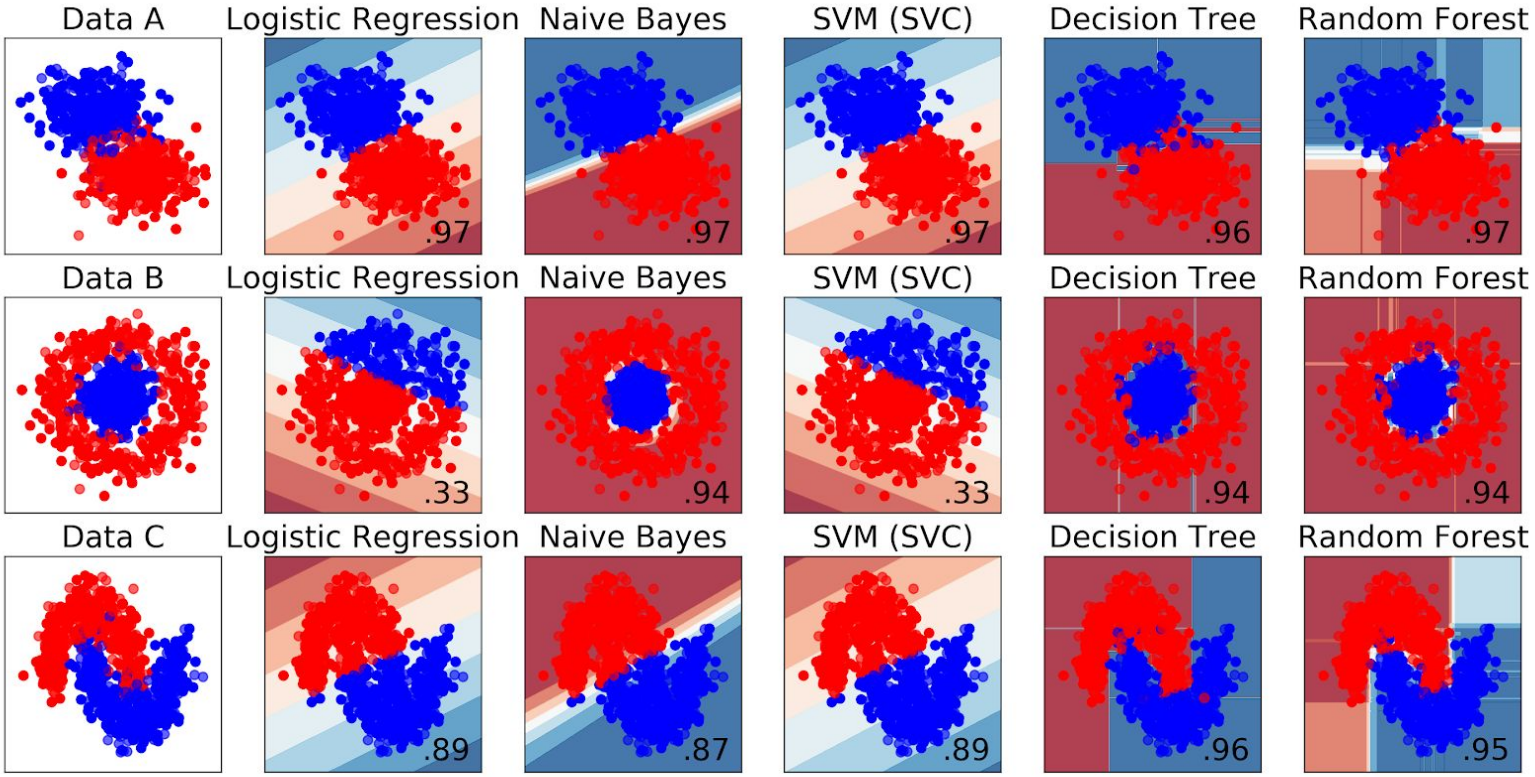
Michael Merola
CSCI3202
Assignment 3 Report
4 November 2019

Part 1, Exploring Decision Boundaries

Results:



a. Each classifier uses different techniques to learn and assign labels to datasets, so the accuracy of each classifier varies depending on the datasets linear separability. All 5 classifiers were able to label Dataset A with accuracy >= 0.96. Logistic Regression is the natural classifier for this problem as A is nearly linearly separable, and interestingly the Decision Tree has a slightly lower accuracy than the other 4. For Dataset B, the accuracy scores were split between 0.33 and 0.94. Logistic Regression and SVM do not classify the data well because it relies on linearly separable data to perform well, meanwhile Naive Bayes, Decision Tree, and Random Forest are able to apply multiple decision boundaries to classify the blue points within the red. Similarly, Decision Tree and Random Forest classify Data C well but Naive Bayes actually struggles the most to accurately label this data.

