**Pima Diabetes**

*Question #1a: Run the code once, record the accuracy and AUC score. What do you notice about the scores?*

Gradient Boosting Acc: 0.76 (+/- 0.07)

Gradient Boosting AUC: 0.82 (+/- 0.06)

CV Runtime: 0.31200504302978516

Ada Boosting Acc: 0.76 (+/- 0.05)

Ada Boosting AUC: 0.83 (+/- 0.06)

CV Runtime: 0.6045501232147217

Gradient boosting and Ada boosting achieved the same accuracy, which is 0.76 with slightly different standard deviation. AUC scores are also similar, which is 0.82 for gradient boosting and 0.83 for Ada boosting with same standard deviation. But Ada boosting had doubled running time for computation. Gradient boosting method on this dataset is more cost-effective.

*Question #1b: In the Scikit API for Ada Boost Classifier, it tells us that when the base\_estimator parameter is set to None, it uses a particular estimator by default. What is this default estimator, and why is it significant?*

If None, then the base estimator is DecisionTreeClassifier(max\_depth=1). Ada boosting is generally using decision stump (consisting of a one-level decision tree) which makes a prediction based on the value of just a single input feature. The base estimator from which the boosted ensemble is built.

*Question #2a: Run the code once, record the accuracy and AUC score. What do you notice about the scores? How do they compare to boosting methods? What about run times?*

Gradient Boosting Acc: 0.76 (+/- 0.07)

Gradient Boosting AUC: 0.82 (+/- 0.06)

CV Runtime: 0.28594517707824707

Ada Boosting Acc: 0.76 (+/- 0.05)

Ada Boosting AUC: 0.83 (+/- 0.06)

CV Runtime: 0.6123809814453125

Neural Network Acc: 0.70 (+/- 0.05)

Neural Network AUC: 0.71 (+/- 0.05)

CV Runtime: 1.305253028869629

Neural Network has lower accuracy and AUC compared to boosting methods, which is 0.70 vs 0.76 in terms of accuracy and 0.71 vs 0.82 in terms of AUC. Also, NN has the longest runtime among three classification algorithms, which is approximately 5 times longer than GB and double of Ada Boosting. It looks like the current NN parameter configuration is not ideal on this dataset.

*Question #2b: In the Scikit API for MLP Classifier, there are different solvers described. When might we use the ‘adam’ solver?*

‘adam’ refers to a stochastic gradient-based optimizer and works pretty well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score.

*Question #3: Run the code once for each setting of the max depth (3,5,7,10), record the accuracy and AUC scores. What do you notice about the scores as the max depth increases? What about run-times?*

Max\_depth = 3

Gradient Boosting Acc: 0.76 (+/- 0.07)

Gradient Boosting AUC: 0.82 (+/- 0.06)

CV Runtime: 0.2943859100341797

Max\_depth = 5

Gradient Boosting Acc: 0.77 (+/- 0.05)

Gradient Boosting AUC: 0.83 (+/- 0.07)

CV Runtime: 0.563629150390625

Max\_depth = 7

Gradient Boosting Acc: 0.77 (+/- 0.08)

Gradient Boosting AUC: 0.81 (+/- 0.08)

CV Runtime: 1.0901520252227783

Max\_depth = 10

Gradient Boosting Acc: 0.74 (+/- 0.07)

Gradient Boosting AUC: 0.80 (+/- 0.09)

CV Runtime: 1.8009307384490967

As the number of max depth increased from 3 to 7, accuracy and AUC didn’t change much. When max depth increased to 10, accuracy slightly dropped from 0.77 to 0.74 but not outside the expected range of variance. Runtime showed more radical change as max depth tripled from 3 to 10, cv runtime was 6 times longer than initial time, which is 1.80 vs 0.29.

*Question #4a: Run the code once, record the accuracy and AUC scores. What do you notice about the scores? How do they compare to the performance above for Gradient Boosting, Ada Boosting, and Neural Networks with no feature selection? Did you notice any changes in run-times?*

Gradient Boosting Acc: 0.77 (+/- 0.05)

Gradient Boosting AUC: 0.83 (+/- 0.07)

CV Runtime: 0.2373650074005127

Performance similar to the CV with all features in gradient boosting with slightly improvement in accuracy, AUC and CV runtime. When compared to Ada boosting with no feature selection, the performance was similar but shorter CV runtime.  Performance is overall better than neural network in any case. In other words we were able to build a model with less features that had similar performance.

