Assignment 1: Supervised Learning

Steven Nord
snord3@gatech.edu

Abstract—Supervised Learning is a subcategory of machine learning where labeled data is used to develop a model or function that can be used later to make predictions on similar data that has not yet been seen. However, there are numerous supervised learning algorithms which can be used to derive a model. The choice for which algorithm is best really depends on the characteristics of the dataset itself and underlying desired output. There is not one that is optimal for all situations. This report will present the results of five algorithms' performances against two separate datasets.

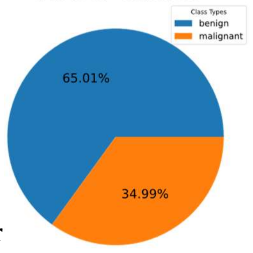
1 Introduction

With so many supervised learning algorithms to choose from, the task of picking only one can be quite the challenge. In this report, analysis from five supervised learning algorithms (K-Nearest Neighbors (KNN), Decision Trees, Boosting with Decision Trees, Artificial Neural Networks (ANN), and Support Vector Machines (SVM)) will be compared to each other. Each algorithm has its own strengths and weaknesses and true behavior can be greatly impacted by the characteristics of the dataset it is trained on. Therefore, each algorithm is analyzed on two characteristically different datasets: Breast Cancer and Steel Plates Faults.

2 Datasets

2.1 Breast Cancer Wisconsin

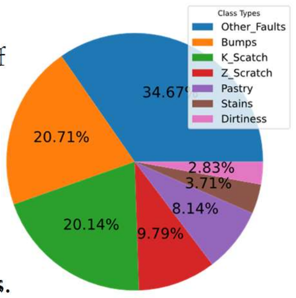
Figure 2.1: Cancer Classes



The Breast Cancer dataset was obtained from the University of Wisconsin Hospital by Dr. William H. Wolberg. It aimed to classify a tumor based on various physical attributes. The dataset contained 699 instances with nine features and one binary target class (benign or malignant). Sixteen instances were dropped due to missing data. **Figure 2.1** shows the breakdown of the target class. The nine features used in learning the target class were all discrete values ranging from 1-10.

2.2 Steel Plates Faults

Figure 2.2: Steel Classes



The Steel Plates dataset was gathered by Semeion, Research Center of Sciences of Communication. It attempts to properly classify the type of surface faults for stainless steel plates. The dataset contains 1,941 samples with 27 features used to categorize each sample into one of seven different fault classes: K_Scatch, Bumps, Dirtiness, Z_Scratch, Pastry, Stains, and Other. **Figure 2.2** shows the breakdown of the target class. The 27 features used for learning the target class were all continuous values with varying ranges.

3 Experiment Setup

The only preprocessing of the data was feature-scaling since algorithms, such as KNN and SVM, rely on measuring distance from data points. Without feature scaling, small discrepancies in large range features would carry more weight than other small range features. This does not impact the Cancer dataset as much since all features range from 1-10, but scaling was still used for consistency. Dimension reduction was not performed as the difference in the number of features for each dataset (9 for Cancer and 27 for Steel) would help illustrate the behavior for the various algorithms.

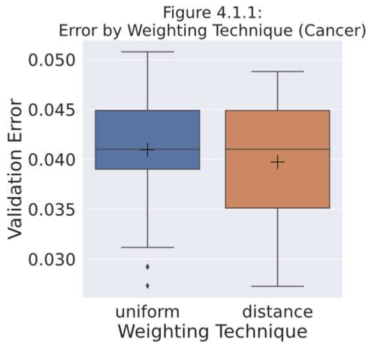
For each algorithm, data was split into two sets. The training set consisted of 75% of the data and the test set consisted of the remaining 25% with stratify sampling so class distributions stay consistent between training and test sets. The algorithms were optimized using five-fold cross-validation with only the training dataset. Several models were trained with Grid-Search technique with various hyperparameters. The aggregate results across all models were utilized for hyperparameter tuning to optimize performance metrics. This allowed algorithm comparison to be on similar scales.

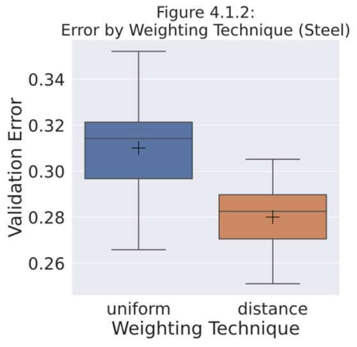
Regarding performance metrics, **accuracy** and **F1-weighted** were the metrics optimized for the Cancer dataset and Steel dataset respectively. The rationale for accuracy was that the Cancer dataset is relatively balanced at 65/35, see **Figure 2.1**. For the Steel dataset, it was between F1-macro (gives equal weight to each class) and F1-weighted (accounts for class imbalance by weighting the score by its percentage of the dataset). F1-weighted was chosen to classify the faults that arise most often since the target classes were highly imbalanced (see **Figure 2.2**).

4 Hyperparameter Tuning

4.1 K-Nearest Neighbors (KNN)

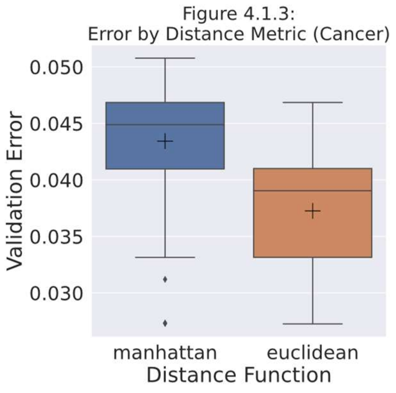
The type of weight technique determines how much of the associated neighbor's prediction vote counted toward the ultimate prediction for the tested data point. The two types of weight considered were uniform (i.e., each neighbor's vote has equal weight) and weighted-distance (i.e., the more similar the features matched the neighbor, the more that neighbor's vote was weighted).

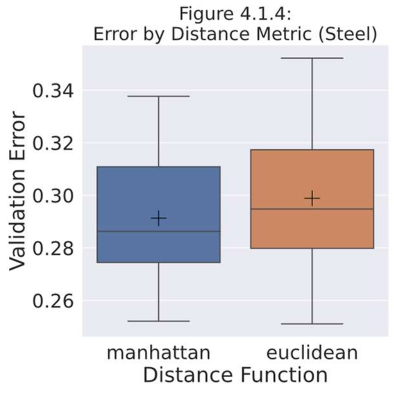




Choosing between these two techniques was important since weighted-distance can lead to added variance in the model as it may prioritize an individual training point that were noisy. On the contrary, uniform can reduce the variance but increase the bias as it may give equal weight to points that are not as similar as the sample to label. **Figures 4.1.1 and 4.1.2** show the weighted-distance has a lower error rate for Cancer and Steel datasets respectively. The "+" in the boxplots indicates the mean across all models with the given weighting technique.

The functions considered for computing the similarity were Euclidean and Manhattan. **Figures 4.1.3 and 4.1.4** show the error rate for each distance function for Cancer and Steel datasets. Interestingly, each dataset preferred differing distance functions to be used as follows: Euclidean for the Cancer data and Manhattan for the Steel data. These results aligned well with previous research that states "The Manhattan distance metric (L1 norm) is consistently more preferable than the Euclidean distance metric (L2 norm) for high dimensional data mining applications" (Aggarwal, 2001). The Cancer was comprised of a much lower dimensional space with only nine features as opposed to 27 features in the Steel dataset.



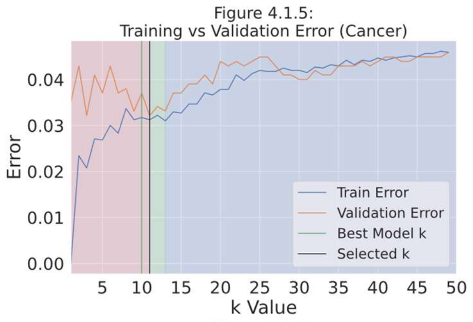


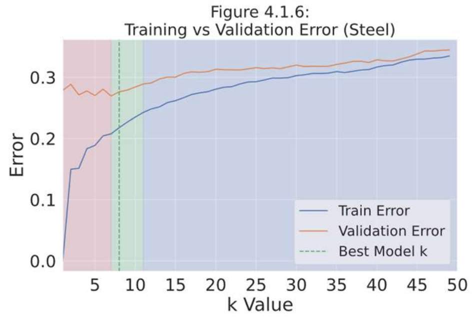
Finally, k represents the number of neighbors to consider when querying to make a prediction. Bias-Variance tradeoff can be observed from the validation curves from **Figures 4.1.5 and 4.1.6**. The bias-variance tradeoff is when fewer neighbors are considered (lower k value) in making a prediction, it typically corresponds to a lower bias, i.e., training error. The train error is lower because with less data points considered, the more say the true training data point label will have. The tradeoff is, however, due to only having a few points to consider when calculating the test error, the variance tends to increase. This is a result of being susceptible to outliers and noisy data. This area of overfitting is indicated by the area in red on the plots.

Underfitting occurred when too many neighbors were considered when making prediction, corresponding to higher k values. Training error goes up because the true training point has less say with the additional points considered in the prediction. This area of underfitting is indicated by the area in blue on the plots.

The green shaded area is where k values that fit the data well and the green dotted line in the plots indicate the model selected during the grid-search process.

For the Cancer dataset, the grid-search process found a model with k equal to ten as the optimal model, but as **Figure 4.1.5** shows, the validation error was beginning upward while the training error was trending down. This k value may not generalize very well to the test data as this was early indication of overfitting. A value between 11 and 13 may be a better fit since the training error was level before bias started to get introduced, which is an indication of underfitting.

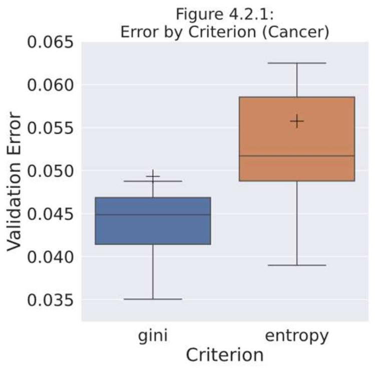




For the Steel dataset, the grid-search process found a k value of eight to be optimal, as shown in **Figure 4.1.6**. There is not much cause for concern with this value since it does not lay in the overfitting region where the Validation and Training errors begin to diverge. Also, the value was not in the underfitting region.

4.2 Decision Tree

Criterion calculates the quality of the splits within decision tree nodes. The two considered for this experiment were Gini impurity and Entropy. According to Wikipedia, Gini is "a measure of how often a randomly chosen element from the set would be incorrectly labeled" (Wikipedia, 2023). Entropy is more computationally complex because the equation involves a log function.



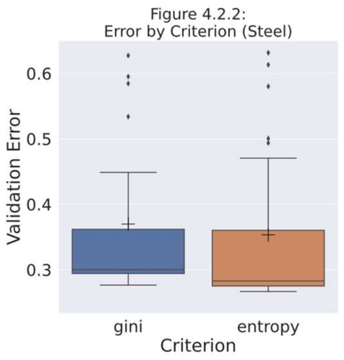
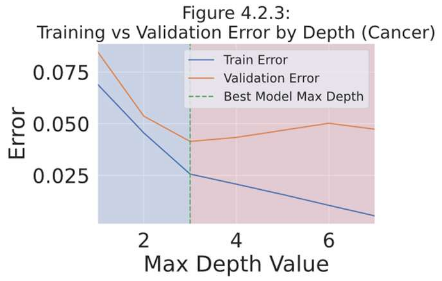
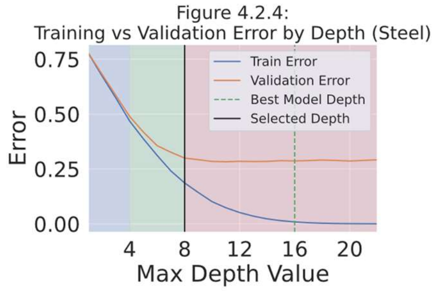


Figure 4.2.1 shows the average cross validation error against the Cancer dataset and Gini had a low mean score. **Figure 4.2.2** represents the runs for the Steel data and results show Entropy with the lower average. This result was due to the high level of imbalance present in the Steel class.

Pruning is a common practice applied with decision trees; otherwise, overfitting can occur. For instance, without any pruning for the Cancer dataset, the algorithm generated a tree with a maximum depth of seven and 24 leaves. This tree was much smaller than the tree generated for the Steel dataset that had maximum depth 22 and 248 leaves. This comes as no surprise since the Steel dataset has more features and more classes to label. Both trees created for each dataset had a perfect score of 1.0 for their training set. This is a result of not pruning as all nodes are expanded until all leaves are pure. This tends to cause an excess amount of variance in the test results. For both datasets, the test result scores were much lower than the previously mentioned training results (.930 for the Cancer test set and .712 for the Steel dataset). For this study, the process of pre-pruning was implemented to help generalize these models. Maximum depth was specifically analyzed to lower the variance for each dataset.



For the Cancer dataset, the grid-search process found a model with depth equal to three as the optimal model. **Figure 4.2.3** shows this maximum depth value to generalize well. If the depth was any shorter, then the plot shows bias beginning to creep into the model. While if the tree was constructed to go deeper through the data, then it introduced high variance and shows the model does not generalize; the validation error rises as the training error steadily declines.

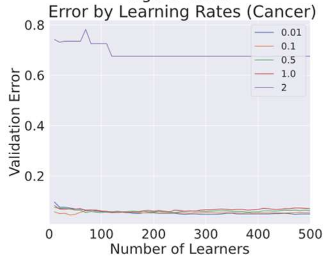


For the Steel dataset, the grid-search process found a depth of 16 to be the optimal model, as shown in **Figure 4.2.4**. A more generalizing depth would be between six and eight. After depth of eight, there was not much reduction in the validation error which indicated information gained from these additional nodes was more prevalent to the training set. Prior to this depth, each additional depth was adding insight as the training and validation errors decreased significantly with each split of the data.

4.3 AdaBoost

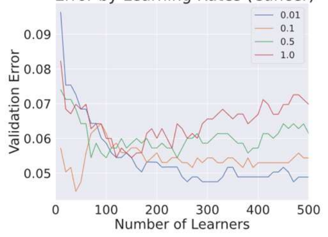
For boosting to be effective it is essential to have a weak base learner, which is a classifier that classifies better than random guessing. The base learner used for this algorithm was Decision Tree, specifically "stumps", for both datasets.

Figure 4.3.1:
Error by Learning Rates (Cancer)



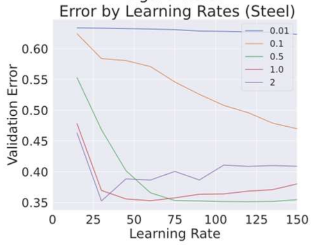
Next, was determining the learning rate. Learning rate's role in the algorithm determines the amount of weight to apply to each subsequent base estimator. **Figure 4.3.1** displays how the various learning rates respond to the boosting algorithm against the Cancer dataset. If the value is too small it can take a long time for the error to decrease since it is not putting enough emphasis on the next iterative base learner. In contrast, if the value of the learning rate is too high, learning rate =2, it can really impact how the model transfers over to the validation data. Since the high learning rate results in a high validation error, removing this value provides a more legible graph in **Figure 4.3.2**. In this figure, even though 0.01 descended at a slower initial rate, this line ultimately settles at a validation rate lower than the higher rates.

Figure 4.3.2:
Error by Learning Rates (Cancer)



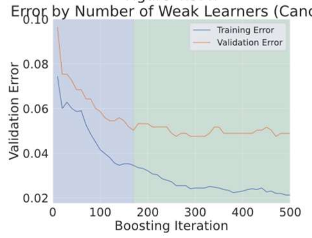
The large learning rate does not have the same impact on the Steel dataset displayed in **Figure 4.3.3**. Due to the dataset being more complex with additional features and more classes to try to classify, the algorithm needs to take more from each succeeding learner. Otherwise, if the rate is too small, the model will take a long time to learn, as plotted by the lines for 0.01 and 0.1. Ultimately, 0.5 achieved a low validation error while not taking too long to learn.

Figure 4.3.3:
Error by Learning Rates (Steel)

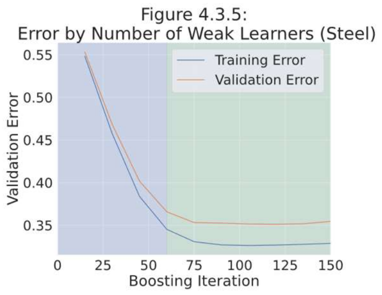


Once the learning rate was decided on, the next step was to determine how many learners to use. With 0.01 selected for the Cancer data, **Figure 4.3.4** shows how the AdaBoost ensemble progresses with added learners. The blue area identified the underfit region, because bias was increasing as the number of learners was reduced. After 170 learners, the validation error is not impacted that much; however, the model stabilized around 300 learners which is a good parameter to choose.

Figure 4.3.4:
Error by Number of Weak Learners (Cancer)



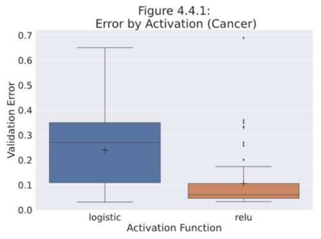
A learning rate of 0.5 was the reasonable choice for the Steel data and the resulting validation curve was plotted in **Figure 4.3.5**. While the number of learners was below 60, the errors benefit from each subsequent learner, so including few learners would introduce significant bias. 100 learners were selected for the optimized model related to the Steel data since the validation error seemed to stabilize at this point.

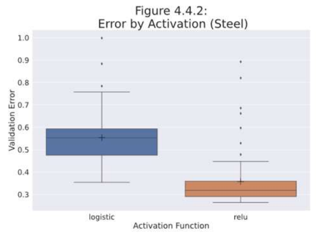


4.4 Artificial Neural Networks (ANN)

Activation functions play a key role in ANNs. They define an output of each node within the network based on the weighted input(s) they receive. Two common activation functions were explored in this exercise: Rectified Linear Unit (ReLU) and logistics sigmoid. ReLU only activates a node if its output is greater than 0, max(0, x). Sigmoid however outputs values between 0 and 1, 1/(1+e^-x).

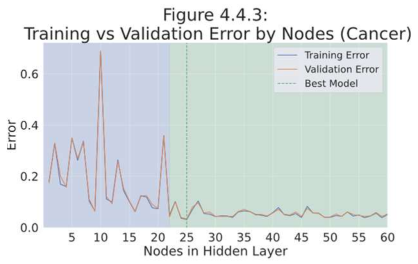
Figures 4.4.1 and 4.4.2 show ReLU outperformed sigmoid for both datasets. These results can be explained since the complexity of the ReLU activation function is computationally less expensive. Therefore, it converges faster than sigmoid resulting in a better fitted model.



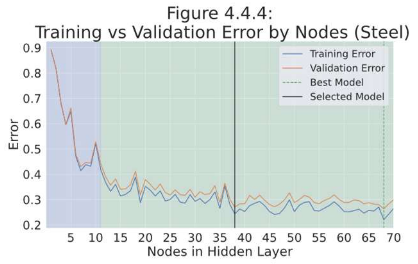


Another advantage of ReLU's cheaper computation was it reduced the testing time for the Cancer dataset across all the grid-search runs by 13.8% compared to the sigmoid testing times. Also, this reduced the Steel's testing time by 10.9%. If future studies were to include more samples, this reduction in time could prove to be significant.

Each neural network was constructed with a single layer, but tuning was performed on the number of nodes appropriate for each dataset. Starting with the Cancer data, **Figure 4.4.3** shows the validation curve for number of nodes. The model preferred by the Grid-Search operation was 25 nodes since less than 22 nodes presented high bias. This value is reasonable and therefore was selected to represent the optimal model.



Now for the Steel data, **Figure 4.4.4** shows a strong case for underfitting with less than 11 nodes. Both training and validation errors drastically increase with fewer nodes included. After this point, there was a small gap between the validation error and training error. A case could be made for the model to be constructed with 11 nodes. At this point the validation error and training are comparable. Alternatively, a case could be made for a model with more nodes since the validation curve never truly diverges from training error while continuing to decrease. This was not a strong indication of overfitting and justifies the node size of 68 which



resulted in the lowest validation error. The ultimate node size selected for the Steel data optimal model was 38, due to the lack of improvement versus the added complexity.

With the activation and the layer size finalized for each model, the last aspect to optimize was the learning rate. Learning rate determines how much to adjust weights of inputs in order to minimize the loss at each iteration. Too large of a rate and the algorithm could experience unstable learning; too small of a rate and the algorithm will take a long time to learn. **Figures 4.4.5 and Figure 4.4.6** show how each dataset responded to the various learning rates. Both dataset's validation error stabilizes quickly with a learning rate of 0.01 while smaller rates are learning the underlying model, but taking longer to learn.

Figure 4.4.5:
Errors by Learning Rates (Cancer)

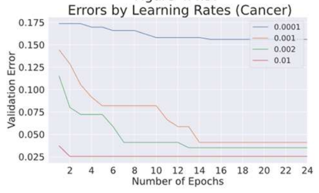
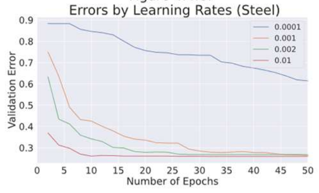


Figure 4.4.6:
Errors by Learning Rates (Steel)



Figures 4.4.7 and 4.4.8 show the final learning curve by iteration for Cancer and Steel respectively. A key difference was since the Cancer dataset was less complex with fewer features and class outputs, the algorithm only needed 2 epochs (i.e., loops through the dataset). To learn a strong model, the Steel dataset had to loop through 10 times as many epochs to converge.

Figure 4.4.7:
Errors by Iteration (Cancer)

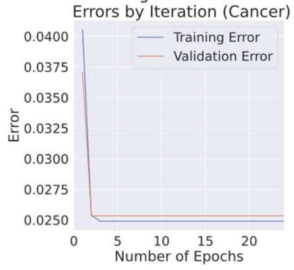
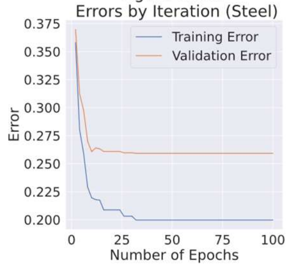


Figure 4.4.8:
Errors by Iteration (Steel)



4.5 Support Vector Machines (SVM)

If data is linearly separable, SVMs can perform very well, but not all datasets are linearly separable. In the case of the Steel dataset, it turned out to be non-linear in nature. Luckily, SVMs take advantage of kernels to transform data into higher dimensional spaces to make them linearly separable. Kernel selection can give insight into whether the underlying data is linearly separable. Linear kernel works well for linearly separable datasets. Radial Basis Function (RBF) works for more complex datasets where the underlying data was non-linearly separable and was a popular kernel when there is no prior knowledge of a dataset.

Adding dimensions can increase opportunity for overfitting, so L2 (Ridge) regularization was considered to help mitigate the risk. Regularization added a penalty component as the model complexity increased. Note, L2 was used as opposed to L1 since L1 shrinks coefficients of less important features to zero. This is better suited for feature selection which was not part of this study since the number of features used in each dataset helps portray the differences amongst different algorithms.

Figure 4.5.1:
Error by Kernel & Regularization (Cancer)

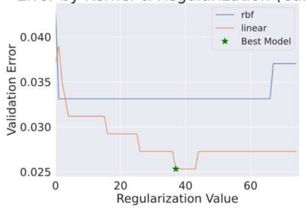


Figure 4.5.1 shows SVM with various kernels matched with regularization values for the Cancer dataset. Without any regularization, RBF kernel outperformed the linear kernel. **Figure 4.5.2** shows this was due to the effects of overfitting occurring with the linear kernel. Once a small penalty was added for the complexity introduced, this kernel beat out RBF kernels. Overfitting occurred when L2 was less than 25.5, but when too much penalty was presented, underfitting became more of a concern over 70. Any value between these values was appropriate to consider for the optimal model. Ultimately, the linear model that generated the lowest validation error with regularization equal to 37 was selected to represent the optimal model. Therefore, the Cancer dataset is linearly separable. Note, the true parameter for SVM is C, which is the inverse of L2 penalty. In this example, C equals 1/37.

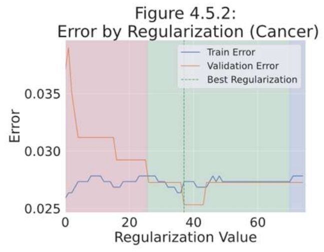
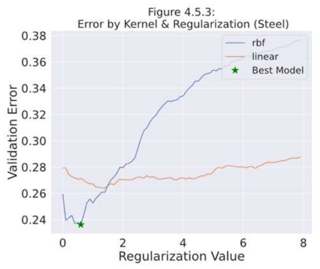
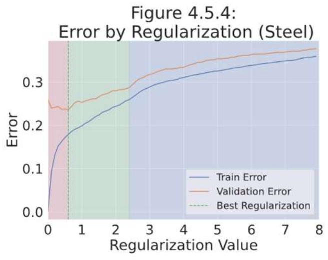


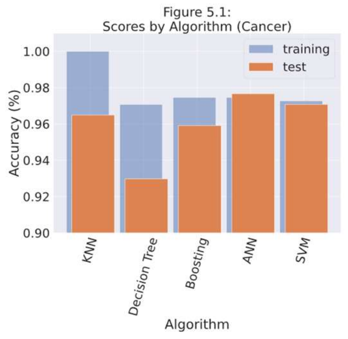
Figure 4.5.3 shows the various models' validation errors for the Steel dataset. The Steel dataset proved to be more non-linearly separable as RBF achieved the lowest error across the various models. **Figure 4.5.4** shows the RBF kernel's errors across different regularization values. The best regularization, indicated by the green dotted line, was not in either of the overfit or underfit regions, so provides evidence it was a strong model to proceed with as the optimal. The parameters of this model were RBF with a regularization of 0.6, or C = 1/0.6. The Steel dataset was non-linearly separable.





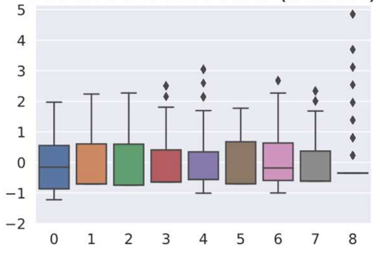
5 Comparing Optimal Algorithms

Figure 5.1 shows how the optimal model for each algorithm performed on the training and test sets from the Cancer dataset. The Neural Networks and the SVM both generated models with both low bias and low variance. SVM's learning curve, **Table 1 (SVM column)**, illustrates that additional data may even improve the model's accuracy. **Figure 5.2** shows the distribution of the features for the dataset. **Figure 5.3** shows the correlation of the features within the Cancer dataset. Each algorithm did fairly well for multiple reasons, including little noise present in the data, most features moderately correlated with the target class, and the classes were relatively balanced.



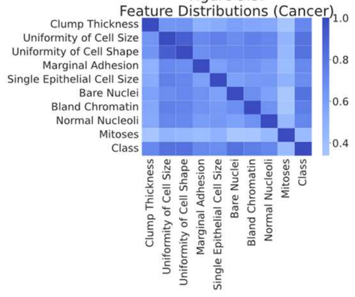
The models with higher variance were KNN, Decision Trees, and Boosting. The gap between the KNN model can be explored first. KNN's test score was equivalent to ANN and SVM, so the gap appears to be a product of the training score. The bias was zero on KNN because when the training point was scored, it first queried the 11 nearest neighbors and then weighted them by similarity. Given the training point queried itself as one of the 11 and then matched exactly, that point received all the weight. Consequently, this dictated the label of that point and always returned with the true labels.

Figure 5.2:
Feature Distributions (Cancer)



Decision Tree did not do as well as the other algorithms. It had relatively low bias comparable to ANN and SVM, but the variance was higher. Decision Trees are greedy algorithms, meaning they choose the best split based on the current node without taking into consideration next splits. This can lead decision trees to local minimums, rather than the global minimum.

Figure 5.3:
Feature Distributions (Cancer)



Boosting helped alleviate some of the overfitting that occurred from the Decision Tree, but still did not outperform ANN or SVM.

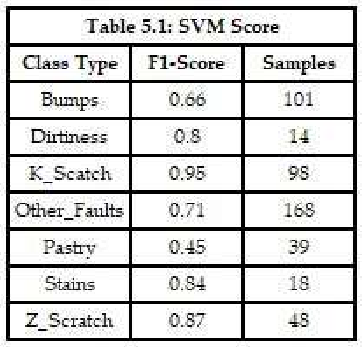
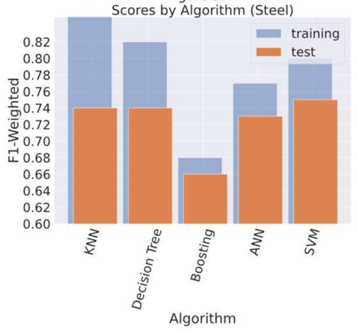


Table 5.1: SVM Score		
Class Type	F1-Score	Samples
Bumps	0.66	101
Dirtiness	0.8	14
K_Scatch	0.95	98
Other_Faults	0.71	168
Pastry	0.45	39
Stains	0.84	18
Z_Scratch	0.87	48

Figure 5.2 shows performance for the optimal model for each algorithm for the Steel dataset's training and test sets. **Table 5.1** presents the F1-scores for the SVM model since it ultimately had the highest test score. This table along with the confusion matrix, in **Table 2 (SVM column)**, offer some insight into how the model classified the test set. Two of the higher sample groupings, Other and Bumps, unfortunately had some of the lower scores. The confusion matrix shows 29 of the 168 Other class types being predicted as Bumps and 30 of the 101 Bumps being predicted as Other. The weighted F1-score for the remaining class types was 0.82, which was seven points higher. This indicated class overlap or noisy class data. Class overlap could be corrected with the addition of features that provide distinguishing qualities for the two classes. Noisy class data would be a product of human data entry error. For example, employees could have misclassified a steel fault by attaching their own interpretational bias when visually inspecting the plate.

Figure 5.4:
Scores by Algorithm (Steel)



Investigating more into the features, the Steel data was also dealing with noisy feature data. **Figure 5.4** shows the scaled feature distributions for the dataset. The Steel dataset has several features with many outliers. The clear presence of feature noise was a more prominent factor for the Steel dataset. This helps explain the struggle of the AdaBoost algorithm for the Steel dataset. AdaBoost uses a loss rate based on misclassification rate which is sensitive to outliers as each additional classifier has to correct the errors of its predecessor.

Figure 5.5:
Feature Distributions (Steel)

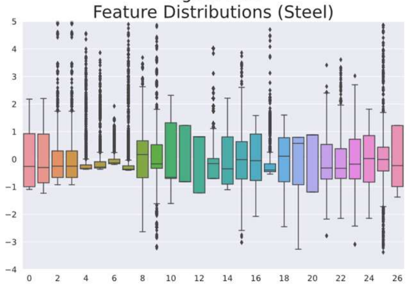
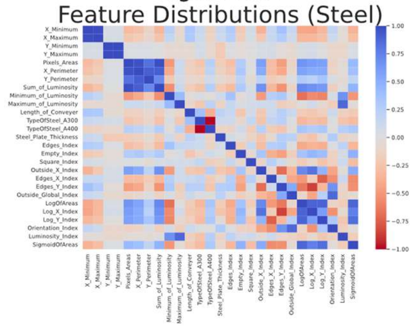


Figure 5.6:
Feature Distributions (Steel)



Steel dataset was imbalanced and the presence of outliers in the data caused issues for KNN. Dirtiness only had 41 training samples while Other category had over 500. With Other having more than 12 times the representation, this provided more chances for an outlier to match more similarly than the true Dirtiness training point and caused the model to mislabel.

SVM algorithms focus on optimizing the margin between boundaries, but the presence of outliers can impact this process. By penalizing these points allowed this algorithm to be more robust with handling the Steel datasets noise.

Even though SVM finished with the highest test score, ANN has a strong case for being considered as the best model. It had a smaller gap between training and test score, while final test results were not significantly lower than SVM. The main justification comes from examining the learning curve, **Table 2 (column ANN)**. The training and test curves offer an expectation that additional data could improve the score for neural networks. Also, the testing time was 8.57% faster on the Steel training set.

Acquiring more data would come at some expense to training time however, since ANN are generally more computationally intensive than SVM. From **Table 2 (columns SVM and ANN)**, the training times were 0.504 and 0.079 seconds, approximately 6 times longer. For this moderately sized dataset it is relatively insignificant, but becomes more of a factor as the dataset grows. This effect can be lessened by taking advantage of parallel computing, such as GPUs.

Detailed results are displayed in **Table 1 and Table 2** for the Cancer and Steel dataset respectively.

The goal of this experiment was not to optimize results, but to see how dataset characteristics are presented in the various algorithms. When Exploritory Data Analysis (EDA) is done prior to optimizing a model, the characteristics provides some foreshadowing on which algorithms will do best.

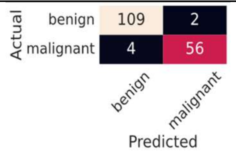
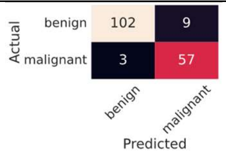
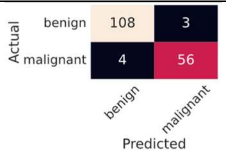
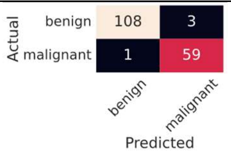
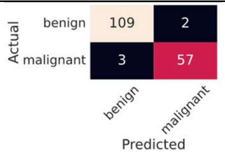
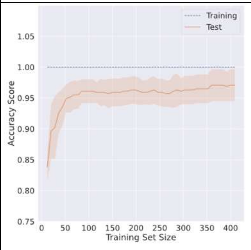
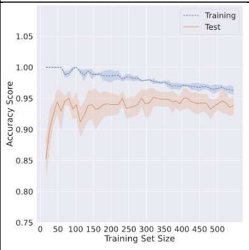
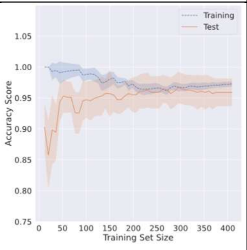
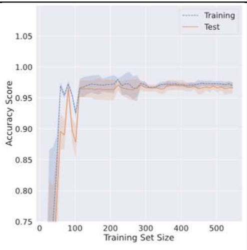
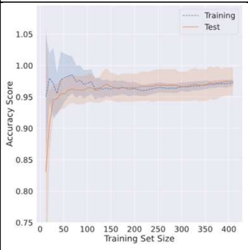
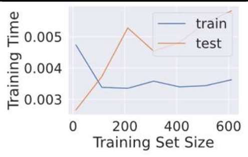
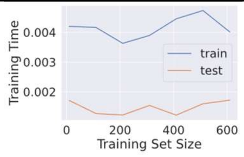
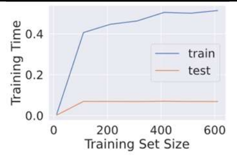
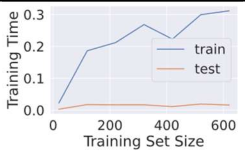
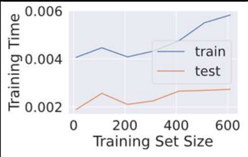
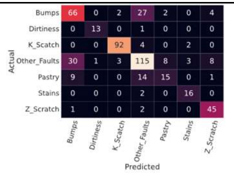
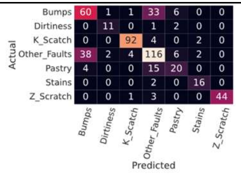
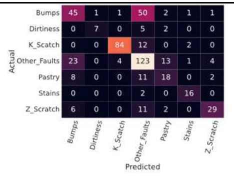
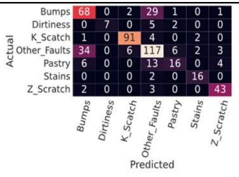
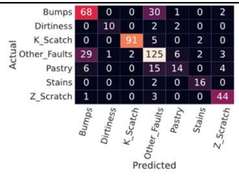
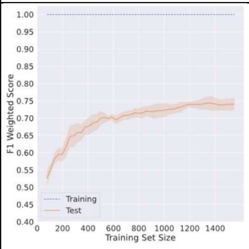
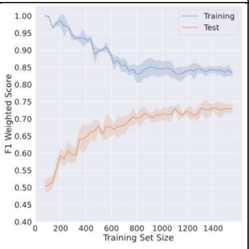
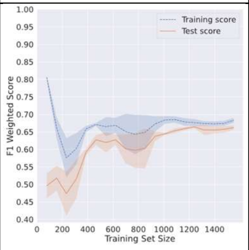
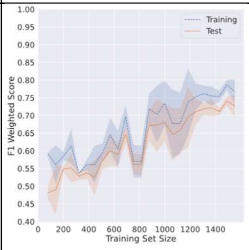
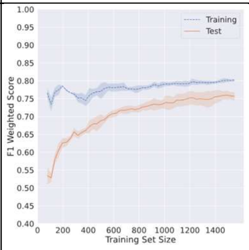
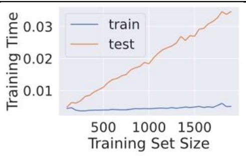
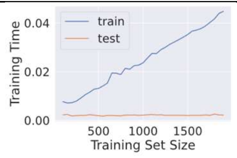
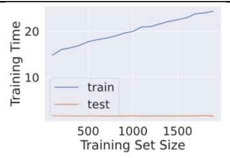
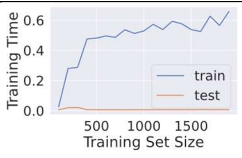
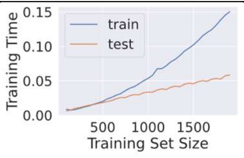
Table 1: Optimal Results Across Algorithms (Cancer)					
	KNN	Decision Trees	Boosting	ANN	SVM
Optimal Parameters	**Distance**: Euclidean **Weights**: Distance **k**: 11	**Criterion**: Gini **Max Depth**: 3	**# of Estimators**: 300 **Learning Rate**: 0.01	**Activation**: ReLU **Nodes**: 25 **Learning Rate**: 0.01	**Kernel**: Linear **C**: 1/37
Accuracy	Test: 96.49% Train: 100.0%	Test: 92.98% Train: 97.07%	Test: 95.91% Train: 97.46%	Test: 97.66% Train: 97.46%	Test: 97.08% Train: 97.27%
Confusion Matrix					
Learning Curve					
Training Curve					
Training Time	0.003 seconds	0.005 seconds	0.512 seconds	0.273 seconds	0.005 seconds
Testing Time	0.006 seconds	0.002 seconds	0.057 seconds	0.003 seconds	0.003 seconds

Table 2: Optimal Results Across Algorithms (Steel)					
	KNN	Decision Trees	Boosting	ANN	SVM
Optimal Parameters	**Distance**: Manhattan **Weights**: Distance **k**: 8	**Criterion**: Entropy **Max Depth**: 8	**# of Estimators**: 100 **Learning Rate**: 0.5	**Activation**: ReLU **Nodes**: 68 **Learning Rate**: 0.01	**Kernel**: RBF **C**: 1/0.6
F1-Weighted	Test: 0.74 Train: 1.00	Test: 0.74 Train: 0.82	Test: 0.66 Train: 0.68	Test: 0.73 Train: 0.77	Test: 0.75 Train: 0.80
Confusion Matrix					
Learning Curve					
Training Curve					
Training Time	0.009 seconds	0.041 seconds	22.202 seconds	0.504 seconds	0.079 seconds
Testing Time	0.033 seconds	0.003 seconds	1.610 seconds	0.032 seconds	0.035 seconds

6 References

1. Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. New York, USA.
2. Wikipedia (2023, February 4). *Decision tree learning*. Retrieved on February 4, 2023 from https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity