In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
```

Telco Company Customer Churn Analysis

Goal

This project is aiming to figure out the behaviour of churning customers of the telco company last month, and come up with a model to predict the churning rate, providing possible strategies for further improvement.

Key takeaway

1. There were over one-quarter of customers churning last month, and churning customers' behavior can be stated as below:

- No obvious patterns in gender
- Churning group mainly consists of the young generation, while it is also nonnegligible that almost 42% of senior citizens churned.
- People with no dependency are easier to churn than people with partners or dependents.
- Over 50% of customers are holding month-to-month contracts, and around 43% of them churned last month. Customers with a long contract are more stable.
- Churning customers tend to have higher charges, in both monthly charges and total charges. Services leading to a relatively larger variety in charges (difference between with certain service and without certain service > 20 dollars) includes phone services (especially with multiple lines), streaming TV, Fiber optic Internet service, and streaming movies.

2. Prediction:

- Both the Logistic regression model and Random forest model have an accuracy of around 80% in predicting customers' retention.
- Most important attributes affecting customer's retention are tenure, month-to-month contract, and total charges, while month-to-month contract and total charges negatively affect retention, which implies that a customer holding a month-to-month contract or a customer charged a higher price will have a higher odds of churning.

3. Recommondation:

- Young generation is taking up 75% of total customers, so earning their loyalty could be the key to decreasing the churning rate.
- Senior citizens and dependent people take up a small proportion of the total customers, and thier churn rates are high, which implies current services or plans may not be suitable for them. Hence, some creative services could be provided to attract senior citizens or independent customers, considering their needs and customizing flexible services for them.
- Attracting customers to purchase longer contracts by lowering monthly charges or rewarding them for a long tenure.
- Collecting customers' feedback, regarding not only technology but also customer services and their current plans. Do they pay bills smoothly? Do they satisfy with the current plan? Is the signal stable? Do they get instant help from online help desk?

Limitation

- We can not be sure about the causation for churning, since the churning may be caused by the customer life cycle, which is normal in a reasonable range. There is only very limited data from one month, if data from previous months are available, then we can compare churn rate in previous months with this month to see whether there is a consistent trend in churning in general or within each group, in order to make informed decisions to improve customer retention.

Analysis

1. Cleasing

In [4]:
```
telecom_cust = pd.read_csv(r"C:/Users/huxia/Downloads/Telco-Customer-Churn.csv")
```

In [5]:
```
telecom_cust.head()
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	Device
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

Check data type:

In [7]:
```
telecom_cust.dtypes
```

Out[7]:
```
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
```

TotalCharges should be a numerical variable

In [8]:
```
telecom_cust.TotalCharges = pd.to_numeric(telecom_cust.TotalCharges,errors='coerce')
```

Check missing values

In [9]:
```python
telecom_cust.isnull().sum()
```

Out[9]:
```
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
```

There are 14 entries in TotalCharges are empty. We will impute the missing values with mean of the TotalCharges.

In [10]:
```python
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan)
telecom_cust['TotalCharges'] = pd.DataFrame(imputer.fit_transform(telecom_cust[['TotalCharges']]))
telecom_cust.isnull().sum()
```

Out[10]:
```
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        0
Churn               0
dtype: int64
```

All data are well-prepared for further analysis now.

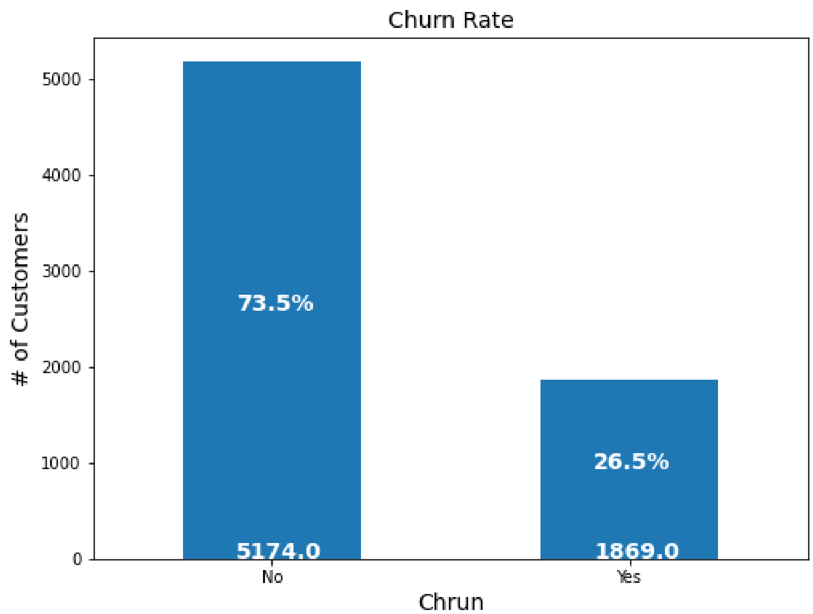
2. Explortary Data Analysis

2.1 Churn Rate Overview

In [11]:

```
churn_rate = telecom_cust['Churn'].value_counts()
ax = churn_rate.plot(kind = 'bar',stacked = True, rot = 0, figsize = (8,6))
#ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_xlabel('Chrun', size = 14)
ax.set_ylabel('# of Customers', size = 14)
ax.set_title('Churn Rate', size = 14)
totals = []
for i in ax.patches:
     totals.append(i.get_height())

total = sum(totals)
for p in ax.patches:
     width, height = p.get_width(), p.get_height()
     x,y = p.get_xy()
     ax.annotate('{:.1f}%'.format(height/total*100),
                 (x+0.3*width,height*0.5),
                 color = 'white',
                 weight = 'bold',
                 size = 14)
     ax.annotate('{:.1f}'.format(height),
                 (x+0.3*width, y),
                 color = 'white',
                 weight = 'bold',
                 size = 14)
```



