**Naïve Bayes Classifier Pseudo code:**

Naive Bayesian Classification is a supervised learning method for classification. It assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems.

f : frequency tables

i : number of instances

C : Number of classes

n : instances per class

function train(class, train\_set){

# increase the instance count

i++;

# update counts for each class

if class of n == class C : # C = <0, 1> in our case

C++;

end if

for pair(attribute, value) in train\_set:

++F[class,attribute,range] #: increment frequency counts

}

function test(test\_set){

max\_likelihood = -100000;

for j in n:

prior = j/i; # Laplace-estimator could be used in case of low frequency

likelihood = log(prior);

for pair(attribute, value) in test\_set:

k = F[j, attribute, value] + prior/j ;

likelihood += log(k);

if(likelihood>max\_likelihood):

max\_likelihood = likelihood;

class = j

end if

return class;

}

Random Forest:

Random forest is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

Pseudo code:

N: number of training cases

M: number of variables in the classifier

m: input variables to be used to determine the decision at a node of the tree; m << M

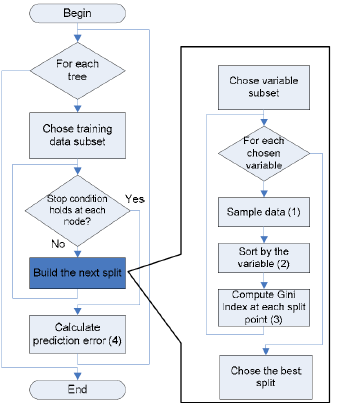
n: bootstrap sample of n with replacement from N in order to train

N-n: test cases

For each node of the tree, randomly choose m variables on which to base the decision at that node.

Calculate the best split based on these m variables in the training set.

Each tree is fully grown and not pruned (as may be done in constructing a normal decision tree classifier).



SVM Classifier:

Given a training set, in supervised learning scenario, an SVM training algorithm builds a model that classifies new examples to one class or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate classes are divided by a finite distance that is as wide as possible. Test data then mapped into that same space and predicted to belong to a class based on which side of the gap they fall.

Let’s take an equation of a linear perceptron:

f(x) = w0 + wTx

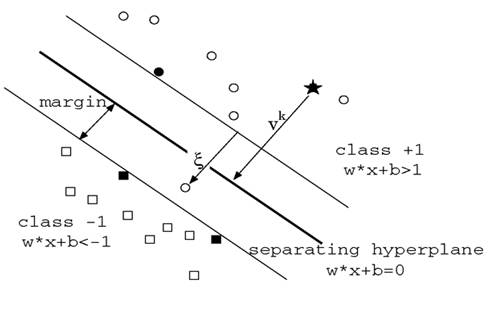
here, w0 = bias and w is known as the weight vector

For support vectors on one side,

w0 + wTx = 1

And for the support vectors on the other side:

w0 + wTx = -1



References:

<http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf>

<https://en.wikipedia.org/wiki/Support_vector_machine>

<http://home.etf.rs/~vm/os/dmsw/Random%20Forest.pptx>