Tucker Atwood
WGU MSDA
D209 Task 1
4/12/24

A1. Research Question

Using data from the "churn" data set, I will answer the question, "Can a k-nearest neighbor classification method be used to predict customer churn?"

A2. Analysis Objectives

The goal of this analysis is to gain an understanding of the relationships between various factors and customer churn. A k-nearest neighbors (KNN) method will be used to calculate the proximity of customers based on several attributes, to classify customers who are most likely to churn, and to identify which customers should be targeted for specific actions to prevent them from leaving the service.

B1. Method Justification

The k-nearest neighbors (KNN) method uses machine learning techniques to compare data points to each other using specified features, aiming to assign unknown values to one of many groups. For example, in this analysis, observations will be split into two groups: "Has churned" and "Has not churned." KNN defines these groups using the proximity of observations to each other; in this analysis, the Euclidean distance (shown below) will be used to measure the distance between data points in several factors, and then group them accordingly.

The data set will be split into a training set and a test set. The training set will be used to identify common characteristics for each group, while the test set will be used to assess the accuracy of the KNN method: the ability to accurately classify customers as "Has churned" or "Has not churned." For each observation in the test set, the method will locate the observation's position using the values of its various selected features, compare the observation to a pre-defined number of similar values (neighbors) from the training set, and assign the observation to one of the two groups. The number of neighbors will be 85 (this value will be explained in section D2) and a **majority voting algorithm** will be used to predict the labels of observations in the test set. This means that, for each observation in the test set, the labels of the nearest 85 neighbors (as determined by Euclidean distance) from the training set will be used to vote on the predicted label for the test set observation.

The **expected outcome** is to create a KNN classification method that will accurately classify customers by their churn status, which can be used to target specific customers at risk of churning.

Euclidean distance. n: number of variables. x and y: individual values of a variable.

$$Distance = \sqrt{\sum_{i=1}^{n}(x_n - y_n)^2}$$

B2. Method Assumption

The KNN method assumes that **observations that exist in the same proximity are inherently similar to each other**, while observations with a greater distance between them are inherently dissimilar. Since the method uses distance to compare points, the presence of common attributes must be able to accurately inform the classification process. For this analysis, this assumption can be deemed accurate; customers with similar attributes are inherently similar to each other and can be classified accordingly.

B3. Packages Used

This research question will be answered using the programming language R in the RStudio environment. Within R, the following packages will be utilized:

- **plyr** and **dplyr**: These packages will be used to perform data frame manipulation tasks, such as revaluing observations to make the KNN method more accurate.
- **naniar**: This package will be used to check for missing (NA) values within the initial data set. These values must be checked and treated to make the KNN method valid.
- **fastDummies**: This package will be used to perform a one-hot encoding process on the necessary categorical data, creating dummy variables for nominal factors. This is essential so that these factors may be included in the KNN comparison method.
- **class**: This package will be used to perform the k-nearest neighbors method, which is the basis for this analysis.
- **pROC**: This package will be used to measure the performance of the KNN method, such as creating an ROC (receiver operating characteristic) curve and calculating the AUC (area under the curve).

C1. Preprocessing Goal

To ensure the KNN method will be as accurate and valid as possible, the data must be transformed. The **goal for this transformation is to include only variables relevant to Churn status, and for all of these independent variables to be set on a scale from 0 to 1.**

This goal is relevant to the KNN classification method because Euclidean distance will be used to measure the proximity of observations to each other. Therefore, all variables must be relevant to comparing Churn status, and distances must all be represented on the same scale to ensure accurate

results. **All categorical variables except for Churn must be re-expressed numerically** using 0's and 1's, and **all numeric variables must be re-expressed on a scale from 0 to 1** so that differences in distance are uniform and balanced.

This goal will be addressed in section C3.

C2. Variables Used

The following independent variables have been determined to be relevant to the classification of customer churn status and will be used in the KNN method:

- **Churn**, a categorical variable that represents whether or not a customer has left the service (Yes or No).
- **Tenure**, a numeric variable that represents the number of months a customer has been with the service.
- **MonthlyCharge**, a numeric variable that represents the average amount of money charged to the customer's account per month.
- **Techie**, a categorical variable that represents whether or not the customer identifies as technologically inclined (Yes or No).
- **Multiple**, a categorical variable that represents whether or not the customer has more than one phone line (Yes or No).
- **OnlineSecurity**, a categorical variable that represents whether or not the customer has signed up for an online security add-on program (Yes or No).
- **TechSupport**, a categorical variable that represents whether or not the customer has signed up for a technical support add-on program (Yes or No).
- **Contract**, a categorical variable that represents the customer's contract type (Month-to-Month, One-year, or Two-year).
- **InternetService**, a categorical variable that represents the customer's internet service provider (DSL, Fiber Optic, or None).

