Alex Teboul

Homework #4 Instructions

Casey Bennett, PhD

DSC540, Winter 2019

DePaul University

<u>Overview</u>

Same two datasets as previous homeworks (Diabetes and Wine Quality), along with the two Python scripts. We will explore ways of using SVMs and other discriminant analysis methods. We're also going to investigate feature selection in a little more detail.

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 are in the accompanying document. We will only use cross-validation in this homework.

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

<u>Pima Diabetes</u>

Open up HW4_Diabetes.py

1) First, let's run an SVM classifier.
   a. First we need to import the functions, on line 11, replace the comment with a call to import SVC () from the sklearn "SVM" package.
   b. On line 278, create a SVC(). Using the API link in the accompanying document, call that function, and pass in the following parameters:
      i. Set kernel to 'rbf'
      ii. Set gamma = 'scale'
      iii. Set C = 1.0
      iv. Set probability = True

v. Set random_state variable to rand_st

c. Add in a cross_validate function on line 279 (use previous homework as an example) with 5 folds, and pass in the clf object.

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

- **--ML Model Output--**
- **SVM Acc:** 0.76 (+/- 0.04)
- **SVM AUC:** 0.82 (+/- 0.07)
- **CV Runtime:** 0.34365177154541016
- I notice that the AUC is slightly higher than the Accuracy and both are relatively good. They don't present a very large range of scores from the cross validation which is promising. This is compared to some of the models we saw previously like NN that had a much larger >0.15 range.
- I also notice that the scores are most similar to the Random Forest implemented in HW2 (0.76 (+/- 0.05) | 0.82 (+/- 0.06)) and the boosting methods from HW3 (0.77 (+/- 0.05) | 0.83 (+/- 0.07)).

Question #1b: In the Scikit API for SVC, it explains the probability parameter … why did we set it equal to 'True'? What does that do?

- By setting probability=True, we are enabling class membership probability estimates with methods *predict_proba* and *predict_log_proba*. In our case I believe *predict_log_proba* will be used because we are interested in a binary class diabetes problem.
- From what I can understand of the documentation, SVC uses a hyperplane to create the appropriate decision regions and does not naturally give us a probability estimate as to whether an observation is a member of a certain class. That said, we can leverage Platt scaling as the sklearn implementation does to first train the SVC and then run a separate cross-validated logistic regression to map SVC outputs to probability values. There are some caveats to using this, where if the dataset is too small, the predicted probabilities might not be very appropriate.
    - I believe we could run some form of MyModel = SVC.fit(X_values_standardized, y_values). Then for an unseen observation new_obsv = [[.1 , .5, .4, .15, ….]] we could use MyModel.predict_log_proba(new_obsv) to get some class probability estimated on the observation we had.

2) Let’s explore how changing some of the parameters for the SVM

   a. Change the kernel parameter of the SVC() to ‘sigmoid’

   b. Now set the kernel to ‘linear’

Question #2: Run the code <u>once</u> for each setting of the kernel, record the accuracy and AUC scores. What do you notice about the scores compared to Question #1? What about run-times?

- **--ML Model Output-- kernel = ‘sigmoid**
- **SVM Acc:** 0.50 (+/- 0.11)
- **SVM AUC:** 0.32 (+/- 0.11)
- **CV Runtime:** 0.5323696136474609

- **--ML Model Output-- kernel = linear**
- **SVM Acc:** 0.77 (+/- 0.05)
- **SVM AUC:** 0.83 (+/- 0.05)
- **CV Runtime:** 81.90785026550293

- First observation is that the kernel = ‘sigmoid’ SVM had much worse Accuracy and AUC. Very poor performance with the sigmoid kernel model was accompanied by a run-time that was similar to that of the SVM with rbf kernel.
- Second observation is that the SVM with linear kernel had better performance in terms of higher Accuracy and AUC. Performance appears to be slightly higher than that of the rbf kernel model. That said, the run-time was significantly longer versus either of the other two kernel methods. The SVM with linear kernel was about 230 times slower than the rbf kernel model. It is clear that of the 3 the rbf kernel was the best option.

