Predicting Diabetes and Heart Disease Using Diagnostic Measurements and Supervised Learning Classification Models

Kunal Sharma

CS 4641 Machine Learning

Abstract

Supervised learning classification algorithms have commonly been used for the detection of various diseases. This paper will focus the diagnosis of diabetes and heart disease using diagnostic measurements. The supervised learning algorithms that will be discussed include: Decision Trees (With Pruning), Boosted Decision Trees, Artificial Neural Networks, Support Vector Machines, and K-nearest neighbor.

Both datasets will be normalized so that features with higher magnitude will not weigh considerably more and so that training time will be reduced. Cross-validation will be used to determine the accuracy of the model. The dataset will be split randomly to create a training set with 75% of the data and a test set with 25%. Particular attention will be given to hyper parameter choices and their performance, accuracy over various numbers of iterations/epochs, and overfitting concerns.

The datasets have been obtained from kaggle.com and their web addresses will be included in the references section. Python and Jupyter Notebook was used for development. Packages used for data manipulation and visualization include Numpy, Matplotlib, Pandas. Machine libraries used include sklearn and keras.

1.1 Diabetes Diagnosis Introduction

Originally, this dataset is from the National Institute of Diabetes and Digestive and Kidney Diseases. The aim of this dataset is to use a pre-selected set of diagnostic measurements to predict whether a patient has diabetes. The patients in this dataset are females of Pima Indian heritages that are at least 21 years of age.

This dataset has a total of 768 rows, a single target column (outcome) and seven medical predictor column detailed in the chart below.

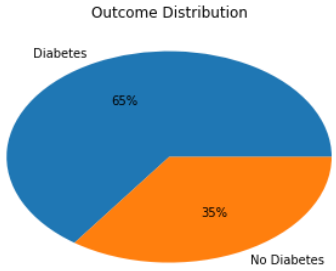
Feature Description

Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration a 2 hr in an oral glucose tolerance test

Blood Pressure	Diastolic blood pressure (mm Hg)
Skin Thickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
Diabetes Pedigree Function	Diabetes pedigree function
Outcome	Class variable (0 or 1)

Outcome Distribution

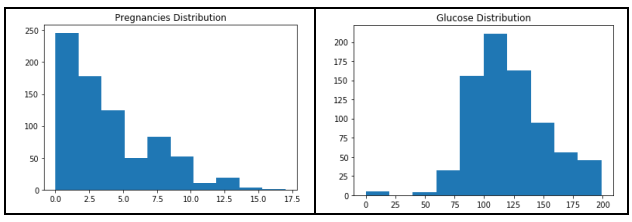
65% of the data represents patients diagnosed with diabetes and 35% represent patients without diabetes. This dataset is therefore unbalanced and we should expect our accuracies should be above 65% if we expect our models to be predictive.

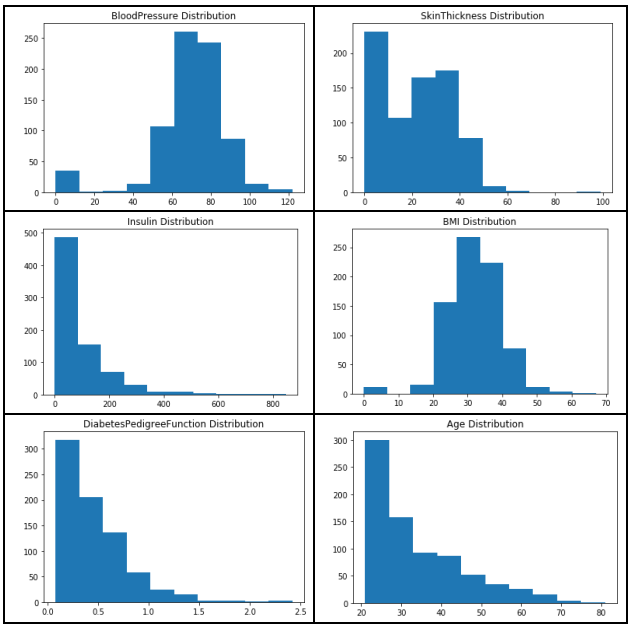


Feature Data Distribution

Glucose, Blood Pressure, and BMI follow a roughly normal distribution while the remaining features are typically skewed to the right.

