**Part I: Research Question**

“Given the dataset of our customer’s individual features and disregarding customer demographics, how well does KNN accurately predict whether a customer churns on unseen data?”

One Data Analysis Goal:

1. The final model should have an AUC above at least 90% and an Accuracy on the unseen test data above at least 85%.

**Part II: Method Justification**

In KNN classification models, the algorithm uses the k-nearest neighbors, or an arbitrary number we can designate, to predict an observation’s classification. When there is a new observation, the algorithm uses k number of nearest neighbors around that observation to classify the new observation. For example, we will assign k = 5. The algorithm will use the 5 nearest observations’ classifications to this new observation and predict the new observation’s classification. There is no one-size-fits-all k value. It is common practice to test several k values in an analysis which can be computationally expensive. For my analysis, I tested values from 1 to 10 for the k parameter. I also am using hyperparameter tuning and cross-validation.

Also, my KNN classification model takes it a step further by adding in hyperparameter tuning and Cross-Validation. For my hyperparameter tuning, I chose to tune the parameter of k from 1 to 10. For Cross-Validation, I chose to split the training dataset into 5 randomly sampled, equally sized parts. As a result, the KNN classification model is trained on each of the 5 parts of the data and tests each of the 10 k parameters to obtain the highest ROC of the parameters.

One Assumption of KNN:

1. KNN is non-parametric and makes no assumptions of the underlying data distributions.

Libraries used:

**caret:** This library was used to train and construction the KNN classification model. It was also used to create the confusion matrix when evaluating how accurate our model was on new, unseen test data.

**fastDummies:** This library was used to transform our categorical variables into binary dummy variables for the knn algorithm. After they are transformed, I can normalize the entire dataset.

**caTools:** This library was used to evaluate and plot our KNN classification model’s AUC.

**Part III: Data Preparation**

Data Preprocessing Goals:

1. Normalize the dataset by using preProcess(). I chose to normalize my dataset as there are many categorical variables I am transforming into binary dummy variables. I want each variable, regardless of continuous or categorical, to be on the same scale and have the same level potential of influence on the KNN classification model. For the continuous variables, there are different ranges presented. It would be best to normalize the variables.
2. Transform categorical variables into binary dummy variables by using fastDummies::dummy\_cols().

|  |  |
| --- | --- |
| **Variable Name** | **Type** |
| Churn | Categorical |
| Outage\_sec\_perweek | Continuous |
| Email | Continuous |
| Contacts | Continuous |
| Yearly\_equip\_failure | Continuous |
| Techie | Categorical |
| Contract | Categorical |
| Port\_modem | Categorical |
| Tablet | Categorical |
| InternetService | Categorical |
| Phone | Categorical |
| Multiple | Categorical |
| OnlineSecurity | Categorical |
| OnlineBackup | Categorical |
| DeviceProtection | Categorical |
| TechSupport | Categorical |
| StreamingTV | Categorical |
| StreamingMovies | Categorical |
| PaperlessBilling | Categorical |
| PaymentMethod | Categorical |
| Tenure | Continuous |
| MonthlyCharge | Continuous |
| Bandwidth\_GB\_Year | Continuous |
| Item1 | Continuous |
| Item2 | Continuous |
| Item3 | Continuous |
| Item4 | Continuous |
| Item5 | Continuous |
| Item6 | Continuous |
| Item7 | Continuous |
| Item8 | Continuous |

Data Prep Steps:

1. Import the raw dataset
2. Remove customer demographics by indexing
3. Transform categorical variables into binary dummy variables using fastDummies::dummy\_cols()
4. Normalize the dataset using preProcess()
5. Set seed for random sampling of data
6. Create the index for the random sampling of data. This is for the splitting of the data into train and test datasets.
7. Split the data into train and test datasets using the index created in step 6.

Please see “prepped\_dataset.csv” for the cleaned, prepped dataset.

**Part IV: Analysis**

Please see “train\_dataset.csv” and “test\_dataset.csv” for the training and testing datasets.

In KNN classification models, the algorithm uses the k-nearest neighbors, or an arbitrary number we can designate, to predict an observation’s classification. When there is a new observation, the algorithm uses k number of nearest neighbors around that observation to classify the new observation. For example, we will assign k = 5. The algorithm will use the 5 nearest observations’ classifications to this new observation and predict the new observation’s classification. There is no one-size-fits-all k value. It is common practice to test several k values in an analysis which can be computationally expensive. For my analysis, I tested values from 1 to 10 for the k parameter by using hyperparameter tuning.

Also, my KNN classification model takes it a step further by adding in Cross-Validation. For my hyperparameter tuning, I chose to tune the parameter of k from 1 to 10. For Cross-Validation, I chose to split the training dataset into 5 randomly sampled, equally sized parts. As a result, the KNN classification model is trained on each of the 5 parts of the data and tests each of the 10 k parameters to obtain the highest ROC of the parameters.

To evaluate my model’s performance, I used two metrics: Accuracy and ROC AUC. For Accuracy evaluation, I used the predict() method on unseen test data and created a confusion matrix using caret::confusionMatrix(). The resulting accuracy was 0.97 which is extremely accurate for unseen data and satisfies my initial data analysis goal. To further evaluate this model against unseen test data, I plotted the AUC of our final model using caTools::colAUC. The results were an AUC of 0.996 which satisfies my initial goal of above 90%. With 1 being a perfect and always correct model, my KNN model is extremely close. Since this is a classification model and problem, I cannot use MSE to evaluate our model. MSE is used in regression models/problems to evaluate the models.

Please see “Task 1 – All Code.R” for the annotated code for classification analysis.

**Part V: Data Summary and Implications**

To evaluate my model’s performance, I used two metrics: Accuracy and ROC AUC. For Accuracy evaluation, I used the predict() method on unseen test data and created a confusion matrix using caret::confusionMatrix(). The resulting accuracy was 0.97 which is extremely accurate for unseen data and satisfies my initial data analysis goal. Accuracy shows the percentage of correctly predicted classifications by my KNN model.

To further evaluate this model against unseen test data, I plotted the AUC of our final model using caTools::colAUC. The results were an AUC of 0.996 which satisfies my initial goal of above 90%. With 1 being a perfect and always correct model, my KNN model is extremely close. Since this is a classification model and problem, I cannot use MSE to evaluate our model. MSE is used in regression models/problems to evaluate the models.

My model produced extremely strong results. The Accuracy and AUC similarities indicate that the model is performing well on unseen data without over or underfitting.

One limitation of my data analysis is that my KNN classification model is currently slow and it would be ineffective in real-time situations where a much larger amount of k values being tested. These results are also only for 1 random seed which can vary depending on the seed set. To minimize these potentially varying results, I used cross-validation to split the data into multiple folds for analysis.

Please see all PNG files for the metric’s code and their outputs.

Recommendation:

1. I recommend putting this model into production, but re-training and testing it on new data in a few months. This model should be used to predict the probability of a customer churning. If a customer has a higher probability, then it is recommended to give the customer more incentives such as a lower monthly charge and access to new features to retain assist in their retention.

**Part VI: Demonstration**

Please see my Panopto Presentation.

**I did not use any outside sources.**