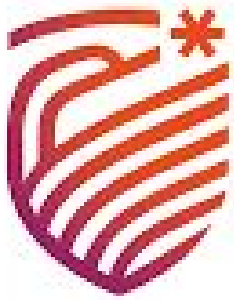


RAMAIAH
Institute of Technology

Data Mining Laboratory

Customer Churn Prediction

Department of Information Science and Engineering

K S Pavan Krishna (1MS17IS051)
Nayan Deep (1MS17IS070)
Harshita S (1MS17IS047)

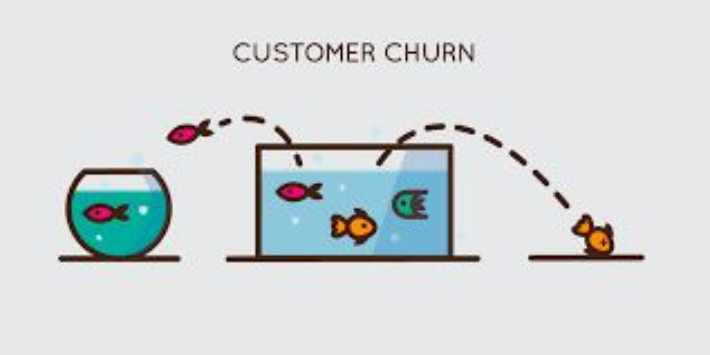
Abstract

In the telecom industry, customers are able to choose from multiple service providers and actively switch from one operator to another. In this highly competitive market, the telecommunications industry experiences an average of 15-25% annual churn rate. Given the fact that it costs 5-10 times more to acquire a new customer than to retain an existing one, customer retention has now become even more important than customer acquisition.

So we need to analyse telecom industry data and predict high value customers who are at high risk of churn and identify main indicators of churn Customers who have not done any usage, either incoming or outgoing - in terms of calls, internet etc. over a period of time. In our analysis we are concentrating on high value customers, as approximately 80% of revenue comes from the top 20% customers. Thus, if we can reduce churn of the high-value customers, we will be able to reduce significant revenue leakage.

Index/Content

Introduction



Customer attrition (*a.k.a customer churn*) is one of the biggest expenditures of any organization. If we could figure out why a customer leaves and when they leave with reasonable accuracy, it would immensely help the organization to strategize their retention initiatives manifold. We made use of a customer transaction dataset from Kaggle to understand the key steps involved in predicting customer attrition in Python.

Supervised Machine Learning is nothing but learning a function that maps an input to an output based on example input-output pairs. A supervised machine learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. Given that we have data on current and prior customer transactions in the telecom dataset, this is a standardized supervised classification problem that tries to predict a binary outcome (Y/N).In the we have made an attempt to solve some of the key business challenges pertaining to customer attrition like say, (1) what is the likelihood of an active customer leaving an organization? (2) what are key indicators of a customer churn? (3) what retention strategies can be implemented based on the results to diminish prospective customer churn?

Code

Import the dataset: We load the dataset into the python notebook in the current working directory.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv(r"C:\Users\91900\Desktop\telecom.csv")
data.head().T
```

Evaluate data structure: In this section, we need to look at the dataset in general and each column in detail to get a better understanding of the input data so as to aggregate the fields when needed.From the head & column methods, we get an idea that this is a telco customer churn dataset where each record entails the nature of subscription, tenure, frequency of payment and churn (signifying their current status).

	0	1	2	3	4
customerID	7590-VHVEG	5575-GNVDE	3668-QPYBK	7795-CFOCW	9237-HQITU
gender	Female	Male	Male	Male	Female
SeniorCitizen	0	0	0	0	0
Partner	Yes	No	No	No	No
Dependents	No	No	No	No	No
tenure	1	34	2	45	2
PhoneService	No	Yes	Yes	No	Yes
MultipleLines	No phone service	No	No	No phone service	No
InternetService	DSL	DSL	DSL	DSL	Fiber optic
OnlineSecurity	No	Yes	Yes	Yes	No
OnlineBackup	Yes	No	Yes	No	No
DeviceProtection	No	Yes	No	Yes	No
TechSupport	No	No	No	Yes	No
StreamingTV	No	No	No	No	No
StreamingMovies	No	No	No	No	No
Contract	Month-to-month	One year	Month-to-month	One year	Month-to-month
PaperlessBilling	Yes	No	Yes	No	Yes
PaymentMethod	Electronic check	Mailed check	Mailed check	Bank transfer (automatic)	Electronic check
MonthlyCharges	29.850000	56.950000	53.850000	42.300000	70.7
TotalCharges	29.85	1889.5	108.15	1840.75	NaN
Churn	No	No	Yes	No	Yes

