TELECOM INDUSTRY CHURNING

THROUGH MACHINE LEARNING

WITH PYTHON

BY

1. SLOK KUMAR MAHTA
2. SNIGDHA SAHU
3. SHALINI KUMARI
4. GOURAB SARKAR
5. HAREKRISHNA MANDAL
6. AVIK SARKAR

SCOPE OF THE PROJECT

What Is Churn Rate?

The churn rate, also known as the rate of attrition or customer churn, is the rate at which customers stop doing business with an entity. It is most commonly expressed as the percentage of service subscribers who discontinue their subscriptions within a given time period.

Source of The Dataset:

For achieving our objective we have taken a public data set from kaggle.com which has been recorded from the year 2011 to 2018.The dataset was present in the form of a ".csv" file named "telecom.csv".

Purpose of the project:

The main contribution of our work is to develop a churn prediction model which assists telecom operators to predict customers who are most likely subject to churn.

The model developed in this work uses machine learning techniques on python platform and builds a new way of features' engineering and selection.

Many approaches have been applied to predict churn in telecom companies. Most of these approaches have used machine learning and data analysis. The majority of related work focused on applying only one method of data analysis to extract knowledge, and the others focused on comparing several strategies to predict churn.

Objectives:

- The objectives of the project will be discussed in particular section.They are as follows:

- Analyze the financial data and behavior of a set of users.
- Find the frequency of the varients of data and behaviour of a set of user.
- Find the correlation of each attribute given.
- Clean the data to ensure it can be fit to any classification model.
- Finding the accuracy of varaity of classification models to get the best classifier.
- Optimizing the best classification model to find the further accuracy.
- Displaying the possibility of churning on the new set of independent variables.

Advantages:

If one out of every 20 subscribers to a high-speed Internet service terminated their subscriptions within a year, then annual churn rate for that internet provider would be 5%.So if we predict and speculate the churn rate using data analysis then it will help the company to prevent a massive loss in net income.Hence in order to make the company's turnout more lucrative data planning and analysing is essential.

Limitations:

The said project has been done on a data set within the time duration 2010 to 2018.For more optimum and accurate data analysis we need data beyond this duration,unfortunately in kaggle.com only this data set with limited time duration was available.

DESCRIPTION OF VARIABLES OF THE DATASET

There are 21 variables in our Dataset which are as follows:

customerID:

Identifier of a customer

Null Values	0
Unique values	7043
Variable Type	Categorical
Features	Independent

Gender:

Whether the customer is a male or a female.

Null Values	0
Unique values	2
Variable Type	Categorical
Features	Independent

SeniorCitizen:

Whether the customer is a senior citizen or not (1, 0).

Null Values	0
Unique values	2
Variable Type	Categorical
Features	Independent

Partner:

Whether the customer has a partner or not (Yes, No).

Null Values	0
Unique values	2
Variable Type	Categorical
Features	Independent

Dependents:

Whether the customer has dependents or not (Yes, No).

Null Values	0
Unique values	2
Variable Type	Categorical
Features	Independent

Tenure:

Number of months the customer has stayed with the company.

Null Values	0
Unique values	73
Variable Type	Numerical(Discrete)
Features	Independent

PhoneService:

Whether the customer has a phone service or not (Yes, No).

Null Values	0
Unique values	2
Variable Type	Categorical
Features	Independent

MultipleLines:

Whether the customer has multiple lines or not (Yes, No, No phone service).

Null Values	0
Unique values	3
Variable Type	Categorical
Features	Independent

InternetService:

Customer’s internet service provider (DSL, Fiber optic, No).

Null Values	0
Unique values	3
Variable Type	Categorical
Features	Independent

OnlineSecurity:

Whether the customer has online security or not (Yes, No, No internet service).

Null Values	0
Unique values	3
Variable Type	Categorical
Features	Independent

OnlineBackup:

Whether the customer has online backup or not (Yes, No, No internet service).

Null Values	0
Unique values	3
Variable Type	Categorical
Features	Independent

DeviceProtection:

Whether the customer has device protection or not (Yes, No, No internet service).

Null Values	0
Unique values	3
Variable Type	Categorical
Features	Independent

TechSupport:

Whether the customer has tech support or not (Yes, No, No internet service).

Null Values 0

Unique values 3

Variable Type Categorical

StreamingTV:

Whether the customer has streaming TV or not (Yes, No, No internet service).

Null Values 0

Unique values 3

Variable Type Categorical

Features Independent

StreamingMovies:

Whether the customer has streaming movies or not (Yes, No, No internet service).

Null Values 0

Unique values 3

Variable Type Categorical

Features Independent

Contract:

The contract term of the customer (Month-to-month, One year, Two year).

Null Values 0

Unique values 3

Variable Type Categorical

Features Independent

PaperlessBilling:

Whether the customer has paperless billing or not (Yes, No).

Null Values 0

Unique values 2

Variable Type Categorical

Features Independent

PaymentMethod:

The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic)).

Null Values 0

Unique values 4

Variable Type Categorical

Features Independent

MonthlyCharges:

The amount charged to the customer monthly.

Null Values 0

Unique values 1585

Variable Type Numerical(Continuous)

Features Independent

TotalCharges:

The total amount charged to the customer.

Null Values	0
Unique values	6531
Variable Type	String type
Features	Independent

Churn:

Whether the customer churned or not (Yes or No).

Null Values	0
Unique values	2
Variable Type	Categorical
Features	Dependent

EXPLORATORY DATA ANALYSIS

Importing Necessary Libraries and Data Set:

The aforementioned libraries of numpy, matplotlib, pandas,seaborn are imported for mathematical operations on matrices, graphical representation of data and data retrieval and manipulation respectively. The pandas library is then used to import the given data set through read_csv( ) function.

Exploring the Data:

The attributes present in the data set, the first 5 entities as well as a summary of some key values is presented to further get an overview of the data set.

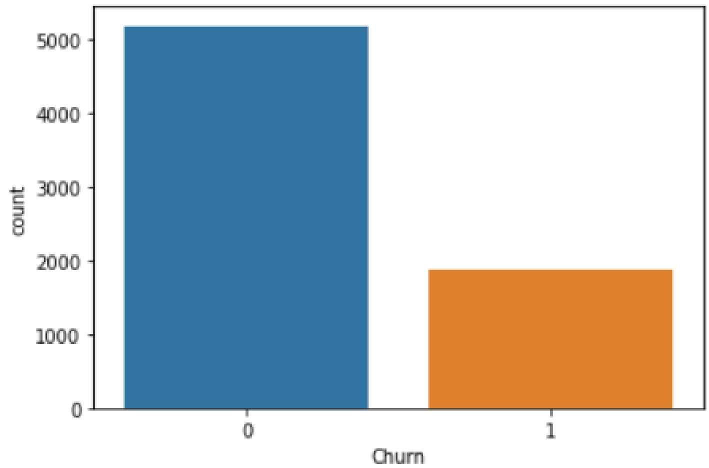
Representing Frequency of Data through Different Types Of Charts:

The charts help us to give a pictorial representation of the datas which enhances the further understanding of the datas.

1. The chart given below is a Bar chart.It depicts the churn in the X axis and count in the Y axis.The chart gives us information about the number of customers who left off the telecom service i.e about 2000 customers churned and it also depicts that 5000 did not churn.

```
import pandas as pd
import seaborn as sns
import math
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
sns.countplot(x='Churn',data=cdf)
```

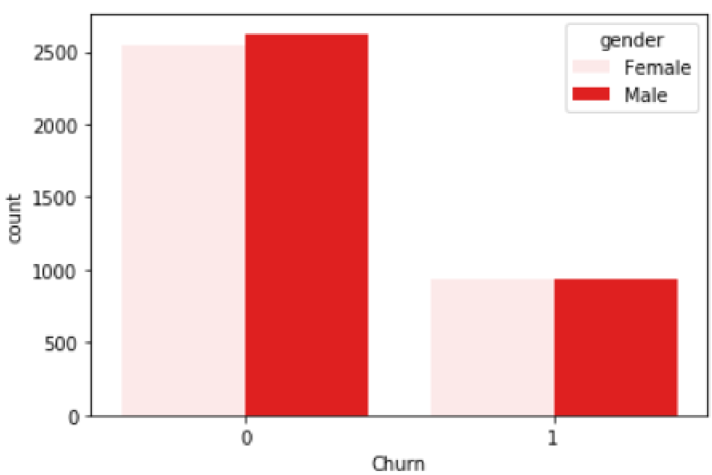
[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1d680164048>



2. The chart given below depicts the churn rate in accordance with the gender.The churn of male and female are equal and the no. of males who did not churn is higher than the no. of females.The X axis marked with ‘0’ depicts that the customer did not churn and the mark ‘1’ depicts that the customer who churned.

```
sns.countplot(x='Churn',hue='gender',data=cdf,color='r')
```

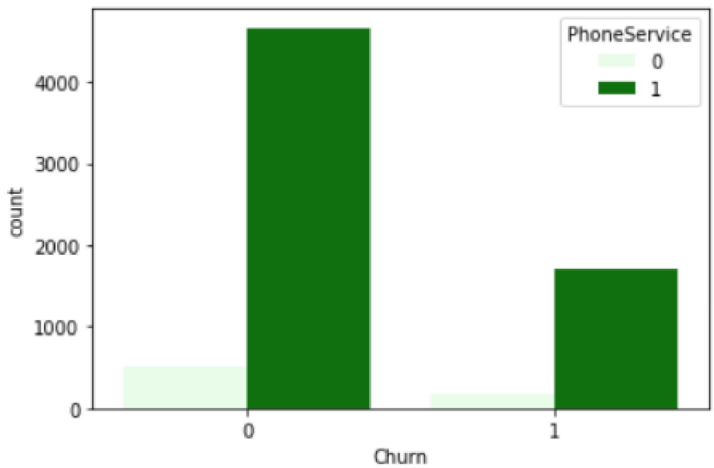
: <matplotlib.axes._subplots.AxesSubplot at 0x1d6802f53c8>



3. In the graph given below,the number of people who did not have any phone service is represented by light green bar and the number of people who had the phone service is represented by dark green bar.The X axis marked with ‘0’ depicts that the customer did not churn and the mark ‘1’ depicts that the customer who churned.

```
sns.countplot(x='Churn',hue='PhoneService',data=cdf,color='g')
```

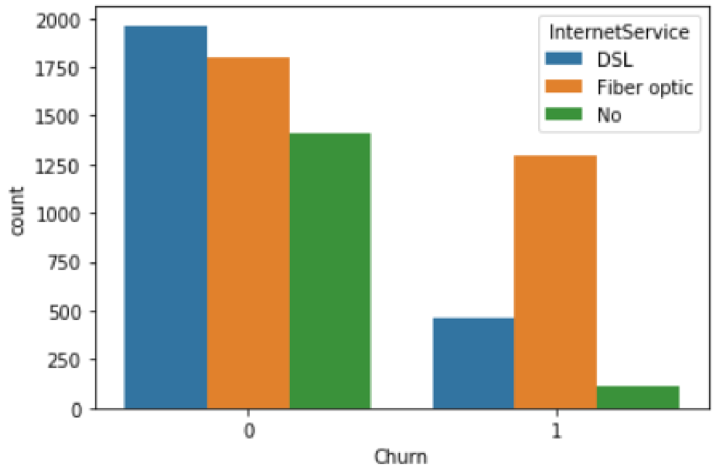
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d6801f9588>
```



4. The X axis marked with ‘0’ depicts that the customer did not churn and the mark ‘1’ depicts that the customer who churned.
   The blue bar represents the DSL type of internet service,orange bar represents the Fibre optic type of internet service and the green bar represents no internet service.

```
sns.countplot(x='Churn',hue='InternetService',data=cdf)
```

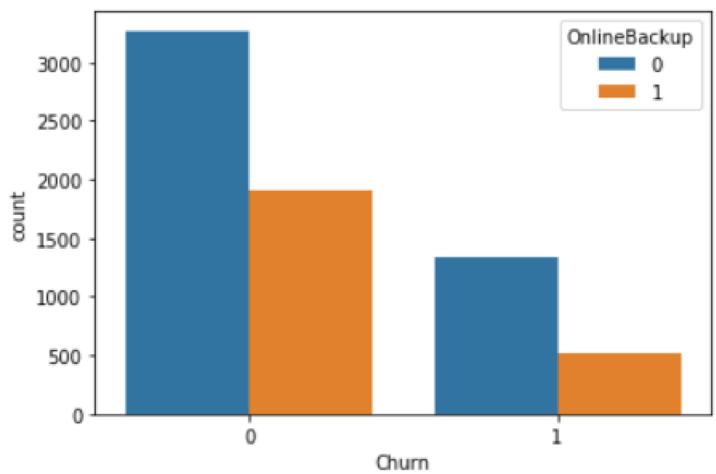
```
: <matplotlib.axes._subplots.AxesSubplot at 0x1d680248b08>
```



5. The X axis marked with '0' depicts that the customer did not churn and the mark '1' depicts that the customer who churned.
   The blue graph represents that the customer has no online backup. The orange graph represents that the customer has an online backup.

```
sns.countplot(x='Churn',hue='OnlineBackup',data=cdf)
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x1d68041f488>
```

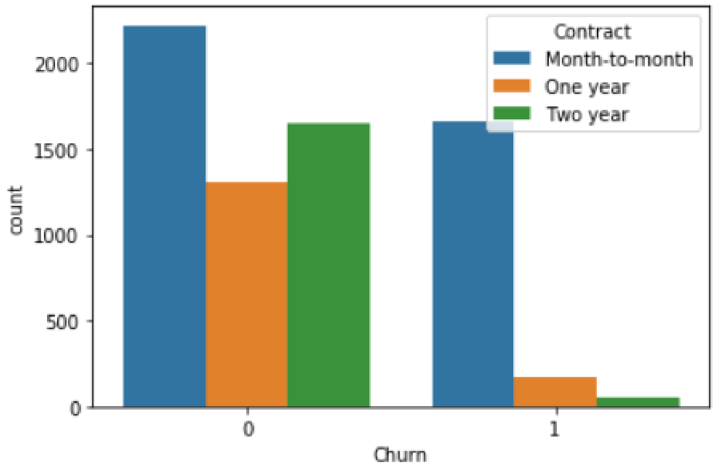


6. The X axis marked with '0' depicts that the customer did not churn and the mark '1' depicts that the customer who churned.

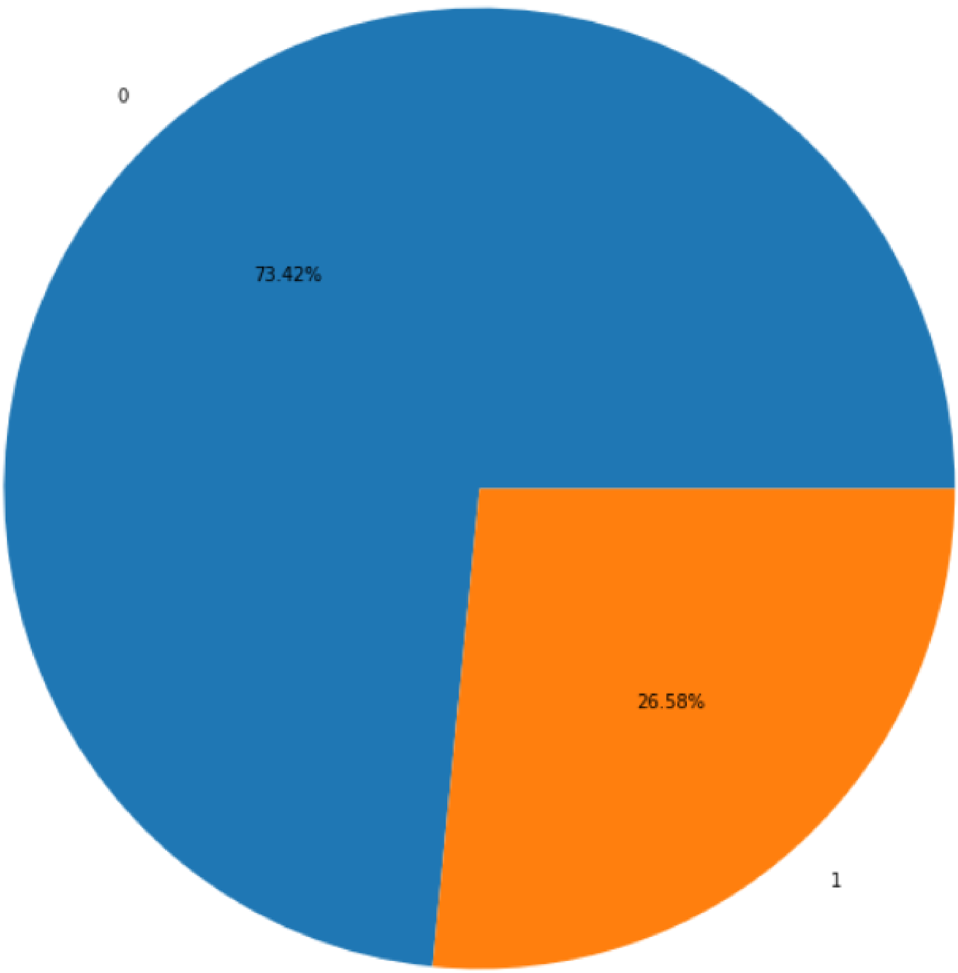
The blue graph represents that the customer has month to month contract. The orange graph represents that the customer has an one year contact. The green graph represents that the customer has two year contact.

