Using k-nearest neighbors, is it possible to predict the customers who will churn?

By:

Kevin Sandoval

**A1.**

The research questions I will be examining is: “Using k-nearest neighbors, is it possible to predict the customers who will churn? This question has important implications for a business as retaining customers is a vital part of the company’s success. Examining the variables that might help determine a given customer is likely to churn could either aide in retaining the customer or help the company focus more on other customers as acquiring new customers is much more expensive than retaining old customers.

**A2.**

The goal(s) of the analysis will be to create a machine learning model that can help the company determine if a customer is likely to churn and then help recommend a course of action that the company can take to retain these customers.

**B1.**

Per Dr. Elleh’s video lecture ‘D209 Task 1: Expectations and Data Preprocessing – Python’, the k-nearest neighbor method looks at the ‘k’ nearest data points to make a determining choice in if the customer has churned. For example, if k is set to 5, the machine learning model will look at the 5 customers that are most similar to the test customer. This is the ‘nearest neighbors’ method and will assign the test customer the outcome which appears the most. So in the 5 customers, if 3 of the nearest neighbors churned, while 2 did not, the test customer would still be read as likely to churn.

**B2.**

The k-nearest neighbor classification method relies on the assumption that similar things exist in proximity to each other. What this assumption is referencing is the fact that to draw a logical conclusion about the test customer, we must assume that similar customers in terms of the variables will be nearby one another, thus allowing us to classify them similarly.

**B3.**

The Python packages I used are explained and shown below.

**Numpy:** Helps preform numerical calculations on arrays

**Pandas:** Allows me to import and export csv files

**Matplotlib.pyplot:** Helpful for visualizing distributions and plots

**Missingno:** Helpful for checking if there are missing values in the dataset

The following are imported from sklearn.model\_selection

**Train\_test\_split:** Allows me to easily split the data into train and test sets

**GridSearchCV:** Helpful for finding the best number of nearest neighbors

The following are imported from sklearn

**Metrics:** Useful for helping build the confusion matrix

**Preprocessing:** Helps with standardizing the numerical variables

The following are imported from sklearn.metrics

**ConfusionMatrixDisplay:** Optional, but I believe it helped visualize the confusion matrix and made it easier to interpret

**Confusion\_matrix:** Useful for creating the confusion matrix to determine false positive and true positive rates

**Roc\_curve:** Made plotting the ROC curve easy

**Roc\_auc\_score:** Helpful in finding the AUC score for the model

**Classification\_report:** Allows me to see metrics for how the model preformed

**KNeighborsClassifier (**from sklearn.neighbors): Helpful in creating the actual model for kNN

**Warnings:** Helpful for ignoring any warnings that pop up when cells are ran

**C1.**

The first thing that I did when I pulled in the data set was to check the shape of the data frame. The shape was (10000 , 50) so I knew that there were 10,000 rows of data with 50 columns. I then checked if there were any duplicate values and got that there were no duplicates.

After that I checked if there were any missing values using the .isna().sum() function to determine if there were any missing values. I found that the data set did not contain any missing values so none were treated. I then decided to check every quantitative variable for outliers. I did find outliers for the columns “Population”, “Children”, “Income”, “Outage\_sec\_perweek”, “Email”, “Contacts”, and “Yearly\_equip\_failure.” I treated the outliers by excluding them to a new variable for each except for “Population” which I decided to retain. After excluding the initial outliers, “Income” and “Outage\_sec\_perweek” still showed that they had a few new outliers, but I decided to retain those as it would be diminishing returns on the exclusion of data. My new data frame shape is (9079, 50).

I then changed all “Yes/No” response variables to 1/0 respectively. I also used one hot encoding to create dummy variables for a few of the variables that had multiple response types. This made it easy to turn every variable into 1/0 or numeric. Per Dr. Elleh’s powerpoint ‘D209 Data Mining 1 Task 1 Cohort’ on Slide 26, I also decided to standardize my data so the distance measure of k-NN would be more appropriate.

**C2.**

All variables used in the analysis are listed below with their classification as well.

**Area:** Categorical

**Children:** Numeric

**Age:** Numeric

**Income:** Numeric

**Gender:** Categorical

**Outage\_sec\_perweek:** Numeric

**Email:** Numeric

**Contacts:** Numeric

**Yearly\_equip\_failure:** Numeric

**Techie:** Categorical

**Contract:** Categorical

**Port\_modem:**

**Tablet:** Categorical

**InternetService:** Categorical

**Phone:** Categorical

**Multiple:** Categorical

**OnlineSecurity:** Categorical

**OnlineBackup:** Categorical

**DeviceProtection:** Categorical

**TechSupport:** Categorical

**StreamingTV:** Categorical

**StreamingMovies:** Categorical

**PaperlessBilling:** Categorical

**MonthlyCharge:** Numeric

**Bandwidth\_GB\_Year:** Numeric

**C3.**

**Step 1:** Checking for duplicates

The first step I performed when preparing the data was to check for duplicate values in the data. This is important as duplicate values can have a misleading effect on the outcome of the analysis. The code snippet and output is shown below.

A screenshot of a computer program

Description automatically generated

**Step 2:** Checking for missing values

The 2nd step of preparing the data was to check for missing values in the data. It is important to check as missing values can skew the data and the results. I used the missingno matrix to check for missing values as well as na.sum() to check. The code and some of the output is shown below.

A screenshot of a computer

Description automatically generated

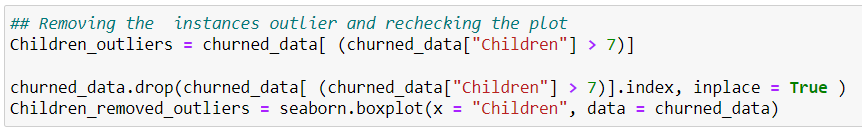
A screenshot of a computer

Description automatically generated

**Step 3:** Removing outliers

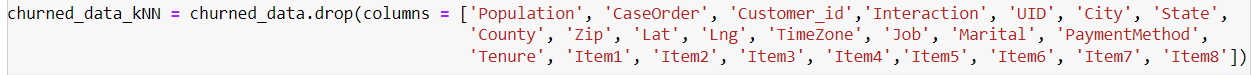
The third step in treating the data is to remove any outliers. I first checked the boxplot of each quantitative variable and if they showed outliers, I treated them. The code segments for the boxplot and outlier removal is shown below with “Children” as the example variable.





**Step 4:** Dropping unneeded columns

In this step I dropped all columns that will not be used for the analysis. The code segment is shown below.



**Step 5:** Creating dummy variables

In this stepped I created dummy variables for all of the variables that had more than 2 responses. In this case, all of the variables I chose for the analysis had 3 different responses so I created a for loop to create the dummy variables. The code segment is shown below.

A computer code with text

Description automatically generated with medium confidence

**Step 6:** Changing Yes/No responses

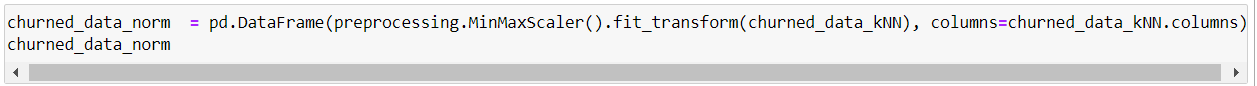
In this step, I changed all of the “Yes”/”No” responses to 1/0s respectively. This makes the analysis much easier to perform. The code segment is shown below.

A white background with text

Description automatically generated

**Step 7:** Standardizing the data

In the last step, I standardized the data. This took age from a range of 18-89 and changed it to 0-1. So 18 would show as 0 and 89 would show as 1. 43 would show as 0.352 for example. The code is shown below.



**C4.**

Prepared CSV is included in the submission.

**D1.**

The files for the training and test sets are provided in the submission.

**D2.**

The first analysis technique I used when creating the model was to create a base model and use the GridSearchCV function with grid.fit to find the optimal parameters for the model. After getting the best parameters for the model, I then created a new model while using those metrics. I also created a confusion matrix to help visualize the effectiveness of the model as well as a classification report. Output shown below.

A chart of different colored squares

Description automatically generated

A screenshot of a graph

Description automatically generated

The other technique I used was to find the accuracy, sensitivity, and specificity of the model. This helps with determining the effectiveness of the model.

**D3.**

The code for all the analysis listed in D2 is shown in screenshots below. The first screenshot is the grid search. Second screenshot is the model creation with the new best parameters. And the third screenshot is of the accuracy, sensitivity, and specificity.

A screenshot of a computer code

Description automatically generated

A computer code with black text

Description automatically generated

A screenshot of a computer code

Description automatically generated

**E1.**

The accuracy of the model as seen in the screenshots in D3 is 0.82929 or 82.929%. This is clearly lower than the 90% threshold that would imply a good fit for the data. The AUC of the model is 0.87588 or 87.588%. This shows that the model has 87.588% of predictions correct. This is a decent fit for the data as it is close to the 90% recommended threshold, but it is not quite at that limit. Since 50% AUC would be a completely random model, we can see that the model is moderately effective at predicting.

**E2.**

Based on the results of the analysis, I can conclude that the model is average at classification and prediction. The accuracy score of classifications being roughly 83% is below the preferred threshold of 90%. The AUC having a prediction accuracy of roughly 88% is closer to the preferred level but still below. From the confusion matrix we can see that the model tends to try and classify as “0” or not churn. From the test set, we can see the model predicted 2,283 “0” to only 441 “1”. That is an 83.8% not churn rate, compared to the true outcome of 2,000 “0” and 724 “1” which is a 73.4% not churn rate. This discrepancy in the model shows that the model prefers to classify as a not churn. We can directly see the impact of that when looking at the F1-score from the classification report. When predicting “0”, the F1-score is 0.89. That is an acceptable score for the prediction but the F1-score for predicting “1” is only 0.60. That is far below the acceptable threshold. The implication of this for the company would be that the model can be effective at predicting customers that will stay, but ineffective at predicting customers who will leave. I think that this is to be expected. The variability in why a customer may leave is large and it makes predicting that difficult. Although the model is not great at predicting when people may leave, it is still better than the random chance of 50/50 so there could be some utility anyway.

**E3.**

One limitation of the analysis is the variables that are chosen. I chose all of the quantitative variables, all of the “Yes”/”No” response variables, and a few of the variables with multiple responses. I did not analyze the Item1-8 variables which checks how people feel about certain questions. This could prove to be more impactful to the analysis than other variables.

**E4.**

To answer the question posed in A1, I would say that this model is partially successful at effectively predict which customers will churn. From the data and conclusions I drew in E2, the model struggles to predict correctly “1” (“Yes”) responses. While it is better than a random chance, it is not near the preferred threshold to be useful. However, the model is considerably better when predicting “0” (“No”) responses. I think that the recommended course of action for the company would be to either try a new model with some additional variables to try and more accurately predict the 1 responses, or accept that a 60% accurate rate of prediction is better than random chance and use the model to primarily check if a customer will stay, rather than leave. I think knowing that a customer is likely to stay can allow the company to shift focus onto the customers that are showing more likeliness to leave.

**F:**

Panopto video provided in the submission.

**G/H:**

Dr. Elleh, F (n.d). *D209 Data Mining 1 Task 1 Cohort Powerpoint.* Retrieved September 25th, 2024,From D209 Announcements

Dr. Elleh, F (n.d). *D209 Webinar: Task 1 Expectations and Data Preprocessing - Python.* Retrieved September 27th, 2024,From D209 Announcements

Dr. Elleh, F (n.d). *D209 Task 1 Splitting the Data and Creating the model - Python.* Retrieved September 27th, 2024,From D209 Announcements

Dr. Elleh, F (n.d). *D209 T1 Building KNN in Python Med.* Retrieved September 27th, 2024,From D209 Announcements

Scikit-learn. (n.d). *ConfusionMatrixDisplay.* Retrieved September 13th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html>