**Homework #1 Instructions**

**Casey Bennett, PhD**

**DSC540, Winter 2019**

**DePaul University**

**Overview**

There are two python scripts, implementing Scikit using the same basic template you were shown in class. The first one uses a dataset related to Diabetes from Pima Indians for classification, the target variable being the presence/absence of diabetes in individuals. The second script uses a dataset of quality ratings for various red wines rated 1-10, for regression. Both scripts implement a basic decision tree from Scikit to make predictions. We will explore different ways to score the predictions and evaluate results, including setting up different scorers, comparing cross-validation to simple test/training splits, and finally look at the effects of simple feature selection.

For classification, we will be creating an object we’ll name ‘clf’, and for regression we’ll name ‘rgr’. These are objects we can call methods on (such as fitting a model to some data), and access their internal variables (such as getting predicted class labels). Scikit API links for decision trees describing methods and variables available can be found in the included links document.

*\*Follow the steps below, record answers to questions in a word document, and turn in both your completed code and the word doc.*

**Pima Diabetes**

Open up HW1\_Diabetes.py

1. First, we need to setup scorers for the simple test/train split. Accuracy can be calculated using the score method within the Decision Tree object (clf), but AUC requires us to use a function from the metrics library on predicted labels.
   1. Note on line 111, the test/training data is already split for you into separate arrays for both the features (data) and the target
   2. Note the Decision Tree classifier being created on line 116, then fitted using the training data on the line below
   3. To calculate accuracy, replace the comment placeholder on line 119 with: clf.score(data\_test, target\_test)
   4. To calculate AUC, replace the comment placeholder on lines 121 with: metrics.roc\_auc\_score(target\_test, clf.predict\_proba(data\_test)[:,1])

*\*Question #1: Run the code 5 times, record the accuracy and AUC scores of each run. What do you notice about the scores?*

*1. ﻿Decision Tree Acc: 0.7063197026022305*

*﻿Decision Tree AUC: 0.6839050476066258*

*2. ﻿Decision Tree Acc: 0.7286245353159851*

*Decision Tree AUC: 0.7369412355600201*

*3. ﻿Decision Tree Acc: 0.6914498141263941*

*Decision Tree AUC: 0.6742249240121581*

*4. ﻿Decision Tree Acc: 0.7063197026022305*

*Decision Tree AUC: 0.6844141488577744*

*5. ﻿Decision Tree Acc: 0.7360594795539034*

*Decision Tree AUC: 0.6988208420792691*

*After running the score 5 times, I noticed that the accuracy score is around .69 to .73 ,and the AUC score has more spread at .67 to .73. Even though the spread of both score are about the same range, they are not consistent.*

1. Next, let’s try changing one of the parameters of the Decision Tree.
   1. On line 116, change the criterion option from ‘gini’ to ‘entropy’

*\*Question #2: Run the code again 5 times, record the accuracy and AUC scores of each run. What do you notice about the scores? How do they compare to scores above in question 1?*

*1. ﻿Decision Tree Acc: 0.6988847583643123*

*Decision Tree AUC: 0.6798261949099939*

*2. ﻿Decision Tree Acc: 0.7174721189591078*

*Decision Tree AUC: 0.6894550342130987*

*3. Decision Tree Acc: 0.7286245353159851*

*Decision Tree AUC: 0.6845088912758086*

*4. ﻿Decision Tree Acc: 0.7323420074349443*

*Decision Tree AUC: 0.7024500907441017*

*5. ﻿Decision Tree Acc: 0.7063197026022305*

*Decision Tree AUC: 0.6797052154195011*

*With changing the criterion, both score look a little bit more consistent which can be noticed from the spread of both score.*

*First of all, it is obvious that the average of accuracy is higher than AUC , but the spread of accuracy is 0.01 point higher which may not make any different. Second, the spread of the both score from ‘entropy’ criteria are higher than ‘gini‘ which make it less consistent.*

1. Now, let’s setup scorers for the cross-validation split. This works a bit differently, we have to set up a dictionary of scorers first, then pass that into the cross\_validate function call. The function will then return a dictionary of scores, which we can call by name.
   1. To turn on cross-validation, we need to first on line 32 change the cross\_val flag to equal 1 instead of 0
   2. To setup the scorers, replace the comment placeholder on lines 127 with:

{'Accuracy': 'accuracy', 'roc\_auc': 'roc\_auc'}

* 1. Note the cross\_validate function call on line 132, with clf object passed in, no need to change this yet
  2. To calculate accuracy, replace the comment placeholder on lines 134 with: scores['test\_Accuracy']
  3. To calculate AUC, replace the comment placeholder on lines 136 with:

scores['test\_roc\_auc']

*\*Question #3: Run the code 5 times, record the accuracy and AUC scores of each run. What do you notice about the scores? How do they compare to the simple test/train split scores in question 1?*

*All 5 run have the same score with the same range. The difference is the CV runtime.*

*Decision Tree Acc: 0.71 (+/- 0.08)*

*Decision Tree AUC: 0.69 (+/- 0.07)*

*Both scores are in moderate performance at accuracy of ~.71 and ACU ~.69 . Compare this to the question 1, it looks like this score has higher spread from the question 1. However, question 1 has only ran 5 times.*

1. Let’s see how the number of cross-validation folds affects performance.
   1. On line 132, change the cv option from 5 to cv=10

Decision Tree Acc: 0.71 (+/- 0.14)

Decision Tree AUC: 0.67 (+/- 0.16)

* 1. Now set the cv option to cv=3

Decision Tree Acc: 0.69 (+/- 0.06)

Decision Tree AUC: 0.67 (+/- 0.05)

* 1. Now set the cv option to cv=8

﻿Decision Tree Acc: 0.70 (+/- 0.09)

Decision Tree AUC: 0.67 (+/- 0.09)

*\*Question #4: Run the code once for each cv setting (3,8,10), record the accuracy and AUC scores. What do you notice about the scores? How do they compare to the CV performance above in question 3?*

First, the accuracy from three cv setting are around .69-.71 with different spread. And, the AUC from three cv setting are the same at .67 with a different in each setting. Moreover, 10 folds setting has the highest spread of score.

Secondly, the performance of cv setting at 3,8,10compared to cv setting at 5 (question 3) can’t really tell much because of the spread of the score is all over the place.

**Wine Quality Dataset**

Open up HW1\_Wine.py … First, let’s repeat the steps we did above for Diabetes, with some tweaks.

1. First, we need to setup scorers for the simple test/train split. For regression problems, both RMSE and Explained Variance requires us to use a function from the metrics library on predicted labels. Note that we have to take the square root of the mean\_squared\_error metric (MSE>>RMSE).
   1. Note on line 186, the test/training data is already split for you into separate arrays for both the features (data) and the target
   2. Note the Decision Tree classifier being created on line 191, then fitted using the training data on the line below
   3. To calculate RMSE, replace the comment placeholder on line 194 with:

math.sqrt(metrics.mean\_squared\_error(target\_test, rgr.predict(data\_test)))

* 1. To calculate Explained Variance, replace the comment placeholder on lines 196 with: metrics.explained\_variance\_score(target\_test, rgr.predict(data\_test))

*\*Question #5: Run the code 5 times, record the RMSE and Expl Variance scores of each run. What do you notice about the scores?*

*1. Decision Tree RMSE: 0.0033872133510769974*

*Decision Tree Expl Var: 0.9996277852116705*

*2. ﻿Decision Tree RMSE: 0.012988662638734483*

*Decision Tree Expl Var: 0.994653905081992*

*3. ﻿Decision Tree RMSE: 0.0060724789007455555*

*Decision Tree Expl Var: 0.9988322515345597*

*4. ﻿Decision Tree RMSE: 0.006853863613216372*

*Decision Tree Expl Var: 0.9986420404656025*

*5. ﻿Decision Tree RMSE: 0.003367385590378044*

*Decision Tree Expl Var: 0.9996228343872587*

*Both scores are relatively good performance with low RMSE and high Explained Variance.*

1. Next, let’s try changing one of the parameters of the Decision Tree.
   1. On line 191, change the criterion option from ‘mse’ to ‘friedman\_mse’

*\*Question #6: Run the code again 5 times, record the RMSE and Expl Variance of each run. What do you notice about the scores? How do they compare to scores above in question 5?*

*1. ﻿Decision Tree RMSE: 0.01107855473426024*

*Decision Tree Expl Var: 0.9963753606664163*

*2. ﻿Decision Tree RMSE: 0.015856851268232837*

*Decision Tree Expl Var: 0.9926113572527766*

*3. ﻿Decision Tree RMSE: 0.019332614929181202*

*Decision Tree Expl Var: 0.9895393722478621*

*4. ﻿Decision Tree RMSE: 0.017461373186878196*

*Decision Tree Expl Var: 0.9899368012720802*

*5. ﻿Decision Tree RMSE: 0.007831227553838537*

*Decision Tree Expl Var: 0.9980767960557615*

*Overall performance with the friedman\_mse is good with low RMSE and high Explained variance. Compare to question 3, the overall RMSE from mse criterion seem to be lower but the Explained look the same from both criterions.*

1. Now, let’s setup scorers for the cross-validation split. This works a bit differently, we have to set up a dictionary of scorers first, then pass that into the cross\_validate function call. The function will then return a dictionary of scores, which we can call by name. For RMSE, we have to again take the square root of MSE (in this case flipping the negative sign first).
   1. To turn on cross-validation, we need to first on line 32 change the cross\_val flag to equal 1 instead of 0
   2. To setup the scorers, replace the comment placeholder on lines 202 with:

{'Neg\_MSE': 'neg\_mean\_squared\_error', 'expl\_var': 'explained\_variance'}

* 1. Note the cross\_validate function call on line 207, with rgr object passed in, no need to change this yet
  2. To calculate RMSE, replace the comment placeholder on lines 209 with: np.asarray([math.sqrt(-x) for x in scores['test\_Neg\_MSE']])
  3. To calculate Explained Variance, replace the comment placeholder on lines 210 with:

scores['test\_expl\_var']

*\*Question #7: Run the code 5 times, record the RMSE and Expl Variance scores of each run. What do you notice about the scores? How do they compare to the simple test/train split scores in question 5?*

﻿Decision Tree RMSE:: 0.01 (+/- 0.01)

Decision Tree Expl Var: 1.00 (+/- 0.01)

Both score are perform really well with low RMSE and high explained variance. The RMSE and Explained variance score from cv have almost the range as they are from split dataset.

1. Let’s see how the number of cross-validation folds affects performance.
   1. On line 207, change the cv option from 5 to cv=10

﻿Decision Tree RMSE:: 0.01 (+/- 0.01)

Decision Tree Expl Var: 0.99 (+/- 0.01)

* 1. Now set the cv option to cv=3

﻿Decision Tree RMSE:: 0.01 (+/- 0.01)

Decision Tree Expl Var: 0.99 (+/- 0.01)

* 1. Now set the cv option to cv=8

﻿Decision Tree RMSE:: 0.01 (+/- 0.02)

Decision Tree Expl Var: 0.99 (+/- 0.02)

*\*Question #8: Run the code once for each cv setting (3,8,10), record the RMSE and Expl Variance. What do you notice about the scores? How do they compare to the CV performance above in question 7?*

The cv setting with 3 folds and 10 folds perform really well with the same number, and cv setting with 8 folds also performs really well but has a bit higher spread. The result from question 8 is not really different from question 7 in term of performance.

1. Finally let’s see how feature selection affects performance. First, let’s turn on the LV Filter, which will filter out variables with low variance, e.g. if there are 100 samples but 95 of them have the exact same value for a variable, it’s probably not that useful a predictor feature.
   1. Set the cv option on line 207 back to cv=5
   2. To turn on the LV Filter, we need to first on line 39 change the lv\_filter flag to equal 1 instead of 0

*\*Question #9: Run the code once, record the RMSE and Expl Variance. What do you notice about the scores? How do they compare to the CV performance above in question 7? What features were selected, and which were removed?*

*﻿Selected: ['fixed acidity', 'residual sugar', 'free sulfur dioxide', 'total sulfur dioxide', 'alcohol']*

*Features (total, selected): 11 5*

*﻿Decision Tree RMSE:: 0.23 (+/- 0.03)*

*Decision Tree Expl Var: -0.72 (+/- 0.46)*

*For the score with the feature selection, RMSE is higher obviously and Explained variance is significant low. Comparing score with filtering out low variance and without show that the performance from question 7 is better. However, the score from question 7 may cause the overfit to the model because all of the features are in the model. Five features are selected which are 'fixed acidity', 'residual sugar', 'free sulfur dioxide', 'total sulfur dioxide', 'alcohol'. Six feature are not selected which are*  *﻿volatile acidity ,citric acid,* *﻿chlorides, density, pH, sulphates*

1. Now let’s see how a more involved feature selection method affects performance. 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 Decision Tree regressor model for this.
   1. Set the lv\_filter on line 39 back to lv\_filter=0
   2. To turn on feature selection, we need to first on line 37 change the feat\_select flag to equal 1 instead of 0
   3. Note that there is an option to change the feature selection type on line 38, but the homework code is hard-coded to only use wrapper-based, so this doesn’t matter for now
   4. You will need to add a DecisionTreeRegressor() call to pass to the rgr object on line 147, you can use something similar to the calls used elsewhere in the code (e.g. line 191 or 206). Don’t forget to set the parameters, particularly the random\_state.
   5. Note the SelectFromModel() function being called on line 148, this is where the actual feature selection occurs, with the rgr object being passed in

*\*Question #10: Run the code once, record the RMSE and Expl Variance. What do you notice about the scores? How do they compare to the CV performance above in question 7? What features were selected, and which were removed?*

﻿ Decision Tree RMSE:: 0.24 (+/- 0.04)

Decision Tree Expl Var: -0.77 (+/- 0.55)

﻿Wrapper Select:

Selected: ['fixed acidity', 'alcohol']

Features (total/selected): 5 2

Right answer

﻿Wrapper Select:

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

Features (total/selected): 11 3

--ML Model Output--

Decision Tree RMSE:: 0.91 (+/- 0.08)

Decision Tree Expl Var: -0.35 (+/- 0.26)

CV Runtime: 0.029790163040161133

I notice RMSE is a bit higher and Explained variance is a bit lower by 0.05. The performance is close in term of score is close to the performance in question 9. Two features are selected which are 'fixed acidity', 'alcohol' and three feature are not selected that are *, 'residual sugar', 'free sulfur dioxide', 'total sulfur dioxide'.*

**Summary Questions**

*\*Question #11: Were there any notable differences in performance between the two datasets? Given that one was a classification problem and the other a regression problem, can we say that the Decision Tree model performed better on one of them? Why or why not?*

*Compare the performance from both dataset, It cannot really tell the differences since both dataset are computed with different score even though both dataset are used Decision tree method. In my opinion, I think we also cannot say that the Decision Tree model perform better on either one of the dataset. This is because there are several things to be considered such as features selection the being apply to just one dataset and the scores that are computed on both datasets are different.*

*\*Question #12: Based on the results you obtained, would you say that Decision Tree is a “good” model for these two datasets? If your boss or a customer asked you to build a decision tree for one of them, what would you tell him/her?*

Based on the results, I think Decision Tree model is a good model for initial start to the problem for these datasets. The result show a moderate to a good performance that the accuracy and AUC in the Diabetes dataset is around ~.70, and for the Wine quality dataset after feature selection show RMSE at ~.24 and explained ~ -.77.

I would tell my boss or customer that it is a great approach since the results that are obtained show at least moderate to a good performance. Also, all the result seem to be consistent with a little of spread.