Machine Learning Homework 3

2020047029
JunYeong Ahn

Q1.
#1a)

	Acc	AUC
Gradient Boosting	0.76 (+/- 0.07)	0.82 (+/- 0.06)
Ada Boosting	0.76 (+/- 0.05)	0.83 (+/- 0.06)

*Though the code has been run only once, the results of Acc and AUC will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of Acc and AUC. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores
Thus 0.76 of Acc and 0.82 of AUC in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.76 of Acc and 0.83 of AUC in Ada Boosting are also averaged version of k (5) cases.
Also (+/- '0.07') and (+/- 0.06) on the back of Acc(0.76) and AUC(0.82) in Gradient Boosting mean they guarantee each 0.69~0.83(Acc) and 0.76~0.88(AUC) of error range with 95% confidence level – for example, '0.07' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of Accs and AUCs in Gradient Boosting for each fold case of CV would be 0.035 and 0.03.
In Ada Boosting, 0.71~0.81(Acc) and 0.77~0.89(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in Gradient Boosting for each fold case of CV would be 0.025 and 0.03.

#1b)
Default Estimator : DecisionTreeClassifier(max_depth=1)

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset. AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

And there are more concrete explanations on 'why it is significant';

1) Decision trees are non-linear. Boosting with linear models simply doesn't work well.
2) The weak learner needs to be consistently better than random guessing. I don't normally need to do any parameter tuning to a decision tree to get that behavior. Training an SVM really does need a parameter search. Since the data is re-weighted on each iteration, I likely need to do another parameter search on each iteration. So I am increasing the amount of work I have to do by a large margin.
3) Decision trees are reasonably fast to train. Since we are going to be building 100s or 1000s (not for sure) of them, that' s a good property. They are also fast to classify, which is again important when I need 100s or 1000s to run before I can output my decision.
4) By changing the depth I have a simple and easy control over the bias/variance trade off, knowing that boosting can reduce bias but also significantly reduces variance. Boosting is known to overfit, so the easy nob to tune is helpful in that regard.

Q2.
#2a)

	Acc	AUC	CV Runtime
Gradient Boosting (In Q1.)	0.76 (+/- 0.07)	0.82 (+/- 0.06)	0.6542456150054932
Ada Boosting (In Q1.)	0.76 (+/- 0.05)	0.83 (+/- 0.06)	1.026254415512085
MLP Classifier	0.71 (+/- 0.08)	0.73 (+/- 0.08)	2.5332133769989014

*Similar with the answer for Q1a, in all the trials of CV the same result comes out unlike the trials in the train/test split situation. In k- fold cross validation, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. The folds are divided as they are now even whenever CV is conducted.

About the scores
Thus the 0.71 of Acc and 0.73 of AUC in MLP Classifier are averaged ones of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset.
Also (+/- '0.08') and (+/- 0.08) on the back of Acc(0.71) and AUC(0.73) mean they guarantee each 0.63~0.79(Acc) and 0.65~-0.81(AUC) of error range with 95% confidence level – for example, '0.08' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. Thus, standard deviation of Accs and AUCs for each fold case of CV would be 0.04.

Gradient Boosting vs. MLP Classifier

Compared to the Gradient Boosting (GB), 1) Acc has dropped by 0.05 and AUC also did by 0.09 in MLP Classifier. Considering error ranges of scores, overlapped sections between Acc of GB and that of MLP exist (AUC too). However, 2) the minimum within error range of Acc (0.71-0.08) and that of AUC (0.73-0.08) in MLP are outside the expected Acc & AUC error range in GB and 3) the Acc&AUC maximum within error range of MLP (0.71+0.08 & 0.73+0.08) is each 0.04 & 0.07 smaller than that of GB (0.76+0.07 & 0.82+0.06) – even the maximum AUC of MLP(0.73+0.08) is smaller than the mean AUC of GB(0.82). Thus, since we usually interpret that the higher Acc and AUC lead to better model ,we can say that MLP didn't perform well with diabetes data compared to GB in terms of two scores even with much longer runtime (0.65 vs. 2.53).

Ada Boosting vs. MLP Classifier

Compared to the Ada Boosting (AB), 1) Acc has dropped by 0.05 and AUC also did by 0.1 in MLP Classifier. Considering error ranges of scores, overlapped sections between Acc of AB and that of MLP exist (AUC too). However, 2) the minimum within error range of Acc (0.71-0.08) and that of AUC (0.73-0.08) in MLP are outside the expected Acc & AUC error range in AB and 3) the Acc&AUC maximum within error range of MLP (0.71+0.08 & 0.73+0.08) is each 0.04 & 0.07 smaller than that of GB (0.76+0.07 & 0.82+0.06) – even the maximum AUC of MLP(0.73+0.08) is smaller than the mean AUC of AB(0.83). Thus, since we usually interpret that the higher Acc and AUC lead to better model ,we can say that MLP didn't perform well with diabetes data compared to AB in terms of two scores even with much longer runtime (1.02 vs. 2.53).

Based on given table, MLP performed worse with diabetes data (if with others, things could change) than boosting methods in terms of Acc and AUC.

#2b)

'adam' solver refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba. 'Adam' works pretty well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. Thus, 'adam' solver could be properly used when we have a lot of data to process.

Q3.
<Gradient Boosting>

	Acc	AUC	CV Runtime
Max_depth = 3	0.76 (+/- 0.07)	0.82 (+/- 0.06)	0.6472682952880859
Max_depth = 5	0.77 (+/- 0.06)	0.83 (+/- 0.07)	1.1190059185028076
Max_depth = 7	0.77 (+/- 0.07)	0.81 (+/- 0.08)	1.8560206890106201
Max_depth = 10	0.74 (+/- 0.06)	0.79 (+/- 0.08)	3.3001973628997803

About the score as the max_depth increases
Based on the result on the table Acc slightly increased by 0.01 and the standard deviation decreased by 0.01 when max_depth changed from 3 to 5. Also AUC improved in that the minimum of AUC is the same but the maximum of it is higher by 0.02 when max_depth = 5.
Then AUC dropped by 0.02 with the standard deviation increased by 0.01 while Acc stayed almost still when max_depth again changed from 5 to 7. When max_depth increased to 10, Acc and AUC noticeably diminished by 0.03 and 0.02 each – though the standard deviation of Accs also decreased by 0.01 but we are still able to say that the model gets degraded in terms of Acc since the minimum, the mean and the maximum within error range both gets lower. This means that scores (in the table above) increase then drop at some point, although the two scores may not always increase and decrease exactly the same. So the overall trend of given results seems to be saying that the scores (Acc and AUC) may be maximized neither in the start of results (max_depth = 3) nor in the end of results (max_depth = 10), but in somewhere middle point (here, max_depth = 5).
In addition, I could infer (this could be wrong since this inference comes from the four trials of the table, which might be few to get 100% answer) that the scores will be somewhat clear hat-shaped graph when the 'x'(here, 'max_depth') is more subdivided into 2, 3, 4, 5, ,,,.

About Run-times
It is expected that the CV runtime increases as max_depth increases because runtimes are associated with the number of calculations. If the maximum depth becomes larger, the time taken to form a single tree gets longer. And after forming multiple trees for Gradient Boosting, those delayed times are accumulated and reflected on CV runtimes. *This is similar to how it takes longer for a person to draw a tree when the tree is larger than when it is smaller – although this time our model draws them.*
And my expectation seems to be true;
0.64(CV Runtime when max_depth = 3) < 1.11(= 5) < 1.85(= 7) < 3.30(= 10).

Q4.
#4a)

<Before Feature Selection>	Acc	AUC	CV Runtime
Gradient Boosting	0.76 (+/- 0.07)	0.82 (+/- 0.06)	0.6542456150054932
Ada Boosting	0.76 (+/- 0.05)	0.83 (+/- 0.06)	1.026254415512085
MLP Classifier	0.71 (+/- 0.08)	0.73 (+/- 0.08)	2.5332133769989014

<**After** Feature Selection>	Acc	AUC	CV Runtime
Gradient Boosting	0.77 (+/- 0.05)	0.83 (+/- 0.07)	0.5176494121551514
Ada Boosting	0.77 (+/- 0.03)	0.84 (+/- 0.07)	0.8368189334869385
MLP Classifier	0.77 (+/- 0.07)	0.83 (+/- 0.08)	2.0245234966278076

*Though the code has been run only once, the results of Acc and AUC will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of Acc and AUC. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores
Thus 0.77 of Acc and 0.83 of AUC in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.77 of Acc & 0.84 of AUC in Ada Boosting and 0.77 of Acc & 0.83 of AUC in MLP are also averaged version of k (5) cases.
Also (+/- '0.05') and (+/- 0.07) on the back of Acc(0.77) and AUC(0.83) in Gradient Boosting mean they guarantee each 0.72~0.82(Acc) and 0.76~0.90(AUC) of error range with 95% confidence level – for example, '0.05' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of Accs and AUCs in Gradient Boosting for each fold case of CV would be 0.025 and 0.035.

In Ada Boosting, 0.74~0.80(Acc) and 0.77~0.91(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in Gradient Boosting for each fold case of CV would be 0.015 and 0.035.
In MLP, 0.70~0.84(Acc) and 0.75~0.91(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in MLP for each fold case of CV would be 0.035 and 0.04.

Comparison: Feature Selection vs. No Feature Selection

-Gradient Boosting (GB)
Acc has increased by 0.01 and 2 * its standard deviation has decreased by 0.02 after feature selection, which means GB gets improved in terms of accuracy. Likewise, AUC and 2 * its standard deviation increased by 0.01 after feature selection, which means GB gets better in terms of AUC.
Thus, considering two performance scores, GB gets better after Wrapper-Based feature selection.

-Ada Boosting (AB)
Acc has increased by 0.01 and 2 * its standard deviation has decreased by 0.02 after feature selection, which means AB gets improved in terms of accuracy. Likewise, AUC and 2 * its standard deviation increased by 0.01 after feature selection, which means AB gets better in terms of AUC.
Thus, considering two performance scores, AB gets better after Wrapper-Based feature selection.

-MLP Classifier
Acc has increased by 0.06 and 2 * its standard deviation has decreased by 0.01 after feature selection, which means MLP gets improved in terms of accuracy. Likewise, AUC increased by 0.1 while 2 * its standard deviation being the same after feature selection, which means MLP gets better in terms of AUC.
Thus, considering two performance scores, MLP gets better after Wrapper-Based feature selection.

About Runtimes
Based on those recorded CV Runtime, <u>CV Runtime for each model has decreased a bit</u> (by around 0.14, 0.19 and 0.51 respectively) <u>after feature selection</u>, although it is hard to say that ‘it is absolute result’ since my computer’s computational speed slightly changes in real time. <u>This fact</u> might stem from the fact: that the decrease of the number of features used in each model simplified (maybe related with parsimonious model?) the calculations within each model.

#4b)

Among total 8 features; 'Times Pregnant', 'Blood Glucose', 'Blood Pressure', 'Skin Fold Thickness', '2-Hour Insulin', 'BMI', 'Family History', 'Age', only three features ('Blood Glucose', 'BMI', 'Age') have survived and the other five features ('Times Pregnant', 'Blood Pressure', 'Family History', 'Skin Fold Thickness', '2-Hour Insulin') have been abandoned after Wrapper-Based feature selection.

Unlike this time, features that survived after Wrapper-Based feature selection using Random Forest classifier in *HW2* were 'Blood Glucose', 'BMI', 'Family History', 'Age' – four features in total with 'Family History' feature added.

In short, <u>two feature selection results in HW2 and HW3 differ</u> since HW2 is using Random Forest but this homework is using Gradient Boosting, although they are based on the same selection method: Wrapper-Based Feature Selection.

Q5.

	Acc	AUC	CV Runtime
MLP Classifier(lbfgs)	0.77 (+/- 0.07)	0.83 (+/- 0.08)	2.0245234966278076
MLP Classifier(adam)	0.72 (+/- 0.07)	0.80 (+/- 0.08)	4.0731024742126465

*Though the code has been run only once, the results of Acc and AUC will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of Acc and AUC. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores

0.77 of Acc and 0.83 of AUC in MLP with lbfgs are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.72 of Acc and 0.80 of AUC in MLP with adam are also averaged version of k (5) cases.

Also (+/- '0.07') and (+/- 0.08) on the back of Acc(0.77) and AUC(0.83) in MLP with lbfgs mean they guarantee each 0.70~0.84(Acc) and 0.75~0.91(AUC) of error range with 95% confidence level – for example, '0.07' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of Accs and AUCs in MLP with lbfgs for each fold case of CV would be 0.035 and 0.04.

In MLP with adam, 0.65~0.79(Acc) and 0.72~0.88(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of both scores in MLP with adam for each fold case of CV would be 0.035 and 0.04.

About the change of scores
After changing the solver from 'lbfgs' to 'adam', Acc decreased by 0.05 while its standard deviation didn't change. Also, AUC decreased by 0.03 while its standard deviation stayed still. Thus, since we usually interpret that the lower Acc and AUC lead to worse model, we can say that using 'adam' doesn't work well with Diabetes data in terms of Acc and AUC compared to 'lbfgs' solver.

About run-times
After changing the solver from 'lbfgs' to 'adam', CV Runtime almost doubled (quite clear gap though CV Runtime is affected by the current condition of our computer). This answer seems to be related to Q2b - for small datasets, 'lbfgs' can converge faster (=> decrease in CV Runtime) and perform better than 'adam'.

Q6.
#6a)

	RMSE	Expl Var
Gradient Boosting	0.64 (+/- 0.01)	0.34 (+/- 0.12)
Ada Boosting	0.66 (+/- 0.03)	0.31 (+/- 0.16)

*Though the code has been run only once, the results of RMSE and Expl Var will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of RMSE and Expl Var. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores
Thus 0.64 of RMSE and 0.34 of Expl Var in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.66 of RMSE and 0.31 of Expl Var in Ada Boosting are also averaged version of k (5) cases.
Also (+/- '0.01') and (+/- 0.12) on the back of RMSE(0.64) and Expl Var(0.34) in Gradient Boosting mean they guarantee each 0.63~0.65(RMSE) and 0.22~0.46(Expl Var) of error range with 95% confidence level – for example, '0.01' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of RMSEs and Expl Vars in Gradient Boosting for each fold case of CV would be 0.005 and 0.06.

In Ada Boosting, 0.63~0.69(RMSE) and 0.15~0.47(Expl Var) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in Gradient Boosting for each fold case of CV would be 0.015 and 0.08.

#6b)

My think of the purpose of the learning rate parameter
*Gradient Boosting is a ML technique for classification and regression problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

In Gradient Boosting, the learning rate, who shrinks the contribution of each new base model - typically a shallow tree - that is added in the series, corresponds to how quickly the error is corrected from each tree to the next and is a simple multiplier / 0 < learning rate ≤ 1.
For example, if the current prediction for a particular example is 0.2 and the next tree predicts that it should actually be 0.8, the correction would be + 0.6. At a learning rate of 1, the updated prediction would be the full 0.2 + 1(0.6) = 0.8, while a learning rate of 0.1 would update the prediction to be 0.2 + 0.1(0.6) = 0.26. By tuning learning rate, I could get fast speed or better optimization as my likes.

Q7.
#7a)

	RMSE	Expl Var	CV Runtime
Gradient Boosting (In Q6.)	0.64 (+/- 0.01)	0.34 (+/- 0.12)	1.3055064678192139
Ada Boosting (In Q6.)	0.66 (+/- 0.03)	0.31 (+/- 0.16)	1.5369338989257812
MLP Regressor	0.66 (+/- 0.05)	0.29 (+/- 0.13)	6.3809239864349365

*Similar with the answer for Q6a, in all the trials of CV the same result comes out unlike the trials in the train/test split situation. In k- fold cross validation, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. The folds are divided as they are now even whenever CV is conducted.

About the scores
Thus the 0.66 of RMSE and 0.29 of Expl Var in MLP Regressor are averaged ones of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset.
Also (+/- '0.05') and (+/- 0.13) on the back of RMSE(0.66) and Expl Var(0.29) mean they guarantee each 0.61~0.71(RMSE) and 0.16~-0.42(Expl Var) of error range with 95% confidence level – for example, '0.05' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. Thus, standard deviation of Accs and AUCs for each fold case of CV would be 0.025 and 0.065 respectively.

Gradient Boosting vs. MLP Regressor
Compared to the Gradient Boosting (GB), 1) Acc has increased by 0.02 and Expl Var dropped by 0.05 in MLP Regressor. Also the 2*standard deviation of RMSEs and Expl Vars are larger by 0.04 and 0.01 in MLP. Considering error ranges of scores, overlapped sections between RMSE of GB and that of MLP exist (Expl Var too). However, 2) the mean of RMSE in MLP (0.66) is higher than the maximum within error range of RMSE in GB (0.64+0.01) and 3) the minimum, mean and maximum of Expl Var are lower in MLP than those in GB. Thus, since we usually interpret that the lower RMSE and the higher Expl Var lead to better model, we can say that MLP didn't perform well with Wine data compared to GB in terms of two scores even with much longer runtime (6.38 vs. 1.30).

Ada Boosting vs. MLP Regressor
Compared to the Ada Boosting (AB), 1) RMSE stayed still – 2 * standard deviation changed from 0.01 to 0.05 but still it is hard to say about RMSE since the averaged score is exactly the same (0.66) - and Expl Var dropped by 0.02 in MLP Regressor. Considering error ranges of scores, overlapped sections between RMSE of AB and that of MLP exist (Expl Var too). However, 2) the mean of Expl Var in MLP (0.29) is still lower than that in AB (0.31) (although the error range of Expl Var of MLP didn't deviate that of AB). Thus, since we usually interpret that the lower RMSE and the higher Expl Var lead to better model, it is hard to say that MLP perform well with Wine data compared to AB in terms of two scores even with much longer runtime (6.38 vs. 1.53) – strictly speaking, MLP did worse than AB.

 Based on given table, MLP performed worse with Wine data (if with others, things could change) than boosting methods in terms of RMSE and Expl Var.

#7b)

I should set the 'hidden_layer_sizes' parameter equal to the tuple '(10, 10,)'.

'(100,)', which is the default value of 'hidden_layer_sizes' parameter, means that default architecture will have one input layer, one hidden layer with 100 units and one output layer. So if I want to create two hidden layers who have 10 units each, '(10, 10,)' will be suit here.

Q8.
#8a)

	Acc	AUC	CV Runtime
Gradient Boosting	0.73 (+/- 0.05)	0.81 (+/- 0.06)	1.8304128646850586
Ada Boosting	0.74 (+/- 0.05)	0.82 (+/- 0.06)	1.7519958019256592
MLP Classifier	0.72 (+/- 0.07)	0.80 (+/- 0.07)	13.191008567810059

*Though the code has been run only once, the results of Acc and AUC will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of Acc and AUC. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores
Thus 0.73 of Acc and 0.81 of AUC in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.74 of Acc & 0.82 of AUC in Ada Boosting and 0.72 of Acc & 0.80 of AUC in MLP are also averaged version of k (5) cases.
Also (+/- '0.05') and (+/- 0.06) on the back of Acc(0.73) and AUC(0.81) in Gradient Boosting mean they guarantee each 0.68~0.78(Acc) and 0.75~0.87(AUC) of error range with 95% confidence level – for example, '0.05' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of Accs and AUCs in Gradient Boosting for each fold case of CV would be 0.025 and 0.03.
In Ada Boosting, 0.69~0.79(Acc) and 0.76~0.88(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in Gradient Boosting for each fold case of CV would be 0.025 and 0.03.
In MLP, 0.65~0.79(Acc) and 0.73~0.87(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in MLP for each fold case of CV would be both 0.035.

Comparison to the regression scores?
We are no longer able to compare these models (or scores) to those in regression. Through discretization, the target variable became binary class which means the problem changed from regression into classification, although original target variable in Wine data was numeric/continuous values (=regression problem). Classification and Regression are fundamentally different and use different totally performance scores (here, Acc&AUC for classification and RMSE&Expl Var for regression). Therefore, to compare these scores to previous one which corresponds to regression version is impossible.

#8b)

What are we actually predicting here?

```
-----------------------------
Bin 0 : 3.0 5.0 744
Bin 1 : 6.0 8.0 855
-----------------------------
```

<The info printed out>

At first, I'll rephrase the result above.

```
-------------------------------------------------------------------------------------------------------------------------
Bin 0(class name) : 3.0(minimum value) 5.0(maximum value) 744(the # of samples in Bin 0)
Bin 1 (class name) : 6.0(minimum value) 8.0(maximum value) 855(the # of samples in Bin 1)
-------------------------------------------------------------------------------------------------------------------------
```

What we tried to do with Wine Quality data was to predict wine quality, whose target variable ranges from 3.0 to 8.0 and is numeric/continuous value, based on some features – regression problem. After discretization using KBinsDiscretizer function, however, the quality values of wine are packed into two bins; Bin 0 and Bin 1. *<The info printed out>* shows us that target values that range from 3.0(min) to 5.0(max) are bound into Bin 0 and it contains 744 observations. Likewise, target values that range from 6.0(min) to 8.0(max) are bound into Bin 0 and it has 855 observations. So we are predicting 'which bin' the novel observation will go into (=whether the wine quality of the new observation will range 3.0~5.0 or 6.0~8.0), not what 'numeric value' of wine quality the novel observation will exactly have.

How would you explain what you did to your boss or customer?
So I'm gonna say this to my boss or customer in easy words;

"Predicting the exact value of wine quality may be tough. Do you also think it is better to predict simply whether the wine will be good(tasty?) or not, not that the wine quality will be 5.39(just an example)? If you want that easy-interpretation and good-prediction mode, FOR YOU, I can make number guessing game into up-down game (I believe you know what this game is) through so-called discretization, though this time we are not randomly guessing but based on certain features."

Q9.

	Acc	AUC	CV Runtime
Gradient Boosting	0.73 (+/- 0.05)	0.81 (+/- 0.06)	2.0885372161865234
Ada Boosting	0.74 (+/- 0.05)	0.82 (+/- 0.06)	1.5578997135162354
MLP Classifier	0.72 (+/- 0.07)	0.80 (+/- 0.07)	14.262721538543701

*Though the code has been run only once, the results of Acc and AUC will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of Acc and AUC. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores

Thus 0.73 of Acc and 0.81 of AUC in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.74 of Acc & 0.82 of AUC in Ada Boosting and 0.72 of Acc & 0.80 of AUC in MLP are also averaged version of k (5) cases.

Also (+/- '0.05') and (+/- 0.06) on the back of Acc(0.73) and AUC(0.81) in Gradient Boosting mean they guarantee each 0.68~0.78(Acc) and 0.75~0.87(AUC) of error range with 95% confidence level – for example, '0.05' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of Accs and AUCs in Gradient Boosting for each fold case of CV would be 0.025 and 0.03.

In Ada Boosting, 0.69~0.79(Acc) and 0.76~0.88(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in Gradient Boosting for each fold case of CV would be 0.025 and 0.03.

In MLP, 0.65~0.79(Acc) and 0.73~0.87(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in MLP for each fold case of CV would be both 0.035.

Comparison to Q8a?

*Normalization is used to alter a feature so it is more similar to a normal (Gaussian) distribution and when features have different scales.

The result in Q9 is exactly the same to the result in Q8a – there is nothing to compare. Seeing the results in Q9 and Q8a, normalization in Q9 might help features to have the same scale with others and become more similar to a normal (Gaussian) distribution but didn't directly change the scores (maybe it's a special case that normalization doesn't work?). Thus, I could infer that the target variable didn't have a weird distribution and had few outliers (or completely not).

Q10.
#10a)

<binned>

	Acc	AUC	CV Runtime
Gradient Boosting	0.73 (+/- 0.05)	0.81 (+/- 0.05)	1.1708855628967285
Ada Boosting	0.73 (+/- 0.06)	0.82 (+/- 0.06)	1.226710319519043
MLP Classifier	0.74 (+/- 0.05)	0.81 (+/- 0.07)	8.890260696411133

<unbinned>

	RMSE	Expl Var	CV Runtime
Gradient Boosting	0.66 (+/- 0.02)	0.29 (+/- 0.14)	0.6452124118804932
Ada Boosting	0.67 (+/- 0.04)	0.30 (+/- 0.13)	0.5885839462280273
MLP Regressor	0.65 (+/- 0.03)	0.33 (+/- 0.12)	4.583731412887573

*Though the code has been run only once, the results of Acc and AUC will be the same whenever I run the code because it is the result of 5-fold CV. Unlike k-fold CV, in train/test split the train set is randomly and newly sampled in each trial and the test set also changes as a result, which results in different outcomes of Acc and AUC. In k- fold cross validation (Here, k is 5 but I'll use k to generalize the CV steps), however, the data is divided into k parts (folds). And k-1 of the parts are used for training, and 1 is used for testing. This procedure is repeated k times rotating the test set then those results are averaged. These folds are divided as they are now even whenever CV is conducted again, which means the outcome will not change whenever I try it.

About the scores

<binned>

Thus 0.73 of Acc and 0.81 of AUC in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.73 of Acc & 0.82 of AUC in Ada Boosting and 0.74 of Acc & 0.81 of AUC in MLP are also averaged version of k (5) cases.

Also (+/- '0.05') and (+/- 0.05) on the back of Acc(0.73) and AUC(0.81) in Gradient Boosting mean they guarantee each 0.68~0.78(Acc) and 0.76~0.86(AUC) of error range with 95% confidence level – for example, '0.05' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of Accs and AUCs in Gradient Boosting for each fold case of CV would be both 0.025.

In Ada Boosting, 0.67~0.79(Acc) and 0.76~0.88(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in Gradient Boosting for each fold case of CV would be both 0.03.

In MLP, 0.69~0.79(Acc) and 0.74~0.88(AUC) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in MLP for each fold case of CV would be 0.025 and 0.035.

<unbinned>

0.66 of RMSE and 0.29 of Expl Var in Gradient Boosting are averaged version of k (5) cases, which means they give an insight on how the model will generalize to an independent dataset. Likewise, 0.67 of RMSE & 0.30 of Expl Var in Ada Boosting and 0.65 of RMSE & 0.33 of Expl Var in MLP are also averaged version of k (5) cases.

Also (+/- '0.02') and (+/- 0.14) on the back of RMSE(0.66) and Expl Var(0.29) in Gradient Boosting mean they guarantee each 0.64~0.68(RMSE) and 0.15~0.43(Expl Var) of error range with 95% confidence level – for example, '0.02' is 2 (originally 1.96 that stands for 95% confidence level) * standard deviation. As a result, standard deviation of RMSEs and Expl Vars in Gradient Boosting for each fold case of CV would be 0.01 and 0.07.

In Ada Boosting, 0.63~0.71(RMSE) and 0.17~0.43(Expl Var) of error range with 95% confidence level are guaranteed, which means standard deviation of both scores in Gradient Boosting for each fold case of CV would be 0.02 and 0.065.

In MLP, 0.62~0.68(RMSE) and 0.21~0.45(Expl Var) of error range with 95% confidence level are guaranteed, which means standard deviation of Both scores in MLP for each fold case of CV would be 0.015 and 0.06.

Comparison: Feature Selection vs. No Feature Selection

<binned>

-In Q8a (No Feature Selection).

	Acc	AUC	CV Runtime
Gradient Boosting	0.73 (+/- 0.05)	0.81 (+/- 0.06)	1.8304128646850586
Ada Boosting	0.74 (+/- 0.05)	0.82 (+/- 0.06)	1.7519958019256592
MLP Classifier	0.72 (+/- 0.07)	0.80 (+/- 0.07)	13.191008567810059

-In Q10a (Feature Selected)

	Acc	AUC	CV Runtime
Gradient Boosting	0.73 (+/- 0.05)	0.81 (+/- 0.05)	1.1708855628967285
Ada Boosting	0.73 (+/- 0.06)	0.82 (+/- 0.06)	1.226710319519043
MLP Classifier	0.74 (+/- 0.05)	0.81 (+/- 0.07)	8.890260696411133

This is the case when target variables are binned. In Gradient Boosting, no significant change has been observed in scores (although the standard deviation of AUC slightly decreased). In Ada Boosting, Acc and 2*its standard deviation decreased by 0.01 and increased by the same figure with AUC being the same – still no 'significant' change in scores. In MLP Classifier, Acc has increased by 0.02 and 2*its standard deviation has decreased by the same figure with even AUC getting increased by 0.01, which means MLP improved in terms of Acc&AUC after feature selection.

We also can say that Gradient Boosting and Ada Boosting get improved after feature selection as well. There was some loss of scores such as decrease of Acc in Ada Boosting. However, we got much more *parsimonious* models which are always welcomed compared to others, although there were very subtle losses.

Also, CV Runtimes in three models have decreased by around 0.56, 0.53 and 4.3 – this roots for my previous argument although it is hard to say that 'it is absolute result' since my computer's computational speed slightly changes in real time.

<unbinned>

-In Q7a. (No Feature Selection)

	RMSE	Expl Var	CV Runtime
Gradient Boosting	0.64 (+/- 0.01)	0.34 (+/- 0.12)	1.3055064678192139
Ada Boosting	0.66 (+/- 0.03)	0.31 (+/- 0.16)	1.5369338989257812
MLP Regressor	0.66 (+/- 0.05)	0.29 (+/- 0.13)	6.3809239864349365

-In Q10a. (Feature Selected)

	RMSE	Expl Var	CV Runtime
Gradient Boosting	0.66 (+/- 0.02)	0.29 (+/- 0.14)	0.6452124118804932
Ada Boosting	0.67 (+/- 0.04)	0.30 (+/- 0.13)	0.5885839462280273
MLP Regressor	0.65 (+/- 0.03)	0.33 (+/- 0.12)	4.583731412887573

This is the case when target variables are unbinned. In Gradient Boosting, RMSE and 2*its standard deviation have increased by 0.02 and 0.01. In addition, Expl Var and 2*its standard deviation have decreased by 0.05 and increased by 0.02 – the scores got worse after feature selection. In Ada Boosting, RMSE and 2*its standard deviation both increased by 0.01. Plus, Expl Var increased by 0.01 as well and 2*its standard deviation decreased by 0.03. In MLP Classifier, RMSE has decreased by 0.01 and 2*its standard deviation has decreased by 0.02 with even AUC increased by 0.04, which means MLP improved in terms of RMSE and Expl Var after feature selection.

We also can say that Gradient Boosting and Ada Boosting get improved after feature selection as well. There was some loss of scores such as increase of RMSE in Gradient Boosting. However, we got much more *parsimonious* models which are always welcomed compared to others, although there were very subtle losses.

Also, CV Runtimes in three models have decreased by around 0.66, 0.95 and 1.8 – this roots for my previous argument although it is hard to say that ‘it is absolute result’ since my computer’s computational speed slightly changes in real time.

#10b)
<binned>
Among total 11 features; 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', only four features ('volatile acidity', 'total sulfur dioxide', 'sulphates', 'alcohol') have survived and the other seven features ('fixed acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'density', 'pH') have been abandoned after Wrapper-Based feature selection.

<unbinned>
Among total 11 features; 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', only three features ('volatile acidity', 'sulphates', 'alcohol') have survived and the other eight features ('fixed acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH') have been abandoned after Wrapper-Based feature selection.

Two feature selection results of <binned> and <unbinned> differ in that 'total sulfur dioxide' feature is selected when target variables are <unbinned> whereas it is not selected when target variables are <binned>.

Q11.
*Again, it is hard to say that 'it is absolute result' about runtimes since my computer's computational speed slightly changes in real time.
*I'll not directly compare Decision Tree to models in this HW since it worked too poor to
to the extent that we don't have to count it in this battle, but RF vs. models in this HW is sufficient.

In Diabetes Dataset
In terms of accuracy and AUC scores, Random Forest(RF) outperformed Decision Tree(DT). If I compare the results from 5-fold cross validation (since we didn't conduct 3, 8 and 10 fold CV against RF), Acc and AUC of DT were 0.71(+/-0.08) and 0.69(+/-0.07) but those of RF are 0.77 (+/- 0.08) and 0.83 (+/- 0.07) (with 100 trees from Q2 in HW2.).
And lets RF vs. boosting models and MLP.

RF) Acc : 0.77 (+/- 0.08) / AUC : 0.83 (+/- 0.07) / Runtime : 1.53
GB) Acc : 0.76 (+/- 0.07) / AUC : 0.82 (+/- 0.06) Runtime : 0.65
AB) Acc : 0.76 (+/- 0.05) / AUC : 0.83 (+/- 0.06) Runtime : 1.02
NN) Acc : 0.71 (+/- 0.08) / AUC : 0.73 (+/- 0.08) / Runtime : 2.53

Actually it is hard to tell the difference between performances (in terms of scores) of models, except for MLP (it performed very poor) – still scores were the highest in RF with subtle gap. However, GB and AB have lower runtime than RF. So more specified rank between those three models may depend on the trade-off between scores and runtime – in my personal opinion, they tied unless there are further analysis.

In Wine Dataset

Likewise, RF outperformed DT comparing the results from 5-fold CV ;
DT) RMSE: 0.90 (+/- 0.10) / Expl Var: -0.31 (+/- 0.17)
RF) RMSE:: 0.65 (+/- 0.02) / Expl Var: 0.33 (+/- 0.11) / Scores in RF greatly improved, deviating far from the error range of DT's.
And lets RF vs. boosting models and MLP.

RF) RMSE : 0.65 (+/- 0.02) / Expl Var: 0.33 (+/- 0.11) / Runtime : 2.21
GB) RMSE : 0.64 (+/- 0.01) / Expl Var: 0.34 (+/- 0.12) / Runtime : 1.30
AB) RMSE : 0.66 (+/- 0.03) / Expl Var: 0.31 (+/- 0.16) / Runtime : 1.53
NN) RMSE : 0.66 (+/- 0.05) / Expl Var: 0.29 (+/- 0.13) / Runtime : 6.38

Again, NN seems to be poor than others with the highest RMSE & Runtime and the lowest Expl Var.
And GB is great in that it has the lowest RMSE & Runtime and the highest Expl Var.
But it is hard to compare RF to AB since RF has lower RMSE and higher Expl Var but has higher Runtime. Thus, the trade-off problem between scores and runtime arises again – but I can somewhat rank them <u>thoroughly based on two scores and Runtime</u>: NN < RF? AB? < GB. As a consequence, the gap between RF and models in HW3 is subtle though – almost draw.

WHY?

Random forest leverages the power of multiple decision trees. It does *not* rely on the feature importance given by a single decision tree. Therefore, the random forest can generalize over the data in a better way. This randomized feature selection makes random forest much more accurate than a decision tree.
However, Gradient Boosting and Ada Boosting are the same ensemble models with Random Forest. Though there are difference between them in that the way each of them learn is different – for example, unlike RF that produces multiple weak learners at once and averages them, the boosting is to train weak learners sequentially, each trying to correct its predecessor - , but no absolute superiority between them They all have their own strengths and weaknesses, and this time they (RF & boosting models) just performed similar in both dataset.

In a bit Easier (But Wordy) Sentences :

"Through exploring many diverse cases (each case corresponds to DT) and averaging them, Random Forest becomes powerful and robust to variation of data compared to a single Decision Tree which is very sensitive to the data it's been trained on. It resembles collective intelligence – one idiot cannot but multiple (numerous) idiots can make a clever decision. And MLP, which is a Neural Network, generally needs a lot of datasets but we have a few, thereby it performed poorly – also MLP have many more possible parameters to tune more than features."

"And one possible reason for why those three ensemble models performed similarly could be this; Soccer shoes are good when playing soccer and basketball shoes are so when playing basketball, baseball shoes when playing baseball. But what if we try to wear them when playing tennis? Or when eating a pizza (an example of 'normal situation')? Which one will be the best? Is it weird to say that any shoes don't matter? Some can be good and some might be bad but those kind of predictions are hard to be based on clear causes."

Q12.

The features we selected are 'Blood Glucose', 'BMI', and 'age'. We can create online health platform for local residences. In this platform, the user reports his current body condition (height and weight for BMI calculation) and age - these are easy to know and report. And recent blood glucose could be reported as well, regardless of whether it is automatically reported from local health institutions or manually reported after large-scale inspection (once) against the whole residences. Since the information including recent blood glucose for three features have been reported, our platform would be able to predict diabetes using any suitable models - these could be determined after further hyper-parameter tuning. As we've learned from the paper on Health Informatics, CNN(Convolutional Neural Network) could be used in order to recognize food accurately within the proposed smartphone-based system, which estimates the calories contained in pictures of food taken by the user. And I think this model can also recognize the sugar components of food. By using this CNN, the user takes a picture of their favorite food (that he frequently eat) then the model sends the result to platform after analyzing calories (for BMI) and the sugar components (for Blood Glucose). Based on these newly reported data for the user's eating behavior, the platform re-predicts his diabetes status then lets him know whether he is likely to develop diabetes within N months if this eating behavior or diet is maintained. In addition, the platform could suggest some probabilities related to diabetes if the percentile concept is introduced then some numeric columns are newly made.

Though this model needs one large-scale inspection against the local, the platform becomes able to keep track of users' diabetes status in real-time even with a tiny participation of users.