Code clause

Data Science Intern

Vedant Sanjay Chaudhari

Project Name

Task 1 - Churn Prediction in Telecom Industry using Logistic Regression

In [1]:
```
# Import necessary libraries
!pip install seaborn

import seaborn as sns
```

In [1]:
```
# Import necessary libraries
!pip install seaborn

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_ma
```

```
Requirement already satisfied: seaborn in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (0.13.0)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (fr
om seaborn) (1.25.2)
Requirement already satisfied: pandas>=1.2 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from seabor
n) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.3 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages
(from seaborn) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from m
atplotlib!=3.6.1,>=3.3->seaborn) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from matpl
otlib!=3.6.1,>=3.3->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from
matplotlib!=3.6.1,>=3.3->seaborn) (4.42.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from
matplotlib!=3.6.1,>=3.3->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from ma
tplotlib!=3.6.1,>=3.3->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from matp
lotlib!=3.6.1,>=3.3->seaborn) (10.0.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (f
rom matplotlib!=3.6.1,>=3.3->seaborn) (2.4.7)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (fr
om matplotlib!=3.6.1,>=3.3->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from panda
s>=1.2->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\vedant\anaconda3\envs\tf\lib\site-packages (from python-da
teutil>=2.7->matplotlib!=3.6.1,>=3.3->seaborn) (1.16.0)
```

In [2]:
```
import pandas as pd

# Load your telecom dataset
data = pd.read_csv('churn prediction.csv')
```

In [2]:
```
import pandas as pd

# Load your telecom dataset
data = pd.read_csv('churn prediction.csv')
```

In [3]:
```
data.head(15)
```

Out[3]:

nerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection

In [3]:
```
data.head(15)
```

Out[3]:

merID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
7590-HVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No
5575-NVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes
3668-PYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No
7795-FOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No
9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	...	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	...	No
6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	...	No
7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	...	Yes
6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	...	No
9763-GRSKD	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	...	No
7469-LKBCI	Male	0	No	No	16	Yes	No	No	No internet service	...	No internet service
8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	...	Yes
0280-XJGEX	Male	0	No	No	49	Yes	Yes	Fiber optic	No	...	Yes
5129-JLPIS	Male	0	No	No	25	Yes	No	Fiber optic	Yes	...	Yes

21 columns

In [4]:
```
data.columns.values
```

Out[4]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
```

In [4]:
```
data.columns.values
```

Out[4]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [5]:
```
data.dtypes
```

Out[5]:
```
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
```

In [6]:
```
# Converting Total Charges to a numerical data type.

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

In [6]:
```
# Converting Total Charges to a numerical data type.

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

Out[6]:
```
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
```

In [7]:
```
#Removing missing values

data.dropna(inplace = True)
```

In [7]:
```
#Removing missing values

data.dropna(inplace = True)

#Remove customer IDs from the data set

df2 = data.iloc[:,1:]

#Converting the predictor variable in a binary numeric variable

df2['Churn'].replace(to_replace='Yes', value=1, inplace=True)
df2['Churn'].replace(to_replace='No',  value=0, inplace=True)

#Let's convert all the categorical variables into dummy variables

df_dummies = pd.get_dummies(df2)
df_dummies.head()
```

Out[7]:

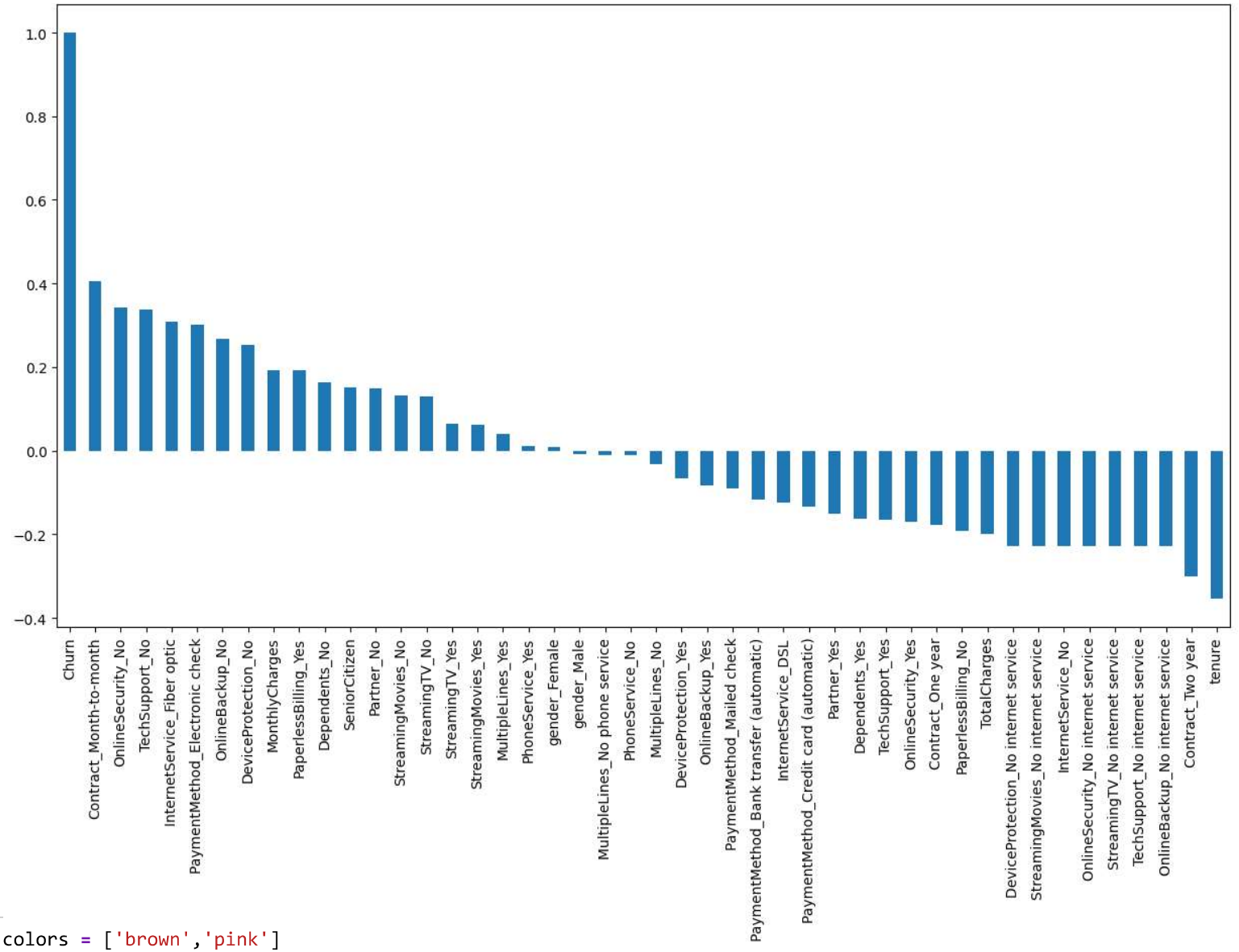
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	...
0	0	1	29.85	29.85	0	1	0	0	1	1	...
1	0	34	56.95	1889.50	0	0	1	1	0	1	...
2	0	2	53.85	108.15	1	0	1	1	0	1	...
3	0	45	42.30	1840.75	0	0	1	1	0	1	...
4	0	2	70.70	151.65	1	1	0	1	0	1	...

5 rows × 46 columns

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

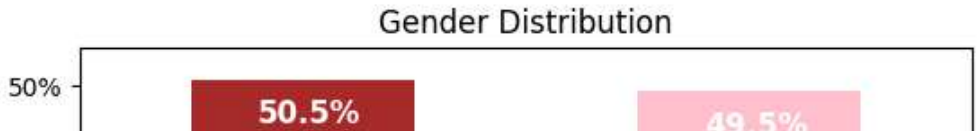
In [8]:
```
#Get Correlation of "Churn" with other variables:
plt.figure(figsize=(15,8))
df_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

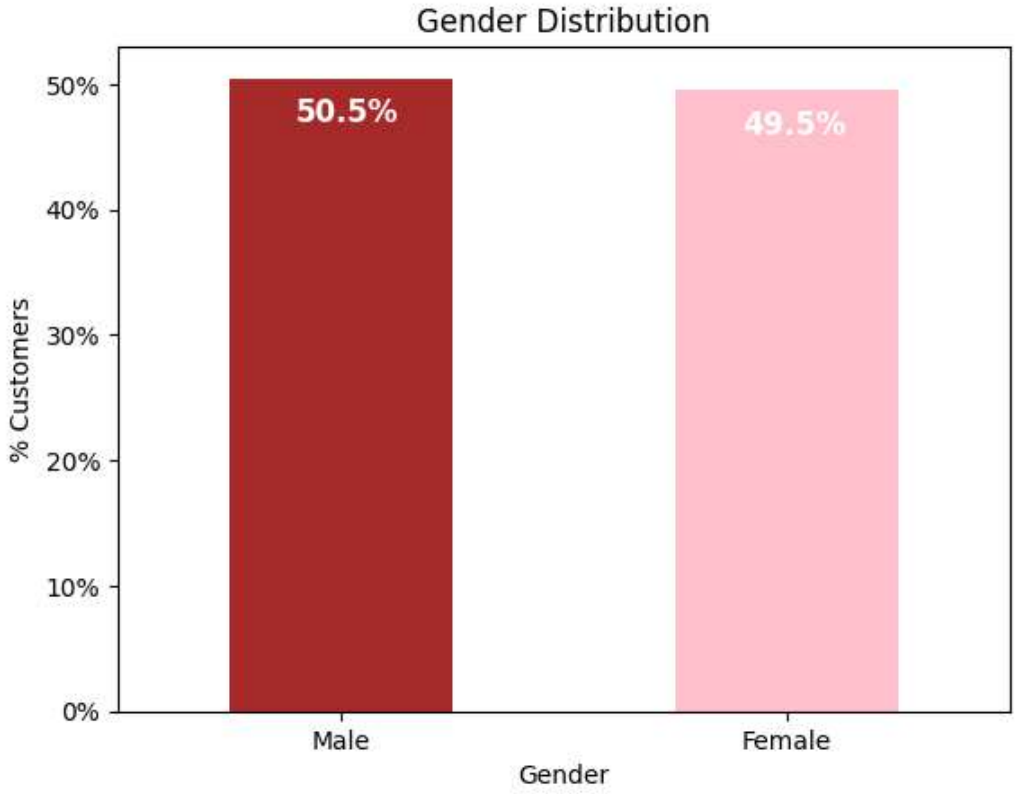
Out[8]: <Axes: >



In [9]:
```
colors = ['brown','pink']
ax = (data['gender'].value_counts()*100.0 /len(data)).plot(kind='bar', stacked = True, rot = 0, color = colors)
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers')
```

