Project Telecom

June 19, 2024

1 Problem Statement:

In the telecom industry, customers are able to choose from a pool of companies to cater their needs regarding communication and internet. Customers are very critical about the kind of services they receive and judge the enitre company based on a single experience! These communication services have become so recurrent and inseparable from the daily routine that a 30 minute maintenance break kicks in anxiety in the users highlighting our taken-for-granted attitude towards these services! Coupled with high customer acquisation costs, churn analysis becomes very pivotal! Churn rate is a metric that describes the number of customers that cancelled or did not renew their subscription with the company. Thus, higher the churn rate, more customers stop buying from your business, directly affecting the revenue! Hence, based on the insights gained from the churn analysis, companies can build strategies, target segments, improve the quality of the services being provided to improve the customer experience, thus cultivating trust with the customers. That is why building predictive models and creating reports of churn analysis becomes key that paves the way for growth!

2 Aim:

1. To classify the potential churn customers based on numerical and categorical features.
2. To classify whether the problem dataset is a binary classification problem for an imbalanced dataset.

2.0.1 Dataset Attributes:

- **customerID** : Customer ID
- **gender** : Whether the customer is a male or a female
- **SeniorCitizen** : Whether the customer is a senior citizen or not (1, 0)
- **Partner** : Whether the customer has a partner or not (Yes, No)
- **Dependents** : Whether the customer has dependents or not (Yes, No)
- **tenure** : Number of months the customer has stayed with the company
- **PhoneService** : Whether the customer has a phone service or not (Yes, No)
- **MultipleLines** : Whether the customer has multiple lines or not (Yes, No, No phone service)
- **InternetService** : Customer's internet service provider (DSL, Fiber optic, No)
- **OnlineSecurity** : Whether the customer has online security or not (Yes, No, No internet service)
- **OnlineBackup** : Whether the customer has online backup or not (Yes, No, No internet service)

- **DeviceProtection** : Whether the customer has device protection or not (Yes, No, No internet service)
- **TechSupport** : Whether the customer has tech support or not (Yes, No, No internet service)
- **StreamingTV** : Whether the customer has streaming TV or not (Yes, No, No internet service)
- **StreamingMovies** : Whether the customer has streaming movies or not (Yes, No, No internet service)
- **Contract** : The contract term of the customer (Month-to-month, One year, Two year)
- **PaperlessBilling** : Whether the customer has paperless billing or not (Yes, No)
- **PaymentMethod** : The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
- **MonthlyCharges** : The amount charged to the customer monthly
- **TotalCharges** : The total amount charged to the customer
- **Churn** : Whether the customer churned or not (Yes or No)

- We start with impoting basic libraries required for analysis of the dataset.

[1]:
```
# Importing basic analytical libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import dtale
%matplotlib inline
sns.set()
import warnings
warnings.filterwarnings("ignore")
pd.options.display.float_format = '{:.2f}'.format
pd.options.display.max_columns = None
```

- Importing the dataset in Jupyter Notebook

[2]:
```
# Importing dataset
df = pd.read_csv("Telecom_Customer_Details.csv")
df.head()
```

[2]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No
2                No             DSL            Yes          Yes
3  No phone service             DSL            Yes           No
```

```
4                 No     Fiber optic             No            No

  DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0               No          No          No              No  Month-to-month
1              Yes          No          No              No        One year
2               No          No          No              No  Month-to-month
3              Yes         Yes          No              No        One year
4               No          No          No              No  Month-to-month

  PaperlessBilling              PaymentMethod  MonthlyCharges TotalCharges  \
0              Yes           Electronic check           29.85        29.85
1               No               Mailed check           56.95       1889.5
2              Yes               Mailed check           53.85       108.15
3               No  Bank transfer (automatic)           42.30      1840.75
4              Yes           Electronic check           70.70       151.65

  Churn
0    No
1    No
2   Yes
3    No
4   Yes
```

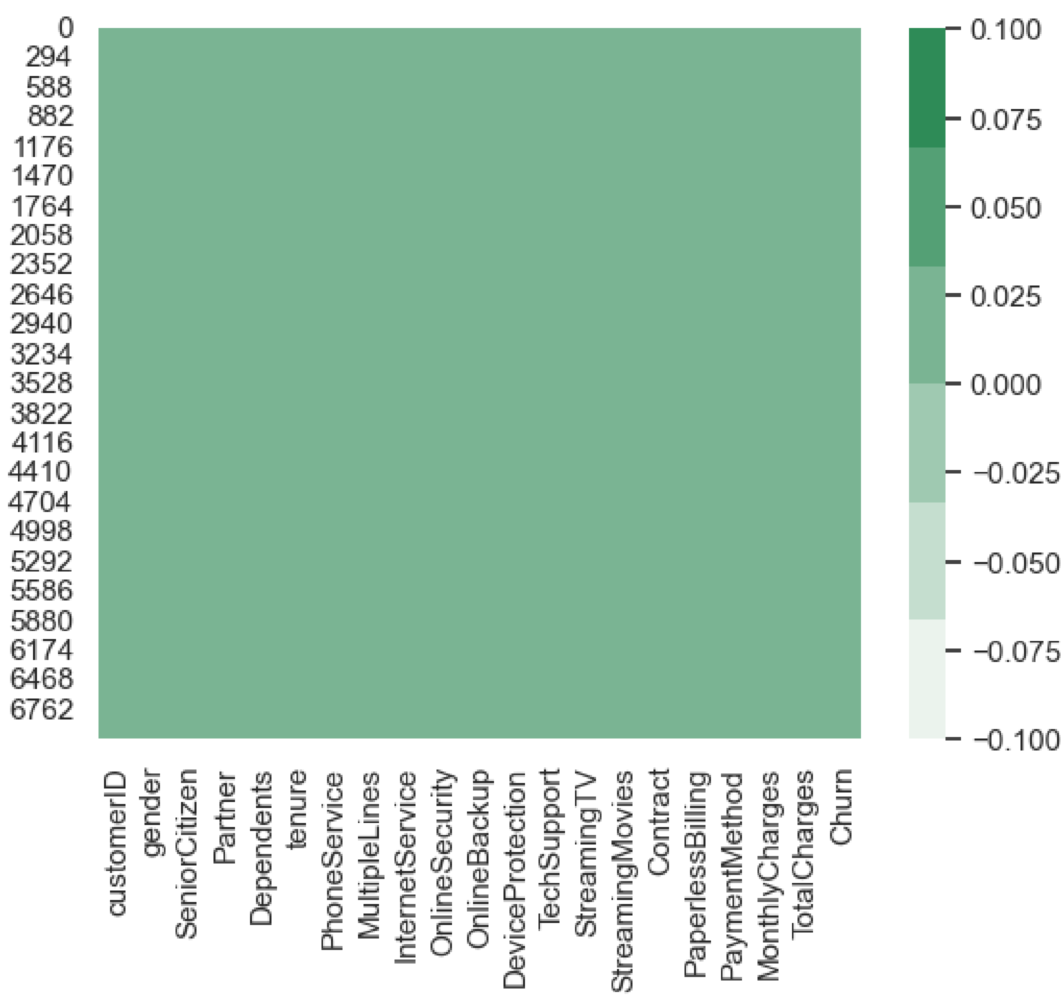
- Checking for null values, datatypes, total information (rows) and features

[3]:
```python
# Checking info of dataset
print("*******************************************************")
print('\t\tDataset Basic Information')
print("*******************************************************")
df.info()
```

```
*******************************************************
                Dataset Basic Information
*******************************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
```

```
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[4]:
```python
# Checking for null values in dataset
sns.heatmap(df.isnull(),cmap=sns.light_palette('seagreen'))
plt.show()
```



- ***Above heatmap shows that there are NO null vales in the above dataset***

[5]:
```python
print("******************************************************")
print('Total Rows\t\tTotalColumns')
print("******************************************************")
print('  ',df.shape[0],'\t\t   ',df.shape[1])
```

```
******************************************************
Total Rows              TotalColumns
******************************************************
   7043                     21
```

[6]:
```python
print("******************************************************")
print('Rows\t\t   Data_Types')
print("******************************************************")
df.dtypes
```

```
******************************************************
Rows                  Data_Types
******************************************************
```

[6]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

[7]:
```python
# Replacing incorrect dtype for MonthlyCharges from 'object' to 'float'
df['TotalCharges'].replace(to_replace=' ',value=np.nan,inplace=True)
```

```
df['TotalCharges'] = df['TotalCharges'].astype('float')
df['TotalCharges'].value_counts()
```

[7]:
```
TotalCharges
20.20      11
19.75       9
20.05       8
19.90       8
19.65       8
           ..
6849.40     1
692.35      1
130.15      1
3211.90     1
6844.50     1
Name: count, Length: 6530, dtype: int64
```

[8]:
```
# %age of missing values
print('Missing values = {:.2f}%'.format(df.TotalCharges.isna().sum()/
 ↪len(df)*100))
```

```
Missing values = 0.16%
```

- Above data shows we currently have 0.16% of the missing data which we may take the liberty to ignore.
- Thumb rule that I follow is if >30% of missing data is there in dataset, drop the feature.

[9]:
```
# Drop missing values
df.dropna(inplace=True)
```

- Customer ID are unique numbers and show no pattern to the trend or analysis.

[10]:
```
# Drop customer ID from main data
df = df.drop('customerID', axis=1)
```

[11]:
```
# Rechecking null values
df.isnull().sum()
```

[11]:
```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
```

```
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

[12]:
```
# Create a copy of dataframe
df1 = df.copy()
```

- Model can only take numerics, therefore character fields must be converted to numerics so machine may understand.
- Here I have used label encoding.

[13]:
```
# Sorting categorical columns out of numerical columns for label encoding
text_data_features = [i for i in df1.columns if i not in df1.describe().columns]
print(text_data_features)
```

```
['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
'PaymentMethod', 'Churn']
```

[14]:
```
# Label Encoding categorical columns
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in text_data_features:
    df1[i] = le.fit_transform(df1[i])
    print(i,':',df1[i].unique(),'=',le.inverse_transform(df1[i].unique()))
```

```
gender : [0 1] = ['Female' 'Male']
Partner : [1 0] = ['Yes' 'No']
Dependents : [0 1] = ['No' 'Yes']
PhoneService : [0 1] = ['No' 'Yes']
MultipleLines : [1 0 2] = ['No phone service' 'No' 'Yes']
InternetService : [0 1 2] = ['DSL' 'Fiber optic' 'No']
OnlineSecurity : [0 2 1] = ['No' 'Yes' 'No internet service']
OnlineBackup : [2 0 1] = ['Yes' 'No' 'No internet service']
DeviceProtection : [0 2 1] = ['No' 'Yes' 'No internet service']
TechSupport : [0 2 1] = ['No' 'Yes' 'No internet service']
StreamingTV : [0 2 1] = ['No' 'Yes' 'No internet service']
StreamingMovies : [0 2 1] = ['No' 'Yes' 'No internet service']
Contract : [0 1 2] = ['Month-to-month' 'One year' 'Two year']
PaperlessBilling : [1 0] = ['Yes' 'No']
PaymentMethod : [2 3 0 1] = ['Electronic check' 'Mailed check' 'Bank transfer
```