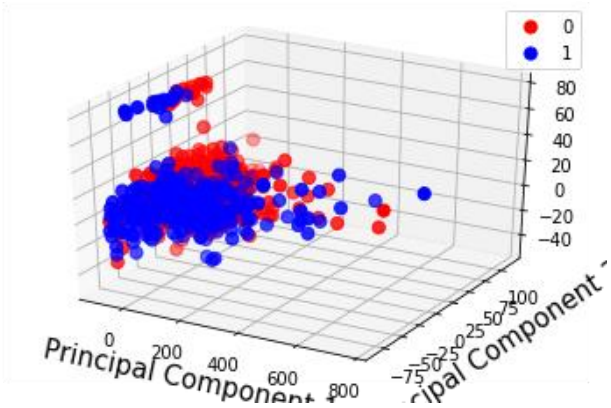
Left to Right: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age





Principal Component Analysis

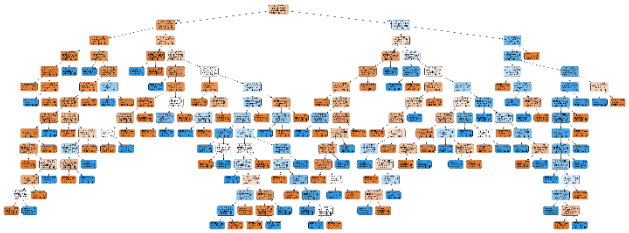
We can use Principal Component Analysis to reduce the dimensionality of the feature space from 8 to 3 dimensions so that we can visualize the data in the figure below. The clustering in the visualization is a good sign that our data can be separated and classified.



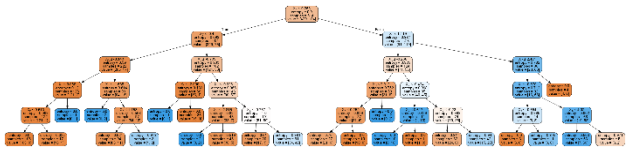
1.1 Classification using Decision Tree (With Pruning)

Decision Tree learning passes observations through a series of if-else conditions based on the training data to reach a target value. Pruning is a technique that reduces the size of the decision trees based on select limited criterion.

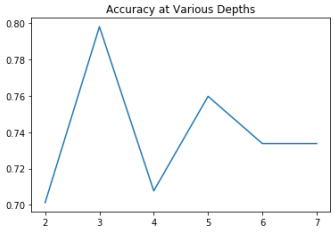
After creating our initial decision tree model without any form of pruning using our model we get an accuracy of 70.78%. Given that 65% of our data is a single class, this model is only slightly predictive. Using Graphviz we can visualize our decision tree.



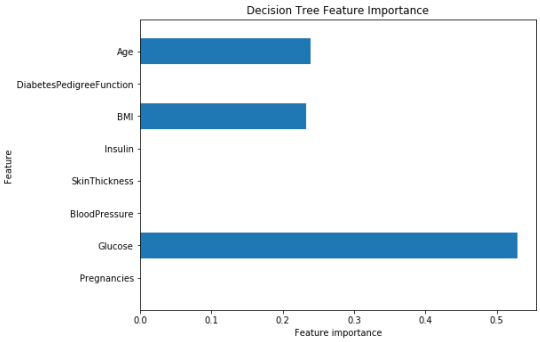
Evidently, our decision tree is incredibly complex making it very likely that overfitting is occurring. Overfitting is when a model is not able to generalize beyond its training data. In this case we can reduce the complexity by pruning the tree by limiting the max depth of nodes. After we limit the depth to 5 the model now achieves an accuracy of 75.97%.



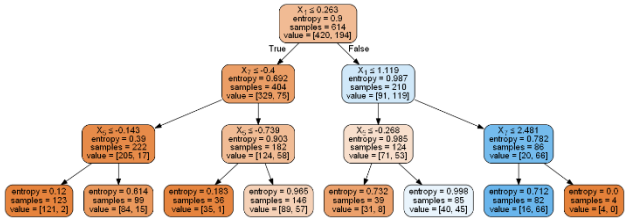
Our simpler model is 5% more accurate. To see if 5 is the ideal limit to set max node depth let's look at the accuracy of nearby numbers. 3 has the highest at 79.8%.



To better understand what the model is primarily making its decisions on, we can look at a plot of the feature importance. Age, BMI and Glucose seem to have the highest impact on determining if a patient should be diagnosed with Diabetes. Our decision trees selected features seem intuitive and are likely to be leading to good predictions.



Model with Max depth of 3 Nodes:



We are using entropy as our criterion for making split decision instead of the Gini index because while logarithmic functions are computationally intensive, our data is less than a thousand rows and entropy would provide a loss error with greater "detail" as it uses the logarithmic function.

1.2 Classification using Boosted Decision Tree

To boost the decision tree we can use Adaptive Boosting, or Adaboost for short. Adaboost is a form of an "Ensemble Learner" which means it uses multiple learners to build an even stronger learning algorithm. This algorithm works by iteratively accounting for the incorrectly classified training set instances.

After running adaboost on our previous decision tree that had an accuracy of 79.8% the accuracy drops down to 70.3%. This may likely be because boosting trees tries to re-classify mislabeled entries. However, some mislabeled entries may have been inconsistent entries or outlies and that's why they were misclassified in the base decision tree.

