Intelligent Customer Retention: Using Machine Learning for Enhanced Prediction of Telecom Customer Churn.

Abstract:- In the Telecommunication Industry, customer churn detection is one of the most important research topics that the company has to deal with retaining on-hand customers. Churn means the loss of customers due to exiting offers of the competitors or maybe due to network issues. In these types of situations, the customer may tend to cancel the subscription to a service. Churn rate has a substantial impact on the lifetime value of the customer because it affects the future revenue of the company and also the length of service. Due to a direct effect on the income of the industry, the companies are looking for a model that can predict customer churn. The model developed in this work uses machine learning techniques. By using machine learning algorithms, we can predict the customers who are likely to cancel the subscription.Using this, we can offer them better services and reduce the churn rate. These models help telecom services to make them profitable. In this model, we used a Decision Tree, Random Forest, and XGBoost.

Keywords: Telecom churn, Xgboost(Extreme Gradient Boosting) Classification algorithms, Decision Trees, Random Forest.

I.INTRODUCTION

The telecommunications sector has displayed one of the central industries in developed countries. Service companies like these suffer, particularly from the loss of valuable customers due to competitors known as customer churn. The scientific progress and the growing number of operators increased the level of opposition. Companies are pulling hard to survive in this aggressive market, depending on complicated strategies. The customer churn causes a considerable loss of telecom services and becomes a severe problem. Three main approaches have been introduced to generate more profits to get new customers, upsell the current customers, and increase the holding period of customers. However, comparing these strategies using the value of return on investment (RoI) of each into account has shown that the third approach is the most successful strategy, proves that maintaining an existing customer costs much lower than getting a new one, in extension to being held much easier than the upselling tactics. To implement the third strategy, companies have to reduce the potential of customer's churn, known as "the customer movement from one provider to another.” Customers’ churn is a significant concern in service sectors with great aggressive services. On the other hand, foretelling the customers who are expected to leave the company will serve a potentially big-hearted extra revenue source if it is given in the early phase. Many types of research confirmed that machine learning technology is highly efficient in predicting this situation. This method is applied learning from past data.

A. <u>Existing System :-</u>

Customer churn prediction has been performed using various techniques, including data mining, machine learning, and hybrid technologies.These techniques enable and support companies in identifying, predicting, and retaining churn customers. They also help industries in CRM and decision making. Most of them used decision trees in common as it is one of the recognized methods to find out the customer churn, but it is not appropriate for complex problems [1]. But the study shows that reducing the data improves the accuracy of the decision tree [2].In some cases, data mining algorithms are used for customer prediction and historical analysis. The techniques of regression trees were discussed with other commonly used data mining methods like decision trees, rule-based learning, and neural networks.

B. <u>Proposed System :-</u>

In this system, we use various algorithms like Random Forest, XGBoost & Logistic Regression to find accurate values and which helps us to predict the churn of the customer. Here we implement the model by having a dataset that is trained and tested, which makes us have maximum correct values. Fig.1 shows the proposed model for churn prediction and describes its steps. In the Initial step, data preprocessing is performed in which we do filtering data and convert data into a similar form, and then we make feature selection. In the further step prediction and classification is done using the algorithms like Random Forest, XGBoost, Logistic Regression(LR). Training and testing the model with the data set, we observe the behavior of the customer and analyze them. In the final step, we do analysis based on the results obtained and predict the customer churn.

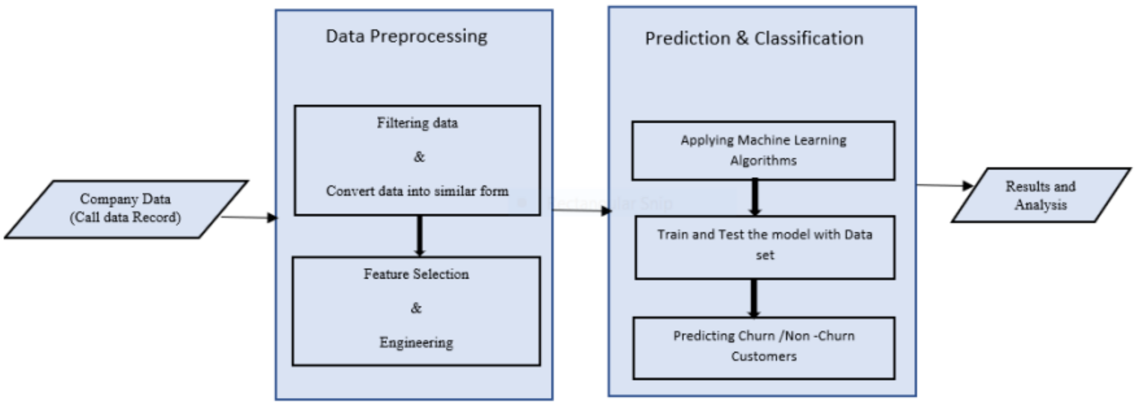
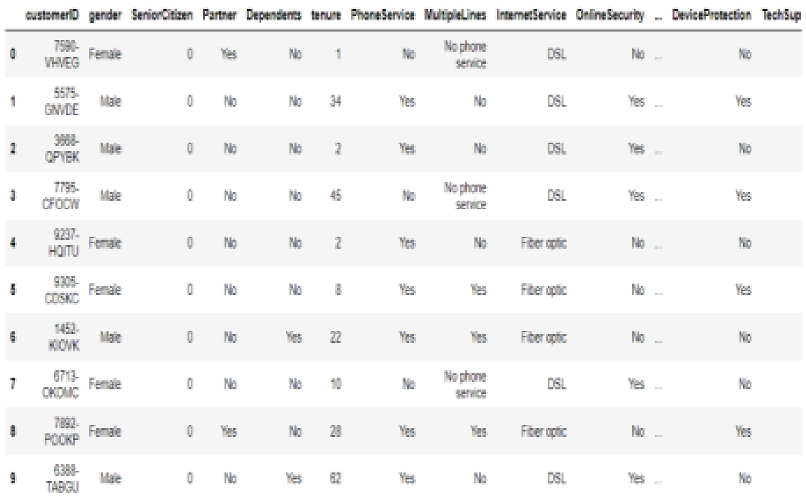

Company Data (Call data Record)
Data Preprocessing
Filtering data & Convert data into similar form
Feature Selection & Engineering
Prediction & Classification
Applying Machine Learning Algorithms
Train and Test the model with Data set
Predicting Churn /Non -Churn Customers
Results and Analysis

Fig 1. Proposed Model for Customer Churn Prediction.

II. METHODOLOGY

A. Data Set :-

As we know, the data set is the starting point for everything; it should have full-fledged data to make the machine learn about the problem. Datasets can be generated or developed from the scrap information available on the internet. Some issues we have to create a dataset that makes sense that tells how to respond based on real-time inputs for the problem datasets can be gathered from the internet every day. A dataset is a collection of data. Most commonly, a data set has contents of a single database table, or a single statistical data matrix, where every column of the table describes a particular variable, and each row matches a given member of the data set in question. The data set lists the values of the variables, such as height, the weight of an object, for each member of the data set. Each value is recognized as a datum. As we know, the data set is the starting point for this process.



	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	...	Yes	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	...	No	
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	...	No	
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	...	Yes	
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	...	No	

B. <u>Data Preprocessing :-</u>

Data set is a collection of feathers and N number of rows. Many values are in different formats. In a dataset, they may be duplicate values or null values that may lead to some loss inaccuracy, and there may be dependent. Data have been collected from different sources, so there use a different type of format to notate a single value like gender someone represents M/F or Male/Female. The machine can understand only 0 and 1, so an image will be in 3-dimension data should be reduced to a 2-dimension format like data show to free from noisy data, null values, an incorrect size. Data cleaning can be performed by panda's tabular data and OpenCV for images.

1. <u>Data Filtering and Noise Removal :-</u>

It is very crucial to make the data useful because unwanted or null values can cause unsatisfactory results or may lead to producing less accurate results. In the data set, there are a lot of incorrect values and missing values. We analyzed the whole dataset and listed out only the useful features. The listing of features can result in better accuracy and contains only valuable features.

2. <u>Feature selection & Engineering :-</u>

Feature selection is a crucial step for selecting the required elements from the data set based on the knowledge. The dataset used here consists of many features out of which we chose the needed features, which enable us to improve performance measurement and are useful for decision-making purposes while remaining will have less importance. The performance of classification increases if the dataset is having only valuable variables and which are highly predictable. Thus having only significant features and reducing the number of irrelevant attributes increases the performance of classification.

C. <u>Prediction & Classification :-</u>

Many techniques have been proposed for customer churn prediction in the telecommunication industry. In these three modeling techniques are used as predictors for the churn prediction. These techniques are outlined as :

1) <u>Random Forest :-</u>

We use Random Forest to predict whether the customer is going to cancel his subscription. Random Forest uses Decision trees for classifying whether the customer is going to cancel his subscription. The random forest consists of a large number of decision trees. A decision tree points to a specific class. A class with more number of votes will be the classifier for a particular customer. Decision trees are sensitive to the data they are trained in. To avoid this, we use Bagging. Bagging is a kind of process where we take a random sample from the dataset for training decision trees.

2) <u>Logistic Regression :-</u>

By using logistic regression, we can predict the probability of a churn i.e., the likelihood of a customer to cancel the subscription. Logistic regression is a supervised learning algorithm used for classification. In Logistic regression, we set a threshold; based on the limit, and only the classification is made using logistic regression. The threshold value is variable, and it is dependent on the classification problem itself.

3) XGBoost :-

XGBoost is the abbreviation for eXtreme Gradient Boosting. The primary purpose of using XGBoost is due to its execution speed, and it's model performance. XGBoost uses ensemble learning methods; i.e., it uses a combination of different algorithms and produces output as a single model. XGBoost supports parallel and distributed computing while offering efficient memory usage.

<u>III. PROPOSED WORK</u> :-

Initially, we will get the dataset from Kaggle, and by data filtering, we removed all the null values. Then we converted all the data into a similar form, which more natural to understand and analyze. By using Logistic regression and having a different approach, we try to implement a predictor model for the Telecom company. Here we have a customer data set, and by preprocessing and feature selection, we divide the data set for training and testing. For this algorithm, we have made some feature engineering to have more efficient and accurate results using that algorithm.

Logistic regression helps us to have a discriminative probabilistic classification and can estimate the probability of occurring event places. The dependent variable presents the event occurrence (e.g., it will be one if the event takes place, 0 otherwise). By training, the data to that model will get a result having their details, and then we will test the model with the remaining amount of data. Therefore we will get an accuracy based on the findings by which we can predict the customer churn and can a clear warning about the customer, and this can help the company to take some measures which will help not to lose the existing customer from the service.(Here we divided data into 80% - training,20%-testing).

By getting both the results, we will try to fix the y1 train data and y2 test data to the model fit to make the model learn from the historical data. In this, the epochs are used to make the model learn the same data repeated times. By using the CNN model, we visualized the data, and by that, we can know the model accuracy of the resulted data, and then we can have a prediction of the churn.(Fig.6,Fig.7)

Similarly, we used the other two techniques to know which will provide us more accurate results. In the Random Forest, we used the same dataset, and by applying the technique, we trained the model and tested it out to get the results in the confusion matrix, which will show us the obtained output, and we can notice the accuracy (fig.3). The result obtained from the XGBoost model is shown in (fig.4), where we can observe the accuracy obtained by using that technique.

IV. RESULT AND ANALYSIS

We performed several experiments on the proposed churn model using machine learning algorithms on the dataset. In Fig.2, we can observe the results obtained while performing the experiment using the Random Forest algorithm and can check the accuracy. Random Forest(RF) is a useful algorithm that suite for classification and can handle nonlinear data very efficiently.RF produced better results and better accuracy and performance compared to the other techniques. As we should need better accuracy to predict the customer churn, we prefer to use the technique, which results in better accuracy. Similarly, we can observe the results obtained when using the Logistic regression technique(Fig.4) .

```
              precision    recall  f1-score   support

           0       0.62      0.52      0.56       440
           1       0.85      0.89      0.87      1321

    accuracy                           0.80      1761
   macro avg       0.73      0.70      0.72      1761
weighted avg       0.79      0.80      0.79      1761
```

Fig 3.Confusion matrix of Random Forest.

```
              precision    recall  f1-score   support

           0       0.57      0.56      0.56       440
           1       0.85      0.86      0.86      1321

    accuracy                           0.79      1761
   macro avg       0.71      0.71      0.71      1761
weighted avg       0.78      0.79      0.78      1761
```

Fig 4.Confusion matrix of Logistic regression.

```
              precision    recall  f1-score   support

           0       0.58      0.50      0.54       440
           1       0.84      0.88      0.86      1321

    accuracy                           0.78      1761
   macro avg       0.71      0.69      0.70      1761
weighted avg       0.78      0.78      0.78      1761
```

Fig 5.Confusion matrix of XGBoost.

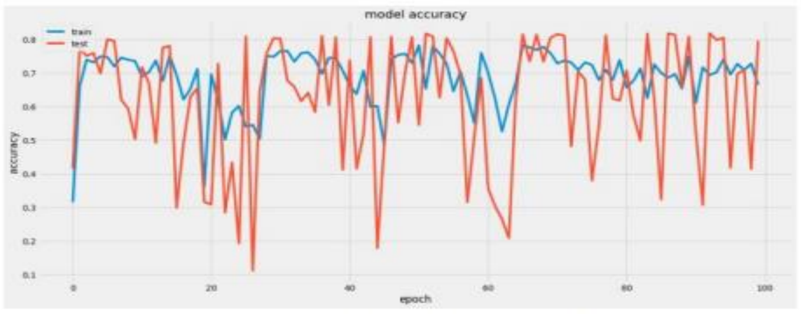


Fig.6.Visualizing CNN model val_acc and accuracy.

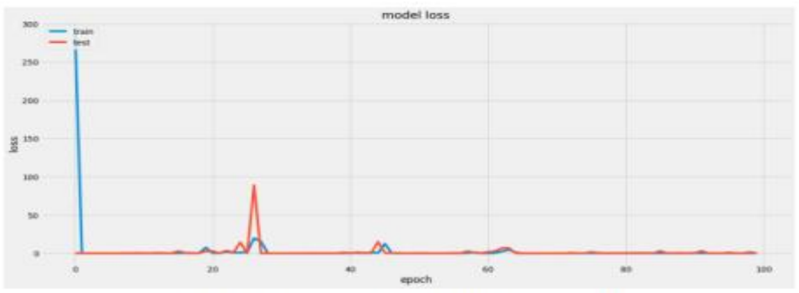


Fig.7.Visualizing CNN model val_loss and loss.

V. CONCLUSION

The importance of churn prediction will help many companies, mainly in telecom industries, to have a profitable income and achieve good revenue. Customer churn prediction is the major issue in the Telecom Industry, and due to this, companies are trying to keep the existing ones from leaving rather than acquiring a new customer. Three tree-based algorithms were chosen because of their applicability and diversity in this type of application. By using Random Forest, XGBoost, and Logistic regression, we will get more accuracy comparing other algorithms. Here we are using the dataset of some customers about their service plan and checking the values of them and have a precise prediction, which will help to identify the customers who are going to migrate to other company services. By this, the Telecom Company can have a clear view and can provide them some exiting offers to stay in that service. The obtained results show that our proposed churn model produced better results and performed better by using machine learning techniques.Random Forest produced better accuracy among the various methods.

In the coming days, we will further research on lazy learning approaches to have better customer churn prediction.To know the changing behavior of the customers, the study can be extended by using Artificial Intelligence techniques for trend analysis and customer prediction.

Code

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%.

In [1]:

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

sns.set(style = 'white')

# Input data files are available in the "../input/" directory.

import os
print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.
```

```
['WA_Fn-UseC_-Telco-Customer-Churn.csv']
```

Let us read the data file in the python notebook

In [2]:

```
telecom_cust = pd.read_csv('../input/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

In [3]:

```
telecom_cust.head()
```

Out[3]:

In [4]:

```
telecom_cust.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)
```

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

In [5]:

```
# Checking the data types of all the columns
telecom_cust.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.
telecom_cust.TotalCharges = pd.to_numeric(telecom_cust.TotalCharges, errors='coerc
e')
telecom_cust.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
```

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

In [7]:

```
#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()
```

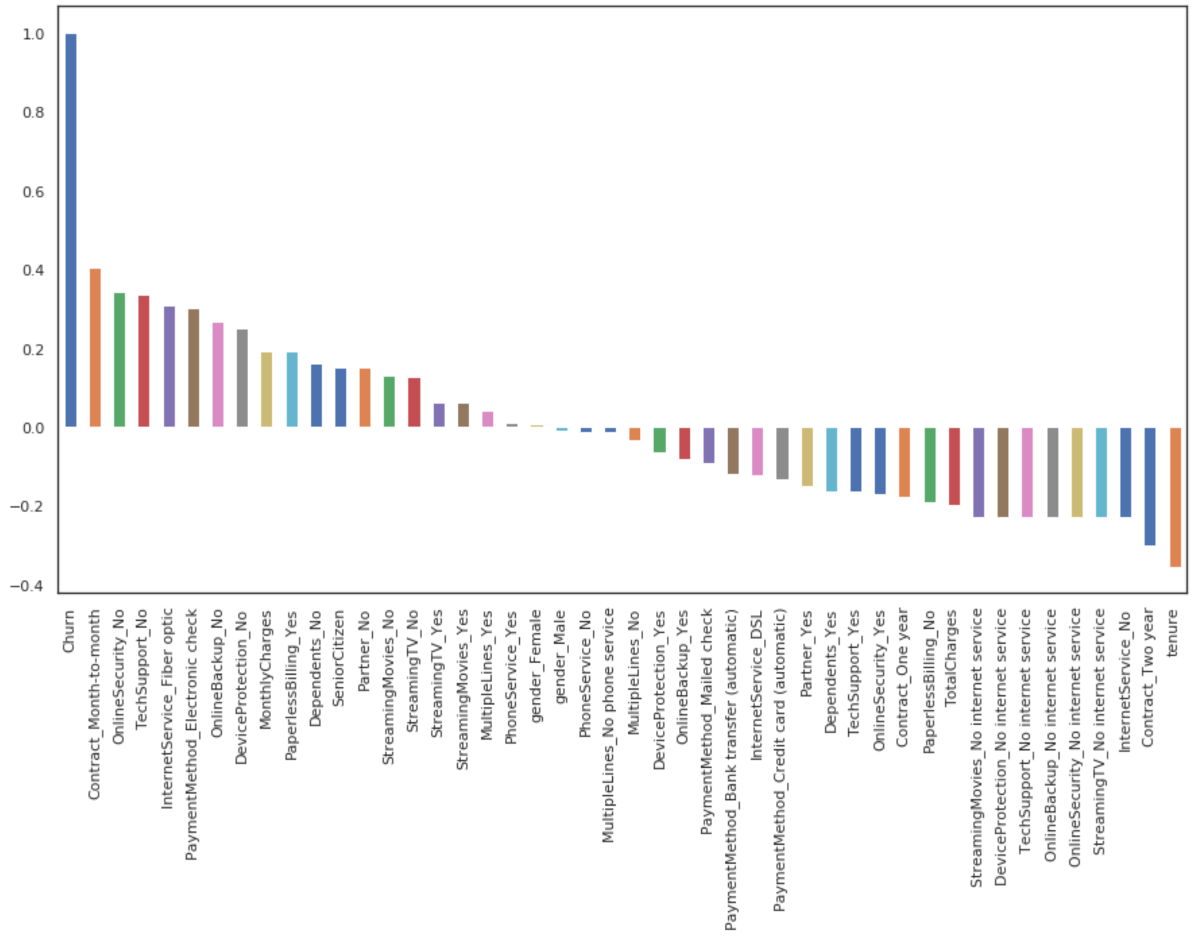
Out[7]:

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]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc65fe4d828>
```



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.

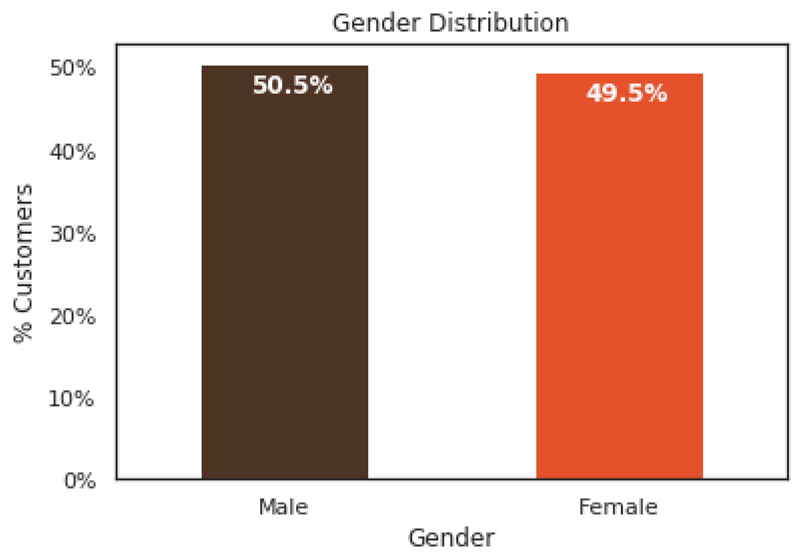
Data Exploration

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

```
                                                                                In [9]:
colors = ['#4D3425','#E4512B']
ax = (telecom_cust['gender'].value_counts()*100.0 /len(telecom_cust)).plot(kind='b
ar',
                                                                               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')
```



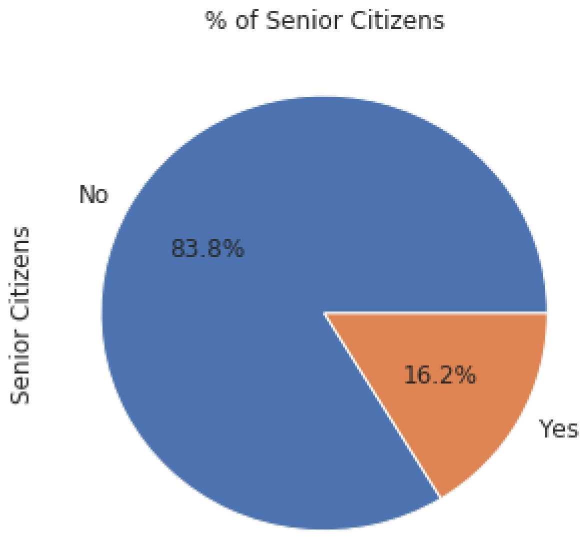
1. **% Senior Citizens** - There are only 16% of the customers who are senior citizens. Thus most of our customers in the data are younger people.

