Project Report: K-Nearest Neighbor Classification
CSCE 633 - Machine Learning

Vandana Bachani (UIN:220009534)

April 10, 2012

Abstract

K-Nearest Neighbor (k-NN) is a simple classification algorithm which classifies new examples based on closest training examples in the feature space. k-NN is an instance-based learning method and can learn non-linear concepts. The project is aimed at analyzing how well a simple algorithm like k-NN is able to effectively solve classification problems on real life datasets. k-NN suffers from the drawbacks of high space complexity and being less sensitive to noise. The main goal is to understand the various techinques to overcome the shortcomings of this algorithm like using condensing, distance-weighted approach, feature selection, feature weighting, etc.

1 Introduction

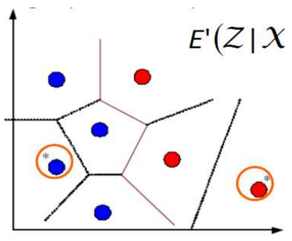
K-Nearest Neighbor is a simple, non-parametric, instance-based classification algorithm, which classifies new examples based on a similarity bias, i.e. it assigns the new example the class of the closest training examples in the feature space. All training examples are saved (if no condensing or editting is applied) which entails high space complexity. The algorithm is slow at classifying new examples in case the training set is huge, as the algorithm considers the distance of the new example from each training example. The algorithm is also less sensitive to noise. A number of measures are used to overcome these drwawbacks and are described in the next section. The following decisions drive the prediction or classification process in k-Nearest Neighbor algorithm:

1. **Distance-Metric**
   Distance-Metric is used to measure the vicinity of a new example to the training examples. There are several distance metrics which may be considered, example Euclidean Distance, Cosine Similarity, Mahalanobis Distance, etc. The implementation uses Euclidean Distance as its distance-metric for continuous attributes and Hamming Distance for binary and nominal attributes.

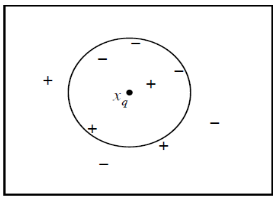
   $$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

2. **k or Distance-Weighted**
   There are several distance based approaches which maybe used to predict the class of new examples from the training examples.

   (a) 1-Nearest Neighbor
   The 1-Nearest Neighbor algorithm measures the new example's distance from all the training examples and assigns the class of the example whose distance from the new example is the minimum. The method is least resilient to noise.

   (b) k-Nearest Neighbor (k-NN)
   Alternatively the algorithm can look at 'k' closest training instances and then take a majority vote to predict the class of the new example to counter excessive similarity bias. 3-NN and 5-NN are frequently used.



(c) Distance-Weighted
In the distance-weighted approach, each training example's class is given a score based on its distance from the test example. At the end all the class scores are accumulated and normalized by dividing by the total score and the class with the highest score is assigned to the test variable.
$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$ where $w_i = \frac{1}{d(x_q,x_i)^2}$

(d) Kernel-Weighted
The kernel-weighted method takes advantage of both parametric and non-parametric methods. The training examples in this case are weighted by a kernel function. The kernel function has a width parameter, h, associated which determines how quickly the influence of the neighbors falls off with distance. A common kernel used is the Gaussian kernel.

$\hat{p}(x) = \frac{1}{Nh} \sum_{t=1}^{N} K(\frac{x-x^t}{h})$ where,

$K(u) = \frac{1}{\sqrt{2\pi}} e^{[\frac{-\mu^2}{2}]}$

The implementation compares k-Nearest Neighbor and Distance-Weighted approaches. The results of the comparison are available in the results section.

2 Drawbacks of k-Nearest Neighbor Algorithm and Techniques to overcome those drawbacks

As mentioned earlier k-Nearest Neighbor algorithm might pose several complexity issues. When the training set is large, saving all examples or efficiently looking up k nearest neighbors becomes challenging. Several techniques have been proposed to combat these issue as discussed below:

1. **Indexing**
If the training set is sorted on each dimension, then a lookup for k-nearest neighbors can be accomplished by binary search in $O(log(n))$ time. But selecting the features to sort the training set might be a concern. In literature, a hierarchical data structure named 'kd-tree' is proposed which is a space partitioning data structure for organizing points in an n-dimensional space.

2. **Condensing - NTGrowth**
Another approach to reducing the space complexity of k-NN algorithm is to use a reduced set of training examples. Though the problem of finding minimal set of examples that make no mistakes on training data is NP-Complete, a greedy approach is followed to discard examples that are classified by others in the training set.
A popular algorithm for the same is the NTGrowth Algorithm proposed by Aha, Kibler and Langley. The algorithm is as follows:

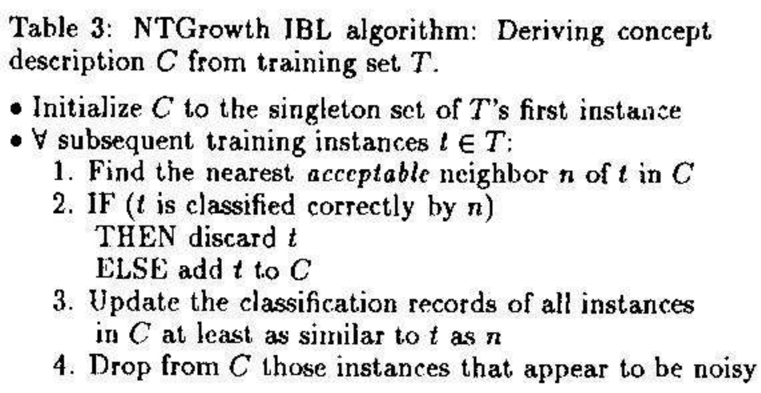


Table 3: NTGrowth IBL algorithm: Deriving concept description C from training set T.

```
• Initialize $C$ to the singleton set of $T$'s first instance
• $\forall$ subsequent training instances $t \in T$:
    1. Find the nearest *acceptable* neighbor $n$ of $t$ in $C$
    2. IF ($t$ is classified correctly by $n$)
         THEN discard $t$
         ELSE add $t$ to $C$
    3. Update the classification records of all instances
         in $C$ at least as similar to $t$ as $n$
    4. Drop from $C$ those instances that appear to be noisy
```

NTGrowth algorithm has been implemented and compared against the regular k-Nearest Neighbor algorithm.
Scaling of attributes is also important for fairness, such that large values do not dominate the distance measures.

3. **Feature Selection**
   Curse of Dimensionality: The various phenomema that arise when analyzing and operating in high-dimensional spaces. Many features in the dataset often leads to lower accuracy.

   Feature selection techniques help identify the key features in a high-dimensional dataset which are able to classify the dataset with high accuracy. Principal Component Analysis (PCA) is a sophisticated technique based on singular value decomposition for doing the same.

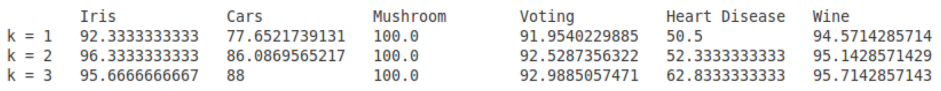
   Two kinds of methods are employed for doing feature selection, namely Filter methods and Wrapper methods.
   Filter methods try to estimate relevance and remove less relevant features. Filter methods include evaluating metrics such as information gain, chi-square, etc. to rank and discard features. RELIEF is a popular filter algorithm for feature selection.
   Wrapper methods use empirical accuracy on training/validation set to pick best features. Stepwise-Forward Selection (SFS), Stepwise-Backward Elimination (SBE), DIET are some wrapper based filter selection methods.

   The implementation uses Stepwise-Forward Selection (SFS) method, with a validation set to predict the accuracy, to pick the best features in a dataset. The plots of the sequence of features identified are shown in the results section. The SFS Algorithm:
   choose single best feature Fa that maximizes acc of 1-NN(D,Fa)
   add next best feature Fb that maximizes acc of 1-NN(D,Fa,Fb)
   keep adding features until accuracy stops increasing

4. **Feature Weighting**
   Features may also be weighted to mark the influence of important features of the dataset on the distance metric. A weighted Euclidean distance metric maybe used to classify new test examples. There are several assess the importance of features and assign appropriate weights to the features, namely, relevance, conditional probability, odds ratio, mutual information, information gain and chi-square metric.
   An experiment with the chi-square feature weighting has been conducted as part of the implementation and the results are presented.
   Chi-Square metric, $\chi^2 = \sum_i \sum_j \frac{(o_{ij}-e_{ij})^2}{e_{ij}}$

3 Experiments and Results

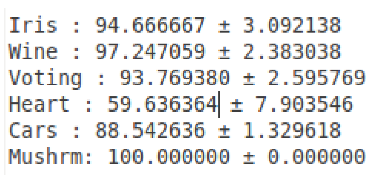
1. **Experiment to determine best 'K'**
   The value of 'k' for k-Nearest Neighbor approach may be different for different datasets. An experiment was conducted using a validation set to determine the best value of k for each dataset, namely 'Iris', 'Wine', 'Cars', 'Voting', 'Mushroom', and 'Heart Disease'. The following table shows best average accuracy reported for a given k-value for each of the datasets.



	Iris	Cars	Mushroom	Voting	Heart Disease	Wine
k = 1	92.3333333333	77.6521739131	100.0	91.9540229885	50.5	94.5714285714
k = 2	96.3333333333	86.0869565217	100.0	92.5287356322	52.3333333333	95.1428571429
k = 3	95.6666666667	88	100.0	92.9885057471	62.8333333333	95.7142857143

   The k values finalized for the datasets:
   'Iris': 3, *'Wine'*: 5, *'Cars'*: 5, *'Voting'*: 5, *'Mushroom'*: 1(any can be used; 100% accuracy in all cases) , and *'Heart Disease'*: 5.

2. **10-Fold Cross Validation and Accuracy**
   Results from 10-fold cross validation depicting the mean accuracy and variance for different datasets:

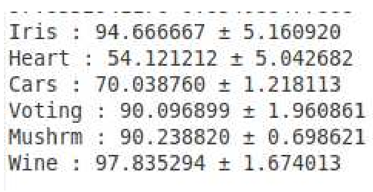


```
Iris : 94.666667 ± 3.092138
Wine : 97.247059 ± 2.383038
Voting : 93.769380 ± 2.595769
Heart : 59.636364 ± 7.903546
Cars : 88.542636 ± 1.329618
Mushrm: 100.000000 ± 0.000000
```

   The observations show that the k-Nearest Neighbor algorithm classifies the datasets with good accuracy. As we see in later sections that the method has comparable efficiency with respect to other methods such as decision trees and neural networks.

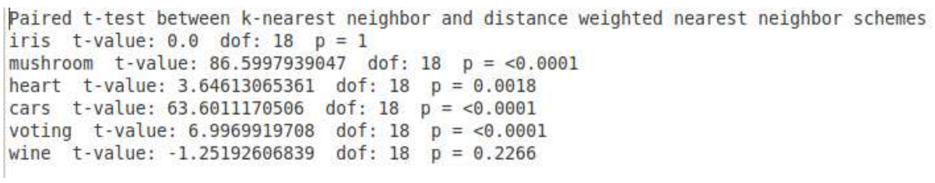
3. **Comparison between k-Nearest Neighbor and distance-weighted approach**
   The implementation compares two instance-based approaches namely, the k-Nearest Neighbor and distance-weighted classification approach as is described in previous sections.

The 10-fold cross validation based mean accuracy for the distance-weighted approach is follows:



```
Iris : 94.666667 ± 5.160920
Heart : 54.121212 ± 5.042682
Cars : 70.038760 ± 1.218113
Voting : 90.096899 ± 1.960861
Mushrm : 90.238820 ± 0.698621
Wine : 97.835294 ± 1.674013
```

A paired T-test is done to compare the two approaches and the following table shows the 't' and 'p' values for the two approaches:



```
Paired t-test between k-nearest neighbor and distance weighted nearest neighbor schemes
iris  t-value: 0.0  dof: 18  p = 1
mushroom  t-value: 86.5997939047  dof: 18  p = <0.0001
heart  t-value: 3.64613065361  dof: 18  p = 0.0018
cars  t-value: 63.6011170506  dof: 18  p = <0.0001
voting  t-value: 6.9969919708  dof: 18  p = <0.0001
wine  t-value: -1.25192606839  dof: 18  p = 0.2266
```

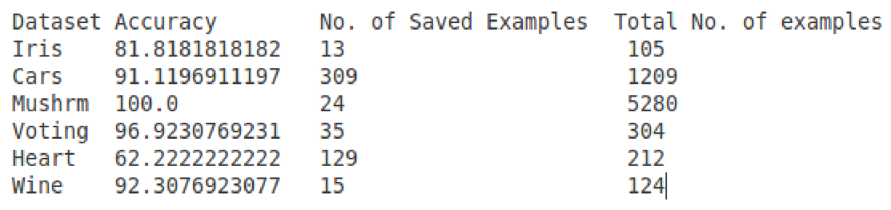
The following conclusions can be drawn from the results:

(a) Iris and Wine datasets have a high p-value indicating that the two methods perform equally well on those datasets. If we take our threshold p-value to be very small like 0.001, then the Heart Disease dataset also is classified marginally equally by the two methods with the k-Nearest Neighbor doing a bit better than the distance-weighted approach.

(b) Observation of very small p-values for the Mushroom, Cars and Voting datasets signify that the k-Nearest Neighbor approach performs statistically much better than the distance-weighted approach for these datasets. The conclusions seem to point that these datasets, in which the training examples are highly clustered especially as in the case of Mushroom dataset with respect to the class boundaries, the weighted distance score might be adding some noise for boundary cases which causes the accuracy to be lower when compared to the k-NN approach which is a good local classifier.

4. **Condensing with NTGrowth**
k-Nearest Neighbor suffers from the drawback of high space complexitiy caused due to saving of all the training examples for class prediction of new examples. The space complexity issue can be resolved by using 'Condensing' techniques which make sure only a representative set of examples are saved in memory.

NTGrowth in one such greedy algorithm which not just reduces the number of training examples that are stored but also is more suceptible to noise. The accuracy results of applying NTGrowth on various datasets are as follows:



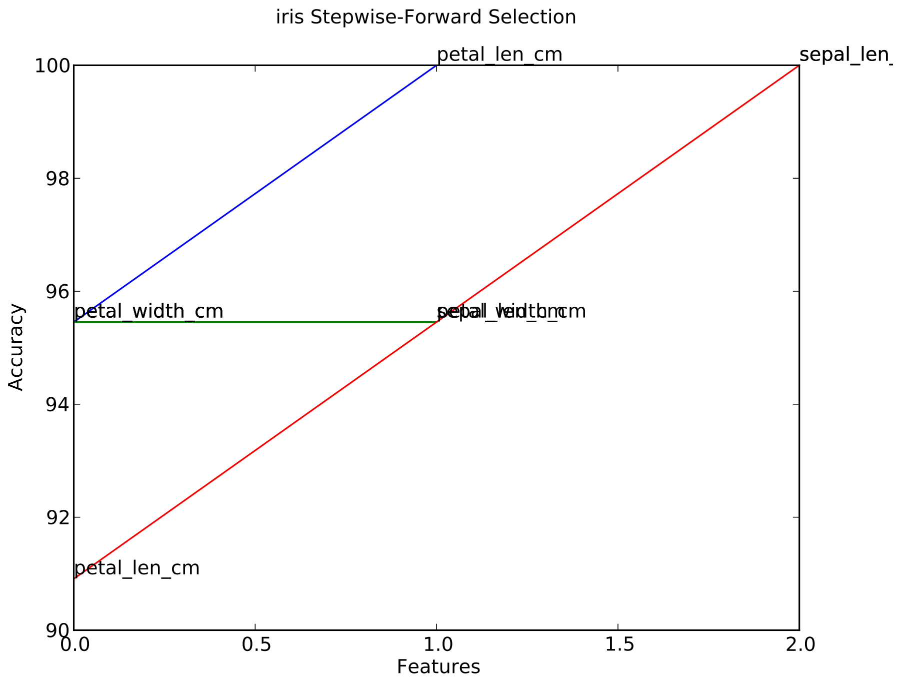
```
Dataset Accuracy        No. of Saved Examples  Total No. of examples
Iris    81.8181818182   13                      105
Cars    91.1196911197   309                     1209
Mushrm  100.0           24                      5280
Voting  96.9230769231   35                      304
Heart   62.2222222222   129                     212
Wine    92.3076923077   15                      124
```

While running the various experiments, the Mushroom dataset which consists of a huge number of training examples i.e. 5280 (this is very large compared to the other datasets), predicting the class of a new example was observed to take a huge amount of time. But as is clear from the results, using NTGrowth reduces the number of examples to be saved down to 24 (in the range of 20-30 as found from multiple runs), and hence the time of prediction was considerably reduced without affecting the accuracy of classification.
For datasets like Iris, and Wine the results from NTGrowth vary across different runs on the algorithm, and for these datasets the sequence of training examples seems to be affecting the performance of the algorithm for the dataset. For a detailed result analysis additional printouts attached to the report can be checked.

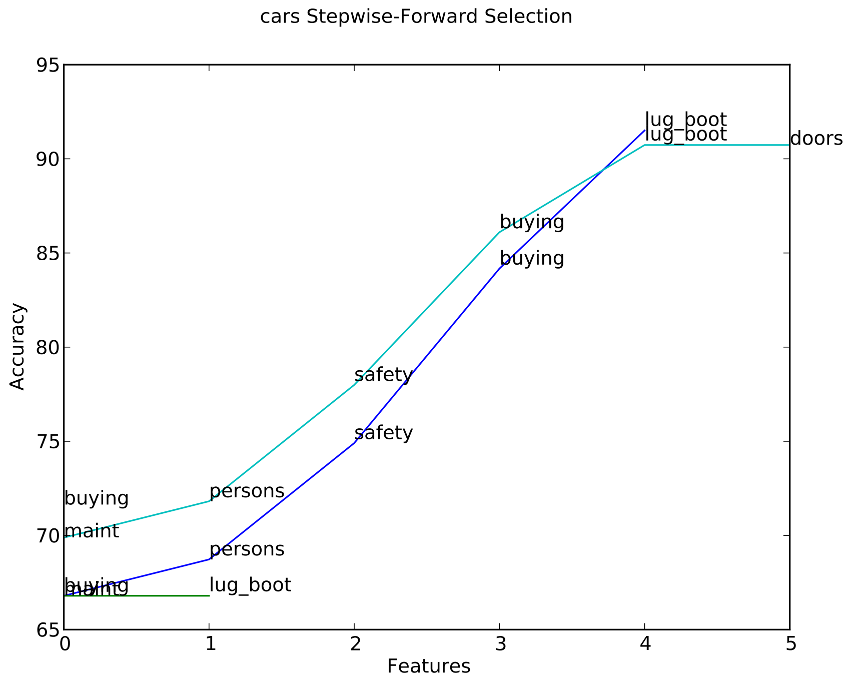
5. **Feature Selection - Stepwise-Forward Selection**
Stepwise-forward selection method for feature selection was implemented and the following graphs were observed for each of the datasets:

(a) Iris dataset
The graph below shows the feature selected at each step of the SFS algorithm with the accuracy reported on a validation set in that step.

As is observed, the feature ’petal_width_cm’ seems to be an important feature which combined with ’petal_length_cm’ give a very high accuracy for Iris dataset for two runs of the algorithm.
Another combination of features which seems to be performing well are, ’petal_length_cm’ followed by ’sepal_width_cm’ followed by ’sepal_length_cm’.

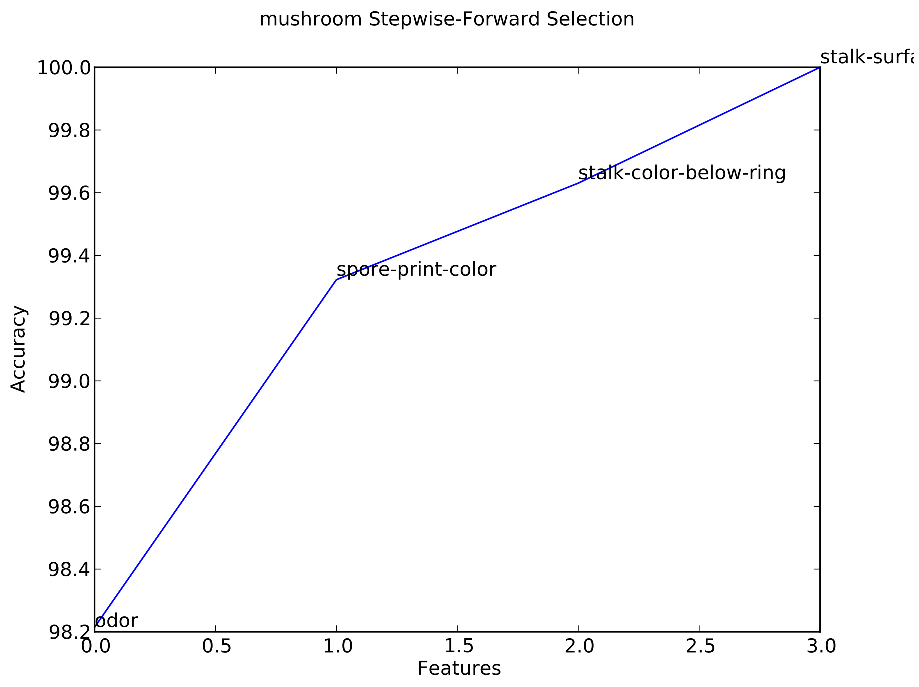


(b) Cars dataset
The graph below gives a good insight into what features push the accuracy of the Cars dataset with each step of SFS.
As is observed, features ’buying’, ’persons’, ’safety’, ’lug_boot’ and ’maintenance’ are the most significant features for the dataset. Another interesting fact that is observed is that, ’maint’, ’persons’, ’safety’ features are selected first and then ’buying’ and ’lug_boot’ do the remaining refinements of accuracy for the dataset.



(c) Mushroom dataset
The ’odor’ feature of the Mushroom dataset is the most significant feature and is able to classify more than 98.2% examples on its own. Other features such as ’spore-print-color’, ’stalk-color-below-ring’ and ’stalk-surface’ further enhance the performance, suggesting that 22 features are too many where actually the performance is driven by only few of the features. This also suggests that many features might be correlated or

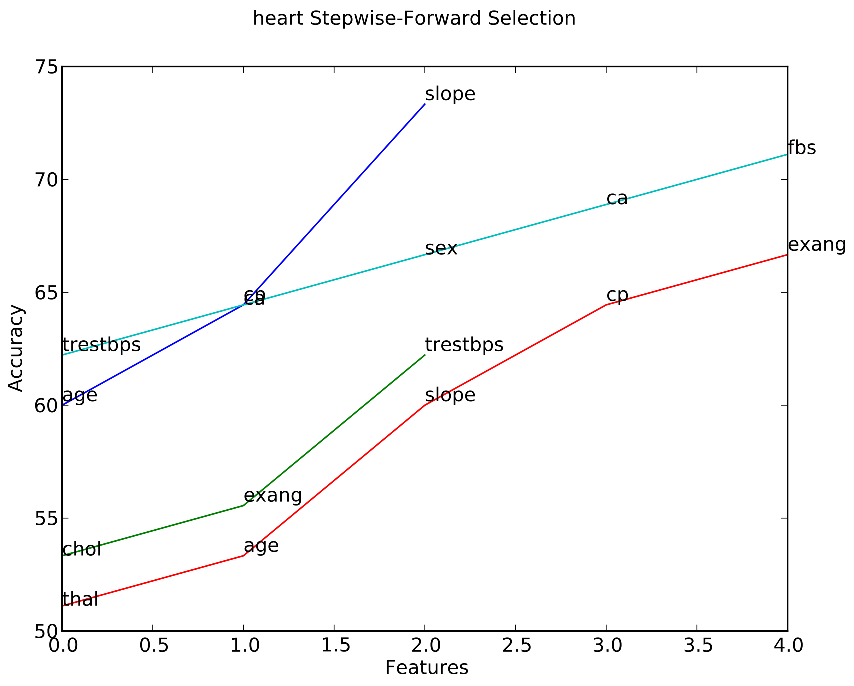
dependent upon these features as they seem to be having no effect on the overall performance.



(d) Heart Disease dataset

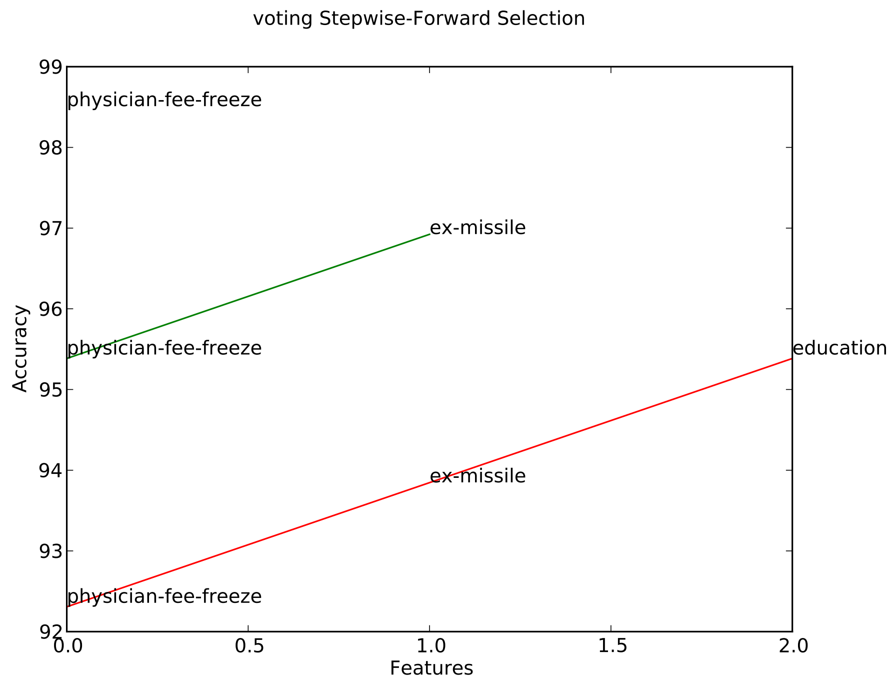
There are several trends observed in the heart disease feature selection. Some features which may seem to be important at first are not able to classify the new examples well. Whereas few other features give better performance but are not always picked.

These observations confirm that the Heart Disease dataset is particularly noisy or there is a lack of appropriate features, which might not have been considered in this dataset.

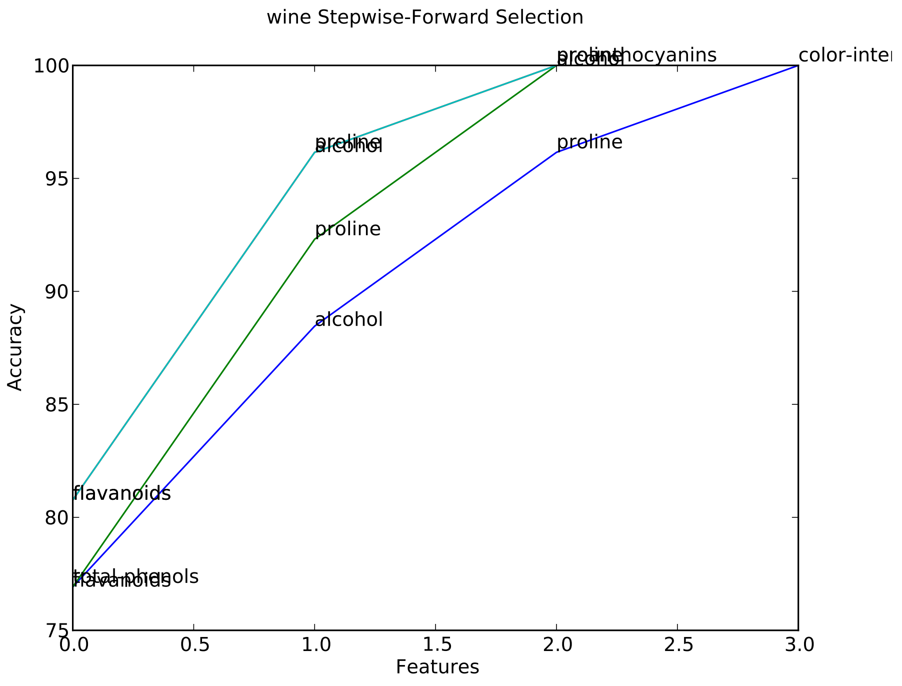


(e) Voting dataset

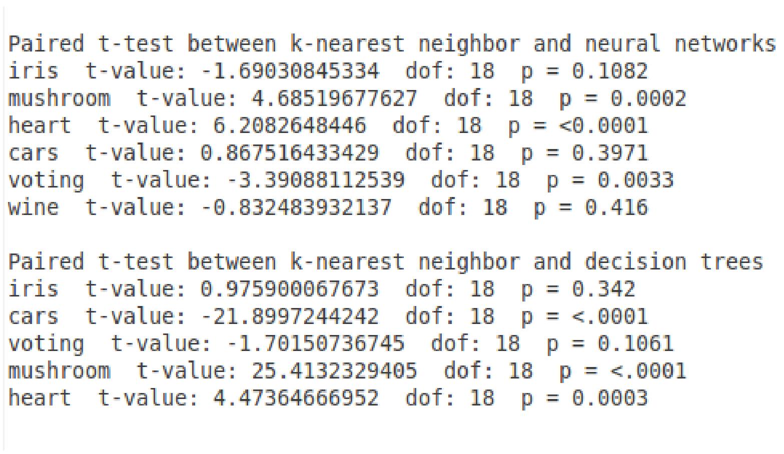
The observations for the Voting dataset are in sync with the known phenomenon. The ’physician-fee-freeze’ feature emerges out to be the most significant feature, followed by ’ex-missle’ and ’education-spending’ features.



(f) Wine dataset
For the Wine dataset, it is clear that the ’flavanoids’, ’alcohol’ and ’proline’ are the most significant features in distinguishing between various categories of wines.



6. **Comparison with Neural Networks and Decision Trees**
Paired T-tests were conducted to compare the performance of k-Nearest Neighbor algorithm to Decision Trees and Neural Networks. The following table shows the observed ’t’ and ’p’ values for the approaches:



```
Paired t-test between k-nearest neighbor and neural networks
iris  t-value: -1.69030845334  dof: 18  p = 0.1082
mushroom  t-value: 4.68519677627  dof: 18  p = 0.0002
heart  t-value: 6.2082648446  dof: 18  p = <0.0001
cars  t-value: 0.867516433429  dof: 18  p = 0.3971
voting  t-value: -3.39088112539  dof: 18  p = 0.0033
wine  t-value: -0.832483932137  dof: 18  p = 0.416

Paired t-test between k-nearest neighbor and decision trees
iris  t-value: 0.975900067673  dof: 18  p = 0.342
cars  t-value: -21.8997244242  dof: 18  p = <.0001
voting  t-value: -1.70150736745  dof: 18  p = 0.1061
mushroom  t-value: 25.4132329405  dof: 18  p = <.0001
heart  t-value: 4.47364666952  dof: 18  p = 0.0003
```

The following conclusions can be drawn from the results obtained:

(a) The observed p-values for the Iris dataset are high and suggest that the k-Nearest Neighbor algorithm performs statistically at par with Decision Trees and Neural Networks. The negative t-value suggests that the neural networks perform better than k-Nearest Neighbor which does better than Decision Trees for Iris classification (positive t-value).

(b) For Cars dataset, k-NN performs statistically as good as Neural Networks, but Decision Trees perform much better than k-NN as is observed by the very low p-value and the negative t-value.

(c) The k-NN algorithm still performs statistically ok with respect to Decision Trees and Neural Networks (given the low p-value, if we consider the threshold p-value to be 0.001). The negative t-values indicate that both Decision Trees and Neural Networks perform better than k-NN for Voting dataset.

(d) The significantly low p-values for the Heart Disease dataset show that the k-NN algorithm performs better than both Decision Trees and Neural Networks.

(e) k-NN algorithm also performs much better than Decision Trees and Neural Networks for the Mushroom dataset as is observed by the very low p-values.

(f) p-value of 0.416 for the Wine dataset suggests that k-NN performs at par with Neural Networks for this dataset with Neural Networks algorithm having a slight edge over the k-NN algorithm.

References

[1] Machine Learning By Tom Mitchell.
http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf

[2] Introduction to Machine Learning By Ethem Alpaydin.
http://www.cmpe.boun.edu.tr/ ethem/i2ml2e/

[3] For finding p-Values
http://vassarstats.net/tabs.html?#csq