In [9]:
```
colors = ['brown','pink']
ax = (data['gender'].value_counts()*100.0 /len(data)).plot(kind='bar', stacked = True, rot = 0, color = colors)
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers')
ax.set_xlabel('Gender')
ax.set_ylabel('% Customers')
ax.set_title('Gender Distribution')

# create a list to collect the plt.patches data
totals = []

# find the values and append to list
for i in ax.patches:
    totals.append(i.get_width())

# set individual bar lables using above list
total = sum(totals)

for i in ax.patches:
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_x()+.15, i.get_height()-3.5, \
            str(round((i.get_height()/total), 1))+'%',
            fontsize=12,
            color='white',
           weight = 'bold')
```

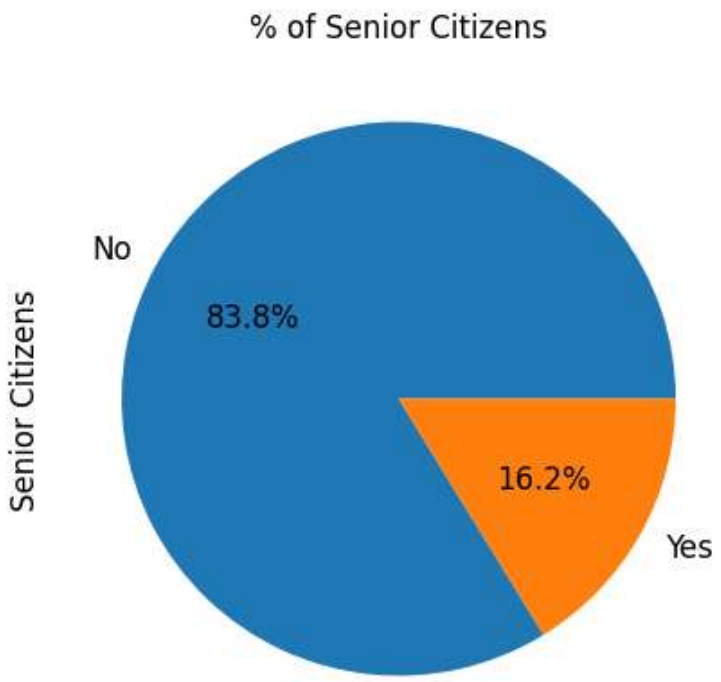




In [10]:
```
ax = (data['SeniorCitizen'].value_counts()*100.0 /len(data))\
.plot.pie(autopct='%.1f%%', labels = ['No', 'Yes'],figsize =(5,5), fontsize = 12 )
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('Senior Citizens',fontsize = 12)
```

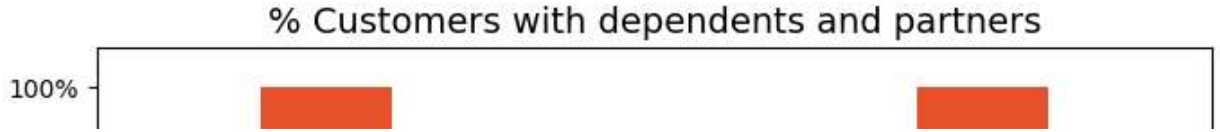
In [10]:
```
ax = (data['SeniorCitizen'].value_counts()*100.0 /len(data))\
.plot.pie(autopct='%.1f%%', labels = ['No', 'Yes'],figsize =(5,5), fontsize = 12 )
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('Senior Citizens',fontsize = 12)
ax.set_title('% of Senior Citizens', fontsize = 12)
```

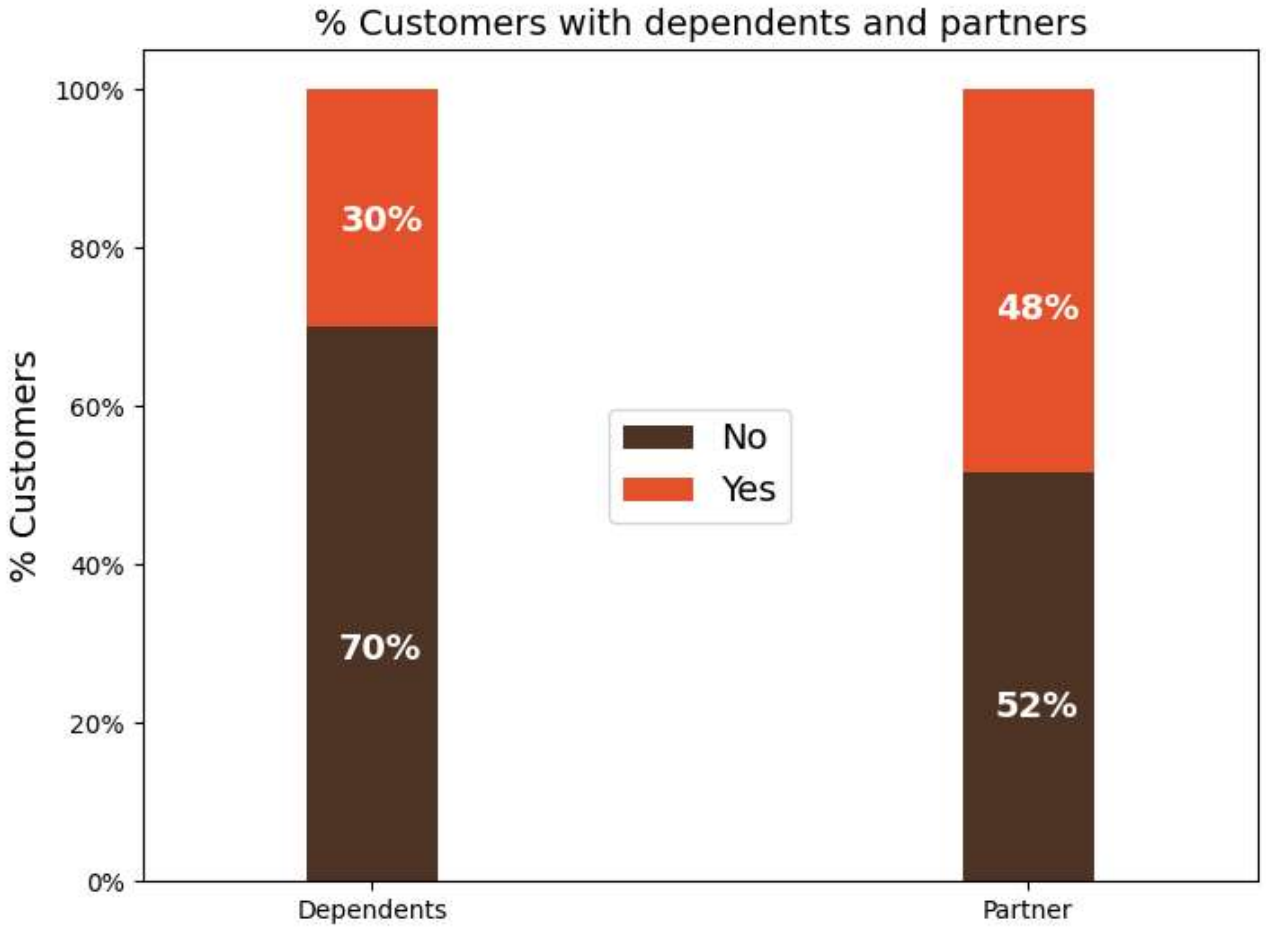
Out[10]: Text(0.5, 1.0, '% of Senior Citizens')



In [11]:
```
df2 = pd.melt(data, id_vars=['customerID'], value_vars=['Dependents','Partner'])
df3 = df2.groupby(['variable','value']).count().unstack()
df3 = df3*100/len(data)
colors = ['#4D3425','#E4512B']
```

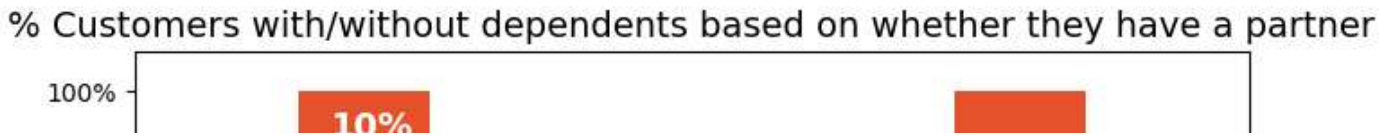
In [11]:
```
df2 = pd.melt(data, id_vars=['customerID'], value_vars=['Dependents','Partner'])
df3 = df2.groupby(['variable','value']).count().unstack()
df3 = df3*100/len(data)
colors = ['#4D3425','#E4512B']
ax = df3.loc[:,'customerID'].plot.bar(stacked=True, color=colors,
                                      figsize=(8,6),rot = 0,
                                     width = 0.2)

ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers',size = 14)
ax.set_xlabel('')
ax.set_title('% Customers with dependents and partners',size = 14)
ax.legend(loc = 'center',prop={'size':14})

for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),color = 'white',weight = 'b
```

