Classification With Machine Learning

- CSCI 446 Artificial Intelligence -

Group 16
James Beck, David Rice, Tyler Wright

Montana State University
Gianforte School of Computing

Abstract

When determining how to best solve a given classification problem, it is important to understand the various tools available and what each is best suited for. In this paper, four different machine learning algorithms will be implemented and tested on five different data sets from the UCI Machine Learning Repository in order to determine what data attributes are best suited for which algorithms. The algorithms implemented are K-Nearest Neighbor(k-NN), Naive Bayes(NB), Tree Augmented Naive Bayes(TAN) and Decision Tree Learning - Iterative Dichotomiser 3 (ID3). The data sets chosen are Breast-Cancer-Wisconsin, Glass, House-Votes-84, Iris and Soybean. All data sets were pre-processed to provide uniform formatting, fill in missing values and discretize any continuous data.

After implementing all algorithms it was determined that, as expected, there is not one solution that will be best at all classification problems and instead, the solution must be chosen after careful evaluation of the data. For the k-NN algorithm, size and number of attributes were found to have the least amount of influence when compared to the other algorithms and performance relied mainly on how linearly separable the classes are. Both of the Bayes algorithms were found to do best with a combination of large point numbers and fewer unique attribute values while the ID3 algorithm, though perhaps the most consistent performer, favored the data sets with the most features for its best results.

1 Problem Statement

The goal of this project is to implement four different machine learning algorithms and test each of them on five different data sets from the UCI Machine Learning Repository to attempt to gauge under what circumstances each performs best. By testing each algorithm on a variety of data sets, it should be possible to determine which should be used on a new set of data just by evaluating the characteristics of the data. In order to compare results across data sets as well as algorithms, each set must be pre-processed so that they not only have similar formatting but also so that all data points are discrete.

The algorithms chosen to be implemented are K-Nearest Neighbor, Naive Bayes, Tree Augmented Naive Bayes(TAN) and Decision Tree Learning - Iterative Dichotomiser 3 (ID3). Each of these algorithms will be run on the following five data sets from the UCI repository. 1. Breast-Cancer-Wisconsin, 2. Glass, 3. House-Votes-84, 4. Iris, 5. Soybean.[4]

1.1 Hypothesis

K-Nearest Neighbor K-nearest neighbor has a bias to preform best on data with distinct class separation values and poorly on data with unequal sample sizes for each classification. We also expect K-nearest neighbor to outperform the other classification algorithms on larger data sets (as long as there is still decent class separation and equal sample sizes) that do not also have a large amount of classification values. This is because k-NN doesn't rely on a trained model, but instead relies on the nearby spatial majority. Fewer classifications may cause less noise, and similar classification stratification will allow for a more effective choice of k.

Because of these reasons, the *House Votes* and *Breast Cancer* data sets are hypothesized to perform best on k-NN, while *Soybean-small* may perform the most poorly. The *Iris* and *Glass* data sets are expected to perform on average with the other classification algorithms.

Naïve Bayes Due to the close similarities between Naive Bayes and TAN, both algorithms should see similar results. Since Naive Bayes does not rely on relations between attributes, it is hypothesized that it should perform better, relative to TAN, when data sets are smaller or have less attributes. However, Naive Bayes should still perform best on larger sets that do not have large ranges where individual attribute values can fall.

Tree Augmented Naïve Bayes (TAN) The TAN algorithm is expected to do well on larger data sets that also have more attributes per classification but still have lower numbers of unique attribute values. Like Naive Bayes, if an attribute is present in a testing set but not in a training set, the probability and therefore prediction of classes containing that attribute may be unnecessarily negatively influenced. Lower number of unique attributes should also help when matching on influential variables in a given tree. TAN should perform best on the *House-Votes* data set but do poorly on *Glass* and *Soybean*.

Decision Tree: ID3 Because decision trees provide paths to a classification based on the value of a specific feature, outliers and linearly separated classification values will not negatively affect performance. However, for the same reasons, over-fitting can easily occur. Deep trees are also a sign of over-fitting. The approach used to handle over-fitting is pruning; however, this requires the data set be split into three sets: a testing set, pruning set, and training set. This is an innate weakness of pruning decision trees with small data sets - there may not be enough data to split into three sets while still having a large enough testing set to produce statistically significant results. Decision trees also branch off of discrete attribute values whether the data was originally categorical or continuous.

Thus, we predict that ID3 will perform best on large data sets with few features that already have discrete feature values[1]. *Breast Cancer* and *Iris* data sets fit this criteria the best. The *Iris* data set does have continuous values, but may perform average or above average due to having only four features and has a reasonable number of instances. *Glass* data set is expected to perform average due to continuous values, average sample size, and average number of features. The *Soybean* data set has only 47 instances and is expected to perform the worst.

