

CodeClause

Data Science Intern

Author : Gajjala Devendra Reddy

Project Name:

Task-1 :Churn Prediction in Telecom Industry using Logistic Regression

In []:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
```

In [3]:

```
telecom_cust = pd.read_csv("C://Users//gajja//Downloads//Tel.csv")
```

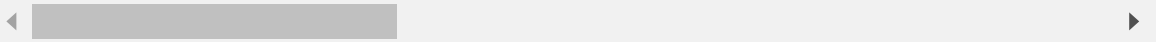
In [4]:

```
telecom_cust.head(10)
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
0	7590-VHVEG	Female	0	Yes	No	1	No	No ph serv
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No ph serv
4	9237-HQITU	Female	0	No	No	2	Yes	
5	9305-CDSKC	Female	0	No	No	8	Yes	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	
7	6713-OKOMC	Female	0	No	No	10	No	No ph serv
8	7892-POOKP	Female	0	Yes	No	28	Yes	
9	6388-TABGU	Male	0	No	Yes	62	Yes	

10 rows × 21 columns



In [5]:

```
telecom_cust.columns.values
```

Out[5]:

```
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 [6]:

```
telecom_cust.dtypes
```

Out[6]:

```
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 [7]:

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

Out[7]:

```
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 [8]:

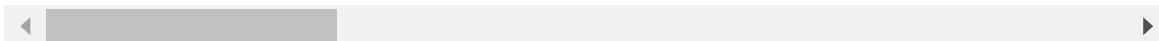
```
#Removing missing values
telecom_cust.dropna(inplace = True)
#Remove customer IDs from the data set
df2 = telecom_cust.iloc[:,1:]
#Convertin the predictor variable in a binary numeric variable
df2['Churn'].replace(to_replace='Yes', value=1, inplace=True)
df2['Churn'].replace(to_replace='No', value=0, inplace=True)

#Let's convert all the categorical variables into dummy variables
df_dummies = pd.get_dummies(df2)
df_dummies.head()
```

Out[8]:

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

5 rows × 46 columns

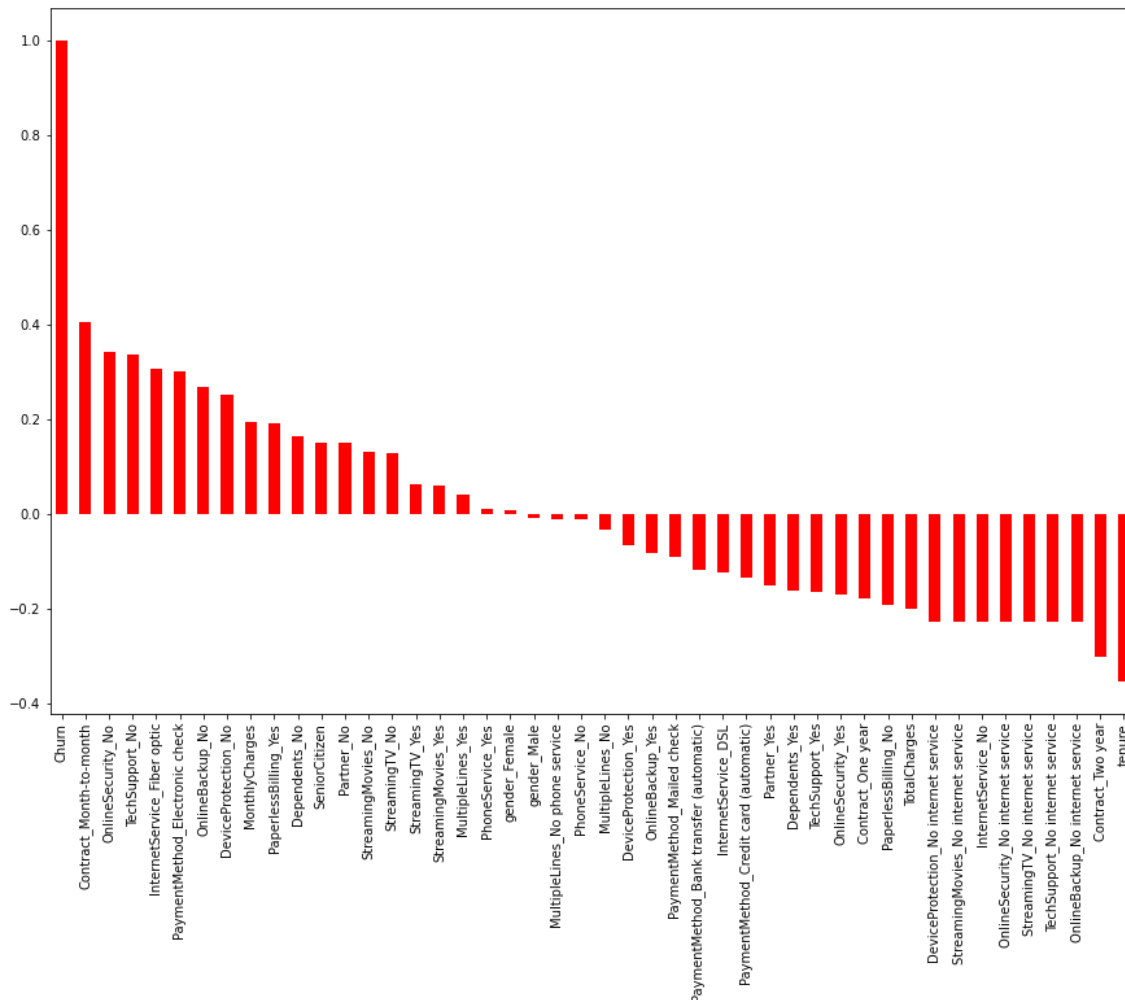


In [37]:

```
#Get Correlation of "Churn" with other variables:
plt.figure(figsize=(15,10),)
m=df_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar',color="red")
m.plot()
```

Out[37]:

[]



In [29]:

```

colors = ['yellow', 'blue']
ax = (telecom_cust['gender'].value_counts()*100.0 /len(telecom_cust)).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')

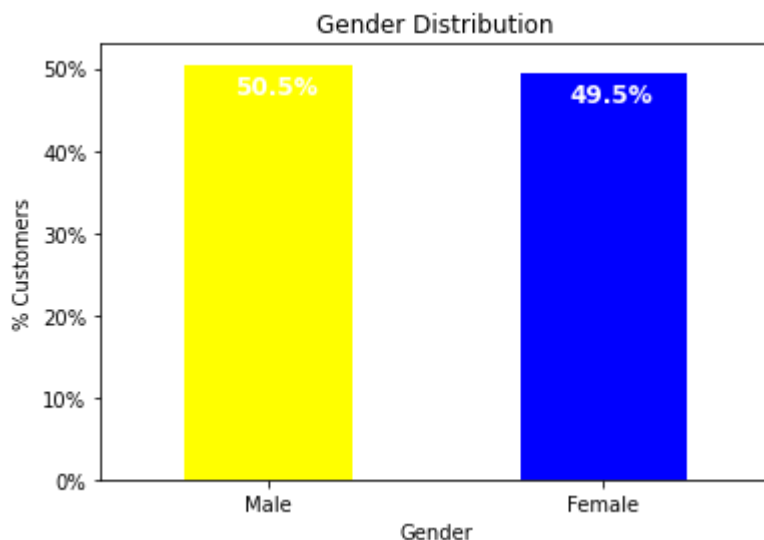
# create a list to collect the plt.patches data
totals = []

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

# set individual bar labels 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')

```

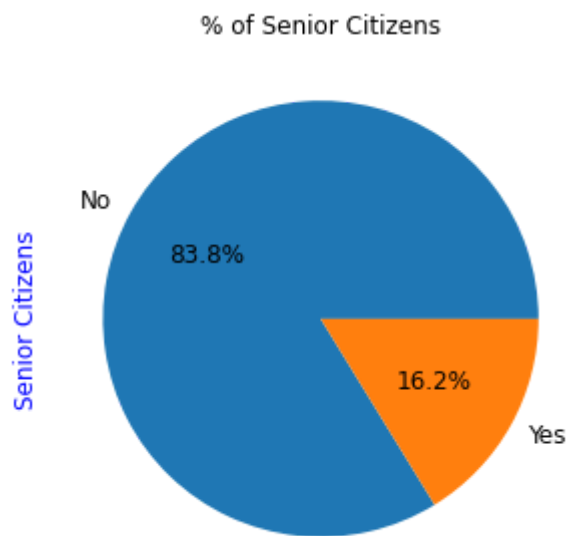


In [45]:

```
ax = (telecom_cust['SeniorCitizen'].value_counts()*100.0 / len(telecom_cust))\
.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,c="b")
ax.set_title('% of Senior Citizens', fontsize = 12)
```

Out[45]:

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

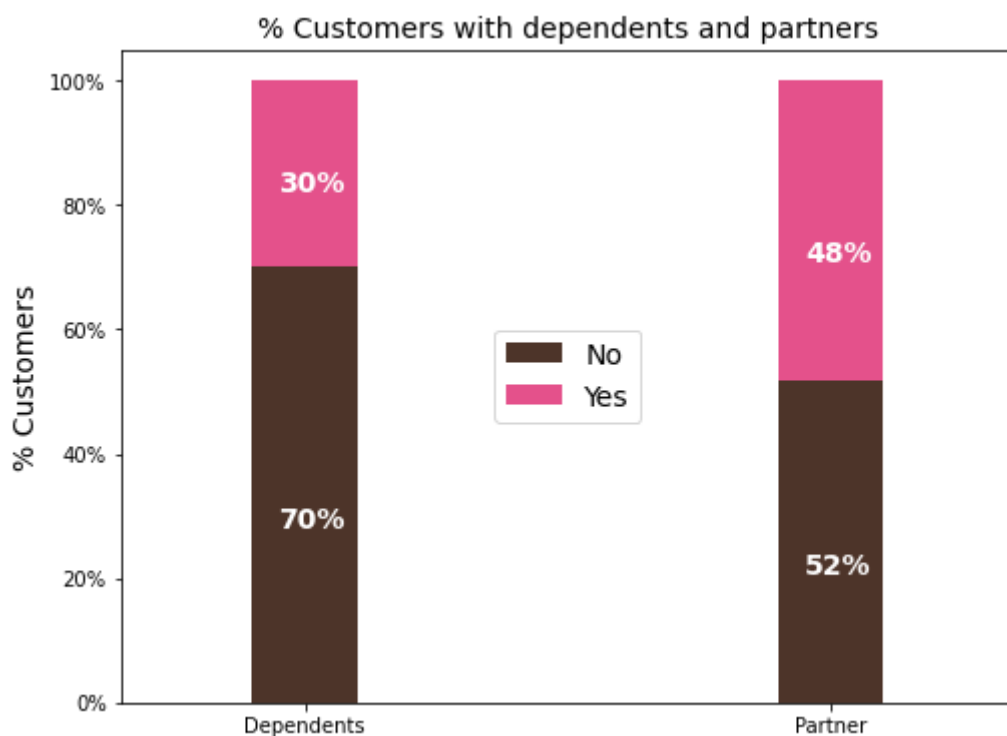


In [48]:

