**Customer Attrition**[**¶**](#gjdgxs)

Customer attrition, also known as customer churn, customer turnover, or customer defection, is the loss of clients or customers.

Telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Companies usually make a distinction between voluntary churn and involuntary churn. Voluntary churn occurs due to a decision by the customer to switch to another company or service provider, involuntary churn occurs due to circumstances such as a customer's relocation to a long-term care facility, death, or the relocation to a distant location. In most applications, involuntary reasons for churn are excluded from the analytical models. Analysts tend to concentrate on voluntary churn, because it typically occurs due to factors of the company-customer relationship which companies control, such as how billing interactions are handled or how after-sales help is provided.

predictive analytics use churn prediction models that predict customer churn by assessing their propensity of risk to churn. Since these models generate a small prioritized list of potential defectors, they are effective at focusing customer retention marketing programs on the subset of the customer base who are most vulnerable to churn.

* [1. Data](#30j0zll)
  + [1.1. Data overview](#1fob9te)
* [2. Data Manipulation](#3znysh7)
* [3. Exploratory Data Analysis](#2et92p0)
  + [3.1. Customer attrition in data](#tyjcwt)
  + [3.2. Varibles distribution in customer attrition](#3dy6vkm)
  + [3.3. Customer attrition in tenure groups](#1t3h5sf)
  + [3.4. Monthly Charges and Total Charges by Tenure and Churn group](#4d34og8)
  + [3.5. Average Charges by tenure groups](#2s8eyo1)
  + [3.6. Monthly charges,total charges and tenure in customer attrition](#17dp8vu)
  + [3.7. Variable Summary](#3rdcrjn)
  + [3.8. Correlation Matrix](#26in1rg)
  + [3.9. Visualising data with principal components](#lnxbz9)
  + [3.10. Binary variables distribution in customer attrition(Radar Chart)](#35nkun2)
* [4. Data preprocessing](#1ksv4uv)
* [5. Model Building](#44sinio)
  + [5.1. Baseline Model](#2jxsxqh)
  + [5.2. Synthetic Minority Oversampling TEchnique (SMOTE)](#z337ya)
  + [5.3. Recursive Feature Elimination](#3j2qqm3)
  + [5.4. Univariate Selection](#1y810tw)
  + [5.5. Decision Tree Visualization](#4i7ojhp)
  + [5.6. KNN Classifier](#2xcytpi)
  + [5.7. Vizualising a decision tree from random forest classifier](#1ci93xb)
  + [5.8. A random forest classifier.](#3whwml4)
  + [5.9. Gaussian Naive Bayes](#2bn6wsx)
  + [5.10. Support Vector Machine](#qsh70q)
  + [5.11. Tuning parameters for support vector machine](#3as4poj)
  + [5.12. LightGBMClassifier](#1pxezwc)
  + [5.13. XGBoost Classifier](#49x2ik5)
* [6. Model Performances](#2p2csry)
  + [6.1. model performance metrics](#147n2zr)
  + [6.2. Compare model metrics](#3o7alnk)
  + [6.3. Confusion matrices for models](#23ckvvd)
  + [6.4. ROC - Curves for models](#ihv636)
  + [6.5. Precision recall curves](#32hioqz)

In [1]:

*#Importing libraries*  
**import** **numpy** **as** **np** *# linear algebra*  
**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*  
*# Input data files are available in the "../input/" directory.*  
**import** **os**  
**import** **matplotlib.pyplot** **as** **plt***#visualization*  
**from** **PIL** **import** Image  
%**matplotlib** inline  
**import** **pandas** **as** **pd**  
**import** **seaborn** **as** **sns***#visualization*  
**import** **itertools**  
**import** **warnings**  
warnings.filterwarnings("ignore")  
**import** **io**  
**import** **plotly.offline** **as** **py***#visualization*  
py.init\_notebook\_mode(connected=**True**)*#visualization*  
**import** **plotly.graph\_objs** **as** **go***#visualization*  
**import** **plotly.tools** **as** **tls***#visualization*  
**import** **plotly.figure\_factory** **as** **ff***#visualization*  
start\_time = pd.datetime.now()

**1.Data**[**¶**](#1hmsyys)

In [2]:

telcom = pd.read\_csv("WA\_Fn-UseC\_-Telco-Customer-Churn.csv")  
*#first few rows*  
telcom.head()

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **customerID** | **gender** | **SeniorCitizen** | **Partner** | **Dependents** | **tenure** | **PhoneService** | **MultipleLines** | **InternetService** | **OnlineSecurity** | **...** | **DeviceProtection** | **TechSupport** | **StreamingTV** | **StreamingMovies** | **Contract** | **PaperlessBilling** | **PaymentMethod** | **MonthlyCharges** | **TotalCharges** | **Churn** |
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | DSL | No | ... | No | No | No | No | Month-to-month | Yes | Electronic check | 29.85 | 29.85 | No |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | DSL | Yes | ... | Yes | No | No | No | One year | No | Mailed check | 56.95 | 1889.5 | No |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | DSL | Yes | ... | No | No | No | No | Month-to-month | Yes | Mailed check | 53.85 | 108.15 | Yes |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | DSL | Yes | ... | Yes | Yes | No | No | One year | No | Bank transfer (automatic) | 42.30 | 1840.75 | No |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | Fiber optic | No | ... | No | No | No | No | Month-to-month | Yes | Electronic check | 70.70 | 151.65 | Yes |

5 rows × 21 columns

**1.1. Data overview**[**¶**](#41mghml)

In [3]:

print ("Rows : " ,telcom.shape[0])  
print ("Columns : " ,telcom.shape[1])  
print ("**\n**Features : **\n**" ,telcom.columns.tolist())  
print ("**\n**Missing values : ", telcom.isnull().sum().values.sum())  
print ("**\n**Unique values : **\n**",telcom.nunique())

Rows : 7043  
Columns : 21  
  
Features :   
 ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']  
  
Missing values : 0  
  
Unique values :   
 customerID 7043  
gender 2  
SeniorCitizen 2  
Partner 2  
Dependents 2  
tenure 73  
PhoneService 2  
MultipleLines 3  
InternetService 3  
OnlineSecurity 3  
OnlineBackup 3  
DeviceProtection 3  
TechSupport 3  
StreamingTV 3  
StreamingMovies 3  
Contract 3  
PaperlessBilling 2  
PaymentMethod 4  
MonthlyCharges 1585  
TotalCharges 6531  
Churn 2  
dtype: int64

**2. Data Manipulation**[**¶**](#2grqrue)

In [4]:

*#Data Manipulation*  
  
*#Replacing spaces with null values in total charges column*  
telcom['TotalCharges'] = telcom["TotalCharges"].replace(" ",np.nan)  
  
*#Dropping null values from total charges column which contain .15% missing data*   
telcom = telcom[telcom["TotalCharges"].notnull()]  
telcom = telcom.reset\_index()[telcom.columns]  
  
*#convert to float type*  
telcom["TotalCharges"] = telcom["TotalCharges"].astype(float)  
  
*#replace 'No internet service' to No for the following columns*  
replace\_cols = [ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',  
 'TechSupport','StreamingTV', 'StreamingMovies']  
**for** i **in** replace\_cols :   
 telcom[i] = telcom[i].replace({'No internet service' : 'No'})  
   
*#replace values*  
telcom["SeniorCitizen"] = telcom["SeniorCitizen"].replace({1:"Yes",0:"No"})  
  
*#Tenure to categorical column*  
**def** tenure\_lab(telcom) :  
   
 **if** telcom["tenure"] <= 12 :  
 **return** "Tenure\_0-12"  
 **elif** (telcom["tenure"] > 12) & (telcom["tenure"] <= 24 ):  
 **return** "Tenure\_12-24"  
 **elif** (telcom["tenure"] > 24) & (telcom["tenure"] <= 48) :  
 **return** "Tenure\_24-48"  
 **elif** (telcom["tenure"] > 48) & (telcom["tenure"] <= 60) :  
 **return** "Tenure\_48-60"  
 **elif** telcom["tenure"] > 60 :  
 **return** "Tenure\_gt\_60"  
telcom["tenure\_group"] = telcom.apply(**lambda** telcom:tenure\_lab(telcom),  
 axis = 1)  
  
*#Separating churn and non churn customers*  
churn = telcom[telcom["Churn"] == "Yes"]  
not\_churn = telcom[telcom["Churn"] == "No"]  
  
*#Separating catagorical and numerical columns*  
Id\_col = ['customerID']  
target\_col = ["Churn"]  
cat\_cols = telcom.nunique()[telcom.nunique() < 6].keys().tolist()  
cat\_cols = [x **for** x **in** cat\_cols **if** x **not** **in** target\_col]  
num\_cols = [x **for** x **in** telcom.columns **if** x **not** **in** cat\_cols + target\_col + Id\_col]

In [5]:

print(cat\_cols)

['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'tenure\_group']

In [6]:

print(num\_cols)

['tenure', 'MonthlyCharges', 'TotalCharges']

In [7]:

telcom.shape

Out[7]:

(7032, 22)

In [8]:

churn.shape

Out[8]:

(1869, 22)

In [9]:

not\_churn.shape

Out[9]:

(5163, 22)

**3. Exploratory Data Analysis**[**¶**](#vx1227)

**3.1. Customer attrition in data**[**¶**](#3fwokq0)

In [10]:

*#labels*  
lab = telcom["Churn"].value\_counts().keys().tolist()  
*#values*  
val = telcom["Churn"].value\_counts().values.tolist()  
  
trace = go.Pie(labels = lab ,  
 values = val ,  
 marker = dict(colors = [ 'royalblue' ,'lime'],  
 line = dict(color = "white",  
 width = 1.3)  
 ),  
 rotation = 90,  
 hoverinfo = "label+value+text",  
 hole = .5  
 )  
layout = go.Layout(dict(title = "Customer attrition in data",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 )  
 )  
  
data = [trace]  
fig = go.Figure(data = data,layout = layout)  
py.iplot(fig)

**3.2. Varibles distribution in customer attrition**[**¶**](#1v1yuxt)

In [11]:

*#function for pie plot for customer attrition types*  
**def** plot\_pie(column) :  
   
 trace1 = go.Pie(values = churn[column].value\_counts().values.tolist(),  
 labels = churn[column].value\_counts().keys().tolist(),  
 hoverinfo = "label+percent+name",  
 domain = dict(x = [0,.48]),  
 name = "Churn Customers",  
 marker = dict(line = dict(width = 2,  
 color = "rgb(243,243,243)")  
 ),  
 hole = .6  
 )  
 trace2 = go.Pie(values = not\_churn[column].value\_counts().values.tolist(),  
 labels = not\_churn[column].value\_counts().keys().tolist(),  
 hoverinfo = "label+percent+name",  
 marker = dict(line = dict(width = 2,  
 color = "rgb(243,243,243)")  
 ),  
 domain = dict(x = [.52,1]),  
 hole = .6,  
 name = "Non churn customers"   
 )  
  
  
 layout = go.Layout(dict(title = column + " distribution in customer attrition ",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 annotations = [dict(text = "churn customers",  
 font = dict(size = 13),  
 showarrow = **False**,  
 x = .15, y = .5),  
 dict(text = "Non churn customers",  
 font = dict(size = 13),  
 showarrow = **False**,  
 x = .88,y = .5  
 )  
 ]  
 )  
 )  
 data = [trace1,trace2]  
 fig = go.Figure(data = data,layout = layout)  
 py.iplot(fig)  
  
  
*#function for histogram for customer attrition types*  
**def** histogram(column) :  
 trace1 = go.Histogram(x = churn[column],  
 histnorm= "percent",  
 name = "Churn Customers",  
 marker = dict(line = dict(width = .5,  
 color = "black"  
 )  
 ),  
 opacity = .9   
 )   
   
 trace2 = go.Histogram(x = not\_churn[column],  
 histnorm = "percent",  
 name = "Non churn customers",  
 marker = dict(line = dict(width = .5,  
 color = "black"  
 )  
 ),  
 opacity = .9  
 )  
   
 data = [trace1,trace2]  
 layout = go.Layout(dict(title =column + " distribution in customer attrition ",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = column,  
 zerolinewidth=1,  
 ticklen=5,  
 gridwidth=2  
 ),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "percent",  
 zerolinewidth=1,  
 ticklen=5,  
 gridwidth=2  
 ),  
 )  
 )  
 fig = go.Figure(data=data,layout=layout)  
   
 py.iplot(fig)  
   
