1. Correlation Matrix

A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to see which pairs have the highest correlation.

2. Feature Importance

Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features.

3. Principal Component Analysis

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form.

Generally this is called a data reduction technique. A property of PCA is that you can choose the number of dimensions or principal component in the transformed result

4. Recursive Feature Elimination

The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.

It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

5. Univariate Selection

Statistical tests can be used to select those features that have the strongest relationship with the output variable.

6. Removing features with low variance

`VarianceThreshold` is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

As an 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

$$\mathrm{Var}[X] = p(1-p)$$

so we can select using the threshold `.8 * (1 - .8)`:

7. Feature selection using SelectFromModel

SelectFromModel is a meta-transformer that can be used along with any estimator that has a coef_ or feature_importances_ attribute after fitting. The features are considered unimportant and removed, if the correspondingcoef_ or feature_importances_ values are below the provided threshold parameter. Apart from specifying the threshold numerically, there are built-in heuristics for finding a threshold using a string argument. Available heuristics are “mean”, “median” and float multiples of these like “0.1*mean”.

8. L1-based feature selection¶

Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. When the goal is to reduce the dimensionality of the data to use with another classifier, they can be used along with **`feature_selection.SelectFromModel`** to select the non-zero coefficients. In particular, sparse estimators useful for this purpose are the **`linear_model.Lasso`** for regression, and of **`linear_model.LogisticRegression`** and **`svm.LinearSVC`** for classification:

9. Randomized sparse models

In terms of feature selection, there are some well-known limitations of L1-penalized models for regression and classification. For example, it is known that the Lasso will tend to select an individual variable out of a group of highly correlated features. Furthermore, even when the correlation between features is not too high, the conditions under which L1-penalized methods consistently select “good” features can be restrictive in general.

10. Tree-based feature selection¶

Tree-based estimators (see the **`sklearn.tree`** module and forest of trees in the **`sklearn.ensemble`** module) can be used to compute feature importances, which in turn can be used to discard irrelevant features (when coupled with the **`sklearn.feature_selection.SelectFromModel`** meta-transformer)

11. Feature selection as part of 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`**:

12. Linear Discriminant Analysis (LDA)

LDA is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (“curse of dimensionality”) and also reduce computational costs.

The general LDA approach is very similar to a Principal Component Analysis (for more information about the PCA, see the previous article **Implementing a Principal Component Analysis (PCA) in Python step by step**), but in addition to finding the component axes that maximize the variance of our data (PCA), we are additionally interested in the axes that maximize the separation between multiple classes (LDA).

All the codes for this methods in python is in one of this three links:

http://machinelearningmastery.com/feature-selection-machine-learning-python/

http://sebastianraschka.com/Articles/2014_python_lda.html#lda-via-scikit-learn

http://scikit-learn.org/stable/modules/feature_selection.html

.