Python SciKit Learn Training

Trainer: Amir Othman

Website:www.tertiarycourses.com.my
Email:malaysiacourses@tertiaryinfotech.com

About me

- Software Engineer @ Instance http://www.instance.com.sg
- Education:
  - Bauhaus Universität Weimar - Masters
  - Hochschule Ulm – Bachelors
- Focus/Expertise:
  - Machine Learning
  - Software Engineering
  - Data Visualization
- Works:
  - www.kronologimalaysia.com
  - www.diezeitachse.de

Agenda

Module 1 – Introduction

Module 2 – Datasets

Module 3 – Supervised Learning

Module 4 – Unsupervised Learning

Module 5 – Model Selection

Materials

Download materials at

https://github.com/amirothman/scikit-learn-course

Module 1
Introduction

What is Machine Learning?

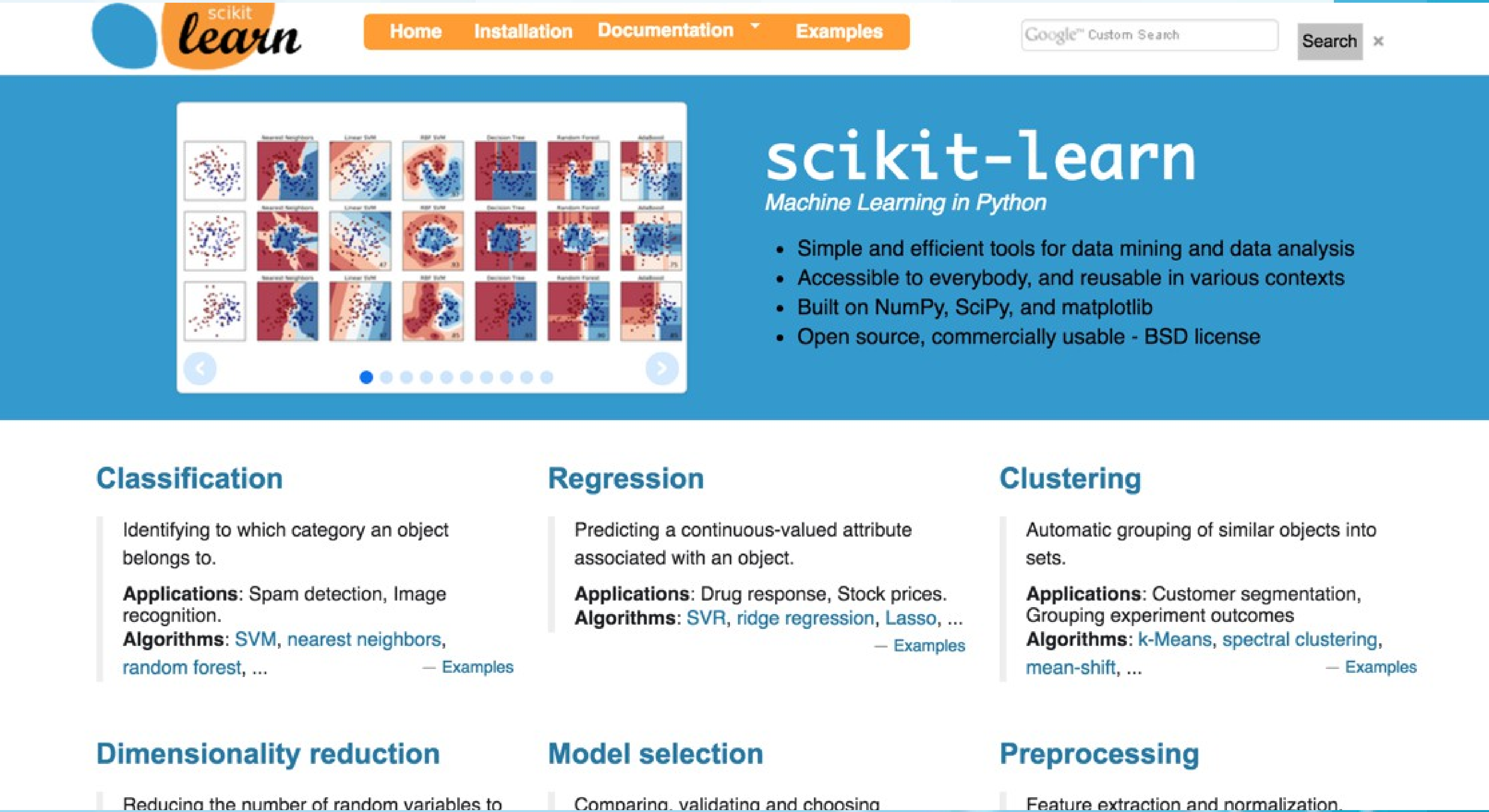
- Machine Learning is about building programs with tunable parameters that are adjusted automatically so as to improve their behavior by adapting to previously seen data
- Machine Learning is a subfield of Artificial Intelligence

Machine Learning Steps

1. Get Data
2. Clean and Preprocess Data
3. Shuffle (Randomize) Data
4. Split into Training/Test Data
5. Set Model Parameters - Learning Rate, Loss Function, Optimizer
6. Training the Model
7. Evaluate the Model
8. Use the Model

SciKit Learn

http://scikit-learn.org/



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to

Model selection

Comparing, validating and choosing

Preprocessing

Feature extraction and normalization.

Scikits Learn Modules

- Classifications
- Regression
- Clustering
- Dimensional Reduction
- Model Selection
- Preprocessing

Module 2
Datasets

What is Dataset

- A dataset is a dictionary-like object that holds all the data and some metadata about the data
- This data is stored in the .data member which is a n_samples, n_features array
- In the case of supervised problem, one or more response variables are stored in the .target member

Numerical Features

Continuous values Eg

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

Categorical Features

Discrete values Eg

- color - RED, GREEN, BLUE
- cities - LONDON, DUBAI, SHANGHAI

Module 3
Supervised Learning

What is Supervised Learning

- In Supervised Learning, we have a dataset consisting of both features and labels.
- The input data (X) is associated with a target label (y)

Supervised Learning Examples

- Given a multicolor image of an object through a telescope, determine whether that object is a star, a quasar, or a galaxy.
- Given a photograph of a person, identify the person in the photo.
- Given a list of movies a person has watched and their personal rating of the movie, recommend a list of movies they would like

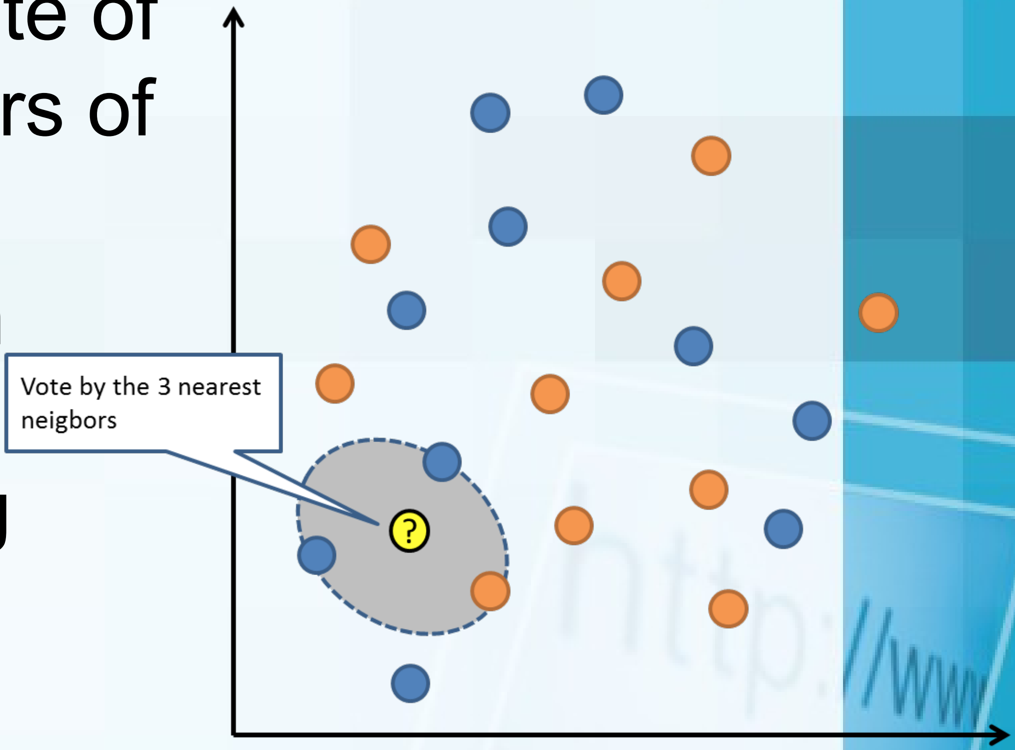
Classification

Key Classification Algorithms

- K Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Gaussian Naive Bayes (GND)
- Stochastic Gradient Descent (SGD)
- Decision Tree (DT)
- Ensemble Methods

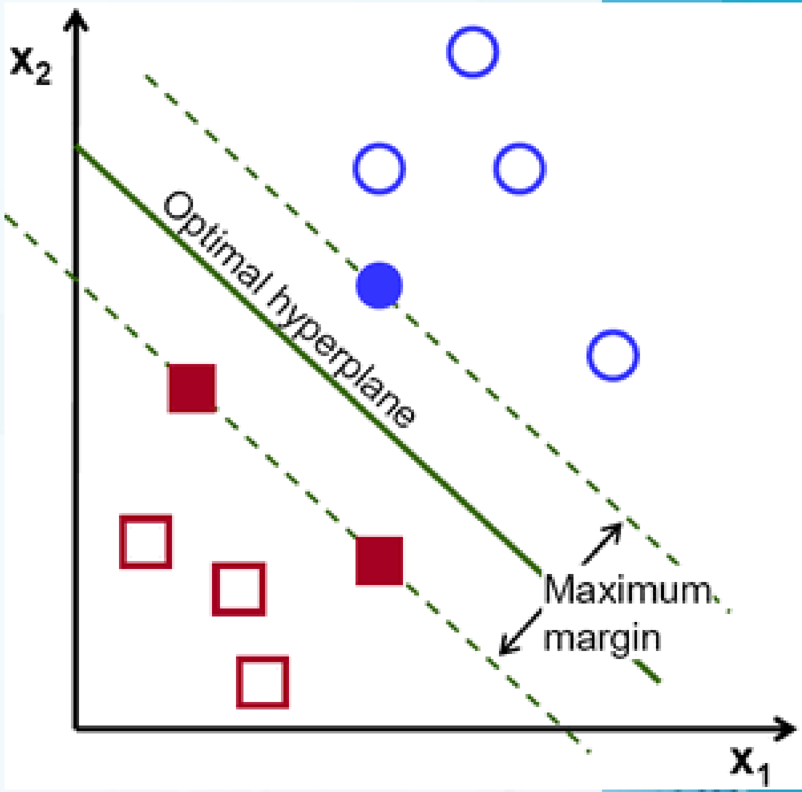
K Nearest Neighbors (KNN)

- A simple majority vote of the nearest neighbors of each point:
- Avoid using KNN on large dataset. It will probably take a long time



Support Vector Machine (SVM)

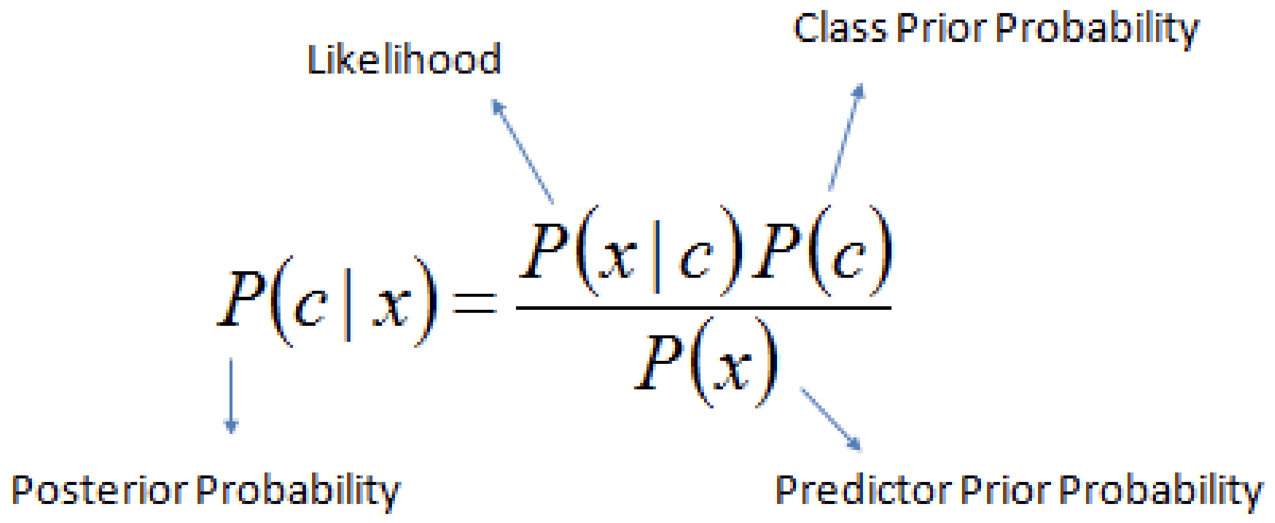
- Identify hyperplane (boundary) with maximum distance apart
- Use Kernels for nonlinear decision boundaries



$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Gaussian Naive Bayes (GNB)

- Use Probability Theory (Bayes Theorem)
- All variables (features) are equally important and independent

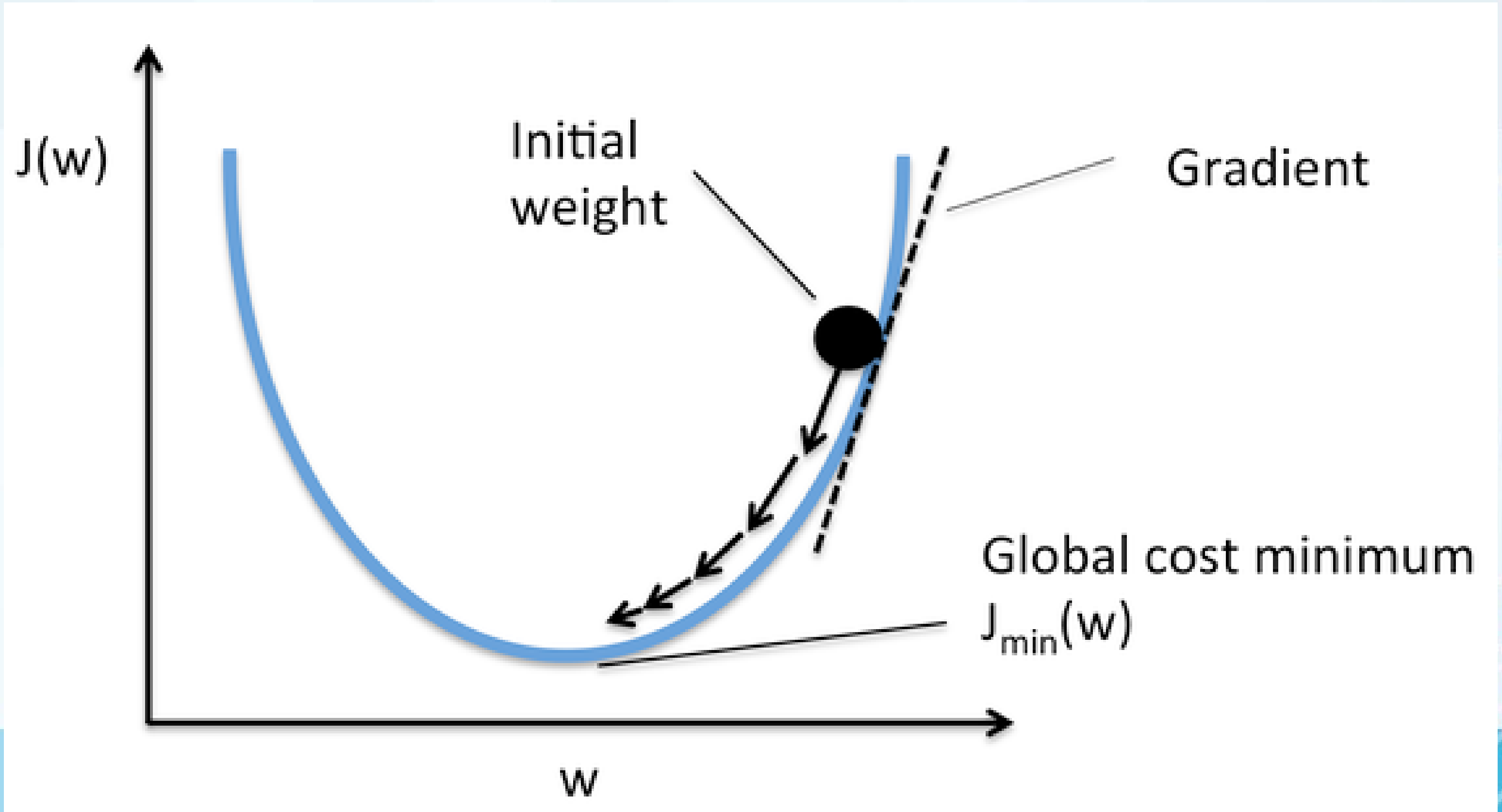


$$P(c \mid x) = \frac{P(x \mid c) P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

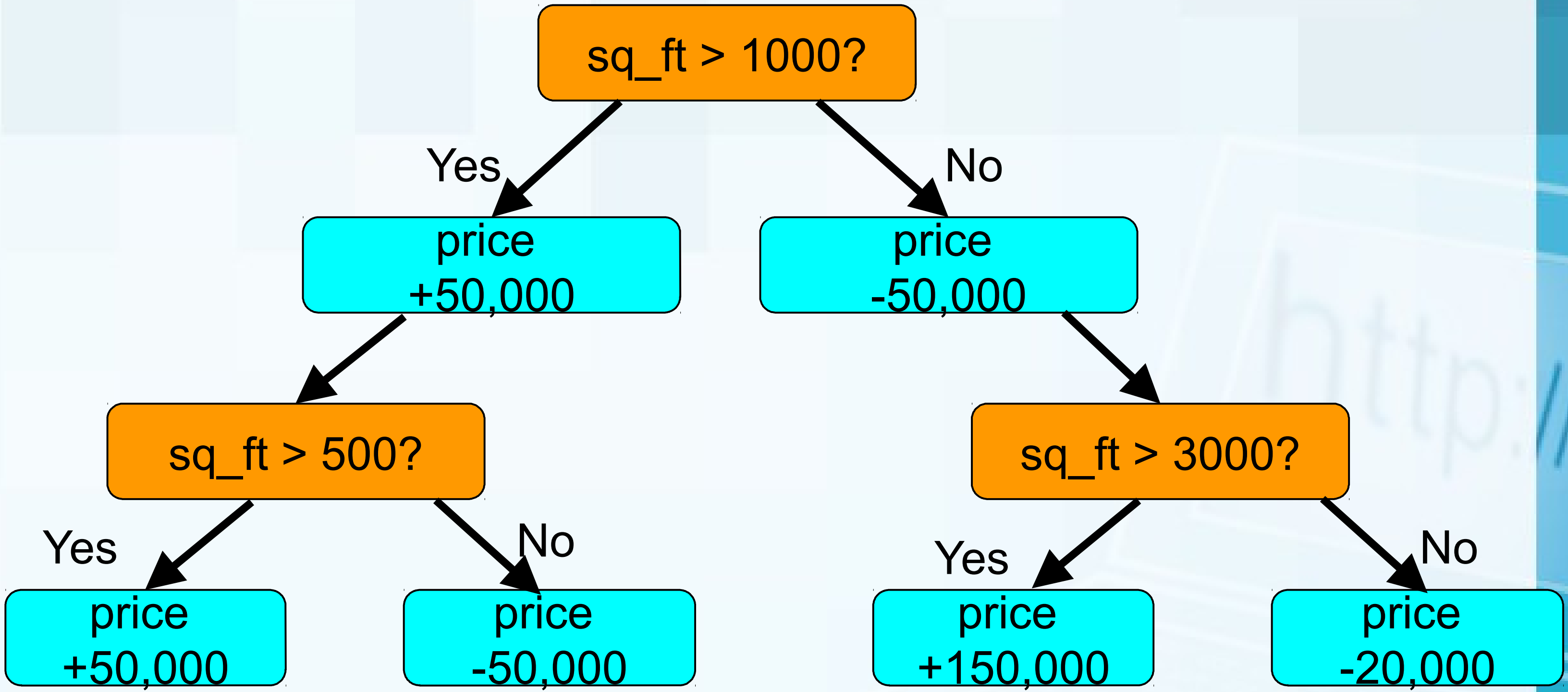
Stochastic Gradient Descent

Stochastic gradient descent (SGD) performs a parameter update for each training using the negative gradient



Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.



Decision Tree Pros and Cons

Pros:

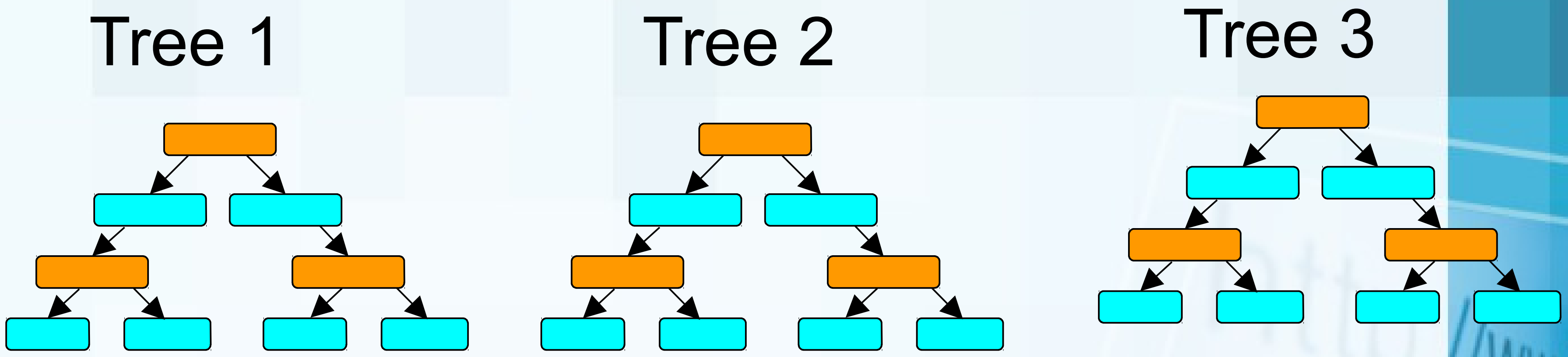
- Simple to understand and to interpret. Trees can be visualized.
- Able to handle both numerical and categorical data.

Cons:

- Decision-tree learners can create over-complex trees that do not generalise the data well (prompt to overfitting)

Random Forest Classifier

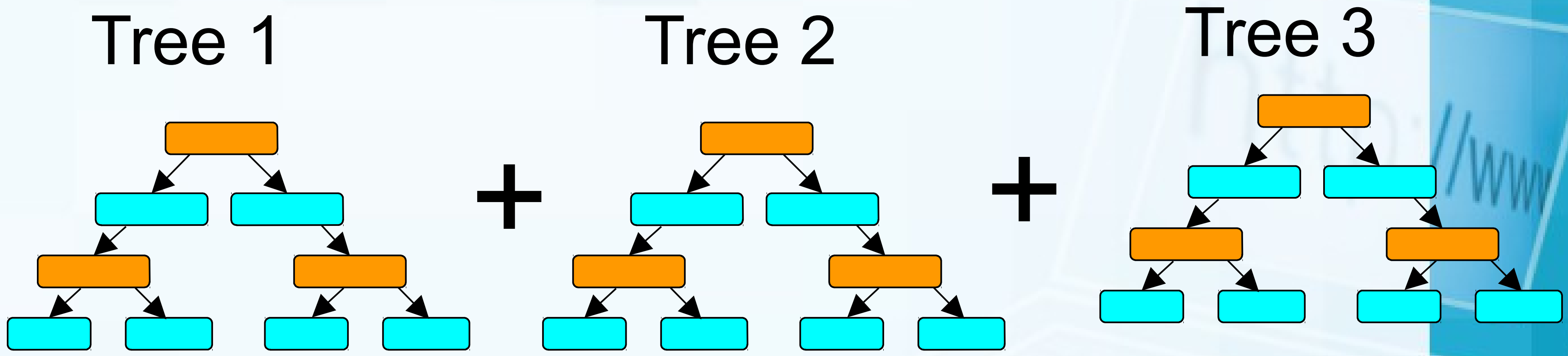
Random Forest classifier is a bagging ensemble method based on lots of decision trees with random selection of subsets of training samples



The result is based on the majority votes from all the decision trees

Gradient Boosting Classifier

Gradient Boosting is a ensemble boosting method that "boosting" many weak predictive models into a strong one, in the form of ensemble of weak models.



Classification Steps

```
# Step 1 Load classifier module
from sklearn import svm
clf = svm.SVC()

# Step 2 Learning/Training
clf.fit(digits.data,digits.target)

# Step 3: Testing/Predict
clf.predict(digits.data[1:2])
```

Classifier Methods

- fit(X,y) : training with the training data set (X,y)
- predict(X): predict with the test data (X)
- score(X,y): Returns the mean accuracy on the given test data (X(and labels (y)

KNN Classifier

```
from sklearn import neighbors

clf = neighbors.KNeighborsClassifier()
clf.fit(X, y)
```

KNN Classifier Parameters

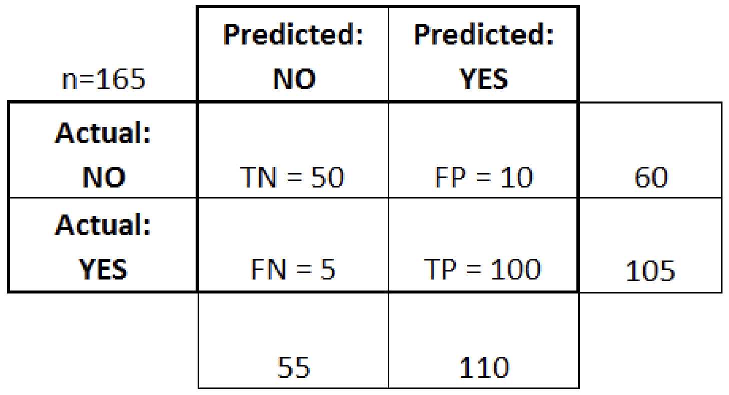
n_neighbor: number of neighbours (default=5)
weights: **uniform**, distance
algorithm: **auto**, ball_tree, kd_tree, brute
metric: **minkowski**

Sckit Learn Score Metric

```
clf.score(X_test,y_test)
```

Confusion Matrix

- **True Positives (TP):** We predicted positive, and the truth is positive.
- **True Negatives (TN):** We predicted negative, and the truth is negative.
- **False Positives (FP):** We predicted positive, but the truth is negative. ("Type I error.")
- **False negatives (FN)**: We predicted negative, but the truth is positive. ("Type II error.")



n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Precision and Recall

Precision = TP/(TP+FP)
Recall = TP/(TP+FN)

We want high precision and high recall (true positive rate) metric for a good model

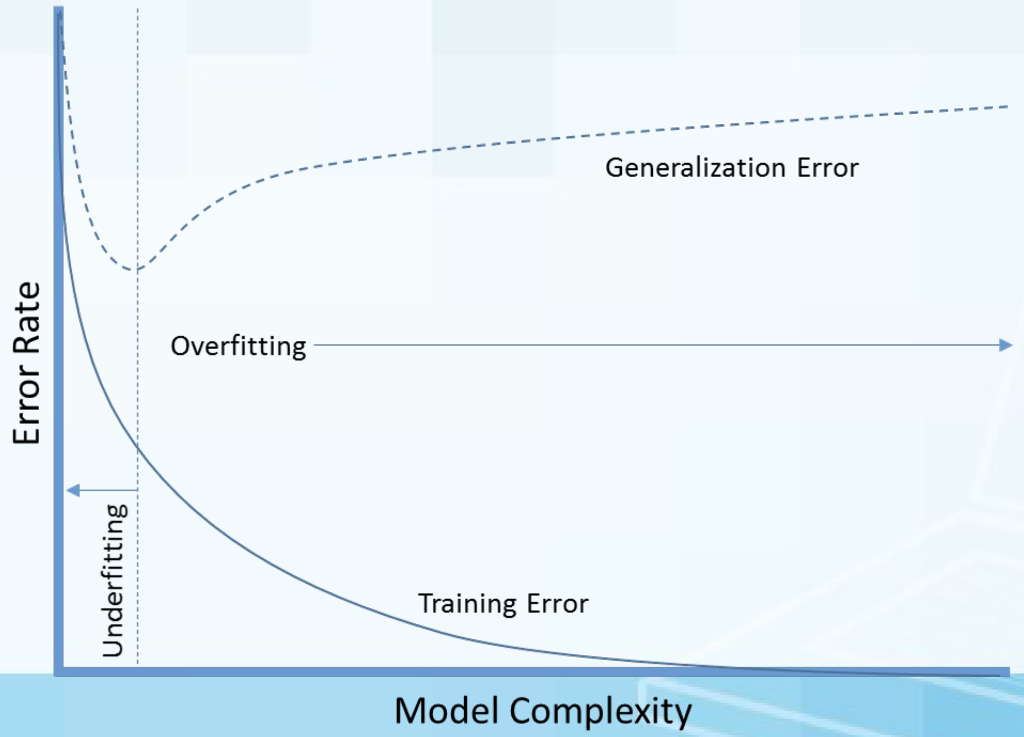
Classification Report

```
from sklearn import metrics
metrics.classification_report(expected, predicted)
```

```
             precision   recall   f1-score   support
        0      1.00       1.00     1.00        11
        1      0.94       0.94     0.94        16
        2      0.91       0.91     0.91        11
avg / total   0.95       0.95     0.95        38
```

OverFitting and UnderFitting

Overfitting – model 'memorizes' training data
Underfitting – model unable to predict training data



Model Persistence

After training a scikit-learn model, it is desirable to have a way to persist the model for future use without having to retrain.

Model Persistence - Joblib

```
# Export the model
from sklearn.externals import joblib
joblib.dump(clf, 'mymodel.pkl')


# Import the model
from sklearn.externals import joblib
clf = joblib.load('mymodel.pkl')
```

Model Persistence - Pickle

```
# Export the model
import pickle
pickle.dump(clf, open("mymodel2.pkl","wb"))

# Import the model
import pickle
clf = pickle.load(open("mymodel2.pkl","rb"))
```

Regression

Key Regression Algorithms

- Linear Regression
- Logistics Regression
- Decision Tree Regression
- SGD Regression

Module 4
Unsupervised Learning

UnSupervised Learning

- In Unsupervised Learning, we only have unlabeled input data
- We are interested in finding similarities between the objects in question.
- You can think of unsupervised learning as a means of discovering labels from the data itself.

Applications

- **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Insurance:** Identifying groups of motor insurance policy holders with a high average claim cost

Clustering

What is Clustering?

- Clustering is the task of grouping a set of objects in such a way that objects in the same group (cluster) are more similar to each other than to those in other groups (clusters).
- Clustering is a unsupervised learning since no labels (targets) are needed for training.

Basic Steps in Clustering

- Feature selection
  - Must be properly selected so as to encode as much information as possible.
- Proximity measure
  - Quantifies how similar or dissimilar two feature vectors are.
- Clustering Criterion
  - Depends on the interpretation the expert gives to the term sensible.

Key Clustering Algorithms

- K-Means
- Mean Shift
- Agglomerative Clustering

K-Means Clustering Method

Given *k*, the *k-means* algorithm is implemented as follows:

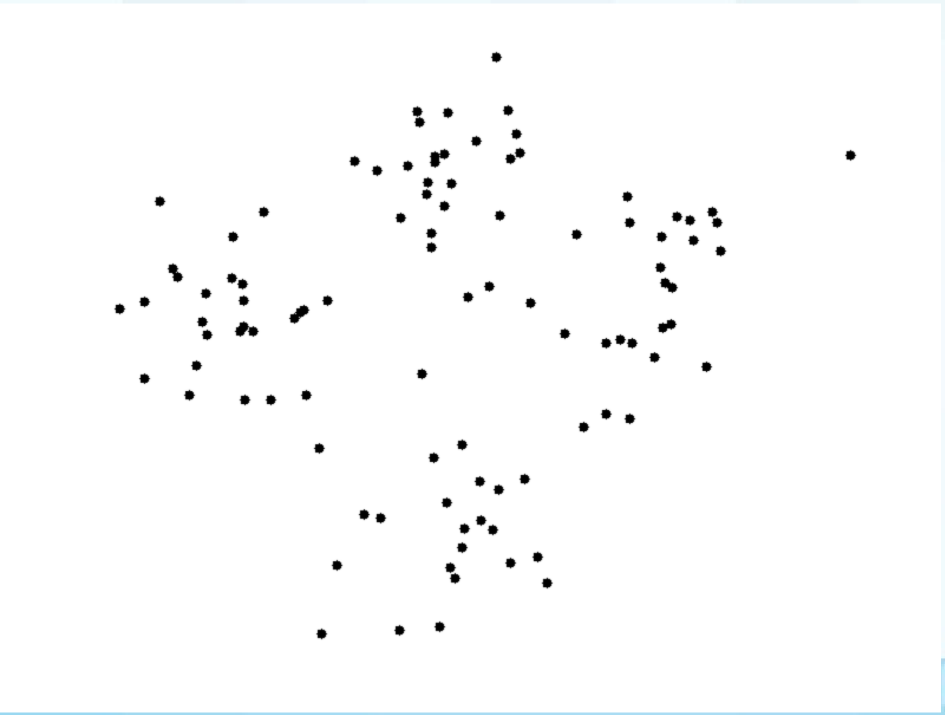
- Partition objects into *k* nonempty subsets
- Compute seed points as the centroids of the clusters of the current partition
- Assign each object to the cluster with the nearest seed point
- Repeat from Step 2, until no more new assignment

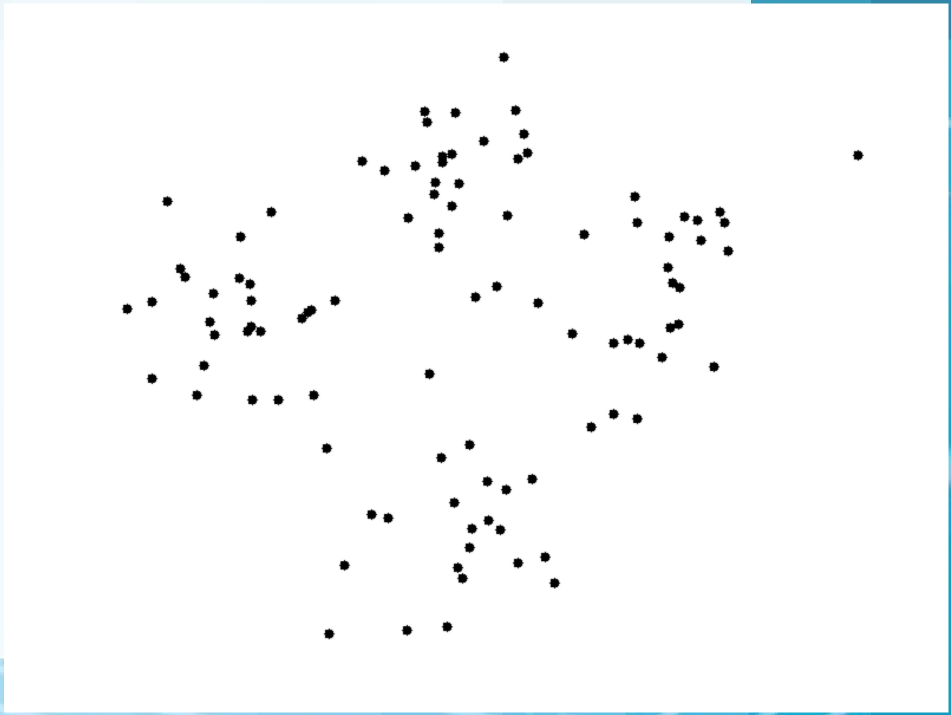
K Mean Algorithm Simulation

Click on the images below to start k-mean clustering simulation (k=4)

Seeds starts on left

Seeds randomly assigned





K-Means Pros & Cons

Pros:

- Simple and fast, Easy to implement

Cons:

- Need to choose K
- Sensitive to outliers
- Prone to local minima.
- Extremely sensitive to initialization. Bad initialization can lead to:
  - poor convergence speed
  - bad overall clustering

K-Means Clustering

sklearn.cluster.KMeans(n_clusters = …)

n_clusters: number of clusters

Clustering Attributes

cluster_centers_: Coordinates of cluster centers
labels_ : Labels of each point

Clustering Steps

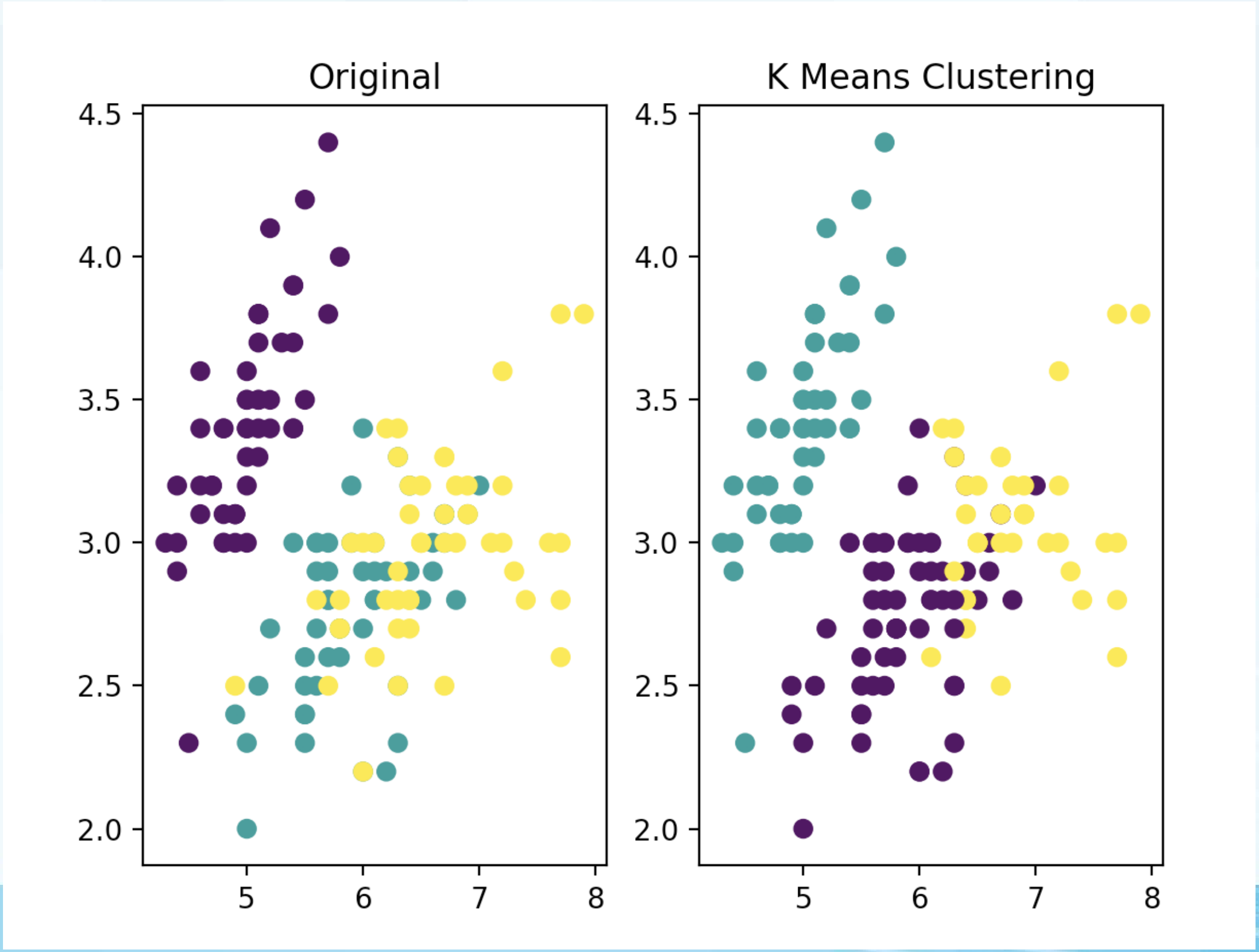
```
# Step 1 Model
from sklearn import cluster
cluster = cluster.KMeans(n_clusters=2)

# Step 2 Training
cluster .fit(X)

# Step 3 Evaluation
plt.scatter(X[:,0],X[:,1],c=cluster.labels_)
plt.show()
```

K- Means Clustering Demo

- Blob generator dataset
- Iris dataset



Challenge: K-Means Clustering

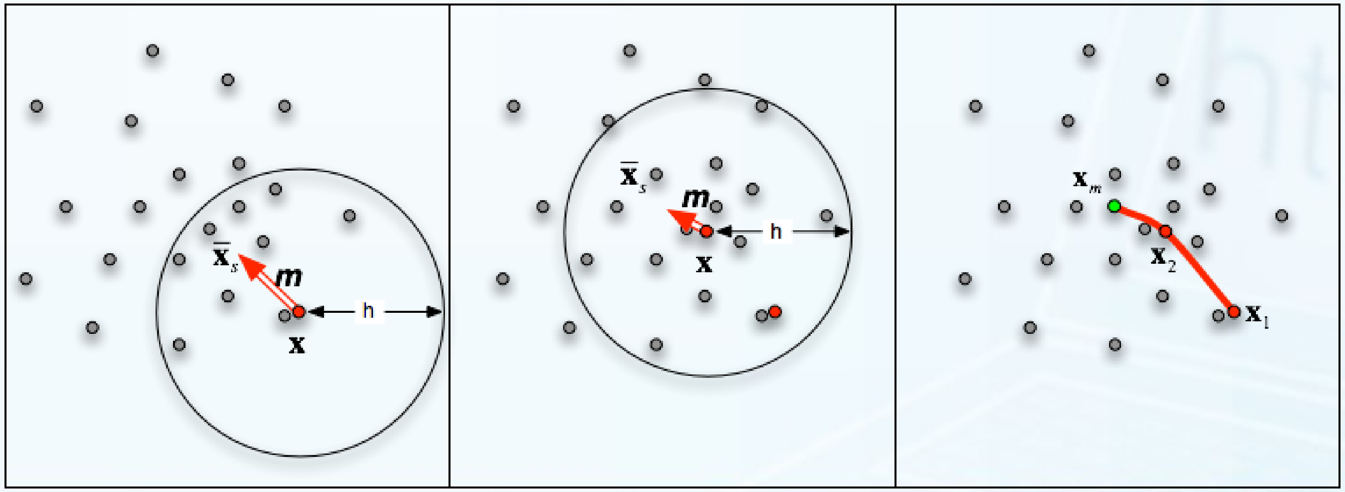
Apply K-Means Clustering to handwritten digits dataset

Time: 5 mins

Mean Shift Clustering

sklearn.cluster.MeanShift()

Mean shift clustering is a centroid-based algorithm, which works by updating candidates for centroids to be the mean of the points within a given region



Mean Shift Pros & Cons

Pros:

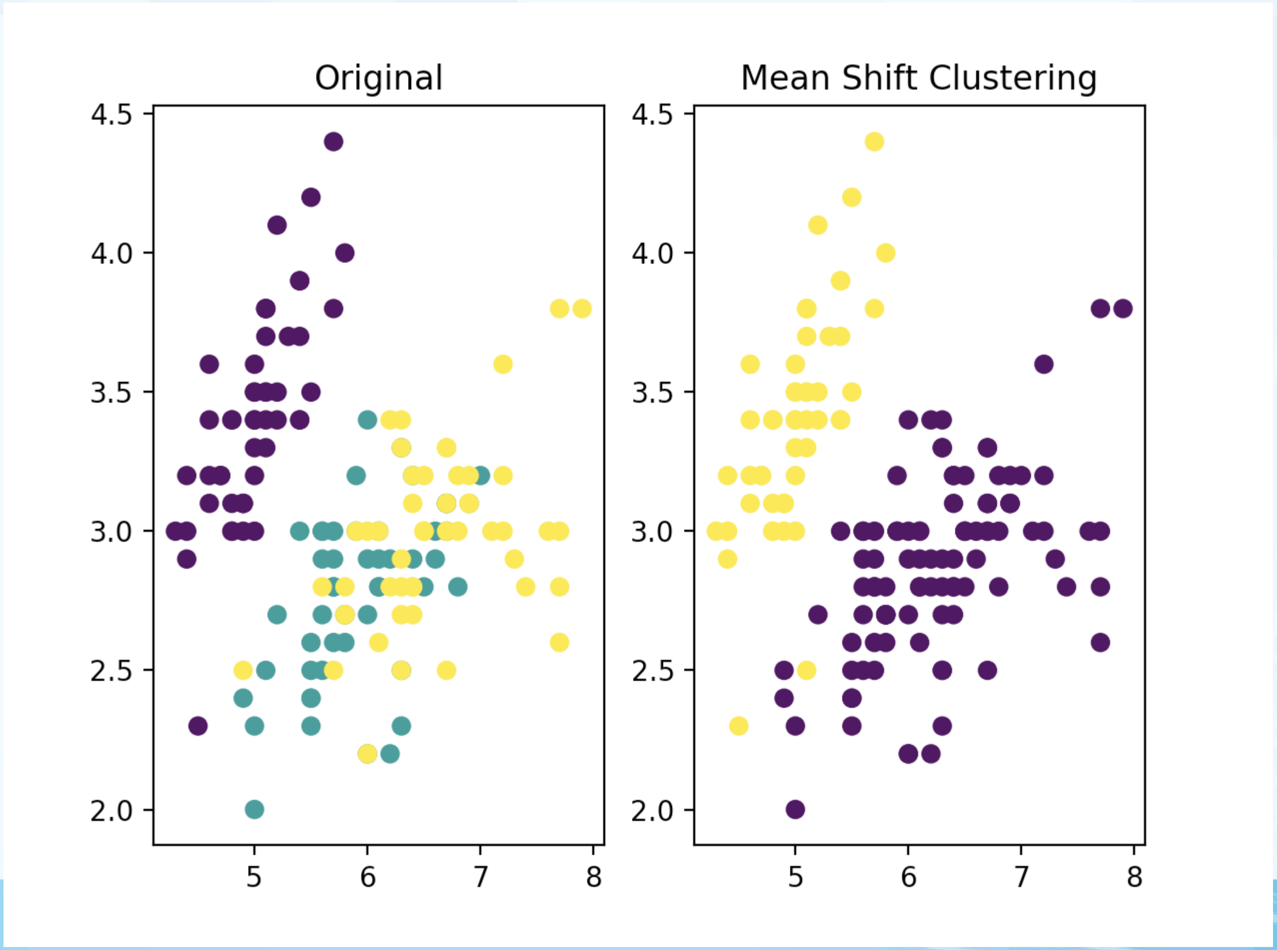
- Don't need to specify number of clusters
- Robust to outliers
- Depend on single parameter - window size

Cons:

- Output depends on window size
- Computationally expensive

Mean Shift Clustering Demo

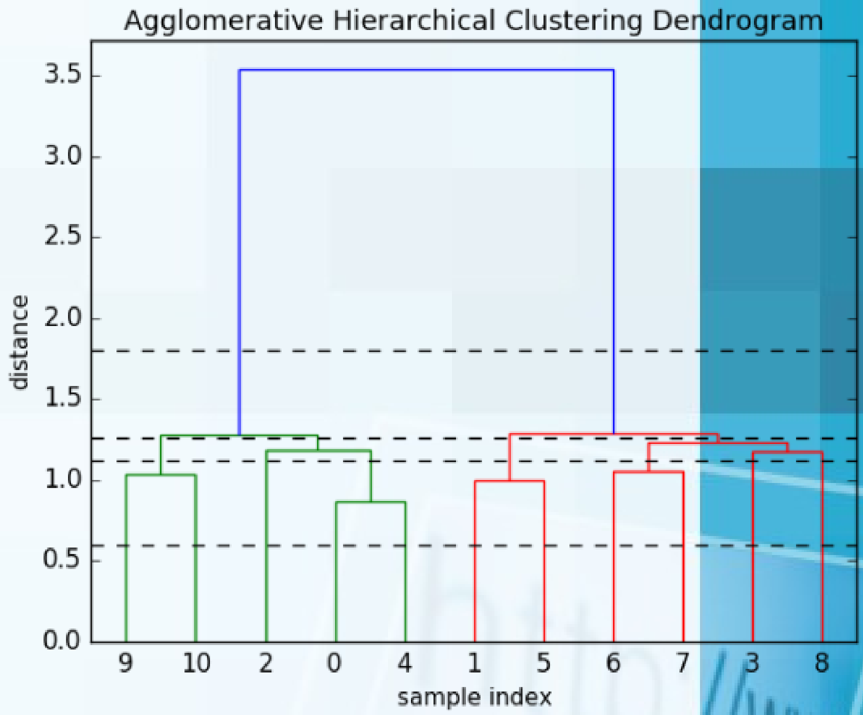
- Blob generator dataset
- Iris dataset



Hierarchical Clustering

Hierarchical clustering seeks to build a hierarchy of clusters. hierarchical clustering generally fall into two types:

- Agglomerative: Bottom up approach
- Divisive: Top down approach



Agglomerative Clustering

sklearn.cluster.AgglomerativeClustering(n_clusters = 2, affinity = 'euclidean', linkage = 'ward'..)

Affinity

Metric used to compute the linkage.

- euclidean,
- l1
- l2
- manhattan
- cosine
- precomputed

If linkage is “ward”, only “euclidean” is accepted.

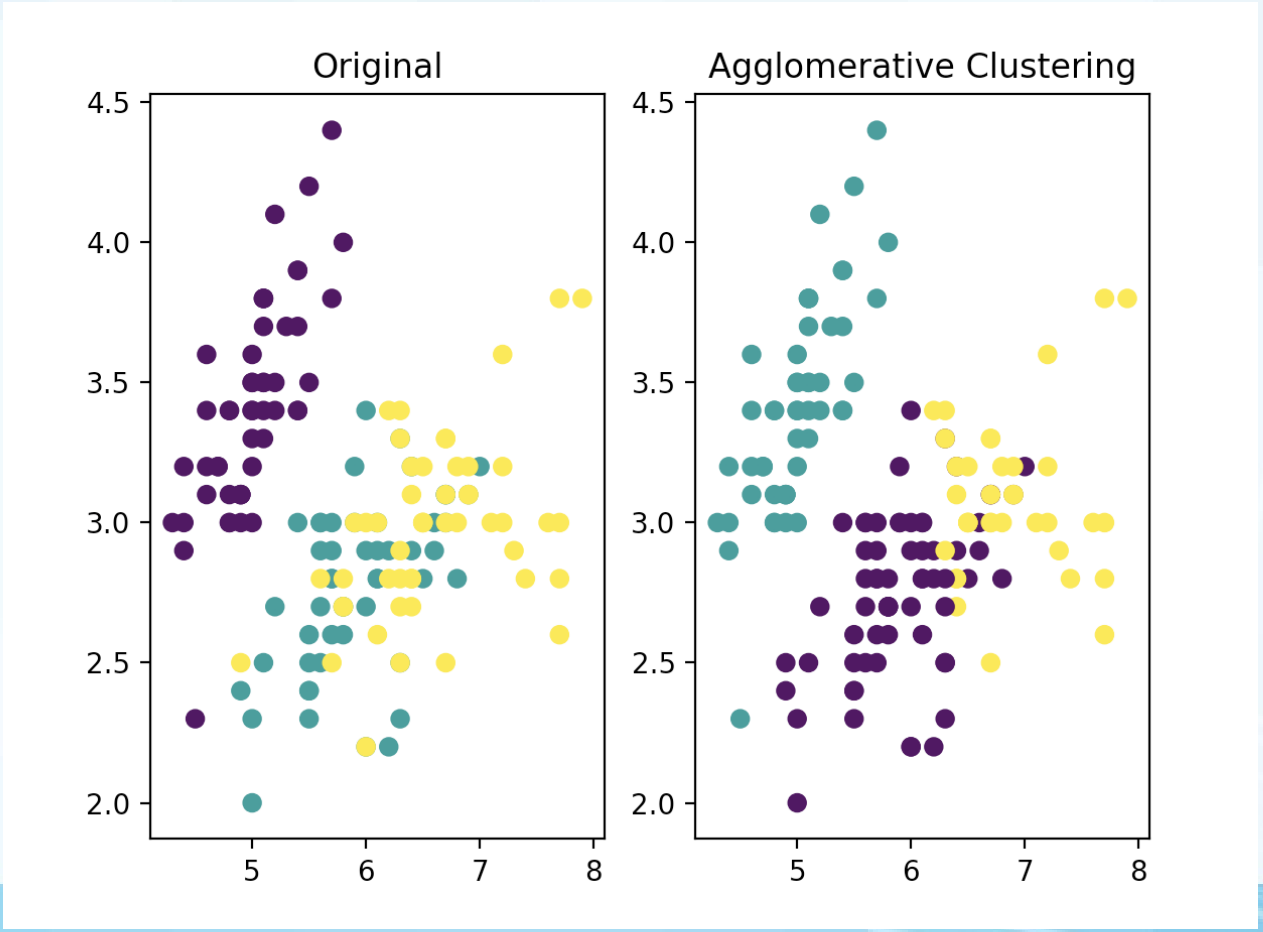
Linkage

The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion.

- ward
- average
- complete

Agglomerative Clustering Demo

- Blob generator dataset
- Iris dataset

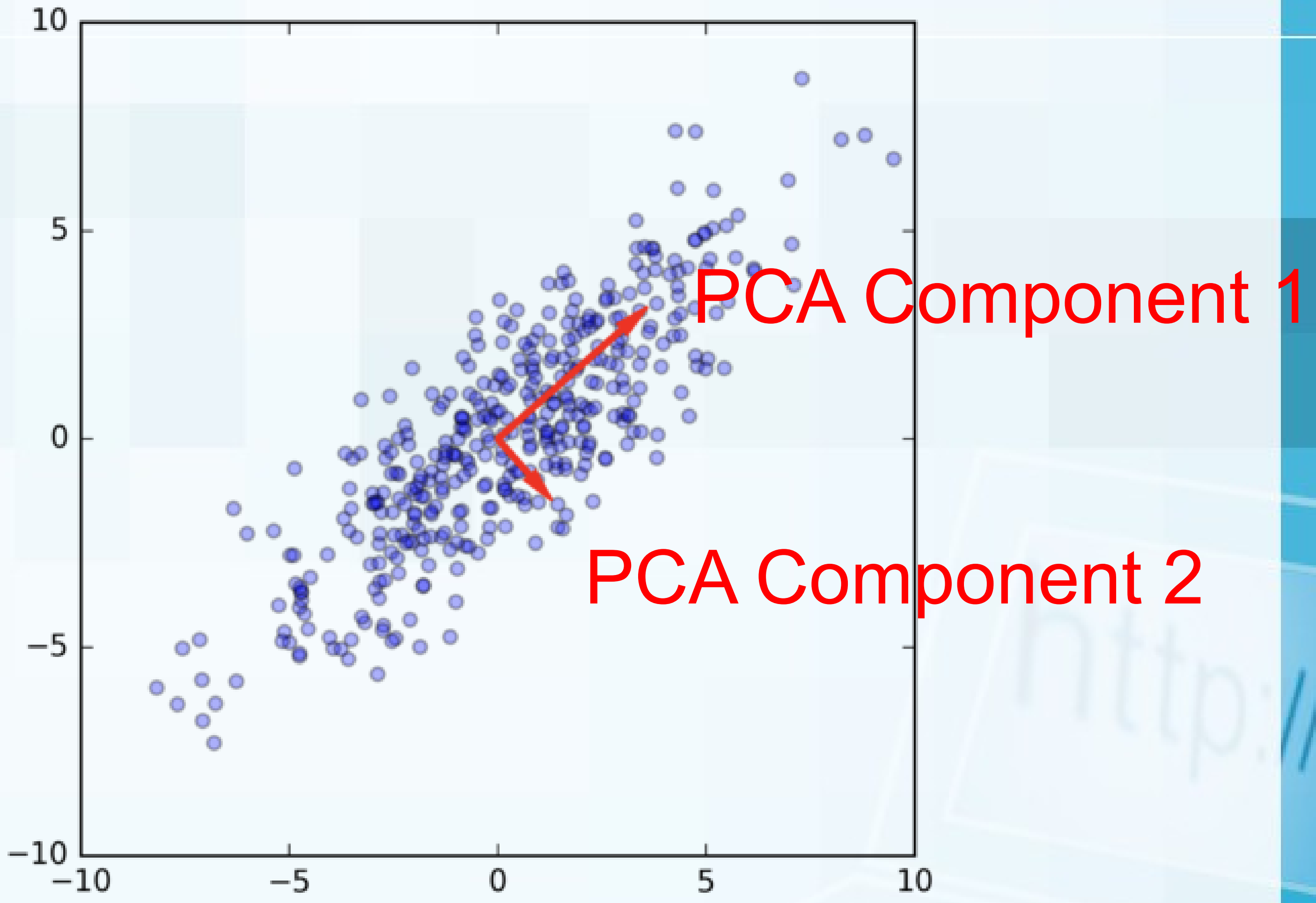


Dimensionality Reduction

Principal Component Analysis

- Principal component analysis (PCA) is a dimensionality reduction technique.
- Suppose you have 10 features (dimensions), and you can use use PCA to reduce it to a few key features (dimensions) that can capture the most variation of the data.
- PCA was invented in 1901 by Karl Pearson

Principal Component Analysis



The principle components have the largest variations

SK PCA Attributes

- **explained_variance_:** The amount of variance explained by each of the selected components.
- **explained_variance_ratio_:** Percentage of variance explained by each of the selected components.
- **components_:** Principal axes in feature space, representing the directions of maximum variance in the data. The components are sorted by explained_variance_.
- **n_components_:** The estimated number of components

SK PCA Methods

- **fit(X):** Fit the model with X
- **transform(X):** Apply dimensionality reduction to X.
- **fit_transform(X):** it the model with X and apply the dimensionality reduction on X.

PCA Simple Demo

```
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1],
[3, 2]])

from sklearn import decomposition
pca = decomposition.PCA(n_components=2)
pca.fit(X)
```

PCA on Iris Dataset

```
pca = decomposition.PCA(n_components=2)
pca.fit(X)
X_t = pca.transform(X)
```

PCA Components

To determine the components,

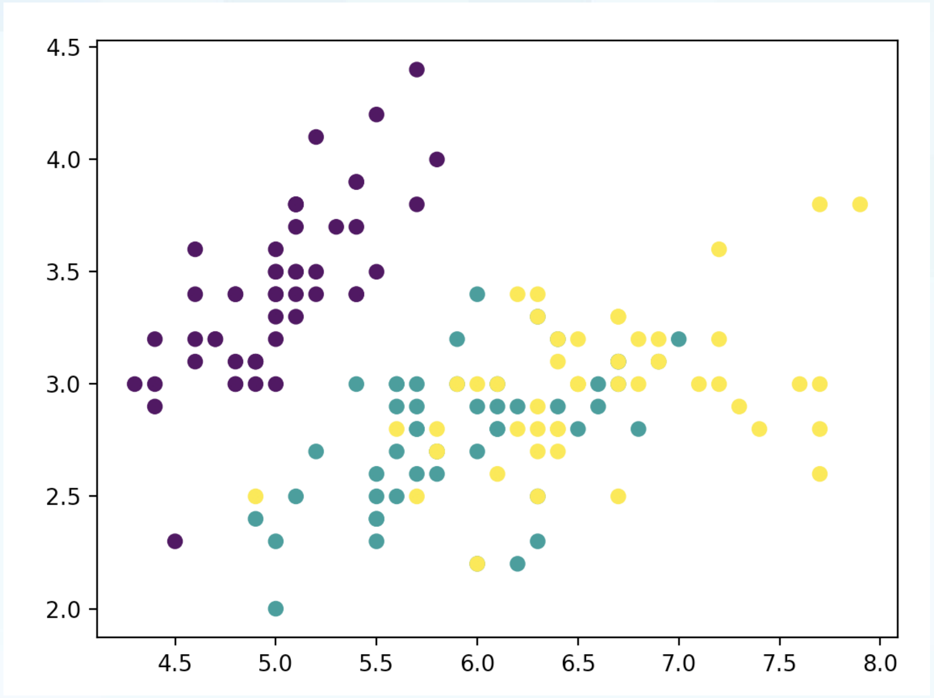
```
comps = pd.DataFrame(pca.components_,
columns=iris.feature_names)
```

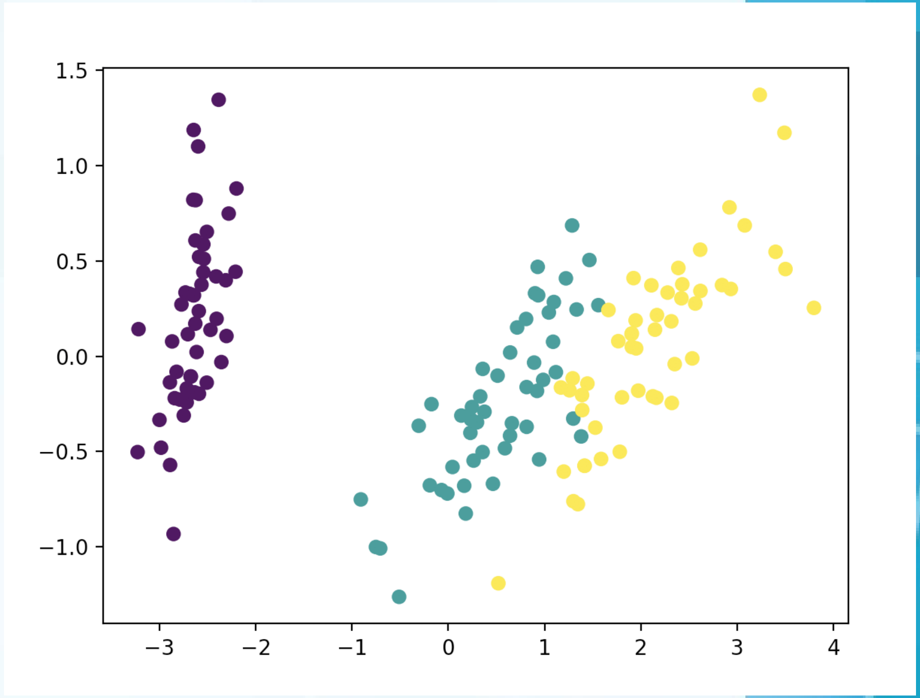
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0           0.361590         -0.082269           0.856572          0.358844
1           0.656540          0.729712          -0.175767         -0.074706
2          -0.580997          0.596418           0.072524          0.549061
3           0.317255         -0.324094          -0.479719          0.751121
```

PCA vs Original

Original X_1 vs X_2

PCA X_1 vs X_2





Module 5
Model Selection

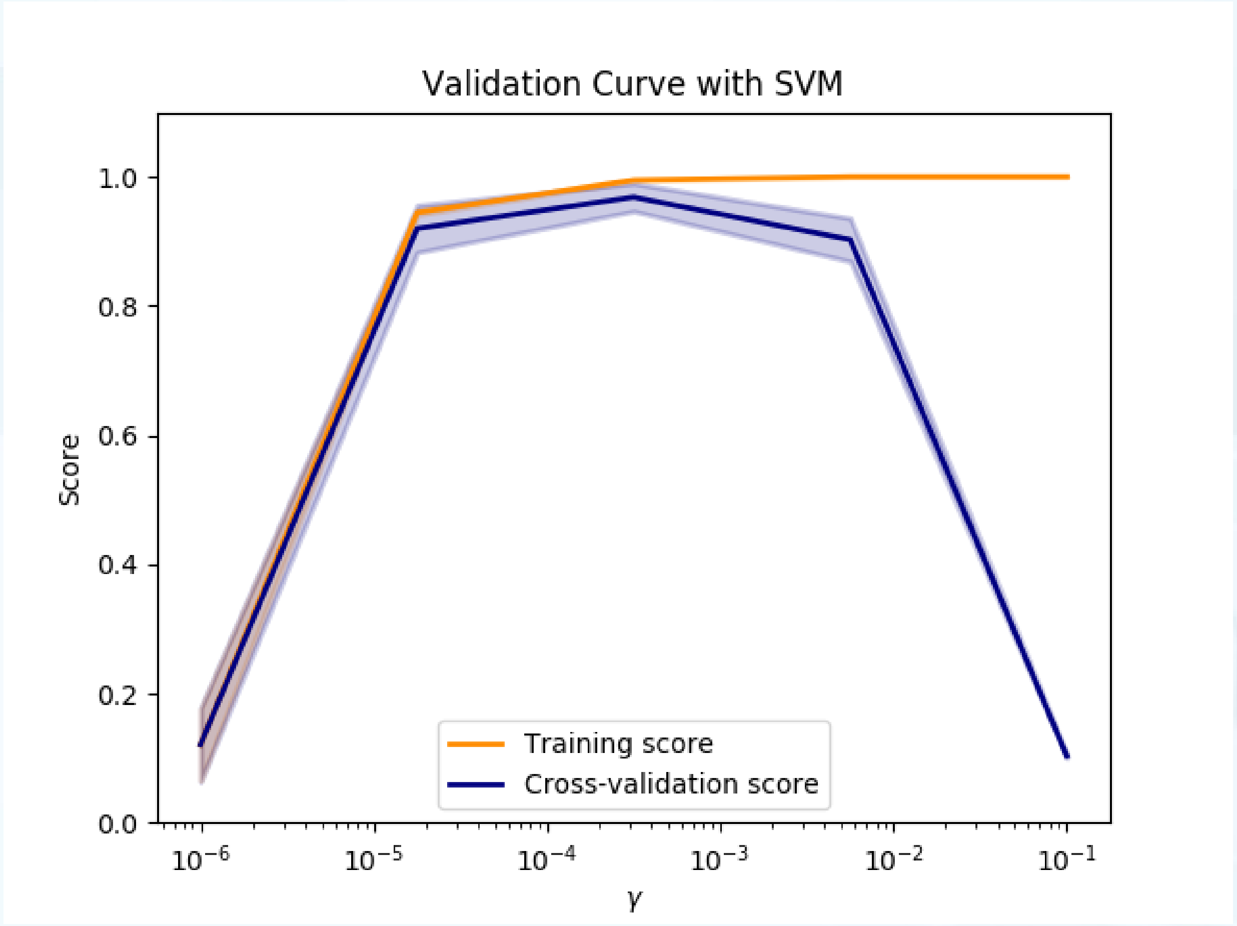
Tuning the hyper-parameters of an estimator

- Hyper-parameters - are not directly learnt within estimators
- Search the hyper-parameter space
  - Exhaustive Grid Search
  - Randomized Parameter Search

Pipeline and FeatureUnion: combining estimators

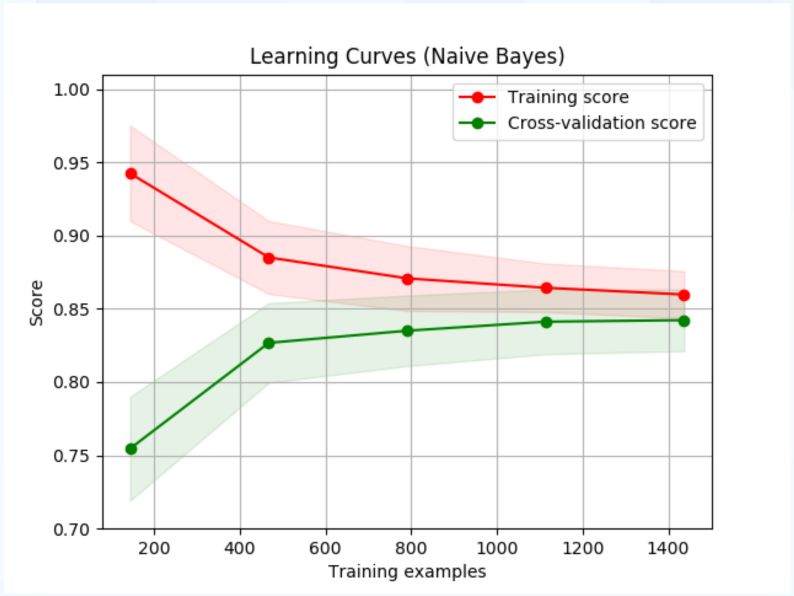
- Useful as there is often a fixed sequence of steps in processing the data
- Convenience and encapsulation
- Joint parameter selection

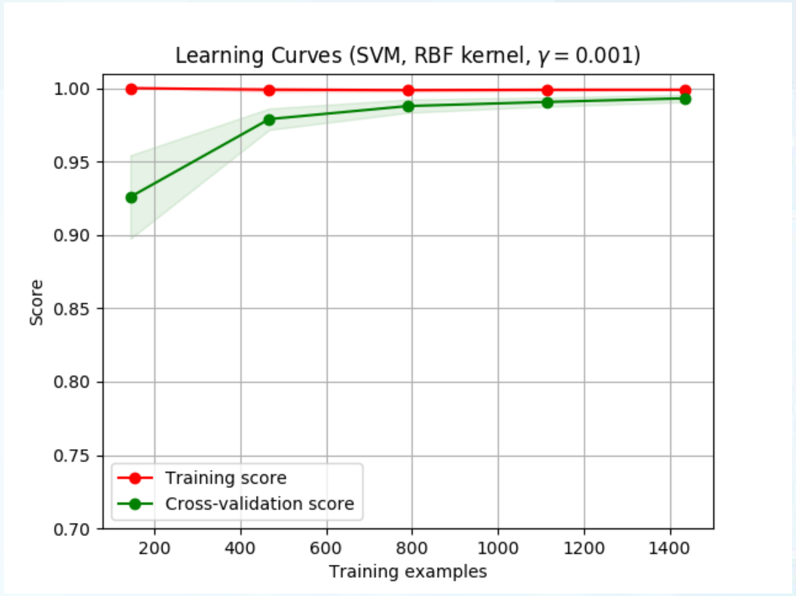
Validation curve



Learning curve

- Shows the validation and training score of an estimator





Summary
Parting Message

Q&A

Feedback

Thank You!

malaysiacourses@tertiaryinfotech.com