Becca Dura
CS167 Project B

Data Science Team Part

Problem and Data

The problem I am trying to solve is how to best predict which customers in the telecom company are going to leave the company, or churn. This is because the telecom company would like to be able to better identify customers at risk of churning, so it can reach out to those customers and improve the company's customer retention rate. The telecom company would also like to be able to use this information to determine which customer features need to be improved to prevent customer churning in the long run. In this model, I used Churn, which tells us whether or not the customer churns, as the target feature and the rest of the variables, aside from the customer ID (an indicator), as predictors.

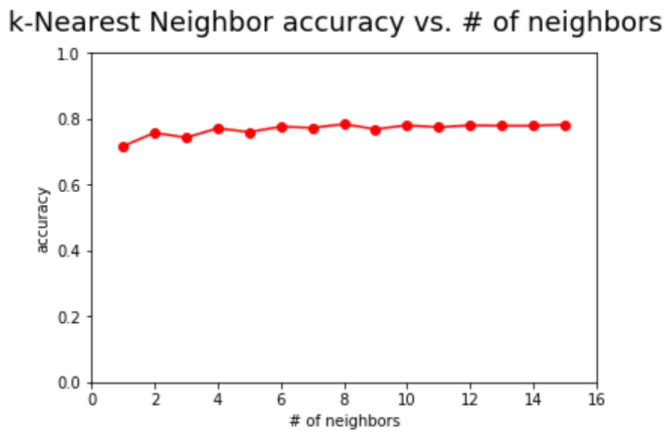
Data Prep

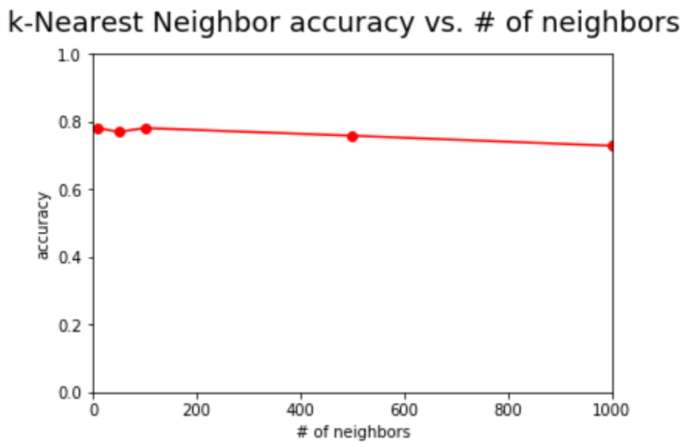
To clean up the data, I first made sure that there were not any null values. Then I checked the data types for each variable to see which variables were categorical and needed to be transformed. I noticed that something was wrong with the Total Charges variable, and after some digging, I discovered that there were 11 rows with missing values, or spaces, in the Total Charges column. To fix that, I changed the missing values to zero, changed the values in the Total Charges column to floats, and then replaced the zeros (previously missing values) with the mean of the Total Charges values. After that, I created dummy variables for the following categorical variables: Internet Service, Contract, and Payment Method. I created dummy variables for those predictors because they had more than two categories each. The rest of the categorical variables only had two categories, aside from a few that had 'No phone service' or 'No internet service'. However, I just counted those as 'No'. To fix the rest of the categorical variables, I assigned 1 to any values that were 'Yes' and 0 to any values that were 'No'. To fix the gender variables, I assigned 1 for males and 0 for females. After fixing all the categorical variables, I checked the data types, and they were all either integers or floats, which meant the data cleaning was done.

Model Experiments

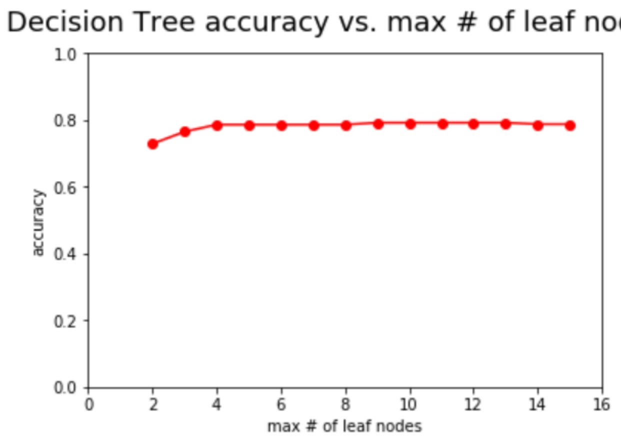
I experimented with the four following models using scikit-learn: a k-Nearest-Neighbor, a decision tree, a random forest, and a linear model. For each model, I tuned a parameter to find the best parameter for the model and to avoid overfitting the model.