```
(automatic)'
 'Credit card (automatic)']
Churn : [0 1] = ['No' 'Yes']
```

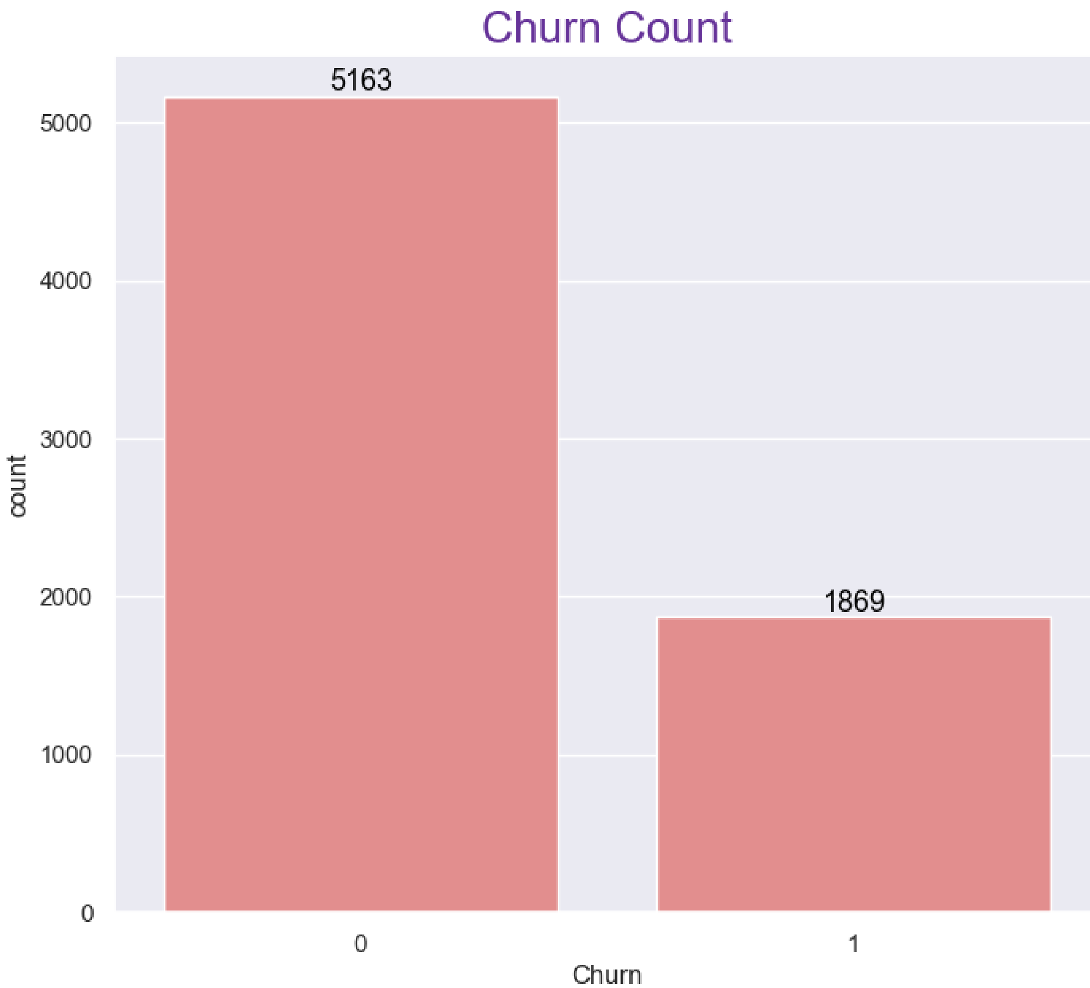
- Statistical description of the whole dataset.

[15]:
```
# Describing data after labelEncoding
df1.describe().T
```

[15]:

	count	mean	std	min	25%	50%	75%	max
gender	7032.00	0.50	0.50	0.00	0.00	1.00	1.00	1.00
SeniorCitizen	7032.00	0.16	0.37	0.00	0.00	0.00	0.00	1.00
Partner	7032.00	0.48	0.50	0.00	0.00	0.00	1.00	1.00
Dependents	7032.00	0.30	0.46	0.00	0.00	0.00	1.00	1.00
tenure	7032.00	32.42	24.55	1.00	9.00	29.00	55.00	72.00
PhoneService	7032.00	0.90	0.30	0.00	1.00	1.00	1.00	1.00
MultipleLines	7032.00	0.94	0.95	0.00	0.00	1.00	2.00	2.00
InternetService	7032.00	0.87	0.74	0.00	0.00	1.00	1.00	2.00
OnlineSecurity	7032.00	0.79	0.86	0.00	0.00	1.00	2.00	2.00
OnlineBackup	7032.00	0.91	0.88	0.00	0.00	1.00	2.00	2.00
DeviceProtection	7032.00	0.90	0.88	0.00	0.00	1.00	2.00	2.00
TechSupport	7032.00	0.80	0.86	0.00	0.00	1.00	2.00	2.00
StreamingTV	7032.00	0.98	0.89	0.00	0.00	1.00	2.00	2.00
StreamingMovies	7032.00	0.99	0.89	0.00	0.00	1.00	2.00	2.00
Contract	7032.00	0.69	0.83	0.00	0.00	0.00	1.00	2.00
PaperlessBilling	7032.00	0.59	0.49	0.00	0.00	1.00	1.00	1.00
PaymentMethod	7032.00	1.57	1.07	0.00	1.00	2.00	2.00	3.00
MonthlyCharges	7032.00	64.80	30.09	18.25	35.59	70.35	89.86	118.75
TotalCharges	7032.00	2283.30	2266.77	18.80	401.45	1397.47	3794.74	8684.80
Churn	7032.00	0.27	0.44	0.00	0.00	0.00	1.00	1.00

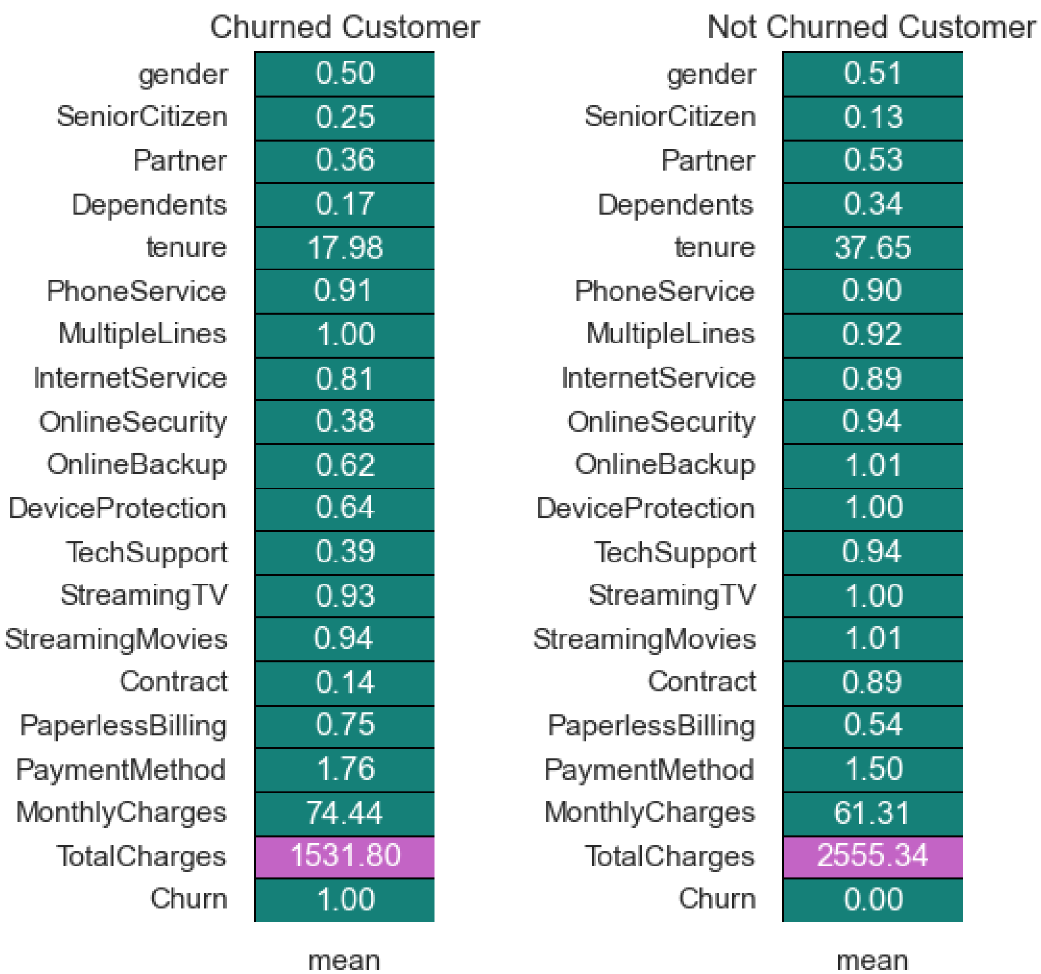
[16]:
```
# Churn column segregation/counts
plt.figure(figsize=(8, 7))
plots = sns.barplot(df1.Churn.value_counts(),color='lightcoral')
for bar in plots.patches:
    plots.annotate(int(bar.get_height()),
                   (bar.get_x() + bar.get_width() / 2,
                    bar.get_height()), ha='center', va='bottom',
                   size=13,color='black')
plt.title("Churn Count",fontdict={'fontsize':20,'color':'rebeccapurple'})
plt.show()
```



- While checking for the total value counts of the dependant variable (Churn) we observe the dataset is imbalanced.

[17]:
```
# Visualising mean of features for Churn and Not_Churn customers through heatmap
colors = ['#158078', '#C364C5']

churn = df1[df1['Churn']==1].describe().T
not_churn = df1[df1['Churn']==0].describe().T

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(6,6))
plt.subplot(1,2,1)
sns.heatmap(churn[['mean']], annot=True, cmap=colors, linewidths=0.4, linecolor␣
↪= 'black',
            cbar=False, fmt='.2f')

plt.title('Churned Customer')
```

```
plt.subplot(1,2,2)
sns.heatmap(not_churn[['mean']], annot=True, cmap=colors, linewidths=0.4,␣
 ↪linecolor = 'black',
            cbar=False, fmt='.2f')
plt.title('Not Churned Customer')

fig.tight_layout(pad=3)
```



3 Exploratory Data Analysis

[18]:
```
df.head()
```

[18]:
```
   gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  Female              0     Yes         No       1           No
```

```
1    Male              0      No          No      34          Yes
2    Male              0      No          No       2          Yes
3    Male              0      No          No      45           No
4  Female              0      No          No       2          Yes

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No
2                No             DSL            Yes          Yes
3  No phone service             DSL            Yes           No
4                No     Fiber optic             No           No

  DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0               No          No          No              No  Month-to-month
1              Yes          No          No              No        One year
2               No          No          No              No  Month-to-month
3              Yes         Yes          No              No        One year
4               No          No          No              No  Month-to-month

  PaperlessBilling              PaymentMethod  MonthlyCharges  TotalCharges  \
0              Yes           Electronic check           29.85         29.85
1               No               Mailed check           56.95       1889.50
2              Yes               Mailed check           53.85        108.15
3               No  Bank transfer (automatic)           42.30       1840.75
4              Yes           Electronic check           70.70        151.65

  Churn
0    No
1    No
2   Yes
3    No
4   Yes
```

[19]:

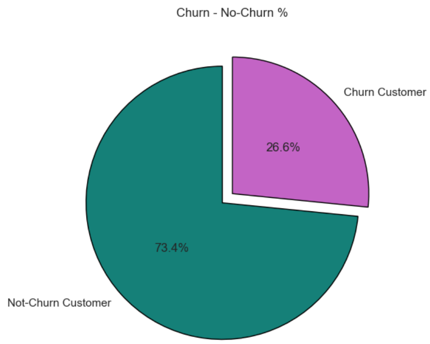
```python
# Segregating cat and numerical featurns in separate lists
col = list(df1.columns)
categorical_features = []
numerical_features = []

for i in col:
    if len(df[i].unique()) >6:
        numerical_features.append(i)
    else:
        categorical_features.append(i)
print('xxxxxxxxxxxxxxxxxxxxxxxx Categorical Features␣
↪xxxxxxxxxxxxxxxxxxxxxxxxxx')
print()
print('categorical_features :', *categorical_features)
```

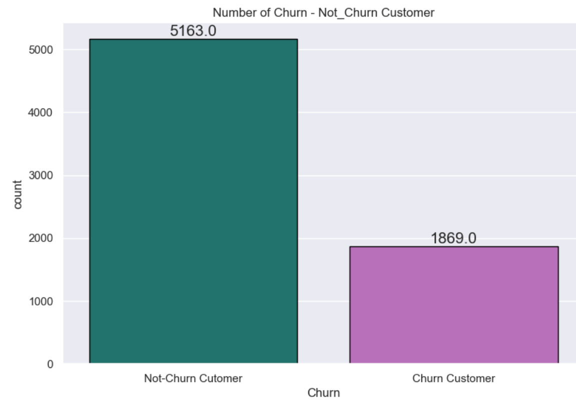
```
print()
print('xxxxxxxxxxxxxxxxxxxxxxxx Numerical Features␣
 ↪xxxxxxxxxxxxxxxxxxxxxxxxxxxx')
print()
print("numerical_features :", *numerical_features)
```

```
xxxxxxxxxxxxxxxxxxxxxxxx Categorical Features xxxxxxxxxxxxxxxxxxxxxxxxxx

categorical_features : gender SeniorCitizen Partner Dependents PhoneService
MultipleLines InternetService OnlineSecurity OnlineBackup DeviceProtection
TechSupport StreamingTV StreamingMovies Contract PaperlessBilling PaymentMethod
Churn

xxxxxxxxxxxxxxxxxxxxxxxx Numerical Features xxxxxxxxxxxxxxxxxxxxxxxxxxxx

numerical_features : tenure MonthlyCharges TotalCharges
```

[20]:

```
l = list(df1['Churn'].value_counts())
circle = l[0]/sum(l)*100, l[1]/sum(l)*100

fig = plt.subplots(nrows=1,ncols=2,figsize=(20,6))
plt.subplot(1,2,1)
plt.pie(circle, labels = ['Not-Churn Customer', 'Churn Customer'], autopct =␣
 ↪'%1.1f%%',pctdistance=0.5, startangle=90,
        explode = (0.1,0), colors = colors, wedgeprops  = {'edgecolor' :␣
 ↪'black',
                                                         'linewidth':1,␣
 ↪'antialiased' : True})
plt.title('Churn - No-Churn %')
plt.subplot(1,2,2)
ax = sns.countplot(x='Churn', data = df, palette = colors, edgecolor = 'black')

for rect1 in ax.patches:
    ax.text(rect1.get_x() + rect1.get_width() / 2, rect1.get_height() + 2,␣
 ↪rect1.get_height(),
            ha = 'center',va = 'bottom' , fontsize=15)
ax.set_xticklabels(['Not-Churn Cutomer', 'Churn Customer'])

plt.title('Number of Churn - Not_Churn Customer');
plt.show()
```





3.0.1 We have segregated the features based upon the below three cases.

- Case 1 : Customer information
- Case 2 : Payment information
- Case 3 : Service Subscribed

[21]:
```
# Categorical features
categorical_features
```

[21]:
```
['gender',
 'SeniorCitizen',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'Churn']
```

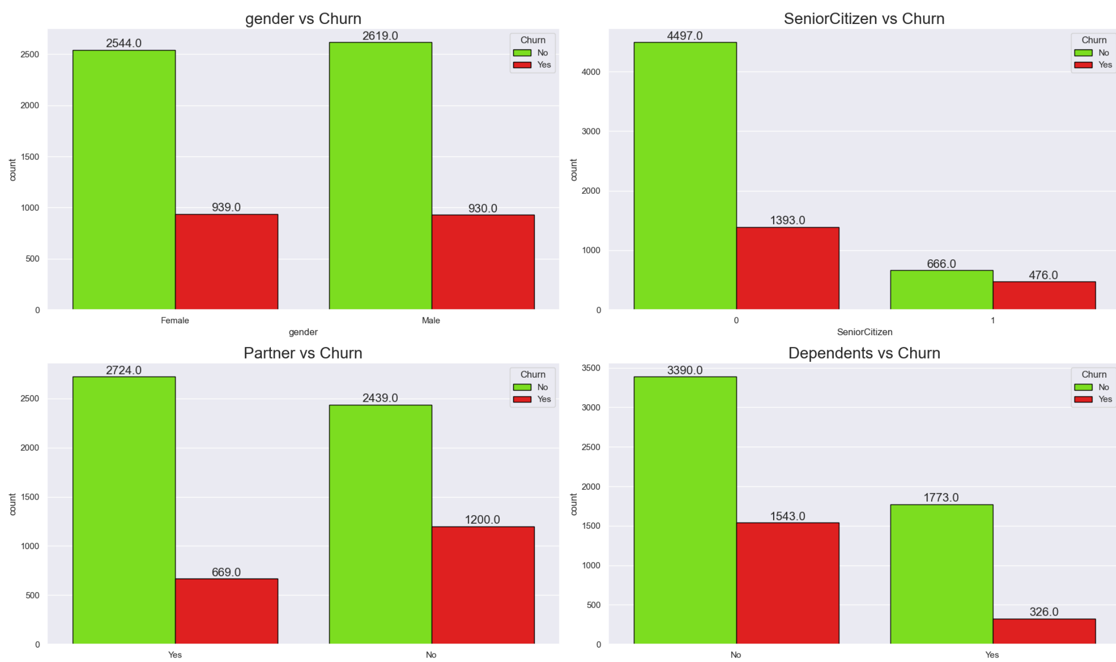
[22]:
```
# Removing churn data from categorical features
categorical_features.remove('Churn')
```

[23]:
```
# Creating list of Customer information, Payment information and Services␣
 ↪Subscribed
l1 = ['gender', 'SeniorCitizen', 'Partner', 'Dependents']
```

```
l2 = ['PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
      'OnlineBackup',  'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies']
l3 = ['Contract',  'PaperlessBilling',  'PaymentMethod']
```

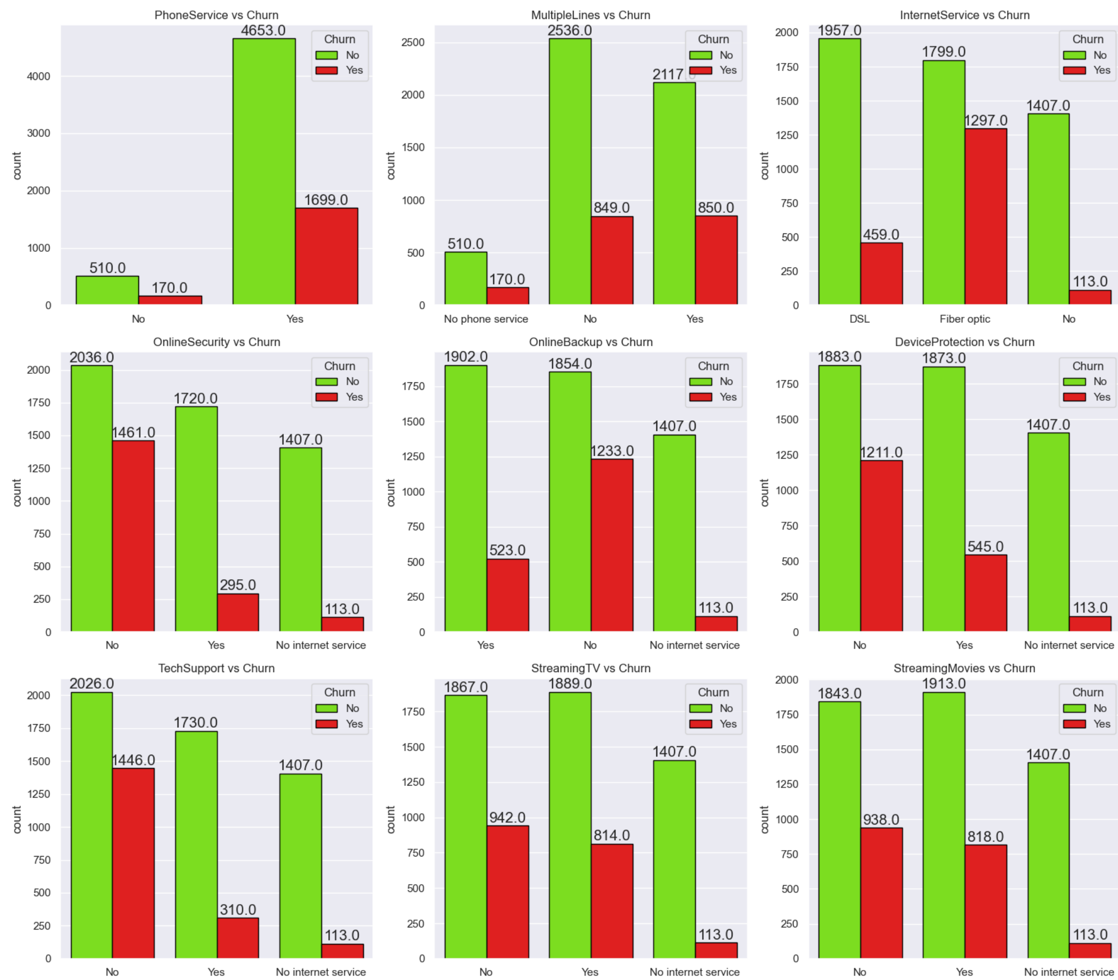
3.0.2 *Below visualization is the comparison count of all the Features vs Churn which is our dependant variable.*

[24]:
```
colors=['lawngreen','red']
fig  = plt.subplots(nrows=2, ncols=2, figsize=(20,12))
for i in range(len(l1)):
    plt.subplot(2,2,i+1)
    ax = sns.countplot(x=l1[i], data=df, hue='Churn', palette=colors,␣
 ↪edgecolor='black')
    for rect in ax.patches:
        if rect.get_xy() != (0,0):
            ax.text(rect.get_x() + rect.get_width()/2,rect.get_height() + 2,␣
 ↪rect.get_height(),
                ha='center',va='bottom',fontdict={'fontsize':15})
            title = l1[i] + ' vs Churn'
            plt.title(title,fontdict={'fontsize':20});
            plt.tight_layout();
```



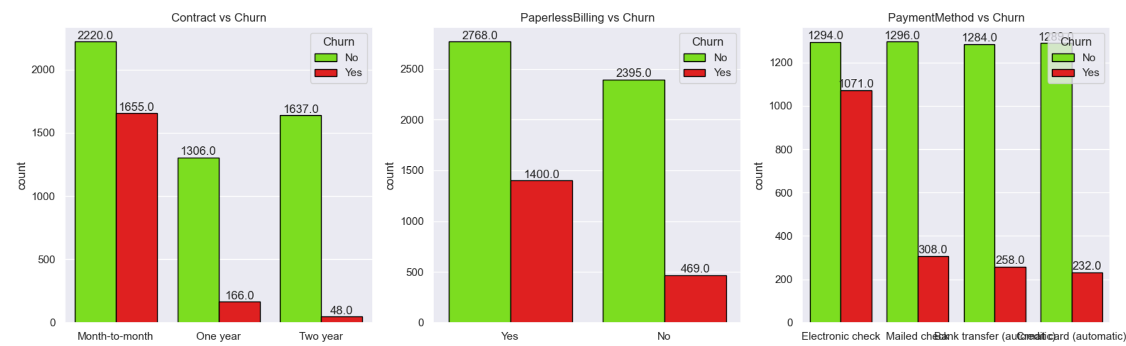
[25]:
```
fig  = plt.subplots(nrows=3, ncols=3, figsize=(16,14))
for i in range(len(l2)):
```

```
    plt.subplot(3,3,i+1)
    ax = sns.countplot(x=l2[i], data=df, hue='Churn', palette=colors,␣
↪edgecolor='black')
    for rect in ax.patches:
        if rect.get_xy() != (0,0):
            ax.text(rect.get_x() + rect.get_width()/2,rect.get_height() + 2,␣
↪rect.get_height(),
                    ha='center',va='bottom',fontdict={'fontsize':15})
            title = l2[i] + ' vs Churn'
            plt.title(title)
            plt.xlabel('')
            plt.tight_layout();
```



[26]:
```
fig  = plt.subplots(nrows=1, ncols=3, figsize=(16,5))
for i in range(len(l3)):
    plt.subplot(1,3,i+1)
```

```
    ax = sns.countplot(x=l3[i], data=df, hue='Churn', palette=colors,␣
↪edgecolor='black')
    for rect in ax.patches:
        if rect.get_xy() != (0,0):
            ax.text(rect.get_x() + rect.get_width()/2,rect.get_height() + 2,␣
↪rect.get_height(),
                    ha='center',va='bottom')
            title = l3[i] + ' vs Churn'
            plt.title(title)
            plt.xlabel('')
            plt.tight_layout();
```

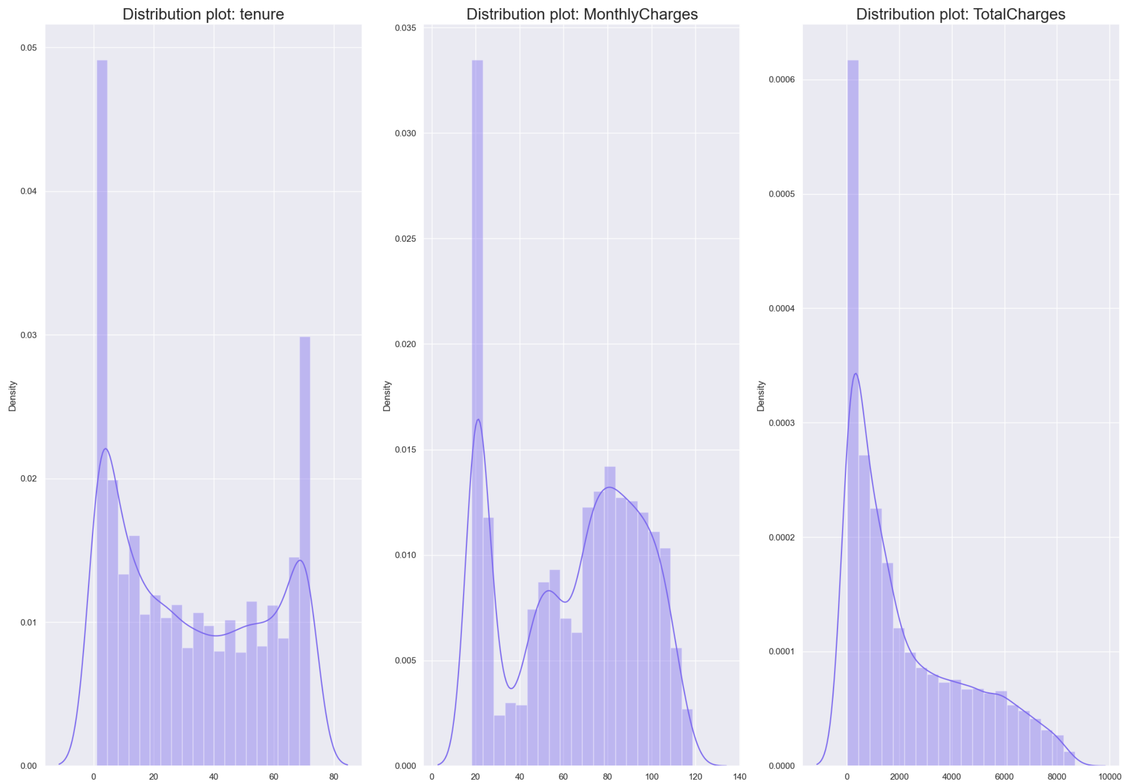


[27]:
```
# Numerical features
numerical_features
```

[27]: ['tenure', 'MonthlyCharges', 'TotalCharges']

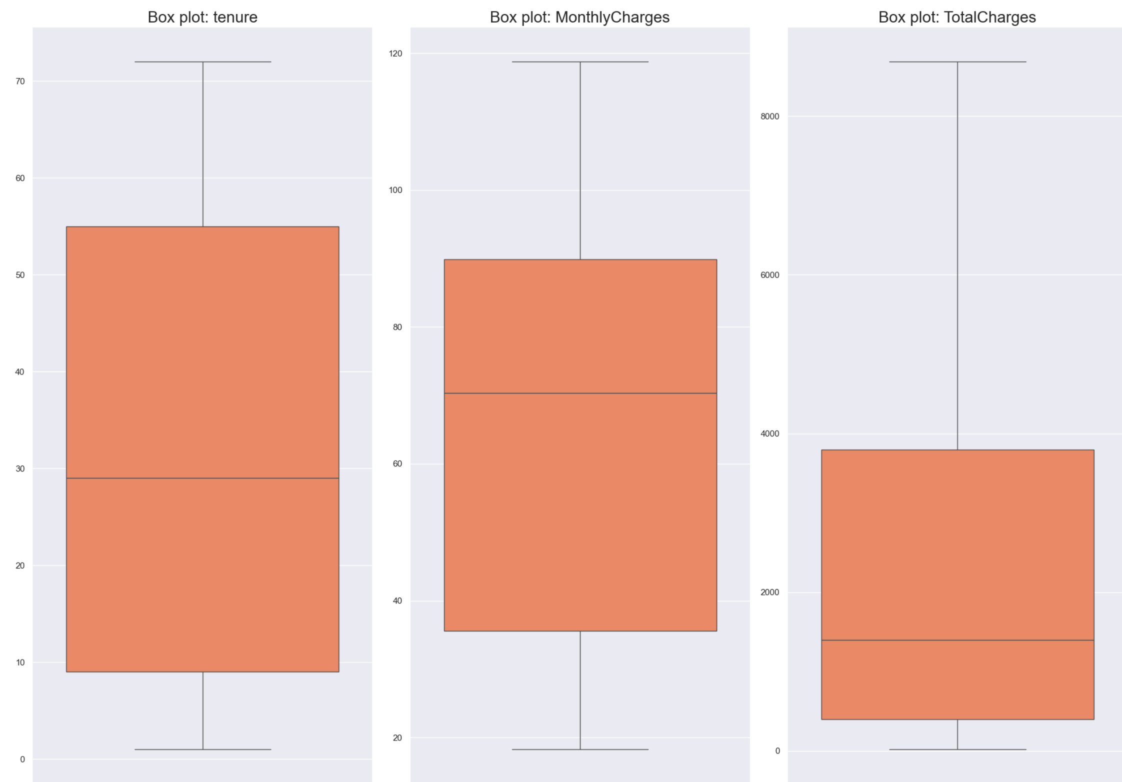
- **Analysis of the normal distribution for numerical features using distribution plot.**

[28]:
```
# Analysing normal distribution using Distplot
fig = plt.subplots(nrows=1,ncols=3,figsize=(20,14))
for i in range(len(numerical_features)):
    plt.subplot(1,3,i+1)
    sns.distplot(df[[numerical_features[i]]],color='mediumslateblue',bins=20)
    title = 'Distribution plot: '+numerical_features[i]
    plt.title(title,fontdict={'fontsize':20})
    plt.tight_layout()
plt.show()
```



- As we analyse from the above distribution plot, none of the numerical features are normally distributed

[29]:
```
# Detecting outliers using Box Plot
fig = plt.subplots(nrows=1,ncols=3,figsize=(20,14))
for i in range(len(numerical_features)):
    plt.subplot(1,3,i+1)
    sns.boxplot(df[[numerical_features[i]]],color='coral',whis=1.5)
    title = 'Box plot: '+numerical_features[i]
    plt.xticks([])
    plt.title(title,fontdict={'fontsize':20})
    plt.tight_layout()
plt.show()
```



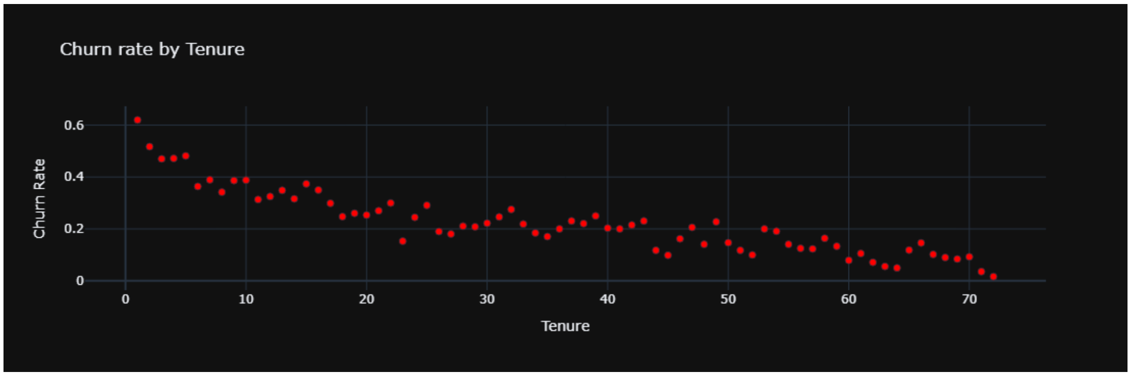
- There are no outliers present while analysing the numerical data.

[30]:
```
# Histogram by Churn/No-Churn
fig,ax = plt.subplots(nrows=3,ncols=1,figsize=(20,14))
for i in range(len(numerical_features)):
    plt.subplot(3,1,i+1)
    sns.
    ↪histplot(data=df,x=numerical_features[i],hue='Churn',multiple='stack',palette=colors,edgeco
    plt.legend(['Churn','No Churn'],loc='upper center')
    title = numerical_features[i] + ' w.r.t churn'
    plt.title(title,fontdict={'fontsize':15})
    plt.xlabel(xlabel="")
    # plt.tight_layout()
plt.show()
```

[31]:
```python
# Import Plotly module
import plotly.graph_objs as go
import plotly.express as px
import plotly.offline as po
```

[32]:
```python
# Grouping data by tenure
tenure_chunk = df1.groupby(by='tenure')[['Churn']].mean().reset_index()
```

[33]:
```python
# Churn Rate by Tenure
fig = go.Figure(data=[go.Scatter(x = tenure_chunk['tenure'],
                                 y = tenure_chunk['Churn'], mode = 'markers',
                                 name='Low', marker = dict(size=7,
                                                           line=dict(width=0.8),
                                                           color='red'))])