*Question #4b: What features were selected, and which were removed? Were there any differences from when you did feature selection with Random Forests in HW2?*

--FEATURE SELECTION ON--

Wrapper Select:

Selected ['Blood Glucose', 'BMI', 'Age']

Features (total/selected): 8 3

Removed: ['Times Pregnant', 'Blood Pressure', 'Skin Fold Thickness', '2-Hour Insulin', 'Family History']

'Family History' was chosen in random forest while it got eliminated in boosting method. The remaining 3 features were same.

*Question #5: Run the code once for each setting of the solver, record the accuracy and AUC scores. What do you notice about the scores when we change the solver? What about run-times?*

Neural Network Acc: 0.72 (+/- 0.07)

Neural Network AUC: 0.80 (+/- 0.08)

CV Runtime: 2.1902809143066406

Performance similar to the CV with all features in Question #1, with perhaps slight reduced performance if any, but not outside the standard deviations we saw in Question #1. Runtime was sharply increased 7 times compared to using ‘lbgfs’ solver. This is reasonable as for small datasets ‘lbfgs’ can converge faster and perform better.

**Wine Quality Dataset**

*Question #6a: Run the code once, record the RMSE and Explained Variance.*

Gradient Boosting RMSE:: 0.64 (+/- 0.01)

Gradient Boosting Expl Var: 0.34 (+/- 0.12)

CV Runtime: 0.3038630485534668

Ada Boosting RMSE:: 0.66 (+/- 0.03)

Ada Boosting Expl Var: 0.31 (+/- 0.16)

CV Runtime: 0.9847791194915771

*Question #6b: In the Scikit API for Gradient Boost Regressor, what do you think is the purpose of the learning rate parameter (hint: do some googling)?*

Learning rate determines the impact of each tree on the final outcome (Update the output with current results taking into account the learning rate). Gradient boosting works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates.

Lower values are generally preferred as they make the model robust to the specific characteristics of tree and thus allowing it to generalize well.

*Question #7a: Run the code once, record the RMSE and Explained Variance. What do you notice about the scores? How do they compare to boosting methods? What about run times?*

Neural Network RMSE:: 0.66 (+/- 0.05)

Neural Network Expl Var: 0.29 (+/- 0.17)

CV Runtime: 1.8054821491241455

Neural Network showed similar performance as compared to boosting methods. Standard deviation was increased to 0.05 as compared to 0.01 in gradient boosting and 0.03 in Ada boosting, but not outside the expected range of variation, so the performance of the models cannot be considered different. Also, NN has the longest runtime among three regression algorithms, which is approximately 6 times longer than GB and double of Ada Boosting.

*Question #7b: In the Scikit API for MLP Regressor, if you wanted to create a neural network to have two hidden layers of 10 and 10, instead of just a single hidden layer of 20, how would you set the hidden\_layers parameter equal to in the function call?*

Change parameter hidden\_layer\_sizes=(10, 10)

*Question #8a: Run the code once, record the accuracy and AUC score. What do you notice about the scores? How do they compare to the regression scores (or can you compare them)?*

Gradient Boosting Acc: 0.73 (+/- 0.06)

Gradient Boosting AUC: 0.81 (+/- 0.06)

CV Runtime: 0.4508168697357178

Ada Boosting Acc: 0.74 (+/- 0.05)

Ada Boosting AUC: 0.82 (+/- 0.06)

CV Runtime: 0.8588459491729736

Neural Network Acc: 0.73 (+/- 0.06)

Neural Network AUC: 0.81 (+/- 0.06)

CV Runtime: 3.2843830585479736

Performance of three classification methods were similar with average accuracy of 0.73 and AUC 0.81. Average RMSE of regression methods were 0.65 and explained about 30% of the variance. It looks like reformat the problem into a classification problem would achieve a better prediction result.

*Question #8b: Look at the bins that were created (some info should be printed out about the # of samples in each bin, and min and max values). How would you explain what you did to your boss or customer? What are we actually predicting here?*

Bin 0 : 3.0 5.0 744

Bin 1 : 6.0 8.0 855

Bin0 represents a group of 744 observations with lowest score of 3 and highest score of 5, which could be labeled as low-class wine. Bin1 represents a group of 855 observations with lowest score of 6 and highest score of 8, which could be labeled as high-class wine. We change the question of predicting the exact class/score of a certain wine into predicting a general binary type of wine, low-class or high-class.

*Question #9: Run the code once, record the accuracy and AUC score. What do you notice about the scores? How do they compare to results in Question #8a?*

Gradient Boosting Acc: 0.73 (+/- 0.06)

Gradient Boosting AUC: 0.81 (+/- 0.06)

CV Runtime: 0.43912720680236816

Ada Boosting Acc: 0.74 (+/- 0.05)

Ada Boosting AUC: 0.82 (+/- 0.06)

CV Runtime: 0.8344099521636963

Neural Network Acc: 0.73 (+/- 0.06)

Neural Network AUC: 0.81 (+/- 0.06)

CV Runtime: 3.315871000289917

Performance similar to the CV without normalization in Question #8a. Boosting trees is about building multiple decision trees. Decision tree doesn't require feature normalization, that's because the model only needs the absolute values for branching. However, it's always a good idea to normalize features because it's easier to visualize and interpret model and compare another model (e.g. SVM) with the same data set.

*Question #10a: Run the code once for both settings of target discretization (binning either 0 or 1). Record the accuracy and AUC scores for binned data, and the RMSE and Explained Variance Scores for un-binned data. What do you notice about the scores? How do they compare to performance above for Gradient Boosting, Ada Boosting, and Neural Networks with no feature selection? Did you notice any changes in run-times?*

binning = 0

Gradient Boosting RMSE:: 0.66 (+/- 0.02)

Gradient Boosting Expl Var: 0.29 (+/- 0.14)

CV Runtime: 0.1479642391204834

Ada Boosting RMSE:: 0.67 (+/- 0.04)

Ada Boosting Expl Var: 0.30 (+/- 0.13)

CV Runtime: 0.25784921646118164

Neural Network RMSE:: 0.65 (+/- 0.03)

Neural Network Expl Var: 0.32 (+/- 0.09)

CV Runtime: 1.6269168853759766

binning = 1

Gradient Boosting Acc: 0.73 (+/- 0.05)

Gradient Boosting AUC: 0.81 (+/- 0.05)

CV Runtime: 0.3073000907897949

Ada Boosting Acc: 0.73 (+/- 0.06)

Ada Boosting AUC: 0.82 (+/- 0.06)

CV Runtime: 0.6897540092468262

Neural Network Acc: 0.73 (+/- 0.06)

Neural Network AUC: 0.81 (+/- 0.06)

CV Runtime: 3.2863149642944336

Performance similar to the CV with all features in Question #6a, #7a and #8a. There was a slight drop in regression performance, but not outside the expected range of variation, so the performance of the models cannot be considered different. Runtime was significantly reduced with feature selection. In other words we were able to build a model with less features that had similar performance with less computation time.

*Question #10b: What features were selected, and which were removed? How do those features differ between binned vs. un-binned runs?*

binning = 0

--FEATURE SELECTION ON--

Wrapper Select:

Selected ['volatile acidity', 'sulphates', 'alcohol']

Features (total/selected): 11 3

Removed: ['fixed acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH']

binning = 1

--FEATURE SELECTION ON--

Wrapper Select:

Selected ['volatile acidity', 'total sulfur dioxide', 'sulphates', 'alcohol']

Features (total/selected): 11 4

Removed: ['fixed acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'density', 'pH']

'total sulfur dioxide' was selected in binning method but eliminated in un-binned data.

**Summary Questions**

*Question #11: Compare the performance of Boosting Methods and Neural Networks here compare to previous methods (decision trees, random forests) from prior Homeworks for both datasets. Did they perform better, worse, or the same in terms of both evaluation scores and run-times? If your boss or customer asked why that might be, how would you explain?*

For classification problem, boosting methods showed similar performance to random forest with approximately 0.76 accuracy and slightly less CV runtime, while neural network achieved a similar accuracy as decision tree but with distinct more computation cost. For regression problem, boosting methods performed slightly better than random forest, especially gradient boosting, which achieved similar result with much less runtime. Neural network could also achieve a similar performance but request much longer computation cost.

Boosting is based on weak learners (high bias, low variance). In terms of decision trees, weak learners are shallow trees, sometimes even as small as decision stumps (trees with two leaves). On the other hand, Random Forest uses fully grown decision trees (low bias, high variance), which is why it takes longer time to tun compared to boosting.

Overall, boosting trees perform better than random forest but with a high price to pay, which is boosting have a few hyperparameters to tune, while random forest is practically tuning-free.

*Question #12: Can we say anything interesting about diabetes based on the features that were selected, if we were for instance trying to create a diabetes screening program for a local healthcare organization?*

Diabetes is correlated with blood glucose, BMI and age, especially with high glucose and BMI. Eyeballing rules are listed as:

* For those who are under 25 years old, if BMI greater than 30 or blood glucose greater than 100, they are very likely to have diabetes.
* For those whose age in range [25, 44], if BMI greater than 30 and blood glucose greater than 100, they are very likely to have diabetes.
* For those whose age in range [45, 64], if BMI greater than 30 and blood glucose greater than 110, they are very likely to have diabetes.
* For those who are older than 65, if BMI greater than 30 and blood glucose greater than 140, they are very likely to have diabetes.