*#function for scatter plot matrix for numerical columns in data*  
**def** scatter\_matrix(df) :  
   
 df = df.sort\_values(by = "Churn" ,ascending = **True**)  
 classes = df["Churn"].unique().tolist()  
 classes  
   
 class\_code = {classes[k] : k **for** k **in** range(2)}  
 class\_code  
  
 color\_vals = [class\_code[cl] **for** cl **in** df["Churn"]]  
 color\_vals  
  
 pl\_colorscale = "Portland"  
  
 pl\_colorscale  
  
 text = [df.loc[k,"Churn"] **for** k **in** range(len(df))]  
 text  
  
 trace = go.Splom(dimensions = [dict(label = "tenure",  
 values = df["tenure"]),  
 dict(label = 'MonthlyCharges',  
 values = df['MonthlyCharges']),  
 dict(label = 'TotalCharges',  
 values = df['TotalCharges'])],  
 text = text,  
 marker = dict(color = color\_vals,  
 colorscale = pl\_colorscale,  
 size = 3,  
 showscale = **False**,  
 line = dict(width = .1,  
 color='rgb(230,230,230)'  
 )  
 )  
 )  
 axis = dict(showline = **True**,  
 zeroline = **False**,  
 gridcolor = "#fff",  
 ticklen = 4  
 )  
   
 layout = go.Layout(dict(title =   
 "Scatter plot matrix for Numerical columns for customer attrition",  
 autosize = **False**,  
 height = 800,  
 width = 800,  
 dragmode = "select",  
 hovermode = "closest",  
 plot\_bgcolor = 'rgba(240,240,240, 0.95)',  
 xaxis1 = dict(axis),  
 yaxis1 = dict(axis),  
 xaxis2 = dict(axis),  
 yaxis2 = dict(axis),  
 xaxis3 = dict(axis),  
 yaxis3 = dict(axis),  
 )  
 )  
 data = [trace]  
 fig = go.Figure(data = data,layout = layout )  
 py.iplot(fig)  
  
*#for all categorical columns plot pie*  
**for** i **in** cat\_cols :  
 plot\_pie(i)  
  
*#for all categorical columns plot histogram*   
**for** i **in** num\_cols :  
 histogram(i)  
  
*#scatter plot matrix*  
scatter\_matrix(telcom)

**3.3. Customer attrition in tenure groups**[**¶**](#4f1mdlm)

In [12]:

*#cusomer attrition in tenure groups*  
tg\_ch = churn["tenure\_group"].value\_counts().reset\_index()  
tg\_ch.columns = ["tenure\_group","count"]  
tg\_nch = not\_churn["tenure\_group"].value\_counts().reset\_index()  
tg\_nch.columns = ["tenure\_group","count"]  
  
*#bar - churn*  
trace1 = go.Bar(x = tg\_ch["tenure\_group"] , y = tg\_ch["count"],  
 name = "Churn Customers",  
 marker = dict(line = dict(width = .5,color = "black")),  
 opacity = .9)  
  
*#bar - not churn*  
trace2 = go.Bar(x = tg\_nch["tenure\_group"] , y = tg\_nch["count"],  
 name = "Non Churn Customers",  
 marker = dict(line = dict(width = .5,color = "black")),  
 opacity = .9)  
  
layout = go.Layout(dict(title = "Customer attrition in tenure groups",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "tenure group",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "count",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 )  
 )  
data = [trace1,trace2]  
fig = go.Figure(data=data,layout=layout)  
py.iplot(fig)

**3.4. Monthly Charges and Total Charges by Tenure and Churn groups**[**¶**](#2u6wntf)

In [13]:

telcom[['MonthlyCharges', 'TotalCharges','tenure',"tenure\_group"]]  
  
*#scatter plot monthly charges & total charges by tenure group*  
  
**def** plot\_tenure\_scatter(tenure\_group,color) :  
 tracer = go.Scatter(x = telcom[telcom["tenure\_group"] == tenure\_group]["MonthlyCharges"],  
 y = telcom[telcom["tenure\_group"] == tenure\_group]["TotalCharges"],  
 mode = "markers",marker = dict(line = dict(color = "black",  
 width = .2),  
 size = 4 , color = color,  
 symbol = "diamond-dot",  
 ),  
 name = tenure\_group,  
 opacity = .9  
 )  
 **return** tracer  
  
*#scatter plot monthly charges & total charges by churn group*  
**def** plot\_churncharges\_scatter(churn,color) :  
 tracer = go.Scatter(x = telcom[telcom["Churn"] == churn]["MonthlyCharges"],  
 y = telcom[telcom["Churn"] == churn]["TotalCharges"],  
 mode = "markers",marker = dict(line = dict(color = "black",  
 width = .2),  
 size = 4 , color = color,  
 symbol = "diamond-dot",  
 ),  
 name = "Churn - " + churn,  
 opacity = .9  
 )  
 **return** tracer  
  
trace1 = plot\_tenure\_scatter("Tenure\_0-12","#FF3300")  
trace2 = plot\_tenure\_scatter("Tenure\_12-24","#6666FF")  
trace3 = plot\_tenure\_scatter("Tenure\_24-48","#99FF00")  
trace4 = plot\_tenure\_scatter("Tenure\_48-60","#996600")  
trace5 = plot\_tenure\_scatter("Tenure\_gt\_60","grey")  
trace6 = plot\_churncharges\_scatter("Yes","red")  
trace7 = plot\_churncharges\_scatter("No","blue")  
  
data1 = [trace1,trace2,trace3,trace4,trace5]   
data2 = [trace7,trace6]  
  
*#layout*  
**def** layout\_title(title) :  
 layout = go.Layout(dict(title = title,  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "Monthly charges",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "Total Charges",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 height = 600  
 )  
 )  
 **return** layout  
  
layout1 = layout\_title("Monthly Charges & Total Charges by Tenure group")  
layout2 = layout\_title("Monthly Charges & Total Charges by Churn group")  
fig1 = go.Figure(data = data1,layout = layout1)  
fig2 = go.Figure(data = data2,layout = layout2)  
py.iplot(fig1)  
py.iplot(fig2)

**3.5. Average Charges by tenure groups**[**¶**](#19c6y18)

In [14]:

avg\_tgc = telcom.groupby(["tenure\_group","Churn"])[["MonthlyCharges",  
 "TotalCharges"]].mean().reset\_index()  
  
*#function for tracing*   
**def** mean\_charges(column,aggregate) :  
 tracer = go.Bar(x = avg\_tgc[avg\_tgc["Churn"] == aggregate]["tenure\_group"],  
 y = avg\_tgc[avg\_tgc["Churn"] == aggregate][column],  
 name = aggregate,marker = dict(line = dict(width = 1)),  
 text = "Churn"  
 )  
 **return** tracer  
  
*#function for layout*  
**def** layout\_plot(title,xaxis\_lab,yaxis\_lab) :  
 layout = go.Layout(dict(title = title,  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',title = xaxis\_lab,  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',title = yaxis\_lab,  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 )  
 )  
 **return** layout  
   
  
*#plot1 - mean monthly charges by tenure groups*  
trace1 = mean\_charges("MonthlyCharges","Yes")  
trace2 = mean\_charges("MonthlyCharges","No")  
layout1 = layout\_plot("Average Monthly Charges by Tenure groups",  
 "Tenure group","Monthly Charges")  
data1 = [trace1,trace2]  
fig1 = go.Figure(data=data1,layout=layout1)  
  
*#plot2 - mean total charges by tenure groups*  
trace3 = mean\_charges("TotalCharges","Yes")  
trace4 = mean\_charges("TotalCharges","No")  
layout2 = layout\_plot("Average Total Charges by Tenure groups",  
 "Tenure group","Total Charges")  
data2 = [trace3,trace4]  
fig2 = go.Figure(data=data2,layout=layout2)  
  
py.iplot(fig1)  
py.iplot(fig2)

**3.6. Monthly charges,total charges and tenure in customer attrition**[**¶**](#3tbugp1)

In [15]:

*##copy data*  
tel\_df = telcom.copy()  
*#Drop tenure column*  
telcom = telcom.drop(columns = "tenure\_group",axis = 1)  
  
trace1 = go.Scatter3d(x = churn["MonthlyCharges"],  
 y = churn["TotalCharges"],  
 z = churn["tenure"],  
 mode = "markers",  
 name = "Churn customers",  
 text = "Id : " + churn["customerID"],  
 marker = dict(size = 1,color = "red")  
 )  
trace2 = go.Scatter3d(x = not\_churn["MonthlyCharges"],  
 y = not\_churn["TotalCharges"],  
 z = not\_churn["tenure"],  
 name = "Non churn customers",  
 text = "Id : " + not\_churn["customerID"],  
 mode = "markers",  
 marker = dict(size = 1,color= "green")  
 )  
  
  
  
layout = go.Layout(dict(title = "Monthly charges,total charges & tenure in customer attrition",  
 scene = dict(camera = dict(up=dict(x= 0 , y=0, z=0),  
 center=dict(x=0, y=0, z=0),  
 eye=dict(x=1.25, y=1.25, z=1.25)),  
 xaxis = dict(title = "monthly charges",  
 gridcolor='rgb(255, 255, 255)',  
 zerolinecolor='rgb(255, 255, 255)',  
 showbackground=**True**,  
 backgroundcolor='rgb(230, 230,230)'),  
 yaxis = dict(title = "total charges",  
 gridcolor='rgb(255, 255, 255)',  
 zerolinecolor='rgb(255, 255, 255)',  
 showbackground=**True**,  
 backgroundcolor='rgb(230, 230,230)'  
 ),  
 zaxis = dict(title = "tenure",  
 gridcolor='rgb(255, 255, 255)',  
 zerolinecolor='rgb(255, 255, 255)',  
 showbackground=**True**,  
 backgroundcolor='rgb(230, 230,230)'  
 )  
 ),  
 height = 700,  
 )  
 )  
   
  
data = [trace1,trace2]  
fig = go.Figure(data = data,layout = layout)  
py.iplot(fig)

**4. Data preprocessing**[**¶**](#28h4qwu)

In [16]:

**from** **sklearn.preprocessing** **import** LabelEncoder  
**from** **sklearn.preprocessing** **import** StandardScaler  
  
*#customer id col*  
Id\_col = ['customerID']  
*#Target columns*  
target\_col = ["Churn"]  
*#categorical columns*  
cat\_cols = telcom.nunique()[telcom.nunique() < 6].keys().tolist()  
cat\_cols = [x **for** x **in** cat\_cols **if** x **not** **in** target\_col]  
*#numerical columns*  
num\_cols = [x **for** x **in** telcom.columns **if** x **not** **in** cat\_cols + target\_col + Id\_col]  
*#Binary columns with 2 values*  
bin\_cols = telcom.nunique()[telcom.nunique() == 2].keys().tolist()  
*#Columns more than 2 values*  
multi\_cols = [i **for** i **in** cat\_cols **if** i **not** **in** bin\_cols]  
  
*#Label encoding Binary columns*  
le = LabelEncoder()  
**for** i **in** bin\_cols :  
 telcom[i] = le.fit\_transform(telcom[i])  
   
*#Duplicating columns for multi value columns*  
telcom = pd.get\_dummies(data = telcom,columns = multi\_cols )  
  
*#Scaling Numerical columns*  
std = StandardScaler()  
scaled = std.fit\_transform(telcom[num\_cols])  
scaled = pd.DataFrame(scaled,columns=num\_cols)  
  
*#dropping original values merging scaled values for numerical columns*  
df\_telcom\_og = telcom.copy()  
telcom = telcom.drop(columns = num\_cols,axis = 1)  
telcom = telcom.merge(scaled,left\_index=**True**,right\_index=**True**,how = "left")

In [17]:

telcom.shape

Out[17]:

(7032, 30)

**3.7. Variable Summary**[**¶**](#nmf14n)

In [18]:

summary = (df\_telcom\_og[[i **for** i **in** df\_telcom\_og.columns **if** i **not** **in** Id\_col]].  
 describe().transpose().reset\_index())  
  
summary = summary.rename(columns = {"index" : "feature"})  
summary = np.around(summary,3)  
  
val\_lst = [summary['feature'], summary['count'],  
 summary['mean'],summary['std'],  
 summary['min'], summary['25%'],  
 summary['50%'], summary['75%'], summary['max']]  
  
trace = go.Table(header = dict(values = summary.columns.tolist(),  
 line = dict(color = ['#506784']),  
 fill = dict(color = ['#119DFF']),  
 ),  
 cells = dict(values = val\_lst,  
 line = dict(color = ['#506784']),  
 fill = dict(color = ["lightgrey",'#F5F8FF'])  
 ),  
 columnwidth = [200,60,100,100,60,60,80,80,80])  
layout = go.Layout(dict(title = "Variable Summary"))  
figure = go.Figure(data=[trace],layout=layout)  
py.iplot(figure)

**3.8. Correlation Matrix**[**¶**](#37m2jsg)

In [19]:

*#correlation*  
correlation = telcom.corr()  
*#tick labels*  
matrix\_cols = correlation.columns.tolist()  
*#convert to array*  
corr\_array = np.array(correlation)  
  
*#Plotting*  
trace = go.Heatmap(z = corr\_array,  
 x = matrix\_cols,  
 y = matrix\_cols,  
 colorscale = "Viridis",  
 colorbar = dict(title = "Pearson Correlation coefficient",  
 titleside = "right"  
 ) ,  
 )  
  
layout = go.Layout(dict(title = "Correlation Matrix for variables",  
 autosize = **False**,  
 height = 720,  
 width = 800,  
 margin = dict(r = 0 ,l = 210,  
 t = 25,b = 210,  
 ),  
 yaxis = dict(tickfont = dict(size = 9)),  
 xaxis = dict(tickfont = dict(size = 9))  
 )  
 )  
  
data = [trace]  
fig = go.Figure(data=data,layout=layout)  
py.iplot(fig)

