Andy Cheng

CS 7641

Assignment 1: Supervised Learning

2/11/2024

1. Introduction

Five different supervised learning algorithms were applied to two different data sets to analyze the impacts and interactions that each method had on the data. The five algorithms are Decision Tree (DT), Boosted Decision Tree (BDT), Neural Network (NN), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). The first data set is on apple quality with features such as size, weight, sweetness, crunchiness, juiciness, ripeness, and acidity with an evenly split overall classification of Good/Bad (Elgiriyewithana, 2024). There are 4000 total data points in the apple data set. This data set is interesting as it represents a relatively limited feature space with seven features that should identify which learning methods and corresponding hyperparameters perform better accordingly. The second data set is on aircraft jet engines with the classification criteria of whether the engine needs to be serviced by maintenance or not (Viswanath, 2024). The feature space includes various engine settings such as throttle levels, temperatures, pressures, and more for 24 unique features. These features also consist of measurements not directly related to needing maintenance such as metal fatigue levels, but secondary measurements that are normally recorded during flight. A secondary goal with the engine data set was seeing if the algorithms could still predict if maintenance is required with only these loosely related measurements. There were 552 total engine servicing data points after preprocessing to achieve a 50/50 split in classification between engine OK/ maintenance needed. This engine data provides an interesting contrast with the apple data as it has over three times more features, but much less overall data available for the models to train on. The goal was to see how the methods handle this richer feature space, but lack of data in comparison to the apple set. All performance was evaluated to determine general trends with the models as well as evaluate bias, variance, and fitting.

2. Methodology

The general methodology was an iterative process where results were generated, and model parameters were tweaked as a result. All analysis was conducted using MATLAB 2022a and the Machine Learning Toolbox (The MathWorks Inc., 2022). Preprocessing was first done to achieve a 50/50 OK/maintenance needed split for the engine data and then splitting both data sets into training/testing sets. The features consisted of all float values for both sets and were normalized to achieve zero mean and standard deviation of 1 for each feature. 85% of the processed data was separated to be the training set with the remaining 15% saved to be the final test set to evaluate on after tuning. Initial learning curves for each method were then generated using default MATLAB values for hyperparameters to evaluate the size of the training sets needed to learn effectively based on overfitting. These curves display both training accuracy and 5-fold cross validation accuracy. Finding the ideal hyperparameters for each algorithm was then conducted by first tuning the parameter that had the greatest predicted impact using validation curves then using this parameter to tune the second hyperparameter using validation curves. The ideal pair of hyperparameters for each algorithm was then saved off and used to generate final learning curves. Each tuned model was then evaluated against the test set to determine final performance.

The hyperparameters tested are shown in Table 1:

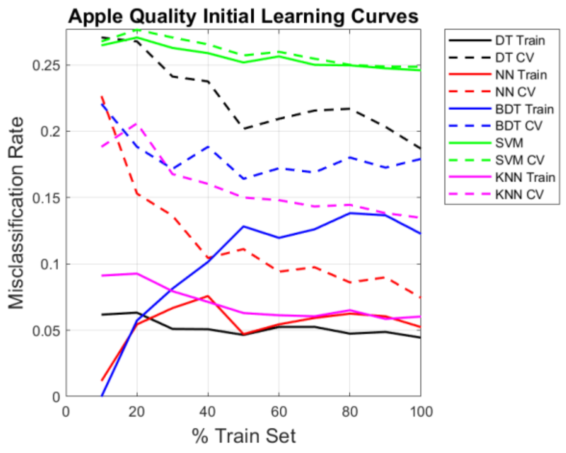
Table 1 - Tuned Hyperparameters for each Algorithm

DT	BDT	NN	SVM	KNN
Pruning On/Off	Max Num Splits	Layer Sizing	Regularization C	K
Minimum Leaf Size	# of Weak Learners	Activation Function	Kernel Type	Distance Function

The DT hyperparameters were chosen to balance the issue of overfitting with deep and complex trees with the need to fit to a dataset with a more complex feature space but limited data points such as the engine set. Seeing how pruning impacted the generalization of the model was key as well as how minimum leaf size impacted the overall bias with model simplification. The BDT hyperparameters focused on how many trees there were with # of weak leaners and how deep each tree can grow with max number of splits. These represent two degrees of freedom for the BDT which allows a wide exploration of model complexity impact to find a balanced level of performance, bias, and generalization for the two vastly different datasets. The NN hyperparameters focused on the fully connected layer count and width as well as the activation function for these layers. The goal was to see if the optimal layer complexity for these datasets differed wildly due to the difference in data outlined previously. The ReLU and Tanh activation functions were chosen for comparison as they are both non-linear but exhibit different outputs for model expressiveness which was interesting to test against the datasets. The SVM hyperparameters were chosen to control overfitting to the limited dataset size with the regularization C having to balance between maximizing the margin and minimizing classification error. The Linear and RBF kernel types were chosen to compare. The RBF for its ability to capture complex, non-linear decision boundaries which may have a positive impact for the engine set with a larger feature space. The Linear for its efficiency in comparison which may reduce overfitting in comparison to the RBF for the limited data set size. The KNN hyperparameters were the # of neighbors K and comparing the Minkowski distance function with the Euclidean distance function. K is the biggest driver for this algorithm's performance with it directly impacting data fitting, model complexity, and also whether the model focuses on local vs global patterns. The distance functions allow for differing neighbor selection which results in differing feature space correlations to explore with the underlying comparison between feature space complexity in the datasets.

3. Results

a. Initial Learning Curves



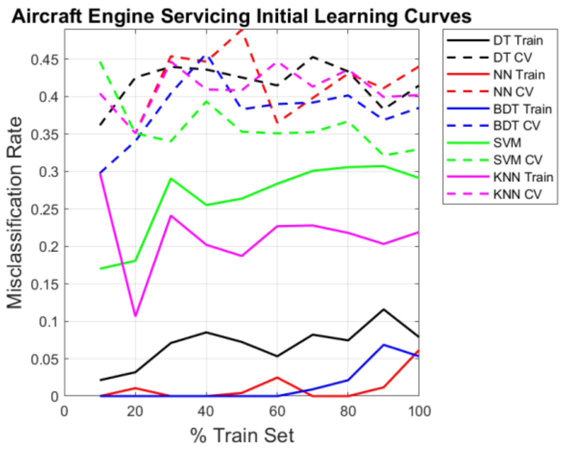
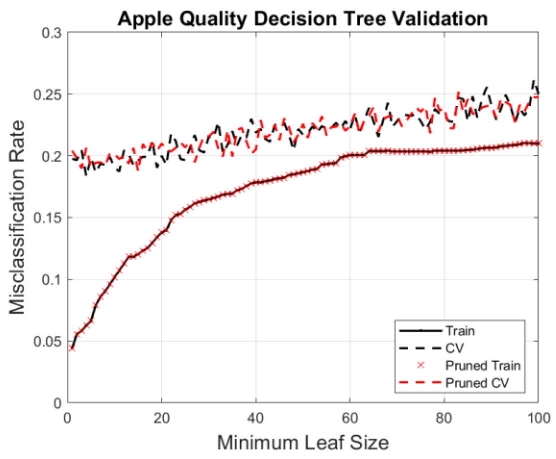


Figure 1 - Initial Learning Curves

Figure 1 shows initial learning curves generated pre-tuning to see what model performance would be with default hyperparameters based on the amount of training data used. The apple dataset curves shows the validation and train curves slowly converging for most algorithms with only SVM reaching a point where misclassification error is equal. This indicates that introducing more data would continue to help as there is no inflection point for cross-validation performance. The training set was kept at 100% for the apple data as a result. The engine dataset curves show a different story with the algorithms having a hard time properly classifying the data. There is large variance between the train and validation curves, except for SVM which had the lowest misclassification rates and improved with more data. The training set was also kept at 100% for the engine data as the number of data points was already very limited and tuning would be able to help performance.

b. Decision Tree



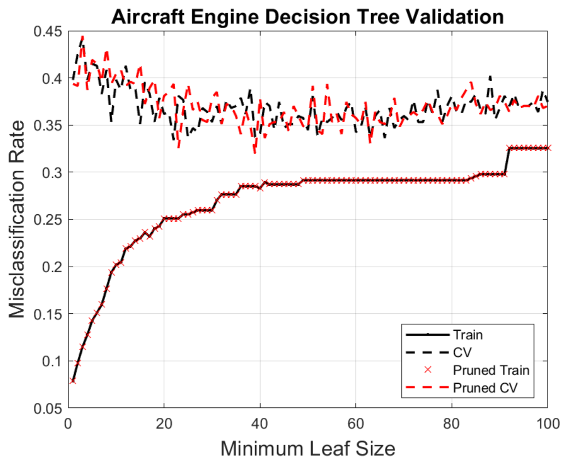


Figure 2 - Decision Tree Hyperparameter Tuning

Figure 2 shows the tuning conducted for DTs where both non-pruned in black and pruned in red are shown against minimum leaf size. Here there is another difference between the data sets shown with the apple dataset curves showing better performance with a lower optimal minimum leaf size of 4 while the engine dataset curves are more parabolic with an optimal minimum leaf size of 39. With the small minimum leaf size, the variance is very large between train and validation performance, but still yields the best performance for the apple dataset. The engine dataset has a more reasonable variance at the optimal minimum leaf size. Both datasets do not show a noticeable difference in performance from pruning which indicates that there is little overfitting occurring and that the decision trees are already low complexity with little depth. The apple data shows validation performance degradation with larger minimum leaf size as the model is forced to make simple splits, increasing bias. The model becomes too simple and gets worse at finding underlying, complex patterns in the apple feature set. The opposite is true for the engine data as the noisy data containing some irrelevant features necessitates a more generalizable model that does not overfit to feature patterns not related to the service required classification.

c. Boosted Decision Tree

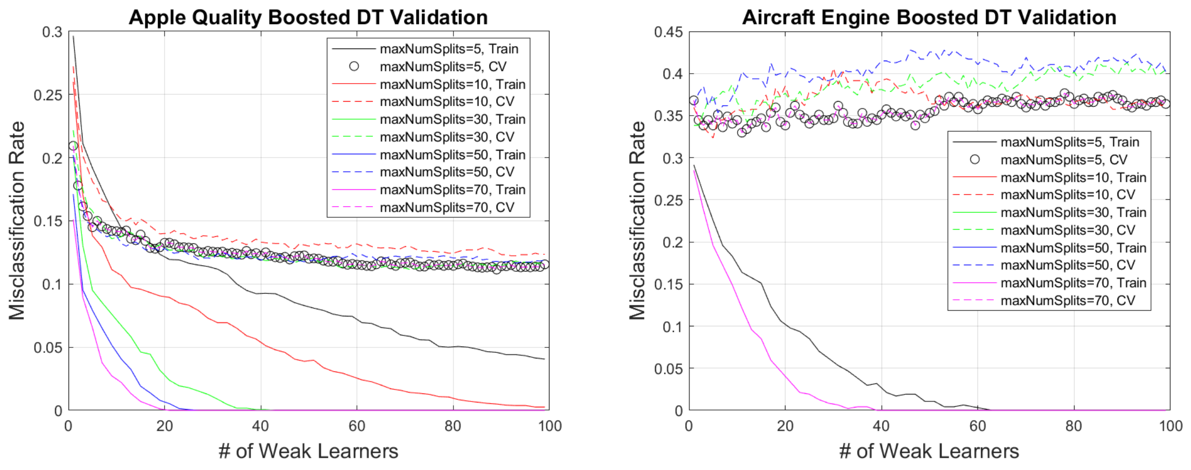


Figure 3 - Boosted Decision Tree Hyperparameter Tuning

Boosted decision trees compared number of weak learners and maximum number of tree splits. The datasets again show interesting behavior in Figure 3 with the apple dataset responding well to tuning while the engine dataset showing little improvement to the validation performance with increasing amounts of weak learners. The apple curves show that performance keeps on increasing with more weak learners, but there reaches a point where model complexity is not worth the small increase in performance. This point is around the 20 weak learner mark with a maximum of five splits where variance is very small as the model is not overfitting to the training data. This performance trend makes sense with more weak learners resulting in greater capacity to learn from the training data and generalize to unseen data, reducing bias. The overfitting that is occurring with large numbers of weak learners is intriguing too and seeing if different boosting algorithms with better regularization techniques would have been good to experiment further with in future work. The engine data has extremely high variance and only really seems to benefit from having a limited number of splits with the lowest tested of five having the best performance. The engine data being circumstantially related to the classification problem does not contain enough information for more weak learners to have a noticeable impact on performance. Both datasets prefer lower number of splits, but the impact is much more pronounced for the engine dataset. This performance stems from having more shallow decision trees that produce smoother decision boundaries that are less likely to capture noise and outliers in the training data, resulting in a more robust model. The noisiness of the engine dataset with circumstantially related features benefits from this simplicity.

d. Neural Network

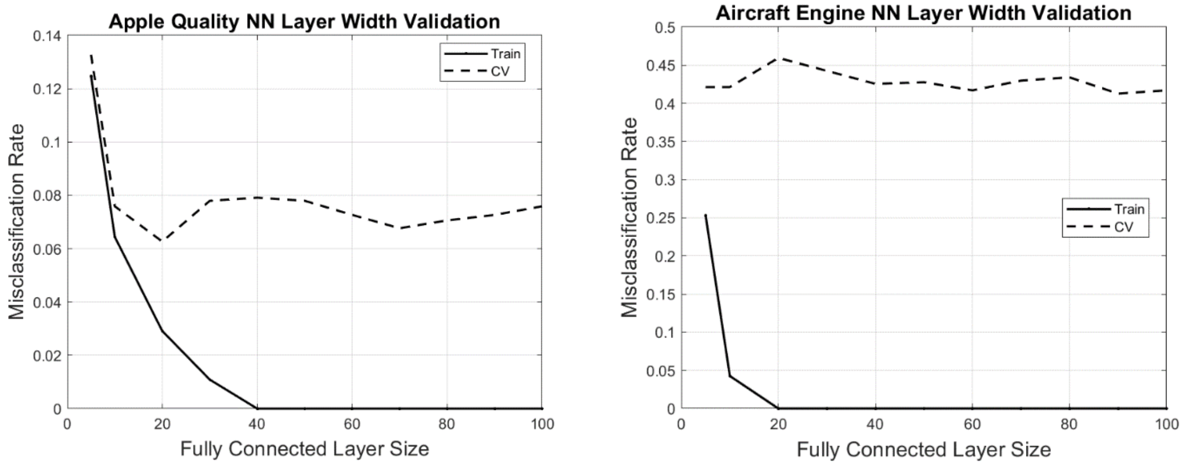


Figure 4 - Neural Network Layer Width Tuning

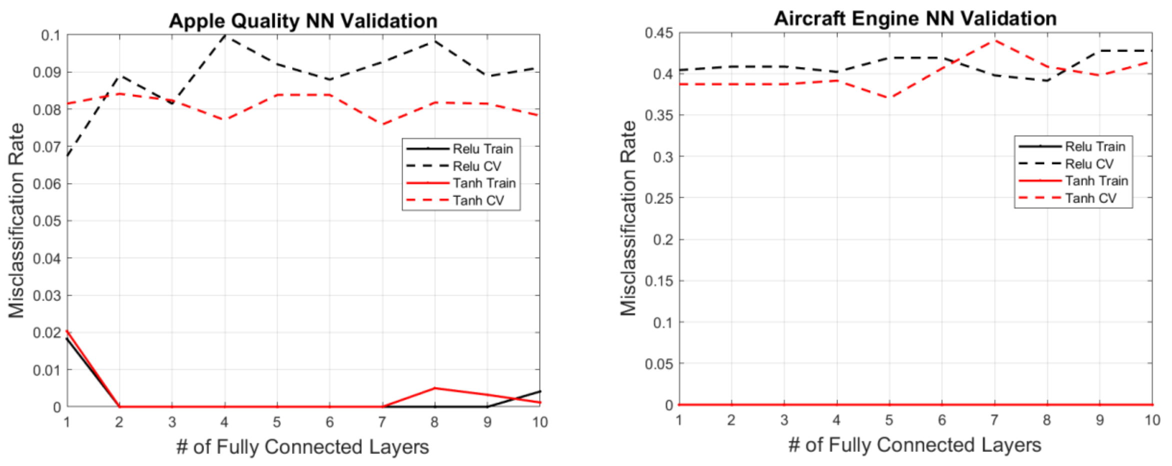


Figure 5 - Neural Network Hyperparameter Tuning

Neural network optimization involved a two-step process with finding the optimal full connected layer width first shown in Figure 4. For the apple dataset, this point was at 20 neurons which resulted in low variance. The engine dataset did not see much improvement with more neurons but had an optimal point of 90. There is large variance showing overfitting to the dataset and the neural network could not generalize to the validation data, no matter how many neurons there were. This trend points to the model becoming too complex relative to the amount of training data available with the small size of the engine train set. There are also not enough features in both datasets to benefit from the increase of dimensionality with more neurons, so the models would not be able to pull any more meaningful relationships from the data. Figure 5 show the next step in tuning where the layer width determined in Figure 4 was used to find the optimal number of fully connected layers. ReLU and Tanh activation functions were also tested. The apple dataset had an optimal point of one fully connected layer while using the ReLU function. The engine dataset had an optimal point of five fully connected layers while using the Tanh function. This behavior of performance staying consistent with more layers is explained similarly to increasing the number of neurons per layer with there not being enough features and data points to benefit from the increased model complexity. The ReLU function performs better for the apple dataset at one layer as this function generally performs better on shallow networks. The Tanh activation function likely performs better for the engine dataset due to the smooth gradient it has compared to ReLU which would make the network more stable when dealing with the noisy, circumstantially related engine data.

e. Support Vector Machine

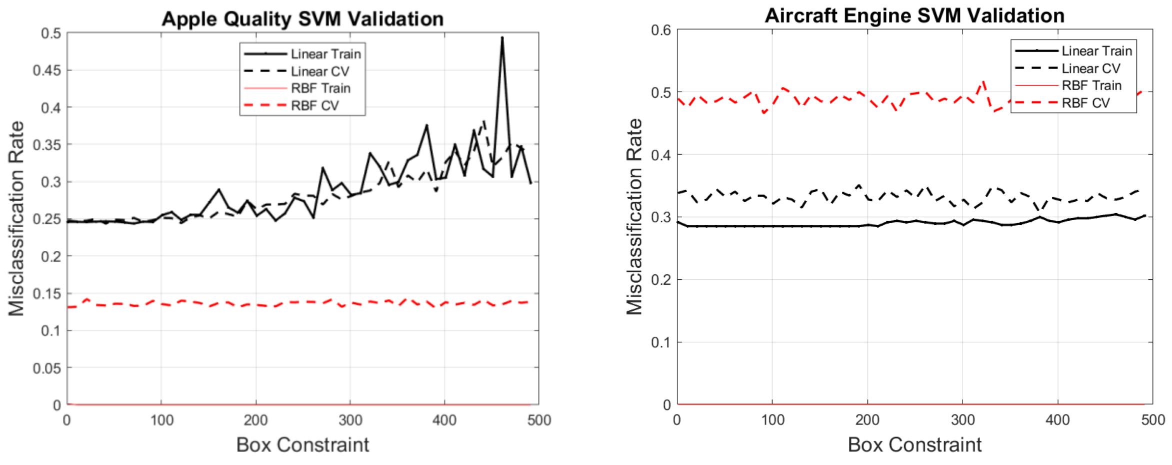


Figure 6 - Support Vector Machine Hyperparameter Tuning

The SVM curves in Figure 6 show more interesting behavior with the RBF kernel performing much better for the apple data while the Linear kernel performing much better for the engine data. The box constraint, or Regularization C, only really impacts the linear kernel performance for the apple dataset at higher C values. It was expected that the RBF would perform better for the more feature dense engine data set, but the opposite is true with the simple kernel having a much lower misclassification rate. This result is likely due to avoiding overfitting using the Linear kernel with how small the engine train set size is. The variance and bias are small for the linear curve, while the RBF is clearly overfitting with zero train set error. The Linear model is more generalizable for the engine validation data which shows in the performance. The apple curves show another story with the RBF model still overfitting but resulting in better performance for the validation data. RBF has non-linearity to model complex decision boundaries which likely benefited the apple data which was features that were directly related to its classification boundary compared to the more difficult feature space for the engine data. As for why only the Linear kernel for the apple data was impacted by higher C values, there may be some underlying misclassification or noise in the data that the model is trying to fit to closely for with larger Regularization C values.

f. K-Nearest Neighbor

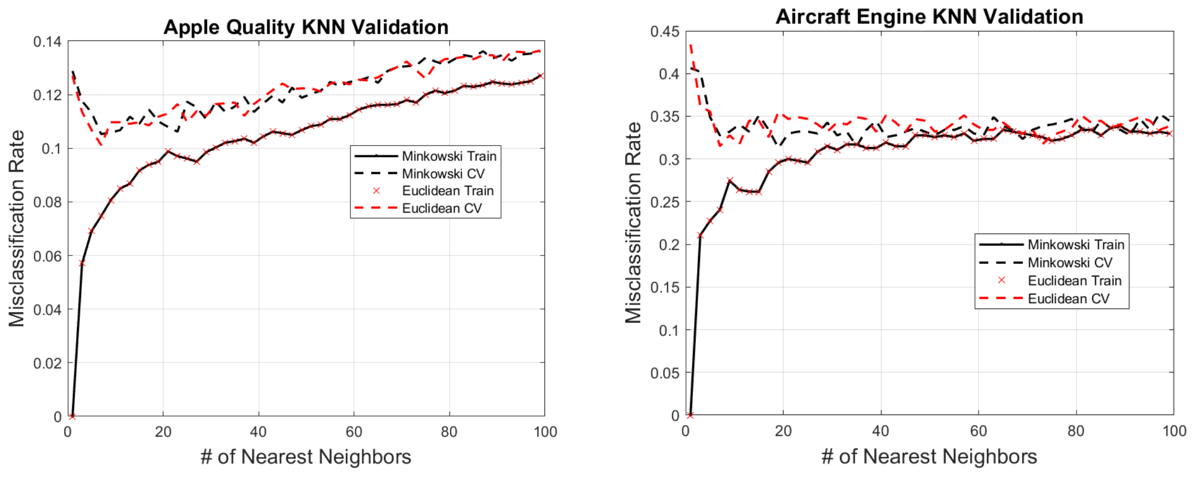
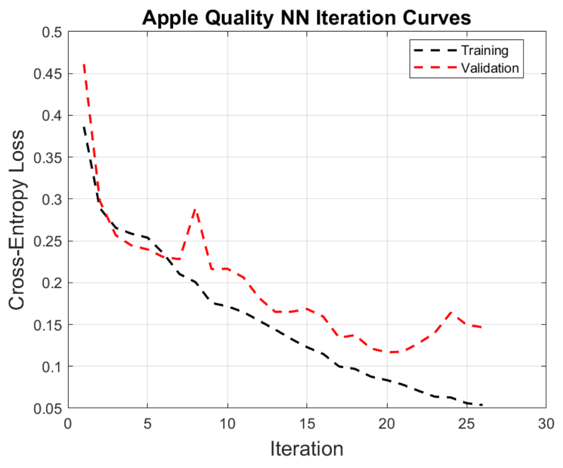


Figure 7 - K-Nearest Neighbors Hyperparameter Tuning

The KNN curves in Figure 7 show similar behavior between the sets with the training and validation curves converging at a low K and having a low variance for increasing K. The apple data has an optimal K of seven using Euclidean distance. Performance begins to regress with larger K values beyond this optimal point. The engine data has an optimal point of 19 using Minkowski distance and performance stays consistent with larger K values. The apple data begins to regress in performance with increasing bias as K gets large due to the model generalizing too much and failing to capture the finer details or complexities of the underlying data distribution. The decision boundary becomes too smooth and cannot resolve the local patterns as well. The engine data performance becomes constant with larger K due to the dataset containing noisy and irrelevant features. The KNN model seems robust to these characteristics after increasing K beyond nine where it captures a balance of local and global structures with its decision boundary. For both datasets, the distance function did not vary results much. This observation is likely due to the datasets being well-structured and scaled correctly through normalization which led to the distance functions producing very similar neighbor rankings.

4. Comparison

a. Final Learning Curves



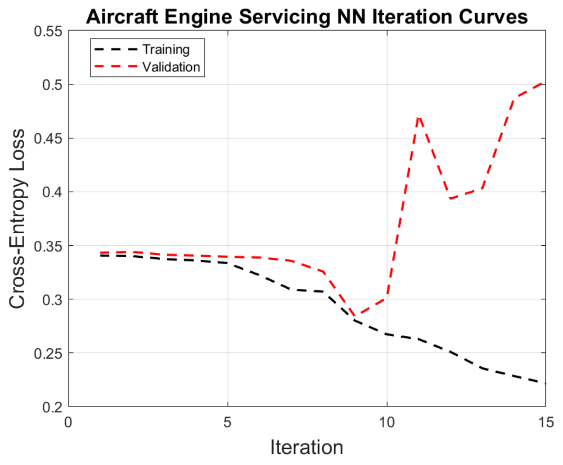
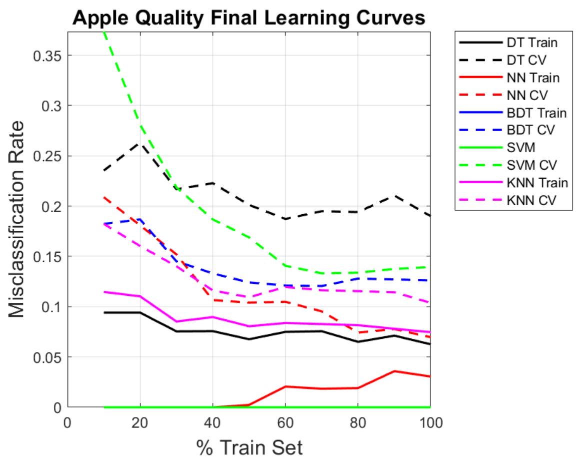


Figure 8 - Final Neural Network Iteration Learning Curves

The final neural network iteration curves in Figure 8 show clear cutoffs where the network begins to overfit and perform worse on the validation data. For the apple dataset, this is at 20 iterations while this cutoff is at 9 iterations. The performance shown in the learning curves is good in general with no large fluctuations in performance with smooth lines until overfitting begins to occur. The neural network training stops early in both cases due to this increase in variance.



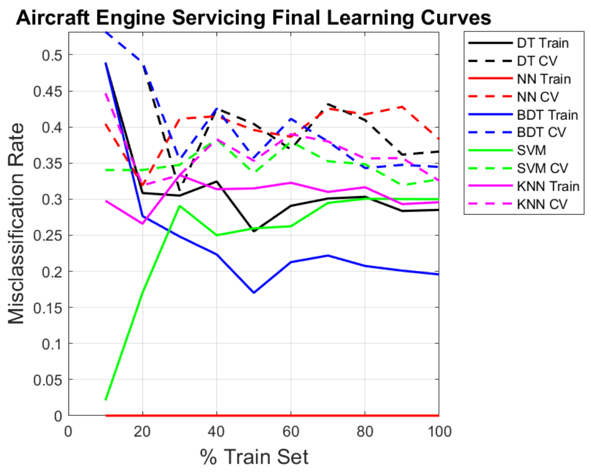


Figure 9 - Final Learning Curves

The final learning curves in Figure 9 for all the models show interesting trends in terms of which algorithm produced the best misclassification error and how the bias-variance tradeoff is presented. The apple data set had the neural network perform the best with low bias and variance at the full train set size. The decision tree performed the worse with larger variance, but a misclassification rate below 0.2. The SVM and BDT models were overfit after tuning with 0% train misclassification rate, but this is reasonable as tuning was driven completely by minimizing validation set error. All algorithms performed well with the apple data to the features directly relating to the classification and having clear classification boundaries between good and bad quality apples. The curve trends also indicate that more training data would be beneficial for better accuracy. The neural network performed the best, especially compared to the decision tree, due to its ability to recognize complex non-linear patterns. The DT could not represent the feature complexity well enough with only its hierarchical splits based on feature thresholds. The NN and KNN had the lowest variance likely due to their flexibility to adapt well to different datasets when tuned well. The engine data set had the SVM and KNN models strike the best balance between bias and variance. Both models had near the best prediction accuracy and low variance. There is still the case where validation performance does not improve with a larger training set beyond 30% which indicates that the underlying issues with the data were not alleviated with hyperparameter tuning. The NN overfits completely to training data which make sense due to the 90 neurons per layer chosen to minimize validation error during tuning. All other algorithms experienced high bias due to the difficulty in classifying the engine servicing threshold from less relevant features.

b. Final Results for Test Set

Table 2 - Test Set Misclassification Results

Dataset	Decision Tree	Neural Net	Boosted DT	SVM	KNN
Apple Quality (Misclassification Rate)	0.193	0.07	0.108	0.11	0.093
Apple Quality (Wall Train Time (s))	0.018	1.12	0.368	0.216	0.028
Aircraft Engine Servicing (Misclassification Rate)	0.366	0.341	0.427	0.341	0.329
Aircraft Engine Servicing (Wall Train Time (s))	0.009	0.361	0.043	1.91	0.010

Table 2 shows the final performance for each tuned algorithm when tested against the partitioned testing set that the models did not train on. For the apple quality data, the NN and KNN models produced the best performance with misclassification less than 0.1. However, the NN had the longest wall time for training with the NN taking 40 times longer to train than the KNN model. As the apple dataset is small in complexity and size, the overall NN train time is still reasonable at 1.12 seconds. The improvement in performance is worth the trade off in wall time here. All misclassification rates for the apple set are good with no model having difficulty in finding feature patterns or classification boundaries. The aircraft engine servicing data shows a different story with the algorithms having a difficult time decerning classification due to the dataset quality issues. The KNN model performed the best which was unexpected at the start of the experiment due to the larger feature space present in the engine data. It appears that creating a model that did not overfit and was very generalizable to new data was the key to having the best performance. The KNN model was kept simple with 9 neighbors and as a result, adapted the best to the noisy engine data with semi-irrelevant features. As for wall train time, the KNN had the second lowest at 0.01 seconds which makes it the best choice of algorithm for the engine data set. The engine SVM model had the longest training time at near two seconds which was much greater than the apple SVM model. Both models used similar regularization values of ~400 so the difference must stem from the use of a linear kernel for the engine model and an RBF kernel for the apple data. The NN train time was the other interesting case where the layer complexity was much higher for the engine model, but the train time was less. This observation indicates that these neurons were most likely not being activated in the needlessly complex engine model and that train set size plays a significant role in training time.

5. Conclusion

The apple quality and aircraft engine servicing datasets did provide an interesting contrast in feature space complexity, train set size, feature relevance, and noisy data that was used to compare these supervised learning algorithms. The apple dataset served well as a control with relevant features, small number of features, and a medium amount of data points that all algorithms performed well on. The engine data set was the opposite with many somewhat relevant features and a small amount of data points which caused difficulty for the algorithms to train. This contrast allowed a look into some of the strengths and shortcomings of the different algorithms. The NN performed best for the apple data due to its ability to readily identify non-linear patterns in features which benefited from the apple features having direct links to the classification. The KNN model performed the best for the engine data due to its simplicity which allowed it to be more generalizable when classifying noisy engine data that it has not seen before. When models overfit to the engine training data, the validation performance degraded which meant that the hyperparameters had to be carefully tuned to strike the balance between bias and variance. The biggest challenge was striking this balance of tuning levels and is where future work on this project should be directed. In terms of the secondary goal, the algorithms were able to predict if maintenance is required to a 77% testing accuracy even with the noisy data with irrelevant features. Some potential research areas to explore with this data include pruning with different criterion, boosting with different regularization techniques to reduce overfitting, testing more complex NN layer configurations, and conducting a PCA to determine relevant features for the engine dataset.

6. References

Elgiriyewithana, N. (2024, January 11). *Apple Quality*. Kaggle. https://www.kaggle.com/datasets/nelgiriyewithana/apple-quality

Viswanath, A. (2024, January 23). *Aircraft Sensor and Engine Performance*. Kaggle. https://www.kaggle.com/datasets/aadharshviswanath/aircraft-sensor-and-engine-performance

The MathWorks Inc. (2022). Statistics and Machine Learning Toolbox (R2022a), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com