Capstone Project
Jason Mark
Aug 20th, 2019

1. Definition:

1.1. Problem Overview:

Diabetes is one of the most common and most expensive chronic diseases worldwide. In 2004 it was estimated that in the US alone, approximately 5 million people unknowingly had the disease while another 13 million were aware of their diagnosis. The expenses involved in treating diabetes once diagnosed are enormous. It is estimated that in 2012 the cost related to treatment of diagnosed diabetes in the U.S. alone was $245 billion and that there were 252,806 deaths related at least in part to diabetes. [1]

Early detection of the disease can help reduce the risk of serious life changing complications such as premature heart disease, stroke, blindness, limb amputations, and kidney failure. [2] Each of these brings with it an increase in other types of risks such as secondary infections. It can reduce both loss of life and cost while simultaneously increasing the quality of living for affected patients.

1.2. Problem Statement:

Models that can help predict an individual with diabetes could be a useful tool to support a physician's decision-making process when working with patients. It could also be leveraged to screen populations of patient data to identify patients most likely to have undiagnosed diabetes and intervene proactively with further testing and monitoring. This can be framed as a binary classification problem to separate those who will vs. those who will not develop diabetes.

1.3. Metrics:

While several other research papers have used straight accuracy to measure the performance of their models, I intend to use the ROC-AUC curve which accounts more for true positive rate vs. false positive rate. This becomes important for a couple of reasons. First of all, the dataset is somewhat imbalanced across classes with 768 total observations, 268 of which are positive for the onset of diabetes and 500 of which are not. Doing a very simple model of predicting that all patients are not positive for the onset of diabetes (class 0) would give an immediate accuracy of 500 / 768 or .65 (65%). This would be an undesirable model because, while more accurate than random chance, it will never make a positive precision which is the entire point of what the model is expected to learn.

Secondly given that a positive prediction from the model could have implications for what a healthcare provider chooses to convey to a patient or how they choose to treat the patient. The provider needs more information about the likelihood of a false negative or positive value from the model. A false negative may result in failing to further test a patient who ends up developing diabetes and misses out on the benefits of early detection. Conversely a false positive, could result in extra tests performed that add costs and potential discomfort or inconvenience to the patient or healthcare provider. It would be up to the healthcare provider or their organization to determine in which of these scenarios they would choose to accept risks and to what degree.

2. Analysis

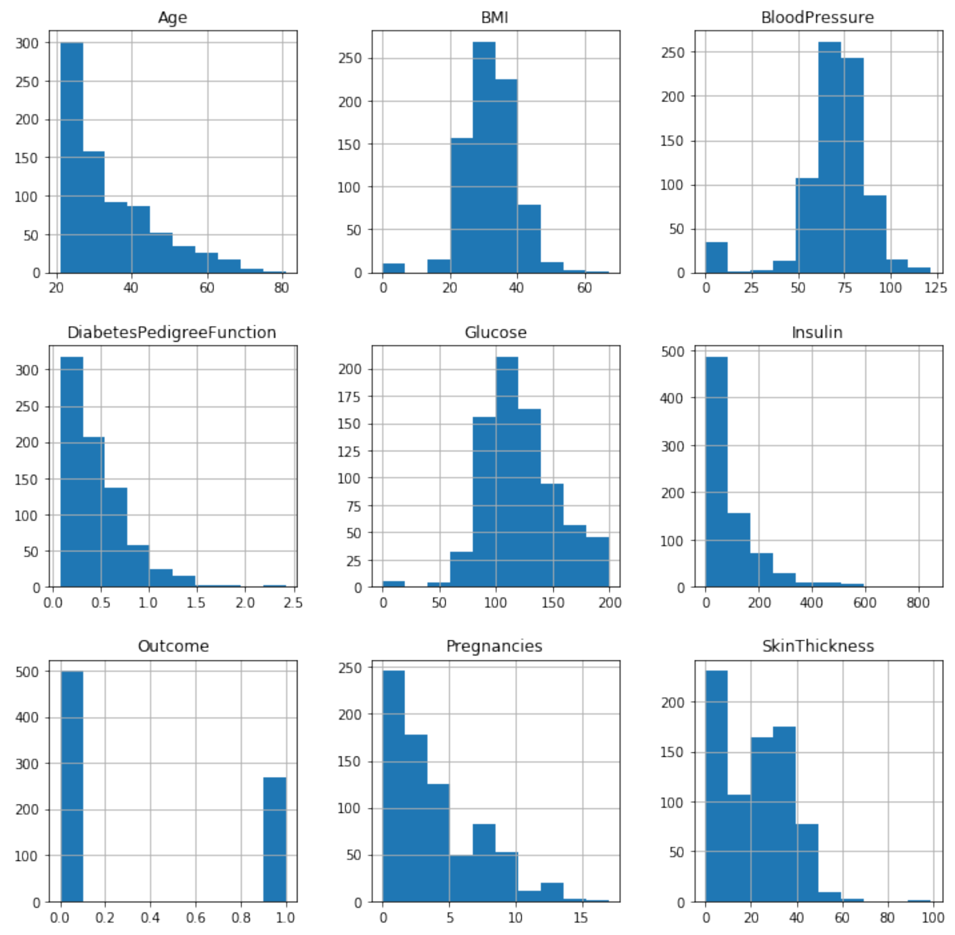
2.1. Data Exploration:

For purposes of this project, I will be using the Pima Indians Diabetes Dataset. This dataset originates from the National Institute of Diabetes and Digestive and Kidney Diseases and is now hosted on Kaggle by the UCI Machine Learning Repository.[4] It contains 8 measurement values for 768 patients of which 268 were determined to have diabetes as indicated in an ‘Outcome’ variable. The 8 included measurement variables are:

Measurement
PregnanciesNumber
BloodPressureDiastolic
GlucosePlasma
SkinThicknessTriceps
Insulin-2Hour
BMIBody
DiabetesPedigreeFunction
Age

Upon further inspection of the Pima Indians Diabetes Dataset, there are 768 patients represented with 268 of these having the diabetes label. Data is not for each feature on each patient observation. Practically this makes sense because unless collected under the same set of controls for each patient, some items may be more difficult to measure than others. An example of a feature that could be particularly problematic for example is the “DiabetesPedigreeFunction”. This feature represents the probability of a patient having diabetes given their family history. This could be problematic because it becomes dependent on the accuracy of family history, whether or not diabetes was correctly identified in relatives, and in the case of something like adoption, whether or not family history information is even available on a patient. Because it is likely that at prediction time, all measurements may also not be present, I chose to leave the missing values in the dataset and am using a model which could work around those values. Fortunately, all “missing” values are represented as zeros in the data so there is not a need to replace them explicitly if I am not replacing them with a different value (e.g. median). It is worth pointing out however, that the meaning of a “0” value varies depending on which feature is being considered. In the case of “BloodPressureDiastolic” for instance, a 0 must represent a missing value because an actual blood pressure value of 0 would represent a deceased patient. On the other hand, in the case of “PregnanciesNumber”, a 0 represents a valid observational value that a patient has had 0 pregnancies.

2.2. Exploratory Visualization:



Visual inspection of a summary of the data shows that there are a number of features with minimum values of 0. These are worth exploring in more detail. It does not look like the data contains any NaN / NA values.

There are missing feature values in the dataset although they're not all immediately apparent because they are coded as zeros rather than NaN or NA values. Because zero can be a valid measurement for some of the variables, we'll need to consider them one by one:

Feature	Whether or not 0 is valid
Pregnancies	0 can be a valid measurement
Glucose	0 is unlikely to be a valid measurement
Blood Pressure	0 is unlikely to be a valid measurement
Skin Thickness	0 is unlikely to be a valid measurement
Insulin	0 is unlikely to be a valid measurement
BMI	0 is unlikely to be a valid measurement
DiabetesPedigreeFunction	0 can be a valid measurement

2.3. **Algorithms and Techniques:**

My initial plan was to try to use a DecisionTreeClassifier to fit the data and make predictions. The reason for this was the fact that a decision tree could be relatively easy to explain to a healthcare provider because the cut points would help show the interaction between features – for example that patients above a certain age who also have high blood pressure might be predicted for the onset of diabetes while patients under a certain age might not be. These types of clearly articulable rules could translate into guidelines for how the medical provider screens patients. Unfortunately, I wasn't able to obtain results with this model that were any better than the original paper.

My next choice was to use the XGBoost algorithm based both on recommendations from proposal reviewers and its current widespread success across a variety of problems. I also wanted to preferably use a model for which AWS SageMaker has a pre-built container image for training and deployment to simplify maintenance and deployment of the model.

At a high level, the process which will be described in more detail in a subsequent section was:

- Prepare / preprocess data
- Split data into train/validation/test sets
- Create an xgboost Estimator
- Create a HyperParameterTuner object
- Run HyperParameterTuner to search for best performing models
- Deploy best performing model as SageMaker endpoint
- Evaluate deployed model using holdout test data set

2.4. **Benchmark:**

I am interested in comparing my results to that of the original paper that was written about the Pima Indian Dataset. [3] In this paper, the sensitivity and specificity achieved was 76%. The original paper also shows the ROC curve which is what I'm intending to compare to my results.

3. **Methodology**

3.1. **Data Preprocessing:**

3.1.1. Feature Scaling:

Because the various features in the dataset are on very different scales, I opted to scale the feature values to the space between 0-1. While this is not necessarily a requirement for tree-based algorithms, it also does not adversely affect them and it can help others algorithms to perform better and so I did this step in case I decided to experiment with further algorithms. I used the MinMaxScaler from the sci-kit learn library to accomplish this.

3.1.2. Split dataset

Using the train_test_split function from the sci-kit learn package, I split the dataset into:

- 80% training
- 10% validation

- 10% testing

Because I wanted to use both a validation and test set and the overall dataset is not that big, I opted to use a bit extra for training purposes.

3.1.3. Prepare for Training

Prepared data was saved to CSV format in train.csv, validation.csv, and test.csv files and loaded into S3 for use in SageMaker model training.

3.2. **Implementation:**

Initially I tried using simpler linear models and the DecisionTreeClassifier from the scikit-learn package to build models and make predictions from the data. My rationale for using these approaches was that they could also offer pretty clear guidance on how the cuts were made which could be more straightforward for a human to understand. Given that this problem is within the medical domain and could be used to guide care for patients, I think that an easily explainable model would be favored over a more opaque or “black box” type model. In applying these models however, I didn’t get results much different from the original paper. Given that my goal was to try to improve on the original paper, I set these models aside.

I determined to use the XGBoost algorithm to predict those cases where diabetes onset will occur. The XGBoost algorithm is currently a widely successful algorithm on a variety of problems. In many cases it can yield performance close to or on par with deep learning models. It is also robust enough to handle the class imbalance in the dataset. Furthermore, AWS SageMaker has a pre-built container for this algorithm which simplifies the development of an initial model without needing to implement a custom container for a deep learning approach like TensorFlow or PyTorch. I'm favoring the most straightforward approach first and can come back and explore those models later if needed.

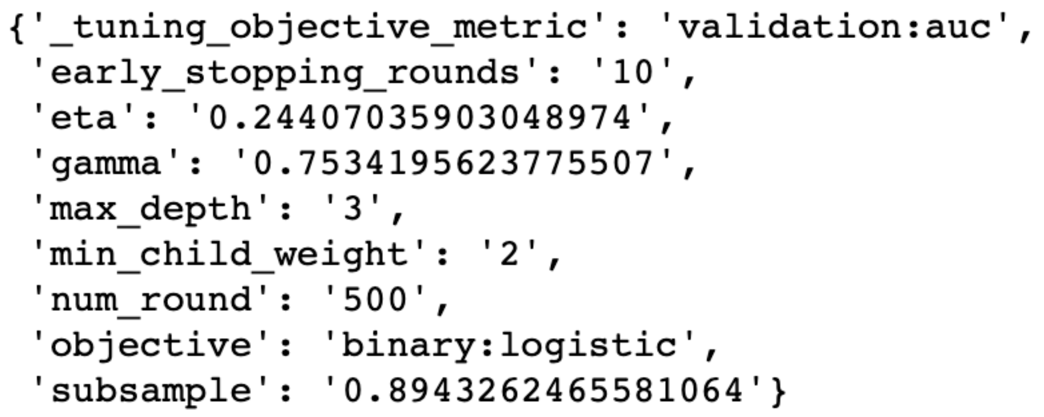
3.3. Refinement:

After building an initial XGBoost Model with an experimental initial set of hyperparameters, it appeared that this model would exceed the previously mentioned simple models. To ensure that a reasonably optimal solution was found, I used a HyperParameterTuning job from within SageMaker to evaluate up to 10 different models across a range of hyper parameters and then selected the best model to evaluate against my test data.

4. Results

4.1. Model Evaluation and Validation

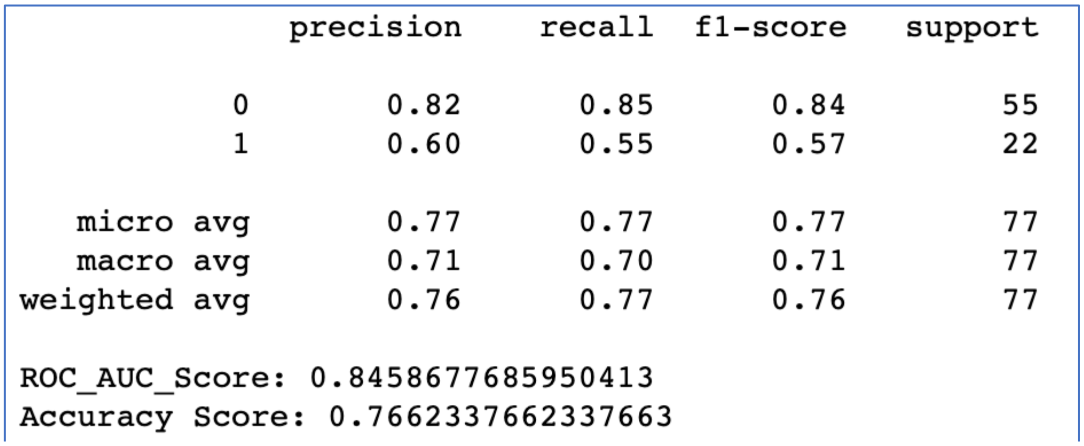
The final model selected after hyperparameter tuning had the following hyperparameter values:



```
{'_tuning_objective_metric': 'validation:auc',
 'early_stopping_rounds': '10',
 'eta': '0.24407035903048974',
 'gamma': '0.7534195623775507',
 'max_depth': '3',
 'min_child_weight': '2',
 'num_round': '500',
 'objective': 'binary:logistic',
 'subsample': '0.8943262465581064'}
```

The accuracy of my model is similar to the original. The ROC-AUC score of .845 is a slight improvement over the original model.

