Steve Su
CSCI3202
4/27/2020
HW5 – Stent/No Stent Prediction
https://github.com/steve303/stent-nostent.git

EDA

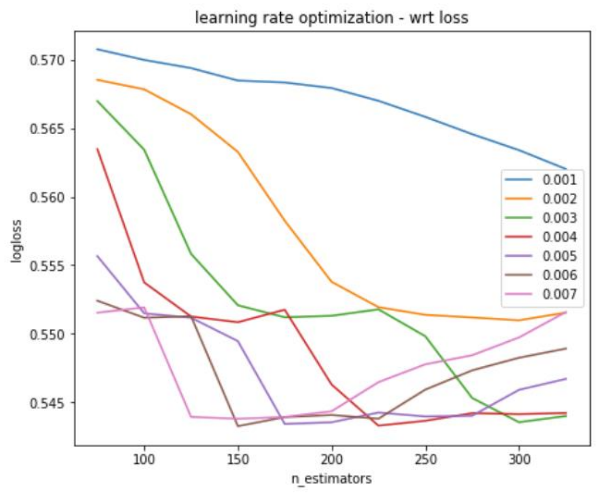
The data set consisted of floats and integers. No null values were found. A histogram of each feature was plotted. Observations show that many of the distributions were skewed. Also the number of True (stent) labels were about half of the False labels.

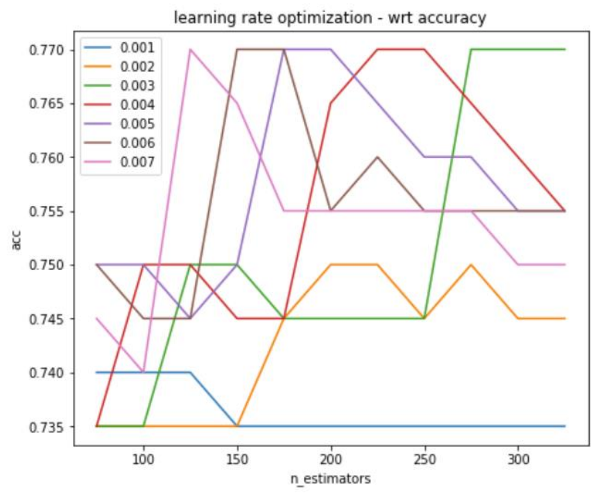
Machine Learning Models

The models used in this study were Adaboost decision tree, Neural Network, and SVM with PCA preprocessing . The libraries used for these models were Sklearn, Tensor Flow, and Keras.

Adaboost Decision Tree

The adaboost decision tree had moderate results. With respect to validation data, the accuracy was 0.77 and log loss was 0.54. The recall was 0.46 and so it does not do well when trying to predict the true label, stent in our case. It's correct less than 50% of the time. When the model does predict the true label it's performance is moderate with precision of 0.78. Hyperparameter search was performed on both the number of estimators (stumps) and learning rate with respect to log loss and accuracy. The plots below show that the best learning rate was 0.005 and the best number of estimators was 175. The max depth of the tree was set to three. This is usually set to a small number because adaboost uses many stumps which by themselves are weak learners and ensembles them to become an effective learner.

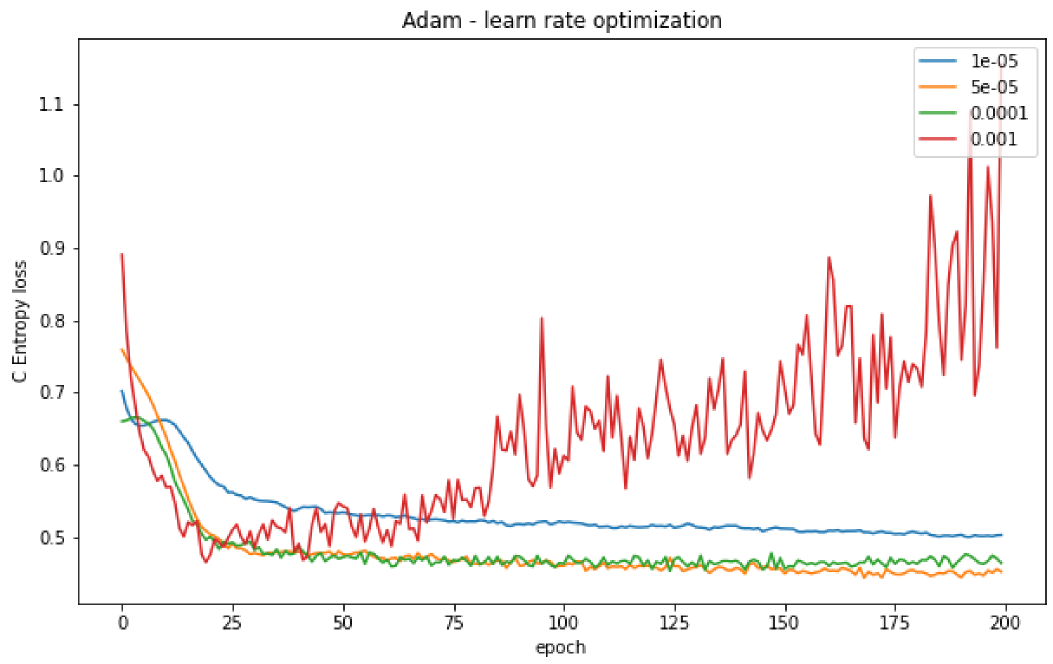




Neural Net

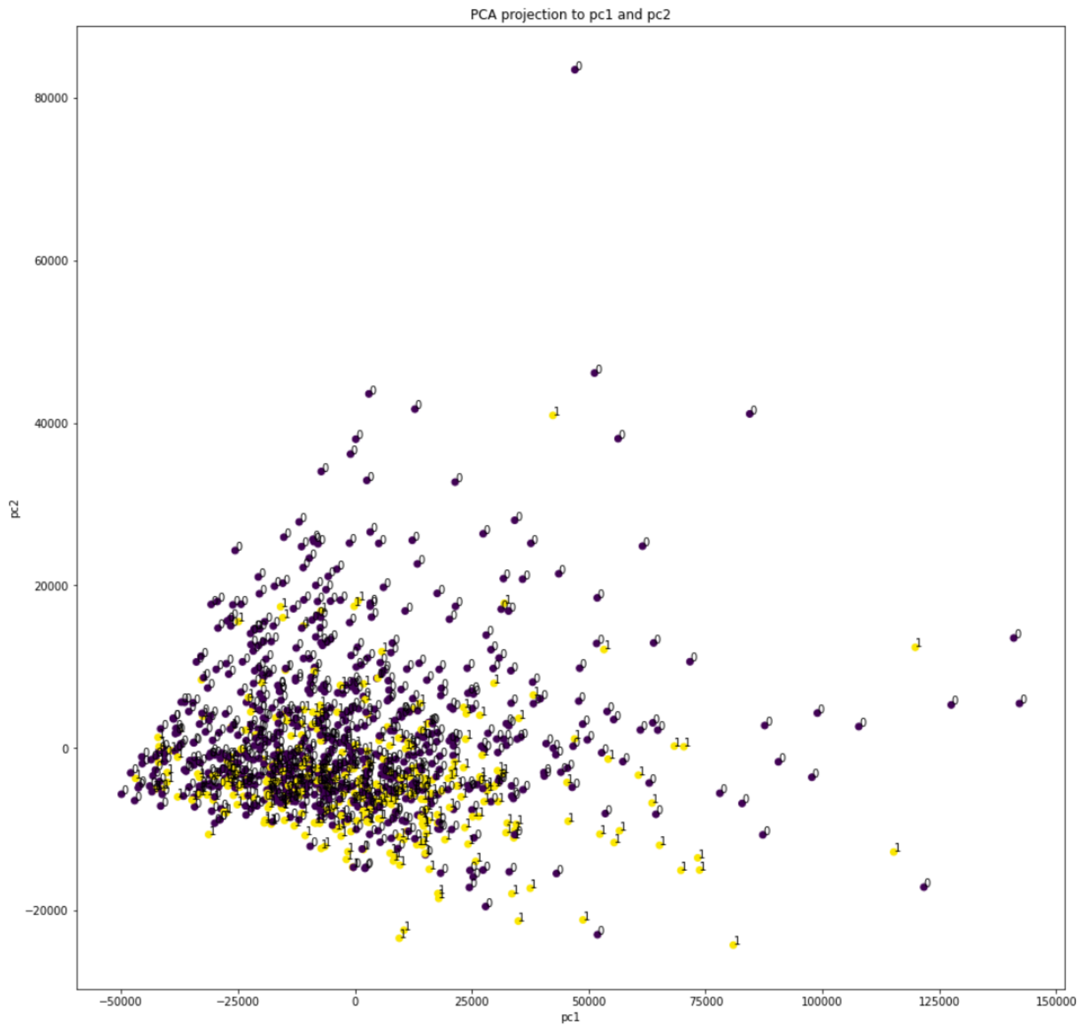
The best NN design of many candidates was one of three fully connected layers. I initially started with a deeper network but scaled back. Layer one had 64 neurons with sigmoid activation function followed by batch normalization. Layer two had 32 neurons with sigmoid activation, again followed by batch normalization. Layer three was the output layer and consisted of a single sigmoid neuron. The batch normalizations were helpful preventing vanishing or exploding gradients especially with deep networks associated with sigmoid activation functions. Simple experimentation demonstrated this. Deeper models were investigated but didn't seem to outperform the more shallow networks so I concentrated on optimizing the smaller network. The optimization hyperparameters I focused on were learning rate, optimizer method, and epoch. The optimizer candidates were SGD and Adam. In the SGD optimizer momentum was set to 0.0 but Nestrov was set to True. Experimentation with setting momentum was a bit tricky. Depending on the value, during convergence, loss would oscillate between values, not settling after many epochs. To keep things simple momentum was set to zero but Nestrov was set to True. All input data was scaled to be between 0 and 1 before feeding into the network.

It turned out that both SGD with Adam models were almost the same for log loss and accuracy, 0.46 and 0.77, respectively. The recall in the SGD model was slightly better than the Adam model, 0.56 versus 0.50, respectively. But SGD's precision was slightly worse than Adam's model, 0.67 versus 0.70, respectively. Overall both optimizers performed similarly. However each model had a different preference towards learning rate. In the SGD model the best learning rate was 5e-4 and for Adam it was 5e-5. It could be seen that learning rates that were to high had huge oscillations. See figure below.



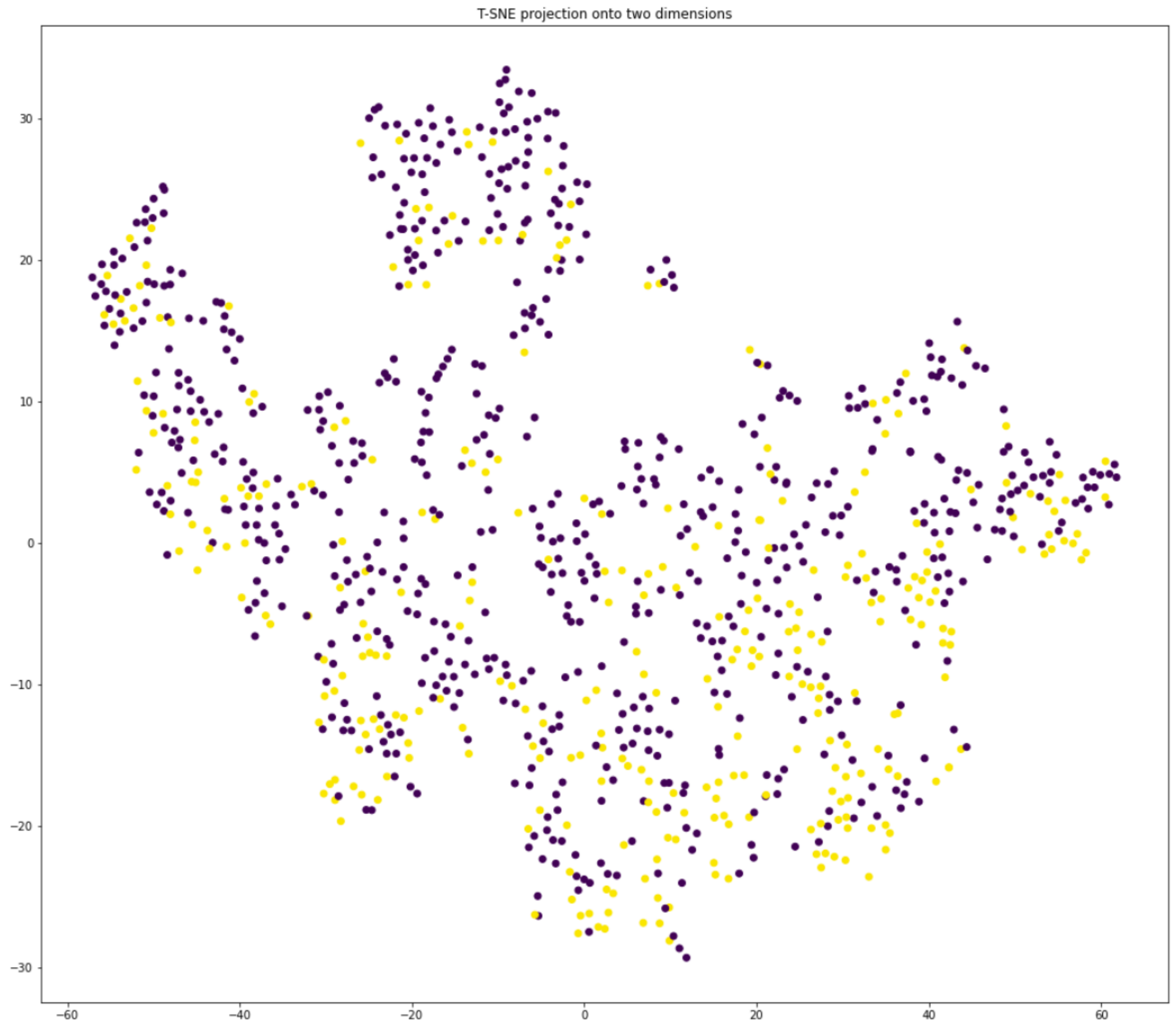
SVM with PCA

PCA was chosen as a preprocessing step before using the SVM classifier. PCA takes a high dimensional data set and projects to lower dimensions through a transformation process by calculating eigen values and eigen vectors of the input data's covariance matrix. The result is a means to identify features which most directly influences the prediction of the label. Before PCA was performed, the input data was mean centered to zero. The PCA showed that 99% of the variance could be attributed to the first six principal components. These six components were then fed into the SVM model. By observing the PCA plot of the first two components there was no clear separation boundary. Because of this, this particular data may be difficult to classify. I chose a non linear kernel (RBF) instead of linear because it may have a better chance of fitting the data. However, results from the SVM-RBF were not particularly good. The SVM classifier reported an accuracy of approximately 0.73, a log loss of 0.58, and a recall of 0.10. This model is not particularly good at predicting when a stent is needed with recall of 0.10. It only predicts correctly 10% of the time when a stent is required. The hyperparameter search of C and gamma were inconclusive. There were no strong trends either way. C is the penalty parameter of the error. Higher values can lead to overfitting . Gamma is a parameter for non linear hyperplanes. Higher gamma values helps to exactly fit the training data set, but again can lead to overfitting.



T-SNE visualization

T-SNE (t distribution stochastic neighbor embedding) is another method like PCA to project high dimensional data set to a lower one but it is better when the data set lives in a manifold like structure such as a cylinder or ball. However, the results do not show any obvious boundaries of separation as in the case of the PCA.



Summary

Out of the three models the Neural Network was the best with lowest log loss of 0.46, followed by Adaboost, 0.54, and SVM, 0.58. None of the models were particularly good at predicting when a stent would be needed with recall values of 0.56 for the Neural Network and 0.50 for Adaboost. The SVM was particularly bad at 0.10. Both the Neural Network and Adaboost tree had accuracy of 0.77 while SVM was 0.73.