Assignment 1: Supervised Learning CS 7641

Connor Beveridge (#902451314)

1. Introduction

Some concepts are much more understandable to humans than others, which to a computer (given current state of algorithms) they may be the same or even harder. This difference is drastic as a computer can theoretically compute anything that is computable, humans not so much. People tend to understand concepts that are more applicable to our spatial, survival-based psychology. One of these concepts many learn very easily is the concept that determines who has won in a tic-tac-toe game. On the other end of the spectrum, we do not understand complex mathematical probabilistic predictions near as easily. In this analysis, I will compare 2 datasets that fairly represent the opposite ends of this spectrum of human understanding among different supervised learning algorithms when training instances are limited (~600). As "human-easy" (HE) concepts often are, the tic-tac-toe dataset is categorical, and as "human-har" (HH) problems often are, the heart disease dataset it mostly numeric (continuous). This analysis provide insight into how machine learning (ML) algorithms learn HE concepts compares to that of more HH ones. Cognitive Psychological comparisons are also discussed.

2. Data Introduction: Two Spectrum-Spanning Datasets

a. Tic-Tac-Toe Endgame (find in "tic-tac-toe.csv")

A tic-tac-toe dataset is used as a simple representative HE problem. Many children can easily learn the concept that determines victory from examples alone. The tic-tac-toe dataset is categorical and has an obvious (to a person who has played tic-tac-toe) true model of a disjunction of conjunctions (3 x's could be here or there or there etc.). This true model is simply set of rules of the spatial organization of sets of 1 letter, x's, with a determined set of combinations (8 different possibilities for 3 x's in a row) of spatial organizations indicating a win. Other, more complicated rule sets could also be considered true models. This analysis assumes all the data are truly reachable endgame states. Each instance is a gameboard state. The target values are + or – indicating victory for x. Each feature is the state of one of the 9 squares with values of x, o, or b for blank.

It will be interesting to know which types of algorithms will represent these concepts and learn similar to humans. This true model for this dataset is much easier for a person to understand than the true models for predicting heart disease which depend on a probabilistic appearance of numerically measured symptoms in relation to certain personal characteristics.

b. Heart Disease (find in "heart.csv")

The heart-disease dataset is largely composed of continuous data from physiological measurements but also has a couple nominal and a couple ordinal features. Each instance is a person, their characteristics such as age and physiological measurements such as cholesterol level. The instances have classification of either having the presence of heart disease or not. The target concept here is a combination of many probabilistic factors and is not intuitively understandable to a person like tic-tac-toe is.

c. An interesting Comparison

To control for the comparison of interest, categorical HE vs. continuous HH target concepts, these datasets are chosen as they are similar in other significant ways. Both datasets have a similar number

of given features (9 vs. 13) and training instances (~600). These datasets both have binary classifications. Both sets are also close to exactly balanced in the number of instances of each class. The basic, unbalanced accuracy score is used for scoring as the data is split almost perfectly in half for both datasets between positive and negative target values. These datasets have no missing data. It will be interesting to see if and which algorithms do better on the conceptually complex heart disease task compared to the simple AND/OR rules of the tic-tac-toe dataset. This will be useful to know in real-world execution when deciding which types of algorithms to use. The most real-world limiting factor for most learning algorithms is the number of available training instances (Mitchell, 1997). It will be interesting to compare how these 2 datasets compare for a small vs. regular number training instances. This will be helpful to know in real-world algorithm selection.

d. Preprocessing

The tic-tac-toe data was encoded numerically for use with one-hot encoding which was done with the *pandas* package. This encoding was selected to prevent any-kind of ordinal attribution to the data whose values are x's, o's, and b's for blanks. The heart dataset was already all numeric, however one feature had multiple options which were nominal in quality. This feature was one hot encoded. The rest were not preprocessed. This was "chest pain type". I wanted to analyze this problem with limited training instances (~600), which is still enough for powerful learning. Thus, I have taken 200 instances out of each dataset randomly for use purely as a testing set with no influence on training for the final assessment. This size will also ensure the final accuracy scoring is statistically significant.

3. Decision Tree Classifier

The decision tree classifier from sci-kitlearn is used to build and execute the decision tree algorithm on both datasets. The 'Gini' information gain criterion was used for the tic-tac-toe dataset not only because it was selected by grid search as will be described next, but it also is computationally less expensive. The decision tree algorithm used does prepruning by issuing a maximum number of layers for the tree's depth, but only in the improved version. 15-fold cross validation (CV) is used for both sets in all algorithms in order to get smooth learning curves and significant results. This will provide better insight into the parameters at play and allow better identification of overfitting. Just as with many AI algorithms, some sort of DT algorithm likely exists as a cognitive architecture in the brain and could be investigated with cognitive neuroscience methods. It will be interesting to compare human learning and machine learnings with these "human easy" and HH concepts.

a. Optimal Learning with training data

GridsearchCV of Sci-Kit Learn was used to find the optimal parameters for each dataset with all available instances used in training (~600), with no pruning to start. These parameters included whether to use 'Gini' or Entropy for the information gain criterion, the minimum number of samples split, and whether to balance the classes or not. Based on my understanding of the DT algorithm, the min number of samples split will influence any noise in the data, possibly by making sure individual or small groups of incorrect values don't create new nodes however if the data is not noisy or . This may help identify noisy datasets via high values for the min split parameter. I was interested in knowing if GridSearch would decide to balance the classes even though they

were already balanced. Balancing the classes with weights is therefore computationally wasteful. The max depth found by Gridsearch is not necessarily

i. **Tic-Tact-Toe Dataset**
Gridsearch found that 'Gigi', 2 samples (min number of splits), and non-balanced were the best parameters. The mean 15-fold CV accuracy on test data with ~600 training instances was 95.5%

ii. **Heart Disease Dataset**
Gridsearch found that 'Entropy', 2 samples (min number of splits), and non-balanced were the best parameters. The mean 15-fold CV accuracy on test data with ~600 training instances was 98.5%. Splitting on at least 4 samples decreases overfitting by not classifying based just a 1 instance leaf, so this indicates there is not much noise or very similar instances with different classes.

b. Learning Curve

Given the optimal parameters of the previous section, a learning curve is created to analyze the learning at a deeper level. **Fig 1** shows the accuracy (mean of CV) of the non-pruned classifier with Tic-Tac-Toe and Heart Disease data. Both sets appear close to convergence.

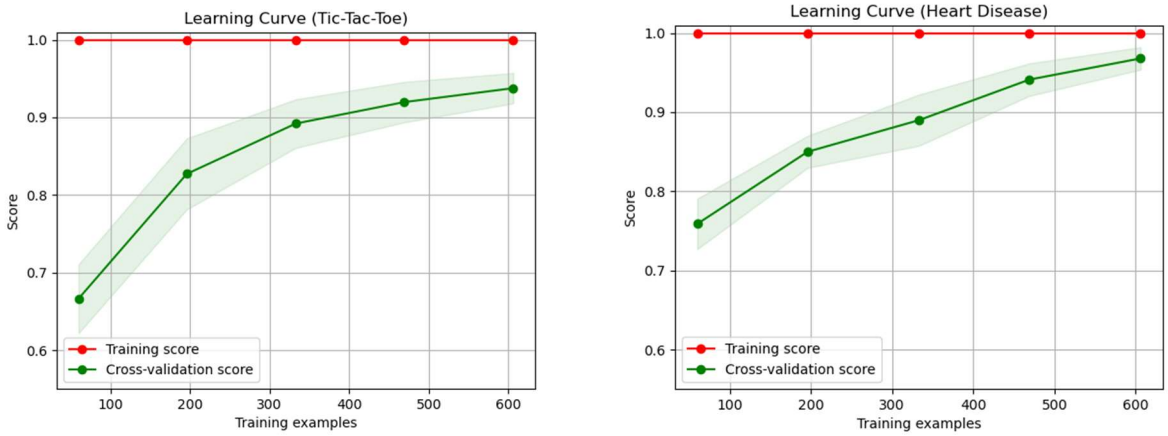

FIG 1. Learning curves, No pruning. Standard deviation is shown along lines.

c. Model Complexity Analysis

In the initial analysis pre-pruning was not done. Its use was tested, and the decreasing complexity of the tree is controlled with the "max-depth" hyperparameter. More pruning leads to a less complex model. As expected for the TTT data assuming correctly label data, there is not apparent overfitting. Both datasets did best on the validation sets with no pruning.
Given the concept complexity (from a human perspective) and potential for noise in the continuous physiological measurements of the HD data, I would much more expect to see overfitting, but its accuracy plateaued with decreased pruning. Model complexity curves are shown in **FIG 2**.

d. Results

i. Tic-Tac Toe Endgame

The tic-tac-toe best model (no prepruning, no min split, Grid Searched with max depth discussed) only achieved 95.5%. Decision trees are biased towards smaller and less complex models but the most accurate model (can be seen in decision_tree.png) had, interestingly, the same number of node layers (8) as disjunctions of conjunctions in the target concept. In other words, there are also 8

different ways or spatial orientations to win for x (3 in a row). However, it still modeled the tic-tac-toe win concept in a way most humans would not.

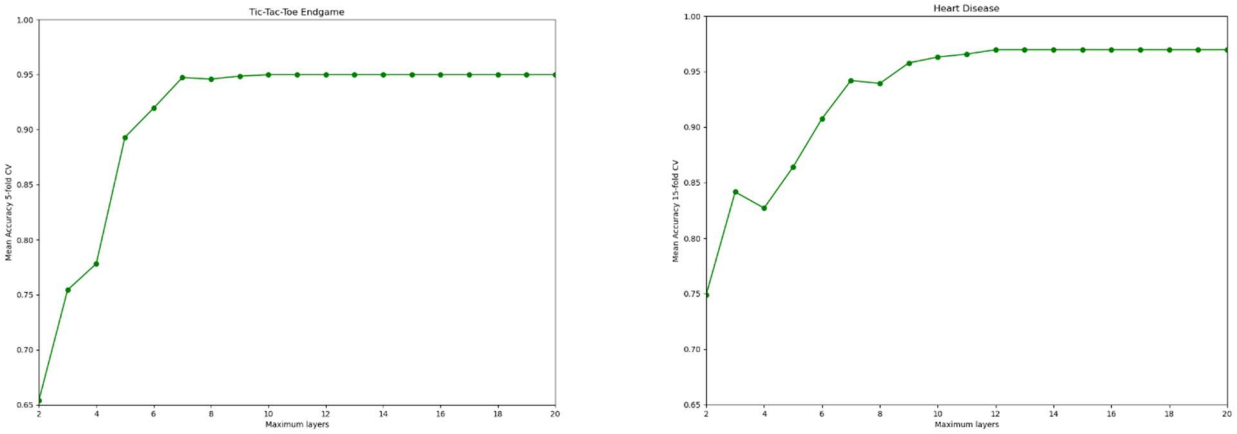

Figure 2. Model Complexity. Prepruning depth and CV score. Optimal max depth for the TTT dataset was 10 layers (But only used 9 with 8 node layers in best model), but converged around 7. This is interesting as there are nine spots on the TTT board and 8 possible ways to win. Any information about the endgame board can be obtained with just 8 node layers. The optimal max depth for the heart-disease data is 12 with mean 15-fold CV score.

It took a "human problem" with a target concept that almost all humans would understand easily and created an equally valid (but not a valid true target concept) concept form and representation structure that would not exist as a cognitive structure in any person's mind. When a person checks a board for a win for x, they almost always look for 3 xs in one of the 8 possible orientations available. This learned DT algorithm however checks each square one by one (which a human would not do) for a certain letter in an dynamic order determined by the tree. It even checks squares for o's. Take for instance a positive sloping diagonal win for o instead of x. The "best" representation of this algorithm, even though it is looking for a win for x, actually checks each of these 3 squares one by one for an o instead of an x. Humans are biased toward focusing on the x's when determining if x won but the DT algorithm didn't have a bias towards focusing on either one.

If there are no errors in the data when it comes to values and classifications, the there should be no overfitting for the data. Perfect data would only be better and better represented with complex models because any correct/noiseless instance will be a win or not according to an exact (or set of) conjunction of disjunctions. Each possibility and thus the target concept could be represented perfectly with a decision tree. Pre-pruning would decrease training accuracy but may improve CV accuracy. This will be investigated in the next section.

ii. Heart-Disease

Amazingly, the best model (no prepruning, no min split) for the heart-disease concept learned on ~600 instances achieved 97.5% accuracy on the 200 instances of test data. Given more data I believe it would reach 100%. Due to the complex nature of the target concept I expected the tic-tac toe data to do better. This may be because of the increased flexibility of the DT algorithm for continues data as in it can choose many numeric values of features to split on vs. the few values for TTT categorical data. With no Pre-pruning, DT models the training data perfect every time, and more training examples only

fit the concept better.

e. Assessment and improvements

These results are against my expectations. I would think that the HE problem would be easily solvable by this algorithm. Both classifiers could be improved with boosting which will be explored in section 5.

i. Tic-Tact-Toe Endgame

After further deliberation and analysis, I have come to an assessment on why the HE tic-tac-toe task did worse in this case. I think with many more examples it would reach a perfect target concept. I looked at the decision tree itself (decision_tee.png) when determining why certain test instances failed. Those that failed had not been seen in the test set and were rare outlier endgame states than only happen if someone is bad at the game. If all possible instance combinations were seen in the training set, it would be perfect, but this would not be impressive. I think this DT algorithm would have done much better if the sklearn version would be able to handle categorical data so it doesn't have to be one-hot encoded. That way, nodes could be split three ways instead of 2 (true and false). Many of the splits in the current version don't provide as much information. For instance, if the node goes false for the center square as an x, that doesn't tell me if it is blank or an O. Knowing this at every node would provide much more information gain. It is surprising that even a child could learn the winning rules to the game perfectly being shown a relatively small amount of examples without any prior knowledge of the game, but the DT algorithm could not learn it perfect after a large number of examples. However, humans are skilled at spatial pattern identification and the DT is not as talented in this respect. Humans can see all features (playable spot) at once in their real spatial layout and DT looks at each spot individually focusing on one of the three values (x, o, b) at a time.

ii. Heart-Disease

In was interesting to see that DT heat-disease data did perfect after ~600 examples. This data likely had minimal noise so did not overfit. A person might take years trying to learn the target concept for this set and not perfect it, while DT has made a near perfect concept in seconds. The results of these two datasets reveal the drastic cognitive differences between general human learning and machine learning with DT, even though human brain computations likely use some form of DT in decision learning.

4. Neural Networks

When comparing humans and ML algorithms, I would predict that Artificial Neural Networks (ANN) show the most similarity as human brains are neural networks and ANNs were neurobiologically-inspired by our own brains (Chen, Lin, Kung, Chung, & Yen, 2019). The logistic activation function is used to make this network more like a brain neural network. A large number of max iterations allowed is set at 750, in order to get converged results, but not too many as to take a long time to train. An initial single hidden layer of 3 hidden units is initially used. The stochastic gradient descent solver is used not only because I understand it best, but also because it allows for the use of momentum.

a. Optimal Learning with training data

GridsearchCV of Sci-Kit Learn was used to find the optimal learning rate and momentum parameter for each dataset with all available instances used in training (~600) with 750 default max iterations for convergence. Optimal learning weights are needed to 'descend' into a global instead of local minimum in the hypothesis space (Mitchell, 1997).

For the TTT dataset, Gridsearch found that '.0025' was the best learning rate, momentum as '1', and accuracy was 96%. For the HD dataset, Gridsearch found that '.0008' was the best learning rate, momentum as '.75', but accuracy was only 66.5%.

b. Model Complexity Analysis

Poor results on the initial implementation for HD are likely due to the fact only 3 hidden unit was used, leading to underfitting and poor generalization. More complexity via more hidden units should provide enough expressive power to better model the target concepts. The TTT concept should be hard to model perfectly with just three units as these units alone would not be expressive enough separating them, there are just too many similar board states that are different win results. Individual inputs alone do not provide much information on win state, but relationships among them do. Based on my own understand of how Neural networks work, I can imagine 1 layer of 8 hidden units to perform perfectly, each representing 1 of the 8 possible winning 'three-x--in-a-row' states. The HD concept seemed like it would do better than TTT with just three units as they can better handle the probabilistic combinations of features which independently do provide significant information on if someone has heart disease. For instance, they would be more likely to have HD if they have high cholesterol. Alternatively, you cannot say X would be more likely to have won a TTT game if the top-right board is an X. However, it did not do well with just three, suggesting a greater relationship among the features. The results are shown in **Fig 3** with the optimal learning rates and momentum from the previous section used.

c. Learning Curve

Fig 4 shows the accuracy (mean of CV) of the optimized learner for both datasets by number of iterations. With enough examples I do think the TTT data would converge on the 8-unit hidden layer just mentioned. The optimal (from this analysis) learner's accuracy for iteration count is shown in Fig 9. The optimal number of units was 11 for the TTT concept and 6 for HD.

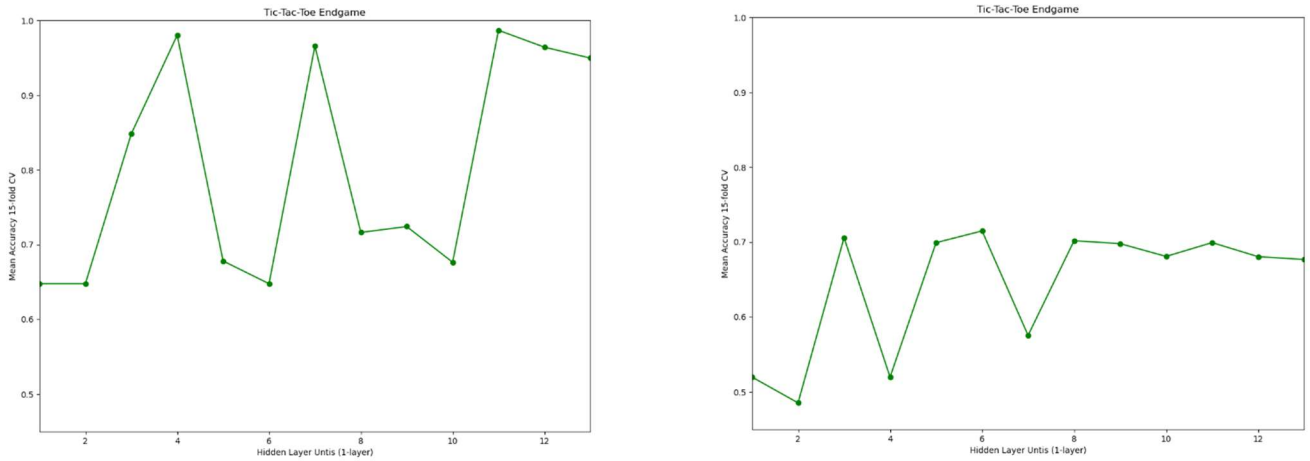

Fig 3 | One layer of hidden units. Y-axis is mean CV accuracy.

a. Assessment and improvements

I was surprised at the iterative speed at which these algorithms converged. I was also surprised that the HD set did not do well at all. This may suggest there are relationships among relationships that would need to be modeled with 2 hidden layers and this may improve accuracy dramatically. ANNs might be susceptible to scalability issued as discussed more in section 7. Scaling the widely ranging continuous HD data would likely improve accuracy.

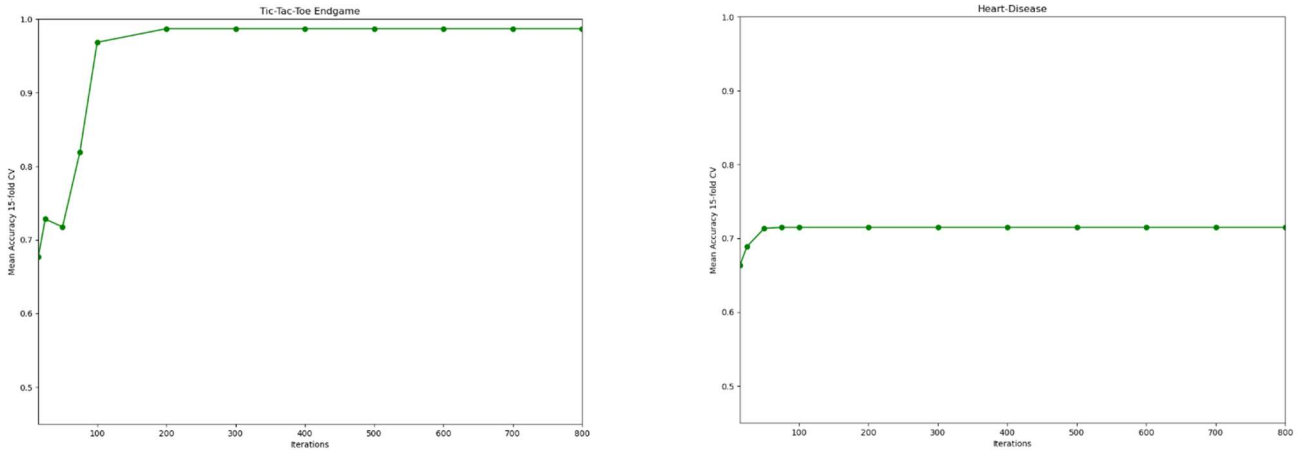

Figure 4 | Iterative Learning Curve

5. Boosting: AdaBoost Decision trees

The optimal DT from section 3 is boosted with several trees. However, the amount of pruning will be reanalyzed as shorter trees can provide similar accuracy with much less training time. I predict that this will improve the performance on the TTT data as it will help focus on those outliers that were causing inaccuracy in a single tree. I also predict boosting will significantly improve the learning curve of the HD concept. Sci-Kitlearn's AdaBoost Classifier is used. 'Gini' is used for both algorithms as it computationally for efficient.

a. Learning Curve

In order to explore use of "weak" learners and reduce training times, I have used heavy pre-pruning to reduce each tree's smaller, 5-layer trees. This will be interesting to compare to an optimal long decision tee from earlier with no pruning. A default max number of trees of 50 is used. The results for both datasets can be seen in **Fig 5**. Compared with the non-boosted tree in figure 1, the boosted version showed significant improvement on the TTT concept, even with much smaller trees. This must be because the boosting algorithm forces the trees to focus on the hard problems, which are the rare in game scenarios which are often classified wrong by the non-boosted tree. Not much improvement was seen in the HD dataset. The boosted algorithm did not a have a significant change compared to a DT on the HD concept. This is likely because the data has less outliers and the trees used are half the size, limiting expressiveness.

b. Model Complexity Analysis and Assessment

No pruning will likely lead to a better accuracy, but only a small amount of accuracy for a longer time of actual learning. Model complexity analysis is done on the Boosted DT to determine the optimal maximum number of weak learners to ensemble. The results are not shown (but can be seen in graph_boost.png and graph_boost_heart.png) as they are uninteresting, for both sets the accuracy was essentially constant with increasing learners, However ,a max of 75 learners for TTT and 25 for HD was found to be slightly optimal. The more significant factor appears to be the amount of pruning done on each tree. This analysis has indicated less pruning results in more

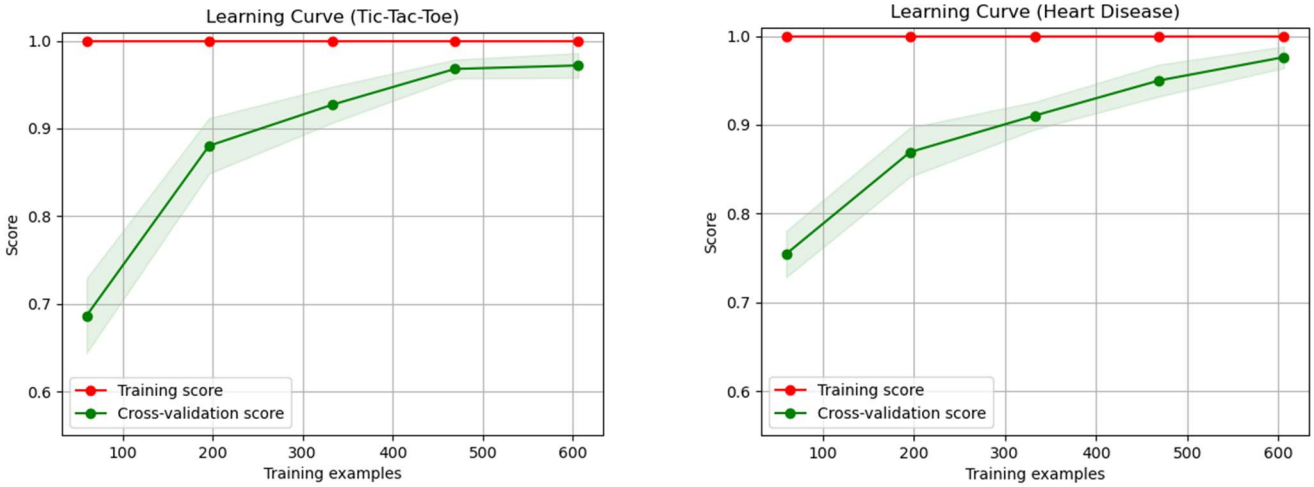

Figure 5 | Boosted decision tree – Y-axis is mean 15-fold CV

accuracy but takes a longer time, so would likely not be worth if for learners who are already doing well such as with the HD single DT training. The Results on the test set for the pruned tree boosted algorithm on TTT reached 98.5% compared to the 95.5% accuracy of the single, unpruned DT. The HD concept improved from the 97.5% accuracy from the single Tree to 99% with a max of 25 Boosted DTs.

6. Support Vector Machines (SVM)

SciKit learn's SVC is used.

a. Optimal Learning

GridsearchCV of Sci-Kit Learn was used to find the optimal kernel parameter for each dataset with all available instances used in training (~600). The different kernels tested were 'linear', 'polynomial', 'radial basis', and 'sigmoid'. 'Linear' was found to be the optimal kernel for both datasets. This makes sense because linear kernels are best for many features, of which the one-hot encoded TTT data has very many. Many of the continuous features of HD are separable linearly.

b. Learning Curve

Learning accuracy based on number of training instances can be seen in **Figure 6.** It was interesting that the TTT converged completely after only 200 training examples and the HD model basically converged at this point. This indicates SVM may be best for small amounts of training instances and for HE problems.

c. Model Complexity Analysis

The regularization hyperparameter 'C' determines how much "risk" is worth taking with higher values of C creating greater margins. Taking more risk in creating decision boundaries sounds bad but may fit the target concept better. The accuracy results for different values of C are not shown because they are not interesting as curves are essentially (especially TTT) constant, but can be seen in graph_SVM and graph_SVM_heart produced by the given code.

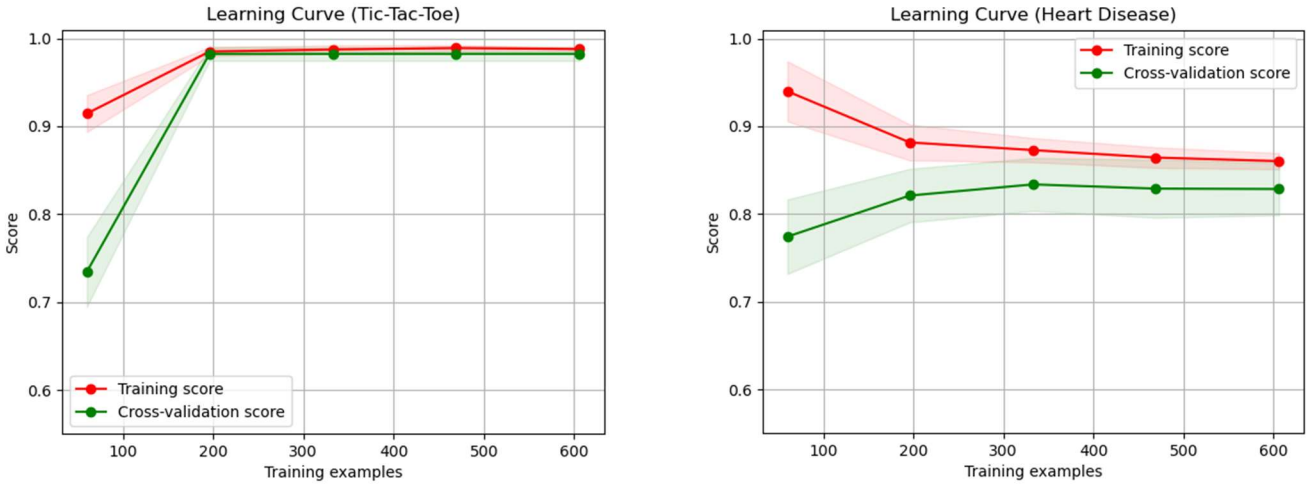

Figure 6 | SVM Learning curve.

d. Assessment and Improvements

These results suggest that SVM are good algorithms for HE problems, especially when data is limited, performing at 97% accuracy on the test set. The HH HD concept was not modeled well at best only doing 82.5% on the test set, however did do well with small numbers of training examples, but quickly converged. The HD concept may have been modeled better with a more specialized kernel created by a physician who knows a lot about heart disease. Again, scaling might have helped as well.

7. K-Nearest Neighbors

When contemplating k-nearest neighbors (KNN) and these datasets, I expect it to do well on the HE problem because there is not much distance between features in one hot encoding, and for it do worse on the HH heart disease concept. It seems it will fall victim to the curse-of-dimensionality as there are many continuous features that could have great distance but may be unhelpful in determining the presence of heart disease. The KNN classifier from sci-kitlearn is used for both datasets. The default k used during gridsearch is k = 5

a. Optimal Learning with training data (5-NN)

GridsearchCV of Sci-Kit Learn was used to find the optimal parameters for each dataset with all available instances used in training (~600). An initial grid search will be done on the weight and distance metric. The weight parameter will determine whether the overall distance is balanced so that nearest neighbors are weighted greater than farther ones. The distance parameter will search between Euclidean and Manhattan distance for optimal performance on the test set.

i. Tic-Tact-Toe Dataset

Gridsearch found that 'uniform' and Manhattan distance were the best parameters. Manhattan distance makes sense because the only distances between any two features is 0,1 in this one-hot encoded data (for Euclidean as well). The spatial distance between instances in this sense does not provide as much information as distant samples often are also wins and often near samples are actually loses to O. The mean 15-fold CV accuracy on test data with ~600 training instances was only 91.5%

ii. Heart Disease Dataset

Gridsearch found that 'distance' and Euclidian distance were the best parameters. Both of these selections make a lot of sense as this numeric data is conceptually valid in a 13-dimensional Euclidian space, and spatially closer neighbors should be much more probable to also have/have not have heart disease. The mean 15-fold CV accuracy on test data with ~600 training instances was 99%

b. Learning Curve

Given the optimal parameters of the previous section, a learning curve is created to analyze the learning at a deeper level. **Fig 7** shows the accuracy (mean of 15-fold CV) of the classifier with Tic-Tac-Toe and Heart Disease data. As expected, cv accuracy converges with the number of training examples.

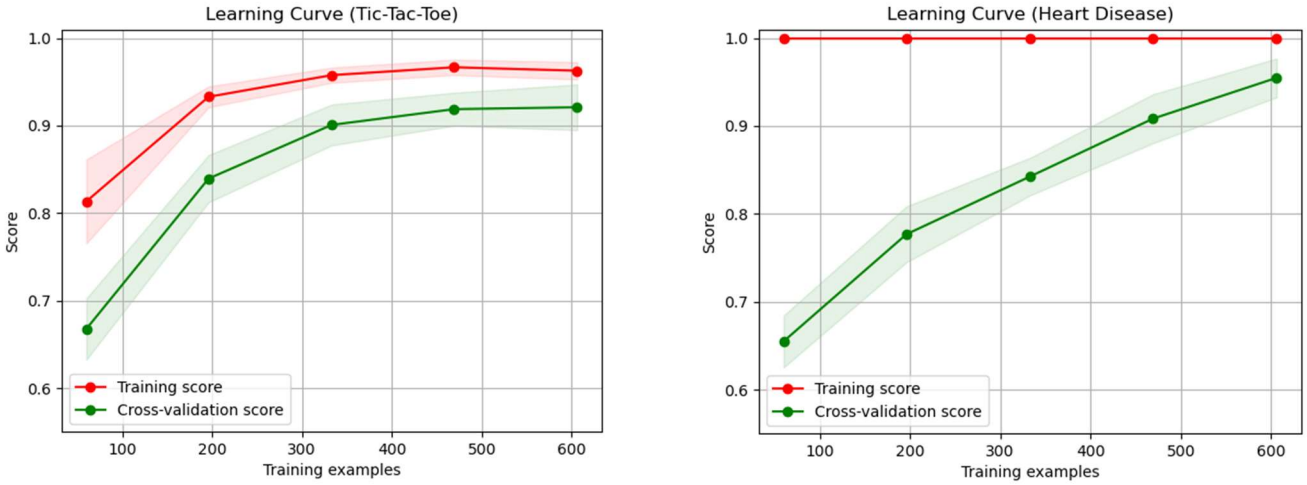

Figure 7. KNN learning curves. Y-axis is mean score for 15-fold CV. Standard deviation is shown along lines.

c. Results

KNN is biased in assuming that 'spatially' close instances will be likely the same class, however, this is not always the case.

i. Tic-tac-toe endgame

Distance as a measure for the "human easy" tic-tac-toe game did turn out to be a poor metric to determine the target class, converging at around 92%. This is obviously because many very similar games are actually different outcomes. For instance, O could play for the win, but doesn't see it and play a different spot leading to x winning next turn, these boards are very similar. In multidimensional Euclidian space, these 2 endgames are not segregated nicely in 'clouds', but are somewhat mixed up. These clouds are classification spaces within a multidimensional classification map. This mixing is so bad, training set score converged to only 95%.

ii. Heart-Disease

Distance appears to be a great measure for the heart-disease data if there could have been more data, looking like it would be converging near 100% with more training examples. Distance weighted

instances were better because unlike the TTT data, Heart-disease instances are likely more 'cloudy' and segregated, given what I know about physiology. This is also evident in the perfect training accuracy. The algorithm didn't succumb to the curse of dimensionality with 5-NN. This is likely because all of the features mattered to predicting heart-disease.

d. Model Complexity Analysis

In the initial analysis the hyperparameter k was defaulted to 5. Its variation was tested, and the increasing complexity of model increased with k. The results of the analysis are shown in **fig 8**. As expected for the TTT because of poorly spatially separated instances, there was apparent overfitting with too many neighbors, or maybe more neighbors reveals even more mixed up classes.
Given the probable segregated clouds of heart-disease instances. I was not surprised to only see a little overfitting. These cloud "bubbles" may be small and so many more neighbors lead to poorer results.

f. Assessment and improvements

In improved learner for the TTT set is created with the optimal k-value of 12, and its improvement can be seen in **Fig 9**. A learning curve was not rerun for the heart disease data because of its good performance and similar optimal k to the original optimized learning.
It was surprising to see the improvement with the optimal k value. This amount was enough to provide correct classifications among impure 'clouds' of classifications. I was surprised to see the test accuracy increase from 91.5% to 98.5% with this k value. This analysis does indicate that optimal values of K maybe be optimal for larger training sizes but not necessarily smaller ones.

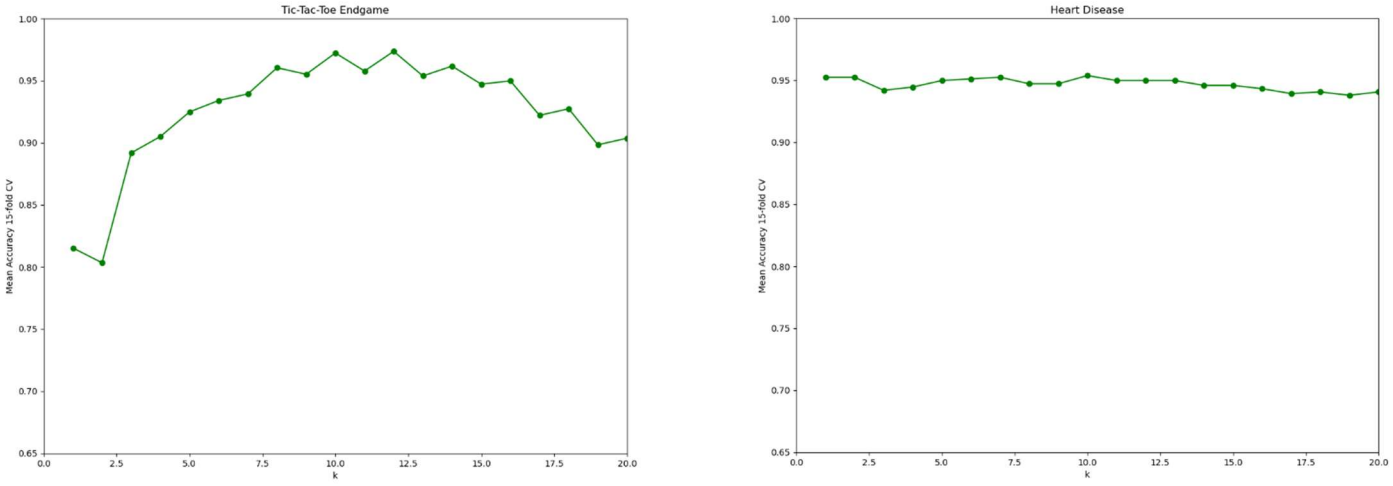

Figure 8. KNN model complexity. Optimal values of k were found to be 12 and 10 respectively.

KNN is biased towards giving larger distances to larger differences in numeric values even if larger numeric differences in features only indicates smaller true feature difference. One improvement I can think when processing varying magnitudes of continuous data would be to scale the continuous data so that KNN doesn't assign a very large feature distance to feature values which have a very large upper bound and lower bound difference. For instance, in the HD set, cholesterol measurements vary from121 to 512 mg/dl and so KNN assigns it the most ultimate importance. This is a massive potential distance compared to other ordinal data which only goes from 0 to 3, but which may be just as important. Another bias of KNN is assigning equal importance to features that have negligible importance. Thus, when using

KNN, it is important to remove features which are known not to be important such as social security number.

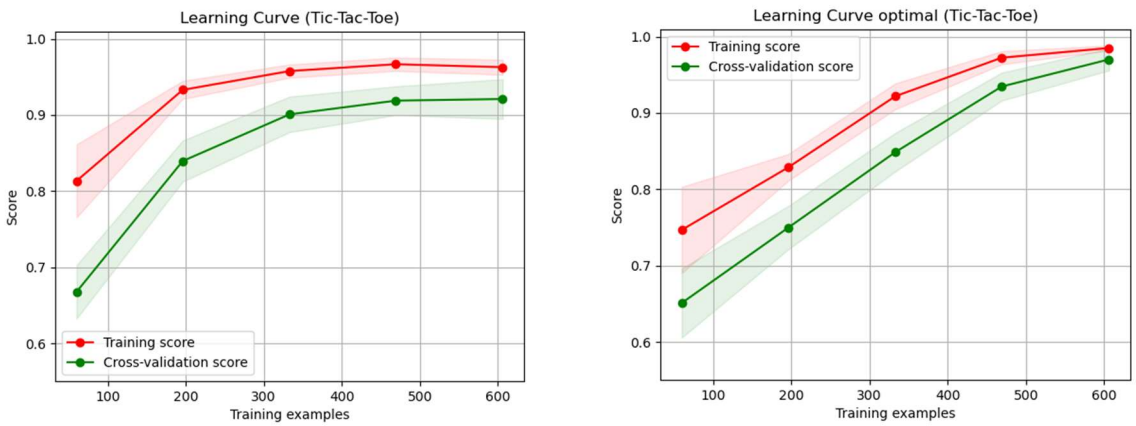

Figure 9. An improved learner. 5-NN (before analysis) vs. 12-NN

8. Comparison Discussion and Conclusion

Times in seconds and accuracies (ACC.) for each algorithm on 200 final test instances are compared in the following table**.** Times are measured from a mean of 10 runs on the optimal learner for each.

	DT	ANN	Boost	SVM	KNN
HE ACC.	95.5%	97.0%	98.5%	97.0%	98.5%
HH ACC.	99.0%	64.5%	99.0%	82.5%	99.0%
HE Time	4.45 ms	206.67 ms	184.3 ms	18.25 ms	18.2 ms
HH Time	4.31 ms	49.12 ms	64.5 ms	1790 ms	6 ms

Which algorithms are ultimately best for overall circumstances will ultimately depend on the situation. When only considering a combination of accuracy and training/prediction times with a moderately limited number of instances (~600) KNN surprisingly proved to be the best for HE and tied for best with HH concepts. All features in the HD data must have been meaningful. When considering a more limited number of training examples available (<= 200), SVM showed rapid converge at a high accuracy for the HE concept. For a very limited number of training examples (<=100), SVM performed with the highest accuracy on the HH problem although training time was long, and more training examples led to little improvement. SVM training times were extremely long for the HH concept with the full available training examples. ANN did not prove relatively useful for either problem, given long training times. More complicated ANNs would likely prove more effective on the HH concepts. Surprisingly, DT were found to be very effective with the HH concept, and not perfectly accurate at the HE concept, apparently as it could not handle the rare outliers well at all. Boosting was found to be worth using if you have sub-optimal accuracies with a base learner but would likely take too long to run with slower base learners. Boosting improved the HE accuracy likely because it made the DTs focus on the rarer outliers.

Bibliography

Chen, Y.-Y., Lin, Y.-H., Kung, C.-C., Chung, M.-H., & Yen, I.-H. (2019). Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes. *Sensors*, 2047.

Mitchell, T. (1997). *Machine Learning.* New York, New York: McGraw Hill.