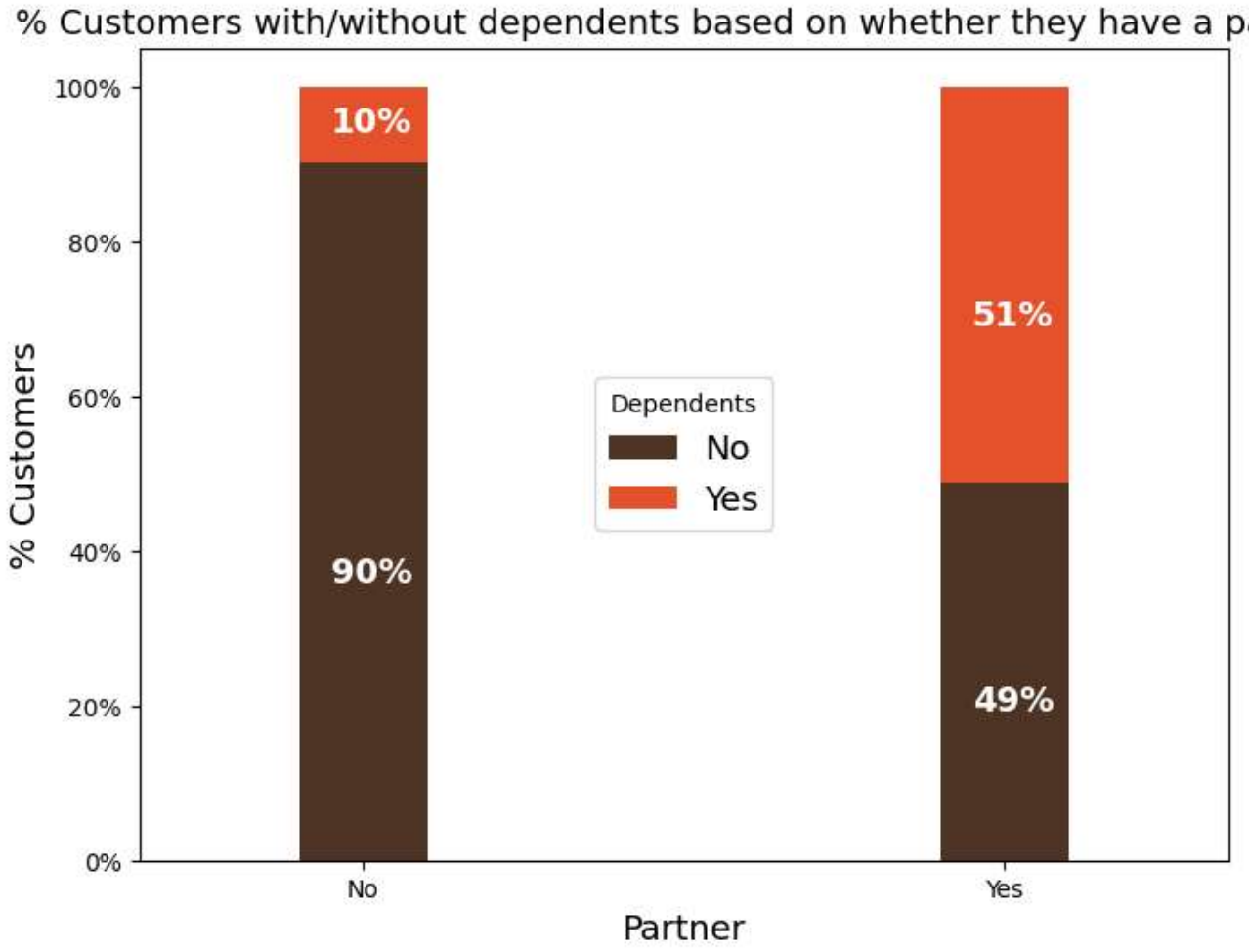




In [12]:
```
colors = ['#4D3425','#E4512B']
partner_dependents = data.groupby(['Partner','Dependents']).size().unstack()

ax = (partner_dependents.T*100.0 / partner_dependents.T.sum()).T.plot(kind='bar',
```

In [12]:
```
colors = ['#4D3425','#E4512B']
partner_dependents = data.groupby(['Partner','Dependents']).size().unstack()

ax = (partner_dependents.T*100.0 / partner_dependents.T.sum()).T.plot(kind='bar',
                                                                      width = 0.2,
                                                                      stacked = True,
                                                                      rot = 0,
                                                                      figsize = (8,6),
                                                                      color = colors)
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc='center',prop={'size':14},title = 'Dependents',fontsize =14)
ax.set_ylabel('% Customers',size = 14)
ax.set_title('% Customers with/without dependents based on whether they have a partner',size = 14)
ax.xaxis.label.set_size(14)

# Code to add the data labels on the stacked bar chart
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),
                color = 'white',
               weight = 'bold',
               size = 14)
```



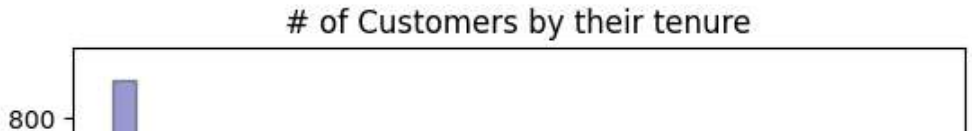


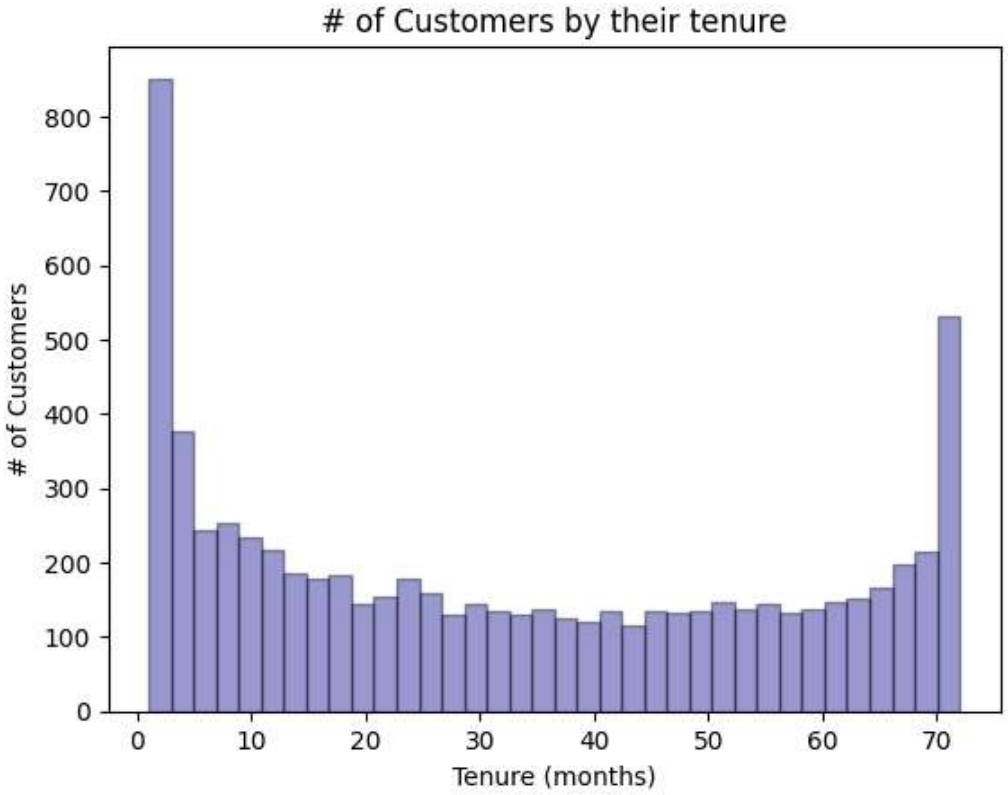
In [13]:
```
ax = sns.distplot(data['tenure'], hist=True, kde=False,
                  bins=int(180/5), color = 'darkblue',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 4})
```

In [13]:
```
ax = sns.distplot(data['tenure'], hist=True, kde=False,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('# of Customers by their tenure')
```

```
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\4060105518.py: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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  ax = sns.distplot(data['tenure'], hist=True, kde=False,
```

Out[13]:
```
Text(0.5, 1.0, '# of Customers by their tenure')
```

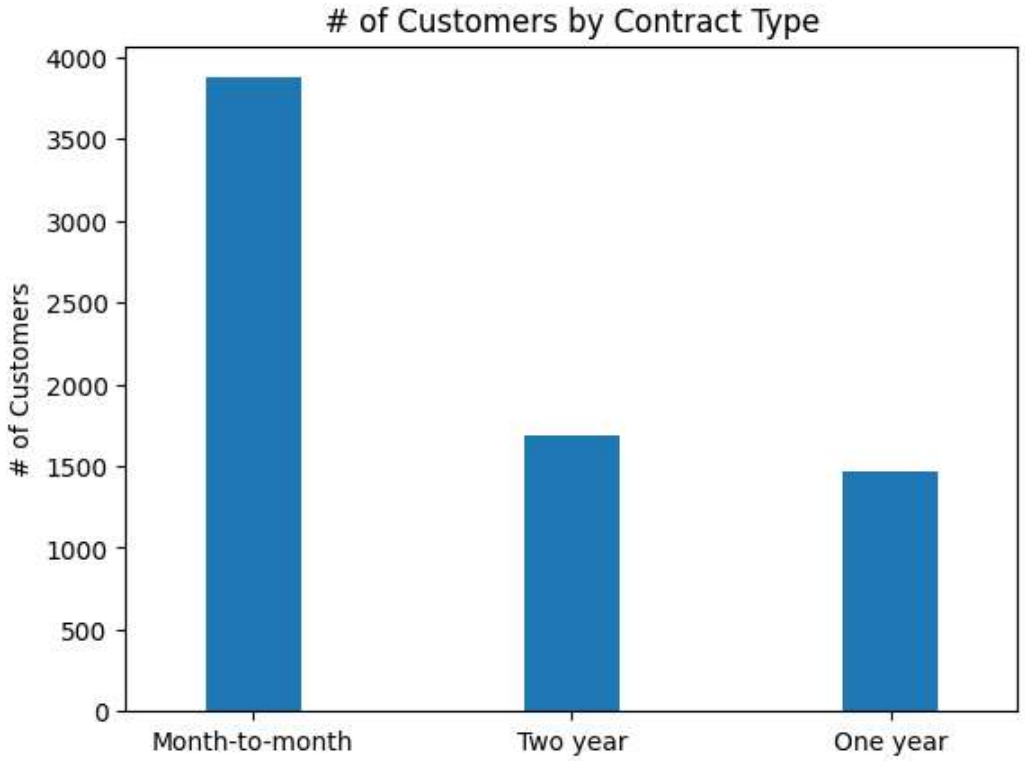




In [14]:
```
ax = data['Contract'].value_counts().plot(kind = 'bar',rot = 0, width = 0.3)
ax.set_ylabel('# of Customers')
ax.set_title('# of Customers by Contract Type')
```

