Proposed ML exercises - "pre-practising" with root concepts

[Salta al contenido principal](https://egela.ehu.eus/mod/page/view.php?id=4325206&inpopup=1#maincontent)

**The following**[**link**](http://www.sc.ehu.es/ccwbayes/master/selected-dbs/nlp-naturallanguageprocessing/)**saves a set of NLP datasets, most of them in WEKA's arff format: there is a link in our "egela" webpage which describes each dataset. Remember that \***[**.arff ("Attribute Relation Format File"**](http://www.cs.waikato.ac.nz/ml/weka/arff.html)**is the standard to codify datasets in WEKA). They are a good option to complete the following exercises. The**[**parent directory**](http://www.sc.ehu.es/ccwbayes/master/selected-dbs/)**saves more datasets of non-NLP domains for different types of analysis.**  
  
**[Exercise 1 -- Supervised classification -- Practising with WEKA -- Prediction of the class-label of unseen sample]**  
  
A good exercise to check your increasing abilities in WEKA is to train a classifier in the "Explorer" interface with labeled samples: and then, use this model to predict the label of unseen-unlabeled samples. This process is also known as "categorization" or "class prediction". Take into account that this process is a kind of "bet", as the model, when it performs the prediction, does know whether its prediction is correct or incorrect.  
  
Let's simulate this process in WEKA's Explorer interface. Load a labaled supervised dataset of your choice in the main "Preprocess" tab. Any classification model is always learned with these samples.  
  
Then, load a second WEKA's \*.arff file composed of unlabeled samples by means of the second "Validation" option of the "Classify" tab: "Test options --> supplied test set".  There are many ways for constructing this dataset. One is to build the unseen samples of your invention from the scratch. You can also modify labeled training samples. Always write the "?" (missing value) symbol as the value of the class label. 5-10 unseen samples can be enough. This second file needs to have the same attribute definition of the first supervised-training file.  
  
Select a supervised classification technique and run. 10-NN or naive Bayes are a good option to explain the output. As the real class of these unlabaled samples is not known, it is not possible to estimate the accuracy of any classifier with these unlabeled instances.   
  
In order to check the predicted labels-class-values for the uncategorized samples it is needed to activate the "Output Predictions" option in the button. It is needed to understand the meaning of the different columns of the output: "actual", "predicted", "probability distribution"...   
  
On the other hand, it is possible to estimate the accuracy of any classifier using only the set of labeled instances. WEKA provides two accuracy estimation techniques to do so: cross-validation and percentage split (also known as hold-out or train-test).

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 2 -- Accuracy estimation of supervised classifiers in WEKA]**

Another key exercise in machine learning: test the accuracy of supervised classifiers. Select a dataset and a classification technique of your interest.  Estimate its global accuracy in different ways. Focus also on the TPR (true positive rate - recall -- sensitivity), FPR (false positive rate) and Precision values of a class-label value of your interest (e.g. "spam" class-label). Remember that WEKA always first learns the final model with all labaled samples, and then applies the validation procedure selected by the user. That is, in the case of a 10-fold cross-validation first learns the final model with all labeled samples; then, 10 sub-models are learned with 90% of the samples, and each tested in the remaining 10%. Information of both processes is output;

* test the classifier in the same set that is trained on;
* percentage split (hold-out), 2/3 train - 1/3 test;
* previous hold-out procedure, repeated 5 times, averaging the results. In order to randomize in each run the samples selected for train or test splits, it is needed to change the "seed" of randomization: see "More options"--"Random seed";
* 10-fold cross-validation;
* 2 times 10-fold cross-validation;
* 5-fold cross-validation;
* leave-one-out: which value needs to be fixed in "folds"?

1. do all estimation methods return the same value? Why?
2. which method returns the best estimated value? Was it expected?
3. based on your intuition, which of the methods is the most reliable?
4. which of the previous methods is the most CPU demanding?
5. is it possible that "classifier A is aparently better than classifier B" for an specific seed value, and then the opposite happens for another seed value (i.e."classifier B is aparently better than classifier A")? how can this be alleviated?

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 3 -- Preprocessing techniques]**

Select a supervised dataset of your interest. We are going to apply different preprocessing techniques. Before starting, take into account the following comments about the application of Preprocessing techniques in WEKA:

* WEKA's Preprocessing techniques can be found in the "Filter-Choose" button of the first WEKA's "Preprocess" tab. Preprocessing techniques are divided by "supervised" or "unsupervised" terms, depending whether they make use of the class-label to complete the preprocessing task. Techniques are further divided by "instances" or "attributes" terms, depending whether they are applied over the samples-rows or [variable](https://egela.ehu.eus/mod/glossary/showentry.php?eid=18228&displayformat=dictionary)-columns of the WEKA file;
* parameter's of any preprocessing filter can be accessed, after selecting it, by clicking in its bold-letter name. Button "More" gives more information about the parameters. They can be modified;
* the selected filter is not applied until the "Apply" button is clicked;
* check the effect of the filter by consulting the histogram and the descriptive-properties of the features-variables.
* the effect of the filter can be removed by clicking the "Undo" button;
* "Edit" button allows to do slight modifications of the instance-variable values.

Among the large plethora of filters provided by WEKA, I propose to apply the following. It is likely that not all of them are useful in NLP domains:

1. The “unsupervised—Instance—**Resample**” filter. Try to understand it.
2. The “unsupervised--Attribute--**AddNoise**” filter. Try to understand it.
3. The “unsupervised--Attribute--**NumericToBinary**” filter. Try to understand it.
4. The "unsupervised--Attribute-**InterquartileRange**" filter. Try to understand it.
5. The “unsupervised-Instace-**RemoveMisclassified**” filter. Try to understand it.
6. The “unsupervised-Instace-**RemoveWithValues**” filter. Try to understand it.
7. The “unsupervised-Instace-**RemovePercentage**” filter. Try to understand it.
8. The "**MultiFilter**" option.
9. **Discretization** is a common filter in classic numerical domains.  However, I am not sure about the usefulness of the popular “unsupervised—Attribute—**Discretize**”  and “supervised—Attribute—**Discretize**” filters in NLP domains. Based on your NLP expert opinion, is it useful to, instead of working with continuous worcounts values (e.g. "4", "7", "1"), discretize these continuous values (e.g. "low","medium","high" wordcount frequencies)? Choose a dataset with numerical variables from this [directory](http://www.sc.ehu.es/ccwbayes/master/selected-dbs/supervised-classification/) and practice with the exposed filter. Analyze its effect in the histogram.
10. Take a look for the "ReplaceMissingValues" filter. Based on your NLP expertise, does it apply in NLP domains? Do missing values appear in NLP domains? Choose a dataset from this [directory](http://www.sc.ehu.es/ccwbayes/master/selected-dbs/supervised-classification/) which has missing values in at least one of its variables and practice with the exposed filter. Analyze its effect.
11. Discover by yourself other filters.
12. The "unsupervised--Attribute-**StringToWordVector**" filter is specially designed to deal with string variables and convert them in a vectorized word-attribute form. Select any file provided in our working directory which has a single string feature codifying the text. Note that the samples are also annotated-labeled. WEKA can not directly deal with string-format attributes. Try to understand the file: visit and understand its parameters. As NLP specialists, you will understand them. It is common to apply, after the **StringToWordVector** operator, the **NumericToBinary** filter, which accounts for the presence/absence of a word (or n-gram in a document).

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 4 -- Data preprocessing -- Classifying class unbalanced datasets]**

Supervised (annotated) datasets where the class-label distributions are highly unbalanced are common in NLP (and other) domains.

* TeleecomTweets dataset offers an interesting unbalanced tasks. Take a look to its class distribution. I recommend to apply the StringToWordVector and NumericToBinary filters before learning classification models.
* An example can be found in the provided "spamversusnospam" dataset, where the large majority of samples belong to a class-label, while a minority of the samples is spread along the rest of the class-labels.

In this type of scenarios it is hard to correctly classify the instances of the minority classes. While the accuracy metric (globaly correctly classified percentage of samples) is an unrealistic measure in this type of scenarios, the sensitivity-recall-TPR (true positive rate) in the minority class(es) is the correct indicator to measure the number of samples of the minority classes correctly classified. As a single populated class "dominates" the dataset, a large TPR in this class will provide a high accuracy ratio, and provide a low number of correclty classified samples in the minority class(es) which does not affect the global accuracy.  

WEKA offers the "CostSensitiveClassifier" ("meta" family) to approach this problem in a simple way: 

* choose a base classifier (e.g. naive Bayes);
* choose an option to test the classifer, e.g. 10-fold cross-validation;
* we tune the values of the cost matrix. Values in the principal diagonal need to be zero, while the other two values are related to both type of errors: while one of these values will be fixed to 1, the other value will represent "how many times the samples of the minority class will be repeated  to train-learn a classifier". By repeating the samples of the minority class, classifiers tend to improve their TPR in the minority class, while reducing it in the majority class. This repetition of the samples is transparent to the user, as the final accuracy percentages are calculated with the original proportion of samples: check the values in the confusion matrix;
* tricks to fix the cost matrix. First, "resize" it with the number of classes of your problem. Second, after fixing a cost value in the matrix, it is needed to click the "tab" key to be fixed;

Check how, depending on the provided costs, TPR changes in the minority class. Does this affect the TPR in the majority class? 

WEKA also offers another option. WEKA's package manager offers the possibility to install the "SMOTE" package, which implements this popular minority-class-oversampling technique. Install the package. The SMOTE function will be located in "Preprocess-supervised-instances". Check its parameters. Apply it. Check how data class distribution changes. Check the performance of different classifiers before and after applying SMOTE. 

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 5 -- Feature selection -- Univariate feature selection -- Feature ranking]**

Choose an NLP dataset of your interest. 

WEKA has a large set options to complete different "feature selection" operations in the "SelectAttributes" tab. As many NLP datasets have a large number of variables (n-grams), the application of dimensionality reduction techniques is mandatory. Remember that while other techniques such as "Principal Component Analysis" do this reduction by constructing new variables, the "feature selection" approach works with original features by selecting a proper subset to construct the final model. 

If you are interested in the feature selection topic the following links [[1](http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf),[2](http://bioinformatics.oxfordjournals.org/content/23/19/2507.short)] provide introductory readings. 

WEKA has different metrics to measure the correlation degree between any feature and the class-label. I recommend you the following ones: "information gain", "information gain ratio", "symmetrical uncertainty", "chi-square". First three metrics are based on the "Information Theory" and they are popular in NLP studies: their formula can be consulted by clicking the "More" button after clicking in the name of the selected metric. Chi-square is a classic statistical metric (consult the Wikipedia). All the metrics need nominal-form (not numeric) features. Thus, it is a good idea to convert the features to a binary-form by means of the "NumericToBinary" preprocessing filter. 

Once any of these metrics is selected, predictors-features will be ranked by means of their correlation degree with the class-label. 

Do all metrics construct the same ranking? Why? 

As a way to provide more stable rankings and mitigate this discrepancy between them, it is common to calculate a "consensus-average ranking". ["Stability on feature selection"](http://cui.unige.ch/%7Ekalousis/papers/stability/KalousisPradosHilarioKIS2007.pdf) is a hot research subject among feature selection technique developers. 

Choose a classifier. Choose an accuracy estimation technique. Follow the next steps: 

* choose a ranking metric and learn a classifier with the "top-5", "top-10", "top-20"... most correlated variables with the class-label. This can be done in two ways in the WEKA interface:
  + the "numToSelect" parameter in the "Ranker" option allows to select a subset of most correlated features. Right-click in the result-list and activate the "Save reduced data" option. This will construct a new \*.arff file with the set of most correlated features and the class-label;
  + the "meta-AttributeSelectedClassifier" in the "Classify" tab also covers this process in a quick way;
* learn a classifier with all the original features;
* is the accuracy improved when working with a subset of features? in all class-labels? Study the TPR in all class-labels;
* did we measure the accuracy of the classifier in a honest way? Take a look to one of the next exercises. We maybe will complete it further, but please reflect about this.
* do this type of rankings measure the level of correlation-redundancy between the features of the ranking? does this approach avoid the inclusion of redundant features in the final classifier?

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 6 -- Feature selection -- Combining univariate + multivariate (multivariate-filters-CFS and wrappers)]**

NLP datasets with hundreds or thousands of features are common. It is popular in NLP studies to first reduce the number of features by means of the exposed feature-ranking techniques. Because of the computational burden, it wouldn't be possible to execute a multivariate feature selection technique over the initial, high-dimensional feature set.

Thus, by means of a univariate feature selection correlation metric, it is common to reduce the dataset to the most correlated top-50, top-100, top-200, etc. features. Over this reduced dataset, most complex, rich and computationally intensive multivariate feature selection approaches can be applied: these approaches try to discover the redundancies among the features and only include "complementary" features in the final feature subset. This implies the use of an optimization heuristic (genetic search, greedy stepwise, etc.) to perform a search in the space of possible feature subsets. As an exhaustive search which guarantees the output of the optimal subset is computationally unfeasible, a suboptimal (but competitive) feature subset is returned. Take into account that in a dataset with "d" features there are 2ddifferent subsets.

WEKA offers the following popular multivariate feature selection approaches:

1. The “Correlated Feature Selection” (CFS) tries to explicitly minimize the correlation degree (redundancy) among selected features, while maximizing the correlation with the class-label. For this purpose, any of the exposed correlation metrics can be used.
2. The "WrapperSubsetEval" approach performs an heuristic search in the space of possible feature subsets by estimating the accuracy percentage of an specific classifier. Selected features are tailored to this specific classification technique (e.g. naive Bayes, 10-NN, etc.). Consult and understand the "classifier" and "folds" parameters of the wrapper approach.

Select a high-dimensional, challenging NLP dataset. Choose and accuracy estimation (validation) technique. Complete in a free way, hybridizing both univariate and multivariate feature selection approaches, different computations with WEKA. Try to enlarge your knowledge and shed light, at least, to the following questions: 

* + is it computationally affordable to apply a multivariate feature selection technique over the initial dataset and its original dimensionality?
  + do the finally selected features change in the wrapper approach when changing the classifier?
  + how does WEKA estimate the accuracy of different feature subsets in the wrapper approach?
  + for both CFS and wrapper approaches, does the final accuracy change when selecting different optimization heuristics to perform the search in the space of possible features?
  + how does a classifier constructed with the features selected by the CFS approach compare with the classifiers learned with feature ranking metrics of previous exercises?

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 7 -- Honest accuracy estimation]**

Recent papers alert about the need to be cautious to honestly estimate the percentage of correctly classified samples of another measure of classifier's goodness. Among them the following papers [ [Reunanen'03](http://www.jmlr.org/papers/volume3/reunanen03a/reunanen03a.pdf)] [[Smialowki'2009](http://bioinformatics.oxfordjournals.org/content/26/3/440.full)] show a clear abstract. Take into account the following facts:

* do not forget that the final supervised classification model is always constructed applying our "analysis-pipeline" to all labeled-annotated samples. The pipeline can consist of several concatenated preprocessing filters, followed by a feature ranking step selecting top-k features and a final classifier modelization;
* it is needed to estimate the accuracy of the exposed analysis-pipeline: not only the final classifier modelization step. How?
  + let's assume that [2/3 train - 1/3 test] procedure is used to test the previous pipeline;
  + if we first construct a feature ranking with all the samples and then do the [2/3 train - 1/3 test] process for accuracy estimation, this process can not be qualified as "honest". Why?
  + remember how a [2/3 train - 1/3 test] works. It assumes that the samples of the testing subset have not been used to take any "decision" of the analysis-pipeline. Is this true in our case?
    - no. "All the samples" (not only those in the 2/3 train) have been used to learn the feature ranking and select the top-k ones.
    - this will raise an optimistic (and not honest) accuracy estimation;
  + in order to honestly estimate the accuracy of our entire pipeline, it is needed to repeat the whole proposed pipeline (feature ranking + classifier) in the 2/3 train subset, and test its accuracy in the remaining 1/3 test subset: this has not been used to construct the feature ranking, neither for training the classifier;
* a similar reasoning is applied for a 10-fold cross-validation procedure, repeating the entire analysis pipeline 10 times, in each different set of 9-folds;
* both "Classifier"-"meta-"AttributeSelectedClassifier" and "Classifier"-"meta-"FilteredClassifier" schemes implement the exposed honest accuracy estimation process;
* you can test by your own that both accuracy estimation procedures ("honest" versus "non-honest") do not output the same estimated accuracy, while the non-honest one returns an optimistic estimation.

**---------------------------------------------------------------------------------------------------------------------------------------------**

**[Exercise 8 -- Statistical comparison between the accuracy of two classification models in a dataset]**

Choose a dataset of your interest. Choose two different classification techniques. As accuracy estimation 5 runs of 10-fold cross-validation will be used: changing in each run the seed in order to randomize the samples.   
Compare both classifiers with a proper statistical test, extracting conclusions such as   
  
"the significance of the differences between compared techniques is p-value=... This means that the results of both compared classifiers can be considered statistically different (or not)..."  
  
Take into account that if the same seed is used for both classifiers in each run, the samples with the accuracy percentages of both classifiers are "paired": that is, the first run (accuracy) of classifier1 with the first run (accuracy) of classifier2. This conditions the selection of the statistical test to compare both classifiers: "2-samples paired t-test" for parametric tests and "Wilcoxon test" for non-parametric tests. For "unpaired" samples, "2-samples independent t-test" for parametric tests and "Mann-Whitney" test for non-parametric tests.   
  
The selection of a parametric or non-parametric test needs other explanation. In order to do a parametric test which is based on the used of "mean" and "standard deviation" parameters, it is needed to test whether the accuracy samples follow a Gaussian distribution. The [Wilks-Shapiro](http://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test) test is designed for this purpose. Different online websites and R software implement it. If the Gaussian distribution is fulfilled the selection of a parametric test is justified; otherwise, a non-parametric test should be applied.   
  
There are different resources which implement the exposed tests: 

1. [VassarStats.net](http://vassarstats.net/)
2. R-software: [parametric tests](http://www.statmethods.net/stats/ttest.html), [non-parametric tests](http://www.statmethods.net/stats/nonparametric.html)
3. Excel - Calc (OpenOffice) implement both paired and independent t-tests

Remember that they keypoint of an statistical test is the p-value. The p-value is a probability, i.e. a value between 0 and 1. It can be translated to percentages. It shows the significance of the difference between both classifiers. The null hypothesis of the exposed tests is that "both classifiers have similar accuracy"; on the other hand, the alternative hypothesis asserts that "classifiers are not similar and the one with the better average accuracy is  considered as superior". The p-value can be interpreted as the "strength" of the null-hypothesis. A small p-value (commonly smaller than 0.05) implies that the null-hypothesis should be rejected and the differences between both classifiers should be considered as "significant".

Última modificación: lunes, 14 de diciembre de 2020, 09:27