Churn Reduction

Abhishek Hupele

27 May 2018

Contents

Introduction

1.1 Problem Statement

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

1.2 Data

Our task is to build classification models which will classify the customer movement or churn based on usage pattern data.

Data Sets -
1) Test_data.csv (1667, 21)

2) Train_data.csv (3333, 21)

The variable indicating customer movement in data :
'Churn'
Given below is a sample of the train data set that we are using to develop models :

Table 1.1: Train data (Columns: 1-11)

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes
KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4
OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5
NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2

Table 1.2: Train data (Columns: 12-21)

total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
99	16.78	244.7	91	11.01	10	3	2.7	1	False.
103	16.62	254.4	103	11.45	13.7	3	3.7	1	False.
110	10.3	162.6	104	7.32	12.2	5	3.29	0	False.

As you can see in the table below we have the following 21 predictors :

Table 1.3: Predictor Variables

• account length
• international plan
• voicemail plan
• number of voicemail messages
• total day minutes used
• day calls made
• total day charge
• total evening minutes
• total evening calls
• total evening charge
• total night minutes
• total night calls
• total night charge
• total international minutes used
• total international calls made
• total international charge
• number of customer service calls made

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 numeric variables. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.
In Figure 2.1 we have plotted the probability density functions of all the numeric predictors we have available in the data as well as the dependent 'Churn' variable. The blue lines indicate Kernel Density Estimations (KDE)[1] of the variable. The red lines represent the normal distribution. You can see in the figure most variables either very closely, or somewhat imitate the normal distribution but not all do.

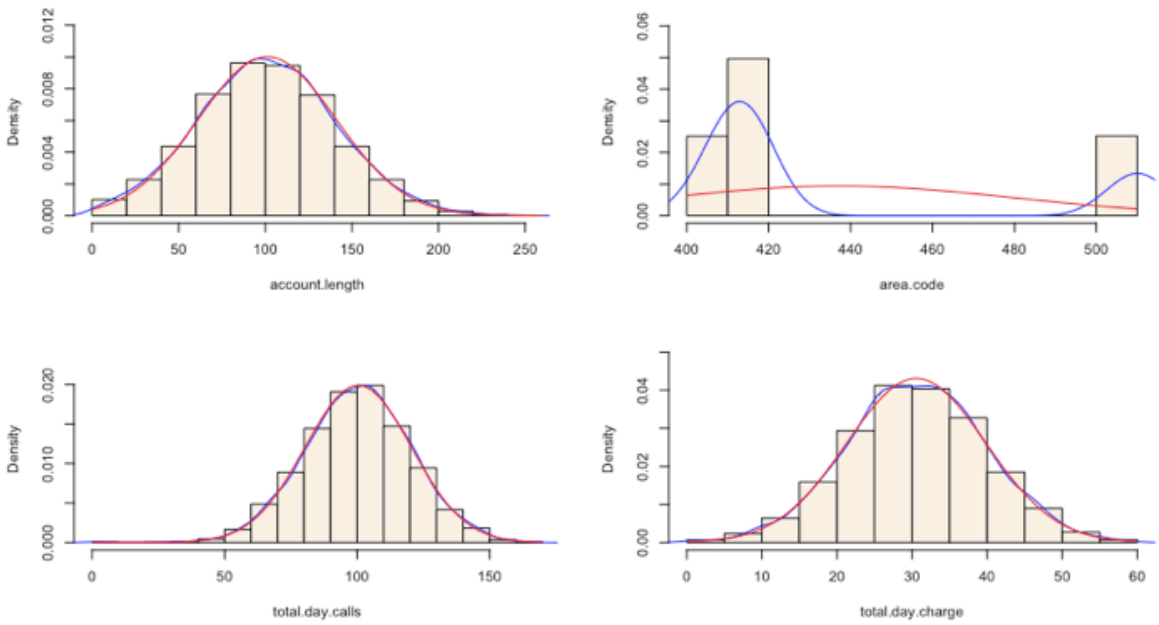


Figure 2.1: Probability Density Functions of Continuous Predictor Variables of Train data (See R code in Appendix)

In Table 1.4 we have plotted the summary functions of all the categorical predictors we have available in the data as well as the dependent 'Churn' variable.

Table 1.4: Categorical Predictor Variables

state		phone.number	international.plan	Churn
WV	106	327-1058: 1	no :3010	False.:2850
MN	84	327-1319: 1	yes: 323	True. : 483
NY	83	327-3053: 1	voice.mail.plan	
AL	80	327-3587: 1	no :2411	
OH	78	327-3850: 1	yes: 922	
OR	78	327-3954: 1		
(Other)	2824	(Other) :3327		

2.1.1 Outlier Analysis

We can clearly observe from these probability distributions that most of the variables are normal, for example, account length, total evening charge , total international calls made ,but some variables have few outliers also, for example, number of voicemail messages .
One of the other steps of pre-processing apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers, Tukey's method. We visualize the outliers using boxplots.
In figure 2.2 we have plotted the boxplots of the 15 continuous predictor variables with respect to each 'Churn' value 'yes' and 'no' . As you can see, we have only a few outliers and extreme values in the the data set, similar results were drawn from probability density function which shows very less skewness and near normal curves.The data is representative talk-time data set of customers and it shows that usage patterns are normal and follows bell curve.

