Hospital Readmittance: A Classification Model Analysis Using Diabetic Patient Data

Andrea Clark[1] and Jonathan Yakubov[2]

Abstract— **This paper extends the analysis presented in Strack et. al. [1] titled *Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records*, by implementing three different classifiers to determine whether or not a diabetic patient was to be readmitted to hospital care based on a selection of attributes from the patient diabetes dataset contributed by the Health Facts (Center Corporation, Kansas City, MO), publicly available on the UCI Machine Learning Repository [2].**

I. Introduction

The motivation behind this work came from a research article titled **Impact of HbA1c Measurement on Hospital Readmission Rates:Analysis of 70,000 Clinical Database Patient Records**. The objective of the article was to examine historical patterns of diabetes care in patients admitted to a US hospital and determine what could lead to improvements in patient safety. Strack et. al hypothesized that the measurement of HbA1c (a measure of glucose control) was associated with a reduction in readmission rates in individuals admitted to the hospital [1]. The data for the study was obtained from the Health Facts database (Center Corporation, Kansas City, MO), a national data warehouse that records clinical records across hospitals throughout the US. The group's conclusion suggests that a measurement of HbA1c for diabetes patients was a strong indicator of hospital readmission. The objective of our study was to understand whether diabetes patients could be classified as Readmitted or Not-readmitted based on the attributes provided in the data. Although Strack et. Al emphasized the statistical relationship between HbA1c and Readmittance, our objective was to build a classification model for the data set with the provided attributes. We implemented three classification models; Support Vector Machines (SVM), Bayes' Classifier, and Back-propagation supported Neural Networks, to examine the accuracy provided by our models. We implemented a Random Forest Regressor to do a feature selection and extract the most vital attributes in the data. The results presented suggests that the attributes in the data are not sufficient to classify patient hospital readmission as the accuracies rendered from the models were low.

II. Data Cleaning and Preprocessing

A. Cleaning the Data

The first step in our analysis was first getting familiar with the attributes and the target categories in the data. The data set, provided by the Center for Clinical and Translational Research, Virginia Commonwealth University [2], was collected over a period of ten years (from 1999 to 2008) of clinical care at 130 US hospitals and integrated delivery networks. The data set included 55 features, many of which were eliminated from our consideration when conducting the analysis. There were a few attributes in the data that were eliminated from the onset, since they had a high percentage of missing values. These features were weight (97% missing), payer code (40% missing), and medical specialty (47% missing). Furthermore, the features `number_outpatient`, `number_emergency`, `number_inpatient` and all the features pertaining to medication names were also excluded, as these were not considered relevant to the classification analysis. In addition, entries that were missing attribute data were eliminated from the analysis by simply removing the entire record.

B. Preprocessing

The data was further processed to include only the first patient encounter, which was determined by finding duplicate entries in the `patient_number` attribute. Additionally, we removed all encounters that resulted in either discharge to a hospice or patient death, as was done for the analysis in the Strack et al. (2014) paper [1]. This was done to avoid biasing the analysis. After performing the above-described operations, we were left with 69,789 entries with 22 attributes. In order to facilitate the implementation of our models, the attributes were converted into either integer or categorical types, depending on whether or not the data was nominal or numeric in nature. Lastly, as the SVM model uses primarily binary classes for classification, the `readmission` target class was compounded from three to only two target categorizations: 'YES' and 'NO', for readmission and no-readmission, respectively.

C. Feature Selection

Our initial preprocessing of the data was able to narrow down the initial 55 attributes to 22. However, this number of attributes is still relatively large, and may be introducing unwanted noise to our classifier. In order to render our models more accurate, we used a Random Forest Regressor to help narrow down the features that account for the most variability in our data. The basic premise of a Random Forest Regressor is that instead of searching for the most important feature while splitting a node, the algorithm searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. The Random Forest Regressor model was implemented using SkLearn, which implements the algorithm by examining how many nodes in the tree use a particular feature, which reduces the impurity of that feature across all trees in the forest. A score is automatically assigned for each feature after training and then are scaled such that the sum of all the features' importance is equal to 1. The following ten features, followed by their respective importance from most important to least, were used to train our models:

- Number of Lab Procedures (0.131)
- Diagnosis 2 (0.1162)
- Diagnosis 3 (0.114)
- Diagnosis 1 (0.1133)
- Number of Medications (0.0993)
- Time in Hospital (0.0694)
- Medical Speciality (0.0533)
- Age (0.0505)
- Number Procedures (0.0416)
- Number Diagnoses (0.0393)

It is important to note that while the model yields these features as the most relevant, these features are not necessarily the most useful for optimizing the models we want to train. A domain expert/physician could give better insight into what factors would reliably affect patient readmission from a biological perspective, which would likely lead to higher model performance. We believe that those relevant attributes were not all included in the data set.