C3. Data Preparation

To prepare the data for analysis, the following steps were performed:

- **Data cleaning**
    - **Full duplicates**, defined as observations for which every variable is a match with another observation, were detected. Zero full duplicates were found.
    - **Partial duplicates**, defined as observations for which a subset of variables match another observation, were detected by searching for matches on the "Customer_id" variable. Zero partial duplicates were found.
    - **Missing values**, defined as entries of "NA" in any variable, were detected. Zero missing values were found.

- **Outliers**, defined as values significantly higher or lower than other established values within the variable, were detected for the relevant numeric variables: Tenure and MonthlyCharge. Zero outliers were found for each variable.

The following code executes the data cleaning process as described. An executable version of this code can be found in the attached file: Atwood_D209_Task1_Code.R.

```
sum(duplicated(churn)) # checks for full duplicates (0)

library(plyr) # using plyr package
library(dplyr) # using dplyr package

churn %>%
  count(Customer_id) %>%
  filter(n > 1) # checks for partial duplicates with matching Customer_id (0)

library(naniar) # using naniar package
n_miss(churn) # total missing values (0)

hist(churn$Tenure) # visualization of Tenure data; bimodal
boxplot(churn$Tenure) # no outliers present
ten_outliers <- churn %>%
  filter((Tenure < quantile(churn$Tenure, 0.25, na.rm = TRUE) - IQR(churn$Tenure, na.rm = TRUE) * 1.5)
         | (Tenure > quantile(churn$Tenure, 0.75, na.rm = TRUE) + IQR(churn$Tenure, na.rm = TRUE) * 1.5))
# find outliers using IQR method
count(ten_outliers) # confirms zero Tenure outliers

hist(churn$MonthlyCharge) # visualization of MonthlyCharge data; normal distribution
boxplot(churn$MonthlyCharge) # no outliers present
mon_outliers <- churn %>%
  mutate(mon_z = scale(churn$MonthlyCharge)) %>%
  filter(mon_z > 3 | mon_z < -3)  # find outliers using z-score method
count(mon_outliers) # confirms zero MonthlyCharge outliers
```

- **Re-expression of categorical variables**
  - The variables Techie, Multiple, OnlineSecurity, and TechSupport were converted to numeric by replacing all "No" responses with "0" and "Yes" responses with "1."
  - Note: the variable Churn was not converted in the same manner, since its unique values of "Yes" and "No" will be used as the groups in the KNN method.

- The variables Contract and InternetService were converted to numeric by using a one-hot encoding process to create dummy variables for each unique value in each variable. This created six new variables: Contract_One_Year, Contract_Two_Year, Contract_Month_to_Month, InternetService_DSL, InternetService_Fiber_Optic, and InternetService_None.
- Note: a k-1 method was **not** used to create these dummy variables; the KNN method does not require this, since it does not create linear combinations of all independent variables. This information was obtained from the following source: https://www.bzst.com/2015/08/categorical-predictors-how-many-dummies.html

The following code executes the re-expression of categorical variables process as described. An executable version of this code can be found in the attached file: Atwood_D209_Task1_Code.R.

```
churn$Techie <- as.numeric(revalue(churn$Techie, replace = c("No" = 0, "Yes" = 1)))
churn$Multiple <- as.numeric(revalue(churn$Multiple, replace = c("No" = 0, "Yes" = 1)))
churn$OnlineSecurity <- as.numeric(revalue(churn$OnlineSecurity, replace = c("No" = 0, "Yes"
= 1)))
churn$TechSupport <- as.numeric(revalue(churn$TechSupport, replace = c("No" = 0, "Yes" = 1)))
# re-expresses necessary categorical binary data as numeric, 0 for No, 1 for Yes

library(fastDummies) # using fastDummies package
churn <- churn %>%
  dummy_cols("Contract") %>%
  rename(Contract_One_Year = 'Contract_One year') %>%
  rename(Contract_Two_Year = 'Contract_Two Year') %>%
  rename(Contract_Month_to_Month = 'Contract_Month-to-month')

churn <- churn %>%
  dummy_cols("InternetService") %>%
  rename(InternetService_Fiber_Optic = 'InternetService_Fiber Optic')
# re-expresses necessary categorical non-binary data as numeric using one-hot encoding
```

- **Scaling data**
  - The numeric variables Tenure and MonthlyCharge were **normalized** by setting them on a scale from 0 to 1. This allows differences in distances to be put on the same scale, which allows them to be more easily compared and creates more accurate results for the KNN process.
  - **Normalization was chosen over standardization** because at least one of the variables (Tenure) does not appear to follow a Gaussian distribution, and so that all variables can be expressed and compared on the same scale (0 to 1).

The following code executes the numeric variable scaling process as described. An executable version of this code can be found in the attached file: Atwood_D209_Task1_Code.R.

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x))) }
# creates normalizing function

churn$Tenure <- normalize(churn$Tenure)
churn$MonthlyCharge <- normalize(churn$MonthlyCharge)
# scales necessary variables
```

- **Selection of relevant variables**
    - The variables from the original "churn" data set were reduced to include only those relevant to the KNN classification method. The remaining variables in the prepared data set are:
        - Churn
        - Tenure
        - MonthlyCharge
        - Techie
        - Multiple
        - OnlineSecurity
        - TechSupport
        - Contract_One_Year
        - Contract_Two_Year
        - Contract_Month_to_Month
        - InternetService_DSL
        - InternetService_Fiber_Optic
        - InternetService_None
    - All of these variables, except for Churn, are expressed on a scale from 0 to 1 in the prepared data set.

The following code executes the variable selection process as described. An executable version of this code can be found in the attached file: Atwood_D209_Task1_Code.R.

```
churn <- churn %>%
  select(Churn, Tenure, MonthlyCharge, Techie, Multiple, OnlineSecurity, TechSupport,
         Contract_One_Year, Contract_Two_Year, Contract_Month_to_Month,
         InternetService_DSL, InternetService_Fiber_Optic, InternetService_None)
# selecting only variables that will be used in the analysis
```

C4. Prepared Data Set

The resulting cleaned and prepared data set has been written into a CSV file and attached with the following name: churn_209_task1.csv.

D1. Data Split

The initial "churn" data set, as well as the referenced prepared data set, contains 10,000 customers and their observed characteristics. To perform a KNN classification method, the data must be split into two subsets: a **training data set**, which will be used to establish KNN classification based on attributes and distances between observations; and a **test data set**, which will be used to evaluate the performance of the KNN classification.

A **random 70/30 split** was utilized to create the two subsets, resulting in a training set consisting of 7000 random customers from the churn data set and a test set consisting of the remaining 3000 customers from the churn data set.

The following code executes the data splitting process as described. An executable version of this code can be found in the attached file: Atwood_D209_Task1_Code.R.

```
set.seed(1)
rand_churn <- churn[sample(10000),]
# randomizes churn rows

churn_train <- rand_churn[1:7000,]
churn_test <- rand_churn[7001:10000,]
# creates 70-30 split for training and test data
```

The resulting training data set has been written into a CSV file and attached with the following name: train_209_task1.csv.

The resulting test data set has been written into a CSV file and attached with the following name: test_209_task1.csv.

D2. Classification Procedure and Technique

The process of the KNN classification method can be summarized as follows:

- The training set of 7000 randomized observations from the churn data set was used to establish relationships among customer factors by measuring Euclidean distances between the relevant factors: Tenure, MonthlyCharge, Techie, Multiple, OnlineSecurity,

TechSupport, Contract_One_Year, Contract_Two_Year, Contract_Month_to_Month, InternetService_DSL, InternetService_Fiber_Optic, and InternetService_None.

- Along with their positions relative to each other, all observations in the training set were labeled by their Churn value, creating two distinct groups: "Has churned" and "Has not churned."
- The remaining 3000 values in the churn data set were used as the test set. For each of these values, the KNN method used Euclidean distance to find the nearest 85 values in the training set and their Churn labels.
- 85 represents the "k" value in the KNN technique, and was selected because it is the approximate square root of the number of observations in the training set (7000). Note: this value was rounded to the nearest odd value to avoid ties in the following classification. The square root rule of thumb was obtained from the following source: https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb
- For each observation in the test set, the labels of the nearest 85 neighbors were used to predict the observation's label. For example, if 44 of the nearest neighbors in the training set were labeled as "Has churned" and 41 were labeled as "Has not churned," the point would be predicted to have the label "Has churned."
- There were no intermediate calculations required to perform this analysis.
- The following intermediate calculations were made to measure the performance of the analysis:
    - A confusion matrix was created to compare the actual Churn values to the predicted Churn values.
    - The Accuracy, Sensitivity, and Specificity values of the confusion matrix were calculated.
    - Each of these values is explained further in Section E1.
    - Screenshots of the inputs and outputs of these calculations are as follows:

Input:

```
conf <- table(churn_actual, churn_predicted)
acc <- (conf[1,1] + conf[2,2])/(conf[1,1] + conf[1,2] + conf[2,1] + conf[2,2])
sens <- (conf[2,2]/(conf[2,2] + conf[1,2]))
spec <- (conf[1,1]/(conf[1,1] + conf[2,1]))
```

Output:

```
> conf
             churn_predicted
churn_actual   No  Yes
         No  2077  101
         Yes  406  416
> acc
[1] 0.831
> sens
[1] 0.8046422
> spec
[1] 0.8364881
```

D3. Classification Code

The following code executes the KNN classification method as described. An executable version of this code can be found in the attached file: Atwood_D209_Task1_Code.R.

```
library(class) # using class package

churn_actual <- rand_churn[7001:10000,]$Churn
churn_predicted <- knn(train = churn_train[-1], test = churn_test[-1],
                       cl = rand_churn[1:7000,]$Churn, k = 85, prob = TRUE)
# creates actual and predicted churn values for test data

conf <- table(churn_actual, churn_predicted)
acc <- (conf[1,1] + conf[2,2])/(conf[1,1] + conf[1,2] + conf[2,1] + conf[2,2])
sens <- (conf[2,2]/(conf[2,2] + conf[1,2]))
spec <- (conf[1,1]/(conf[1,1] + conf[2,1]))
# creates table and calculates accuracy, sensitivity, and specificity for churn predictions using KNN

conf
acc
sens
spec
# displays the table and each of the calculated values

library(pROC) # using pROC package

churn_prob <- attr(churn_predicted, "prob")
ROC <- roc(churn_actual, churn_prob)
plot(ROC, col = 2)
auc(ROC)
# visualizes ROC curve and calculates area under the curve (AUC)
```

E1. Accuracy and Area Under Curve

The following table represents a confusion matrix of all customers in the test data set by comparing their "actual" Churn status to their "predicted" Churn status from the KNN method.

	churn_predicted	
churn_actual	No	Yes
No	2077	101
Yes	406	416

The following metrics were calculated and assessed from the confusion matrix.

- **Accuracy** represents the proportion of all predictions, Yes and No, that were correct. This is calculated as follows:

$$\frac{2077 + 416}{2077 + 416 + 406 + 101} = \mathbf{0.831}$$

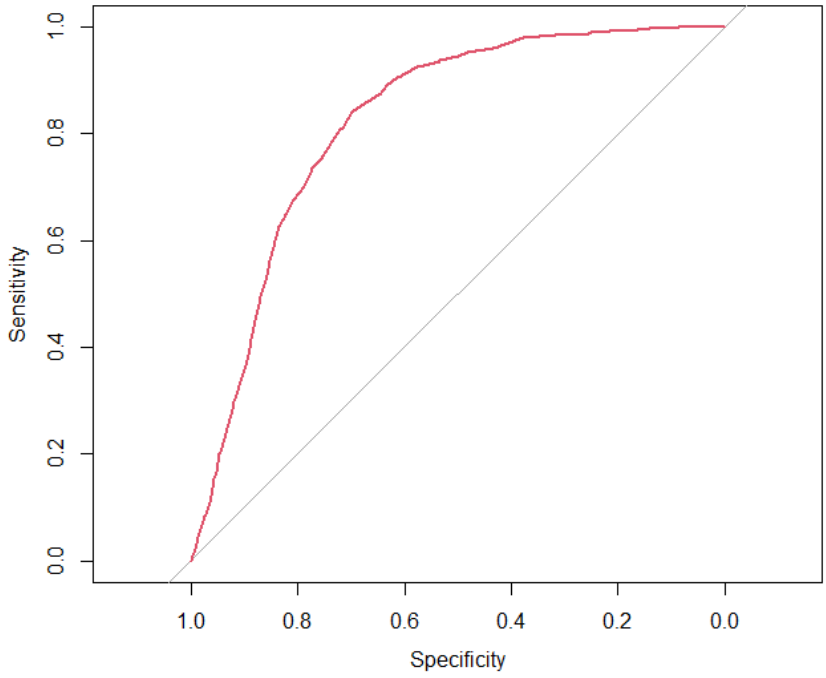
- The Accuracy value shows that 83.1% of all observations in the test set were correctly predicted to be labeled "Has churned" or "Has not churned."
- **Sensitivity** represents the proportion of true positives: predictions of "Yes" which were correct. This is calculated as follows:

$$\frac{416}{416 + 101} \approx \mathbf{0.805}$$

- The Sensitivity value shows that 80.5% of all observations in the test set that were predicted to be labeled "Has churned" were correct.
- **Specificity** represents the proportion of true negatives: predictions of "No" which were correct. This is calculated as follows:

$$\frac{2077}{2077 + 406} \approx \mathbf{0.836}$$

- The Specificity value shows that 83.6% of all observations in the test set that were predicted to be labeled "Has not churned" were correct.
- The **ROC (receiver operating characteristic) curve** plots specificity and sensitivity at every possible threshold, creating a visualization of the tradeoff between true positives and true negatives as one increases and the other decreases. The ROC curve for the application of this KNN method can be visualized as follows:



- The **area under the curve (AUC)** is used to measure the effectiveness of the classification method. An AUC value of 0.5 represents the area under the diagonal line above, where specificity and sensitivity are always equal. A classification close to this value has not predicted labels any better than a random selection would be expected to. An AUC value of 1.0 would represent a perfect classification for all labels.
- The AUC value for this classification method was determined to be **0.819**. This indicates that the KNN method was competent at distinguishing between the two indicated labels.
- **Overall, the KNN method performed adequately in predicting Churn labels in the test data set.** The method was better at predicting true negatives than true positives, but was able to predict at least 80% of each. The Accuracy value of 0.831 and the AUC value of 0.819 both indicate that the KNN method was useful in classifying customers' churn labels, though there is some room for improvement.

E2. Results and Implications

The k-nearest neighbors (KNN) classification method was able to accurately predict customer churn for approximately 83% of observations in the test data set. The sensitivity of the model was approximately 81%, the specificity was approximately 84%, and the area under the ROC curve was 0.819. The Accuracy value indicates the **KNN method can be deemed useful in predicting churn classification**, and the other metrics indicate that the method's ability to predict true positives and true negatives was **sufficiently balanced**.

These results imply that KNN classification should be used to predict and prevent customer churn. The selected features represent characteristics of customers that can be used to make predictions regarding each customer's risk of leaving the service. For current customers with similar features to previous customers who have churned, informed decisions can be made to target customers at high risk of leaving the service and to try to prevent them from doing so.

E3. Limitations

There are potential **limitations** to the processes and results of this KNN classification method:

- Only a subset of variables was selected to be included in the analysis. These variables were selected due to their perceived practical connection to customer churn; however, it is possible for one or more other unselected variables in the initial dataset to have a significant impact on churn classification. Including other variables in future KNN classifications may increase the Accuracy, AUC, and overall performance of the method.
- The value of k was determined based on the square root of the number of observations in the training data set. This decision was made based on the general rule of thumb for selecting the optimal k value. However, different values of k may have created a better-performing model.

E4. Recommended Course of Action

Based on the given results, it was determined that customer churn can be usefully predicted using a k-nearest neighbors (KNN) classification method. It is recommended to use the factors applied to this method, potentially along with other factors if they increase the performance of this model, to determine customers who are at high risk of leaving the service. Customers who share common characteristics in these factors with previous customers who have left the service should be targeted specifically in an attempt to prevent them from leaving.

Potential actions include, but are not limited to: offering exclusive deals to these at-risk customers, surveying them regarding their satisfaction with the service, and analyzing other services that may offer better incentives to these customers in particular.

F. Panopto Video

Please refer to the link attached with a Panopto video recording. The video includes an explanation, execution, and output results of the referred code used to perform this analysis.

G. Third-Party Code References

WGU Courseware was used as a resource to learn the methods, concepts, and functions used to create the codes in this project, including DataCamp course tracks (datacamp.com), Dr. Festus Elleh's D209 PowerPoint presentation, and the book *Data Science Using Python and R* by Chantal D. Larose and Daniel T. Larose.

H. Content References

The general understanding and reasoning for not using k-1 groups for dummy variables was obtained with assistance from the following resource:
https://www.bzst.com/2015/08/categorical-predictors-how-many-dummies.html

The rule of thumb for using the square root of the number of observations in the training data set as the optimal value for k was obtained with assistance from the following resource:
https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb