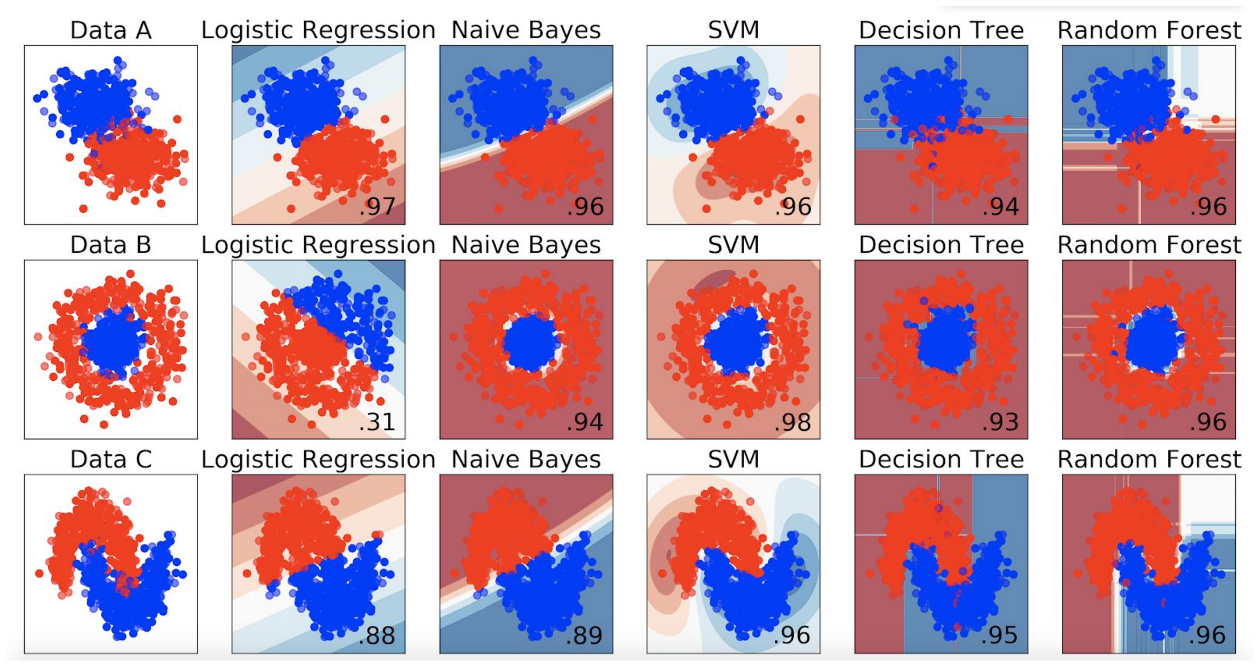
Calvin Zikakis
November 11, 2019
CSCI 3202
Professor D’Mello

Assignment 3 - Report

Part 1-

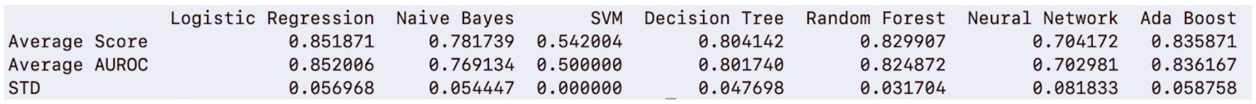


1a - Most classifiers did an excellent job with the data sets. Logistic Regression struggled the most with dataset B and C being hard for it to correctly classify. The lowest score of the classifiers was Logistic Regression on dataset B and the highest score was SVM on dataset B. The other classifiers sat around scores of 0.93 to 0.96.

1b - We can clearly see the effects the datasets shapes play on the classification score. Notice for Logistic Regression, when the data is easily linearly separable it performs great (Data A), but when the data cannot be separated with a line (or a logistic function in this case), the classifier performs very poorly. This is to be expected with less advance classifiers due to their limited ability in producing complex decision bounds.

Functions that are able to make more advanced shaped decision bounds tend to perform much better with all shapes of data. Take for instance SVM. SVM has the ability to take non-linearly separable data and map it into a new dimension where it can be linearly separated. This allows us to take the data from dataset B and map it to a new dimension where we can fit a line through the data. These transformations are called kernels and allow us to map more complex shapes of data.

Part 2.A -



```
               Logistic Regression  Naive Bayes       SVM  Decision Tree  Random Forest  Neural Network  Ada Boost
Average Score             0.851871     0.781739  0.542004       0.804142       0.829907        0.704172   0.835871
Average AUROC             0.852006     0.769134  0.500000       0.801740       0.824872        0.702981   0.836167
STD                       0.056968     0.054447  0.000000       0.047698       0.031704        0.081833   0.058758
```

2a. 1- Above you can see a table depicting the average different score each model produced as well as the standard deviation of those models.

2a. 2- Logistic Regression performed fantastically for this dataset. It ended its cross validated training and testing with an average AUROC of 0.85. This is a fantastic value for an untuned classifier. It also provided a relatively low standard deviation of only 0.057. This means this classifier's performance barely varied per fold in the dataset.

Adaboost also performed excellently on the data set with an average AUROC of 0.84 and a standard deviation of 0.058.

The worst performing classifier was SVM with an average score of 0.54. This score is close to chance for this dataset being 0.5. The standard deviation on this classifier oddly was 0. This is probably due to the standard deviation of that dimension being zero. More investigation is required here.

2a. 3- Neural Network Classifier - This classifier works by modelling neurons in the brain. This algorithm is designed to recognize patterns in data and make a prediction based off of these patterns. Data enters into a network of neurons. Between the input and the output of the neurons lies hidden units. The data travels through these weighted hidden units and is transformed into an output the classifier can then used to predict. I picked this classifier to use because I was interested in the performance differences of a Neural Net and a Decision Tree.

Adaptive Boosting - This classifier method works by converting a series of weak classifiers into a strong one. Data enters the algorithm and the algorithm fits weak classifiers to the data set and chooses the one with the lowest error. It then updates the weights of the data to be higher for more accurate classifiers. This classification technique is used in junction with another algorithm for best classification (SAMME.R in our case). I choose this classifier due to the amount of features present in the data set.

Part 2.B -

2b. 1 - I used gridsearch for my parameter tuning. This function allows you to feed it a dictionary of parameter options and it will test all of them using the AUROC metric and return the parameters that resulted in the best score. This allows me to test lots of parameters easily and fast.

2b. 2 - SVM - Gamma - This parameter decides how far the influence of each training example reaches. When gamma is high it means close, low values mean far.

C - This parameter helps to regularize the function. It controls the trade off of correct classification of training and maximization of the functions margins for decision.

Random Forest - max_depth - This represents the depth of each tree in the forest as the name implies.

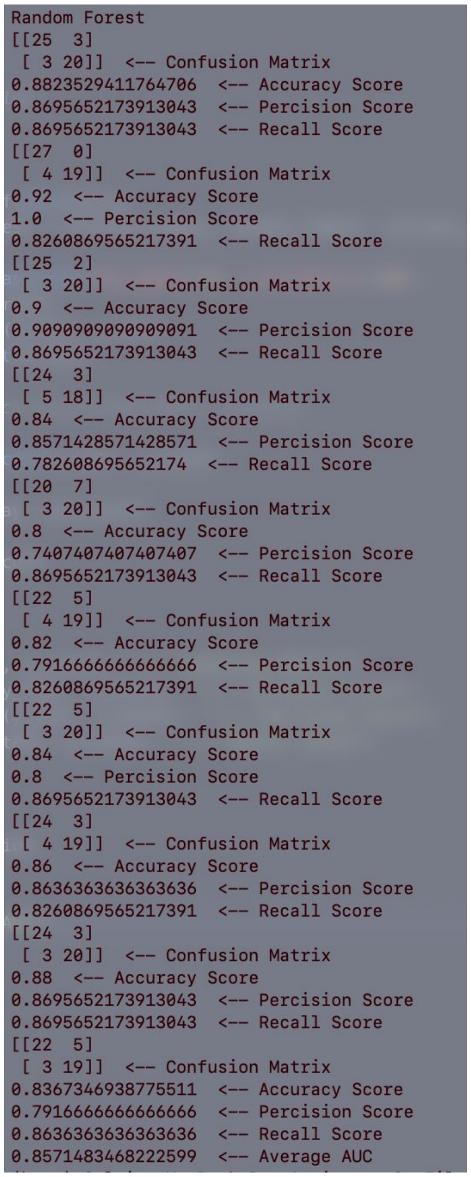
N-estimators - This parameter is used to control the number of trees in the forest.

2b. 3- SVM - I was able to improve my SVM model with hyperparameter tuning but not by much. Tuning this SVM model increased my AUROC to be 0.57.

Random Forest - I was able to improve this model when tuning my parameters to 0.865. I am happy with this tuning but I do think it is possible to get this higher with more computation time.

Part 2.C -

2c.1 -



```
Random Forest
[[25  3]
 [ 3 20]]  <-- Confusion Matrix
0.8823529411764706  <-- Accuracy Score
0.8695652173913043  <-- Percision Score
0.8695652173913043  <-- Recall Score
[[27  0]
 [ 4 19]]  <-- Confusion Matrix
0.92  <-- Accuracy Score
1.0  <-- Percision Score
0.8260869565217391  <-- Recall Score
[[25  2]
 [ 3 20]]  <-- Confusion Matrix
0.9  <-- Accuracy Score
0.9090909090909091  <-- Percision Score
0.8695652173913043  <-- Recall Score
[[24  3]
 [ 5 18]]  <-- Confusion Matrix
0.84  <-- Accuracy Score
0.8571428571428571  <-- Percision Score
0.782608695652174  <-- Recall Score
[[20  7]
 [ 3 20]]  <-- Confusion Matrix
0.8  <-- Accuracy Score
0.7407407407407407  <-- Percision Score
0.8695652173913043  <-- Recall Score
[[22  5]
 [ 4 19]]  <-- Confusion Matrix
0.82  <-- Accuracy Score
0.7916666666666666  <-- Percision Score
0.8260869565217391  <-- Recall Score
[[22  5]
 [ 3 20]]  <-- Confusion Matrix
0.84  <-- Accuracy Score
0.8  <-- Percision Score
0.8695652173913043  <-- Recall Score
[[24  3]
 [ 4 19]]  <-- Confusion Matrix
0.86  <-- Accuracy Score
0.8636363636363636  <-- Percision Score
0.8260869565217391  <-- Recall Score
[[24  3]
 [ 3 20]]  <-- Confusion Matrix
0.88  <-- Accuracy Score
0.8695652173913043  <-- Percision Score
0.8695652173913043  <-- Recall Score
[[22  5]
 [ 3 19]]  <-- Confusion Matrix
0.8367346938775511  <-- Accuracy Score
0.7916666666666666  <-- Percision Score
0.8636363636363636  <-- Recall Score
0.8571483468222599  <-- Average AUC
```

2c.2 - My model appears to do a pretty solid job in classifying if a person has good credit or not. It has a high accuracy rating as well as high precision and recall scores. This is topped off by an average AUC of 0.86.

Part 2.D -

The best settings I found for my best model are as follows below.

```
trainOnAllData(df, RandomForestClassifier(max_depth=10, n_estimators=100))
```