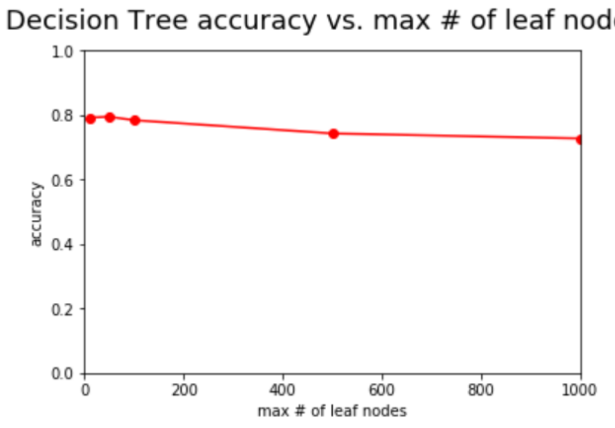
To start, I created a k-Nearest-Neighbor model, where I tuned the number of neighbors. The number of neighbors (n_neighbors) parameter is used to enter in numbers for how many neighbors the model uses to predict whether or not a customer churns, instead of having the model default to five neighbors. I created two plots to display how the k-Nearest-Neighbor model performed as the number of neighbors changed, which helped me tune the parameter. To measure the performance of my model, I used its accuracy. The first plot below shows how the accuracy of my model changes among smaller numbers of neighbors, while the second plot shows how the accuracy of my model changes if a larger number of neighbors is used. As the plots show, the accuracy is low for one to three neighbors, relatively high (and relatively the same) for four to fifteen neighbors, and begins to decrease steadily for larger numbers of neighbors. That is most likely because too few neighbors do not accurately portray the data, while too many neighbors can lead to overfitting and include a lot of outliers in the data. Based off the plots, I would say that using eight neighbors produces the k-Nearest-Neighbor model with the best performance.



