**MACHINE LEARNING**

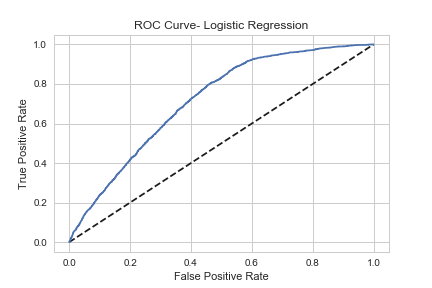
**Data ML Pre-Processing**

In order to run machine learning algorithms on this data, the information needs to be translated into a format that can be interpreted by the algorithms. Each column that we are using must be in either a binary form of 0 or 1. While some columns were already in this form—e.g. SMS\_received—the rest needed to be converted from a categorical variable to binary values with the pandas get\_dummies function. We dropped the first dummy column to prevent variable correlation in the columns. Next, we split the data and targets into two groups, a training group and a testing group, using train\_test\_split. At this point we ran machine learning algorithms on the dataset, but we found that we were not getting reliable predictions. To increase the accuracy of the algorithms, we split the data again and then ran SMOTE—Synthetic Minority Over-Sampling Technique—on the second training segment. For K-Nearest Neighbors and Random Forest we used grid search to tune the hyperparameters. To determine the effectiveness of the models in no-show prediction, we ran f1-tests. We used a custom threshold to maximize f1-scores for no-show prediction. Below are the scores for different configurations of these machine learning algorithms.

**Logistical Regression CV Classifier**

We first implemented LogisticREgressionCV from sklearn’s linear model using only the top features of our dataset. These top features determined previously were SMS\_received, Age\_Binned, Wait\_Binned and Neighborhood. Our initial model predicted that all patients would arrive for their appointments. Using 10-fold cross validation and f1 scoring gave a few prediction of not showing up and a f1-score of 0.01. This is because the predict function uses a threshold of 0.5 for predictions, but our data has a natural no-show rate of approximately 0.2. We used predict\_proba() to implement different thresholds and found a threshold of greater than 0.21 gave the highest f1-score of 0.446. Additionally, we had an AUROC (Area Under the Receiver Operating Characteristic curve) of 0.725. To further improve our model’s performance, we split the data again and ran the SMOTE algorithm from imblearn’s over sampling package to balance the number of patients who showed up and did not show up to their appointment. Here we found the same max f1-score of 0.446 but now with a threshold of 0.5. Additionally, after applying SMOTE we the AUROC decreased slightly to 0.724.

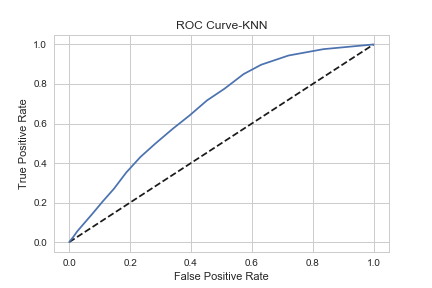
We then included the following additional features of statistical significance: scholarship, hypertension, diabetes and appointment day. We ran SMOTE algorithm as again on this larger dataset. With the balanced larger dataset, we found a max f1-score of 0.439 with a threshold of greater than 0.51. The AUROC was 0.714.



**K-Nearest Neighbors Classifier**

The next program we utilized was the KNeighborsClassifier algorithm. We first tried this with an 80%-train, 20%-test split. We used only the top features of the data. This resulted in an AUROC of 0.65 and a f1-score of 0.245. A grid search found that 18 neighbors seemed to be ideal. When we implemented a threshold, we could get a maximum f1-score of 0.424 with a threshold of 0.22. We applied SMOTE on just the top variables which resulted in AUROC of 0.69. The balanced data had a f1-score with no threshold of 0.347. When we applied a threshold of 0.22 we found a max f1-score of 0.431.

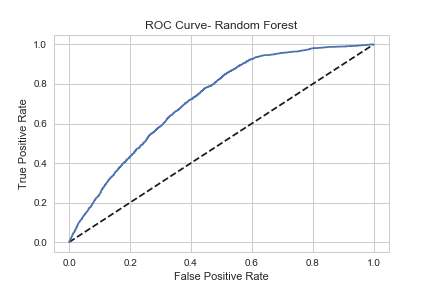
We then included all the variables and rebalanced the data using SMOTE. We could get a maximum f1-score of 0.414 with a threshold of 0.22 and 18 neighbors. The data generated an AUROC of 0.68.



**Random Forest Classifier**

We first ran the RandomForestClassifier() on our top categories and found a small f1-score of 0.11, as almost everyone was classified as having successfully arrived for their appointments. The AUROC was 0.704. Running a grid search cv for hyperparameter tuning advised using max\_depth= 30, min\_samples\_leaf= 1, n\_estimators= 9. This change increased the AUROC to 0.716; however, without balancing the data, the f1-score dropped to 0.05. Applying a threshold of over 0.22 to the predic\_proba function increased the f1-score to 0.434.

We rebalanced the data and included the remaining variables. We tuned the hyperparameters with grid search and a manual method which resulted in an AUROC of 0.72 and a maximum f1-score of 0.44 with a threshold of greater than 0.43. The best hyperparameters we found were a maximum depth of 35 and min\_samples\_leaf of 12 using 26 estimators.



**Conclusion**

We found the best predictions using logistic regression cv, followed by random forest, then, followed by k-nearest neighbors. However, these analyses were all very similar in their overall effectiveness. Using these algorithms, our client can now predict which appointments are more likely to be missed. Healthcare groups and physicians’ offices will be able use this information to schedule more patients or to bring in fewer practitioners on days when more patients are predicted to fail to arrive for their appointments. Additionally, medical offices can use this algorithm to implement different reminder methods patients who are predicted to no-show in order to increase the likelihood that these clients do not actually miss their appointments. Another recommendation to medical office staff would be to call likely no-show patients early on the date of their appointment to find out if the patients anticipate arriving to their appointments. The appointments freed by clients who cancel in this manner could then be offered to a back-up waiting list of patients who would like to take last-minute appointments. Alternatively, if there are days where the algorithm predicts very unlikely no-shows for the scheduled appointments, more doctors could be brought in to help with back up.