In [14]:
```
ax = data['Contract'].value_counts().plot(kind = 'bar',rot = 0, width = 0.3)
ax.set_ylabel('# of Customers')
ax.set_title('# of Customers by Contract Type')
```

Out[14]:
```
Text(0.5, 1.0, '# of Customers by Contract Type')
```



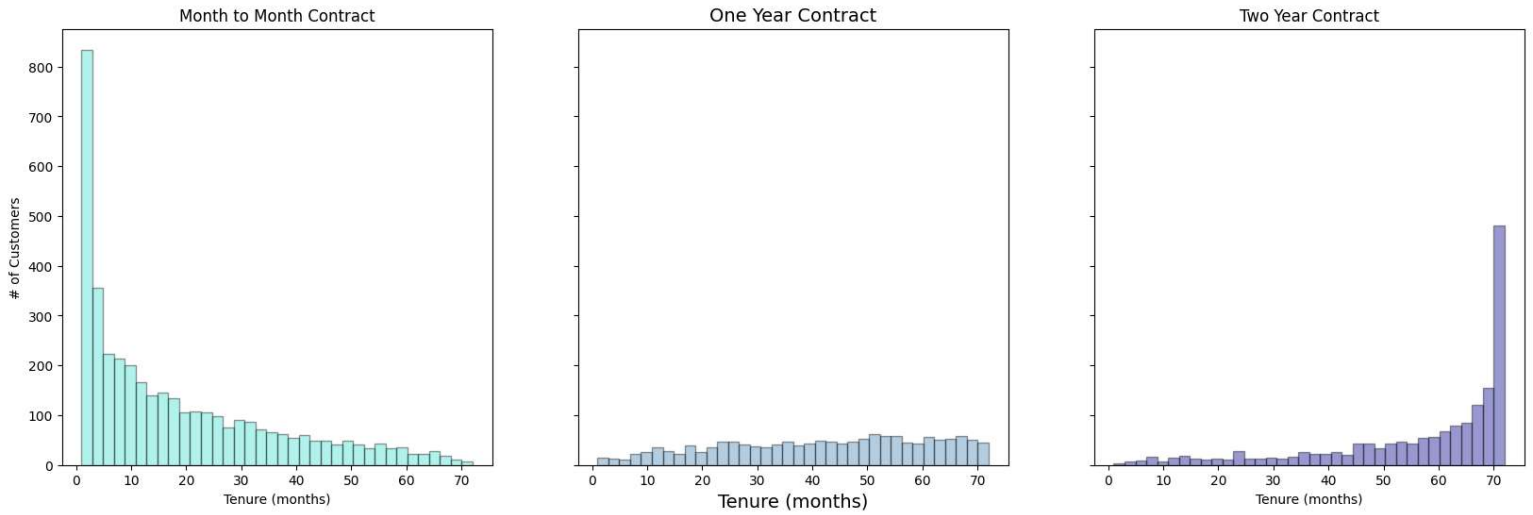
In [15]:
```
fig, (ax1,ax2,ax3) = plt.subplots(nrows=1, ncols=3, sharey = True, figsize = (20,6))

ax = sns.distplot(data[data['Contract']=='Month-to-month']['tenure'],
                  hist=True, kde=False,
```

In [15]:
```
fig, (ax1,ax2,ax3) = plt.subplots(nrows=1, ncols=3, sharey = True, figsize = (20,6))

ax = sns.distplot(data[data['Contract']=='Month-to-month']['tenure'],
                  hist=True, kde=False,
                  bins=int(180/5), color = 'turquoise',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 4},
                 ax=ax1)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('Month to Month Contract')

ax = sns.distplot(data[data['Contract']=='One year']['tenure'],
                  hist=True, kde=False,
                  bins=int(180/5), color = 'steelblue',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 4},
                 ax=ax2)
ax.set_xlabel('Tenure (months)',size = 14)
ax.set_title('One Year Contract',size = 14)

ax = sns.distplot(data[data['Contract']=='Two year']['tenure'],
                  hist=True, kde=False,
                  bins=int(180/5), color = 'darkblue',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 4},
                 ax=ax3)

ax.set_xlabel('Tenure (months)')
ax.set_title('Two Year Contract')
```

```
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\435576459.py: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
```

```
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\435576459.py: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 (https://gist.github.com/mwaskom/de44147ed297
4457ad6372750bbe5751)

  ax = sns.distplot(data[data['Contract']=='Month-to-month']['tenure'],
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\435576459.py: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 (https://gist.github.com/mwaskom/de44147ed297
4457ad6372750bbe5751)

  ax = sns.distplot(data[data['Contract']=='One year']['tenure'],
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\435576459.py: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 (https://gist.github.com/mwaskom/de44147ed297
4457ad6372750bbe5751)

  ax = sns.distplot(data[data['Contract']=='Two year']['tenure'],
```

Out[15]: Text(0.5, 1.0, 'Two Year Contract')

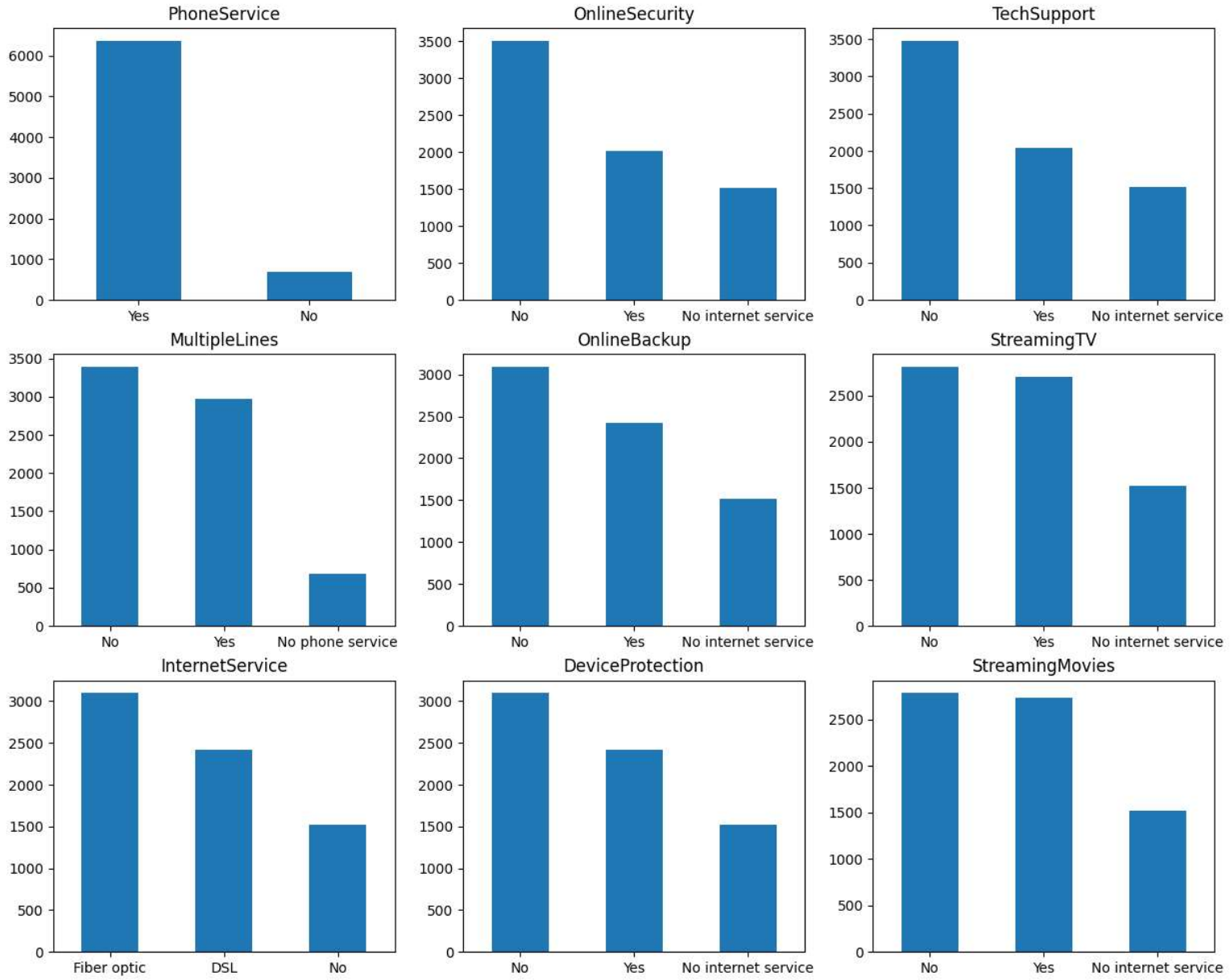


In [16]:
```
data.columns.values
```

Out[16]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [17]:
```
services = ['PhoneService','MultipleLines','InternetService','OnlineSecurity',
            'OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies']

fig, axes = plt.subplots(nrows = 3,ncols = 3,figsize = (15,12))
```

In [17]:
```
services = ['PhoneService','MultipleLines','InternetService','OnlineSecurity',
            'OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies']

fig, axes = plt.subplots(nrows = 3,ncols = 3,figsize = (15,12))
for i, item in enumerate(services):
    if i < 3:
        ax = data[item].value_counts().plot(kind = 'bar',ax=axes[i,0],rot = 0)

    elif i >=3 and i < 6:
        ax = data[item].value_counts().plot(kind = 'bar',ax=axes[i-3,1],rot = 0)

    elif i < 9:
        ax = data[item].value_counts().plot(kind = 'bar',ax=axes[i-6,2],rot = 0)
    ax.set_title(item)
```

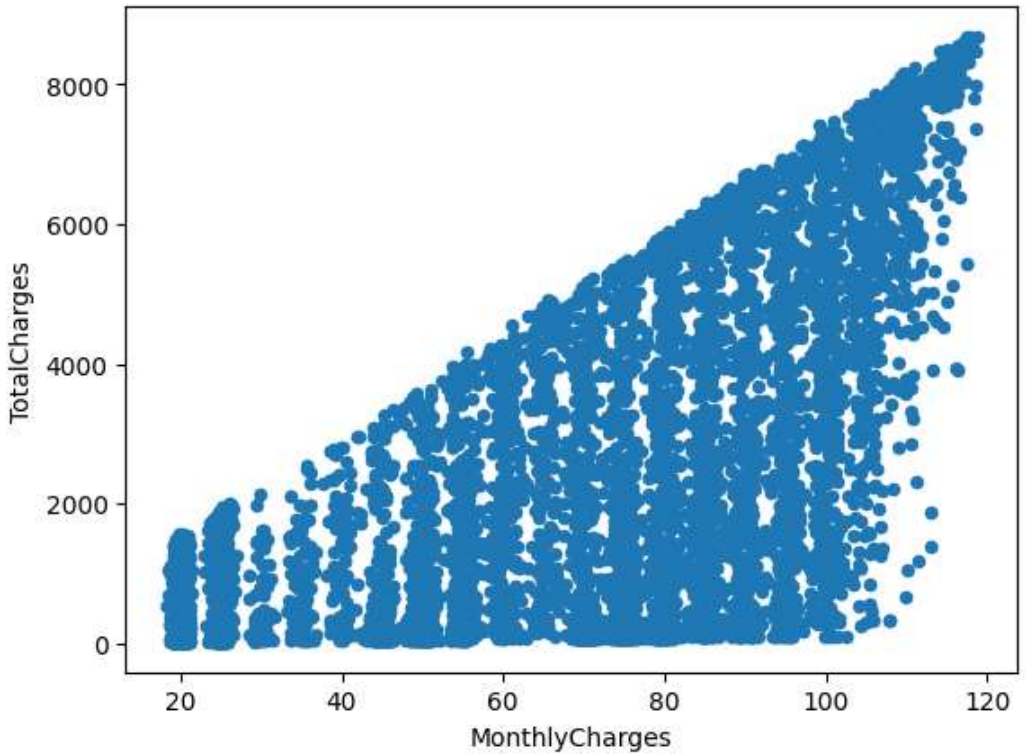


In [18]:
```
data[['MonthlyCharges', 'TotalCharges']].plot.scatter(x = 'MonthlyCharges', y='TotalCharges')
```

Out[18]: <Axes: xlabel='MonthlyCharges', ylabel='TotalCharges'>

In [18]:
```
data[['MonthlyCharges', 'TotalCharges']].plot.scatter(x = 'MonthlyCharges', y='TotalCharges')
```

Out[18]: <Axes: xlabel='MonthlyCharges', ylabel='TotalCharges'>



churn attribute

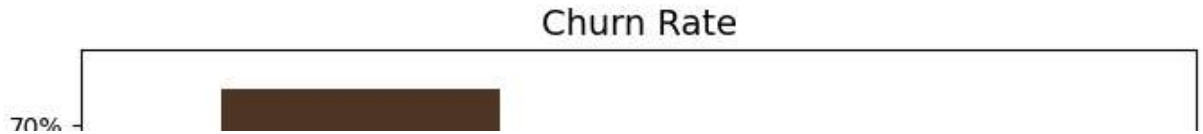
In [19]:
```
colors = ['#4D3425','#E4512B']
ax = (data['Churn'].value_counts()*100.0 /len(data)).plot(kind='bar', stacked = True, rot = 0, color = colors,fi
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers',size = 14)
```

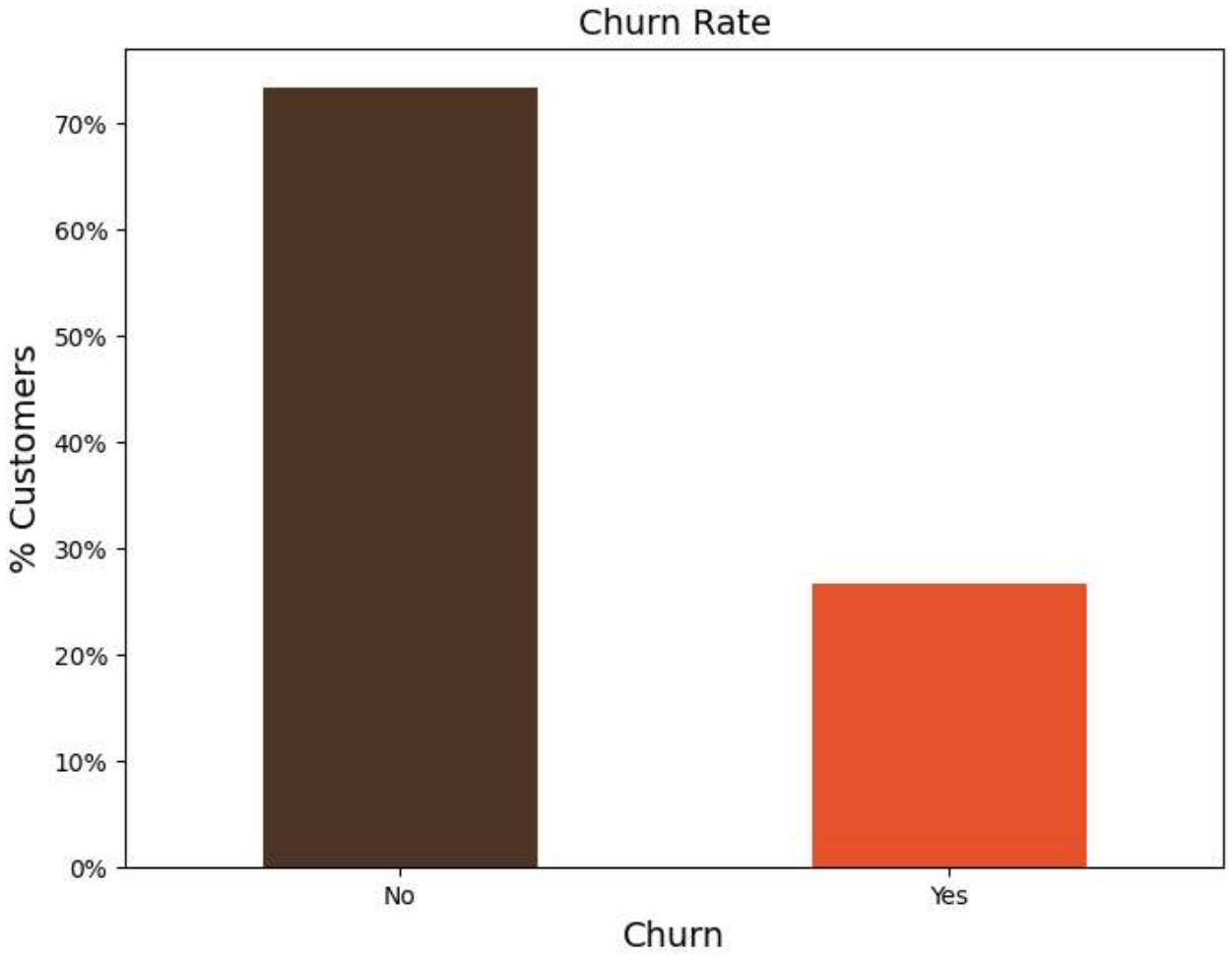
In [19]:
```
colors = ['#4D3425','#E4512B']
ax = (data['Churn'].value_counts()*100.0 /len(data)).plot(kind='bar', stacked = True, rot = 0, color = colors,fi
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers',size = 14)
ax.set_xlabel('Churn',size = 14)
ax.set_title('Churn Rate', size = 14)

# create a list to collect the plt.patches data
totals = []

# find the values and append to list
for i in ax.patches:
    totals.append(i.get_width())

# set individual bar lables using above list
total = sum(totals)

for i in ax.patches:
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_x()+.15, i.get_height()-4.0, \
            str(round((i.get_height()/total), 1))+'%',
            fontsize=12,
            color='white',
           weight = 'bold',
           size = 14)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 20
     16 total = sum(totals)
     18 for i in ax.patches:
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 20
     16 total = sum(totals)
     18 for i in ax.patches:
     19     # get_width pulls left or right; get_y pushes up or down
---> 20     ax.text(i.get_x()+.15, i.get_height()-4.0, \
     21             str(round((i.get_height()/total), 1))+'%',
     22             fontsize=12,
     23             color='white',
     24            weight = 'bold',
     25            size = 14)

File ~\anaconda3\envs\tf\lib\site-packages\matplotlib\axes\_axes.py:689, in Axes.text(self, x, y, s, fontdict,
**kwargs)
    628 """
    629 Add text to the Axes.
    630
   (...)
    679     >>> text(x, y, s, bbox=dict(facecolor='red', alpha=0.5))
    680 """
    681 effective_kwargs = {
    682     'verticalalignment': 'baseline',
    683     'horizontalalignment': 'left',
   (...)
    687     **kwargs,
    688 }
--> 689 t = mtext.Text(x, y, text=s, **effective_kwargs)
    690 t.set_clip_path(self.patch)
    691 self._add_text(t)

File ~\anaconda3\envs\tf\lib\site-packages\matplotlib\_api\deprecation.py:454, in make_keyword_only.<locals>.w
rapper(*args, **kwargs)
    448 if len(args) > name_idx:
    449     warn_deprecated(
    450         since, message="Passing the %(name)s %(obj_type)s "
    451         "positionally is deprecated since Matplotlib %(since)s; the "
    452         "parameter will become keyword-only %(removal)s.",
    453         name=name, obj_type=f"parameter of {func.__name__}()")
--> 454 return func(*args, **kwargs)

File ~\anaconda3\envs\tf\lib\site-packages\matplotlib\text.py:183, in Text.__init__(self, x, y, text, color, v
erticalalignment, horizontalalignment, multialignment, fontproperties, rotation, linespacing, rotation_mode, u
nset, wrap, transform_rotates_text, parse_math, **kwargs)
    167 self._text = ''
    168 self._reset_visual_defaults(
    169     text=text,
    170     color=color,
(...)
    181     rotation_mode=rotation_mode,
    182 )
--> 183 self.update(kwargs)
```

```
setex, wrap, transform_rotates_text, parse_math, **kwargs)
    167 self.text = ''
    168 self._reset_visual_defaults(
    169     text=text,
    170     color=color,
    171     fontproperties=fontproperties,
    172     clip_on=clip_on,
    173     usetex=usetex,
    174     parse_math=parse_math,
    175     wrap=wrap,
    176     transform_rotates_text=transform_rotates_text,
    177     rotation_mode=rotation_mode,
    178 )
--> 179 self.update(kwargs)

File ~\anaconda3\envs\tf\lib\site-packages\matplotlib\text.py:223, in Text.update(self, kwargs)
    221 def update(self, kwargs):
    222     # docstring inherited
--> 223     kwargs = cbook.normalize_kwargs(kwargs, Text)
    224     sentinel = object()  # bbox can be None, so use another sentinel.
    225     # Update fontproperties first, as it has lowest priority.

File ~\anaconda3\envs\tf\lib\site-packages\matplotlib\cbook\__init__.py:1779, in normalize_kwargs(kw, alias_ma
pping)
   1777 canonical = to_canonical.get(k, k)
   1778 if canonical in canonical_to_seen:
-> 1779     raise TypeError(f"Got both {canonical_to_seen[canonical]!r} and "
   1780                     f"{k!r}, which are aliases of one another")
   1781 canonical_to_seen[canonical] = k
   1782 ret[canonical] = v

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



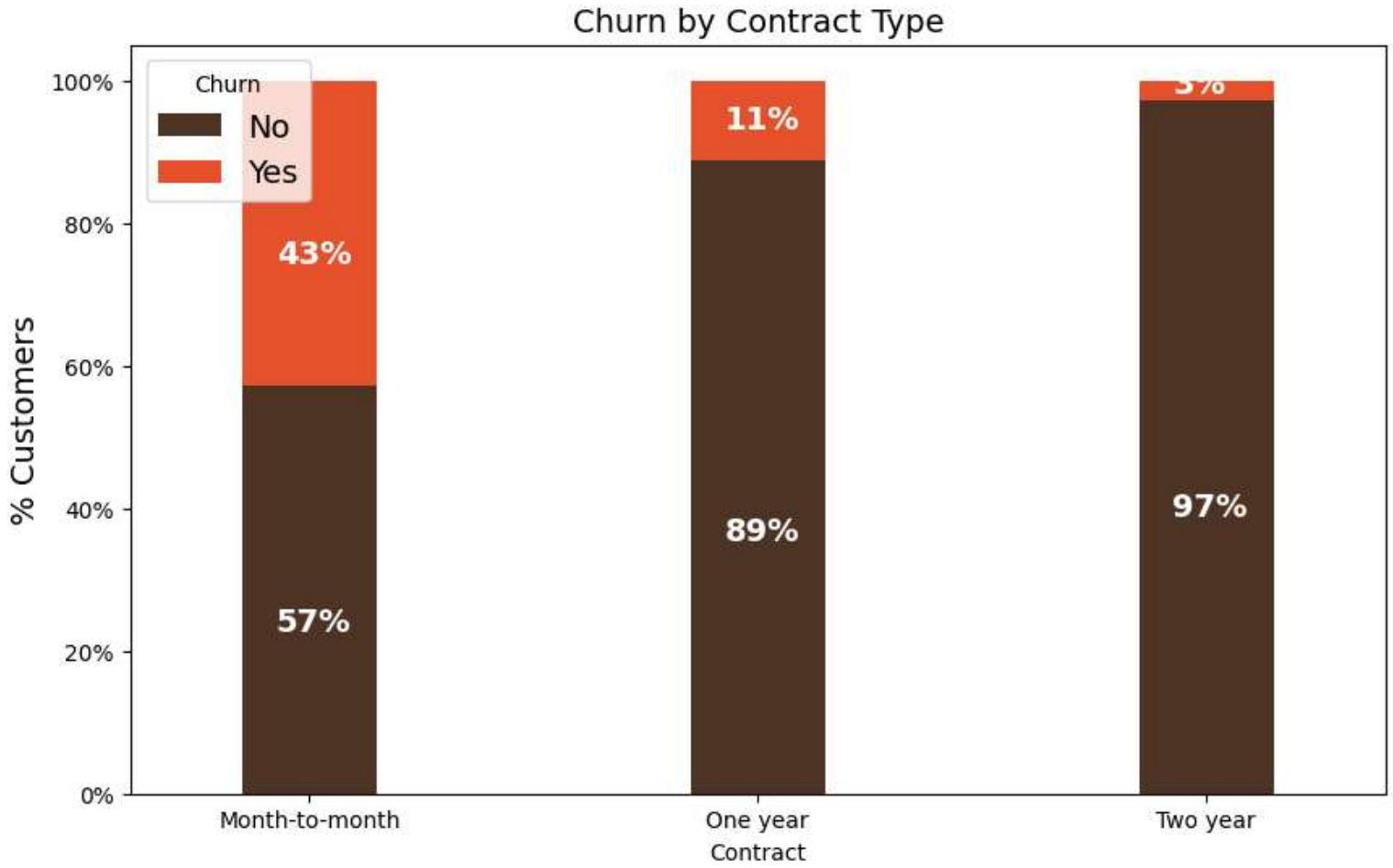


churn by contract

In [20]:
```
colors = ['#4D3425','#E4512B']
contract_churn = data.groupby(['Contract','Churn']).size().unstack()

