Using random forests, is it possible to predict the tenure length of customers?

By:

Kevin Sandoval

**A1.**

The research questions I will be examining is: “Using random forests, is it possible to predict the tenure length of customers? 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 the length a customer is likely to stay with the company could aide in prioritizing retaining those customers.

**A2.**

The goal(s) of the analysis will be to create a machine learning model and use the random forest technique to help the company determine the length of tenure of a customer, and then help recommend a course of action that the company can take to retain these customers.

**B1.**

Random forests (RF) are a machine learning method for classification. It takes multiple decision trees during the training phase of the model and gives the mean of the predictions for the regression. Since each decision tree makes it’s own prediction, the predictions are averaged to get the mean in a single result. I chose to use random forests as it is a more robust method than just using one singular decision tree. It reduces overfitting and improves accuracy. The expected outcome of using the random forest method is that the accuracy should increase while overfitting decreases. Per Dr. Elleh’s PowerPoint “D209 Data Mining 1 Task 2 Cohort.”

**B2.**

The main assumption used in the random forest method is that the sampling is representative of the population. Since the model depends heavily on the training dataset to make informed decisions, a misrepresentation of the population data could skew the model into making incorrect classifications.

**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

**Seaborn:** Helpful for visualizations

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

The following are imported from sklearn.metrics

**Make\_scorer:** Used as a scoring function in the GridSearchCV

**Mean\_squared\_error:** Useful for obtaining the MSE of the model

**R2\_score:** Useful for obtaining the R2 value of the model

**Mean\_absolute\_error:** Useful for obtaining the mean absolute error score

**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).

**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

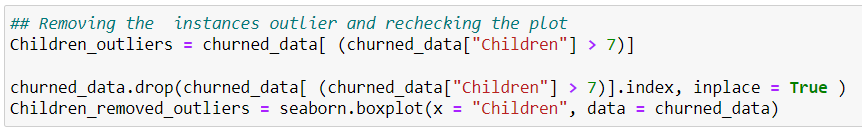
A screenshot of a computer

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 screen shot of a computer code

Description automatically generated

**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.



**Step 7:** Checking for correlation

In the last step, I checked for correlation. It is important to see if any variables are highly correlated with the response variable, as well as if they are highly correlated with other explanatory variables. This could create problems for the model in the future.



**C4.**

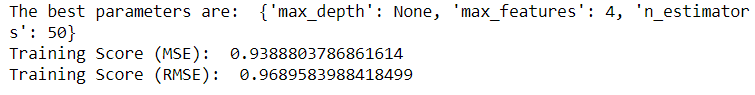
Prepared CSV is included in the submission.

**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 found the MSE, RMSE, and the R2 value for the initial and new model in these steps. Lastly, I created a “Feature Importances” plot to look at the most impactful features on the model and data.





A black text on a white background

Description automatically generated

A graph with text overlay

Description automatically generated

**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 feature importance graphic.

A computer screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

A close-up of a computer code

Description automatically generated

**E1.**

From the output, we can see the MSE for the training model is 0.9388 while it is 41.878 for the testing model. This is a large increase in the MSE and could suggest that the model is not as effective. When looking at the R2 for the initial model, we can see that the output for the training model is 0.9899, which is very high. The number drops to 0.9397 for the test model. 0.9397 is still a respectable R2 value and suggests that the model fits the data well. Although 41.878 seems high for the testing model’s MSE score, the metric is more useful when comparing two models to when another. Since the R2 value is still acceptable, the accuracy of the testing model should be ok.

**E2.**

Based on the results of the analysis, I can conclude that the model is average at classification and prediction. The testing MSE of 41.878 and RSME of 6.47 implies that the model struggled to accurately predict the values of tenure to a high degree. However, the R2 score is still acceptable, which could mean that the model is moderately accurate. There could be better models created and could have an MSE lower than this model, but this testing model still performed acceptably at predicting a customer’s tenure. Aside from the metrics, the features importance graph shows that Bandwidth\_GB\_Year is easily the single most important variable when it comes to predicting Tenure. This could have implications on what the company might focus their efforts on going forward.

**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. I also kept Bandwidth\_GB\_Year in the model despite it showing a high correlation with the dependent variable Tenure. I felt it would be best for the model to keep it in, but there are arguments for removing it as it is too closely related.

**E4.**

To answer the question posed in A1, I would say that this model is partially successful at effectively predicting Tenure length in customers. I think that the R2 values for both the training and testing models indicates that the models are good fits for the data. The MSE and RMSE are both significantly higher in the testing model than the training model, but that could be within the noise of error for predicting values. As a mentioned briefly in E3, the Feature Importance graph shows that Bandwidth\_GB\_Year is by far the variable with the greatest importance in the model. On my initial thinking, I thought that I would recommend to the company to prioritize high Bandwidth\_GB\_Year users, as they tend to have greater Tenure. However, I think a large part of the importance that Bandwidth\_GB\_Year has is that it is a counting stat. Specifically meaning that the longer a customer is with the company, they will be using more and more data. If a customer was to leave early, then their Bandwidth\_GB\_Year value would remain low for the rest of the year, as they are not using any data. Because of this, I’m not sure that Bandwidth\_GB\_Year is a good predictor for the company to focus on. I would recommend looking at the other predictors in the Feature Importance graph, such as Income, MonthlyCharge, Outage\_sec\_perweek, Age, Email, Children, and Contacts. Those variables are more independent from Tenure, and might provide more useful insight into when a customer might leave.

**F:**

Panopto video provided in the submission.

**G/H:**

Dr. Elleh, F (n.d). *D209 Data Mining 1 Task 2 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