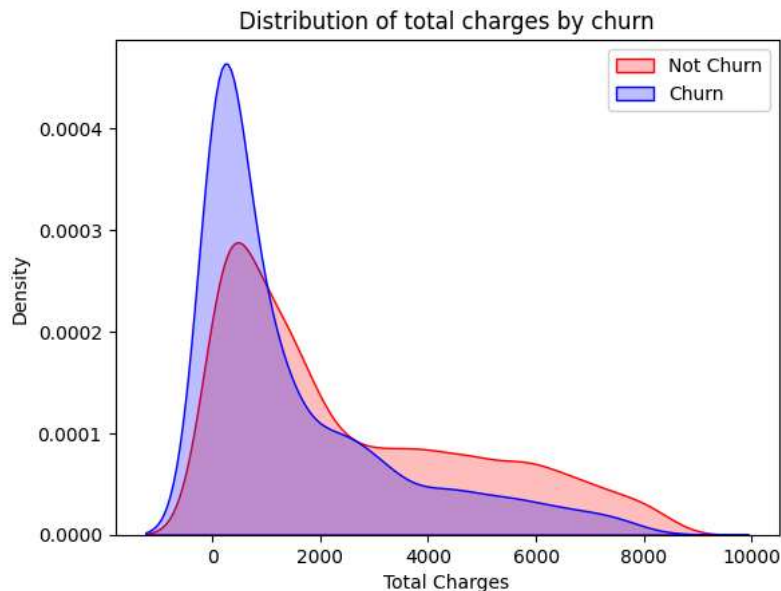
```
<ipython-input-25-5a731f014001>:1: FutureWarning:
```

```
`shade` is now deprecated in favor of `fill`; setting `fill=True`.  
This will become an error in seaborn v0.14.0; please update your code.
```

```
ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust["Churn"] == 'No') ],  
<ipython-input-25-5a731f014001>:3: FutureWarning:
```

```
`shade` is now deprecated in favor of `fill`; setting `fill=True`.  
This will become an error in seaborn v0.14.0; please update your code.
```

```
ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust["Churn"] == 'Yes') ],  
Text(0.5, 1.0, 'Distribution of total charges by churn')
```



✓ After going through the above EDA we will develop some predictive models and compare them.

We will develop Logistic Regression, Random Forest, SVM, ADA Boost and XG Boost

1. Logistic Regression

```
1 # We will use the data frame where we had created dummy variables  
2 y = df_dummies['Churn'].values  
3 X = df_dummies.drop(columns = ['Churn'])  
4  
5 # Scaling all the variables to a range of 0 to 1  
6 from sklearn.preprocessing import MinMaxScaler  
7 features = X.columns.values  
8 scaler = MinMaxScaler(feature_range = (0,1))  
9 scaler.fit(X)  
10 X = pd.DataFrame(scaler.transform(X))  
11 X.columns = features
```

It is important to scale the variables in logistic regression so that all of them are within a range of 0 to 1. This helped me improve the accuracy from 79.7% to 80.7%. Further, you will notice below that the importance of variables is also aligned with what we are seeing in Random Forest algorithm and the EDA we conducted above.

```
1 # Create Train & Test Data  
2 from sklearn.model_selection import train_test_split  
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
1 # Running logistic regression model  
2 from sklearn.linear_model import LogisticRegression  
3 model = LogisticRegression()  
4 result = model.fit(X_train, y_train)
```

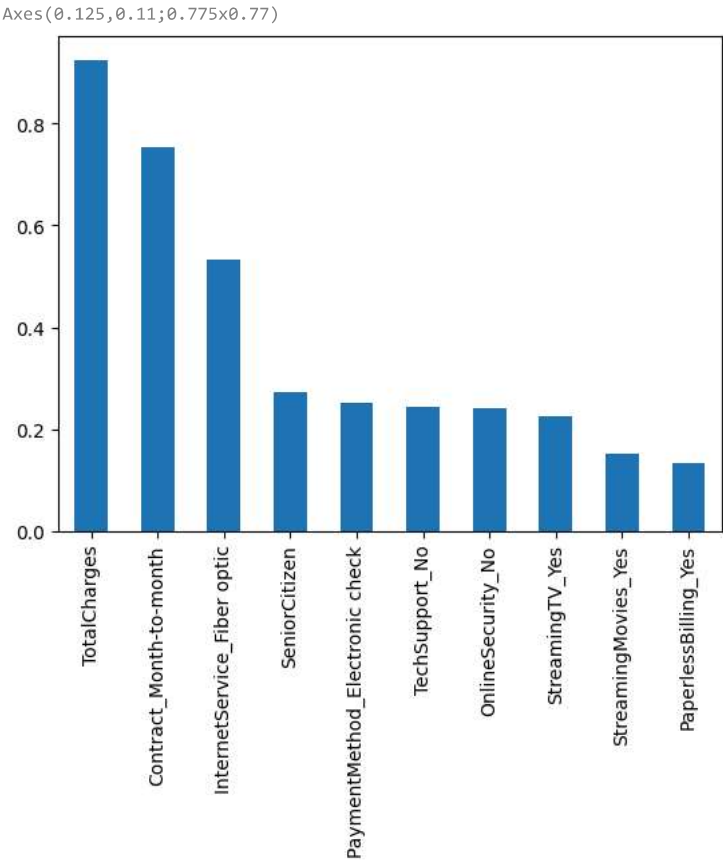
```
1 from sklearn import metrics  
2 prediction_test = model.predict(X_test)  
3 # Print the prediction accuracy  
4 print (metrics.accuracy_score(y_test, prediction_test))
```

0.8075829383886256


```

1 # To get the weights of all the variables
2 weights = pd.Series(model.coef_[0],
3                     index=X.columns.values)
4 print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
5

```



```

1 print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))

```

