Contents

Notes on feature selection

Notes on basic feature selection methods.

1.0. From `scikit-learn` User Guide

`scikit-learn` has a feature selection section in its user guide (link), which provides basic feature selection functionalities.

1.1. Remove low-variance features

- Implemented in `sklearn.feature_selection.VarianceThreshold`, . Notice that variance threshold is not just arbitrary and could have statistical basis.
  - For example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are **Bernoulli** random variables, and the variance of such variables is given by $\mathrm{Var}[X] = p(1-p)$ where $p = 0.8$.

1.2. Univariate feature selection

Commonly used best univariate feature selectors include the following:

- `SelectKBest(score_func=<function f_classif>, k=10)` removes all but the k highest scoring features.

- `SelectPercentile(score_func=<function f_classif>, percentile=10` removes all but a user-specified highest scoring percentage of features.

Common scoring functions based on univariate statistical tests are as follows:

- For regression tasks (continuous target):
  - `f_regression` computes the p-value and F-value of fitted univariate regression model between feature and continuous target.
  - `mutual_info_regression` estimates mutual information for a continuous target variable.
- For classification tasks (label target):
  - `chi2` computes chi-squared stats between each non-negative feature and class
  - `f_classif` computes the ANOVA F-value for the provided sample.
  - `mutual_info_classif` estimates mutual information for a discrete target variable.
- An example of the comparison between `f_regression` and `mutual_info_regression` is linked here, and key result is that mutual information works for non-linear dependencies whereas F-value or p-value only captures linear dependency.
- **TODO**: *Read on and eventually write a summary/note on mutual information, and Kullback–Leibler divergence*

1.3. Recursive feature elimination (RFE)

- `sklearn.feature_selection.RFE` selects features by recursively considering smaller and smaller sets of features based on importance obtained either through a `coef_` attribute or through a `feature_importances_` attribute.
- `sklearn.feature_selection.RFECV` performs RFE in a cross-validation loop to find the optimal number of features.

1.4. Model-based feature selection

- `sklearn.feature_selection.SelectFromModel` is a meta-transformer that can be used along with any estimator that has a `coef_` or `feature_importances_` attribute after fitting. Features with corresponding `coef_/feature_importance_` values below the provided threshold are removed.
- Typical model-based feature selection approaches include:

- ℓ_1-based methods (e.g., LASSO, Support Vector Machine with regularization), and
- tree based methods that provide `feature_importances_`

1.5. Feature selection as a pipeline

Feature selection is usually used as a pre-processing step before doing the actual learning. The recommended way to do this in `scikit-learn` is to use a `sklearn.pipeline.Pipeline` as follows:

```
clf = Pipeline([
  ('feature_selection', SelectFromModel(LinearSVC(penalty="l1"))),
  ('classification', RandomForestClassifier())
])
clf.fit(X, y)
```

References

- Scikit-learn User Guide 1.13. Feature selection¶
- wiki - Mutual Information