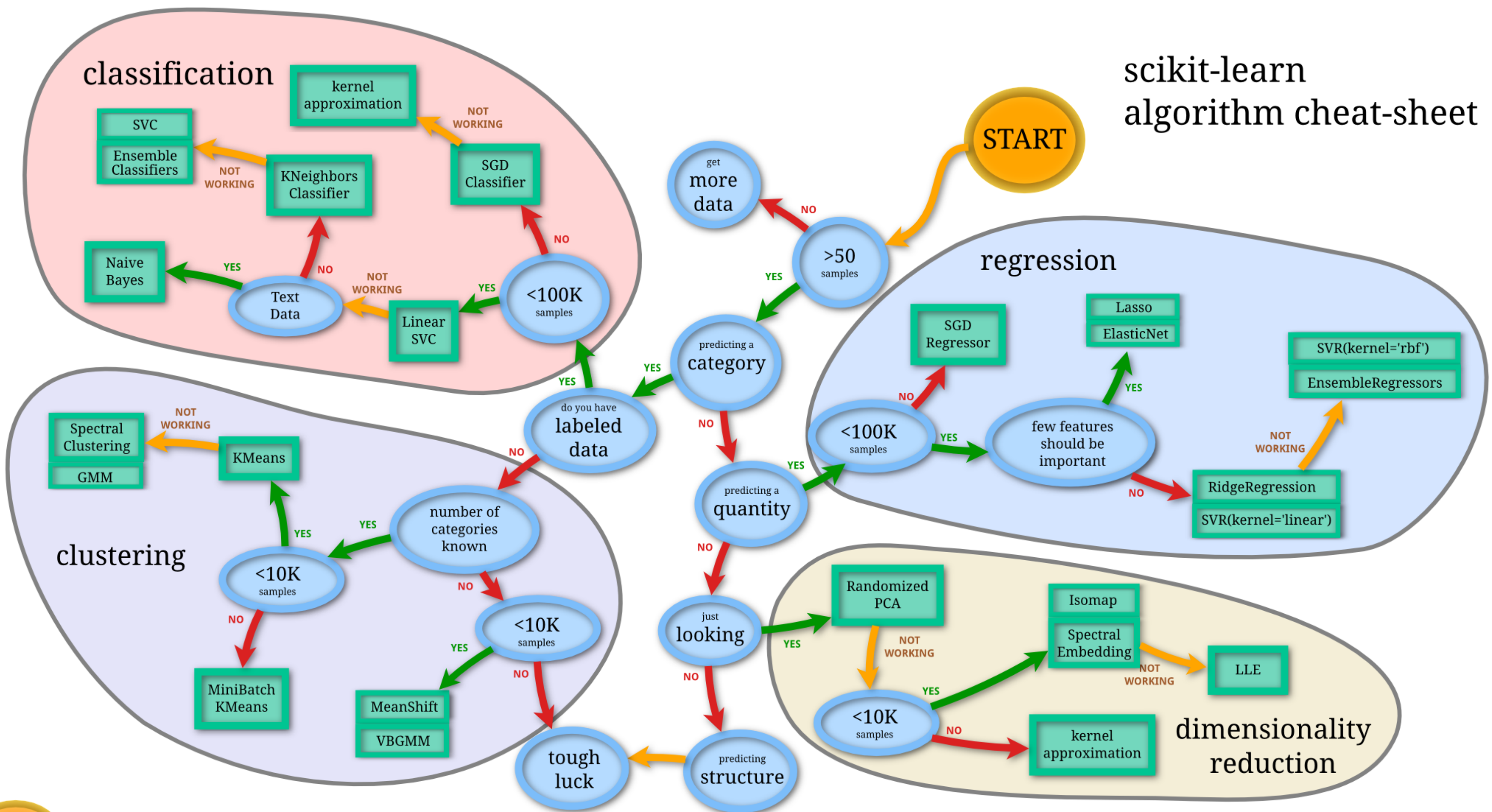
Scikit

Aplying Machine Learning

David Arroyo Menéndez

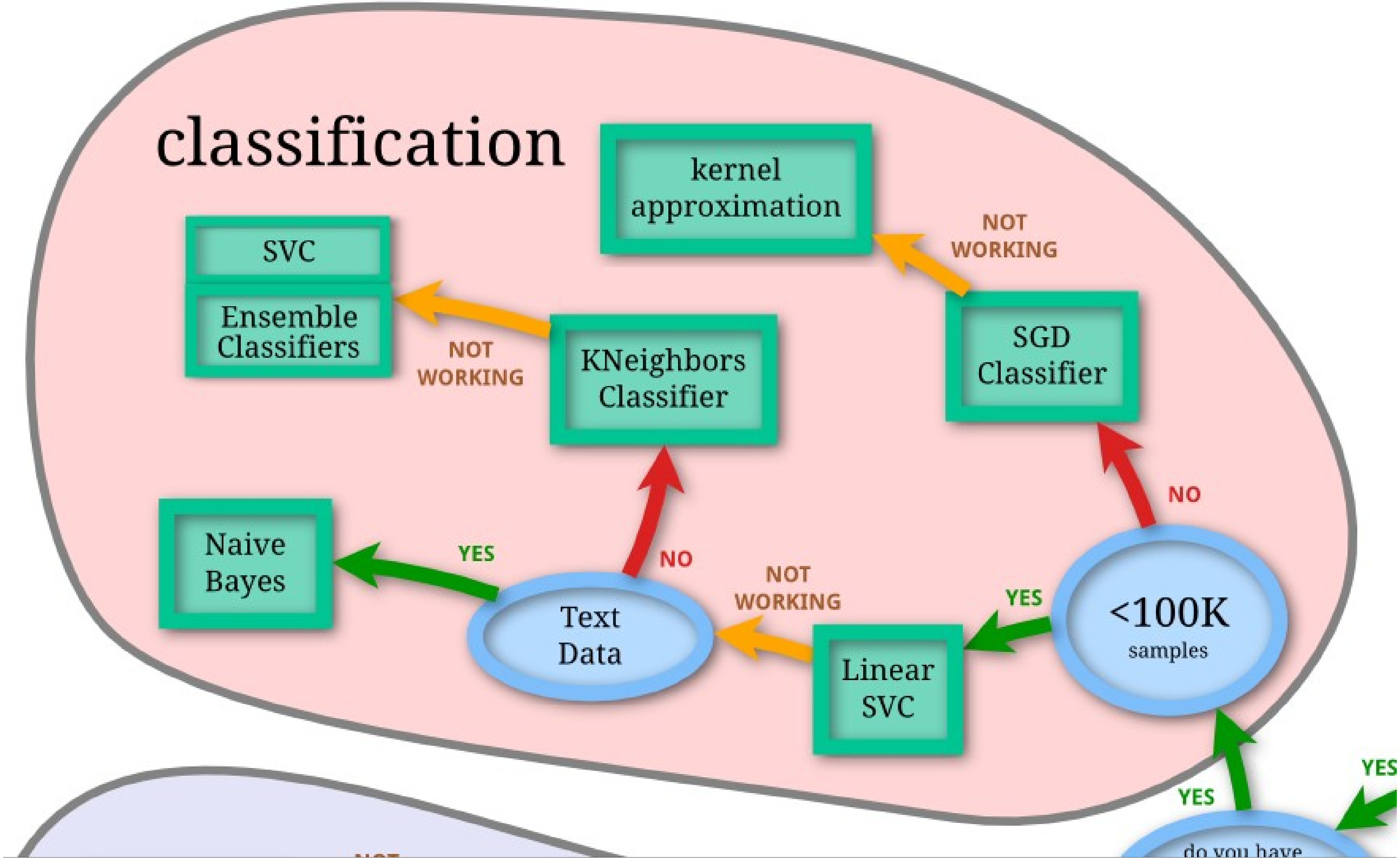
Scikit



scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC
Ensemble Classifiers
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering
GMM
MiniBatch KMeans
MeanShift
VBGMM
regression
SGD Regressor
few features should be important
Lasso
ElasticNet
RidgeRegression
SVR(kernel='linear')
SVR(kernel='rbf')
EnsembleRegressors
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING

Back

Classification



Classification: SGD

We want predict a category and we’ve labeled data with <100k samples

```
$ python3 plot_sgd_iris.py
$ python3 sgd.py
```

Classification: Kernel approximation

We want predict a category and we've labeled data with <100k samples and SGD is not working

$ python3 kernel-approximation.py

Classification: Naive Bayes

What we want to know; the Posterior probability of Class j given a predictor x

The Likelihood; the probability of the predictor given a Class j. Its computed from the training-set.

The Prior probability of Class j; what we know about the class distribution before we consider x.

$$P(Class_j \mid x) = \frac{P(x \mid Class_j) \times P(Class_j)}{P(x)}$$

The Evidence. In practice, there's interest only in the numerator (denominator is effectively constant)

Applying the independence assumption

$$P(x \mid Class_j) = P(x_1 \mid Class_j) \times P(x_2 \mid Class_j) \times \ldots \times P(x_k \mid Class_j)$$

Substituting the independence assumption, we derive the Posterior probability of Class j given a new instance x' as...

$$P(Class_j \mid x') = P(x'_1 \mid Class_j) \times P(x'_2 \mid Class_j) \times \ldots \times P(x'_k \mid Class_j) \times P(Class_j)$$

Classification: Naive Bayes

	Predictors				Response
	Outlook	**Temperature**	**Humidity**	**Wind**	**Class** **Play=Yes** **Play=No**
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Classification: Naive Bayes

P(Outlook=o|Class $_{Play=Yes\|No}$)	Frequency		Probability in Class	
Outlook =	Play=Yes	Play=No	Play=Yes	Play=No
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	total= 9	total=5		

P(Temperature=t|Class $_{Play=Yes\|No}$)	Frequency		Probability in Class	
Temperature =	Play=Yes	Play=No	Play=Yes	Play=No
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
	total= 9	total=5		

P(Humidity=h|Class $_{Play=Yes\|No}$)	Frequency		Probability in Class	
Humidity =	Play=Yes	Play=No	Play=Yes	Play=No
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
	total= 9	total=5		

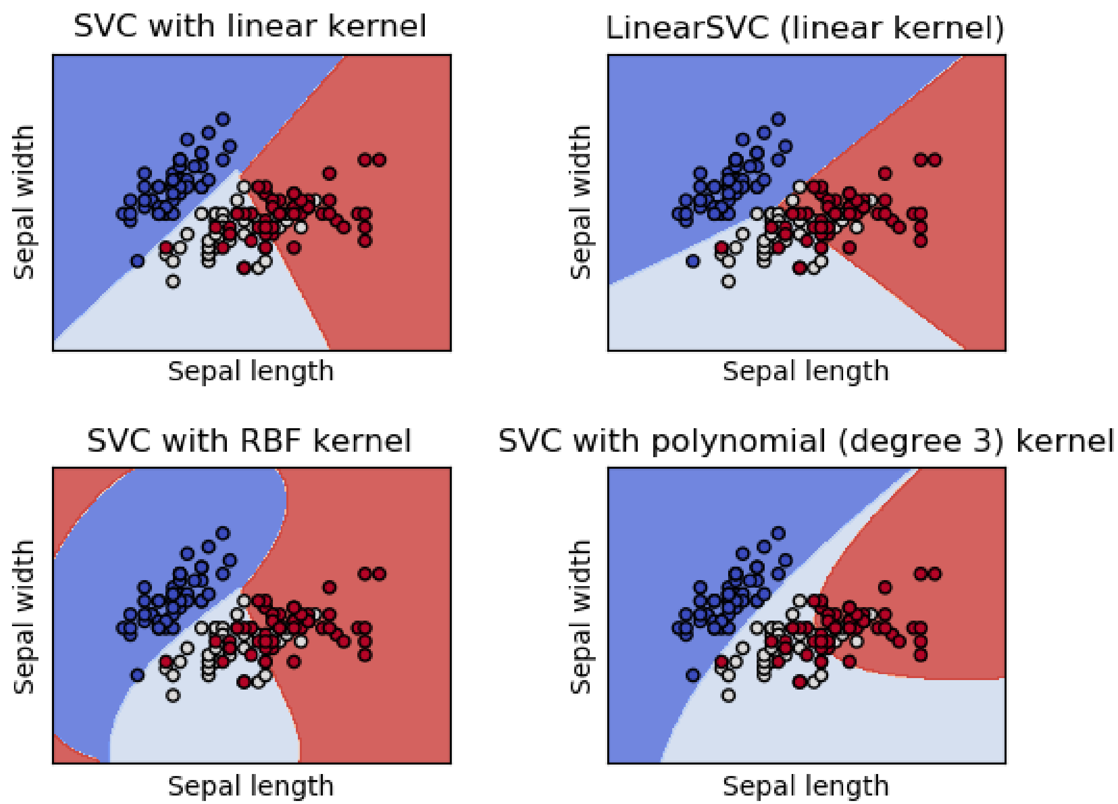
P(Wind=w|Class $_{Play=Yes\|No}$)	Frequency		Probability in Class	
Wind =	Play=Yes	Play=No	Play=Yes	Play=No
strong	3	3	3/9	3/5
weak	6	2	6/9	2/5
	total= 9	total=5		

Classification: Naive Bayes

```
$ python3 gaussiannb.py
$ python3 bernoullinb.py
$ python3 multinomialnb.py
$ python3 nltk/sexmachine.py
```

Classification: SVC

We want predict a category and we've labeled data with <100k samples



Classification: SVC
What is a kernel?

1.4.6. Kernel functions

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$. d is specified by keyword `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$. γ is specified by keyword `gamma`, must be greater than 0.
- sigmoid ($\tanh(\gamma \langle x, x' \rangle + r)$), where r is specified by `coef0`.

Different kernels are specified by keyword kernel at initialization:

```
>>> linear_svc = svm.SVC(kernel='linear')
>>> linear_svc.kernel
'linear'
>>> rbf_svc = svm.SVC(kernel='rbf')
>>> rbf_svc.kernel
'rbf'
```

Classification: SVC

We want predict a category and we’ve labeled data with <100k samples

$ python3 svc-example.py

Classification: Trees

ID3: The algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalise to unseen data.

C4.5: is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals.