```
sns.countplot(x='Churn',hue='Contract',data=cdf)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d6800d8688>
```

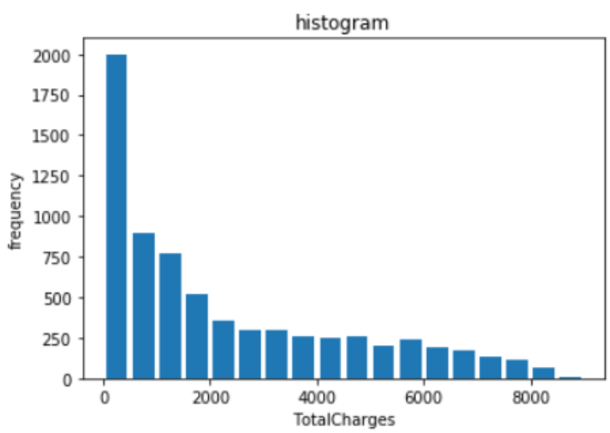


```
lab = cdf["Churn"].value_counts().keys().tolist()
#values
val = cdf["Churn"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=3,autopct='%0.2f%%')
plt.show()
```



7.The histogram given below represents TotalCharges in the X axis and the Frequency in the Y axis.The different bars represents the different intervals.The interval 0-500 has maximum frequency i.e 2000.

```
ten=cdf['TotalCharges']
bins=[0,500,1000,1500,2000,2500,3000,3500,4000,4500,5000,5500,6000,6500,7000,7500,8000,8500,9000]
plt.hist(ten,bins,histtype='bar',rwidth=0.8)
plt.xlabel('TotalCharges')
plt.ylabel('frequency')
plt.title('histogram')
plt.show()
```

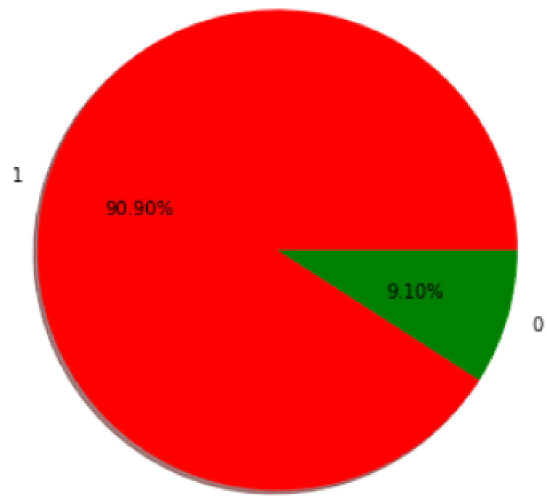


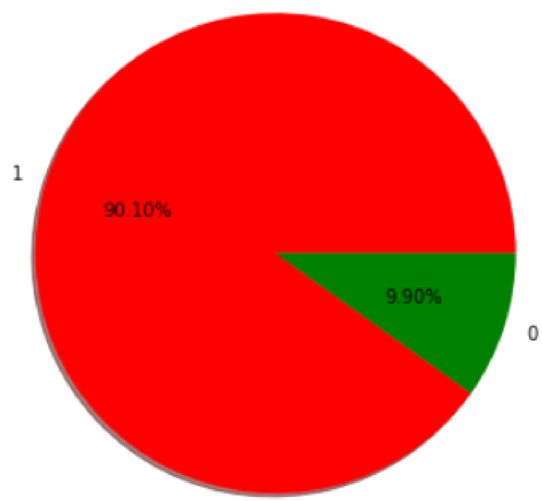
```
churn     = cdf[cdf["Churn"] == "Yes"]
not_churn = cdf[cdf["Churn"] == "No"]
```

```
lab = churn["PhoneService"].value_counts().keys().tolist()
val = churn["PhoneService"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(243,243,243)",shadow=True)
plt.show()
lab = not_churn["PhoneService"].value_counts().keys().tolist()
val = not_churn["PhoneService"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(243,243,243)",shadow=True)
plt.show()
```

1.The first pie chart given below depicts the percentage of the customers who churned by having a phone service and by not having a phone service.

The second pie chart given below depicts the percentage of the customers who did not churn by having a phone service and by not having a phone service.



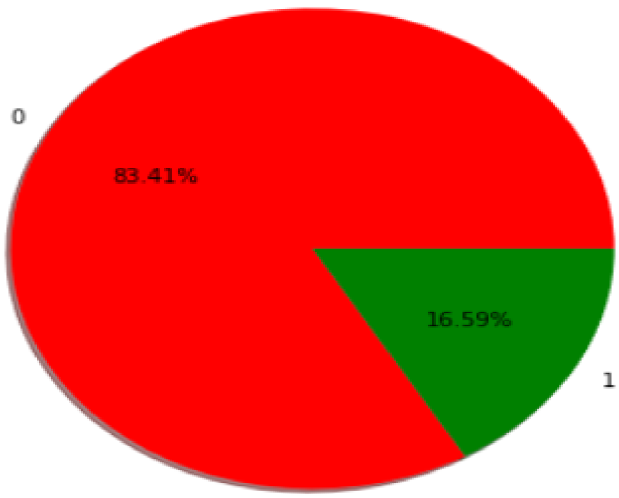


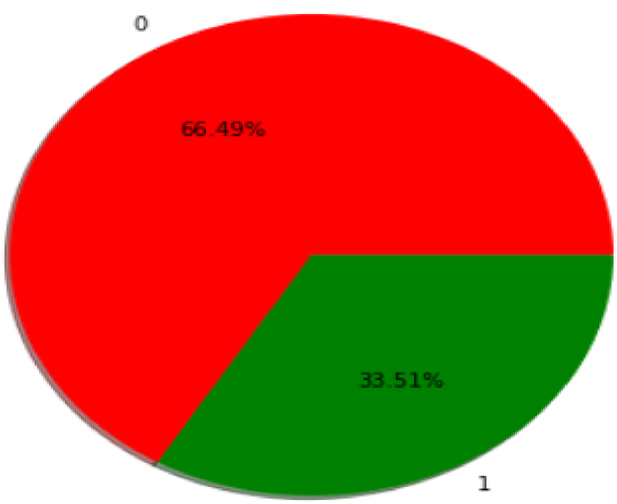

```
lab = churn["InternetService"].value_counts().keys().tolist()
val = churn["InternetService"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(255, 255, 255)",shadow=True)
plt.show()
lab = not_churn["InternetService"].value_counts().keys().tolist()
val = not_churn["InternetService"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(255, 255, 255)",shadow=True)
plt.show()
```

2.The first pie chart given below depicts the percentage of the customers who churned by having a InternetService and by not having a InternetService.

The second pie chart given below depicts the percentage of the customers who did not churned by having a InternetService and by not having a InternetService.

.

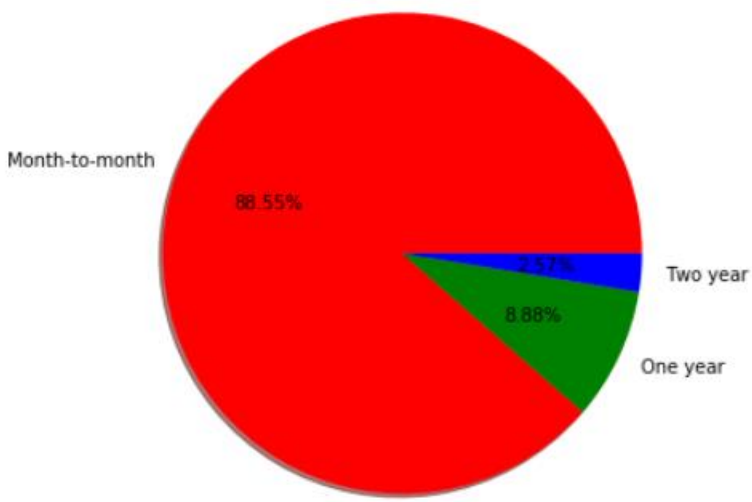


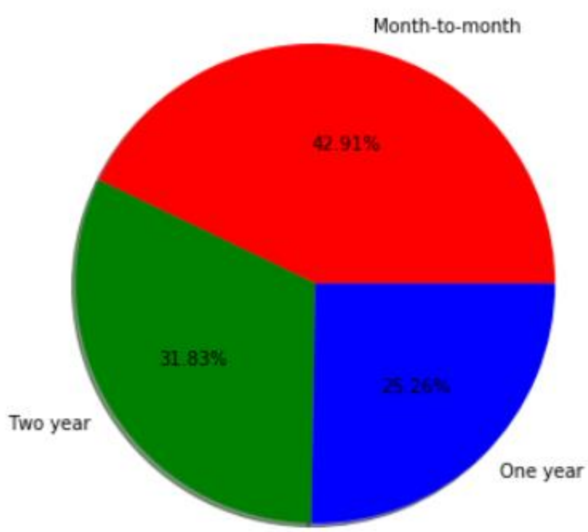



```
lab = churn["Contract"].value_counts().keys().tolist()
val = churn["Contract"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(255, 255, 255)",shadow=True)
plt.show()
lab = not_churn["Contract"].value_counts().keys().tolist()
val = not_churn["Contract"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(255, 255, 255)",shadow=True)
plt.show()
```

3.The first pie chart given below depicts the percentage of the customers who churned by having a Contract and by not having a Contract.

The first pie chart given below depicts the percentage of the customers who did not churned by having a Contract and by not having a Contract.

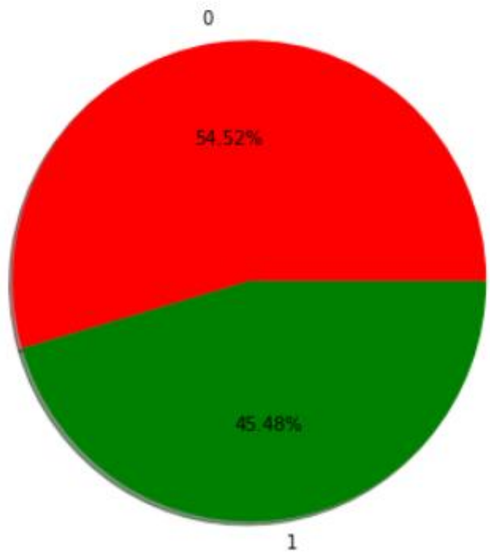


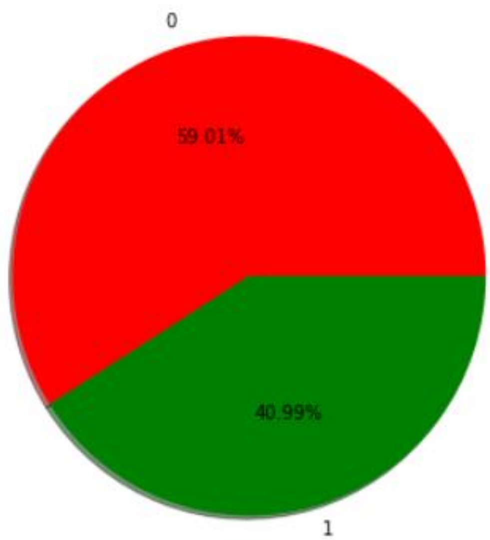


```
lab = churn["MultipleLines"].value_counts().keys().tolist()
val = churn["MultipleLines"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(255, 255, 255)",shadow=True)
plt.show()
lab = not_churn["MultipleLines"].value_counts().keys().tolist()
val = not_churn["MultipleLines"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=1.5,autopct='%0.2f%%',colors = "rgb(255, 255, 255)",shadow=True)
plt.show()
```

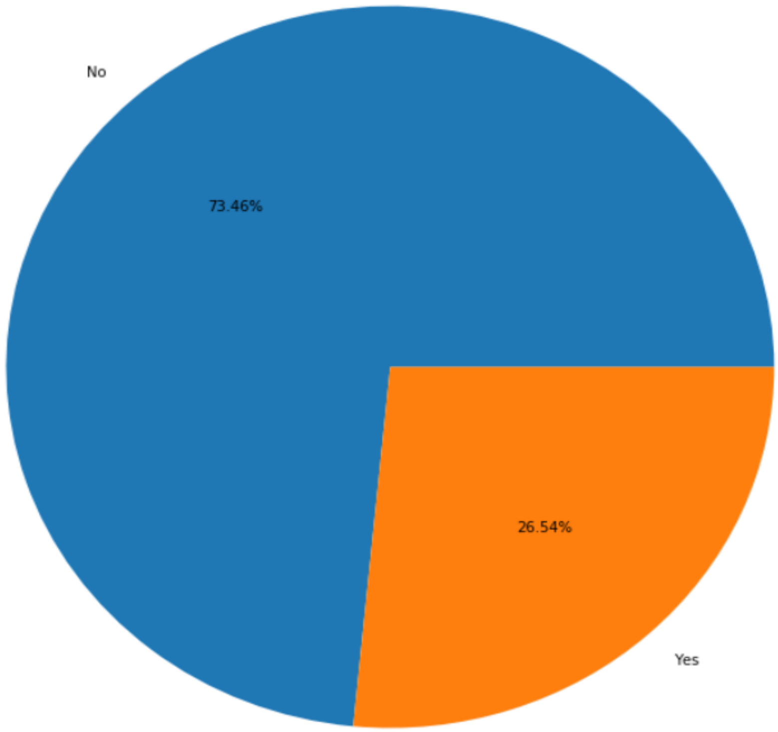
4.The first pie chart given below depicts the percentage of the customers who churned by having a MultipleLines and by not having a MultipleLines.

The second pie chart given below depicts the percentage of the customers who did not churned by having a MultipleLines and by not having a MultipleLines.


0
54.52%
45.48%
1


0
59.01%
40.99%
1

```
lab = cdf["Churn"].value_counts().keys().tolist()
#values
val = cdf["Churn"].value_counts().values.tolist()
plt.pie(val,labels=lab,radius=3,autopct='%0.2f%%')
plt.show()
```



5.The above pie chart depicts the customers who churned or did not churn.

Dataset Preparation

- All the required modules such as NumPy, pandas, sklearn etc. has been imported.

```
import numpy as np
import  pandas as pd
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2,f_classif
```

- We have created a dataframe named ‘cdf’ from the csv file “telecom.csv” and found out the information of the dataframe by using the info function. Further we described the dataframe by using the describe function.

  Code:

```
cd=pd.read_csv("telecom.csv")
cdf=pd.DataFrame(cd)
print(cdf.info())
print(cdf.describe())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
customerID          7043 non-null object
gender              7043 non-null object
SeniorCitizen       7043 non-null int64
Partner             7043 non-null object
Dependents          7043 non-null object
tenure              7043 non-null int64
PhoneService        7043 non-null object
MultipleLines       7043 non-null object
InternetService     7043 non-null object
OnlineSecurity      7043 non-null object
OnlineBackup        7043 non-null object
DeviceProtection    7043 non-null object
```

```
TechSupport          7043 non-null object
StreamingTV          7043 non-null object
StreamingMovies      7043 non-null object
Contract             7043 non-null object
PaperlessBilling     7043 non-null object
PaymentMethod        7043 non-null object
MonthlyCharges       7043 non-null float64
TotalCharges         7043 non-null object
Churn                7043 non-null object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None
       SeniorCitizen       tenure  MonthlyCharges
count    7043.000000  7043.000000     7043.000000
mean        0.162147    32.371149       64.761692
std         0.368612    24.559481       30.090047
min         0.000000     0.000000       18.250000
25%         0.000000     9.000000       35.500000
50%         0.000000    29.000000       70.350000
75%         0.000000    55.000000       89.850000
max         1.000000    72.000000      118.750000
```

- We then found out all the unique values of corresponding variables of the dataframe by using the unique function and counted the no. of the unique values of corresponding variables of the dataframe by using the nunique function.

```
cdf.nunique()
```

```
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
```

- We changed the “No phone service”: ‘N0’ and “No internet service”: “No” in required columns of DataFrame for better interpretation.

Code:

```
list1 = [ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                 'TechSupport','StreamingTV', 'StreamingMovies']
for i in list1:
    cdf[i] = cdf[i].replace({'No internet service':'No'})
cdf['MultipleLines'] = cdf['MultipleLines'].replace({'No phone service':'No'})
```

- We then replaced the values=’Yes’ to ‘1’ and values=’No’ to ‘0’ for the variables such as :'OnlineSecurity', 'PhoneService' ,'OnlineBackup','DeviceProtection','TechSupport','StreamingTV', 'StreamingMovies','MultipleLines','Partner','Dependents','PaperlessBilling,’Churn’ by using the replace function.

Code:

```
list2=[ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection','Churn',
                 'TechSupport','StreamingTV',
'StreamingMovies','MultipleLines','Partner','Dependents','PaperlessBilling','PhoneService']
for i in list2:
     cdf[i]= cdf[i].replace({'Yes':1,'No':0})
print(cdf.head())
```

Output:

```
   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  \
0  7590-VHVEG  Female              0        1           0       1   
1  5575-GNVDE    Male              0        0           0      34   
2  3668-QPYBK    Male              0        0           0       2   
3  7795-CFOCW    Male              0        0           0      45   
4  9237-HQITU  Female              0        0           0       2   

   PhoneService  MultipleLines InternetService  OnlineSecurity  ...  \
0             0              0             DSL               0  ...   
1             1              0             DSL               1  ...   
2             1              0             DSL               1  ...   
3             0              0             DSL               1  ...   
4             1              0     Fiber optic               0  ...   

   DeviceProtection  TechSupport  StreamingTV  StreamingMovies  \
0                 0            0            0                0   
1                 1            0            0                0   
2                 0            0            0                0   
3                 1            1            0                0   
4                 0            0            0                0   

         Contract PaperlessBilling              PaymentMethod MonthlyCharges  \
0  Month-to-month                1           Electronic check          29.85   
1        One year                0               Mailed check          56.95   
2  Month-to-month                1               Mailed check          53.85   
3        One year                0  Bank transfer (automatic)          42.30   
4  Month-to-month                1           Electronic check          70.70   

   TotalCharges Churn  
0         29.85     0  
1        1889.5     0  
2        108.15     1  
3       1840.75     0  
4        151.65     1  

[5 rows x 21 columns]
```

- We then found the dummies of the variables such as “gender”, "InternetService", "Contract", and concatenated the dummy variable of the respective variables to the original dataframe and dropped the initial variables.We dropped the variables such as "customerID", "PaymentMethod" as they do not have a high correlation with the dependent variable i.e the ‘Churn’ variable.

- We observed from the above described table that the standard deviations of “tenure” was very high so we changed all the null values in the “tenure” variable to ‘0.1’ .As 0 signifies that the no. of month is zero. Therefore we can replace it by 0.1 as it represents that customer is engaged for sometime(some days here) but not for a month of period.

Code:

```
gen=pd.get_dummies(cdf["gender"],drop_first = True)
cdf=pd.concat((cdf,gen),axis=1)
cdf.drop(['gender'],axis=1,inplace=True)
y=pd.get_dummies(cdf["InternetService"])
y.drop(['No'],axis=1,inplace=True)
cdf.drop(['InternetService'],axis=1,inplace=True)
cdf=pd.concat((cdf,y),axis=1)
x=pd.get_dummies(cdf["Contract"])
cdf.drop(["Contract"],axis=1,inplace=True)
cdf=pd.concat((cdf,x),axis=1)
cdf.drop(["PaymentMethod"],axis=1,inplace=True)
cdf.drop(["customerID"],axis=1,inplace=True)
cdf['tenure']=cdf['tenure'].replace({0:0.1})

cdf.head()
```

Output:

```
   SeniorCitizen  Partner  Dependents  tenure  PhoneService  MultipleLines  \
0              0        1           0     1.0             0              0
1              0        0           0    34.0             1              0
2              0        0           0     2.0             1              0
3              0        0           0    45.0             0              0
4              0        0           0     2.0             1              0

   OnlineSecurity  OnlineBackup  DeviceProtection  TechSupport  ...  \
0               0             1                 0            0  ...
1               1             0                 1            0  ...
2               1             1                 0            0  ...
3               1             0                 1            1  ...
4               0             0                 0            0  ...

   PaperlessBilling  MonthlyCharges  TotalCharges  Churn Male  DSL  \
0                 1           29.85         29.85      0    0    1
1                 0           56.95        1889.5      0    1    1
2                 1           53.85        108.15      1    1    1
3                 0           42.30       1840.75      0    1    1
4                 1           70.70        151.65      1    0    0

   Fiber optic  Month-to-month  One year  Two year
0            0               1         0         0
1            0               0         1         0
2            0               1         0         0
3            0               0         1         0
4            1               1         0         0

[5 rows x 22 columns]
```

- Now to bring the tenure in normal distribution we took log of it.

- All the NaN values in the 'TotalCharges' col. was replaced with the predicted value .The prediction was done by a LinearRegression Model.

- The 'TotalCharges' and 'MonthlyCharges' variables were scaled by taking their respective logs.

- All the negative values of the dataframe were converted to positive by using abs().

Code:

```
ten=np.log(cdf['tenure'])
cdf.drop(["tenure"],axis=1,inplace=True)
cdf=pd.concat((cdf,ten),axis=1)
cdf['TotalCharges'] =pd.to_numeric(cdf['TotalCharges'], errors ='coerce')
z=cdf.dropna()
index =cdf['TotalCharges'].index[cdf['TotalCharges'].apply(np.isnan)]
c=[]
c.extend(index)
# values of the monthlycharges corresponding to which the values of totalcharges is null or nan!
d=cdf.loc[c,["MonthlyCharges"]]
x= z["MonthlyCharges"].values
m=len(x)
x=x.reshape((m,-1))
y= z["TotalCharges"].values
y = y.reshape((m,-1))
regmodel=linear_model.LinearRegression()
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3,random_state=42)
regmodel.fit(x_train,y_train)
a= regmodel.predict(d)
list1=[]
for i in a:
    list1.extend(i)
r2= regmodel.score(x,y)
print("Accuracy of linear model used for predicting a null values of TotalCharges in the data set : ", r2)
for i in list1:
    cdf['TotalCharges']=(cdf['TotalCharges'].fillna(i,limit= 1))
tc=np.log(cdf['TotalCharges'])
cdf.drop(['TotalCharges'],axis=1,inplace=True)
cdf=pd.concat((cdf,tc),axis=1)
mc=np.log(cdf['MonthlyCharges'])
cdf.drop(['MonthlyCharges'],axis=1,inplace=True)
cdf=pd.concat((cdf,mc),axis=1)
# Changing all negative values into non-negative values:
cdf=cdf.abs()
print(cdf.describe())
cdf.head()
```

Output:

```
Accuracy of linear model used for predicting a null values of TotalCharges
 in the data set :  0.4238558780398537
       SeniorCitizen      Partner   Dependents  PhoneService  MultipleLine
s  \
count    7043.000000  7043.000000  7043.000000   7043.000000    7043.00000
0
mean        0.162147     0.483033     0.299588      0.903166       0.42183
7
std         0.368612     0.499748     0.458110      0.295752       0.49388
8
min         0.000000     0.000000     0.000000      0.000000       0.00000
0
25%         0.000000     0.000000     0.000000      1.000000       0.00000
0
50%         0.000000     0.000000     0.000000      1.000000       0.00000
0
75%         0.000000     1.000000     1.000000      1.000000       1.00000
0
max         1.000000     1.000000     1.000000      1.000000       1.00000
0

       OnlineSecurity  OnlineBackup  DeviceProtection  TechSupport  \
count     7043.000000   7043.000000       7043.000000  7043.000000
mean         0.286668      0.344881          0.343888     0.290217
std          0.452237      0.475363          0.475038     0.453895
min          0.000000      0.000000          0.000000     0.000000
25%          0.000000      0.000000          0.000000     0.000000
50%          0.000000      0.000000          0.000000     0.000000
75%          1.000000      1.000000          1.000000     1.000000
max          1.000000      1.000000          1.000000     1.000000

       StreamingTV  ...        Churn         Male          DSL  Fiber opti
c  \
count  7043.000000  ...  7043.000000  7043.000000  7043.000000  7043.00000
0
mean      0.384353  ...     0.265370     0.504756     0.343746     0.43958
5
std       0.486477  ...     0.441561     0.500013     0.474991     0.49637
2
min       0.000000  ...     0.000000     0.000000     0.000000     0.00000
0
25%       0.000000  ...     0.000000     0.000000     0.000000     0.00000
0
50%       0.000000  ...     0.000000     1.000000     0.000000     0.00000
0
75%       1.000000  ...     1.000000     1.000000     1.000000     1.00000
0
```

```
max       1.000000  ...     1.000000     1.000000     1.000000     1.00000
0

       Month-to-month     One year     Two year       tenure  TotalCharges
  \
count     7043.000000  7043.000000  7043.000000  7043.000000   7043.000000

mean         0.550192     0.209144     0.240664     2.916909      6.938430

std          0.497510     0.406726     0.427517     1.324141      1.553139

min          0.000000     0.000000     0.000000     0.000000      2.933857

25%          0.000000     0.000000     0.000000     2.197225      5.991839

50%          1.000000     0.000000     0.000000     3.367296      7.242297

75%          1.000000     0.000000     0.000000     4.007333      8.239224

max          1.000000     1.000000     1.000000     4.276666      9.069330


       MonthlyCharges
count     7043.000000
mean         4.021917
std          0.594424
min          2.904165
25%          3.569533
50%          4.253483
75%          4.498142
max          4.777020

[8 rows x 22 columns]
```

Glimpse of Prepared Dataset

- Information:

```
cdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 22 columns):
SeniorCitizen       7043 non-null float64
Partner             7043 non-null float64
Dependents          7043 non-null float64
PhoneService        7043 non-null float64
MultipleLines       7043 non-null float64
OnlineSecurity      7043 non-null float64
OnlineBackup        7043 non-null float64
DeviceProtection    7043 non-null float64
TechSupport         7043 non-null float64
StreamingTV         7043 non-null float64
StreamingMovies     7043 non-null float64
PaperlessBilling    7043 non-null float64
Churn               7043 non-null float64
Male                7043 non-null float64
DSL                 7043 non-null float64
Fiber optic         7043 non-null float64
Month-to-month      7043 non-null float64
One year            7043 non-null float64
Two year            7043 non-null float64
tenure              7043 non-null float64
TotalCharges        7043 non-null float64
MonthlyCharges      7043 non-null float64
dtypes: float64(22)
memory usage: 1.2 MB
```

- Description:

```
       SeniorCitizen      Partner   Dependents  PhoneService  MultipleLine
s  \
count    7043.000000  7043.000000  7043.000000   7043.000000    7043.00000
0
mean        0.162147     0.483033     0.299588      0.903166       0.42183
7
```

```
std         0.368612     0.499748     0.458110      0.295752       0.49388
8
min         0.000000     0.000000     0.000000      0.000000       0.00000
0
25%         0.000000     0.000000     0.000000      1.000000       0.00000
0
50%         0.000000     0.000000     0.000000      1.000000       0.00000
0
75%         0.000000     1.000000     1.000000      1.000000       1.00000
0
max         1.000000     1.000000     1.000000      1.000000       1.00000
0

       OnlineSecurity  OnlineBackup  DeviceProtection  TechSupport  \
count     7043.000000   7043.000000       7043.000000  7043.000000
mean         0.286668      0.344881          0.343888     0.290217
std          0.452237      0.475363          0.475038     0.453895
min          0.000000      0.000000          0.000000     0.000000
25%          0.000000      0.000000          0.000000     0.000000
50%          0.000000      0.000000          0.000000     0.000000
75%          1.000000      1.000000          1.000000     1.000000
max          1.000000      1.000000          1.000000     1.000000

       StreamingTV  ...        Churn         Male          DSL  Fiber opti
c  \
count  7043.000000  ...  7043.000000  7043.000000  7043.000000  7043.00000
0
mean      0.384353  ...     0.265370     0.504756     0.343746     0.43958
5
std       0.486477  ...     0.441561     0.500013     0.474991     0.49637
2
min       0.000000  ...     0.000000     0.000000     0.000000     0.00000
0
25%       0.000000  ...     0.000000     0.000000     0.000000     0.00000
0
50%       0.000000  ...     0.000000     1.000000     0.000000     0.00000
0
75%       1.000000  ...     1.000000     1.000000     1.000000     1.00
max       1.000000  ...     1.000000     1.000000     1.000000     1.00




       Month-to-month     One year     Two year       tenure  TotalCharges
  \
count     7043.000000  7043.000000  7043.000000  7043.000000   7043.000000

mean         0.550192     0.209144     0.240664     2.916909      6.938430
```

```
std        0.497510    0.406726    0.427517    1.324141     1.553139

min        0.000000    0.000000    0.000000    0.000000     2.933857

25%        0.000000    0.000000    0.000000    2.197225     5.991839

50%        1.000000    0.000000    0.000000    3.367296     7.242297

75%        1.000000    0.000000    0.000000    4.007333     8.239224

max        1.000000    1.000000    1.000000    4.276666     9.069330

       MonthlyCharges
count     7043.000000
mean         4.021917
std          0.594424
min          2.904165
25%          3.569533
50%          4.253483
75%          4.498142
max          4.777020

[8 rows x 22 columns]
```

- First ‘5’ row of dataset:

```
print(cdf.head())
```

```
   SeniorCitizen  Partner  Dependents  PhoneService  MultipleLines  \
0            0.0      1.0         0.0           0.0            0.0
1            0.0      0.0         0.0           1.0            0.0
2            0.0      0.0         0.0           1.0            0.0
3            0.0      0.0         0.0           0.0            0.0
4            0.0      0.0         0.0           1.0            0.0

   OnlineSecurity  OnlineBackup  DeviceProtection  TechSupport  StreamingTV  \
0             0.0           1.0               0.0          0.0          0.0
1             1.0           0.0               1.0          0.0          0.0
2             1.0           1.0               0.0          0.0          0.0
3             1.0           0.0               1.0          1.0          0.0
4             0.0           0.0               0.0          0.0          0.0

   ...  Churn  Male  DSL  Fiber optic  Month-to-month  One year  Two year  \
0  ...    0.0   0.0  1.0          0.0             1.0       0.0       0.0
1  ...    0.0   1.0  1.0          0.0             0.0       1.0       0.0
2  ...    1.0   1.0  1.0          0.0             1.0       0.0       0.0
3  ...    0.0   1.0  1.0          0.0             0.0       1.0       0.0
4  ...    1.0   0.0  0.0          1.0             1.0       0.0       0.0

     tenure  TotalCharges  MonthlyCharges
0  0.000000      3.396185        3.396185
1  3.526361      7.544068        4.042174
2  0.693147      4.683519        3.986202
3  3.806662      7.517928        3.744787
4  0.693147      5.021575        4.258446

[5 rows x 22 columns]
```

➢ Last '5' row of the dataset:

```
print(cdf.tail())
```

```
      SeniorCitizen  Partner  Dependents  PhoneService  MultipleLines  \
7038            0.0      1.0         1.0           1.0            1.0
7039            0.0      1.0         1.0           1.0            1.0
7040            0.0      1.0         1.0           0.0            0.0
7041            1.0      1.0         0.0           1.0            1.0
7042            0.0      0.0         0.0           1.0            0.0

      OnlineSecurity  OnlineBackup  DeviceProtection  TechSupport  \
7038             1.0           0.0               1.0          1.0
7039             0.0           1.0               1.0          0.0
7040             1.0           0.0               0.0          0.0
7041             0.0           0.0               0.0          0.0
7042             1.0           0.0               1.0          1.0

      StreamingTV  ...  Churn  Male  DSL  Fiber optic  Month-to-month  \
7038          1.0  ...    0.0   1.0  1.0          0.0             0.0
7039          1.0  ...    0.0   0.0  0.0          1.0             0.0
7040          0.0  ...    0.0   0.0  1.0          0.0             1.0
7041          0.0  ...    1.0   1.0  0.0          1.0             1.0
7042          1.0  ...    0.0   1.0  0.0          1.0             0.0

      One year  Two year    tenure  TotalCharges  MonthlyCharges
7038       1.0       0.0  3.178054      7.596141        4.440296
7039       1.0       0.0  4.276666      8.904209        4.636669
7040       0.0       0.0  2.397895      5.847739        3.387774
7041       0.0       0.0  1.386294      5.725544        4.309456
7042       0.0       1.0  4.189655      8.831201        4.660132

[5 rows x 22 columns]
```

Feature Selection & Selection of features using f_classif and chi2 method

Features with their scores:

- Code:-

```python
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2,f_classif

print()
print("Best feature selection")
for i in range(1,22):
    print("Best feature selection using f_classif")
    print()
    print("Number of Columns:",i )
    print()
    bestfeatures=SelectKBest(score_func=f_classif,k=i)
    fit=bestfeatures.fit_transform(x,y)
    col1=x.columns.values[bestfeatures.get_support()]
    scores=bestfeatures.scores_[bestfeatures.get_support()]
    name_scores=list(zip(col1,scores))
    print()
    print(name_scores)
    print()
    print("Best feature selection using chi2")
    print()
    print("Number of Columns:",i )
    print()
    bestfeatures=SelectKBest(score_func=f_classif,k=i)
    fit=bestfeatures.fit_transform(x,y)
    col1=x.columns.values[bestfeatures.get_support()]
    scores=bestfeatures.scores_[bestfeatures.get_support()]
    name_scores=list(zip(col1,scores))
    print()
    print(name_scores)
    print()
```

Output:

- **Best feature selection**

Best feature selection using f_classif

Number of Columns: 1

[('Month-to-month', 1382.340696976842)]

Best feature selection using chi2

Number of Columns: 1

[('Month-to-month', 1382.340696976842)]

Best feature selection using f_classif

Number of Columns: 2

[('Month-to-month', 1382.340696976842), ('tenure', 1162.7579488308727)]

Best feature selection using chi2

Number of Columns: 2

[('Month-to-month', 1382.340696976842), ('tenure', 1162.7579488308727)]

Best feature selection using f_classif

Number of Columns: 3

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('tenure', 1162.7579488308727)]

Best feature selection using chi2

Number of Columns: 3

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('tenure', 1162.7579488308727)]

Best feature selection using f_classif

Number of Columns: 4

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727)]

Best feature selection using chi2

Number of Columns: 4

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727)]

Best feature selection using f_classif

Number of Columns: 5

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308)]

Best feature selection using chi2

Number of Columns: 5

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308)]

Best feature selection using f_classif

Number of Columns: 6

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two yea
r', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002
950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 6

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two yea
r', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002
950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 7

[('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to
-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.75794
88308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.889786539909
3)]

Best feature selection using chi2

Number of Columns: 7

[('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to
-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.75794
88308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.889786539909
3)]

Best feature selection using f_classif

Number of Columns: 8

[('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to
-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.919
2540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('
MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 8

[('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 9

[('OnlineSecurity', 212.66619940319887), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 9

[('OnlineSecurity', 212.66619940319887), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 10

[('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 10

[('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 11

[('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 11

[('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 12

[('SeniorCitizen', 164.04142445613567), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 12

[('SeniorCitizen', 164.04142445613567), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 13

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 13

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 14

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 14

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25
540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901),
('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year
', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308
727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 15

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.962
95545820452), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.985218092
8093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-mo
nth', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.919254
0580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('Mon
thlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 15

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.962
95545820452), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.985218092
8093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-mo
nth', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.919254
0580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('Mon
thlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 16

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 16

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 17

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 17

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('PaperlessBilling', 268.9852180928093)

, ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 18

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 18

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 19

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('MultipleLines', 11.341439011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 13

82.340696976842), ('One year', 229.90574088120178), ('Two year', 707.919254058077
9), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCha
rges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 19

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('MultipleLines', 11.341439011576513), ('OnlineSecurity', 212.66
619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.95478043
9130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395)
, ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('
DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 13
82.340696976842), ('One year', 229.90574088120178), ('Two year', 707.919254058077
9), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCha
rges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 20

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('PhoneService', 1.0042664747911525), ('MultipleLines', 11.3414
39011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.9629554582
0452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175)
, ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('Pa
perlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.04
60424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120
178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges
', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 20

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('PhoneService', 1.0042664747911525), ('MultipleLines', 11.3414
39011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.9629554582
0452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175)
, ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('Pa

perlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using f_classif

Number of Columns: 21

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('PhoneService', 1.0042664747911525), ('MultipleLines', 11.341439011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('Male', 0.5222569018409975), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Best feature selection using chi2

Number of Columns: 21

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('PhoneService', 1.0042664747911525), ('MultipleLines', 11.341439011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('Male', 0.5222569018409975), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Input Feature Selection

- Feature selection based on the score :

Performance of the different models (Logistic Regression, Decision tree, KNN, NB) based on kth best number of feature selection using f_classif and chi2 method respectively

Feature selection
Code:

```
x=cdf.drop('Churn',axis=1)
y=cdf['Churn']
list_a=[]
list_f=[]
list1=[f_classif,chi2]
for w in list1:

    print("The feature selection based on", w," Method:")
    print()
    for i in range(1,22):
        print("No. of features selected:   ",i )
        print()
        bestfeatures=SelectKBest(score_func=w,k=i)
        fit=bestfeatures.fit_transform(x,y)
        col1=x.columns.values[bestfeatures.get_support()]
        scores=bestfeatures.scores_[bestfeatures.get_support()]
        print("The name of the columns corresponding to the scores:")
        print()
        name_scores=list(zip(col1,scores))
        print(name_scores)
        x1=cdf[list(col1)]
        x_train,x_test,y_train,y_test= train_test_split(x1,y,test_size=0.3,random_state=1)
        kfold = model_selection.KFold(n_splits=10, random_state =42)
        logmodel= LogisticRegression(solver='lbfgs', multi_class='auto')
        logmodel.fit(x_train,y_train)
        pre=logmodel.predict(x_test)
        print()
        print("Performance of the different models with",i,"th","best columns of the dataset: ")
        print()
        print("Accuracy of the logistic model:  " , accuracy_score(y_test, pre)*100)
        print("F1_score of the model: ", f1_score(y_test,pre)*100)
```

```python
        list_a.append(accuracy_score(y_test, pre))
        list_f.append(f1_score(y_test,pre))
        res=confusion_matrix(y_test,pre)
        print(res)
        print()
        print()

#decision tree

        classifier_entropy = DecisionTreeClassifier(criterion='entropy', random_state=42, max_depth = 20 )
        classifier_entropy.fit(x_train,y_train)
        y_ped=classifier_entropy.predict(x_test)
        print("Accuracy of Decision tree model: ",accuracy_score(y_test,y_ped)*100)
        print("F1_score of Decision tree model: ", f1_score(y_test,y_ped)*100)
        print(confusion_matrix(y_test,y_ped))
        print()
        print()
#KNN
        classifier=KNeighborsClassifier(n_neighbors=83,metric='euclidean')
        classifier.fit(x_train,y_train)
        y_pred =classifier.predict(x_test)
        print("Accuracy of KNN model: ",accuracy_score(y_test,y_pred)*100)
        print("F1_score of KNN model: ", f1_score(y_test,y_pred)*100)
        print(confusion_matrix(y_test,y_pred))
        print()
```

```python
#NB
        print()
        label=list(y_train)
        l=x_train
        model=GaussianNB()
        model.fit(l,label)
        predicted=model.predict(x_test)
        print(" Accuracy of Naive byes model: ",accuracy_score(y_test,predicted)*100)
        print("F1 score of Naive byes model:  ",f1_score(y_test,predicted)*100)
        print(confusion_matrix(y_test,predicted))
```

OUTPUT:

No. of features selected: 1

The name of the columns corresponding to the scores:

[('Month-to-month', 1382.340696976842)]

Performance of the different models with 1st best columns of the dataset:

Accuracy of the logistic model: 75.01183151916706
F1_score of the model: 0.0
[[1585 0]
 [528 0]]

Accuracy of Decision tree model: 75.01183151916706
F1_score of Decision tree model: 0.0
[[1585   0]
 [528   0]]

Accuracy of KNN model: 75.01183151916706
F1_score of KNN model: 0.0
[[1585   0]
 [528   0]]

 Accuracy of Naive byes model: 65.97255087553242
F1 score of Naive byes model: 57.07462686567164
[[916 669]
 [50 478]]

No. of features selected: 2

The name of the columns corresponding to the scores:

[('Month-to-month', 1382.340696976842), ('tenure', 1162.7579488308727)]

Performance of the different models with 2nd best columns of the dataset:

Accuracy of the logistic model: 77.04685281590156
F1_score of the model: 41.91616766467065
[[1453 132]
 [353 175]]

Accuracy of Decision tree model: 76.28963558920965
F1_score of Decision tree model: 42.74285714285715
[[1425 160]
 [341 187]]

Accuracy of KNN model: 76.52626597255087
F1_score of KNN model: 40.24096385542168
[[1450 135]
 [361 167]]

Accuracy of Naive byes model: 75.9110269758637
F1 score of Naive byes model: 53.25987144168962
[[1314 271]
 [238 290]]

No. of features selected: 3

The name of the columns corresponding to the scores:

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('tenure', 1162.7579488308727)]

Performance of the different models with 3rd best columns of the dataset:

Accuracy of the logistic model: 80.07572172266919
F1_score of the model: 51.66475315729048
[[1467 118]
 [303 225]]

Accuracy of Decision tree model: 78.75059157595835
F1_score of Decision tree model: 56.534365924491766
[[1372 213]
 [236 292]]

Accuracy of KNN model: 79.27117841930904
F1_score of KNN model: 56.1122244488978
[[1395 190]
 [248 280]]

Accuracy of Naive byes model: 77.56743965925224
F1 score of Naive byes model: 62.20095693779905
[[1249 336]
 [138 390]]

No. of features selected: 4

The name of the columns corresponding to the scores:

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two yea
r', 707.9192540580779), ('tenure', 1162.7579488308727)]

Performance of the different models with 4th best columns of the dataset:

```
Accuracy of the logistic model:   80.07572172266919
F1_score of the model:  51.66475315729048
[[1467  118]
 [ 303  225]]


Accuracy of Decision tree model:  78.84524372929485
F1_score of Decision tree model:  56.72797676669894
[[1373  212]
 [ 235  293]]


Accuracy of KNN model:  79.27117841930904
F1_score of KNN model:  56.1122244488978
[[1395  190]
 [ 248  280]]


 Accuracy of Naive byes model:  70.9891150023663
F1 score of Naive byes model:   60.01304631441617
[[1040  545]
 [  68  460]]
```

No. of features selected: 5

The name of the columns corresponding to the scores:

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two yea
r', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002
950308)]

Performance of the different models with 5th best columns of the dataset:

```
Accuracy of the logistic model:   80.31235210601041
F1_score of the model:  55.17241379310344
[[1441  144]
 [ 272  256]]
```

```
Accuracy of Decision tree model:  74.82252721249408
F1_score of Decision tree model:  51.28205128205128
[[1301  284]
 [ 248  280]]


Accuracy of KNN model:  79.5551348793185
F1_score of KNN model:  54.33403805496829
[[1424  161]
 [ 271  257]]


 Accuracy of Naive byes model:  71.08376715570279
F1 score of Naive byes model:   59.88181221273802
[[1046  539]
 [  72  456]]


No. of features selected:    6

The name of the columns corresponding to the scores:

[('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two yea
r', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002
950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 6th best columns of the dataset:

Accuracy of the logistic model:   80.17037387600567
F1_score of the model:  54.80043149946062
[[1440  145]
 [ 274  254]]


Accuracy of Decision tree model:  75.57974443918599
F1_score of Decision tree model:  50.66921606118547
[[1332  253]
 [ 263  265]]


Accuracy of KNN model:  79.83909133932798
F1_score of KNN model:  55.1578947368421
[[1425  160]
```

[266 262]]

Accuracy of Naive byes model: 72.9294841457643
F1 score of Naive byes model: 61.035422343324264
[[1093 492]
 [80 448]]

No. of features selected: 7

The name of the columns corresponding to the scores:

[('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 7th best columns of the dataset:

Accuracy of the logistic model: 80.50165641268339
F1_score of the model: 56.076759061833684
[[1438 147]
 [265 263]]

Accuracy of Decision tree model: 73.73402744912447
F1_score of Decision tree model: 46.48023143683703
[[1317 268]
 [287 241]]

Accuracy of KNN model: 80.21769995267393
F1_score of KNN model: 57.17213114754098
[[1416 169]
 [249 279]]

Accuracy of Naive byes model: 73.11878845243729
F1 score of Naive byes model: 61.36054421768707
[[1094 491]
 [77 451]]

No. of features selected: 8

The name of the columns corresponding to the scores:

[('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 8th best columns of the dataset:

Accuracy of the logistic model: 80.50165641268339
F1_score of the model: 56.076759061833684
[[1438 147]
[265 263]]

Accuracy of Decision tree model: 74.20728821580691
F1_score of Decision tree model: 48.14462416745956
[[1315 270]
[275 253]]

Accuracy of KNN model: 80.26502602934217
F1_score of KNN model: 57.230769230769226
[[1417 168]
[249 279]]

Accuracy of Naive byes model: 71.17841930903927
F1 score of Naive byes model: 60.27397260273971
[[1042 543]
[66 462]]

No. of features selected: 9

The name of the columns corresponding to the scores:

[('OnlineSecurity', 212.66619940319887), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 9th best columns of the dataset:

Accuracy of the logistic model: 80.21769995267393
F1_score of the model: 56.27615062761506
[[1426 159]
 [259 269]]

Accuracy of Decision tree model: 74.39659252247989
F1_score of Decision tree model: 47.526673132880696
[[1327 258]
 [283 245]]

Accuracy of KNN model: 80.02839564600094
F1_score of KNN model: 56.76229508196721
[[1414 171]
 [251 277]]

Accuracy of Naive byes model: 71.84098438239471
F1 score of Naive byes model: 60.465116279069775
[[1063 522]
 [73 455]]

No. of features selected: 10

The name of the columns corresponding to the scores:

[('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 10th best columns of the dataset:

Accuracy of the logistic model: 80.17037387600567
F1_score of the model: 56.759545923632615
[[1419 166]
 [253 275]]

```
Accuracy of Decision tree model:  73.92333175579743
F1_score of Decision tree model:  48.83936861652739
[[1299  286]
 [ 265  263]]


Accuracy of KNN model:  80.21769995267393
F1_score of KNN model:  58.36653386454182
[[1402  183]
 [ 235  293]]


 Accuracy of Naive byes model:  72.50354945575012
F1 score of Naive byes model:   60.76975016880487
[[1082  503]
 [  78  450]]


No. of features selected:    11

The name of the columns corresponding to the scores:

[('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSu
pport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 73
8.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088
120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCha
rges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 11 th best columns of the dataset:

Accuracy of the logistic model:   80.26502602934217
F1_score of the model:  57.230769230769226
[[1417  168]
 [ 249  279]]


Accuracy of Decision tree model:  74.34926644581165
F1_score of Decision tree model:  49.81481481481482
[[1302  283]
 [ 259  269]]


Accuracy of KNN model:  80.26502602934217
F1_score of KNN model:  59.39629990262901
```

```
[[1391  194]
 [ 223  305]]


 Accuracy of Naive byes model:  73.16611452910554
F1 score of Naive byes model:   61.34969325153374
[[1096  489]
 [  78  450]]


No. of features selected:    12

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Dependents', 195.1493137732415), ('OnlineSe
curity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling',
268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.3406
96976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('ten
ure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 3
25.8897865399093)]

Performance of the different models with 12th best columns of the dataset:

Accuracy of the logistic model:   80.78561287269285
F1_score of the model:  58.4016393442623
[[1422  163]
 [ 243  285]]


Accuracy of Decision tree model:  74.86985328916232
F1_score of Decision tree model:  52.1190261496844
[[1293  292]
 [ 239  289]]


Accuracy of KNN model:  79.9810695693327
F1_score of KNN model:  57.91044776119403
[[1399  186]
 [ 237  291]]


 Accuracy of Naive byes model:  73.450070989115
F1 score of Naive byes model:   61.44329896907216
[[1105  480]
```

[81   447]]

No. of features selected: 13

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 13 th best columns of the dataset:

Accuracy of the logistic model: 80.69096071935637
F1_score of the model: 58.282208588957054
[[1420   165]
 [243   285]]

Accuracy of Decision tree model: 74.82252721249408
F1_score of Decision tree model: 50.280373831775705
[[1312   273]
 [259   269]]

Accuracy of KNN model: 80.17037387600567
F1_score of KNN model: 58.39126117179742
[[1400   185]
 [234   294]]

Accuracy of Naive byes model: 73.40274491244676
F1 score of Naive byes model: 61.401098901098905
[[1104   481]
 [81   447]]

No. of features selected: 14

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 14 th best columns of the dataset:

Accuracy of the logistic model: 80.50165641268339
F1_score of the model: 57.70020533880904
[[1420 165]
[247 281]]

Accuracy of Decision tree model: 74.86985328916232
F1_score of Decision tree model: 51.14995400183992
[[1304 281]
[250 278]]

Accuracy of KNN model: 80.64363464268813
F1_score of KNN model: 59.86261040235525
[[1399 186]
[223 305]]

Accuracy of Naive byes model: 73.82867960246095
F1 score of Naive byes model: 61.62387231089521
[[1116 469]
[84 444]]

No. of features selected: 15

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 15th best columns of the dataset:

Accuracy of the logistic model: 80.12304779933743
F1_score of the model: 57.31707317073172
[[1411 174]
 [246 282]]

Accuracy of Decision tree model: 75.76904874585897
F1_score of Decision tree model: 52.059925093632955
[[1323 262]
 [250 278]]

Accuracy of KNN model: 80.35967818267865
F1_score of KNN model: 59.353574926542606
[[1395 190]
 [225 303]]

Accuracy of Naive byes model: 73.68670137245623
F1 score of Naive byes model: 61.495844875346265
[[1113 472]
 [84 444]]

No. of features selected: 16

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 16th best columns of the dataset:

Accuracy of the logistic model: 79.9810695693327
F1_score of the model: 56.96846388606307
[[1410 175]

[248 280]]

Accuracy of Decision tree model: 74.86985328916232
F1_score of Decision tree model: 51.14995400183992
[[1304 281]
 [250 278]]

Accuracy of KNN model: 80.59630856601989
F1_score of KNN model: 59.4059405940594
[[1403 182]
 [228 300]]

Accuracy of Naive byes model: 73.82867960246095
F1 score of Naive byes model: 61.570535093815145
[[1117 468]
 [85 443]]

No. of features selected: 17

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 17th best columns of the dataset:

Accuracy of the logistic model: 80.8329389493611
F1_score of the model: 58.79959308240081
[[1419 166]
 [239 289]]

Accuracy of Decision tree model: 74.49124467581638
F1_score of Decision tree model: 49.9535747446611

[[1305 280]
 [259 269]]

Accuracy of KNN model: 80.50165641268339
F1_score of KNN model: 58.88223552894212
[[1406 179]
 [233 295]]

Accuracy of Naive byes model: 73.49739706578325
F1 score of Naive byes model: 61.11111111111111
[[1113 472]
 [88 440]]

No. of features selected: 18

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents', 195.1493137732415), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 18 th best columns of the dataset:

Accuracy of the logistic model: 81.21154756270705
F1_score of the model: 59.61342828077314
[[1423 162]
 [235 293]]

Accuracy of Decision tree model: 73.68670137245623
F1_score of Decision tree model: 49.270072992700726
[[1287 298]
 [258 270]]

Accuracy of KNN model: 81.11689540937056

F1_score of KNN model: 60.6896551724138
[[1406 179]
 [220 308]]

 Accuracy of Naive byes model: 73.73402744912447
F1 score of Naive byes model: 61.43154968728284
[[1116 469]
 [86 442]]

No. of features selected: 19

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('MultipleLines', 11.341439011576513), ('OnlineSecurity', 212.66
619940319887), ('OnlineBackup', 47.96295545820452), ('DeviceProtection', 30.95478043
9130214), ('TechSupport', 196.25540507248175), ('StreamingTV', 28.261123665052395)
, ('StreamingMovies', 26.628665202834036), ('PaperlessBilling', 268.9852180928093), ('
DSL', 110.33853175234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 13
82.340696976842), ('One year', 229.90574088120178), ('Two year', 707.919254058077
9), ('tenure', 1162.7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCha
rges', 325.8897865399093)]

Performance of the different models with 19 th best columns of the dataset:

Accuracy of the logistic model: 81.16422148603881
F1_score of the model: 59.797979797979785
[[1419 166]
 [232 296]]

Accuracy of Decision tree model: 73.63937529578799
F1_score of Decision tree model: 50.22341376228775
[[1275 310]
 [247 281]]

Accuracy of KNN model: 80.8329389493611
F1_score of KNN model: 59.621136590229305
[[1409 176]
 [229 299]]

Accuracy of Naive byes model: 73.78135352579271
F1 score of Naive byes model: 61.42061281337048
[[1118 467]
[87 441]]

No. of features selected: 20

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('PhoneService', 1.0042664747911525), ('MultipleLines', 11.3414
39011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.9629554582
0452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175)
, ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('Pa
perlessBilling', 268.9852180928093), ('DSL', 110.33853175234901), ('Fiber optic', 738.04
60424544476), ('Month-to-month', 1382.340696976842), ('One year', 229.90574088120
178), ('Two year', 707.9192540580779), ('tenure', 1162.7579488308727), ('TotalCharges
', 434.5374002950308), ('MonthlyCharges', 325.8897865399093)]

Performance of the different models with 20 th best columns of the dataset:

Accuracy of the logistic model: 81.30619971604354
F1_score of the model: 59.89847715736041
[[1423 162]
[233 295]]

Accuracy of Decision tree model: 73.07146237576904
F1_score of Decision tree model: 49.51197870452529
[[1265 320]
[249 279]]

Accuracy of KNN model: 80.78561287269285
F1_score of KNN model: 59.88142292490117
[[1404 181]
[225 303]]

Accuracy of Naive byes model: 74.11263606247041
F1 score of Naive byes model: 61.72148355493352
[[1125 460]

```
 [  87  441]]


No. of features selected:     21

The name of the columns corresponding to the scores:

[('SeniorCitizen', 164.04142445613567), ('Partner', 163.06003598399556), ('Dependents'
, 195.1493137732415), ('PhoneService', 1.0042664747911525), ('MultipleLines', 11.3414
39011576513), ('OnlineSecurity', 212.66619940319887), ('OnlineBackup', 47.9629554582
0452), ('DeviceProtection', 30.954780439130214), ('TechSupport', 196.25540507248175)
, ('StreamingTV', 28.261123665052395), ('StreamingMovies', 26.628665202834036), ('Pa
perlessBilling', 268.9852180928093), ('Male', 0.5222569018409975), ('DSL', 110.338531
75234901), ('Fiber optic', 738.0460424544476), ('Month-to-month', 1382.340696976842)
, ('One year', 229.90574088120178), ('Two year', 707.9192540580779), ('tenure', 1162.
7579488308727), ('TotalCharges', 434.5374002950308), ('MonthlyCharges', 325.8897865
399093)]

Performance of the different models with 21 th best columns of the dataset:

Accuracy of the logistic model:    81.21154756270705
F1_score of the model:   59.77710233029382
[[1421  164]
 [ 233  295]]


Accuracy of Decision tree model:   74.15996213913867
F1_score of Decision tree model:   49.81617647058824
[[1296  289]
 [ 257  271]]


Accuracy of KNN model:   80.88026502602933
F1_score of KNN model:   60.15779092702169
[[1404  181]
 [ 223  305]]


 Accuracy of Naive byes model:   74.06530998580217
F1 score of Naive byes model:    61.678321678321666
[[1124  461]
 [  87  441]]
```

Feature selection using chi2 method

The feature selection based on <function chi2 at 0x0000015AA0FE91F8> Method:

No. of features selected: 1

The name of the columns corresponding to the scores:

[('tenure', 599.9539968592322)]

Performance of the different models with 1st best columns of the dataset:

```
Accuracy of the logistic model:   76.7628963558921
F1_score of the model:  41.617122473246134
[[1447  138]
 [ 353  175]]


Accuracy of Decision tree model:  76.62091812588736
F1_score of Decision tree model:  36.828644501278774
[[1475  110]
 [ 384  144]]


Accuracy of KNN model:  76.62091812588736
F1_score of KNN model:  36.828644501278774
[[1475  110]
 [ 384  144]]


 Accuracy of Naive byes model:  76.7628963558921
F1 score of Naive byes model:   41.617122473246134
[[1447  138]
 [ 353  175]]
```

No. of features selected: 2

The name of the columns corresponding to the scores:

[('Month-to-month', 519.8953106092886), ('tenure', 599.9539968592322)]

Performance of the different models with 2nd best columns of the dataset:

```
Accuracy of the logistic model:   77.04685281590156
F1_score of the model:  41.91616766467065
[[1453  132]
 [ 353  175]]


Accuracy of Decision tree model:  76.28963558920965
F1_score of Decision tree model:  42.74285714285715
[[1425  160]
 [ 341  187]]


Accuracy of KNN model:  76.52626597255087
F1_score of KNN model:  40.24096385542168
[[1450  135]
 [ 361  167]]


 Accuracy of Naive byes model:  75.9110269758637
F1 score of Naive byes model:   53.25987144168962
[[1314  271]
 [ 238  290]]
No. of features selected:     3

The name of the columns corresponding to the scores:

[('Month-to-month', 519.8953106092886), ('Two year', 488.578090256898), ('tenure', 59
9.9539968592322)]

Performance of the different models with 3rd best columns of the dataset:

Accuracy of the logistic model:   77.04685281590156
F1_score of the model:  41.91616766467065
[[1453  132]
 [ 353  175]]


Accuracy of Decision tree model:  76.28963558920965
F1_score of Decision tree model:  42.74285714285715
[[1425  160]
 [ 341  187]]
```

Accuracy of KNN model: 76.62091812588736
F1_score of KNN model: 40.33816425120773
[[1452 133]
 [361 167]]

 Accuracy of Naive byes model: 65.83057264552768
F1 score of Naive byes model: 56.97258641239571
[[913 672]
 [50 478]]
No. of features selected: 4

The name of the columns corresponding to the scores:

[('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('Two yea r', 488.578090256898), ('tenure', 599.9539968592322)]

Performance of the different models with 4 th best columns of the dataset:

Accuracy of the logistic model: 80.07572172266919
F1_score of the model: 51.66475315729048
[[1467 118]
 [303 225]]

Accuracy of Decision tree model: 78.84524372929485
F1_score of Decision tree model: 56.72797676669894
[[1373 212]
 [235 293]]

Accuracy of KNN model: 79.27117841930904
F1_score of KNN model: 56.1122244488978
[[1395 190]
 [248 280]]

 Accuracy of Naive byes model: 70.9891150023663
F1 score of Naive byes model: 60.01304631441617
[[1040 545]
 [68 460]]
No. of features selected: 5

The name of the columns corresponding to the scores:

[('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One yea
r', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.95399685923
22)]

Performance of the different models with 5 th best columns of the dataset:

Accuracy of the logistic model: 80.07572172266919
F1_score of the model: 51.66475315729048
[[1467 118]
 [303 225]]

Accuracy of Decision tree model: 78.84524372929485
F1_score of Decision tree model: 56.72797676669894
[[1373 212]
 [235 293]]

Accuracy of KNN model: 79.27117841930904
F1_score of KNN model: 56.1122244488978
[[1395 190]
 [248 280]]

Accuracy of Naive byes model: 65.97255087553242
F1 score of Naive byes model: 57.07462686567164
[[916 669]
 [50 478]]
No. of features selected: 6

The name of the columns corresponding to the scores:

[('OnlineSecurity', 147.29585790556192), ('Fiber optic', 374.4762164498734), ('Month-to
-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578
090256898), ('tenure', 599.9539968592322)]

Performance of the different models with 6 th best columns of the dataset:

Accuracy of the logistic model: 79.69711310932324
F1_score of the model: 50.74626865671642
[[1463 122]
 [307 221]]

Accuracy of Decision tree model: 78.23000473260767
F1_score of Decision tree model: 55.42635658914728
[[1367 218]
 [242 286]]

Accuracy of KNN model: 79.27117841930904
F1_score of KNN model: 55.48780487804878
[[1402 183]
 [255 273]]

Accuracy of Naive byes model: 68.9540937056318
F1 score of Naive byes model: 58.63808322824716
[[992 593]
 [63 465]]
No. of features selected: 7

The name of the columns corresponding to the scores:

[('OnlineSecurity', 147.29585790556192), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 7 th best columns of the dataset:

Accuracy of the logistic model: 80.12304779933743
F1_score of the model: 55.0321199143469
[[1436 149]
 [271 257]]

Accuracy of Decision tree model: 75.57974443918599
F1_score of Decision tree model: 53.2608695652174
[[1303 282]
 [234 294]]

Accuracy of KNN model: 79.2238523426408
F1_score of KNN model: 52.74488697524219
[[1429 156]
 [283 245]]

Accuracy of Naive byes model: 70.04259346900142
F1 score of Naive byes model: 59.08209437621203
[[1023 562]
 [71 457]]
No. of features selected: 8

The name of the columns corresponding to the scores:

[('OnlineSecurity', 147.29585790556192), ('TechSupport', 135.55978268636352), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 8 th best columns of the dataset:

Accuracy of the logistic model: 80.12304779933743
F1_score of the model: 56.06694560669456
[[1425 160]
 [260 268]]

Accuracy of Decision tree model: 76.24230951254141
F1_score of Decision tree model: 53.69003690036901
[[1320 265]
 [237 291]]

Accuracy of KNN model: 79.5551348793185
F1_score of KNN model: 54.23728813559322
[[1425 160]
 [272 256]]

Accuracy of Naive byes model: 70.37387600567912
F1 score of Naive byes model: 59.297789336801046
[[1031 554]
 [72 456]]
No. of features selected: 9

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('OnlineSecurity', 147.29585790556192), ('TechSupport', 135.55978268636352), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 9 th best columns of the dataset:

Accuracy of the logistic model: 79.649787032655
F1_score of the model: 55.578512396694215
[[1414 171]
 [259 269]]

Accuracy of Decision tree model: 75.4850922858495
F1_score of Decision tree model: 52.4770642201835
[[1309 276]
 [242 286]]

Accuracy of KNN model: 79.9810695693327
F1_score of KNN model: 56.52620760534429
[[1415 170]
 [253 275]]

Accuracy of Naive byes model: 70.89446284902982
F1 score of Naive byes model: 59.29847782925215
[[1050 535]
 [80 448]]
No. of features selected: 10

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('TechSupport', 135.55978268636352), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 10 th best columns of the dataset:

Accuracy of the logistic model: 79.83909133932798
F1_score of the model: 56.17283950617284
[[1414 171]
 [255 273]]

```
Accuracy of Decision tree model:  75.0591575958353
F1_score of Decision tree model:  52.39385727190605
[[1296  289]
 [ 238  290]]


Accuracy of KNN model:  79.88641741599622
F1_score of KNN model:  56.04963805584282
[[1417  168]
 [ 257  271]]


 Accuracy of Naive byes model:  71.60435399905349
F1 score of Naive byes model:   59.94659546061416
[[1064  521]
 [  79  449]]
No. of features selected:    11

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Dependents', 133.03644287868082), ('OnlineS
ecurity', 147.29585790556192), ('TechSupport', 135.55978268636352), ('PaperlessBilling'
, 105.68086299962546), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.89
53106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('t
enure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 11 th best columns of the dataset:

Accuracy of the logistic model:   80.88026502602933
F1_score of the model:  58.52156057494866
[[1424  161]
 [ 243  285]]


Accuracy of Decision tree model:  74.68054898248934
F1_score of Decision tree model:  50.41705282669139
[[1306  279]
 [ 256  272]]


Accuracy of KNN model:  80.54898248935163
F1_score of KNN model:  58.35866261398176
[[1414  171]
```

[240 288]]

Accuracy of Naive byes model: 71.93563653573119
F1 score of Naive byes model: 60.06734006734007
[[1074 511]
 [82 446]]
No. of features selected: 12

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('TechSupport', 135.55978268636352), ('PaperlessBilling', 105.68086299962546), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 12 th best columns of the dataset:

Accuracy of the logistic model: 81.02224325603407
F1_score of the model: 59.03983656792645
[[1423 162]
 [239 289]]

Accuracy of Decision tree model: 74.96450544249882
F1_score of Decision tree model: 50.7906976744186
[[1311 274]
 [255 273]]

Accuracy of KNN model: 80.35967818267865
F1_score of KNN model: 57.69622833843017
[[1415 170]
 [245 283]]

Accuracy of Naive byes model: 71.55702792238523
F1 score of Naive byes model: 59.47403910991233
[[1071 514]
 [87 441]]
No. of features selected: 13

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('TechSupport', 135.559 78268636352), ('PaperlessBilling', 105.68086299962546), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 13 th best columns of the dataset:

Accuracy of the logistic model: 80.45433033601515
F1_score of the model: 57.90010193679919
[[1416 169]
[244 284]]

Accuracy of Decision tree model: 74.91717936583058
F1_score of Decision tree model: 50.83487940630798
[[1309 276]
[254 274]]

Accuracy of KNN model: 80.88026502602933
F1_score of KNN model: 60.079051383399204
[[1405 180]
[224 304]]

Accuracy of Naive byes model: 72.55087553241836
F1 score of Naive byes model: 60.27397260273973
[[1093 492]
[88 440]]
No. of features selected: 14

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.217 694023039066), ('TechSupport', 135.55978268636352), ('PaperlessBilling', 105.68086299 962546), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-m onth', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.57809 0256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349)]

Performance of the different models with 14 th best columns of the dataset:

Accuracy of the logistic model: 80.35967818267865
F1_score of the model: 57.868020304568525
[[1413 172]
 [243 285]]

Accuracy of Decision tree model: 75.0591575958353
F1_score of Decision tree model: 52.47971145175835
[[1295 290]
 [237 291]]

Accuracy of KNN model: 80.26502602934217
F1_score of KNN model: 58.50746268656716
[[1402 183]
 [234 294]]

Accuracy of Naive byes model: 72.40889730241364
F1 score of Naive byes model: 60.25903203817313
[[1088 497]
 [86 442]]
No. of features selected: 15

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.217694023039066), ('TechSupport', 135.55978268636352), ('PaperlessBilling', 105.68086299962546), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349), ('MonthlyCharges', 27.36797848545879)]

Performance of the different models with 15 th best columns of the dataset:

Accuracy of the logistic model: 80.12304779933743
F1_score of the model: 57.31707317073172
[[1411 174]
 [246 282]]

Accuracy of Decision tree model: 75.76904874585897

```
F1_score of Decision tree model:  52.059925093632955
[[1323  262]
 [ 250  278]]


C:\Users\Slok\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:947: Converge
nceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
Accuracy of KNN model:  80.35967818267865
F1_score of KNN model:  59.353574926542606
[[1395  190]
 [ 225  303]]


 Accuracy of Naive byes model:  73.68670137245623
F1 score of Naive byes model:   61.495844875346265
[[1113  472]
 [  84  444]]
No. of features selected:    16

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents',
133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.217
694023039066), ('DeviceProtection', 20.226662195984314), ('TechSupport', 135.5597826
8636352), ('PaperlessBilling', 105.68086299962546), ('DSL', 71.31318025219977), ('Fibe
r optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 17
6.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), (
'TotalCharges', 142.3115981590349), ('MonthlyCharges', 27.36797848545879)]

Performance of the different models with 16 th best columns of the dataset:

Accuracy of the logistic model:   79.9810695693327
F1_score of the model:  56.96846388606307
[[1410  175]
 [ 248  280]]


Accuracy of Decision tree model:  74.86985328916232
F1_score of Decision tree model:  51.14995400183992
[[1304  281]
 [ 250  278]]

Accuracy of KNN model:  80.59630856601989
```

```
F1_score of KNN model:  59.4059405940594
[[1403  182]
 [ 228  300]]


 Accuracy of Naive byes model:  73.82867960246095
F1 score of Naive byes model:   61.570535093815145
[[1117  468]
 [  85  443]]
No. of features selected:     17
```

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.217694023039066), ('DeviceProtection', 20.226662195984314), ('TechSupport', 135.55978268636352), ('StreamingTV', 17.334234804462824), ('PaperlessBilling', 105.68086299962546), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349), ('MonthlyCharges', 27.36797848545879)]

Performance of the different models with 17 th best columns of the dataset:

```
Accuracy of the logistic model:   80.8329389493611
F1_score of the model:  58.79959308240081
[[1419  166]
 [ 239  289]]


Accuracy of Decision tree model:  74.49124467581638
F1_score of Decision tree model:  49.9535747446611
[[1305  280]
 [ 259  269]]


Accuracy of KNN model:  80.50165641268339
F1_score of KNN model:  58.88223552894212
[[1406  179]
 [ 233  295]]


 Accuracy of Naive byes model:  73.49739706578325
F1 score of Naive byes model:   61.111111111111
```

[[1113 472]
 [88 440]]
No. of features selected: 18

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.217694023039066), ('DeviceProtection', 20.226662195984314), ('TechSupport', 135.55978268636352), ('StreamingTV', 17.334234804462824), ('StreamingMovies', 16.242530716789197), ('PaperlessBilling', 105.68086299962546), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349), ('MonthlyCharges', 27.36797848545879)]

Performance of the different models with 18 th best columns of the dataset:

Accuracy of the logistic model: 81.21154756270705
F1_score of the model: 59.61342828077314
[[1423 162]
 [235 293]]

Accuracy of Decision tree model: 73.68670137245623
F1_score of Decision tree model: 49.270072992700726
[[1287 298]
 [258 270]]

Accuracy of KNN model: 81.11689540937056
F1_score of KNN model: 60.6896551724138
[[1406 179]
 [220 308]]

Accuracy of Naive byes model: 73.73402744912447
F1 score of Naive byes model: 61.43154968728284
[[1116 469]
 [86 442]]
No. of features selected: 19

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('MultipleLines', 6.548511590728465), ('OnlineSecurity', 147.2958

5790556192), ('OnlineBackup', 31.217694023039066), ('DeviceProtection', 20.226662195984314), ('TechSupport', 135.55978268636352), ('StreamingTV', 17.334234804462824), ('StreamingMovies', 16.242530716789197), ('PaperlessBilling', 105.68086299962546), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.3115981590349), ('MonthlyCharges', 27.36797848545879)]

Performance of the different models with 19 th best columns of the dataset:

Accuracy of the logistic model: 81.16422148603881
F1_score of the model: 59.797979797979785
[[1419 166]
 [232 296]]

Accuracy of Decision tree model: 73.63937529578799
F1_score of Decision tree model: 50.22341376228775
[[1275 310]
 [247 281]]

Accuracy of KNN model: 80.8329389493611
F1_score of KNN model: 59.621136590229305
[[1409 176]
 [229 299]]

Accuracy of Naive byes model: 73.78135352579271
F1 score of Naive byes model: 61.42061281337048
[[1118 467]
 [87 441]]
No. of features selected: 20

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('MultipleLines', 6.548511590728465), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.217694023039066), ('DeviceProtection', 20.226662195984314), ('TechSupport', 135.55978268636352), ('StreamingTV', 17.334234804462824), ('StreamingMovies', 16.242530716789197), ('PaperlessBilling', 105.68086299962546), ('Male', 0.2586986175941518), ('DSL', 71.31318025219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886), ('One year', 176.12317121760617), ('Tw o year', 488.578090256898), ('tenure', 599.9539968592322), ('TotalCharges', 142.31159 81590349), ('MonthlyCharges', 27.36797848545879)]

Performance of the different models with 20 th best columns of the dataset:

Accuracy of the logistic model: 81.06956933270232
F1_score of the model: 59.677419354838705
[[1417 168]
[232 296]]

Accuracy of Decision tree model: 74.39659252247989
F1_score of Decision tree model: 50.046168051708214
[[1301 284]
[257 271]]

Accuracy of KNN model: 80.92759110269758
F1_score of KNN model: 60.059464816650156
[[1407 178]
[225 303]]

Accuracy of Naive byes model: 73.82867960246095
F1 score of Naive byes model: 61.46341463414634
[[1119 466]
[87 441]]
No. of features selected: 21

The name of the columns corresponding to the scores:

[('SeniorCitizen', 134.35154479888715), ('Partner', 82.41208263843043), ('Dependents', 133.03644287868082), ('PhoneService', 0.09726062494293952), ('MultipleLines', 6.54851 1590728465), ('OnlineSecurity', 147.29585790556192), ('OnlineBackup', 31.21769402303 9066), ('DeviceProtection', 20.226662195984314), ('TechSupport', 135.55978268636352) , ('StreamingTV', 17.334234804462824), ('StreamingMovies', 16.242530716789197), ('Pa perlessBilling', 105.68086299962546), ('Male', 0.2586986175941518), ('DSL', 71.313180 25219977), ('Fiber optic', 374.4762164498734), ('Month-to-month', 519.8953106092886) , ('One year', 176.12317121760617), ('Two year', 488.578090256898), ('tenure', 599.95 39968592322), ('TotalCharges', 142.3115981590349), ('MonthlyCharges', 27.3679784854 5879)]

Performance of the different models with 21 th best columns of the dataset:

```
Accuracy of the logistic model:   81.21154756270705
F1_score of the model:  59.77710233029382
[[1421  164]
 [ 233  295]]


Accuracy of Decision tree model:  74.15996213913867
F1_score of Decision tree model:  49.81617647058824
[[1296  289]
 [ 257  271]]


Accuracy of KNN model:  80.88026502602933
F1_score of KNN model:  60.15779092702169
[[1404  181]
 [ 223  305]]


