Machine Learning Project 1

Introduction I chose the Contraceptive Method Choice (CU) Lim, 1997 and Wine Quality Red and White (WQ) **Cortez:2009** datasets. The Contraceptive data was split into 3 categories None, Short Term, and Long term. I wanted to see what factors contribute to people's decisions about using contraceptives, so I changed it to a binary class and grouped short and long term together and renamed it to Contraceptive Used. Wine Quality is rated on a scale of 1-10 but most of the wine only got 5/6. From my online research and playing with the data I saw the 5/6 ratings were the ones that ML algorithms had trouble classifying; they were able to easily rate the outliers but the ratings closer to the middle had a higher error rate. I wanted to target this subset of the data as it was noisy, harder and would provide a learning challenge to create a good ML algorithm. Contraceptive Used has 9 features and 1473 instances and Wine Quality has 12 features and 4432 instances by combining the red and white data-sets. The data-sets are similar percentage-wise with both Wine Quality and Contraceptive Used around 42.5/57.5 {0, 1} class distribution. This makes it interesting to see 2 binary data-sets with similar class distributions but different number of instances. Are learning curves tied to a number of instances or a percent of data? I used a percent of the sample size for the Learning Curve (LC) to try and answer this. Additionally, as the classes were slightly imbalanced I used F1 for scoring to include precision in the equation and penalize false positives. As Brownlee, 6/20/2019 put it "A low precision can also indicate a large number of False Positives". As 1/2 of F1 is precision, a low precision score will weigh it down so that even if the model labels everything as positive the score will still be under 57%. A `random_state() = 42` was used for all models to standardize results. Another difference in the data-sets is that Contraceptive Used has many categorical features with a logical ordering but implicit scaling; whereas Wine Quality is pure numerical data.

The data was first split 80-20 for train-test so that the test set wouldn't leak into the training data then pre-processed with `Quantile_Transformer()`. I used `Quantile_Transformer()` as some features in both data-sets had orders of magnitude in difference. The pre-processing to scale the data is to equate features for KNN and ANN and made it into a Gaussian Distribution for SVM. All model-Complexity Curves were run with 5-Fold validation and the scores where averaged to find the best Hyper Parameter (HP) for the model. `GridSearchCV` was used to find the hyperparameter with the method described by Hsu et al., 5/19/2016 in which a coarse `GridSearchCV` is computed to find a good region after which a `validation_curve` is computed for the best hyperparameters. I first plotted all the Learning Curves with basic default parameters for a baseline. The things I changed were for Decision Trees (DT) I used Entropy as that is what we talked about in class and understand. For KNNs algorithm I used brute so that it would be pure KNN and not resemble a Decision Tree if auto would use a Tree type KNN. For SVM and ANN I set the `max_iter` to 10,000 to start and for ANN I used `warm_start` hoping it would give me better results faster.

Contraceptive Used: DT, KNN, ADA These are the Learning Curves for the 1st data-set. Immediately, we see Decision Trees high variance compared to all the other models; KNN has much less variance and ADA Boost seems to be the best out-of-the-box model. The default settings in SKLearn allow the Decision Tree to grow as much as possible which creates high variance as the Tree will keep splitting and classifying the training data as much as possible. This ends up mis-classifying the validation set leading to lower scores. Contraceptive Used has a very small Dimensionality; only 1 column has 2 significant digits and the rest are under 10. There-

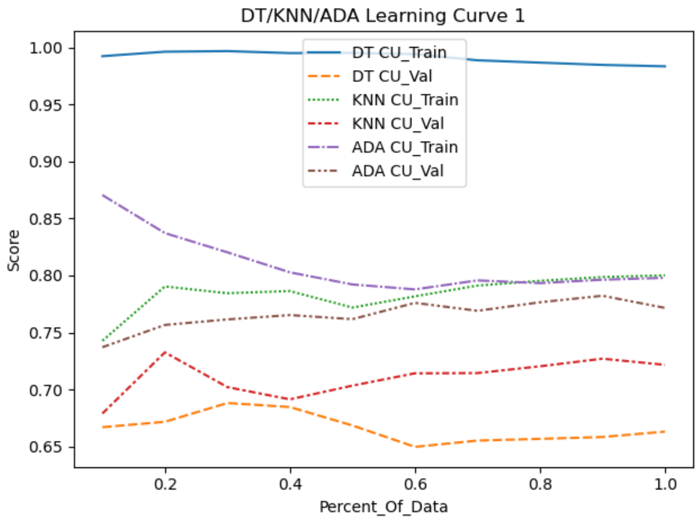


Figure 1: DT-KNN-ADA LC1

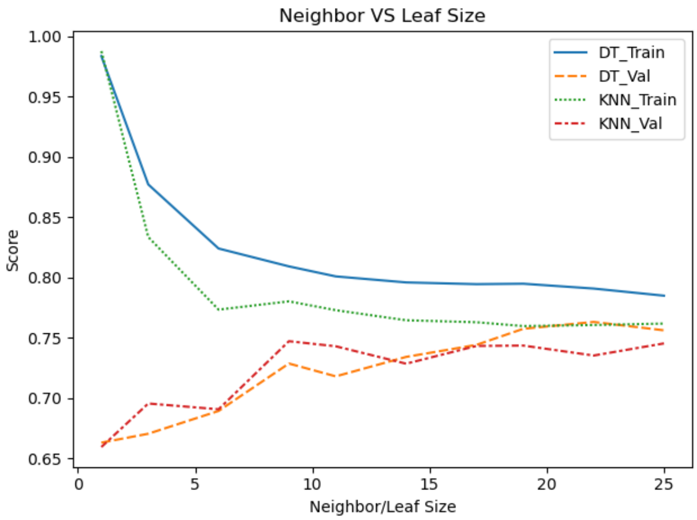


Figure 2: Neighbor VS Leaf Size

fore, it has less options and some instances have the same values with opposing classifications leading to a slightly less accurate score as the Tree can't create 2 leaves with all the same features but opposite predictions. Note that as the score slightly declines for Contraceptive Used train - indicating this mis-classification - the validation score has a corresponding incline indicating a slight generalization and that sometimes mis-classification in train is good for the model. KNN has less variance as there is no training and the model just finds the instances that are closest to classify the data. Obviously the training data is closer to itself than the validation data which gives it some variance. ADA Boost has the least as it focuses on the data that it got wrong and harder to classify which gives it much less variance than Decision Trees or KNN. Interestingly, I noticed that if `n_neighbors = 1` it looks like the Decision Tree split. After thinking about it this made sense as the 1st nearest neighbor in the training set will always be itself; it's basically saying i=i or an identity function. Un-pruned Decision Trees behave the same way as they will attempt to classify all data instances leading to as many leaves as possible. This correlation between leaves and neighbors made me think that they would have similar validation Curves (VCs) and so I plotted them.

While these curves look similar there are still differences. KNN still has less variance; this is because the way a Decision Tree is split is that it looks at what is the biggest Information Gain. Once an instance is sent down one branch there is no coming back. The Decision Tree will try and classify it correctly in that branch but that may be impossible because the instance may have been split based on a smaller difference in features up the branch and there are really big differences later. KNN looks at all differences at once. This can lead to a lower variance in its model in spite of noisy data. However, the Decision Trees validation score is better than KNNs train score. One reason may be that since Contraceptive Used features have some Categorical features with an assumed scale (i.e. Standard-of-living index 1-4) does the index at 2 have double/half the effect of 1 or 4 on a decision? So while the ordering is logical as 4 is a higher standard of living than 1, the distance which is so integral to KNN may be off. Note that there are no features which have no ordering (i.e. location) all features are either logically or numerically (i.e. Age) ordered. Additionally, as Leaf size goes up for the Decsion Tree the Fit and Score time will go down as it will have less instances to classify; in contrast the score time for KNN will go up as it increases `n_neighbors` (fit will always be 0). So for the Decision Tree will have the benefits of a better model and reduced score time whereas KNN gets a better model but worse score time. However, it has a lower variance than the Decision Tree so we will have to consider all of these factors when picking our final model. I picked `n_neighbors = 25` and `min_samples_leaf = 22` as those seem to have the best scores.

Another intriguing idea was to compare the `max_depth()` of ADA Boost and Decision Trees; if ADA Boost is so good can it benefit from being more expressive? We can clearly see ADA Boost hit maximum variance before the Decision Tree in Figure 3. However, in the end its variance is *less* than Decision Tree and has a higher score. This is due to its constant refocusing on the harder data and despite having more variance ends up doing better. The Decision Tree starts out with a high bias and does best at a `max_depth()` of 3 (which allows for a maximum of 8 leaves). Afterwords, the Decision Tree just increases its variance and our validation score gets worse as the model doesn't generalise well to unseen data. So we see the bias-variance trade here as we start with a high bias-low variance with an extremely simple model with a depth of 1. We hit a good spot in which our train and Test scores increase in step; then go to low bias-high variance as our model gets more complex and over-fits on the train data making our validation score go down. We see that if the underlying Tree is changed even slightly it has a big effect on ADA Boost as the variance issue will be multiplied by the number of Estimators.

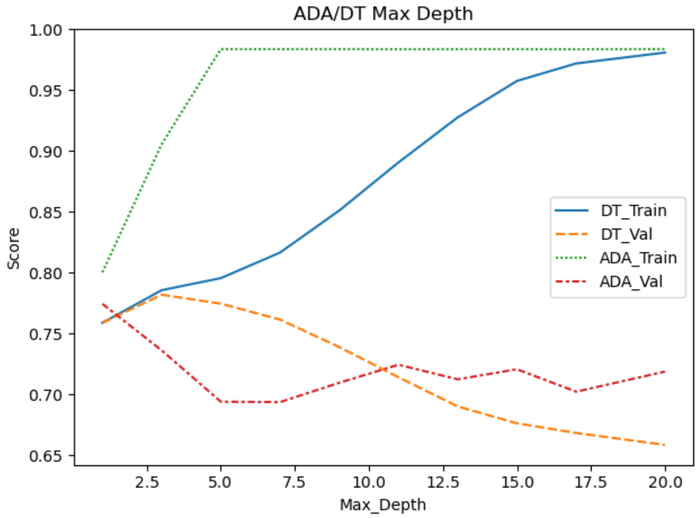


Figure 3: Max Depth

Another interesting comparison was to think of each Decision Tree depth layer as another Estimator. The reason why a depth layer is like 1 estimator is because they both judge the whole data-set. Would ADA Boost have high variance if we allow it to grow its Estimators just like a Decision Tree? Interestingly enough, ADA Boost did well in Figure 4. ADA Boost seems to get a slight increase around 5 estimators, whereas the Decision Tree scores start shooting down at 3. This reinforces that ADA Boost overfits less than other models and that unbounded depth in Decision Trees is bad as it increases variance. On the other hand, being too aggressive with pruning can lead to high bias and reduces our score as well; as seen in the graph where the score goes up in the beginning. Also, note how both Decision Tree and ADA start at the same score. The reason for this is that 1 Estimator with a depth of 1 = 1 Decision Tree with a depth of 1. In contrast, Figure 3 ADA Boost started out with a slightly higher score from the increased number of Estimators. For ADA Boost I picked `n_estimators =`

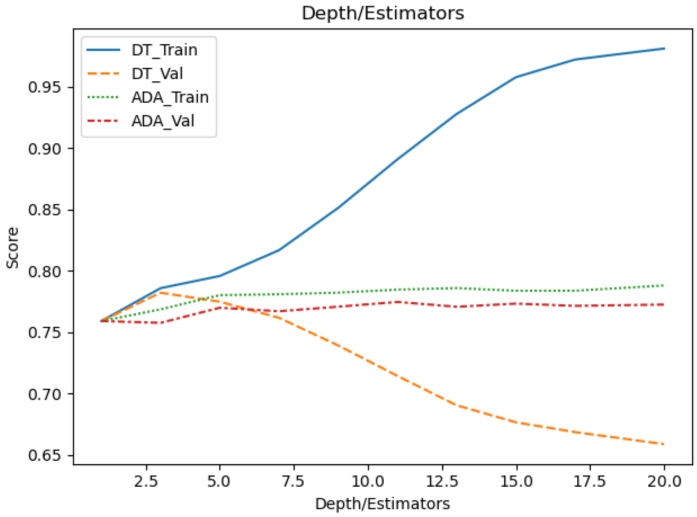


Figure 4: Depth-Estimators

5 as it seems to have a slight boost there and `max_depth = 3` as that seems to be the best for the Decision Tree. This also obviates the `min_samples_leaf = 22` due to the extremely aggressive pruning and the size of the data-set every leaf should have many more samples than 22.