**3.9. Visualising data with principal components**[**¶**](#1mrcu09)

In [20]:

**from** **sklearn.decomposition** **import** PCA  
  
pca = PCA(n\_components = 2)  
  
X = telcom[[i **for** i **in** telcom.columns **if** i **not** **in** Id\_col + target\_col]]  
Y = telcom[target\_col + Id\_col]  
  
principal\_components = pca.fit\_transform(X)  
pca\_data = pd.DataFrame(principal\_components,columns = ["PC1","PC2"])  
pca\_data = pca\_data.merge(Y,left\_index=**True**,right\_index=**True**,how="left")  
pca\_data["Churn"] = pca\_data["Churn"].replace({1:"Churn",0:"Not Churn"})  
  
**def** pca\_scatter(target,color) :  
 tracer = go.Scatter(x = pca\_data[pca\_data["Churn"] == target]["PC1"] ,  
 y = pca\_data[pca\_data["Churn"] == target]["PC2"],  
 name = target,mode = "markers",  
 marker = dict(color = color,  
 line = dict(width = .5),  
 symbol = "diamond-open"),  
 text = ("Customer Id : " +   
 pca\_data[pca\_data["Churn"] == target]['customerID'])  
 )  
 **return** tracer  
  
layout = go.Layout(dict(title = "Visualising data with principal components",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "principal component 1",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "principal component 2",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 height = 600  
 )  
 )  
trace1 = pca\_scatter("Churn",'red')  
trace2 = pca\_scatter("Not Churn",'royalblue')  
data = [trace2,trace1]  
fig = go.Figure(data=data,layout=layout)  
py.iplot(fig)

**3.10. Binary variables distribution in customer attrition(Radar Chart)**[**¶**](#46r0co2)

In [21]:

*#separating binary columns*  
bi\_cs = telcom.nunique()[telcom.nunique() == 2].keys()  
dat\_rad = telcom[bi\_cs]  
  
*#plotting radar chart for churn and non churn customers(binary variables)*  
**def** plot\_radar(df,aggregate,title) :  
 data\_frame = df[df["Churn"] == aggregate]   
 data\_frame\_x = data\_frame[bi\_cs].sum().reset\_index()  
 data\_frame\_x.columns = ["feature","yes"]  
 data\_frame\_x["no"] = data\_frame.shape[0] - data\_frame\_x["yes"]  
 data\_frame\_x = data\_frame\_x[data\_frame\_x["feature"] != "Churn"]  
   
 *#count of 1's(yes)*  
 trace1 = go.Scatterpolar(r = data\_frame\_x["yes"].values.tolist(),  
 theta = data\_frame\_x["feature"].tolist(),  
 fill = "toself",name = "count of 1's",  
 mode = "markers+lines",  
 marker = dict(size = 5)  
 )  
 *#count of 0's(No)*  
 trace2 = go.Scatterpolar(r = data\_frame\_x["no"].values.tolist(),  
 theta = data\_frame\_x["feature"].tolist(),  
 fill = "toself",name = "count of 0's",  
 mode = "markers+lines",  
 marker = dict(size = 5)  
 )   
 layout = go.Layout(dict(polar = dict(radialaxis = dict(visible = **True**,  
 side = "counterclockwise",  
 showline = **True**,  
 linewidth = 2,  
 tickwidth = 2,  
 gridcolor = "white",  
 gridwidth = 2),  
 angularaxis = dict(tickfont = dict(size = 10),  
 layer = "below traces"  
 ),  
 bgcolor = "rgb(243,243,243)",  
 ),  
 paper\_bgcolor = "rgb(243,243,243)",  
 title = title,height = 700))  
   
 data = [trace2,trace1]  
 fig = go.Figure(data=data,layout=layout)  
 py.iplot(fig)  
  
*#plot*  
plot\_radar(dat\_rad,1,"Churn - Customers")  
plot\_radar(dat\_rad,0,"Non Churn - Customers")

**5. Model Building**[**¶**](#2lwamvv)

## **5.1. Baseline Model**[**¶**](#111kx3o)

In [22]:

**from** **sklearn.model\_selection** **import** train\_test\_split  
**from** **sklearn.linear\_model** **import** LogisticRegression  
**from** **sklearn.metrics** **import** confusion\_matrix,accuracy\_score,classification\_report  
**from** **sklearn.metrics** **import** roc\_auc\_score,roc\_curve,scorer  
**from** **sklearn.metrics** **import** f1\_score  
**import** **statsmodels.api** **as** **sm**  
**from** **sklearn.metrics** **import** precision\_score,recall\_score  
**from** **yellowbrick.classifier** **import** DiscriminationThreshold  
*#splitting train and test data*   
train,test = train\_test\_split(telcom,test\_size = .25 ,random\_state = 111)  
   
*##seperating dependent and independent variables*  
cols = [i **for** i **in** telcom.columns **if** i **not** **in** Id\_col + target\_col]  
train\_X = train[cols]  
train\_Y = train[target\_col]  
test\_X = test[cols]  
test\_Y = test[target\_col]  
  
*#Function attributes*  
*#dataframe - processed dataframe*  
*#Algorithm - Algorithm used*   
*#training\_x - predictor variables dataframe(training)*  
*#testing\_x - predictor variables dataframe(testing)*  
*#training\_y - target variable(training)*  
*#training\_y - target variable(testing)*  
*#cf - ["coefficients","features"](cooefficients for logistic*   
 *#regression,features for tree based models)*  
  
*#threshold\_plot - if True returns threshold plot for model*  
   
**def** telecom\_churn\_prediction(algorithm,training\_x,testing\_x,  
 training\_y,testing\_y,cols,cf,threshold\_plot) :  
   
 *#model*  
 algorithm.fit(training\_x,training\_y)  
 predictions = algorithm.predict(testing\_x)  
 probabilities = algorithm.predict\_proba(testing\_x)  
 *#coeffs*  
 **if** cf == "coefficients" :  
 coefficients = pd.DataFrame(algorithm.coef\_.ravel())  
 **elif** cf == "features" :  
 coefficients = pd.DataFrame(algorithm.feature\_importances\_)  
   
 column\_df = pd.DataFrame(cols)  
 coef\_sumry = (pd.merge(coefficients,column\_df,left\_index= **True**,  
 right\_index= **True**, how = "left"))  
 coef\_sumry.columns = ["coefficients","features"]  
 coef\_sumry = coef\_sumry.sort\_values(by = "coefficients",ascending = **False**)  
   
 print (algorithm)  
 print ("**\n** Classification report : **\n**",classification\_report(testing\_y,predictions))  
 print ("Accuracy Score : ",accuracy\_score(testing\_y,predictions))  
 *#confusion matrix*  
 conf\_matrix = confusion\_matrix(testing\_y,predictions)  
 *#roc\_auc\_score*  
 model\_roc\_auc = roc\_auc\_score(testing\_y,predictions)   
 print ("Area under curve : ",model\_roc\_auc,"**\n**")  
 fpr,tpr,thresholds = roc\_curve(testing\_y,probabilities[:,1])  
   
 *#plot confusion matrix*  
 trace1 = go.Heatmap(z = conf\_matrix ,  
 x = ["Not churn","Churn"],  
 y = ["Not churn","Churn"],  
 showscale = **False**,colorscale = "Picnic",  
 name = "matrix")  
   
 *#plot roc curve*  
 trace2 = go.Scatter(x = fpr,y = tpr,  
 name = "Roc : " + str(model\_roc\_auc),  
 line = dict(color = ('rgb(22, 96, 167)'),width = 2))  
 trace3 = go.Scatter(x = [0,1],y=[0,1],  
 line = dict(color = ('rgb(205, 12, 24)'),width = 2,  
 dash = 'dot'))  
   
 *#plot coeffs*  
 trace4 = go.Bar(x = coef\_sumry["features"],y = coef\_sumry["coefficients"],  
 name = "coefficients",  
 marker = dict(color = coef\_sumry["coefficients"],  
 colorscale = "Picnic",  
 line = dict(width = .6,color = "black")))  
   
 *#subplots*  
 fig = tls.make\_subplots(rows=2, cols=2, specs=[[{}, {}], [{'colspan': 2}, **None**]],  
 subplot\_titles=('Confusion Matrix',  
 'Receiver operating characteristic',  
 'Feature Importances'))  
   
 fig.append\_trace(trace1,1,1)  
 fig.append\_trace(trace2,1,2)  
 fig.append\_trace(trace3,1,2)  
 fig.append\_trace(trace4,2,1)  
   
 fig['layout'].update(showlegend=**False**, title="Model performance" ,  
 autosize = **False**,height = 900,width = 800,  
 plot\_bgcolor = 'rgba(240,240,240, 0.95)',  
 paper\_bgcolor = 'rgba(240,240,240, 0.95)',  
 margin = dict(b = 195))  
 fig["layout"]["xaxis2"].update(dict(title = "false positive rate"))  
 fig["layout"]["yaxis2"].update(dict(title = "true positive rate"))  
 fig["layout"]["xaxis3"].update(dict(showgrid = **True**,tickfont = dict(size = 10),  
 tickangle = 90))  
 py.iplot(fig)  
   
 **if** threshold\_plot == **True** :   
 visualizer = DiscriminationThreshold(algorithm)  
 visualizer.fit(training\_x,training\_y)  
 visualizer.poof()  
   
logit = LogisticRegression(C=1.0, class\_weight=**None**, dual=**False**, fit\_intercept=**True**,  
 intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,  
 penalty='l2', random\_state=**None**, solver='liblinear', tol=0.0001,  
 verbose=0, warm\_start=**False**)  
  
telecom\_churn\_prediction(logit,train\_X,test\_X,train\_Y,test\_Y,  
 cols,"coefficients",threshold\_plot = **True**)

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,  
 intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,  
 penalty='l2', random\_state=None, solver='liblinear', tol=0.0001,  
 verbose=0, warm\_start=False)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.83 0.90 0.87 1268  
 1 0.68 0.54 0.60 490  
  
 micro avg 0.80 0.80 0.80 1758  
 macro avg 0.76 0.72 0.73 1758  
weighted avg 0.79 0.80 0.79 1758  
  
Accuracy Score : 0.8003412969283277  
Area under curve : 0.7194714478851477

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**5.2. Synthetic Minority Oversampling TEchnique (SMOTE)**[**¶**](#3l18frh)

* Randomly pick a point from the minority class.
* Compute the k-nearest neighbors (for some pre-specified k) for this point.
* Add k new points somewhere between the chosen point and each of its neighbors

In [23]:

**from** **imblearn.over\_sampling** **import** SMOTE  
  
cols = [i **for** i **in** telcom.columns **if** i **not** **in** Id\_col+target\_col]  
  
smote\_X = telcom[cols]  
smote\_Y = telcom[target\_col]  
  
*#Split train and test data*  
smote\_train\_X,smote\_test\_X,smote\_train\_Y,smote\_test\_Y = train\_test\_split(smote\_X,smote\_Y,  
 test\_size = .25 ,  
 random\_state = 111)  
  
*#oversampling minority class using smote*  
os = SMOTE(random\_state = 0)  
os\_smote\_X,os\_smote\_Y = os.fit\_sample(smote\_train\_X,smote\_train\_Y)  
os\_smote\_X = pd.DataFrame(data = os\_smote\_X,columns=cols)  
os\_smote\_Y = pd.DataFrame(data = os\_smote\_Y,columns=target\_col)  
*###*  
  
  
  
logit\_smote = LogisticRegression(C=1.0, class\_weight=**None**, dual=**False**, fit\_intercept=**True**,  
 intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,  
 penalty='l2', random\_state=**None**, solver='liblinear', tol=0.0001,  
 verbose=0, warm\_start=**False**)  
  
telecom\_churn\_prediction(logit\_smote,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 cols,"coefficients",threshold\_plot = **True**)

Using Theano backend.

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,  
 intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,  
 penalty='l2', random\_state=None, solver='liblinear', tol=0.0001,  
 verbose=0, warm\_start=False)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.90 0.75 0.82 1268  
 1 0.55 0.79 0.65 490  
  
 micro avg 0.76 0.76 0.76 1758  
 macro avg 0.73 0.77 0.73 1758  
weighted avg 0.80 0.76 0.77 1758  
  
Accuracy Score : 0.7605233219567691  
Area under curve : 0.769503637417112

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**5.3. Recursive Feature Elimination**[**¶**](#206ipza)

Recursive Feature Elimination (RFE) is based on the idea to repeatedly construct a model and choose either the best or worst performing feature, setting the feature aside and then repeating the process with the rest of the features. This process is applied until all features in the dataset are exhausted. The goal of RFE is to select features by recursively considering smaller and smaller sets of features.

In [24]:

**from** **sklearn.feature\_selection** **import** RFE  
  
logit = LogisticRegression()  
  
rfe = RFE(logit,10)  
rfe = rfe.fit(os\_smote\_X,os\_smote\_Y.values.ravel())  
  
rfe.support\_  
rfe.ranking\_  
  
*#identified columns Recursive Feature Elimination*  
idc\_rfe = pd.DataFrame({"rfe\_support" :rfe.support\_,  
 "columns" : [i **for** i **in** telcom.columns **if** i **not** **in** Id\_col + target\_col],  
 "ranking" : rfe.ranking\_,  
 })  
cols = idc\_rfe[idc\_rfe["rfe\_support"] == **True**]["columns"].tolist()  
  
  
*#separating train and test data*  
train\_rf\_X = os\_smote\_X[cols]  
train\_rf\_Y = os\_smote\_Y  
test\_rf\_X = test[cols]  
test\_rf\_Y = test[target\_col]  
  
logit\_rfe = LogisticRegression(C=1.0, class\_weight=**None**, dual=**False**, fit\_intercept=**True**,  
 intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,  
 penalty='l2', random\_state=**None**, solver='liblinear', tol=0.0001,  
 verbose=0, warm\_start=**False**)  
*#applying model*  
telecom\_churn\_prediction(logit\_rfe,train\_rf\_X,test\_rf\_X,train\_rf\_Y,test\_rf\_Y,  
 cols,"coefficients",threshold\_plot = **True**)  
  
tab\_rk = ff.create\_table(idc\_rfe)  
py.iplot(tab\_rk)

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,  
 intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,  
 penalty='l2', random\_state=None, solver='liblinear', tol=0.0001,  
 verbose=0, warm\_start=False)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.91 0.73 0.81 1268  
 1 0.53 0.80 0.64 490  
  
 micro avg 0.75 0.75 0.75 1758  
 macro avg 0.72 0.77 0.73 1758  
weighted avg 0.80 0.75 0.76 1758  
  
Accuracy Score : 0.7502844141069397  
Area under curve : 0.7667884503959312

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**5.4. Univariate Selection**[**¶**](#4k668n3)

* Feature Extraction with Univariate Statistical Tests (Chi-squared for classification)
* uses the chi squared (chi^2) statistical test for non-negative features to select the best features

In [25]:

**from** **sklearn.feature\_selection** **import** chi2  
**from** **sklearn.feature\_selection** **import** SelectKBest  
  
*#select columns*  
cols = [i **for** i **in** telcom.columns **if** i **not** **in** Id\_col + target\_col ]  
  
*#dataframe with non negative values*  
df\_x = df\_telcom\_og[cols]  
df\_y = df\_telcom\_og[target\_col]  
  
*#fit model with k= 3*  
select = SelectKBest(score\_func = chi2,k = 3)  
fit = select.fit(df\_x,df\_y)  
  
*#Summerize scores*  
print ("scores")  
print (fit.scores\_)  
print ("P - Values")  
print (fit.pvalues\_)  
  
*#create dataframe*  
score = pd.DataFrame({"features":cols,"scores":fit.scores\_,"p\_values":fit.pvalues\_ })  
score = score.sort\_values(by = "scores" ,ascending =**False**)  
  
  
*#createing new label for categorical and numerical columns*  
score["feature\_type"] = np.where(score["features"].isin(num\_cols),"Numerical","Categorical")  
  
*#plot*  
trace = go.Scatter(x = score[score["feature\_type"] == "Categorical"]["features"],  
 y = score[score["feature\_type"] == "Categorical"]["scores"],  
 name = "Categorial",mode = "lines+markers",  
 marker = dict(color = "red",  
 line = dict(width =1))  
 )  
  
trace1 = go.Bar(x = score[score["feature\_type"] == "Numerical"]["features"],  
 y = score[score["feature\_type"] == "Numerical"]["scores"],name = "Numerical",  
 marker = dict(color = "royalblue",  
 line = dict(width =1)),  
 xaxis = "x2",yaxis = "y2"  
 )  
layout = go.Layout(dict(title = "Scores for Categorical & Numerical features",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 tickfont = dict(size =10),  
 domain=[0, 0.7],  
 tickangle = 90,zerolinewidth=1,  
 ticklen=5,gridwidth=2),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "scores",  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 margin = dict(b=200),  
 xaxis2=dict(domain=[0.8, 1],tickangle = 90,  
 gridcolor = 'rgb(255, 255, 255)'),  
 yaxis2=dict(anchor='x2',gridcolor = 'rgb(255, 255, 255)')  
 )  
 )  
  
data=[trace,trace1]  
fig = go.Figure(data=data,layout=layout)  
py.iplot(fig)

scores  
[2.54297062e-01 1.33482766e+02 8.18577694e+01 1.31271509e+02  
 9.29483891e-02 1.47165601e+02 3.12098318e+01 2.02160070e+01  
 1.35439602e+02 1.73206148e+01 1.59306111e+01 1.04979224e+02  
 3.88864216e+00 8.68247305e-01 6.51465136e+00 7.11376111e+01  
 3.72082851e+02 2.85475152e+02 5.16714004e+02 1.76608724e+02  
 4.86223101e+02 7.66190658e+01 9.99725387e+01 4.24113152e+02  
 4.47251434e+01 1.63773281e+04 3.65307468e+03 6.29630810e+05]  
P - Values  
[6.14065505e-001 7.08954608e-031 1.46240915e-019 2.15953960e-030  
 7.60461827e-001 7.21988253e-034 2.31590182e-008 6.91717063e-006  
 2.64595220e-031 3.15742928e-005 6.57073922e-005 1.23423173e-024  
 4.86137123e-002 3.51440986e-001 1.06989295e-002 3.33158163e-017  
 6.58713045e-083 4.81399951e-064 2.19511926e-114 2.66631661e-040  
 9.45428638e-108 2.07328356e-018 1.54524820e-023 3.10584857e-094  
 2.26727030e-011 0.00000000e+000 0.00000000e+000 0.00000000e+000]

**5.5. Decision Tree Visualization**[**¶**](#2zbgiuw)

* Using top three numerical features

In [26]:

**from** **sklearn.tree** **import** DecisionTreeClassifier  
**from** **sklearn.tree** **import** export\_graphviz  
**from** **sklearn** **import** tree  
**from** **graphviz** **import** Source  
**from** **IPython.display** **import** SVG,display  
  
*#top 3 categorical features*  
features\_cat = score[score["feature\_type"] == "Categorical"]["features"][:3].tolist()  
  
*#top 3 numerical features*  
features\_num = score[score["feature\_type"] == "Numerical"]["features"][:3].tolist()  
  
  
*#Function attributes*  
*#columns - selected columns*  
*#maximum\_depth - depth of tree*  
*#criterion\_type - ["gini" or "entropy"]*  
*#split\_type - ["best" or "random"]*  
*#Model Performance - True (gives model output)*  
  
**def** plot\_decision\_tree(columns,maximum\_depth,criterion\_type,  
 split\_type,model\_performance = **None**) :  
   
 *#separating dependent and in dependent variables*  
 dtc\_x = df\_x[columns]  
 dtc\_y = df\_y[target\_col]  
   
 *#model*  
 dt\_classifier = DecisionTreeClassifier(max\_depth = maximum\_depth,  
 splitter = split\_type,  
 criterion = criterion\_type,  
 )  
 dt\_classifier.fit(dtc\_x,dtc\_y)  
   
 *#plot decision tree*  
 graph = Source(tree.export\_graphviz(dt\_classifier,out\_file=**None**,  
 rounded=**True**,proportion = **False**,  
 feature\_names = columns,   
 precision = 2,  
 class\_names=["Not churn","Churn"],  
 filled = **True**   
 )  
 )  
   
 *#model performance*  
 **if** model\_performance == **True** :  
 telecom\_churn\_prediction(dt\_classifier,  
 dtc\_x,test\_X[columns],  
 dtc\_y,test\_Y,  
 columns,"features",threshold\_plot = **True**)  
 display(graph)  
   
plot\_decision\_tree(features\_num,3,"gini","best")

tenure <= 16.5 gini = 0.39 samples = 7032 value = [5163, 1869] class = Not churn MonthlyCharges <= 68.62 gini = 0.5 samples = 2539 value = [1375, 1164] class = Not churn True MonthlyCharges <= 69.97 gini = 0.26 samples = 4493 value = [3788, 705] class = Not churn False tenure <= 3.5 gini = 0.42 samples = 1386 value = [975, 411] class = Not churn TotalCharges <= 120.0 gini = 0.45 samples = 1153 value = [400, 753] class = Churn gini = 0.49 samples = 605 value = [342, 263] class = Not churn gini = 0.31 samples = 781 value = [633, 148] class = Not churn gini = 0.25 samples = 239 value = [35, 204] class = Churn gini = 0.48 samples = 914 value = [365, 549] class = Churn MonthlyCharges <= 28.55 gini = 0.12 samples = 1957 value = [1832, 125] class = Not churn tenure <= 43.5 gini = 0.35 samples = 2536 value = [1956, 580] class = Not churn gini = 0.04 samples = 949 value = [932, 17] class = Not churn gini = 0.19 samples = 1008 value = [900, 108] class = Not churn gini = 0.46 samples = 979 value = [621, 358] class = Not churn gini = 0.24 samples = 1557 value = [1335, 222] class = Not churn

**\* Using top three categorical features**[**¶**](#1egqt2p)

In [27]:

plot\_decision\_tree(features\_cat,3,"entropy","best",  
 model\_performance = **True** ,)

DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=3,  
 max\_features=None, max\_leaf\_nodes=None,  
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
 min\_samples\_leaf=1, min\_samples\_split=2,  
 min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,  
 splitter='best')  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.83 0.84 0.84 1268  
 1 0.58 0.54 0.56 490  
  
 micro avg 0.76 0.76 0.76 1758  
 macro avg 0.70 0.69 0.70 1758  
weighted avg 0.76 0.76 0.76 1758  
  
Accuracy Score : 0.7610921501706485  
Area under curve : 0.6947675915792185

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

Contract\_Month-to-month <= 0.5 entropy = 0.84 samples = 7032 value = [5163, 1869] class = Not churn Contract\_Two year <= 0.5 entropy = 0.36 samples = 3157 value = [2943, 214] class = Not churn True PaymentMethod\_Electronic check <= 0.5 entropy = 0.98 samples = 3875 value = [2220, 1655] class = Not churn False PaymentMethod\_Electronic check <= 0.5 entropy = 0.51 samples = 1472 value = [1306, 166] class = Not churn PaymentMethod\_Electronic check <= 0.5 entropy = 0.19 samples = 1685 value = [1637, 48] class = Not churn entropy = 0.44 samples = 1125 value = [1023, 102] class = Not churn entropy = 0.69 samples = 347 value = [283, 64] class = Not churn entropy = 0.16 samples = 1517 value = [1482, 35] class = Not churn entropy = 0.39 samples = 168 value = [155, 13] class = Not churn entropy = 0.91 samples = 2025 value = [1364, 661] class = Not churn entropy = 1.0 samples = 1850 value = [856, 994] class = Churn

In [28]:

*#using contract,tenure and paperless billing variables*  
columns = ['tenure','Contract\_Month-to-month', 'PaperlessBilling',  
 'Contract\_One year', 'Contract\_Two year']  
  
plot\_decision\_tree(columns,3,"gini","best",model\_performance= **True**)

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=3,  
 max\_features=None, max\_leaf\_nodes=None,  
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
 min\_samples\_leaf=1, min\_samples\_split=2,  
 min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,  
 splitter='best')  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.85 0.74 0.79 1268  
 1 0.50 0.66 0.57 490  
  
 micro avg 0.72 0.72 0.72 1758  
 macro avg 0.68 0.70 0.68 1758  
weighted avg 0.75 0.72 0.73 1758  
  
Accuracy Score : 0.7218430034129693  
Area under curve : 0.7038724006952939

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

Contract\_Month-to-month <= 0.5 gini = 0.39 samples = 7032 value = [5163, 1869] class = Not churn Contract\_Two year <= 0.5 gini = 0.13 samples = 3157 value = [2943, 214] class = Not churn True tenure <= 5.5 gini = 0.49 samples = 3875 value = [2220, 1655] class = Not churn False PaperlessBilling <= 0.5 gini = 0.2 samples = 1472 value = [1306, 166] class = Not churn PaperlessBilling <= 0.5 gini = 0.06 samples = 1685 value = [1637, 48] class = Not churn gini = 0.13 samples = 673 value = [625, 48] class = Not churn gini = 0.25 samples = 799 value = [681, 118] class = Not churn gini = 0.03 samples = 902 value = [887, 15] class = Not churn gini = 0.08 samples = 783 value = [750, 33] class = Not churn PaperlessBilling <= 0.5 gini = 0.49 samples = 1318 value = [578, 740] class = Churn PaperlessBilling <= 0.5 gini = 0.46 samples = 2557 value = [1642, 915] class = Not churn gini = 0.49 samples = 535 value = [309, 226] class = Not churn gini = 0.45 samples = 783 value = [269, 514] class = Churn gini = 0.36 samples = 754 value = [574, 180] class = Not churn gini = 0.48 samples = 1803 value = [1068, 735] class = Not churn

**5.6. KNN Classifier**[**¶**](#3ygebqi)

* Applying knn algorithm to smote oversampled data.

In [29]:

**def** telecom\_churn\_prediction\_alg(algorithm,training\_x,testing\_x,  
 training\_y,testing\_y,threshold\_plot = **True**) :  
   
 *#model*  
 algorithm.fit(training\_x,training\_y)  
 predictions = algorithm.predict(testing\_x)  
 probabilities = algorithm.predict\_proba(testing\_x)  
   
 print (algorithm)  
 print ("**\n** Classification report : **\n**",classification\_report(testing\_y,predictions))  
 print ("Accuracy Score : ",accuracy\_score(testing\_y,predictions))  
 *#confusion matrix*  
 conf\_matrix = confusion\_matrix(testing\_y,predictions)  
 *#roc\_auc\_score*  
 model\_roc\_auc = roc\_auc\_score(testing\_y,predictions)   
 print ("Area under curve : ",model\_roc\_auc)  
 fpr,tpr,thresholds = roc\_curve(testing\_y,probabilities[:,1])  
   
 *#plot roc curve*  
 trace1 = go.Scatter(x = fpr,y = tpr,  
 name = "Roc : " + str(model\_roc\_auc),  
 line = dict(color = ('rgb(22, 96, 167)'),width = 2),  
 )  
 trace2 = go.Scatter(x = [0,1],y=[0,1],  
 line = dict(color = ('rgb(205, 12, 24)'),width = 2,  
 dash = 'dot'))  
   
 *#plot confusion matrix*  
 trace3 = go.Heatmap(z = conf\_matrix ,x = ["Not churn","Churn"],  
 y = ["Not churn","Churn"],  
 showscale = **False**,colorscale = "Blues",name = "matrix",  
 xaxis = "x2",yaxis = "y2"  
 )  
   
 layout = go.Layout(dict(title="Model performance" ,  
 autosize = **False**,height = 500,width = 800,  
 showlegend = **False**,  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(title = "false positive rate",  
 gridcolor = 'rgb(255, 255, 255)',  
 domain=[0, 0.6],  
 ticklen=5,gridwidth=2),  
 yaxis = dict(title = "true positive rate",  
 gridcolor = 'rgb(255, 255, 255)',  
 zerolinewidth=1,  
 ticklen=5,gridwidth=2),  
 margin = dict(b=200),  
 xaxis2=dict(domain=[0.7, 1],tickangle = 90,  
 gridcolor = 'rgb(255, 255, 255)'),  
 yaxis2=dict(anchor='x2',gridcolor = 'rgb(255, 255, 255)')  
 )  
 )  
 data = [trace1,trace2,trace3]  
 fig = go.Figure(data=data,layout=layout)  
   
 py.iplot(fig)  
   
 **if** threshold\_plot == **True** :   
 visualizer = DiscriminationThreshold(algorithm)  
 visualizer.fit(training\_x,training\_y)  
 visualizer.poof()  
  
   
**from** **sklearn.neighbors** **import** KNeighborsClassifier  
knn = KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
 metric\_params=**None**, n\_jobs=1, n\_neighbors=5, p=2,  
 weights='uniform')  
telecom\_churn\_prediction\_alg(knn,os\_smote\_X,test\_X,  
 os\_smote\_Y,test\_Y,threshold\_plot = **True**)

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
 metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,  
 weights='uniform')  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.86 0.69 0.76 1268  
 1 0.47 0.71 0.56 490  
  
 micro avg 0.69 0.69 0.69 1758  
 macro avg 0.66 0.70 0.66 1758  
weighted avg 0.75 0.69 0.71 1758  
  
Accuracy Score : 0.6939704209328783  
Area under curve : 0.6989506212579669

**5.7. Vizualising a decision tree from random forest classifier**[**¶**](#2dlolyb)

In [30]:

**from** **sklearn.ensemble** **import** RandomForestClassifier  
  
*#function attributes*  
*#columns - column used*  
*#nf\_estimators - The number of trees in the forest.*  
*#estimated\_tree - tree number to be displayed*  
*#maximum\_depth - depth of the tree*  
*#criterion\_type - split criterion type ["gini" or "entropy"]*  
*#Model performance - prints performance of model*  
  
**def** plot\_tree\_randomforest(columns,nf\_estimators,  
 estimated\_tree,maximum\_depth,  
 criterion\_type,model\_performance = **None**) :  
   
 dataframe = df\_telcom\_og[columns + target\_col].copy()  
   
 *#train and test datasets*  
 rf\_x = dataframe[[i **for** i **in** columns **if** i **not** **in** target\_col]]  
 rf\_y = dataframe[target\_col]  
   
 *#random forest classifier*  
 rfc = RandomForestClassifier(n\_estimators = nf\_estimators,  
 max\_depth = maximum\_depth,  
 criterion = criterion\_type,  
 )  
 rfc.fit(rf\_x,rf\_y)  
   
 estimated\_tree = rfc.estimators\_[estimated\_tree]  
   
 graph = Source(tree.export\_graphviz(estimated\_tree,out\_file=**None**,  
 rounded=**True**,proportion = **False**,  
 feature\_names = columns,   
 precision = 2,  
 class\_names=["Not churn","Churn"],  
 filled = **True**))  
 display(graph)  
   
 *#model performance*  
 **if** model\_performance == **True** :  
 telecom\_churn\_prediction(rfc,  
 rf\_x,test\_X[columns],  
 rf\_y,test\_Y,  
 columns,"features",threshold\_plot = **True**)  
   
  
cols1 = [ i **for** i **in** train\_X.columns **if** i **not** **in** target\_col + Id\_col]   
plot\_tree\_randomforest(cols1,100,99,3,"entropy",**True**)

Contract\_Two year <= 0.5 entropy = 0.84 samples = 4453 value = [5143, 1889] class = Not churn InternetService\_No <= 0.5 entropy = 0.93 samples = 3386 value = [3535, 1834] class = Not churn True tenure <= 38.5 entropy = 0.21 samples = 1067 value = [1608, 55] class = Not churn False OnlineSecurity <= 0.5 entropy = 0.96 samples = 2829 value = [2747, 1734] class = Not churn PaymentMethod\_Electronic check <= 0.5 entropy = 0.51 samples = 557 value = [788, 100] class = Not churn entropy = 0.99 samples = 2026 value = [1761, 1481] class = Not churn entropy = 0.73 samples = 803 value = [986, 253] class = Not churn entropy = 0.48 samples = 489 value = [696, 80] class = Not churn entropy = 0.68 samples = 68 value = [92, 20] class = Not churn PaymentMethod\_Mailed check <= 0.5 entropy = 0.03 samples = 169 value = [279, 1] class = Not churn StreamingTV <= 0.5 entropy = 0.24 samples = 898 value = [1329, 54] class = Not churn entropy = 0.07 samples = 78 value = [126, 1] class = Not churn entropy = 0.0 samples = 91 value = [153, 0] class = Not churn entropy = 0.15 samples = 431 value = [660, 14] class = Not churn entropy = 0.31 samples = 467 value = [669, 40] class = Not churn

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='entropy',  
 max\_depth=3, max\_features='auto', max\_leaf\_nodes=None,  
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
 min\_samples\_leaf=1, min\_samples\_split=2,  
 min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None,  
 oob\_score=False, random\_state=None, verbose=0,  
 warm\_start=False)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.76 0.97 0.85 1268  
 1 0.71 0.19 0.30 490  
  
 micro avg 0.75 0.75 0.75 1758  
 macro avg 0.73 0.58 0.57 1758  
weighted avg 0.74 0.75 0.70 1758  
  
Accuracy Score : 0.7525597269624573  
Area under curve : 0.5792876456576321

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**5.8. A random forest classifier.**[**¶**](#sqyw64)

* A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement .
* Below are the trees produced by random forest model with 10 estimated trees with maximum depth of three for each tree. Each tree produced is slightly different from other.

In [31]:

*#making 10 trees with random forest.*  
n = np.arange(0,10).tolist()  
cols1 = [ i **for** i **in** train\_X.columns **if** i **not** **in** target\_col + Id\_col]   
**for** i **in** n :  
 plot\_tree\_randomforest(cols1,10,i,3,"entropy",model\_performance=**False**)

InternetService\_No <= 0.5 entropy = 0.85 samples = 4477 value = [5105, 1927] class = Not churn Contract\_Month-to-month <= 0.5 entropy = 0.91 samples = 3505 value = [3694, 1803] class = Not churn True tenure <= 3.5 entropy = 0.4 samples = 972 value = [1411, 124] class = Not churn False Contract\_One year <= 0.5 entropy = 0.46 samples = 1356 value = [1961, 210] class = Not churn tenure <= 10.5 entropy = 1.0 samples = 2149 value = [1733, 1593] class = Not churn entropy = 0.23 samples = 659 value = [1016, 40] class = Not churn entropy = 0.62 samples = 697 value = [945, 170] class = Not churn entropy = 0.95 samples = 926 value = [533, 904] class = Churn entropy = 0.95 samples = 1223 value = [1200, 689] class = Not churn PaperlessBilling <= 0.5 entropy = 0.91 samples = 152 value = [163, 79] class = Not churn Contract\_Two year <= 0.5 entropy = 0.22 samples = 820 value = [1248, 45] class = Not churn entropy = 0.83 samples = 111 value = [130, 47] class = Not churn entropy = 1.0 samples = 41 value = [33, 32] class = Not churn entropy = 0.33 samples = 416 value = [629, 40] class = Not churn entropy = 0.07 samples = 404 value = [619, 5] class = Not churn

Dependents <= 0.5 entropy = 0.83 samples = 4468 value = [5200, 1832] class = Not churn MonthlyCharges <= 68.83 entropy = 0.89 samples = 3150 value = [3452, 1534] class = Not churn True Contract\_One year <= 0.5 entropy = 0.6 samples = 1318 value = [1748, 298] class = Not churn False TotalCharges <= 223.38 entropy = 0.7 samples = 1346 value = [1713, 403] class = Not churn TotalCharges <= 1536.33 entropy = 0.97 samples = 1804 value = [1739, 1131] class = Not churn entropy = 0.98 samples = 420 value = [367, 257] class = Not churn entropy = 0.46 samples = 926 value = [1346, 146] class = Not churn entropy = 0.93 samples = 660 value = [350, 671] class = Churn entropy = 0.81 samples = 1144 value = [1389, 460] class = Not churn PaymentMethod\_Electronic check <= 0.5 entropy = 0.65 samples = 964 value = [1250, 251] class = Not churn StreamingMovies <= 0.5 entropy = 0.42 samples = 354 value = [498, 47] class = Not churn entropy = 0.5 samples = 746 value = [1032, 127] class = Not churn entropy = 0.94 samples = 218 value = [218, 124] class = Not churn entropy = 0.22 samples = 213 value = [307, 11] class = Not churn entropy = 0.63 samples = 141 value = [191, 36] class = Not churn