```
                                                                               In [10]:
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)
ax.set_title('% of Senior Citizens', fontsize = 12)
```

Out[10]:

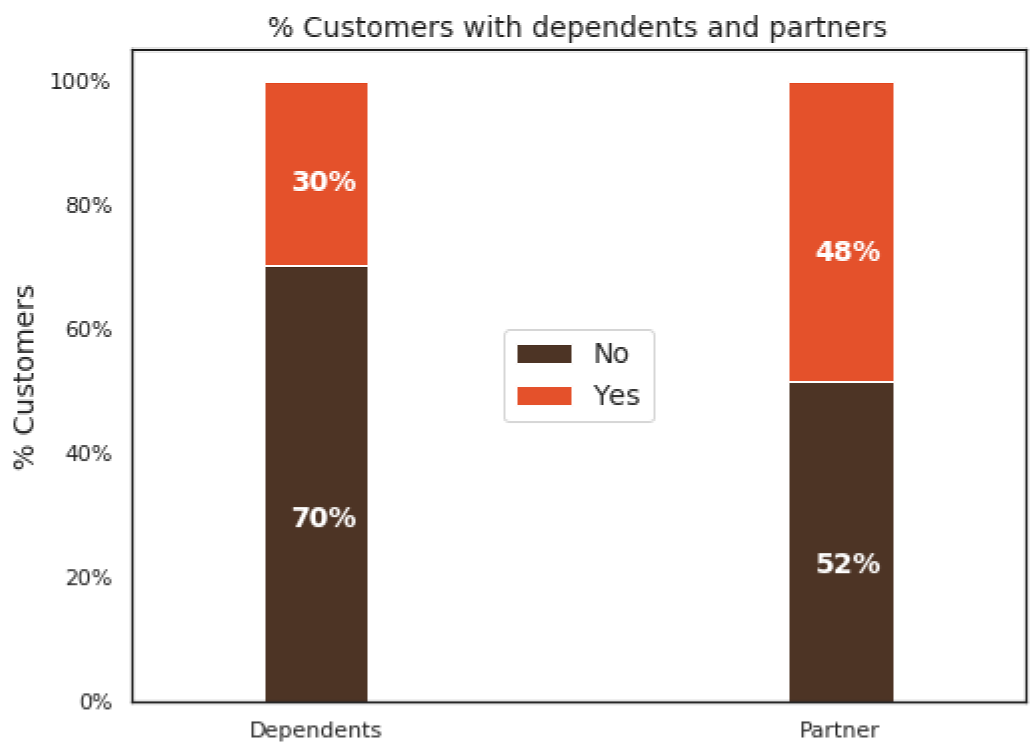
```
Text(0.5,1,'% of Senior Citizens')
```



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

In [11]:

```
df2 = pd.melt(telecom_cust, id_vars=['customerID'], value_vars=['Dependents','Part
ner'])
df3 = df2.groupby(['variable','value']).count().unstack()
df3 = df3*100/len(telecom_cust)
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*heigh
t),
                color = 'white',
               weight = 'bold',
               size = 14)
```

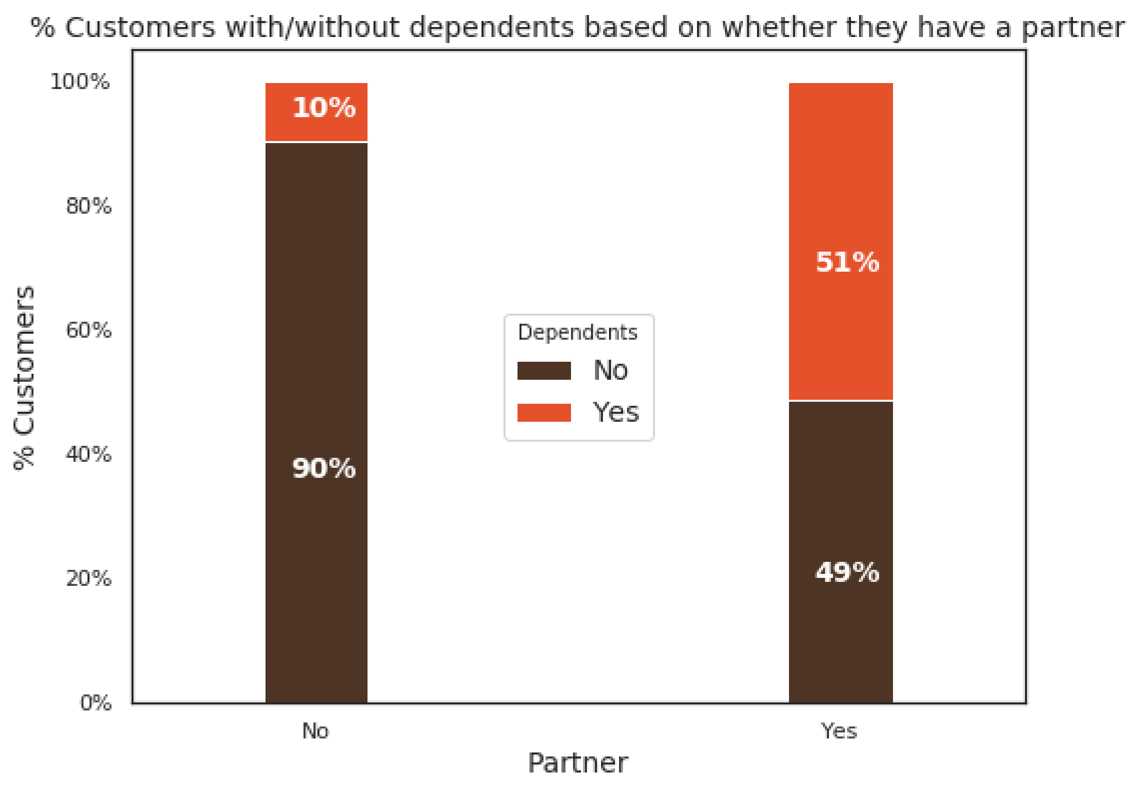


What would be interesting is to look at the % of customers, who have partners, also have dependents. We will explore this next.

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

In [12]:

```
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 par
tner',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*heigh
t),
                color = 'white',
               weight = 'bold',
               size = 14)
```



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

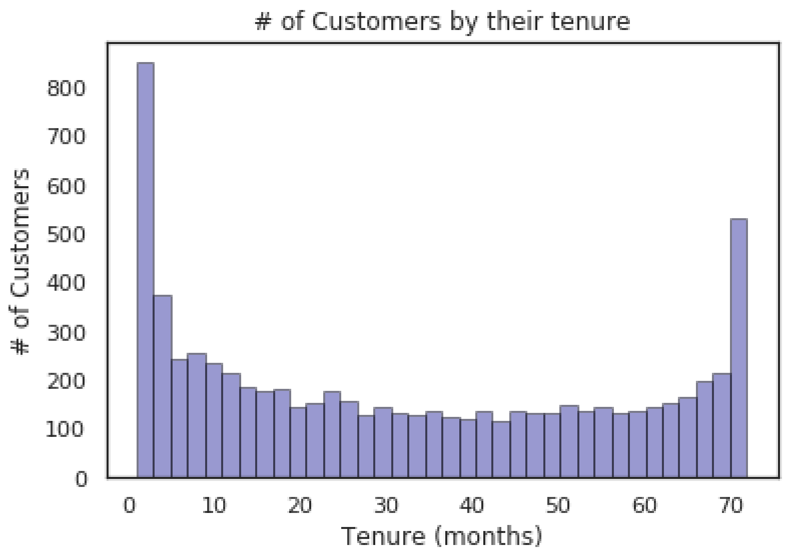
1. Tenure: After looking at the below histogram we can see that a lot of customers have been with the telecom company for just a month, while quite a many are there for about 72 months. This could be potentially because different customers have different contracts. Thus based on the contract they are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

In [13]:

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

Out[13]:

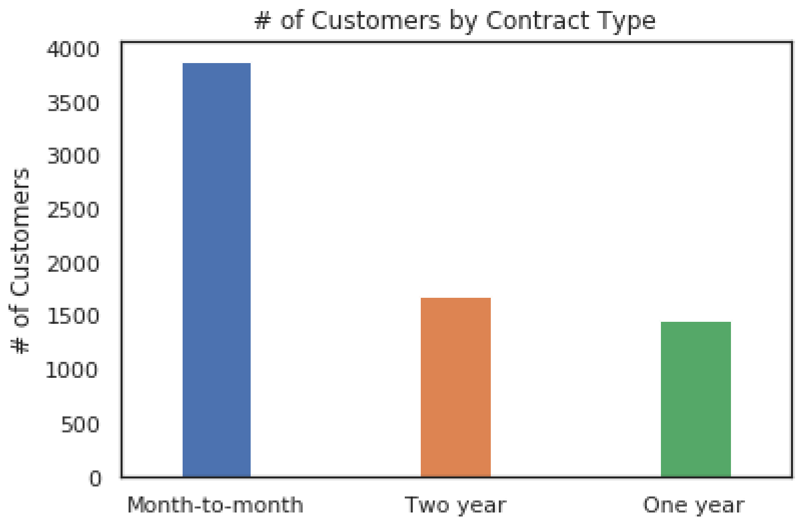
```
Text(0.5,1,'# of Customers by their tenure')
```



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

```
In [14]:
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[14]:
Text(0.5,1,'# of Customers by Contract Type')
```



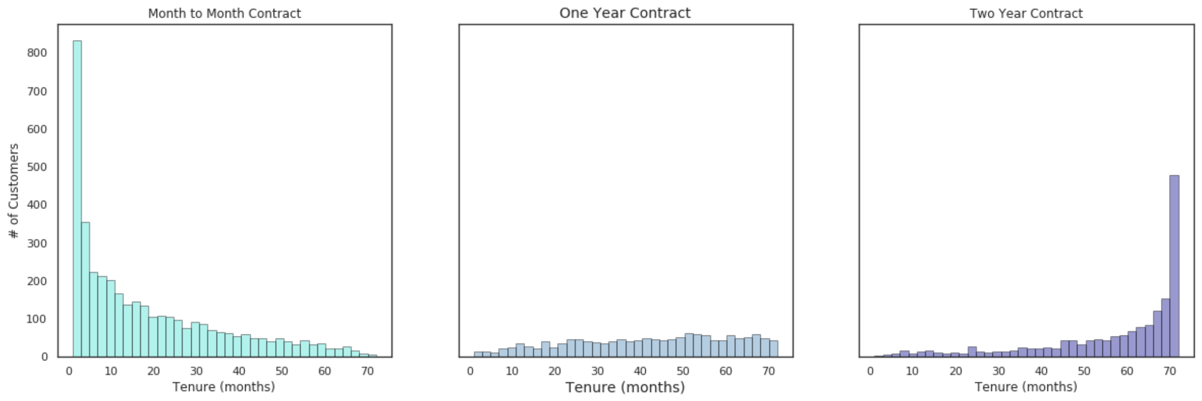
As we can see from this graph most of the customers are in the month to month contract. While there are equal number of customers in the 1 year and 2 year contracts.

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

```
In [15]:
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[15]:

```
Text(0.5,1,'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

In [16]:

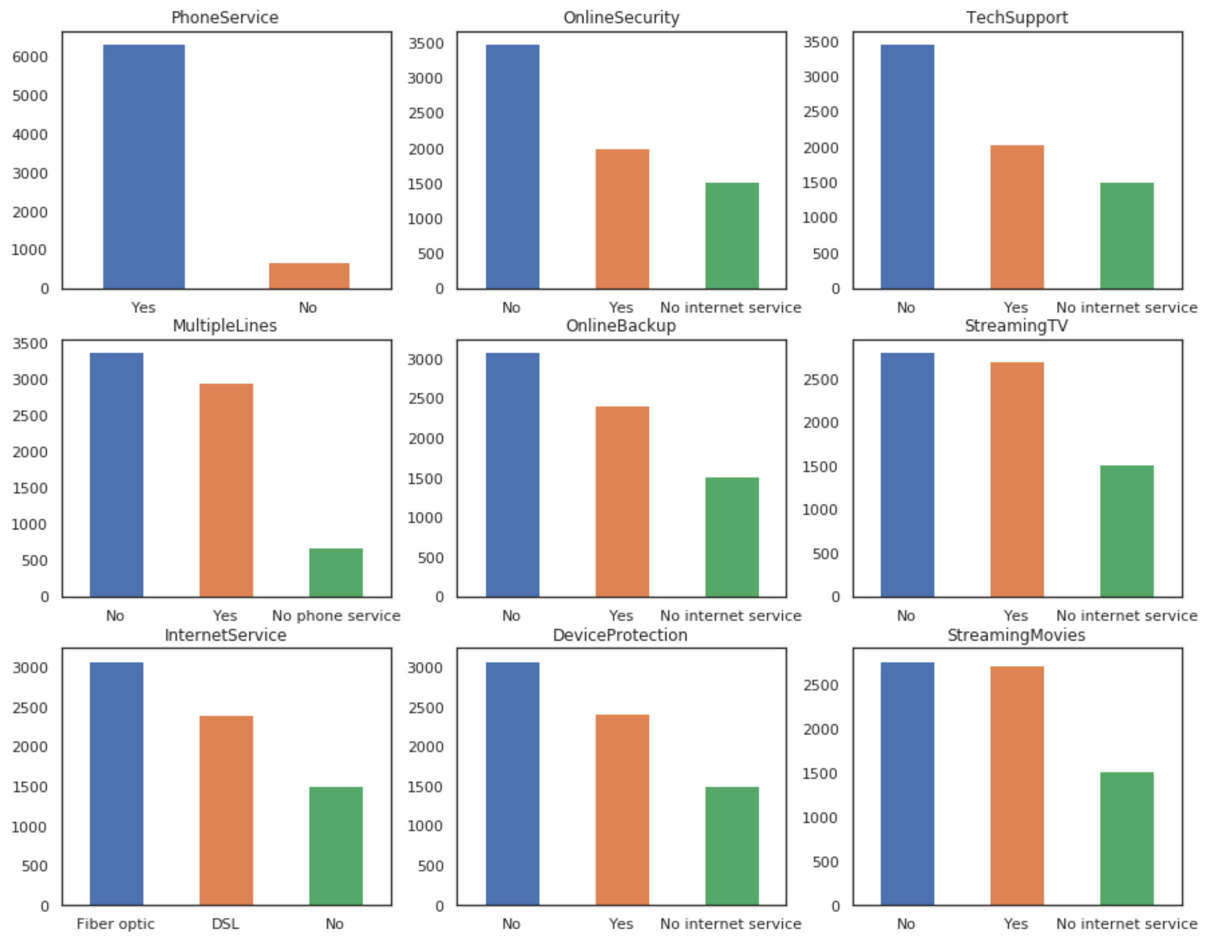
```
telecom_cust.columns.values
```

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','Streamin
gMovies']

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],ro
t = 0)

    elif i < 9:
        ax = telecom_cust[item].value_counts().plot(kind = 'bar',ax=axes[i-6,2],ro
t = 0)
    ax.set_title(item)
```



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

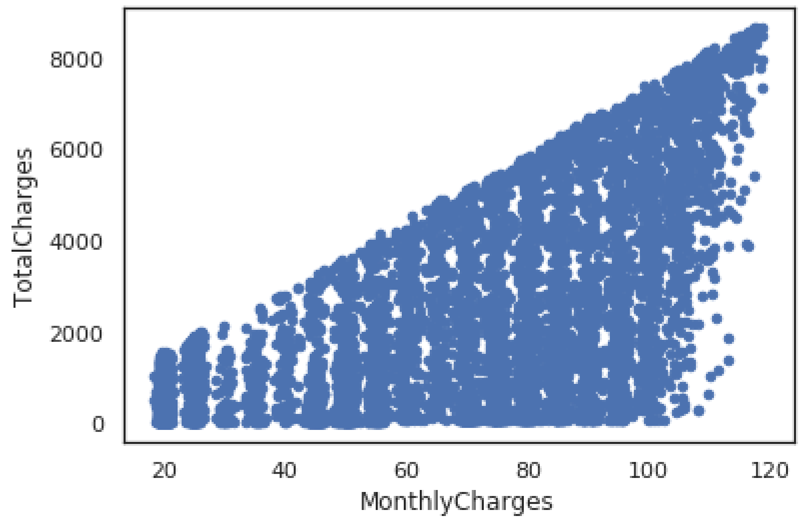
We will observe that the total charges increases as the monthly bill for a customer increases.

In [18]:

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

Out[18]:

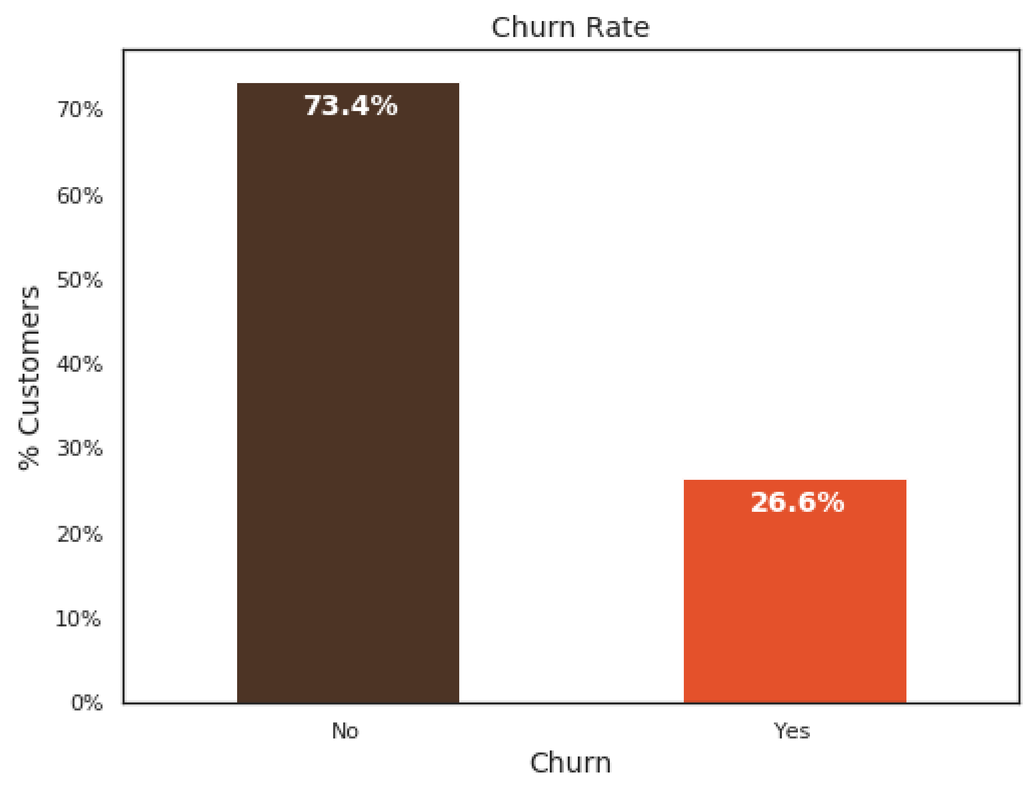
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6582a89e8>
```



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

1. Lets first look at the churn rate in our data

```
                                                                      In [19]:
colors = ['#4D3425','#E4512B']
ax = (telecom_cust['Churn'].value_counts()*100.0 /len(telecom_cust)).plot(kind='ba
r',
                                                                       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 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)
```



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 skeweness could lead to a lot of false negatives. We will see in the modelling section on how to avoid skewness in the data.

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

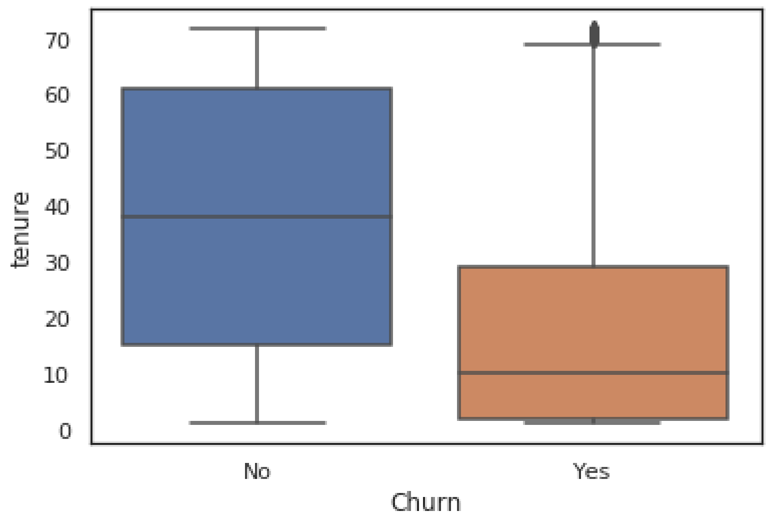
i.) Churn vs Tenure: As we can see form the below plot, the customers who do not churn, they tend to stay for a longer tenure with the telecom company.

In [20]:

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

Out[20]:

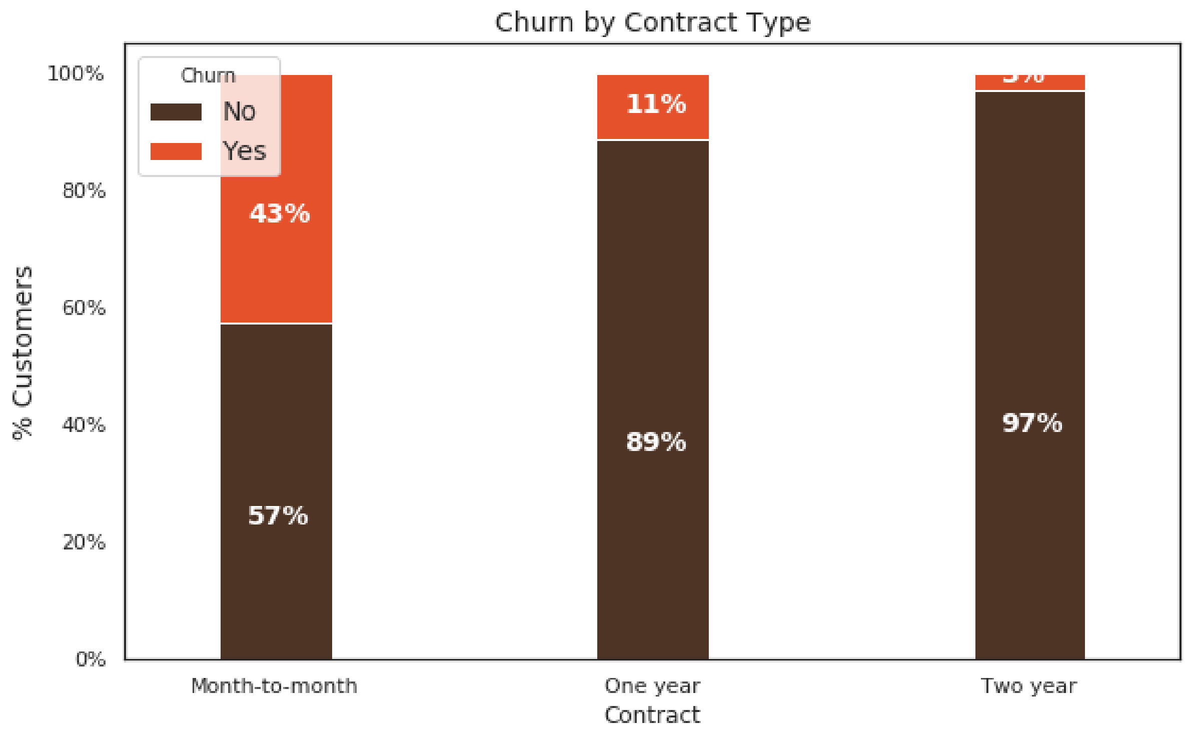
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc658246438>
```



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.

In [21]:

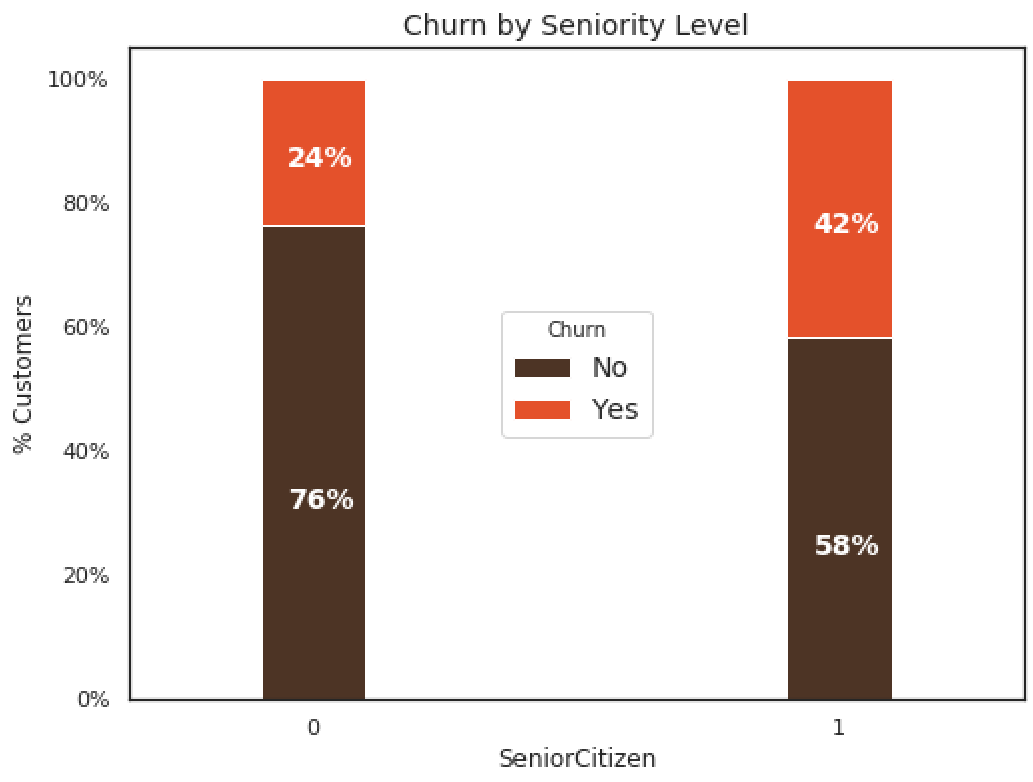
```
colors = ['#4D3425','#E4512B']
contract_churn = telecom_cust.groupby(['Contract','Churn']).size().unstack()

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*heigh
t),
                color = 'white',
               weight = 'bold',
               size = 14)
```



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

In [22]:

```
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*heigh
t),
                color = 'white',
               weight = 'bold',size =14)
```



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

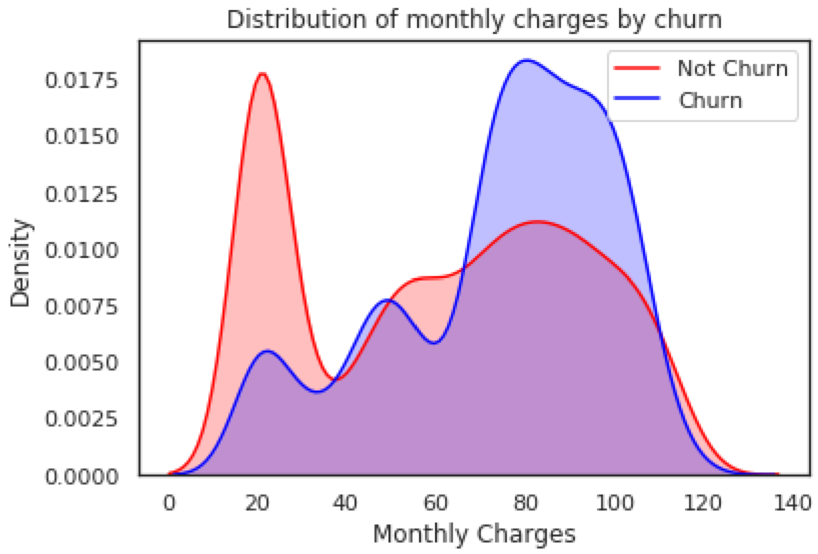
```
                                                                      In [23]:
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'No') ],
                color="Red", shade = True)
ax = sns.kdeplot(telecom_cust.MonthlyCharges[(telecom_cust["Churn"] == 'Yes') ],
                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')
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interprete
```

```
d as an array index, `arr[np.array(seq)]`, which will result either in an erro
r or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[23]:

```
Text(0.5,1,'Distribution of monthly charges by churn')
```



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

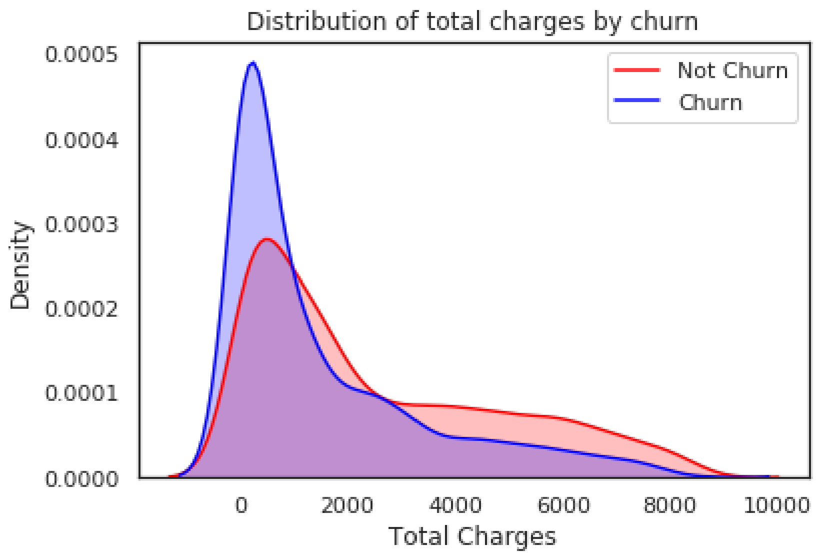
In [24]:

```python
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')
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
g: Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interprete
d as an array index, `arr[np.array(seq)]`, which will result either in an erro
r or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[24]:

```
Text(0.5,1,'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

In [25]:

```
# We will use the data frame where we had created dummy variables
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
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:323: Data
ConversionWarning: Data with input dtype uint8, int64, float64 were all conver
ted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

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.

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_st
ate=101)
```

In [27]:

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

In [28]:

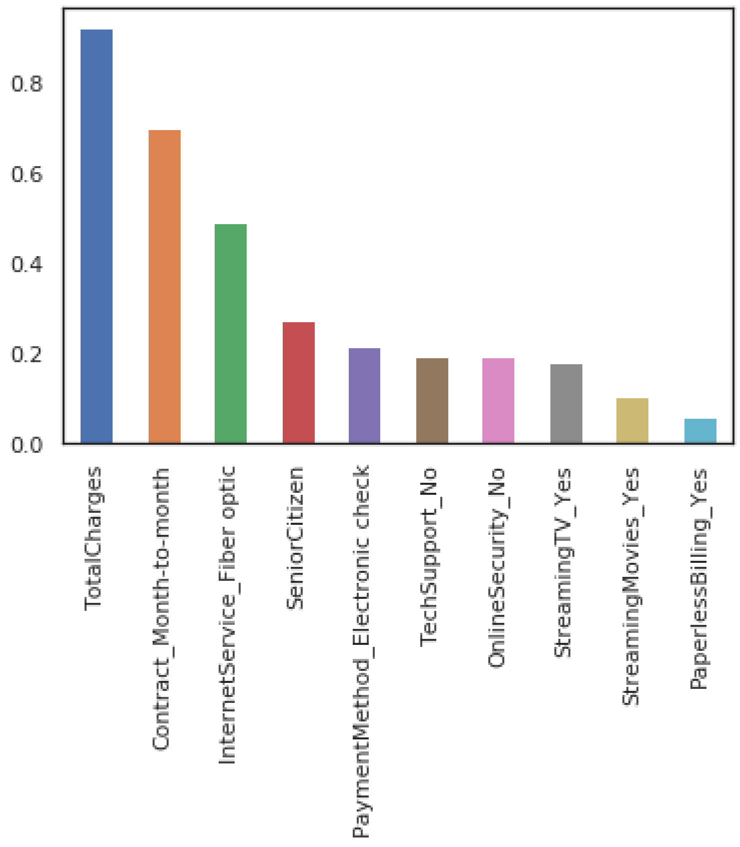
```
from sklearn import metrics
prediction_test = model.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
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'))
```

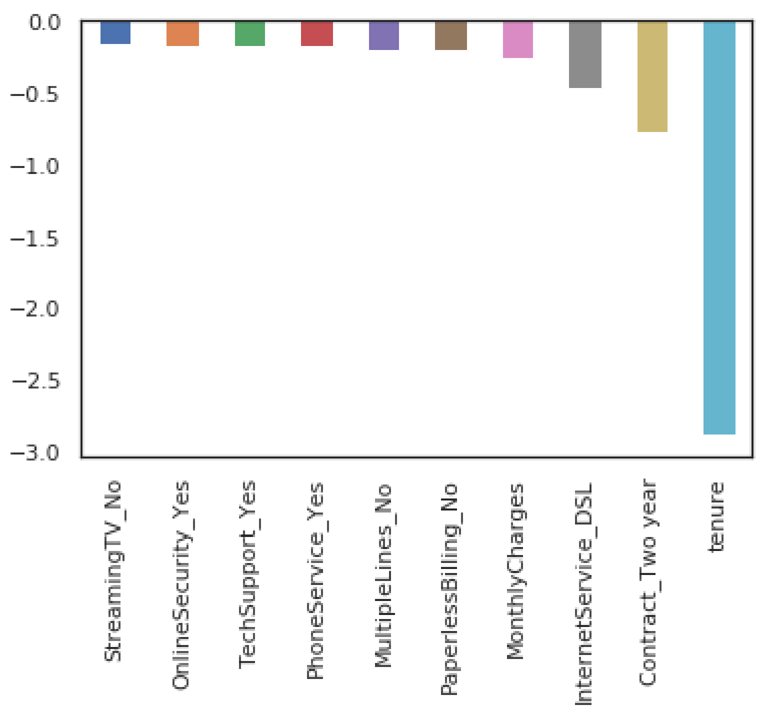
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



In [30]:

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

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



Observations

We can see that some variables have a negative relation to our predicted variable (Churn), while some have positive relation. Negative relation means that likeliness of churn decreases with that variable. Let us summarize some of the interesting features below:

- As we saw in our EDA, having a 2 month contract reduces chances of churn. 2 month contract along with tenure have the most negative relation with Churn as predicted by logistic regressions
- Having DSL internet service also reduces the proability of Churn
- Lastly, total charges, monthly contracts, fibre optic internet services and seniority can lead to higher churn rates. This is interesting because although fibre optic services are faster, customers are likely to churn because of it. I think we need to explore more to better understad why this is happening.

Any hypothesis on the above would be really helpful!

2. Random Forest

In [31]:

```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
ate=101)
model_rf = RandomForestClassifier(n_estimators=1000 , oob_score = True, n_jobs = -
1,
                                  random_state =50, max_features = "auto",
                                  max_leaf_nodes = 30)
model_rf.fit(X_train, y_train)

# Make predictions
prediction_test = model_rf.predict(X_test)
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8088130774697939
```

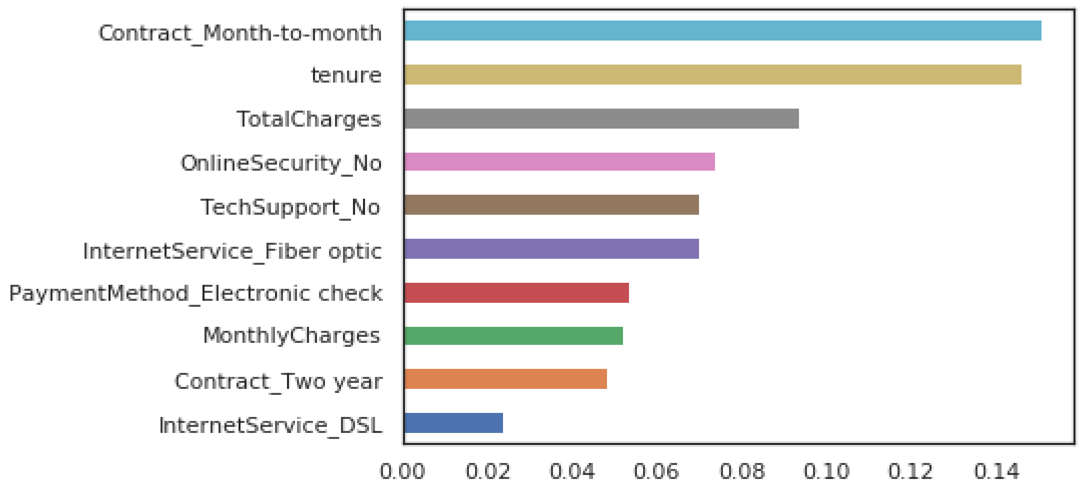
In [32]:

```
importances = model_rf.feature_importances_
```

```
weights = pd.Series(importances,
                    index=X.columns.values)
weights.sort_values()[-10:].plot(kind = 'barh')
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc64bd500f0>
```



Observations:

- From random forest algorithm, monthly contract, tenure and total charges are the most important predictor variables to predict churn.
- The results from random forest are very similar to that of the logistic regression and in line to what we had expected from our EDA

3. Support Vecor Machine (SVM)

In [33]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
ate=99)
```

In [34]:

```
from sklearn.svm import SVC

model.svm = SVC(kernel='linear')
model.svm.fit(X_train,y_train)
preds = model.svm.predict(X_test)
metrics.accuracy_score(y_test, preds)
```

Out[34]:

```
0.820184790334044
```

In [35]:

```
# Create the Confusion matrix
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,preds))
```

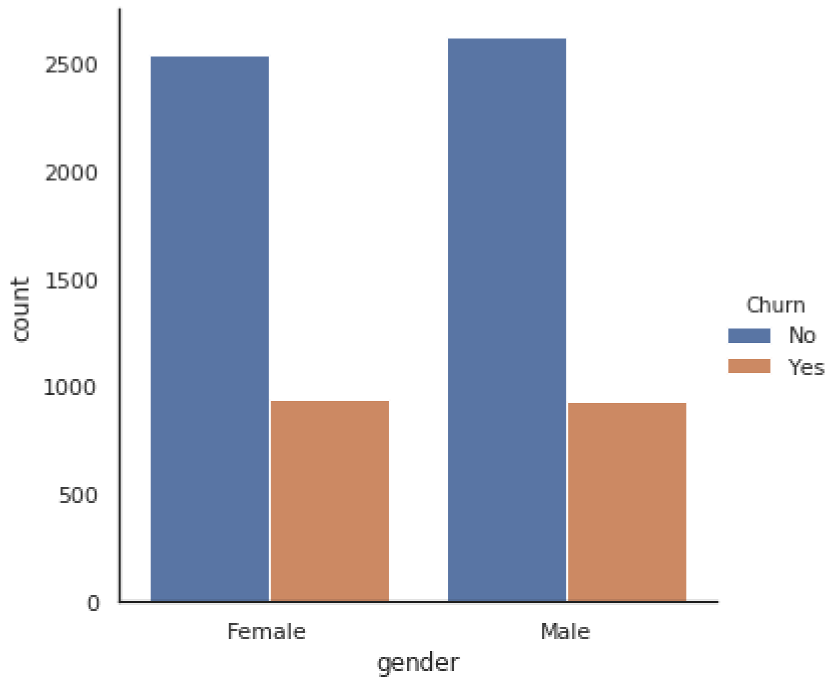
```
[[953  89]
 [164 201]]
```

Wth SVM I was able to increase the accuracy to upto 82%. However, we need to take a deeper look at the true positive and true negative rates, including the Area Under the Curve (AUC) for a better prediction. I will explore this soon. Stay Tuned!

In [36]:

```
ax1 = sns.catplot(x="gender", kind="count", hue="Churn", data=telecom_cust,
                  estimator=lambda x: sum(x==0)*100.0/len(x))
```

```
#ax1.yaxis.set_major_formatter(mtick.PercentFormatter())
```



4. ADA Boost

In [37]:

```
# AdaBoost Algorithm
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier()
# n_estimators = 50 (default value)
# base_estimator = DecisionTreeClassifier (default value)
model.fit(X_train,y_train)
preds = model.predict(X_test)
metrics.accuracy_score(y_test, preds)
```

Out[37]:

```
0.8159203980099502
```

5. XG Boost

In [38]:

```
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(X_train, y_train)
preds = model.predict(X_test)
metrics.accuracy_score(y_test, preds)
```

Out[38]:

```
0.8294243070362474
```

linkcode
Interestingly with XG Boost I was able to increase the accuracy on test data to almost 83%. Clearly, XG Boost is a winner among all other techniques. XG Boost is a slow learning model and is based on the concept of Boosting