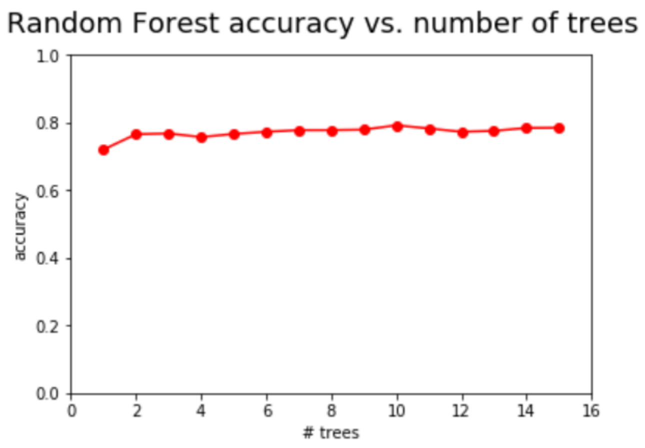


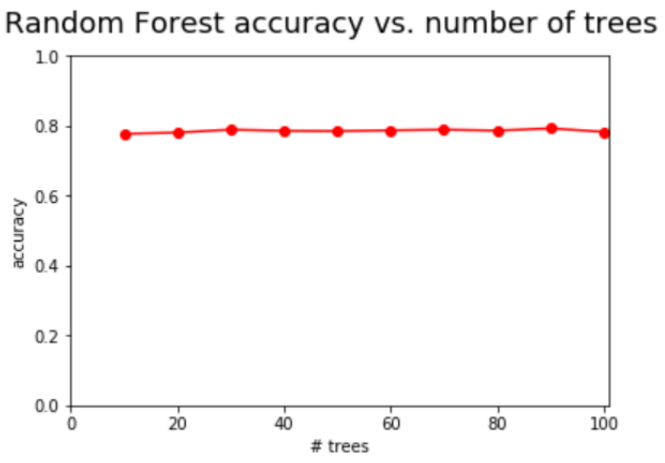
Next, I created a decision tree model, where I tuned the maximum number of leaf nodes (max_leaf_nodes). That parameter is used to put a cap on the number of leaf nodes in the decision tree, which ensures that the tree is not excessively large and only uses relevant features to predict whether or not a customer churns. I created two plots to display how my decision tree model performed as the maximum number of leaf nodes changed, which helped me tune the parameter. To measure the performance of my model, I used its accuracy. The first plot below shows how the accuracy of my model changes among decision trees with smaller numbers of leaf nodes, while the second plot shows how the accuracy of my model changes if the maximum number of leaf nodes is larger. As the first plot below shows, there must be more than one leaf node for the decision tree model to work. The accuracy of the decision tree model looks to be lower if the maximum number of leaf nodes is two or three. However, the accuracy is highest and steady if the maximum number of leaf nodes is between four and fifty. Although the accuracy is steady, as more leaf nodes are used, the risk of overfitting increases. As the second plot shows, if the maximum number of leaf nodes is greater than fifty, the performance of the model tends to decrease. That is most likely because there are not that many features with significantly high importance. Based off the plots, I would say that a decision tree model with a maximum of four leaf nodes performs the best.

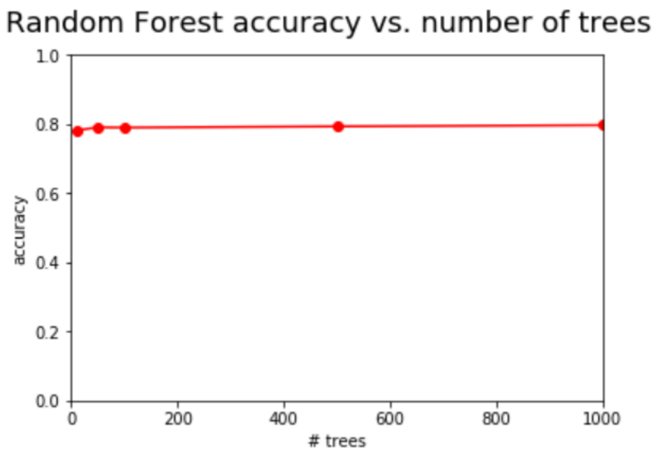




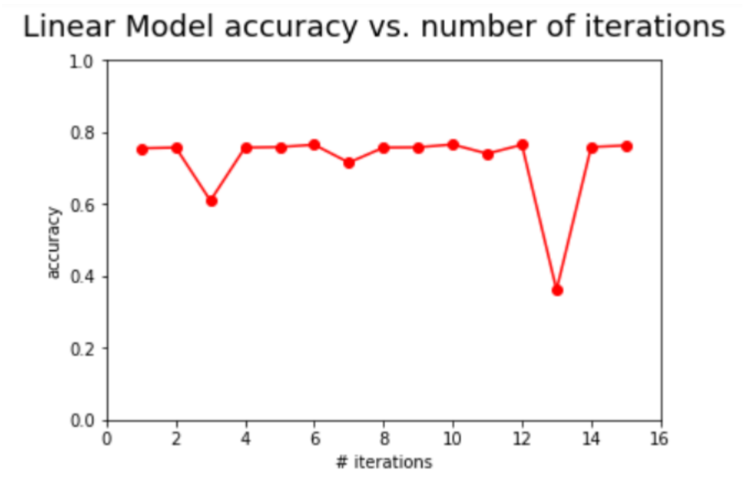
Then, I created a random forest model, where I tuned the number of trees (n_estimators). That parameter just changes how many decision trees are used in the creation of the random forest model from the default of ten trees. I created three plots to display how my random forest model performed as the number of decision trees used in the model changed, which helped me tune the parameter. To measure the performance of my model, I used its accuracy. The first plot below shows how the accuracy of my model changes among random forests with less decision trees, while the third plot shows how the accuracy of my model changes if there are a lot of decision trees used. The second plot is something I added in because it looked like the accuracy was the highest when around fifty decision trees were used in the random forest model, and I wanted to tune the parameter a little more. Although the accuracy is a little worse when fewer decision trees are used, the number of decision trees used in this random forest model does not seem to have a large effect on its performance, as shown in the plots below. The model seems to hit its peak accuracy when ten to thirty decision trees are used and plateau off after that; however, using too many decision trees could potentially lead to overfitting. That is most likely because there are a lot of features with some significance, but certain features are much more significant than others. Based off the plots, I would say that the random forest model predicts whether or not customers churn best when thirty decision trees are used to create the model.