```
data.shape
```

```
(7043, 21)
```

```
data.dtypes
```

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

```
# Converting Total Charges to numerical data type
data.TotalCharges = pd.to_numeric(data.TotalCharges, errors='coerce')
data.isnull().sum()
```

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        13
Churn                0
dtype: int64
```

```
data['TotalCharges'].dtypes
```

```
dtype('float64')
```

```
data.groupby('Churn')[['MonthlyCharges', 'tenure', 'TotalCharges']].agg(['min', 'max', 'mean'])
```

	MonthlyCharges			tenure			TotalCharges		
	min	max	mean	min	max	mean	min	max	mean
Churn									
No	18.25	118.75	61.257275	0	72	37.572754	18.80	8672.45	2554.441923
Yes	18.85	118.35	74.459957	1	72	17.988223	18.85	8684.80	1533.334547

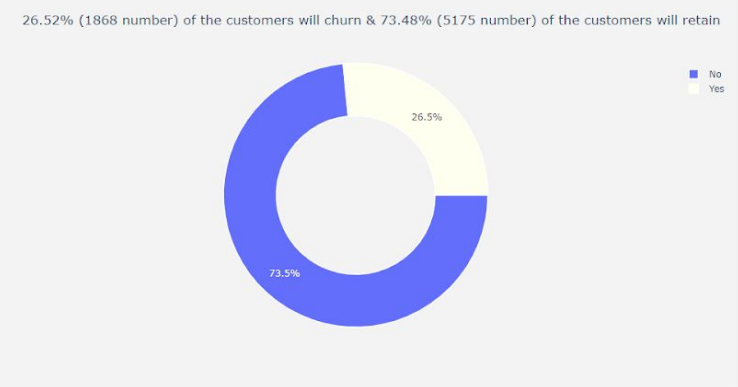
```
df = pd.DataFrame(data)
```

```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
...          ...     ...            ...     ...        ...     ...
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  ...  \
0              No  No phone service             DSL             No  ...
1             Yes                No             DSL            Yes  ...
2             Yes                No             DSL            Yes  ...
3              No  No phone service             DSL            Yes  ...
4             Yes                No     Fiber optic             No  ...
...           ...               ...             ...            ...  ...
7038          Yes               Yes             DSL            Yes  ...
7039          Yes               Yes     Fiber optic             No  ...
7040           No  No phone service             DSL            Yes  ...
7041          Yes               Yes     Fiber optic             No  ...
7042          Yes                No     Fiber optic            Yes  ...
```

```
7040           No  No phone service             DSL            Yes  ...
7041          Yes               Yes     Fiber optic             No  ...
7042          Yes                No     Fiber optic            Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
...               ...         ...         ...             ...             ...
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95       1889.50
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70           NaN
...               ...                        ...            ...           ...
7038              Yes               Mailed check          84.80       1990.50
7039              Yes    Credit card (automatic)         103.20       7362.90
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40        306.60
7042              Yes  Bank transfer (automatic)         105.65       6844.50

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
...    ...
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]
```

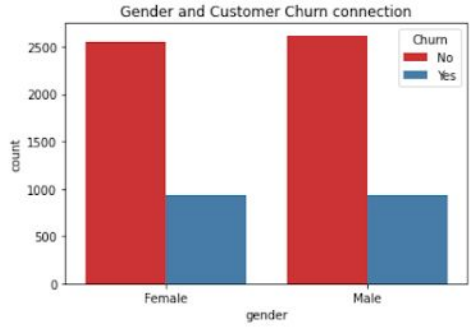
VISUALIZATION

```
import plotly.graph_objs as go
import plotly.offline as po

# Visualization of Total Customer Churn

churn_labels = data["Churn"].value_counts().keys().tolist()
churn_values = data["Churn"].value_counts().values.tolist()

plot_data= [go.Pie(labels = churn_labels,values = churn_values,marker = dict(colors = [ 'yello', 'Ivory'],
                        line = dict(color = "white",width = 1.5)),rotation = 90,hoverinfo = "label+value+text",hole = .6)]

# Total Number of Customers that will churn
count_churn_yes = data[data.Churn == 'Yes'].shape[0]
# Total Number of Customers that will not churn
count_churn_no = data[data.Churn == 'No'].shape[0]

# Percentage of customer that will churn
percent_churn_yes = round((count_churn_yes / (count_churn_yes + count_churn_no) * 100),2)
# Percentage of customer that will not churn (retain)
percent_churn_no = round((count_churn_no / (count_churn_yes + count_churn_no) * 100 ),2)


plot_layout = go.Layout(dict(title = f'{percent_churn_yes}% ({count_churn_yes} number) of the customers will churn &
                             {percent_churn_no}% ({count_churn_no} number) of the customers will retain',
                    plot_bgcolor = "rgb(243,243,243)",
                   paper_bgcolor = "rgb(243,243,243)",))


fig = go.Figure(data=plot_data, layout=plot_layout)
po.iplot(fig)
```



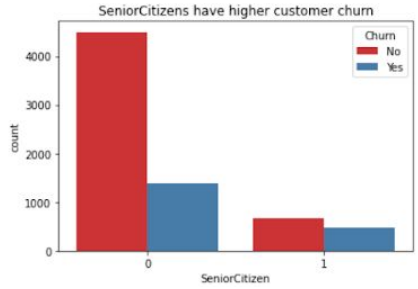

```
ax = sns.countplot(x='gender', hue='Churn', data=data, palette="Set1")
ax.set_title('Gender and Customer Churn connection')
```

```
Text(0.5, 1.0, 'Gender and Customer Churn connection')
```



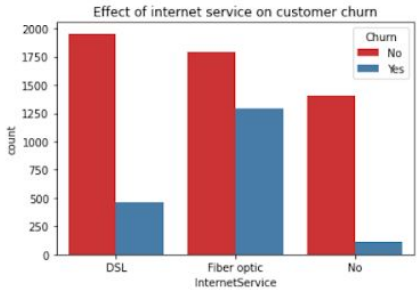
```
ax = sns.countplot(x= 'SeniorCitizen', hue='Churn', data=data, palette="Set1")
ax.set_title(f'SeniorCitizens have higher customer churn')
```

```
Text(0.5, 1.0, 'SeniorCitizens have higher customer churn')
```

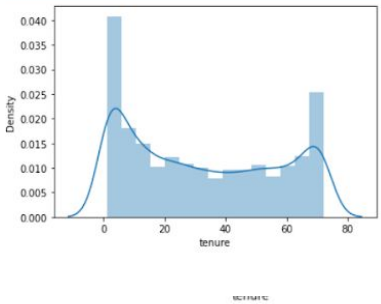



```
ax = sns.countplot(x= 'InternetService', hue='Churn', data=data, palette="Set1")
ax.set_title(f'Effect of internet service on customer churn')
```

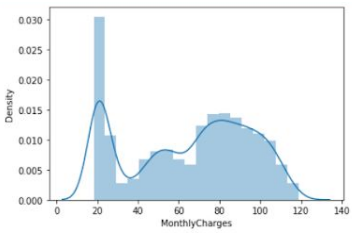
```
Text(0.5, 1.0, 'Effect of internet service on customer churn')
```




```
ax = sns.distplot(data['tenure'])
```

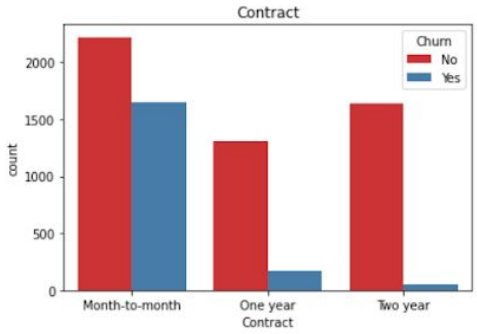



```
ax=sns.distplot(data['MonthlyCharges']);
```



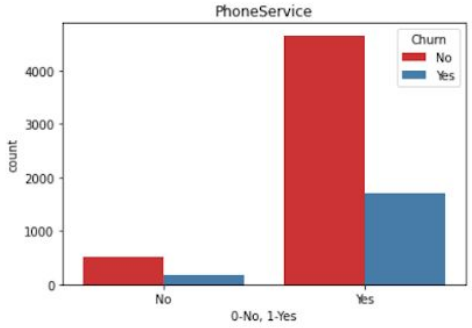
```
ax = sns.countplot(x= 'Contract', hue='Churn', data=data, palette="Set1")
ax.set_title(f'Contract')
```

```
Text(0.5, 1.0, 'Contract')
```



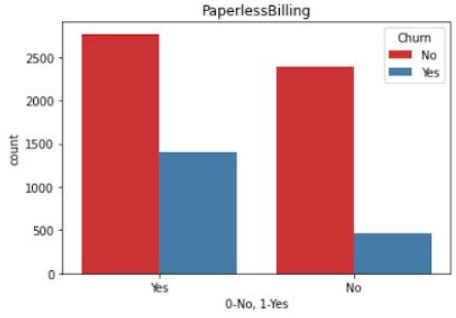

```
ax = sns.countplot(x= 'PhoneService', hue='Churn', data=data, palette="Set1")
ax.set_title(f'PhoneService')
plt.xlabel('0-No, 1-Yes')
```

```
Text(0.5, 0, '0-No, 1-Yes')
```




```
ax = sns.countplot(x= 'PaperlessBilling', hue='Churn', data=data, palette="Set1")
ax.set_title(f'PaperlessBilling')
plt.xlabel('0-No, 1-Yes')
```

```
Text(0.5, 0, '0-No, 1-Yes')
```



DATA PREPROCESSING

Revalidate NA's: It's always a good practice to revalidate and ensure that we don't have any more null values in the dataset.

```
# drop all rows with any NaN and NaT values
data = df.dropna()
```

```
# Calculate the missing value counts for each column
data.isnull().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
#dropping insignificant attributes
x = data.drop('customerID', axis = 'columns')

x=data['SeniorCitizen'] = data['SeniorCitizen'].map({0:'No', 1:'Yes'})
x=data['Churn'] = data['Churn'].map({'No':0, 'Yes':1})
```

```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
# Checking its datatype
data['TotalCharges'].dtypes
```

```
# Converting tenure into smaller buckets
bins = [0,6,12,18,24,36,48,60,72,84]
data['Tenure_grouped'] = pd.cut(data['tenure'], bins)
data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	TechSupport	StreamingTV
0	7590-VHVEG	Female	No	Yes	No	1	No	No phone service	DSL	No	...	No	No
1	5575-GNVDE	Male	No	No	No	34	Yes	No	DSL	Yes	...	No	No
2	3668-QPYBK	Male	No	No	No	2	Yes	No	DSL	Yes	...	No	No
3	7795-CFOCW	Male	No	No	No	45	No	No phone service	DSL	Yes	...	Yes	No
5	9305-CDSKC	Female	No	No	No	8	Yes	Yes	Fiber optic	No	...	No	Yes

5 rows × 22 columns

```
df_churn_model = df_dummy
df_churn_model.head()
```

	tenure	MonthlyCharges	TotalCharges	Churn	customerID_0003-MKNFE	customerID_0004-TLHLJ	customerID_0011-IGKFF	customerID_0013-EXCHZ	customerID_0013-MHZWF
0	1	29.85	29.85	0	0	0	0	0	0
1	34	56.95	1889.50	0	0	0	0	0	0
2	2	53.85	108.15	1	0	0	0	0	0
3	45	42.30	1840.75	0	0	0	0	0	0
5	8	99.65	820.50	1	0	0	0	0	0

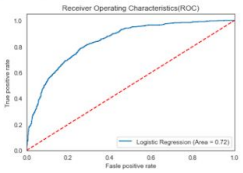
5 rows × 7059 columns

```
from sklearn.model_selection import train_test_split
y = df_churn_model['Churn']
X = df_churn_model.drop('Churn', axis=1, inplace=False)

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

y_pred_logreg = logreg.predict(X_test)

logreg_roc_auc = roc_auc_score(y_test,y_pred_logreg)
fpr, tpr, thresholds = roc_curve(y_test,logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (Area = %0.2f)' % logreg_roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('Fasle positive rate')
plt.ylabel('True positive rate')
plt.title('Receiver Operating Characteristics(ROC)')
plt.legend(loc="lower right")
plt.savefig('LOG_ROC')
plt.show()
```




```
from sklearn.metrics import confusion_matrix
confusion_matrix_logreg = confusion_matrix(y_test,y_pred_logreg)
confusion_matrix_logreg
```

```
array([[1394,  154],
       [ 262,  299]], dtype=int64)
```

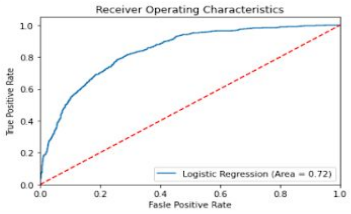
```
from sklearn.metrics import classification_report
classify_logreg = classification_report(y_test, y_pred_logreg)
print(classify_logreg)
```

```
              precision    recall  f1-score   support

           0       0.84      0.90      0.87      1548
           1       0.66      0.53      0.59       561

    accuracy                           0.80      2109
   macro avg       0.75      0.72      0.73      2109
weighted avg       0.79      0.80      0.80      2109
```

Results and Discussions

Evaluate the model using ROC Graph: It's good to re-evaluate the model using ROC Graph. ROC Graph shows us the capability of a model to distinguish between the classes based on the AUC Mean score. The orange line represents the ROC curve of a random classifier while a good classifier tries to remain as far away from that line as possible. As shown in the graph below, the fine-tuned Logistic Regression model showcased a higher AUC score.

```
]: from sklearn.metrics import roc_auc_score
   from sklearn.metrics import roc_curve

   y_pred_logreg = logreg.predict(X_test)

   logreg_roc_auc = roc_auc_score(y_test,y_pred_logreg)
   fpr, tpr, thresholds = roc_curve(y_test,logreg.predict_proba(X_test)[:,1])
   plt.figure()
   plt.plot(fpr, tpr, label='Logistic Regression (Area = %0.2f)' % logreg_roc_auc)
   plt.plot([0,1],[0,1],'r--')
   plt.xlim([0.0,1.0])
   plt.ylim([0.0,1.05])
   plt.xlabel('Fasle Positive Rate')
   plt.ylabel('True Positive Rate')
   plt.title('Receiver Operating Characteristics')
   plt.legend(loc="lower right")
   plt.savefig('LOG_ROC')
   plt.show()
```




```
]: from sklearn.metrics import confusion_matrix
   confusion_matrix_logreg = confusion_matrix(y_test,y_pred_logreg)
   confusion_matrix_logreg

]: array([[1394,  154],
          [ 262,  299]], dtype=int64)
```

```
]: from sklearn.metrics import classification_report
   classify_logreg = classification_report(y_test, y_pred_logreg)
   print(classify_logreg)
```

	precision	recall	f1-score	support
0	0.84	0.90	0.87	1548
1	0.66	0.53	0.59	561
accuracy			0.80	2109
macro avg	0.75	0.72	0.73	2109
weighted avg	0.79	0.80	0.80	2109

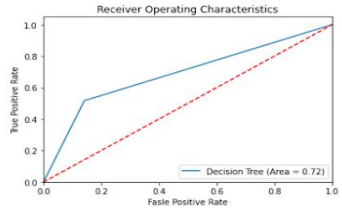
Decision Tree

A **decision tree** is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (**decision** taken after computing all attributes).

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dectree = DecisionTreeClassifier()
dectree.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred_dectree = dectree.predict(X_test)

dectree_roc_auc = roc_auc_score(y_test,y_pred_logreg)
fpr, tpr, thresholds = roc_curve(y_test,dectree.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Decision Tree (Area = %0.2f)' % dectree_roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('Fasle Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristics')
plt.legend(loc="lower right")
plt.savefig('DEC_ROC')
plt.show()
```




```
confusion_matrix_dectree = confusion_matrix(y_test,y_pred_dectree)
confusion_matrix_dectree
```

```
array([[1329,  219],
       [ 271,  290]], dtype=int64)
```

```
classify_dectree = classification_report(y_test, y_pred_dectree)
print(classify_dectree)
```