3) Finally, let’s run feature selection again on the Diabetes dataset, but this time do it using SVMs. Since SVMs depending on the kernel type can function like a linear regression method, and produce coefficients that we can use as a measure of “feature importance” natively.

   a. Let’s leave the SVC() kernel set as ‘linear’

   b. To turn on feature selection, we need to first on line 38 change the feat_select flag to equal 1 instead of 0

   c. Note that there is an option to change the feature selection type is already set to 2 (wrapper-based) on line 39

d. You will need to add a SVC(), call to pass to the clf object on line 191, you can use something similar to the calls used elsewhere in the code. Don't forget to set the parameters, particularly the kernel.

e. Make sure you set the kernel to the 'linear' when using it for feature selection (so it produces coefficients for FS), or your code will give errors.

Question #3a: Run the code once, record the accuracy and AUC scores. What do you notice about the scores? How do they compare to the performance Question 2 above for SVMs using a linear kernel with no feature selection?

- **--ML Model Output--**
- **SVM Acc:** 0.65 (+/- 0.00)
- **SVM AUC:** 0.50 (+/- 0.25)
- **CV Runtime:** 0.11620306968688965

- I notice that the scores with feature selection on are much lower than they were when feature selection was not turned on. Accuracy dropped from 0.77 (+/- 0.05) to 0.65 (+/- 0.00) and AUC dropped from 0.83 (+/- 0.05) to 0.50 (+/- 0.25). The only benefit was that runtime dropped from roughly 83 seconds to 0.1 seconds with feature selection turned on for the SVM with linear kernel. One other observation is that the accuracy stayed the same for each cross validation, the no range in accuracy could be partially due to the single feature that was selected for.

Question #3b: What features were selected, and which were removed? Were there any differences from when you did feature selection with Boosting in HW3, or Random forests in HW2?

- **-FEATURE SELECTION ON--**
- **Wrapper Select:**
- Selected ['Family History']
- **Features (total/selected):** 8 1

- There was only a single feature selected for here: 'Family History'
- The other 7 features were removed. These were: 'Times Pregnant', 'Blood Glucose', 'Blood Pressure', 'Skin Fold Thickness', '2-Hour Insulin', 'BMI', and 'Age'

- This time, the only feature selected was 'Family History, which wasn't selected in HW3 but was in HW2. The accuracy and AUC were higher in HW2 and HW3 compared to this homework for feature selection though. I have summarized the differences in feature selection between these homeworks in the table below.

	Selected Features	Removed Features
HW4	‘Family History’	'Times Pregnant', 'Blood Glucose', 'Blood Pressure', 'Skin Fold Thickness', '2-Hour Insulin', 'BMI', ‘Age’
HW3	Blood Glucose, BMI, Age	Family History, Times Pregnant, Blood Pressure, Skin Fold Thickness, and 2-Hour Insulin
HW2	Blood Glucose, BMI, Family History, Age	Times Pregnant, Blood Pressure, Skin Fold Thickness, 2-Hour Insulin

Wine Quality Dataset

Open up HW4_Wine.py … First, let’s repeat the steps we did above for Diabetes.

4) First, let’s run an SVM regressor.

   a. First we need to import the functions, on line 12, add a call to import SVR() from the sklearn “SVM” package.

   b. On line 368, create a SVR(). Using the API link in the accompanying document, call that function, and pass in the following parameters:

      i. Set kernel to ‘rbf’

      ii. Set gamma = 0.1

      iii. Set C = 1.0

   c. Add in a cross_validate function on line 369 (use previous homework as an example) with 5 folds, and pass in the rgr object.

Question #4a: Run the code once, record the RMSE and Explained Variance.