ax = (contract_churn.T*100.0 / contract_churn.T.sum()).T.plot(kind='bar',width = 0.3,stacked = True, rot = 0,fig
```

In [20]:

```
colors = ['#4D3425','#E4512B']
contract_churn = data.groupby(['Contract','Churn']).size().unstack()

ax = (contract_churn.T*100.0 / contract_churn.T.sum()).T.plot(kind='bar',width = 0.3,stacked = True, rot = 0,fig
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc='best',prop={'size':14},title = 'Churn')
ax.set_ylabel('% Customers',size = 14)
ax.set_title('Churn by Contract Type',size = 14)

# Code to add the data labels on the stacked bar chart
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height), color = 'white', weight =
```



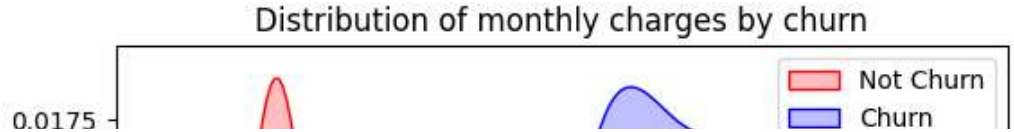
churn by monthly charges

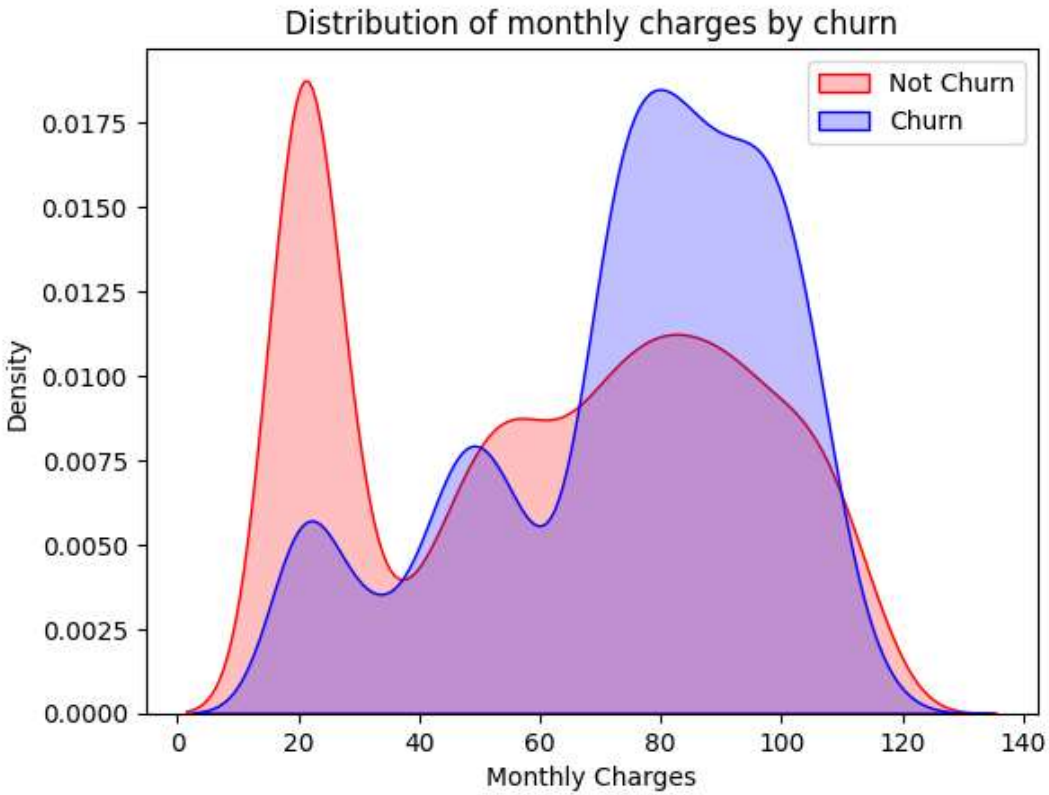
In [21]:
```
ax = sns.kdeplot(data.MonthlyCharges[(data["Churn"] == 'No') ], color="Red", shade = True)
ax = sns.kdeplot(data.MonthlyCharges[(data["Churn"] == 'Yes') ],ax =ax, color="Blue", shade= True)
ax.legend(["Not Churn","Churn"],loc='upper right')
ax.set_ylabel('Density')
```

In [21]:
```
ax = sns.kdeplot(data.MonthlyCharges[(data["Churn"] == 'No') ], color="Red", shade = True)
ax = sns.kdeplot(data.MonthlyCharges[(data["Churn"] == 'Yes') ],ax =ax, color="Blue", shade= True)
ax.legend(["Not Churn","Churn"],loc='upper right')
ax.set_ylabel('Density')
ax.set_xlabel('Monthly Charges')
ax.set_title('Distribution of monthly charges by churn')
```

```
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\2217197768.py: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(data.MonthlyCharges[(data["Churn"] == 'No') ], color="Red", shade = True)
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\2217197768.py:2: 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(data.MonthlyCharges[(data["Churn"] == 'Yes') ],ax =ax, color="Blue", shade= True)
```

Out[21]:
```
Text(0.5, 1.0, 'Distribution of monthly charges by churn')
```





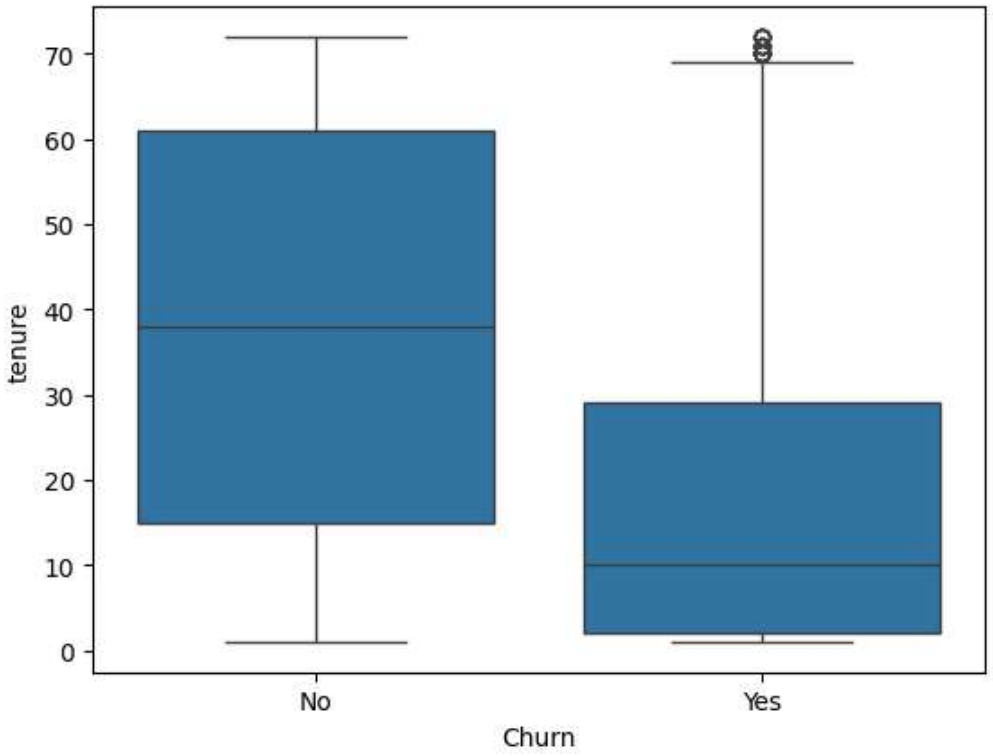
churn by tenure

In [22]:
```
sns.boxplot(x = data.Churn, y = data.tenure)
```

Out[22]:
```
<Axes: xlabel='Churn', ylabel='tenure'>
```

In [22]:
```
sns.boxplot(x = data.Churn, y = data.tenure)
```

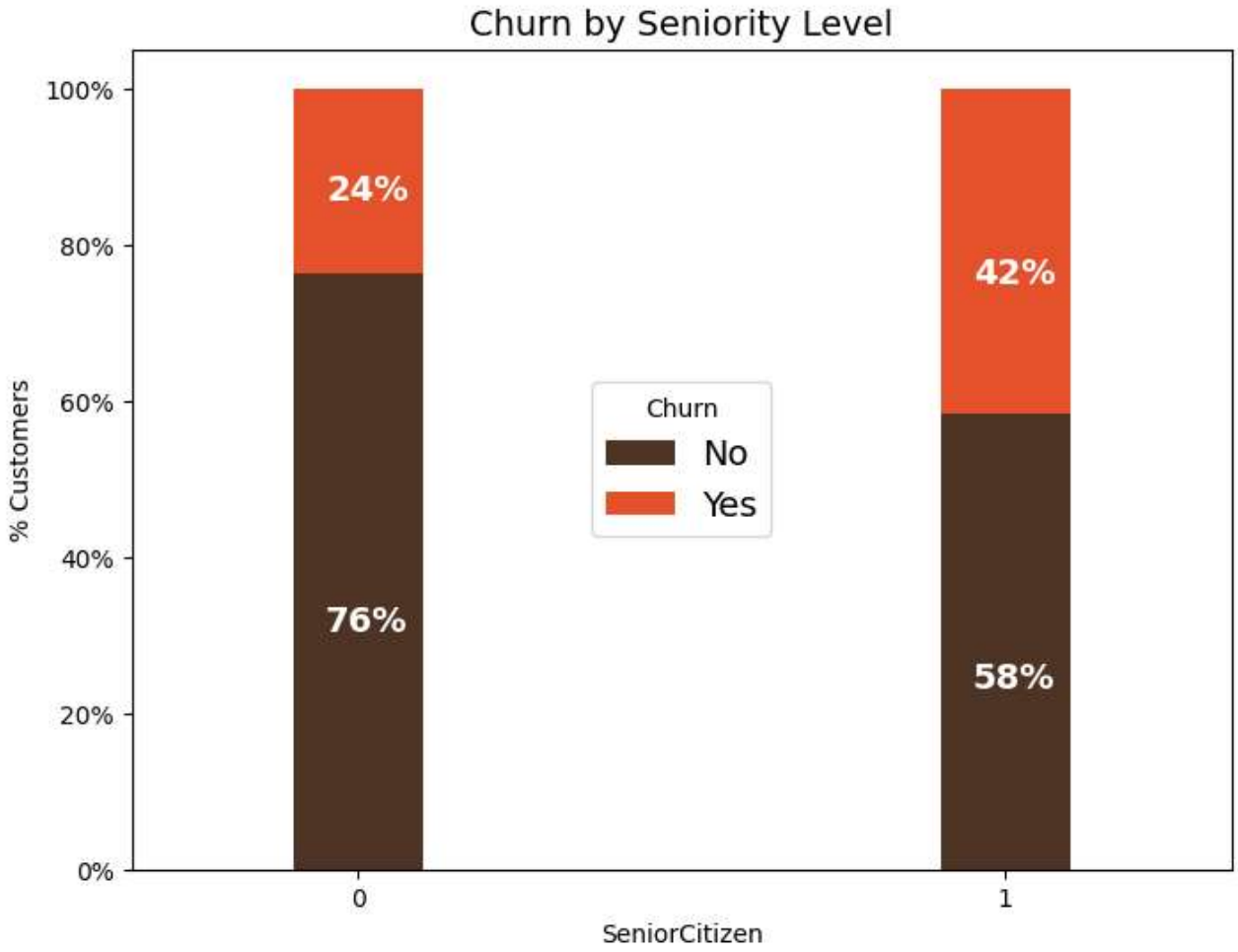
Out[22]:
```
<Axes: xlabel='Churn', ylabel='tenure'>
```



churn by seniority

In [23]:
```
lors = ['#4D3425','#E4512B']
niority_churn = data.groupby(['SeniorCitizen','Churn']).size().unstack()

 = (seniority_churn.T*100.0 / seniority_churn.T.sum()).T.plot(kind='bar', width = 0.2,stacked = True,rot = 0,figs
```

In [23]:
```
lors = ['#4D3425','#E4512B']
niority_churn = data.groupby(['SeniorCitizen','Churn']).size().unstack()

 = (seniority_churn.T*100.0 / seniority_churn.T.sum()).T.plot(kind='bar', width = 0.2,stacked = True,rot = 0,figs
.yaxis.set_major_formatter(mtick.PercentFormatter())
.legend(loc='center',prop={'size':14},title = 'Churn')
.set_ylabel('% Customers')
.set_title('Churn by Seniority Level',size = 14)

Code to add the data labels on the stacked bar chart
r p in ax.patches:
  width, height = p.get_width(), p.get_height()
  x, y = p.get_xy()
  ax.annotate('{:.0f}%'.format(height), (p.get_x()+.25*width, p.get_y()+.4*height),color = 'white',weight = 'bold
```



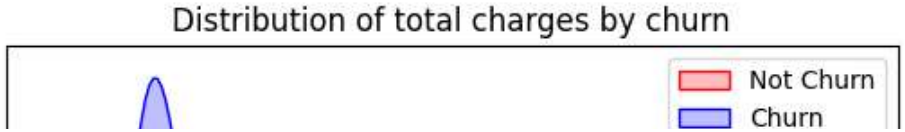
churn by total charges

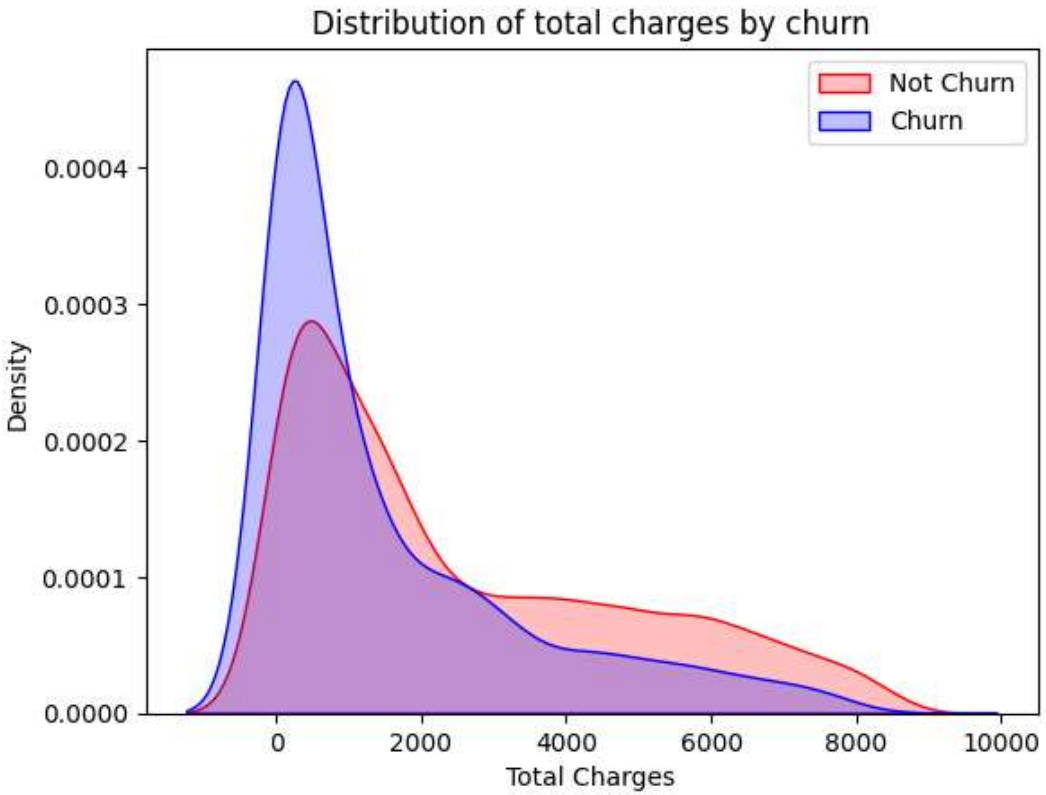
In [24]:
```
ax = sns.kdeplot(data.TotalCharges[(data["Churn"] == 'No') ],color="Red", shade = True)
ax = sns.kdeplot(data.TotalCharges[(data["Churn"] == 'Yes') ],ax =ax, color="Blue", shade= True)
ax.legend(["Not Churn","Churn"],loc='upper right')
ax.set_ylabel('Density')
```

In [24]:
```
ax = sns.kdeplot(data.TotalCharges[(data["Churn"] == 'No') ],color="Red", shade = True)
ax = sns.kdeplot(data.TotalCharges[(data["Churn"] == 'Yes') ],ax =ax, color="Blue", shade= True)
ax.legend(["Not Churn","Churn"],loc='upper right')
ax.set_ylabel('Density')
ax.set_xlabel('Total Charges')
ax.set_title('Distribution of total charges by churn')
```

```
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\446959753.py: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(data.TotalCharges[(data["Churn"] == 'No') ],color="Red", shade = True)
C:\Users\Vedant\AppData\Local\Temp\ipykernel_15140\446959753.py:2: 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(data.TotalCharges[(data["Churn"] == 'Yes') ],ax =ax, color="Blue", shade= True)
```

Out[24]:
```
Text(0.5, 1.0, 'Distribution of total charges by churn')
```





finally by Logistic regression

In [25]:
```
y = df_dummies['Churn'].values
X = df_dummies.drop(columns = ['Churn'])

# Scaling all the variables to a range of 0 to 1
from sklearn.preprocessing import MinMaxScaler
features = X.columns.values
scaler = MinMaxScaler(feature_range = (0,1))
scaler.fit(X)
X = pd.DataFrame(scaler.transform(X))
X.columns = features
```

In [26]:
```
# Create Train & Test Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

In [26]:
```
# Create Train & Test Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

In [27]:
```
# Running logistic regression model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result = model.fit(X_train, y_train)
```

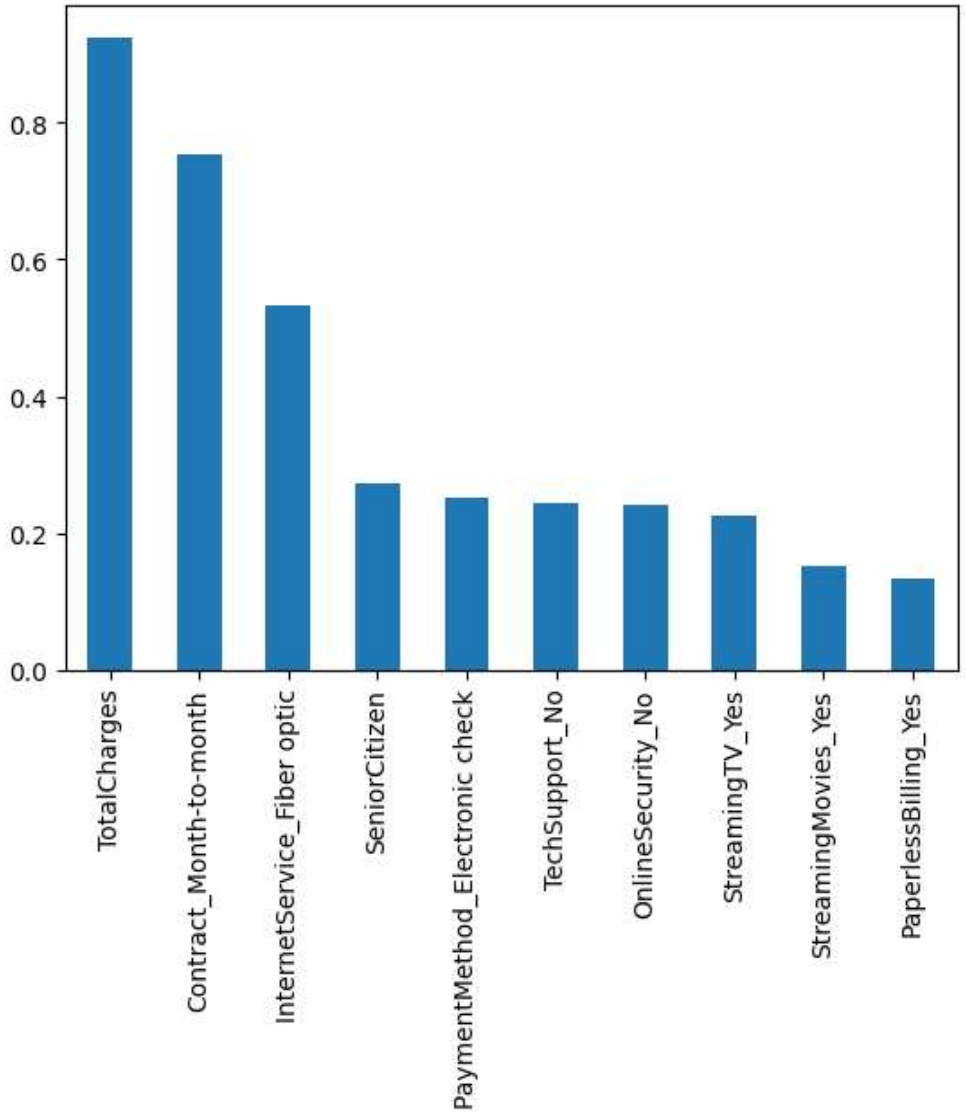
In [28]:
```
from sklearn import metrics
prediction_test = model.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8075829383886256
```

In [29]:
```
# To get the weights of all the variables
weights = pd.Series(model.coef_[0],
                 index=X.columns.values)
print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
```

In [29]:
```
# To get the weights of all the variables
weights = pd.Series(model.coef_[0],
                    index=X.columns.values)
print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
```

```
Axes(0.125,0.11;0.775x0.77)
```

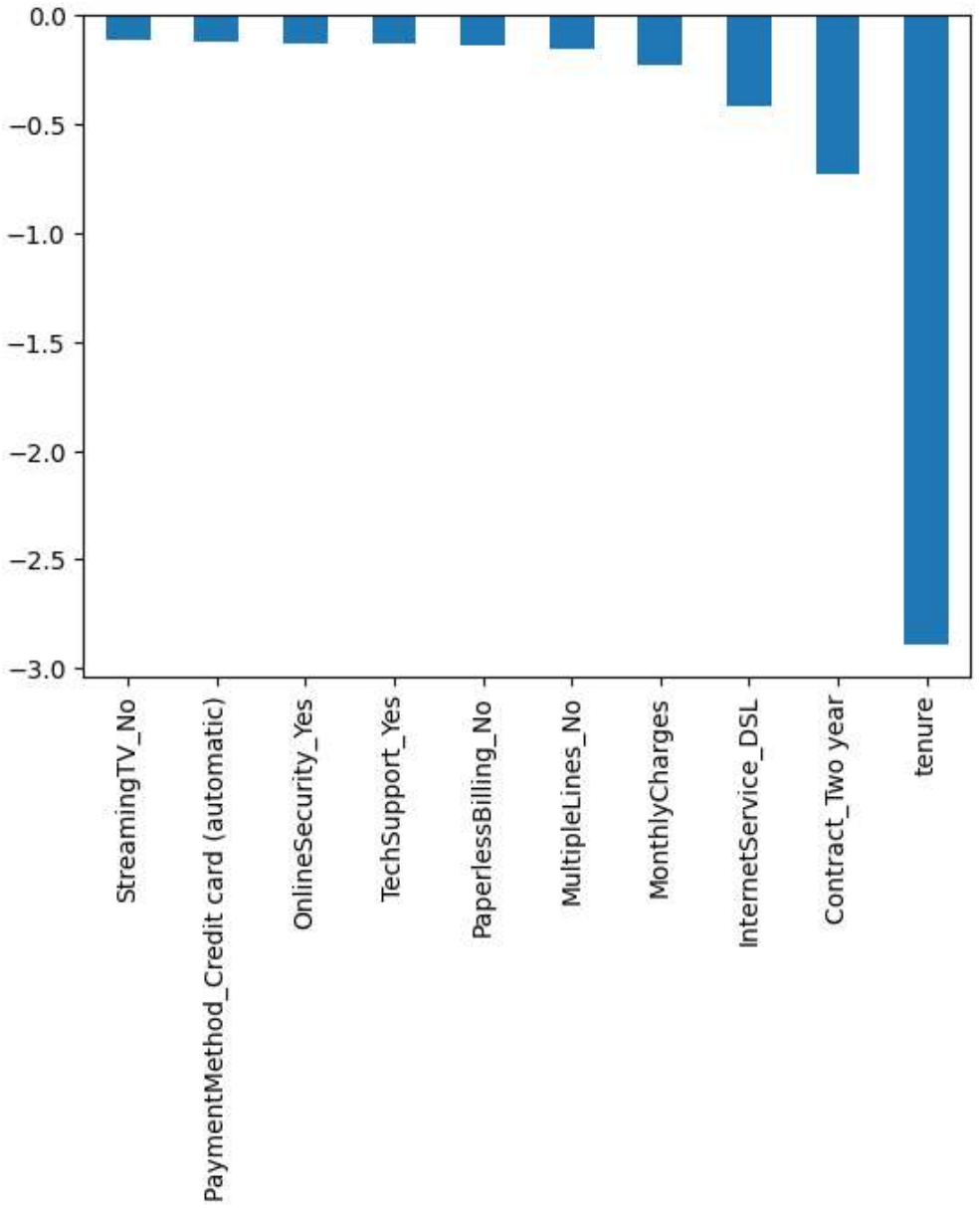


In [30]:
```
print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))
```

```
Axes(0.125,0.11;0.775x0.77)
```

In [30]:
```
print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))
```

```
Axes(0.125,0.11;0.775x0.77)
```



In [ ]:

In [ ]: