Sabriya Alam & Parker Crain

ECE 20875

Final Project

December 6, 2019

Personalized Healthcare Data Analytics

Introduction

The data set analyzed for this project was a labelled dataset detailing the medical histories of a set of patients. It included details about their lifestyles, medical conditions, and other personal characteristics. The parameter of interest was whether or not a patient was readmitted to the hospital after their release. The goal was to create predictive models to relate these input factors of a patient's medical history and lifestyle details to their likelihood of being readmitted. The created models were then evaluated to analyze whether such a correlation existed, and to understand how useful a predictive model may be in this context. If such a model could be created, it was determined whether particular characteristics were more influential in making this prediction than others.

Data Description

The data set acquired is one that simulates real patient data, with factors recorded including:

- Length of most recent hospital visit
- Number of hospital visit
- Age
- Whether or not this is a readmission
- Admission from emergency department
- Height / weight
- Sex
- Marital status
- Zip code
- Health insurance provider
- Attending doctors
- Smoking habits
- Diabetes

- Psychological conditions (anxiety, depression, etc.)
- Alcohol and substance abuse history
- Month and year of hospital discharge
- Readmission status and time elapsed between readmissions

In total, there are 89 recorded categories for each patient, with several characteristics being one-hot encoded (e.g. zip code, marital status). The output variable analyzed was readmission status, recorded as 1 if the patient was readmitted, and 0 if he or she was not. As provided, the continuous variable characteristics such as height, weight, and of each patient were already normalized. The rest of the data was represented as binary values 1 and 0.

Basic Data Analysis Models

Initially, some fundamental models were created to explore the data and see if any correlations could be detected between the input and output factors of this data set via simple predictive modeling. The models implemented included linear and logistic regression, linear discriminant analysis, decision trees, and random forests. The original data was split into training and testing sets, and the mean squared errors and variance scores were calculated based on the actual outcomes of the testing data and the outcomes predicted by the models. If a strong relationship fitting these models could be determined for this data set, one would expect to see low mean squared errors for predicted values and high variance (R^2) scores.

The results of running the models are summarized below:

Linear Regression

Linear Regression Mean squared error: 0.21

Linear Regression Variance score: 0.16

Logistic Regression

Logistic Regression Mean squared error: 0.33

Logistic Regression Variance score: -0.33

Linear Discriminant Analysis

Linear Discriminant Analysis Mean squared error: 0.33

Linear Discriminant Analysis Variance score: -0.33

Decision Trees

Decision Tree Mean squared error: 0.34

Decision Tree Variance score: -0.37

Random Forests

Random Forest Mean squared error: 0.34

Random Forest Variance score: -0.37

The models created above all indicate relatively high mean squared errors and low variance scores. Based on this evaluation, it is unlikely that these models are good choices to make predictions on this data set. The lack of any strong correlation here also casts some doubt as to whether a predictive relationship actually exists in this data set, so to explore this further, a neural network model was created to see if improved predictions could be made. Neural networks are able to find relationships in data that are more complex than linear correlations. Multilayer networks can be used to represent convex regions in higher dimensional space, which can lead to more effective classification models for complex data. If a neural network model can be used to make reasonable predictions for this data, one would expect to see a high accuracy rate on testing data, and a plotted ROC curve would have an area underneath it close to 1 to indicate a robust classifier.

Neural Network Model

In order to create a neural network, a Keras model was defined and compiled, and then fit with the training dataset. The model was evaluated, saved, and then used to make predictions on testing data. A summary of the model was output as well for ease of analysis.

