Diabetes Readmission

Parth Desai

Diabetes is a disease that affects the body's ability to use blood sugar and can lead to a build up of sugar in the blood that can lead to serious problems. Normally in the body, the pancreas produces insulin and that insulin lets the sugar enter cells for use in the body. However, in diabetes patients the insulin proteins are destroyed, or the body builds up a great tolerance to insulin and the sugar doesn't enter the cell [1]. One in ten Americans are diabetic while one in three are prediabetic [2]. This puts a great strain on our hospital systems in finding the best way to help these people. Diabetes patients usually are admitted to hospitals due to hypoglycemia, which is low blood sugar [3]. If a patient is readmitted, this means that not enough was done with the patient, either form a medical or personal standpoint. A key to be able to better care for these patients is early recognition of who will need to be readmitted. The goal of this project is to be able to predict whether or not a patient will be readmitted within 30 days to better narrow down which patients might need extra care. This project I broken down into three main parts that can be summed up as Dataset Preprocessing & Cleaning, Model Training & Tuning, and Conclusions. Dataset Preprocessing & Cleaning contains a look at the input data, how we understand it, and what was needed to be done in preparation for our models.

I. Dataset Preprocessing & Cleaning

A. About the data

The dataset for this report was from a public dataset provided by the Center for Clinical and Translational Research at VCU [4]. It is a set gathered form data from 130 hospitals over 10 years in the united states. The data was in the form of two CVSs. One was made up of the data and targets and was read into python using pandas. The other was a referential document about the numbers used to represent categorical data. Coming in, there was 55 attributes with slightly more categorical than numerical features. Some were completely filled while many had a lot of missing values. The scales were very different across many of the features and there was a total of 101766 rows of patients. The target label was named readmitted and consisted of three different outcomes. If the outcome is no, the patient was not readmitted to the hospital. If a patient was readmitted within the last 30 days the outcome was <30, and if the patient was readmitted after 30 days, the result was >30. Only about 10% of patients are readmitted.

B. Missing Values Feature Removal

There were a lot of attributes that went into this dataset, but this needed to be trimmed down for many reasons. Starting with the columns of missing values, the weight attribute was dropped because 96.6% of the rows had a '?' value. This was because the data is from between 1999 – 2008 and weight data was not required to be recorded in a structured manner until after 2009. The payer code attribute was missing 40% of its values and should have no impact on a diabetes readmission so that feature was also removed. The medical specialty feature was missing 49% of its values so that column was also dropped. Patient number also was dropped do to having no effect.

C. Correlation Feature Removal

The next lot of attributes had extremely low correlation ($-.005<r<.005$) to the target data. These categories included possible tests that were run on the patients. Testing for acetohexamide, nateglinide, chlorpropamide, glimepiride, tolbutamide, pioglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, citoglipton, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, and metformin-pioglitazone all had extremely low correlation. Many of these tests had an extremely high rate of not being administered. Another group of these tests had the same result every time, giving no new information. This knowledge about the data lines up with the lack of correlation. Another group of attributes that did not belong were the id numbers. Encounter id and patient id so those columns were thrown out. A category that had too much correlation with three other categories was num_medication which made sense because the number_diagnosis, num_procedure and diabeatesMed? all would influence that number, so num_medications as removed.

D. Cleaning

The dataset now contains only 26 features, however many of those features need more work in order to be useful to us in our models.

1) Missing numeric Values

Missing values were contained in diagnosis one, two, and three. These were on a range 0-999. I decided that replacing them with the mean would be the best way to fix these missing values. I made the '?' values NaN values in pandas and then converted the column to numeric to take the mean.

2) Categoric Demographic

The age, gender, and race categories. This nominal data needed to be represented with numbers, so I chose to use 0 to represent the missing values and 1-5 for the races and 1-2 for gender. The age was given in a range of 10 (10-20), so I used 1-10 to represent the different ranges.

3) Categorical Test Results

Next was the test results. They had values of No, Steady, Up, or Down, with a few exceptions. The feature max_glu_serum had values of None, Norm, >200, and >300. A1Cresult had values of None Norm, 7 and 8. They all had 4 outcomes with the first being that the test was not administered. I gave a 0 for No or None, and a 1-3 value for the rest.

4) Target Feature

With the target feature, the goal is to predict whether or not a patient will be readmitted within 30 days, so the No and >30 categories were made into 0 and <30 was made a 1. These will be the outcomes that we will predict.

5) Feature Scaling

The variance in ranges of the features was quite great with some things being on 0-1 scale with others on a 0-999 scale. Looking at all of the columns that were numeric data, I decided to put everything on the same scale using the minmax scaler form sklearn. Preprocessing. Now the values in each column are similar, which helps in the training process. The data was then split into a test and train groups for training with the training size being 70% of the dataset.

6) Class imbalance

The split in our classes is about 90% negative to 10% positive. There are two things we can do for class imbalance. We can undersample and remove a number of majority class rows so our set looks balanced and the model doesn't go for exclusively one answer. The other way will be having class weights for our minority class be higher than for the majority, since a false positive is better than a false negative. We will test data at 1:9 ratio without weights, a 1:1 ratio with out weights and a 1:9 ratio with weights.

II. Model Training & Tuning

Three model types were selected as candidates for the classifier. Those would include a multilayer-perceptron model, a random forest classifier and a k nearest neighbors' classifier. For each model, the data was ran with the full 101,766 rows of data with a 1:9 ratio of positives to negatives and with a dataset 22,714 row dataset with a 1:1 ratio of positives to negatives. The random forest classifier, another run was done with a 1:9 ratio with class weights. MLP and KNN did not support class weights.

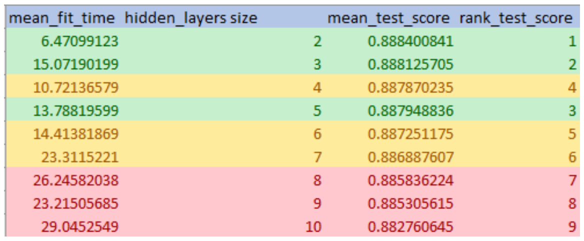
A. Multi-Layer Perceptron

For this type of classifier, one-hot encoded nominal categorical features can help because they get rid of the bias of bigger number categories. So for features race, gender, age, admission type, discharge disposition and admission source, they were dropped and replaced by their one-hot encoded counterparts.

1) 1:9 Ratio without Weights

There are many things to take into consideration like number of hidden layers what solver and what solver to use. At the beginning, a model was created with 5 hidden layers, with a L-BFGS optimizer starting with an initial learning rate of .05 and running it for a maximum of 300 iterations. The accuracy of this model was 88.9289%.

To check how many hidden layers being the best was, a function called GridSearchCV was used, from sklearn, to go through hidden layers ranging from two to ten and ran 5-fold validation on it. GridSearchCV uses the entire dataset and splits it itself into k, in this case 5 and tries all of them and performs cross validation trying every combination.

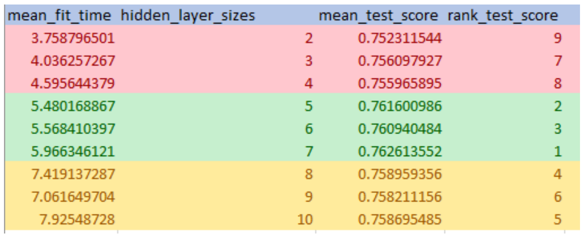


mean_fit_time	hidden_layers size	mean_test_score	rank_test_score
6.47099123	2	0.888400841	1
15.07190199	3	0.888125705	2
10.72136579	4	0.887870235	4
13.78819599	5	0.887948836	3
14.41381869	6	0.887251175	5
23.3115221	7	0.886887607	6
26.24582038	8	0.885836224	7
23.21505685	9	0.885305615	8
29.0452549	10	0.882760645	9

Table 1

In *Table 1* we can see the lowest values were the best with a hidden layer size of 2 taking the highest mean test scores. The test score is the accuracy score. 5 seemed to be a good choice, however we can choose 2 for our hidden layer parameter now that we validated it.

2) 1:1 Ratio



mean_fit_time	hidden_layer_sizes	mean_test_score	rank_test_score
3.758796501	2	0.752311544	9
4.036257267	3	0.756097927	7
4.595644379	4	0.755965895	8
5.480168867	5	0.761600986	2
5.568410397	6	0.760940484	3
5.966346121	7	0.762613552	1
7.419137287	8	0.758959356	4
7.061649704	9	0.758211156	6
7.92548728	10	0.758695485	5

Table 2

The same steps were taken as in the 1:9 ratio, however we have an even balance of positive to negative readings now in our dataset. Training the MLP with the same initial parameters as before, 70.3154%. The results of running GridSearchCV with the same possible parameters are in *Table 2*. The best number of hidden layers for this data was middle values with 7 having the highest mean. This model had a mean accuracy of 76.2613%.

B. Random Forest Classifier

1) 1:9 Ratio without Weights

Initially the model was to have decision trees with a max depth of 3 and 10 estimator trees in the classifier. The result of this is an accuracy of 88.7913%.

Checking the accuracy value of various number of estimators without specifying a max depth in *Figure 1*.

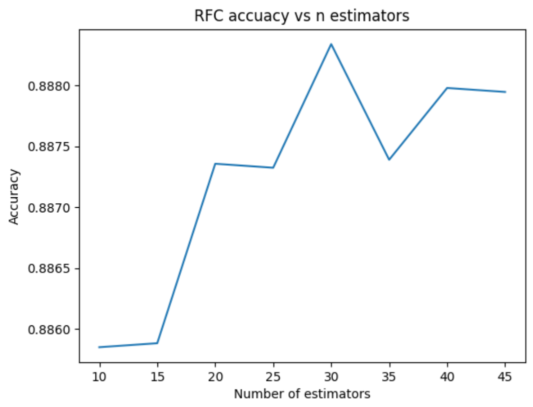
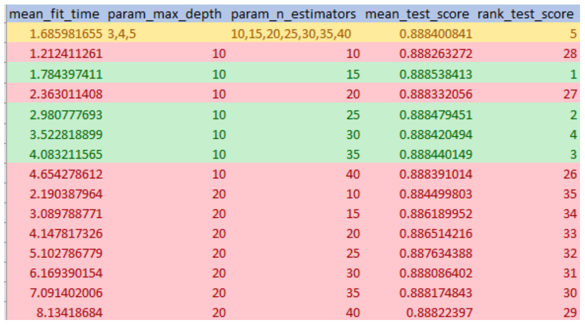


Figure 1

30 estimators yielded the highest accuracy (88.8053%) when not specifying a max depth. Specifying a max depth of three and running it again, the accuracy does not change with the number of estimators and a flat line at 88.7913% is created.

Then, GridSearchCV was run with the options of 10,15,20,25,30,35, and 40 for the estimators and 3,4,5,10, and 20. These combinations were ran through the grid search with 5-fold validation and the results are as shown in *Table 3*.

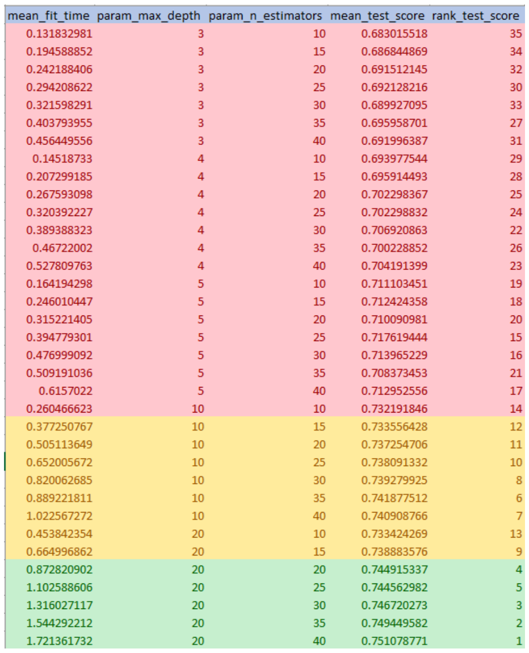


mean_fit_time	param_max_depth	param_n_estimators	mean_test_score	rank_test_score
1.685981655	3,4,5	10,15,20,25,30,35,40	0.888400841	5
1.212411261	10	10	0.888263272	28
1.784397411	10	15	0.888538413	1
2.363011408	10	20	0.888332056	27
2.980777693	10	25	0.888479451	2
3.522818899	10	30	0.888420494	4
4.083211565	10	35	0.888440149	3
4.654278612	10	40	0.888391014	26
2.190387964	20	10	0.884499803	35
3.089788771	20	15	0.886189952	34
4.147817326	20	20	0.886514216	33
5.102786779	20	25	0.887634388	32
6.169390154	20	30	0.888086402	31
7.091402006	20	35	0.888174843	30
8.13418684	20	40	0.88822397	29

Table 3

For the depths 3, 4 and 5 at any number of estimators returned the same accuracy of 88.8401% which is ok compared to some of the other combinations. A bettor estimator is at a max depth of 10 with 15 estimators. This has a mean accuracy of 88.8538% and thus those will be our parameters for the 1:9 no weight random forest classifier model.

2) 1:1 Ratio

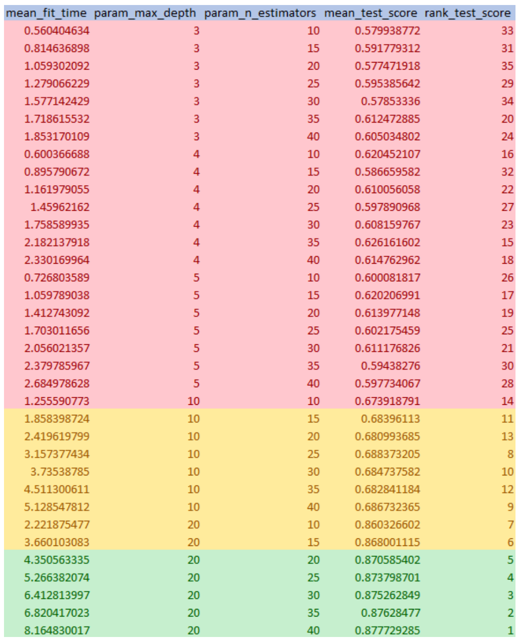


mean_fit_time	param_max_depth	param_n_estimators	mean_test_score	rank_test_score
0.131832981	3	10	0.683015518	35
0.194588852	3	15	0.686844869	34
0.242188406	3	20	0.691512145	32
0.294208622	3	25	0.692128216	30
0.321598291	3	30	0.689927095	33
0.403793955	3	35	0.695958701	27
0.456449556	3	40	0.691996387	31
0.14518733	4	10	0.693977544	29
0.207299185	4	15	0.695914493	28
0.267593098	4	20	0.702298367	25
0.320392227	4	25	0.702298832	24
0.389388323	4	30	0.706920863	22
0.46722002	4	35	0.700228852	26
0.527809763	4	40	0.704191399	23
0.164194298	5	10	0.711103451	19
0.246010447	5	15	0.712424358	18
0.315221405	5	20	0.710090981	20
0.394779301	5	25	0.717619444	15
0.476999092	5	30	0.713965229	16
0.509191036	5	35	0.708373453	21
0.6157022	5	40	0.712952556	17
0.260466623	10	10	0.732191846	14
0.377250767	10	15	0.733556428	12
0.505113649	10	20	0.737254706	11
0.652005672	10	25	0.738091332	10
0.820062685	10	30	0.739279925	8
0.889221811	10	35	0.741877512	6
1.022567272	10	40	0.740908766	7
0.453842354	20	10	0.733424269	13
0.664996862	20	15	0.738883576	9
0.872820902	20	20	0.744915337	4
1.102588606	20	25	0.744562982	5
1.316027117	20	30	0.746720273	3
1.544292212	20	35	0.749449582	2
1.721361732	20	40	0.751078771	1

Table 4

Running the initial RFC with a max depth of 3 and 10 estimator trees, we get an accuracy of 68.9214%. Running search with various number of estimators without specifying a max depth, a peak was at 30 estimators. Running GridSearchCV with the same options as in the first RFC section, the best mean accuracy was 75.1078% with a max depth of 20 and 40 estimators.

3) 1: Ratio with Weights



mean_fit_time	param_max_depth	param_n_estimators	mean_test_score	rank_test_score
0.560404634	3	10	0.579938772	33
0.814636898	3	15	0.591779312	31
1.059302092	3	20	0.577471918	35
1.279066229	3	25	0.595385642	29
1.577142429	3	30	0.57853336	34
1.718615532	3	35	0.612472885	20
1.853170109	3	40	0.605034802	24
0.600366688	4	10	0.620452107	16
0.895790672	4	15	0.586659582	32
1.161979055	4	20	0.610056058	22
1.45962162	4	25	0.597890968	27
1.758589935	4	30	0.608159767	23
2.182137918	4	35	0.626161602	15
2.330169964	4	40	0.614762962	18
0.726803589	5	10	0.600081817	26
1.059789038	5	15	0.620206991	17
1.412743092	5	20	0.613977148	19
1.703011656	5	25	0.602175459	25
2.056021357	5	30	0.611176826	21
2.379785967	5	35	0.59438276	30
2.684978628	5	40	0.597734067	28
1.255590773	10	10	0.673918791	14
1.858398724	10	15	0.68396113	11
2.419619799	10	20	0.680993685	13
3.157377434	10	25	0.688373205	8
3.73538785	10	30	0.684737582	10
4.511300611	10	35	0.682841184	12
5.128547812	10	40	0.686732365	9
2.221875477	20	10	0.860326602	7
3.660103083	20	15	0.868001115	6
4.350563335	20	20	0.870585402	5
5.266382074	20	25	0.873798701	4
6.412813997	20	30	0.875262849	3
6.820417023	20	35	0.87628477	2
8.164830017	20	40	0.877729285	1

Table 5

Running the initial RFC, we get an accuracy of 62.3026%. The best number of estimators with no depth seems to be 40 and running GridSearchCV, we see 20 max depth and 40 estimators gives the best accuracy at 87.7729%.

C. K nearest neighbors

This is a simple classifier with only one optimization and that is the number of nearest neighbors.

1) 1:9 Ratio without Weights

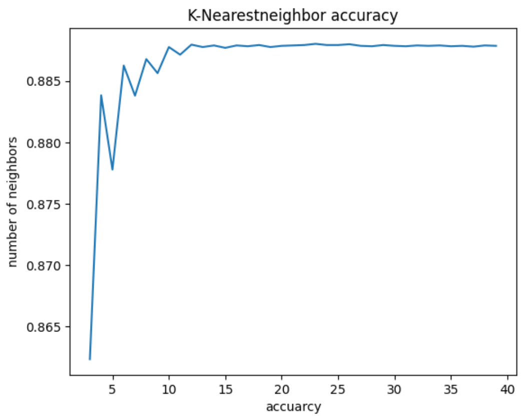


Figure 2

In this classifier, we start out with running Knn with values three through forty for n number of neighbors. The accuracy of those models predictions was then plotted in *Figure 2*. 13 neighbors yields the highest accuracy.

2) 1:1 Ratio

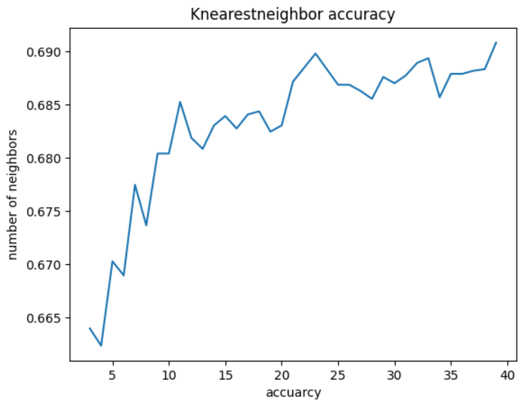


Figure 3

Running the same search as the previous Knns for *Figure 3*, it yielded the highest accuracy at 40 but 23 seems equally comparably at 69.0124%.

II. Analysis and Conclusions

A. Deeper Dive into Accuracies

1) 1:9 Ratio without weights Confusion Matrices

Looking at the accuracies from prior along with the confusion matrices for our 1:9 ratio unweighted models in *Figures 4, 5, and 6*, while a high accuracy rate, the false positivity rate is also high. It seems that the learners started going with a negative result more often because that was rewarded by accuracy. In the case of the MLP, it only predicted 3 postive test correctly out of the 3463 that there were. However accuracy hides that because of the low positivity rate.

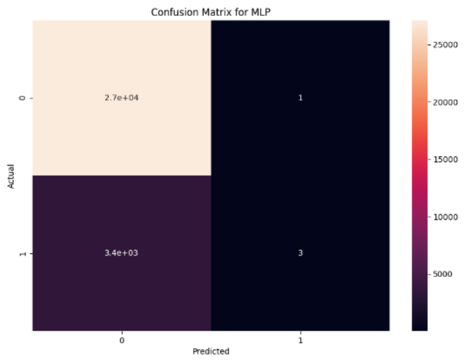


Figure 4

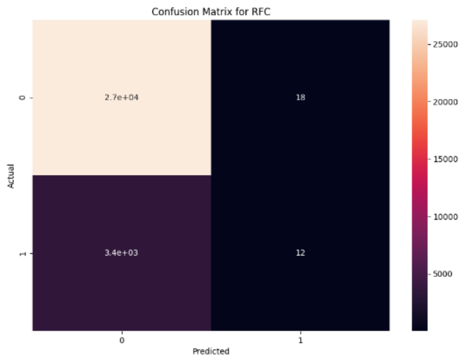


Figure 5

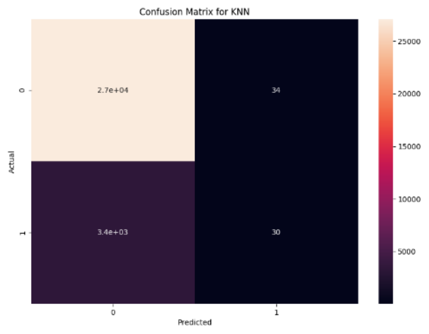


Figure 6

2) 1:1 Ratio Confusion Matrices

Looking at our even distribution models confusion matrices in *Figures 7 ,8, and 9*, these models are catching more patients that would have to return to the hospital with significantly less false negatives. The tradeoff is many more people will be looked at than needed as the false positives increase. This is ok as its only a few hundred more compared to the thousands extra caught early. These are good models with Random Forest Classifier doing the best job.

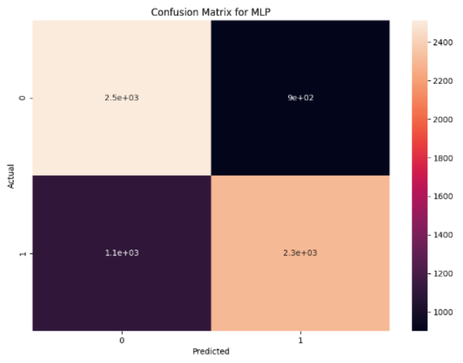


Figure 7

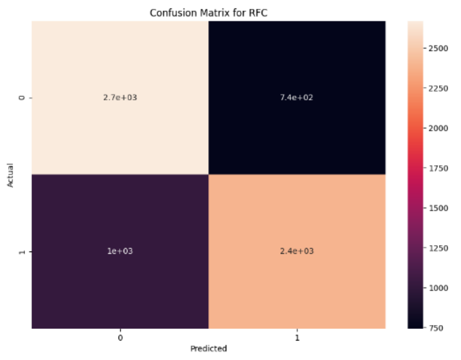


Figure 8

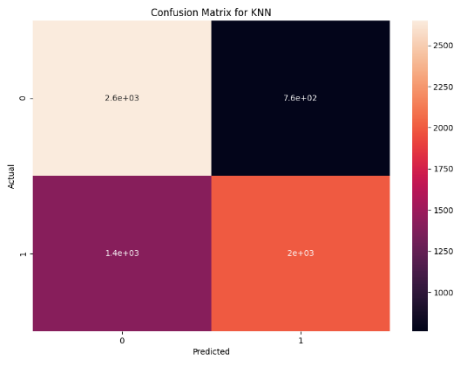


Figure 9

3) 1:9 Ratio with weights Confusion Matrices

Here we take a look at our RFC 1:9 ratio with weights confusion matrix in *Figure 10* and compared to no weights it fared much better actually attempting to classify many as positive compared to before.

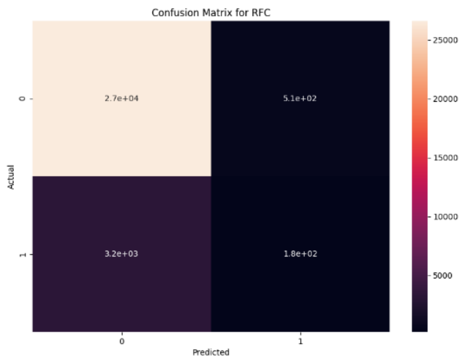


Figure 10

B. *What was Learned*

1) Lessons

Dealing with imbalanced data can be very difficult. For much of this project I thought I had a very good classifier with the full dataset getting 89% accuracy, but that was because 89% of the answers were 0 and it gets 0 for almost all of them. I fi had more time I would like to see how a logistic regression model would handle the class weights and if it would help more. Also checking out some more of the custom activations would have been nice and maybe could've raised the threshold for a negative. I have learned that it is 80% about the preprocessing and 20% models, if even that. Making sure the inputs to the model are what they need to be takes the most time because the real world data isn't as nice as our in class problems.

2) Information

For this data, it seemed that using a balanced distribution was better than using balanced weights, which was better than nothing at all. Using a Random Forest Classifier with an even distribution of 1:1 for the classes works best when trying to find patients that may be readmitted into the hospital within the next 30 days.

APPENDIX A

The Github Repo with my three files of, each slightly different for each dataset run, code can be found at https://github.com/ParfDesai/CSI5810Project1

REFERENCES

[1] https://www.mayoclinic.org/diseases-conditions/diabetes/symptoms-causes/syc-20371444

[2] https://www.cdc.gov/diabetes/library/features/diabetes-stat-report.html

[3] https://www.medicalnewstoday.com/articles/317436#diabetic-ketoacidosis

[4] https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

Final note: I don’t know why my columns are all weird, but they are. I tried to fix them server times but they would just get worse.