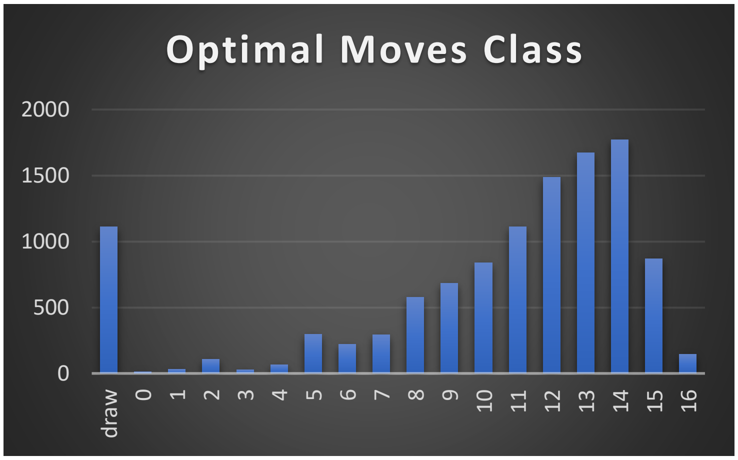
George Maroun CS 4641- Guzdial
Assignment #1

Supervised Learning Report

Datasets

Krkopt Chess (King-Rook vs. King):

This is a Chess Endgame Database for White King and Rook against Black King (KRK) - Black-to-move Positions Drawn or Lost in N Moves. This data set is obtained from a team of researchers at the Turing Institute who generated the database. The learning system is provided with examples of chess positions described only by the coordinates of the pieces on the board. The task is to predict the optimal number of moves required for the White player to win the game through checkmate given the relative positions of the two white pieces and one black piece. There are 7 attributes and a total of 11,222 instances. Below is the class attribute distribution for all the endgames in the file, ranging from a draw to 16 moves.



Diabetes:
This is a binary-valued dataset gathered from a population in Phoenix, Arizona for identifying potential signs of diabetes according to the World Health Organization. This has been used as an adaptive learning routine to forecast the onset of diabetes mellitus. There is a total of 768 instances with 9 attribute including class.

Why are these interesting datasets?

These datasets are interesting for both their practical implications and their supervised learning application results in machine learning. As far as the Krkopt data is concerned, these types of problems have been put to use in industry for the simple reason that a reliable cornerstone of logging our progress in the realm of artificial intelligence is checking algorithms against human beings. Companies like IBM and Intel have poured millions into research for supercomputers that outperform and consistently beat expert humans at strategy games like chess. Consolidating and optimizing searches in chess endgames could allow us to effectuate chess AI systems with more cost efficiency.
The Diabetes dataset is first and foremost blatantly interesting in the world of medicine. There is certainly a lot of medical innovation left to be done, and a huge part of it is going to come with computing power being able to glean through huge archives of data to extract patterns and generalize for further patients. On a small scale, this is what the dataset is doing in that it tracks common health abnormalities associated with diabetes and generates a system for diagnosing, quickly mind you, potential positive-testing individuals. These both do a great job in defining how machine learning will come into play in a practical sense in quotidian life.

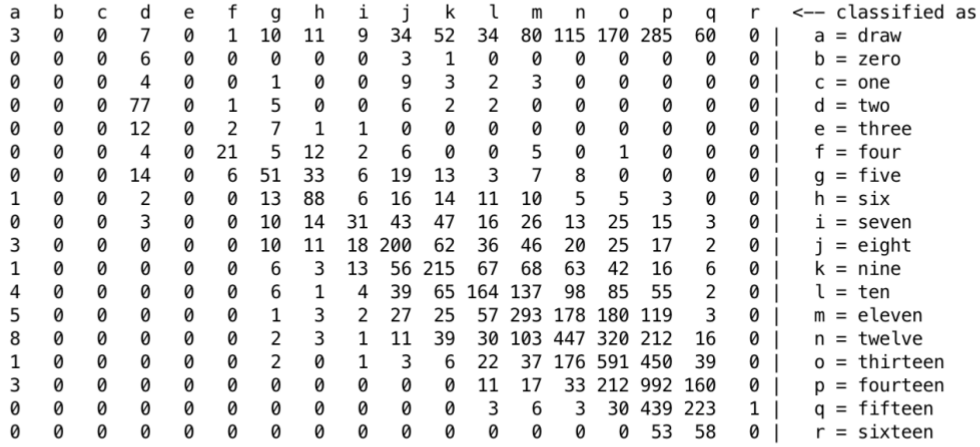
With respect to machine learning, the Krkopt dataset is of interest because of the noise association with it classification. The setup, albeit simple at a high level, provides ambiguity to the generalization in the sense that many endgame outputs that are different (draw vs. checkmate) have highly similar setups that overlap and blur the output. Also, worthy of noting, the dataset is of considerable value, but nowhere near the space of possible scenarios that the algorithm could encounter, which makes paying attention to runtime and overfitting suppressing inherently interesting to document.

With the diabetes dataset from an ML standpoint, it is a strong example of how sporadic a seemingly ordered and behaved a set of data entries can be when algorithms are run over it. While the testing accuracy did seem to long run improve within classifications, there was still a fair amount of error even with extensive parameter tweaking. This begs the question of hidden variables that are not exactly highlighted in datasets but sway results in an un-generalizable way.

Decision Trees: J48

My decision tree analysis incorporated several different levels of pruning (CF of .01, .05, .10. .25, .50) and a single unpruned tree.

Prune State	CF	Leaves	Tree Size	Train Time	Test Time	% Training	% Testing	% CV
Pruned	.01	365	417	.33	.06	41.0893	38.2799	37.8258
Pruned	.05	799	913	.35	.07	47.0038	39.6613	39.4186
Pruned	.1	1282	1465	.35	.06	51.8935	40.9982	41.1339
Pruned	.25	2892	3305	.37	.07	64.8363	42.8699	43.1611
Pruned	.5	4537	5185	.37	.06	73.4016	42.7807	43.4952
Unpruned	---	6182	7065	.31	.06	77.857	42.4688	43.4618



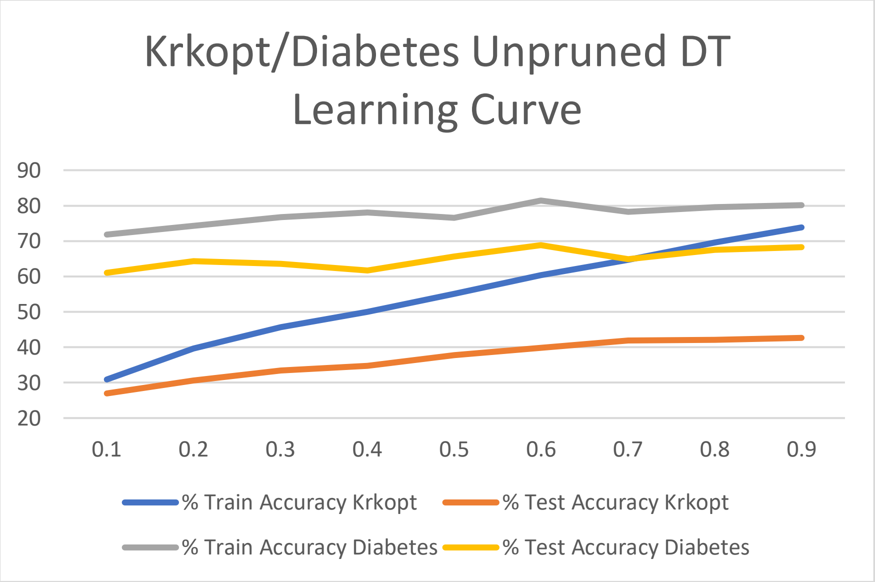
```
  a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   <-- classified as
  3   0   0   7   0   1  10  11   9  34  52  34  80 115 170 285  60   0 |   a = draw
  0   0   0   6   0   0   0   0   0   3   1   0   0   0   0   0   0   0 |   b = zero
  0   0   0   4   0   0   1   0   0   9   3   2   3   0   0   0   0   0 |   c = one
  0   0   0  77   0   1   5   0   0   6   2   2   0   0   0   0   0   0 |   d = two
  0   0   0  12   0   2   7   1   1   0   0   0   0   0   0   0   0   0 |   e = three
  0   0   0   4   0  21   5  12   2   6   0   0   5   0   1   0   0   0 |   f = four
  0   0   0  14   0   6  51  33   6  19  13   3   7   8   0   0   0   0 |   g = five
  1   0   0   2   0   0  13  88   6  16  14  11  10   5   5   3   0   0 |   h = six
  0   0   0   3   0   0  10  14  31  43  47  16  26  13  25  15   3   0 |   i = seven
  3   0   0   0   0   0  10  11  18 200  62  36  46  20  25  17   2   0 |   j = eight
  1   0   0   0   0   0   6   3  13  56 215  67  68  63  42  16   6   0 |   k = nine
  4   0   0   0   0   0   6   1   4  39  65 164 137  98  85  55   2   0 |   l = ten
  5   0   0   0   0   0   1   3   2  27  25  57 293 178 180 119   3   0 |   m = eleven
  8   0   0   0   0   0   2   3   1  11  39  30 103 447 320 212  16   0 |   n = twelve
  1   0   0   0   0   0   2   0   1   3   6  22  37 176 591 450  39   0 |   o = thirteen
  3   0   0   0   0   0   0   0   0   0   0  11  17  33 212 992 160   0 |   p = fourteen
  0   0   0   0   0   0   0   0   0   0   0   3   6   3  30 439 223   1 |   q = fifteen
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  53  58   0 |   r = sixteen
```

A variety of output features in the decision tree results become evident. Some phenomena to note are the trees are very leaf-heavy and, as shown in the confusion matrix, most misclassifications of the endgame refer to draw. These can be explained by the fact that the attributes are nominal which makes the model aggregate them and allows for noisy combinations. The relative information gain from a single attribute is trivial, so these trees become very wide horizontally. Thus, this lets the draw misclassification manifest itself in a myriad of leaves, instead of being caught in a linear manner down the tree, which is why the effect is prevalent. It is also important to point out that overfitting is not taking control of this classification since accuracies improve across the board, no matter the case, with correspondingly lower pruning levels.

Prune State	CF	Leaves	Tree Size	Train Time	Test Time	% Training	% Testing	% CV
Pruned	.01	3	5	.323	.01	77.8502	67.5325	73.9414
Pruned	.05	3	5	.24	.01	77.8502	67.5326	75.57
Pruned	.1	3	5	.23	.01	77.8503	67.5325	75.2443
Pruned	.25	13	15	.24	.01	80.78178	68.8312	75.2443
Pruned	.5	13	25	.25	.01	80.7819	68.8312	74.7557
Unpruned	---	13	25	.24	.01	80.7819	68.8312	75.43

```
   a   b   <-- classified as
 355  49 |   a = tested_negative
 111  99 |   b = tested_positive
```

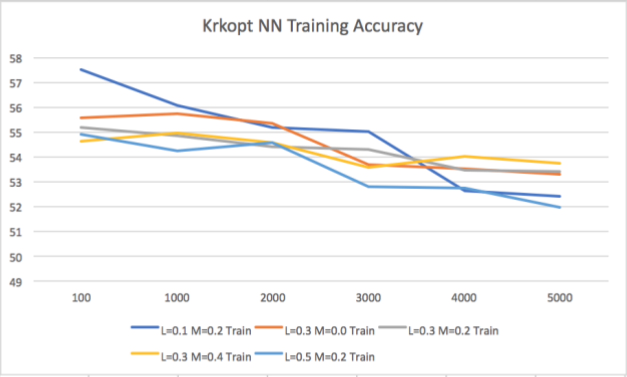
While this classification is not as extreme with respect to differences created by pruning, it does follow a similar path of higher error being associated with more aggressive pruning for testing and training. There is sort of a paradox that we can look at in accordance with the tree size in relation to the accuracy. While the extremely small size does correlate to a decently fit model for this dataset, pointing out that there may be a considerable number of irrelevant attributes, it is interesting that the error can be dually seen as high given how certain the model seems to be behaving and the fact that there aren't a large amount of instances. There seems to be a level of uncertainty under the hood in the data because otherwise the algorithm would be performing near perfection, all things considered. An interesting part of this data is the fact that more B's were misclassified as A's than correctly classified. Diabetes patients are human and, as arises, are highly case by case which points to the conclusion that these "WHO"-endorsed criteria are not the whole picture when it comes to diagnosis.

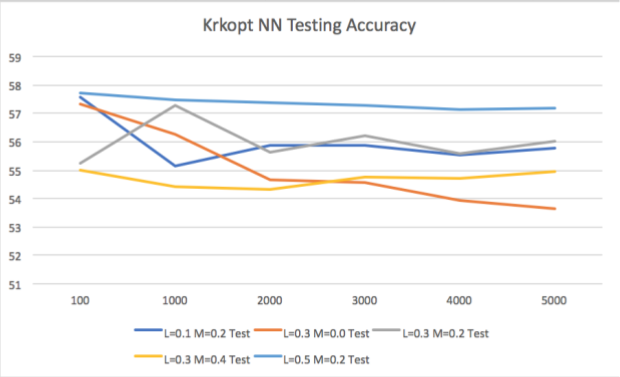


Note: The Krkopt learning curve testing rate stagnated, indicating little relation between partitioning and accuracy.

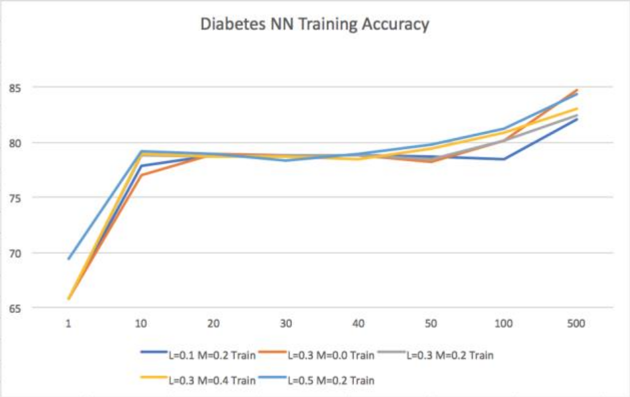
Note: The Diabetes learning curve is similarly stagnant, hinting at no instance-accuracy relationship.

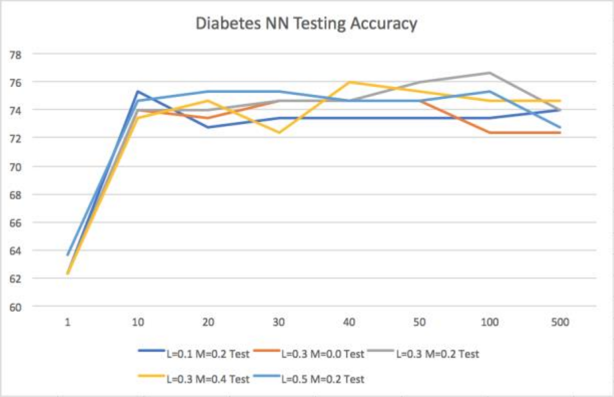
Neural Network: Multilayer Perceptron



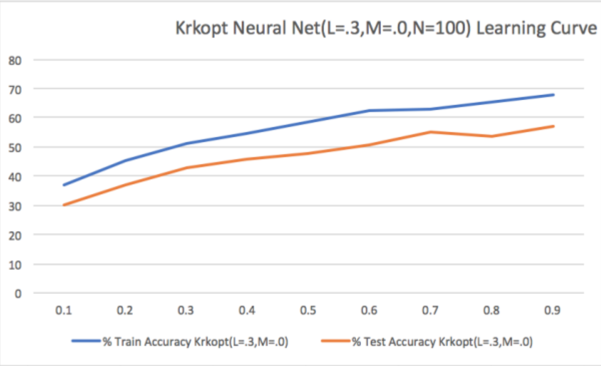


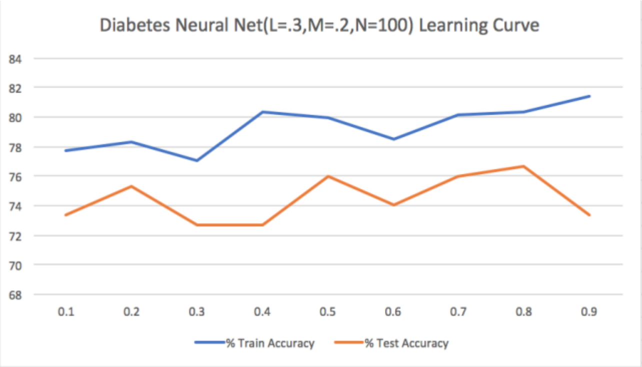
For this modeling experiment, Krkopt was run with varying learning rates and momentums, along with epochs ranging from 100-5000. The Neural Net is arguably one of the more dynamic models in a general sense, which is an important notion to adopt when considering running many datasets with different underpinnings. Since they can delineate many functions, overfitting can be a side effect. This can be seen in a direct context as the graph fits a linearly negative slope as the epochs (training times) increase with respect to the testing data. One of the most interesting relationships I have seen in this report is the dubious nature of the training/testing relationship, as the concept would suggest that the training data have the opposite trend line. This is hypothetically due to some stipulation of the bias/variance tradeoff, but I don't know. The learning curve parameter also seems to have a slight negative effect on the training accuracy. There were times, however, when the testing accuracy was exceptional when stacked up against a DT classification.





The Diabetes Neural Net analysis incorporated similar learning and momentum rates, as well as epochs ranging from 1-500. I included the former just to visualize the instantaneous result from altering the epochs on the order of 10. One feature to highlight is how well the testing accuracy seemed to perform in a relative sense, likely since the model at hand is robust and works well with a binary classification, at least more so than the other dataset we are analyzing. There is a slight indication of overfitting towards the end, looking in specific at the last epoch level that heightened the training accuracy while having a negative effect on the testing accuracy. The optimal training size was hit at some point in the curve, most likely.

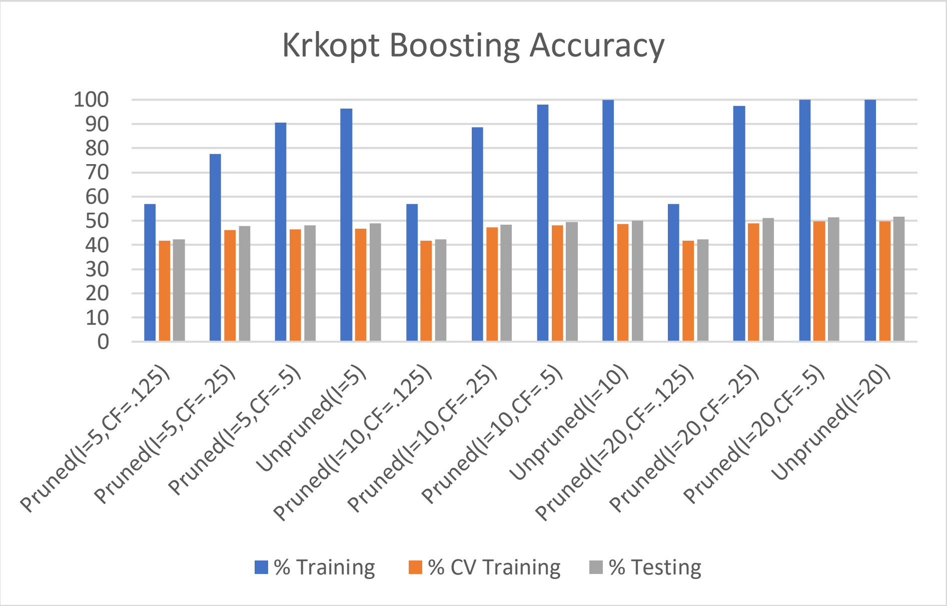




Note: Tweaking the Krkopt instance size had a strong positive effect on the model, as the training and testing accuracies both saw a boost on the same level, roughly speaking.

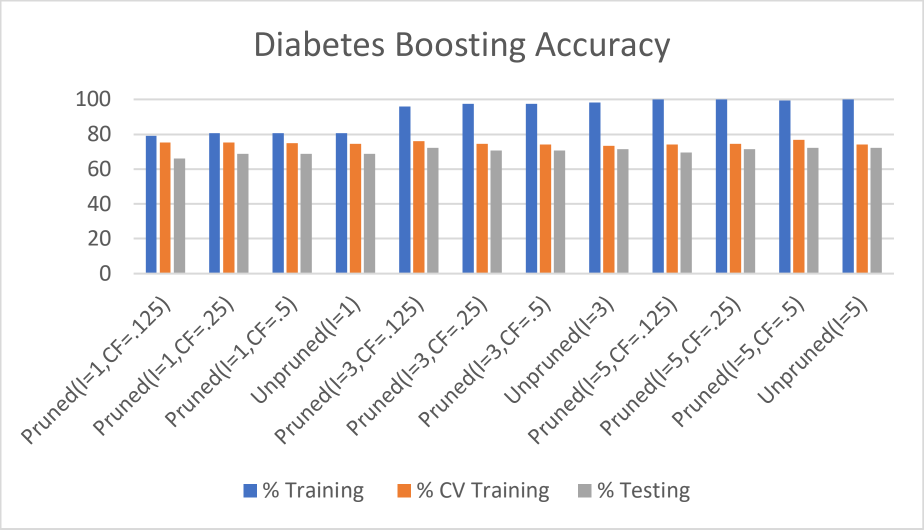
Note: Interest enough, fitting a general model to these Diabetes lines would be a challenge, pointing out the sporadic nature of the model as well as defining through the image how partitioning data leads to untraceable variables such as vital attributes being left out.

Boosting: AdaboostM1

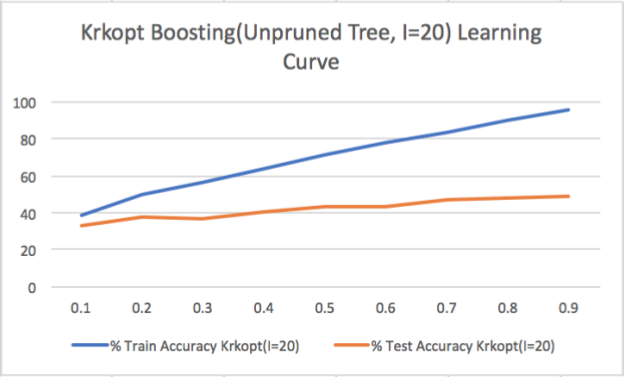


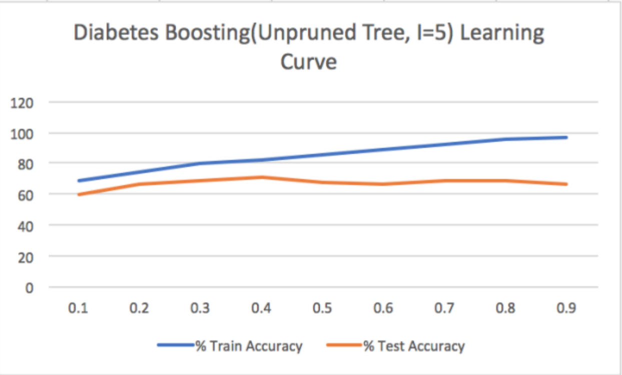
The Boosting algorithm took on the relative run times indicative of a weak classifier, which can be easily referenced in my attached .xlsx file. Similar to the decision tree, the classifier takes much more time to train the data than to test it.

The Krkopt Boosting improved the testing accuracy of the dataset by anywhere from 10-20% given the specific number of iterations and corresponding confidence factors. The most accurate the model ever became was at the 20-iteration level with higher confidence factors in the base DT algorithm. One can easily point out the overfitting that occurred when looking at the sharp increase in the training accuracy compared against how it actually performed. The classifier reached in the neighborhood of 99-100% accuracy on some later iterations, which proves that the classifier was too caught up in the noise.



As in the other model comparison, the Diabetes Boosting classifier did a better job than the Decision Tree at testing accuracy, somewhere on the order of 5-10%, increasing linearly with higher iterations and less pruning. Something to point out, however, is the fact that the accuracy plateaued, and perhaps even peaked, at a middle-valued iteration value and low level of pruning, despite the general linear upward trend. The model started to overfit the data as it was given more freedom, and as we have seen in past examples, this leads to a heightened sense of training accuracy, but not rightfully so. I want to point out the level of abstruseness that can exist even in the simplest of binary classifications. The task is to classify the patient as pos. or neg. diabetes-ridden, but the parameters struggle to deal with that nevertheless, making for a very interesting dataset.

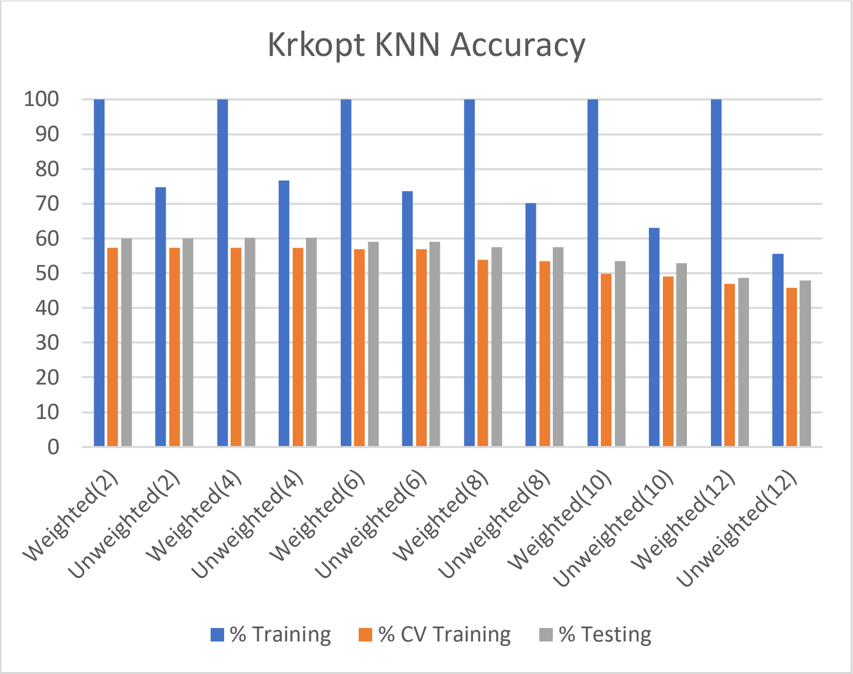




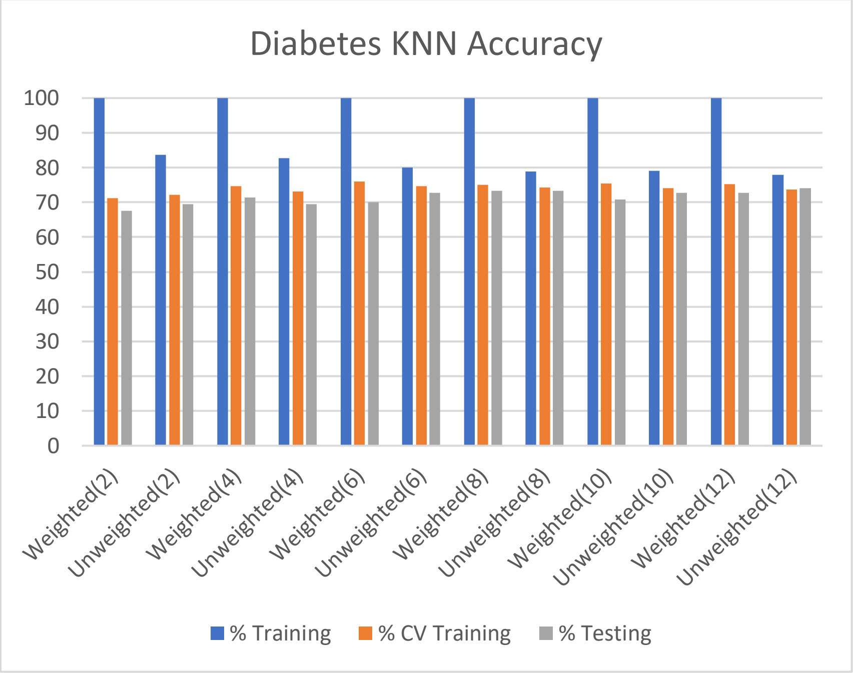
Note: The change in Krkopt training accuracy is cause for overfitting, since it clearly wasn't reciprocated with the testing data. Although they are both on the uptrend, they aren't proportional.

Note: A robust model, the Diabetes learning curve indicates a beginning fair to good accuracy level that didn't become dynamic through the instance size iterations.

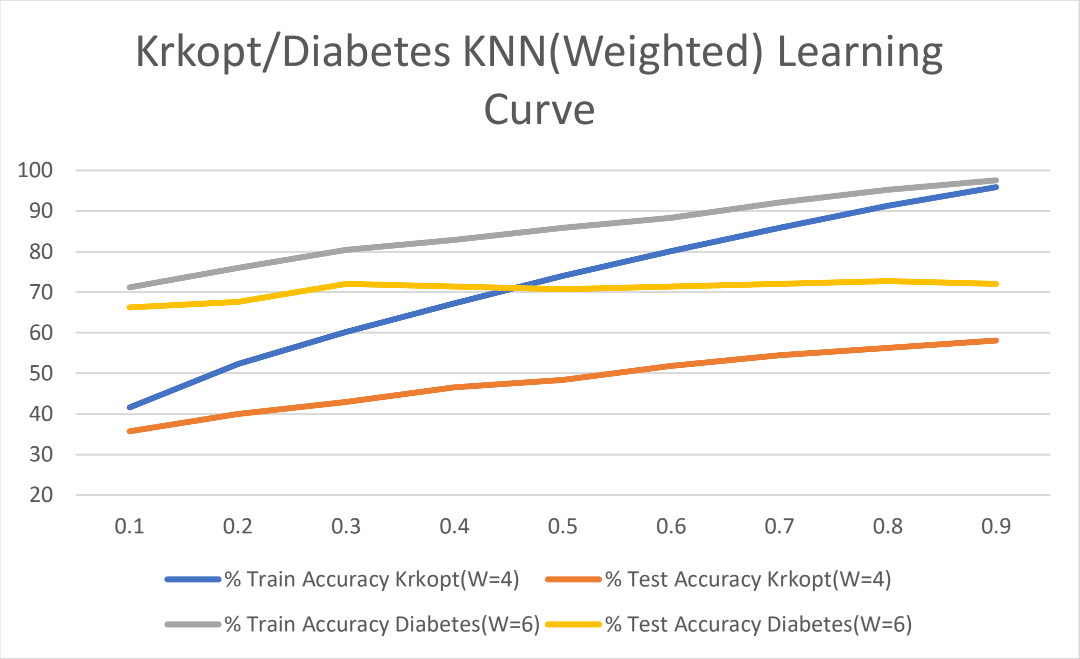
K-NN: IBK



On the nature of K-Nearest Neighbor characteristics, it is known to be true that the algorithm deals well with filtering out noise in data, which is the first quality of this output that catches my eye. Comparing with what we have seen so far, Krkopt's KNN is outperforming testing accuracies of Boosting by more than 20% in some cases, which we already said was performing better than DT's alone. The essence of this difference can be highlighted with respect to the behavior of KNN's classification, in that they localize the model to the nearest instance to agree on an output, rather than taking the data holistically and generalizing from there.



As far as the diabetes dataset is concerned, the algorithm performs well, perhaps slightly better than DT's and Boosting, and about as well as Neural Nets. A potential reasoning as to why the contrast for the other dataset might have been more stark is that while KNN localizes the nearest variables, diabetes has been shown through this data to be a fairly, albeit binary, whimsical condition to classify so looking at near neighbors might be misleading. That's not to say this model isn't working, as it seems to perform well stacked up against others, just not VERY well.

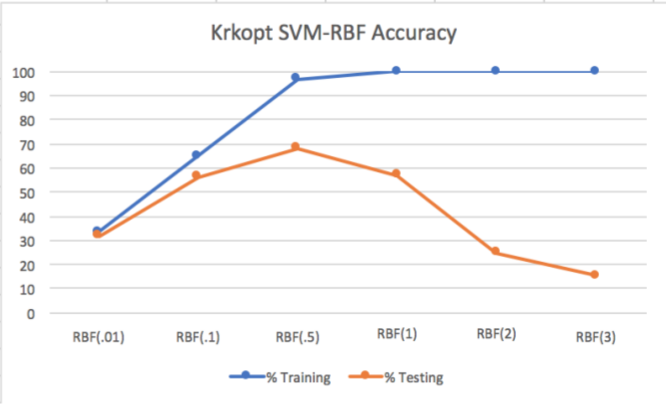


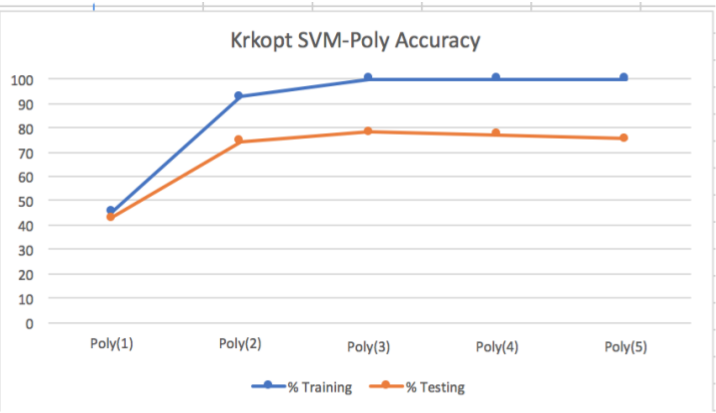
Note: For both types of accuracies with Krkopt, especially training, a higher instance base leads to better potential neighbors, and therefore, a better classification as shown.

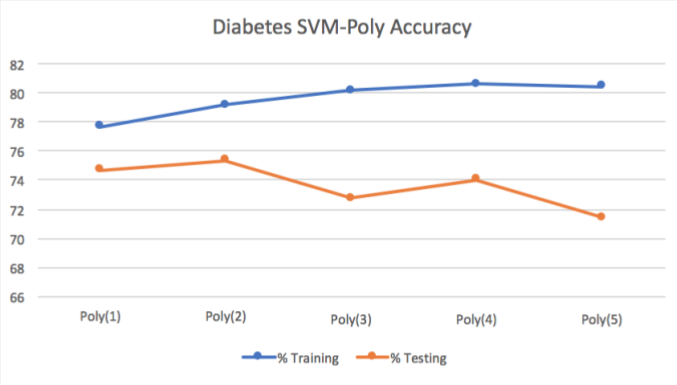
Note: As discussed in the Diabetes-specific analysis for KNN, the neighboring classification, no matter the size, seems to have a plateau effect on the model, so it stagnates.

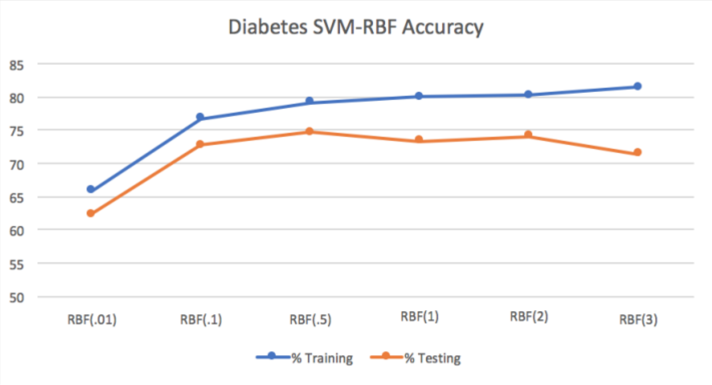
Looking at the runtimes (referencing the excel sheet attached in the file), KNN is following its expected learning approach, the lazy learner. The train times are taking little to no time, whereas the testing times are taking relatively much longer, and increase at a slight rate with the amount of neighbors taken into account.

Support Vector Machines: libSVM


Krkopt SVM-RBF Accuracy

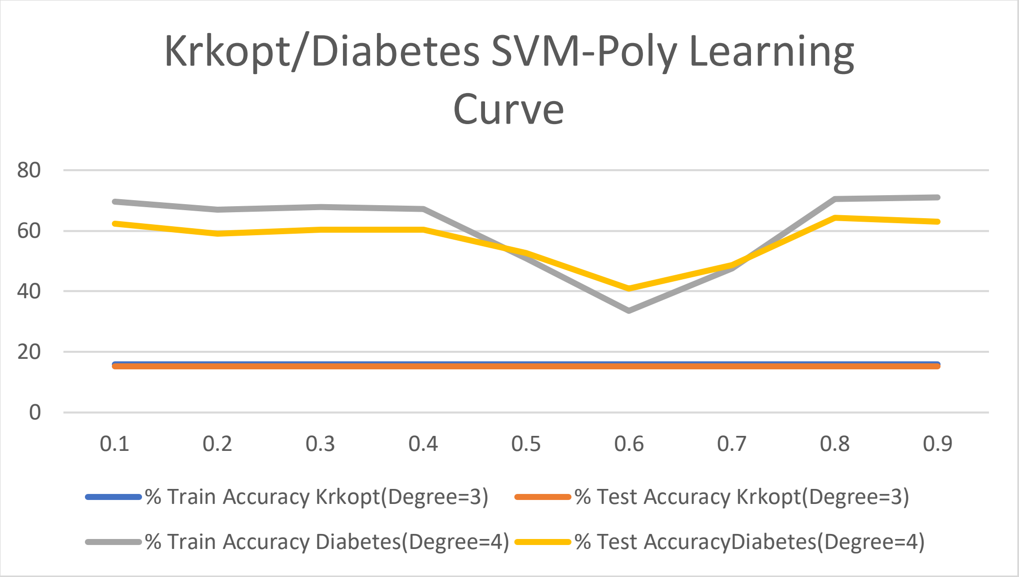

Krkopt SVM-Poly Accuracy


Diabetes SVM-Poly Accuracy


Diabetes SVM-RBF Accuracy

SVM’s are known to be resistant to overfitting, however there is some evidence shown against that point with the sudden drop-off in the RBF testing accuracy curve with some of the higher gamma values. A very high gamma value (3) in fact plummeted the testing accuracy levels to almost nothing. Notably, a higher value of gamma leads to larger bias towards data which could be the source of the overfitting. However, as with any function, there is a sweet spot (in this case .5) in regards to a gamma value that leads to the most accurate results because allowing the algorithm to block out noise could be and is beneficial in the broader scope of the model.

It is evident with both cases that increasing the exponent value in the Polynomial fit leads to improvements in the training and testing of the model. There is a tradeoff with these parameters in that, with SVM, the training and testing times of building the classification with these higher values to yield higher results is not exactly cost efficient, which is another variable to consider. Despite this relationship there is an interesting dip in the Diabetes Poly graph in the middle of the run, which possibly points to the fact that SVM might not need more freedom past a certain threshold to fit its best model



Note: Tuning the poly data size did not seem to influence the Krkopt learning curve because given that degree it essentially did not matter.

Note: The Diabetes poly learning curve had a large dip when a specific amount of data was passed in, which could indicate that certain outlier attributes certainly might have been left out at a whim to concentrate the model on irrelevant information.

Conclusion:

Krkopt: All the models examined had their interesting qualities, but in terms of optimization given a specific set of parameters, referencing all the learning curves in total, SVM had the potential to test the most accurately on the RBF side, and also performed VERY well with the optimal Polynomial degree and more instances fed in. Oddly enough the learning curve argues this point with a very low testing accuracy, so, on a broader scope, the Neural Net had arguably the most consistently good results of the 5 algorithms. Depending on the criteria we're looking for (reliability/consistency vs. optimality), different classifications categorize these winner, respectively. I would place the SVM first and Neural Net close behind for this dataset.

Diabetes: Once again comparing trends with the learning curves, none of the algorithms necessarily performed poorly in regards to testing, but there were some better than others. Throughout the 5, I would conclude based on the analysis and curve shape/gravity that Neural Nets performed best at classifying this data. I think looking at this as an aggregate is very interesting and imperative to understanding the framework for these algorithms and how even binary-valued targets can be misconstrued. The fact remains: Neural Nets have proven to be a robust and abstractly reliable model all the same.

***** For all the learning curves, I generated them after picking and choosing the optimal parameters for each algorithm.**