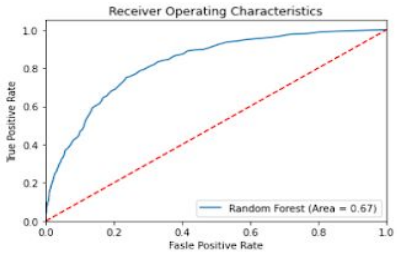
```
              precision    recall  f1-score   support

           0       0.83      0.86      0.84      1548
           1       0.57      0.52      0.54       561

    accuracy                           0.77      2109
   macro avg       0.70      0.69      0.69      2109
weighted avg       0.76      0.77      0.76      2109
```

Random forest

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
ranfor = RandomForestClassifier(n_estimators=100, random_state=0)
ranfor.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=0)
```

```
y_pred_ranfor = ranfor.predict(X_test)
ranfor_roc_auc = roc_auc_score(y_test,y_pred_ranfor)
fpr, tpr, thresholds = roc_curve(y_test,ranfor.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Random Forest (Area = %0.2f)' % ranfor_roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('Fasle Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristics')
plt.legend(loc="lower right")
plt.savefig('RF_ROC')
plt.show()
```




```
confusion_matrix_ranfor = confusion_matrix(y_test,y_pred_ranfor)
confusion_matrix_ranfor
```

```
array([[1422,  126],
       [ 323,  238]], dtype=int64)
```

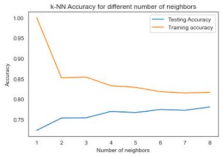
```
classify_ranfor = classification_report(y_test, y_pred_ranfor)
print(classify_ranfor)
```

```
              precision    recall  f1-score   support

           0       0.81      0.92      0.86      1548
           1       0.65      0.42      0.51       561

    accuracy                           0.79      2109
   macro avg       0.73      0.67      0.69      2109
weighted avg       0.77      0.79      0.77      2109
```

```
# K Nearest Neighbors Classifier
from sklearn.neighbors import KNeighborsClassifier

neighbors = np.arange(1,9)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))
```

```
# K Nearest Neighbors Classifier
from sklearn.neighbors import KNeighborsClassifier

neighbors = np.arange(1,9)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)
```

```
plt.title('k-NN Accuracy for different number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
```

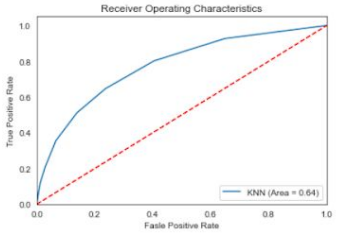


K Nearest Neighbours

```
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=8)
```

```
y_pred_knn = knn.predict(X_test)

knn_roc_auc = roc_auc_score(y_test,y_pred_knn)
fpr, tpr, thresholds = roc_curve(y_test,knn.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='KNN (Area = %0.2f)' % knn_roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('Fasle Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristics')
plt.legend(loc="lower right")
plt.savefig('KNN_ROC')
plt.show()
```




```
confusion_matrix_knn = confusion_matrix(y_test,y_pred_knn)
confusion_matrix_knn
```

```
array([[1447,  101],
       [ 362,  199]], dtype=int64)
```

```
classify_knn = classification_report(y_test, y_pred_knn)
print(classify_knn)
```

```
              precision    recall  f1-score   support

           0       0.80      0.93      0.86      1548
           1       0.66      0.35      0.46       561

    accuracy                           0.78      2109
   macro avg       0.73      0.64      0.66      2109
weighted avg       0.76      0.78      0.76      2109
```

Comparing performance of four models

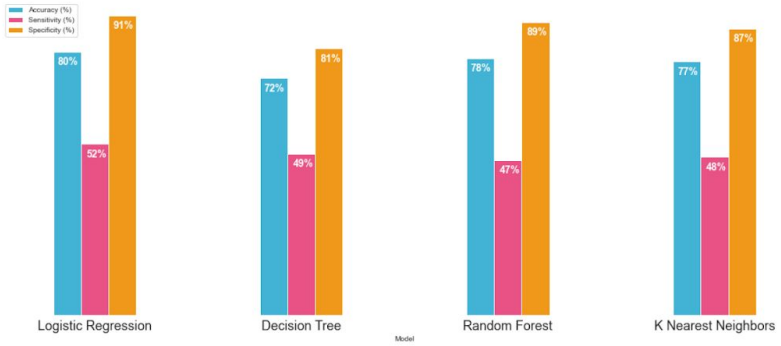
```
# Comparing performance of the 4 models

performance = {'Model': ['Logistic Regression', 'Decision Tree', 'Random Forest', 'K Nearest Neighbors'],
               'Accuracy (%)': [80, 72,78, 77],
               'Sensitivity (%)': [52, 49, 47, 48],
               'Specificity (%)': [91, 81, 89, 87]
              }
perf = pd.DataFrame(data=performance)
perf
```

	Model	Accuracy (%)	Sensitivity (%)	Specificity (%)
0	Logistic Regression	80	52	91
1	Decision Tree	72	49	81
2	Random Forest	78	47	89
3	K Nearest Neighbors	77	48	87

	Model	Accuracy (%)	Sensitivity (%)	Specificity (%)
0	Logistic Regression	80	52	91
1	Decision Tree	72	49	81
2	Random Forest	78	47	89
3	K Nearest Neighbors	77	48	87

```
perf = perf.set_index('Model')
```

```
sns.set_style(style='white')

ax = perf.plot(kind='bar', figsize=(20,8), width=0.4, color=['#42b3d5', '#e85285', '#f09819'], fontsize=18)

ax.legend(loc='best')
plt.xticks(rotation=0, ha="center")

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_linewidth(False)
ax.spines['left'].set_linewidth(False)

ax.get_yaxis().set_visible(False)

for n in range(3):
    for index, val in enumerate(perf.iloc[:,n].transpose()):
        label=str(val) + '%'

        if n==0:
            ax.annotate(label, xy=(index - 0.18, val-4), color='#ffffff', fontsize=14, fontweight='bold')
        elif n==1:
            ax.annotate(label, xy=(index - 0.04, val-4), color='#ffffff', fontsize=14, fontweight='bold')
        elif n==2:
            ax.annotate(label, xy=(index + 0.09, val-4), color='#ffffff', fontsize=14, fontweight='bold')
```



From the above bar graph it can be seen that logistic regression gives the best accuracy and decision tree results in a low accuracy.

Conclusion

We have used data mining techniques to predict the results of churn customers on the benchmark Churn dataset available at https://www.kaggle.com/blastchar/telco-customer-churn . We have evaluated the number of churns using the classification techniques using Python. We have represented the large dataset churn in the form of graphs which depicts the outcomes vividly and in a unique pattern visualization manner. The Churn Factor is used in many functions to depict the various areas or scenarios when the churn rate is high. We predict that there is a huge deviation in the graph of churners when customer service calls are measured. The graphs are made taking churn factors as the deciding parameters. Graphs represent the different ways of observing the number of churners from the dataset. Once the root area is recognized the steps can be taken by Telecom Company to improve their services and retain their old customers from churning

Future Work

Share key insights about the customer demographics and churn rate that you have garnered from the exploratory data analysis sections to the sales and marketing team of the organization. Let the sales team know the features that have positive and negative correlations with churn so that they could strategize the retention initiatives accordingly.

Further, classify the upcoming customers based on the propensity score as high risk (for customers with propensity score > 80%), medium risk (for customers with a propensity score between 60–80%) and lastly low-risk category (for customers with propensity score <60%). Focus on each segment of customers upfront and ensure that their needs are well taken care of.

Lastly, measure the return on investment (ROI) of this assignment by computing the attrition rate for the current financial quarter. Compare the quarter results with the same quarter last year or the year before and share the outcome with the senior management of your organization.

References

1. https://www.kaggle.com/
2. https://www.researchgate.net/
3. https://www.geeksforgeeks.org/
4. https://analyticsindiamag.com/
5. https://towardsdatascience.com/
6. https://scikit-learn.org/stable/
7. https://www.erpublication.org/published_paper/IJETR032129.pdf
8. Prof S. Chandrasekhar, - Predicting The Churn In Telecom Industry