 Accuracy of Naive byes model:  74.06530998580217
F1 score of Naive byes model:   61.678321678321666
[[1124  461]
 [  87  441]]
```

Out of all above experimental data we found that all the combination of column in data set gives high percentage of F1_score. So, we decided to keep all the column of the data set to predict the dependent variable.

BUILDING A MODEL

&

EVALUATION OF MODEL

- Until now we have prepared the data set i.e., means converted all the entries of the all column in normal distribution form.
- Now we going to build various model and check the Accuracy of those model accordingly with given dataset.

Different types of model used for analyzing:

1. Logistic Regression
2. Decision Tree
3. K- Nearest Neighbors
4. Naïve Bayes
5. Bagging classifier using decision tree
6. Random forest
7. Voting Classifier using logistic reg. and decision tree
8. Bagging classifier using Naïve Bayes
9. Bagging classifier using K-NN
10. Bagging classifier using logistic
11. Voting With Logistic & Naïve Bayes
12. Voting classifier using Naïve Bayes, KNN and logistic Reg.
13. Voting classifier using Naïve Bayes, KNN, logistic Reg. & Decision Tree

1. Logistic Regression:

Code:

```
logmodel= LogisticRegression()
logmodel.fit(x_train,y_train)
pre=logmodel.predict(x_test)

print()

print("Accuracy of the logistic model:  " , accuracy_score(y_test, pre)*100)
print("F1_score of the model: ", f1_score(y_test,pre)*100)
print()
res=confusion_matrix(y_test,pre)
print("Confusion matrix :")
print(res)
print()
results=model_selection.cross_val_score(logmodel,x,y,cv=kfold)*100
print("cross validation of the model:  ",results)
print()
print("Mean of the result of the cross validation: ",results.mean())
```

Output:

```
Accuracy of the logistic model:   81.21154756270705
F1_score of the model:  59.69543147208122

Confusion matrix :
[[1422  163]
 [ 234  294]]

cross validation of the model:   [80.70921986 80.14184397 80.         81.96022727 80.68181818 79.11931818
 81.67613636 79.26136364 79.26136364 81.53409091]

Mean of the result of the cross validation:  80.43453820116054
```

2. Decision Tree:

Code:

```
classifier_entropy = DecisionTreeClassifier(criterion='entropy', random_state=42, max_depth = 20 )
classifier_entropy.fit(x_train,y_train)
y_ped=classifier_entropy.predict(x_test)

print("Accuracy of Decision tree model : ",accuracy_score(y_test,y_ped)*100)
print()
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of Decision tree model : ", f1_score(y_test,y_ped)*100)
print()
classifier=DecisionTreeClassifier(criterion='entropy', random_state=42, max_depth = 3 )
results=model_selection.cross_val_score(classifier,x,y,cv=kfold)*100
print("cross validation of Decision tree model :  ",results)
print()
print("Mean of the result of the cross validation: ",results.mean() )
```

Output:

```
Accuracy of Decision tree model :  74.15996213913867

Confusion Matrix:
[[1296  289]
 [ 257  271]]

F1_score of Decision tree model :  49.81617647058824

cross validation of Decision tree model :   [80.28368794 78.86524823 77.73049645 80.68181818 79.11931818 76.42045455
 80.39772727 76.98863636 79.40340909 79.54545455]

Mean of the result of the cross validation:  78.94362508059316
```

3. K- Nearest Neighbors:

Code:

```
classifier=KNeighborsClassifier(n_neighbors=83,metric='euclidean')
classifier.fit(x_train,y_train)
y_pred =classifier.predict(x_test)

print("Accuracy of KNN model : ",accuracy_score(y_test,y_pred)*100)
print()
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
print()
print("F1_score of KNN model : ", f1_score(y_test,y_pred)*100)
print()
classifier=KNeighborsClassifier(n_neighbors=83,metric='euclidean')
results=model_selection.cross_val_score(classifier,x,y,cv=kfold)*100
print("cross validation of KNN model:  ",results)
print()
print("Mean of the result of the cross validation: ",results.mean())
```

Output:

```
Accuracy of KNN model :  80.88026502602933

Confusion Matrix:
[[1404  181]
 [ 223  305]]

F1_score of KNN model :  60.15779092702169

cross validation of KNN model:   [81.41843972 80.85106383 79.43262411 82.52840909 79.6875     77.69886364
 80.53977273 78.97727273 79.11931818 81.25      ]

Mean of the result of the cross validation:  80.15032640232108
```

4. Naïve Bayes:

Code:

```
label=list(y_train)
l=x_train
model=GaussianNB()
model.fit(l,label)
predicted=model.predict(x_test)
print(" Accuracy of Naive bayes model : ",accuracy_score(y_test,predicted)*100)
print()
print("Confusion matrix:")
print(confusion_matrix(y_test,predicted))
print()
print("F1 score of Naive bayes model  :  ",f1_score(y_test,predicted)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("cross validation of navie bayes model:  ",results)
print()
print("Mean of the result of the cross validation: ",results.mean())
```

Output:

```
 Accuracy of Naive bayes model :  74.06530998580217

Confusion matrix:
[[1124  461]
 [  87  441]]

F1 score of Naive bayes model  :   61.678321678321666

cross validation of navie bayes model:   [73.19148936 73.33333333 75.17730496 74.43181818 72.15909091 70.45454545
 71.875      73.15340909 71.875      73.15340909]

Mean of the result of the cross validation:  72.88044003868472
```

5. Bagging Classifier Using Decision Tree:

Code:

```
cart= DecisionTreeClassifier()
num_tree =100
model = BaggingClassifier(base_estimator=cart, n_estimators= num_tree, random_state=7)

model.fit(x_train,y_train)
y_ped=model.predict(x_test)

print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix:")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print()
print("Ensemble learner model of type bagging: ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of bagging classifier using Decision tree model: ", results.mean())
print()
```

Output:

```
Accuracy of the model :  79.08187411263606

confusion Matrix:
[[1389  196]
 [ 246  282]]

F1_score of the model :  56.063618290258454


Ensemble learner model of type bagging:

list of accuracy:  [79.85815603 77.73049645 77.30496454 77.84090909 79.11931818 74.57386364
 78.125      78.26704545 80.39772727 77.98295455]

Accuracy: of bagging classifier using Decision tree model:  78.12004352030947
```

6. Random Forest:

Code:

```
num_tree =100
max_features =5
kfold= model_selection.KFold(n_splits=10, random_state=7)
model = RandomForestClassifier( n_estimators= num_tree,max_features=max_features,random_state=7)

model.fit(x_train,y_train)
y_ped=model.predict(x_test)

print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("Ensemble model of type Randomforest : ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of random forest using Decision tree model: ", results.mean())
```

Output:

```
Accuracy of the model :  80.17037387600567

confusion Matrix:
[[1417  168]
 [ 251  277]]

F1_score of the model :  56.93730729701954

Ensemble model of type Randomforest :

list of accuracy:  [80.70921986 80.56737589 77.73049645 78.125      79.82954545 75.28409091
 80.82386364 77.69886364 80.25568182 79.11931818]

Accuracy: of random forest using Decision tree model:  79.01434558349452
```

7. Voting Classifier Using Logistic Regression & Decision Tree:

Code:

```
estimators= []
model1= LogisticRegression()
estimators.append(('logistic',model1))
model2= DecisionTreeClassifier()

estimators.append(('cart',model2))
ensemble=VotingClassifier(estimators)
ensemble.fit(x_train,y_train)
y_ped=ensemble.predict(x_test)
print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("voting model:  ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of the voting classifier using logistic regression and Decision tree: ", results.mean())
```

Output:

```
Accuracy of the model :  80.17037387600567

confusion Matrix:
[[1484  101]
 [ 318  210]]

F1_score of the model :  50.05959475566149

voting model:

list of accuracy:  [80.70921986 80.56737589 77.73049645 78.125      79.82954545 75.28409091
 80.82386364 77.69886364 80.25568182 79.11931818]

Accuracy: of the voting classifier using logistic regression and Decision tree:  79.01434558349452
```

8. Bagging Classifier Using Naïve Bayes:

Code:

```
cart=GaussianNB()
num_tree =100
model = BaggingClassifier(base_estimator=cart, n_estimators= num_tree, random_state=7)

model.fit(x_train,y_train)
y_ped=model.predict(x_test)

print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print()
print("Ensemble learner model of type bagging: ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of bagging classifier using Decision tree model: ", results.mean())
print()
```

Output:

```
Accuracy of the model :  74.3019403691434

confusion Matrix:
[[1129  456]
 [  87  441]]

F1_score of the model :  61.89473684210526


Ensemble learner model of type bagging:

list of accuracy:  [73.19148936 73.4751773  75.31914894 74.43181818 72.30113636 70.59659091
 71.875      73.4375     72.15909091 73.29545455]

Accuracy: of bagging classifier using Decision tree model:  73.00824065119276
```

9.Bagging Classifier Using KNN:

Code:

```
cart=KNeighborsClassifier(n_neighbors=83,metric='euclidean')
num_tree =100
model = BaggingClassifier(base_estimator=cart, n_estimators= num_tree, random_state=7)

model.fit(x_train,y_train)
y_ped=model.predict(x_test)

print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print()
print("Ensemble learner model of type bagging: ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of bagging classifier using K-NN model: ", results.mean())
print()
```

Output:

```
Accuracy of the model :  80.8329389493611

confusion Matrix:
[[1407  178]
 [ 227  301]]

F1_score of the model :  59.781529294935446


Ensemble learner model of type bagging:

list of accuracy:  [80.9929078  80.56737589 79.43262411 82.10227273 79.82954545 77.27272727
 80.96590909 78.97727273 79.26136364 80.39772727]

Accuracy: of bagging classifier using K-NN model:  79.97997259832366
```

9. Bagging Classifier Using Logistic:

Code:

```
cart=LogisticRegression()
num_tree =100
model = BaggingClassifier(base_estimator=cart, n_estimators= num_tree, random_state=7)

model.fit(x_train,y_train)
y_ped=model.predict(x_test)

print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print()
print("Ensemble learner model of type bagging: ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of bagging classifier using logistic regression model: ", results.mean())
print()
```

Output:

```
Accuracy of the model :  81.21154756270705

confusion Matrix:
[[1423  162]
 [ 235  293]]

F1_score of the model :  59.61342828077314
```

```
Ensemble learner model of type bagging:

list of accuracy:  [80.9929078  80.         79.71631206 81.96022727 80.53977273 79.11931818
 81.53409091 79.26136364 79.26136364 81.25      ]

Accuracy: of bagging classifier using logistic regression model:  80.36353562217924
```

11. Voting With Logistic & Naïve Bayes:

Code:

```
estimators= []
model1= LogisticRegression()
estimators.append(('logistic',model1))
model2= GaussianNB()
estimators.append(('cart',model2))
ensemble=VotingClassifier(estimators)
ensemble.fit(x_train,y_train)
y_ped=ensemble.predict(x_test)
print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("voting model:  ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of the voting classifier using logistic regression and naive bayes: ", results.mean())
```

Output:

```
Accuracy of the model :  81.21154756270705

confusion Matrix:
[[1423  162]
 [ 235  293]]

F1_score of the model :  59.61342828077314

voting model:

list of accuracy:  [73.19148936 73.4751773  75.31914894 74.43181818 72.30113636 70.59659091
 71.875      73.4375     72.15909091 73.29545455]

Accuracy: of the voting classifier using logistic regression and naive bayes:  73.00824065119276
```

12. Voting classifier using Naïve Bayes, KNN and logistic Regression:

Code:

```
estimators= []
model1= LogisticRegression()
estimators.append(('logistic',model1))
model2= GaussianNB()
estimators.append(('cart',model2))
model3= KNeighborsClassifier(n_neighbors=83,metric='euclidean')
estimators.append(('KNN',model3))
ensemble=VotingClassifier(estimators)
ensemble.fit(x_train,y_train)
y_ped=ensemble.predict(x_test)
print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("voting model:  ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of the voting classifier using logistic regression,naive bayes and K-NN: ", results.mean())
```

Output:

```
Accuracy of the model :  80.78561287269285

confusion Matrix:
[[1388  197]
 [ 209  319]]

F1_score of the model :  61.11111111111111

voting model:

list of accuracy:  [80.9929078  80.56737589 79.43262411 82.10227273 79.82954545 77.27272727
 80.96590909 78.97727273 79.26136364 80.39772727]

Accuracy: of the voting classifier using logistic regression,naive bayes and K-NN:  79.97997259832366
```

13. Voting classifier using Naïve Bayes, K-NN and logistic Regression & Decision Tree:

Code:

```
estimators= []
model1= LogisticRegression()
estimators.append(('logistic',model1))
model2= GaussianNB()
estimators.append(('naive',model2))
model3= KNeighborsClassifier(n_neighbors=83,metric='euclidean')
estimators.append(('KNN',model3))
model4=DecisionTreeClassifier()
estimators.append(('cart',model4))
ensemble=VotingClassifier(estimators)
ensemble.fit(x_train,y_train)
y_ped=ensemble.predict(x_test)
print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("voting model:  ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of the voting classifier using logistic regression,naive bayes ,K-NN and decision tree : ", results.mean())
```

Output:

```
Accuracy of the model :  81.06956933270232

confusion Matrix:
[[1414  171]
 [ 229  299]]

F1_score of the model :  59.919839679358724

voting model:

list of accuracy:  [73.19148936 73.4751773  75.31914894 74.43181818 72.30113636 70.59659091
 71.875      73.4375     72.15909091 73.29545455]

Accuracy: of the voting classifier using logistic regression,naive bayes ,K-NN and decision tree :  73.00824065119276
```

COMPARISON OF MODELS BY PICTORIAL REPRESENTATION

COMPARISON OF THE MODELS BY BOX PLOT:

In descriptive statistics, a **box plot** or **boxplot** is a method for graphically depicting groups of numerical data through their quartiles.

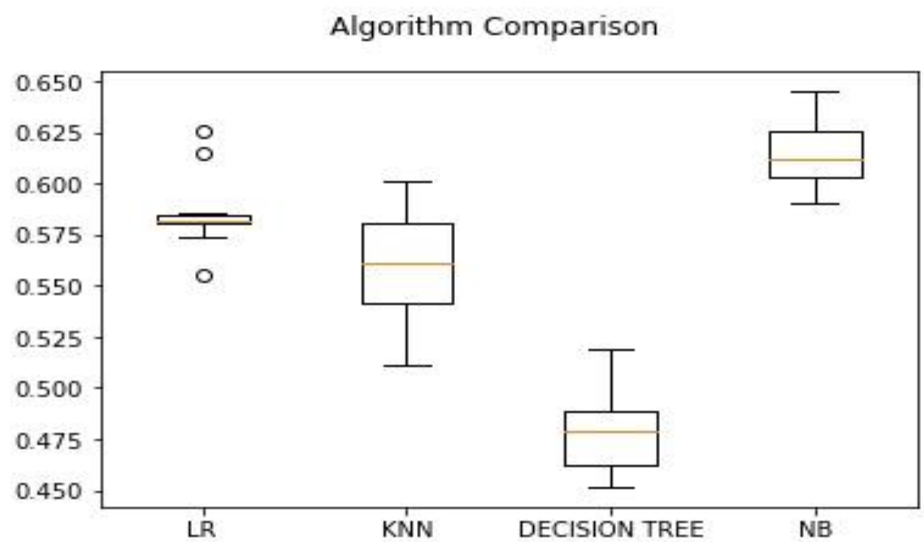
CODE:

```
x=cdf.drop('Churn',axis=1)
y=cdf['Churn']
# prepare configuration fro cross validation test harness
seed=7
#prepare models
models=[]
models.append(('LR',LogisticRegression()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('DECISION TREE',DecisionTreeClassifier()))
models.append(('NB',GaussianNB()))

results=[]
names=[]
scoring='f1'
for name,model in models:
    kfold=model_selection.KFold(n_splits=10,random_state=seed)
    cv_results=model_selection.cross_val_score(model,x,y,cv=kfold,scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg="%s: %f (%f)" % (name,cv_results.mean(),cv_results.std())
fig=plt.figure()
fig.suptitle('Algorithm Comparison')
ax=fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
print("Comparision chart  based on F1 Score of the models : ")
plt.show()
```

❖ **OUTPUT:**

```
Comparision chart  based on F1 Score of the models :
```

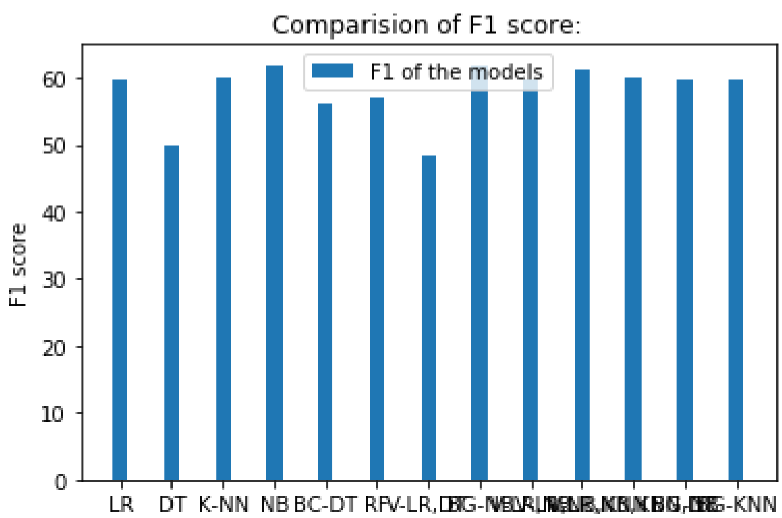


The graph above describes the comparison between the F1 Scores of the models namely Logistic Regression,K Nearest Neighbour,Decision Tree and Naive Bayes. The graph is a box plot which depicts that the F1 Score of Naïve Bayes model is the maximum which is about 61.5%(approx.), then comes the KNN model with F1 Score of about 60.1%(approx.).The third is Logistic Regression model with F1 Score of about 59.6% (approx.). And the last is the Decision Tree with the least F1 Score of about 49.7% (approx.).

COMPARISON OF THE MODELS BY BAR GRAPH:

```python
import matplotlib.pyplot as plt
import numpy as np
list4=[59.69543147208122, 49.81617647058824, 60.15779092702169, 61.678321678321666, 56.063618290258454, 56.9373072970195
      48.426150121065376, 61.89473684210526, 59.61342828077314, 61.11111111111111, 60.10050251256281, 59.61342828077314
      59.781529294935446]
list3=['LR','DT','K-NN','NB','BC-DT','RF','V-LR,DT','BG-NB','V-LR,NB','V-LR,NB,KNN','V-LR,NB,KNN,DT','BG-NB','BG-KNN']

xpos=np.arange(len(list3))
plt.xticks(xpos,list3)
plt.ylabel("F1 score")
plt.title("Comparision of F1 score:")
plt.bar(xpos,list4,width=0.3,label="F1 of the models")
plt.legend()
plt.show()
```



We have compared all the thirteen models by a bar graph and found that the maximum F1 Score of about 61.894% is obtained for the Bagging classifier using Naïve Bayes model. So we chose this model for the prediction of the Churn rate.

THE PROPOSED MODEL

The proposed model is composed of six steps. These steps are: identify problem domain, data selection, investigate data set, classification, clustering and knowledge usage. The classification step produces two types of customers (churners and non-churners) while the clustering step produces 3 clusters which are used to be evaluated according to the retention strategy in further usage.

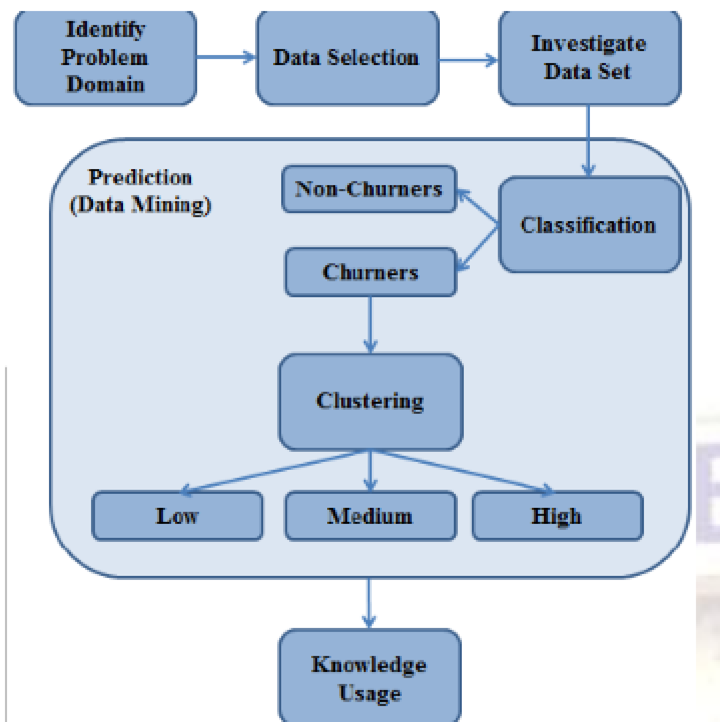


Figure 2: The Proposed Churn Prediction model

The proposed model can produce more than 3 clusters based on the types of acquired knowledge. Knowledge usage receives the produced clusters for assign a retaining solution for each type of churners. Churners can be clustered according to many criteria such as profitability or dissatisfactory of customers.

Code:

```
import numpy as np
import   pandas as pd
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier


cd=pd.read_csv("telecom.csv")
cdf=pd.DataFrame(cd)
list1 = [ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                'TechSupport','StreamingTV', 'StreamingMovies']
for i in list1:
    cdf[i] = cdf[i].replace({'No internet service':'No'})
```

```
cdf['MultipleLines'] = cdf['MultipleLines'].replace({'No phone service':'No'})


list2=[ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection','Churn',
                    'TechSupport','StreamingTV',
'StreamingMovies','MultipleLines','Partner','Dependents','PaperlessBilling','PhoneService']
for i in list2:
    cdf[i]= cdf[i].replace({'Yes':1,'No':0})
gen=pd.get_dummies(cdf["gender"],drop_first = True)
cdf=pd.concat((cdf,gen),axis=1)
cdf.drop(['gender'],axis=1,inplace=True)
y=pd.get_dummies(cdf["InternetService"])
y.drop(['No'],axis=1,inplace=True)
cdf.drop(['InternetService'],axis=1,inplace=True)
cdf=pd.concat((cdf,y),axis=1)
x=pd.get_dummies(cdf["Contract"])
cdf.drop(["Contract"],axis=1,inplace=True)
cdf=pd.concat((cdf,x),axis=1)
cdf.drop(["PaymentMethod"],axis=1,inplace=True)
cdf.drop(["customerID"],axis=1,inplace=True)
cdf['tenure']=cdf['tenure'].replace({0:0.1})
ten=np.log(cdf['tenure'])
cdf.drop(["tenure"],axis=1,inplace=True)
cdf=pd.concat((cdf,ten),axis=1)
cdf['TotalCharges'] =pd.to_numeric(cdf['TotalCharges'], errors ='coerce')
z=cdf.dropna()
index =cdf['TotalCharges'].index[cdf['TotalCharges'].apply(np.isnan)]
c=[]
c.extend(index)


d=cdf.loc[c,["MonthlyCharges"]]
```

```
x= z["MonthlyCharges"].values

m=len(x)
x=x.reshape((m,-1))

y= z["TotalCharges"].values
y = y.reshape((m,-1))

regmodel=linear_model.LinearRegression()
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3,random_state=42)
regmodel.fit(x_train,y_train)
a= regmodel.predict(d)

list1=[]
for i in a:
    list1.extend(i)

r2= regmodel.score(x,y)
print("Accuracy of linear model used for predicting a null values of TotalCharges in the data set : ", r2)

for i in list1:
    cdf['TotalCharges']=(cdf['TotalCharges'].fillna(i,limit= 1))

tc=np.log(cdf['TotalCharges'])
cdf.drop(['TotalCharges'],axis=1,inplace=True)
```

```
cdf=pd.concat((cdf,tc),axis=1)
mc=np.log(cdf['MonthlyCharges'])
cdf.drop(['MonthlyCharges'],axis=1,inplace=True)
cdf=pd.concat((cdf,mc),axis=1)

cdf=cdf.abs()

from sklearn.model_selection import train_test_split
x=cdf.drop('Churn',axis=1)
y=cdf['Churn']



x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3,random_state=1)



kfold = model_selection.KFold(n_splits=10, random_state =42)

# Naive bayes

label=list(y_train)
l=x_train
model=GaussianNB()
model.fit(l,label)
predicted=model.predict(x_test)
print(" Accuracy of Naive bayes model : ",accuracy_score(y_test,predicted)*100)
print()
print("Confusion matrix:")
print(confusion_matrix(y_test,predicted))
print()
print("F1 score of Naive bayes model   :   ",f1_score(y_test,predicted)*100)
print()
```

```
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print("cross validation of navie bayes model:   ",results)
print()
print("Mean of the result of the cross validation: ",results.mean())
#bagging classifier using naive bayes!
cart=GaussianNB()
num_tree =100
model = BaggingClassifier(base_estimator=cart, n_estimators= num_tree,
random_state=7)


model.fit(x_train,y_train)
y_ped=model.predict(x_test)


print("Accuracy of the model : ",accuracy_score(y_test,y_ped)*100)
print()
print("confusion Matrix: ")
print(confusion_matrix(y_test,y_ped))
print()
print("F1_score of the model : ", f1_score(y_test,y_ped)*100)
print()
results=model_selection.cross_val_score(model,x,y,cv=kfold)*100
print()
print("Ensemble learner model of type bagging: ")
print()
print("list of accuracy: ",results)
print()
print("Accuracy: of bagging classifier using Decision tree model: ", results.mean())
print()
```

❖ Output:

```
Accuracy of Naive Bayes model:  74.06530998580217

Confusion matrix:
[[1124  461]
 [  87  441]]

F1 score of Naive Bayes model:   61.678321678321666

cross validation of Naive Bayes model: [73.19148936 73.33333333 75.1773049
6 74.43181818 72.15909091 70.45454545
 71.875      73.15340909 71.875      73.15340909]

Mean of the result of the cross validation:  72.88044003868472




Accuracy of the Bagging classifier model:  74.3019403691434

confusion Matrix:
[[1129  456]
 [  87  441]]

F1_score of the model:  61.89473684210526


Ensemble learner model of type bagging:

list of accuracy: [73.19148936 73.4751773  75.31914894 74.43181818 72.3011
3636 70.59659091
 71.875      73.4375     72.15909091 73.29545455]

Accuracy: of bagging classifier using Decision tree model:  73.00824065119
276
```

CONCLUSION

After testing accuracy of multiple models, namely - Logistic Regression, Naïve Bayes, K Nearest Neighbour, Decision Tree, Bagging Classifier, Voting Classifier and Random Forest and calculating their prediction accuracy and F1 Score, we have found that Naive Bayes provides the best F1 Score of 61.678% shows our algorithm is quite accurate. We have worked with 7,043 cases and 21 variables.

We have purposefully left the date of the expected churn open - ended because we are focused on only gauging the features that indicate the disengagement with the product, and not the exact manner (like time-frame) in which users will disengage.

The aim is to distinctly single out those customers who are likely to churn so that the company may engage with them again and rekindle their interest in the service.

In conclusion, we have obtained an accurate model **"Naïve Bayes"** and predicted possible customers' churning out at **74.065% accuracy** and **F1 Score** of **61.678** so that churn-rate can be effectively minimized by the company.

ACKNOWLEDGEMENT

I am very happy to complete this project, along with my talented group members. However, it would not have been possible without aid from multiple individuals. I am highly indebted to Globsyn Finishing School for training me and providing me with all the necessary knowledge.

I would like to express my gratitude to Mr. Kaushik Ghosh, whose advice and guidance has proven invaluable in bringing about the end product. It is through his constant encouragement that I have been able to see this project to its completion. There was also a lot of support throughout the course of the project from Mr. Kaushik who oversaw the entire process and extended his aid whenever required. I also have immense appreciation for my group members for their constant cooperation and help.

CERTIFICATION

Winter Training Organised by Globsyn Finishing School

This is to certify that **Slok Kumar Mahto,** a student of Government College of Engineering and Ceramic Technology under Roll No. GCECTB-R17-3028 has completed the real time project under the guidance of trainer **Kaushik Ghosh** on the topic of Telecom Industry Churning through Machine Learning with Python. Globsyn Finishing School ID-42653

Duration-17-1-2020 to 10-2-2020

CERTIFICATION

Winter Training Organised by Globsyn Finishing School

This is to certify that **Snigdha Sahu,** a student of Asansol Engineering College under MAKAUT, Roll-10800217016 has completed the real time project under the guidance of trainer **Kaushik Ghosh** on the topic of Telecom Industry Churning through Machine Learning with Python. Globsyn Finishing School ID-42298.

Duration-17-1-2020 to 10-2-2020

CERTIFICATION

Winter Training Organised by Globsyn Finishing School

This is to certify that **Shalini Kumari,** a student of Asansol Engineering College under MAKAUT, Roll-10800217023 has completed the real time project under the guidance of trainer **Kaushik Ghosh** on the topic of Telecom Industry Churning through Machine Learning with Python. Globsyn Finishing School ID-42299

Duration-17-1-2020 to 10-2-2020

CERTIFICATION

Winter Training Organised by Globsyn Finishing School

This is to certify that **Gourab Sarkar,** a student of Regent Education and Research Foundation under MAKAUT, Roll-26300117051 has completed the real time project under the guidance of trainer **Kaushik Ghosh** on the topic of Telecom Industry Churning through Machine Learning with Python. Globsyn Finishing School ID-42341

Duration-17-1-2020 to 10-2-2020

CERTIFICATION

Winter Training Organised by Globsyn Finishing School

This is to certify that **Harekrishna Mandal,** a student of Regent Education and Research Foundation under MAKAUT, Roll-26300117050 has completed the real time project under the guidance of trainer **Kaushik Ghosh** on the topic of Telecom Industry Churning through Machine Learning with Python. Globsyn Finishing School ID-42340

Duration-17-1-2020 to 10-2-2020

CERTIFICATION

Winter Training Organised by Globsyn Finishing School

This is to certify that **Avik Sarkar,** a student of Global Institute of Management and Technology under MAKAUT, Roll-25900117031 has completed the real time project under the guidance of trainer **Kaushik Ghosh** on the topic of Telecom Industry Churning through Machine Learning with Python. Globsyn Finishing School ID-42423

Duration-17-1-2020 to 10-2-2020