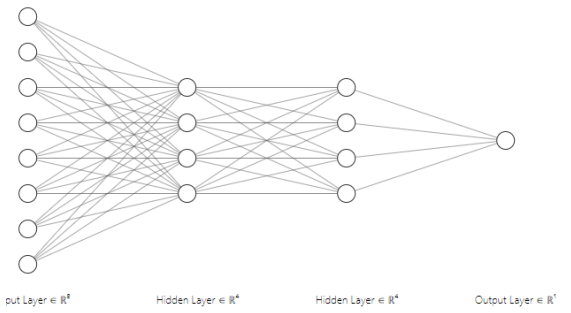
Seeing as only 3 out of 8 of the features follow a normal distribution this is very likely to be the case. Adaboost would attempt to re-fit the tree to the data points and reduce the overall accuracy.

Diabetes is a complex problem that medically has various factors that play into it. The available data set is only 768 instances, having more data would likely solve this issue with Adaboost as more features may begin to follow a normal distribution and outliers/inconsistent data would not have as large of an effect.

1.3 Classification using Artificial Neural Network

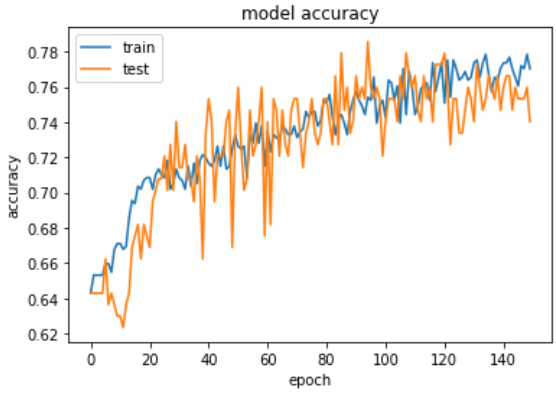
Neural Networks is an algorithm inspired by the neurons in the brain. Neural Networks use a network of perceptrons that multiply a series of weights to determine a final output.

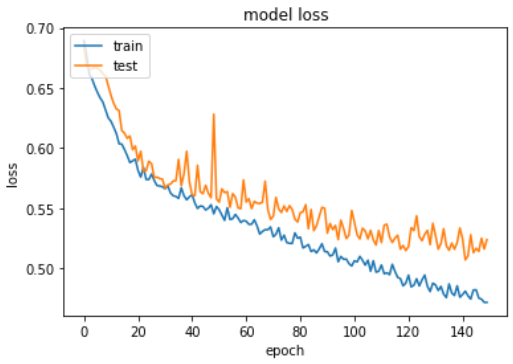
The first approach for the neural network that we can try is two hidden layers that are the average between the number of features (input nodes) and number of classes (output nodes). In this case, we have 8 input feature nodes and a single output node for binary classification. We will thus use 4 nodes in our hidden layers



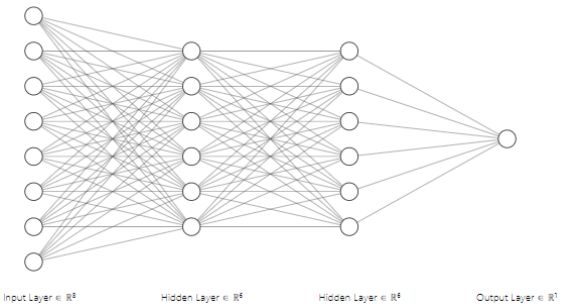
.

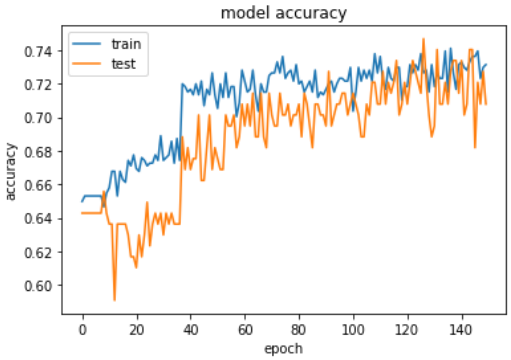
This resulted in an accuracy of 75%. The test data accuracy got very close to the training accuracy. As expected the loss for both the training and test data decreased as the number of epochs increased.

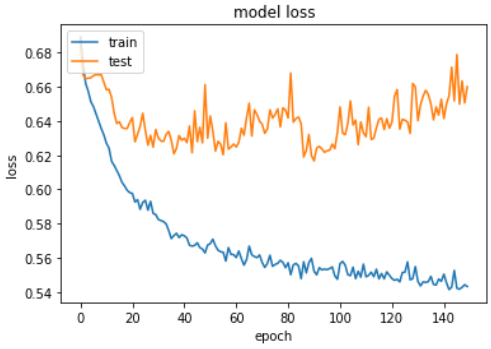




We can now try to increase the number of neurons in the hidden layers. The intuition here is that if the data is more complex then more neurons may be needed to create an effective model, if not we should see signs of overfitting.

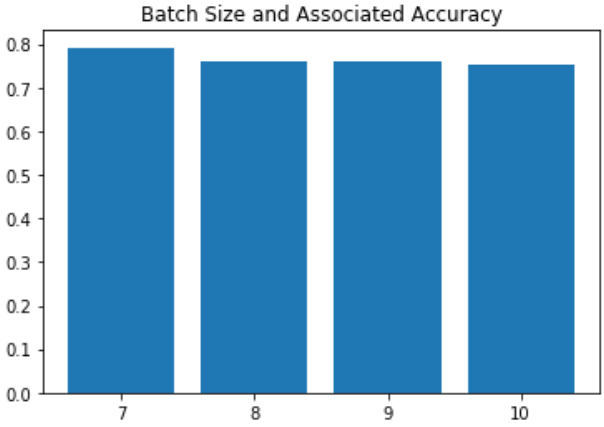






Since the loss is starting to increase we can see that the model is now overfitting and that adding more neurons to the hidden layer is not the best choice.

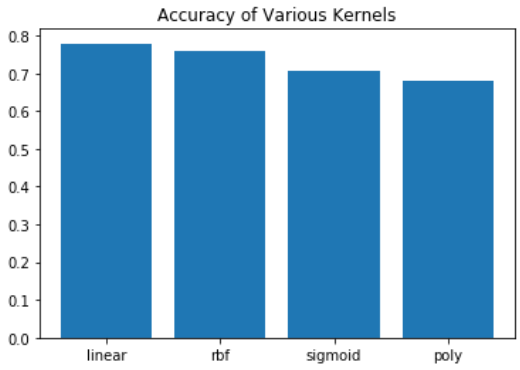
Next, we can try changing the batch size. The batch size is the number of training samples that will be propagated through the network. After trying different batch sizes, we find that the batch size of 7 gives us the highest accuracy of 79.2%.



There are several hyper parameters that can be tuned to optimize accuracy. The ‘tanh’ activation function was attempted instead of sigmoid activation, however, the accuracy dropped to 70.7%. Changing the loss function to “mean squared error” instead of “binary cross entropy” also dropped the accuracy, to 73.4%.

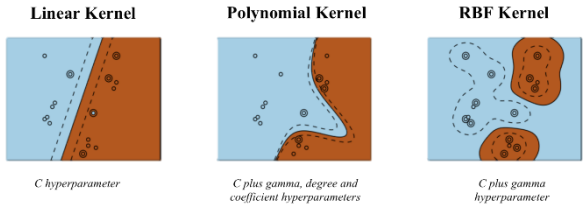
1.4 Classification using Support Vector Machine

Support Vector Machines split the feature space by finding a suitable hyperplane that will have the highest accuracy in classification. Kernels allow the SVM to generate the hyperplane without needing to compute in higher dimensions.



After trying different kernels on the default hyper parameter settings we find that the linear kernel performs the best with an

accuracy of 77.9%. This is likely because this is the simplest kernel and the least likely to over fit. This can be visualized with the example graph below.



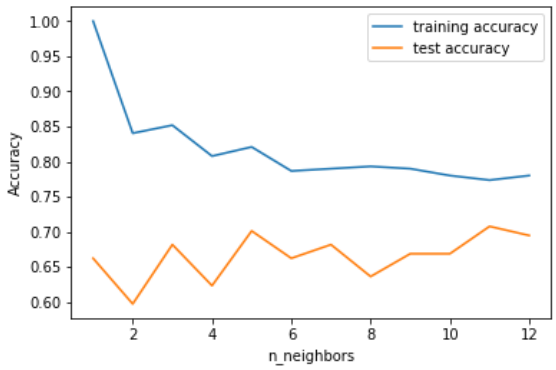
Source: beta.cambridgespark.com

After varying the kernel type, we can also vary the penalty parameter C of the error term. Trying values C as 1, 3, 30, 3000 all led to the same result of 77.9%.

1.5 Classification using K-Nearest Neighbors

K-nearest neighbors is an algorithm that uses all of the points in the training set and classifies a point by considering the K closest data points to it and assigning the new point to the class with the highest count.

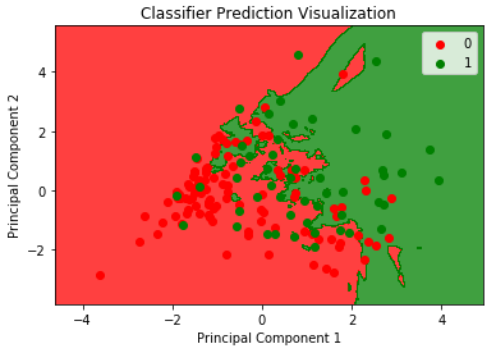
Keeping all other default settings, we can vary K from 1 to 12 to find that 11 is the optimal number for K in terms of maximizing the test accuracy to 70.77%



KNN can use various algorithms to compute the nearest neighbors. However ‘ball_tree’, ‘kd_tree’, ‘brute’ and ‘auto’ all gave the same result of 70.77%.

KNN by default uses the Euclidian distance but it can also use the Manhattan distance. Using the Manhattan distance dropped the accuracy to 0.66% making model less predictive.

We can visualize the classification of KNN by reducing the dimensionality from 8 dimensions to 2 and plotting the points and the regions that the KNN is classifying as positive and negative as seen below. Keep in mind that a large percentage of the variance in the data is lost due to the nature of dimensionality reduction. We can clearly see that despite the loss of variance, the model is clearly overfitting even at its optimal K value. KNN may not be the ideal model for this dataset.



One reason why the performance of KNN might be so poor is the curse of dimensionality. The curse of dimensionality is the idea that as the dimensionality grows there are fewer observations per region. In about 100 dimensions all points become essentially equidistant. With the feature space in 8 dimensions, this may explain why KNN isn’t performing as well as the other classifiers.

1.6 Conclusion for Diabetes Diagnosis

Here are the final best test results of each of the classifiers on this diabetes dataset.

Rank	Model	Result
1	**Decision Tree with Pruning**	**79.8%**
2	Artificial Neural Network	79.2%
3	Support Vector Machine	77.9%
4	K Nearest Neighbors	70.8%
5	Boosted Decision Tree	70.3%

Decision Trees with Pruning performed the best on this dataset and this is likely because the model is not affected by higher dimensionality like K-Nearest Neighbors is or by outliers (since it's pruned) as Adaboost is. Artificial Neural Networks comes very close in terms of performance and while it takes longer to train, it has the potential to surpass the performance of the decision tree with further hyper parameter tuning.

2 Heart Disease Diagnosis Introduction

Originally, this dataset is from the Cleveland database. The aim of this dataset is to also use a pre-selected set of diagnostic measurements to predict whether a patient has diabetes. The original dataset contained 76 attributes, however, this dataset contains the 13 most statistically significant attributes.

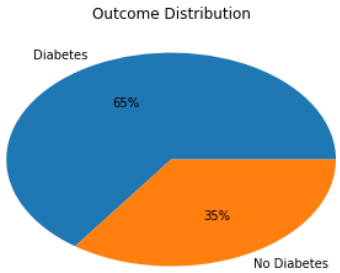
This dataset has a total of 303 rows, a single target column (outcome) and thirteen medical predictor columns detailed in the following table.

Feature Description

Age	Patient Age
Sex	1 = Male, 0 = Female
Chest Pain Type	4 Values associated with chest pain types
Resting Blood Pressure	Units in mm Hg
Serum Cholesterol	Units in mg/dl
Fasting Blood Sugar	>120 mg/dl
Resting Electrocardiographic results	Values 0,1,2
Maximum Heart Rate Achieved	Beats/Min
Exercise Induced Angina	numerical
Oldpeak	ST depression induced by exercise relative to rest
Peak exercise slope	Slope of ST segment
Number of Major Vessels	(0-3) Colored by flourosopy
Thal	3 = normal, 6 = fixed defect, 7 = reversible defect
Heart Disease	1 = Diagnosed, 0 = Not Diagnosed

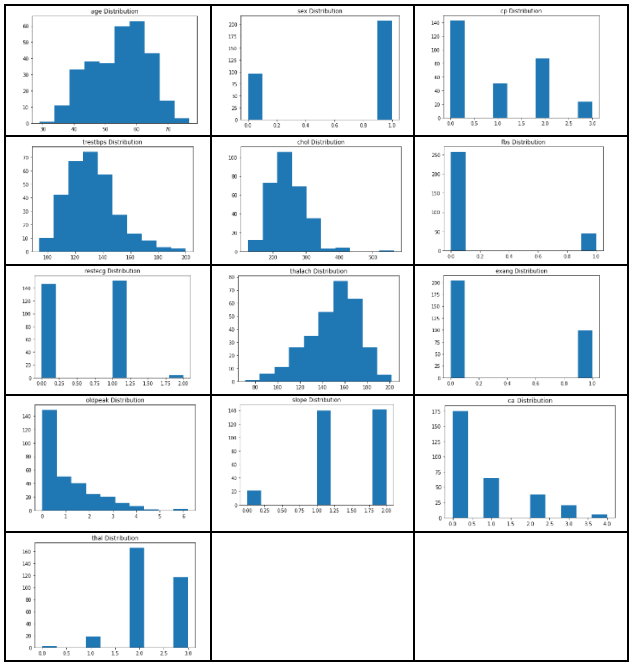
Outcome Distribution

46% of the data represents patients diagnosed with heart disease and 54% represents patients without. This dataset is nearly balanced and we should expect our accuracies should be above 54% if we expect our models to be predictive.



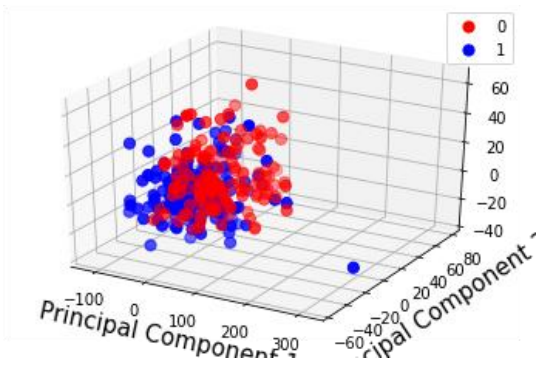
Feature Data Distribution

Left to Right: Age, Sex, CP, Trestbps, Chol, fbs, Restecg, Thalach, Exang, Old Peak, Slope, Ca, thal



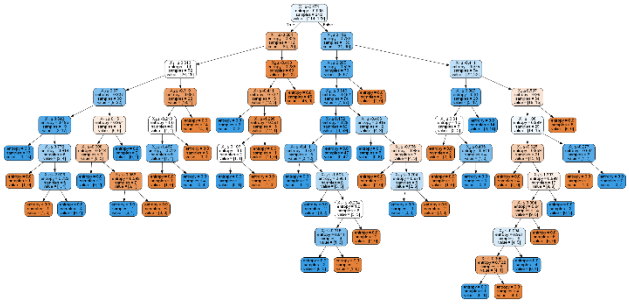
Principal Component Analysis

We can use Principal Component Analysis to reduce the dimensionality of the feature space from 13 to 3 dimensions so that we can visualize the data in the figure below. Again, clustering in the visualization is a good sign that our data can be separated and classified.

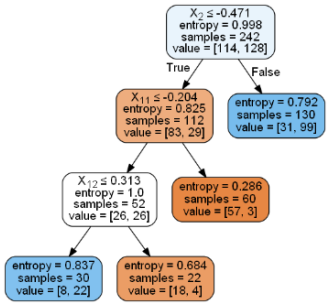


2.1 Classification using Decision Tree (With Pruning)

After creating our initial decision tree model without any form of pruning using our model we get an accuracy of 86.88%. Given that 54% of our data is a single class, this model is very predictive. Decision tree visualization:



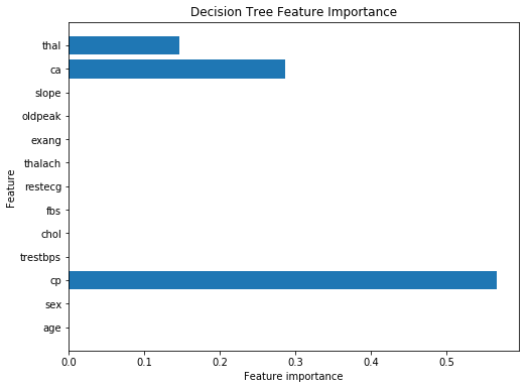
As expected, our initial decision tree is incredibly complex. Overfitting is likely occurring. We select limited the depth of nodes as our main pruning strategy since decisions at lower depth get very specific and at some point too specific that the model won't generalize well. After we limit the depth to 4 the model now achieves an accuracy of 88.5%.



Our simpler model is slightly more accurate but is less complex so is likely to generalize well. According to Occam's razor, the simpler solution should be selected.

To better understand what the model is primarily making its decisions on, we can look at a plot of the feature importance. Thalassemia (Blood Disorder), Chest Pain

Type, and CA (Number of Major Vessels colored by fluoroscopy) Glucose seem to have the highest impact on determining if a patient should be diagnosed with Heart Disease. Our decision trees selected features seem intuitive and are likely to be leading to good predictions.



When we switch our criterion for information to the Gini index and increase the max depth to 5, accuracy increases to 91.8%.

2.2 Classification using Boosted Decision Tree

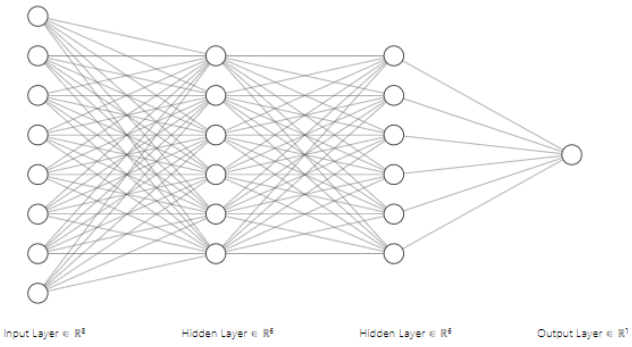
After running Adaboost on our previous decision tree that had an accuracy of 91.8% the accuracy drops down to 90.6%. While this drop in accuracy is not as severe as the one in the diabetes dataset, it may likely be the same issue. Given that the dataset is relatively small outliers may have an even bigger effect and would be further weighed by Adaptive Boost.

Again, as most features are not normally distributed there is a good chance that a few outliers are reducing accuracy. Heart Disease, too, is a complex problem that medically has various factors and thus may require more data for higher accuracy.

Changing other variables does not particularly improve performance. Changing the learning rate to anything lower than 1 only decreases the accuracy. Switching the algorithm from SAMME.R to SAMME reduces accuracy. Reducing or adding more estimators than the default 50 seems to either under fit or over fit the data.

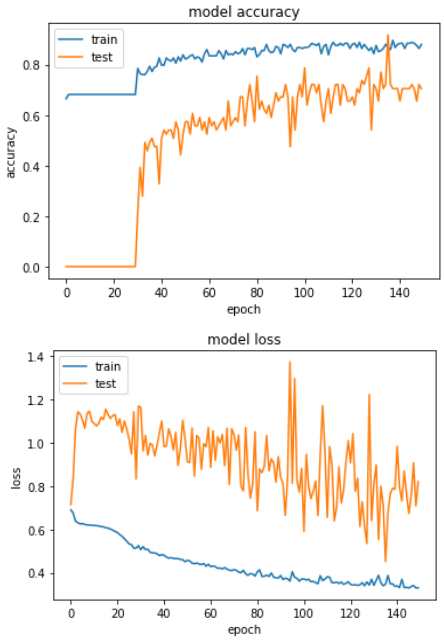
2.3 Classification using Artificial Neural Network

We can try a similar architecture approach as with the diabetes dataset: two hidden layers that are the average between the number of features (input nodes) and number of classes (output nodes). In this case, we have 13 input feature nodes and a single output node for binary classification. We will thus use 7 nodes in our hidden layers.

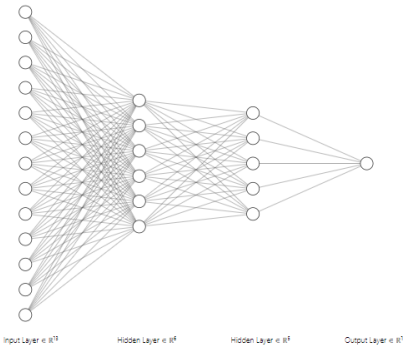


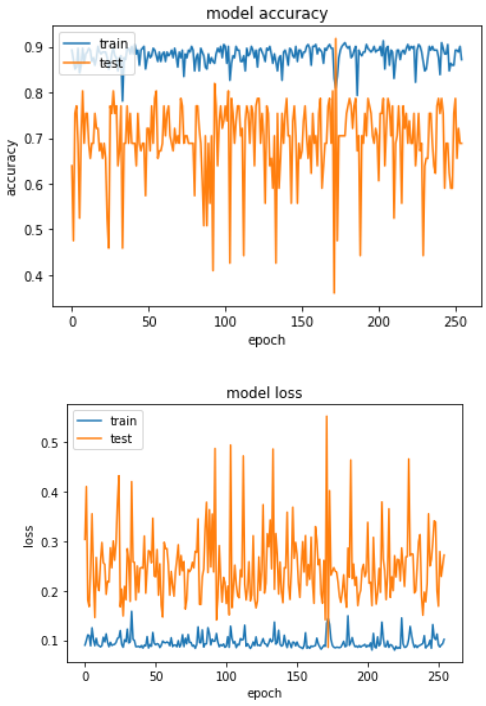
.

This resulted in an accuracy of 73.77%. The test data accuracy did not get as close to the training accuracy as we would hope. As expected the average loss for both the training and test data decreased as the number of epochs increased, although the testing data did so in a volatile manner.



We can change a few parameters to improve the performance. First we can change the activation function, relu (Rectified Linear Unit) to tanh. We can change the loss function from binary-cross entropy to mean squared errors since we have many continuous numerical values. Finally we can reduce the number of neurons in the hidden layers in case we have overfitting from 7 and 7 to 6 and 5.

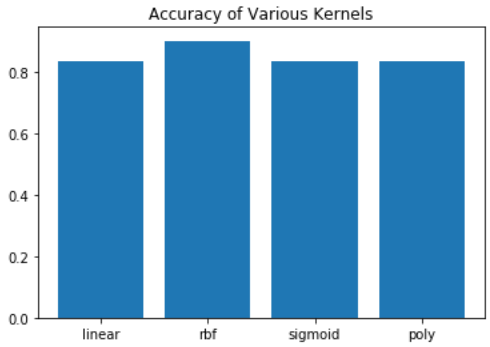




These changes have improved the accuracy to 86.89%. The loss and test graphs are extremely volatile, this could likely be because the learning rate is too high and the model is not able to converge near the optimal point.

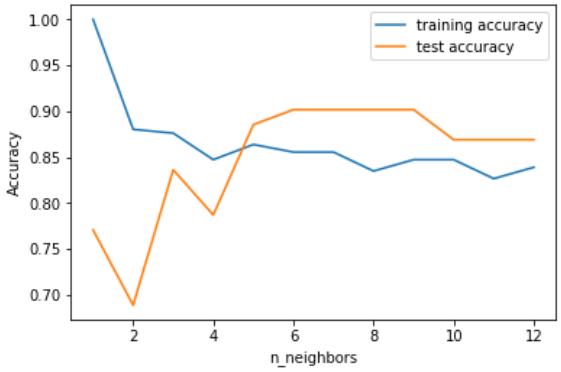
2.4 Classification using Support Vector Machine

The main difference between different SVM models is the choice of kernels for reasons explained earlier. After trying different kernels on the default hyper parameter settings we find that the rbf kernel performs the best with an accuracy of 90.2% while all other accuracies are 83.6%. This is likely rbf has a lot of freedom in that its boundary can take a very flexible shape around the data without overfitting quickly like the polynomial kernel.



2.5 Classification using K-Nearest Neighbors

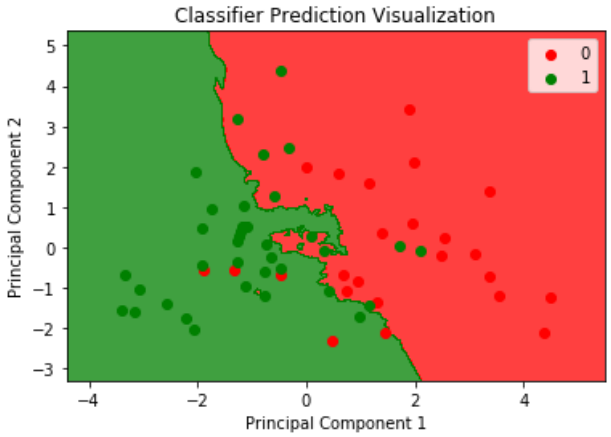
Keeping all other default settings, we can vary K from 1 to 12 to find that 7 is the optimal number for K in terms of maximizing the test accuracy to 90.16%



Notably, the K values from 6 to 9 all have a similar result. KNN can use various algorithms to compute the nearest neighbors. Once again, 'ball_tree', 'kd_tree', 'brute' and 'auto' all gave the same result (90.2%).

KNN by default uses the Euclidian distance but it can also use the Manhattan distance. Using the Manhattan distance dropped the accuracy to 86.89% making the model less predictive.

We can visualize the classification of KNN by reducing the dimensionality from 13 dimensions to 2 and plotting the points and the regions that the KNN is classifying as positive and negative as seen below. Even though much of the variance is lost as a result of dimensionality reduction we can verify that KNN separates the points fairly well.



2.6 Conclusion for Heart Disease Diagnosis

Here are the final best test results of each of the classifiers on this diabetes dataset.

Rank	Model	Result
1	**Decision Tree with Pruning**	**91.8%**
2	Boosted Decision Tree	90.6%
3	Support Vector Machine	90.2%
3	K Nearest Neighbors	90.2%
4	Artificial Neural Network	86.9%

Decision Trees with Pruning performed the best once again on this dataset, however, this time the Gini Index was used. SVM typically works well for

smaller datasets as it is easier to find a hyperplane to divide the subspace. K-Nearest neighbors worked well even despite higher dimensionality. ANN ranked the lowest but with different architectures it may be able to perform better. Overall, these accuracies are very high and the models are quite predictive.

3 Conclusion

In sum, despite the fact that both dataset were less than 1,000 instances the final accuracies of the models was relatively high. Overall, the pruned decision trees seemed to have had the best performance (highest accuracy) on both datasets. For further improvement, tuned hyper parameters can be tuned to a more specific degree and hyper parameters that were not closely considered can be revisited.

4 References

Diabetes Dataset:
https://www.kaggle.com/uciml/pima-indians-diabetes-database

Heart Disease Dataset:
https://www.kaggle.com/ronitf/heart-disease-uci