Learning Curves DT/KNN/ADA Round 2

Continuing the iterative process of tuning the hyperparameters, I plotted my next set of Learning Curves with best hyperparameters seen above. I'll start with the DT/KNN/ADA in both data-sets. We see some clear gain in the reduction of variance in the Decision Tree for `max_depth = 3` and KNN for `n_neighbors = 25`, unfortunately we seemed to have introduced bias to ADA Boost with `n_estimators = 5`. We can also see the effect of more data on a high bias model in that as we do get more data our model gets more accurate. This is because it gets to build the model based on more data which ends up generalizing better to the validation set. Seeing as both ADA Boost and Decision Tree have issues focusing on improving the underlying Tree will hopefully help; as opposed to aggressively pruning maxing out depth at 3. Hopefully, using the underlying Tree can also help ADA Boost do better too. Since depth implicitly limits the nodes to a breadth first search we can use that as a baseline region for how many nodes do we need to retain the score we have and hopefully improve it by finding better nodes deeper in the Tree while sacrificing some width nodes. As Depth = 3 means we can have 8 nodes (since it's binary classification $2^3 = 8$). This doesn't end up helping but it does reinforce that the time to hit high variance takes much longer than Depth as we would need double the amount of nodes for each layer of depth. It hurts ADA Boost much more easily as we saw previously. Again this reinforces the fact that ADA Boosts strength comes from being a weak learner and we should be very careful while introducing model Complexity to the underlying Learner. However, 1 strength of ADA Boost is that it is hard to introduce variance to the same level as a non-weak learner. This can be seen here because past the initial drop in accuracy it remains stable and we know from Figure 3 that the Decision Tree will eventually do worse than ADA Boost.

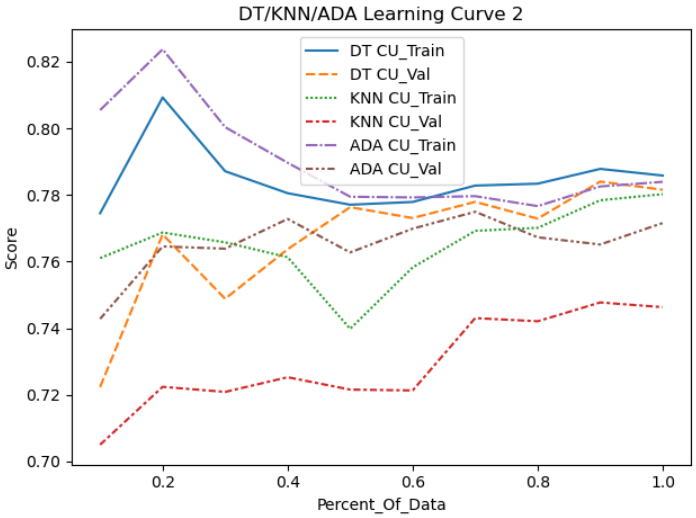


Figure 5: CU DT-KNN-ADA LC2

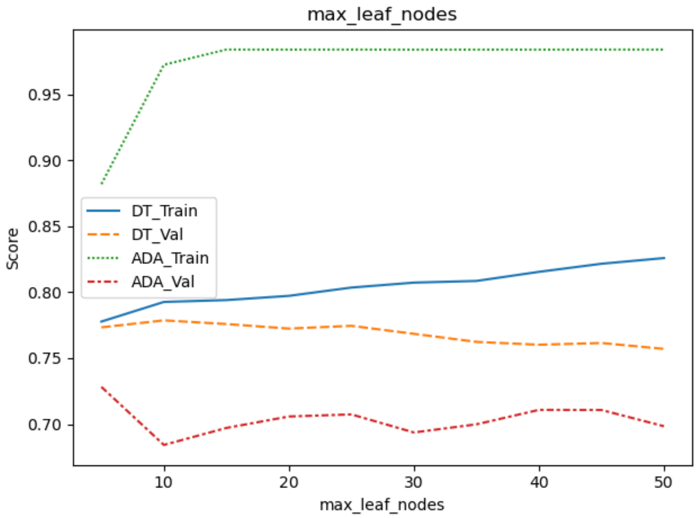


Figure 6: CU max-leaf-nodes

Contraceptive Used: SVM/ANN

From examining this graph alone I would think that the more expressive a model can be the more variance it can/will have; ANNs > RBF > Linear in terms of expressiveness or degrees of freedom. However, from our past graphs we know this is not true. ADA, in Figure 4, with 20 estimators is more ex-

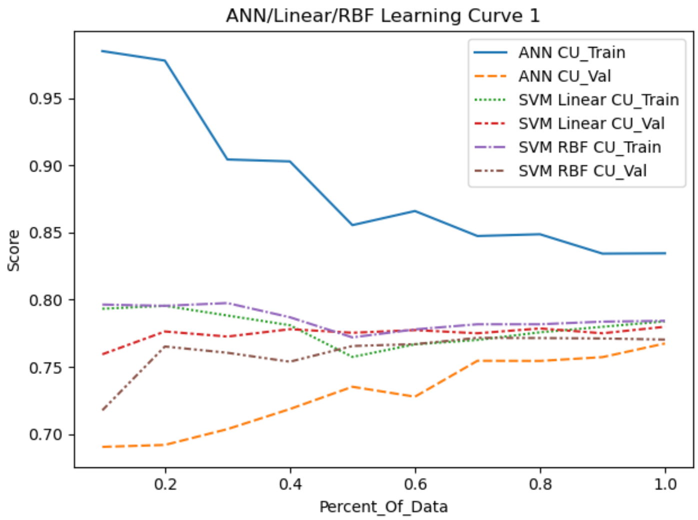


Figure 7: ANN-Linear-RBF LC1

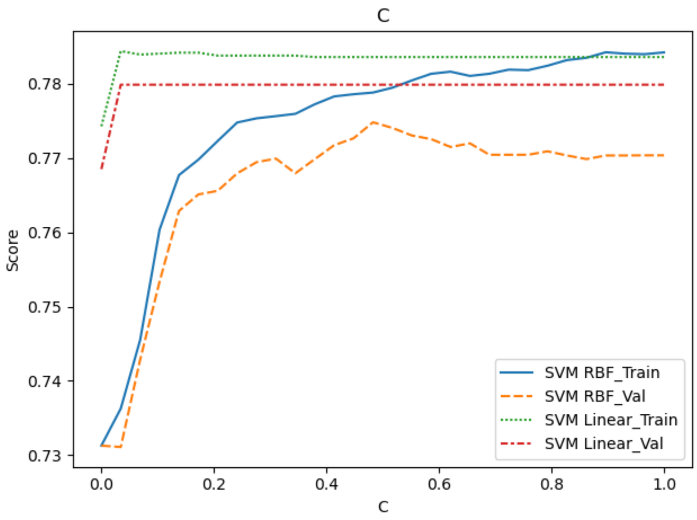


Figure 8: C Regularization

pressive yet has less variance than our Decision Tree with `max_depth()` of 4 (which would have max 16 leaves). Additionally, these models have an order of magnitude longer for there fit times vs Decision Trees and ADA Boost.

However, in SVM ”A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly” (SKLearn, n.d.-a); meaning that C will have a bias/variance trade. Figure 8 shows this beautifully. The RBF Kernel has a lower score due to its bias at 1st, has a point where the score increases in comparative amounts in both train and validation, then the train shoots up with high variance and the validation set suffers. This is similar to Figure 4 where we saw the same bias turns to variance pattern with the best spot in the middle. In contrast, the Linear kernel does start out with some bias but never even gets past that point nor does this turn into variance for a higher C. I believe that since the Linear kernel isn’t as expressive as RBF it cannot get as much variance in this data-set which has noisy instances that it can’t pin down without sacrificing other examples. It can only search and fit its model to Linearly separable Hyper-planes. RBF, which has greater freedom, does get that high variance from overemphasizing the noisy data in the train set by being able to separate the Hyper-planes to the noisy data. The key is to find where to stop emphasizing the train data by reducing C. I set C = .5 as that seems to be the spot right before the variance is introduced into the model.

To better the model even further in RBF I searched for a good Gamma. Gamma is how much an effect do the nearest instances in the Hyper-plane will have. I think of it as a SVM version of `n_neighbors` with distance based weighting because a higher Gamma says that closer instances have more weight. The graph shows this to be true as the more we weight closer

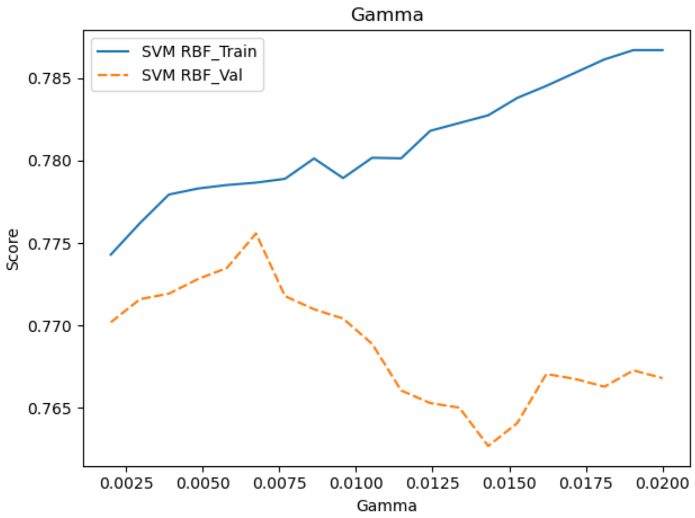


Figure 9: CU Gamma

instances the better the train set does as its instances are always going to be closer to themselves. Again we see how both the train and validation scores go up together and then train continues rising but validation goes down due to train using its own close instances to score itself just like KNNs neighbors = 1 case. I decided to set Gamma to .007 based on this chart as that is the highest point in score with not to much variance.

ANN: Contraceptives Used-Wine Quality

ANNs are unique compared to the other models and will be best showcased if compared side by side. Here too we see that providing more freedom in the form of more nodes is not necessarily a good thing. The training set uses the extra node to make itself more accurate which ends up hurting the validation set. Wine Quality validation set does worse than Contraceptive Used precisely because its train set was able to over-fit itself and introduce more variance than Contraceptive Used. Additionally, since the validation sets aren't doing better with more model complexity I had low hopes that adding Layers would make a difference. This proved to be true that adding more Hidden Layers did not help the validation score but increased the fit time of the model. I even tried to have a node per instance which didn't help either and leads me to believe that there is significant noise in both data-sets. As nobody wants a model that takes extra time yet has the same score I kept the model at 100 nodes. I believe there is a correlation between a small node size and the small depth of a Decision Tree; in both Wine Quality and Contraceptive Used the `max_depth()` is small, 2 and 3 respectively. This means there aren't that many significant factors; in fact on the UCI Wine Quality page it says ”We are not sure if all input variables are relevant” (**Cortez:2009**) which seems to be true. This would lead to smaller Decision Trees as we would be able to achieve maximum score with less splits. In the context of ANNs this translates into less nodes as it only would only have to significantly deal with less Features.

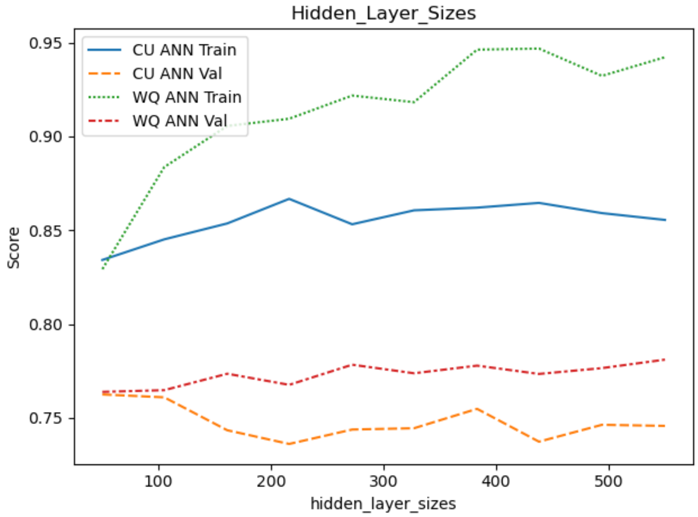


Figure 10: hidden-layer-sizes

The next thing I examined was the Learning Rate. As we see the bias increases as the Learning rate goes up. The reason is again-degrees of freedom. The smaller the node weight update is the greater flexibility the model has to mold itself to the training data. Forcing it to bigger weight update decreases the variance and increases the bias as the model can't mold itself as well to the training data; therefore it generalizes better to the validation set. However, to high a Learning Rate would increase the bias

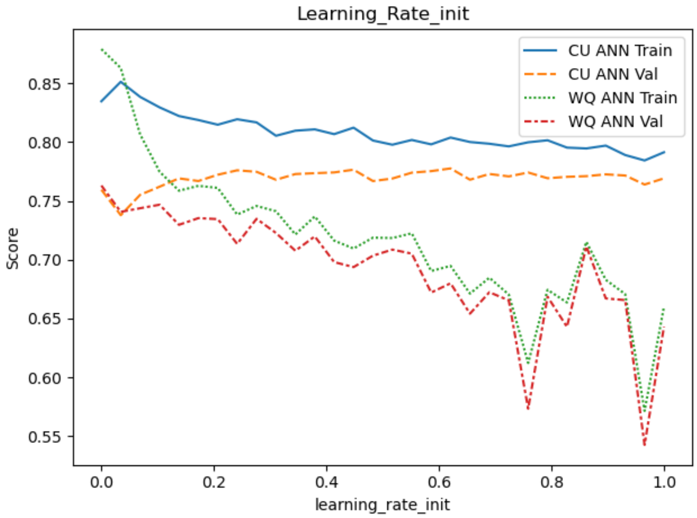


Figure 11: learning-rate-init

and create a lot of error. After setting the Learning Rate for Contraceptive Used to 1.0 and Wine Quality to .2 I plotted the Loss Curves. I picked those numbers as they seemed to have the best reduction in variance for the given score. However, when plotted, both 1.0 and .2 Learning Rates gave an extremely high error. This indicated to me that there was high bias and the models were under-fit. I went back and searched for some better learning rates on a log scale similar to a coarse `GridSearchCV` and plotted the Loss Curves as those are also good indicators of Learning Rates. Learning Rates tied to error as a smaller Learning Rate increases the Hypotheses space of the model and allows it to find one with lower error. However, to low of a Rate would increase the Hypotheses space but it leads to different problems. The model can more easily get stuck in a local minimum, run out of iterations or just be unfeasible time-wise. Learning Rates also have an effect on the time it takes to train the model. A smaller learning rate leads to more weight updates and more training time vs a larger learning rate will translate into less training time for the model. The learning rates chosen were .02 for Wine Quality and .002 for Contraceptive Used.

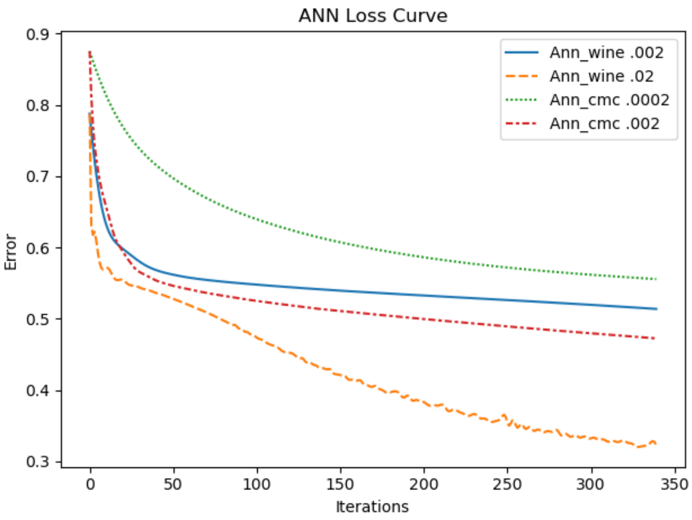
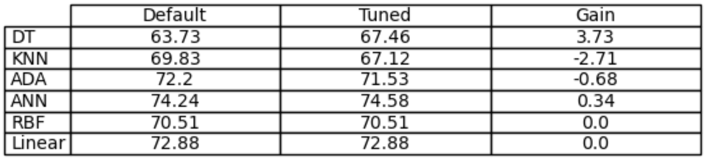


Figure 12: ANN Loss Curve

Final Results

The biggest difference in the gains is Decision Tree and KNN. KNN, as we mentioned suffers in this Categorical data-set as an implicit scaling is assumed in the features which may be an incorrect scaling. So despite increasing `n_neighbors = 25` and our score time, we did worse. In contrast, the Decision Tree did better with aggressive pruning to a depth of 3. We didn't set the Max Leaves to 15 as the pruning was so aggressive at a depth of 3 it wouldn't have made a difference to this model. So our Tree lowered its fit and score time and did better than both the default Decision Tree model and KNNs final model. Additionally, ADA Boost suffered I assume because I lowered the estimators to 5 from 50 as I didn't see a discrepancy in the score line to my naked eye. Apparently there was one and using 10% of the Estimators only decreased it by a little bit. ANN benefited from doubling the learning rate which would decrease the training time as the step updates would be larger and also prevents it from introducing variance to the model. As noted above the Linear kernel is not good for this data and didn't increase its score beyond that which it used to overcome a bit of bias. RBF having no gains surprised me. It may be that I set C to high which overemphasised the training set and led to the same results in the validation and test sets.



	Default	Tuned	Gain
DT	63.73	67.46	3.73
KNN	69.83	67.12	-2.71
ADA	72.2	71.53	-0.68
ANN	74.24	74.58	0.34
RBF	70.51	70.51	0.0
Linear	72.88	72.88	0.0

Figure 13: CU Final Results

Wine Quality: DT, KNN, ADA

Now that we are in the 2nd data-set I want to start addressing the question of instance/based Learning Curves. The datasets have differences which will play out in the upcoming graphs and will help highlight strength and weaknesses of the algorithms.

If we contrast/compare Figure 1 and Figure 14 we can see that they are similar,

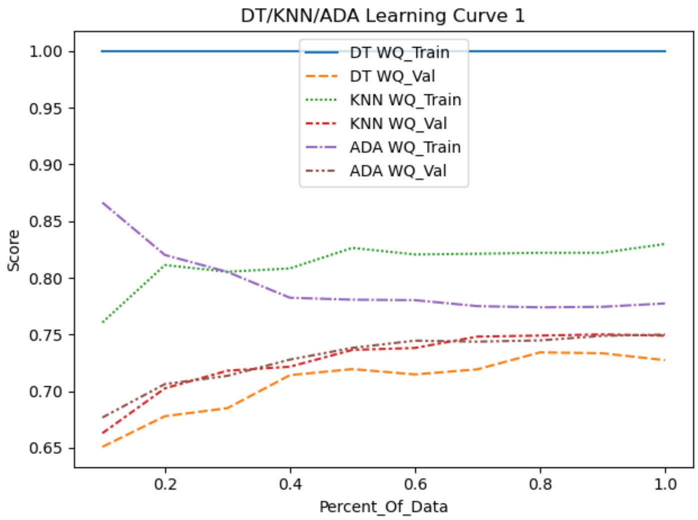


Figure 14: DT-KNN-ADA LC1

which leans us towards a percentage based approach. However, note that Wine Qualities lines in general are smoother. This can best be seen in KNN train, validation and Decision Tree validation. The Contraceptive Used data-set has less samples so 20% (189) of its data may be a bad representation of the true domain, as opposed to Wine Quality which is much bigger and 20% (568) of Wine Quality may be a better representation of the whole data-set. Revisiting the question of ”Are Learning Curves tied to a percentage of the data or is it instance based?”; not to sound like a quantum physicist but it seems like it’s both. They both are very steep until 20% and near the high at 50%. They differ though that the smaller data-set Contraceptive Used has higher highs and lower lows. This can be more easily seen from the steep decline of Contraceptive Used vs the shallow decline of Wine Quality between 20-40%. In this regard it is instance based as a larger 10% is less prone to change than a smaller 10%; i.e. given 2 uniform distributions 10 and 1000 receiving an extra 10% can alter the smaller than the larger more significantly.

Additionally, Wine Quality was over-fit more as there are 1) more features to split on which allows for a greater degree of complexity and 2) the input values themselves are more varied which allows the Tree to differentiate between instances more easily. Wine Quality suffers from a greater Curse of Dimensionality than Contraceptive Used despite having more instances. The Wine Quality data has 5 columns with 3 significant digits (1000 discrete options) and 6 with 2 significant digits (100 options); Contraceptive Used only has 1 column with 2 significant digits and the rest are under 10. This means that there are many more value options for an instance in Wine Quality than in Contraceptive Used and that either 1) no instances are exactly the same allowing the Tree to create a leaf for each instance; or, 2) if 2 instances are the same they have the same classification.

Re-plotting our Neighbor VS Leaf Size shows us that whereas in Contraceptive Used the Decision Tree beat KNN, here the results are the opposite. Why is KNN better in Wine Quality than in Contraceptive Used? This can be attributed to the differences in Features between Contraceptive Used and Wine Quality. In Contraceptive Used there are many Categorical features. They have a logical ordering but the scale is assumed. For example, Wife’s education range 1-4, Husband’s education 1-4, Husband’s occupation 1-4, Standard-of-living index 1-4. These Attributes have inherent assumptions about the data such as an Undergraduate degree has double the

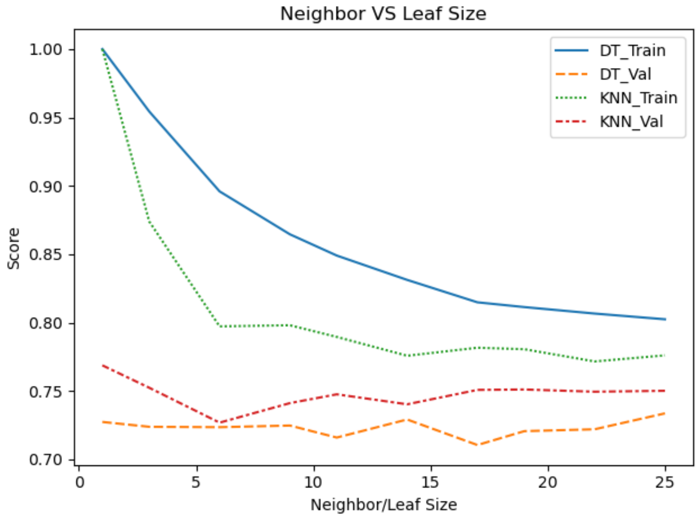


Figure 15: WQ Neighbor VS Leaf Size

”distance” than a High School degree and .5 of a Masters. The ordering is understandable but the scale may or may not be true. In contrast, Wine Quality has pure numerical Attributes which makes it better for KNN (8% alcohol is double 4%). Additionally, why is the Decision Tree curve is strangely flat here as opposed to Figure 2 where the Decision Tree has a 10% gain. (For the final hyperparameter I chose `n_neighbors = 25` here too as that gives the best score.)

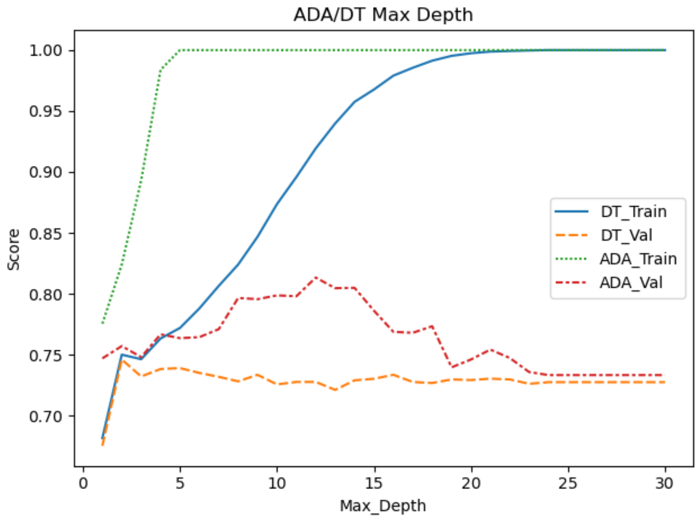


Figure 16: WQ Max Depth

To answer that question we must look at the `max_depth()` parameter chart. Here too we see the bias-variance pattern but Decision Tree validation line remains stable instead of losing accuracy as in Figure 3. variance still increases but it would seem that even if the training data is over-fitting itself doesn’t mean we lose accuracy. The reason over here is that once we hit the `max_depth()` of 2 with 4 leaves all the validation data is taken care of. The Decision Tree is just making extra branches and leaves for its own mis-classified instances while being able to maintain the correctly classified data that the validation set uses. Inversely to Figure 3, ADA Boost gains here as it too is able to remain stable while correcting via a new Estimator the mis-classified data. Again, this echoes our quote from the authors of the Wine Quality data that not all features may be relevant. Once the initial splits are created and leaves classified the Tree just seems to be expanding the mis-classified data to over-fit and create high variance in the train set. As Harris, 2017 (2017) describes a Decision Tree search ”The search usually stops when the sample becomes too class pure”. So whatever is being split is still able to maintain the same class purity that corresponds to the validation set and retaining its score. Amazingly, though ADA Boost is able to also hold and gain in score; presumably for the same reason. It can create Trees that at the first couple Depths have the same splits and reinforce what it learnt and in later depths is where it starts focusing on the problems it got wrong. Restraining its Depth would force it to use those first couple of layers for the harder problems, but they still may be mixed up with other data and inseparable. Only later can we get this effect. Yet once we expand the `max_depth` past a certain point it falls to overfitting and high variance and returns to the base tree here. So for my ADA Boost model I left it at `n_estimators = 50` and `max_depth = 12` and the Decision Tree `max_depth = 3`.

Learning Curves DT/KNN/ADA Round 2

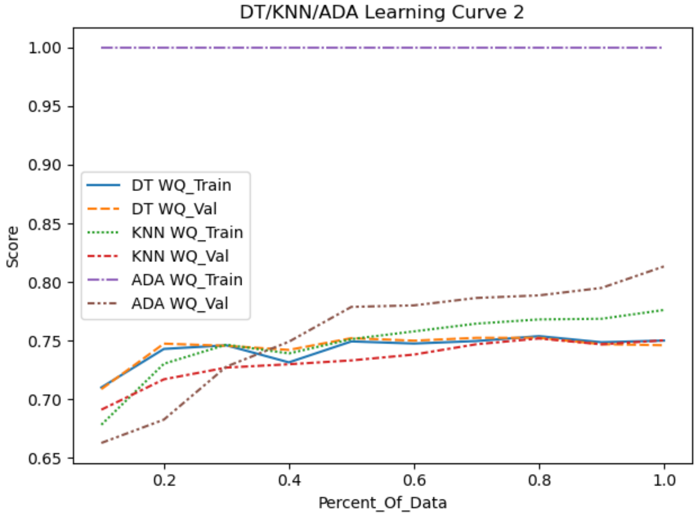


Figure 17: WQ DT-KNN-ADA LC2

After tuning my models I plotted another learning curve and had some interesting results. The variance for ADA Boost is expected as we saw it happen in Figure 16 but the gain in score is fantastic. I believe that this is a good model. The reason why it's getting a good score on the training set is just because those cases are easy for the model but this doesn't end up hurting our validation score. We were successful in reducing the variance for all the other models but the scores look to be about the same. This reduction in variance helps our confidence that the model will generalize well to the held out test set. However, despite the variance looking to be high in ADA Boost I believe this model should be kept as it's good for the train score and good for the validation score by a significant amount which implies that it will generalize well to the test set.

Wine Quality: SVM

The Learning Curve produced looks pretty similar to Contraceptive Used (Figure 7 except for 1 significant change – the scores are lower here. This can be attributed to the lack of relevant Features as seen in the smaller Decision Tree. Having less relevant Features makes it hard for ANN and SVM models to create a Hypotheses. This is compounded by the fact that these features are still present in our data creating a Domain that inaccurately presents itself to these models as being relevant. Additionally, ANNs and SVMs are well suited for Categorical data with a logical ordering as their weighting systems can implicitly infer the different weights and adjust them accordingly. 1 other interesting point is the greater variance here VS Contraceptive Used. Now that we see a the effect of Categorical data on KNN, we can examine the data again and see if there is another data reason why our Linear model performed poorly VS RBF in Contraceptive Used and here. So we had transformed the data using Quantile_Transformer() because the orders of magnitude and scale was different across features as described previously; however, this distorts the linear correlation (SKLearn, n.d.-b). Using Standard_Scalar() on the data is bad for LinearSVM because it would give the outliers to much weight. So despite being good for Categorical features the underlying data values prevent us from using its full capacity. To paraphrase the SKLearn, n.d.-c docs ”RBF... or Linear models assume centered features around zero and same order variance. Otherwise, it might dominate... and make the estimator unable to learn...as expected”. So in our data we have a catch-22 for Linear kernels. There will either be outliers or a distortion of the linear correlation. RBF is fine because it doesn't need linear correlations to be effective and re-

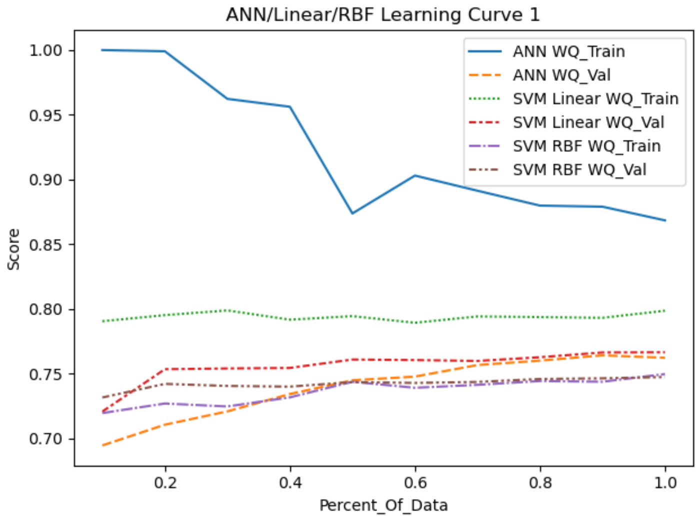


Figure 18: WQ ANN-Linear-RBF LC1

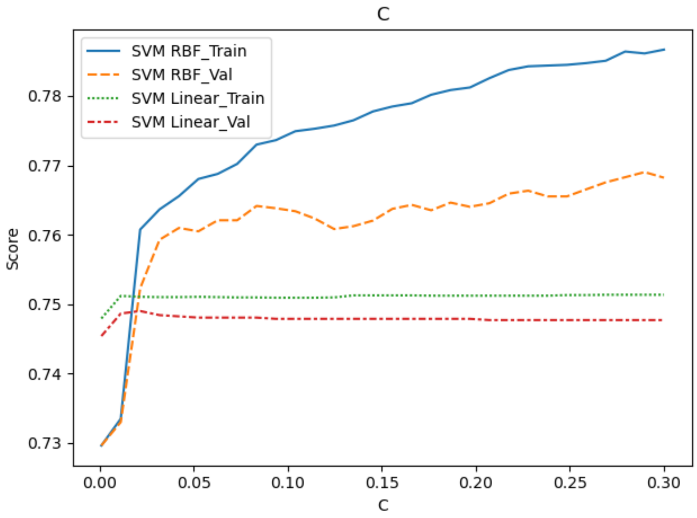


Figure 19: WQ C Regularization

quires the `Quantile_Transformer()` to prevent outliers from taking over.

Gamma here has an interesting behaviour This graph doesn't exhibit the classic pattern of splitting of a low score for both train-validation followed by an increase in both and then validation going down with training going up. Examining the hyperparameters of the SVM we will see C is set to .1 and Gamma starts diverge right around there. A high C focuses on getting the training examples right whereas Gamma is a weight of how much to affect closer instances. It seems that as Gamma gets larger the instances affect each other more leading to a general score for the data-set. This would be equivalent to setting a high number of neighbors on uniform, the higher you go the closer train and validation get.

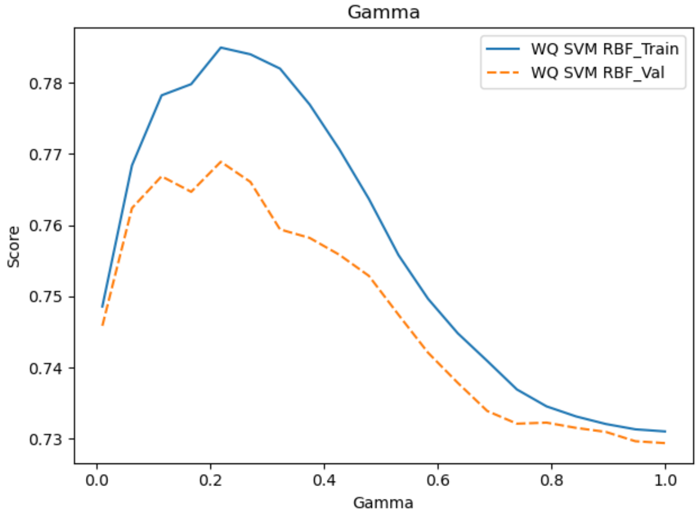
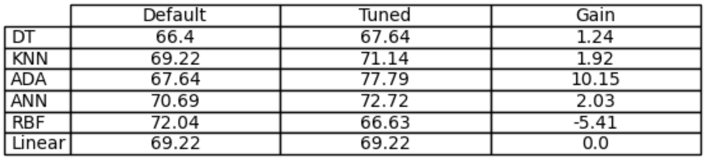


Figure 20: WQ Gamma

Final Results: Wine Quality



	Default	Tuned	Gain
DT	66.4	67.64	1.24
KNN	69.22	71.14	1.92
ADA	67.64	77.79	10.15
ANN	70.69	72.72	2.03
RBF	72.04	66.63	-5.41
Linear	69.22	69.22	0.0

Figure 21: WQ Final Results

The biggest gain model here is from ADA Boost. I believe this is because of the data that was able to be separated by the underlying Decision Tree without losing accuracy allowed ADA Boost to also get some extra instances correct. KNN did well here too and as we mentioned the reason why Wine Quality and KNN work better together is because the features are purely numerical and distance is exactly as stated. In Contraceptives used as we mentioned the distance implied by the logical ordering may or may not be true and it seems to have been not true as we see from the final results. Note that KNNs neighbors both did well at 25 on the validation Curve; yet, we see that it doesn't mean our held out Test set will do well. This is especially true of Categorical features. We made more modest gains here by the Decision Tree. One factor that may play a role in this is that Wine Quality suffers from a greater Curse of Dimensionality than Contraceptives used. In ANN our Learning Rate was .02 which is 20 times the default of .001. This indicates that the model was previously overfitting to the training data and a bigger step size allowed us to overcome that variance. My RBF kernel was a complete shock to me that it did so poorly with all that tuning. I assume that C to Gamma ratio was the cause and that there were too many support vectors due to a low C but the low Gamma was very influential and all of them ended up affecting each other. We mentioned an additional reason in the Wine Quality data that applies to Contraceptive Used too. The Linear Kernel is caught in a catch-22 between the difference in orders of magnitude among the features and the scaling distorting the linear correlations leading it to a high bias model.

Additionally, while comparing and contrasting the 2 data-sets we noted that it is useful to compare 2 data-sets as a percentage of the data; with the caveat that the less data you have the more jagged the learning curves might look.

References

Lim, T.-S. (1997). *Uci repository: Contraceptive method choice*. https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice (accessed: 02/18/2021)

Brownlee, J. (6/20/2019). Classification accuracy is not enough: More performance measures you can use. https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/. (Accessed: 2/10/21)

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (5/19/2016). A practical guide to support vector classification, 5. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf. (Accessed: 2/10/21)

SKLearn. (n.d.-a). *Svm-kernels*. https://scikit-learn.org/stable/modules/svm.html#svm-kernels (accessed: 02/18/2021)

Harris, E. J. (2017). *Information gain versus gain ratio: A study of split method biases*. https://www.mitre.org/sites/default/files/pdf/harris_biases.pdf (accessed: 02/18/2021)

SKLearn. (n.d.-b). *Preprocessing data*. https://scikit-learn.org/stable/modules/preprocessing.html#non-linear-transformation (accessed: 02/18/2021)

SKLearn. (n.d.-c). *Preprocessing data*. https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling (accessed: 02/18/2021)