Using random forests, is it possible to predict the customers who will churn?

By:

Kevin Sandoval

**A.**

All parts for A can be found in the GitLab.

**B1.**

The research questions I will be examining is: “Using random forests, 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.

**B2.**

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.

**C1.**

A random forest classifier is a machine learning method for classification. This properly analyzes the dataset as the classifier requires a categorical variable as the target variable. Since I am using Churn, the responses are either “Yes” or “No”. The classifier then uses “majority voting” across the decision trees to determine the final class label. This works well with the given dataset as we want to view similar customers to determine if future customers will churn. Using the decision tree classifiers on different sub samples helps with the classification of those customers. 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.” The other expected outcome for an individual classification is a “1” for Churn and a “0” for not churn as the prediction.

**C2.**

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

**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

**D1.**

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).

**D2.**

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

**Churn:** Categorical

**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:** Categorical

**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

**D3.**

**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.



A screen shot of a computer code

Description automatically generated

**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.

A close-up of a white background

Description automatically generated

**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.



**D4.**

Prepared CSV is included in the submission.

**E1.**

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

**E2.**

After using the RandomForestClassifier to create the initial classification model on the test data, I then used ClassificationReport to easily find the accuracy, precision, recall, and F1-score of the model. They are typed out below and also included in a screenshot.

A screenshot of a computer

Description automatically generated

**Accuracy Score:** 0.89

**Precision Score (For Churned):** 0.82

**Precision Score (For Not Churned):** 0.91

**Recall Score (For Churned):** 0.73

**Recall Score (For Not Churned):** 0.95

**F1-Score (For Churned):** 0.77

**F1-Score (For Not Churned):** 0.93

Confusion Matrix is shown below.

A chart of a blue yellow and purple box

Description automatically generated with medium confidence

ROC Curve with the calculated AUC is shown in the screenshot below.

A graph of a positive rate

Description automatically generated

**AUC**: 0.95

**E3.**

For the hyperparameter tuning part of the process, I chose to tune “max\_depth”, “n\_estimators”, and “max\_features”. I selected these because they are the most effective at changing the robustness of the model. For “n\_estimators”, it would be valuable to know how many data points would be the optimal amount to use when creating the classifier. The “max\_depth” looks at the depth of the tree that is created. “If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.” (scikit-learn, DeciscionTreeClassifier). “max\_features” checks for the number of features to consider when checking for the best split for the model. Since I will be passing in and integer value, it will consider features at each split. (scikit-learn, DeciscionTreeClassifier). I used the GridSearchCV function to perform the hyperparameter tuning and used 10 as the number of cross fold validations.

A screenshot of the code and output of the best hyperparameters is shown below.

A screenshot of a computer code

Description automatically generated

**E4.**

After using the GridSearchCV to tune the hyperparameters, I then used ClassificationReport to easily find the accuracy, precision, recall, and F1-score of the model. They are typed out below and also included in a screenshot.

**Accuracy Score:** 0.88

**Precision Score (For Churned):** 0.92

**Precision Score (For Not Churned):** 0.88

**Recall Score (For Churned):** 0.62

**Recall Score (For Not Churned):** 0.97

**F1-Score (For Churned):** 0.74

**F1-Score (For Not Churned):** 0.92

Confusion Matrix is shown below.

A chart of different colors

Description automatically generated with medium confidence

ROC Curve with the calculated AUC is shown in the screenshot below.

A graph with a line

Description automatically generated

**AUC**: 0.94

**F1.**

For ease of access, I decided to create a table to show the difference in metrics between the initial model and the tuned model. Any values less than 0.03 apart I considered in the noise and chose to represent those as equivalent.

|  |  |  |
| --- | --- | --- |
| Scores | Initial Model | Tuned Model |
| Accuracy | 0.89 | 0.88 |
| Precision (Churn) | 0.83 | 0.90 |
| Precision (No Churn) | 0.91 | 0.88 |
| Recall (Churn) | 0.75 | 0.62 |
| Recall (No Churn) | 0.95 | 0.97 |
| F1 (Churn) | 0.78 | 0.74 |
| F1 (No Churn) | 0.93 | 0.92 |
| AUC | 0.95 | 0.94 |

For some added insight, I’ve chosen to add the definitions of each score per the scikit-learn website for help with contextualization the results.