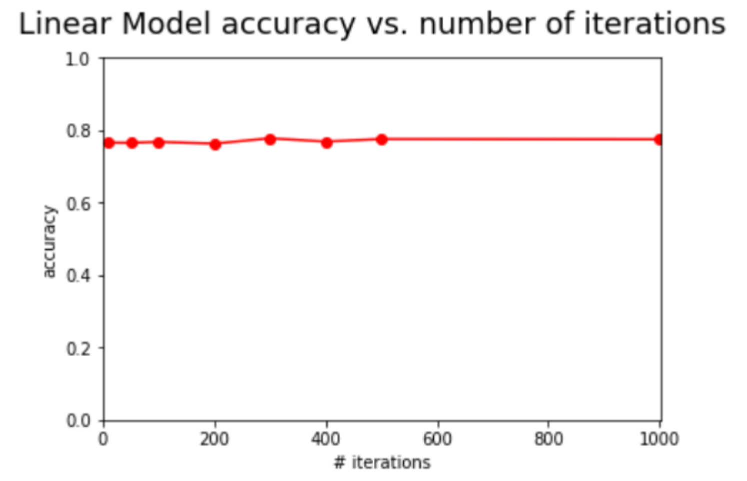






Finally, I created a linear model, where I tuned the maximum number of iterations (max_iter). That parameter indicates the maximum number of times the algorithm should pass over the training data when building the model, so it is not set at the default, which is five. In other words, it is the maximum number of epochs. I created two plots to display how my linear model performed as the maximum number of iterations changed, which helped me tune the parameter. To measure the performance of my model, I used its accuracy. The first plot below shows how the accuracy of my model changes among low values for the parameter, while the second plot shows how the accuracy of my model changes if the maximum number of iterations is higher. As the plots show, the accuracy is not very stable among lower numbers of iterations. However, it seems to remain stable for larger numbers of iterations. The peak accuracy, according to the plot, occurs when the maximum number of iterations parameter is set at 300. After that, the accuracy looks like it plateaus off. Since I do not want too many iterations or to risk overfitting the model, I would say that the linear model best predicts whether or not customers churn when the maximum number of iterations is 300.





