K-nearest neighbor (knn) classification

# Part I: Research Question

Which customers have the highest likelihood for churn?

The goal in answering this question is to decrease costs associated with account maintenance. As the cost to on-board a new customer is already known to be 10 times that of retaining an existing customer, one way to cut costs is by raising retention rates among the existing customer base. Therefore, it would be a valuable undertaking for an organization to have the ability to identify existing customers likely to churn. Armed with this knowledge, new strategic initiatives could be developed as needed to target the retention of highly valued existing customers.

The research question will be addressed using the K-nearest neighbor (knn) classification method

# Part II: Method Justification

The knn classification method is chosen as it functions to determine classifications based on historic observations. In training the model using a training data set, the model will register each observation’s properties as coordinates in a feature space. Having been trained, the model can then plot any new observations to be classified within the same feature space. Using distance measurement, the model will classify the new observation based on the classification of the nearest trained or historical observation. In other words, the classification of any new observation is the result of the likeness of its attributes in terms of measured proximity to previous observations. The assumption of the knn algorithm is that similar observations will exist closely to one another when plotting by attribute within a feature space. R will be used for this analysis. R is open source software that was specifically made for statistical analysis. Using R, we can ingest the data set, and leveraging an extensive library of data manipulation and visualization packages, perform the necessary classification steps. More information can be found on the R project website (<https://www.r-project.org/>). The dplyr package will be used for data preparation and manipulation within R. More information for the dplyr package can be found on the tidyverse website (<https://dplyr.tidyverse.org/>). The class package will be used to construct the knn classification model. More information for the class package can be found on the Comprehensive R Archive Network (CRAN) website (<https://cran.r-project.org/web/packages/class/index.html>). Finally, the pROC package will be used for testing the model performance. More information for the pROC package can be found on the CRAN website (<https://cran.r-project.org/web/packages/pROC/index.html>).

# Part III: Data Preparation

For the knn algorithm, all variables to be used within the data set shall be numerical. First, it is necessary to investigate the data set to identify continuous and categorical variables. Once this is known, the data set is ready to be prepared.  
The target variable in this instance being Churn, is categorical in the data set provided. This requires creation of a dummy variable, with 0s representing the “No” values and 1s representing the “Yes” values. Next, it is best to transform the any obscure variable names into a more readily understood naming convention. This is done with the variables “Item 1…. Item8” using the data dictionary provided. Finally, the variables for use in the knn model are selected and the cleaned data set is provided in the Cleaned\_Data.csv attachment.

# Load Dataset  
df<-read.csv("c:/users/shua/documents/Data Mining\_D209/churn\_clean.csv")  
  
# Identify numeric columns for use  
str(df)

## 'data.frame': 10000 obs. of 50 variables:  
## $ CaseOrder : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ Customer\_id : chr "K409198" "S120509" "K191035" "D90850" ...  
## $ Interaction : chr "aa90260b-4141-4a24-8e36-b04ce1f4f77b" "fb76459f-c047-4a9d-8af9-e0f7d4ac2524" "344d114c-3736-4be5-98f7-c72c281e2d35" "abfa2b40-2d43-4994-b15a-989b8c79e311" ...  
## $ UID : chr "e885b299883d4f9fb18e39c75155d990" "f2de8bef964785f41a2959829830fb8a" "f1784cfa9f6d92ae816197eb175d3c71" "dc8a365077241bb5cd5ccd305136b05e" ...  
## $ City : chr "Point Baker" "West Branch" "Yamhill" "Del Mar" ...  
## $ State : chr "AK" "MI" "OR" "CA" ...  
## $ County : chr "Prince of Wales-Hyder" "Ogemaw" "Yamhill" "San Diego" ...  
## $ Zip : int 99927 48661 97148 92014 77461 31030 37847 73109 34771 45237 ...  
## $ Lat : num 56.3 44.3 45.4 33 29.4 ...  
## $ Lng : num -133.4 -84.2 -123.2 -117.2 -95.8 ...  
## $ Population : int 38 10446 3735 13863 11352 17701 2535 23144 17351 20193 ...  
## $ Area : chr "Urban" "Urban" "Urban" "Suburban" ...  
## $ TimeZone : chr "America/Sitka" "America/Detroit" "America/Los\_Angeles" "America/Los\_Angeles" ...  
## $ Job : chr "Environmental health practitioner" "Programmer, multimedia" "Chief Financial Officer" "Solicitor" ...  
## $ Children : int 0 1 4 1 0 3 0 2 2 1 ...  
## $ Age : int 68 27 50 48 83 83 79 30 49 86 ...  
## $ Income : num 28562 21705 9610 18925 40074 ...  
## $ Marital : chr "Widowed" "Married" "Widowed" "Married" ...  
## $ Gender : chr "Male" "Female" "Female" "Male" ...  
## $ Churn : chr "No" "Yes" "No" "No" ...  
## $ Outage\_sec\_perweek : num 7.98 11.7 10.75 14.91 8.15 ...  
## $ Email : int 10 12 9 15 16 15 10 16 20 18 ...  
## $ Contacts : int 0 0 0 2 2 3 0 0 2 1 ...  
## $ Yearly\_equip\_failure: int 1 1 1 0 1 1 1 0 3 0 ...  
## $ Techie : chr "No" "Yes" "Yes" "Yes" ...  
## $ Contract : chr "One year" "Month-to-month" "Two Year" "Two Year" ...  
## $ Port\_modem : chr "Yes" "No" "Yes" "No" ...  
## $ Tablet : chr "Yes" "Yes" "No" "No" ...  
## $ InternetService : chr "Fiber Optic" "Fiber Optic" "DSL" "DSL" ...  
## $ Phone : chr "Yes" "Yes" "Yes" "Yes" ...  
## $ Multiple : chr "No" "Yes" "Yes" "No" ...  
## $ OnlineSecurity : chr "Yes" "Yes" "No" "Yes" ...  
## $ OnlineBackup : chr "Yes" "No" "No" "No" ...  
## $ DeviceProtection : chr "No" "No" "No" "No" ...  
## $ TechSupport : chr "No" "No" "No" "No" ...  
## $ StreamingTV : chr "No" "Yes" "No" "Yes" ...  
## $ StreamingMovies : chr "Yes" "Yes" "Yes" "No" ...  
## $ PaperlessBilling : chr "Yes" "Yes" "Yes" "Yes" ...  
## $ PaymentMethod : chr "Credit Card (automatic)" "Bank Transfer(automatic)" "Credit Card (automatic)" "Mailed Check" ...  
## $ Tenure : num 6.8 1.16 15.75 17.09 1.67 ...  
## $ MonthlyCharge : num 172 243 160 120 150 ...  
## $ Bandwidth\_GB\_Year : num 905 801 2055 2165 271 ...  
## $ Item1 : int 5 3 4 4 4 3 6 2 5 2 ...  
## $ Item2 : int 5 4 4 4 4 3 5 2 4 2 ...  
## $ Item3 : int 5 3 2 4 4 3 6 2 4 2 ...  
## $ Item4 : int 3 3 4 2 3 2 4 5 3 2 ...  
## $ Item5 : int 4 4 4 5 4 4 1 2 4 5 ...  
## $ Item6 : int 4 3 3 4 4 3 5 3 3 2 ...  
## $ Item7 : int 3 4 3 3 4 3 5 4 4 3 ...  
## $ Item8 : int 4 4 3 3 5 3 5 5 4 3 ...

# Dummy column for churn  
library(dplyr)

## Warning: replacing previous import 'vctrs::data\_frame' by 'tibble::data\_frame'  
## when loading 'dplyr'

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

str(df$Churn)

## chr [1:10000] "No" "Yes" "No" "No" "Yes" "No" "Yes" "Yes" "No" "No" "No" ...

df$dummyChurn<-case\_when(  
 df$Churn=="Yes" ~ 1,  
 TRUE ~ 0  
)  
str(df$dummyChurn)

## num [1:10000] 0 1 0 0 1 0 1 1 0 0 ...

# Identify column order for renaming  
colnames(df)

## [1] "CaseOrder" "Customer\_id" "Interaction"   
## [4] "UID" "City" "State"   
## [7] "County" "Zip" "Lat"   
## [10] "Lng" "Population" "Area"   
## [13] "TimeZone" "Job" "Children"   
## [16] "Age" "Income" "Marital"   
## [19] "Gender" "Churn" "Outage\_sec\_perweek"   
## [22] "Email" "Contacts" "Yearly\_equip\_failure"  
## [25] "Techie" "Contract" "Port\_modem"   
## [28] "Tablet" "InternetService" "Phone"   
## [31] "Multiple" "OnlineSecurity" "OnlineBackup"   
## [34] "DeviceProtection" "TechSupport" "StreamingTV"   
## [37] "StreamingMovies" "PaperlessBilling" "PaymentMethod"   
## [40] "Tenure" "MonthlyCharge" "Bandwidth\_GB\_Year"   
## [43] "Item1" "Item2" "Item3"   
## [46] "Item4" "Item5" "Item6"   
## [49] "Item7" "Item8" "dummyChurn"

colnames(df)[43:50]

## [1] "Item1" "Item2" "Item3" "Item4" "Item5" "Item6" "Item7" "Item8"

colnames(df)[43:50]<-list("Timely.Response", "Timely.Fixes", "Timely.Replacements", "Reliability", "Options", "Respectful.Response", "Courteous.Exchange", "Active.Listening")  
colnames(df)

## [1] "CaseOrder" "Customer\_id" "Interaction"   
## [4] "UID" "City" "State"   
## [7] "County" "Zip" "Lat"   
## [10] "Lng" "Population" "Area"   
## [13] "TimeZone" "Job" "Children"   
## [16] "Age" "Income" "Marital"   
## [19] "Gender" "Churn" "Outage\_sec\_perweek"   
## [22] "Email" "Contacts" "Yearly\_equip\_failure"  
## [25] "Techie" "Contract" "Port\_modem"   
## [28] "Tablet" "InternetService" "Phone"   
## [31] "Multiple" "OnlineSecurity" "OnlineBackup"   
## [34] "DeviceProtection" "TechSupport" "StreamingTV"   
## [37] "StreamingMovies" "PaperlessBilling" "PaymentMethod"   
## [40] "Tenure" "MonthlyCharge" "Bandwidth\_GB\_Year"   
## [43] "Timely.Response" "Timely.Fixes" "Timely.Replacements"   
## [46] "Reliability" "Options" "Respectful.Response"   
## [49] "Courteous.Exchange" "Active.Listening" "dummyChurn"

# Select Columns for Model  
df<-df%>%select(dummyChurn, Children, Age, Income, Outage\_sec\_perweek, Tenure, MonthlyCharge, Bandwidth\_GB\_Year, Timely.Response, Timely.Fixes, Timely.Replacements, Reliability, Options, Respectful.Response, Courteous.Exchange, Active.Listening)  
  
# Write cleaned data set  
write.csv(df, "c:/users/shua/documents/Data Mining\_D209/Cleaned\_Data.csv")

# Part IV: Analysis

To begin the analysis, the predictor variable data values are normalized. This is done using applying the function (x - min(x)) / (max(x) - min(x)) to each predictor variable value. This is an important step to mitigate the impact of outliers on the knn algorithm. Next the data is split into training and test data. In this case, 90% of the data set provided will be used as a training data set. This fits the model with historical observations which the model will use to make future predictions. The remaining 10% of the data set will be the test data on which the predictions of churn will be made. The split data sets are stored in the “trainDF” and “testDF” data frames respectively. An analysis of the train and test data frames suing R’s summary function confirms the data has been appropriately normalized and split. Now having normalized the data and split into training and test data sets, the knn classification model is built. First the model is built using the default k=1 attribute. This means the classification for each test observation is determined by its single nearest training neighbor (i.e. historical observation). The resulting accuracy of this approach is 76%. Next, the model is rerun changing the k attribute to 7. This means the classification for each test observation is determined by a majority of the 7 nearest training neighbors in the feature space. The resulting accuracy shows some improvement at 81%. Finally, the model is rerun changing the k attribute to 15. The resulting accuracy nudges even higher to 82%.

# Normalize predictor columns  
normFunc<- function(x) { (x - min(x)) / (max(x) - min(x))}  
normDF<- as.data.frame(lapply(df[,c(2:(length(df)))], normFunc))  
summary(normDF)

## Children Age Income Outage\_sec\_perweek  
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.2394 1st Qu.:0.07301 1st Qu.:0.3751   
## Median :0.1000 Median :0.4930 Median :0.12695 Median :0.4699   
## Mean :0.2088 Mean :0.4941 Mean :0.15261 Mean :0.4691   
## 3rd Qu.:0.3000 3rd Qu.:0.7465 3rd Qu.:0.20459 3rd Qu.:0.5623   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## Tenure MonthlyCharge Bandwidth\_GB\_Year Timely.Response   
## Min. :0.00000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.09743 1st Qu.:0.2855 1st Qu.:0.1543 1st Qu.:0.3333   
## Median :0.48494 Median :0.4163 Median :0.4461 Median :0.3333   
## Mean :0.47220 Mean :0.4408 Mean :0.4622 Mean :0.4151   
## 3rd Qu.:0.85184 3rd Qu.:0.5745 3rd Qu.:0.7754 3rd Qu.:0.5000   
## Max. :1.00000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Timely.Fixes Timely.Replacements Reliability Options   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3333 1st Qu.:0.2857 1st Qu.:0.3333 1st Qu.:0.3333   
## Median :0.5000 Median :0.2857 Median :0.3333 Median :0.3333   
## Mean :0.4175 Mean :0.3553 Mean :0.4163 Mean :0.4155   
## 3rd Qu.:0.5000 3rd Qu.:0.4286 3rd Qu.:0.5000 3rd Qu.:0.5000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Respectful.Response Courteous.Exchange Active.Listening  
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2857 1st Qu.:0.3333 1st Qu.:0.2857   
## Median :0.2857 Median :0.5000 Median :0.2857   
## Mean :0.3568 Mean :0.4183 Mean :0.3565   
## 3rd Qu.:0.4286 3rd Qu.:0.5000 3rd Qu.:0.4286   
## Max. :1.0000 Max. :1.0000 Max. :1.0000

# Split training and testing  
sampNumbers<-sort(sample(nrow(normDF), nrow(normDF)\*.9))  
trainDF<-normDF[sampNumbers,]  
testDF<-normDF[-sampNumbers,]  
summary(trainDF)

## Children Age Income Outage\_sec\_perweek  
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.2394 1st Qu.:0.07257 1st Qu.:0.3733   
## Median :0.1000 Median :0.4930 Median :0.12639 Median :0.4688   
## Mean :0.2084 Mean :0.4931 Mean :0.15290 Mean :0.4683   
## 3rd Qu.:0.3000 3rd Qu.:0.7465 3rd Qu.:0.20532 3rd Qu.:0.5620   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## Tenure MonthlyCharge Bandwidth\_GB\_Year Timely.Response   
## Min. :0.0000682 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0977659 1st Qu.:0.2855 1st Qu.:0.1543 1st Qu.:0.3333   
## Median :0.5058934 Median :0.4163 Median :0.4585 Median :0.3333   
## Mean :0.4729152 Mean :0.4409 Mean :0.4627 Mean :0.4151   
## 3rd Qu.:0.8512964 3rd Qu.:0.5718 3rd Qu.:0.7750 3rd Qu.:0.5000   
## Max. :1.0000000 Max. :1.0000 Max. :0.9911 Max. :1.0000   
## Timely.Fixes Timely.Replacements Reliability Options   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3333 1st Qu.:0.2857 1st Qu.:0.3333 1st Qu.:0.3333   
## Median :0.3333 Median :0.2857 Median :0.3333 Median :0.3333   
## Mean :0.4174 Mean :0.3548 Mean :0.4160 Mean :0.4149   
## 3rd Qu.:0.5000 3rd Qu.:0.4286 3rd Qu.:0.5000 3rd Qu.:0.5000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Respectful.Response Courteous.Exchange Active.Listening  
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2857 1st Qu.:0.3333 1st Qu.:0.2857   
## Median :0.2857 Median :0.5000 Median :0.2857   
## Mean :0.3559 Mean :0.4176 Mean :0.3553   
## 3rd Qu.:0.4286 3rd Qu.:0.5000 3rd Qu.:0.4286   
## Max. :1.0000 Max. :1.0000 Max. :1.0000

summary(testDF)

## Children Age Income Outage\_sec\_perweek  
## Min. :0.000 Min. :0.0000 Min. :0.003539 Min. :0.01384   
## 1st Qu.:0.100 1st Qu.:0.2535 1st Qu.:0.077392 1st Qu.:0.38656   
## Median :0.200 Median :0.5070 Median :0.130052 Median :0.47908   
## Mean :0.212 Mean :0.5028 Mean :0.150004 Mean :0.47685   
## 3rd Qu.:0.300 3rd Qu.:0.7606 3rd Qu.:0.199811 3rd Qu.:0.56330   
## Max. :1.000 Max. :1.0000 Max. :0.676029 Max. :0.90812   
## Tenure MonthlyCharge Bandwidth\_GB\_Year Timely.Response   
## Min. :0.00000 Min. :0.0000 Min. :0.01253 Min. :0.0000   
## 1st Qu.:0.09447 1st Qu.:0.2854 1st Qu.:0.15519 1st Qu.:0.3333   
## Median :0.32335 Median :0.4044 Median :0.34370 Median :0.5000   
## Mean :0.46579 Mean :0.4400 Mean :0.45751 Mean :0.4152   
## 3rd Qu.:0.85498 3rd Qu.:0.5829 3rd Qu.:0.77943 3rd Qu.:0.5000   
## Max. :0.99993 Max. :1.0000 Max. :1.00000 Max. :0.8333   
## Timely.Fixes Timely.Replacements Reliability Options   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3333 1st Qu.:0.2857 1st Qu.:0.3333 1st Qu.:0.3333   
## Median :0.5000 Median :0.4286 Median :0.5000 Median :0.5000   
## Mean :0.4190 Mean :0.3599 Mean :0.4187 Mean :0.4212   
## 3rd Qu.:0.5000 3rd Qu.:0.4286 3rd Qu.:0.5000 3rd Qu.:0.5000   
## Max. :0.8333 Max. :0.8571 Max. :0.8333 Max. :1.0000   
## Respectful.Response Courteous.Exchange Active.Listening  
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2857 1st Qu.:0.3333 1st Qu.:0.2857   
## Median :0.4286 Median :0.5000 Median :0.4286   
## Mean :0.3644 Mean :0.4245 Mean :0.3673   
## 3rd Qu.:0.4286 3rd Qu.:0.5000 3rd Qu.:0.4286   
## Max. :0.8571 Max. :0.8333 Max. :0.8571