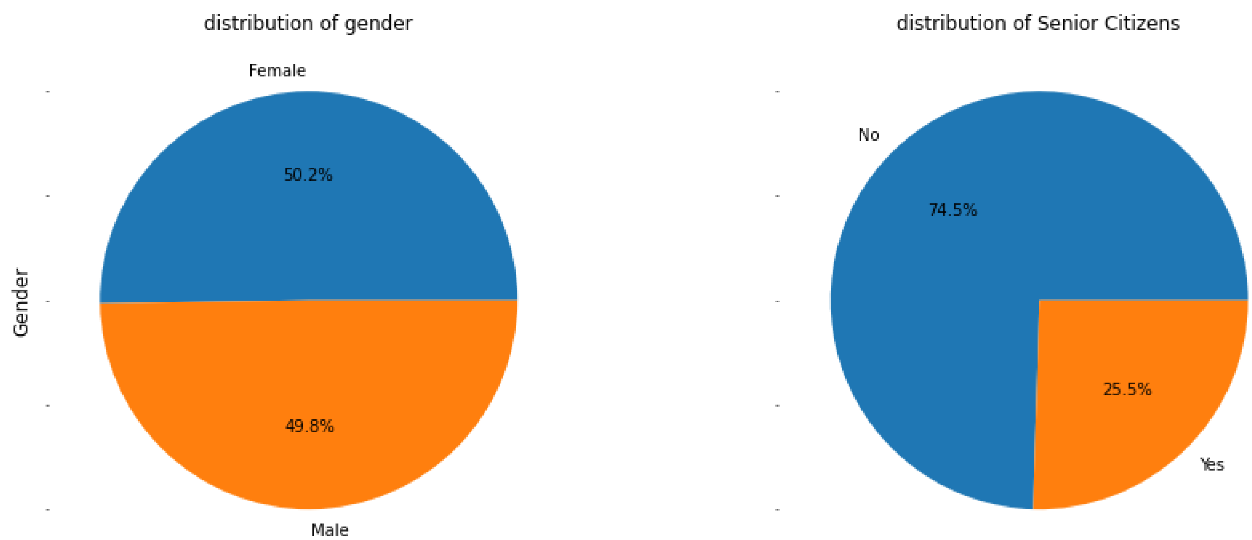
There were 26.5% customer churning in the last month.

2.2 Churning Customer Distribution in Gender and Age

In [12]:
```
fig,(ax1,ax2) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize= (15,6))
temp = telecom_cust.gender[telecom_cust['Churn'] == 'Yes']
temp.replace(0,'Female',inplace = True)
temp.replace(1,'Male',inplace = True)
a_counts = temp.value_counts()
total_num = len(temp)
ax1 = (a_counts*100/total_num).plot.pie(labels=['Female','Male'],fontsize = 10,ax = ax1,autopct='%1.1f%%')
ax1.yaxis.set_major_formatter(mtick.PercentFormatter())
ax1.set_ylabel('Gender', fontsize = 12)
ax1.set_title('distribution of gender')

b_counts = telecom_cust.SeniorCitizen[telecom_cust['Churn'] == 'Yes'].value_counts()
ax2 = (b_counts*100/total_num).plot.pie(labels=['No','Yes'],fontsize = 10,ax = ax2,autopct='%1.1f%%')
ax2.yaxis.set_major_formatter(mtick.PercentFormatter())
ax2.set_ylabel('Senior Citizens', fontsize = 12)
ax2.set_title('distribution of Senior Citizens')
```

Out[12]:
```
Text(0.5, 1.0, 'distribution of Senior Citizens')
```



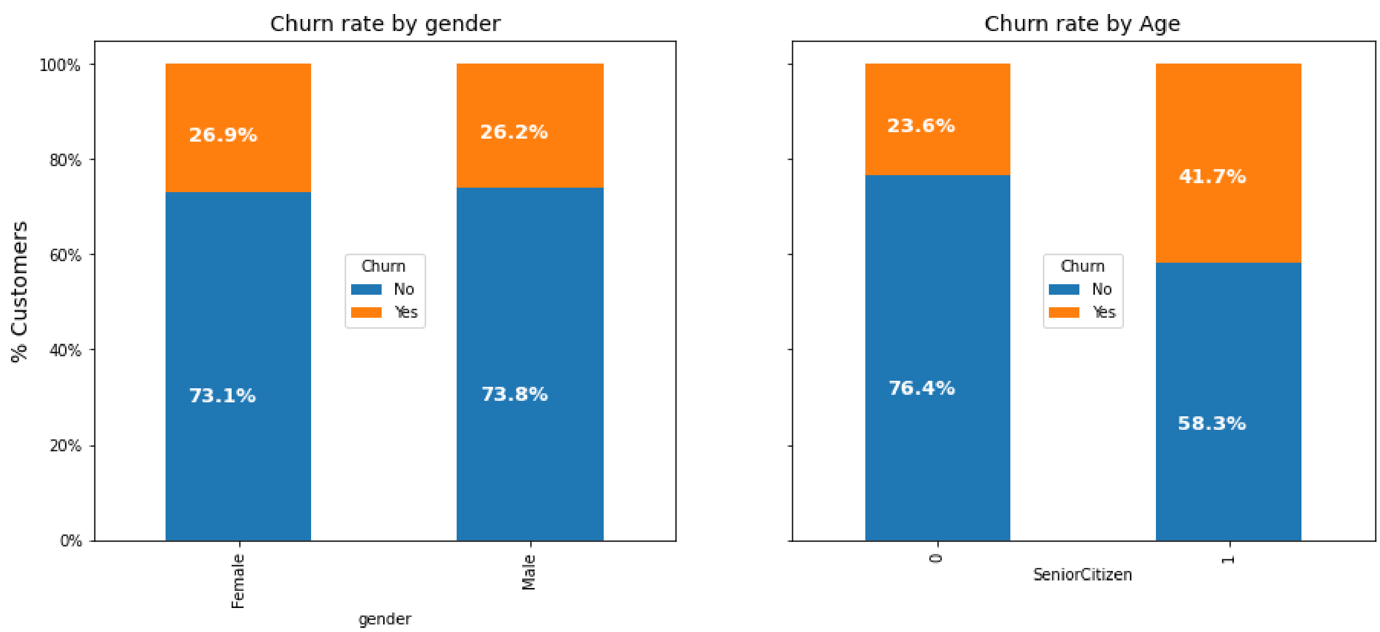

```
The churning customers are distributed evenly in females and males, and most of them are young generation. However,
the distribution may be affected by original consumer features, so the reason for the large proportion of young
citizens in churning customers may be that the majority of customers in this telco company are young. In order to
look into an accurate and unbiased changing pattern, we need to check the variety of retention within each group.
```