- **--ML Model Output--**
- **SVM RMSE:** 0.77 (+/- 0.06)
- **SVM Expl Var:** 0.08 (+/- 0.10)
- **CV Runtime:** 1.0178759098052979

- I notice that the explained variance is very low and the RMSE is higher than most of the other methods we have tried in previous assignments.

Question #4b: In the Scikit API for SVR, you will notice there is no probability parameter (averse to for the classifier version), why do you think that is?

- For the classifier version, the probability parameter can be used to output a probability estimate that an observation is a member of a certain class. In the case of SVR, or linear regression problems in general, we are not predicting classes but continuous values. You can't really predict the probability of a new observation leading to a particular continuous value unless you were treating all continuous values as unique classes - but that also doesn't make sense / wouldn't be useful to us here.

5) Let's explore how changing some of the parameters for the SVM
    a. Change the kernel parameter of the SVR() to 'sigmoid'
    b. Now set the kernel to 'linear'

Question #5: Run the code <u>once</u> for each setting of the kernel, record the RMSE and Explained Variance. What do you notice about the scores compared to Question #4? What about run-times?

- **--ML Model Output-- kernel = 'sigmoid'**
- **SVM RMSE:** 0.91 (+/- 0.22)
- **SVM Expl Var:** 0.00 (+/- 0.00)
- **CV Runtime:** 0.4709208011627197

-
- **--ML Model Output-- kernel = 'linear'**
- **SVM RMSE:** 0.66 (+/- 0.02)
- **SVM Expl Var:** 0.29 (+/- 0.19)
- **CV Runtime:** 45.67411732673645

- I notice that the sigmoid model is worse in terms of RMSE and Expl Var than the rbf model and the linear model. The linear model has better RMSE and Expl Var than either of the other two models.
- If we ignore runtime, then I would argue that in terms of performance on this particular wine dataset linear > rbf > sigmoid.
- In terms of runtimes, sigmoid is the fastest, followed by rbf, and finally linear is the slowest. The runtime for linear is much longer than the other two models - being almost 45 times slower than the rbf kernel model and 90 times slower than the sigmoid model.

For this dataset though, 45 seconds isn't a terribly long amount of time to wait. Also, the runtime speed increase gained by the sigmoid model is not worth the very poor performance. None of the models are particularly impressive, but linear followed by rbf would be my picks for this question as the best models of the three.

6) Discriminant Analysis methods can be prone to issues with data distributions, so let's see if normalizing the features has any effect.

a. Let's leave the SVC() kernel set as 'linear'

b. On line 35, change the norm_features flag to equal 1 instead of 0

Question #6: Run the code once, record the RMSE and Explained Variance. What do you notice about the scores, or the run times? How do they compare to results in Question #5?

- **--ML Model Output-- kernel = 'linear' | norm_features = 1**
- **SVM RMSE:** 0.66 (+/- 0.02)
- **SVM Expl Var:** 0.29 (+/- 0.16)
- **CV Runtime:** 1.86958646774292

- I notice that the RMSE and Expl Var scores remain virtually the same as before, but we gain a great increase in terms of speed. Specifically, run-time dropped from 45.7 seconds to 1.9 seconds - or about 46 seconds to 2 seconds making it 23 times faster than before. This certainly makes the linear kernel SVR the best model of the bunch explored in Q4-5.

7) Let's run feature selection again on the Wine dataset, just like we did for Diabetes above.

a. Let's leave the norm_features flag turned on (set to 1)

b. To turn on feature selection, we need to first on line 38 change the feat_select flag to equal 1 instead of 0

c. Note that there is an option to change the feature selection type is already set to 2 (wrapper-based) on line 39

d. You will need to add a SVR(), call to pass to the rgr object on line 254, you can use something similar to the calls used elsewhere in the code. Don't forget to set the parameters, particularly the kernel.

e. Make sure you set the kernel to the 'linear' when using it for feature selection (so it produces coefficients for FS), or your code will give errors.

Question #7a: Run the code once Record the RMSE and Explained Variance Score. What do you notice about the scores? How do they compare to performance in Question 6 above for SVMs using a linear kernel with no feature selection?

- **--ML Model Output--**
- **SVM RMSE:** 0.68 (+/- 0.04)
- **SVM Expl Var:** 0.27 (+/- 0.25)
- **CV Runtime:** 0.8909296989440918

- I notice that there is a very slight difference in RMSE that is not really significant. Also the explained variance is less stable than it was in Q6 but the average is consistent between the two problems. There is also an improvement in speed with feature selection turned on.
  - **RMSE Q6 → Q7a:** 0.66 (+/- 0.02) → 0.68 (+/- 0.04)
  - **Expl Var Q6 → Q7a:** 0.29 (+/- 0.16) → 0.27 (+/- 0.25)
  - **Runtime:** 1.9 → 0.9
- Overall, it seems that the feature selection is beneficial here.

Question #7b: What features were selected, and which were removed? How did this compare with features selected in previous homeworks (Random Forests, Gradient Boosting)?

- **--FEATURE SELECTION ON--**
- **Wrapper Select:**
- **Selected** ['volatile acidity', 'sulphates', 'alcohol']
- Features (total/selected): 11 3

- Volatile acidity, sulphates, and alcohol were selected, while fixed acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, density, pH, and total sulphur dioxide were removed. These are the same three features that were selected in the previous homeworks for RF and GB models. This is in contrast to what we observed in Q3b of this homework where there were some differences in features selection between the homeworks / models.
- I have summarized the selected and removed features from the three homeworks in the table below.

	Selected Features	**Removed Features**
HW4 SVR	volatile acidity, sulphates, alcohol	fixed acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, density, pH, total sulphur dioxide
HW3 GB	volatile acidity, sulphates, alcohol (*+total sulphur dioxide when binned only)	fixed acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, density, pH, total sulphur dioxide
HW2 RF	volatile acidity, sulphates, alcohol	fixed acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, density, pH, total sulphur dioxide

8) Let's compare some other feature selection methods, such as using mutual information between each feature and the target (based on information theory and entropy).

a. Set fs_type line 39 to 3

Question #8a: Run the code once Record the RMSE and Explained Variance Score. What do you notice about the scores? How do they compare to performance above using wrapper feature selection in Question 7?

- **--ML Model Output--**
- **SVM RMSE:** 0.67 (+/- 0.03)
- **SVM Expl Var:** 0.27 (+/- 0.22)
- **CV Runtime:** 1.112194538116455

- I notice that the scores are comparable to what we observed in the previous question for this model when wrapper feature selection was used. There is no significant change in RMSE or Explained Variance here. There is a slight runtime increase due to the greater number of features included, but it's not important here.
    - **RMSE Q7a → Q8a:** 0.68 (+/- 0.04) → 0.67 (+/- 0.03)
    - **Expl Var Q7a → Q8a:** 0.27 (+/- 0.25) → 0.27 (+/- 0.22)
    - **Runtime:** 0.9 → 1.1

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

- **--FEATURE SELECTION ON--**
- Univariate Feature Selection - Mutual Info:
- **Ranked Features**
    - 0 alcohol : 0.18172894446798882
    - 1 volatile acidity : 0.11915824956411836
    - 2 sulphates : 0.09069194713695783
    - 3 density : 0.09046658193209911
    - 4 total sulfur dioxide : 0.07635446522631817
    - 5 citric acid : 0.07193012346967098
    - 6 fixed acidity : 0.05912361895228635
    - 7 chlorides : 0.05889844314002257
    - 8 free sulfur dioxide : 0.05577295659560466
    - 9 residual sugar : 0.032717605876168676
    - 10 pH : 0.017162869877964226
- **Selected** ['volatile acidity', 'total sulfur dioxide', 'density', 'sulphates', 'alcohol']
- Features (total/selected): 11 5
- Here we had:
    - **Selected (5):** alcohol, volatile acidity, sulphates, density, and total sulphur dioxide
    - **Removed (6):** fixed acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, pH
    - So we have density and total sulphur dioxide included in the model now for no significant performance improvement. Interestingly, sulphates and density are very close in terms of their ranked importance here.

9) Let's run feature selection again on the Wine dataset, except using a full-blown wrapper. I've already written the helper function for you (line 55), you just need to turn it on. This is a very straightforward exhaustive search method, with no regularization, so it will probably pick more features than truly necessary. Warning: it can take a while to run, perhaps a couple minutes depending on your computer.

a. Set fs_type line 39 to 4

b. You will need to add a SVR(), call to pass to the rgr object on line 304, you can use something similar to the calls used elsewhere in the code. Don't forget to set the parameters, particularly the kernel.

Question #9a: Run the code once Record the RMSE and Explained Variance Score. What do you notice about the scores? How do they compare to performance above for feature selection using the simple wrapper in Question 7 and the univariate mutual info in

Question 8?

- **--ML Model Output--**
- **SVM RMSE:** 0.66 (+/- 0.02)
- **SVM Expl Var:** 0.30 (+/- 0.18)
- **CV Runtime:** 1.644324779510498
- **Wrapper Feat Sel Runtime:** 218.39492392539978

- I notice that there is a slight improvement in RMSE and Expl Var, but in reality it is a small and not significant change. Really the performance of the simple wrapper in Q7 was probably sufficient for this case. The univariate mutual info Q8 was about the same performance, maybe a little insignificantly better, but also drew in 2 more features. Therefore Q7 simple wrapper seems to be the most appropriate here.
    - **RMSE Q7a→Q8a→Q9a:** 0.68 (+/- 0.04) → 0.67 (+/- 0.03) → 0.66 (+/- 0.02)
    - **ExplVar Q7a→Q8a→Q9a:** 0.27 (+/- 0.25) → 0.27 (+/- 0.22) → 0.30 (+/- 0.18)
    - **Runtime:** 0.9 → 1.1 → 220

Question #9b: What features were selected, and which were removed?

- Combos Searched so far: 231 Current Best Score: 0.6596825116948701
- # of Feature Combos Tested: 231
- 0.6596825116948701 [0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1] 11
- **Selected** ['volatile acidity', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']
- Features (total/selected): 11 9

- **Selected (9):** volatile acidity, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol
- **Removed (2):** citric acid, fixed acidity
- We can see here that many more features were selected than in any other method. This brings it back to the comment about more features being selected than necessary.

10) You might note the long run times for Question #9. You can imagine that would be even more exacerbated with different kernels or ML methods. Given that context, look at the comment on line 56. No changes are needed for this question, nor code to run here.

****Question #10a: Based on the comment on line 56, explain how we might search the feature set space in a more optimal manner.***

- The comment explains that we can change from exhaustive search to greedy search, random search, genetic algos, etc. We can start with regularization to more efficiently search the feature space. Then implement one of the methods the comment mentions.

Looking at papers on the topic online, it is clear that there are many proposed approaches/algorithms for feature selection using these methods - it seems to be a field itself. In general, exhaustive search is too computationally expensive - as was mentioned in class.

Question #10b: If you uncomment the print statements on lines 87 and 96, and watch the code run, you may notice that there are actually several different feature sets that perform nearly the same as the optimal feature set, some of which have much fewer features than others. What is one way we could force the wrapper method to select smaller feature sets, even if they have slightly less performance? (HINT: line 56 also mentions something about this)

- A type of greedy search could function search for the best single feature selected, and then add features and take the best model until a threshold of improvement is failed to be reached. Let's say this threshold is a 0.02 RMSE
- 1) You could run for each feature like:
  - [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] .. → .. [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
  - (11 steps)
  - Then take the best RMSE:
  - [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1] → 0.7266641580854134
    - This is just ^alcohol selected
- 2) Then keep alcohol and add each other option and keep the best pair:
  - [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1] .. → .. [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1].
  - (10 steps)
  - Then take the best RMSE:
  - [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1] → 0.6765697591042016
    - This is volatile acidity + alcohol
  - The threshold improvement of RMSE of 0.02 was reached so we continue.
- 3) Then keep alcohol and volatile acidity, and repeat:
  - [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1] .. → .. [0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1]
  - The threshold improvement of RMSE of 0.02 was not reached, so we can stop.
  - (9 steps)
  - **Final RMSE and Features selected:**
    - [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1] → RMSE 0.68 for volatile acidity and alcohol.
    - This is slightly less performance than the best model, but it took 11+10+9= 30 steps to arrive at rather than 231. Also these steps are using the fewest number of features so it's slightly faster there as well. I know this won't always work but it's one method for this particular problem that could work.

Summary Questions

***Question #11: Line up the results from Homeworks 1,2,3,4 as a table for both Diabetes and Wine, with rows of performance metrics for each ML method (Decision Trees, Random Forests, Gradient Boosting, Ada Boost, Neural Networks, SVMs). Looking at this table of performance metrics, how would you explain the table to a boss or customer?**

The table below presents the best performance of feature selected, un-binned models for each of the homeworks and models tested.

Question 11: Compare RF, DT, GB, Ada, NN, SVM Un-Binned & Feature Selected or Best Models				
	Diabetes		**Wine**	
Model	**Accuracy**	**AUC**	**RMSE**	**Expl Var**
H2: RF	0.76 (+/- 0.05)	0.82 (+/- 0.06)	0.65 (+/- 0.02)	0.33 (+/- 0.11)
H1: DT	0.71 (+/- 0.14)	0.67 (+/- 0.16)	0.91 (+/- 0.08)	-0.35 (+/- 0.26)
H3: GB	0.77 (+/- 0.05)	0.83 (+/- 0.07)	0.66 (+/- 0.02)	0.29 (+/- 0.14)
H3: Ada	0.77 (+/- 0.03)	0.84 (+/- 0.07)	0.67 (+/- 0.04)	0.30 (+/- 0.13)
H3: NN	0.73 (+/- 0.15)	0.78 (+/- 0.17)	0.64 (+/- 0.04)	0.34 (+/- 0.11)
H4: SVM	0.76 (+/- 0.04)	0.82 (+/- 0.07)	0.66 (+/- 0.02)	0.29 (+/- 0.16)

- ***Note that these were the best models for the given tests in terms of performance metrics and runtimes. The 'Feature Selected or Best Models' takes into account that the feature selected models weren't always the best choice. In the case of HW4 for example, on the Diabetes dataset, the rbf kernel model without feature selection performed the best and those metrics were included in the table below. For the HW4 Wine dataset, the chosen SVM is linear kernel SVM with simple wrapper and normalized features.
- **In terms of how I would explain this table to a boss or customer:**
    - I would start by explaining that there were 2 datasets that we looked at, the first having to do with Diabetes and the second with Wine.

- In the Diabetes dataset, we were primarily interested in predicting whether or not someone had diabetes based on different factors. It turned out that blood glucose, BMI, and age are important to making an accurate prediction of diabetes. In the second dataset, we wanted to predict Wine quality based on chemical characteristics of the wine. In this case, features of the wine like volatile acidity, sulphates, and alcohol turned out to be important towards predicting the wine's quality.
- We tested 6 different types of models for these datasets: Decision Trees, Random Forests, Gradient Boosting, Ada Boost, Neural Networks, and Support Vector Machines. We assessed the performance on the Diabetes dataset in terms of Accuracy and Area Under the Curve (AUC) as we wanted to predict *has* vs *has no* diabetes. High Accuracy and AUC mean the model performed well. For wine, we used Root Mean Squared Error (RMSE) and Explained Variance (Expl Var). The lower RMSE and higher Expl Var indicate a better model.

- **Overall,** it appears that the Support Vector Machine models performed the best across the two datasets in terms of getting higher Accuracy and AUC for Diabetes, or lower RMSE and higher Expl Var for Wine, while also having relatively low run-times.

- **In terms of the best models for Wine:**
    - None of the models was particularly good at predicting Wine Quality on a numeric scale from 3-8, and RF, GB, Ada, and SVM performed quite similarly. SVMs seem to be the best in terms of run-times and scores, though it is very close with others like feature selected GB and Ada. That said, regression was not ideal for this dataset.
    - A better approach is to classify the wines as *high* or *low* and try to predict that, which is also well modeled by GB and Ada. Accuracy of 0.73 (+/- 0.05) and AUC of 0.81 (+/- 0.06) was found for GB on this binary classification task.
    - Across all models, volatile acidity, sulphates, alcohol were found to be important.
- **In terms of the best models for Diabetes:**
    - Blood Glucose, BMI, and Age appear to be important features regardless of the model used, so these are important data points to track.
    - Model performance is pretty comparable for RF, GB, and Ada, SVM. With DT and NN having some worse stability in terms of accuracy and AUC scores that vary more. For the NN in particular, turning feature selection on actual makes the model less stable.
    - I would again suggest that SVMs performed the best in terms of keeping run-time low and being relatively simpler than other comparably performing models.
- **In terms of Run-times:**
    - Neural Networks were the slowest models, with Decision Trees being the fastest models. I could suggest to a boss or customer that NN are the most complex model and Decision Trees are the simplest and this is why one takes longer to run than the other. Random Forests, Gradient Boosted, and Ada Boosted models were about the same in terms of run-time. I could explain these run-times being

slower than Decision Trees because these models work by creating ensembles of weaker models or tuning themselves to build stronger models - taking time.The top SVM models were slightly faster than the NN, RF, GB, and Ada models but still slower than decision trees.

Question #12: If we had to explain to someone what really drives peoples' perception of wine quality, what would you say based on your findings in this homework (e.g. Q8) and previous ones? Are there 2-3 features we can say are consistently most important? If so, can you hypothesize why those features might be important?

- Our findings in this assignment suggest that volatile acidity, sulphates, alcohol drive people's perception of wine quality. Across all of our models, these chemical characteristics of wine were the most useful towards predicting wine quality with the least amount of error or greatest accuracy. If we simplify the problem of predicting wine quality to classifying a wine as either high or low quality, we can with high accuracy predict high vs low quality using those three features alone.
- The sulfate feature used in our dataset is actually likely referring to sulfites, which are naturally occurring compounds found in all wines. They can act as a preservative that inhibit microbial growth and maintain freshness. It makes sense that higher quality wines might have more sulfite content, in effect leading to a fresher tasting or smelling wine. That said, a small percentage of the population could be allergic to sulfites, so it is good to limit this amount to average-high average content to ensure freshness without to far influencing taste or otherwise. For reference though, dried fruits can have ten times as much sulfite content at this is on the order of hundreds of parts per million anyways so quite small amounts.
- Volatile acidity is another important feature that refers to the presence of steam distillable acids in the wine. In this case, lower volatile acidity is better than higher volatile acidity. The volatile acidity of a wine can influence its smell, where higher volatile acidity could lead to an unpleasant aroma. For reference, nail polish remover is a solution with a high volatile acidity - and poor smell.
- Finally, alcohol content is important to the quality of wine. There are individual consumer differences in taste for how much alcohol is desired in wine. That said, higher quality wines tend to be on the higher side of alcohol content as well. Go too high though, and you might turn off wine drinkers as well. A sweet spot tends to be between 12-14 percent alcohol content for many wine drinkers.
- To summarize, I think wines with average sulfate content, low volatile acidity, and medium-high alcohol content are likely to appeal to the majority of wine drinkers based on the models we have created.