**Pima Diabetes**

*Question #1a: Run the code 5 times, record the accuracy and AUC score. What do you notice about the scores?*

--ML Model Output--

Random Forest Acc: 0.7546468401486989

Random Forest AUC: 0.8175987841945289

--ML Model Output--

Random Forest Acc: 0.7843866171003717

Random Forest AUC: 0.8398510242085662

--ML Model Output--

Random Forest Acc: 0.7843866171003717

Random Forest AUC: 0.8422145953757225

--ML Model Output--

Random Forest Acc: 0.7472118959107806

Random Forest AUC: 0.7988808227465215

--ML Model Output--

Random Forest Acc: 0.7509293680297398

Random Forest AUC: 0.8375766148043375

The scores of accuracy and AUC appear to oscillate. Accuracy fluctuates between 0.75 and 0.78 and AUC goes up and down, which varies from 0.80 to 0.84. The higher accuracy, the larger AUC. Accuracy changed within 4%, so the model result is stable and consistent.

*Question #1b: For the fit() method of a RandomForestClassifier, it lists three possible parameters on the API webpage, what are they? Define what you could pass in to each one?*

Parameters: X, y, sample\_weight

X : array-like or sparse matrix of shape = [n\_samples, n\_features]. The training input samples. In this case, we can pass in data\_train.

y : array-like, shape = [n\_samples] or [n\_samples, n\_outputs]. The target values (class labels in classification, real numbers in regression). In this case, we can pass in target\_train.

sample\_weight : array-like, shape = [n\_samples] or None. Sample weights. If None, then samples are equally weighted. In this case, we used default setting that sample weight was set to none.

*Question #2: Run the code once, record the accuracy and AUC score.*

--ML Model Output--

Random Forest Acc: 0.77 (+/- 0.08)

Random Forest AUC: 0.83 (+/- 0.07)

CV Runtime: 0.5776009559631348

Cross validation result is consistent with results in question #1a.

*Question #3: Run the code once for each setting of the number of trees (5,10,20,50,100,200,500, 1000), record the accuracy and AUC scores. What do you notice about the scores? How do they change as the number of trees increases?*

n\_estimators=5

--ML Model Output--

Random Forest Acc: 0.74 (+/- 0.06)

Random Forest AUC: 0.77 (+/- 0.07)

CV Runtime: 0.06230974197387695

n\_estimators=10

--ML Model Output--

Random Forest Acc: 0.74 (+/- 0.05)

Random Forest AUC: 0.80 (+/- 0.08)

CV Runtime: 0.08503389358520508

n\_estimators=20

--ML Model Output--

Random Forest Acc: 0.76 (+/- 0.06)

Random Forest AUC: 0.81 (+/- 0.07)

CV Runtime: 0.14176392555236816

n\_estimators=50

--ML Model Output--

Random Forest Acc: 0.77 (+/- 0.06)

Random Forest AUC: 0.83 (+/- 0.07)

CV Runtime: 0.3154129981994629

n\_estimators=200

--ML Model Output--

Random Forest Acc: 0.77 (+/- 0.07)

Random Forest AUC: 0.83 (+/- 0.06)

CV Runtime: 1.1612539291381836

n\_estimators=500

--ML Model Output--

Random Forest Acc: 0.78 (+/- 0.07)

Random Forest AUC: 0.83 (+/- 0.06)

CV Runtime: 2.8338210582733154

n\_estimators=1000

--ML Model Output--

Random Forest Acc: 0.77 (+/- 0.07)

Random Forest AUC: 0.83 (+/- 0.07)

CV Runtime: 5.661194801330566

As we increased the number of trees, model accuracy showed limited improvement, which increased by 5.4% from 0.74 to 0.78. AUC scores increased by 7.8% from 0.77 to 0.83. After 50 trees, both accuracy and AUC score seemed stable and showed little change. When we double the number of trees, runtime also doubled each time.

*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 in question #2?*

--ML Model Output--

Random Forest Acc: 0.76 (+/- 0.05)

Random Forest AUC: 0.82 (+/- 0.06)

CV Runtime: 0.6006970405578613

Accuracy dropped from 0.77 to 0.76 and standard deviation dropped from 0.08 to 0.05. AUC scored also dropped from 0.83 to 0.82 and standard deviation dropped from 0.07 to 0.06. This means the model generated a narrower range compared to question #2. Since we implemented feature selection, the process took slightly longer to compute, which is 0.02s difference.

*Question #4b: What features were selected, and which were removed?*

--FEATURE SELECTION ON--

Wrapper Select:

Selected: ['Blood Glucose', 'BMI', 'Family History', 'Age']

Features (total/selected): 8 4

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

*Question #5: Run the code once, record the accuracy and AUC scores. What features were selected, and which were removed? How do the selected features compare to what you saw in Question #4 above? Was the performance (accuracy, AUC) different than in Question #4?*

--FEATURE SELECTION ON--

Selected: ['Blood Glucose', 'BMI', 'Family History', 'Age']

Features (total/selected): 8 4

--ML Model Output--

Random Forest Acc: 0.76 (+/- 0.05)

Random Forest AUC: 0.82 (+/- 0.06)

CV Runtime: 0.6113541126251221

The same 4 features as in question #4 were selected in this model, i.e. 'Blood Glucose', 'BMI', 'Family History' and 'Age'. Both accuracy and AUC scores are same as in question #4. Runtime was slightly longer with 0.01s difference.

**Wine Quality Dataset**

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

--ML Model Output--

Random Forest RMSE:: 0.65 (+/- 0.02)

Random Forest Expl Var: 0.33 (+/- 0.11)

CV Runtime: 0.8945357799530029

*Question #7a: Run the code once for each setting of the number of trees (5,10,20,50,100,200,500, 1000), record the RMSE and Expl Variance. What do you notice about the scores? How do they change as the number of trees increases? Is this the same as you for the Diabetes dataset in Question #3?*

n\_estimators=5

--ML Model Output--

Random Forest RMSE:: 0.70 (+/- 0.04)

Random Forest Expl Var: 0.20 (+/- 0.13)

CV Runtime: 0.06547808647155762

n\_estimators=10

--ML Model Output--

Random Forest RMSE:: 0.67 (+/- 0.04)

Random Forest Expl Var: 0.28 (+/- 0.09)

CV Runtime: 0.1088259220123291

n\_estimators=20

--ML Model Output--

Random Forest RMSE:: 0.66 (+/- 0.04)

Random Forest Expl Var: 0.30 (+/- 0.09)

CV Runtime: 0.19843578338623047

n\_estimators=50

--ML Model Output--

Random Forest RMSE:: 0.65 (+/- 0.03)

Random Forest Expl Var: 0.32 (+/- 0.11)

CV Runtime: 0.46114182472229004

n\_estimators=200

--ML Model Output--

Random Forest RMSE:: 0.64 (+/- 0.02)

Random Forest Expl Var: 0.34 (+/- 0.10)

CV Runtime: 1.7489662170410156

n\_estimators=500

--ML Model Output--

Random Forest RMSE:: 0.64 (+/- 0.02)

Random Forest Expl Var: 0.34 (+/- 0.11)

CV Runtime: 4.3421947956085205

n\_estimators=1000

--ML Model Output--

Random Forest RMSE:: 0.64 (+/- 0.02)

Random Forest Expl Var: 0.34 (+/- 0.10)

CV Runtime: 8.663026094436646

As we increased the number of trees, model RMSE showed limited improvement, which decreased by 8.6% from 0.70 to 0.64. AUC scores increased by 70%, which is from 0.20 to 0.34. Standard deviation of RMSE and AUC didn’t have distinct changes. It maintained the pattern in question #3 that even though the number of trees increased, model performance didn’t improve much.

*Question #7b: What about run-times, how do those change as you change the number of trees? What do the changes in scores and run-times tell us about choosing the right number of trees?*

As we double the number of trees, runtime also doubled, ranging from 0.07s to 8.66s. As we noticed that after 50 trees, the result almost stopped changing. We could set up an early stopping filter when we observe less than 5% change, which help us minimize computing time and cost.

*Question #8a: Run the code once, record the RMSE and Expl Variance. What do you notice about the scores? How do they compare to the performance above in question #6?*

--ML Model Output--

Random Forest RMSE:: 0.68 (+/- 0.03)

Random Forest Expl Var: 0.25 (+/- 0.11)

CV Runtime: 0.5873231887817383

Model RMSE went up by 0.03 and standard deviation went down by 0.01 which led explained variance to decrease from 0.33 to 0.25 with no standard deviation change. 3 instead of 11 features were computed so that runtime was improved from 0.89 to 0.59. This means that we could implement feature selection to improve model performance with respect to cost of time while maintaining a similar result.

*Question #8b: What features were selected, and which were removed?*

--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']

*Question #9: Run the code once and record the RMSE and Expl Variance. What do you notice about the scores? How do they differ from question #6?*

--ML Model Output--

Random Forest RMSE:: 0.65 (+/- 0.02)

Random Forest Expl Var: 0.33 (+/- 0.11)

CV Runtime: 0.8748352527618408

The scores were consistent with previous mode results, high RMSE and low explained variance. No change was observed on RMSE and explained variance compared with question #6, except slightly faster runtime after feature normalization. Random Forest is a tree-based model and hence does not require feature scaling.

*Question #10a: Run the code 5 times and record the RMSE and Expl Variance. What do you notice about the scores? How stable are they? How do they differ from question #6?*

--ML Model Output--

Decision Tree RMSE: 0.6139014578904337

Decision Tree Expl Var: 0.4187038857120168

--ML Model Output--

Decision Tree RMSE: 0.6068654828590222

Decision Tree Expl Var: 0.40829327367146295

--ML Model Output--

Decision Tree RMSE: 0.5978831706230622

Decision Tree Expl Var: 0.422769078418023

--ML Model Output--

Decision Tree RMSE: 0.6375735251718032

Decision Tree Expl Var: 0.39229894503478335

--ML Model Output--

Decision Tree RMSE: 0.647357265017879

Decision Tree Expl Var: 0.40373940488167925

After running the model 5 times, the best RMSE reached 0.59 with explained variance equal to 0.42. RMSE changed between 0.60 to 0.65. Even if the highest RMSE using this model (0.65) would explain 0.40 of the variances. Bagging results as a whole are better than model in question #6, i.e. lower RMSE and higher explained variance.

*Question #10b: Based on the API webpage for the BaggingRegressor() in the accompanying API links document, what two parameters do we need to change to create a Random Subspaces model?*

When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces. By definition, we need to change max\_features (default=1.0) and bootstrap\_features (default=False).

*Question #10c: Based on the API webpage for the BaggingRegressor(), notice that the function when called also calculates the out-of-bag error (oob\_score). Should we be using that metric then rather than a test/train split or cross-validation with bagging, or can we use them together (hint: do some googling)?*

As compared to the validation score OOB score is computed on data that was not necessarily used in the analysis of the model. Whereas for calculation validation score, a part of the original training dataset is actually set aside before training the models. Additionally, the OOB score is calculated using only a subset of DTs not containing the OOB sample in their bootstrap training dataset. This leads to reducing the overall aggregation effect in bagging. While the validation score is calculated using all the DTs of the ensemble. Thus in general, validation on a full ensemble of DTs is better than a subset of DT for estimating the score. However, occasionally the dataset is not big enough and hence set aside a part of it for validation is unaffordable. Consequently, in cases where we do not have a large dataset and want to consume it all as the training dataset, the OOB score provides a good trade-off.

**Summary Questions**

*Question #11: Compare the performance of Random Forests here to the Decision Tree models for both datasets in Homework #1. Did Random Forest perform better, worse, or the same? If your boss or customer asked why that might be, how would you explain?*

Random forest did better than decision tree on both datasets. For classification problem, random forest increased accuracy from range [0.70, 0.73] in decision tree to [0.75, 0.78]. For regression problem, random forest lowered RMSE to from 0.80 in decision tree to 0.60 and increased explained variance from 0 to 0.40. Such outperformance is due to ensemble, simply put is building multiple trees with various combinations of features from the same dataset at the same time. The final model is voted on the best one among them all. Instead of computing just one possible outcome like decision tree, random forest tests different outcomes.

*Question #12: We’ve now seen several different sampling and evaluation techniques. When it comes to evaluating model performance, what is the “gold standard” approach?*

The best way to evaluate the performance of a model would be to make predictions for new data to which the answers are known. The second-best way is to use techniques like random train and test split and k-fold cross validation to measure how well the model generalizes.