In [13]:

```
gender_churn = telecom_cust.groupby(['gender','Churn']).size().unstack()
SeniorCitizen_churn = telecom_cust.groupby(['SeniorCitizen','Churn']).size().unstack()

fig,(ax1,ax2) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize= (15,6))
ax= (gender_churn.T/gender_churn.T.sum()*100).T.plot(kind= 'bar',stacked=True,ax=ax1)
#adjust fig
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc= 'center',title= 'Churn')
ax.set_ylabel('% Customers', size= 14)
ax.set_title('Churn rate by gender', size= 14)
#add data labels
for p in ax.patches:
    width, height=p.get_width(),p.get_height()
    x,y=p.get_xy()
    ax.annotate('{:.1f}%'.format(height),
            (x+0.15*width,y+0.4*height),
            color= 'white',
            weight= 'bold',
            size= 13)

ax= (SeniorCitizen_churn.T/SeniorCitizen_churn.T.sum()*100).T.plot(kind= 'bar',stacked=True,ax=ax2)
#adjust fig
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc= 'center',title= 'Churn')
ax.set_ylabel('% Customers', size= 14)
ax.set_title('Churn rate by Age', size= 14)
#add data labels
for p in ax.patches:
    width, height=p.get_width(),p.get_height()
    x,y=p.get_xy()
    ax.annotate('{:.1f}%'.format(height),
            (x+0.15*width,y+0.4*height),
            color= 'white',
            weight= 'bold',
            size= 13)
```



```
Although the young generation takes up a large proportion of churning customers, there were 41.7% senior citizens
churning last month, which is way more than the churning percentage among young citizens (23.6%).
```

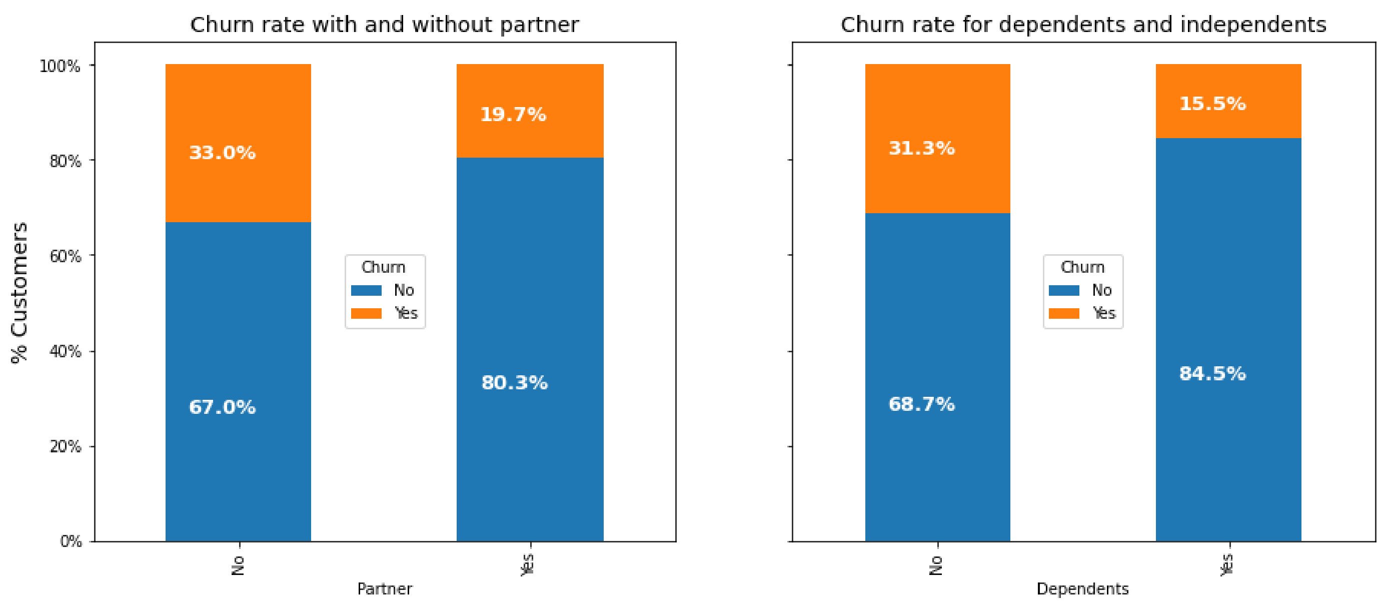
2.3 Churn rate by dependency

In [14]:

```
partner_churn = telecom_cust.groupby(['Partner','Churn']).size().unstack()
dependent_churn = telecom_cust.groupby(['Dependents','Churn']).size().unstack()

fig,(ax1,ax2) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize= (15,6))
ax= (partner_churn.T/partner_churn.T.sum()*100).T.plot(kind= 'bar',stacked=True,ax=ax1)
#adjust fig
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc= 'center',title= 'Churn')
ax.set_ylabel('% Customers', size= 14)
ax.set_title('Churn rate with and without partner', size= 14)
#add data labels
for p in ax.patches:
    width, height=p.get_width(),p.get_height()
    x,y=p.get_xy()
    ax.annotate('{:.1f}%'.format(height),
            (x+0.15*width,y+0.4*height),
            color= 'white',
            weight= 'bold',
            size= 13)

ax= (dependent_churn.T/dependent_churn.T.sum()*100).T.plot(kind= 'bar',stacked=True,ax=ax2)
#adjust fig
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.legend(loc= 'center',title= 'Churn')
ax.set_ylabel('% Customers', size= 14)
ax.set_title('Churn rate for dependents and independents', size= 14)
#add data labels
for p in ax.patches:
    width, height=p.get_width(),p.get_height()
    x,y=p.get_xy()
    ax.annotate('{:.1f}%'.format(height),
            (x+0.15*width,y+0.4*height),
            color= 'white',
            weight= 'bold',
            size= 13)
```



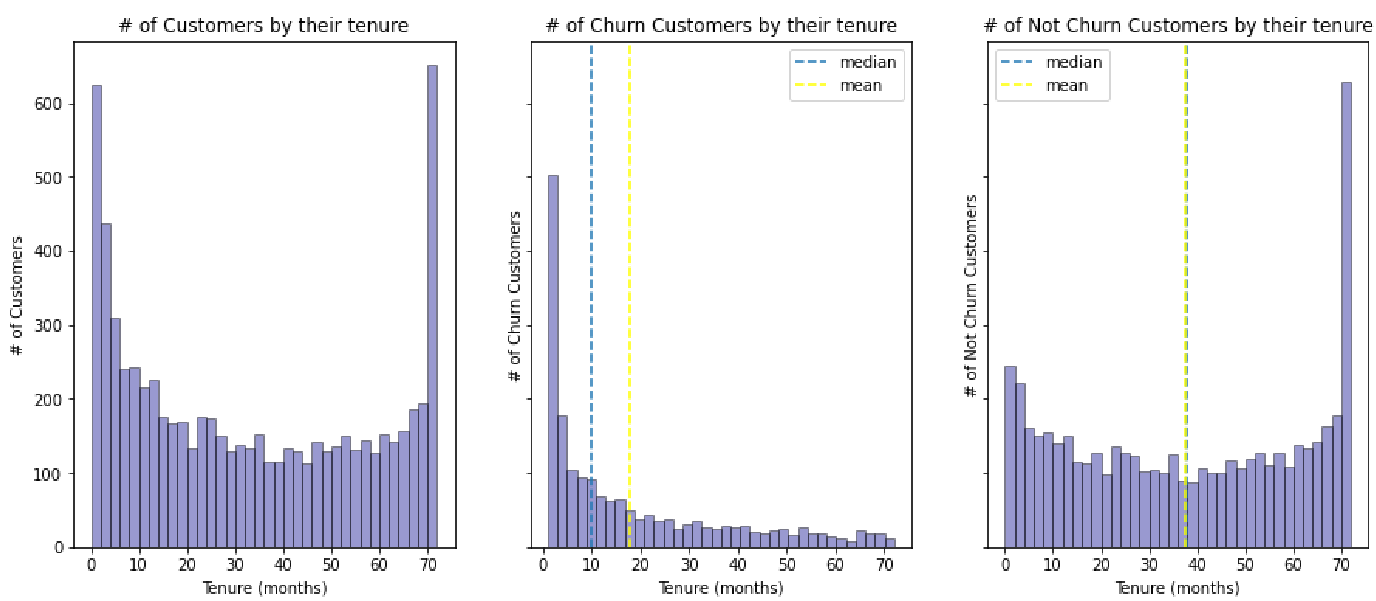
Customers who have no partner or who are independents are easier to churn.

2.4 Chur rate and Tenture

In [15]:
```
fig,(ax1,ax2,ax3) = plt.subplots(nrows = 1, ncols=3, sharey=True, figsize = (15,6))
ax = sns.distplot(telecom_cust['tenure'],hist=True, kde = False,
 bins = 36,
 color= 'darkblue',hist_kws={'edgecolor':'black'},
 kde_kws={'linewidth':4},
 ax = ax1)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('# of Customers by their tenure')
x = telecom_cust.tenure[telecom_cust['Churn'] == 'Yes']
ax = sns.distplot(x,hist=True, kde = False,
 bins = 36,
 color= 'darkblue',hist_kws={'edgecolor':'black'},
 kde_kws={'linewidth':4},
 ax = ax2)
ax.set_ylabel('# of Churn Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('# of Churn Customers by their tenure')
ax.axvline(x.median(),linestyle = '--')
ax.axvline(x.mean(),linestyle = '--',color = 'yellow')
ax.legend(['median','mean'],loc = 'upper right')
x = telecom_cust.tenure[telecom_cust['Churn'] == 'No']
ax = sns.distplot(x,hist=True, kde = False,
 bins = 36,
 color= 'darkblue',hist_kws={'edgecolor':'black'},
 kde_kws={'linewidth':4},
 ax = ax3)
ax.axvline(x.median(),linestyle = '--')
ax.axvline(x.mean(),linestyle = '--',color = 'yellow')
ax.legend(['median','mean'],loc = 'upper left')
ax.set_ylabel('# of Not Churn Customers')
ax.set_xlabel('Tenure (months)')
ax.set_title('# of Not Churn Customers by their tenure')
```

```
C:\Users\huxia\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]: Text(0.5, 1.0, '# of Not Churn Customers by their tenure')



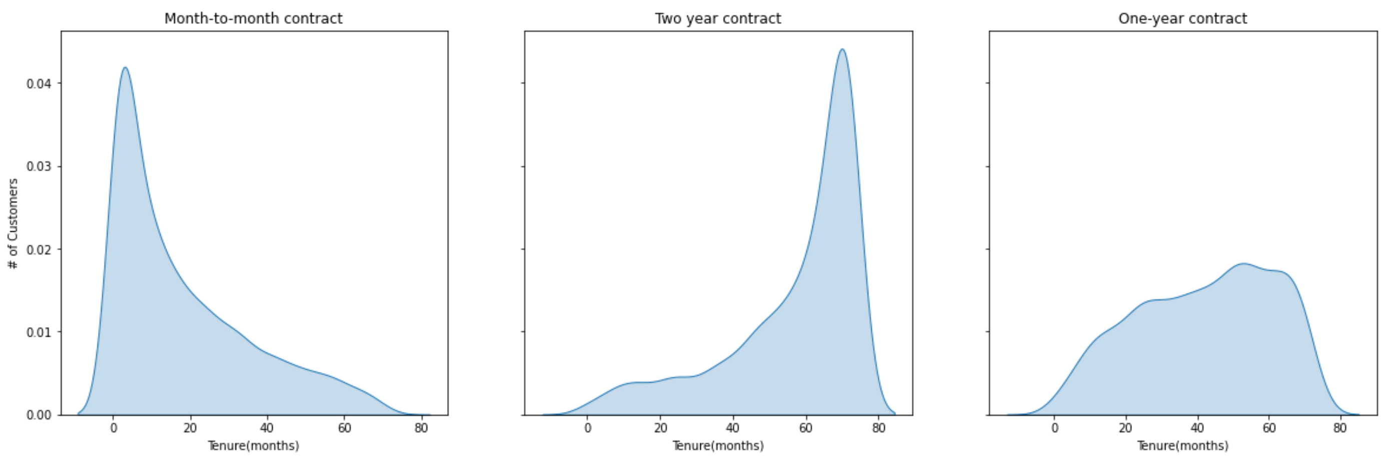
- Churning customers tend to have shorter tenure, and the medium tenure for churning customers is less than 20 months.
- Retention customers have much more even distribution in tenure between zero to sixties months, while many people have 72 months tenure. The medium tenure for retention customers is around 40 months, which is twice more than churning customers'.

2.5 Distribution of Churning Customer over Tenure by Different Contract Type

In [16]:

```
#subplot
fig,(ax1,ax2,ax3) = plt.subplots(nrows = 1, ncols=3, sharey=True, figsize = (20,6))
#plot 1 conditional subset's count: telecom_cust[telecom_cust['Contract']=='Month-to-month']
ax = sns.kdeplot(telecom_cust[telecom_cust['Contract']=='Month-to-month']['tenure'],
 shade=True,
 ax=ax1)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure(months)')
ax.set_title('Month-to-month contract')
ax = sns.kdeplot(telecom_cust[telecom_cust['Contract']=='Two year']['tenure'],
 shade = True,
 ax=ax2)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure(months)')
ax.set_title('Two year contract')
ax = sns.kdeplot(telecom_cust[telecom_cust['Contract']=='One year']['tenure'],
 shade = True,
 ax=ax3)
ax.set_ylabel('# of Customers')
ax.set_xlabel('Tenure(months)')
ax.set_title('One-year contract')
```

Out[16]: Text(0.5, 1.0, 'One-year contract')



- People with Month-to-Month contracts tend to churn with less than 1-year tenure
- People with Two-year contracts tend to churn after 3-year tenure
- People with One-year contracts have an even distribution of tenure, and the medium is around 40 months.

2.6 Churn Rate by Contract Type

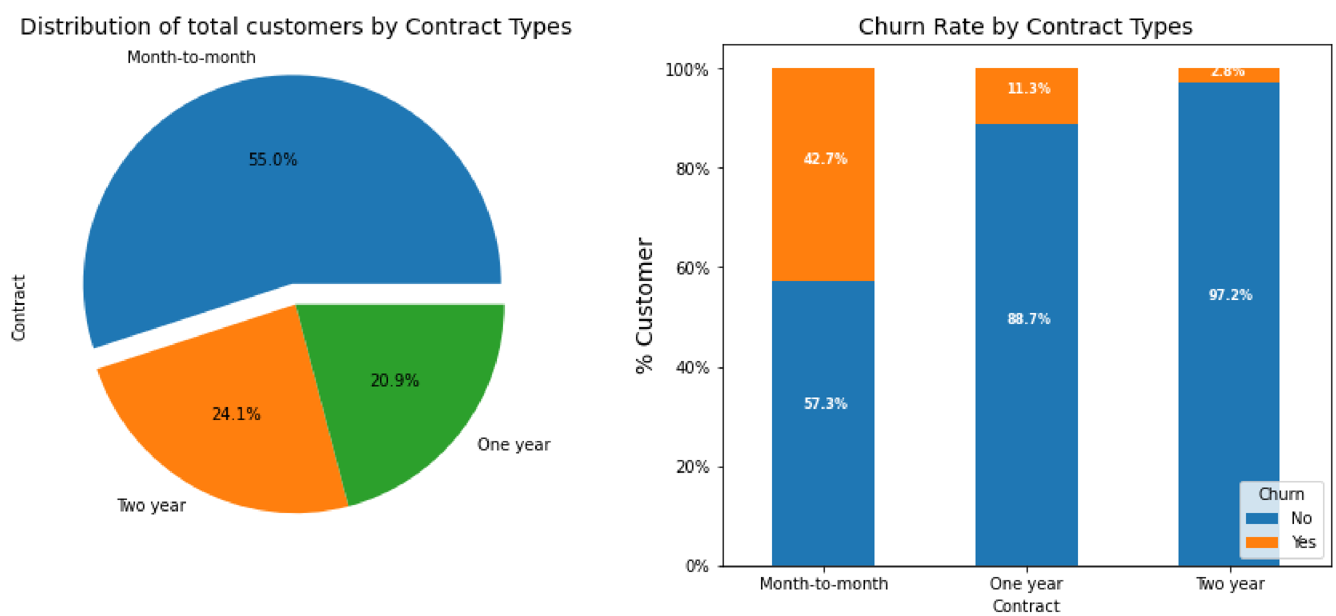
In [17]:
```python
fig,(ax1,ax2) = plt.subplots(nrows=1, ncols=2, sharey=False, figsize= (15,6))

counts = telecom_cust['Contract'].value_counts()
ax1 = (counts*100/total_num).plot.pie(ax = ax1,autopct='%1.1f%%',explode=(0.1, 0, 0))
ax1.yaxis.set_major_formatter(mtick.PercentFormatter())
ax1.set_title('Distribution of total customers by Contract Types',size = 14)


df1 = telecom_cust.groupby(['Churn','Contract']).size().unstack()
df2 = df1/df1.sum()*100
ax2 = df2.T.plot(kind = 'bar', stacked = True, rot = 0, ax = ax2)
ax2.yaxis.set_major_formatter(mtick.PercentFormatter())
ax2.set_ylabel('% Customer', size = 14)
ax2.set_title('Churn Rate by Contract Types', size = 14)
for p in ax2.patches:
     width, height = p.get_width(),p.get_height()
     x,y = p.get_xy()
     ax2.annotate('{:.1f}%'.format(height),
     (x+0.3*width,y+0.55*height),
     color = 'white',
     weight = 'bold',
     size = 8)

