International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-6)

Research Article

June 2015

Performance Comparison between Naive Bayes, Decision Tree and k-Nearest Neighbor

Kalavathi K
4PA13SCS07, 4th Sem M.Tech,
P.A. College of Engineering, Mangalore,
Karnataka, India

Nimitha Safar PV
4PA13SCS11, 4th Sem M.Tech,
P.A. College of Engineering, Mangalore,
Karnataka, India

Abstract-

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.
A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time. In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.
We are comparing Naive Bayes, Decision Tree, and k-Nearest Neighbor using Neural Network Toolbox. Neural network tool box consists of tools for designing, implementing, visualizing and simulating neural networks.

Keywords- MLP, AUC

I. INTRODUCTION

Machine learning, a technique which makes use of various disciplines like cognitive science, computer science, pattern recognitions and statistics, can be used to classify and predict data. Neural networks are used for applications where formal analysis would be difficult or impossible such as pattern recognition and nonlinear identification and control. Tool box supports feed forward networks, radial basis networks, dynamic networks, self-organizing maps and other networks. It would be impossible to cover the total range of applications for which neural networks have provided outstanding solutions [1]. At most basic level what we needed are neural network tools to classify / predict the physical, chemical and biological properties of various types of datasets.

II. LITERATURE SURVEY

Neural networks: Neural networks offer a mathematical model that attempts to mimic the human brain. Knowledge is represented as a layered set of interconnected processors, which are called neurons. Each node has a weighted connection to other nodes in adjacent layers. Individual nodes take the input received from connected nodes and use the weights together with a simple function to compute output values. Learning in neural networks is accomplished by network connection weight changes while a set of input instances is repeatedly passed through the network [2]. Once trained, an unknown instance passing through the network is classified according to the values seen at the output layer.Typically,neural network model is having a configuration in its basic form. Neurons only fire when input is bigger than some threshold. It should, however, be noted that firing doesn't get bigger as the stimulus increases, it is an all or nothing arrangement. Suppose a firing rate is there at each neuron. Also suppose that a neuron connects with m other neurons and so receives m-many inputs "x 1 ..... x m. This configuration is actually called a Perceptron [9].
Training: Vectors from a training set are presented to the network one after another. If the network's output is correct, no change is made. Otherwise, the weights and biases are updated using the perceptron learning rule (as shown
above). When each epoch (an entire pass through all of the input training vectors is called an epoch) of the training set has occurred without error, training is complete. When the training is completed, if any input training vector is presented to the network and it will respond with the correct output vector. If a
vector, P, not in the training set is presented to the network, the network will tend to exhibit generalization by responding with an output similar to target vectors for input vectors close to the previously unseen input vector P. The transfer function used in the Hidden layer is Log- Sigmoid while that in the output layer is Pure Linear [14]. Neural networks are very good in classification and regression tasks where the attributes have missing values and also when the attribute values are categorical in nature. The accuracy observed is very good, but the only bottle neck is the extra training time and complexity in the learning process when the number of training set examples seems very high. It describe how neural networks can be applied in data mining. There are some algorithms for extracting comprehensible representations from neural networks. Describes research to generalize and extend the capabilities of these algorithms. The application of the data mining technology based on neural network is vast. One such area of application is in the design of mechanical structure. Introduces one such application of the data mining based on neural network to analyze the effects of structural technological parameters on stress in the weld region of the shield engine rotor in a submarine. It explains an application of neural networks in study of proteins. In that work, global adaptive techniques from multiple alignments are used for

prediction of Beta-turns. This also introduces global adaptive techniques like Conjugate gradient method, Preconditioned Conjugate gradient method etc.
In an approach to discover symbolic classification rules using neural networks, the network is trained to achieve the required accuracy rate, and then activation values of the hidden units in the network are analyzed. Classification rules are generated using the result of this analysis.

III. METHODOLOGIES

The neural network tool box consists of a set of functions and structures which easily handle neural networks; we can use GUI or command-line operations or simple coding by using m-file. Neural networks are very good at pattern recognition problems. A neural network with enough elements (called neurons) can classify any data with arbitrary accuracy. The procedure is, first the user has to decide the structure of the MLP network architecture such as the number of hidden layers and neurons (nodes) in each layer. The activation functions for each layer are also chosen at this stage, that is, they are assumed to be known. The unknown parameters to be estimated are the weights and biases. Many algorithms exist for determining the network parameters. In neural network literature the algorithms are called learning or teaching algorithms, in system identification they belong to parameter estimation algorithms.
The most well-known are back-propagation and Levenberg-Marquardt algorithms. Back-propagation is a gradient based algorithm, which has many variants. creation function like newlin (create a linear layer), newp (create a perceptron) or newff (create a feed forward back propagation). They are particularly well suited for complex decision boundary problems over many variables. Therefore neural networks are a good candidate for solving the wine and iris classification problem. The neighborhood attributes will act as inputs to a neural network, and the respective target for each will be an n-element class vector. The network will be designed by using the attributes of neighborhoods to train the network to produce the correct target classes. The work flow of any problem can consists of six primary steps they are collection of data, create the network, initialize the weights and biases, train the network, validate the network and use the network. First load the dataset e.g. load wine it automatically arrange the dataset attributes X and classes y. then use P=X; then find its size by using size(P) then give T=y'; then find its size by giving size(T). We need not split the data set for training and testing separately it could be done be the neural net works. The data preparation task consists of three stages like 75% training, 15% validation and 15% testing data by using the coding part [trainV,valV,testV]=dividevec(P,T,0.10,0.10). here P is inputs (attributes) and T is targets (classes). Then design the network by using patternnet=10; then use net=patternnet; create a network function like newlin(create a linear layer) or newp(create a perceptron) or newff(create a feed forward back propagation network). Then it will display the sub object structures.

Algorithms:

Naïve Bayes

P(Outcome/Multiple Evidence) =
P(Evidence1/Outcome) x P(Evidence2/outcome) x ... x P(EvidenceN/outcome) x P(Outcome)
scaled by P(Multiple Evidence)
Or
P(outcome/evidence) =
P(Likelihood of Evidence) x Prior prob of outcome
P(Evidence)

Decision Tree

TreeGrowing (S,A,y)
Where:
S - Training Set
A - Input Feature Set
y - Target Feature
Create a new tree T with a single root node.
IF One of the Stopping Criteria is fulfilled THEN
Mark the root node in T as a leaf with the most
common value of y in S as a label.
ELSE
Find a discrete function f(A) of the input
attributes values such that splitting S
according to f(A)'s outcomes (v1,...,vn)gains
the best splitting metric.IF best splitting metric
>treshold THENLabel t with f(A)
FOR each outcomeviof f(A):
Set Subtree i= TreeGrowing (æf(A)=viS,A,y).
Connect the root node of tT to Subtree I with an edge that is labelled as vi
END FOR
ELSE

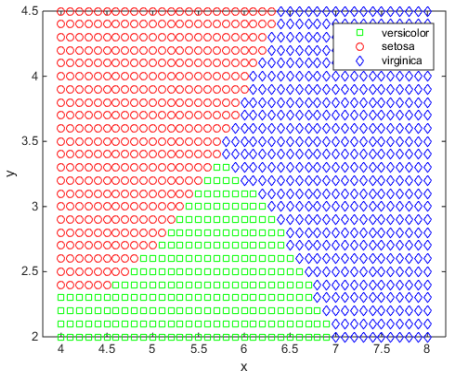
Mark the root node in T as a leaf with the most
common value of y in S as a label.
END IF
END IF
RETURN T
TreePruning (S,T,y)
Where:
S - Training Set
y - Target Feature
T - The tree to be pruned
DO
Select a node t in T such that pruning it
maximally improve some evaluation criteria
IF t6=Ø
THEN T=pruned(T,t)
UNTIL t=Ø
RETURN T

k-NN
Input:
D, the set of k training objects, and test object z=(x',y')
Process:
Compute d(x', x), the distance between z
and every object, (x,y)ЄD.
Select Dz≤D, the set of k closest training objects to z..
Output:y'=argmax∑(xi,yi)ЄDz I(V=yi)

A) Naive Bayes Classifiers

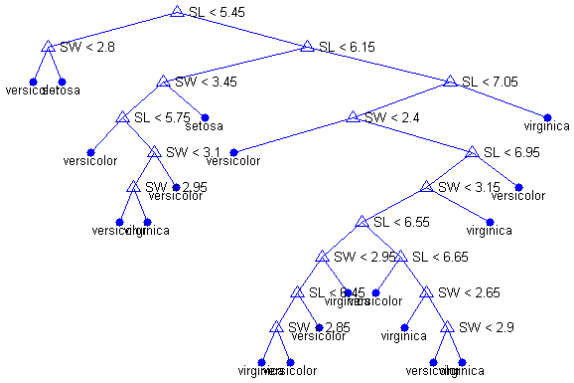
The classify function has other two other types, 'diagLinear' and 'diagQuadratic'. They are similar to 'linear' and 'quadratic', but with diagonal covariance matrix estimates. These diagonal choices are specific examples of a naive Bayes classifier, because they assume the variables are conditionally independent given the class label. Naive Bayes classifiers are among the most popular classifiers. While the assumption of class-conditional independence between variables is not true in general, naive Bayes classifiers have been found to work well in practice on many data sets.
The ClassificationNaiveBayes class can be used to create a more general type of naive Bayes classifier.
First model each variable in each class using a Gaussian distribution.we can compute the resubstitution error and the cross-validation error.

```
nbGau = fitcnb(meas(:,1:2), species);
nbGauResubErr = resubLoss(nbGau)
nbGauCV = crossval(nbGau, 'CVPartition',cp);
nbGauCVErr = kfoldLoss(nbGauCV)
labels = predict(nbGau, [x y]);
gscatter(x,y,labels,'grb','sod')
nbGauResubErr =
        0.2200
nbGauCVErr =
   0.2200
```



So far we have assumed the variables from each class have a multivariate normal distribution. Often that is a reasonable assumption, but sometimes we may not be willing to make that assumption or we may see clearly that it is not valid. Now try to model each variable in each class using a kernel density estimation, which is a more flexible nonparametric technique. Here we set the kernel to box.


```
nbKD = fitcnb(meas(:,1:2), species, 'DistributionNames','kernel', 'Kernel','box');
nbKDResubErr = resubLoss(nbKD)
nbKDCV = crossval(nbKD, 'CVPartition',cp);
nbKDCVErr = kfoldLoss(nbKDCV)
labels = predict(nbKD, [x y]);
gscatter(x,y,labels,'rgb','osd')
nbKDResubErr =
   0.2067
nbKDCVErr =
   0.2133
```



For this data set, the naive Bayes classifier with kernel density estimation gets smaller resubstitution error and cross-validation error than the naive Bayes classifier with a Gaussian distribution.

B) Decision Tree

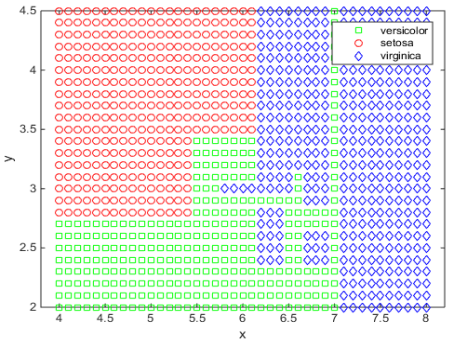
Another classification algorithm is based on a decision tree. A decision tree is a set of simple rules, such as "if the sepal length is less than 5.45, classify the specimen as setosa." Decision trees are also nonparametric because they do not require any assumptions about the distribution of the variables in each class.

The classregtree class creates a decision tree. Create a decision tree for the iris data and see how well it classifies the irises into species.

```
t = classregtree(meas(:,1:2), species,'names',{'SL' 'SW' });
```

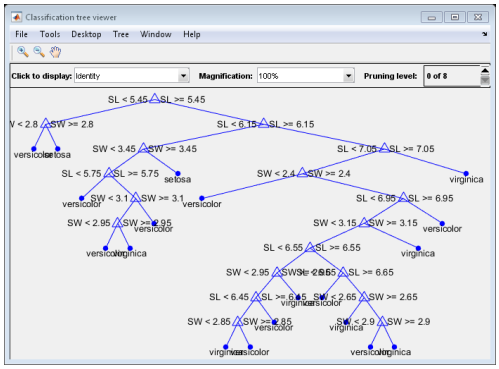
It's interesting to see how the decision tree method divides the plane. Use the same technique as above to visualize the regions assigned to each species.

```
[grpname,node] = eval(t,[x y]);
gscatter(x,y,grpname,'grb','sod')
```



Another way to visualize the decision tree is to draw a diagram of the decision rule and class assignments.

```
view(t);
```



This cluttered-looking tree uses a series of rules of the form "SL < 5.45" to classify each specimen into one of 19 terminal nodes. To determine the species assignment for an observation, start at the top node and apply the rule. If the point satisfies the rule we take the left path, and if not we take the right path. Ultimately you reach a terminal node that assigns the observation to one of the three species.

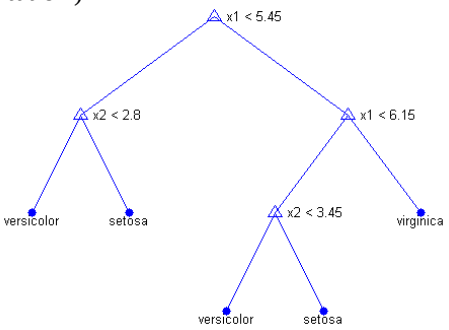
Compute the resubstitution error and the cross-validation error for decision tree.

```
dtclass = eval(t,meas(:,1:2));
bad = ~strcmp(dtclass,species);
dtResubErr = sum(bad) / N
dtClassFun = @(xtrain,ytrain,xtest)(eval(classregtree(xtrain,ytrain),xtest));
dtCVErr  = crossval('mcr',meas(:,1:2),species, ...
        'predfun', dtClassFun,'partition',cp)
dtResubErr = 0.1333 dtCVErr =  0.2933
```

For the decision tree algorithm, the cross-validation error estimate is significantly larger than the resubstitution error. This shows that the generated tree overfits the training set. In other words, this is a tree that classifies the original training set well, but the structure of the tree is sensitive to this particular training set so that its performance on new data is likely to degrade. It is often possible to find a simpler tree that performs better than a more complex tree on new data.

Prune the tree. First compute the resubstitution error for various of subsets of the original tree. Then compute the cross-validation error for these sub-trees. A graph shows that the resubstitution error is overly optimistic. It always decreases as the tree size grows, but beyond a certain point, increasing the tree size increases the cross-validation error rate.

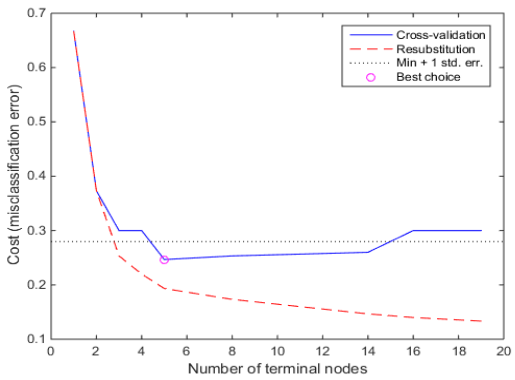
```
resubcost = test(t,'resub');
[cost,secost,ntermnodes,bestlevel] = test(t,'cross',meas(:,1:2),species);
plot(ntermnodes,cost,'b-', ntermnodes,resubcost,'r--')
figure(gcf);
xlabel('Number of terminal nodes');
ylabel('Cost (misclassification error)')
legend('Cross-validation','Resubstitution')
```



A simple rule would be to choose the tree with the smallest cross-validation error. While this may be satisfactory, we might prefer to use a simpler tree if it is roughly as good as a more complex tree. For this example, take the simplest tree that is within one standard error of the minimum. That's the default rule used by the classregtree/test method.

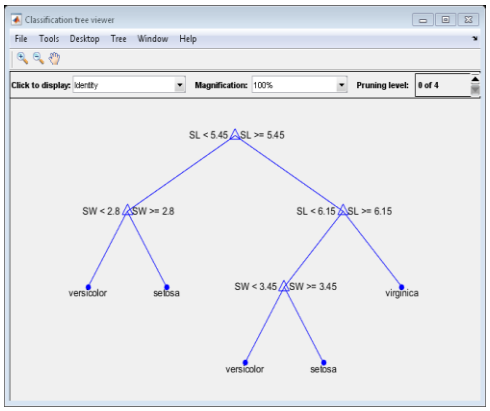
We can show this on the graph by computing a cutoff value that is equal to the minimum cost plus one standard error. The "best" level computed by the classregtree/test method is the smallest tree under this cutoff. (Note that bestlevel=0 corresponds to the unpruned tree, so you have to add 1 to use it as an index into the vector outputs from classregtree/test.)

```
[mincost,minloc] = min(cost);
cutoff = mincost + secost(minloc);
hold on
plot([0 20], [cutoff cutoff], 'k:')
plot(ntermnodes(bestlevel+1), cost(bestlevel+1), 'mo')
legend('Cross-validation','Resubstitution','Min + 1 std. err.','Best choice')
hold off
```



Finally,we can look at the pruned tree and compute the estimated misclassification error for it.

```
pt = prune(t,bestlevel);view(pt)
```

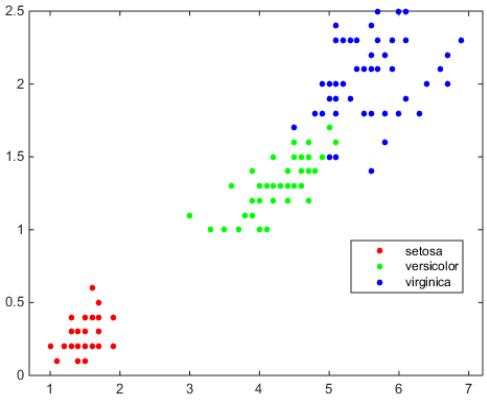



```
cost(bestlevel+1)
ans =  0.2467
```

C) Classifying Query Data Using knnsearch

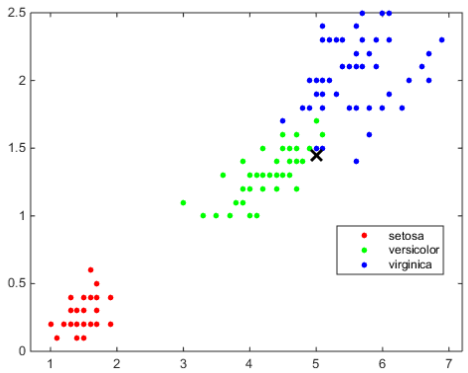
This example shows how to classify query data using knnsearch .It shows how to classify query data using knnsearch .

```
load fisheriris
x = meas(:,3:4);
gscatter(x(:,1),x(:,2),species)
legend('Location','best')
```



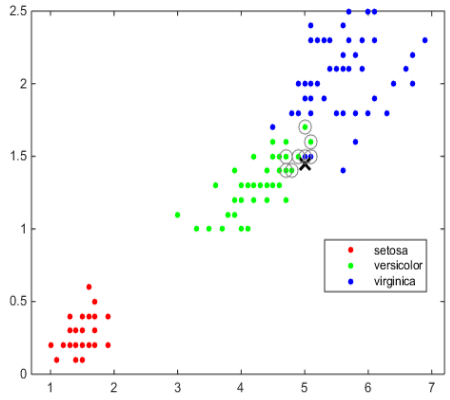
Plot the new point.

```
newpoint = [5 1.45];
line(newpoint(1),newpoint(2),'marker','x','color','k',...
  'markersize',10,'linewidth',2)
```



Find the 10 sample points closest to the new point.

```
[n,d] = knnsearch(x,newpoint,'k',10);
line(x(n,1),x(n,2),'color',[.5 .5 .5],'marker','o',...
   'linestyle','none','markersize',10)
```

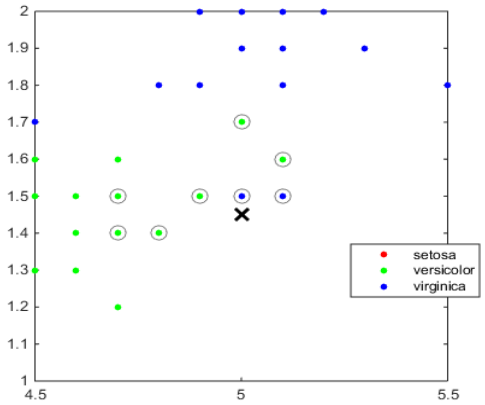



It appears that knnsearch has found only the nearest eight neighbors. In fact, this particular dataset contains duplicate values.

```
x(n,:)
ans =    5.0000    1.5000
   4.9000    1.5000
   4.9000    1.5000
   5.1000    1.5000
   5.1000    1.6000
   4.8000    1.4000
   5.0000    1.7000
   4.7000    1.4000
   4.7000    1.4000
   4.7000    1.5000
```

Make the axes equal so the calculated distances correspond to the apparent distances on the plot axis equal and zoom in to see the neighbors better.

```
xlim([4.5 5.5]);
ylim([1 2]);
axis square
```

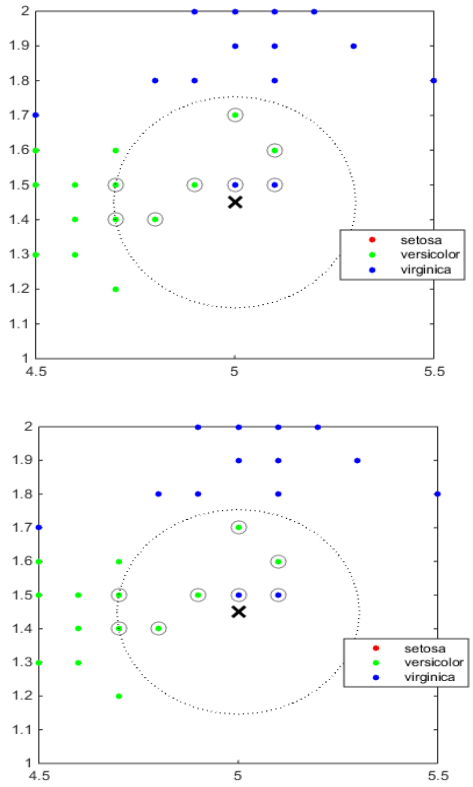


Find the species of the 10 neighbors.

```
tabulate(species(n))
      Value    Count   Percent
  virginica        2    20.00%
 versicolor        8    80.00%
```

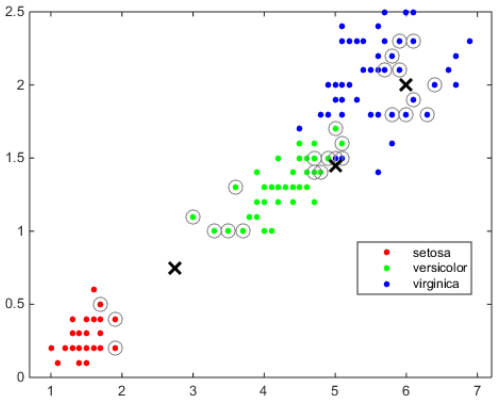
Using a rule based on the majority vote of the 10 nearest neighbors, we can classify this new point as a versicolor.
Visually identify the neighbors by drawing a circle around the group of them. Define the center and diameter of a circle, based on the location of the new point.

```
ctr = newpoint - d(end);
diameter = 2*d(end);
% Draw a circle around the 10 nearest neighbors.
h = rectangle('position',[ctr,diameter,diameter],...
  'curvature',[1 1]);
  h.LineStyle = ':';
```

Using the same dataset, find the 10 nearest neighbors to three new points.

```
figure
newpoint2 = [5 1.45;6 2;2.75 .75];
gscatter(x(:,1),x(:,2),species)
legend('location','best')
[n2,d2] = knnsearch(x,newpoint2,'k',10);
line(x(n2,1),x(n2,2),'color',[.5 .5 .5],'marker','o',...
  'linestyle','none','markersize',10)
line(newpoint2(:,1),newpoint2(:,2),'marker','x','color','k',...
  'markersize',10,'linewidth',2,'linestyle','none')
```



Find the species of the 10 nearest neighbors for each new point.

```
tabulate(species(n2(1,:)))
      Value    Count   Percent
  virginica        2    20.00%
 versicolor        8    80.00%
tabulate(species(n2(2,:)))
      Value    Count   Percent
  virginica       10   100.00%
tabulate(species(n2(3,:)))
      Value    Count   Percent
 versicolor        7    70.00%
     setosa        3    30.00%
```


IV. RESULTS AND DISCUSSION

Marketing Campaign

Value	Count	Percent
no	39922	88.30%
yes	5289	11.70%

We partition the data into training set and test set. The training set will be used to calibrate/train the model parameters. The trained model is then used to make a prediction on the test set. Predicted values will be compared with actual data to compute the confusion matrix. Confusion matrix is one way to visualize the performance of a machine learning technique.

Training Set

Value	Count	Percent
No	23948	88.28%
Yes	3179	11.72%

Test Set

Value	Count	Percent
No	15974	88.33%
Yes	2110	11.67%

Confusion matrix for knn =

93.865	6.135
62.133	37.867

Confusion matrix for Naïve Bayes =

92.018	7.9817
46.256	53.744

Confusion matrix for Decision Tree=

94.072	5.9284
51.611	48.389

Elapsed time is 1.705000 seconds.

Comparison between Naïve Bayes, Decision tree and k-NN

In case accuracy, Naive Bayes is the best. Naïve Bayes is the most accurate classifier compared to Decision Tree and k-NN.Decision Tree has the fastest classification time followed by Naïve Bayes and k-Nearest Neighbor. The differences between classification time of Decision Tree and Naïve Bayes also between Naïve Bayes and k-NN are about an order of magnitude. Decision Trees are very flexible, easy to understand, and easy to debug.

Another parameter for comparing classifier performance is area under the curve (AUC). In this parameter Naïve Bayes is also the biggest among the three classifiers.

V. CONCLUSION

Our experiment shows that Decision Tree is the fastest and k-Nearest Neighbor is the slowest. The fast classification time of Decision Tree because there is no calculation in its classification. Meanwhile k-Nearest Neighbor is the slowest classifier because the classification time is directly related to the number of data. The bigger the data, the larger distance calculations must be performed. This causes the classification is extremely slow.

Naïve Bayes' good performance is caused by the zero-one loss function used in the classification.

ACKNOWLEDGEMENT

I would like to thank Dr.Zahid Ansari, Professor of P.A.Enggineering College for being my advisor and his valuable help to work better.

REFERENCES

[1] G. K. Gupta, Introduction to Data Mining with Case Studies. Prentice Hall of India, New Delhi, 2006.
[2] P-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining. Addison Wesley Publishing, 2006.
[3] J. Han and M. Kamber, Data Mining: Concepts and Techniques. Morgan-Kaufmann Publishers, San Francisco, 2001.
[4] O. Maimon and L. Rokach, Data Mining and Knowledge Discovery. Springer Science and Business Media, 2005.

[5] X. Niuniu and L. Yuxun, "Review of Decision Trees," IEEE, 2010.
[6] V. Mohan, "Decision Trees: A comparison of various algorithms for building Decision Trees," Available at: http://cs.jhu.edu/~vmohan3/document/ai_dt.pdf
[7] T. Miquelez, E. Bengoetxea, P. Larranaga, "Evolutionary Computation based on Bayesian Classifier," Int. J. Appl. Math. Comput. Sci. vol. 14(3), pp. 335 – 349, 2004.
[8] M. K. Stern, J. E. Beck, and B. P. Woolf, "Naïve Bayes Classifiers for User Modeling," Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.979
[9] Wikipedia, "k-Nearest Neighbor Algorithm," Available at: http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm
[10] V. Garcia, C. Debreuve, "Fast k Nearest Neighbor Search using GPU," IEEE, 2008.
[11] J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, 2006.
[12] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of Classification Methods Based on the Type of Attributes and Sample Size," Available at: http://www4.ncsu.edu/~arezaei2/paper/JCIT4-184028_Camera%20Ready.pdf
[13] Wikipedia, "Receiver Operating Characteristics," Available at: http://en.wikipedia.org/wiki/Receiver_operating_characteristic
[14] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers," Kluwer Academic Publishers, Netherland, 2004.
[15] D. Xhemali, C. J. Hinde, and R. G. Stone, "Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages," International Journal of Computer Science Issue, Vol. 4(1), 2009.

