Customer Churning in the Telecom Industry

Dasun Wellawalage
DSC 680
https://github.com/dasun27/Bellevue/blob/gh-pages/index.md

Business Problem

Customer churning is a major problem to many leading industries and Telecom is no exception. Due to the finite number of customers in a given area, it is important that Telecom companies retain their customers as it is really hard to win back those who churn away. Some studies have shown that it is way more profitable to retain customers than going after new customers. For this reason, many organizations now have a retention team specialized for retaining customers. Specifically, in the field of telecommunications, there are many attributes that would cause a customer to move out. The purpose of this analysis is to study some of those variables and figure out which ones have the most impact.

Method

For this analysis I am using a dataset taken from Kaggle. Unfortunately, it doesn't state which service provider it belongs to. However, this dataset consists of 7000+ records which is not huge, but it is still enough for us to perform some analysis. There are 21 variables in the dataset and most of them have binary or categorical values. There is a 'Churn' feature which is what we are trying to predict. I did some exploratory data analysis first followed by building models using three different methods to predict the churning probability. Below is a step by step guide to the methods I used to conduct this analysis.

1. Checking for null values and missing values – Most of the ML algorithms will give you errors if there are non-numeric values and even those that take categorical values will not tolerate missing or null values. In my dataset there were no missing or null values.

```
data = pd.read_csv('datasets_13996_18858_WA_Fn-UseC_-Telco-Customer-Churn.csv')

data.dtypes

data.isnull().sum()
```

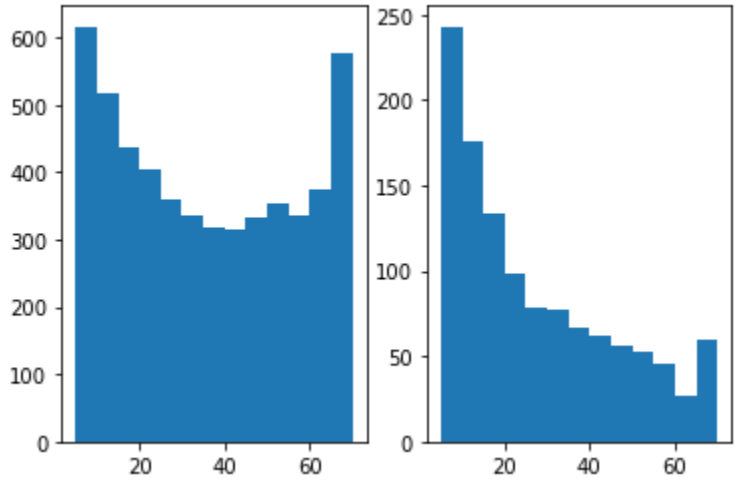
2. Data Cleanup – There were a few features that needed a little clean up. For example, some features had a value 'No internet service' that gave an identical meaning to ‘No’ in that given context. I replaced those instances with ‘No’. I also replaced the 1s and 0s in the ‘SeniorCitizen’ column with ‘Yes’ & ‘No’ to match the other binary features in the dataset.

```
# Converting 'SeniorCitizen' column to Yes/No to make it categorical
data["SeniorCitizen"] = data["SeniorCitizen"].replace({1:"Yes",0:"No"})
data["SeniorCitizen"] = data["SeniorCitizen"].astype(object)

# Changing 'No internet service' value to 'No' in service columns as they mean the same
services = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
'StreamingMovies']

for x in services:
   data[x]  = data[x].replace({'No internet service' : 'No'})
```

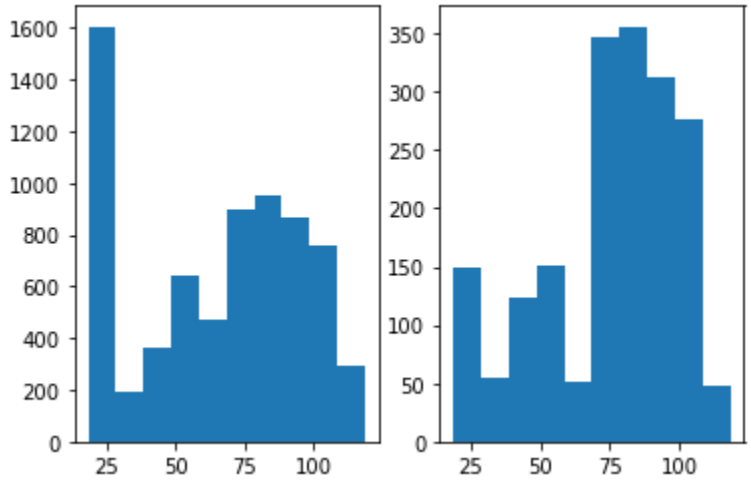
3. Studying individual variables – I studied some of the features using histograms to get an idea about the distribution of those individual variables. I split the dataset into two categories based on the ‘Churn’ variable. Then I compared individual features both in the large dataset and in the ‘Churn’ dataset. Below are some of the features that had prominent differences.

```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
binlist = [5, 10, 15, 20, 25 , 30, 35, 40, 45, 50, 55, 60, 65, 70]
churned = data[data["Churn"] == "Yes"]

axes[0].hist(data["tenure"], bins = binlist) # all records
axes[1].hist(churned["tenure"], bins = binlist) # churned customers
```

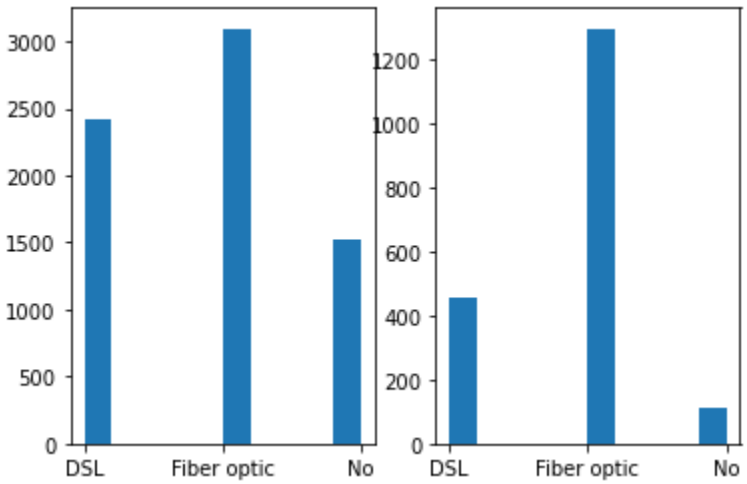
Tenure – main dataset (left) vs Churned dataset (right)



Monthly Charges – main dataset (left) vs Churned dataset (right)



Internet Service – main dataset (left) vs Churned dataset (right)



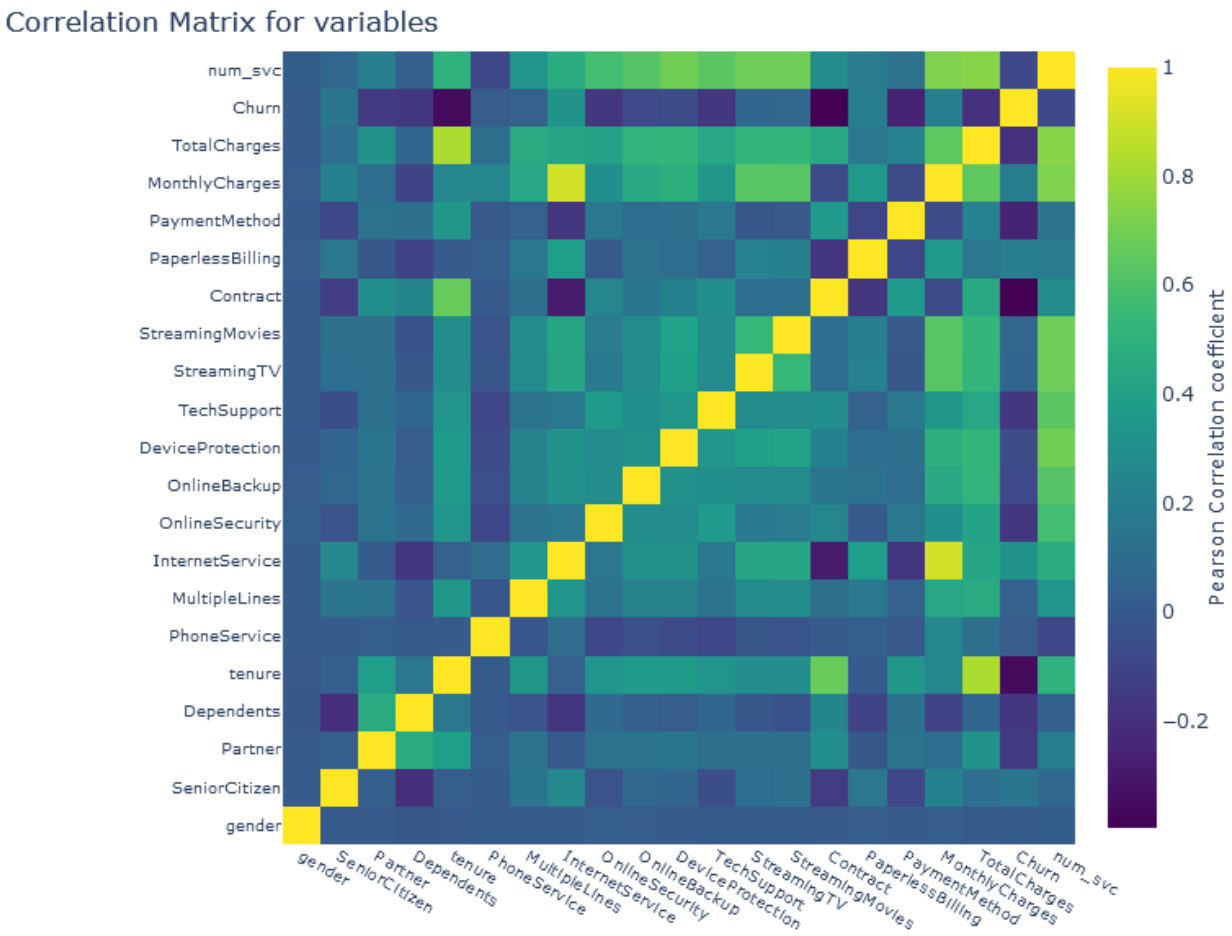
4. Next, I wanted to check if having additional services (apart from Internet & Phone) have an impact on churning. For this, I created a new feature which was a count of how many additional services customers have. Based on that feature, customers with more than four additional services had very little tendency to churn. However, this is somewhat obvious as you would hesitate to make changes that involve a lot of work for you.

```
svc = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
'StreamingMovies']

for x,row in data.iterrows():
    if row['InternetService'] == "No":
        data.at[x,'num_svc'] = 0
    else:
        n = 0
        for y in svc:
            if row[y] == "Yes":
                n += 1
        data.at[x,'num_svc'] = n
```

5. I used a correlation matrix to visualize the impact each variable has on the other. According to that, ‘tenure’, ‘contract’, and ‘payment method’ seemed to have a big impact on the ‘Churn’ while other features also had some impact. I used ‘plotly’ package to create the correlation matrix.

```
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go

correlation = datanum.corr()
#tick labels
matrix_cols = correlation.columns.tolist()
#convert to array
corr_array  = np.array(correlation)

#Plotting
trace = go.Heatmap(z = corr_array,
                   x = matrix_cols,
                   y = matrix_cols,
                   colorscale = "Viridis",
                   colorbar   = dict(title = "Pearson Correlation coefficient",
                                     titleside = "right"
                                    ) ,
                  )

layout = go.Layout(dict(title = "Correlation Matrix for variables",
                        autosize = False,
                        height  = 720,
                        width   = 800,
                        margin  = dict(r = 0 ,l = 210,
                                       t = 25,b = 210,
                                      ),
                        yaxis   = dict(tickfont = dict(size = 9)),
                        xaxis   = dict(tickfont = dict(size = 9))
```

```
                )
            )

data = [trace]
fig = go.Figure(data=data,layout=layout)
py.iplot(fig)
```



6. Finally, I started working on my ML models. I started with Logistic regression as the dependent variable is binary. I used the classes from SciKit Learn module.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

```
from sklearn import metrics

feature_cols = ['tenure', 'Contract', 'PaymentMethod', 'MonthlyCharges', 'InternetService',
'Dependents', 'SeniorCitizen']
X = datanum[feature_cols]
y = datanum['Churn']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)

logreg = LogisticRegression()
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7853492333901193
Precision: 0.6190476190476191
Recall: 0.4773218142548596
```

7. Next, I tried a KNN classifier. I tried a few different K values but k=5 seemed to be the optimal.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test,
y_pred))print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7671777399204998
```

8. Last, I tried a Random Forest classifier.

```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
```

```
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7597955706984668
```

Conclusion

Customer churning is a major problem to many leading industries and Telecom is no exception. Due to the finite number of customers in a given area, it is important that Telcos retain their customers as it is really hard to win back those who churn away. Purpose of this analysis is to build a model to predict customer churning based on this dataset. However, this should only be used to gain some insight and as a steppingstone to a much larger production scale prediction model, as such a system should include many more factors than what is used in this analysis.

References

- Datacamp (Aug 2018). KNN Classification using Scikit-learn. Retrieved Sep 20, 2020, from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

  Detailed explanation of using KKN classifier with SciKit Learn

- Datacamp (Dec 2019). Understanding Logistic Regression in Python. Retrieved Sep 20, 2020, from https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

  Detailed explanation of using Logistic Regression with SciKit Learn

- Datacamp (May 2018). Understanding Random Forest Classifiers in Python. Retrieved Sep 20, 2020, from https://www.datacamp.com/community/tutorials/random-forests-classifier-python

  Detailed explanation of using Random Forest Classifier with SciKit Learn

- Medium (Oct 2019). Analysis of Telco Customer Churn Dataset. Retrieved Sep 20, 2020, from https://medium.com/@kmacver/analysis-of-telco-customer-churn-dataset-926ff04d2295

  An analysis on customer churning in the Telecom industry

- Rstudio (2020). Telecom Churn Analysis. Retrieved Sep 20, 2020, from http://rstudio-pubs-static.s3.amazonaws.com/443094_bc2c15d74e7e4b7b96d8fc95f3162b08.html

A detailed analysis on customer churning using Rstudio

- Statista (Jun 2020). Customer churn rate in the United States in 2018, by industry. Retrieved Sep 20, 2020, from https://www.statista.com/statistics/816735/customer-churn-rate-by-industry-us/

  This shows the customer churn rate in US in 2018 for different industries. You can see how the Telecom industry does compared to others.

- NYU (2017). Churn in the Telecom Industry – Identifying customers likely to churn and how to retain them. Retrieved Sep 20, 2020, from https://wp.nyu.edu/adityakapoor/2017/02/17/churn-in-the-telecom-industry-identifying-customers-likely-to-churn-and-how-to-retain-them/

  This describes the current churning problem within the Telecom industry and how to address that using a data driven approach.

- McKinsey & Company (Dec 2017). Reducing churn in telecom through advanced analytics. Retrieved Sep 20, 2020, from https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/reducing-churn-in-telecom-through-advanced-analytics#

  A comprehensive, analytics-driven approach to base management which can help telecom companies reduce churn by as much as 15%.

- Medium (Nov 2019). Telco Customer Churn Prediction. Retrieved Sep 20, 2020, from https://towardsdatascience.com/telco-customer-churn-prediction-72f5cbfb8964

  This is a detailed article on building a machine learning model for Churn prediction.

- Medium (Nov 2018). Cutting the Cord: Predicting Customer Churn for a Telecom Company. Retrieved Sep 20, 2020, from https://towardsdatascience.com/cutting-the-cord-predicting-customer-churn-for-a-telecom-company-268e65f177a5

  An analysis on customer churning in the Telecom industry using a Kaggle dataset.

- Database Marketing Institute (Sep 2020). Churn reduction in the telecom industry. Retrieved Sep 20, 2020, from http://www.dbmarketing.com/telecom/churnreduction.html

  This is a marketing approach to customer churning in the Telecom industry.

- Rstudio (2020). Telecom Churn Analysis. Retrieved Sep 20, 2020, from http://rstudio-pubs-static.s3.amazonaws.com/277278_427ca6a7ce7c4eb688506efc7a6c2435.html

  A detailed analysis on customer churning using Rstudio

- Techsee (2019). Reasons for Customer Churn in the Telecom Industry: 2019 Survey Results. Retrieved Sep 20, 2020, from https://techsee.me/resources/surveys/2019-telecom-churn-survey/

  A survey conducted in 2019 on the reasons for customer churning in the Telecom industry.

- Rutgers (2020). Telecom customer churn prediction. Retrieved Sep 20, 2020, from https://rucore.libraries.rutgers.edu/rutgers-lib/62514/

  A research done by Rutgers university on Telecom customer churn prediction

- Parcus Group (2020). Telecom Customer Churn Prediction Models. Retrieved Sep 20, 2020, from https://parcusgroup.com/Telecom-Customer-Churn-Prediction-Models

  Building customer churn prediction model using consumer data.

Appendix I – Dataset

Dataset - https://www.kaggle.com/blastchar/telco-customer-churn

There is no codebook. Below are the variables.

- customerID – Alpha numeric
- gender – Text (Male/Female)
- SeniorCitizen – Binary
- Partner – Text (Yes/No)
- Dependents - Text (Yes/No)
- Tenure – integer
- PhoneService - Text (Yes/No)
- MultipleLines - Text
- InternetService - Text
- OnlineSecurity - Text
- OnlineBackup - Text
- DeviceProtection - Text
- TechSupport - Text
- StreamingTV - Text
- StreamingMovies - Text
- Contract - Text
- PaperlessBilling - Text (Yes/No)
- PaymentMethod - Text
- MonthlyCharges - Currency
- TotalCharges - Currency
- Churn - Text (Yes/No)

Appendix II – Complete Code

Importing Libraries

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

Loading Data

