NAME

bayes_fss - Select an optimal feature subset for Naive Bayes classification

SYNOPSIS

bayes_fss [options] [--] <dataset>

DESCRIPTION

Overview

Bayes FSS is a program for selecting optimal features subsets for Naive Bayes classification. It generates and evaluates several features subsets in turn, looking for the best performing one. When the search algorithm has come to completion or the program is interrupted, a summary of the performance of the best classifier found so far is displayed on the standard output in JSON notation.

Main options

−m, −−mode=<binary|multiclass> [multiclass]

Classification mode. Applying a binary classifier to a dataset that includes more than two labels is valid and allowed. If binary classification is chosen, the **−−truth** option must also be given.

−t, −−truth=<label_name>

Name of the label corresponding to the positive class, when doing binary classification.

−k, −−folds=<integer> [5]

Number of folds for cross-validation. Must be strictly greater than one and smaller than or equal to the number of samples.

−S, −−smooth=<float> [0.5]

Frequency increment for additive smoothing. Must be strictly greater than zero.

−s, −−search=<none|forward|forward-join|backward|backward-join> [forward-join]

Search strategy.

none Don’t search the attribute space, merely measure the performance of the classifier that uses the full attributes set. The obtained results can be used as a baseline for comparison.

forward Start with the empty feature set and incrementally try to add features. Stop when no improvement can be made.

backward Same as **forward** , but start with the full feature set and progressively remove features instead.

forward-join Same as forward, except that, in addition to checking if the addition of a feature improves the model, also check if joining this feature with one of the selected features would be better, and choose the best solution.

backward-join Same as backward, except that, in addition to checking if the removal of a feature improves the model, also check if joining two selected features would be better, and choose the best solution. This is the most expensive search method.

−M, −−measure=<accuracy|precision|recall|F1> [F1]
Measure to maximize.

−a, −−average=<micro|macro> [macro]
Averaging mode to use for performance measurement when doing multiclass classification.

−F, −−max-features=<integer> [inf]
Maximum number of features to select. This is the number of original features: joined features, if any, count as the number of original features they join. This option can only be used when **−−search** is set to **forward** or **forward-join.**

−L, −−max-links=<integer> [inf]
Maximum number of features dependencies to model. This option can only be given if **−−search** is set to **forward-join** or **backward-join.** If, for example, **−−max-links** is set to 1, the search algorithm will attempt to combine up to two features. The default is to not restrict the number of features dependencies.

Output options

−v, −−verbose
Output performance measures for each model evaluated instead of merely for the best performing one.

−c, −−compact
Compress the output of the program so that JSON documents fit on a single line. Per default, JSON documents are pretty-printed.

General options

−h, −−help
Display a short help message.

−−version
Display the current version.

INPUT FORMAT

A file containing tab-separated values is expected as input. The first line of the file must contain the features names. Remaining lines should contain extracted features for each sample, preceded by the sample label. For example:

	tail	num_legs	wings	continent
mammal	yes	4	no	africa
reptile	yes	0	no	asia
mammal	no	2	yes	asia

Empty fields are not allowed. If a feature value is missing for some samples, you can either insert a dummy unique value, or remove these samples.

It is important to ensure that your data is not nicely sorted per label, as can happen if it was generated automatically, because this can screw up cross-validation. To shuffle an existing dataset, you can use the following:

```
$ head -n 1 dataset.tsv > shuffled.tsv
```

```
$ tail -n +2 dataset.tsv | sort -R >> shuffled.tsv
```

OUTPUT FORMAT

The program output looks as follows:

```
{
  "subset": ["tail",["num_legs","wings"],"continent"],
  "accuracy": 98.891013,
  "precision": 97.564507,
  "recall": 87.069047,
  "F1": 91.612369,
  "subsets_evaluated": 12,
  "interrupted": false
}
```

The most important field is "subset", which gives the best performing subset found so far. It is set to the empty JSON array if the best solution is to not use any features. Dependent features are represented as nested JSON arrays. For example, the above summary indicates that the features "num_legs" and "wings" depend on each other and should be merged together. Applying the necessary modifications to our dataset, we obtain:

	tail	num_legs+wings	continent
mammal	yes	4+no	africa
reptile	yes	0+no	asia
mammal	no	2+yes	asia

And the formula for computing the probability of a label given features values becomes:

$$P(label|tail,num\_legs,wings,continent) \propto P(label)\,P(tail|label)\,P(num\_legs,wings|label)\,P(continent|label)$$

COPYRIGHT