|  |
| --- |
|  |
| **Hospital Readmission Prediction for Diabetic Patients** |
|  |

**Anuj JainNeil Raya**

Department of Computer Science Department of Computer Science

Boston University Boston University

Brookline, MA 02446 Allston, MA 02134

*anuj12@bu.edu* *neilraya@bu.edu*

|  |
| --- |
|  |

**Shravanth Venkatesh**

Department of Computer Science

Boston University

*tvsg@bu.edu*

**Abstract**

Diabetes is an issue that is prevalent globally and finding out if there was some parameter that influences the readmission of a patient, could help a lot of people. Our aim with this project was to see if we could make some changes that would help people or even if we found the effectiveness of various algorithms on this kind of data set would be a success for us. Families all over the world spend over billions of dollars every year on diabetic treatment and our aim is to try and give a better answer. We used a kaggle dataset and based our code on a kernel we found along with this dataset.

**1 Data Processing**

**1.1 Loading and Cleaning**

We start by loading our dataset and a second pre-processed dataset, for the second dataset we will talk about the processing in the next section. For our original dataset using the kernel’s pre-processing we start by cleaning the data of ‘?’ and incomplete fields so that we can actually work on our data. Any rows that consist of mainly missing fields or NaN values we decided to drop as it wouldn’t help our results at all.

* 1. **Pre-Processing**

For our main dataset, we follow the kernels processing and start by placing the medicines in an array called keys so that we can access our data to find the number of changes made in each column. The kernel then re-encodes the admission-type, discharge type and admission source into lesser categories so it would easier to access in the future. We then start changing the columns with the type object to have binary values so that we could use those for our algorithms. Columns like gender, race all were a few which were made binary. Columns like age where the age was stored as a string which we decided we should change to an integer as that would be helpful. The kernel also reduced the fields in both the A1Cresult and max-glu-serum from Normal, Abnormal and Not Tested. Moreover, the kernel dropped multiple entries because they could not find any results because of the biases. The kernel looks at the average stay at hospital across multiple encounters. The percentage of the medication changes across multiple encounters. Certain features such as “diagnosis”, for example, was not found to be meaningful while trying to combine multiple categorical values into an array to build a data model. Hence, the kernel then considers first encounter and last encounter individually which works us representations of multiple encounters. This resulted in dataset being reduced to about 70,000 encounters: The code then converts the readmitted parameter to 0 if the readmitted value is greater than 30 or No and to 1 if the readmitted value is less than 30. The kernel then processes some other variables by cleaning, copying data frames and eventually classifying value ranges of these variables to smaller values so it was easier to access later.

* 1. **Pre-Processing Part 2**

Based on our results using the kernel we realized that a few functions like logistic regression and linear regression do not work too well so we decided to alter the pre-processing. Since these classifications work best with integers so we removed a few of the non-integer columns which did not help our results and just added an unnecessary parameter. We checked the values in the columns and figured which made the most sense to drop. We decided not to continuously classify the values into smaller ranges because that sort of reduces the accuracy as it would combine various ranges into binary values. We also do not use the diagnoses fields as it did not add any value to our result and our results turned out to be better without it. We also used the basic cleaning and dropping mentioned in the processing above. We use kurtosis in both the pre-processing to remove the skewness of our data i.e. the noise. We convert age to ranges dividing as the mid-point values of incrementing tens which changes how he classified age earlier from 1-8. We use the kernels pre-processing to convert certain classes as integers based on the values in the column. We then normalize our data as we want our final training set values to be between 0-1 which is what is compared for accuracy. We divide our feature sets separately based on logistic and linear regression which uses the feature set from our second file. For the other functions we use the kernels feature sets with integers and with everything. This is where we began our modelling to check how our data points fared.

1. **Modeling**

The kernel had already implemented logistic regression, decision tree, and random forest to classify the data. The decision tree and random forest gave excellent results, with 92% accuracy for the decision tree and 94% accuracy for the random forest. As stated earlier, we used a different preprocessing method when training the logistic regression model which yielded much more relevant results than the implementation from the kernel. This is because we did not cut the diagnostic data into smaller bins, giving us no loss of generality.

To stop minority classes from being overshadowed, we used a Synthetic Minority Oversampling Technique (SMOTE) provided by the imblearn. This function works by first taking the nearest neighbors of a minority class to determine if it is in danger of being overshadowed by another much larger feature. If oversampling is needed, it inserts extra points to the minority class on the lines between the points. This makes smaller features have more of an effect on the result, which gave us more reasonable answers. Our data set had many features that were mostly zeros but would have a significant effect on prediction.

The models that we implemented are linear regression, SVM with linear kernel, SVM with polynomial kernel, SVM with RBF kernel, SVM with sigmoid kernel, and a multilayer perceptron classifier. We used the scikit-learn implementations for each one. For linear regression, we omitted the preprocessing step of grouping the diagnostic data into smaller bins to achieve a more accurate regression. We omitted the SVMs with linear and polynomial kernel; the linear kernel took a long time to converge and could not outperform the linear regression prediction. The polynomial kernel SVM ran very quickly with high degree, but gave us 50% accuracy, 50% precision, 100% recall. With a degree of 8 or lower, we could not get the polynomial kernel SVM to converge. SVM with RBF or sigmoid kernel gave much more relevant results.

After speaking to the TF, we got the idea to try a multilayer perceptron. This performed the best out of all the models we tested.

1. **Results and Analysis**

Decision Tree, Random Forest and Logistic Regression were done previously and taken from Kaggle [2].

Decision Tree took a singular root and made decisions based on that root and therefore ended up coming with Really good accuracy but took number of medications (Figure.1) as the top priority variable, which is correct, but insulin as taken as the least important variable which was not right is the real world. However, Random Forest gave a good accuracy and correctly predicted that insulin was the major factor in readmission (Figure. 2)of the patients given their medical history.

We made changes to the dataset that affected the logistic regression’s outcome significantly and make it have a lesser accuracy but we were able to get the exact same results for linear regression as well which we felt makes sense because we are predicting binary results which means log curve is fitted from 0-1.

Linear Regression was average result given that it was a 50-dimensional space and that the parameter space was not binary, only the result space was binary (Table 1).

Support Vector Machine with Radial Basis Function (Gaussian) without over-sampling technique to change the imbalances we got a good accuracy but with bad precision and recall(Over-sampled data did not converge in the limited time we had).

Support Vector Machine with Sigmoid function without over-sampling technique to change the imbalances again we got a good accuracy but low precision and recall which was a direct result of not over sampling as this kernel did not converge in the limited time we had.

Multi-Layer Perceptron was our best model (Table 1)given that the dataset had a good amount of data and instead of the convolutional neural network which we planned to work with from the beginning.

**4       Conclusion**

Even though our results were not the most accurate we still learned a lot by just seeing how the kernel was implemented and we realized there is a few things we could have improved if we had more time. We would have done the pre-processing more differently and would have tried using different class types and maybe other ideas. We were limited because of the time shortage and with our data being 10 years old at the earliest. If we had more complete fields and more recent data, we could have done more and probably gotten more relevant data. Overall, this project was a great learning experience for us and it taught us a lot of new things which was really helpful.

**5.1 Figures**

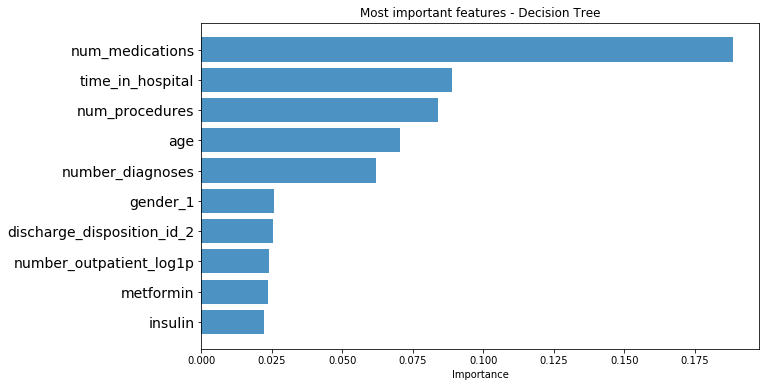


Figure.1: Decision Tree Results

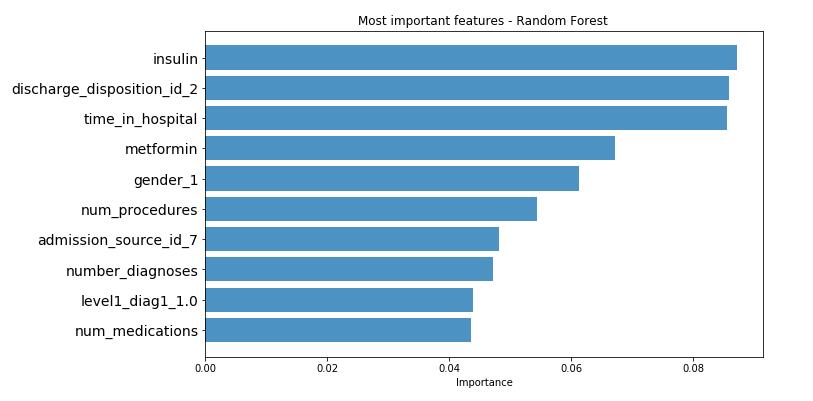


Figure.2: Random Forest

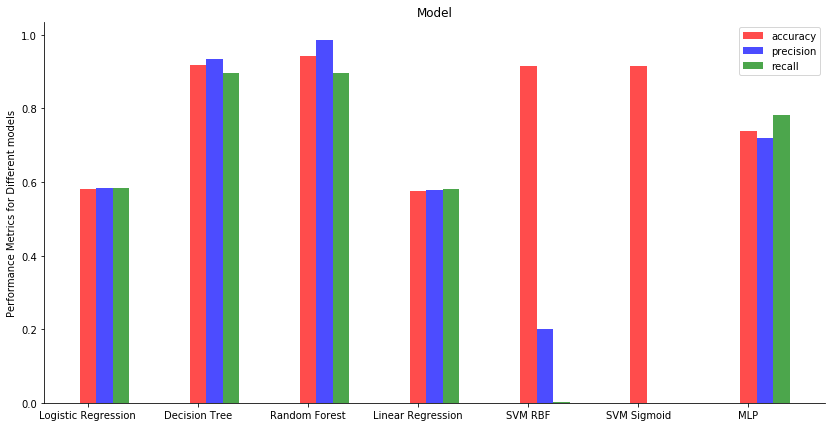
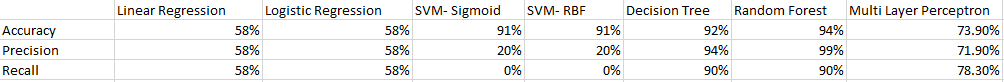
****

Figure 3: Model Performance Graph

**5.2 Table**

Table.1: Accuracy, Precision and Recall Tabulation



**6 Citations**

[1] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records,” BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.

[2] Prediction on Hospital Readmission. (n.d.). Retrieved June 28, 2019, from <https://www.kaggle.com/iabhishekofficial/prediction-on-hospital-readmission>

**7 Acknowledgements**

[a] Ryan Yu, Teaching Fellow, ryu1@bu.edu

[b] Gavin Brown, Teaching Fellow, grbrown@bu.edu