We will process this data for whatever small outlier are present and plot it again after outlier removal in the next sections. Outlier analysis is done for all the continuous predictor variables.

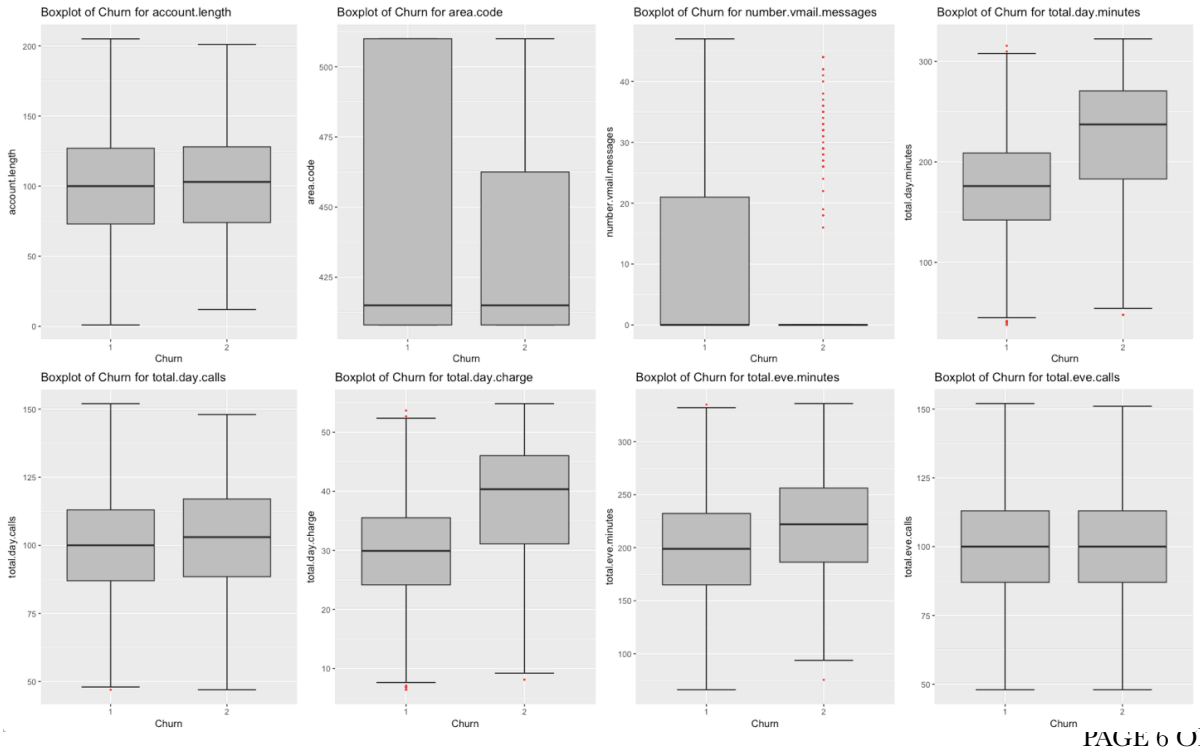


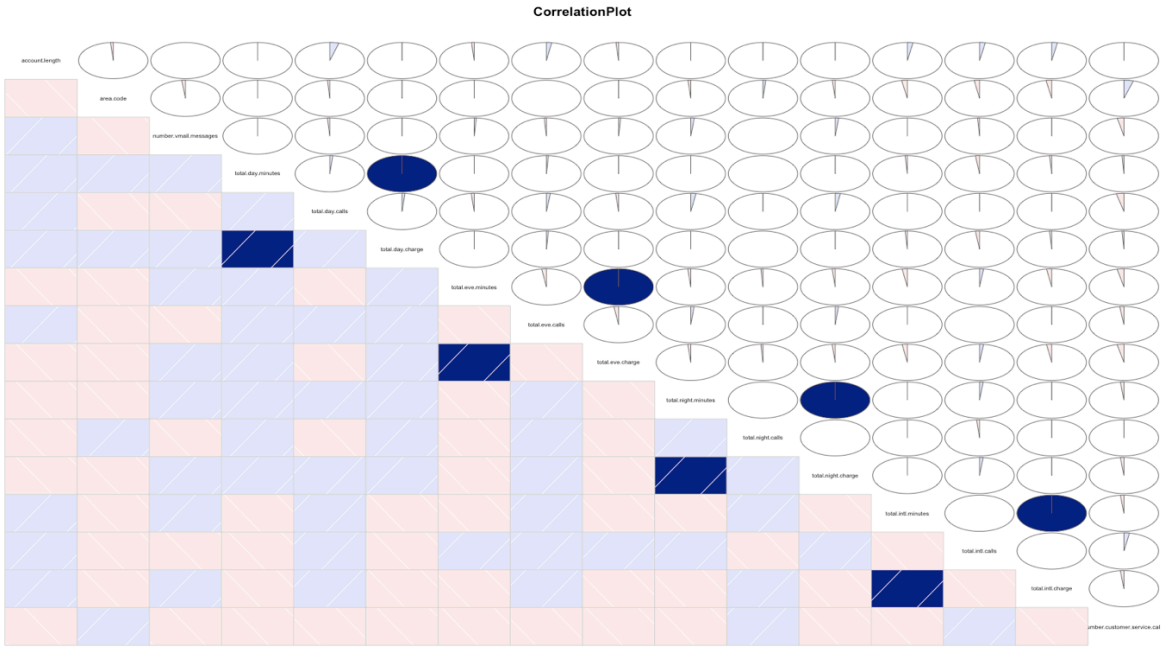
Figure 2.2: Boxplots for all the Continuous Predictor Variables for Train Data (See R code in Appendix)

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. We try to select a subset of relevant features (variables, predictors) for use in model construction. There are several methods to identify and remove irrelevant attributes from data that do not contribute much information.

Below we have used correlation analysis (continuous variables) and chi-square test (categorical variables) to perform features selection.

From figure 2.3 , we can see that variables 'total.day.minutes', 'total.eve.minutes', 'total.night.minutes', 'total.intl.minutes' are highly correlated.




```
                          ac. ar. nm.. ttl.dy.m ttl.dy.cl ttl.dy.ch ttl.v.m ttl.v.cl ttl.v.ch ttl.nght.m ttl.nght.cl ttl.nght.ch ttl.ntl.m ttl.ntl.cl ttl.ntl.ch n...
account.length            1
area.code                     1
number.vmail.messages             1
total.day.minutes                     1
total.day.calls                               1
total.day.charge                      1               1
total.eve.minutes                                             1
total.eve.calls                                                       1
total.eve.charge                                              1               1
total.night.minutes                                                                   1
total.night.calls                                                                             1
total.night.charge                                                                    1               1
total.intl.minutes                                                                                            1
total.intl.calls                                                                                                      1
total.intl.charge                                                                                             1               1
number.customer.service.calls                                                                                                                 1
attr(,"legend")
[1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

Figure 2.3: Correlation plot for all the Continuous Predictor Variables for Train Data (See R code in Appendix)

Chi-square test compares categorical variables to target variable to see if they are related. If chi-square statistic is greater than critical value then we reject the null hypothesis : Two variables are independent.

From table 1.5, we can see that for variable 'phone.number' the p-value is greater than critical value 0.05 thus null hypothesis is rejected.

Table 1.5: Chi-sq Probability Value for categorical value

```
[1] "state"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 83.044, df = 50, p-value = 0.002296

[1] "area.code"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 0.17754, df = 2, p-value = 0.9151

[1] "phone.number"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 85.655, df = 95, p-value = 0.743

[1] "international.plan"

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 222.57, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 34.132, df = 1, p-value = 5.151e-09

Warning messages:
1: In chisq.test(table(factor_data$Churn, factor_data[, i])) :
  Chi-squared approximation may be incorrect
2: In chisq.test(table(factor_data$Churn, factor_data[, i])) :
  Chi-squared approximation may be incorrect
```

2.1.3 Feature Scaling

In order to reduce unwanted variation either within or between variables, we perform either normalisation or standardisation. As all the variables are not normally distributed, we will bring the continuous variables within range(0,1) by performing normalisation.

$$z = \frac{\mathrm{x} - min(x)}{[\max(x) - min(x)]}$$

Where x is an original value, z is the normalised value

2.2 Modeling

2.2.1 Model Selection

We are trying to predict results in a discrete output, which makes it a classification problem. The dependent variable ‘Churn’ is expected to have output, that is, if the customer has moved (1=yes; 0 = no).

We can build classification models by using several learning algorithms like decision tree, knn, naive bayes, random forest, logistic regression. As acquiring new customer is costly and it is financially less burdensome to retain existing customer we would like to predict the churn ‘yes’ customers more accurately and reduce inaccuracy in predicting how many customers are going to churn ‘yes’ but model predicted churn ‘no’.

Based on the performance measured which is accuracy and false negative rate in this case, we can select appropriate model.

2.2.1 KNN

KNN is time consuming but could be accurate learning algorithm, as it compares all variables by measuring distance between data set.

Confusion Matrix and Statistics

```
knn_pred    1    2
       1 1361  171
       2   82   53

               Accuracy : 0.8482
                 95% CI : (0.8301, 0.8651)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 0.9817

                  Kappa : 0.216
 Mcnemar's Test P-Value : 3.157e-08

            Sensitivity : 0.9432
            Specificity : 0.2366
         Pos Pred Value : 0.8884
         Neg Pred Value : 0.3926
             Prevalence : 0.8656
         Detection Rate : 0.8164
   Detection Prevalence : 0.9190
      Balanced Accuracy : 0.5899

       'Positive' Class : 1
```

2.2.2 Naive Bayes

It works on Bayes theorem of probability to predict the class of unknown data set, with assumption that all independent input variables are conditionally independent.

Confusion Matrix and Statistics

```
          predicted
observed    1    2
       1 1423   20
       2  177   47

               Accuracy : 0.8818
                 95% CI : (0.8654, 0.8969)
    No Information Rate : 0.9598
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2784
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8894
            Specificity : 0.7015
         Pos Pred Value : 0.9861
         Neg Pred Value : 0.2098
             Prevalence : 0.9598
         Detection Rate : 0.8536
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.7954

       'Positive' Class : 1
```

2.2.3 Random Forest

Random Forest is an ensemble that consists of many decision trees. It also gives what are important variables which help to predict or classify for new test cases.

Confusion Matrix and Statistics

```
  rf_pred
     1    2
1 1416   27
2  135   89

               Accuracy : 0.9028
                 95% CI : (0.8876, 0.9166)
    No Information Rate : 0.9304
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4754
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9130
            Specificity : 0.7672
         Pos Pred Value : 0.9813
         Neg Pred Value : 0.3973
             Prevalence : 0.9304
         Detection Rate : 0.8494
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8401

       'Positive' Class : 1
```

2.2.4 Decision Tree

Decision tree is a predictive model based on a branching series of boolean tests. We are using C5.0 decision tree algorithm.

Confusion Matrix and Statistics

```
  c50_pred
     1    2
1 1442    1
2  126   98

               Accuracy : 0.9238
                 95% CI : (0.91, 0.9361)
    No Information Rate : 0.9406
    P-Value [Acc > NIR] : 0.9978

                  Kappa : 0.5715
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9196
            Specificity : 0.9899
         Pos Pred Value : 0.9993
         Neg Pred Value : 0.4375
             Prevalence : 0.9406
         Detection Rate : 0.8650
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9548

       'Positive' Class : 1
```

2.2.5 Logistic Regression

Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

Confusion Matrix and Statistics

```
> conf_mat
  logis_pred
     0    1
1 1411   32
2  165   59
```

Conclusion

3.1 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. As acquiring new customer is costly and it is financially less burdensome to retain existing customer we would like to predict the churn ‘yes’ customers more accurately and reduce inaccuracy in predicting how many customers are going to churn ‘yes’ but model predicted churn ‘no’.

Based on the performance measured which is Accuracy and False Negative Rate in this case, we can select appropriate model.

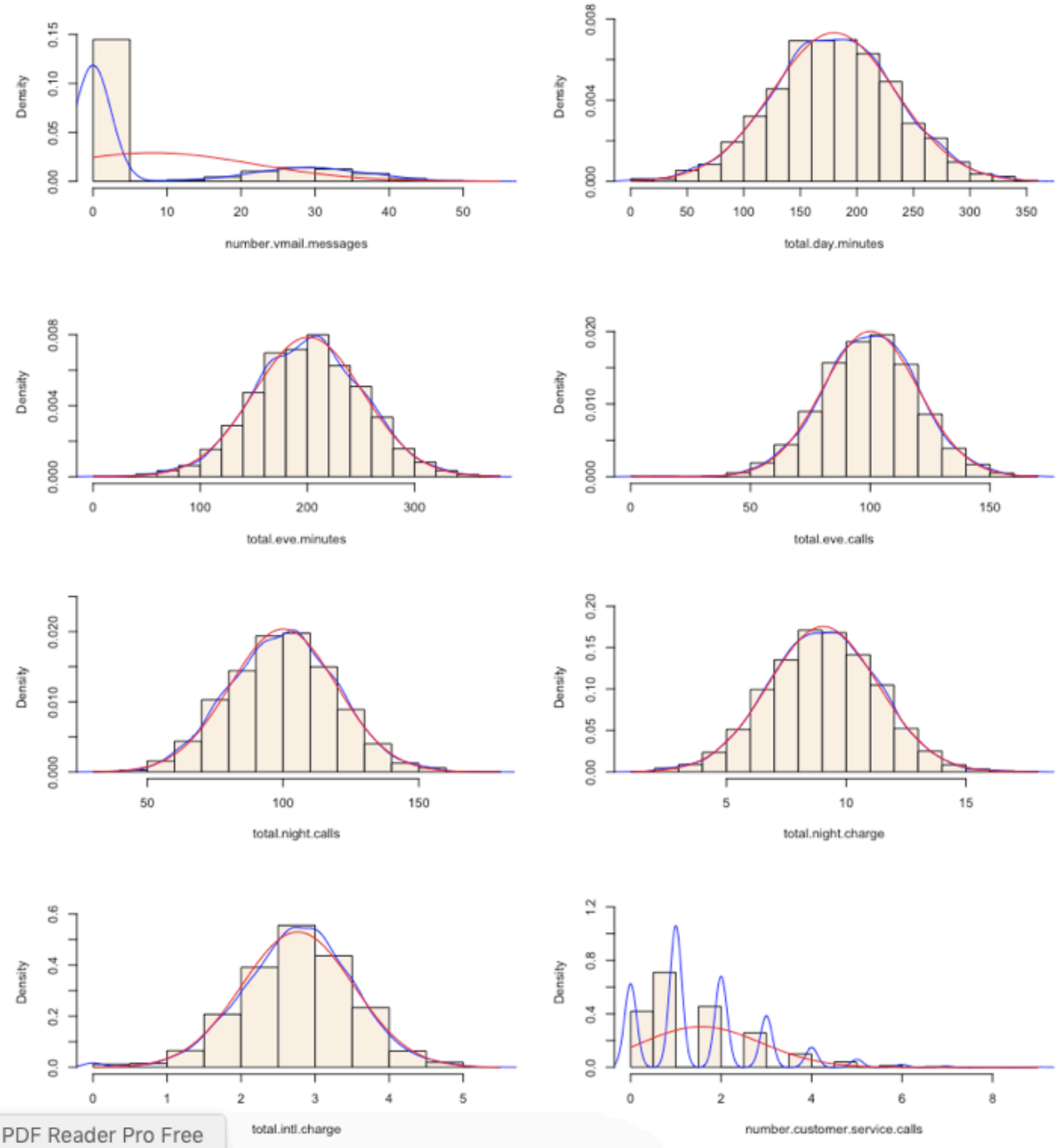
Table 1.6: FNR & Accuracy table for Models

	FNR	Accuracy
KNN	**61**	**85**
Naive Bayes	**79**	**88**
Random Forest	**60**	**90**
Decision Tree	**56**	**92**
LogisticRegression	**73**	**88**

3.2 Model Selection

We can see that Decision Tree gives lowest FNR and highest Accuracy, making it suitable for the current problem.

Appendix A - Extra Figures



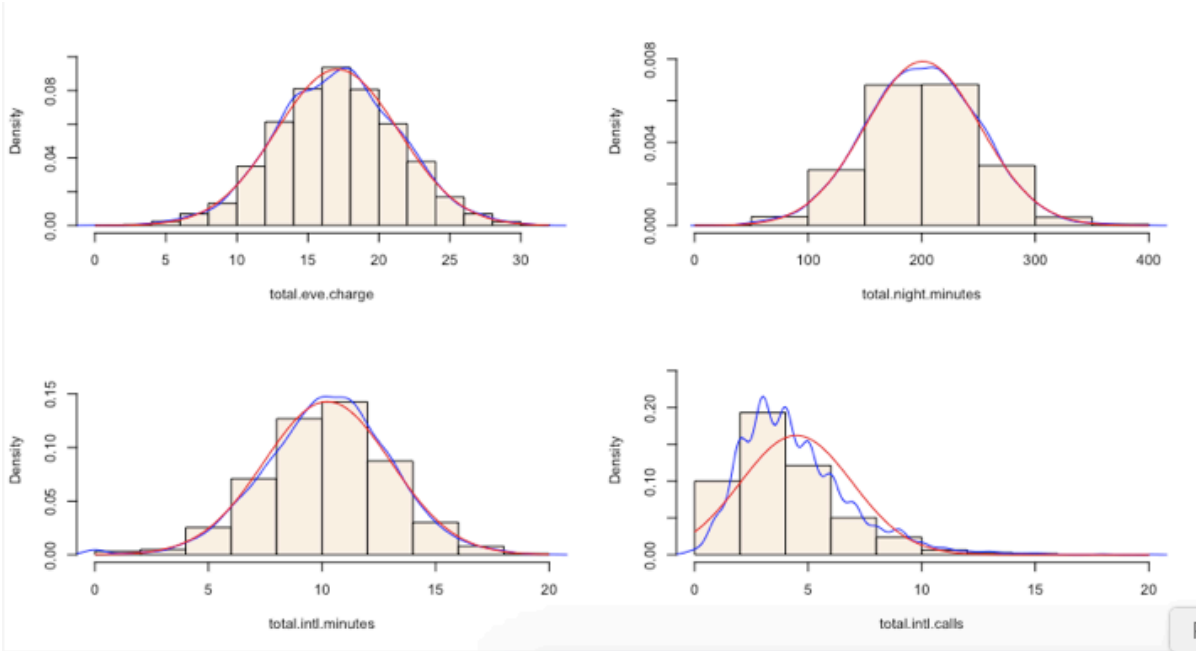


Figure 2.1: Probability Density Functions of Continuous Predictors Variables of Train data

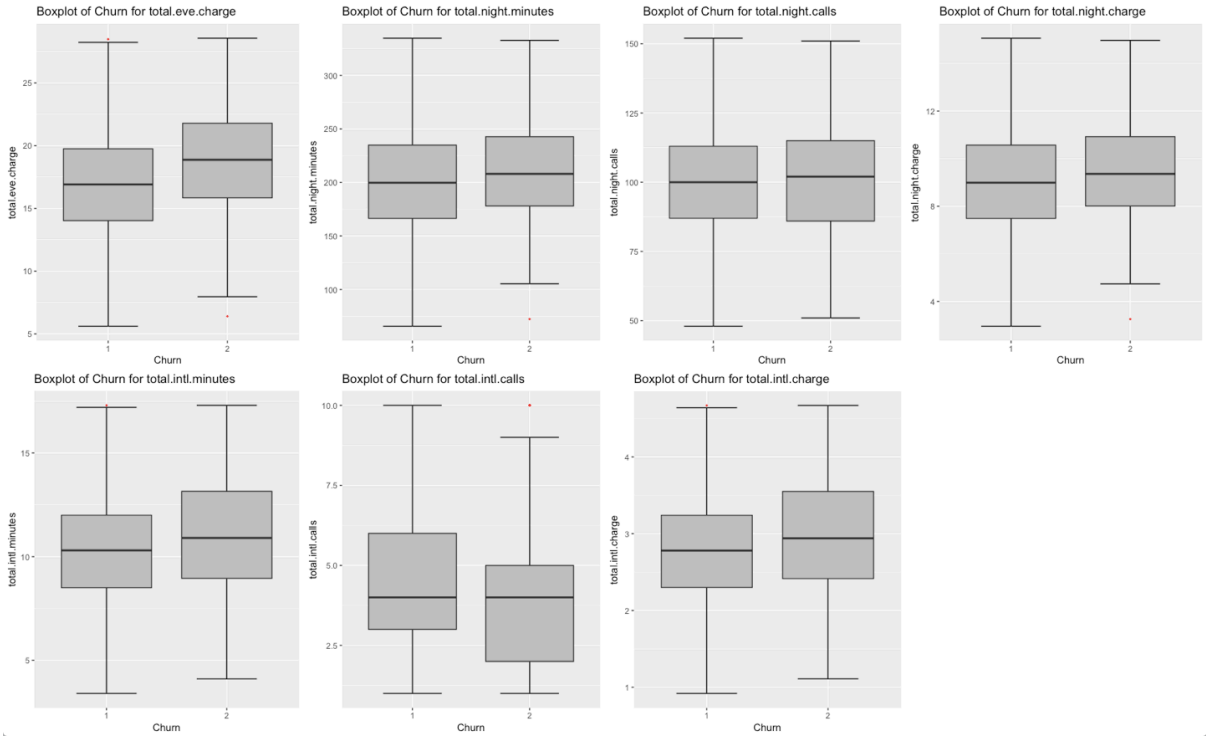


Figure 2.2: Boxplots for all the Continuous Predictor Variables for Train Data

Appendix B - R Code

Complete R File

```
rm(list = ls())
#   install.packages("corrgram",   lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/
library")
#   install.packages("sampling",   lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/
library")
#  install.packages("DataCombine",  lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/
library")
# install.packages("caret", lib="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
library(psych)
library(ggplot2)
library(corrgram)
library(sampling)
library(corrgram)
library(class)
library(e1071)
library(caret)
library(DataCombine)
library(caret)
library(randomForest)
library(inTrees)
library(C50)

train = read.csv('Train_data.csv', header = TRUE)
test = read.csv('Test_data.csv', header = TRUE)

numeric_index = sapply(train,is.numeric)
numeric_data = train[,numeric_index]
cnames = colnames(numeric_data)
# Figure 2.1: Probability Density Functions of Continuous Predictor Variables of Train data
# multi.hist(train[,numeric_index], main = NA, dcol = c("blue", "red"), dlty = c("solid", "solid"), bcol =
"linen")
train$area.code = factor(train$area.code, labels = (1:length(levels(factor(train$area.code)))))
train$phone.number = as.factor(substr(train$phone.number,2,4))
numeric_index = sapply(train,is.numeric)
numeric_data = train[,numeric_index]
cnames = colnames(numeric_data)
factor_index = sapply(train, is.factor)

# Table 1.4: Categorical Predictor Variables
summary(train[,factor_index])
factor_index = sapply(train,is.factor)
factor_data = train[,factor_index]
cnames_f = colnames(factor_data)
for (i in 1:ncol(train)) {
  if (class(train[,i]) == 'factor') {
    train[,i] = factor(train[,i],labels = 1:length(levels(factor(train[,i]))))
  }
}
```

```
for (i in 1:length(cnames)){
  assign(paste0('plot',i), ggplot(aes_string(y = cnames[i], x = 'Churn'), data = subset(train)) +
          stat_boxplot(geom = 'errorbar', width = 0.5) +
             geom_boxplot(outlier.colour = 'red', fill = 'grey', outlier.shape = 18, outlier.size = 1, notch =
FALSE) +
          theme(legend.position = 'bottom') +
          labs(y = cnames[i], x = 'Churn') +
          ggtitle(paste('Boxplot of Churn for', cnames[i])))
}
# Figure 2.2: Boxplots for all the Continuous Predictor Variables for Train Data
gridExtra::grid.arrange(plot1,plot2,plot3,plot4,plot5,plot6,plot7,plot8, ncol = 4)
gridExtra::grid.arrange(plot9,plot10,plot11,plot12,plot13,plot14,plot15,ncol = 4)

for (i in cnames) {
  outlier =  train[,i][train[,i] %in% boxplot.stats(train[,i])$out]
  train = train[which(!train[,i] %in% outlier),]
}
# Figure 2.3: Correlation plot for all the Continuous Predictor Variables for Train Data
corrgram(train[,numeric_index], order = F, upper.panel = panel.pie, text.panel = panel.txt, main =
'CorrelationPlot')
symnum(cor(train[,numeric_index]))
# Table 1.5: Chi-sq Probability Value for categorical value
for (i in cnames_f[1:5]) {
  print(i)
  print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}
train_deleted = subset(train, select = -c(area.code, phone.number, total.day.minutes,
total.eve.minutes, total.night.minutes, total.intl.minutes))
numeric_index = sapply(train_deleted,is.numeric)
numeric_data = train_deleted[,numeric_index]
cnames = colnames(numeric_data)

for ( i in cnames) {
      train_deleted[,i] = (train_deleted[,i] - min(train_deleted[,i]))/(max(train_deleted[,i]) -
min(train_deleted[,i]))
}

test = read.csv('Test_data.csv', header = TRUE)
test$area.code = factor(test$area.code, labels = (1:length(levels(factor(test$area.code)))))
test$phone.number = as.factor(substr(test$phone.number,2,4))
numeric_index = sapply(test,is.numeric)
factor_index = sapply(test, is.factor)
for (i in 1:ncol(test)) {
  if (class(test[,i]) == 'factor') {
    test[,i] = factor(test[,i],labels = 1:length(levels(factor(test[,i]))))
  }
}
numeric_data = test[,numeric_index]
cnames = colnames(numeric_data)
factor_index = sapply(test,is.factor)
factor_data = test[,factor_index]
cnames_f = colnames(factor_data)
test_deleted = subset(test, select = -c(area.code,phone.number, total.day.minutes, total.eve.minutes,
total.night.minutes, total.intl.minutes))
numeric_index = sapply(test_deleted,is.numeric)
cnames = colnames(test_deleted[,numeric_index])
for ( i in cnames) {
  test_deleted[,i] = (test_deleted[,i] - min(test_deleted[,i]))/(max(test_deleted[,i]) - min(test_deleted[,i]))
```

```
}

rmExcept(c('test_deleted','train_deleted'))
train = train_deleted
test = test_deleted

knn_pred = knn(train[,1:14],test[,1:14],train$Churn,k=1)
conf_mat = table(knn_pred, test$Churn)

nb_model = naiveBayes(Churn~.,data = train)
nb_pred = predict(nb_model, test[,1:14], type = 'class')
conf_mat = table(observed = test[,15], predicted = nb_pred)

rf_model = randomForest(Churn~.,train, importance = TRUE, ntree = 550)
rf_pred = predict(rf_model, test[,-15])
conf_mat = table(test$Churn,rf_pred)

c50_model = C5.0(Churn~., train, trials = 100, rules = TRUE)
c50_pred = predict(c50_model,test[,-15],type = 'class')
conf_mat = table(test$Churn,c50_pred)

logis_model= glm(Churn~., data = train, family = 'binomial')
logis_pred = predict(logis_model, newdata = test, type = 'response')
logis_pred = ifelse(logis_pred > 0.5,1,0)
conf_mat = table(test$Churn, logis_pred)

# conf_mat
# confusionMatrix(conf_mat)
TN = conf_mat[1,1]
FN = conf_mat[1,2]
TP = conf_mat[2,1]
FP = conf_mat[2,2]
(FN*100)/(FN+TP)
(TP+TN)*100/(TN+FN+TP+FN)
```

Complete Python File

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm

train = pd.read_csv('Train_data.csv')
test = pd.read_csv('Test_data.csv')

train['area code'] = train['area code'].astype(object)

for i in range(train.shape[0]) :
    temp = train['phone number'][i].split('-')
    train['phone number'][i] = temp[0]

for i in train.columns :
    if (train[i].dtypes == 'object')  :
        print(i)
        train[i] = pd.Categorical(train[i])
        train[i] = train[i].cat.codes


get_ipython().run_line_magic('matplotlib', 'inline')
plt.boxplot(train['total day calls'])

cnames = ['account length','number vmail messages','total day minutes','total day calls',
            'total day charge', 'total eve minutes','total eve calls','total eve charge',
            'total night minutes','total night calls', 'total night charge','total intl minutes',
            'total intl calls','total intl charge', 'number customer service calls']

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)

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

```
df_corr = train.loc[:,cnames]
f, ax = plt.subplots(figsize = (7,5))
corr = df_corr.corr()
sns.heatmap(corr, mask = np.zeros_like(corr, dtype = np.bool), cmap =
sns.diverging_palette(220,10,as_cmap = True),
          square = True, ax = ax)
cat_names = ['state','area code','phone number','international plan','voice mail plan']
for i in cat_names :
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(train['Churn'],train[i]))
    print(p)

train = train.drop(['area code', 'phone number','total day minutes','total eve minutes',
                    'total night minutes', 'total intl minutes'], axis = 1)

get_ipython().run_line_magic('matplotlib', 'inline')
plt.hist(train['account length'], bins = 'auto')

cnames = ['account length', 'number vmail messages', 'total day calls', 'total day charge',
        'total eve calls', 'total eve charge', 'total night calls',
        'total night charge', 'total intl calls', 'total intl charge',
        'number customer service calls']

for i in cnames :
    train[i] = (train[i] - train[i].min())/(train[i].max() - train[i].min())

test['area code'] = test['area code'].astype(object)
for i in range(test.shape[0]) :
    temp = test['phone number'][i].split('-')
    test['phone number'][i] = temp[0]

for i in test.columns :
    if (test[i].dtypes == 'object')  :
        print(i)
        test[i] = pd.Categorical(test[i])
        test[i] = test[i].cat.codes
test = test.drop(['area code', 'phone number','total day minutes','total eve minutes',
                    'total night minutes', 'total intl minutes'], axis = 1)
cnames = ['account length', 'number vmail messages', 'total day calls', 'total day charge',
        'total eve calls', 'total eve charge', 'total night calls',
        'total night charge', 'total intl calls', 'total intl charge',
        'number customer service calls']
for i in cnames :
    test[i] = (test[i] - test[i].min())/(test[i].max() - test[i].min())

train['Churn'] = train['Churn'].replace(1,'yes')
train['Churn'] = train['Churn'].replace(0,'no')

x_train = train.values[:,0:14]
y_train = train.values[:,14]

test['Churn'] = test['Churn'].replace(1,'yes')
test['Churn'] = test['Churn'].replace(0,'no')

x_test = test.values[:,0:14]
y_test = test.values[:,14]

clf = tree.DecisionTreeClassifier(criterion = 'entropy').fit(x_train,y_train)
```

```
# clf = tree.DecisionTreeClassifier(criterion = 'gini').fit(x_train,y_train)
dt_pred = clf.predict(x_test)
CM = pd.crosstab(y_test,dt_pred)

knn_model = KNeighborsClassifier(n_neighbors = 1).fit(x_train,y_train)
knn_pred = knn_model.predict(x_test)
CM = pd.crosstab(y_test,knn_pred)

nb_model = GaussianNB().fit(x_train,y_train)
nb_pred = nb_model.predict(x_test)
CM = pd.crosstab(y_test,nb_pred)

rf_model = RandomForestClassifier(n_estimators = 100).fit(x_train, y_train)
rf_pred = rf_model.predict(x_test)
CM = pd.crosstab(y_test,nb_pred)

train['Churn'] = train['Churn'].replace('no',0)
train['Churn'] = train['Churn'].replace('yes',1)
train_logit = pd.DataFrame(train['Churn'])
test_logit = pd.DataFrame(test['Churn'])
cnames = ['account length','number vmail messages','total day calls','total day charge',
        'total eve calls','total eve charge','total night calls',
        'total night charge','total intl calls','total intl charge',
        'number customer service calls']
train_logit = train_logit.join(train[cnames])
test_logit = test_logit.join(test[cnames])
cat_names = ['state','international plan', 'voice mail plan']
for i in cat_names :
    temp = pd.get_dummies(train[i], prefix = i)
    train_logit = train_logit.join(temp)
    temp = pd.get_dummies(test[i], prefix = i)
    test_logit = test_logit.join(temp)
train_cols = train.columns[0:14]
test_cols = test.columns[0:14]
logit = sm.Logit(train['Churn'],train[train_cols]).fit()
test['predict_prob'] = logit.predict(test[test_cols])
test['predict_val'] = 1
test.loc[test.predict_prob < 0.5, 'predict_val'] = 0

CM = pd.crosstab(test['Churn'],test['predict_val'])
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

(FN*100)/(FN+TP)
accuracy_score(y_test,nb_pred)*100
```