CS 7641 – Supervised Learning (Assignment 1)

Introduction

This paper will analyze five different supervised learning algorithms – Decision Trees, Boosting, Neural Networks, Support Vector Machines, and k-Nearest Neighbors on two different datasets - Faulty Steel Plates and Phishing Websites. Weka, a suite of machine learning software written in Java, is used for analysis.

Description of datasets

Both datasets can be found on the UCI Machine Learning Repository.

Faulty Steel Plates:

The Steel Plates Faults dataset contains 1,941 instances with 27 attributes that classifies faulty steel plates into 7 different categories – Pastry, Z_Scratch, K_Scratch, Stains, Dirtiness, Bumps, and Other_Faults. Other_Faults comprises about 35% of the classes, while Bumps and K_Scratch make up about 20% each. All of the features are numerical.

Phishing Websites:

The Phishing Websites dataset classifies 11,055 instances with 30 attributes as a binary response of -1 (not phishing) or 1 (phishing). All data is nominal, either -1, 0, or 1. There are 4,898 instances that are not phishing websites and 6,157 instances that are, which shows that the data isn't unbalanced. To do better than average would mean at least a 56% positive classification. The data is not noisy.

Data wrangling

For the faulty steel plates dataset, I found a converted csv file on Kaggle that had the 7 different labels one-hot encoded. I transformed the labels into one column with 7 different nominal values. I took out the TypeOfSteel_A400 feature because it's redundant since there are only two types of steel and TypeOfSteel_A300 is binary.

There was no data wrangling performed on the phishing websites datasets.

Why are the datasets interesting?

Analyzing faulty steel plates is important in improving safety and reducing costs. Automatic pattern recognition of faulty steel plates can reduce the amount of defective plates that are in circulation and possibly used. Defective plates may be dangerous and pose safety concerns or not be aesthetically pleasing. Classifying defective steel plates before they leave the factory can save high return shipping fees and lead to better customer satisfaction. This technique can also be widely applicable to evaluate other types of defective metals. It's also good because it's challenging to classify the dataset into seven different classes.

Identifying phishing websites is extremely important for online security. It can prevent identity theft or credit card fraud. Internet browsers like Chrome sometimes notify the user when it believes a website is unsafe to proceed. This can prevent someone from mistakenly giving out bank login information or even their social security number to phishing websites. This dataset is interesting because it has great features, isn't noisy, and can classify phishing websites well.

CS 7641 – Supervised Learning (Assignment 1)

Analysis Steps:

Data is split 80-20, 80% training and 20% test data. 10-fold cross validation will be used on the training set to help generalize the data more and avoid overfitting. Stratified k-Fold cross-validation estimates and uses the data more effectively. The accuracy rate for training and cross-validation will be averaged across the folds. I'll be using the training set to build the model, validation set to tune the hyperparameters and pick a model, and the held-out test set to estimate the final performance and quality of the chosen model.

For each algorithm, I will graph two learning curves:

1. Training and validation accuracy against training size
    a. This is used to evaluate bias and variance. Training size will be in increments of 10%.
2. Training and validation accuracy against a range of a single hyperparameter value
    a. To see how the performance of training and cross-validation accuracy vary for different inputs of a single hyperparameter. This helps with model evaluation.

Algorithms are also compared and contrasted within and between datasets.

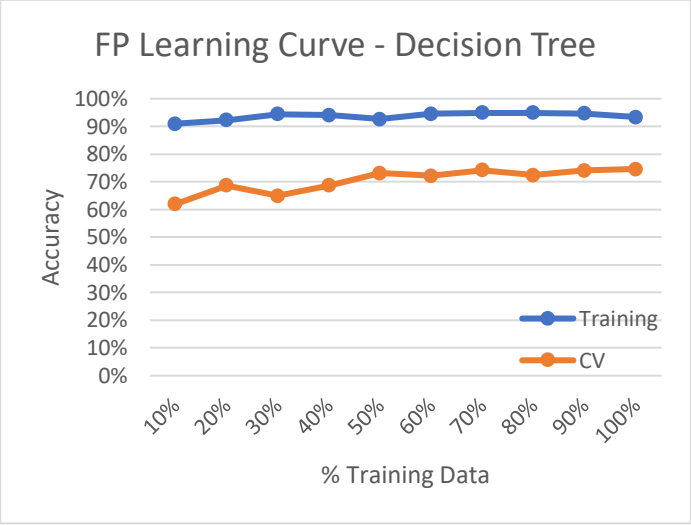
Decision Tree

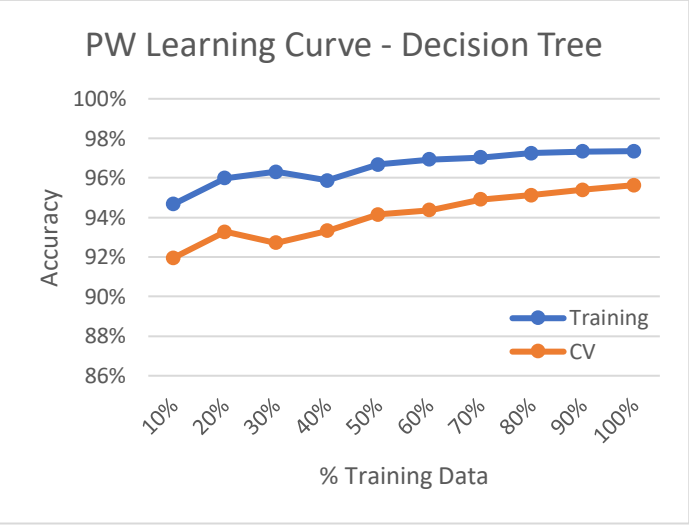
In Weka, the J48 classifier is used to build the decision tree. This is a top down learning algorithm that splits the best attributes (decision nodes) based on information gain. The branches are the outcomes of the binary splits (≥,≤ or T, F) and the leaf nodes represent a classifying label.

For model complexity, I'll be adjusting the hyperparameters for pruning (T, F) and confidence factor (default = 0.25), where lower values incur more pruning. Pruning serves to simplify the tree, making it easier to understand the results, and avoid the risk of overfitting the training data.

The decision tree algorithm is an eager learner, which means it has a long train time and low prediction time.

Learning Curve:





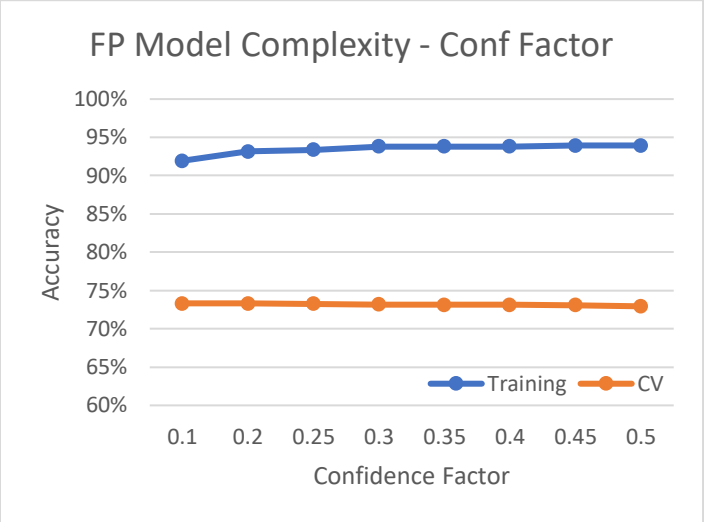
The learning curve for faulty plates indicate high variance/overfitting. The training accuracy is around 93%, while the validation accuracy is around 70%. The validation accuracy does improve from 62% to 75% when there's more training data to build the model.

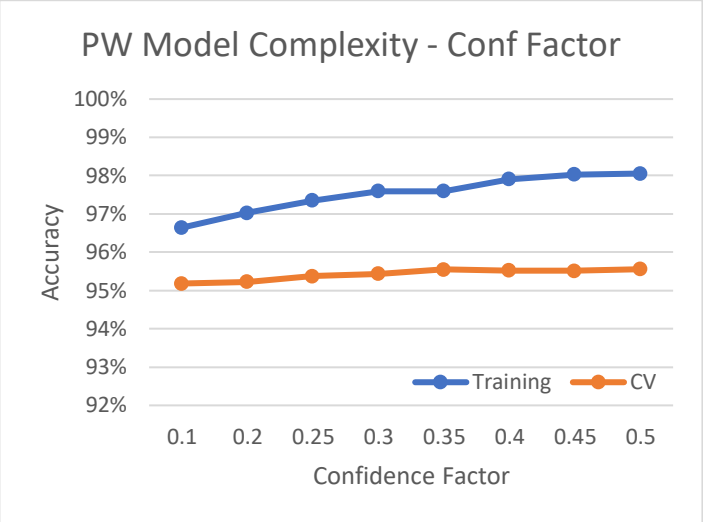
The learning curve for phishing websites shows low variance and low bias, which is what we want. Accuracy is high for both the training set and validation set, which means the model is generalizable and performs well. The phishing websites model classifies well even with little training data at 10%.

The phishing websites model has a much better classification accuracy than faulty plates, 95% vs 75%, respectively. This could be primarily due to two reasons. Faulty plates has seven different class labels, while phishing websites only has a binary label, making it easier to classify. Also, the faulty steel plates dataset contains more noisy data.

The large difference in the classification accuracy between the two will probably be a recurring trend in all the algorithms.

Model Complexity:





In evaluating model complexity, I will tune the hyperparameters and pick the best performing one. For both training models, I adjusted the confidence factor, where lower values incur more pruning. I also ran models with an unpruned tree, which isn't shown here. The cross-validation accuracy within datasets, was similar for confidence factor intervals ranging from 0.1 to 0.5. The Phishing dataset stayed around 95% while Faults was around 73% accuracy.

For phishing, the unpruned tree actually has the highest cv-accuracy by a few tenths of a percent. However, the slight improvement in accuracy isn't worth it. The size of the unpruned tree is 579, which is much higher than the pruned trees. It's better to have a pruned tree that is less likely to overfit. The second highest cv-accuracy has a confidence factor = 0.5 with a tree size of 432. We could even choose a more pruned tree, but I'll use this model for algorithm comparison in the end.

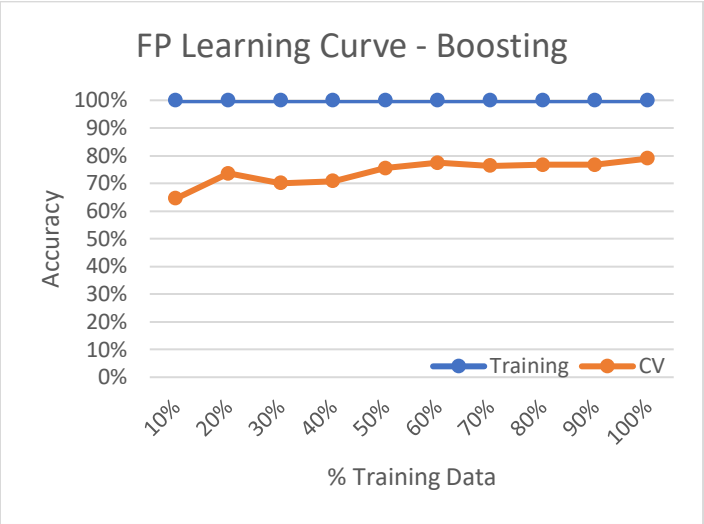
For faulty plates, the highest cv-accuracy was 73.32% with a confidence factor = 0.2. The size of the tree is 257 with 129 leaves. I'll use this model for algorithm comparison.

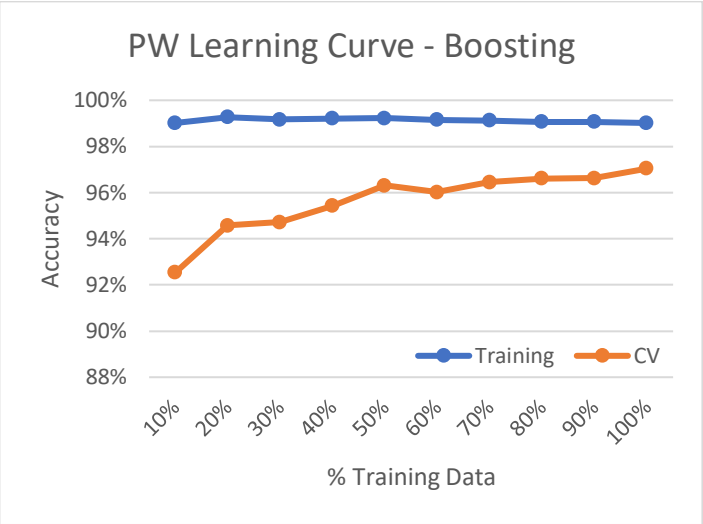
Boosting

Boosting is an ensemble learning method that incrementally builds the model by placing more weight on the misclassified data. It basically turns a set of weak learners, which always do better than chance, into a single strong learner.

In Weka, the boosting algorithm is called AdaBoostM1. The main hyperparameters for this model is the base classifier and the number of iterations to improve the weak learners. I'll use the J48 Decision Tree as my base classifier with default settings of a pruned tree and a 0.25 confidence factor because these settings did well in my original model evaluation for Decision Trees. I'll modify the hyperparameter for number of iterations from 10 to 50 for my model complexity curve. Higher iterations use more weak learners.

Learning Curve:



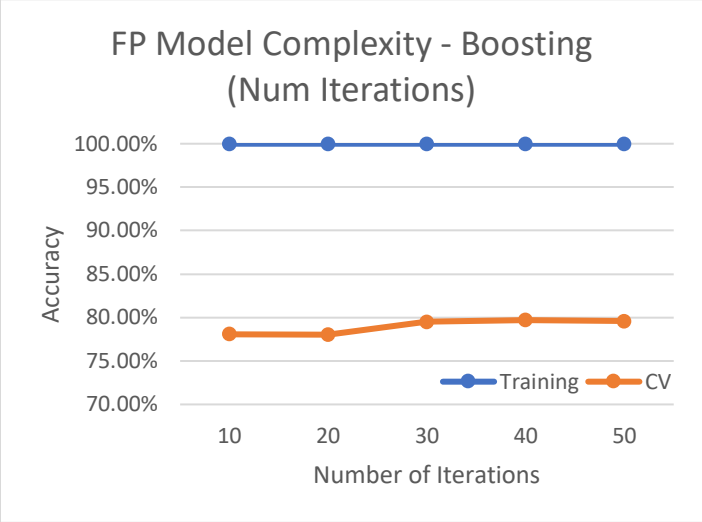


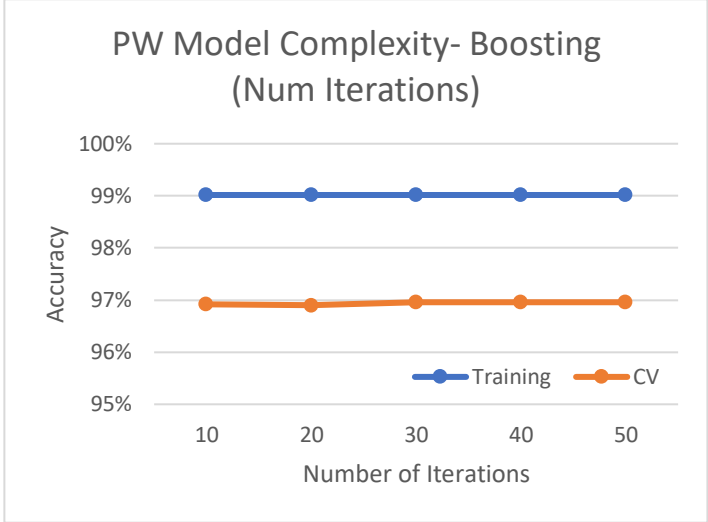
The learning curve for faulty plates show that the model has high variance/overfitting. The trained model has 100% accuracy on the training set, while the validation set only has around 71% accuracy. It doesn't generalize well. Phishing websites again has low bias and low variance with accuracy above 90% for training and validation data.

Both models improve a lot with more training data – faulty plates from 64% to 79% and phishing from 92.5% to 97.04%. This shows that with more training data, the model is able to generalize better and achieve a lower variance/bias.

The accuracy for both datasets have increased from the Decision Tree algorithm. The validation accuracy for faulty plates improves a little more than 4% while the accuracy for phishing improves by 2%. This is expected because boosting improves upon our decision tree base classifier by turning weak learners into a set of strong learners.

Model Complexity:





For model evaluation, I'll be varying the number of iterations from 10 to 50 to improve weak learners. Faulty plates has the best cv-accuracy at 79.70% where iterations = 40. At 40 iterations, the tree size is 293 with 147 leaves.

Phishing websites has the best model with number of iterations = 30 and an accuracy of 96.96%. At 30 iterations, the tree size is lowest at 19. Iterations 40 and 50 have the same cv-accuracy, but the tree sizes are 73 and 51, respectively. A lower number of iterations will allow this eager learner to train faster.

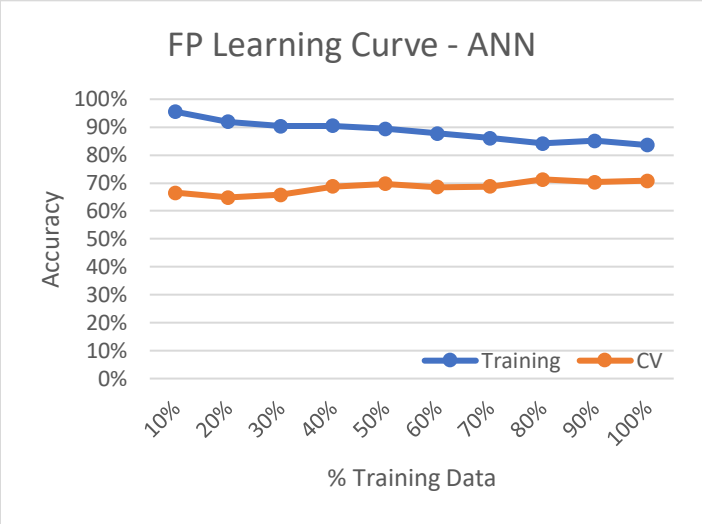
Again, the boosting model with decision tree classifies better than the original decision tree with these optimal hyperparameters.

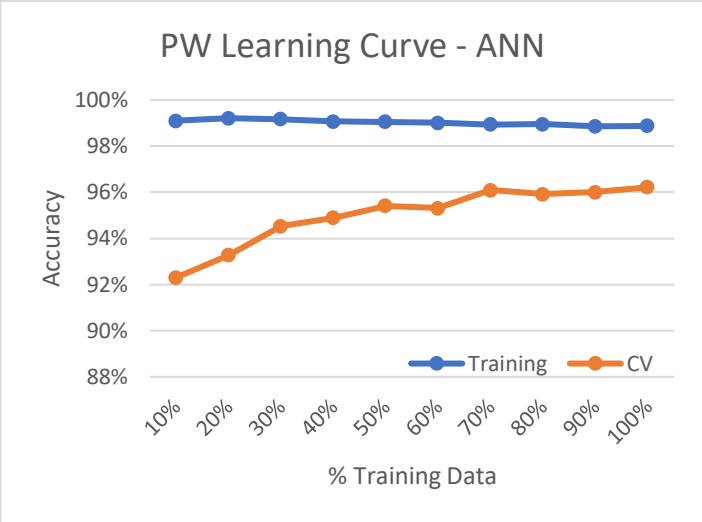
Artificial Neural Network

A neural network tries to mimic the brain in mapping inputs to outputs. Weka uses MultiLayerPerceptron - a network of perceptions . It has a default of one hidden layer with (attributes + classes)/2 neurons in each layer. The weights for the perceptrons are learned from the training set and updated via “backpropagation”. The change in weight is the learning rate times the gradient plus the previous change in weight times the momentum. Thus, hidden layers, # of neurons, and weights (learning rate, momentum) are the hyperparameters.

This is also an eager learning algorithm that produces good results, but is very slow.

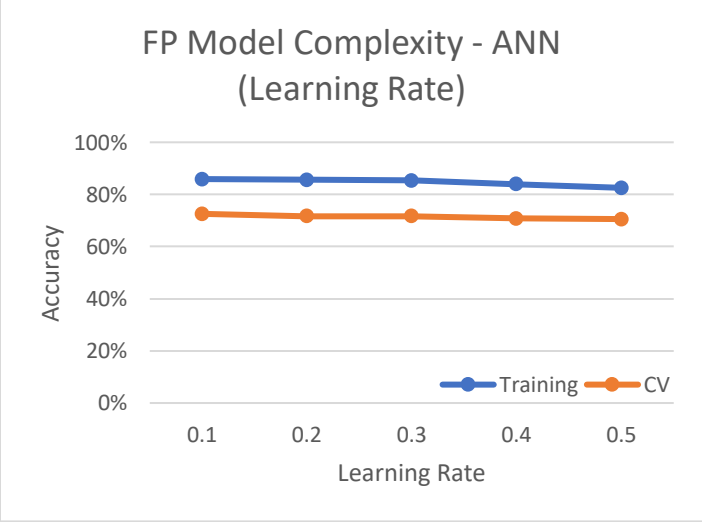
Learning Curve:

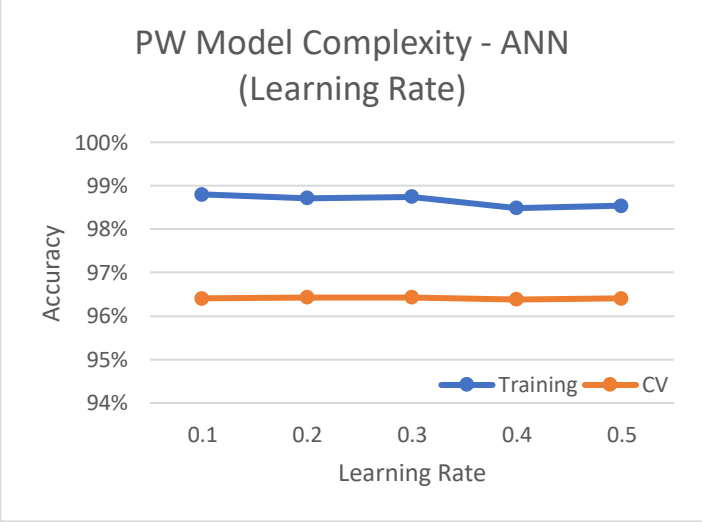




The learning curve for faulty plates show high variance/overfitting. Variance decreases as more of the data is used for training since the training and cv-accuracy slightly converge, but bias increases slightly as the model performs a lot worse on the training set. The variance for phishing websites decreases with more training data. Phishing websites displays low variance/bias, indicating a good classifier.

Model Complexity:





In the model complexity graph above, I plotted the learning rate against the accuracy.
For model evaluation, I tried different amounts of hidden layers, neurons, learning rate, and momentum. For both trained models, as the number of hidden layers increased, the cv-accuracy decreased. When I altered the amount of neurons for a single hidden layer, I found the optimal for both to be 16 neurons. This was the default [attributes + classes)/2] neurons. After plugging in different values for learning rate (default = 0.2) and momentum (default = 0.3), I found that that lower rates produced better results. This means the algorithm takes smaller steps in calculating the weight change, which can lead to higher accuracy, but also longer training time.

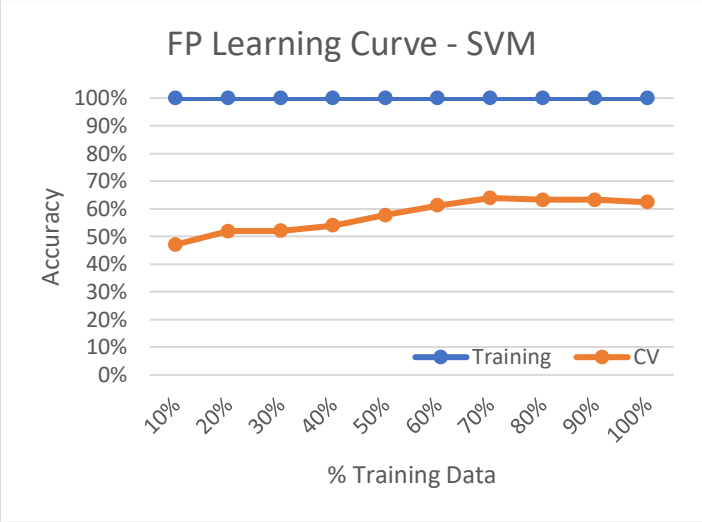
The optimal hyperparameters for faulty plates and phishing websites had 1 hidden layer, 16 neurons, and a learning rate and momentum of 0.1. The accuracy was 72.87% for faults and 96.69% for phishing. I will use these models for algorithm comparison in the end.

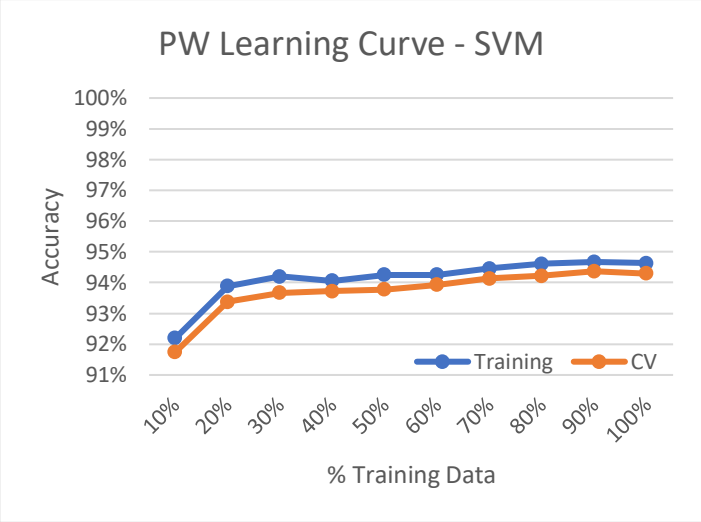
Support Vector Machines

Support Vector Machines uses hyperplanes to maximize the margins between the different classes. The larger the distance between the hyperplanes, the better the generalization and separation of classes.

In Weka, LibSVM is the most popular and robust SVM algorithm. LibSVM uses one vs one classification, where each different pair of labels has a separately trained classifier. For the Fault dataset that has 7 classes, the one vs. one classification has 21 different SVM classifiers. The final prediction is based off of a vote of the combined classifiers. SVM is an eager learner. Default settings were used for each kernel.

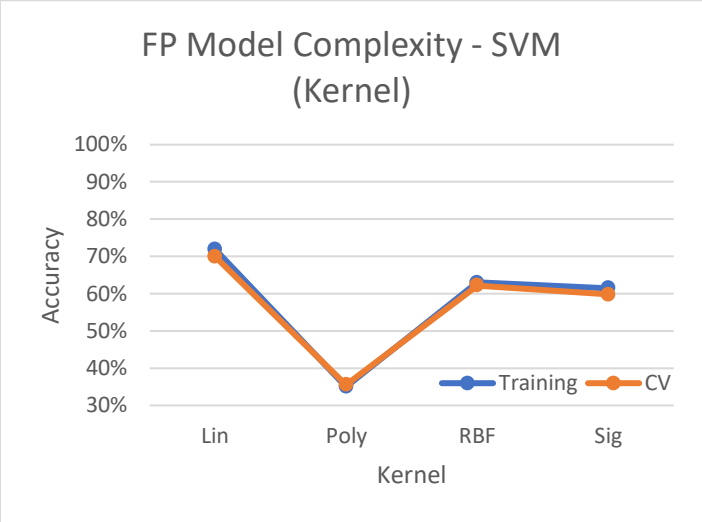
Learning Curve:

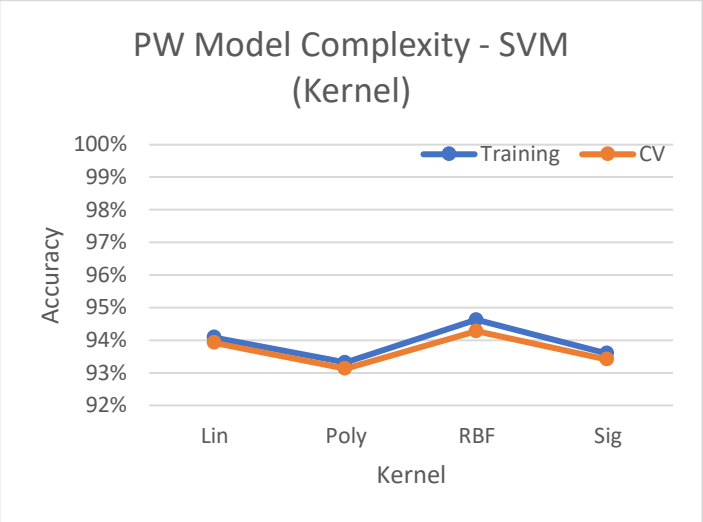




The learning curve for faulty plates indicates high variance/overfitting. The training set has 100% accuracy, while the validation set only performs around 62%. This is our worst performing algorithm yet! The phishing website SVM model has low bias, which is demonstrated by the training accuracy line being almost identical to the cv accuracy line. Variance is low as well. Both models generalize better as the training set increases.

Model Complexity:





For faulty plates, the linear kernel does the best with a cv-accuracy of 69.91%. The polynomial kernel for faulty plates does not do a good job at all as it classifies all labels into Other_Faults, which is 35% of the class.

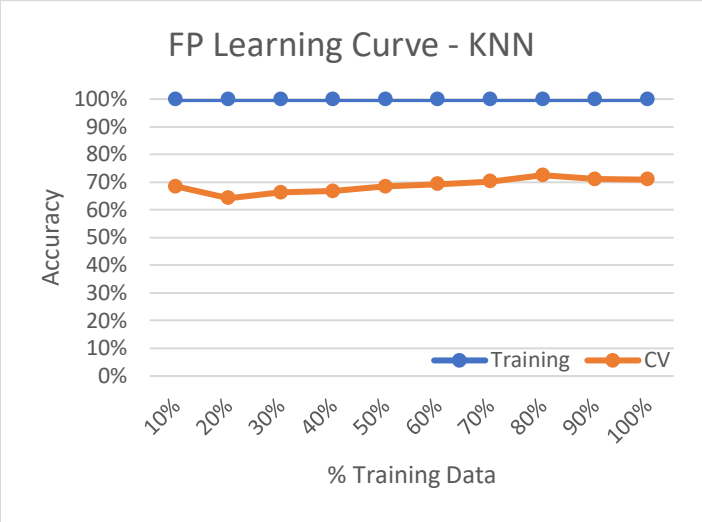
SVM turns out to be the poorest performer for faulty plates, most likely because the hyperplanes aren't able to sufficiently separate the seven classes. SVM performs better on binary classification. This is shown in phishing websites, where the RBF kernel with default hyperparameters performs better.

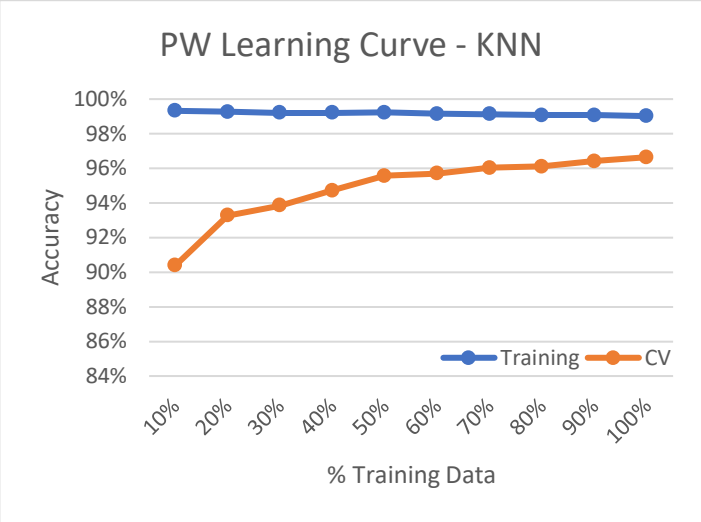
k-Nearest Neighbor

The k-Nearest Neighbor algorithm outputs a classification based on the majority of the number of neighbors. This is an instance based, lazy learner, which means it does nothing until you have to make a prediction.

In Weka, IBK is used. The main hyperparameters are the number of nearest neighbors and distance function. The default algorithm uses the Euclidean distance function and one nearest neighbor.

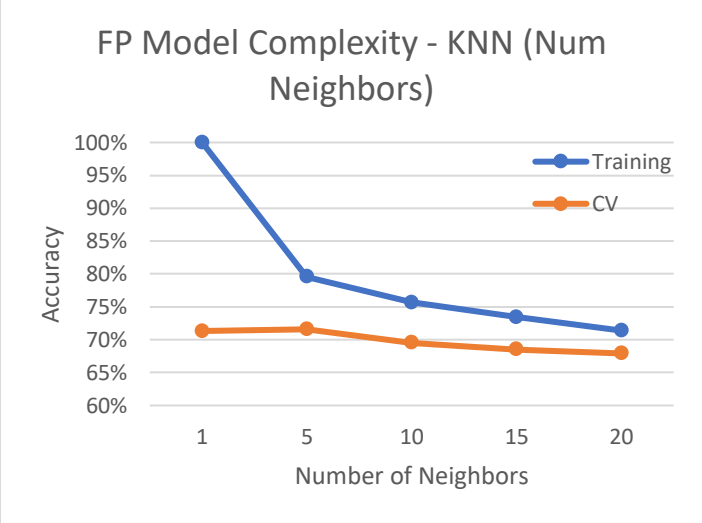
Learning Curve:

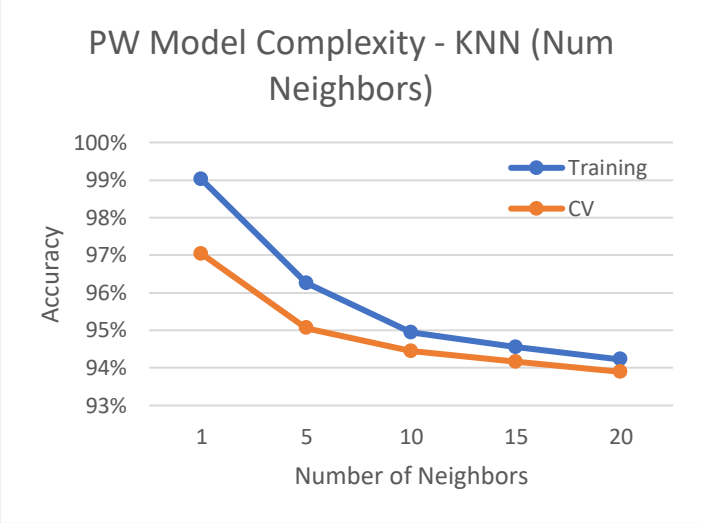




The faulty plates model again shows high variance/overfitting. The training set performs at 100%, while the accuracy for cv is 71%. The model for faulty plates doesn't improve as much with more training data while the kNN for phishing websites improves by over 6%. The variance for phishing websites decreases with more training data.

Model Complexity:





For model complexity, I experimented with different numbers of nearest neighbors. As the number of neighbors increases, more bias is introduced for both because the cv-accuracy and training accuracy get worse. The optimal number of neighbors for faulty plates is 5 with 71.59% cv-accuracy and phishing websites is 1 with 97.04% accuracy. These optimal models will be used for algorithm comparison.

Conclusion

Faulty Steel Plates:

Model	Training	CV	Test	Time
DT	93.17%	73.32%	73.52%	Fast
Boosting	100.00%	79.70%	77.12%	Medium
ANN	84.41%	72.87%	70.95%	Very Slow
SVM	71.84%	69.91%	69.15%	Fast/Medium
KNN	79.57%	71.59%	70.69%	Fast

Phishing Website:

Model	Training	CV	Test	Time
DT	98.06%	95.56%	96.16%	Fast
Boosting	99.02%	96.96%	96.79%	Medium
ANN	98.78%	96.69%	96.83%	Very Slow
SVM	94.63%	94.28%	94.30%	Fast/Medium
KNN	99.02%	97.04%	97.06%	Fast

After tuning the parameters, I used my test set to determine the final performance and quality of the models. Accuracy and confusion matrix were used to determine the best models.

For faulty steel plates, the boosting algorithm with a decision tree base classifier clearly produces the best test accuracy at 77.12%. Decision tree was second.

For phishing websites, the decision tree, boosting, and multilayer perceptron model all do well with an accuracy of 96% . KNN performs the best on the test set with a 97.06% accuracy. This is a fast algorithm since it's a lazy learner that only tests for 5 nearest neighbors.

The algorithms classified faulty steel plates significantly worse because there were seven classes as opposed to a binary classification in phishing websites. SVM did not do well in either because the hyperplanes could not separate labels well. ANN performed the slowest by a far margin.