# Set Target and Test Dependent Variables  
targetChurn<-df[sampNumbers,1]  
testChurn<-df[-sampNumbers,1]  
  
#Build classification model  
library(class)  
churn\_pred<- knn(trainDF, testDF, cl=targetChurn)  
churn\_actual<-testChurn  
  
# Confusion Matrix  
table(churn\_pred, churn\_actual)

## churn\_actual  
## churn\_pred 0 1  
## 0 622 131  
## 1 106 141

# Model Accuracy  
mean(churn\_pred==churn\_actual)

## [1] 0.763

#Build classification model with 7k  
churn\_pred7<- knn(trainDF, testDF, cl=targetChurn, k=7)  
churn\_actual7<-testChurn  
  
# Confusion Matrix 7k  
table(churn\_pred7, churn\_actual7)

## churn\_actual7  
## churn\_pred7 0 1  
## 0 662 121  
## 1 66 151

# Model Accuracy 7k  
mean(churn\_pred7==churn\_actual7)

## [1] 0.813

#Build classification model with 15k  
churn\_pred15<- knn(trainDF, testDF, cl=targetChurn, k=15)  
churn\_actual15<-testChurn  
  
# Confusion Matrix 15k  
table(churn\_pred15, churn\_actual15)

## churn\_actual15  
## churn\_pred15 0 1  
## 0 671 126  
## 1 57 146

# Model Accuracy 15k  
mean(churn\_pred15==churn\_actual15)

## [1] 0.817

#Rerun highest accuracy model with probability parameter  
churn\_pred15p<- knn(trainDF, testDF, cl=targetChurn, k=15, prob=TRUE)  
churn\_prob<-attr(churn\_pred15p, "prob")  
head(churn\_prob)

## [1] 0.8000000 0.9333333 0.8000000 0.7333333 0.8666667 0.9333333

# Part V: Data Summary and Implications

Having found the overall accuracy to be 82%, this means that the model is correctly predicting the churn outcome (whether positive or negative churn) 82% of the time. However, as it is more important to predict when a customer will churn than when they will not, it is important to take extra steps in gauging the model’s performance beyond understanding the overall accuracy. An ROC curve based on the classification model is drawn. A poor fitting model would appear as a diagonal line, meaning that the model’s ability to distinguish between classes is 50/50 or no better than a coin flip. This ROC curve however does not follow the diagonal line. Instead it rises forming a curve with an area underneath. This area under the curve (AUC) can be measured as another metric for how well the model is performing. In this case, the AUC is .7809. This is well above the .5 AUC of an ineffective model, but still below the 1 AUC of a perfectly performing model. With an accuracy of 82% and an AUC of .7809 the model is performing well but will have some limitations. This analysis did not incorporate all of the variables in the original data set such as latitude and longitude. Also, this analysis only tried k attributes for the knn classification model of 1, 7, and 15. Other values may be tried to yield optimal results. The recommended course of action to the organization is to implement this model for regular use to identify customer likely to churn. These customer can then become part of targeted service retention initiatives, thereby decreasing the organization’s customer on-boarding costs.

library(pROC)

## Warning: package 'pROC' was built under R version 4.0.5

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

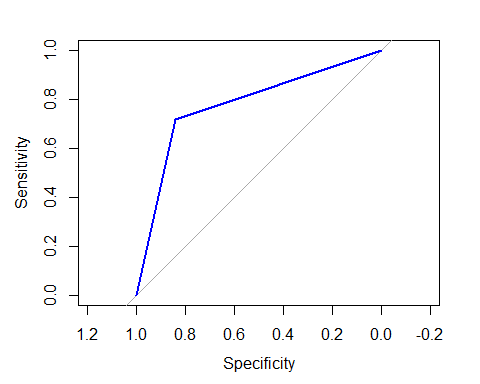
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

churn\_pred15<-as.numeric(as.character(churn\_pred15))  
  
ROC<-roc(churn\_pred15, testChurn)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

plot(ROC, col="blue")



auc(ROC)

## Area under the curve: 0.7806