Contract\_Two year <= 0.5 entropy = 0.84 samples = 4454 value = [5163, 1869] class = Not churn PaperlessBilling <= 0.5 entropy = 0.92 samples = 3385 value = [3553, 1808] class = Not churn True PaymentMethod\_Mailed check <= 0.5 entropy = 0.23 samples = 1069 value = [1610, 61] class = Not churn False tenure <= 5.5 entropy = 0.77 samples = 1241 value = [1532, 443] class = Not churn PaymentMethod\_Credit card (automatic) <= 0.5 entropy = 0.97 samples = 2144 value = [2021, 1365] class = Not churn entropy = 0.98 samples = 360 value = [348, 240] class = Not churn entropy = 0.6 samples = 881 value = [1184, 203] class = Not churn entropy = 0.99 samples = 1779 value = [1594, 1213] class = Not churn entropy = 0.83 samples = 365 value = [427, 152] class = Not churn PaperlessBilling <= 0.5 entropy = 0.26 samples = 835 value = [1257, 57] class = Not churn InternetService\_Fiber optic <= 0.5 entropy = 0.09 samples = 234 value = [353, 4] class = Not churn entropy = 0.14 samples = 409 value = [608, 12] class = Not churn entropy = 0.35 samples = 426 value = [649, 45] class = Not churn entropy = 0.05 samples = 218 value = [332, 2] class = Not churn entropy = 0.43 samples = 16 value = [21, 2] class = Not churn

tenure <= 17.5 entropy = 0.83 samples = 4465 value = [5167, 1865] class = Not churn PaymentMethod\_Electronic check <= 0.5 entropy = 0.99 samples = 1654 value = [1468, 1148] class = Not churn True tenure <= 57.5 entropy = 0.64 samples = 2811 value = [3699, 717] class = Not churn False Contract\_Month-to-month <= 0.5 entropy = 0.9 samples = 933 value = [1019, 464] class = Not churn InternetService\_Fiber optic <= 0.5 entropy = 0.97 samples = 721 value = [449, 684] class = Churn entropy = 0.29 samples = 170 value = [259, 14] class = Not churn entropy = 0.95 samples = 763 value = [760, 450] class = Not churn entropy = 1.0 samples = 258 value = [220, 189] class = Not churn entropy = 0.9 samples = 463 value = [229, 495] class = Churn Contract\_Month-to-month <= 0.5 entropy = 0.75 samples = 1793 value = [2202, 607] class = Not churn MonthlyCharges <= 90.58 entropy = 0.36 samples = 1018 value = [1497, 110] class = Not churn entropy = 0.41 samples = 901 value = [1292, 115] class = Not churn entropy = 0.93 samples = 892 value = [910, 492] class = Not churn entropy = 0.17 samples = 603 value = [945, 24] class = Not churn entropy = 0.57 samples = 415 value = [552, 86] class = Not churn

tenure <= 18.5 entropy = 0.82 samples = 4424 value = [5224, 1808] class = Not churn Contract\_Month-to-month <= 0.5 entropy = 0.99 samples = 1671 value = [1467, 1213] class = Not churn True SeniorCitizen <= 0.5 entropy = 0.58 samples = 2753 value = [3757, 595] class = Not churn False StreamingTV <= 0.5 entropy = 0.31 samples = 193 value = [299, 18] class = Not churn MonthlyCharges <= 68.62 entropy = 1.0 samples = 1478 value = [1168, 1195] class = Churn entropy = 0.21 samples = 167 value = [268, 9] class = Not churn entropy = 0.77 samples = 26 value = [31, 9] class = Not churn entropy = 0.93 samples = 745 value = [781, 410] class = Not churn entropy = 0.92 samples = 733 value = [387, 785] class = Churn InternetService\_DSL <= 0.5 entropy = 0.5 samples = 2299 value = [3234, 401] class = Not churn Contract\_Two year <= 0.5 entropy = 0.84 samples = 454 value = [523, 194] class = Not churn entropy = 0.6 samples = 1447 value = [1925, 325] class = Not churn entropy = 0.31 samples = 852 value = [1309, 76] class = Not churn entropy = 0.91 samples = 359 value = [392, 190] class = Not churn entropy = 0.19 samples = 95 value = [131, 4] class = Not churn

tenure <= 17.5 entropy = 0.82 samples = 4453 value = [5220, 1812] class = Not churn InternetService\_No <= 0.5 entropy = 0.99 samples = 1679 value = [1488, 1130] class = Not churn True MonthlyCharges <= 74.88 entropy = 0.62 samples = 2774 value = [3732, 682] class = Not churn False TechSupport <= 0.5 entropy = 1.0 samples = 1294 value = [954, 1045] class = Churn tenure <= 3.5 entropy = 0.58 samples = 385 value = [534, 85] class = Not churn entropy = 0.99 samples = 1060 value = [733, 911] class = Churn entropy = 0.96 samples = 234 value = [221, 134] class = Not churn entropy = 0.85 samples = 143 value = [164, 62] class = Not churn entropy = 0.32 samples = 242 value = [370, 23] class = Not churn PaymentMethod\_Electronic check <= 0.5 entropy = 0.32 samples = 1336 value = [1985, 121] class = Not churn tenure <= 57.5 entropy = 0.8 samples = 1438 value = [1747, 561] class = Not churn entropy = 0.23 samples = 1117 value = [1718, 66] class = Not churn entropy = 0.66 samples = 219 value = [267, 55] class = Not churn entropy = 0.92 samples = 830 value = [882, 451] class = Not churn entropy = 0.51 samples = 608 value = [865, 110] class = Not churn

Contract\_Month-to-month <= 0.5 entropy = 0.83 samples = 4435 value = [5170, 1862] class = Not churn Contract\_One year <= 0.5 entropy = 0.33 samples = 1990 value = [2946, 191] class = Not churn True InternetService\_Fiber optic <= 0.5 entropy = 0.99 samples = 2445 value = [2224, 1671] class = Not churn False TotalCharges <= 3042.47 entropy = 0.17 samples = 1063 value = [1639, 42] class = Not churn PaymentMethod\_Electronic check <= 0.5 entropy = 0.48 samples = 927 value = [1307, 149] class = Not churn entropy = 0.03 samples = 478 value = [765, 2] class = Not churn entropy = 0.26 samples = 585 value = [874, 40] class = Not churn entropy = 0.4 samples = 709 value = [1022, 89] class = Not churn entropy = 0.67 samples = 218 value = [285, 60] class = Not churn PhoneService <= 0.5 entropy = 0.85 samples = 1131 value = [1301, 496] class = Not churn MonthlyCharges <= 108.17 entropy = 0.99 samples = 1314 value = [923, 1175] class = Churn entropy = 0.95 samples = 255 value = [249, 149] class = Not churn entropy = 0.81 samples = 876 value = [1052, 347] class = Not churn entropy = 0.99 samples = 1287 value = [887, 1165] class = Churn entropy = 0.76 samples = 27 value = [36, 10] class = Not churn

MonthlyCharges <= 69.08 entropy = 0.83 samples = 4453 value = [5171, 1861] class = Not churn MultipleLines\_No phone service <= 0.5 entropy = 0.64 samples = 2129 value = [2803, 536] class = Not churn True PaymentMethod\_Electronic check <= 0.5 entropy = 0.94 samples = 2324 value = [2368, 1325] class = Not churn False TotalCharges <= 217.0 entropy = 0.57 samples = 1687 value = [2261, 353] class = Not churn tenure <= 6.5 entropy = 0.82 samples = 442 value = [542, 183] class = Not churn entropy = 0.88 samples = 447 value = [503, 213] class = Not churn entropy = 0.38 samples = 1240 value = [1758, 140] class = Not churn entropy = 0.99 samples = 100 value = [71, 94] class = Churn entropy = 0.63 samples = 342 value = [471, 89] class = Not churn TotalCharges <= 1180.0 entropy = 0.78 samples = 1275 value = [1583, 469] class = Not churn OnlineSecurity <= 0.5 entropy = 1.0 samples = 1049 value = [785, 856] class = Churn entropy = 0.97 samples = 217 value = [143, 215] class = Churn entropy = 0.61 samples = 1058 value = [1440, 254] class = Not churn entropy = 0.98 samples = 828 value = [539, 759] class = Churn entropy = 0.86 samples = 221 value = [246, 97] class = Not churn

MonthlyCharges <= 68.92 entropy = 0.84 samples = 4452 value = [5135, 1897] class = Not churn Contract\_Month-to-month <= 0.5 entropy = 0.63 samples = 2097 value = [2798, 521] class = Not churn True Contract\_Two year <= 0.5 entropy = 0.95 samples = 2355 value = [2337, 1376] class = Not churn False MonthlyCharges <= 41.38 entropy = 0.21 samples = 1055 value = [1615, 57] class = Not churn TotalCharges <= 250.6 entropy = 0.86 samples = 1042 value = [1183, 464] class = Not churn entropy = 0.11 samples = 698 value = [1082, 16] class = Not churn entropy = 0.37 samples = 357 value = [533, 41] class = Not churn entropy = 0.96 samples = 520 value = [506, 310] class = Not churn entropy = 0.69 samples = 522 value = [677, 154] class = Not churn tenure <= 20.5 entropy = 0.99 samples = 1864 value = [1633, 1327] class = Not churn MonthlyCharges <= 92.38 entropy = 0.35 samples = 491 value = [704, 49] class = Not churn entropy = 0.94 samples = 816 value = [465, 836] class = Churn entropy = 0.88 samples = 1048 value = [1168, 491] class = Not churn entropy = 0.1 samples = 243 value = [360, 5] class = Not churn entropy = 0.51 samples = 248 value = [344, 44] class = Not churn

PaymentMethod\_Electronic check <= 0.5 entropy = 0.83 samples = 4457 value = [5193, 1839] class = Not churn PaperlessBilling <= 0.5 entropy = 0.65 samples = 2965 value = [3877, 781] class = Not churn True InternetService\_Fiber optic <= 0.5 entropy = 0.99 samples = 1492 value = [1316, 1058] class = Not churn False InternetService\_Fiber optic <= 0.5 entropy = 0.53 samples = 1436 value = [1973, 274] class = Not churn TechSupport <= 0.5 entropy = 0.74 samples = 1529 value = [1904, 507] class = Not churn entropy = 0.45 samples = 1194 value = [1704, 176] class = Not churn entropy = 0.84 samples = 242 value = [269, 98] class = Not churn entropy = 0.84 samples = 962 value = [1118, 406] class = Not churn entropy = 0.51 samples = 567 value = [786, 101] class = Not churn tenure <= 22.5 entropy = 0.88 samples = 509 value = [554, 238] class = Not churn TotalCharges <= 3052.82 entropy = 1.0 samples = 983 value = [762, 820] class = Churn entropy = 0.98 samples = 285 value = [266, 193] class = Not churn entropy = 0.57 samples = 224 value = [288, 45] class = Not churn entropy = 0.95 samples = 617 value = [373, 639] class = Churn entropy = 0.9 samples = 366 value = [389, 181] class = Not churn

In [32]:

*#making 10 trees with random forest for columns*   
*#selected from recursive feature elimination*  
  
n = np.arange(0,10).tolist()  
cols = idc\_rfe[idc\_rfe["rfe\_support"] == **True**]["columns"].tolist()   
**for** i **in** n :  
 plot\_tree\_randomforest(cols,10,i,3,"gini",model\_performance=**False**)

Contract\_Two year <= 0.5 gini = 0.39 samples = 4469 value = [5169, 1863] class = Not churn InternetService\_No <= 0.5 gini = 0.45 samples = 3398 value = [3533, 1799] class = Not churn True PhoneService <= 0.5 gini = 0.07 samples = 1071 value = [1636, 64] class = Not churn False OnlineSecurity <= 0.5 gini = 0.47 samples = 2836 value = [2780, 1683] class = Not churn PaperlessBilling <= 0.5 gini = 0.23 samples = 562 value = [753, 116] class = Not churn gini = 0.49 samples = 2023 value = [1763, 1401] class = Not churn gini = 0.34 samples = 813 value = [1017, 282] class = Not churn gini = 0.2 samples = 398 value = [533, 69] class = Not churn gini = 0.29 samples = 164 value = [220, 47] class = Not churn TotalCharges <= 3122.35 gini = 0.03 samples = 100 value = [155, 2] class = Not churn InternetService\_Fiber optic <= 0.5 gini = 0.08 samples = 971 value = [1481, 62] class = Not churn gini = 0.0 samples = 43 value = [74, 0] class = Not churn gini = 0.05 samples = 57 value = [81, 2] class = Not churn gini = 0.05 samples = 696 value = [1086, 26] class = Not churn gini = 0.15 samples = 275 value = [395, 36] class = Not churn

InternetService\_Fiber optic <= 0.5 gini = 0.39 samples = 4483 value = [5132, 1900] class = Not churn tenure <= 5.5 gini = 0.25 samples = 2517 value = [3377, 590] class = Not churn True Contract\_Two year <= 0.5 gini = 0.49 samples = 1966 value = [1755, 1310] class = Not churn False TotalCharges <= 60.38 gini = 0.48 samples = 494 value = [476, 305] class = Not churn tenure <= 22.5 gini = 0.16 samples = 2023 value = [2901, 285] class = Not churn gini = 0.5 samples = 286 value = [237, 201] class = Not churn gini = 0.42 samples = 208 value = [239, 104] class = Not churn gini = 0.28 samples = 596 value = [765, 158] class = Not churn gini = 0.11 samples = 1427 value = [2136, 127] class = Not churn TechSupport <= 0.5 gini = 0.5 samples = 1699 value = [1367, 1285] class = Not churn PaperlessBilling <= 0.5 gini = 0.11 samples = 267 value = [388, 25] class = Not churn gini = 0.5 samples = 1346 value = [1004, 1106] class = Churn gini = 0.44 samples = 353 value = [363, 179] class = Not churn gini = 0.04 samples = 74 value = [101, 2] class = Not churn gini = 0.14 samples = 193 value = [287, 23] class = Not churn

TotalCharges <= 335.1 gini = 0.4 samples = 4446 value = [5098, 1934] class = Not churn PaperlessBilling <= 0.5 gini = 0.49 samples = 1010 value = [893, 712] class = Not churn True Contract\_Two year <= 0.5 gini = 0.35 samples = 3436 value = [4205, 1222] class = Not churn False tenure <= 5.5 gini = 0.41 samples = 482 value = [558, 229] class = Not churn tenure <= 4.5 gini = 0.48 samples = 528 value = [335, 483] class = Churn gini = 0.47 samples = 346 value = [337, 213] class = Not churn gini = 0.13 samples = 136 value = [221, 16] class = Not churn gini = 0.45 samples = 440 value = [233, 455] class = Churn gini = 0.34 samples = 88 value = [102, 28] class = Not churn InternetService\_No <= 0.5 gini = 0.43 samples = 2405 value = [2643, 1168] class = Not churn tenure <= 71.5 gini = 0.06 samples = 1031 value = [1562, 54] class = Not churn gini = 0.45 samples = 2175 value = [2278, 1162] class = Not churn gini = 0.03 samples = 230 value = [365, 6] class = Not churn gini = 0.08 samples = 802 value = [1207, 50] class = Not churn gini = 0.02 samples = 229 value = [355, 4] class = Not churn

InternetService\_Fiber optic <= 0.5 gini = 0.39 samples = 4500 value = [5162, 1870] class = Not churn Contract\_Month-to-month <= 0.5 gini = 0.24 samples = 2542 value = [3454, 574] class = Not churn True OnlineSecurity <= 0.5 gini = 0.49 samples = 1958 value = [1708, 1296] class = Not churn False PaperlessBilling <= 0.5 gini = 0.06 samples = 1436 value = [2182, 75] class = Not churn PhoneService <= 0.5 gini = 0.4 samples = 1106 value = [1272, 499] class = Not churn gini = 0.04 samples = 857 value = [1353, 26] class = Not churn gini = 0.11 samples = 579 value = [829, 49] class = Not churn gini = 0.48 samples = 242 value = [238, 153] class = Not churn gini = 0.38 samples = 864 value = [1034, 346] class = Not churn TotalCharges <= 1182.0 gini = 0.5 samples = 1423 value = [1079, 1105] class = Churn Contract\_Two year <= 0.5 gini = 0.36 samples = 535 value = [629, 191] class = Not churn gini = 0.4 samples = 571 value = [241, 647] class = Churn gini = 0.46 samples = 852 value = [838, 458] class = Not churn gini = 0.43 samples = 369 value = [390, 175] class = Not churn gini = 0.12 samples = 166 value = [239, 16] class = Not churn

tenure <= 16.5 gini = 0.4 samples = 4417 value = [5105, 1927] class = Not churn PaperlessBilling <= 0.5 gini = 0.5 samples = 1597 value = [1339, 1193] class = Not churn True Contract\_Two year <= 0.5 gini = 0.27 samples = 2820 value = [3766, 734] class = Not churn False tenure <= 5.5 gini = 0.44 samples = 658 value = [687, 341] class = Not churn InternetService\_No <= 0.5 gini = 0.49 samples = 939 value = [652, 852] class = Churn gini = 0.49 samples = 365 value = [320, 258] class = Not churn gini = 0.3 samples = 293 value = [367, 83] class = Not churn gini = 0.48 samples = 838 value = [536, 818] class = Churn gini = 0.35 samples = 101 value = [116, 34] class = Not churn TechSupport <= 0.5 gini = 0.36 samples = 1844 value = [2244, 684] class = Not churn TechSupport <= 0.5 gini = 0.06 samples = 976 value = [1522, 50] class = Not churn gini = 0.39 samples = 1251 value = [1465, 530] class = Not churn gini = 0.28 samples = 593 value = [779, 154] class = Not churn gini = 0.03 samples = 485 value = [779, 13] class = Not churn gini = 0.09 samples = 491 value = [743, 37] class = Not churn

InternetService\_Fiber optic <= 0.5 gini = 0.4 samples = 4431 value = [5121, 1911] class = Not churn tenure <= 5.5 gini = 0.26 samples = 2485 value = [3334, 596] class = Not churn True tenure <= 17.5 gini = 0.49 samples = 1946 value = [1787, 1315] class = Not churn False tenure <= 1.5 gini = 0.47 samples = 514 value = [495, 314] class = Not churn InternetService\_No <= 0.5 gini = 0.16 samples = 1971 value = [2839, 282] class = Not churn gini = 0.5 samples = 245 value = [205, 184] class = Not churn gini = 0.43 samples = 269 value = [290, 130] class = Not churn gini = 0.23 samples = 1229 value = [1697, 254] class = Not churn gini = 0.05 samples = 742 value = [1142, 28] class = Not churn TechSupport <= 0.5 gini = 0.44 samples = 704 value = [369, 754] class = Churn tenure <= 43.5 gini = 0.41 samples = 1242 value = [1418, 561] class = Not churn gini = 0.43 samples = 631 value = [320, 681] class = Churn gini = 0.48 samples = 73 value = [49, 73] class = Churn gini = 0.49 samples = 531 value = [499, 362] class = Not churn gini = 0.29 samples = 711 value = [919, 199] class = Not churn

TotalCharges <= 229.65 gini = 0.39 samples = 4437 value = [5154, 1878] class = Not churn PaperlessBilling <= 0.5 gini = 0.5 samples = 809 value = [665, 629] class = Not churn True Contract\_Month-to-month <= 0.5 gini = 0.34 samples = 3628 value = [4489, 1249] class = Not churn False InternetService\_No <= 0.5 gini = 0.45 samples = 384 value = [396, 207] class = Not churn InternetService\_No <= 0.5 gini = 0.48 samples = 425 value = [269, 422] class = Churn gini = 0.5 samples = 183 value = [135, 143] class = Churn gini = 0.32 samples = 201 value = [261, 64] class = Not churn gini = 0.39 samples = 330 value = [139, 386] class = Churn gini = 0.34 samples = 95 value = [130, 36] class = Not churn InternetService\_Fiber optic <= 0.5 gini = 0.13 samples = 1926 value = [2791, 214] class = Not churn PaperlessBilling <= 0.5 gini = 0.47 samples = 1702 value = [1698, 1035] class = Not churn gini = 0.07 samples = 1317 value = [1976, 78] class = Not churn gini = 0.25 samples = 609 value = [815, 136] class = Not churn gini = 0.36 samples = 476 value = [604, 184] class = Not churn gini = 0.49 samples = 1226 value = [1094, 851] class = Not churn

InternetService\_Fiber optic <= 0.5 gini = 0.39 samples = 4435 value = [5170, 1862] class = Not churn TotalCharges <= 223.53 gini = 0.25 samples = 2479 value = [3393, 575] class = Not churn True tenure <= 17.5 gini = 0.49 samples = 1956 value = [1777, 1287] class = Not churn False InternetService\_No <= 0.5 gini = 0.46 samples = 559 value = [580, 314] class = Not churn Contract\_Month-to-month <= 0.5 gini = 0.16 samples = 1920 value = [2813, 261] class = Not churn gini = 0.5 samples = 270 value = [208, 223] class = Churn gini = 0.32 samples = 289 value = [372, 91] class = Not churn gini = 0.06 samples = 1329 value = [2054, 71] class = Not churn gini = 0.32 samples = 591 value = [759, 190] class = Not churn TotalCharges <= 120.0 gini = 0.43 samples = 705 value = [347, 772] class = Churn Contract\_Two year <= 0.5 gini = 0.39 samples = 1251 value = [1430, 515] class = Not churn gini = 0.18 samples = 156 value = [24, 215] class = Churn gini = 0.46 samples = 549 value = [323, 557] class = Churn gini = 0.44 samples = 985 value = [1033, 487] class = Not churn gini = 0.12 samples = 266 value = [397, 28] class = Not churn

Contract\_Two year <= 0.5 gini = 0.39 samples = 4437 value = [5142, 1890] class = Not churn InternetService\_No <= 0.5 gini = 0.45 samples = 3391 value = [3548, 1834] class = Not churn True tenure <= 70.5 gini = 0.07 samples = 1046 value = [1594, 56] class = Not churn False TotalCharges <= 545.83 gini = 0.48 samples = 2813 value = [2721, 1736] class = Not churn Contract\_Month-to-month <= 0.5 gini = 0.19 samples = 578 value = [827, 98] class = Not churn gini = 0.48 samples = 811 value = [516, 778] class = Churn gini = 0.42 samples = 2002 value = [2205, 958] class = Not churn gini = 0.04 samples = 235 value = [373, 8] class = Not churn gini = 0.28 samples = 343 value = [454, 90] class = Not churn InternetService\_No <= 0.5 gini = 0.08 samples = 771 value = [1153, 50] class = Not churn InternetService\_No <= 0.5 gini = 0.03 samples = 275 value = [441, 6] class = Not churn gini = 0.12 samples = 438 value = [643, 43] class = Not churn gini = 0.03 samples = 333 value = [510, 7] class = Not churn gini = 0.03 samples = 208 value = [333, 6] class = Not churn gini = 0.0 samples = 67 value = [108, 0] class = Not churn

Contract\_Two year <= 0.5 gini = 0.39 samples = 4482 value = [5182, 1850] class = Not churn InternetService\_Fiber optic <= 0.5 gini = 0.45 samples = 3410 value = [3578, 1809] class = Not churn True PhoneService <= 0.5 gini = 0.05 samples = 1072 value = [1604, 41] class = Not churn False OnlineSecurity <= 0.5 gini = 0.32 samples = 1694 value = [2140, 532] class = Not churn TechSupport <= 0.5 gini = 0.5 samples = 1716 value = [1438, 1277] class = Not churn gini = 0.34 samples = 1244 value = [1524, 420] class = Not churn gini = 0.26 samples = 450 value = [616, 112] class = Not churn gini = 0.5 samples = 1350 value = [1032, 1118] class = Churn gini = 0.4 samples = 366 value = [406, 159] class = Not churn gini = 0.0 samples = 107 value = [153, 0] class = Not churn InternetService\_Fiber optic <= 0.5 gini = 0.05 samples = 965 value = [1451, 41] class = Not churn gini = 0.03 samples = 693 value = [1047, 15] class = Not churn gini = 0.11 samples = 272 value = [404, 26] class = Not churn

**5.9. Gaussian Naive Bayes.**[**¶**](#3cqmetx)

In [33]:

**from** **sklearn.naive\_bayes** **import** GaussianNB  
gnb = GaussianNB(priors=**None**)  
  
telecom\_churn\_prediction\_alg(gnb,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y)

GaussianNB(priors=None, var\_smoothing=1e-09)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.90 0.74 0.81 1268  
 1 0.54 0.79 0.64 490  
  
 micro avg 0.75 0.75 0.75 1758  
 macro avg 0.72 0.77 0.73 1758  
weighted avg 0.80 0.75 0.77 1758  
  
Accuracy Score : 0.7542662116040956  
Area under curve : 0.7657921843816391

**5.10. Support Vector Machine**[**¶**](#1rvwp1q)

* “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space .where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes

In [34]:

**from** **sklearn.svm** **import** SVC  
  
*#Support vector classifier*  
*#using linear hyper plane*  
svc\_lin = SVC(C=1.0, cache\_size=200, class\_weight=**None**, coef0=0.0,  
 decision\_function\_shape='ovr', degree=3, gamma=1.0, kernel='linear',  
 max\_iter=-1, probability=**True**, random\_state=**None**, shrinking=**True**,  
 tol=0.001, verbose=**False**)  
  
cols = [i **for** i **in** telcom.columns **if** i **not** **in** Id\_col + target\_col]  
telecom\_churn\_prediction(svc\_lin,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 cols,"coefficients",threshold\_plot = **False**)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,  
 decision\_function\_shape='ovr', degree=3, gamma=1.0, kernel='linear',  
 max\_iter=-1, probability=True, random\_state=None, shrinking=True,  
 tol=0.001, verbose=False)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.91 0.73 0.81 1268  
 1 0.54 0.81 0.64 490  
  
 micro avg 0.75 0.75 0.75 1758  
 macro avg 0.72 0.77 0.73 1758  
weighted avg 0.80 0.75 0.76 1758  
  
Accuracy Score : 0.7508532423208191  
Area under curve : 0.7684349449559004

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**5.11. Tuning parameters for support vector machine**[**¶**](#4bvk7pj)

In [35]:

*#tuning parameters*  
*#Support vector classifier*  
*#using non-linear hyper plane("rbf")*  
  
svc\_rbf = SVC(C=1.0, kernel='rbf',   
 degree= 3, gamma=1.0,   
 coef0=0.0, shrinking=**True**,  
 probability=**True**,tol=0.001,  
 cache\_size=200, class\_weight=**None**,  
 verbose=**False**,max\_iter= -1,  
 random\_state=**None**)  
  
telecom\_churn\_prediction\_alg(svc\_rbf,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,threshold\_plot = **False**)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,  
 decision\_function\_shape='ovr', degree=3, gamma=1.0, kernel='rbf',  
 max\_iter=-1, probability=True, random\_state=None, shrinking=True,  
 tol=0.001, verbose=False)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.82 0.86 0.84 1268  
 1 0.58 0.51 0.55 490  
  
 micro avg 0.76 0.76 0.76 1758  
 macro avg 0.70 0.69 0.69 1758  
weighted avg 0.75 0.76 0.76 1758  
  
Accuracy Score : 0.7622298065984073  
Area under curve : 0.6855388527650809

**5.12. LightGBMClassifier**[**¶**](#2r0uhxc)

In [36]:

**from** **lightgbm** **import** LGBMClassifier  
  
lgbm\_c = LGBMClassifier(boosting\_type='gbdt', class\_weight=**None**, colsample\_bytree=1.0,  
 learning\_rate=0.5, max\_depth=7, min\_child\_samples=20,  
 min\_child\_weight=0.001, min\_split\_gain=0.0, n\_estimators=100,  
 n\_jobs=-1, num\_leaves=500, objective='binary', random\_state=**None**,  
 reg\_alpha=0.0, reg\_lambda=0.0, silent=**True**, subsample=1.0,  
 subsample\_for\_bin=200000, subsample\_freq=0)  
  
cols = [i **for** i **in** telcom.columns **if** i **not** **in** Id\_col + target\_col]  
telecom\_churn\_prediction(lgbm\_c,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 cols,"features",threshold\_plot = **True**)

LGBMClassifier(boosting\_type='gbdt', class\_weight=None, colsample\_bytree=1.0,  
 importance\_type='split', learning\_rate=0.5, max\_depth=7,  
 min\_child\_samples=20, min\_child\_weight=0.001, min\_split\_gain=0.0,  
 n\_estimators=100, n\_jobs=-1, num\_leaves=500, objective='binary',  
 random\_state=None, reg\_alpha=0.0, reg\_lambda=0.0, silent=True,  
 subsample=1.0, subsample\_for\_bin=200000, subsample\_freq=0)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.82 0.85 0.84 1268  
 1 0.58 0.53 0.55 490  
  
 micro avg 0.76 0.76 0.76 1758  
 macro avg 0.70 0.69 0.70 1758  
weighted avg 0.76 0.76 0.76 1758  
  
Accuracy Score : 0.7627986348122867  
Area under curve : 0.6909418657052726

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**5.13. XGBoost Classifier**[**¶**](#1664s55)

In [37]:

**from** **xgboost** **import** XGBClassifier  
  
xgc = XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,  
 colsample\_bytree=1, gamma=0, learning\_rate=0.9, max\_delta\_step=0,  
 max\_depth = 7, min\_child\_weight=1, missing=**None**, n\_estimators=100,  
 n\_jobs=1, nthread=**None**, objective='binary:logistic', random\_state=0,  
 reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, seed=**None**,  
 silent=**True**, subsample=1)  
  
  
telecom\_churn\_prediction(xgc,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 cols,"features",threshold\_plot = **True**)

XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,  
 colsample\_bynode=1, colsample\_bytree=1, gamma=0, learning\_rate=0.9,  
 max\_delta\_step=0, max\_depth=7, min\_child\_weight=1, missing=None,  
 n\_estimators=100, n\_jobs=1, nthread=None,  
 objective='binary:logistic', random\_state=0, reg\_alpha=0,  
 reg\_lambda=1, scale\_pos\_weight=1, seed=None, silent=True,  
 subsample=1, verbosity=1)  
  
 Classification report :   
 precision recall f1-score support  
  
 0 0.83 0.85 0.84 1268  
 1 0.59 0.53 0.56 490  
  
 micro avg 0.76 0.76 0.76 1758  
 macro avg 0.71 0.69 0.70 1758  
weighted avg 0.76 0.76 0.76 1758  
  
Accuracy Score : 0.764505119453925  
Area under curve : 0.6933770037983648

/Users/stevecoggeshall/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: DeprecationWarning:  
  
plotly.tools.make\_subplots is deprecated, please use plotly.subplots.make\_subplots instead

**6. Model Performances**[**¶**](#3q5sasy)

## **6.1. model performance metrics**[**¶**](#25b2l0r)

In [38]:

**from** **sklearn.metrics** **import** f1\_score  
**from** **sklearn.metrics** **import** cohen\_kappa\_score  
  
*#gives model report in dataframe*  
**def** model\_report(model,training\_x,testing\_x,training\_y,testing\_y,name) :  
 model.fit(training\_x,training\_y)  
 predictions = model.predict(testing\_x)  
 accuracy = accuracy\_score(testing\_y,predictions)  
 recallscore = recall\_score(testing\_y,predictions)  
 precision = precision\_score(testing\_y,predictions)  
 roc\_auc = roc\_auc\_score(testing\_y,predictions)  
 f1score = f1\_score(testing\_y,predictions)   
 kappa\_metric = cohen\_kappa\_score(testing\_y,predictions)  
   
 df = pd.DataFrame({"Model" : [name],  
 "Accuracy\_score" : [accuracy],  
 "Recall\_score" : [recallscore],  
 "Precision" : [precision],  
 "f1\_score" : [f1score],  
 "Area\_under\_curve": [roc\_auc],  
 "Kappa\_metric" : [kappa\_metric],  
 })  
 **return** df  
  
*#outputs for every model*  
model1 = model\_report(logit,train\_X,test\_X,train\_Y,test\_Y,  
 "Logistic Regression(Baseline\_model)")  
model2 = model\_report(logit\_smote,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "Logistic Regression(SMOTE)")  
model3 = model\_report(logit\_rfe,train\_rf\_X,test\_rf\_X,train\_rf\_Y,test\_rf\_Y,  
 "Logistic Regression(RFE)")  
decision\_tree = DecisionTreeClassifier(max\_depth = 9,  
 random\_state = 123,  
 splitter = "best",  
 criterion = "gini",  
 )  
model4 = model\_report(decision\_tree,train\_X,test\_X,train\_Y,test\_Y,  
 "Decision Tree")  
model5 = model\_report(knn,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "KNN Classifier")  
rfc = RandomForestClassifier(n\_estimators = 1000,  
 random\_state = 123,  
 max\_depth = 9,  
 criterion = "gini")  
model6 = model\_report(rfc,train\_X,test\_X,train\_Y,test\_Y,  
 "Random Forest Classifier")  
model7 = model\_report(gnb,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "Naive Bayes")  
model8 = model\_report(svc\_lin,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "SVM Classifier Linear")  
model9 = model\_report(svc\_rbf,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "SVM Classifier RBF")  
model10 = model\_report(lgbm\_c,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "LGBM Classifier")  
model11 = model\_report(xgc,os\_smote\_X,test\_X,os\_smote\_Y,test\_Y,  
 "XGBoost Classifier")  
  
*#concat all models*  
model\_performances = pd.concat([model1,model2,model3,  
 model4,model5,model6,  
 model7,model8,model9,  
 model10,model11],axis = 0).reset\_index()  
  
model\_performances = model\_performances.drop(columns = "index",axis =1)  
  
table = ff.create\_table(np.round(model\_performances,4))  
  
py.iplot(table)

**6.2. Compare model metrics**[**¶**](#kgcv8k)

In [39]:

model\_performances  
**def** output\_tracer(metric,color) :  
 tracer = go.Bar(y = model\_performances["Model"] ,  
 x = model\_performances[metric],  
 orientation = "h",name = metric ,  
 marker = dict(line = dict(width =.7),  
 color = color)  
 )  
 **return** tracer  
  
layout = go.Layout(dict(title = "Model performances",  
 plot\_bgcolor = "rgb(243,243,243)",  
 paper\_bgcolor = "rgb(243,243,243)",  
 xaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 title = "metric",  
 zerolinewidth=1,  
 ticklen=5,gridwidth=2),  
 yaxis = dict(gridcolor = 'rgb(255, 255, 255)',  
 zerolinewidth=1,ticklen=5,gridwidth=2),  
 margin = dict(l = 250),  
 height = 780  
 )  
 )  
  
  
trace1 = output\_tracer("Accuracy\_score","#6699FF")  
trace2 = output\_tracer('Recall\_score',"red")  
trace3 = output\_tracer('Precision',"#33CC99")  
trace4 = output\_tracer('f1\_score',"lightgrey")  
trace5 = output\_tracer('Kappa\_metric',"#FFCC99")  
  
data = [trace1,trace2,trace3,trace4,trace5]  
fig = go.Figure(data=data,layout=layout)  
py.iplot(fig)

**6.3. Confusion matrices for models**[**¶**](#34g0dwd)

In [40]:

lst = [logit,logit\_smote,decision\_tree,knn,rfc,  
 gnb,svc\_lin,svc\_rbf,lgbm\_c,xgc]  
  
length = len(lst)  
  
mods = ['Logistic Regression(Baseline\_model)','Logistic Regression(SMOTE)',  
 'Decision Tree','KNN Classifier','Random Forest Classifier',"Naive Bayes",  
 'SVM Classifier Linear','SVM Classifier RBF', 'LGBM Classifier',  
 'XGBoost Classifier']  
  
fig = plt.figure(figsize=(13,15))  
fig.set\_facecolor("#F3F3F3")  
**for** i,j,k **in** itertools.zip\_longest(lst,range(length),mods) :  
 plt.subplot(4,3,j+1)  
 predictions = i.predict(test\_X)  
 conf\_matrix = confusion\_matrix(predictions,test\_Y)  
 sns.heatmap(conf\_matrix,annot=**True**,fmt = "d",square = **True**,  
 xticklabels=["not churn","churn"],  
 yticklabels=["not churn","churn"],  
 linewidths = 2,linecolor = "w",cmap = "Set1")  
 plt.title(k,color = "b")  
 plt.subplots\_adjust(wspace = .3,hspace = .3)

**6.4. ROC - Curves for models**[**¶**](#1jlao46)

In [41]:

lst = [logit,logit\_smote,decision\_tree,knn,rfc,  
 gnb,svc\_lin,svc\_rbf,lgbm\_c,xgc]  
  
length = len(lst)  
  
mods = ['Logistic Regression(Baseline\_model)','Logistic Regression(SMOTE)',  
 'Decision Tree','KNN Classifier','Random Forest Classifier',"Naive Bayes",  
 'SVM Classifier Linear','SVM Classifier RBF', 'LGBM Classifier',  
 'XGBoost Classifier']  
  
plt.style.use("dark\_background")  
fig = plt.figure(figsize=(12,16))  
fig.set\_facecolor("#F3F3F3")  
**for** i,j,k **in** itertools.zip\_longest(lst,range(length),mods) :  
 qx = plt.subplot(4,3,j+1)  
 probabilities = i.predict\_proba(test\_X)  
 predictions = i.predict(test\_X)  
 fpr,tpr,thresholds = roc\_curve(test\_Y,probabilities[:,1])  
 plt.plot(fpr,tpr,linestyle = "dotted",  
 color = "royalblue",linewidth = 2,  
 label = "AUC = " + str(np.around(roc\_auc\_score(test\_Y,predictions),3)))  
 plt.plot([0,1],[0,1],linestyle = "dashed",  
 color = "orangered",linewidth = 1.5)  
 plt.fill\_between(fpr,tpr,alpha = .4)  
 plt.fill\_between([0,1],[0,1],color = "k")  
 plt.legend(loc = "lower right",  
 prop = {"size" : 12})  
 qx.set\_facecolor("k")  
 plt.grid(**True**,alpha = .15)  
 plt.title(k,color = "b")  
 plt.xticks(np.arange(0,1,.3))  
 plt.yticks(np.arange(0,1,.3))

**6.5. Precision recall curves**[**¶**](#43ky6rz)

In [42]:

**from** **sklearn.metrics** **import** precision\_recall\_curve  
**from** **sklearn.metrics** **import** average\_precision\_score  
  
  
lst = [logit,logit\_smote,decision\_tree,knn,rfc,  
 gnb,svc\_lin,svc\_rbf,lgbm\_c,xgc]  
  
length = len(lst)  
  
mods = ['Logistic Regression(Baseline\_model)','Logistic Regression(SMOTE)',  
 'Decision Tree','KNN Classifier','Random Forest Classifier',"Naive Bayes",  
 'SVM Classifier Linear','SVM Classifier RBF', 'LGBM Classifier',  
 'XGBoost Classifier']  
  
fig = plt.figure(figsize=(13,17))  
fig.set\_facecolor("#F3F3F3")  
**for** i,j,k **in** itertools.zip\_longest(lst,range(length),mods) :  
   
 qx = plt.subplot(4,3,j+1)  
 probabilities = i.predict\_proba(test\_X)  
 predictions = i.predict(test\_X)  
 recall,precision,thresholds = precision\_recall\_curve(test\_Y,probabilities[:,1])  
 plt.plot(recall,precision,linewidth = 1.5,  
 label = ("avg\_pcn : " +   
 str(np.around(average\_precision\_score(test\_Y,predictions),3))))  
 plt.plot([0,1],[0,0],linestyle = "dashed")  
 plt.fill\_between(recall,precision,alpha = .2)  
 plt.legend(loc = "lower left",  
 prop = {"size" : 10})  
 qx.set\_facecolor("k")  
 plt.grid(**True**,alpha = .15)  
 plt.title(k,color = "b")  
 plt.xlabel("recall",fontsize =7)  
 plt.ylabel("precision",fontsize =7)  
 plt.xlim([0.25,1])  
 plt.yticks(np.arange(0,1,.3))

In [43]:

print('duration: ',pd.datetime.now() - start\_time)

duration: 0:03:39.956246