2 Description of Algorithms Implemented

2.1 K-Nearest Neighbor

K-Nearest Neighbor is a lazy learner that uses a distance function and threshold value k to determine classification of a point based on the k closest points. The design points of interest are the value to set k to and the distance metric.

Choice of k: The value of k needs to be large enough to soften the influence of noisy data points, but small enough to stay within the spatial boundary of a point's correct category population. This implementation sets k to be the size of the smallest category population minus one as shown in equation 1.

$$k = n - 1 \mid n = min(|s|, s \in category\_sets) \tag{1}$$

If there was a data set with 10 training data points for classification A while other other classifications had at least 100 training points, a value of $k > 20$ would cause a misclassification any time the testing data point was of class A. This approach guarantees that will not happen while adapting for some noise.

[1]Continuous feature values will be discretized during pre-processing, but this makes an assumption that already-categorical data has a more helpful representation of which branch to take at each decision node.

Distance function: Because all features are categorical (discussed in section 3.2), the hamming value difference metric in equation 2 is used.

$$\begin{gathered} D_H = \sum_{i=1}^{k} |x_i - y_i| \\ classification(x_i) == classification(y_i) => |x_i - y_i| = 1 \\ \neg(classification(x_i) == classification(y_i)) => |x_i - y_i| = 0 \end{gathered} \tag{2}$$

Many more complex categorical distance functions could be used such as the Eskin, IOF, Goodall, and Burnaby functions [1]. However, the hamming distance is used because this k-nearest neighbor algorithm is meant to be our base case algorithm for comparative analysis of the NB, TAN, and ID3 algorithms. In other words, k-nearest neighbor should be as naïve and simple as possible.

Algorithm The procedure follows the approach described by Leung [3] and is shown in algorithm 1. Procedure ClosetPoints(k, p) generates a subset of k data that are closest to point p using the hamming distance of equation 2.

Algorithm 1 k-Nearest Neighbor Algorithm

1: **procedure** k-NN(D, C, p)
2: **pre:** D is the preprocessed, categorical training data sets
3: **pre:** C is the classifications of each training data point
4: **pre:** p is the normalized query point to assign a classification
5: $k \leftarrow min(|d|, d \in D_{category\_set}) - 1)$
6: $neighborhood \leftarrow$ ClosestPoints(k, p)
7: $classification \leftarrow$ MajorityClass($neighborhood$)
8: **return** $classification$

2.2 Naïve Bayes

Naïve Bayes model uses conditional independence and decomposition to reduce the number of probability calculations that general Bayes' Rule naturally requires [7]. The general case of Bayes' Rule in equation 3 can be reduced by utilizing the general form of conditional independence to yield equation 4.

$$\boldsymbol{P}(Y|X) = \frac{\boldsymbol{P}(X|Y)\boldsymbol{P}(Y)}{\boldsymbol{P}(X)} \tag{3}$$

$$\boldsymbol{P}(X|Y,Z) = \boldsymbol{P}(X|Z)\boldsymbol{P}(Z) \tag{4}$$

Decomposition is performed to produce the Naïve Bayes (NB) model in equation 5.

$$\boldsymbol{P}(C|X) = \boldsymbol{P}(C)\prod_i \boldsymbol{P}(x_i|C) \tag{5}$$

The left side of the NB equation is the posterior probability or the probability of an object X belonging to a particular class given a set of attributes $X =< x_1, x_2, \ldots, x_n >$. The $\boldsymbol{P}(C)$ is the prior probability or the probability of an object belonging to the class in question. Finally, the likelihood of each attribute given the class are all multiplied together. When used with a training set, the values on the right are calculated using frequency tables based on the training set. With the probabilities from the training data, equation 5 is then performed on each object in the test set for each class. The class which returns the highest probability is the class the object is classified as.

2.3 Tree Augmented Naïve Bayes (TAN)

TAN [5] augments NB by relaxing the conditional independence assumption by allowing more complex relationships among attributes. This is accomplished by introducing a deeper tree than the flat tree in NB (which is so simple that it seems a tree is not even necessary in the implementation). The issue that arises is that determining the optimal tree becomes intractable [5]. A simple restriction would be to allow each attribute to have, as parents, the class (like NB) and at most one other attribute [5]. The method used in [5] uses conditional mutual information 6 to determine some relationships between attributes.

$$I_P = (X;Y|Z) = \sum_{x,y,z} P(x,y,c)\log\frac{P(x,y|z)}{P(x|z)P(y|z)} \tag{6}$$

The conditional mutual information in essence measures the information Y provides to X when Z is known [5]. The relationship which maximizes I_P is the one wanted. The TAN algorithm 2 augments NB with additional, yet fairly simple, structure.

Algorithm 2 Tree Augmented Naïve Bayes

```
1: procedure TAN(D, C)
2:     G = undirected graph with an edge between attributes
3:     for all attribute pairs do
4:         attr − edge(x_i, x_j) = I_P(x_i, x_j)
5:     build a maximum spanning tree MSP
6:     convert MSP to a directed graph
7:     build TAN model by connecting C to all X
```

This model can then be used to classify the objects. The formulation used is in equation 7 and used much the same way as equation 5.

$$\boldsymbol{P}(C|X) = \boldsymbol{P}(C)\boldsymbol{P}(X_{root}|C)\prod_i \boldsymbol{P}(X_i|C, X_{parent}) \tag{7}$$

2.4 Decision Tree Learning - Iterative Dichotomiser 3 (ID3)

ID3 is a decision tree learner that sets a model according to training data, and then classifies new data according to the model. Leaf-node values are the classification to apply to the query data, inner-node values represent a feature of the data, and branches represent the feature-value of the query data point for the feature node currently visited. The fitness functions and training algorithm follow the procedure described by J. R. Quinlan [6].

Classification using ID3 has three main procedures:

1. Train - build the decision tree until it cannot be built any further as shown in algorithm 3.
2. Prune - remove inner nodes if their removal causes the tree to perform better.
3. Classify - assign query data a classification according to the leaf node they traverse to when sent through the decision tree.

Fitness Functions: ID3 greedily evaluates the best feature to use at a given inner-node using entropy and gain fitness functions. S is the data set, c is the numbers of features that can be obtained from S, p_i is the ratio of $s \in S$ belonging to some feature i. F is a specific feature within S and $S_v \subset S$ is the data values in S with some specific feature.

$$Entropy(S) = \sum_{i=1}^{c} -p_i log_2(p_i) \tag{8}$$

$$Gain(S, F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{9}$$

$$S_v = s \in S \mid F(s) = v \tag{10}$$

Over-fitting: Decision trees can be prone to over-fitting due to a bias of memorizing the best path of many training data points and creating a tree that is too deep as a result. Reduced error pruning handles this by considering each non-leaf node as a potential pruning candidate, turning each inner-node into a leaf node with the classification most common among its child leaf nodes, and keeping the pruned state if the performance of the tree is better than the non-pruned state.

The performance measure used in this implementation is the average true positive ratio over each classification of the data set. This measure is chosen to give a pruning preference of correct classification over correct negative classifications.

Algorithm: Recursively build decision sub-trees by choosing node feature assignments based on maximal gain. Remove the chosen node's feature from the list of available features. If all values in the data set have the came classification, have no remaining features

to assign as an inner-node, or if there are no data points to assign to in the next recursive iteration, then generate a new leaf node with the classification value equal to the most common classification left in the data set.

Algorithm 3 Decision Tree Learning - ID3

procedure ID3-TRAIN($S, node, features$)
  Pre: S is a pre-processed data set of values with categorical features and known classifications
  if $|S_{classifications}| = 1$ **then**
    return a new node as leaf with classification $s_{classification}$
  if $|features| = 0$ **then**
    return new node as leaf with classification of the majority category among data points still in S
  else
    $best\_feature \leftarrow \{f \mid \forall features \in S, f = MAX(Gain(S, features_i))\}$
    $node_{value} \leftarrow best\_feature$
    for each feature value $f_i \in best\_feature$ **do**
      Generate new branch with value f_i
      $S' \leftarrow \{S_i \subset S \mid$ the subset is only values from S with feature value f_i for feature $best\_feature$ $\}$
      if $|S'| = 0$ **then**
        Attach new node as leaf with classification $MAX(|S'_{classification\_type}|)$
      else
        $new\_node \leftarrow$ child of $node$ from branch f_i
        ID3-TRAIN($S', new\_node, features - \{best\_feature\}$)
  return current node as leaf

3 Experimental Approach

This experiment is highly statistic in nature. The data sets vary in size, stratification, and have both categorical and continuous values. The goal of this experiment is to use the same algorithm and hypothesis test for each data set in order to reveal the bias an algorithm has toward the type of data set it can perform best on without specific tuning for a known set.

Special attention must also be given to the performance metrics. The true positive recall rate will not reveal weaknesses in algorithms that fail to assign the correct classification to proportionally small classification populations. Discretization of continuous data is discussed in section 3.2, performance metrics are discussed in section 3.3, and section 3.4 discusses hypothesis testing.

