# Concatenating multiple feature extraction methods

In many real-world examples, there are many ways to extract features from a dataset. Often it is beneficial to combine several methods to obtain good performance. This example shows how to use FeatureUnion to combine features obtained by PCA and univariate selection.

Combining features using this transformer has the benefit that it allows cross validation and grid searches over the whole process.

The combination used in this example is not particularly helpful on this dataset and is only used to illustrate the usage of FeatureUnion.

auto\_examples\_jupyter/[plot\_cv\_predict.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/plot_cv_predict.ipynb" \t "_blank)

Applications

# Outlier detection on a real data set

This example illustrates the need for robust covariance estimation on a real data set. It is useful both for outlier detection and for a better understanding of the data structure.

We selected two sets of two variables from the Boston housing data set as an illustration of what kind of analysis can be done with several outlier detection tools. For the purpose of visualization, we are working with two-dimensional examples, but one should be aware that things are not so trivial in high-dimension, as it will be pointed out.

In both examples below, the main result is that the empirical covariance estimate, as a non-robust one, is highly influenced by the heterogeneous structure of the observations. Although the robust covariance estimate is able to focus on the main mode of the data distribution, it sticks to the assumption that the data should be Gaussian distributed, yielding some biased estimation of the data structure, but yet accurate to some extent. The One-Class SVM does not assume any parametric form of the data distribution and can therefore model the complex shape of the data much better.

auto\_examples\_jupyter/applications/[plot\_outlier\_detection\_housing.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/applications/plot_outlier_detection_housing.ipynb" \t "_blank)

[Classification](http://scikit-learn.org/stable/auto_examples/index.html#classification)

# Classifier comparison

A comparison of a several classifiers in scikit-learn on synthetic datasets. The point of this example is to illustrate the nature of decision boundaries of different classifiers. This should be taken with a grain of salt, as the intuition conveyed by these examples does not necessarily carry over to real datasets.

Particularly in high-dimensional spaces, data can more easily be separated linearly and the simplicity of classifiers such as naive Bayes and linear SVMs might lead to better generalization than is achieved by other classifiers.

The plots show training points in solid colors and testing points semi-transparent. The lower right shows the classification accuracy on the test set.

auto\_examples\_jupyter/Classification/plot\_classifier\_comparison. ipynb

# Plot classification probability

Plot the classification probability for different classifiers. We use a 3 class dataset, and we classify it with a Support Vector classifier, L1 and L2 penalized logistic regression with either a One-Vs-Rest or multinomial setting, and Gaussian process classification.

The logistic regression is not a multiclass classifier out of the box. As a result it can identify only the first class.

auto\_examples\_jupyter/Classification/plot\_lda\_qda.ipynb

[Clustering](http://scikit-learn.org/stable/auto_examples/index.html#clustering)

# K-means Clustering

The plots display firstly what a K-means algorithm would yield using three clusters. It is then shown what the effect of a bad initialization is on the classification process: By setting n\_init to only 1 (default is 10), the amount of times that the algorithm will be run with different centroid seeds is reduced. The next plot displays what using eight clusters would deliver and finally the ground truth.

[plot\_cluster\_iris.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/cluster/plot_cluster_iris.ipynb" \t "_blank)

[Covariance estimation](http://scikit-learn.org/stable/auto_examples/index.html#covariance-estimation)

# Outlier detection with several methods.

When the amount of contamination is known, this example illustrates three different ways of performing [Novelty and Outlier Detection](http://scikit-learn.org/stable/modules/outlier_detection.html#outlier-detection):

* based on a robust estimator of covariance, which is assuming that the data are Gaussian distributed and performs better than the One-Class SVM in that case.
* using the One-Class SVM and its ability to capture the shape of the data set, hence performing better when the data is strongly non-Gaussian, i.e. with two well-separated clusters;
* using the Isolation Forest algorithm, which is based on random forests and hence more adapted to large-dimensional settings, even if it performs quite well in the examples below.
* using the Local Outlier Factor to measure the local deviation of a given data point with respect to its neighbors by comparing their local density.

The ground truth about inliers and outliers is given by the points colors while the orange-filled area indicates which points are reported as inliers by each method.

