

# **Machine Learning : An Introduction**

*Prof. Dr. Noman Islam*

---

# Outline

- Introduction to machine learning
- Getting started with Scikit-learn
- Machine learning concepts
- Basic mathematics
- Advanced concepts of machine learning
- Deep dive into scikit-learn

# Introduction

- Deep learning is a specific kind of machine learning.
- A machine learning algorithm is an algorithm that is able to learn from data.
- A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  , if its performance at tasks in  $T$  , as measured by  $P$  , improves with experience  $E$

# Task

- Machine learning allows us to tackle tasks that are too difficult to solve with fixed programs written and designed by human beings.
- Machine learning tasks are usually described in terms of how the machine learning system should process an example
- An example is a collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process.
- The features of an image are usually the values of the pixels in the image

# Common machine learning tasks

- Classification: the learning algorithm is usually asked to produce a function  $f: \mathbb{R}^n \rightarrow \{1, \dots, k\}$ . An example of a classification task is object recognition, where the input is an image
- Regression: In this type of task, the computer program is asked to predict a numerical value given some input. To solve this task, the learning algorithm is asked to output a function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$

- Transcription: In this type of task, the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe it into discrete, textual form. For example, in optical character recognition, the computer program is shown a photograph containing an image of text and is asked to return this text in the form of a sequence of characters. Another example is speech recognition, where the computer program is provided an audio waveform and emits a sequence of characters or wordID codes describing the words that were spoken in the audio recording.

- Machine Translation: In a machine translation task, the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language.
- Anamoly Detection: In this type of task, the computer program sifts through a set of events or objects, and flags some of them as being unusual or atypical. An example of an anomaly detection task is credit card fraud detection.

- Synthesis and Sampling: In this type of task, the machine learning algorithm is asked to generate new examples that are similar to those in the training data. For example, video games can automatically generate textures for large objects or landscapes, rather than requiring an artist to manually label each pixel



# Scikit-learn

- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.
- It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python.
- This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

# Installation

- `pip install -U scikit-learn`
- `conda install scikit-learn`

# Datasets

- A collection of data is called dataset. It is having the following two components –
  - Features – The variables of data are called its features. They are also known as predictors, inputs or attributes.
    - *Feature matrix* – It is the collection of features, in case there are more than one.
    - *Feature Names* – It is the list of all the names of the features.

- Response – It is the output variable that basically depends upon the feature variables. They are also known as target, label or output.
  - *Response Vector* – It is used to represent response column. Generally, we have just one response column.
  - *Target Names* – It represent the possible values taken by a response vector.
- Scikit-learn have few example datasets like iris and digits for classification and the Boston house prices for regression.

# Loading data in sk-learn

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])
```

# Splitting the data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size = 0.3, random_state = 1  
)
```

# Training a classifier

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.4, random_state=1
)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
classifier_knn = KNeighborsClassifier(n_neighbors = 3)
classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)
```



```
# Finding accuracy by comparing actual response values(y_test)with  
    predicted response value(y_pred)  
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))  
# Providing sample data and the model will make prediction out of that data  
sample = [[5, 5, 3, 2], [2, 4, 3, 5]]  
preds = classifier_knn.predict(sample)  
pred_species = [iris.target_names[p] for p in preds] print("Predictions:",  
    pred_species)
```

# Saving model





```
from sklearn.externals import joblib  
joblib.dump(classifier_knn, 'iris_classifier_knn.joblib')
```

```
joblib.load('iris_classifier_knn.joblib')
```

# Performance Measure

- Accuracy is just the proportion of examples for which the model produces the correct output
- Error rate , the proportion of examples for which the model produces an incorrect output
- Test dataset
- Overfitting
- Underfitting

# Precision and Recall

		Predicted	
			
Actual		TP	FN
		FP	TN

$$precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

# Experience

- Supervised Learning
- Unsupervised Learning
- A dataset is a collection of many examples
- One of the oldest datasets studied by statisticians and machine learning researchers is the Iris dataset

# Unsupervised learning

- Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset
- Clustering

# Supervised Learning

- Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target
- A supervised learning algorithm can study the Iris dataset and learn to classify iris plants into three different species based on their measurements

- Roughly speaking, unsupervised learning involves observing several examples of a random vector  $x$ , and attempting to implicitly or explicitly learn the probability distribution  $p(x)$ , or some interesting properties of that distribution
- Unsupervised learning and supervised learning are not formally defined terms.
- Some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences



# Linear Regression

- The goal is to build a system that can take a vector  $x \in \mathbb{R}^n$  as input and predict the value of a scalar  $y \in \mathbb{R}$  as its output
- We define the output to be  $y = wx$ , where  $w \in \mathbb{R}^n$  is a vector of parameters
- We can think of  $w$  as a set of weights that determine how each feature affects the prediction

- One way of measuring the performance of the model is to compute the mean squared error of the model on the test set.
- If  $\hat{\mathbf{y}}(\text{test})$  gives the predictions of the model on the test set, then the mean squared error is given by

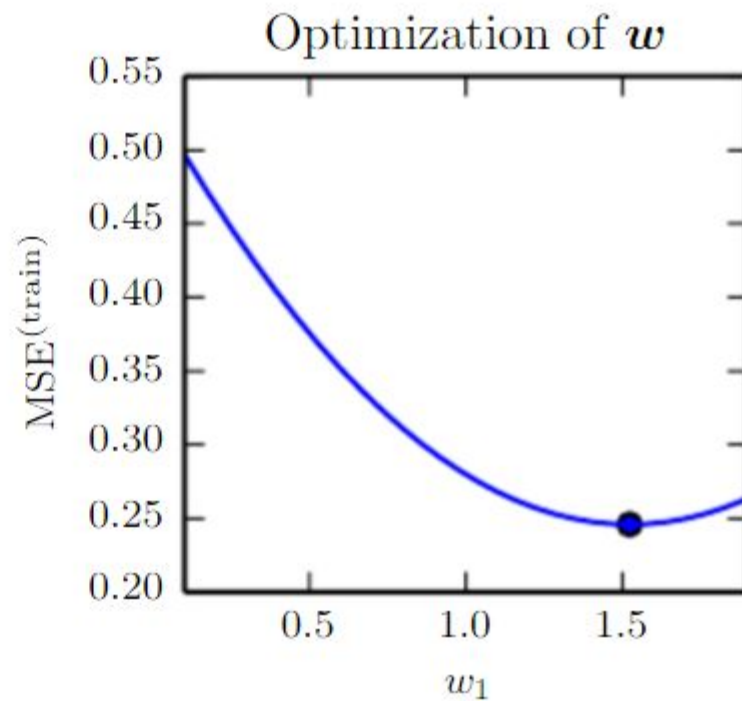
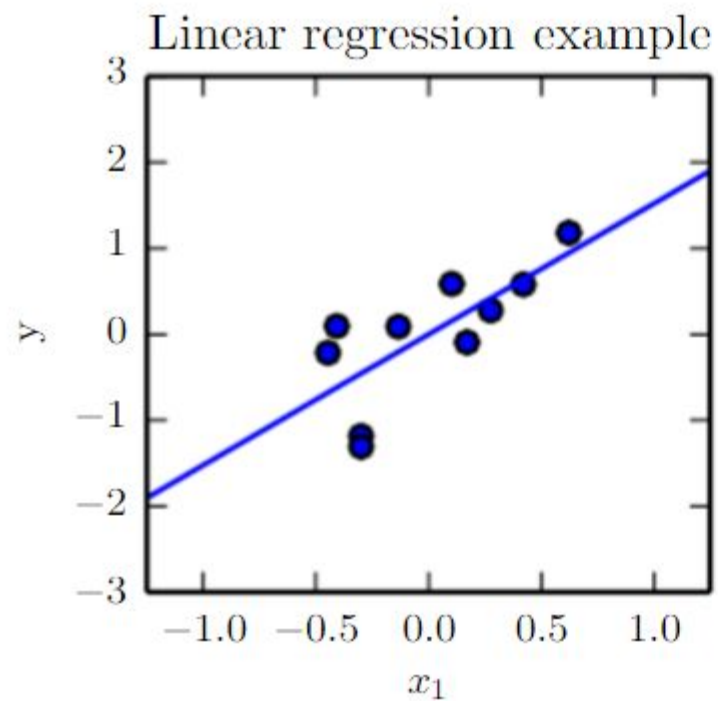
$$\text{MSE}_{\text{test}} = \frac{1}{m} \sum_i (\hat{\mathbf{y}}^{(\text{test})} - \mathbf{y}^{(\text{test})})_i^2.$$

To minimize  $\text{MSE}_{\text{train}}$ , we can simply solve for where its gradient is  $\mathbf{0}$ :

$$\nabla_{\mathbf{w}} \text{MSE}_{\text{train}} = \mathbf{0}$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m} \|\hat{\mathbf{y}}^{(\text{train})} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_{\mathbf{w}} \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$



- It is worth noting that the term linear regression is often used to refer to a slightly more sophisticated model with one additional parameter—an intercept term  $b$ . In this model










$$y = wx + b$$

- The intercept term  $b$  is often called the bias parameter

# Capacity, overfitting and underfitting

- The ability to perform well on previously unobserved inputs is called generalization
- Training error
- What separates machine learning from optimization is that we want the generalization error, also called the test error, to be low as well.
- The train and test data are generated by a probability distribution over datasets called the data generating process
- We typically make a set of assumptions known collectively as the i.i.d. assumptions

- The factors determining how well a machine learning algorithm will perform are its ability to:
  1. Make the training error small.
  2. Make the gap between training and test error small
- These two factors correspond to the two central challenges in machine learning:
  - Underfitting
  - and overfitting
- Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.
- Overfitting occurs when the gap between the training error and test error is too large.

	Train data	Test data	Result
ideal			
underfit			
overfit			



- We can control whether a model is more likely to overfit or underfit by altering its capacity
- Informally, a model's capacity is its ability to fit a wide variety of functions
- Models with low capacity may struggle to fit the training set.
- Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set

# Hypothesis space

- One way to control the capacity of a learning algorithm is by choosing its hypothesis space, the set of functions that the learning algorithm is allowed to select as being the solution

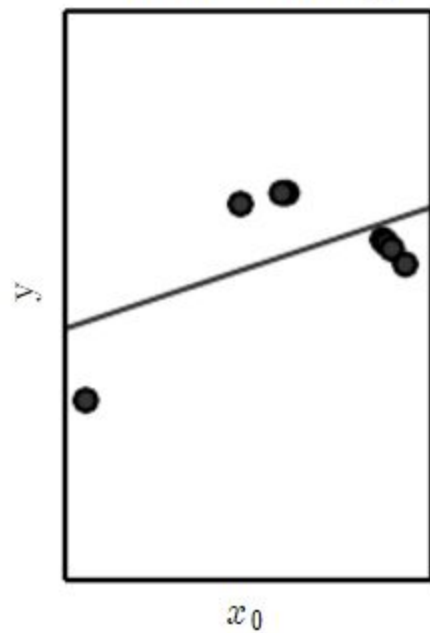
$$\hat{y} = b + wx.$$

$$\hat{y} = b + w_1x + w_2x^2.$$

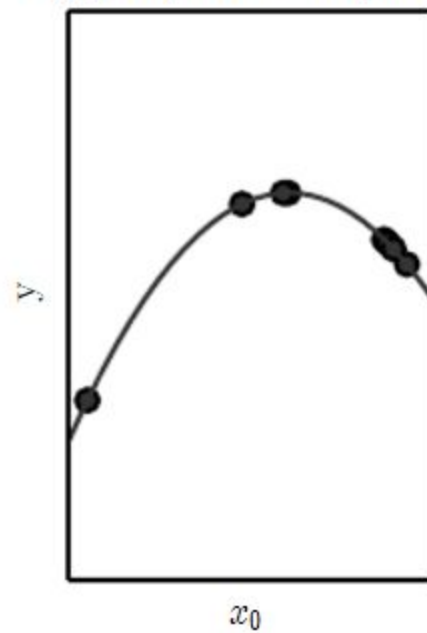
$$\hat{y} = b + \sum_{i=1}^9 w_i x^i.$$

- Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with.
- Models with insufficient capacity are unable to solve complex tasks.
- Models with high capacity can solve complex tasks, but when their capacity is higher than needed to solve the present task they may overfit.

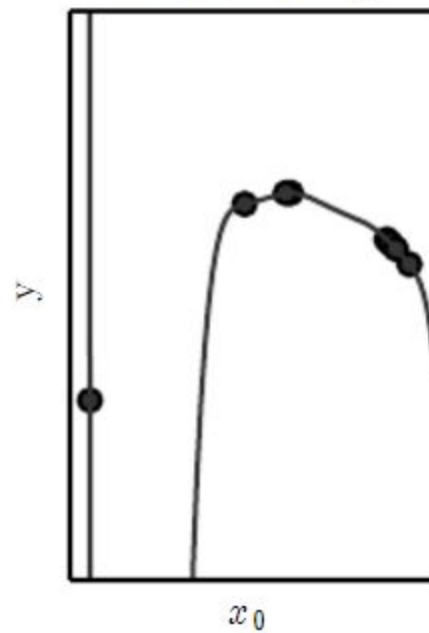
Underfitting



Appropriate capacity



Overfitting



# Nearest neighbor regression

- When asked to classify a test point  $x$ , the model looks up the nearest entry in the training set and returns the associated regression target.

# Hyper parameters

- Most machine learning algorithms have several settings that we can use to control the behavior of the learning algorithm
- The degree of the polynomial

# Bias and Variance

- Variance is how sensitive a prediction is to what training set was used. Ideally, how we choose the training set shouldn't matter – meaning a lower variance is desired.
- Bias is the strength of assumptions made about the training dataset. Making too many assumptions might make it hard to generalize, so we prefer low bias as well.
- A too flexible model has high variance and low bias, whereas a too strict model has low variance and high bias.
- Ideally we would like a model with both low variance error and low bias error. That way, it both generalizes to unseen data and captures the regularities of the data.

# Consistency

- Usually, we are also concerned with the behavior of an estimator as the amount of training data grows
- In particular, we usually wish that, as the number of data points  $m$  in our dataset increases, our point estimates converge to the true value of the corresponding parameters



# Logistic Regression

Linear regression uses the general linear equation  $Y = b_0 + \sum(b_i X_i) + \epsilon$  where  $Y$  is a continuous dependent variable and independent variables  $X_i$  are *usually* continuous (but can also be binary, e.g. when the linear model is used in a t-test) or other discrete domains.  $\epsilon$  is a term for the variance that is not explained by the model and is usually just called "error". Individual dependent values denoted by  $Y_j$  can be solved by modifying the equation a little:  $Y_j = b_0 + \sum(b_i X_{ij}) + \epsilon_j$

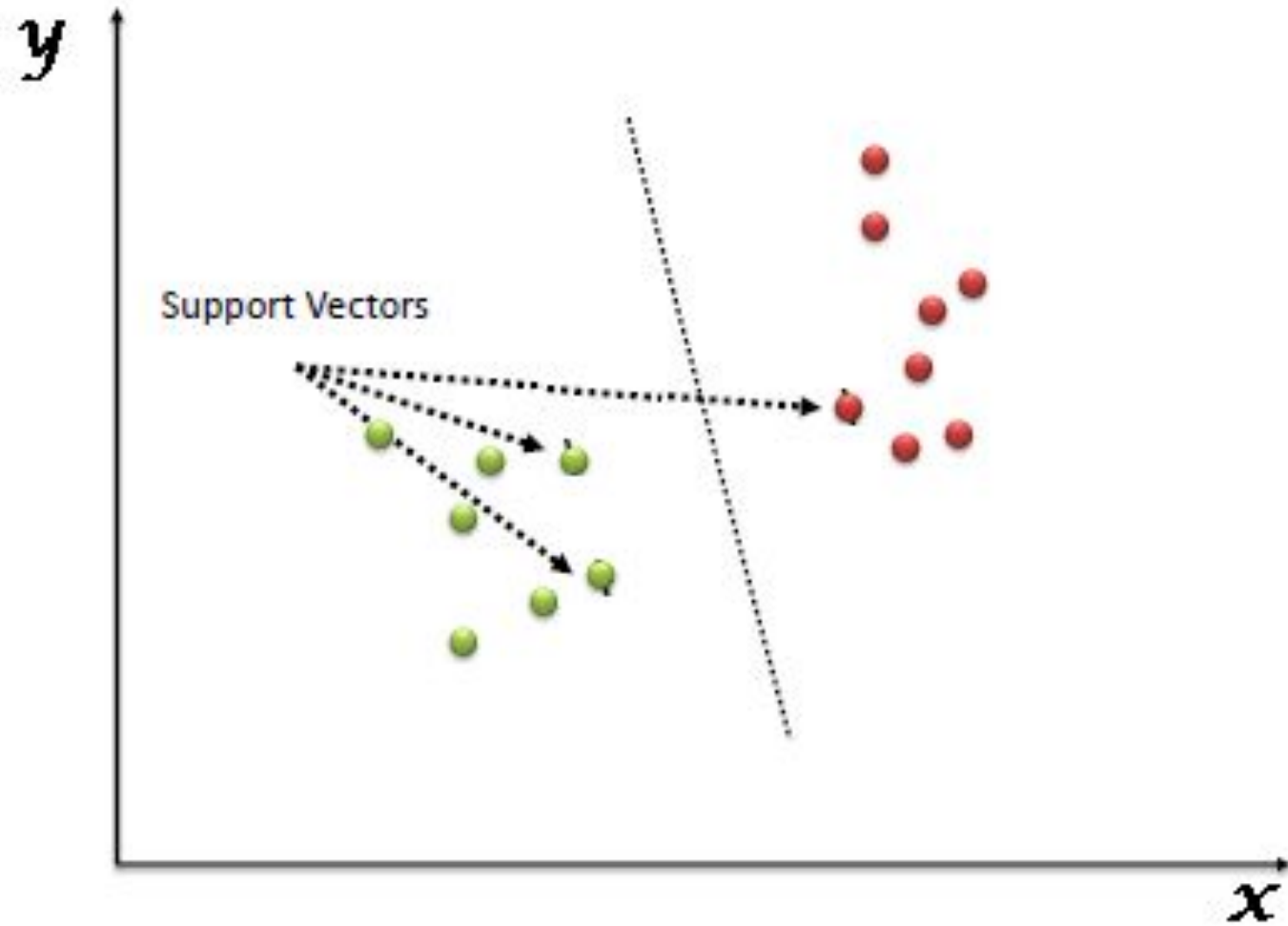
Logistic regression is another generalized linear model (GLM) procedure using the same basic formula, but instead of the continuous  $Y$ , it is regressing for the probability of a categorical outcome. In simplest form, this means that we're considering just one outcome variable and two states of that variable- either 0 or 1.

The equation for the probability of  $Y = 1$  looks like this:

$$P(Y = 1) = \frac{1}{1 + e^{-(b_0 + \sum(b_i X_i))}}$$

# Support Vector Machine

- In this algorithm, we plot each data item as a point in  $n$ -dimensional space (where  $n$  is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well



# Decision Tree Classifier

- Each node of the decision tree is associated with a region in the input space, and internal nodes break that region into one sub-region for each child of the node
- Space is thus sub-divided into non-overlapping regions, with a one-to-one correspondence between leaf nodes and input regions. Each leaf node usually maps every point in its input region to the same output

# Unsupervised Learning Algorithm

- A classic unsupervised learning task is to find the “best” representation of the data.
- We are looking for a representation that preserves as much information about  $x$  as possible while obeying some penalty or constraint aimed at keeping the representation simpler or more accessible than  $x$  itself
- Low-dimensional representations attempt to compress as much information about  $x$  as possible in a smaller representation

# Principal Component Analysis

- A representation that has lower dimensionality than the original input.
- It also learns a representation whose elements have no linear correlation with each other
- This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

# K-means clustering

- The k-means clustering algorithm divides the training set into  $k$  different clusters of examples that are near each other.
- The k-means algorithm works by initializing  $k$  different centroids  $\{\mu(1), \dots, \mu(k)\}$  to different values, then alternating between two different steps until convergence. In one step, each training example is assigned to cluster  $i$ , where  $i$  is the index of the nearest centroid  $\mu(i)$ .
- In the other step, each centroid  $\mu(i)$  is updated to the mean of all training examples  $x(j)$  assigned to cluster  $i$ .

# Stochastic Gradient Descent

- Nearly all of deep learning is powered by one very important algorithm: stochastic gradient descent or SGD
- The cost function used by a machine learning algorithm often decomposes as a sum over training examples of some per-example loss function
- It is the main way to train large linear models on very large datasets.



# Building a machine learning algorithm

- Nearly all deep learning algorithms can be described as particular instances of a fairly simple recipe:
  - combine a specification of a dataset,
  - a cost function,
  - An optimization procedure
  - and a model.

# Challenges that motivate deep learning

- The simple machine learning algorithms described in this lecture work very well on a wide variety of important problems.
- However, they have not succeeded in solving the central problems in AI, such as recognizing speech or recognizing objects.
- The challenge of generalizing to new examples becomes exponentially more difficult when working with high-dimensional data
- The mechanisms used to achieve generalization in traditional machine learning are insufficient to learn complicated functions in high-dimensional spaces.

# Curse of Dimensionality

- Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high.

# Preprocessing data using scikit-learn

```
import numpy as np
from sklearn import preprocessing
Input_data = np.array(
    [
        [2.1, -1.9, 5.5],
        [-1.5, 2.4, 3.5],
        [0.5, -7.9, 5.6],
        [5.9, 2.3, -5.8]
    ]
)
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0,1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print ("\nMin max scaled data:\n", data_scaled_minmax)
```

# Estimator API

- It is one of the main APIs implemented by Scikit-learn.
- It provides a consistent interface for a wide range of ML applications that's why all machine learning algorithms in Scikit-Learn are implemented via Estimator API.
- The object that learns from the data (fitting the data) is an estimator.
- It can be used with any of the algorithms like classification, regression, clustering or even with a transformer, that extracts useful features from raw data.
- `estimator.fit(data)`

# SVM

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.svm import SVC
X, y = load_iris(return_X_y = True)
clf = SVC()
clf.set_params(kernel = 'linear').fit(X, y)
clf.predict(X[:5])
clf.set_params(kernel = 'rbf', gamma = 'scale').fit(X, y)
clf.predict(X[:5])
```

# Decision trees

```
from sklearn import tree
from sklearn.model_selection import train_test_split
X=[[165,19],[175,32],[136,35],[174,65],[141,28],[176,15]
,[131,32],[166,6],[128,32],[179,10],[136,34],[186,2],[12
6,25],[176,28],[112,38],[169,9],[171,36],[116,25],[196,2
5], [196,38], [126,40], [197,20], [150,25], [140,32],[136,35]]
Y=['Man','Woman','Woman','Man','Woman','Man','Woman','Ma
n','Woman','Man','Woman','Man','Woman','Woman','Woman','
Man','Woman','Woman','Man', 'Woman', 'Woman', 'Man', 'Man', 'Woman', 'Woman']
data_feature_names = ['height','length of hair']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 1)
DTclf = tree.DecisionTreeClassifier()
DTclf = clf.fit(X,Y)
prediction = DTclf.predict([[135,29]])
print(prediction)
```

# Decision trees

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
digits = load_digits()
digits.data.shape
kmeans = KMeans(n_clusters = 10, random_state = 0)
clusters = kmeans.fit_predict(digits.data)
kmeans.cluster_centers_.shape
```