3.1 Data Set Descriptions

Five total data sets [4] were classified: breast cancer, glass, house votes, iris, and soybean. The data sets can be broken up into the relatively simple discrete and the more complicated continuous case. The discrete case includes house votes, breast cancer, and soybean. The preprocessing was minor in this case. House votes was simple since it was all boolean random variables with two classes: republican and democrat. However, the data set was tricky because some variables took on a ? value which signified the value was neither yes nor no, not that the value was missing. The value for the ? values are assigned a random boolean. The breast cancer data set has random variables with discrete values zero through ten with two classes benign and malignant. Soybean has discrete values zero through six but has seven classes. The continuous case includes the glass and iris data sets. The preprocessing was far more involved which required discretization which is discussed further in section 3.2.

3.2 Pre-processing

Unwanted features and missing data values: In order to account for possible missing values in the data sets, each set was pre-processed before being passed to the machine learning algorithms. During this pre-processing, it was first be determined if less than 5 percent of the total data has a missing value. If this was the case, the data with the missing values was completely removed from the set. If more than 5 percent are missing values, a range was calculated from the existing data which contains all attribute values for each classifier and missing attributes are randomly chosen from within this range.

While the pre-processing is taking place, the data sets are also re-formatted to be more easily used. All classifier values become the first values for a given row and are also given an integer representation. Any non numeric values, such as Y and N, were also changed to an integer representation. The final step of the pre-processing is to determine if a set has any completely unique attributes, such as patient ID numbers[2], and eliminate these attributes.

Discretization of Continuous Attributes: Because classification algorithms such as ID3 and TAN rely on discretized attribute values to make decisions, all continuous data is compartmentalized according to the discretization process described by Fayad and Irani [2]. This section uses the same variable notation used in Fayad and Irani's publication.

The challenge of a discretization heuristic is to determine enough cuts points such that each bin has primarily data of one classification, but not generate so many cut points such that the learning algorithms do not have enough data to infer attribute traits common to each classification. Cut points are thus chosen using entropy and gain equations similar

[2]This occurs in the breast-cancer-wisconsin set

to what is used in the ID3 algorithm in section 2.4 and the halting condition uses the minimum description length principle (MDLP) shown in equation 14.

The entropy, class entropy, gain, and MDLP equations shown in equations 11 - 14 are used in algorithm 4 to discretize a data set with known classifications and continuous feature values. The algorithm iterates over each feature of the data set and recursively generates n cut points based on maximal gain until the halting condition has been reached. The continuous values of the current feature are then reassigned arbitrary integer values unique to each generated bin.

S is the set of data. N is the size of S: $|S|$. $P(C_i, S)$ is the ratio of examples in S that have class C_i. k is the number of classifications in S. A is an attribute of S. T is a cut point on S where $S_1 \subset S$ is the data with values $\leq S$ at T and $S_2 \subset S$ is the data with values $> S$ at T.

$$Ent(S) = -\sum_{i=1}^{k} P(C_i, s) log_2(P(C_i, S)) \tag{11}$$

The "*class information entropy of the partition induced by T*" [2] is defined by:

$$E(A, T : S) = \frac{S_1}{S} Ent(S_1) + \frac{S_2}{S} Ent(S_2) \tag{12}$$

Gain is the set entropy minus class entropy:

$$Gain(A, T : S) = Ent(S) - E(A, T : S) \tag{13}$$

And the MDLP halting criterion does not stop if

$$Gain(A, T : S) > \frac{1}{N}[log_2(N-1) + log_2(3^k - 2) - (kEnt(S) - k_1 Ent(S_1) - k_2 Ent(S_2))] \tag{14}$$

3.3 Performance Metrics

The metrics of choice are macro-average[3], average precision[4], average accuracy[5], and the average F1-score. Macro-average is chosen over micro-average because some of the data sets have classifications with very few data points as members. It is desired to have the statistics be sensitive to small-sized classification subsets.

Let TP stand for true positive, TN is true negative, FP is false positive, and FN is false negative. Sensitivity is calculated by $\frac{TP}{TP+FN}$. Precision is calculated by $\frac{TP}{TP+FP}$. Accuracy is calculated by $\frac{TP+TN}{P+N}$. F1-score is calculated by $2 * \frac{precision * sensitivity}{precision + sensitivity}$ The macro-average takes the average of a binary measure for each classification λ defined as follows:

$$macro\_average = \frac{1}{q} \sum_{\lambda}^{q} B(TP_\lambda, TN_\lambda, FP_\lambda, FN_\lambda) \tag{15}$$

[3]True positive rate averaged over all classifications.

[4]Positive predictive value averaged over all classifications

[5]The trueness of positive and negative classifications compared to the actual positive and negative classifications averaged over all classifications.

Algorithm 4 Multi-Interval Discretization

procedure DISCRETIZE(S) returns S with discretized feature values
  for each feature $f \in S$ **do**
    Sort S by the values of f from least to greatest
    $C \leftarrow$ DISC-RECURSIVE(S, f) ▷ C is the list of cut points
    for cut point $c \in C$ **do**
      $A \leftarrow$ values of f within cut point partition c
      Reassign all values in A with the same arbitrary, unique discrete value
  Return S

procedure DISC-RECURSIVE(S,f) returns a list of cut points for feature f
  Pre: A is an initially empty set outside of the recursive scope that holds the cut point locations
  Remark: $value(S(i), f)$ means the value of data vector $i \in S$ for feature f
  if |S| == 1 **then**
    $A \cup value(S(1), f)$
    Return
  $t \leftarrow ARGMIN_{s \in S}(Ent(S))$
  $S_1 \leftarrow$ data vectors $\leq i$
  $S_1 \leftarrow$ data vectors $> i$
  $threshold \leftarrow \frac{1}{N}[log_2(N-1) + log_2(3^k - 2) - (kEnt(S) - k_1 Ent(S_1) - k_2 Ent(S_2))]$
  if $Gain(A, t : S) < threshold$ **then**
    Return
  else
    $A \cup value(S(t), f)$
    DISC-RECURSIVE(S_1, f)
    DISC-RECURSIVE(S_2, f)
  Return A

3.4 K-Fold Cross Validation

K-fold cross validation is used to avoid over-fitting, data memorization, and lucky runs. The goal is to take the average metrics of many runs while ensuring that the testing data was not used in training. This experiment uses 10-fold cross validation which involves partitioning the data set into ten stratified folds. A loop runs that unions nine of the folds into a training set, and uses the leftover fold as the testing set. The results of the testing set is saved for all ten iterations, and the average of the results is used as the final value.

The 10-fold approach has a weakness with data sets having few[6] data samples for a specific classification such as the soybean and glass data. In this case, two or more testing

[6]Less than ten in this situation.

partitions will test the same data point. If the algorithm happens to be particularly good or bad at classifying that shared point, some accuracy in results has been lost. However, 10-fold cross validation is still used because the benefit of avoiding over-fitting on small data set sizes outweighs the sample point bias that can occur.

3.5 Experiment Run

The experiment is designed to run all data sets across all algorithms while recording the desired averaged metrics described by section 3.3 and 3.4. The experiment is run in the procedure described by algorithm 5. Note that the data sets have already been pre-processed.

Algorithm 5 Experiment Run

```
1:  procedure EXPERIMENT
2:      for each data set do
3:          partitions ← data_set_partitions[1...10]
4:          for each machine learning classifier do
5:              loop 10 times
6:                  training_set ← partitions[1...9]
7:                  testing_set ← partitions[10]
8:                  TRAIN(training_set, classifier)
9:                  fold_results[i] ← TEST(testing_set, classifier)    ▷ Returns a conf. matrix
10:                 training_set.SHIFT()
11:                 testing_set.SHIFT()
12:             results ← AVERAGE(fold_results)
```

4 Experiment Results

The following table shows the results for each algorithm on each data set after running 10-fold cross validation. Appendix A shows the detailed confusion matrix for each run.

Figure 1 shows a comparison of each algorithm for the primary metric: macro-average.

Algorithm	*Macro-Avg*	*Precision*	*Accuracy*	*F1-Score*
BREAST CANCER DATA SET				
K-NN	0.801	0.903	0.859	0.849
NB	0.929	0.910	0.923	0.919
TAN	0.956	0.940	0.951	0.948
ID3	0.936	0.933	0.941	0.935

Glass Data Set				
K-NN	0.865	0.946	0.973	0.904
NB	0.257	0.216	0.790	0.235
TAN	0.317	0.361	0.803	0.338
ID3	0.943	0.950	0.989	0.946
House Votes Data Set				
K-NN	0.907	0.891	0.900	0.899
NB	0.500	0.310	0.621	0.383
TAN	0.978	0.987	0.983	0.982
ID3	0.981	0.975	0.979	0.978
Iris Data Set				
K-NN	0.933	0.935	0.956	0.934
NB	0.167	0.178	0.444	0.172
TAN	0.300	0.290	0.533	0.295
ID3	0.927	0.926	0.951	0.926
Soybean Data Set				
K-NN	1.000	1.000	1.000	1.000
NB	0.500	0.281	0.787	0.360
TAN	0.826	0.887	0.925	0.856
ID3	0.972	0.972	0.988	0.972

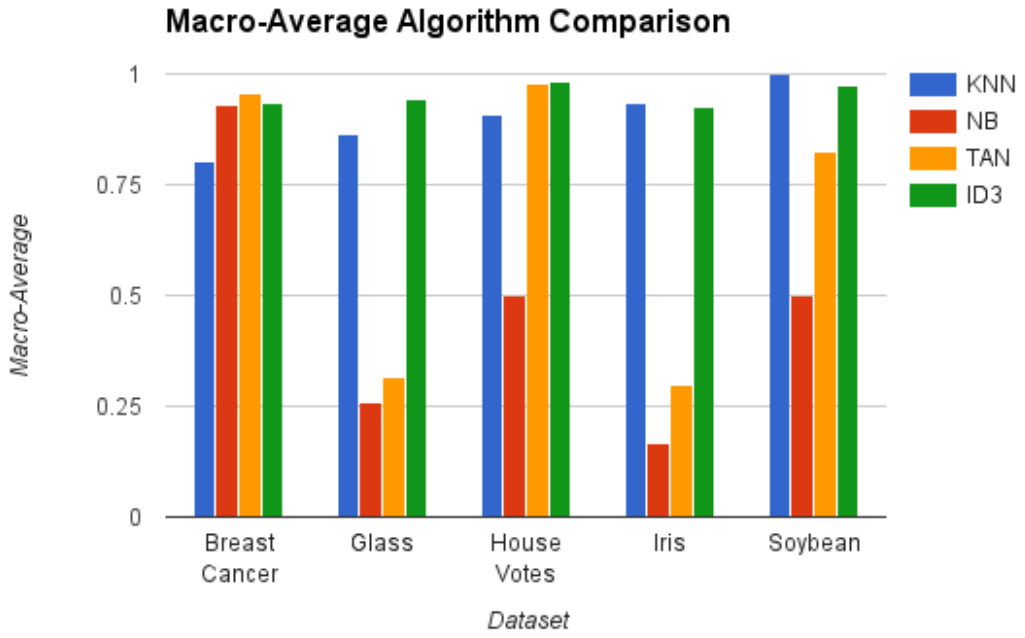
5 Algorithm Behavior

5.1 K-Nearest Neighbor

Section 1.1 expected k-NN to preform well on *House Votes* and *Breast Cancer* due to their larger data set size and equal subset per classification size. *Soybean-small* was expected to have the worst performance due to its smallest data set size. The actual results reveal the opposite. *Soybean-small* and *Iris* had the best performance for macro-average and F1-score while *Breast Cancer* had the lowest macro average and *House Votes* had average performance in all metrics.

House Votes did not perform best likely because of a weakness with the distance function shown in equation 2. The binary feature-wise distance evaluation is notably weak on this voting data set because a Democrat classification will often vote with the same value on many features as a Republican classification. The distinguishing factors between the two classifications is the few specific voting features they will vote differently on. As a result, there is poor spatial separation between data of different classifications. To solve this issue, a different distance value metric should be used that is able to identify feature

Figure 1: Algorithm Performance



votes most common to only one classification subset and give additional weight to those features.

Iris likely performed well in all metrics because the classification subset stratification was equal among all classifications and the iris continuous feature value data is naturally clustered. This means the data set size had less influence on k-NN than expected, and the discretization of continuous feature values discussed in section 3.2 may have been more effective than hypothesized. In similar fashion, *Soybean-small* likely had perfect classification due to notably good separation of feature-values to classifications. The natural spatial clustering is also very strong in this set.

5.2 Naïve Bayes

The Naive Bayes algorithm performed fairly consistently across all data sets with the exception of the breast-cancer set where it performed much better, almost classifying all unknowns correctly. Naive Bayes also performed the worst on the glass set. Interestingly, both of these sets have larger ranges over which attribute values fall when compared to all other data sets. The results from breast-cancer could have been predicted due to it having the largest number of instances but the results from the glass set suggest that having the largest attribute value range may be most detrimental when not having enough instances

for test data to match to.

5.3 Tree Augmented Naïve Bayes (TAN)

Though the House-Votes data set did do well with the TAN algorithm, as was predicted, it was not the best performer by a large margin. In fact, The Breast Cancer set was found to be a close runner up for being best suited for TAN. On the opposite end of the results, Iris and Glass turned out to be the two worst performers. Both Glass and Iris have some of the higher ranges for values that attributes can hold, showing that, as expected, the TAN algorithm likely favors having less options for attribute values. The Breast Cancer set also has high variability of attributes but is also the largest set that tests were run on, meaning that there was much higher chance of matching test data to training data. Soybean was also a low performer and the smallest set that was run.

These results show that the TAN algorithm favors a combination of larger data sets and lower numbers of unique attribute values with size likely being the most important factor as sets with more attribute values tended to outperform those with less, when there were more instances of possible classifications.

5.4 Decision Tree Learning - Iterative Dichotomiser 3 (ID3)

ID3 was hypothesized to do best on *Breast Cancer* and *Iris* due to their already-categorical feature values and large data set size with few features. *Soybean* was expected to do poorly due to its small data set size. The actual results revealed the opposite with *Breast Cancer* and *Iris* being the two worst performers, although not by much as the all metrics for all data sets were greater than 0.926.

The three data sets with best results also had the most features. This suggests that the hypothesis of decision trees primarily preferring large, discretized data sets should be changed to a bias of data sets with many features and many possible feature-values. This will drive generation of a tree that is wide (due to more feature-values) and deep (due to many features), but the depth of a tree can be shortened through pruning techniques.

6 Summary

Though, on average, K-Nearest Neighbor and ID3 performed better than the Bayes algorithms, true to the idea of no free lunch, all algorithms had at least one data set where they performed better than all the other algorithms (with Naive Bayes as an exception almost outperforming on the *Breast Cancer* data set). Naive Bayes and TAN performed similarly as both had the same best result (*Breast-Cancer*) and worst results (*Glass* and *Iris*) and had similar trends across the remaining sets. As expected, TAN performer better than Naive Bayes on all data sets. Results for K-Nearest Neighbor show that the algorithm is not necessarily dependent on the size of the data set but instead on the amount of spacial

separation between the different classifications. Finally, the results from the ID3 algorithm show that of the earlier hypotheses were incorrect and the size of the data sets actually had little to do with the final classification results and instead, the number of attributes seems to have the most influence, with the best results coming from sets with the largest number of attributes per class.

References

[1] Kumar Vipin Boriah Shyam Chandola Varun. *Similarity Measures for Categorical Data: A comparative Evaluation*. 1st. University of Minnesota Department of Computer Science and Engineering, 2007.

[2] Irani K.B. Fayyad U.M. "Multi-interval Discretization of Continuous-valued Attributes for Classification Learning". In: *Proc. Thirteenth International Join Conference on Artificial Intelligence* (1993), pp. 1022–10278.

[3] Ming Leung K. *k-Nearest Neighbor algorithm for classification*. 1st. Polytechnic University Department of Computer Science, 2007.

[4] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[5] Moises Goldszmidt Nir Friedman Dan Geiger. "Induction of Decision Trees". In: *Machine Learning* 29 (1997), pp. 161–205.

[6] J. R. Quinlan. "Computational Intelligence, Concepts to Implementations". In: *Machine Learning* 81 (1986).

[7] Peter Norvig Stuart J. Russell. *Artificial Intelligence: A Modern Approach*. 3rd. Pearson, 2009.

A Appendix A: Confusion Matrices for each Run

A.1 K-Nearest Neighbor

Breast Cancer Data Set:

		Predicted
Actual	44.700	0.300
	9.400	14.600

Macro Average	Precision	Accuracy	F1-Score
0.801	0.903	0.859	0.849

Glass Data Set:

			Predicted			
Actual	6.900	0.000	0.000	0.000	0.000	0.000
	0.400	6.900	0.000	0.000	0.000	0.000
	0.700	0.200	0.800	0.000	0.000	0.000
	0.000	0.000	0.000	1.100	0.100	0.100
	0.000	0.000	0.000	0.000	0.900	0.000
	0.200	0.000	0.000	0.000	0.000	2.700

Macro Average	Precision	Accuracy	F1-Score
0.865	0.946	0.973	0.904

House Votes Data Set:

		Predicted
Actual	8.500	0.600
	1.800	13.100

Macro Average	Precision	Accuracy	F1-Score
0.907	0.891	0.900	0.899

Iris Data Set:

		Predicted	
Actual	5.000	0.000	0.000
	0.000	4.700	0.300
	0.000	0.700	4.300

Macro Average	Precision	Accuracy	F1-Score
0.933	0.935	0.956	0.934

Soybean Data Set:

		Predicted		
Actual	0.900	0.000	0.000	0.000
	0.000	0.800	0.000	0.000
	0.000	0.000	0.800	0.000
	0.000	0.000	0.000	1.500

Macro Average	Precision	Accuracy	F1-Score
1.000	1.000	1.000	1.000

A.2 Naïve Bayes

Breast Cancer Data Set:

		Predicted
Actual	40.900	4.100
	1.200	22.800

Macro Average	Precision	Accuracy	F1-Score
0.929	0.910	0.923	0.919

Glass Data Set:

		Predicted				
Actual	2.400	1.600	0.100	0.200	0.800	1.800
	2.200	4.900	0.000	0.100	0.000	0.100
	0.500	1.100	0.000	0.000	0.100	0.000
	0.300	0.500	0.000	0.100	0.300	0.100
	0.300	0.200	0.000	0.000	0.400	0.000
	0.600	2.200	0.000	0.100	0.000	0.000

Macro Average	Precision	Accuracy	F1-Score
0.257	0.216	0.790	0.235

House Votes Data Set:

		Predicted
Actual	0.000	9.100
	0.000	14.900

Macro Average	Precision	Accuracy	F1-Score
0.500	0.310	0.621	0.383

Iris Data Set:

		Predicted	
Actual	0.300	1.300	3.400
	3.400	1.200	0.400
	2.100	1.900	1.000

Macro Average	Precision	Accuracy	F1-Score
0.167	0.178	0.444	0.172

Soybean Data Set:

		Predicted		
Actual	0.000	0.900	0.000	0.000
	0.000	0.800	0.000	0.000
	0.000	0.000	0.000	0.800
	0.000	0.000	0.000	1.500

Macro Average		Precision	Accuracy	F1-Score
0.500	0.281	0.787	0.360	

A.3 Tree Augmented Naïve Bayes (TAN)

Breast Cancer Data Set:

		Predicted
Actual	42.200	2.800
	0.600	23.400

Macro Average	Precision	Accuracy	F1-Score
0.956	0.940	0.951	0.948

Glass Data Set:

			Predicted			
Actual	2.600	1.400	0.100	0.100	0.900	1.800
	2.200	4.900	0.000	0.100	0.000	0.100
	0.500	0.800	0.300	0.000	0.100	0.000
	0.300	0.400	0.000	0.100	0.400	0.100
	0.300	0.000	0.000	0.100	0.500	0.000
	0.600	1.700	0.000	0.100	0.100	0.400

Macro Average	Precision	Accuracy	F1-Score
0.332	0.383	0.806	0.356

House Votes Data Set:

		Predicted
Actual	8.700	0.400
	0.000	14.900

Macro Average	Precision	Accuracy	F1-Score
0.978	0.987	0.983	0.982

Iris Data Set:

		Predicted	
Actual	0.300	1.300	3.400
	2.300	2.700	0.000
	2.100	1.400	1.500

Macro Average	Precision	Accuracy	F1-Score
0.300	0.290	0.533	0.295

Soybean Data Set:

		Predicted		
Actual	0.400	0.500	0.000	0.000
	0.000	0.800	0.000	0.000
	0.000	0.000	0.500	0.300
	0.000	0.000	0.000	1.500

Macro Average	Precision	Accuracy	F1-Score
0.767	0.862	0.900	0.812

A.4 Decision Tree Learning - Iterative Dichotomiser 3 (ID3)

Breast Cancer Data Set:

		Predicted
Actual	42.800	2.200
	1.900	22.100

Macro Average	Precision	Accuracy	F1-Score
0.936	0.933	0.941	0.935

Glass Data Set:

		Predicted				
Actual	6.800	0.100	0.000	0.000	0.000	0.000
	0.100	7.200	0.000	0.000	0.000	0.000
	0.000	0.100	1.600	0.000	0.000	0.000
	0.000	0.000	0.100	1.200	0.000	0.000
	0.000	0.000	0.000	0.100	0.800	0.000
	0.000	0.100	0.000	0.000	0.100	2.700

Macro Average	Precision	Accuracy	F1-Score
0.943	0.950	0.989	0.946

House Votes Data Set:

		Predicted	
Actual	9.000	0.100	
	0.400	14.500	

Macro Average	Precision	Accuracy	F1-Score
0.981	0.975	0.979	0.978

Iris Data Set:

		Predicted	
Actual	5.000	0.000	0.000
	0.200	4.400	0.400
	0.000	0.500	4.500

Macro Average	Precision	Accuracy	F1-Score
0.927	0.926	0.951	0.926

Soybean Data Set:

		Predicted		
Actual	0.800	0.000	0.100	0.000
	0.000	0.800	0.000	0.000
	0.000	0.000	0.800	0.000
	0.000	0.000	0.000	1.500

Macro Average	Precision	Accuracy	F1-Score
0.972	0.972	0.988	0.972