III. Classification Models

We implemented three classification models to examine the data: a naïve Bayes' classifier, a support vector machines (SVM), and a neural network trained using backpropagation. With the initial implementation of the models, we observed overfitting, as the non-readmitted class-label was the most frequent in the data. As a result, we implemented an over-sampling and under-sampling technique to allow the models to learn and classify the data more easily. While these yielded slightly better accuracies, it was noted that the myriad of attributes were not optimal for classification. This is when a Random Forest Regressor was used to extract the most relevant features from the 22 attributes we were working with. We noticed that this resulted in slightly higher accuracies, and yielded better results than the over-sampling/under-sampling techniques previously implemented. Therefore, the over-sampling/under-sampling techniques were not used for the analysis and rather the most relevant attributes extracted by the Random Forest Regressor were used for the training of the models.

A. *Naïve Bayes' Classifier*

The SkLearn Python library was used to implement the naïve Bayes' classifier. [3] Before the model was trained on the data, the LabelEncoder class was introduced to transform the categorical features into numerical categories, since the model can only handle numeric attributes. The features extracted by the Random Forest regressor were used to the analysis. These features included `Num_Lab_Procedures`, `Diag_1`, `Diag_2`, `Diag_3`, `Num_Medications`, `Time_in_Hospital`, `Medical_Specialty`, `Age`, `Num_Procedures`, and `Number_Diagnoses` The data was split to a 40% test set and a 60% training set. The model we implemented assumed a Gaussian distribution to represent the class-conditional probability for the attributes. A naïve Bayes' classifier works by estimating the class-conditional probability by assuming that the attributes are conditionally independent, given the class label y. [4] The conditional independence assumption can be stated formally as follows:

$$P(\mathbf{X} \mid Y = y) = \prod_{i=1}^{d} P(X_i \mid Y = y)$$

where each attribute set $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ consists of d attributes. Assuming conditional independence, instead of having to compute the class-conditional probability for every combination of attributes, $\mathbf{X}$, it suffices to estimate the conditional probability of each X_i given Y. This assumption reduces the number of computations required to estimate the probability. The posterior probability for each class Y is as follows:

$$P(Y \mid \mathbf{X}) = \frac{P(Y) \prod_{i=1}^{d} P(X_i \mid Y)}{P(\mathbf{X})}$$

Since $P(\mathbf{X})$ is fixed for every class Y, it is sufficient to choose the class that maximizes $P(Y) \prod_{i=1}^{d} P(X_i \mid Y)$. [5] The results from running the naïve Bayes' classifier are shown in the figures below:

```
Accuracy: 0.6087548359363806
Precision: 0.458319550115779
Recall: 0.26598195430984833
              precision    recall  f1-score   support

           0       0.65      0.81      0.72     17498
           1       0.46      0.27      0.34     10418

   micro avg       0.61      0.61      0.61     27916
   macro avg       0.55      0.54      0.53     27916
weighted avg       0.58      0.61      0.58     27916
```

Fig. 1. Naïve Bayes' Metrics

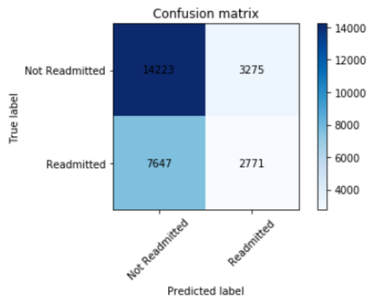


Fig. 2. Naïve Bayes' Confusion Matrix

This classifier performed surprisingly well, despite its "naïve" assumption of class-conditional independence. While this independence assumption may overlook potential hidden dependencies within the attributes, assuming that the attributes are independent yields a computationally more tractable model. The naïve Bayes' classifier yielded a 61%

accuracy, with a precision of 46% and recall of 27%. The accuracy here represents how many times the classifier correctly predicted a patient being readmitted, ‘YES‘ and not readmitted, ‘NO‘, which can be inferred from the diagonal entries in the confusion matrix. However, the classifier was not very precise, that is, its ability to correctly predict a patient being readmitted is not very high. This is supported by the low recall rate, which specifies the rate at which the classifier is correctly predicting a patient being readmitted.

B. Support Vector Machines

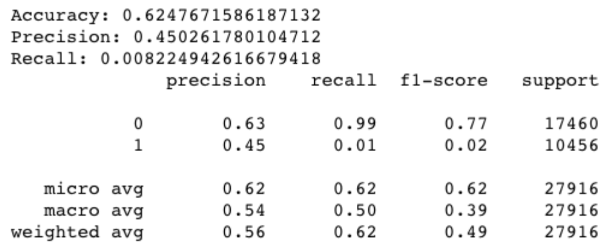
The SkLearn Python library [3] was imported to implement the SVM model. The LabelEncoder class was introduced to transform the categorical features into numerical categories. This was useful when training the data. The results returned from the Random Forest Regressor influenced our decision to use the first 10 most relevant features in the training of the data. These features included `Num_Lab_Procedures`, `Diag_1`, `Diag_2`, `Diag_3`, `Num_Medications`, `Time_in_Hospital`, `Medical_Specialty`, `Age`, `Num_Procedures`, and `Number_Diagnoses`. Given that an SVM is a supervised learning algorithm, the class-label ‘Readmitted‘ had two possible results for each data object, which were either ‘YES‘ or ‘NO‘. A Support Vector Machine works by determining the hyperplane that maximizes the distance between the two distinct classes. The support vectors are the data objects that have an equal measure of distance from opposite sides of the hyperplane, i.e. from both sides of the classes to the hyperplane. Since our dataset was not linearly separable, we implemented the SVM with a radial basis function kernel to determine the non-linear decision boundary in a higher dimensional feature space. This ‘kernel trick‘ is a method used for computing similarity in the transformed vector space using the original attribute set. Taking the dot product between two input vectors $\mathbf{u}$ and $\mathbf{v}$ in a transformed vector space, we can express this dot product in terms of a similarity function in the original space, given as follows:

$$K(\mathbf{u},\mathbf{v}) = \phi(\mathbf{u})\cdot\phi(\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^2$$

The similarity function , K, which is computed in the original attribute space is known as the kernel function. It is not necessary to know the exact form of the mapping function ϕ because the kernel function must satisfy Mercer’s theorem, which asserts that kernel functions can always be expressed as the dot product between two input vectors in some higher-dimensional space. [5] For completeness, the similarity function for the radial basis function is given below:

$$\begin{aligned} K(\mathbf{u},\mathbf{v}) &= \phi(\mathbf{u})^\top\phi(\mathbf{v}) \\ &= \exp\Big(-\gamma\|\mathbf{u}-\mathbf{v}\|^2\Big),\ \gamma > 0 \end{aligned}$$

The data was split to a 40% test set and a 60% training set. After training the model on the top ten most relevant features given by the Random Forest regressor yielded an accuracy 62% with a precision of 45% for the ’YES’ readmitted label.



```
Accuracy: 0.6247671586187132
Precision: 0.450261780104712
Recall: 0.008224942616679418
              precision    recall  f1-score   support

           0       0.63      0.99      0.77     17460
           1       0.45      0.01      0.02     10456

   micro avg       0.62      0.62      0.62     27916
   macro avg       0.54      0.50      0.39     27916
weighted avg       0.56      0.62      0.49     27916
```

Fig. 3. SVM Metrics

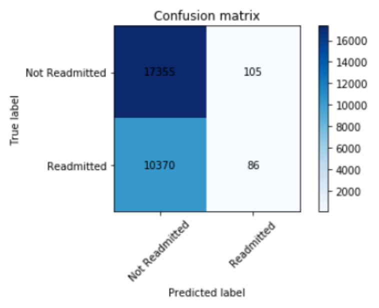


Fig. 4. SVM Confusion Matrix

C. Neural Network

A neural network model was implemented using Sklearn’s LabelEncoder to convert categorical attributes to their respective numeric values. [3] The implementation of the Neural Network required installing the Keras [6], TensorFlow [7], and Theano [8] libraries. The data was split to a 40% test set and a 60% training set. The attributes used for the training set were determined from the feature selection utilized of the Random Tree Regressor. These ten attributes are the same from the previous models. The model was initialized using the Sequential class from the Keras library. Thereafter, the input layer was defined with the ten attributes making up the 10 nodes. The network used three hidden layers, where each had 20 nodes. A basic framework for this network is pictures below [9]:

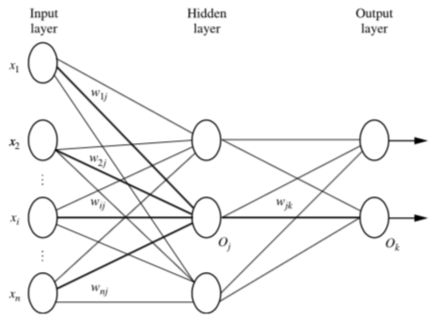
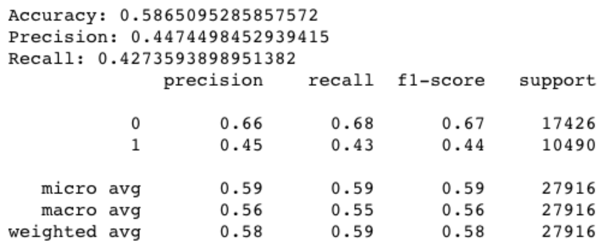


Fig. 5. A multilayer, feed-forward neural network

The activation function used for both the input layer and the hidden layer was the *rectified linear unit* (ReLu) function. The *sigmoid* activation function was used for the output layer, as the classification was a binary one between ‘YES‘

for readmission and 'NO' for no readmission. The "Adam" method was implemented as the optimizer for the stochastic gradient descent step, which simply optimizes the weights and bias of the neural network during back-propagation. The logarithmic loss function implemented was the "binary-crossentropy". Once the data was trained, the predictions were made on the test data. The accuracy of the model was about 59% and the precision for the value 'YES' in class-label 'Readmitted', was 44%. It was concluded again, that the accuracy was not optimal because the features recovered from the feature selection were not suited for classification of diabetes patients in hospital readmission. As previously discussed, seeking physician expertise and incorporating more relevant features into the data set would have increased the accuracy.



```
Accuracy: 0.5865095285857572
Precision: 0.4474498452939415
Recall: 0.4273593898951382
              precision    recall  f1-score   support

           0       0.66      0.68      0.67     17426
           1       0.45      0.43      0.44     10490

   micro avg       0.59      0.59      0.59     27916
   macro avg       0.56      0.55      0.56     27916
weighted avg       0.58      0.59      0.58     27916
```

Fig. 6. Neural Network Metrics

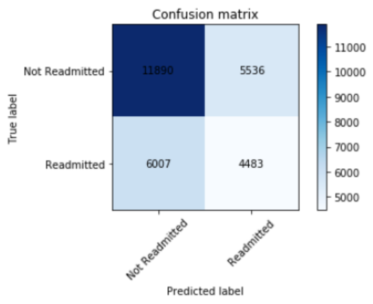


Fig. 7. Neural Network Confusion Matrix

D. Apriori Algorithm

The Apriori rule-mining algorithm was used to extract frequent itemsets from the data set. We were interested in analyzing whether there were any strong relationships between various features in the data that could potentially be used for classification purposes. The Apyori Python library was implemented to extract the frequent itemsets of the data. [10] We observed 3 items as the maximum k in the extraction process using a minimum support of 0.5 and a maximum length of 5 items. Features such as `Caucasian`, `Max_Glu_Serum: None`, `DiabetesMed: Yes` were the most frequently-appearing. However, the frequent itemsets of the features were not sufficient for classification purposes. We tried to implement the CBA classification method using the recently introduced PyARC Python library based on association rules. [11] However, the itemsets were not discriminate enough for classification. Additionally, the information gain was not very high for these itemsets. As a result, the CBA algorithm was not employed for classification purposes. Nonetheless, it is interesting to note the different frequent sets of the data.

IV. Conclusions

Of the given models implemented, the Support Vector Machine (SVM) yielded the highest accuracy, at about 62%. The accuracy across all three models was not very high, but this can be explained by the fact that the attributes provided in the data were not optimal for hospital re-admittance classification purposes. For future work, if the models were to be used again, we would try to seek the expertise of physicians and obtain data more directly linked to diabetic patients' hospital readmission. Thereafter, we would run the Random Forest regressor to do a feature selection. We believe that this would help increase the accuracy and the precision of the 'Readmitted' class in the data. Implementing various models have shown that the accuracy does not necessarily increase, and supports our theory that the the data did not have the appropriate attributes to build a high-performing model. The `weight` attribute, for instance, would likely be a strong predictor for diabetes and a potential hospitalization factor, yet this attribute was far too sparse in our dataset to be included in our analysis. It would be a helpful and interesting exercise to run our models on another diabetic patient dataset with a slightly more concise feature set but a more robust population sample. It would be beneficial for physicians to have such a model, such as the ones we trained, to predict future readmission of patients, and plan patient care accordingly. In order for this to be feasible, we recommend that future data sets record other aspects of patients that directly correlate to hospital readmission.

References

[1] B. Strack, J. Deshazo, C. Gennings, J. L. Olmo Ortiz, S. Ventura, K. Cios, and J. N Clore, "Impact of hba1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records," *BioMed research international*, vol. 2014, 2014.
[2] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[4] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001. [Online]. Available: http://www.scipy.org/
[5] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson Education, 2006.
[6] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io
[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, and G. S. Corrado, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.
[8] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, and N. Ballas, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, 2016. [Online]. Available: http://arxiv.org/abs/1605.02688
[9] J. Han, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2005.
[10] Y. Mochizuki, "Apyori," 2016.
[11] J. Filip, "pyarc," 2018.