Initially, a sequential Keras neural network model was created with an input layer of all 89 features of the data, one fully-connected hidden layer with a few nodes, and one output layer. This model yielded an accuracy rate of 77%, but it was found that the model was simply predicting a negative output regardless of input values, and since 77% of the data had negative outputs, there was a high accuracy rate despite the model not actually predicting anything useful. This was to be expected, since such a simplistic model could not capture the scope, structure, and complexity of this dataset. To remedy this, a few steps were taken:

- Create evenly split (positive/negative outcome) training and testing data
- Tune the model to be able to handle more complex data by adding more layers and nodes, and adjusting the number of epochs
- Speed up model generation by decreasing the total number of features without losing variation in the data
    - Principal Component Analysis
    - Lasso Regression
- Generate an ROC curve to evaluate the predictive power of the model

Data Preprocessing

Some preprocessing became necessary in the case of building a neural network model that had not been necessary when building regression models. To ensure that the model was not biased toward one output over another solely because the training data set was more heavily weighted for one class, the original data set was divided into one training set and one testing set, each of which had 50% positive output and 50% negative output as the variable of interest.

Additionally, since generating the model was found to be a time-consuming task, it was determined that reducing the number of features of the data set could improve this performance. To this end, initially, principal component analysis was performed to synthesize a lower-dimension matrix that could still preserve 95% of the variance in the original data. This brought the new number of input variables down to 60, which did indeed speed up the process of generating a new model. The accuracy level of the model stayed nearly the same with this new input matrix, which led to the conclusion that data variance had indeed been preserved with fewer dimensions.

As an alternative to principal component analysis, lasso regression was also implemented to help find non-unique features that may be correlated. Lasso regression penalizes non-zero coefficients in

order to find a minimizing set of features. When lasso regression was performed on the raw data, the following coefficients were returned:

```
[-0.        0.03442898  0.        0.        0.       -0.0004045
 -0.       0.       -0.       -0.       -0.        0.
 -0.       0.       -0.        0.        0.        0.
 -0.       0.       -0.       -0.       -0.        0.
  0.      -0.       -0.        0.        0.        0.
 -0.      -0.        0.        0.        0.        0.
 -0.       0.       -0.       -0.        0.        0.
  0.       0.       -0.        0.       -0.        0.
  0.       0.        0.        0.       -0.        0.
 -0.       0.       -0.       -0.        0.       -0.
 -0.      -0.        0.        0.       -0.        0.
 -0.      -0.       -0.        0.        0.        0.
  0.       0.       -0.       -0.       -0.       -0.
  0.       0.        0.        0.       -0.       -0.
 -0.       0.        0.       -0.       -0.       ]
```

As seen above, almost all 89 of the coefficients returned are 0 or very close to 0. This outcome is generalized to conclude that no linear combination of any subset of the regressors will be useful in generating predictions. In the end, the neural network model was ultimately trained with the 60 synthesized features generated by principal component analysis.

Tuning the Model

To tune a neural network model for higher accuracy, it is recommended that the number of layers and number of nodes in each layer be increased. Generally, deeper neural networks perform better. The complexity of the model in this regard is typically determined through experimentation.

In our model, all the layers are densely connected, so all the neurons in one layer are connected to those in the next layer. Activation functions, which are responsible for transforming summed weighted inputs of a node into a defined output, were assigned to each layer. Non-linear activation functions improve models by finding more complex structures in data, so in this case, a combination of sigmoid functions (logistic, so any input less than 0 is 0, and any input greater than 1 is 1, while anything in between retains its value) and rectified linear unit functions (0 for any input value below 0 and a retained value for anything greater than 0) were utilized.

The last model optimization technique explored was to adjust the number of epochs of the model. In an iterative process, one epoch is defined as when the entire dataset is passed forward and backward through the neural network once. One epoch updates the weights of the nodes once (using gradient descent and backpropagation to minimize errors in prediction), and depending on the diversity of the dataset, the number of epochs determines whether the overall model is underfit or overfit to the data.

Model Evaluation, Predictions, and Visualization

Some metrics utilized to evaluate the output of a model include precision (the number of true positives divided by the total number of elements labeled as belonging to the positive class) and recall (the number of true positives retrieved over the total number of positives). Visually, a receiver operating characteristic (ROC) curve plots the true positive rate against the false positive rate at various threshold settings, which allows it to illustrate the overall capability of a binary classifier. A classifier that does an excellent job of discerning between two classes has an ROC curve that tends toward the upper left-hand corner of the graph, while a classifier that is no better than random guessing has an ROC curve that is close to the diagonal. Additionally, the closer that the area under the ROC curve is to 1, the better the classifier is.

In this case, after tuning the model as specified above (increasing the number of layers and nodes per layer, and increasing the number of epochs), the output of the model was found to have

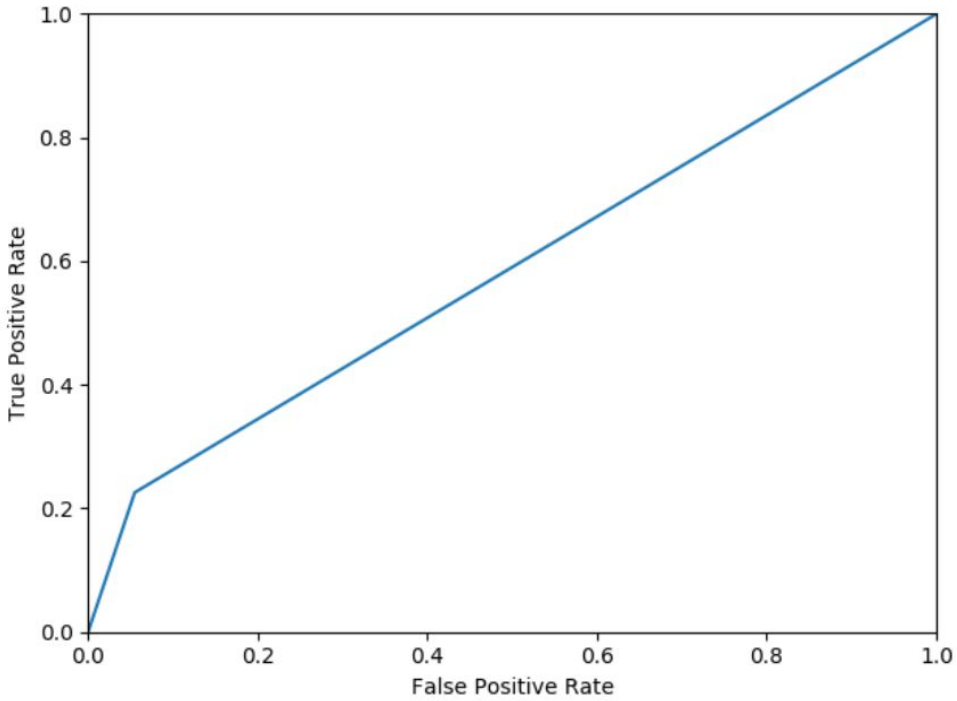
improved in accuracy. The generated ROC curve also indicates that the classifier had an improved performance compared to the initial model. Details of this are outlined below.

<u>Original model summary</u> (using equally weighted data without PCA, prior to tuning neural network model):