```
data = pd.read_csv('datasets_13996_18858_WA_Fn-UseC_-Telco-Customer-Churn.csv')
data.head()
```

Exploring Data

```
data.shape
```

```
(7043, 21)
```

```
data.isnull().sum()
```

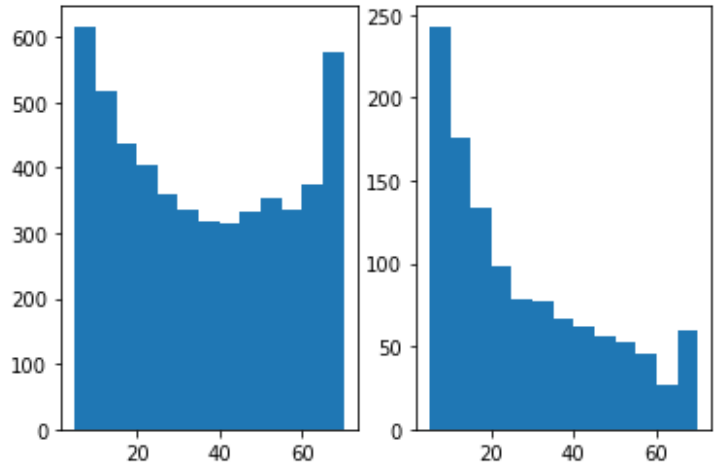
Cleanup

```
# Converting 'SeniorCitizen' column to Yes/No to make it categorical
data["SeniorCitizen"] = data["SeniorCitizen"].replace({1:"Yes",0:"No"})
data["SeniorCitizen"] = data["SeniorCitizen"].astype(object)
```

```
# Changing 'No internet service' value to 'No' in service columns as they mean the same
services = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

for x in services:
    data[x]  = data[x].replace({'No internet service' : 'No'})
```

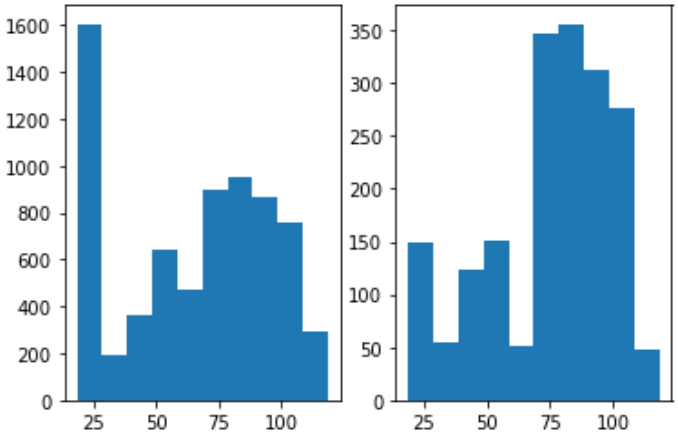
Visualizing current data

```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
binlist = [5, 10, 15, 20, 25 , 30, 35, 40, 45, 50, 55, 60, 65, 70]
churned = data[data["Churn"] == "Yes"]

axes[0].hist(data["tenure"], bins = binlist) # all records
axes[1].hist(churned["tenure"], bins = binlist) # churned customers
```



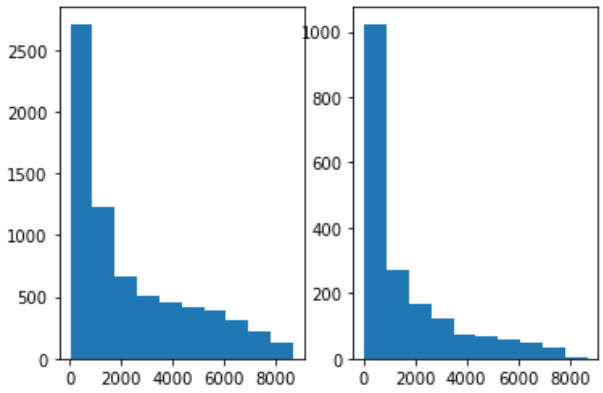
As you can see above, the churning possibility is significantly less as the tenure increases

```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
axes[0].hist(data["MonthlyCharges"]) # all records
axes[1].hist(churned["MonthlyCharges"]) # churned customers
```



This shows that most customers who churned were paying more than $75 per month

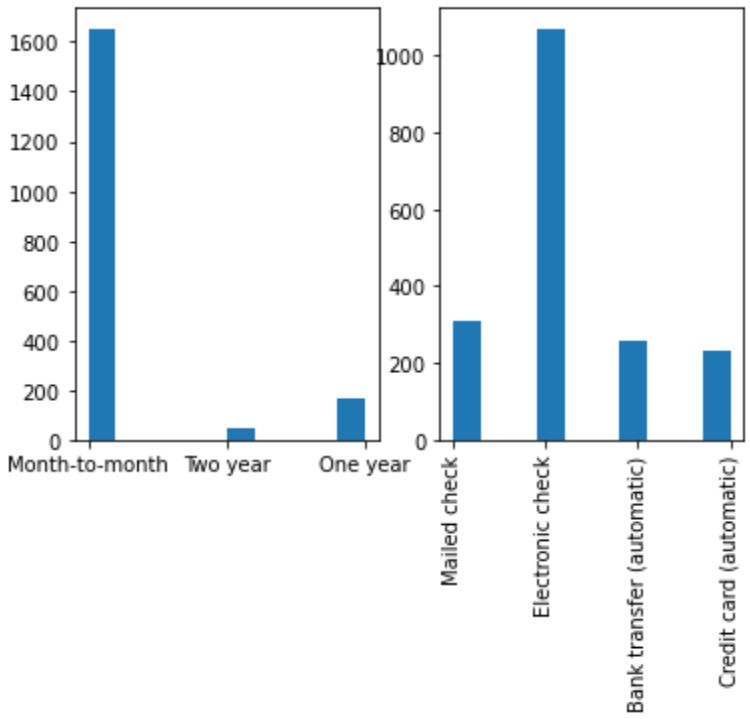
```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
axes[0].hist(data["TotalCharges"]) # all records
axes[1].hist(churned["TotalCharges"]) # churned customers
```



As both the graphs show an identical distribution, the Total charges don't seem to have an impact on churning

```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
axes[0].hist(churned["Contract"])
axes[1].hist(churned["PaymentMethod"])
plt.xticks(rotation=90)
```

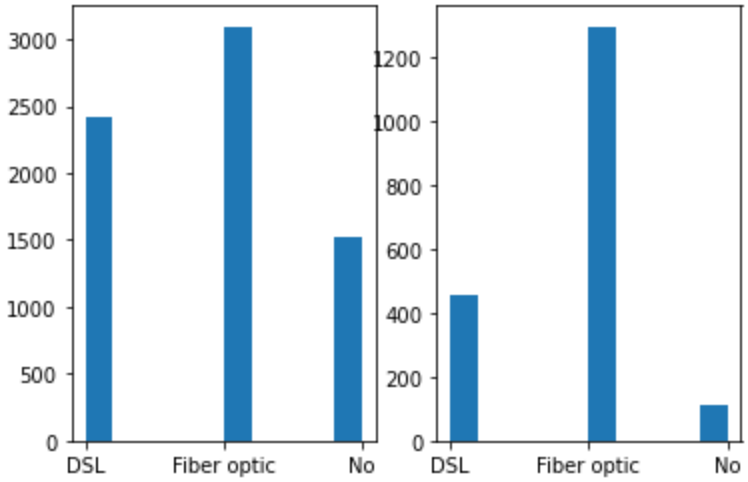
```
([0, 1, 2, 3], <a list of 4 Text major ticklabel objects>)
```



Most of the churned customers were on month-to-month contract and they used electronic check payment method.

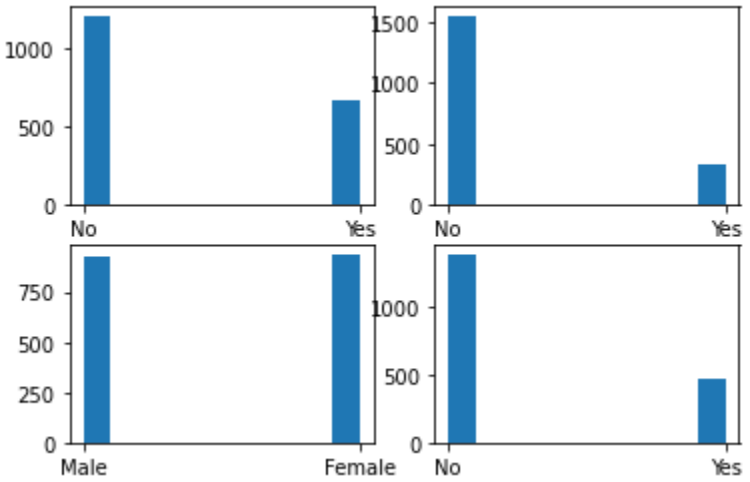
```
fig, axes = plt.subplots(nrows = 1, ncols = 2)
axes[0].hist(data["InternetService"]) # all records
axes[1].hist(churned["InternetService"]) # churned customers
```

```
(array([ 459.,    0.,    0.,    0.,    0., 1297.,    0.,    0.,    0.,
         113.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <a list of 10 Patch objects>)
```



Most of the churned customers were using fiber optic services

```
fig, axes = plt.subplots(nrows = 2, ncols = 2)
axes[0, 0].hist(churned["Partner"])
axes[0, 1].hist(churned["Dependents"])
axes[1, 0].hist(churned["gender"])
axes[1, 1].hist(churned["SeniorCitizen"])
```



Having dependents and being a senior citizen seem to have a big impact on churning while the gender and having a partner don't seem to contribute much

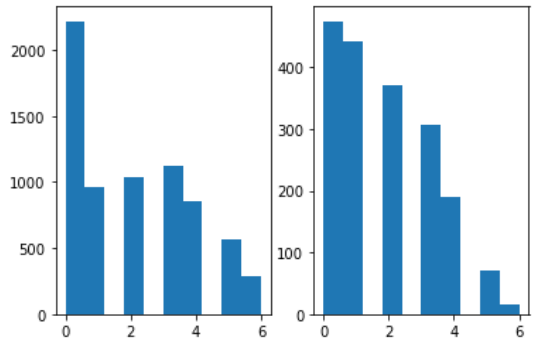
Creating a new column based on the number of additional online services

```
svc = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

for x,row in data.iterrows():
    if row['InternetService'] == "No":
        data.at[x,'num_svc'] = 0
    else:
        n = 0
        for y in svc:
            if row[y] == "Yes":
                n += 1
        data.at[x,'num_svc'] = n
```

```
data['num_svc'] = data['num_svc'].astype(int)
```

```
churned = data[data["Churn"] == "Yes"]
fig, axes = plt.subplots(nrows = 1, ncols = 2)
axes[0].hist(data["num_svc"]) # all records
axes[1].hist(churned["num_svc"]) # churned customers
```

```
(array([475., 442.,   0., 370.,   0., 306., 190.,   0.,  71.,  15.]),
 array([0. , 0.6, 1.2, 1.8, 2.4, 3. , 3.6, 4.2, 4.8, 5.4, 6. ]),
 <a list of 10 Patch objects>)
```



According to the graphs, those customer who have 4 or more additional services are very less likely to churn

Converting all columns to numeric values for correlation

```
# Creating a new dataframe
del datanum
datanum = data.copy(deep=True)
```

```
# converting 'gender'
datanum["gender"] = datanum["gender"].replace({"Male": 0, "Female": 1})
datanum["gender"] = pd.to_numeric(datanum["gender"], errors='coerce')
```

```
cat_var = ['SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
           'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'Churn']

for x in cat_var:
    datanum[x] = datanum[x].replace({"Yes": 1, "No": 0})
    datanum[x] = pd.to_numeric(datanum[x], errors='coerce')
```

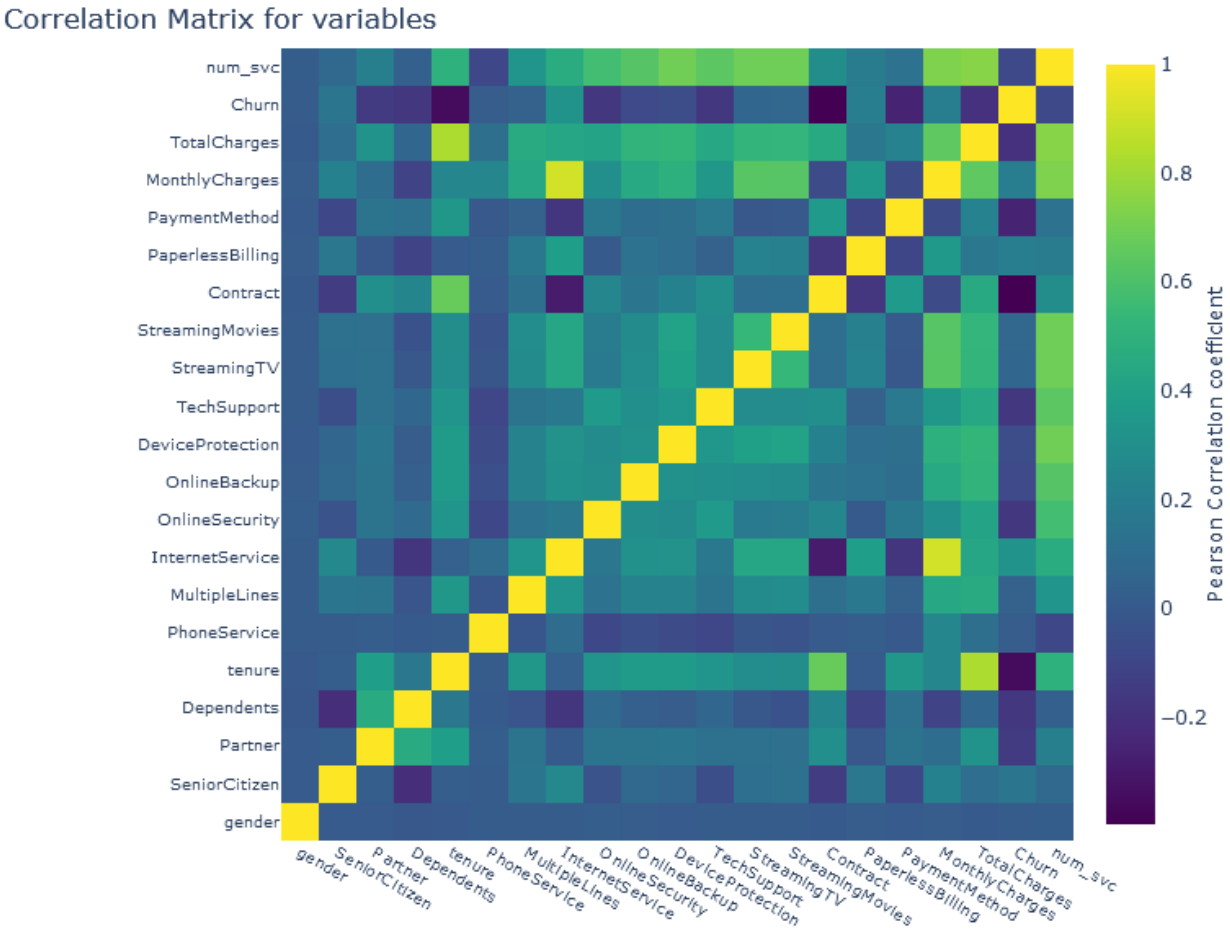
```
# converting 'MultipleLines'
datanum["MultipleLines"] = datanum["MultipleLines"].replace({"No": 0, "No phone service": 1, "Yes": 2})
datanum["MultipleLines"] = pd.to_numeric(datanum["MultipleLines"], errors='coerce')
```

```
# converting 'InternetService'
datanum["InternetService"] = datanum["InternetService"].replace({"No": 0, "DSL": 1, "Fiber optic": 2})
datanum["InternetService"] = pd.to_numeric(datanum["InternetService"], errors='coerce')
```

```
# converting 'Contract'
datanum["Contract"] = datanum["Contract"].replace({"Month-to-month": 0, "One year": 1, "Two year": 2})
datanum["Contract"] = pd.to_numeric(datanum["Contract"], errors='coerce')
```

```
# converting 'PaymentMethod'
datanum["PaymentMethod"] = datanum["PaymentMethod"].replace({"Electronic check": 0, "Mailed check": 1,
                                                             "Bank transfer (automatic)": 2, "Credit card (automatic)": 3})
datanum["PaymentMethod"] = pd.to_numeric(datanum["PaymentMethod"], errors='coerce')
```

```
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go

correlation = datanum.corr()
#tick labels
matrix_cols = correlation.columns.tolist()
#convert to array
corr_array  = np.array(correlation)

#Plotting
trace = go.Heatmap(z = corr_array,
                   x = matrix_cols,
                   y = matrix_cols,
                   colorscale = "Viridis",
                   colorbar   = dict(title = "Pearson Correlation coefficient",
                                     titleside = "right"
                                    ) ,
                  )

layout = go.Layout(dict(title = "Correlation Matrix for variables",
                        autosize = False,
                        height  = 720,
                        width   = 800,
                        margin  = dict(r = 0 ,l = 210,
                                       t = 25,b = 210,
                                      ),
                        yaxis   = dict(tickfont = dict(size = 9)),
                        xaxis   = dict(tickfont = dict(size = 9))
                       )
                  )

data = [trace]
fig = go.Figure(data=data,layout=layout)
py.iplot(fig)
```



Building Models

Logistic Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn import metrics
```

```
feature_cols = ['tenure', 'Contract', 'PaymentMethod', 'MonthlyCharges', 'InternetService', 'Dependents', 'SeniorCitizen']
X = datanum[feature_cols]
y = datanum['Churn']
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

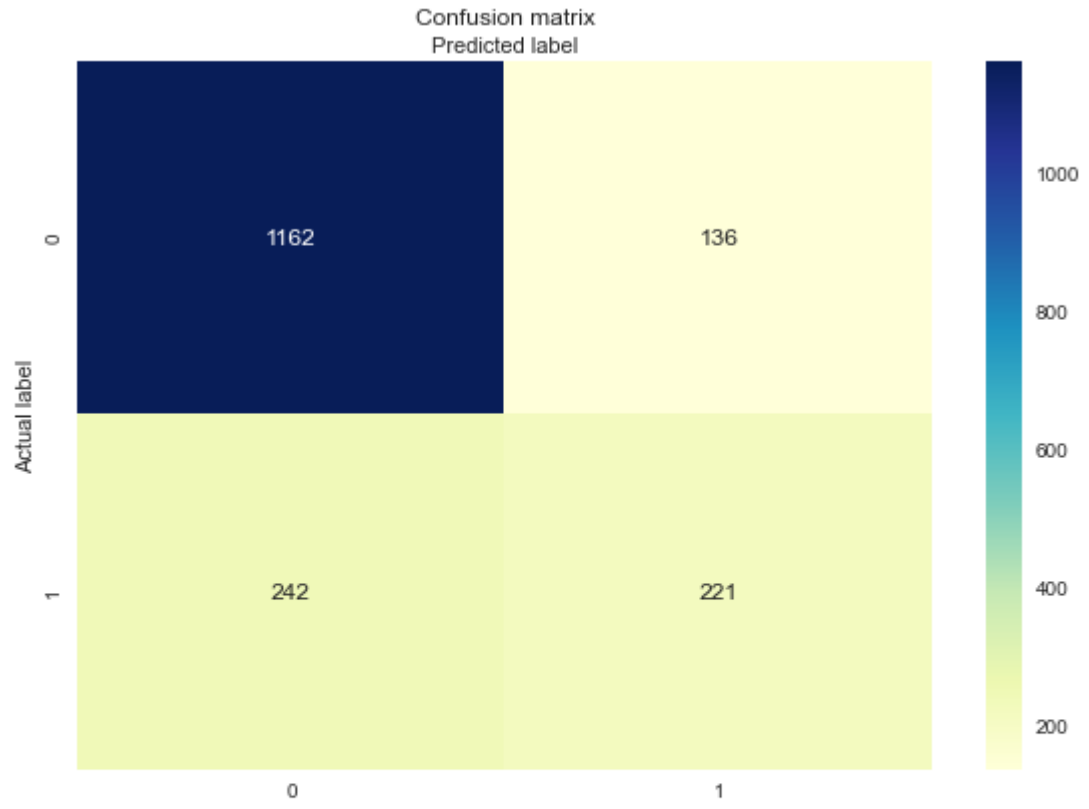
```
logreg = LogisticRegression()

logreg.fit(X_train,y_train)

y_pred=logreg.predict(X_test)
```

Creating the confusion matrix

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[1162,  136],
       [ 242,  221]], dtype=int64)
```

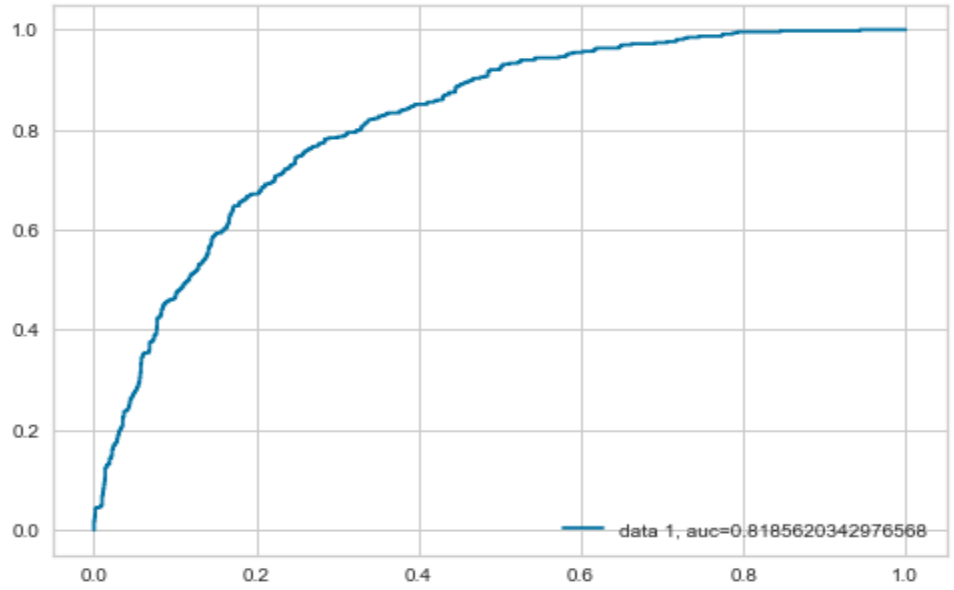
```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```




```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7853492333901193
Precision: 0.6190476190476191
Recall: 0.4773218142548596
```

```
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



KNN Classifier

K=3

```
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=3)

#Train the model using the training sets
knn.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(X_test)
```

```
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7541169789892107
```

K=5

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7671777399204998
```

K=7

```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7671777399204998
```

K=5 seems to lead best results

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7660420215786485
```

```
clf=RandomForestClassifier(n_estimators=150)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7632027257240205
```

```
clf=RandomForestClassifier(n_estimators=50)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7597955706984668
```

Default estimator (100) seems to yield best results