Homework #2

Chaonan Shi (1901412)
Casey Bennett, PhD
DSC540, Winter 2019
DePaul University

Pima Diabetes

1) First, let's run a simple test/train split using a random forest. To do so, we need to do two things, first create a Random Forest Classifier object (clf), then "fit" some data using that object.

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

Performance/Times	1	2	3	4	5
Accuracy	0.75	0.76	0.78	0.74	0.74
AUC	0.80	0.83	0.83	0.79	0.80

Analysis:

By running code 5 times, I found that Accuracy score has changed each time roughly in range 0.74~0.78; and the AUC score as changed each time roughly in range 0.79~0.83. From my point of view, the changing of result score from Accuracy and AUC is because of data split and Random Forest algorithm.
1. Since each time we run the code and processing of data split will select different data as Training & Testing. In this case, I am not surprise that fluctuation of Accuracy Score and the ROC score. Also since AUC is calculating the possibility of positive sample over negative sample:

$$AUC = P(P_{postive} > P_{negative})$$

So For ROC score, close to 0.5 represents the positive sample= negative sample.
2. For random forest algorithm, since this algorithm build by bunch of Decision Trees, and iterate through each tree to determine the best one. Thus, this is another reason to make our performance become differently.

However, the rang from both ACC and ROC is not quite huge, so we can consider it as stable. Moreover, I believe the CV (Cross Validation) is till necessary for random forest algorithm. The crucial point for correctness is the implicit assumption that each row of your data is an independent case (out-of-bag). If this assumption is not met, the out-of-bag estimate will be overoptimistic.

**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?*

Name	X	Y	Sample_weight
Description	The training input samples Also be described as dependent variable, parameter, and schema as well.	The target values (class labels in classification, real numbers in regression). Also be described as independent variable or output.	Sample weights. If None, then samples are equally weighted. From my point of view, we could pass in different weight for child node to achieve different purposes.

2) Let's repeat step 1 above, this time using cross-validation. To do so, we need to do two things, first create a Random Forest Classifier object (clf), then second pass that object and some data arrays into a "cross-validate" function.

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

Performance	*ACC*	*AUC*	*CV Runtime*
Scores	*0.77*	*0.83*	*1.067*

3) Let's explore how the number of trees affects performance of a Random Forest.

**Question #3: Run the code <u>once</u> 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?*

Performance/Times	*1(5)*	*2(10)*	*3(20)*	*4(50)*	*5(100)*	*6(200)*	*7(500)*	*8(1000)*
Accuracy	*0.74*	*0.74*	*0.76*	*0.77*	*0.77*	*0.77*	*0.78*	*0.77*
AUC	*0.77*	*0.80*	*0.81*	*0.83*	*0.83*	*0.83*	*0.83*	*0.83*
Running Time	*0.08*	*0.12*	*0.22*	*0.51*	*0.95*	*1.87*	*4.69*	*9.40*

<u>Analysis:</u>
By changing number of trees from 5 to 1000, I found that Accuracy score and AUC score has become stable after setting over 50 number of trees. The ACC score increased from 0.74 to 0.77; and AUC score increased from 0.77 to 0.83 respectively.

Throughout this experiment, the results illustrate that there are certain threshold for number of random forest trees, increasing number of tree may not improve accuracy and roc as we expect. On the top of that, since number of tree increased, running time also increased significantly from 0.08 to 9.40.

From my point of view, it is important that determine what the best number of trees. Not only for increasing precision, but also decreasing the cost of running time.

4) Now let's try applying feature selection method we used for the wine dataset in Homework #1 to the diabetes dataset. We will turn on the Wrapper-Based Feature Selection, which essentially builds lots of models with different subsets of features, and picks the subset that performs the best. For simplicity here though, we will just build a single subset and select the top variables. We will use the same Random Forest model for this.

**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?*

Performance	*ACC*	*AUC*	*CV Runtime*
Scores	*0.76*	*0.82*	*1.069*

Total Features (9)	*Class*	*Blood Glucose*	*BMI*	*'Family History'*	*'Age'*	*Times Pregnant*	*Blood Pressure*	*Skin Fold Thickness*	*2-Hour Insulin*
Selected (4)		√	√	√	√				

Analysis:
By comparing the performance from question 2, I found that scores remaining same level. In this case, we can summarized that features are least significant:

Times Pregnant	*Blood Pressure*	*Skin Fold Thickness*	*2-Hour Insulin*

So that we can drop off them from our model, only keeping:

Class	*Blood Glucose*	*BMI*	*'Family History'*	*'Age'*

and that would not affect our model's performance significantly.

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

Analysis:
As I mentioned above, the features: Class, Blood Glucose, BMI, Family History, Age been selected during the feature selection process;

And features: Times Pregnant, Blood Pressure, Skin Fold Thickness, 2-Hour Insulin been removed from model.

In the summary, since we done this job by using wrapper selection, even the result could be reliable than others, but it is worth to try other selections in this case, eg: filter and embedded, etc, to see the difference among these algorithms.

5) Random Forests and similar tree methods also produce a "feature importance" score that can also be used for feature selection. Let's try manually setting up a feature selection section for that.

** 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?*

Analysis:

Total Features (9)	*Class*	*Blood Glucose*	*BMI*	*'Family History'*	*'Age'*	*Times Pregnant*	*Blood Pressure*	*Skin Fold Thickness*	*2-Hour Insulin*
Selected (4)		√	√	√	√				

Performance	*ACC*	*AUC*	*CV Runtime*
Scores	*0.76*	*0.82*	*1.1023*

In the summary, the features been selected from 'feature importances' are same as question#4. From my point of view, the algorithm from question2 called 'Wrapper Select' is based on the classifier performance and focus on the "optimal" feature subset by iteratively selecting features. Indeed, there is several disadvantages for that: easy to overfitting, caused by multicollinearity; cost significant computation time when the number of features is large;

On the other hand, the 'feature importances' based on the gradient boosting algorithm. So The more an attribute is used to make key decisions with decision trees, the higher its relative importance. In this case, the accuracy of 'feature importances' will heavily depending on number of trees.

Wine Quality Dataset
Open up HW2_Wine.py … First, let's repeat the steps we did above for Diabetes, with some tweaks. We will skip the Train/Test version, and jump right to the Cross-Val version.

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

RMSE	Explained_Variance_Score	Running Time
0.65	0.33	1.35

Analysis:

For RMSE, it standards for standard deviation of the residuals (prediction errors). RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

For Explained_Variance, which also measured by R^2, which standards for how accuracy of dependent variable explained by independent variables.

**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?*

Performance/Times	*1(5)*	*2(10)*	*3(20)*	*4(50)*	*5(100)*	*6(200)*	*7(500)*	*8(1000)*
RMSE	*0.70*	*0.67*	*0.66*	*0.65*	*0.65*	*0.64*	*0.64*	*0.64*
Expl Variance	*0.20*	*0.28*	*0.30*	*0.32*	*0.33*	*0.34*	*0.34*	*0.34*
Running Time	*0.086*	*0.15*	*0.29*	*0.68*	*1.35*	*2.65*	*6.80*	*13.17*

Analysis:
By changing number of trees from 5 to 1000, I found that RMSE score and Expl Variance score has become stable after over 50 number of trees. The RMSE score decreased from 0.70 to 0.64; and Expl Variance score increased from 0.20 to 0.34 respectively.

Throughout this experiment, the results roughly same as question 3 in Diabetes dataset, illustrating that there are certain threshold for number of random forest trees, increasing number of tree may not improve explained variance and RMSE as we expect. On the top of that, since number of tree increased, running time also increased significantly from 0.086s to 13.17s

From my point of view, it is important that determine what the best number of trees. Not only for increasing precision and decreasing RMSE, but also decreasing the cost of running time.

**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?*

Analysis:
As I mentioned above, running time also increased significantly from 0.086s to 13.17s by increasing number of trees from 5 to 1000. Moreover, throughout this experiment, the result illustrates that increase trees does not guarantee higher explained variance and lower RMSE after certain threshold, and it only costs us more computation time.

**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?*

RMSE	Explained_Variance_Score	Running Time
0.68	0.25	0.97

Analysis:

By comparing with performance in question #6, the result of RMES and Explained_Variance_Score roughly stable at same level. However, the running time of this selection much more lower then the running time at question #6 (1-(0.97/1.35)) which is 30% faster.

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

Total Features (10)	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
Selected (3)		√								√	√

Analysis:

From above table, we can summarize that features: volatile acidity, sulphates, and alcohol been selected during Wrapper Selection;
Also, features: fixed acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, and PH been removed from dataset.

**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?*

RMSE	Explained_Variance_Score	Running Time
0.65	0.33	1.35

Analysis:

By comparing with performance in question #6, the result of RMES Explained_Variance_Score, and running time from this question stay at exact same level. From my point of view, I support that scores will increase after standardizing data frame. However, the results illustrate that the data frame is normal distributed already, so that normalize function does not improve the result significantly.

**Question #10a: Run the code <u>5 times</u> 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?*

Performance/Times	1	2	3	4	5
RMSE	0.66	0.65	0.62	0.65	0.62
Explained_Variance_Score	0.40	0.38	0.38	0.38	0.36

<u>Analysis:</u>

From question #6, the result of RMES and Explained_Variance_Score are 0.65 and 0.33, respectively, comparing with RMES and Explained_Variance_Score are 0.62~0.66 and 0.38~0.40 respectively in this question. Since the range of RMES and Explained_Variance_Score is not large, we can consider it as stable result.

From my point of view, I support that scores will increase after bagging comparing with question 6, however, the result does not show what I supposed to. I believe it is because of setting of criterion, since all of our scores based on the 'MSE' in this case, then it may generates same scores and does not improve too much for boosting methodology. Moreover, according to previous question, the dataset is close to normal distribution, if we consider it as well, then this results does not surprised us too much.

**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?*

Name	max_samples	random_state
Description	The number of samples to draw from X to train each base estimator	If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by `np.random`.

<u>Analysis:</u>

Max_samples decide percentage of sample from dataset to train. Basically higher percentage of sample, more accuracy of our result but more computation costs;

random_state makes sure that the sample from dataset be selected randomly, so that we get stable result which can apply to other dataset.

**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)?*

Analysis:

In this case, the out-of -bag error is the accuracy of examples xi using all the trees in the random forest ensemble for which it was omitted during training. Thus it kind of acts as a semi-testing instance, we can get a sense of how well our classifier can generalize using this metric. On the other hand, this approach also easily can be overfit since Random forest uses bootstrap aggregation of decision trees.
From my point of view, OOB can be used for determining hyper-parameters, but in order to estimate performance of model, I believe CV or data split also need to be applied.

**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?*

Analysis:

In this case, comparing with decision tree model from HW1, the Random Forests perform better than the decision tree algorithm. There are several advantages for random forest:

1) The random forest algorithm build on the further step of decision tree, so that it has same information gain algorithm from decision tree but more flexibility and reliable.
2) Unlike single decision tree, since the random forest randomly select trees which perform best, so that it can avoid overfitting problem in most of time.
3) Also random Forest using a random subset of features, This means if we have 30 features, random forests will only use a certain number of those features in each model.

**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?*

Analysis:

From my point of view, this is no "gold standard" approach. Depending on the dataset and goals, even one technique works perfectly on one dataset, but it does not guarantee will also work perfectly on another dataset. For example cross-section and time series dataset are totally different, also, weather dataset and human activity dataset as well.

Moreover, there are always more than one approach can be applied during the analysis. For instance, we can use both naïve bayes and random forest to do same job and compare the difference between them. On the other hand, the criterion of evaluation of each model may different from one to another, if our target is binary, logistic regression and naïve bayes may be applied; if type of target is continuous, then we can apply regular regression to do prediction.

On the top of that, we also need to evaluate more than one indicator. Accuracy and ROC can measure how precision of model with binary classification. RMSE, AIC/BIC and R^2 can help us to decide which regression model is best.

Based on my experience, there are several criterions when I choose approach for my model:

1) Handle missing value problem at first place;
2) Apply normalization to penalize noise/residual, to make dataset close to normal distribution as much as possible;
3) Avoid overfitting as much as possible;
4) Less features with high accuracy scores;
5) Test performance using more than one approach/algorithm