ax2.legend(loc = 'lower right',title = 'Churn')
```

Out[17]: <matplotlib.legend.Legend at 0x2305dc16160>



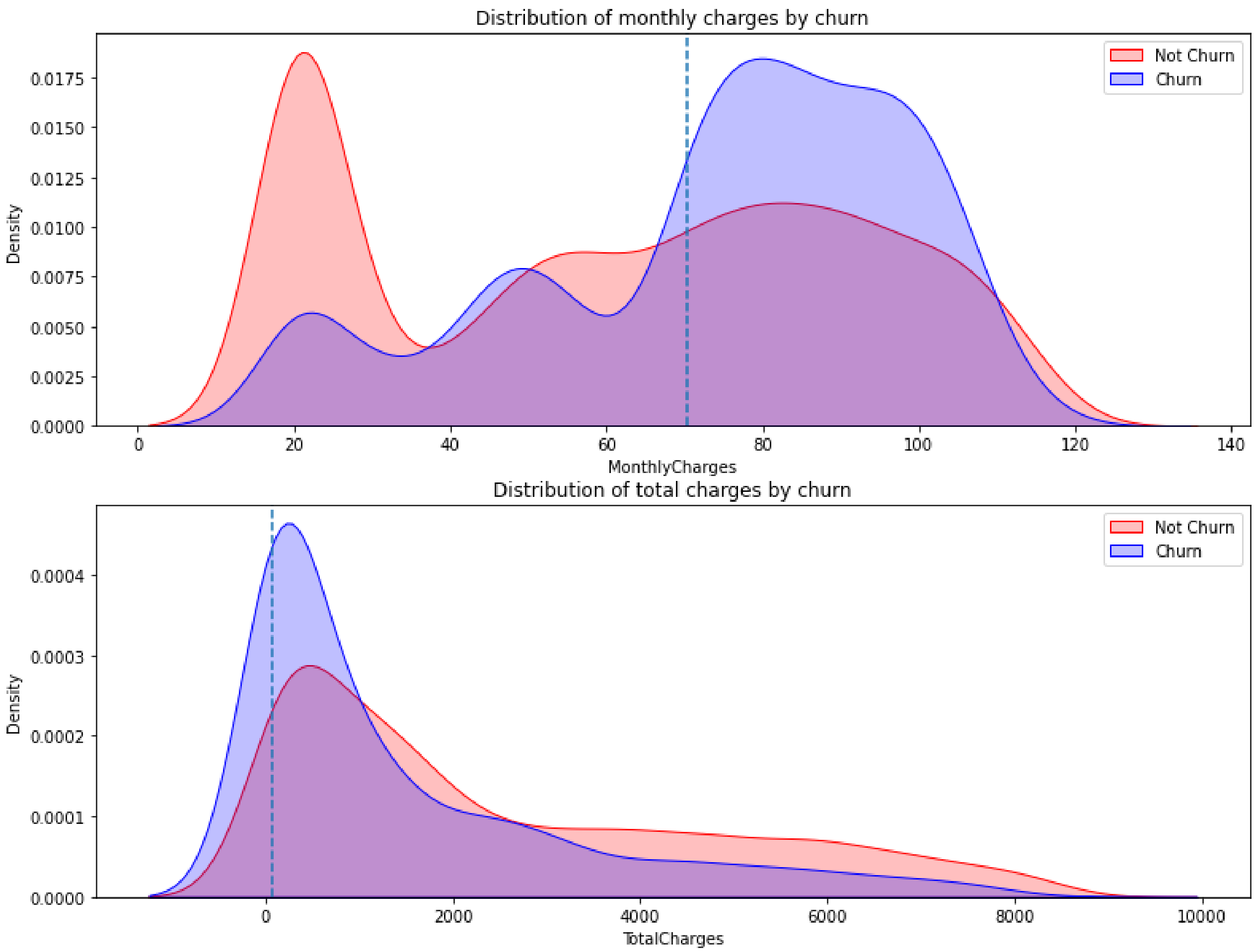
Short-term contract customers take up 55% of total customers, but almost half of Month-to-Month contract customers churned last month, while long-term contract customers are relatively more stable. There were 11.3% churning in one-year contract customers and 2.8% churning in two-year contract customers.

2.7 Density of Mothly Charges by Churning and Not Churning Customers

In [18]:

```
fig,(ax1,ax2) = plt.subplots(nrows = 2, ncols=1, sharey=False, figsize = (13,10))
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust['Churn'] == 'No')], color = 'Red', shade = True,ax = a
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust['Churn'] == 'Yes')], color = 'Blue', shade = True, ax
ax.legend(['Not Churn','Churn'],loc='upper right')
ax.set_title('Distribution of monthly charges by churn')
ax.axvline(telecom_cust.MonthlyCharges.median(),linestyle = '--')
ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust['Churn'] == 'No')], color = 'Red', shade = True, ax = ax
ax = sns.kdeplot(telecom_cust.TotalCharges[(telecom_cust['Churn'] == 'Yes')], color = 'Blue', shade = True, ax =
ax.legend(['Not Churn','Churn'],loc='upper right')
ax.set_title('Distribution of total charges by churn')
ax.axvline(telecom_cust.MonthlyCharges.median(),linestyle = '--')
```

Out[18]: <matplotlib.lines.Line2D at 0x2305de8b130>

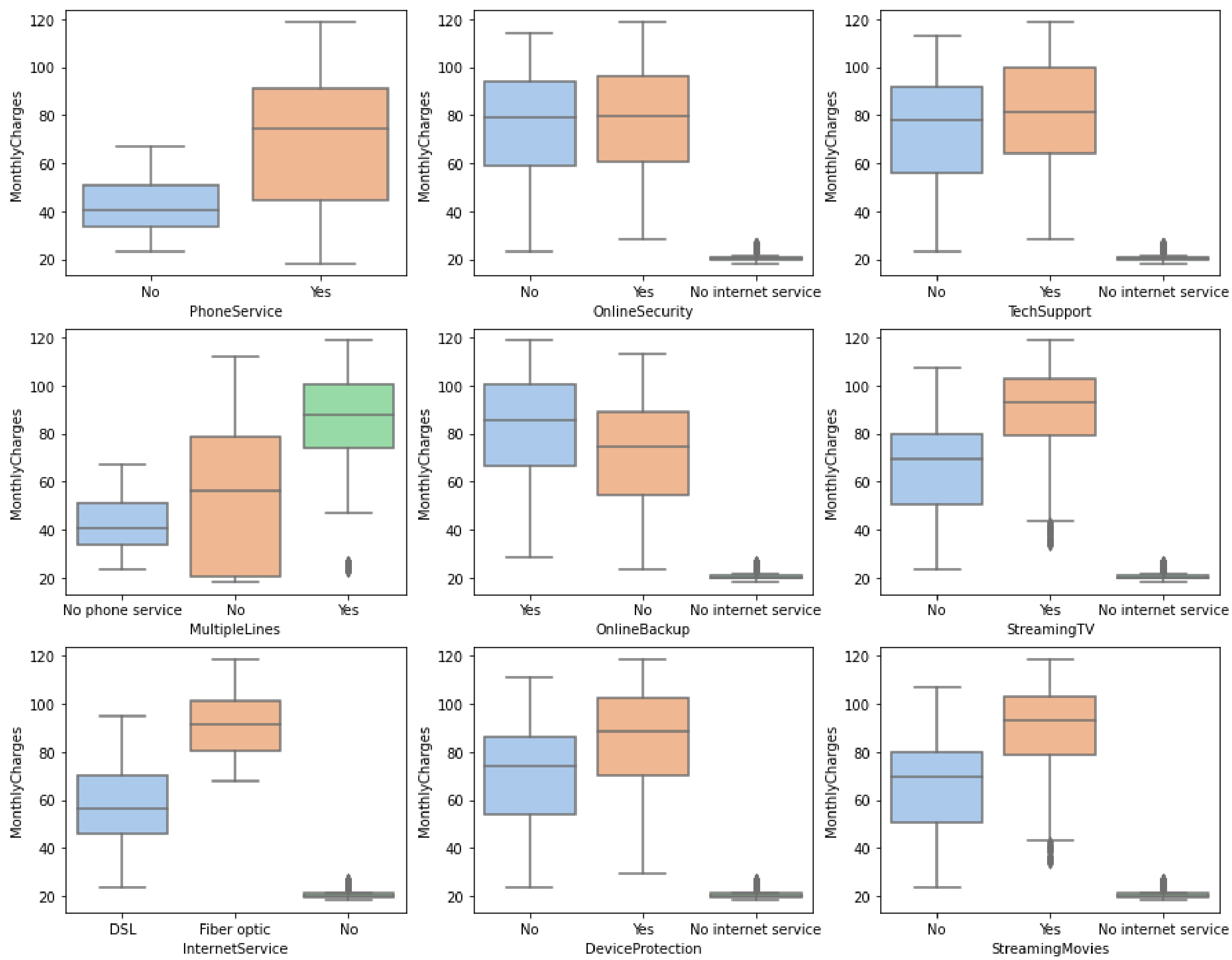


Churing customers tend to have higher charges, in both monthly charges and total charges.

2.8 What affects monthly charges?

In [19]:

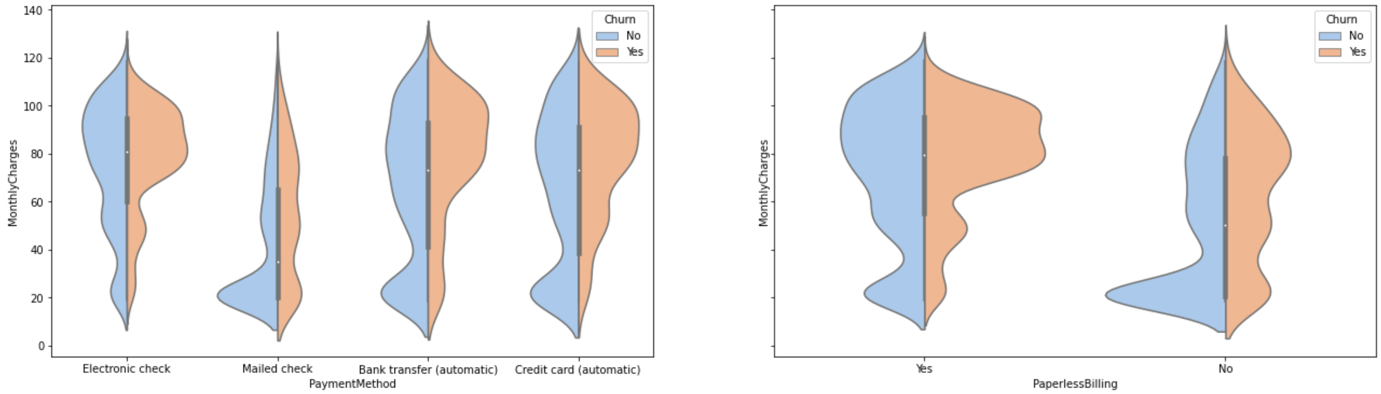
```
services = ['PhoneService','MultipleLines','InternetService','OnlineSecurity',
 'OnlineBackup','DeviceProtection','TechSupport','StreamingTV','StreamingMovies']
fig, axes = plt.subplots(nrows = 3, ncols = 3, figsize =(15,12))
for i,item in enumerate(services):
 if i < 3:
     sns.boxplot(x=telecom_cust[item], y="MonthlyCharges",
     palette="pastel", data=telecom_cust, ax =axes[i,0])
 elif i>=3 and i<6:
     sns.boxplot(x=telecom_cust[item], y="MonthlyCharges",
     palette="pastel", data=telecom_cust, ax =axes[i-3,1])
 elif i>=6:
     sns.boxplot(x=telecom_cust[item], y="MonthlyCharges",
     palette="pastel", data=telecom_cust, ax =axes[i-6,2])
     ax.set_title(item)
```



Seviece leading to relative large variaty in charges includes: pnhone services (espectially with multiple lines), streaming TV, Fiber optic Internet service and streaming movies.

2.9 Churn Rate by Payment method

In [20]:
```
fig,(ax1,ax2) = plt.subplots(nrows = 1, ncols=2, sharey=True, figsize = (22,6))
ax = sns.violinplot(x="PaymentMethod", y="MonthlyCharges", hue="Churn",
 split=True, palette="pastel", data=telecom_cust, height=4.2, aspect=1.4, ax = ax1)
ax = sns.violinplot(x="PaperlessBilling", y="MonthlyCharges", hue="Churn",
 split=True, palette="pastel", data=telecom_cust, height=4.2, aspect=1.4,ax = ax2)
```



Churing customers tend to pay larger amount of charges through electronic check, bank transfer or creadit card, and prefer paperless billing.

Prediction

1. Logistic Regression

In [21]:
```
#get dummy variable
df = telecom_cust.iloc[:,1:]
df['Churn'].replace(to_replace = 'Yes', value = 1, inplace = True)
df['Churn'].replace(to_replace = 'No', value = 0, inplace = True)
df_dummies = pd.get_dummies(df)
y = df_dummies['Churn'].values
x = df_dummies.drop(columns=['Churn'])
#scale all the variables to arange of 0 to 1
from sklearn.preprocessing import MinMaxScaler
features = x.columns.values #get columns name
scaler = MinMaxScaler(feature_range= (0,1)) #fit a scaler
scaler.fit(x)
#transform x through the scaler and reset the dataframe
x = pd.DataFrame(scaler.transform(x))
#set dataframe's column name
x.columns = features
```

In [22]:
```
#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 = 100)
```

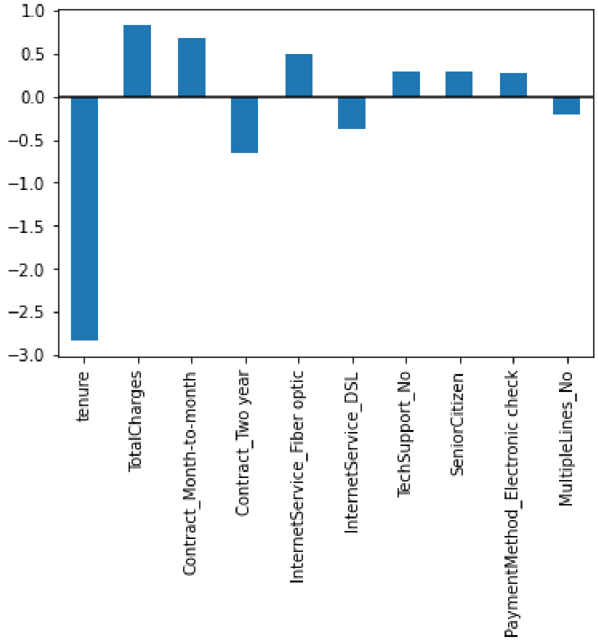
In [23]:
```
#fit logistic regression model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result = model.fit(x_train,y_train)
```

In [24]:
```
#accuracy
from sklearn import metrics
prediction_test = model.predict(x_test)
print(metrics.accuracy_score(y_test,prediction_test))
```

```
0.791292001893043
```

In [25]:
```python
#weight of all the variable
weights = pd.Series(model.coef_[0],index=x.columns.values) #coef is [[]], need to take [0]
ax = weights.sort_values(ascending = False,key=abs)[:10].plot(kind = 'bar')#print varabiles with largest weights
ax.axhline(0,color = 'black')
```

Out[25]: <matplotlib.lines.Line2D at 0x2305d870b50>



Most important attributes affecting customer's retention are tenure, month-to-month contract and total charges, while month-to-month contract and total charges are giving negative impacts.

2. Random Forest

In [26]:
```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
#split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 100)
n_estimators = [int(x) for x in np.linspace(start= 500, stop=2000,num=10)]
max_features = ['auto','sqrt']
max_leaf_nodes = [int(x) for x in np.linspace(start= 10, stop=500,num=10)]
#min_samples_split = [2,5]
#min_samples_leaf = [1,2]
oob_score = [True,False]
param_grid = {'n_estimators' : n_estimators,
 'max_features' : max_features,
 'max_leaf_nodes': max_leaf_nodes,
 'oob_score': oob_score
 }
```

In [27]:
```python
rf = RandomForestClassifier()
from sklearn.model_selection import RandomizedSearchCV
rf_Grid = RandomizedSearchCV(estimator = rf, param_distributions = param_grid, cv = 3, verbose = 2, n_jobs = -1)
```

In [28]:
```
rf_Grid.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:   39.2s finished
```

Out[28]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
                   param_distributions={'max_features': ['auto', 'sqrt'],
                                        'max_leaf_nodes': [10, 64, 118, 173,
                                                           227, 282, 336, 391,
                                                           445, 500],
                                        'n_estimators': [500, 666, 833, 1000,
                                                         1166, 1333, 1500, 1666,
                                                         1833, 2000],
                                        'oob_score': [True, False]},
                   verbose=2)
```

In [29]:
```
rf_Grid.best_params_
```

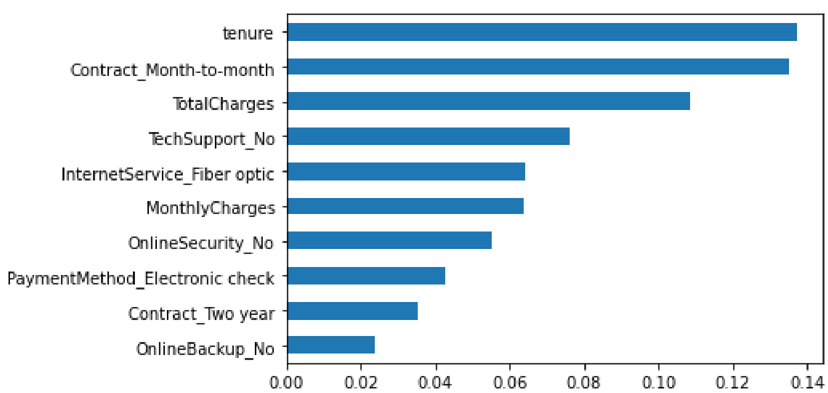
Out[29]:
```
{'oob_score': True,
 'n_estimators': 2000,
 'max_leaf_nodes': 64,
 'max_features': 'sqrt'}
```

In [30]:
```
model_rf_rscv = RandomForestClassifier(n_estimators=1166,
 oob_score= True,
 n_jobs=-1,
 random_state = 123,
 max_features='sqrt',
max_leaf_nodes=64)
#fit
model_rf_rscv.fit(x_train,y_train)
#predict
prediction_test_rscv = model_rf_rscv.predict(x_test)
#compare and get accuracy
metrics.accuracy_score(y_test,prediction_test_rscv)
```

Out[30]:
```
0.791292001893043
```

In [31]:
```
importances = model_rf_rscv.feature_importances_
weights = pd.Series(importances,index = x.columns.values)
weights.sort_values(ascending = True)[-10:].plot(kind = 'barh')
```

Out[31]: <AxesSubplot:>



Most important attributes affecting customer's retention are tenure, month-to-month contract and total charges.