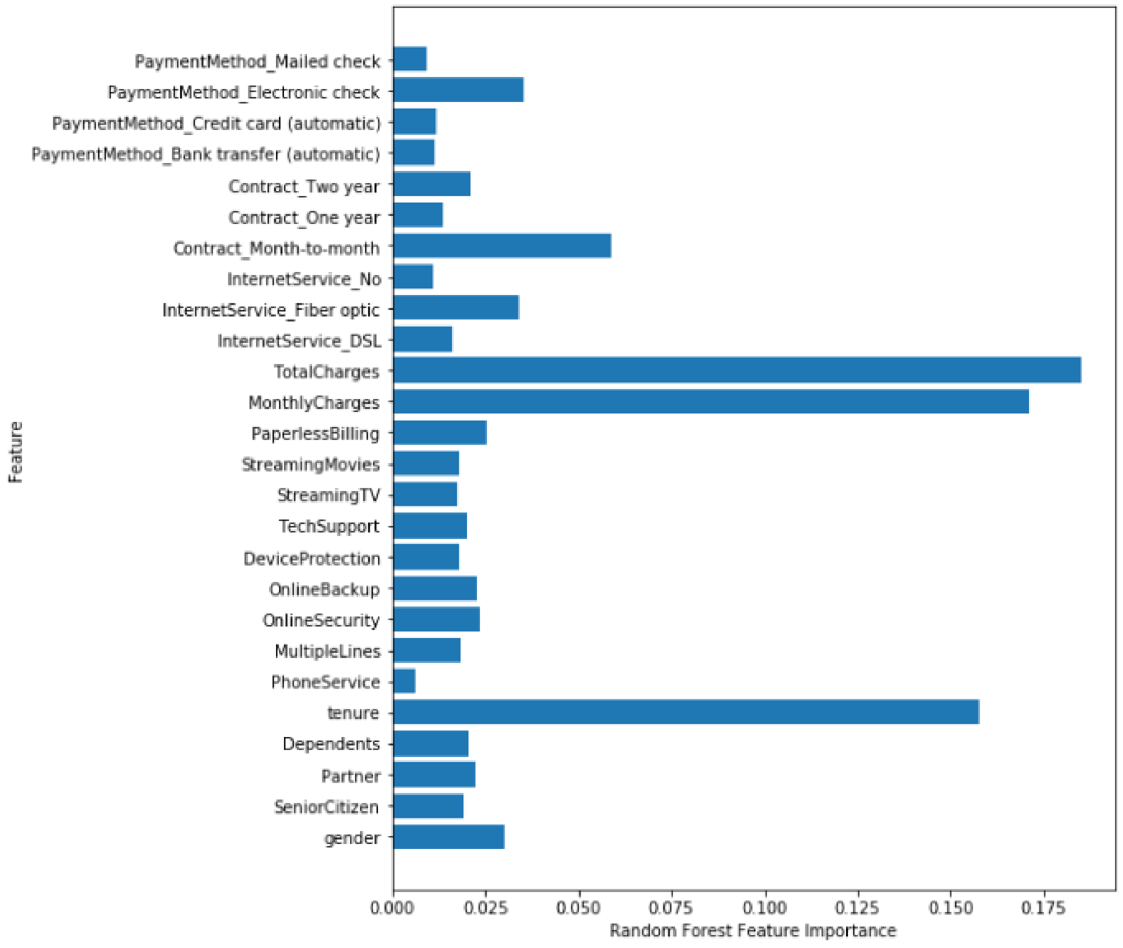
Experiment Interpretation

After examining all the models and tuning the parameters, none of the models really stood out. The random forest model had the highest accuracy (0.787), but not by much. I think the random forest model performed the best because, although there were only a few extremely important features, there were a lot of features with some importance. When there are few important features, decision trees perform better. Since the random forest model is created based off decision trees, it performed the best, especially since the parameter I tuned was the number of decision trees used to build the random forest model.

When it came to tuning the parameters, the models all showed a slight difference in performance among different parameter values; however, the linear model had the greatest variance in accuracy for various numbers of iterations (the parameter). When certain values under ten were tested as the maximum number of iterations in the linear model, the accuracy dropped as low as 0.36. That is why I decided a higher maximum number of iterations would create a better performing model.

Feature Importance

I created the bar graph below to show the importance of the features in the random forest model. As the bar graph shows, there are three features that are highly important in the random forest model: TotalCharges, MonthlyCharges, and tenure. However, a lot of the features have some importance. The first conclusion I would make based off the bar graph is that what a customer is charged, both monthly and in total, is a key component in determining whether or not a customer churns. A customer with higher charges, both monthly and in total, is more likely to churn than a customer with less charges, and vice versa. The other conclusion I would make is that the number of months a customer has been with the telecom company is an important predictor of whether or not a customer will churn. If a customer has been with the telecom company for a longer period of time, the customer is less likely to churn due to loyalty to the company, and vice versa. In conclusion, there were only three features of great importance in the random forest model, but those three features can tell us a lot about customer churning at the telecom company.



Non-Technical Part

Recommendation

Based off my experiment, customers who have been with the company for a longer period of time and have more tenure are less likely to churn. To improve customer retention for the telecom company, I would recommend looking into how to better retain newer customers. The telecom company already does a good job of keeping customers who have been with them for a while, but there is a high turnover rate for newer customers. The company should undergo a review of its newer customers and what they are looking for in a telecom company. Since a customer's total and monthly charges also plays a key role in whether or not a customer churns, the company may be charging newer customers more, while freezing prices for older customers. The company should look into not raising prices for newer customers, or at least offering discounts to new customers for the first few months. In other words, I would recommend that the telecom company look into how to better retain newer customers and lower charges for customers to prevent them from churning.