**Accuracy:** “this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y\_true.” Essentially this checks the model for a 1 to 1 match on if the model predicted correctly.

**Precision:** “The precision is the ratio TP / (TP + FP) where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.” In the Precision (Churn) instance, this would be the models ability to not label someone as “Churned” when they haven’t. The ratio mentioned can be used in conjunction with the confusion matrix where “TP” is the bottom right quadrant, and “FP” is the upper right quadrant.

**Recall:** “The recall is the ratio TP / (TP + FN) where TP is the number of true positives and FN the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.” In the Recall (Churn) instance, this would be the models ability to correctly label someone as “Churned” when they have. The ratio mentioned can be used in conjunction with the confusion matrix where “TP” is the bottom right quadrant, and “FN is the bottom left quadrant.

**F1:** “The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is: F1 = (2\*TP) / (2\*TP + FP + FN).” This gives weight to the True Positive results and then adds on the incorrect predictions to determine an overall score for the effectiveness of the model at predicting the true positives.

From this table, we can see that the initial and tuned model performed similarly in Accuracy score, Recall (No Churn) score, F1 (No Churn) score, and AUC. The tuned model performed significantly better than the initial model in Precision Score for customers that had churned. Intuitively, this means that the tuned model does a better job at not giving false positives compared to the initial model. The tuned model performed slightly worse on the Precision (No Churn) section, which implies that the model had more false positives on people that did not churn. Similarly, the tuned model struggled heavily with the Recall Score for churned customers. While the initial model did not perform great at a score of only 0.75, the tuned model only recorded a score of 0.62. This heavily implies that the tuned model had far too many false negative results, i.e. labeling customers as not churned when they in fact did churn. Since F1 score is a combination of Precision and Recall, we can see that the F1 score for the tuned model also is slightly lower than the initial, largely due to the Recall (Churn) score being significantly lower.

**F2.**

Based on the results of the analysis, I can conclude that the tuned model tends to take a more cautious approach when classifying a customer has churned. This can be seen directly in the lower Recall Score, but higher Precision Score for the Churned customers. Overall, I would say that the tuned model does not perform better than the initial model. Given that customers are 10x more expensive to attain than retain, I believe that an effective tuned model should return more false positives than false negatives. A false positive would indicate to the company that a customer will churn, which might cause the company to spend time, money, and/or resources trying to retain a customer that isn’t going to churn. However, I do think that that is better than the false negative, which is telling the company a customer won’t churn, and then they lose the customer because of the implication that the customer would stay.

**F3.**

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.

**F4.**

To answer the question posed in B1 I would say that the tuned model does not effectively predict the customers who will churn but does a good job at predicting customers who will not churn. There are some positives in the model to consider, such as a high precision score for churned customers, solid accuracy score, and great AUC, F1(No Churn), and Recall (No Churn) scores. These are all mostly related to labeling a customer as not churning. Since there is negligible loss in accuracy score, the tuned model is effective at correctly labeling people who will not churn. However, the tuned model struggles to effectively label people who will churn. As indicated by the low Recall (Churn) score, the model tends to label more “Not Churn” and opposed to more “Churn.” As mentioned in F2, new customers are 10x harder to attain than to retain new customers. Because of this, the model might not be useful for the company to try and determine the most at risk customers. However, I do believe the model can still have a valid practical effect, as it does a good job at finding people who will not churn. Because of this, I would recommend to the company to try and create a new model that better predicts customers who will churn, and then use that other model in conjunction with this tuned model. While 1 model might not be sufficient to predict customers who will churn and customers who won’t, having a model good at doing both would be an effective way to help the company retain customers.

**G:**

Panopto video provided in the submission.

**H/I:**

\*As a note, much of the code and resources were re-used from my D209 Task 1 and 2 submissions.

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

Scikit-learn. (n.d). *accuracy\_score.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>

Scikit-learn. (n.d). *ConfusionMatrixDisplay.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html>

Scikit-learn. (n.d). *DecisionTreeClassifier.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

Scikit-learn. (n.d). *f1\_score.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html>

Scikit-learn. (n.d). *precision\_score.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html>

Scikit-learn. (n.d). *RandomForestClassifier.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Scikit-learn. (n.d). *recall\_score.* Retrieved February 8th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html>