Máster Oficial Universitario en
Ingeniería Informática
muiinf.webs.upv.es

Data Science (CDA)

Practical 7:
Model evaluation

José Hernández Orallo (jorallo@dsic.upv.es)
(Adapted from material by M. José Ramírez Quintana)

In this session we are going to work on some issues related to evaluation as seen at the end of Unit 3.

We are going to work with the tic-tac-toe dataset from the UCI repository (and available on poliformat). 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):

- top-left-square: {x,o,b}
- top-middle-square: {x,o,b}
- top-right-square: {x,o,b}
- middle-left-square: {x,o,b}
- middle-middle-square: {x,o,b}
- middle-right-square: {x,o,b}
- bottom-left-square: {x,o,b}
- bottom-middle-square: {x,o,b}
- bottom-right-square: {x,o,b}
- 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. This package requires package 'pbkrtest', which is not available for R version <= 3.2.2. If your version or R is old, update R first.

We will construct our models by using cross-validation (with 10 folds and 1 repetition) and we will learn 5 different classification models: a Naive Bayes, a Decision Tree, a Neural Network, k-Nearest Neighbors and a Support Vector Machine (with a linear kernel), using the training data. Finally, we will evaluate our models with respect to the test set.

Exercises

1. Load the data into R and name the columns to better identify the board (following an ordering from left to right and from top to bottom). Check for missing values.

2. Read the "data splitting" section on the caret web page. Then split the data into 70% training and 30% test by keeping the original class proportion (check “createDataPartition()” function).

3. Read "the model training and parameter tuning" section (http://topepo.github.io/caret/model-training-and-tuning.html) on the caret web page to know how to train a model using cross validation (i.e., use “trainControl()” method; the classProbs argument of the trainControl object must be set to TRUE). Also read the list of models available at the "Sortable Model List" link (http://topepo.github.io/caret/available-models.html) 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 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. Complete the following table with the final values of accuracy and kappa used for the models:

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

4. Read the "model performance" section on the caret web page. 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 the kappa values obtained by each one (to do this you can also use the postResample function). Add the AUC value to the table. 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:

   a. First, we calculate again the predictions on the test set but now setting the "type" parameter of the predict function to "prob".

   b. We construct a "prediction()" object for each classifier using the vector of estimated probabilities for the positive class as the first parameter (calculated above), 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, which also takes the "prediction()" object computed above as a first parameter.

   d. Draw all the curves in the same plot using different colours. Add a legend.

   e. Which models should you keep, and which models should you discard, according to the ROC curves?