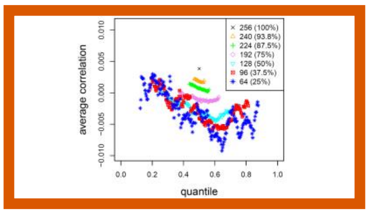
Asociación Española para la Inteligencia Artificial (**AEPIA**)

UIMP Universidad Internacional Menéndez Pelayo

Máster Universitario en Investigación en Inteligencia Artificial



CIENCIA DE DATOS Y APRENDIZAJE AUTOMÁTICO

Practical 2: Model Evaluation

José Hernández-Orallo, DSIC, UPV, jorallo@dsic.upv.es

Based on M.José Ramírez Quintana's material

We are going to work with the tic-tac-toe dataset from the UCI repository (and available on the platform). This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). The dataset consists of 958 instances (65.3% are positive) and 9 attributes (x= has played "x", o= has played "o" (the second player), b= empty):

1. top-left-square: {x,o,b}
2. top-middle-square: {x,o,b}
3. top-right-square: {x,o,b}
4. middle-left-square: {x,o,b}
5. middle-middle-square: {x,o,b}
6. middle-right-square: {x,o,b}
7. bottom-left-square: {x,o,b}
8. bottom-middle-square: {x,o,b}
9. bottom-right-square: {x,o,b}
10. Class: positive,negative

To create the models we are going to use the caret package (http://topepo.github.io/caret/index.html). This package provides useful tools for data splitting, model generation and model evaluation, among others. We will construct our models by using cross-validation (with 10 folds and 1 repetition) and we will learn 5 different classification models: a Naïve Bayes, a Decision Tree, a Neural Network, k-Nearest Neighbours and a Support Vector Machine (with a linear kernel), using the training data. Finally we evaluate our models with respect to the test set.

1. Load the data into R. Name the columns to better identify the board, as visited from left to right and from top to down. Check for missing values.

2. Read the "data splitting" section at the web page of caret. Then split the data into 70% training and 30% test by keeping the original proportion of classes.

3. Read "the model training and parameter tuning" section at the web page of caret to know how to train a model using cross validation. Read also the list of models available at the "Sortable Model List" link in order to know the name of the method that implements each technique we are going to apply as well as the package you must install to use it. Remember to use the same seed for training the models in order to make a fair comparison of their performances. Do not modify the tuning parameters chosen by default for the *train* function. The values of the measures used for selecting the best model (by default, the accuracy and/or the kappa measures are used for classification models) are printed by typing the name of the model in the console. An alternative way of collecting the resampling results is to use the *resamples* function (from the caret package). Complete the following table with the final values of accuracy and kappa for the training data:

	Accuracy	Kappa
Naive Bayes		
Decision Tree		
Neural Network		
Nearest Neighbour		
SVM (linear kernel)		

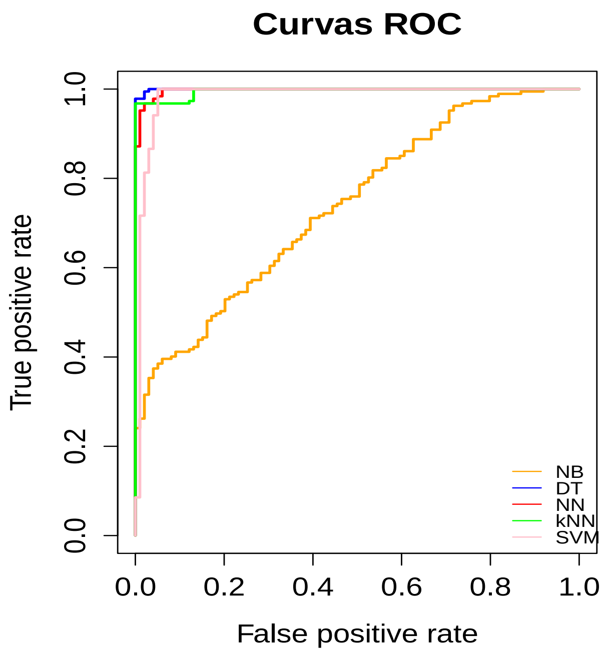
4. Read the "model performance" section at the web page of caret. Apply the models to the test dataset. Print the confusion matrix of each model and observe the information it provides. Construct a table (similar to the previous one) with the accuracy and kappa values obtained by each model on the test set (to do this you can also use the *postResample* function). Add to the table the AUC value. It can be calculated using the *AUC* package (https://cran.r-project.org/web/packages/AUC/AUC.pdf).

5. Plot the ROC curves of the models. We are going to use the *ROCR* package (http://rocr.bioinf.mpi-sb.mpg.de/ROCR.pdf):
   a. First, we calculate again the predictions on the test set but now setting the type parameter of the *predict* function to "prob", in order to generate a probabilistic classifier (a probability estimator).
   b. We construct a "*prediction*" object for each classifier using the vector of estimated probabilities for the positive class as the first

parameter, and the vector of actual class labels as the second parameter.

c. We calculate the measures we want to plot on the y-axis (TPR) and on the x-axis (FPR) by using the *performance* function.
d. Draw all the curves in the same plot.

The curves should look like this:


Curvas ROC
True positive rate
False positive rate
NB
DT
NN
kNN
SVM