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                900
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 88
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9
=================================================================
Total params: 997
Trainable params: 997
Non-trainable params: 0
_________________________________________________________________
```

acc: 60.02%

ROC Curve (prior to model tuning)



AUC Score = 0.5853871128871129

Classification Summary

	precision	recall	f1-score	support
0.0	0.55	0.94	0.69	1000
1.0	0.80	0.23	0.35	1001
accuracy			0.59	2001
macro avg	0.68	0.59	0.52	2001
weighted avg	0.68	0.59	0.52	2001

Confusion Matrix:

[[974 26]
[835 166]]

Interpretation

If the model says the patient will not be readmitted, probability that they will actually not be readmitted is .55

If the model says the patient will be readmitted, the probability that they will actually be readmitted is .80

If the patient is not readmitted, the probability of the model telling me so is .94

If the patient is readmitted, the probability of the model telling me so is .23

Confusion matrix indicates that of the 2001 data points used for testing, there were 974 true negatives, 166 true positives, 26 false positives, and 835 false negatives.

<u>Final model summary</u> (using fully preprocessed data, tuning model with more layers/nodes/epochs):

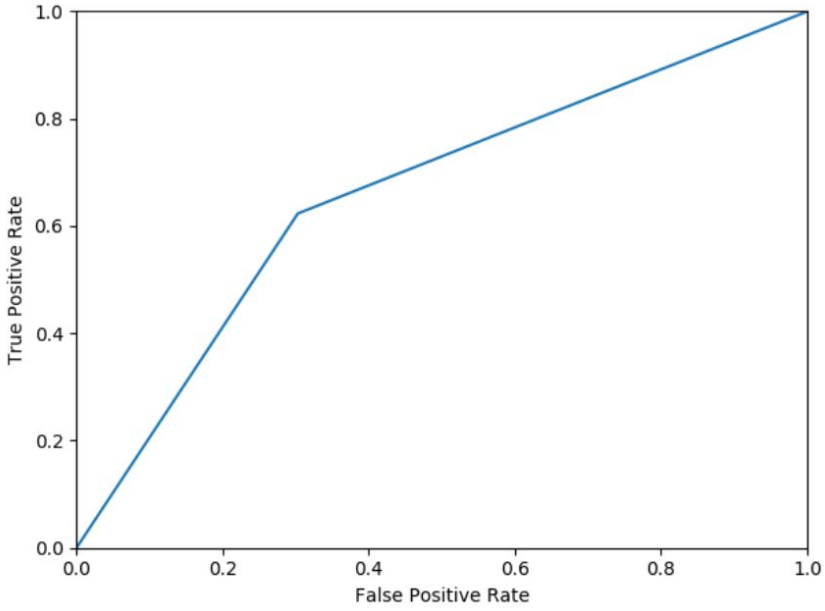
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 30)	1830
dense_2 (Dense)	(None, 20)	620
dense_3 (Dense)	(None, 20)	420
dense_4 (Dense)	(None, 15)	315

dense_5 (Dense)	(None, 15)	240
dense_6 (Dense)	(None, 15)	240
dense_7 (Dense)	(None, 10)	160
dense_8 (Dense)	(None, 10)	110
dense_9 (Dense)	(None, 10)	110
dense_10 (Dense)	(None, 15)	165
dense_11 (Dense)	(None, 8)	128
dense_12 (Dense)	(None, 5)	45
dense_13 (Dense)	(None, 1)	6

Total params: 4,389
Trainable params: 4,389
Non-trainable params: 0

acc: 67.28%

ROC Curve (after tuning model)



AUC score = 0.6601883116883117

Classification Report

---

	precision	recall	f1-score	support
0.0	0.65	0.70	0.67	1000
1.0	0.67	0.62	0.65	1001
accuracy			0.66	2001
macro avg	0.66	0.66	0.66	2001
weighted avg	0.66	0.66	0.66	2001

Confusion Matrix:

[[697 303]

[377 624]]

Interpretation

If the model says the patient will not be readmitted, probability that they will actually not be readmitted is .65

If the model says the patient will be readmitted, the probability that they will actually be readmitted is .67

If the patient is not readmitted, the probability of the model telling me so is .70

If the patient is readmitted, the probability of the model telling me so is .62

Confusion matrix indicates that of the 2001 data points used for testing, there were 697 true negatives, 624 true positives, 303 false positives, and 377 false negatives.

As seen from the plots and metrics above, tuning the neural network model resulted in improved accuracy and a classifier that is better at discerning between the positive and negative classes. The area under the ROC curve, which is closer to 1 as performance improves, increased from 0.585 to .660. Moreover, the accuracy of predictions on the same testing data set increased from 60.02% to 67.28%.

The confusion matrix of the tuned model indicates that the proportion of true positive and true negative predictions are about the same, and shows improved performance compared to the initial model--which had skewed results and was especially inaccurate in predicting positive cases. The classification report generated for the tuned model also provided the following information. The f1 score is a metric that combines the precision and recall using their harmonic mean, and a model that maximizes this has an optimal balance of precision and recall. In this case, the original model had an f1 score of .69 for classification 0, and .35 for classification 1, while the improved model had an f1 score of .67 for classification 0, and .65 for classification 1. The f1 score should be close to 1 for both classes to indicate a better classifier, so these metrics also confirm that the second model is a more reliable predictive model than the original one.