fig.update_layout(title = "Churn rate by Tenure", xaxis_title = "Tenure",
                  yaxis_title="Churn Rate", template = "plotly_dark")
fig.show()
```

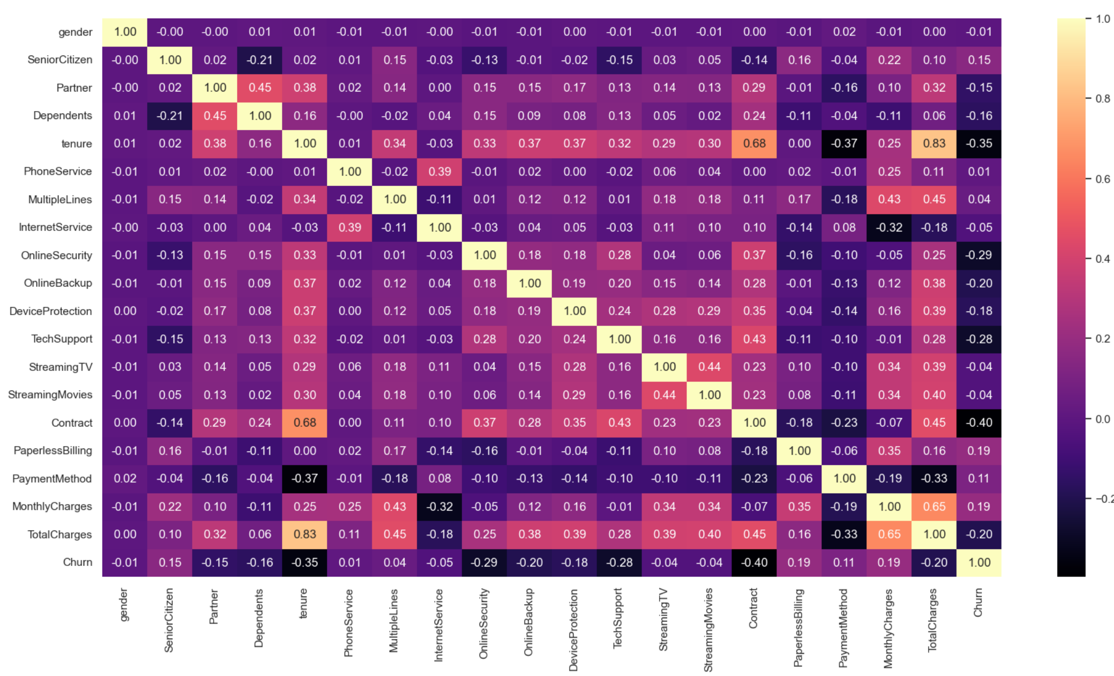


- Standadizing the numerical features

[34]:
```
# Standard scaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df1[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.
↪fit_transform(df1[['tenure', 'MonthlyCharges', 'TotalCharges']])
```

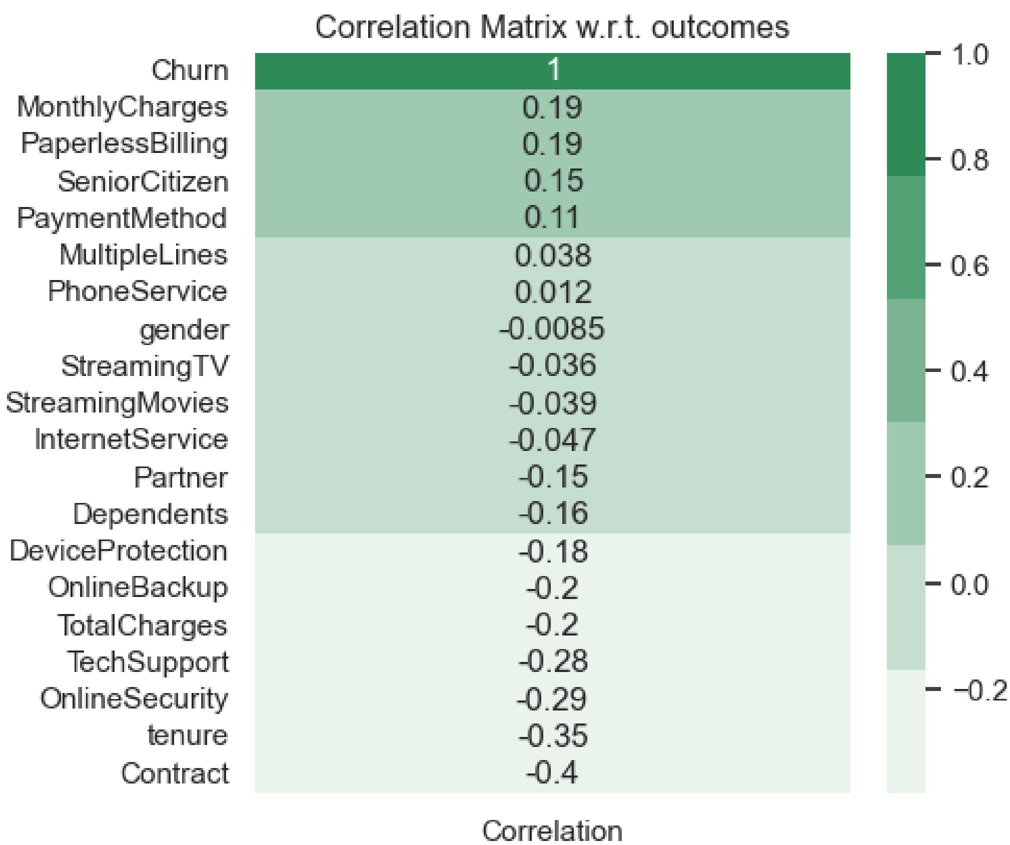
- **Observing the correlation of each of the features in the dataset using heatmap**

[35]:
```
# Correlation dataset
sns.heatmap(df1.corr(),cmap='magma',annot=True,figure = plt.
↪figure(figsize=(20,10)),fmt='.2f');
```



- Correlation of independent features vs dependent features

[36]:
```
corr = df1.corrwith(df1['Churn']).sort_values(ascending = False).to_frame()
corr.columns = ['Correlation']
plt.subplots(figsize = (5,5))
sns.heatmap(corr, annot=True, cmap=sns.light_palette('seagreen'))
plt.title("Correlation Matrix w.r.t. outcomes")
plt.show()
```



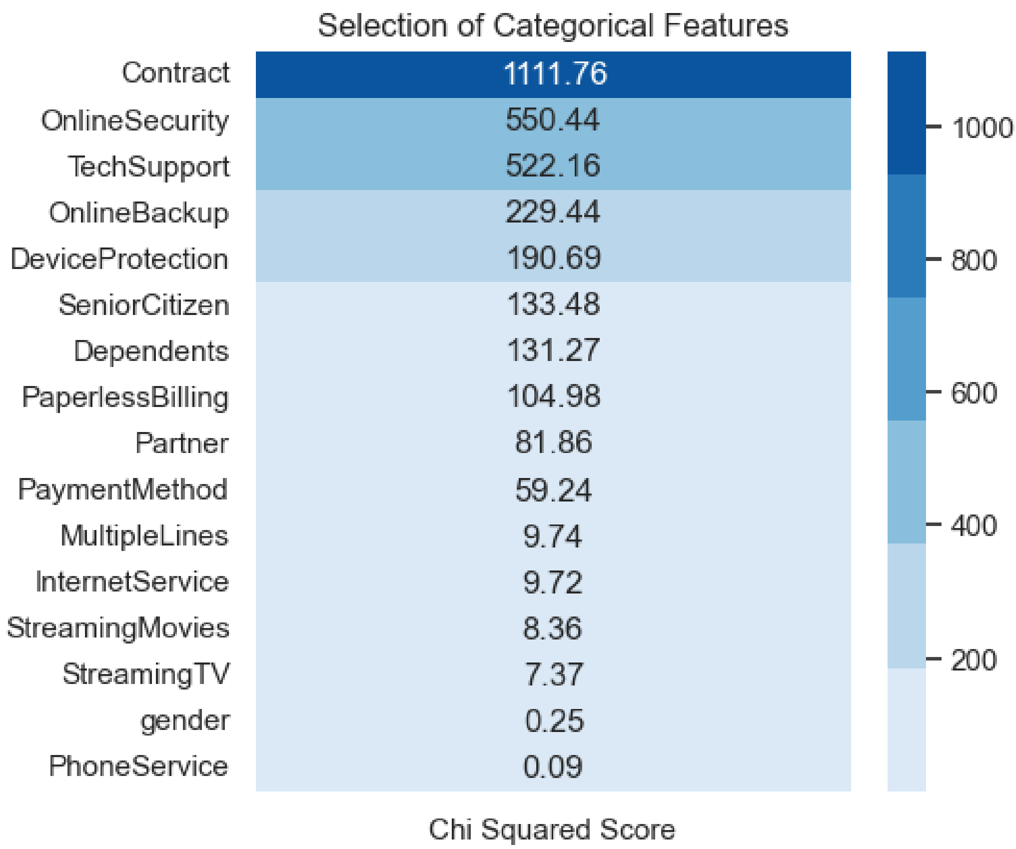
3.0.3 Feature Importance of categorical fields using Chi-Square method

- Here we are segregating categorical features that are heavily impacting the Churning (dependent variable) and will remove features that are least impacting.

[37]:
```
# Chi-Square Test
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, mutual_info_classif
```

[38]:
```
# Heatmap of categorical features
features = df1.loc[:,categorical_features] # categorocal
```