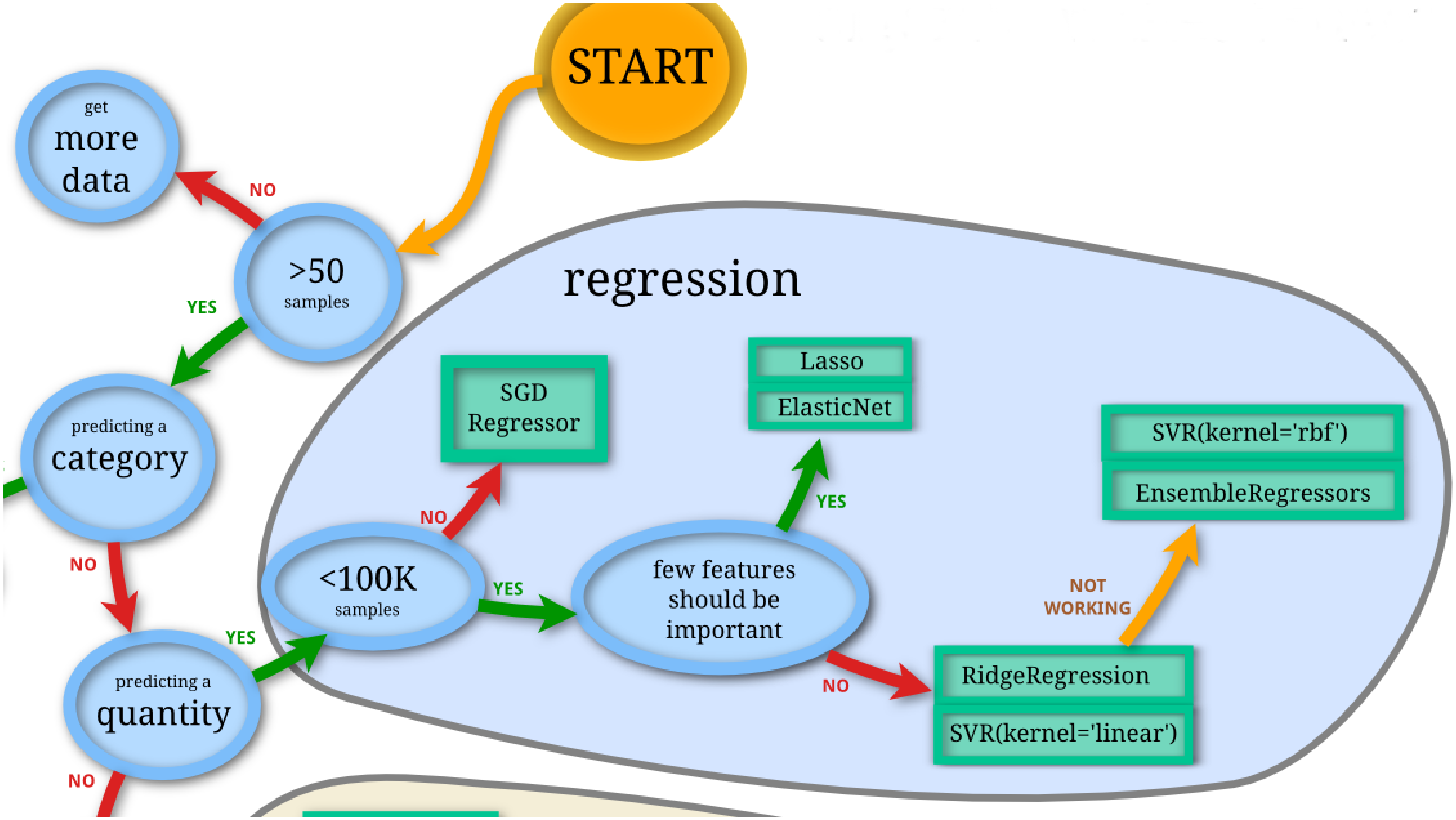
CART: is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets.

Classification: Trees (II)

```
$ python3 plot_iris1.py
$ python3 plot_unveil_tree_structure.py
$ python3 plot_classifier_comparison.py
```

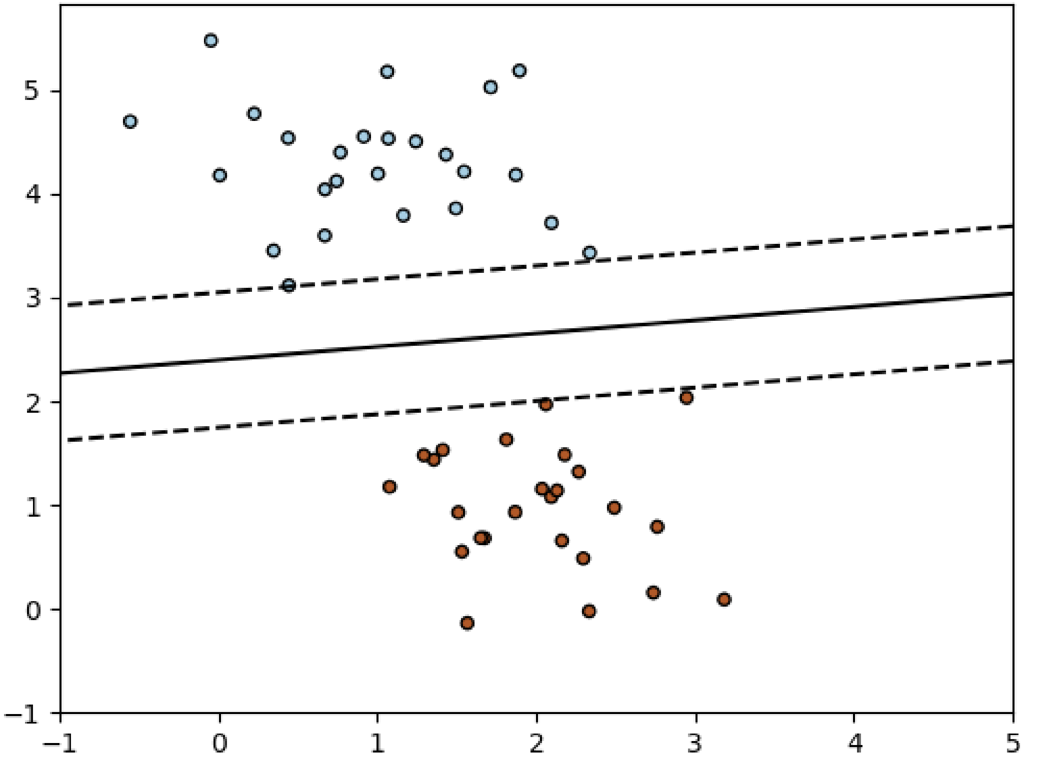
http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Regression


START
get more data
NO
>50 samples
YES
predicting a category
NO
predicting a quantity
YES
NO
regression
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors

SGD Regressor

The class SGDClassifier implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification.



SGD Regressor

As other classifiers, SGD has to be fitted with two arrays: an array X of size [n_samples, n_features] holding the training samples, and an array Y of size [n_samples] holding the target values (class labels) for the training samples:

```
$ python3 plot_sgd_iris.py
$ python3 sgd.py
```

Lasso

Linear Model trained with L1 prior as regularizer (aka the Lasso)
The optimization objective for Lasso is:
(1 / (2 * n_samples)) * ||y - Xw||^2_2 + alpha * ||w||_1

Technically the Lasso model is optimizing the same objective function as the Elastic Net with l1_ratio=1.0 (no L2 penalty).

```
$ python3 plot_lasso_and_elasticnet.py
$ python3 plot_multi_task_lasso_support.py
```

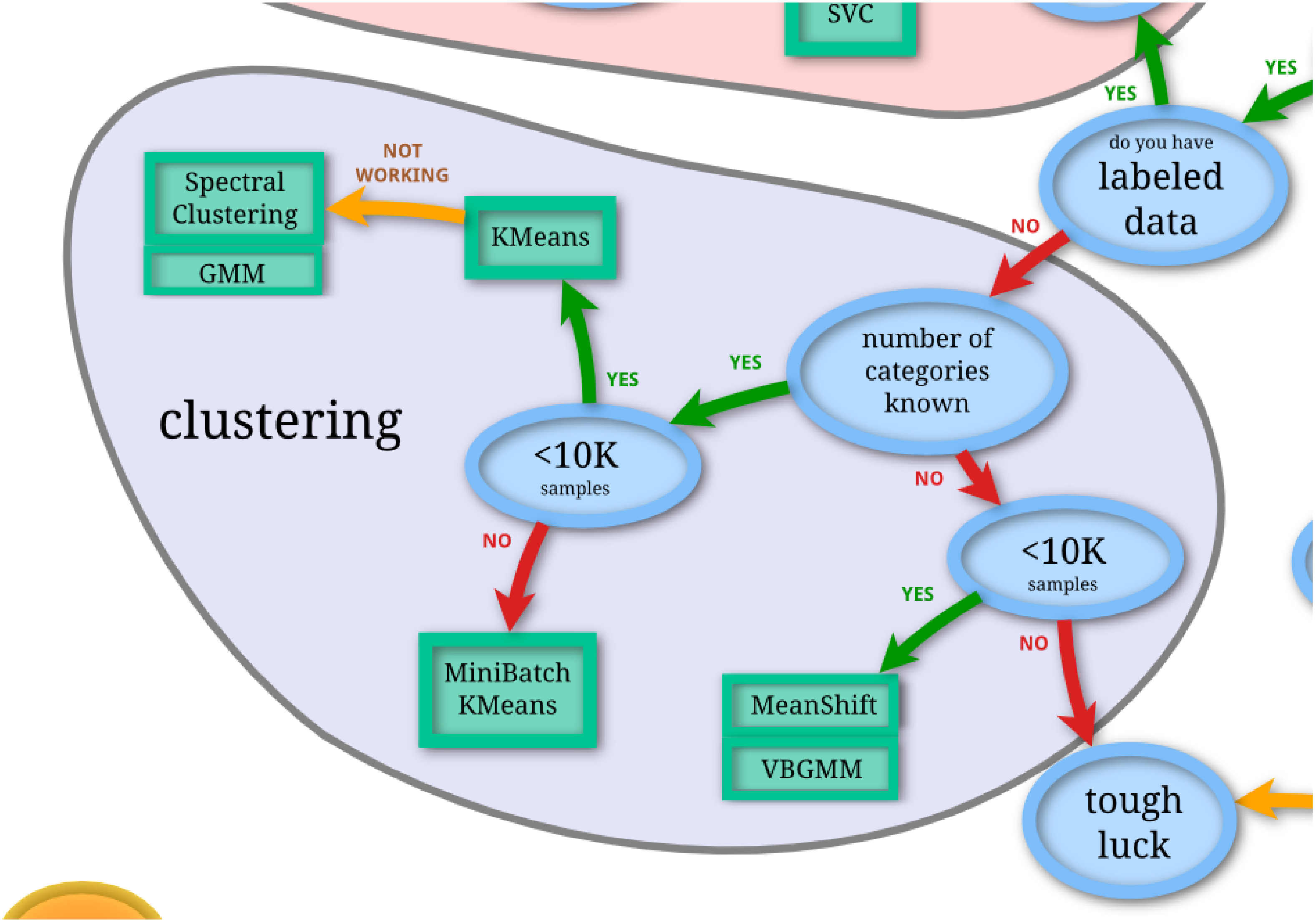
Kernel Ridge

```
$ python3 plot_kernel_ridge_regression.py
```

SVR

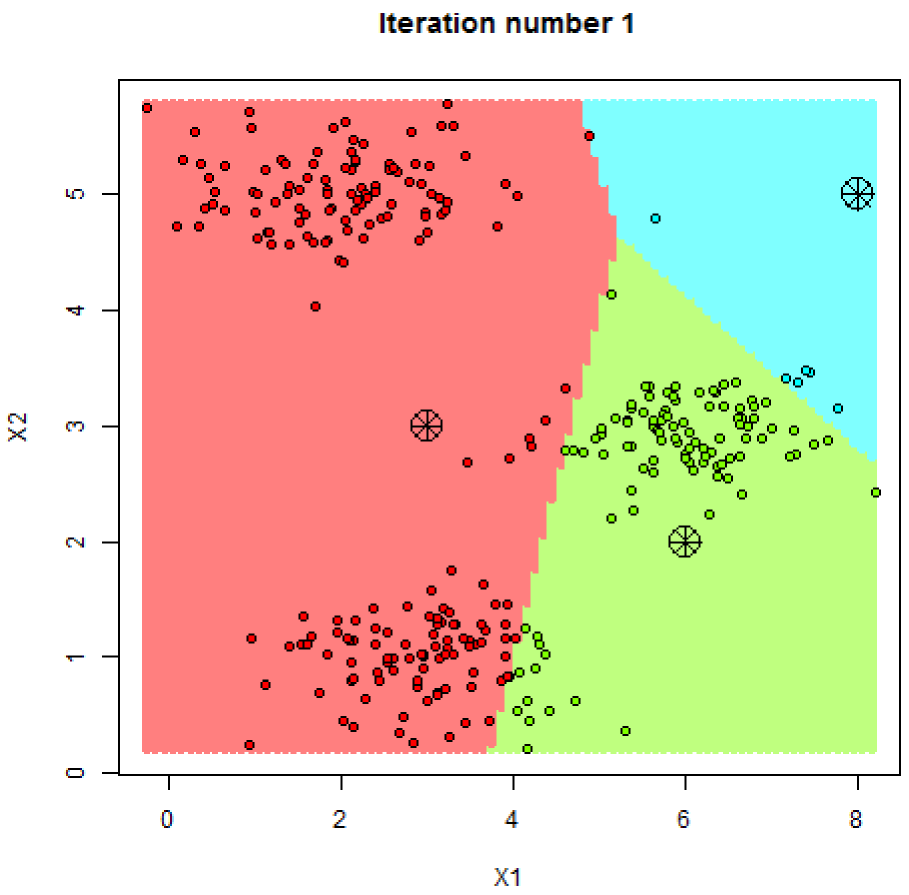
```
$ python3 svr-example.py
```

Clustering



SVC
YES
YES
do you have
labeled
data
NO
NOT
WORKING
Spectral
Clustering
GMM
KMeans
number of
categories
known
YES
YES
clustering
<10K
samples
NO
<10K
samples
NO
YES
NO
MiniBatch
KMeans
MeanShift
VBGMM
tough
luck

Kmeans



Kmeans

Subdivide the space making regions from reference points called centroides

$ python3 plot_kmeans_assumptions.py
$ python3 plot_cluster_iris.py

GMM

A typical finite-dimensional mixture model is a hierarchical model consisting of the following components:

- N **random variables that are observed**, each distributed according to a mixture of K components, with the components belonging to the same parametric family of distributions (e.g., all normal, all Zipfian, etc.) but with different parameters
- N **random latent variables** specifying the identity of the mixture component of each observation, each distributed according to a K-dimensional categorical distribution
- A set of K **mixture weights**, which are probabilities that sum to 1.
- A set of K **parameters**, each specifying the parameter of the corresponding mixture component. In many cases, each "parameter" is actually a set of parameters. For example, if the mixture components are Gaussian distributions, there will be a mean and variance for each component. If the mixture components are categorical distributions (e.g., when each observation is a token from a finite alphabet of size V), there will be a vector of V probabilities summing to 1.

$ python3 plot_gmm_covariances.py

Spectral Clustering

Make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions

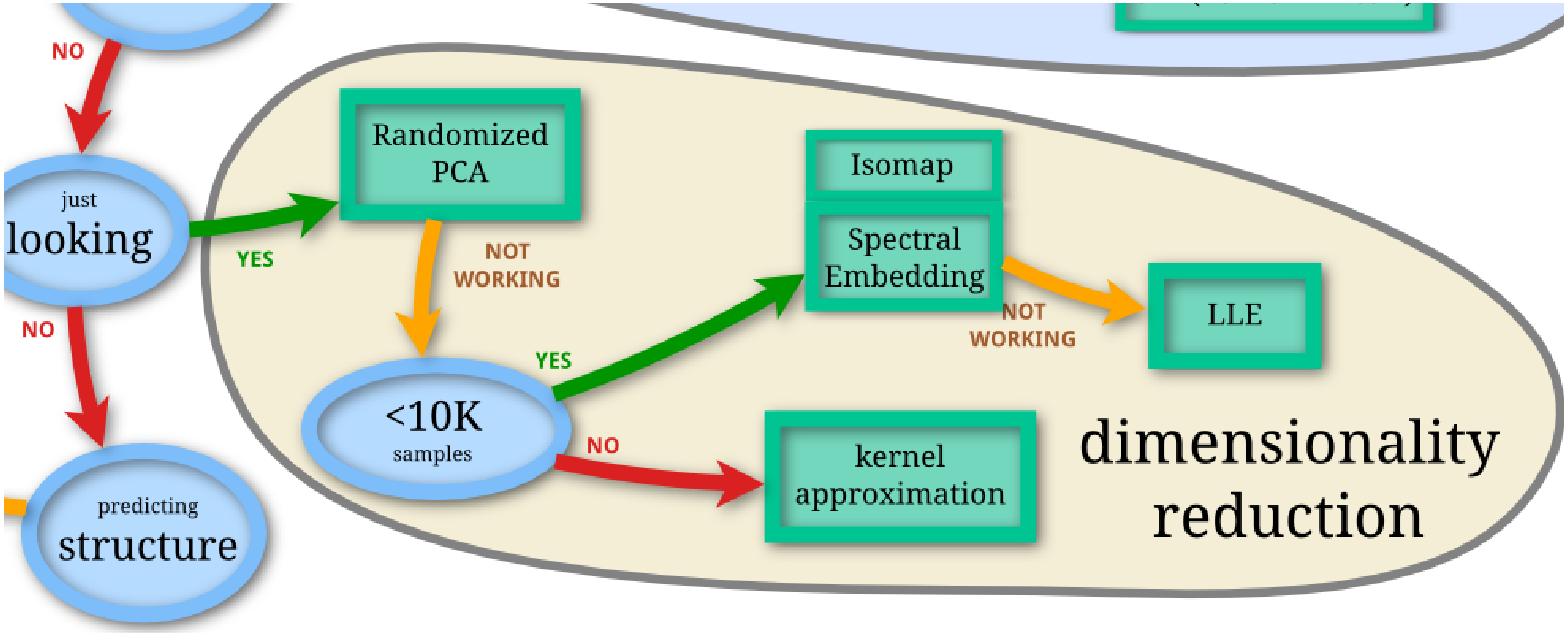
```
$ python3 plot_cluster_comparison.py
```

Mean Shift

Mean shift is a non-parametric feature-space analysis technique for locating the maxima of a density function, a so-called mode-seeking algorithm.

```
$ python3 plot_mean_shift.py
```

Dimensionality Reduction



The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets

Randomized PCA

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It's often used to visualize genetic distance and relatedness between populations. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering

```
$ python3 pca-example.py
$ python3 scikit-plot-pca.py
$ python3 plot_pca_iris.py
$ python3 incremental-pca.py
```

Kernel Approximation

The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets.

For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into feature vector representations via a user-specified feature map: in contrast, kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation.

```
$ python3 kernel-approximation.py
```

LLE and Spectral Embedding

```
$ python3 plot_lle_digits.py
$ python3 plot_spectral_grid.py
```

Spectral embedding for non-linear dimensionality reduction.
Forms an affinity matrix given by the specified function and applies spectral decomposition to the corresponding graph laplacian. The resulting transformation is given by the value of the eigenvectors for each data point.

Isomap

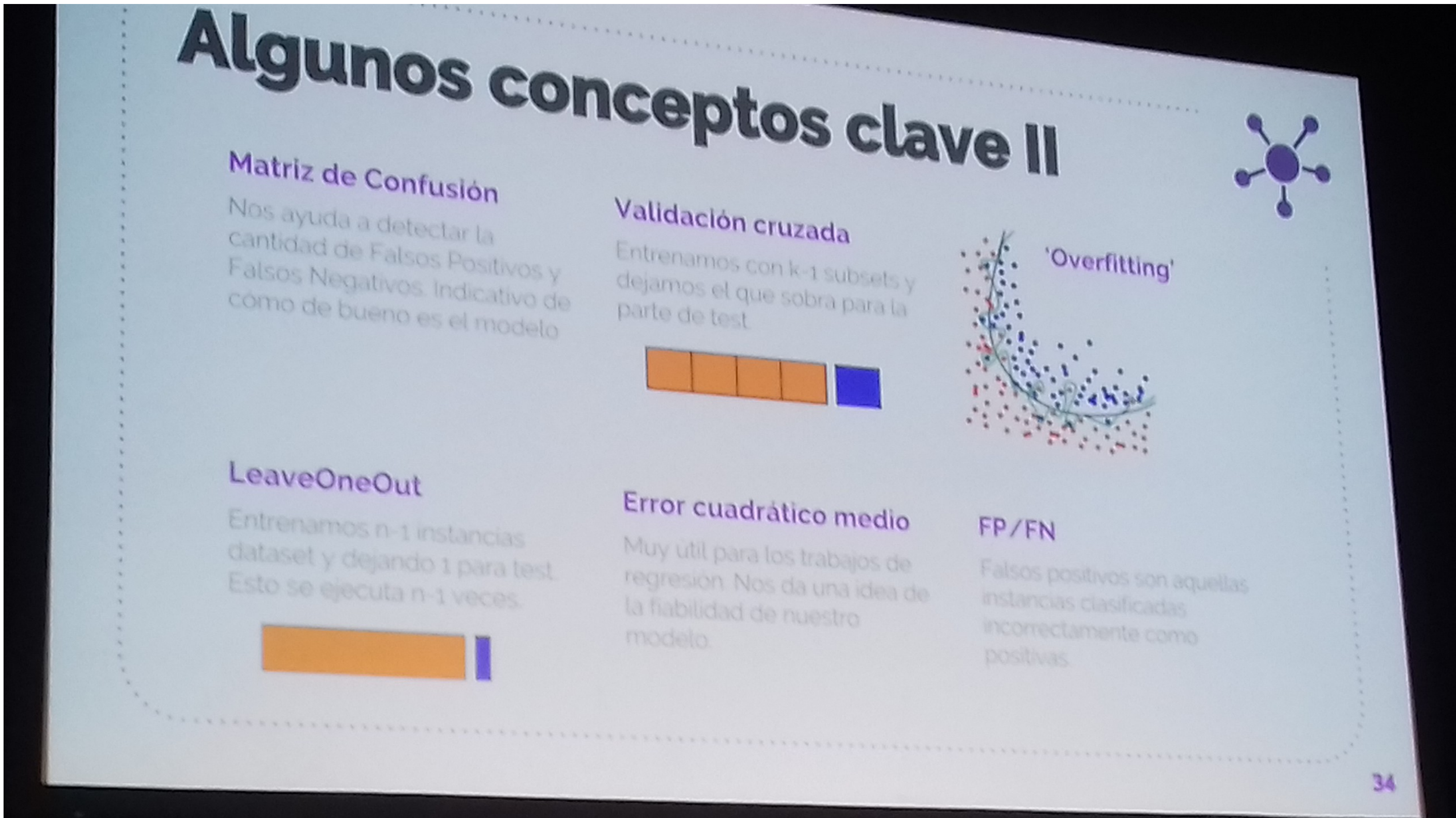
Isomap Embedding
Non-linear dimensionality reduction through Isometric Mapping

```
$ python3 plot_compare_methods.py
```

Scikit Image

```
$ python3 plot_marching_cubes.py
```

Conceptos Clave



Matriz de Confusión

```
$ python3 plot_confusion_matrix.py
```

LeaveOneOut

```
$ python3 leaveoneout.py
$ python3 leavepout.py
```

Validación Cruzada

```
$ python3 crossvalidation.py
```

Overfitting

```
$ python3 nooverfitting.py
```

Outliers

Son valores atípicos que se salen del doble de la desviación típica, por ejemplo.

```
$ python3 numpy/reject-outliers.py
$ python3 scikit/plot_outlier_detection_housing.pyo
```