4.2. Justification



```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        55
           1       0.60      0.55      0.57        22

   micro avg       0.77      0.77      0.77        77
   macro avg       0.71      0.70      0.71        77
weighted avg       0.76      0.77      0.76        77

ROC_AUC_Score: 0.8458677685950413
Accuracy Score: 0.7662337662337663
```

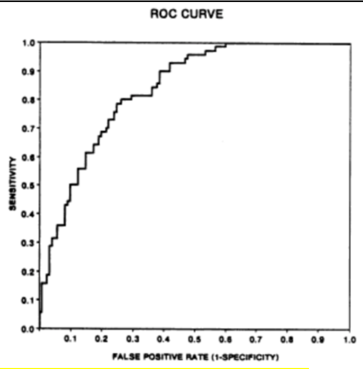
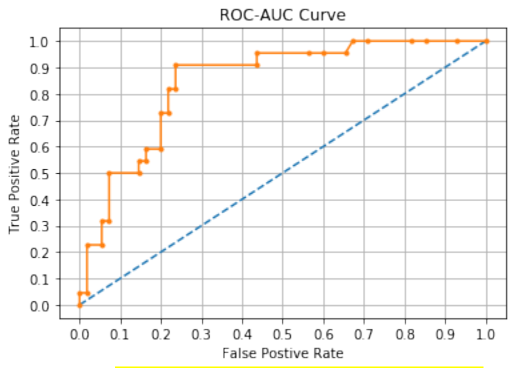
While the accuracy value of my model is similar to the original, the ROC-AUC value is slightly higher. Given that an original value was not specified, this has to be interpreted from the charts in the following section.

It is interesting to look at the precision and recall values by class. My model does do a significantly better job at predicting the outcome class 0 than it does at predicting outcome class 1. Despite choosing an algorithm and model type to account for the imbalance in the data, there is still a bias towards predicting the majority class. This could also indicate that there are just more features needed to better predict outcome class 1 – or diabetes onset. It is possible that missing values play some role here and that could be further explored as well.

In practice this model would not be sufficient to directly guide care. It could however be a supporting tool for a healthcare provider with an understanding that if the model predicts that a patient will not develop diabetes, that is more likely to be correct than a prediction that the patient will develop diabetes. Even in this scenario though, some unnecessary further screening tests might be avoided if only patients with a prediction of onset diabetes were required to undergo those tests.

5. Conclusion

5.1. Free-Form Visualization

 ROC from original paper	 ROC from project model

Based on the ROC-AUC curve, my model's performance is slightly better than the original paper model. per observation.

5.2. Reflection

In comparing to other results using the same dataset, they have similar results using a variety of different models. I think that the similarity in results from a variety of researchers over a span of many years indicates that we've likely reached the best obtainable results on this very limited dataset with current methods. Diabetes and health in general are complicated problems that are influenced by a wide variety of factors. I would say that the ability of the models to have some positive predictive power supports the idea that better models could be produced given a larger dataset or a dataset with more available features.

A significant challenge with models affecting healthcare is quantifying what tolerance for error is to be allowed. While human healthcare providers are not infallible and make mistakes in diagnosis and screening, it is somehow generally not seen as quite the same "failure" when they get it wrong as when a machine or model gets it wrong. Our tolerance for accepting one vs. the other is not the same and so even when a model does reasonably well, the consequence of ever making a mistake is often seen as too high to accept from a machine learned model.

5.3. Improvement

One other consideration for this dataset is that it is collected specifically from a study using members of the Pima Indian tribe. It is entirely conceivable that there are genetic or lifestyle differences within this population that make any model built using this data not robust enough or valid for a larger more general population. With this in mind, any resulting model would need to be validated against populations of patients of either their respective population subgroup or against the general population before being widely deployed or used.

References:

[1] CDC – Deaths and Cost related to Diabetes.
https://www.cdc.gov/diabetes/data/statistics-report/deaths-cost.html

[2] Diabetes: A National Plan for Action. The Importance of Early Diabetes Detection.
https://aspe.hhs.gov/report/diabetes-national-plan-action/importance-early-diabetes-detection

[3] Using the ADAP Learning Algorithm for Forecast the Onset of Diabetes Mellitus
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/

[4] Pima Indians Diabetes Dataset
https://www.kaggle.com/uciml/pima-indians-diabetes-database/activity