Ritu Kumari

10/12/2018

Contents

**1 Introduction**

1.1 Problem Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

**2 Methodology**

2.1 Pre Processing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.1.1 Outlier Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.1.2 Feature Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . 12

2.2.1 Model Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.2.2 Logidtics Regression . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.2.3 Deasion Trees . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.2.4 Random forest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

3 Conclusion

3.1 Model Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.1.1 Mean Absolute Error (MAE) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.1.2 Mean Squared Error (MSE) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.2 Model Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

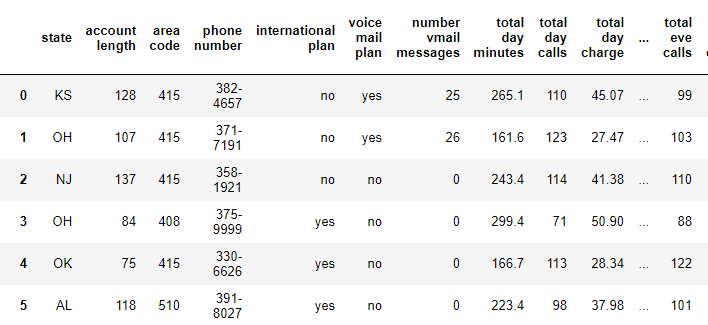
Appendix A - python Code

1.1 Problem statement

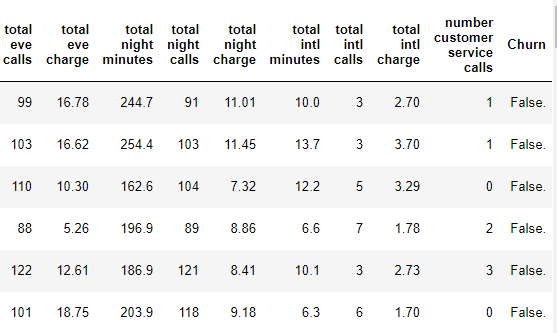
The objective of this Case is to predict customer behavior. We are providing you a public dataset that has customer usage pattern and if the customer has moved or not. We expect you to develop an algorithm to predict the churn score based on usage pattern.

1.2 Data

Our task is to develop an algorithm to predict the churn score based on usage pattern. Given below is a sample of the data set that we are using to predict the churn score.



|  |
| --- |
| 1 |



As you can see in the table below we have the following 21 variables, using which we have to correctly predict the churn score:

1. account length
2. international plan
3. voicemail plan
4. number of voicemail messages
5. total day minutes used
6. day calls made
7. total day charge
8. total evening minutes
9. total evening calls
10. total evening charge
11. total night minutes
12. total night calls
13. total night charge
14. total international minutes used
15. total international calls made
16. total international charge
17. number of customer service calls made
18. state

|  |
| --- |
| 2 |

1. area code
2. phone number
3. churn

**Chapter 2**

**Methodology**

**2.1 Pre Processing**

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

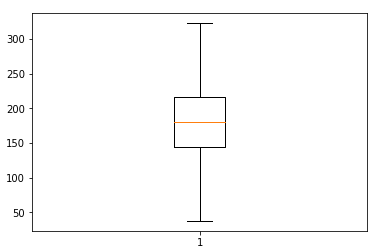
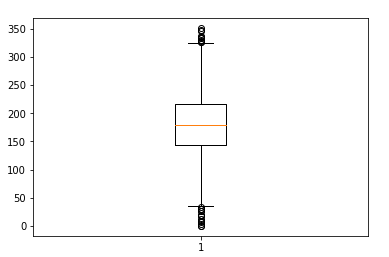
2.1.1 Outlier Analysis

We can clearly observe from these probability distributions that most of the variables are skewed, for example, total day charge , total night charge etc. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data. We can see the effect of the skew in figures.

|  |
| --- |
| 3 |

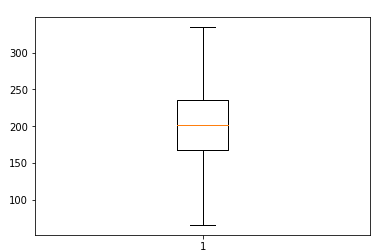
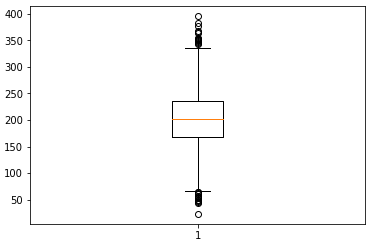
Effect of outliers using python

total day minutes



With outliers without outliers

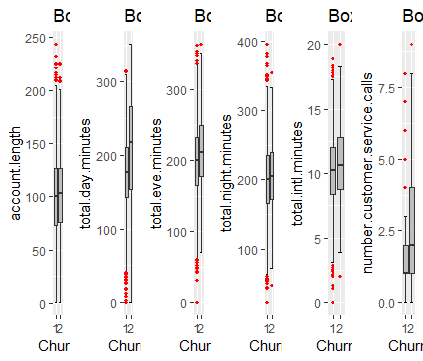
Total night minutes



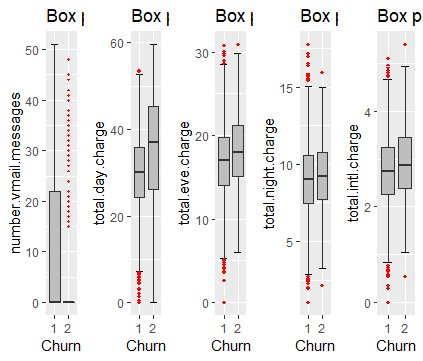
With outliers without outliers

|  |
| --- |
| 4 |

Effect of outliers using R

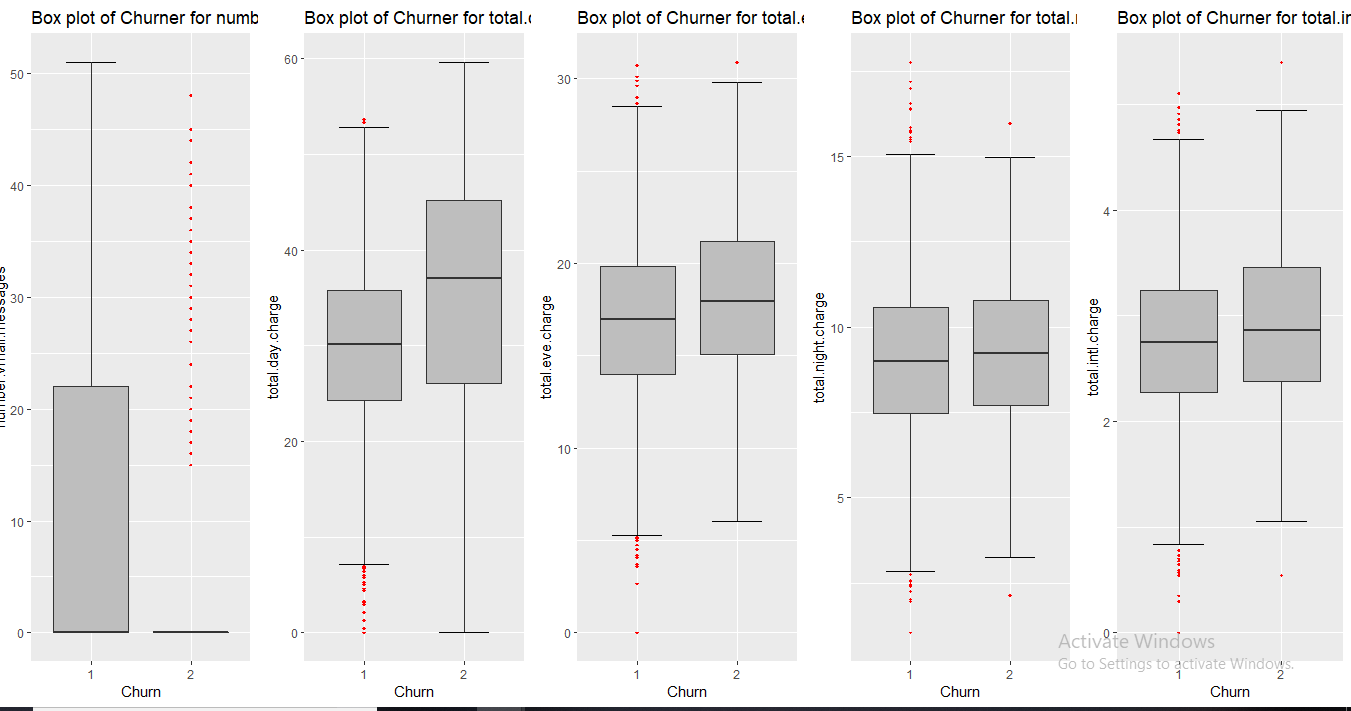


With outliers

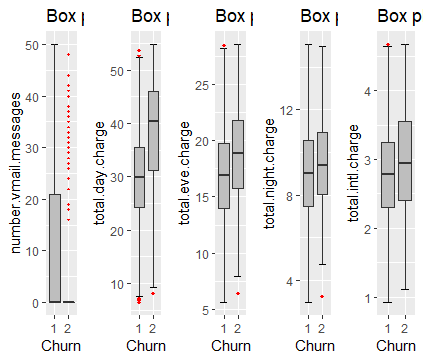


With otliers

|  |
| --- |
| 5 |



Without outliers



Without outliers

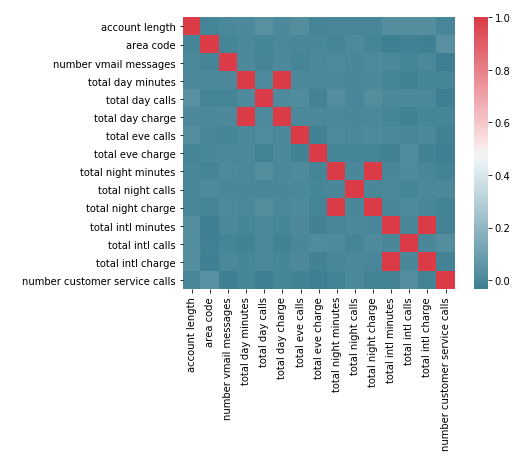
|  |
| --- |
| 6 |

2.1.2 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. Below we have used few methods to perform features selection.

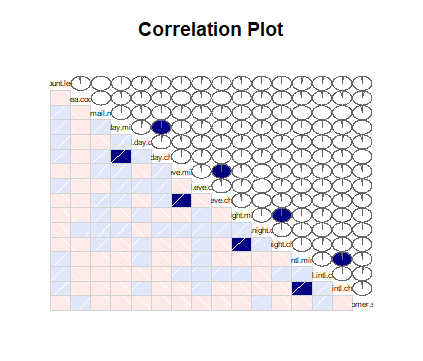
Using python

Correlation analysis for numeric variable



|  |
| --- |
| 7 |

Using R

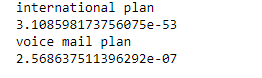


Here, we can see Four variable are highly correlated so no need to use these variables but before drop the variable we will check dependency with target variable.

We found that our target variable are dependent but their will no be problem when we drop these variable because same dependency having that correlated variable.

|  |
| --- |
| 8 |

Chie\_square test for categorical variable



There are no items having value greater than to 0.05 means categorical variable are no correlated so we no need to drop out any attributes as well as our target attributes are dependent on these two attributes also.

|  |
| --- |
| 9 |

2.2 Modeling

2.2.1 Model Selection

In our early stages of analysis during pre-processing we have come to understand the data. We should Start our model building from the simplest to more complex. Therefore we use Multiple Linear Regression.

**Decision Tree**

Decision tree is a type of supervised learning algorithm(having a pre-defined target variable) that is mostly used in classification problems. In this technique, we split the population or sample into two or more homogeneous sets(or sub-populations) based on most significant splitter / differentiator in input variables.

Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable.

**Random Forest**

The mean validation accuracy for this model is 0.856

* RandomForest is a tree based bootstrapping algorithm wherein a certain no. of weak learners (decision trees) are combined to make a powerful prediction model.
* For every individual learner(decision trees), a random sample of rows and a few randomly chosen variables are used to build a decision tree model.
* Final prediction can be a function of all the predictions made by the individual learners.

|  |
| --- |
| 10 |

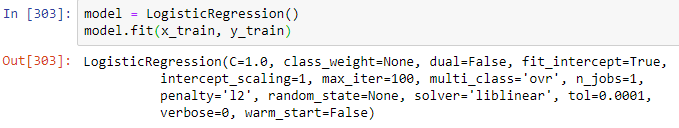
**Logistic Regression**

We started with Logistic Regression which is used for predicting binary outcome.

* Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, Churn / nun\_churn) given a set of independent variables.
* Logistic regression is an estimation of Logit function. Logit function is simply a log of odds in favor of the event.

Sklearn requires the target variable in a separate dataset. So, we droped our target variable from the train dataset and save it in another dataset.

we maked dummy variables for the categorical variables. Dummy variable turns categorical variables into a series of 0 and 1, making them lot easier to quantify and compare.



|  |
| --- |
| 11 |

**Chapter 3**

**Model Evaluation**

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria: 1. Predictive Performance 2. Interpretability 3.Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

**3.1.1 Mean Absolute Error (MAE)**

MAE is one of the error measures used to calculate the predictive performance of the model

**3.1.2 Mean Squared Error (MSE)**

|  |
| --- |
| 12 |

3.2 Model Selection

We can see that both models perform comparatively on average and therefore we can select one of the models without any loss of information.

|  |
| --- |
| 13 |

**Appendix A - python Code**

#!/usr/bin/env python

# coding: utf-8

# In[ ]:

#Load libraries

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import chi2\_contingency

import seaborn as sns

from random import randrange, uniform

# In[ ]:

train = pd.read\_csv(r"C:\Users\anupr\Desktop\churn\Train\_data.csv")

test = pd.read\_csv(r"C:\Users\anupr\Desktop\churn\Test\_data.csv")

# In[ ]:

train.shape

test.shape

# In[ ]:

train.columns

test.columns

# In[ ]:

train.dtypes

test.dtypes

# In[ ]:

plt.boxplot(train['number vmail messages'])

# In[ ]:

plt.boxplot(train['total day calls'])

# In[ ]:

plt.boxplot(train['total night minutes'])

# In[ ]:

#save numeric names

cnames = ["account length","area code","number vmail messages","total day minutes", "total day calls", "total day charge", "total eve calls", "total eve charge", "total night minutes", "total night calls", "total night charge", "total intl minutes

# In[ ]:

# #Detect and delete outliers from data

for i in cnames:

print(i)

q75, q25 = np.percentile(train.loc[:,i], [75 ,25])

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

#print(min)

#print(max)

train = train.drop(train[train.loc[:,i] < min].index)

train = train.drop(train[train.loc[:,i] > max].index)

# In[ ]:

# #Detect and delete outliers from data

for i in cnames:

print(i)

q75, q25 = np.percentile(test.loc[:,i], [75 ,25])

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

#print(min)

#print(max)

test = test.drop(test[test.loc[:,i] < min].index)

test = test.drop(test[test.loc[:,i] > max].index)

# In[ ]:

train.isnull().sum()

# # Feature Selection

# In[ ]:

##Correlation analysis

#Correlation plot

df\_corr = train.loc[:,cnames]

# In[ ]:

#Set the width and hieght of the plot

f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix

corr = df\_corr.corr()

#Plot using seaborn library

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),

square=True, ax=ax)

# In[ ]:

train = train.drop(['total day charge', 'total intl charge', 'total night charge'], axis=1)

# In[ ]:

test = test.drop(['total day charge', 'total intl charge', 'total night charge'], axis=1)

# In[ ]:

#Chisquare test of independence

#Save categorical variables

cat\_names = ["state", "international plan", "voice mail plan"]

# In[ ]:

#loop for chi square values

for i in cat\_names:

print(i)

chi2, p, dof, ex = chi2\_contingency(pd.crosstab(train['Churn'], train[i]))

print(p)

# In[ ]:

df = train.copy()

train = df.copy()

# In[ ]:

#save continous names

cnames = ["total day minutes", "total day calls", "total eve minutes", "total eve charge", "total night minutes", "total intl minutes"]

# In[ ]:

def Normalise(df):

for i in cnames:

#print(i)

df[i] = (df[i] - df[i].min())/(df[i].max()-df[i].min())

return df

#Normalization

train = Normalise(train)

test = Normalise(test)

# In[ ]:

X = train.drop('Churn',1)

y = train.Churn

# In[ ]:

X = test.drop('Churn',1)

y = test.Churn

# In[ ]:

X=pd.get\_dummies(X)

train=pd.get\_dummies(train)

test=pd.get\_dummies(test)

# In[ ]:

from sklearn.model\_selection import train\_test\_split

# In[ ]:

x\_train, x\_cv, y\_train, y\_cv = train\_test\_split(X,y, test\_size =0.6)

# In[ ]:

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# In[ ]:

model = LogisticRegression()

model.fit(x\_train, y\_train)

# In[ ]:

pred\_cv = model.predict(x\_cv)

# In[ ]:

accuracy\_score(y\_cv,pred\_cv)

# In[ ]:

pred\_test = model.predict(test)

# In[ ]:

sample = pd.read\_csv(r"C:\Users\anupr\Desktop\churn\Sample\_output.csv")

# In[ ]:

sample['Churn']=pred\_test

sample['phone\_number']=test\_original['phone\_number']

# In[ ]:

sample['Churn'].replace(0, 'N',inplace=True)

sample['Churn'].replace(1, 'Y',inplace=True)

# In[ ]:

pd.df(sample, columns=['phone\_number','Churn']).to\_csv('logistic.csv')

# # Decision Tree

# In[ ]:

from sklearn import tree

# In[ ]:

i=1

kf = StratifiedKFold(n\_splits=5,random\_state=1,shuffle=True)

for train\_index,test\_index in kf.split(X,y):

print('\n{} of kfold {}'.format(i,kf.n\_splits))

xtr,xvl = X.loc[train\_index],X.loc[test\_index]

ytr,yvl = y[train\_index],y[test\_index]

model = tree.DecisionTreeClassifier(random\_state=1)

model.fit(xtr, ytr)

pred\_test = model.predict(xvl)

score = accuracy\_score(yvl,pred\_test)

print('accuracy\_score',score)

i+=1

pred\_test = model.predict(test)

# In[ ]:

sample['Churn']=pred\_test

sample['phone\_number']=test\_original['phone\_number']

# In[ ]:

sample['Churn'].replace(0, 'N',inplace=True)

sample['Churn'].replace(1, 'Y',inplace=True)

# In[ ]:

pd.df(sample, columns=['phone\_number','Churn']).to\_csv('decesion\_tree.csv')

# # Random Forest

# In[ ]:

from sklearn.ensemble import RandomForestClassifier

# In[ ]:

i=1

kf = StratifiedKFold(n\_splits=5,random\_state=1,shuffle=True)

for train\_index,test\_index in kf.split(X,y):

print('\n{} of kfold {}'.format(i,kf.n\_splits))

xtr,xvl = X.loc[train\_index],X.loc[test\_index]

ytr,yvl = y[train\_index],y[test\_index]

model = RandomForestClassifier(random\_state=1, max\_depth=10)

model.fit(xtr, ytr)

pred\_test = model.predict(xvl)

score = accuracy\_score(yvl,pred\_test)

print('accuracy\_score',score)

i+=1

pred\_test = model.predict(test)

# In[ ]:

sample['Churn']=pred\_test

sample['phone\_number']=test\_original['phone\_number']

# In[ ]:

sample['Churn'].replace(0, 'N',inplace=True)

sample['Churn'].replace(1, 'Y',inplace=True)

# In[ ]:

pd.df(sample, columns=['phone\_number','Churn']).to\_csv('random.csv')