```
df2 = pd.melt(telecom_cust, id_vars=['customerID'], value_vars=['Dependents', 'Partner'])
df3 = df2.groupby(['variable', 'value']).count().unstack()
df3 = df3*100/len(telecom_cust)
colors = ['#4D3429', '#E4518B']
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 = 'bold',
               size = 14)
```



In [49]:

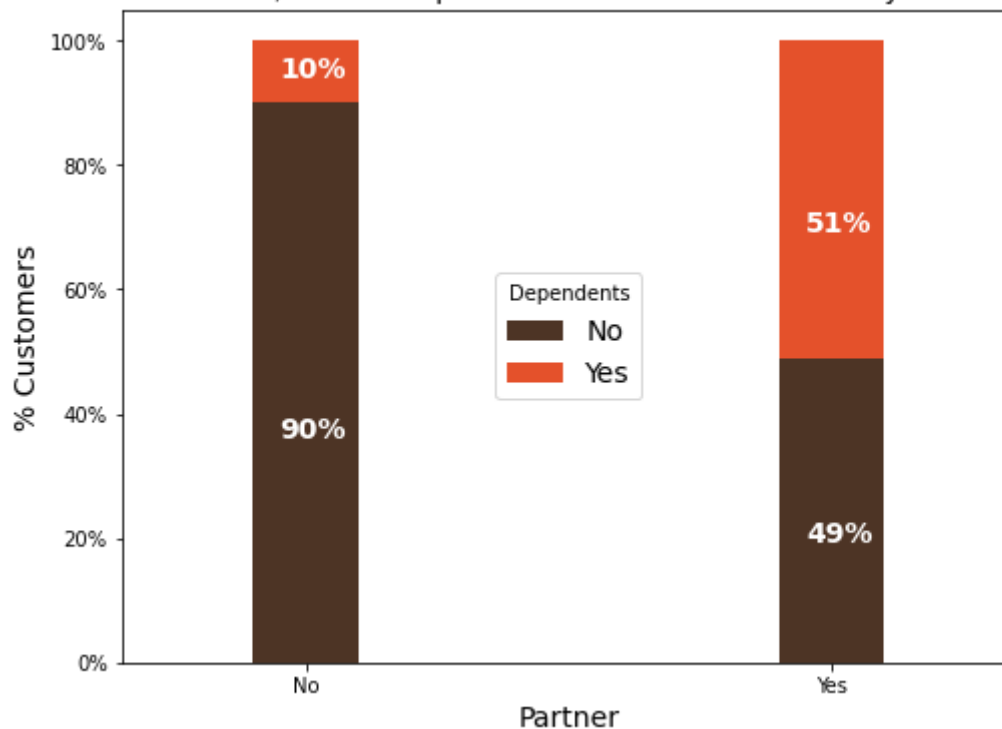
```
colors = ['#4D3425', '#E4512B']
partner_dependents = telecom_cust.groupby(['Partner', 'Dependents']).size().unstack()

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',
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)
```

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

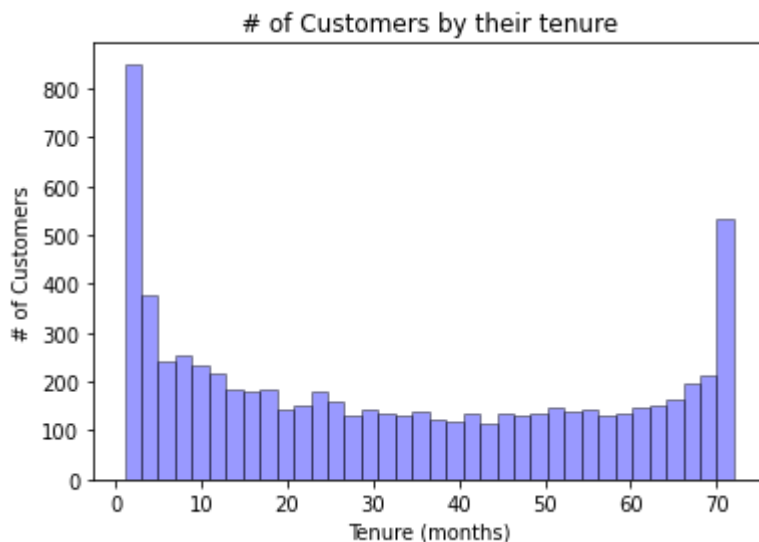


In [55]:

```
ax = sns.distplot(telecom_cust['tenure'], hist=True, kde=False,
                  bins=int(180/5), color = 'blue',
                  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')
```

Out[55]:

Text(0.5, 1.0, '# of Customers by their tenure')

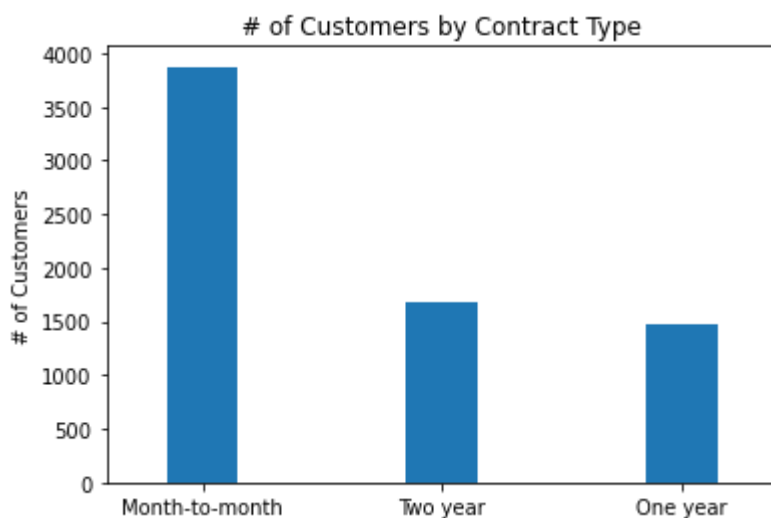


In [57]:

```
ax = telecom_cust['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[57]:

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



In [58]:

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

ax = sns.distplot(telecom_cust[telecom_cust['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')

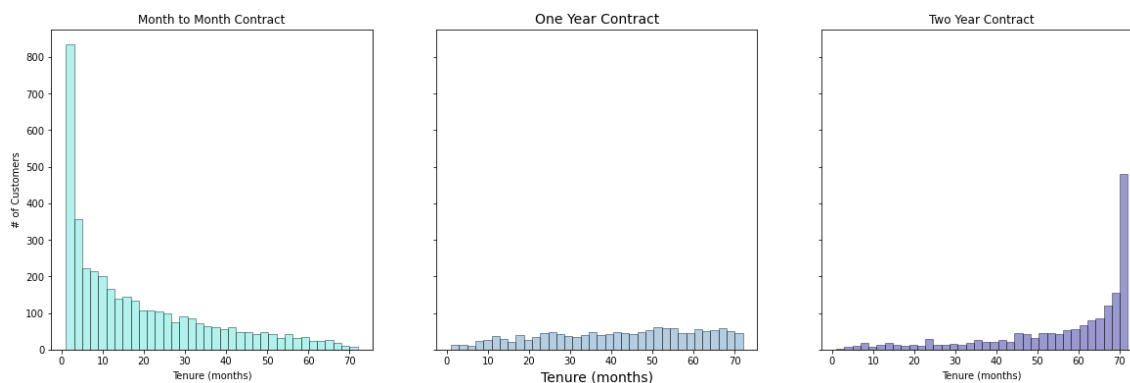
ax = sns.distplot(telecom_cust[telecom_cust['Contract']=='One year']['tenure'],
                  hist=True, kde=False,
                  bins=int(180/5), color = 'steelblue',
                  hist_kws={'edgecolor':'black'},
                  kde_kws={'linewidth': 4},
                  ax=ax2)
ax.set_xlabel('Tenure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

ax = sns.distplot(telecom_cust[telecom_cust['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')
```

Out[58]:

Text(0.5, 1.0, 'Two Year Contract')



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

In [59]:

```

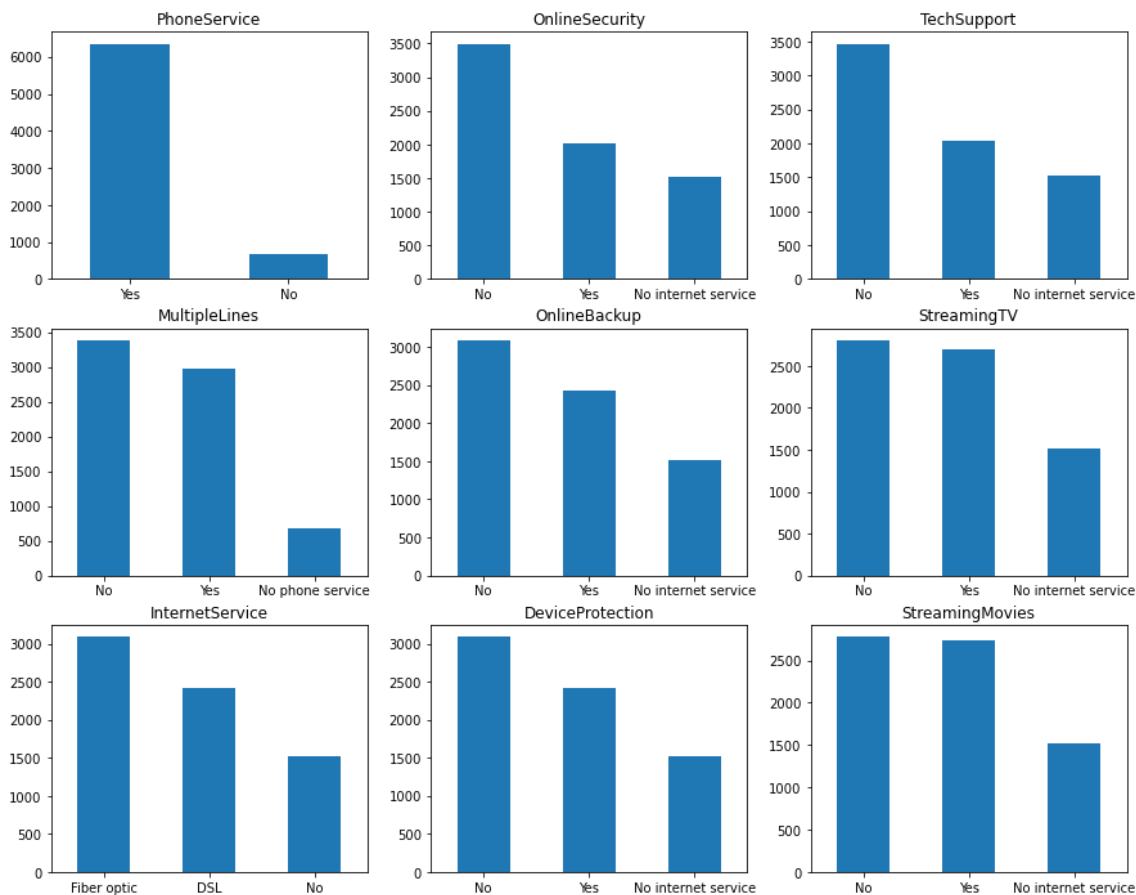
services = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
            'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovie']