Conclusion

Based on the models created above to analyze this patient dataset, it is clear that it may be possible to make a prediction about a patient's likelihood of being readmitted to the hospital based on the influence of their lifestyle factors and other personal characteristics. Although this prediction cannot be definitive, when used in tandem with a professional's advice, such a model may prove to be useful in certain medical contexts. Whereas the linear and logistic regression models had low variance scores (indicating that the models did not capture all the variation in the data) and relatively high mean squared errors, a neural network model was found to be more useful in providing accurate predictions. Regression models (including lasso regression) did not show any indication that a particular set of input factors were significantly more influential on the outcome than others, so using principal component analysis is what allowed for a lower dimension input that made the training process more efficient while maintaining the original variation in the data set. The final model created had an accuracy of 67.28% on the testing data. With more exhaustive tuning of this model, there is a prospect of creating a model that is increasingly reliable in providing predictions based on the input factors of this data set.

The original data set provided was intended to simulate real-world patient data. Real data is diverse and complicated, and rarely are there extremely clear-cut patterns that emerge that might give us the confidence to be able to make significant decisions based solely on the outcome of a model alone. Models are tools to augment a human's judgement, not to replace them. In order to abide by

ethical considerations, domain expertise is crucial before the implementation or use of such predictive models. In recent years, and in the years to come, the use of machine learning tools and algorithms are becoming more widespread in areas of criminal justice, law-enforcement, healthcare, and education. Ensuring that this is done ethically and responsibly will be a duty that data scientists and domain experts must remain vigilantly mindful of.

Resources

Agrawal, Aayush. "Building Neural Network from Scratch." *Medium*, Towards Data Science, 18 Feb. 2019, towardsdatascience.com/building-neural-network-from-scratch-9c88535bf8e9.

Allibhai, Eijaz. "Building A Deep Learning Model Using Keras." *Medium*, Towards Data Science, 21 Nov. 2018, towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37.

Brownlee, Jason. "A Gentle Introduction to the Rectified Linear Unit (ReLU)." *Machine Learning Mastery*, 6 Aug. 2019, machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

Brownlee, Jason. "How to Configure the Number of Layers and Nodes in a Neural Network." *Machine Learning Mastery*, 6 Aug. 2019, machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/.

Brownlee, Jason. "How to Save and Load Your Keras Deep Learning Model." *Machine Learning Mastery*, 13 Sept. 2019, machinelearningmastery.com/save-load-keras-deep-learning-models/.

Brownlee, Jason. "How to Visualize a Deep Learning Neural Network Model in Keras." *Machine Learning Mastery*, 11 Sept. 2019, machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/.

Brownlee, Jason. "Use Random Forest: Testing 179 Classifiers on 121 Datasets." *Machine Learning Mastery*, 12 Aug. 2019, machinelearningmastery.com/use-random-forest-testing-179-classifiers-121-datasets/.

Galarnyk, Michael. "PCA Using Python (Scikit-Learn)." *Medium*, Towards Data Science, 4 Nov. 2019, towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60.

"How to Get the ROC Curve and AUC for Keras Model?" *Knowledge Transfer*, 15 July 2019, androidkt.com/get-the-roc-curve-and-auc-for-keras-model/.

Ivanov, Slav. "37 Reasons Why Your Neural Network Is Not Working." *Medium*, Slav, 16 Apr. 2019, blog.slavv.com/37-reasons-why-your-neural-network-is-not-working-4020854bd607.

Ujjwalkarn. "A Quick Introduction to Neural Networks." *The Data Science Blog*, 10 Aug. 2016, ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.