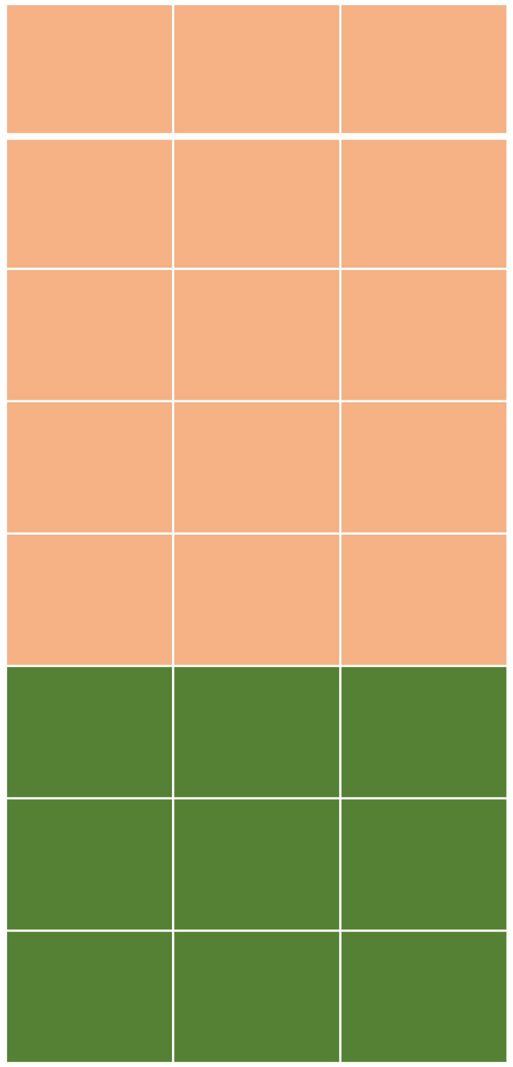
Scikitlearn in Machine Learning

Python

Training and Test Data

- Usually when creating a ML model, the total data is divided into two categories: Training data and Test data.
- The training data is used to create a model. (orange)
- The test data is used to measure the goodness of the model. (green)
- The target values from training set is used for training while the target values of test data is compared to the target values returned by the trained model to measure how good the train was trained.



How to calculate the errors (in-error and out-error)

- In-error is usually referred to the the amount of mismatched classified data in the training set and out-error is for test set.
- In the case of regression the error is the difference between the actual target value and the target value that the model returns.

- tree_clf.fit(trainX, TRAINy)
- print(tree_clf.score(trainX, TRAINy))

- tree_clf.fit(testX, TESTy)
- print(tree_clf.score(testX, TESTy))

Datasets

- Load and fetch popular reference datasets (e.g. Iris)

```
# load a default dataset
from sklearn import datasets
iris = datasets.load_iris()
```

http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html

- Artificial data generators (e.g. binary classification)

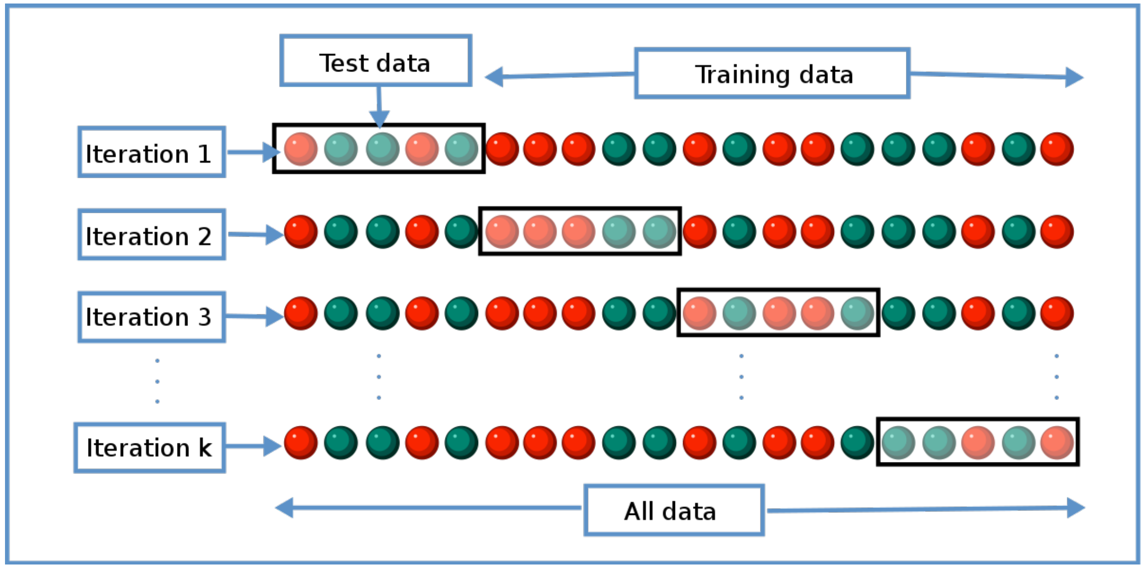
```
dataset = datasets.make_classification(n_samples=1000, n_features=10,
            n_informative=2, n_redundant=2, n_repeated=0, n_classes=2)
```

http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

- Inspect the data structures

```
# dataset description
print iris.DESCR
# data examples (features)
print iris.data
# data target labels (classes)
print iris.target
```

Cross validation



k-fold cross-validation

- Split the dataset D in k equally sized disjoint subsets D_i
- For $i \in [1, k]$
    - Train the predictor on $T_i = D \setminus D_i$
    - Compute the score of the predictor on the test set D_i
- Return the average score across the folds

Cross validation

- We can use Scikit-learn for K-fold cross-validation:

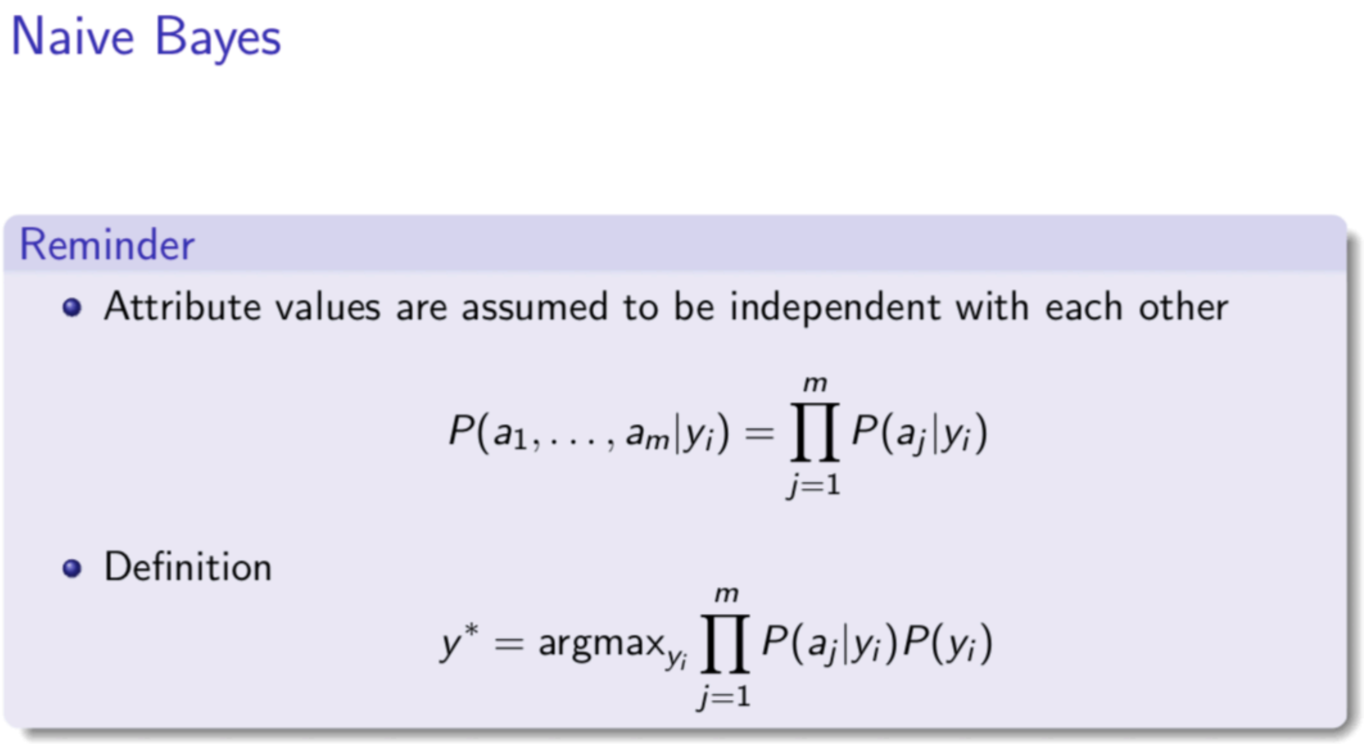
```
from sklearn import cross_validation
kf = cross_validation.KFold(len(iris.data), n_folds=5, shuffle=True)
for train_index, test_index in kf:
    X_train, y_train = iris.data[train_index], iris.target[train_index]
    X_test, y_test   = iris.data[test_index],  iris.target[test_index]
```

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

- Inspect the structures:

```
print X_train
print y_train
print X_test
print y_test
```

Naïve Bayes



Naive Bayes

Reminder

- Attribute values are assumed to be independent with each other

$$P(a_1, \ldots, a_m | y_i) = \prod_{j=1}^{m} P(a_j | y_i)$$

- Definition

$$y^* = \operatorname{argmax}_{y_i} \prod_{j=1}^{m} P(a_j | y_i) P(y_i)$$

Naïve Bayes

- Scikit-learn implements several naive Bayes algorithms
- e.g. Gaussian naive Bayes

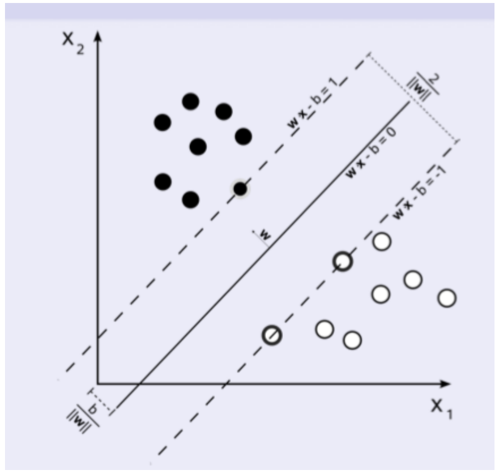
```
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
```

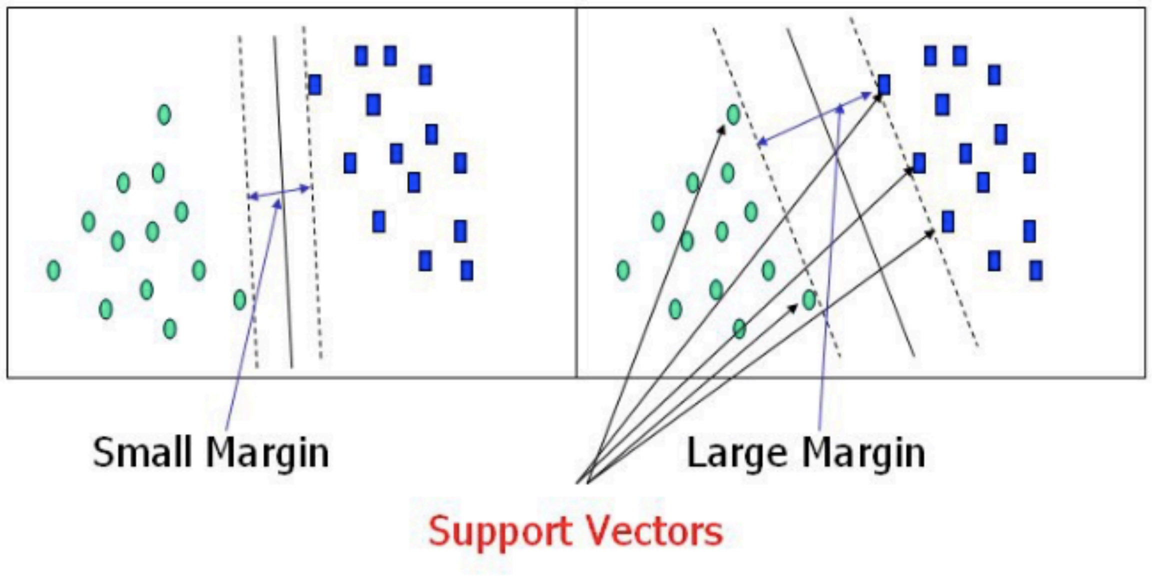
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

- Inspect the data structures:

```
print pred
print y_test
```

Support Vector Machine (SVM)





SVM

- Scikit-learn also includes Support Vector Machine algorithms
- e.g. Support-C Vector Classification

```
#SVM
from sklearn.svm import SVC
clf = SVC(C=1e-01, kernel='rbf', gamma=0.1)
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
```

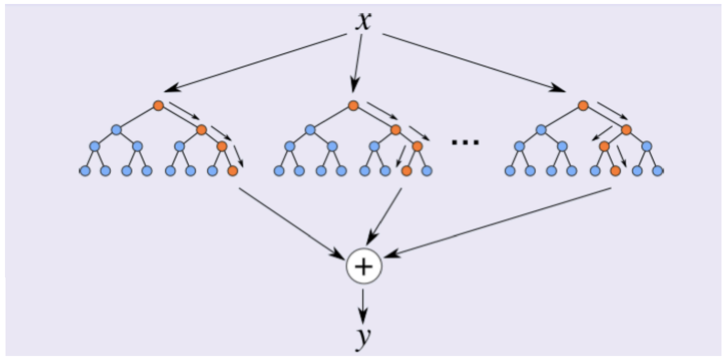
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

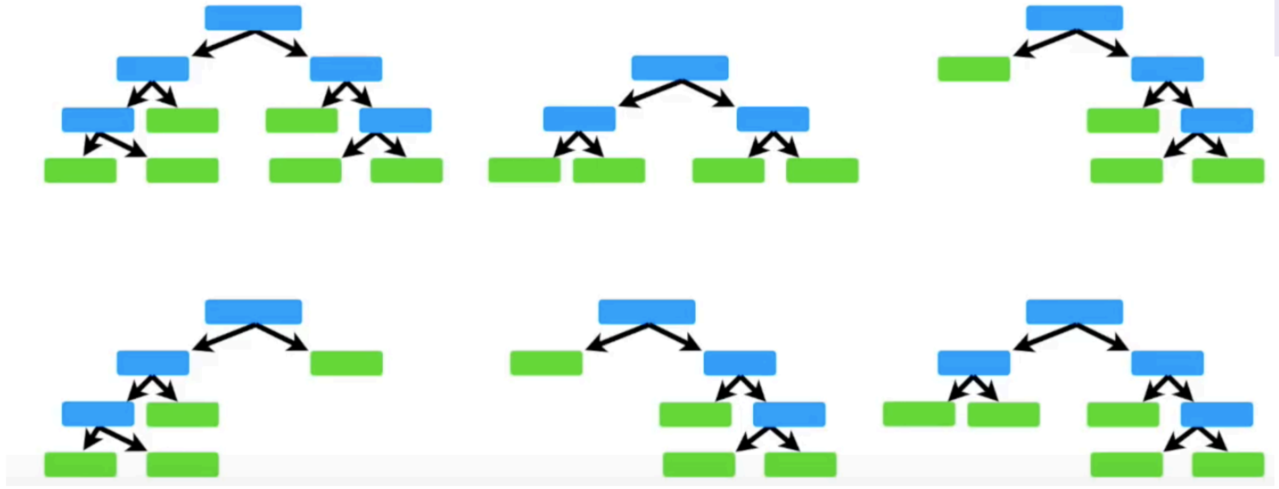
- Inspect the data structures:

```
print pred
print y_test
```

Random Forest

1. The idea comes from decision tree
2. Create a number of bootstraps i.e., a subset of data set
3. For each bootstrap create multiple decision trees by randomly selecting a number of attributes
4. Apply the new entry on all the trees and see which target value is returned more.





Random Forest

- Scikit-learn includes ensemble-based methods for classification
- e.g. Random Forest Classifier

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=6)
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
```

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- Inspect the data structures:

```
print pred
print y_test
```