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

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

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

```

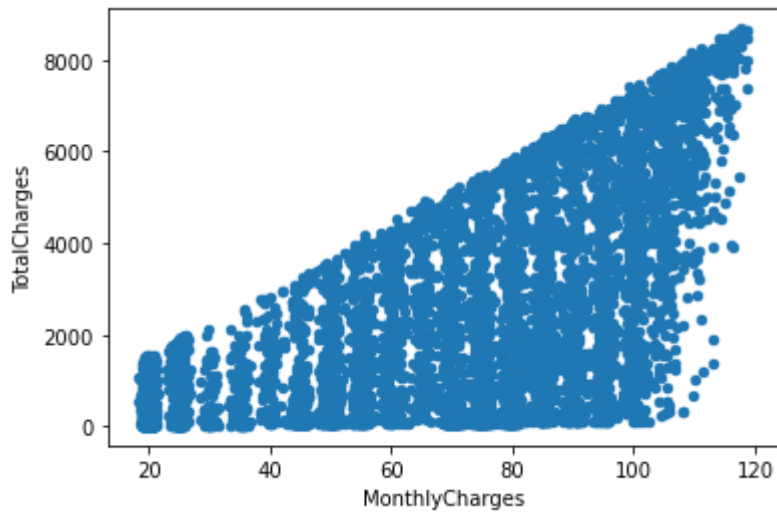


In [61]:

```
telecom_cust[['MonthlyCharges', 'TotalCharges']].plot.scatter(x = 'MonthlyCharges',  
y='TotalCharges')
```

Out[61]:

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



In [65]:

```
colors = ['#4D8425', '#E2912B']
ax = (telecom_cust['Churn'].value_counts()*100.0 / len(telecom_cust)).plot(kind='bar',
                                     stacked = True,
                                     rot = 0,
                                     color = colors,
                                     figsize = (8,6))

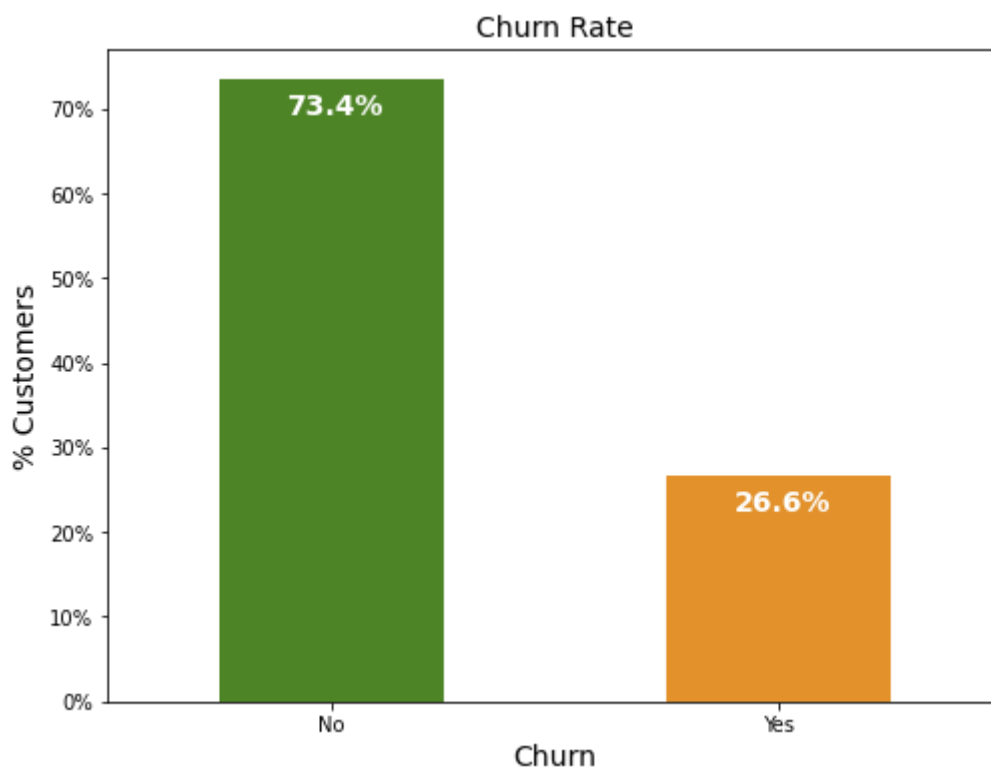
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('% Customers', size = 14)
ax.set_xlabel('Churn', size = 14)
ax.set_title('Churn Rate', size = 14)

# 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 labels 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)
```

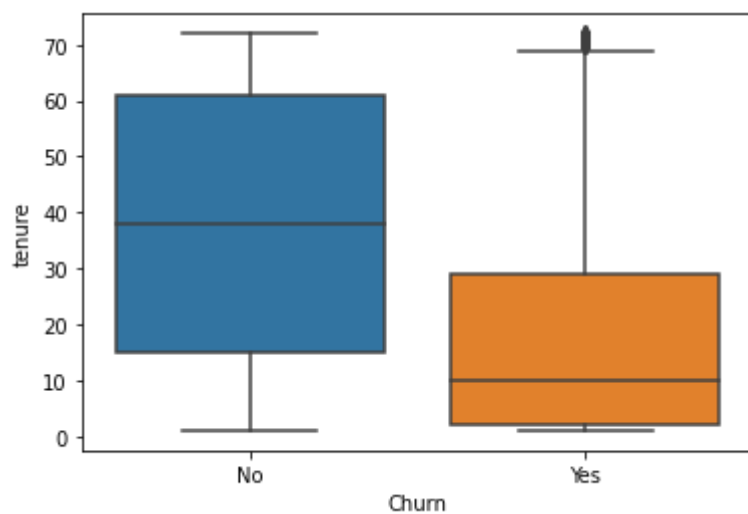


In [66]:

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

Out[66]:

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



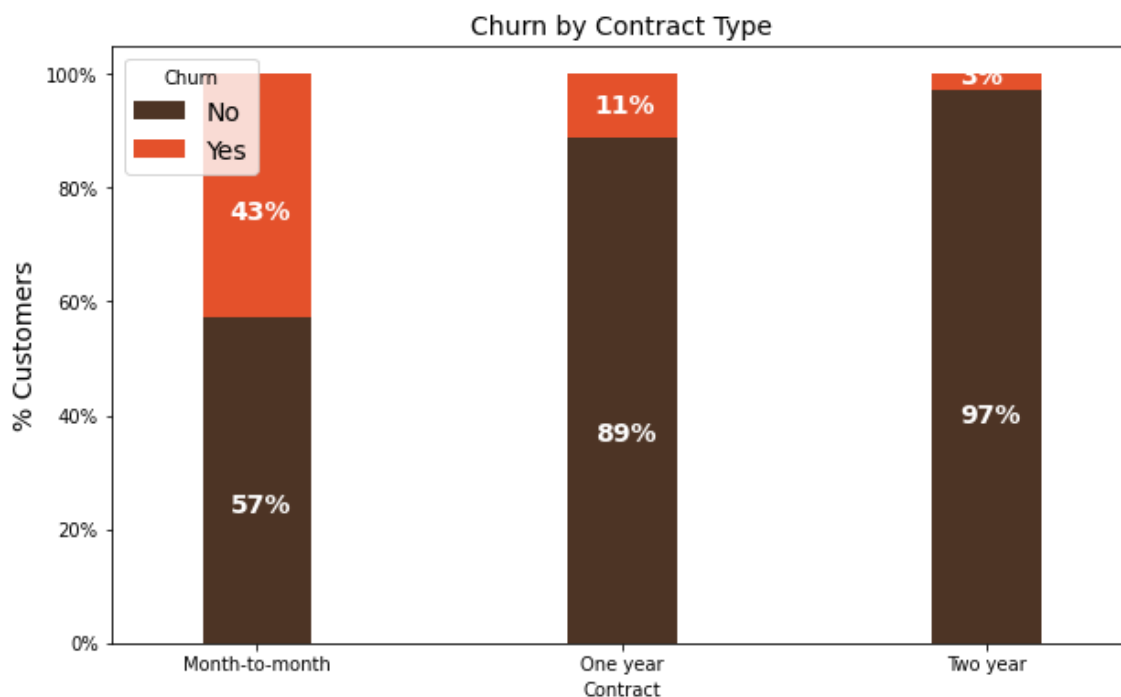
In [67]:

```
colors = ['#4D3425', '#E4512B']
contract_churn = telecom_cust.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,
                                                             figsize = (10,6),
                                                             color = colors)

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 = 'bold',
               size = 14)
```



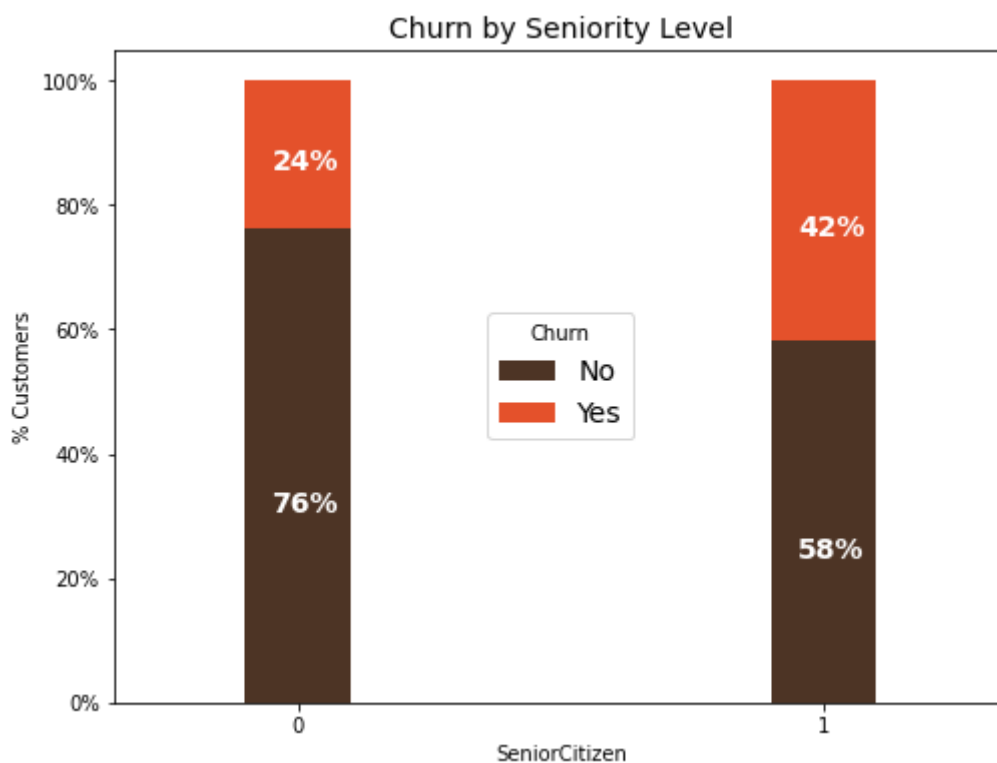
In [68]:

```
colors = ['#4D3425', '#E4512B']
seniority_churn = telecom_cust.groupby(['SeniorCitizen', 'Churn']).size().unstack()

ax = (seniority_churn.T*100.0 / seniority_churn.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 = 'Churn')
ax.set_ylabel('% Customers')
ax.set_title('Churn by Seniority Level',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)
```

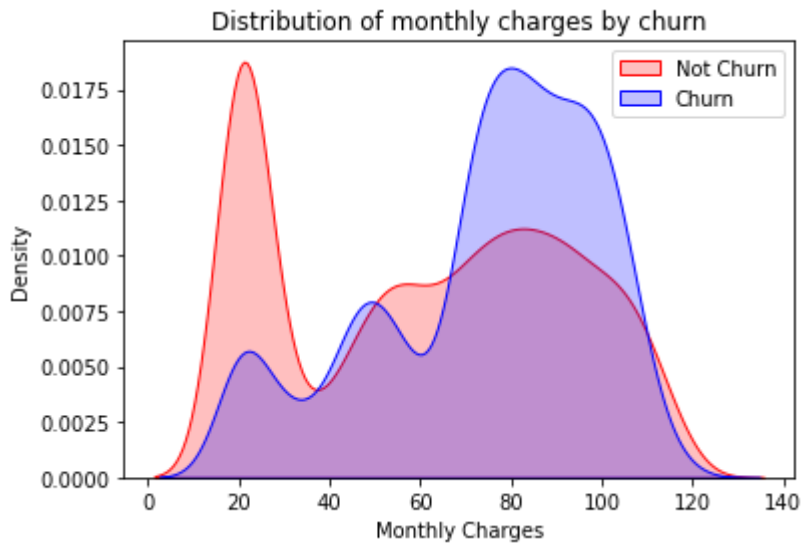


In [69]:

```
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'No') ],  
                color="Red", shade = True)  
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["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')
```

Out[69]:

Text(0.5, 1.0, 'Distribution of monthly charges by churn')

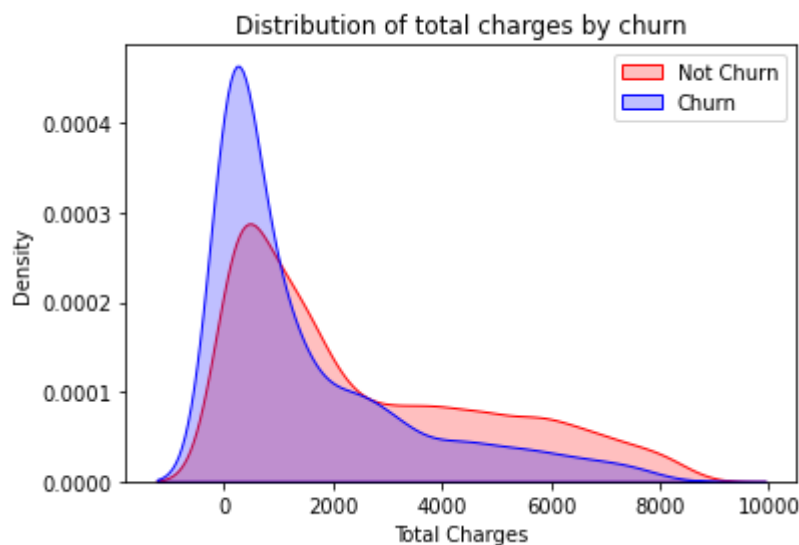


In [70]:

```
ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust["Churn"] == 'No') ],
                color="Red", shade = True)
ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust["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')
```

Out[70]:

Text(0.5, 1.0, 'Distribution of total charges by churn')



In [71]:

```
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 [72]:

```
# 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=10)
```

In [73]:

```
# Running Logistic regression model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result = model.fit(X_train, y_train)
```

In [74]:

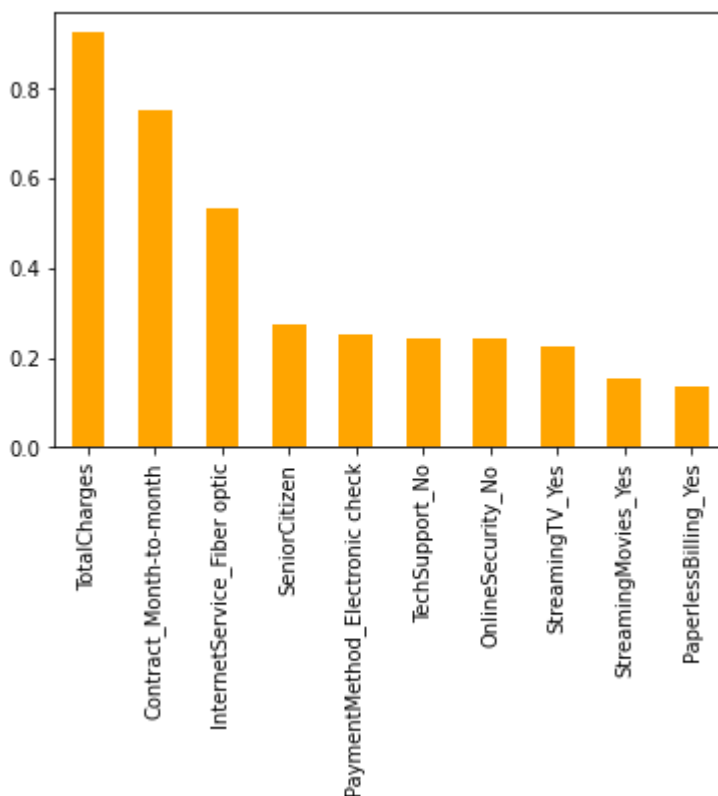
```
from sklearn import metrics
prediction_test = model.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

0.8075829383886256

In [78]:

```
# 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',color="orange"))
```

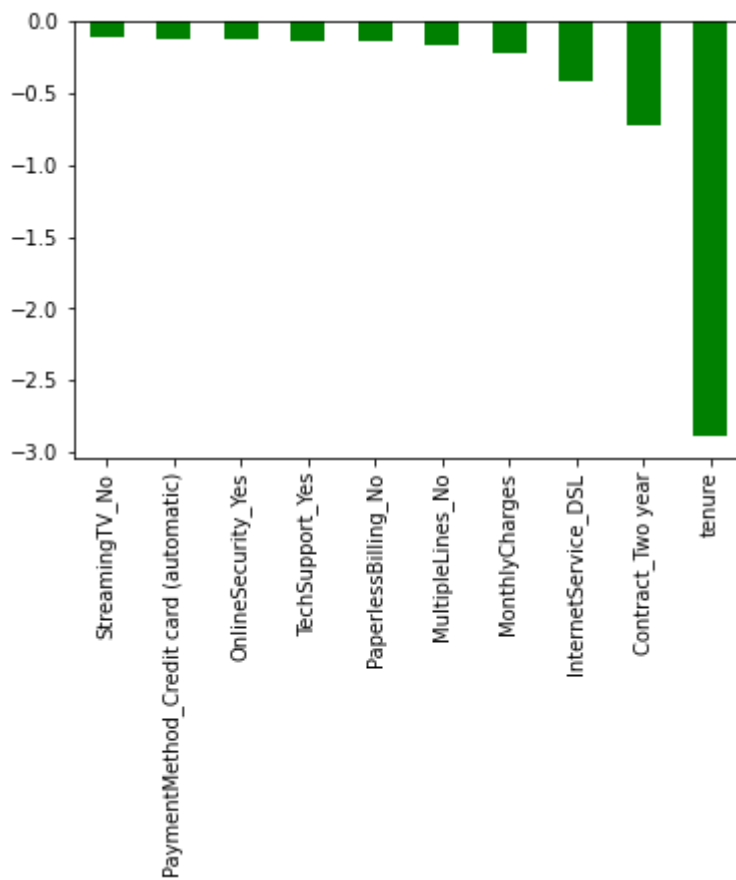
AxesSubplot(0.125,0.125;0.775x0.755)



In [82]:

```
print(weights.sort_values(ascending = False)[-10:].plot(kind='bar',color="green"))
```

AxesSubplot(0.125,0.125;0.775x0.755)



In []: