Advance Machine Learning – Homework 3

Name: - Umme Athiya

Pima Diabetes: -

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

Ans: -

Algorithm	Accuracy	AUC	CV Runtime
Gradient Boosting Classifier	0.76 (+/- 0.06)	0.82 (+/- 0.06)	0.9507982730865479
AdaBoost Classifier	0.76 (+/- 0.07)	0.83 (+/- 0.08)	1.2282774448394775

- Upon running the code once, and having both the model results, their accuracy is the same which is approx.. 0.76, which means they perform well in predicting the correct class.
- The AUC score for Ada Boost (0.83) is a little higher than Gradient Boosting (0.82), thus from these scores, Ada Boost is better.
- And lastly, the CV Runtime of the Ada Boost classifier is longer than the other, which means the computation is more expensive.

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?

Ans: -

- So, the default estimator for a parameter that is set to None, is max_depth=1 which means that the decision tree would be a very shallow tree with only one split.
- Also, AdaBoost works by correcting the errors of weak parameters or learners in a sequential mode, thus making it more robust.

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?

Algorithm	Accuracy	AUC	CV Runtime
Gradient Boosting Classifier	0.76 (+/- 0.06)	0.82 (+/- 0.06)	1.0981919765472412
AdaBoost Classifier	0.76 (+/- 0.07)	0.83 (+/- 0.08)	1.344412088394165
MLPClassifier	0.70 (+/- 0.07)	0.73 (+/- 0.09)	5.481795072555542

- From the above results, it looks like MLPClassifier is low for both its Accuracy and AUC along with the CV runtime when compared to the other models.
- This suggests that neural networks generally need more computation time, to backpropagation updates.

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

Ans: -

- Generally, 'adam' solvers in MLP Classifier are for adaptive optimization algorithm that usually aggregates the momentum and adaptive learning rates.
- They come into the picture when working for larger datasets, when dataset is noisy, suitable for training on high dimensional data, and stochastic gradient descent (SGD) optimization.

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

Ans: -

Algorithm	max_depth	Accuracy	AUC	CV Runtime
Gradient Boosting Classifier	3	0.76 (+/- 0.06)	0.82 (+/- 0.06)	1.0981919765472412
Gradient Boosting Classifier	5	0.77 (+/- 0.05)	0.83 (+/- 0.07)	1.5650091171264648
Gradient Boosting Classifier	7	0.76 (+/- 0.05)	0.81 (+/- 0.07)	2.435595750808716
Gradient Boosting Classifier	10	0.73 (+/- 0.07)	0.79 (+/- 0.08)	4.524122953414917

❖ **<u>Accuracy and AUC Trends: -</u>**
  - At the start, when max_depth is increasing from 3 to 5, the accuracy and AUC slightly improves.
  - When max_depth = 7, accuracy remain stable but AUC slightly drops.
  - At max_depth = 10, both accuracy and AUC decreases but to over fitting.

❖ **<u>Runtime Increases: -</u>**

- We see that when max_depth increases, the CV runtime increases from 1.09 seconds to 4.52 seconds at depth=10.
- This is solely due to the deeper trees as they have more nodes and splits thus increasing computational cost.

Question #4a: Run the code once, and 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?

Ans: -

Algorithm	**Feature selection**	**Accuracy**	**AUC**	**CV Runtime**
Gradient Boosting Classifier	0	0.76 (+/- 0.06)	0.82 (+/- 0.06)	1.0981919765472412
AdaBoost Classifier	0	0.76 (+/- 0.07)	0.83 (+/- 0.08)	1.344412088394165
MLPClassifier	0	0.70 (+/- 0.07)	0.73 (+/- 0.09)	5.481795072555542
Gradient Boosting Classifier	1	0.77 (+/- 0.05)	0.83 (+/- 0.07)	0.6854889392852783
AdaBoost Classifier	1	0.76 (+/- 0.06)	0.83 (+/- 0.08)	0.9021971225738525
MLPClassifier	1	0.75 (+/- 0.13)	0.81 (+/- 0.12)	2.466346025466919

❖ **Gradient Boosting Classifier: -**

- With feature selection, the GBC displays a marginal improvement in accuracy and AUC (accuracy rose from 0.76 to 0.77, AUC climbed from 0.82 to 0.83). Furthermore, the CV duration is greatly decreased, suggesting that feature selection enhanced model efficiency by lowering the feature count.

❖ **AdaBoost Classifier: -**

- With or without feature selection, AdaBoost's accuracy and AUC stay the same, but the CV runtime drops, demonstrating that feature selection enhances the model's effectiveness without compromising performance.

❖ **MLP Classifier: -**

- Feature selection significantly improves the accuracy and AUC of the MLP Classifier (accuracy went from 0.70 to 0.75, AUC went from 0.73 to 0.81). Additionally, the runtime is greatly decreased, suggesting that feature selection lowers computing costs while simultaneously enhancing 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?

Ans: -

Feature selection type = 2	
Selected	**Removed**
Blood Glucose	Time Pregnant
BMI	Blood Pressure
Age	Skin Fold Thickness
	2 – Hour Insulin
	Family History

- When compared with Random Forest from HW2, the selected feature – family history is removed in this homework. So, the features selected in this code are Blood Glucose, BMI, and Age.

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

Ans: -

Algorithm	Solver	Accuracy	AUC	CV Runtime
MLP Classifier	**lbgfs**	0.76 (+/- 0.06)	0.82 (+/- 0.06)	1.0981919765472412
MLP Classifier	**adam**	0.72 (+/- 0.07)	0.80 (+/- 0.08)	3.363502025604248

❖ **<u>Accuracy</u>**

- Compared to Adam's accuracy of 0.72 (+/- 0.07), the accuracy of L-BFGS is higher at 0.76 (+/- 0.06).
- Adam: Compared to L-BFGS, the accuracy is lower, indicating that L-BFGS may be more appropriate for this specific situation or that it has better convergence with less iterations.

- ❖ **AUC:**
    - Compared to Adam's AUC of 0.80 (+/- 0.08), L-BFGS's AUC is 0.82 (+/- 0.06), which is somewhat higher. According to the AUC score, this demonstrates that L-BFGS also does somewhat better at class distinction.
- ❖ **Runtime for CV:**
    - L-BFGS: At 1.10 seconds, the runtime is much shorter.
    - Adam: In this instance, Adam needs more iterations and time for convergence than L-BFGS, as evidenced by the significantly longer runtime (3.36 seconds).

Accuracy and AUC are both improved by L-BFGS, which also has a higher runtime efficiency.

Wine Quality Dataset: -

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

Ans: -

Algorithm	RMSE	Expl Var	CV Runtime
Gradient Boosting Regressor	0.64 (+/- 0.01)	0.34 (+/- 0.12)	1.5174193382263184
AdaBoost Regressor	0.66 (+/- 0.03)	0.31 (+/- 0.16)	1.9401097297668457

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)?

Ans: -

- A gradient boosting regression's learning rate regulates how much each tree contributes to the final prediction. To put it simply, it establishes the extent to which the model should adapt to the mistakes produced by the existing tree.
- **Gradient Boosting** is an ensemble technique that creates a powerful predictive model by combining several weak models (decision trees). Every new tree fixes the mistakes of the ones that came before it.
- **Learning Rate:** The learning rate reduces the contribution of a new tree to the model. It takes more trees to accurately represent the data when the learning rate is lower because each tree's impact is reduced. A higher learning rate, on the other hand, indicates that each tree has a greater influence, which can expedite training but, if set too high, may result in overfitting.

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?

Ans: -

Algorithm	**RMSE**	**Variance**	**CV Runtime**
Gradient Boosting Regressor	0.64 (+/- 0.01)	0.34 (+/- 0.12)	1.907599687576294
AdaBoost Regressor	0.66 (+/- 0.03)	0.31 (+/- 0.16)	2.436420440673828
MLP Regressor	0.66 (+/- 0.05)	0.31 (+/- 0.12)	9.116678953170776

- Because it has the lowest RMSE and the highest explained variance, the Gradient Boosting Regressor performs better than the AdaBoost and MLP regressors.
- With RMSE values of 0.66, the AdaBoost and MLP regressions perform quite similarly; however, the MLP regression exhibits greater performance variability (wider standard deviation in RMSE).
- With a runtime of 1.91 seconds, the Gradient Boosting Regressor model is the fastest.
- At 2.44 seconds, the AdaBoost Regressor is a little slower.
- With a runtime of 9.12 seconds, MLP Regressor takes a lot longer, which may indicate that it is more computationally demanding.

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?

Ans: -

```
MLPRegressor(hidden_layer_sizes=(10, 10), activation='logistic', solver='lbfgs',
alpha=0.0001, max_iter=1000, random_state=rand_st)
```

- In the MLP Regressor from Scikit-learn, you would set the hidden_layer_sizes argument to a tuple with two values (10, 10) to generate a neural network with two hidden layers, each with 10 units.
- (10, 10) designates two buried layers with ten neurons per.
- Activation='logistic' sets the logistic (sigmoid) activation function.
- solver='lbfgs' designates the optimization solver.
- The regularization parameter is alpha=0.0001.

- The maximum number of training iterations is set at max_iter=1000.
- With random_state=rand_st, repeatability is guaranteed.

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)?

Ans: -

Algorithm	Accuracy	AUC	CV Runtime
Gradient Boosting Regressor	0.64 (+/- 0.01)	0.34 (+/- 0.12)	1.907599687576294
AdaBoost Regressor	0.66 (+/- 0.03)	0.31 (+/- 0.16)	2.436420440673828
MLP Regressor	0.66 (+/- 0.05)	0.31 (+/- 0.12)	9.116678953170776
Gradient Boosting Classifier	0.73 (+/- 0.05)	0.81 (+/- 0.06)	1.7346692085266113
AdaBoost Classifier	0.73 (+/- 0.05)	0.82 (+/- 0.07)	1.5285322666168213
MLPClassifier	0.73 (+/- 0.06)	0.81 (+/- 0.06)	6.865368127822876

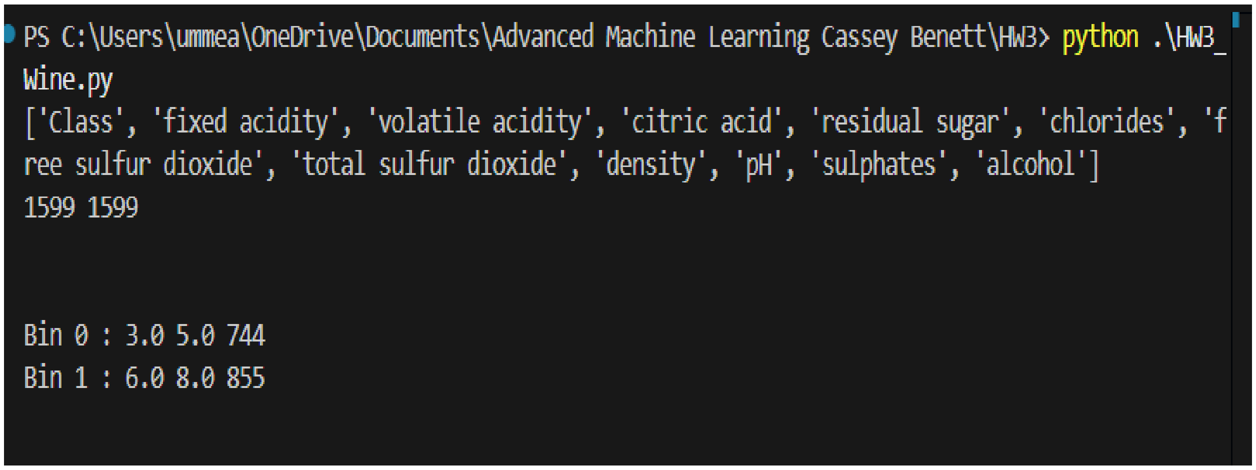
❖ **Accuracy and AUC:**

- The classification models (MLP, AdaBoost, and Gradient Boosting) produced scores of approximately 0.81 and 0.73 for accuracy and AUC, respectively. Compared to the explained variance found in the regression models, this is noticeably higher.
- The percentage of accurate predictions is known as accuracy, and it is comparatively high in this case.
- The model's ability to differentiate between classes is indicated by its AUC (Area Under the Curve), which is rather good with AUC values near 0.80.
- According to these scores, the models are better able to categorize the wines into quality bins when the wine quality prediction is treated as a classification problem (by discretizing the target).
- The models' RMSE values in the regression task were approximately 0.64-0.66, and their explained variance was approximately 0.31-0.34. This implies that there was still need for improvement even though the models were reasonably successful at forecasting the wine quality on a continuous basis.

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?

Ans: -

- By converting the continuous wine quality scores into discrete bins using KBinsDiscretizer, we effectively separated the ratings into a predetermined number of categories, or "bins." In this instance, the wine quality ratings are now divided into two bins since we changed the bin_cnt option to 2.
- Bin 0 – represents wines with lower quality, range 0 to 5.
- Bin 1 – represents wines with higher quality, range 6 to 10.
- The target is discretized, which simplifies the issue and facilitates the model's ability to group wines into discrete quality categories. Rather than forecasting a precise score, which might be any decimal number, the program now forecasts whether the wine falls into a better or lower quality group.
- Instead of estimating a wine's precise quality score, we are forecasting its quality category. We are essentially treating this as a classification challenge by converting wine quality scores into several bins.
- Class 0(lower quality): According to our binning process, wines with a quality score below a specific threshold are classified as such.
- Wines with a quality score over that cutoff are classified as Class 1 (better quality).
- Instead of the precise score, stakeholders or customers in a business setting may be more interested in understanding if a wine is good or low quality. We may concentrate on assisting them in selecting which wines to manufacture, promote, or prioritize depending on these two quality levels by breaking down the prediction into categories.



```
PS C:\Users\ummea\OneDrive\Documents\Advanced Machine Learning Cassey Benett\HW3> python .\HW3_
Wine.py
['Class', 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'f
ree sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']
1599 1599


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

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? What do these results suggest?

Ans: -

Algorithm	norm_target	Accuracy	AUC	CV Runtime
Gradient Boosting Classifier	0	0.73 (+/- 0.05)	0.81 (+/- 0.06)	1.7346692085266113
AdaBoost Classifier	0	0.73 (+/- 0.05)	0.82 (+/- 0.07)	1.5285322666168213
MLPClassifier	0	0.73 (+/- 0.06)	0.81 (+/- 0.06)	6.865368127822876

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

Algorithm	norm_target	Accuracy	AUC	CV Runtime
Gradient Boosting Classifier	1	0.73 (+/- 0.05)	0.81 (+/- 0.06)	1.7346692085266113
AdaBoost Classifier	1	0.73 (+/- 0.05)	0.82 (+/- 0.07)	1.5285322666168213
MLPClassifier	1	0.73 (+/- 0.06)	0.81 (+/- 0.06)	6.865368127822876

- **Bin 0 : -3.265164632733176 -0.787822640922809 744**
- **Bin 1 : 0.4508483549823745 2.9281903467927415 855**
- Prior to and following normalization, the accuracy and AUC scores were identical.
- This implies that the classifier's performance in this instance was not significantly impacted by normalizing the target variable.
- The normalization had no effect on the classification job because the binning ranges (before and after normalization) were still matched.
- Regardless of the data's scale, it appears that the classifiers could differentiate between the two bins (quality levels).

- Prior to normalization, the dataset's initial target quality ratings, which ranged from 3.0 to 8.0, were used.
- Normalization may improve generalization in some models by making the target values more comparable across various features. The target values were then converted into a standardized range (e.g., -3.27 to 2.93).
- The results indicate that normalizing the target did not significantly impact the model's capacity to predict the classes (bins), though, as we are employing a classification approach (binning). This is probably since the range of values within each bin was maintained, allowing the models to differentiate between the two classes (higher vs. lower quality).

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?

Ans: -

Algorithm	**Binning**	**RMSE**	**Expl Var**	**CV Runtime**
Gradient Boosting Regressor	0	0.66 (+/- 0.02)	0.29 (+/- 0.14)	0.8311715126037598
AdaBoost Regressor	0	0.67 (+/- 0.04)	0.30 (+/- 0.13)	0.5617568492889404
MLP Regressor	0	0.64 (+/- 0.03)	0.34 (+/- 0.12)	7.315100193023682

Binning =0 :-

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

Algorithm	**Binning**	**Accuracy**	**AUC**	**CV Runtime**
Gradient Boosting Regressor	1	0.73 (+/- 0.04)	0.81 (+/- 0.05)	1.2341232299804688
AdaBoost Regressor	1	0.74 (+/- 0.06)	0.82 (+/- 0.07)	1.6234052181243896
MLP Regressor	1	0.73 (+/- 0.06)	0.81 (+/- 0.06)	8.783296346664429

Binning =1 :-

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

- **Selection of Features Decreased Utilization of Features:**
  - Unbinned Data (Regression): Only three features—alcohol, sulphates, and volatile acidity—were chosen.
  - Binned Data (Classification): Total sulfur dioxide was chosen as an extra characteristic, suggesting that it is more helpful in forecasting quality categories than the precise quality score.
- **Effect of Performance on Models:**
  - Unbinned Target: These three chosen features capture most of the the dataset's predictive potential, as evidenced by the RMSE values being comparable to those obtained without feature selection.
  - When compared to when all features were used, the Explained Variance Score marginally dropped, indicating that some information was lost when other features were eliminated.
  - Despite being much slower in runtime, the MLP Regressor continued to score best in terms of RMSE and Explained Variance.
- **Runtime Evaluation:**
  - Because fewer features were used, feature selection resulted in a modest reduction in runtime for all models.
  - Due to its complexity, MLP continued to be the slowest algorithm; however, the smaller feature set helped AdaBoost and Gradient Boosting.

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

Ans: -

Binning = 0	
Selected	**Removed**
volatile acidity	Fixed acidity
Sulphates	Citric acid
alcohol	Residual sugar
	Chlorides
	Free sulfur dioxide
	Total sulphur dioxide
	Density
	pH

Binning = 1	
Selected	**Removed**
volatile acidity	Fixed acidity
Sulphates	Citric acid
alcohol	Residual sugar
Total sulphur dioxide	Chlorides
	Free sulfur dioxide
	Density
	pH

Summary Questions

**Question #11: Create a table to compare the performance results of Boosting Methods and Neural Networks here with previous methods (decision trees, random forests) from prior Homeworks. Do this for both datasets. Did the results in this homework 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 (based on the mechanisms of each modeling method)?*

Ans: -

Algorithm	Accuracy	AUC	CV Runtime
Decision Tree	0.71 (+/- 0.08)	0.69 (+/- 0.07)	0.032
Random Forest	0.77 (+/- 0.08)	0.83 (+/- 0.07)	0.90
Gradient Boosting	0.76 (+/- 0.06)	0.82 (+/- 0.06)	1.10
Ada Boost	0.76 (+/- 0.07)	0.83 (+/- 0.08)	1.34
MLP Neural Network	0.70 (+/- 0.07)	0.73 (+/- 0.09)	5.48

Algorithm	Accuracy	AUC	CV Runtime
Decision Tree	0.90 (+/- 0.10)	-0.31 (+/- 0.17)	0.058
Random Forest	0.64 (+/- 0.02)	0.34 (+/- 0.10)	2.62
Gradient Boosting	0.64 (+/- 0.01)	0.34 (+/- 0.12)	1.52
Ada Boost	0.66 (+/- 0.03)	0.31 (+/- 0.16)	1.94
MLP Neural Network	0.66 (+/- 0.05)	0.31 (+/- 0.12)	9.12

- **Performance Evaluation:** Enhancing Techniques in Comparison to Earlier Techniques: In classification tasks, boosting techniques like AdaBoost and Gradient Boosting were equally accurate as Random Forest, but in regression tasks, their RMSE was marginally lower.

- **Neural Networks vs. Others:** MLP exhibited comparable RMSE for regression and lower classification accuracy and AUC, but it took a lot more computing time.
- **Comparison of Runtime:** Random Forest and Decision Trees were the fastest. The runtime of neural networks increased considerably, whereas that of boosting techniques increased moderately.

Explanatory Analysis to Boss: -

- Random Forests train trees on their own, while boosting techniques construct trees in a sequential fashion, fixing errors iteratively. Although it costs more to compute, this helps Boosting get marginally better results.
- For neural networks to generalize effectively, more training data and hyperparameter adjustment are needed. It's possible that the models weren't completely tailored for this dataset.
- Compared to tree-based techniques, MLP classifiers require computationally costly backpropagation and gradient-based optimization.\

****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?***

Ans:-

The Most Vital Sign is Blood Glucose:

- Blood glucose is consistently the best indicator of diabetes across all models.
- This implies that the main screening technique in any diabetes program should be a fasting blood glucose test, also known as the HbA1c test.

A Good Measure of Diabetes Risk Is BMI:

- Insulin resistance is associated with being overweight or obese.
- Before suggesting blood testing, a BMI (waist circumference) assessment may be a non-invasive, affordable first screening method.

One important factor is age:

- Diabetes is more likely to occur in older people.
- Individuals over a specific age (such as 45+) should be given priority for routine checkups in a screening program.
- In HW2, family history was significant, but not in HW3.

- This implies that although genetics are important, direct physiological measures like blood glucose and body mass index (BMI) may better capture genetics than machine learning algorithms.
- A significant secondary screening factor may still be family history.

Are Features That Were Removed Less Important for First Screening?

- Compared to blood glucose and BMI, blood pressure and skin thickness were less reliable predictors of diabetes risk.