```
target = df1.loc[:,'Churn'] # categorical
best_features = SelectKBest(score_func = chi2, k='all')
fit = best_features.fit(features, target)

featureScores_1 = pd.DataFrame(data = fit.scores_, index= fit.feature_names_in_,
                             columns = ['Chi Squared Score'])

plt.subplots(figsize=(5,5))
sns.heatmap(featureScores_1.sort_values(ascending = False, by = 'Chi Squared␣
 ↪Score'),
            annot=True, cmap = sns.color_palette("Blues"), fmt = '.2f');
plt.title('Selection of Categorical Features')
plt.show()
```



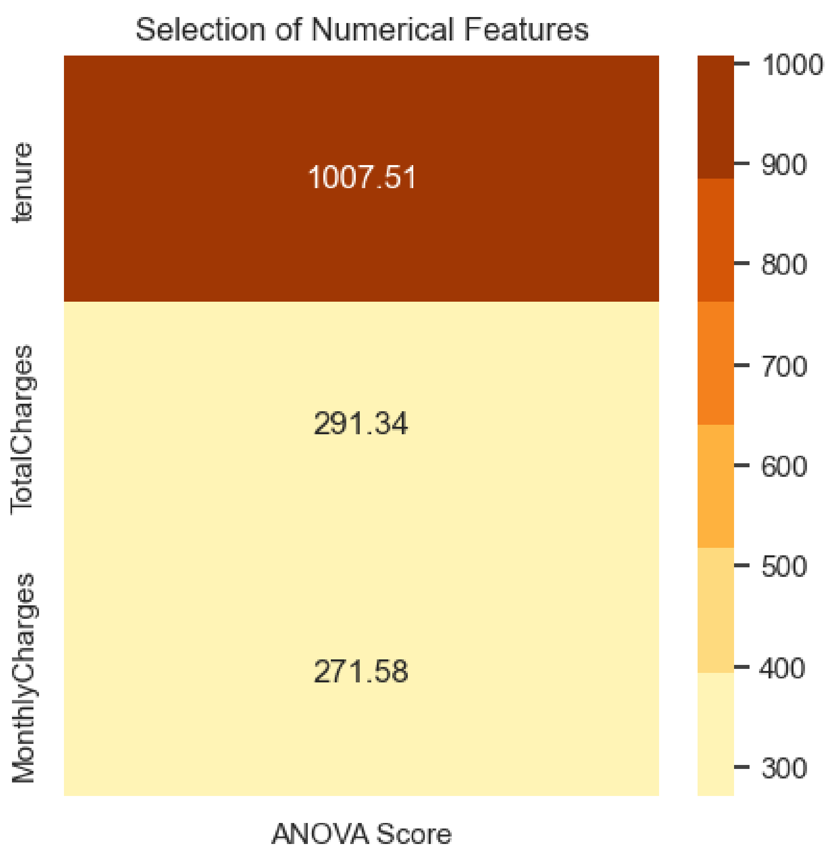
- *From the above feature selection of categorical columns using Chi-Square test, we can observe that the features 'Contract', 'Online Security' and 'TechSupport' is highly impacting the Churn (dependent variable) whereas 'StreamingTV', 'gender' and 'PhoneService' being the least important.*

3.0.4 Feature Importance of numerical fields using ANOVA testing

- Here we are segregating numerical features that are heavily impacting the Churning (dependent variable) and will remove features that are least impacting.

[39]:
```
# Imporing ANOVA feature selection
from sklearn.feature_selection import f_classif
```

[40]:
```
# Heatmap of numerical features
features = df1.loc[:,numerical_features] # numerical
target = df1.loc[:,'Churn'] # categorical
best_features = SelectKBest(score_func = f_classif, k='all')
fit = best_features.fit(features, target)

featureScores_2 = pd.DataFrame(data = fit.scores_, index= fit.feature_names_in_,
                               columns = ['ANOVA Score'])

plt.subplots(figsize=(5,5))
sns.heatmap(featureScores_2.sort_values(ascending = False, by = 'ANOVA Score'),
            annot=True, cmap = sns.color_palette("YlOrBr"), fmt = '.2f');
plt.title('Selection of Numerical Features')
plt.show()
```



- ***From the above feature selection of numerical columns using ANOVA test, we can observe that the feature 'tenure' is highly impacting the Churn (dependent variable) whereas 'TotalCharges' and 'MonthlyCharges' being the moderately important.***

[41]:
```
# Removing less important independent variables
df1.drop(columns=featureScores_1[featureScores_1['Chi Squared Score']<10].
 ↪index,inplace=True)
```

[42]:
```
df1.head()
```

[42]:
```
   SeniorCitizen  Partner  Dependents  tenure  OnlineSecurity  OnlineBackup  \
0              0        1           0   -1.28               0             2
1              0        0           0    0.06               2             0
2              0        0           0   -1.24               2             2
3              0        0           0    0.51               2             0
4              0        0           0   -1.24               0             0

   DeviceProtection  TechSupport  Contract  PaperlessBilling  PaymentMethod  \
0                 0            0         0                 1              2
1                 2            0         1                 0              3
2                 0            0         0                 1              3
3                 2            2         1                 0              0
4                 0            0         0                 1              2

   MonthlyCharges  TotalCharges  Churn
0           -1.16         -0.99      0
1           -0.26         -0.17      0
2           -0.36         -0.96      1
3           -0.75         -0.20      0
4            0.20         -0.94      1
```

[43]:
```
# Checking if we have imbalance dataset
df1['Churn'].value_counts()
```

[43]:
```
Churn
0    5163
1    1869
Name: count, dtype: int64
```

[44]:
```
# Handling imbalance dataset
from imblearn.over_sampling import SMOTE, RandomOverSampler
smote = SMOTE()
f1 = df1.iloc[:,:-1]
t1 = df1.iloc[:,-1]
f1,t1 = smote.fit_resample(f1,t1)
```

- **In the above analysis, we understood that our dataset is imbalanced and hence we fixed the same using SMOTE technique.**

  In Layman's terms - SMOTE technique is used to balance the number of counts for both the classes (0 and 1). Hence in the below cell we can find out that the data is now balanced.

[45]:
```
# Balanced dataset
t1.value_counts()
```

[45]:
```
Churn
0    5163
1    5163
Name: count, dtype: int64
```

4 Moldel Building

[46]:
```
# Model Building
from sklearn.model_selection import train_test_split, cross_val_score,␣
↪RepeatedStratifiedKFold
from sklearn.metrics import confusion_matrix,classification_report,␣
↪accuracy_score, roc_auc_score, roc_curve,␣
↪precision_recall_curve,RocCurveDisplay, auc
```

- Splitting the data into train and test

[47]:
```
x_train, x_test, y_train, y_test = train_test_split(f1,t1, test_size=0.2,␣
↪random_state=101)
```

[48]:
```
def model_eval(*classifier):
    for estimator in classifier:
        print()
        print('**********************',str(type(estimator)).split('.')[-1][:
↪-2],'********************')
        print()
        estimator.fit(x_train, y_train)
        cm = confusion_matrix(y_test,estimator.predict(x_test))
        names= ['True Negative','False Positive','False Negative','True␣
↪Positive']
        counts = [value for value in cm.flatten()]
        percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.
↪sum(cm)]
        labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names, counts,␣
↪percentages)]
        labels = np.asarray(labels).reshape(2,2)
        sns.heatmap(cm, annot=labels, cmap='Blues',fmt='')
        print(classification_report(y_test,estimator.predict(x_test)))
        plt.show()
```

•

[49]:
```
# RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
Random_Forest = RandomForestClassifier(max_depth=4, random_state=1)
```

[50]:
```
# XGBoost Classifier
from xgboost import XGBClassifier
XGBoost = XGBClassifier(learning_rate=0.01,n_estimators=500)
```

[51]:
```
# LightGBM Classifier
from lightgbm import LGBMClassifier
LightGBM = LGBMClassifier(learning_rate=0.01,␣
 ↪n_estimators=500,force_col_wise=True)
```

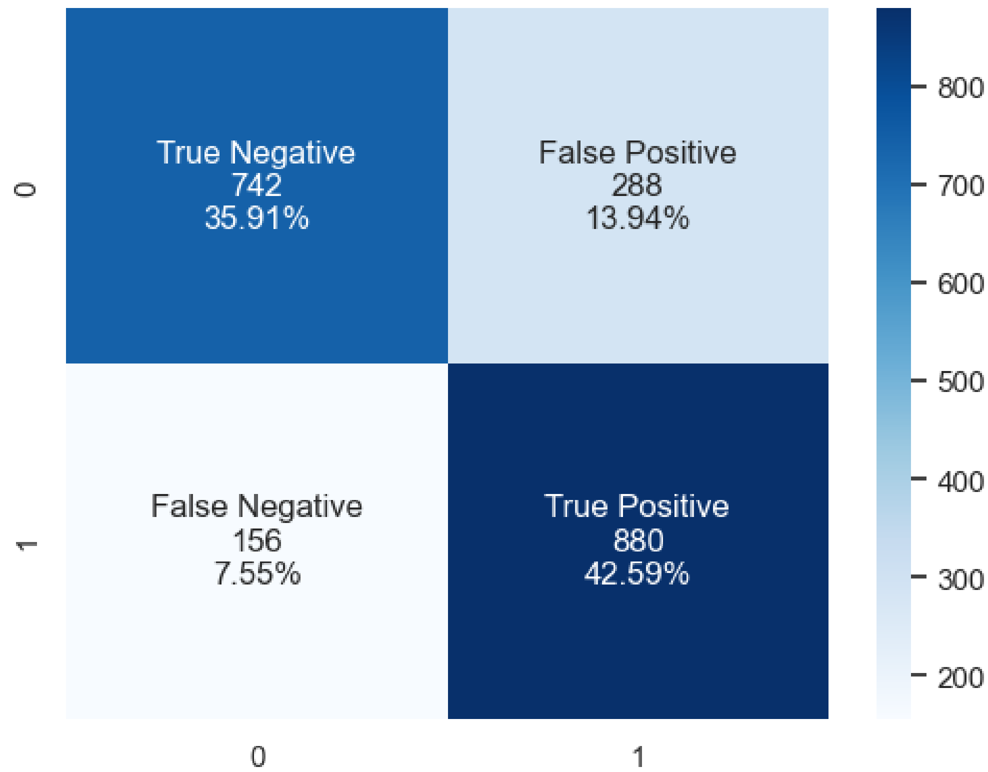
[52]:
```
# Naive Bayes Classifier
from sklearn.naive_bayes import GaussianNB
Gaussian_Naive_Bayes = GaussianNB()
```

[53]:
```
# CatBoost Classifier
from catboost import CatBoostClassifier
Cat_Boost = CatBoostClassifier(learning_rate=0.
 ↪01,eval_metric='AUC',logging_level='Silent')
```

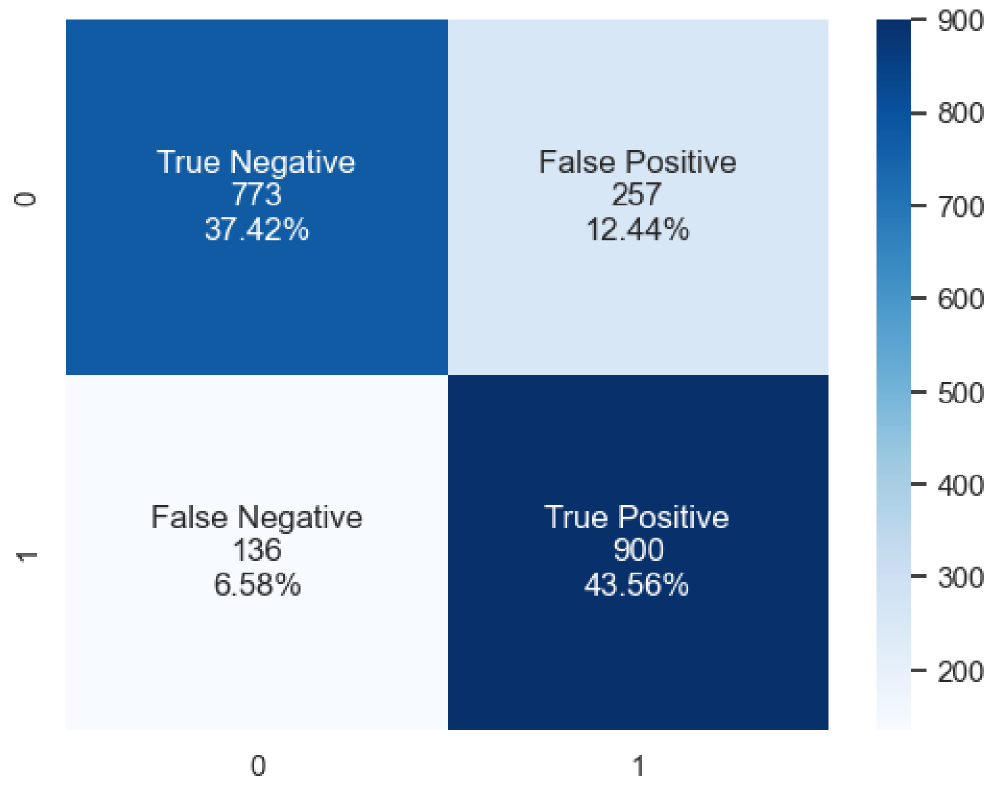
[54]:
```
from sklearn.svm import SVC
SVM_Classifier = SVC()
```

[55]:
```
model_eval(Random_Forest,XGBoost,LightGBM,Gaussian_Naive_Bayes,Cat_Boost)
```

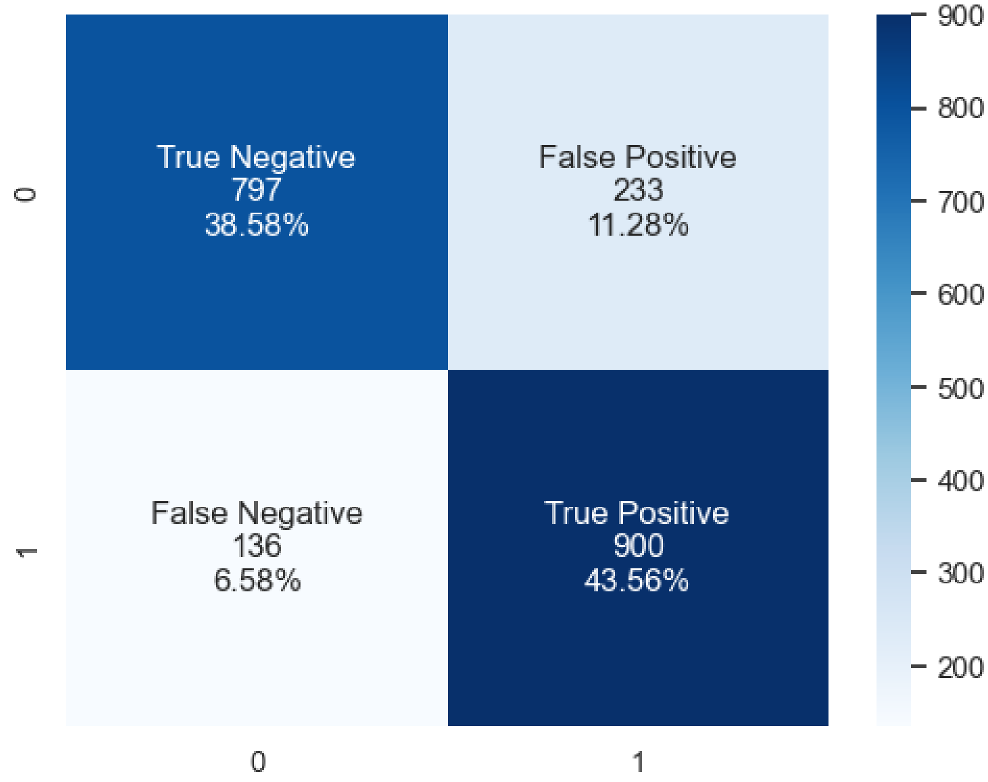
```
********************** RandomForestClassifier *******************

              precision    recall  f1-score   support

           0       0.83      0.72      0.77      1030
           1       0.75      0.85      0.80      1036

    accuracy                           0.79      2066
   macro avg       0.79      0.78      0.78      2066
weighted avg       0.79      0.79      0.78      2066
```



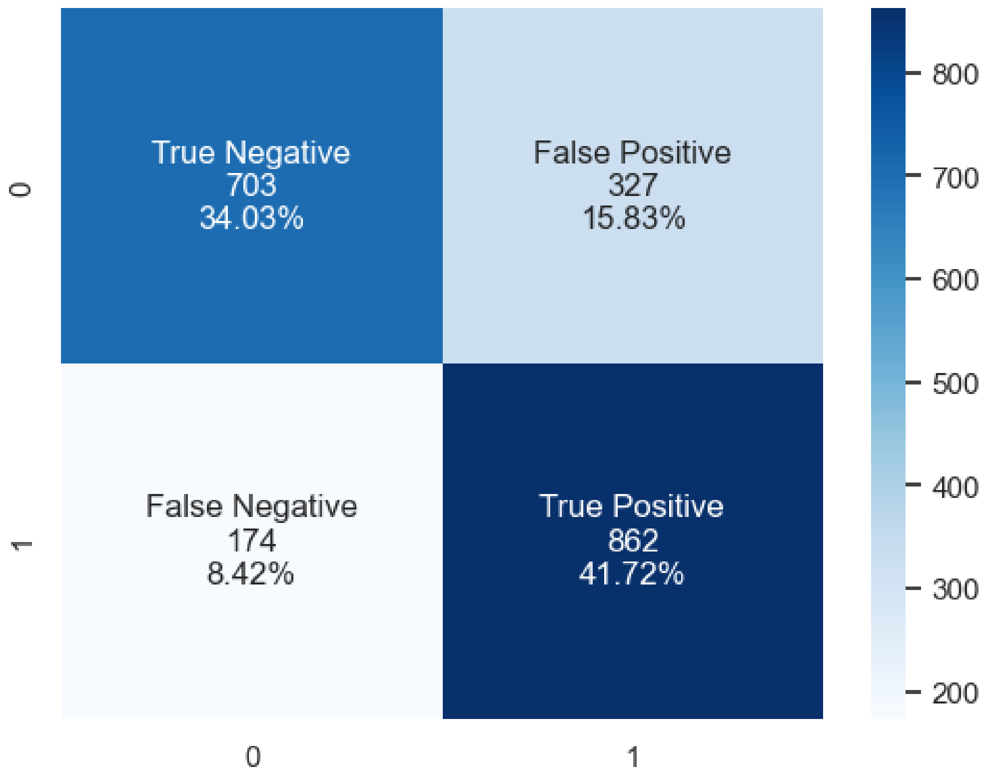

```
********************** XGBClassifier ******************

              precision    recall  f1-score   support

           0       0.85      0.75      0.80      1030
           1       0.78      0.87      0.82      1036

    accuracy                           0.81      2066
   macro avg       0.81      0.81      0.81      2066
weighted avg       0.81      0.81      0.81      2066
```



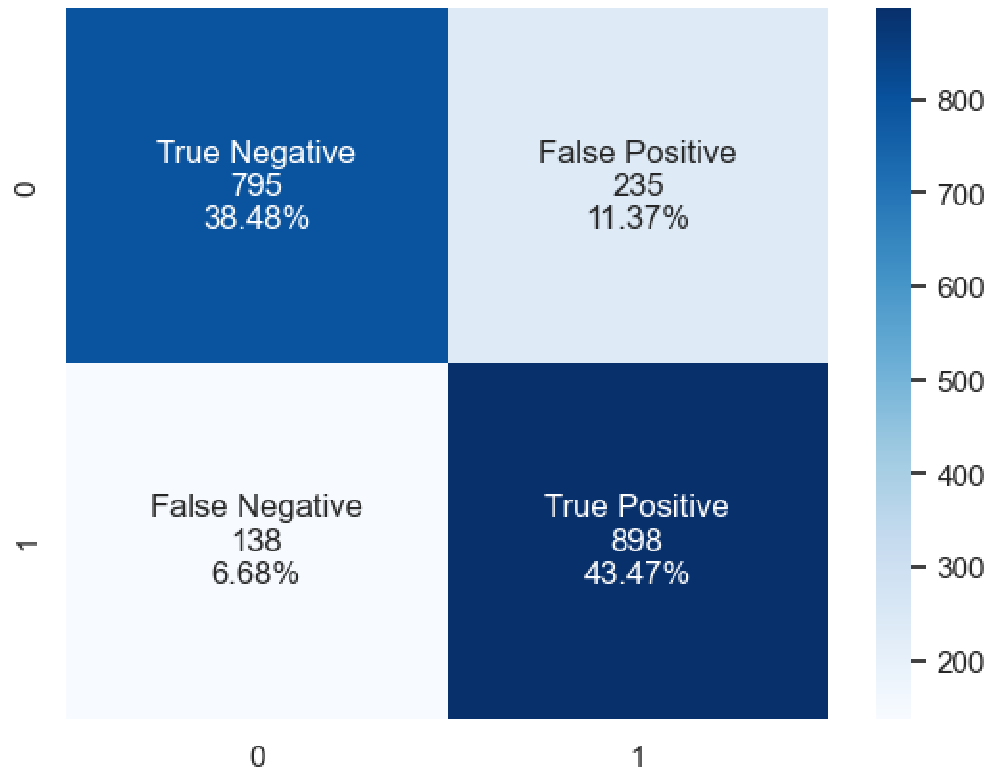

```
********************** LGBMClassifier ******************

[LightGBM] [Info] Number of positive: 4127, number of negative: 4133
[LightGBM] [Info] Total Bins 792
[LightGBM] [Info] Number of data points in the train set: 8260, number of used
features: 13
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499637 -> initscore=-0.001453
[LightGBM] [Info] Start training from score -0.001453
              precision    recall  f1-score   support

           0       0.85      0.77      0.81      1030
           1       0.79      0.87      0.83      1036

    accuracy                           0.82      2066
   macro avg       0.82      0.82      0.82      2066
weighted avg       0.82      0.82      0.82      2066
```




```
********************** GaussianNB ******************

              precision    recall  f1-score   support

           0       0.80      0.68      0.74      1030
           1       0.72      0.83      0.77      1036

    accuracy                           0.76      2066
   macro avg       0.76      0.76      0.76      2066
weighted avg       0.76      0.76      0.76      2066
```




```
********************** CatBoostClassifier *******************

              precision    recall  f1-score   support

           0       0.85      0.77      0.81      1030
           1       0.79      0.87      0.83      1036

    accuracy                           0.82      2066
   macro avg       0.82      0.82      0.82      2066
weighted avg       0.82      0.82      0.82      2066
```



- **Above information depicts the classification report where we can establish the fact that LGBMClassifier and Cat Boosting technique gives us the max accuracy whereas Gaussian Naive Bayes being the least**
- Confusion matrix is represented using heatmap.

4.0.1 Fitting the dataset into the models.

[56]:
```
Random_Forest.fit(x_train, y_train)
XGBoost.fit(x_train, y_train)
LightGBM.fit(x_train, y_train)
Cat_Boost.fit(x_train,y_train)
Gaussian_Naive_Bayes.fit(x_train,y_train)
```

```
[LightGBM] [Info] Number of positive: 4127, number of negative: 4133
[LightGBM] [Info] Total Bins 792
[LightGBM] [Info] Number of data points in the train set: 8260, number of used
features: 13
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499637 -> initscore=-0.001453
```

```
[LightGBM] [Info] Start training from score -0.001453
```

[56]:

```
GaussianNB()
```

[57]:

```python
rf_probs = Random_Forest.predict_proba(x_test)[:,1]
xgb_probs = XGBoost.predict_proba(x_test)[:,1]
lgbm_probs = LightGBM.predict_proba(x_test)[:,1]
nb_probs = Gaussian_Naive_Bayes.predict_proba(x_test)[:,1]
cat_probs = Cat_Boost.predict_proba(x_test)[:,1]
```

[58]:

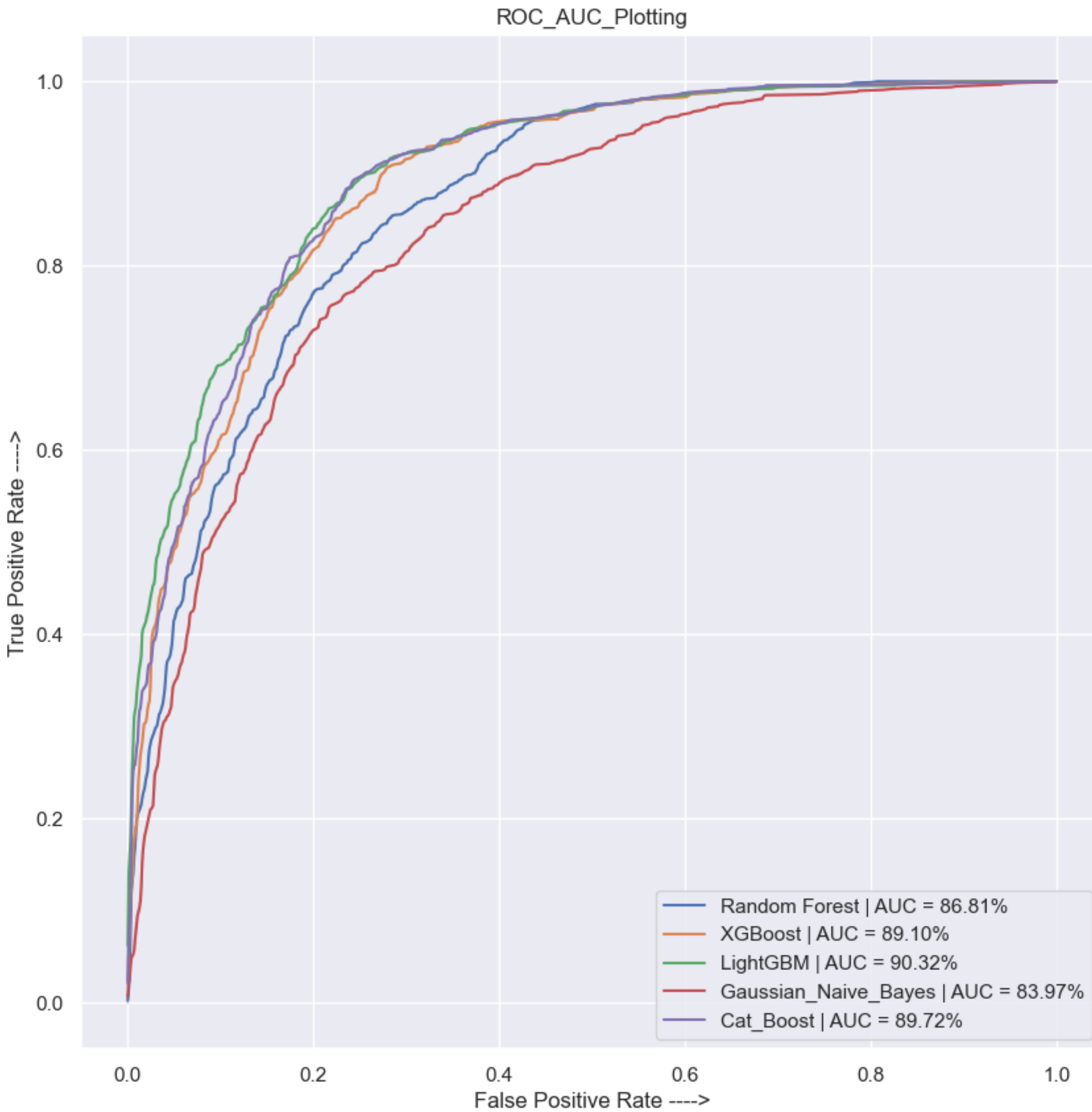
```python
rf_auc = roc_auc_score(y_test,rf_probs) * 100
xgb_auc = roc_auc_score(y_test,xgb_probs) * 100
lgbm_auc = roc_auc_score(y_test,lgbm_probs) * 100
nb_auc = roc_auc_score(y_test,nb_probs) * 100
cat_auc = roc_auc_score(y_test,cat_probs) * 100
```

[59]:

```python
rf_fpr,rf_tpr,rf_threshold = roc_curve(y_test,rf_probs)
xgb_fpr,xgb_tpr,xgb_threshold = roc_curve(y_test,xgb_probs)
lgbm_fpr,lgbm_tpr,lgbm_threshold = roc_curve(y_test,lgbm_probs)
nb_fpr,nb_tpr,nb_threshold = roc_curve(y_test,nb_probs)
cat_fpr,cat_tpr,cat_threshold = roc_curve(y_test,cat_probs)
```

[60]:

```python
plt.figure(figsize=(10,10))
sns.lineplot(x=rf_fpr,y=rf_tpr,label='Random Forest | AUC = {:.2f}%'.
 ↪format(rf_auc))
sns.lineplot(x=xgb_fpr,y=xgb_tpr,label='XGBoost | AUC = {:.2f}%'.
 ↪format(xgb_auc))
sns.lineplot(x=lgbm_fpr,y=lgbm_tpr,label='LightGBM | AUC = {:.2f}%'.
 ↪format(lgbm_auc))
sns.lineplot(x=nb_fpr,y=nb_tpr,label='Gaussian_Naive_Bayes | AUC = {:.2f}%'.
 ↪format(nb_auc))
sns.lineplot(x=cat_fpr,y=cat_tpr,label='Cat_Boost | AUC = {:.2f}%'.
 ↪format(cat_auc))
plt.xlabel("False Positive Rate ---->")
plt.ylabel("True Positive Rate ---->")
plt.title("ROC_AUC_Plotting")
plt.legend()
plt.show()
```



5 What we started with?

- We started with a sample of dataset related to telecom industry where the problem statement was to classify the potential churn customers based on numerical and categorical features. Along with we also had to identify whether the provided problem was a binary classification and if it's an imbalanced dataset. We had certain dataset attributes where each features described its dependency on the final outcome.

6 What we observed and course of actions initiated?

- The very first step of analysis was metadata, that is knowing data about the data where I get to know the dataset attributes, it's meaning, count of total records and drawing segregation between a classification or a regression problem statement. Formal proceedings included data cleaning where importing of basic analytical and visualisation libraries (like numpy, pandas, seaborn, etc). We imported the dataset into Jupyter notebook and check for the missing values. Dropped missing data where necessary. Encoded the categorical columns using label encoder and scaled the numerical columns. I also checked for any imbalances in the dataset and fixed using oversampling technique called SMOTE. Correlations were drawn out with each of the independent features with respect to Churn data (dependent column) and using sklearn's feature importance technique I able to drop the least important features for better model prediction. Segregation of features into three cases helped my model to get a better fit and accuracy score.

7 Conclusion

- Here I have used 5 of the machine learning classifiers — RandomForest, XGBoosting, LGBM-Classifier, Cat Boosting and Gaussian Naive Bayes out of which LGBM Classifier proved to provide the best accuracy score. We can see the same in the ROC_AUC curve where the area under the same is maximum, that is 90.32%.

8 Improvements

- I deliberately added this section in my project as Artificial Intelligence is itself an emerging subject of our new era which has rapidly gained strength in past few years and Machine Learning is one of the element under its umbrella. As I believe am a student and to gain mastery over the same takes years. Being said so, my project has a lot of scope for improvements which due to time stringent I could not able to do so. I would also appreciate those masters out there in platform to advise on my project to lift my progress.