Andrew Cudworth
CS 7641 Machine Learning (Spring 2020)
Assignment 1 Supervised Learning

CS 7641 Project 1 Supervised Learning

Classification Problem

This paper explores K-Nearest Neighbor (KNN), Decision Tree (DT),Boosted Decision Trees, Neural Networks (NN) and Support Vector Machine (SVM) based supervised learning agents on two contrasting datasets to drive analysis of the Supervised Learning methods.

The first dataset is a summary of Airbnb listings in 2019 in New York city [3]. The target to be predicted is "room type" which can be "Entire Home", "private room", or "Shared Room" encodes as 0,1, and 2 respectively. The features shown in Table 1 are all continuous values creating a relatively standard machine learning problem seeking to maximize testing accuracy. There are 49k unique records broken into a 80/20 train test split in all analysis and experiments.

Table 1 Airbnb Data Record Example

price	Min Nights	Reviews	Reviews/month	Host Listings	Room Type (TARGET)
149	1	9	0.21	6	Private room

Some potential challenges with the data are:

- An exact hypothesis may not exist. There is no reason to assume a set of features have a given label.
- Random samples of the data could easily be skewed to not represent the general distribution. For example high prices are rare, but highly predictive of "Entire Home."
- The "Shared Room" is only 2.3% of the data which may make it hard to predict

Some potential advantages of this data for supervised learning are

- Features and target each have some level of grouping or correlation as summarized in Figure 1 which suggest they provide information about the target value
- Many of the features are independent from each other which provide more avenues of information to the learners

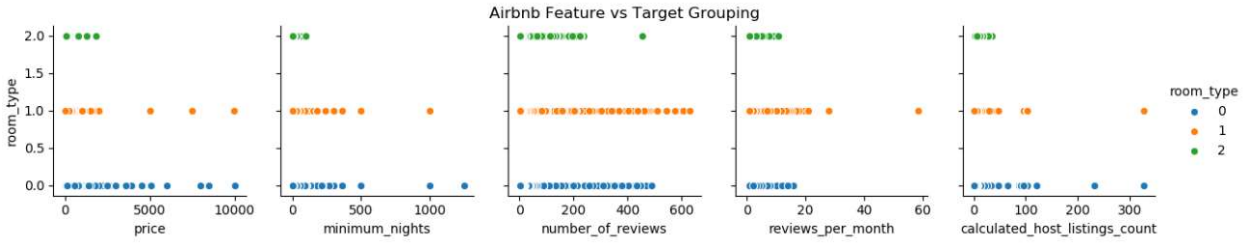

Figure 1 Airbnb Feature vs Room Type Grouping

The second set of data are real hands of poker drawn from a well shuffled deck with a label of the suit and value for each card dealt matching expected probability distributions [2]. For any hand better than 2 of a kind, the hand is labeled 1 for a "good hand" else it is labeled 0. The type of hand is

predicted when 4 of 5 cards are known. There are 25k records with 1547 unique hands with an example in Table 2. Again, all experiments use an 80/20 train test split.

Table 2 Poker Data Record Example (Potential Royal Flush)

Suit1	card1	suit2	card2	suit3	card3	suit4	card4	hand (TARGET)
1	10	1	11	1	12	1	13	1

Some potential challenges with the data are:

- The data is highly imbalanced. There are only 712 good hands among 25k records. High test accuracy can be achieved by always guessing a record is a bad hand.
- Given 4 of 5 cards, two identical records are likely to lead to different results as the 5th card is random.
- The order of cards does not matter, but each feature is unique to the learning algorithm. Because of this, hands that are identical, but dealt in different orders will appear as distinct records.

Some potential advantages of this data for supervised learning are

- The rules for probability are relatively easy to state explicitly. This implies that there is an exact (non-deterministic) solution within the hypothesis space which could be learned given 4 of 5 cards.
- The data very accurately reflects the probability distributions for hands that could be dealt as [2]. This means the data is consistent and accurate. So we know the hypothesis exists and the data reflects it.
- Given 4 of 5 cards, in many cases, a hand already is explicitly good or bad. If a definition of the target is well learned, this can allow for easier prediction.

For both data sets, there is some amount of imbalance summarized in Table 3. For the Airbnb data, only the "Shared Room" is imbalanced.

Table 3 Target Value Distribution

Target Distribution		
Target	**Airbnb data**	**Poker data**
0	51.97%	97.15%
1	45.66%	2.85%
2	2.37%	0.00% (NA)

Results

Both Data sets were originally tested on a "dummy process" including random uniform guessing as well as guessing the most frequent target value. [19] Accuracy was measured purely as the number of records correctly predicted in the test data. Dummy guessing methods do not provide much benefit on the Airbnb data as the records are almost evenly split between 0 and 1. Poker on the other hand has

very few “good hands.” So guessing that every hands is bad or guessing “bad hand” according to the distribution in the data immediately leads to a higher than 97% accuracy. In both cases, uniform random guessing does worse that guessing the most common target.

A grid search was performed on both datasets for all learning agents as well. Results are shown in Figure 2 with boosted decision trees achieving the highest test accuracy at 88.94% on the Airbnb data. Notably, accuracy does not change much for the poker data as most models only guess “bad hand” when tuned for accuracy.

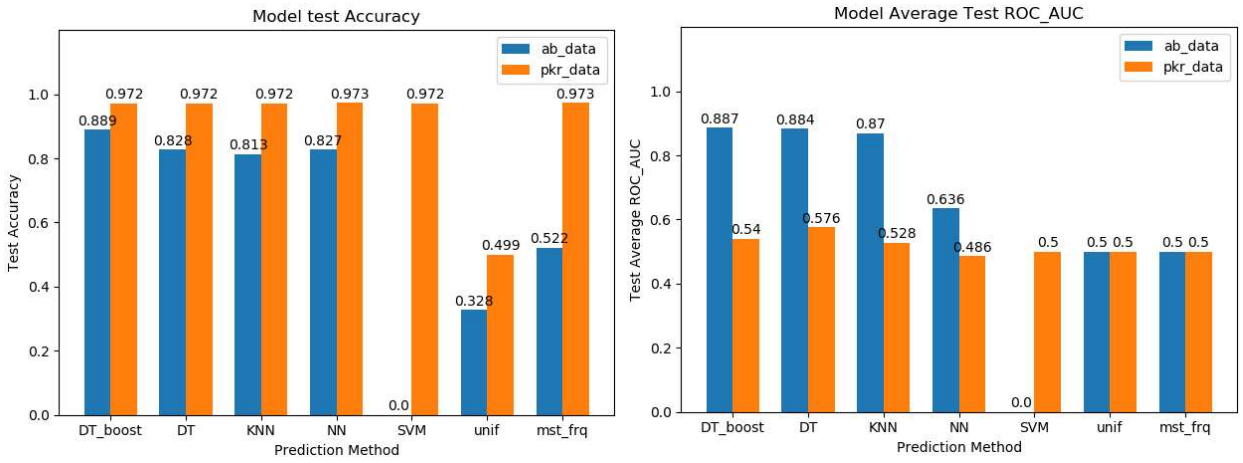

Figure 2 Learner and Naïve method Performance

To account for imbalanced prediction, a new grid search was performed seeking to maximize area under the recall operating characteristic curve for all targets. Each target value was weighted evenly initially to encourage accuracy across all targets rather than the most common. More detailed tuning was performed as discussed in the analysis section as well. A perfect score on always guessing “bad hand” would lead to a score of 0.5 as occurred for random guessing. The results are presented in figure 2 and a single decision tree achieved the highest roc_auc at 57.6%. The poker data generally remained a challenge for all models. With Airbnb models varying in their AUC performance. Results for individual models are discussed in the analysis section.

ANALYSIS

K Nearest Neighbor Learner

K nearest neighbors is a learner that relies on the assumption that records similar to each other will have the same label. The number of neighbors, “K” was varied as in figure 5. Performance of the most accurate model is pictured in figures 3 and 4 which had an accuracy of 81.33% for the Airbnb data and 97.18% for the poker data.

An important property of KNN is that training occurs by adding records vs testing occurs by finding N nearest elements to the record being queries. This means that training time does not increase as more training examples are added. However, querying does take longer as more data is added or K is increased. This is reflected directly in the “scalability of the model” pane in figure 4 for both data sets

where fit or training time is proportional to the amount of training data, but very low relative to other models.

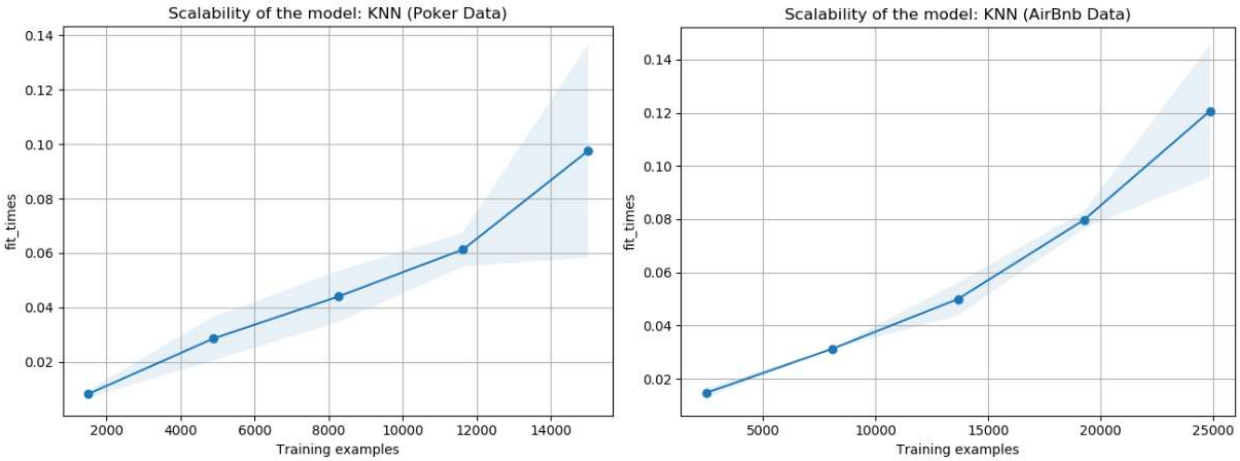

Figure 3 KNN Model Scalability

Accuracy of the KNN model depends on the assumption neighbors are similar. This assumption of similarity and distance manifests differently in each data set though. For Airbnb data, all features are continuous and distance between them is meaningful. For example, a price of $100 is twice as far from $50 as it is from $75. Additionally there are groupings of features for each target. Because of this, the model performs well above naive methods on both accuracy and average roc_auc.