Here, we assume that we know the fraction of outliers in the datasets. Thus rather than using the ‘predic

plot\_outlier\_detection.ipynb

[Cross decomposition](http://scikit-learn.org/stable/auto_examples/index.html#cross-decomposition)

# Compare cross decomposition methods

Simple usage of various cross decomposition algorithms: - PLSCanonical - PLSRegression, with multivariate response, a.k.a. PLS2 - PLSRegression, with univariate response, a.k.a. PLS1 - CCA

Given 2 multivariate covarying two-dimensional datasets, X, and Y, PLS extracts the ‘directions of covariance’, i.e. the components of each datasets that explain the most shared variance between both datasets. This is apparent on the **scatterplot matrix** display: components 1 in dataset X and dataset Y are maximally correlated (points lie around the first diagonal). This is also true for components 2 in both dataset, however, the correlation across datasets for different components is weak: the point cloud is very spherical.

[plot\_compare\_cross\_decomposition.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/cross_decomposition/plot_compare_cross_decomposition.ipynb" \t "_blank)

[Dataset examples](http://scikit-learn.org/stable/auto_examples/index.html#dataset-examples)

# The Iris Dataset

This data sets consists of 3 different types of irises’ (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarray

The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.

The below plot uses the first two features. See [here](https://en.wikipedia.org/wiki/Iris_flower_data_set) for more information on this dataset.

plot\_iris\_dataset.ipynb

[Decomposition](http://scikit-learn.org/stable/auto_examples/index.html#decomposition)

# PCA example with Iris Data-set

Principal Component Analysis applied to the Iris dataset.

[plot\_pca\_iris.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/decomposition/plot_pca_iris.ipynb" \t "_blank)

# Comparison of LDA and PCA 2D projection of Iris dataset

The Iris dataset represents 3 kind of Iris flowers (Setosa, Versicolour and Virginica) with 4 attributes: sepal length, sepal width, petal length and petal width.

Principal Component Analysis (PCA) applied to this data identifies the combination of attributes (principal components, or directions in the feature space) that account for the most variance in the data. Here we plot the different samples on the 2 first principal components.

Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance between classes. In particular, LDA, in contrast to PCA, is a supervised method, using known class labels.

[plot\_pca\_vs\_lda.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/decomposition/plot_pca_vs_lda.ipynb" \t "_blank)

[Ensemble methods](http://scikit-learn.org/stable/auto_examples/index.html#ensemble-methods)

# Plot the decision surfaces of ensembles of trees on the iris dataset

Plot the decision surfaces of forests of randomized trees trained on pairs of features of the iris dataset.

plot\_forest\_iris.ipynb

[Tutorial exercises](http://scikit-learn.org/stable/auto_examples/index.html#tutorial-exercises)

# SVM Exercise

A tutorial exercise for using different SVM kernels.

This exercise is used in the [Using kernels](http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html#using-kernels-tut) part of the [Supervised learning: predicting an output variable from high-dimensional observations](http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html#supervised-learning-tut) section of the [A tutorial on statistical-learning for scientific data processing](http://scikit-learn.org/stable/tutorial/statistical_inference/index.html#stat-learn-tut-index).

[plot\_iris\_exercise.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/exercises/plot_iris_exercise.ipynb" \t "_blank)

# Cross-validation on Digits Dataset Exercise

A tutorial exercise using Cross-validation with an SVM on the Digits dataset.

[plot\_cv\_digits.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/exercises/plot_cv_digits.ipynb" \t "_blank)

[Feature Selection](http://scikit-learn.org/stable/auto_examples/index.html#feature-selection)

# Feature selection using SelectFromModel and LassoCV

Use SelectFromModel meta-transformer along with Lasso to select the best couple of features from the Boston dataset.

[plot\_select\_from\_model\_boston.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/feature_selection/plot_select_from_model_boston.ipynb" \t "_blank)

# Recursive feature elimination with cross-validation

A recursive feature elimination example with automatic tuning of the number of features selected with cross-validation.

[plot\_rfe\_with\_cross\_validation.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/feature_selection/plot_rfe_with_cross_validation.ipynb" \t "_blank)

# Univariate Feature Selection

An example showing univariate feature selection.

Noisy (non informative) features are added to the iris data and univariate feature selection is applied. For each feature, we plot the p-values for the univariate feature selection and the corresponding weights of an SVM. We can see that univariate feature selection selects the informative features and that these have larger SVM weights.

In the total set of features, only the 4 first ones are significant. We can see that they have the highest score with univariate feature selection. The SVM assigns a large weight to one of these features, but also Selects many of the non-informative features. Applying univariate feature selection before the SVM increases the SVM weight attributed to the significant features, and will thus improve classification

[plot\_feature\_selection.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/feature_selection/plot_feature_selection.ipynb" \t "_blank)

[Gaussian Process for Machine Learning](http://scikit-learn.org/stable/auto_examples/index.html#gaussian-process-for-machine-learning)

# Gaussian process classification (GPC) on iris dataset

This example illustrates the predicted probability of GPC for an isotropic and anisotropic RBF kernel on a two-dimensional version for the iris-dataset. The anisotropic RBF kernel obtains slightly higher log-marginal-likelihood by assigning different length-scales to the two feature dimensions.

plot\_gpc\_iris.ipynb

[Generalized Linear Models](http://scikit-learn.org/stable/auto_examples/index.html#generalized-linear-models)

# Plot multi-class SGD on the iris dataset

Plot decision surface of multi-class SGD on iris dataset. The hyperplanes corresponding to the three one-versus-all (OVA) classifiers are represented by the dashed lines.

[plot\_sgd\_iris.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/linear_model/plot_sgd_iris.ipynb" \t "_blank)

# Logistic Regression 3-class Classifier

Show below is a logistic-regression classifiers decision boundaries on the iris <https://en.wikipedia.org/wiki/Iris\_flower\_data\_set>\_ dataset. The datapoints are colored according to their labels.

plot\_iris\_logistic.ipynb

[Model Selection](http://scikit-learn.org/stable/auto_examples/index.html#model-selection)

# Underfitting vs. Overfitting

This example demonstrates the problems of underfitting and overfitting and how we can use linear regression with polynomial features to approximate nonlinear functions. The plot shows the function that we want to approximate, which is a part of the cosine function. In addition, the samples from the real function and the approximations of different models are displayed. The models have polynomial features of different degrees. We can see that a linear function (polynomial with degree 1) is not sufficient to fit the training samples. This is called **underfitting**. A polynomial of degree 4 approximates the true function almost perfectly. However, for higher degrees the model will **overfit** the training data, i.e. it learns the noise of the training data. We evaluate quantitatively **overfitting** / **underfitting** by using cross-validation. We calculate the mean squared error (MSE) on the validation set, the higher, the less likely the model generalizes correctly from the training data.

plot\_underfitting\_overfitting.ipynb

[Nearest Neighbors](http://scikit-learn.org/stable/auto_examples/index.html#nearest-neighbors)

# Nearest Neighbors Classification

Sample usage of Nearest Neighbors classification. It will plot the decision boundaries for each class.

plot\_classification.ipynb

[Neural Networks](http://scikit-learn.org/stable/auto_examples/index.html#neural-networks)

# Varying regularization in Multi-layer Perceptron

A comparison of different values for regularization parameter ‘alpha’ on synthetic datasets. The plot shows that different alphas yield different decision functions.

Alpha is a parameter for regularization term, aka penalty term, that combats overfitting by constraining the size of the weights. Increasing alpha may fix high variance (a sign of overfitting) by encouraging smaller weights, resulting in a decision boundary plot that appears with lesser curvatures. Similarly, decreasing alpha may fix high bias (a sign of underfitting) by encouraging larger weights, potentially resulting in a more complicated decision boundary.

plot\_mlp\_alpha.ipynb

[Preprocessing](http://scikit-learn.org/stable/auto_examples/index.html#preprocessing)

# Importance of Feature Scaling

Feature scaling though standardization (or Z-score normalization) can be an important preprocessing step for many machine learning algorithms. Standardization involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one.

While many algorithms (such as SVM, K-nearest neighbors, and logistic regression) require features to be normalized, intuitively we can think of Principle Component Analysis (PCA) as being a prime example of when normalization is important. In PCA we are interested in the components that maximize the variance. If one component (e.g. human height) varies less than another (e.g. weight) because of their respective scales (meters vs. kilos), PCA might determine that the direction of maximal variance more closely corresponds with the ‘weight’ axis, if those features are not scaled. As a change in height of one meter can be considered much more important than the change in weight of one kilogram, this is clearly incorrect.

To illustrate this, PCA is performed comparing the use of data with [StandardScaler](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler) applied, to unscaled data. The results are visualized and a clear difference noted. The 1st principal component in the unscaled set can be seen. It can be seen that feature #13 dominates the direction, being a whole two orders of magnitude above the other features. This is contrasted when observing the principal component for the scaled version of the data. In the scaled version, the orders of magnitude are roughly the same across all the features.

The dataset used is the Wine Dataset available at UCI. This dataset has continuous features that are heterogeneous in scale due to differing properties that they measure (i.e alcohol content, and malic acid).

The transformed data is then used to train a naive Bayes classifier, and a clear difference in prediction accuracies is observed wherein the dataset which is scaled before PCA vastly outperforms the unscaled version.

[plot\_scaling\_importance.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/preprocessing/plot_scaling_importance.ipynb" \t "_blank)

# Compare the effect of different scalers on data with outliers

Feature 0 (median income in a block) and feature 5 (number of households) of the [California housing dataset](http://www.dcc.fc.up.pt/%7Eltorgo/Regression/cal_housing.html) have very different scales and contain some very large outliers. These two characteristics lead to difficulties to visualize the data and, more importantly, they can degrade the predictive performance of many machine learning algorithms. Unscaled data can also slow down or even prevent the convergence of many gradient-based estimators.

Indeed many estimators are designed with the assumption that each feature takes values close to zero or more importantly that all features vary on comparable scales. In particular, metric-based and gradient-based estimators often assume approximately standardized data (centered features with unit variances). A notable exception are decision tree-based estimators that are robust to arbitrary scaling of the data.

This example uses different scalers, transformers, and normalizers to bring the data within a pre-defined range.

Scalers are linear (or more precisely affine) transformers and differ from each other in the way to estimate the parameters used to shift and scale each feature.

QuantileTransformer provides a non-linear transformation in which distances between marginal outliers and inliers are shrunk.

Unlike the previous transformations, normalization refers to a per sample transformation instead of a per feature transformation.

The following code is a bit verbose, feel free to jump directly to the analysis of the [results](http://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#results).

plot\_all\_scaling.ipynb

[Semi Supervised Classification](http://scikit-learn.org/stable/auto_examples/index.html#semi-supervised-classification)

# Decision boundary of label propagation versus SVM on the Iris dataset

Comparison for decision boundary generated on iris dataset between Label Propagation and SVM.

This demonstrates Label Propagation learning a good boundary even with a small amount of labeled data.

plot\_label\_propagation\_versus\_svm\_iris.ipynb

[Support Vector Machines](http://scikit-learn.org/stable/auto_examples/index.html#support-vector-machines)

# Plot different SVM classifiers in the iris dataset

Comparison of different linear SVM classifiers on a 2D projection of the iris dataset. We only consider the first 2 features of this dataset:

* Sepal length
* Sepal width

This example shows how to plot the decision surface for four SVM classifiers with different kernels.

The linear models LinearSVC() and SVC(kernel='linear') yield slightly different decision boundaries. This can be a consequence of the following differences:

* LinearSVC minimizes the squared hinge loss while SVC minimizes the regular hinge loss.
* LinearSVC uses the One-vs-All (also known as One-vs-Rest) multiclass reduction while SVC uses the One-vs-One multiclass reduction.

Both linear models have linear decision boundaries (intersecting hyperplanes) while the non-linear kernel models (polynomial or Gaussian RBF) have more flexible non-linear decision boundaries with shapes that depend on the kind of kernel and its parameters.

[plot\_iris.ipynb](http://localhost:8888/notebooks/Desktop/EMSE6992_DataAnalytics/EMSE6992_LabsandAssignments/auto_examples_jupyter/svm/plot_iris.ipynb" \t "_blank)

[Decision Trees](http://scikit-learn.org/stable/auto_examples/index.html#decision-trees)

# Plot the decision surface of a decision tree on the iris dataset

Plot the decision surface of a decision tree trained on pairs of features of the iris dataset.

See [decision tree](http://scikit-learn.org/stable/modules/tree.html#tree) for more information on the estimator.

For each pair of iris features, the decision tree learns decision boundaries made of combinations of simple thresholding rules inferred from the training samples.

plot\_iris.ipynb