Predicting Hospital Readmission Using Different Machine Learning Models

Abstract

This paper presents a study about hospital readmission rates for diabetic patients. Four machine learning models; Decision Trees, Support Vector Machines, Random Forest, and Gradient Boost Classifier were used in order to most accurately predict if a patient with diabetes will be readmitted to the hospital within 30 days. In order to achieve this, data exploration, feature engineering, building training and validation samples, model selection, and model evaluation were all used in this process. These models will be training on a data set that represents 10 years of clinical care at 130 US hospitals.

1 Data Processing

Before training this large amount of data, the data will be prepared for the best possible outcome. Upon reading and understanding the distribution of the data in the diabetes data set, it was realized that there may be missing values in which most of them were written in strings. The data was selected and then dropped from the data set, then processed and finally transformed to specific data accordingly.

In the diabetes data set of 50 labels, it was noticed there were 37 categorical labels. Majority of these categorical data have string values which could be a problem to train machine learning model. Thus, it was decided to encode all these string values into integers.

It was decided to manually look at the various columns of data has been made readily available. Once this has been accomplished the appropriate columns were selected. It was decided to leave out all of the medications as it does not bring significant value to the prediction of patient readmission with diabetes. Within this data set, the data has been selected from the X data values to be all of the columns except diag 1, diag 2, diag 3, encounter id, patient nbr, and readmitted. These columns specifically were dropped because according to the IDs-mapping file, it can be seen that these columns are related to death or hospice. Therefore, these specific columns were removed these samples from the predictive model.

In the data there were many different values missing that were given, specifically marked as '?' under the columns race, player, code, medical specialty, and weight. The '?' within the data was additionally converted into NumPy Nan values in order to be used to preprocess the missing data of the data set. The most important column within the data supplied is the readmitted as it describes if a patient was hospitalized within 30 days, more than 30 days or was not readmitted at all. The output Y label into the data frame by having '>30' (readmitted after more than 30 days) as value of 0, '<30' (readmitted after less than 30 days) as value of 1, and 'NO' not readmitted at all with the value of 2. Using the LabelEncoder non-numerical labels were transformed to numerical labels. Two data frames were then created by extracting all columns except the 'output' column and the columns dropped that were specified earlier was used as a X label. In addition, the Y label was constructed by extracting the 'readmission' column from the main preprocessed data. Once this was completed, the team began to split the data into x and y training/validation data through using sklearn's train-test-split function to split the data into 70 percent training set and 30 percent validation set. Preprocessing of the X training and validation was also done. The validation and training data was specifically preprocessed through the use of a standard scaler.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

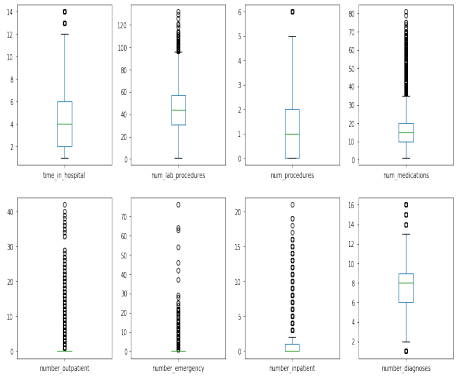

Figure 1: Box-Whisker Plots of the Important Features

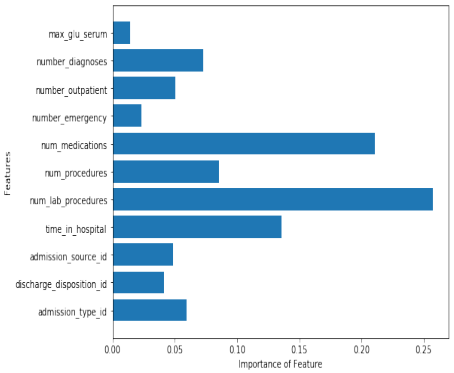

Figure 2: Values of the Importance of Features

As shown above the box-whisker plots demonstrate of all the features that are considered important that will affect the outcome of the models. Using a decision tree model, the second graph now displays the importance of the features and its weight when figuring out the readmission rates.

2 Decision Trees

To begin classifying , the Decision Tree model was chosen initially. Varying maximum depths from 1 to 27 were used on the this model. Here, it was found that the smallest maximum depth of 1 had an accuracy of 0.557 on the training data set and 0.560 on the validation set. While the largest tested maximum depth of 27 had an accuracy of 0.928 on the training data set and an accuracy rate of 0.491 on the validation set. The decrease in accuracy score on the validation data from 0.560 with a maximum depth of 1 to 0.491 accuracy score with a maximum depth of 27 leads us to believe that the data was over fitting the Decision Tree model. Therefore, there will be a lower accuracy score as the maximum depth is increased. Upon reviewing the results with vary maximum depths, it can be seen that the best maximum depth for the decision tree would be 7 as it has the highest accuracy score on the validation data set of 0.578.

In addition, with a maximum decision tree depth of 7, the accuracy score on the training data is 0.582 which when compared to the accuracy score of 0.578 on the validation data there does not appear to be any over fitting as the accuracy scores between the training and validation data is very close. A confusion matrix, with a maximum depth of 7, was constructed to show how well the decision tree model predicted the outcomes of readmission in patients as looking at accuracy scores alone will not be enough to access the decision tree model's performance on its predictions. It was found that there were 3425 true positive outcomes and 84 false positive outcomes. This is fairly good as the number of true positive outcomes is much higher than the false positive predictions the model made. This is good to see as the true positive is an outcome where the decision tree model correctly predicts the positive class of having readmission. It was found that there were 1248 false negative outcomes and 74 true negative outcomes. In contrast, it can be seen that the decision tree model did not do as well when predicting true negative outcomes as there is a drastic difference between false negative outcomes and true negative outcomes. Also, the overall measure of recall (the proportion of actual positives was identified correctly) is 0.399. This model found an overall precision score of 0.501. However, it must be noted that precision is defined as the percentage of the results that are relevant. Meaning the decision tree model found about half of the results to be relevant.

As shown in figure 3 above, there appears to not be any over fitting from the maximum depths of 1 to 10. At maximum depth from 11 to 27, there is extreme over fitting within the data as the area under the curve for the training values compared to the area under the curve for the test values are much higher as can be seen in the figure.

Another set of data to demonstrate the model's scores is from a classification report. A classification report is used to provide measurement on the quality of predictions provided by the model. Demonstrated by the classification report, the decision tree had a weighted average precision of 0.54, along with a weighted recall average of 0.58, and a weighted f1-score average of 0.53.

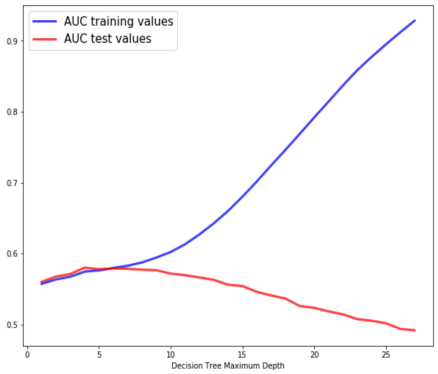

Figure 3: Area Under the Curve for Decision Tree

3 Random Forest

After testing the data set with the Decision Tree model, it was decided to move onto the Random Forest model. Unlike the Decisions Tree model, the Random Forest model utilizes multiple decision trees of random depth and uses averaging to improve the accuracy of the various sub samples the model takes. Instead of figuring out the accuracy of each depth, the model was ran with the default 100 estimators (trees). After running the model, the training set had the accuracy of 0.5448 and the test set had the accuracy of 0.5434.

Given the accuracy of the training data and the test data, it can be concluded that there isn't any over fitting though the accuracy scores do come very close. A confusion matrix was constructed to further demonstrate the outcome of the model as the accuracy of both data sets alone will not be enough to assess the model's performance. It was found that there were 3962 true positive outcomes and 75 false positive outcomes. These results are good because this demonstrates the model's ability to predict the positive class of readmission. It was also shown that the model had 1331 false negatives and 76 true negatives. Like the decision tree model, the random forest model did not do as well when predicting true negative outcomes. In addition, the model's had the an overall recall score of 0.399. Alongside its recall score, the random forest model also had an overall precision score of 0.501. Again it needs to be noted that precision is defined as the percentage of the results that were found to be relevant.

Alongside the usage of a confusion matrix, the classification report shows that the decision tree had a weighted average precision of 0.55, along with a weighted recall average of 0.58, and a weighted f1-score average of 0.54.

4 Gradient Boosting

Another model apart from decision tree and random forest that was decided to train our model to test the accuracy of it was Gradient Boosting. In this case, Gradient Boosting Classifier will be taking random forest as a baseline model to optimize the performance. To perform these, our team will be taking the base random forest model and apply gradient boosting as an ensemble method to help reduce bias and variance, and is a sequential classifier. To get the best accuracy outcome, the learning rate has been varied between floating values from 0 to 1. To train this Gradient Boosting Classifier model, a value of 20 was set to the number of boosting stages, max feature and max depth was set to 2. After running this model on varies learning rate, there have been a linear rate of increase in both the training and validation accuracy score.

Given this range of data, our team have noticed some relationship between the learning rate and the accuracy of the result. Starting off with a learning rate of 0.5, our team was getting training accuracy score of 0.5426 and validation accuracy score of 0.5421. The difference was subtle. As the learning rate increase to 1, the training accuracy score was 0.5794 and validation accuracy score was 0.5669. The difference were more noticeable when learning rate is at the value of 1. There is an increase in the validation accuracy value from random forest (0.543400) after applying gradient boosting with a learning rate of 1 (0.566885). However, after getting the classification report, the precision value (weighted average) was about 0.44 and the recall (weighted average) was about 0.37. These values were lower than what would have been forecasted. Comparing to the base model

random forest, these were actually lower than the precision(0.55) and recall(0.58) values obtained from the random forest's performance.

5 Search Vector Machine

Lastly, the Search Vector Machine (SVM) was used to test the accuracy against all of the current models that were being used. Amongst all of the models being used, the SVM has the highest accuracy. Unlike the decision tree and random forest model, SVMs use mathematical functions (kernels) these kernels are utilized to take data as input and transform it into the required form. After going through the data set with the SVM and its default kernel (RBF), the model was able to achieve an accuracy of 0.583 on the training data set and 0.580 on the validation set. Given that the SVM had a linear kernel, the run time to process all of the data would've been significantly faster however it comes at a cost of its own accuracy. With the linear kernel, the SVM would have the accuracy of 0.521 on the training data set and 0.519 on the validation set. This indicates that the complexity of the kernel. This shows that the kernel complexity is crucial to the SVM.

When comparing the SVM's accuracy score on the training data set and the validation set, the scores do not appear to be displaying any form of over fitting as the accuracy scores between the training and validation data are very close. To further look into the SVM's precision, and other data, a confusion matrix was created. Shown from the matrix, there are 2505 true positive outcomes and 2 false positive outcomes. This is very good as the number of true positives outcomes out weight the small number of false positive outcomes. This is a good indication that the model is very reliable on making a correct prediction on the positive class of having readmission. In addition, it was found that there were 940 false negative outcomes and 4 true negative outcomes. Similar to the positive classes, the SVM performed very well on predicting true negative outcomes. Overall, the SVM ended up having the recall score of 0.234 and the precision score of 0.522. The overall recall measure of the SVM is on the lower side in which it is predicting the three different labels of readmission. Given the precision score of the model, it should be noted that the score is defined as a percentage of the results that were considered to be relevant.

Alongside the usage of a confusion matrix, the classification report shows that the SVM had a weighted average precision of 0.56, along with a weighted recall average of 0.58, and a weighted f1-score average of 0.50.

6 Comparing Classifiers

When it comes down to performance and accuracy amongst all of the tested models, the SVM had the best accuracy in terms of precision, recall value, and f1 scores. However despite its accuracy, the SVM model has an issue of having a high training time when it comes to large datasets. This model also works poorly with overlapping classes and tends to be be very sensitive to the type of kernel used. SVM kernels determine the speed in which a dataset can be processed and its accuracy.

The worst performer amongst all of the models used was the Gradient Booster. The model's precision score, recall score and f1 scores were lower than the decision tree model scores. It however has one main advantage in which as their is an increase of ensembles or iterations, the Gradient Booster's scores increase. Despite this fact the model was limited to a certain amount of iterations to keep testing simple and fair.

Another model that was covered is the Decision Tree model. As the model's depth increased to it max, the precision, recall, and f1 scores increased however due to testing purposes the model was limited to a depth of 7. The Decision Tree model provided fair scores for its depth and what was given to it. This model outperformed the Gradient Booster, and the SVM in terms of both performance and run time.

Lastly, a major improvement above the Decision Tree is the Random Forest model. This model runs multiple Decision Trees which therefore causes it to have very fast run times with good precision, recall and f1 scores. Amongst all of the models that were tested, the Random Forest model proved to be very useful for finding the readmission rates of the patients derived from the data set.

After testing all of the various models multiple times with trial and error, it was concluded that the Random Forest model was very useful in predicting the readmission per patient. Overall, the Random Forest model had the one of the faster run times while giving standard scores. Given the time, the SVM would've been chosen but the run time were too long. If the Gradient Boosting was allowed to have a larger amount of boosting stages, the Gradient Booster would've have the best scores and run time.

References

[1] "1.10. Decision Trees." Scikit, scikit-learn.org/stable/modules/tree.html#id2.

[2] "1.4. Support Vector Machines." Scikit, scikit-learn.org/stable/modules/svm.html#kernel-functions.

[3] "3.2.4.3.1. Sklearn.ensemble.RandomForestClassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.Belle, Vanya Van, et al.

[4] "Difference between SVM Linear, Polynmial and RBF Kernel?" ResearchGate, 24 Nov. 2018, www.researchgate.net/post/Diffference_ between_ SVM_ Linear_ polynmial_ and_ RBF_ kernel.

[5] "Classification Report." Classification Report, Yellowbrick v1.1 Documentation, www.scikit-yb.org/en/latest/api/classifier/classification_ report.html.

[6] "Decision Tree Classification in Python." DataCamp Community, www.datacamp.com/community/tutorials/decision-tree-classification-python.

[7] "Gradient Boosting in Python Using Scikit-Learn." Ben Alex Keen, benalexkeen.com/gradient-boosting-in-python-using-scikit-learn/.

[8] "Gradient Boosting Regression." Scikit, scikit-learn.org/stable/auto_ examples/ensemble/plot_ gradient_ boosting_ regression.html?highlight=gradient boosting.

[9] "Gradient Boosting Regularization." Scikit, scikit-learn.org/stable/auto_ examples/ensemble/plot_ gradient_ boosting_ regularization.html?highlight=gradient boosting.

[10] Gupa, Shashank. "Why Is RBF Kernel Used in SVM?" Cross Validated, 1 July 1965, stats.stackexchange.com/questions /172554/why-is-rbf-kernel-used-in-svm.

[11] "How to Select Support Vector Machine Kernels.", KDnuggets www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html.Huneycutt, Jake.

[12] "Implementing a Random Forest Classification Model in Python." Medium, Medium, 21 May 2018, medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652.

[13] Krishnan, Muthu. "Understanding the Classification Report through Sklearn." Muthukrishnan, 19 Oct. 2019 muthu.co/understanding-the-classification-report-in-sklearn/.

[14] Maklin, Cory. "Gradient Boosting Decision Tree Algorithm Explained." Medium, Towards Data Science, 21 July 2019 towardsdatascience.com/machine-learning-part-18-boosting-algorithms-gradient-boosting-in-python-ef5ae6965be4.

[15] Nelson, Dan. "Gradient Boosting Classifiers in Python with Scikit-Learn." Stack Abuse, Stack Abuse, 17 July 2019 stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/.

[16] "Parameter Estimation Using Grid Search with Cross-Validation." Scikit, scikit-learn.org/stable/auto_ examples/model_ selection/plot_ grid_ search_ digits.html.

[17] "Sklearn.model_ selection.GridSearchCV." Scikit, scikit-learn.org/stable/modules/generated/ sklearn.model_ selection.GridSearchCV.html.

[18] "Sklearn.svm.SVC." Scikit, scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html# sklearn.svm.SVC.

[19] "Sklearn.tree.DecisionTreeClassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.tree. DecisionTreeClassifier.html.Yiu, Tony.

[20] "Understanding Random Forest." Medium, Towards Data Science, 14 Aug. 2019 towardsdatascience.com/understanding-random-forest-58381e0602d2.