This does not hold true for the poker data. Both suite and card value are discrete numeric values. Similarity is only meaningful when a feature is identical for the sake of prediction. For example, 9 of clubs may be close to 8 of clubs, but is not close when the most common hand is 3 of a kind. Additionally, the separation of suite and numeric values means that suits are given a distance from each other that is not meaningful. A Diamond is not further from one suit than another for example. Finally, a major concept of similarity is missed due to order the cards are dealt not mattering. The hand 9,10,J,Q is identical to Q,J,10,9 in its likelihood of being a good hand. However, the 9 feature would be treated as "far away" from the corresponding Q feature in both cases. This failure is reflected in the low 52.8% auc score that is close to naïve methods. Similarity and distance of the data must be well defined and meaningful to prediction for KNN to outperform naïve methods.

Overfitting for a KNN classifier occurs if the feature and target relationship of the test data changes or K is too high. As discussed, a slice of the Airbnb data is grouped by features, but is largely unique. This means much of the testing data presented to KNN is unique. Overfitting can be seen in Figure 4 for Airbnb data where training data reaches 100% accuracy, but cross validation accuracy remains well below that. Naturally, as more training data is provided, more representative neighborhoods become available improving the CV score. In contrast, the poker training and cv data is nearly identical only varying in the distribution of hands dealt. This is seen in a lack of overfitting. However, More training data does not increase the accuracy or ROC_AUC of the model due to the withheld data not varying much from the training data.

Andrew Cudworth
CS 7641 Machine Learning (Spring 2020)
Assignment 1 Supervised Learning

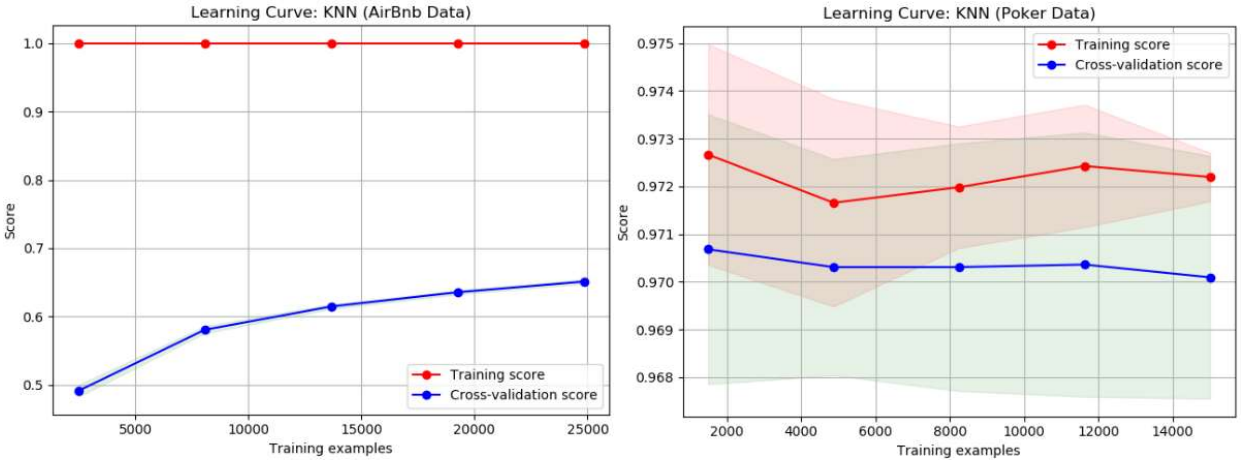

Figure 4 KNN Learning Curves

Increasing K decreases the training score, but also increases testing performance on withheld test data (as well as CV score) for airbnb. This can be seen in figure 6 where both training and testing score converge to 82.7% once more than 17 neighbors are considered. This makes sense as more neighbors allow better generalization. However, this comes at the cost of fit time to make the accurate predictions. Notably, KNN manages above random guessing auc scores with k less than 5 despite the challenges of similarity. This is likely due to hands of 3 and 4 of the same card having similar features. Naturally this does not extend to flushes straights and other hands.

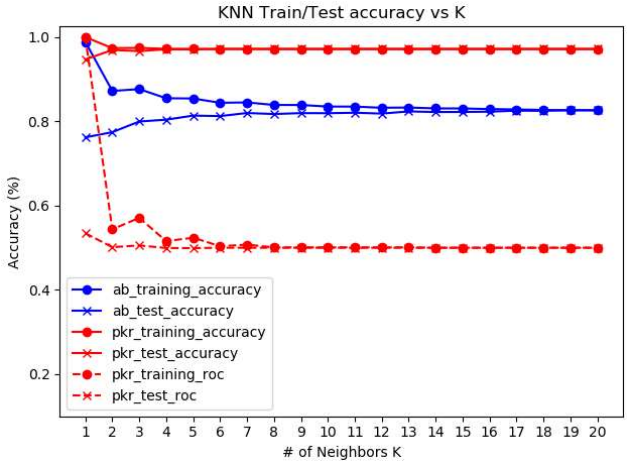

Figure 5 KNN Accuracy Vs K

Decision Tree Learner

Decision trees create a series of criteria to check based on the features and ends with a prediction. They rely on the assumption that individual records can be explicitly defined by their features and future records with those features will have the same target value. The poker data explicitly breaks this assumption. However, the decision tree learns to usually guess bad hand when splitting on random features or features with the highest gini coefficient. This allowed an accuracy of 97.18% on the poker data (slightly worse than always guessing bad hand) and a respectable 82.77%

accuracy on the Airbnb data. The decision trees presented split on highest gini coefficient and use an implementation of the ID3 algorithm. All models used minimum cost complexity pruning. [10][11][13]

Building the decision tree requires a recursive evaluation of all the data which causes training or fit time to grow with number of training examples provided as shown in Figure 6 [10]. Training time is of the same order of magnitude as KNN classifiers. However, query time is a function of the structure of the tree for DT as opposed to size of data and K for KNN. The size of the tree can be adjusted via the construction algorithm or post training pruning providing an alternate route to reduce query time.

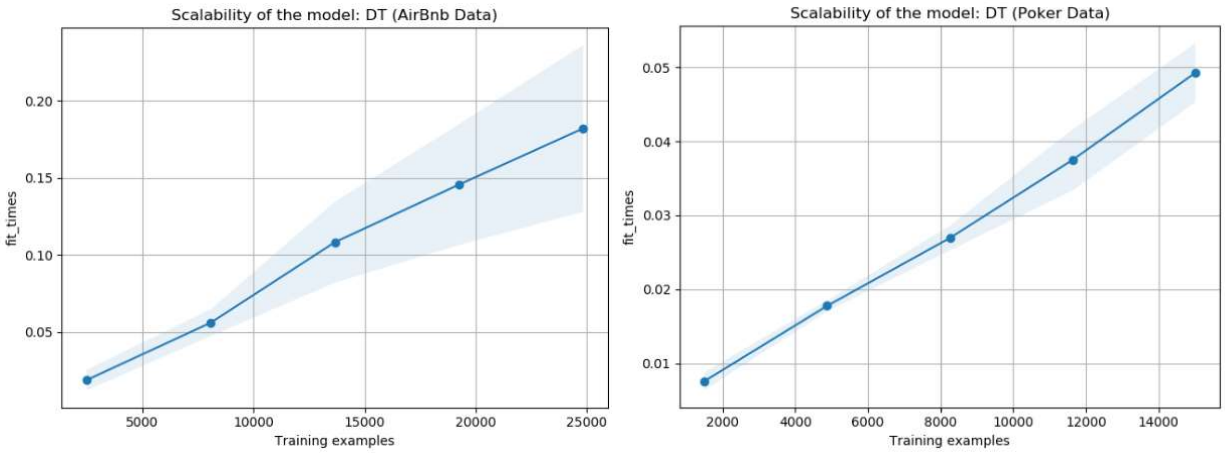

Figure 6 Scalability of Decision Trees

Overfitting will occur in Decision Trees if there is noise in the data or splitting criteria is not well defined. Given unique records, a naïve decision tree could quickly achieve 100% training accuracy at the expense of testing accuracy. This implementation effectively avoids overfitting as seen in figure 7 via minimal cost complexity pruning and a maximum tree depth of 50. [11] MCC Pruning calculates α as defined in Equation 1 [11] and prunes the node with lowest α until no nodes have α less than model specific alpha_min. This reduces the size of the tree as a function of the difference in sample impurity between a node and its terminal children (weighted by sample distribution) where impurity is its Gini coefficient defined in equation 2 [10]. This effectively removes nodes that do not change the outcome of a prediction on training data. Which means the tree is more generalized when presented with new data while not losing accuracy on training data. In fact, in figure 8 as alpha is increased testing accuracy slightly outperforms training accuracy. Unfortunately general accuracy decreases as the tree is pruned more.

$$\alpha_{node\_t} = \frac{H(t) - H(T_t)}{|T| - 1}$$ (T is Terminal Nodes below node t) (H is Gini Coefficient of node)

Equation 1 α for minimum complexity cost coefficient [11]

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$

$$P_{mk} = \frac{Accurate\ Predictions\ for\ K}{Total\ Predictions} \in given\ region$$

Equation 2 Gini Coefficient H(X) [10]

Finally, Decision Trees allow weighting of training accuracy for individual targets. This leads to the highest ROC accuracy of any models for the poker data at 57.5 % with a slight drop in overall accuracy to 95.7% at a sample weight of 0.18 for bad hand vs 0.82 for good hand. (see figure 9)

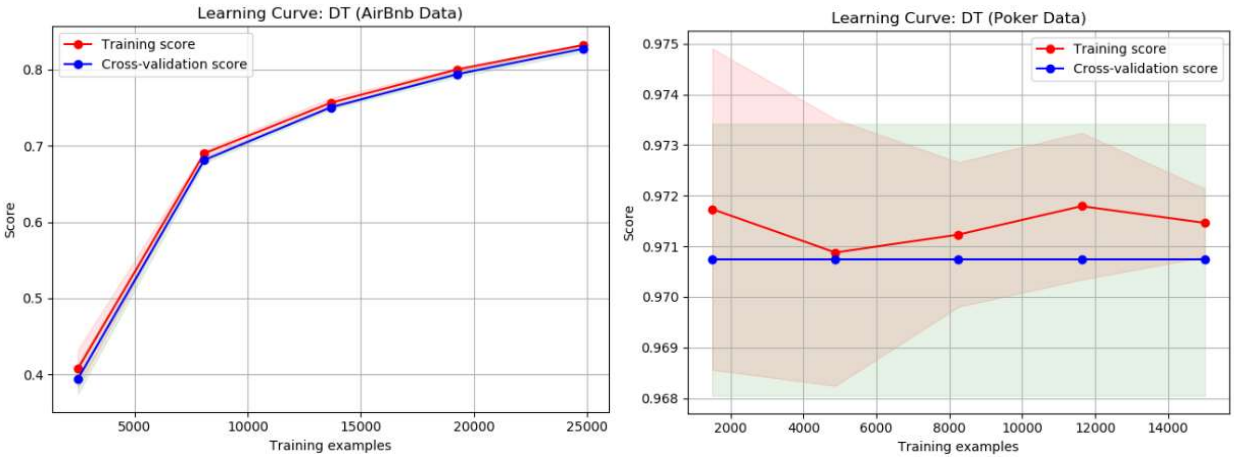

Figure 7 Decision Tree Learning Curves

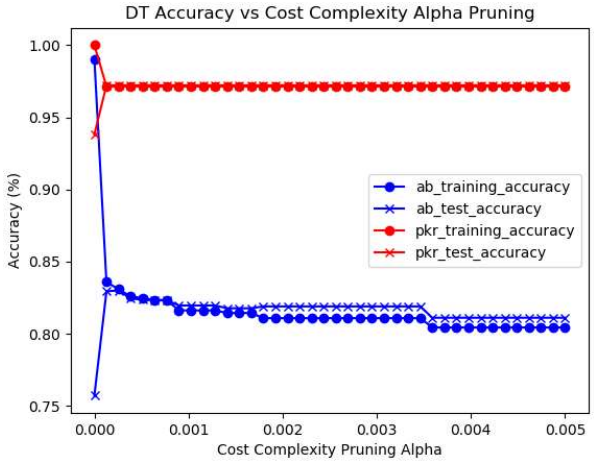

Figure 8 Decision Tree CCP Accuracy α Weight

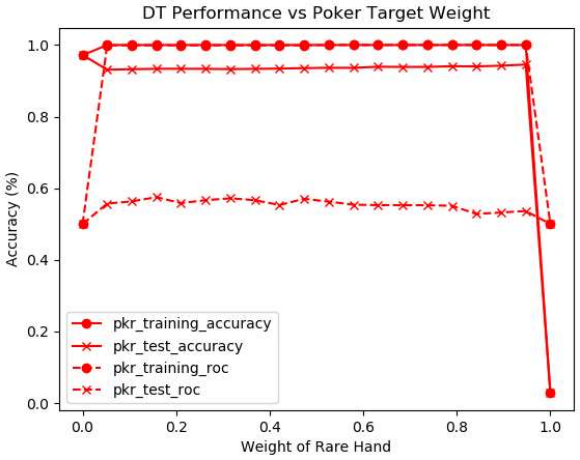

Figure 9 Decision Tree Performance vs Class

Boosted Decision Trees

Boosting takes the average of multiple estimators (decision trees in this case). However, it also chooses training data according to a probability distribution. Data that is predicted incorrectly has its weight increased in this probability distribution [18]. Boosting decision trees lead to the highest accuracy performance of the Airbnb data of 88.94%. On the poker data, weighting the hand leads to similar high ROC values around 54.0% (second only to a single decision tree) seen in figure 11. In both cases, the higher or better performance was accompanied by more aggressive pruning of individual trees. As seen in Figure 10 pruning can be more aggressive with a larger α and accuracy dropping off more slowly than in an individual tree. This holds true for both datasets. However, poker data ROC

remains a challenge as pruning leads to lower ROC due to less predictions of good hand due to their higher likelihood to be inaccurate.

Training time for an individual trees remains the same as in the decision tree though because pruning occurs after training (as implemented). However, the boosting requires many decision trees increasing training time by the factor of trees used to estimate. In this implementation 50 trees were used leading to much longer training time. Query time follows a similar patter requiring an average of all the trees.

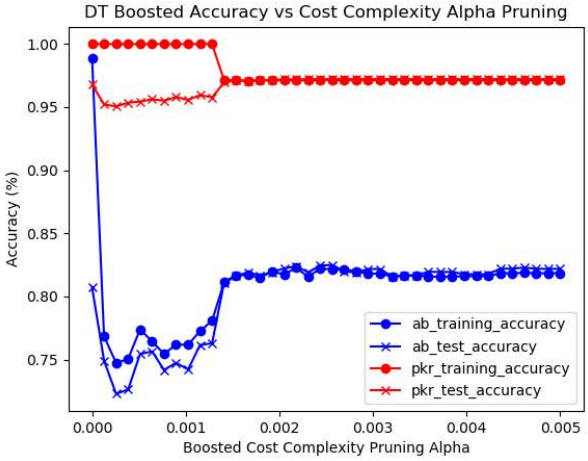

Figure 10 Boosted Complexity Pruning

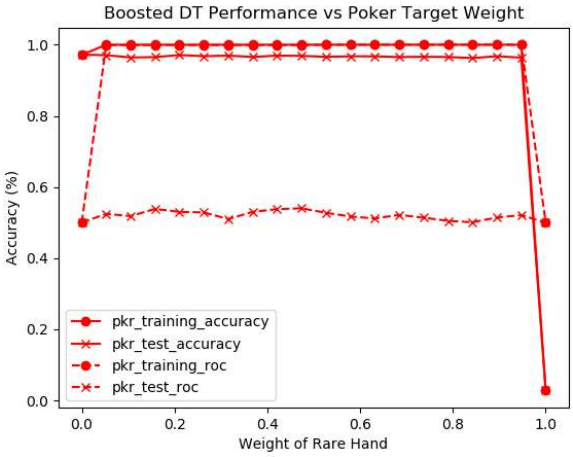

Figure 11 Boosted Performance vs Class Weight

Neural Net Learner

Neural Nets or multi-layered-perceptron (MLP) use a network of weights on an input vector (the value of features for this case) to return a vector of probabilities for the output values (the target values). At each layer of the network, the weights pass through an activation function that passes the weight to the next layer. A solver performs the backpropagation algorithm until the weights converge or a stopping criteria is found [SOURCE]. In this implementation, MLPs achieved 82.7% accuracy for Airbnb data using a 6x2x2x3 network, in line with other models. Interestingly, despite high accuracy its roc_auc was much lower than other models showing imbalanced accuracy. Poker data MLPs achieved accuracy of 97.2% and roc of 50.0% performing similar to always guessing bad hand using an 8x2 network. In this implementation, max iterations for backpropagation were set to 200 for the grid search and 5000 for direct observations in figure 14. Final network structure was found via grid search. However, performance was in line with directly created networks in Figure 14. Many models failed to converge. However, accuracy did not closely track allowed iterations. So iterations were limited for feasibility.

MLPs present a number of challenges including long training times, and a non-deterministic training process, and difficult to interpret models. As with other models, figure 12 shows training time is proportional to training examples, however, the scale of training time is far higher. The MLP models have training time near 100 seconds for all Airbnb data vs < 0.25 seconds for KNN or DT. The initial weights for the network effect also the final weights and data variation from cross validation lead significantly different performance even with identical model parameters. This is seen in the high

standard deviation (the highest of any model) and low scores in for Airbnb data in Figure 13. Non random shuffling of data could be used to address this at the risk of overfitting.

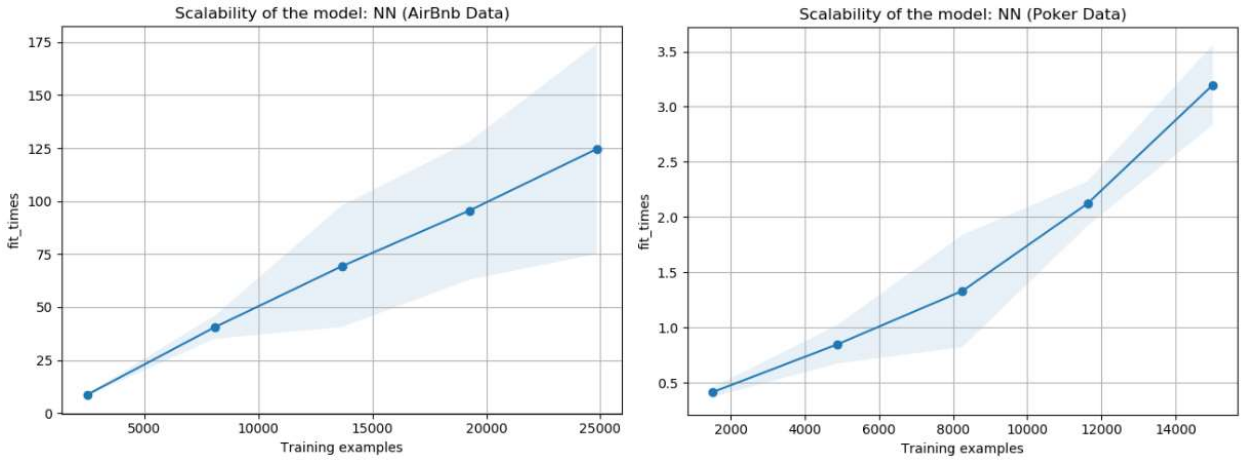

Figure 12 Scalability of Neural Nets

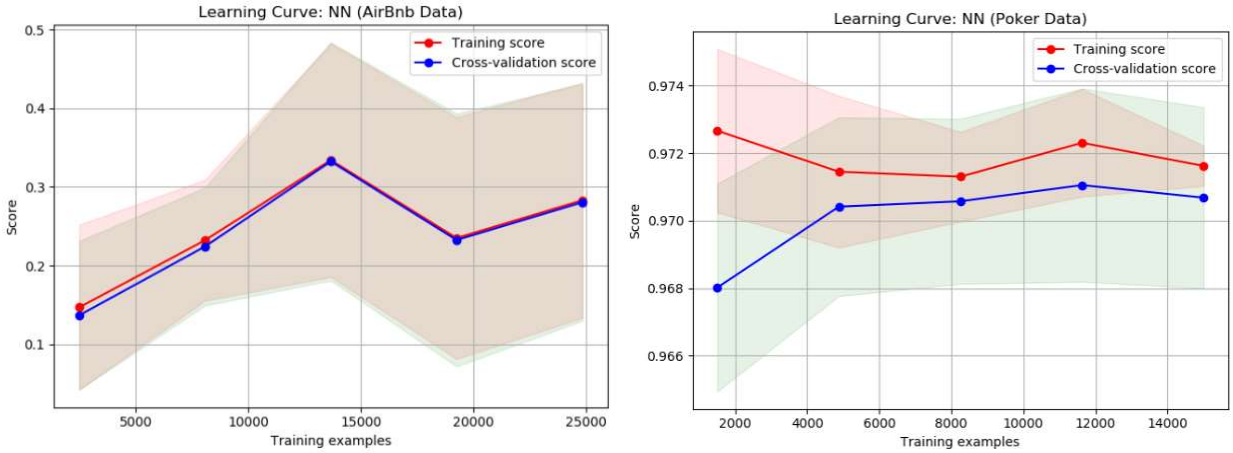

Figure 13 NN Performance

MLPs exhibited little overfitting for the data tested. The MLP creates a nonlinear approximation of the data which allows for good generalization as the data tested is non-linear. However, this comes at the expense of training time and accuracy for non-optimal network structures. Notably, the NN is one of the few models to get better than most frequent guess accuracy on the poker data. However, this comes at the expense of roc accuracy. This makes sense as the target outcome of the poker data is highly non-linear due to suits role in a good hand. The NN can approximate some good hands such as flushes. However, the probability aspect of the 5th card reduces overall accuracy. Finally, the structure of the network does not have a clear impact on the accuracy of the model as seen in Figure 14 making tuning difficult.

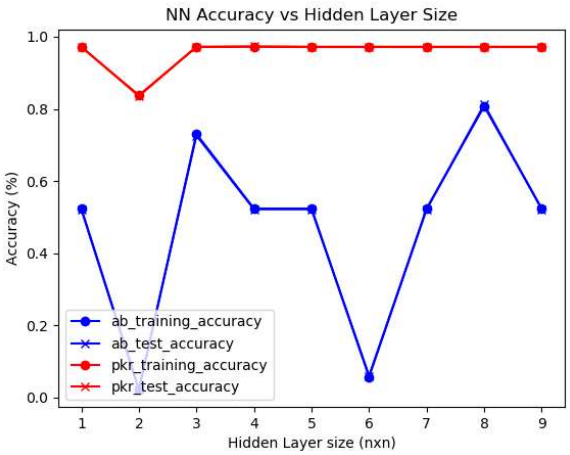

Figure 14 NN Accuracy vs Hidden Layer Size

Support Vector Machine Learner

Support Vector machines separate the data into one of two groups via a kernel function. In the simplest case, this can be pictured as drawing a line (linear kernel function) between two data points of a different class. When a clear delineation is not found, the SVM projects the data into a higher dimension and separates along that dimension[18]. SVMs are designed for binary label classification as separation loses meaning when there are more than two classes. For the Airbnb data, the implementation breaks each class into a probability of 'one vs rest' or the probability that a record is or is not that class then returns the class with highest probability. This causes a significant increase in the training time. For this reason, Airbnb scores are reported on a 1000 record sub sample and their performance should not be directly compared to other models. Four kernels were tested as shown in Table 4 with linear performing most effectively.

Table 4 SVM Performance (1000 record sample of Airbnb Data)

Performance	Kernel			
	Linear	Polynomial	Radial Basis	Sigmoid
ab train score (sample)	84.21%	77.63%	78.95%	64.47%
ab test score (sample)	73.68%	63.16%	52.63%	68.42%
ab train roc auc score (sample)	95.67%	91.80%	93.85%	74.30%
ab test roc auc score (sample)	75.57%	64.77%	55.68%	71.02%
pkr train score	97.13%	97.13%	97.13%	95.61%
pkr test score	97.30%	97.30%	97.30%	95.76%
pkr train roc auc score	50.00%	50.00%	50.00%	52.55%
pkr test roc auc score	50.00%	50.00%	50.00%	52.88%

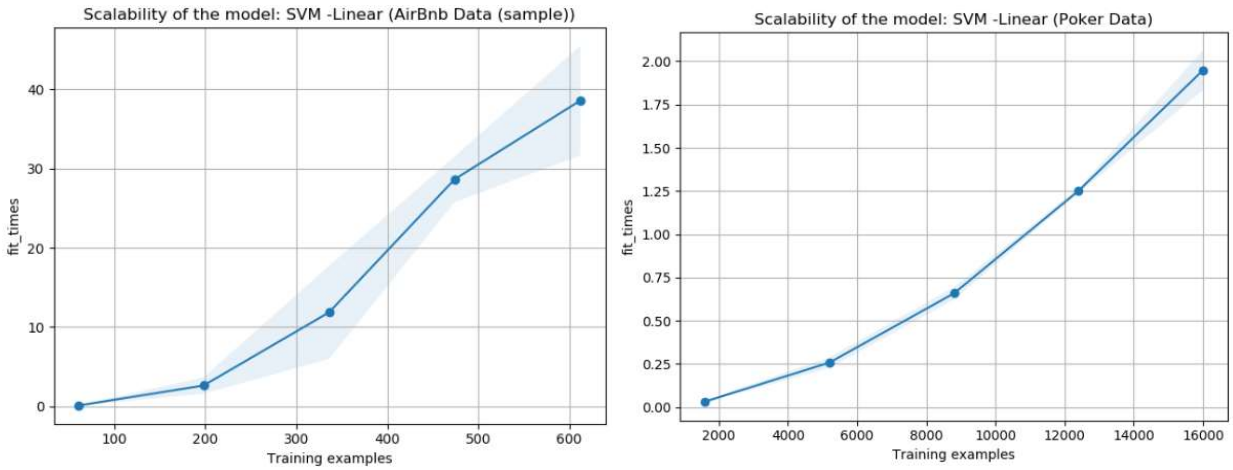

Figure 15 Scalability of SVMS

Support Vector Machines have the worst training time of any model for the Airbnb data. However, the poker data training time is in line with neural nets, but much slower than KNN or DTs. Notably, the training time for Airbnb data is a function of the data being multiclass. If the rare target were removed for example, the model would train very quickly.

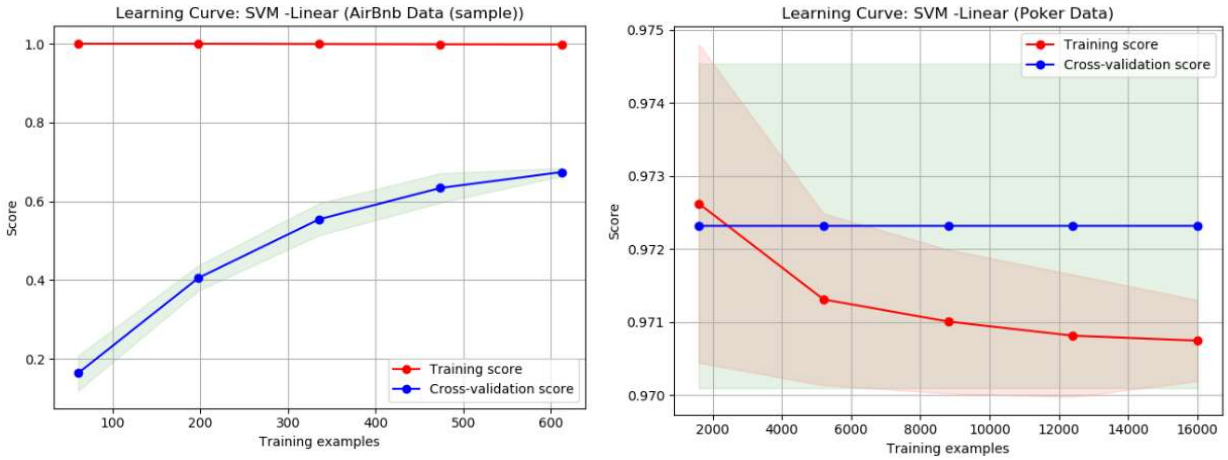

Figure 16 SVM Learning Curve

Andrew Cudworth
CS 7641 Machine Learning (Spring 2020)
Assignment 1 Supervised Learning

SOURCES:

[1] Mitchell, T. M. (1997). *Machine learning*. New York: McGraw Hill.
[2] Cattral, R. (2001, July 1). Poker Hand Data Set. Retrieved from https://archive.ics.uci.edu/ml/datasets/Poker Hand
[3] Dgomonov. (2019, August 12). New York City Airbnb Open Data. Retrieved from https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
[4] Samal, C. (2020, February 5). Airbnb Analysis, Visualization and Prediction. Retrieved from https://www.kaggle.com/chirag9073/airbnb-analysis-visualization-and-prediction
[5] Gaudard, O. (2017, September 7). #11 Grouped barplot. Retrieved from https://python-graph-gallery.com/11-grouped-barplot/
[6] Grouped bar chart with labels¶. (n.d.). Retrieved from https://matplotlib.org/3.1.1/gallery/lines_bars_and_markers/barchart.html
[7] David ZwickerDavid Zwicker 17.7k44 gold badges4747 silver badges6868 bronze badges. (2013, March 1). Plot a bar using matplotlib using a dictionary. Retrieved from https://stackoverflow.com/questions/16010869/plot-a-bar-using-matplotlib-using-a-dictionary
[8] Sanjay.M. (2018, November 2). KNN using scikit-learn. Retrieved from https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75
[9] 3.3. Metrics and scoring: quantifying the quality of predictions¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/model_evaluation.html#model-evaluation
[10] 1.10. Decision Trees¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/tree.html#mathematical-formulation
[11] 1.10. Decision Trees¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning
[12] Post pruning decision trees with cost complexity pruning¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html
[13] Plotting Learning Curves¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
[14] 3.5. Validation curves: plotting scores to evaluate models¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/learning_curve.html#learning-curve
[15] Receiver Operating Characteristic (ROC)¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
[16] Confusion matrix¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
[17] Precision-Recall¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html
[18] Isbell, C., & Littman, M. (n.d.). Machine Learning. Retrieved from https://classroom.udacity.com/courses/ud262
[19] sklearn.dummy.DummyClassifier¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html