b. Based on the above performance, each classifier is uniquely good at classifying different dataset problems. Logistic Regression is the natural solution to data that is linearly

separable because it tries to fit a single line and assert labels based on fit; however, it suffers with Datasets like B and C that are more complex. All the other classifiers are able to accurately label Dataset A, but have other advantages/disadvantages when it comes to B and C. Naive Bayes can accurately classify B, but for dataset C it acts as a linear classifier and ends up being less accurate than Linear Regression. SVM is a linear classifier and has the same accuracies as Linear Regression. Decision Tree and Random Forest both utilize multiple decision boundaries and have very similar accuracies over the datasets, but it is interesting to note that Decision Tree is worse on Dataset A but better for C while Random Forest is better for A and worse for C. Overall, Decision Tree is the most consistent over the three datasets, and Random Forest has more variability between the two.

Part 2, Implementing a Machine Learning Pipeline

A. *Basic Model Building*

1.

MODEL	AVERAGE ▼	STANDARD DEV
Gradient Boosting	0.920	0.043
Logistic Regression	0.917	0.045
Random Forest	0.909	0.055
AdaBoost	0.885	0.050
Naive Bayes	0.870	0.074
SVM (SVC)	0.811	0.082
Decision Tree	0.794	0.050

2. I chose the Gradient Boosting model as the best overall fit for this data because it had both the highest average AUROC score across folds and the lowest standard deviation. This means that on average the Gradient Boost has a more acceptable ratio of true positives to false positives and performs with lower error in accuracy than the other models. The low Standard Deviation also tells us that Gradient Boost has consistent results between the scores at each fold, so having both best scores is a good indicator that we should use this model for the data. Logistic Regression is also a good overall model choice because it has the second highest average and the second lowest standard deviation.

3. Gradient Tree Boosting (or Gradient Boosted Regression Trees) is a greedy supervised learning technique that uses a loss function and a weak-learning decision tree to make predictions about data. The model is additive using gradient descent to minimize the loss while it adds trees. The gradient descent ensures that each tree's output reduces

residual loss and the model performs well. Logistic Regression is a technique for learning that uses a sigmoid function (S curve) to fit a set of data and predict the probability that a certain data point can be classified under a label. It has a threshold parameter to change the point on the curve that classifies points, and the technique is generally used for linearly separable data.

B. *Hyperparameter Tuning*

1. I utilized the *RandomizedSearchCV* library function to optimize my parameters for SVC and RandomForest classifiers. By defining a grid of parameters and values, the function performs a K-fold CV with each combination of supplied values until determining the most effective parameters. Rather than searching exhaustively, it randomly samples parameter values.

2. For SVC, the *C* parameter determines the penalty error term and *gamma* changes the number and type of features to be applied. For Random Forest, the *n_estimators* parameter is the number of trees in the forest and the *max_depth* is the max number of levels in each decision tree.

3.

MODEL	AVERAGE ▼	STANDARD DEV
Random Forest	0.9274729699	0.03942406938
SVM (SVC)	0.7311170713	0.05771842235

**parameters used:* SVC(C=1000, gamma=0.0001)
RandomForestClassifier(n_estimators=250, max_depth=96)

By using hyperparameter optimization and the AUROC metric, I was actually able to improve on the results from the no-parameter tests. The Random Forest classifier had an average score .0074 higher and a standard deviation .004 lower than the Gradient Boosting classifier's best scores.

C. *Analysis of Best Model*

Confusion Matrix

	ACTUAL/Positive	ACTUAL/Negative
PREDICTED/Positive	46	8
PREDICTED/Negative	11	60

Stats

Accuracy	0.848
Precision	0.8484611372
Recall	0.8446852425
AUROC	0.9296589584

Based on the detailed statistics and confusion matrix, I can reasonably conclude that my model has good accuracy for this dataset. The true accuracy is ~85% with precision and recall scores that are similar, meaning that the model has a strong fit for the data.

D. *Generalizability*

Model Rebuild:
sklearn.ensemble.RandomForestClassifier(n_estimators=250, max_depth=96)