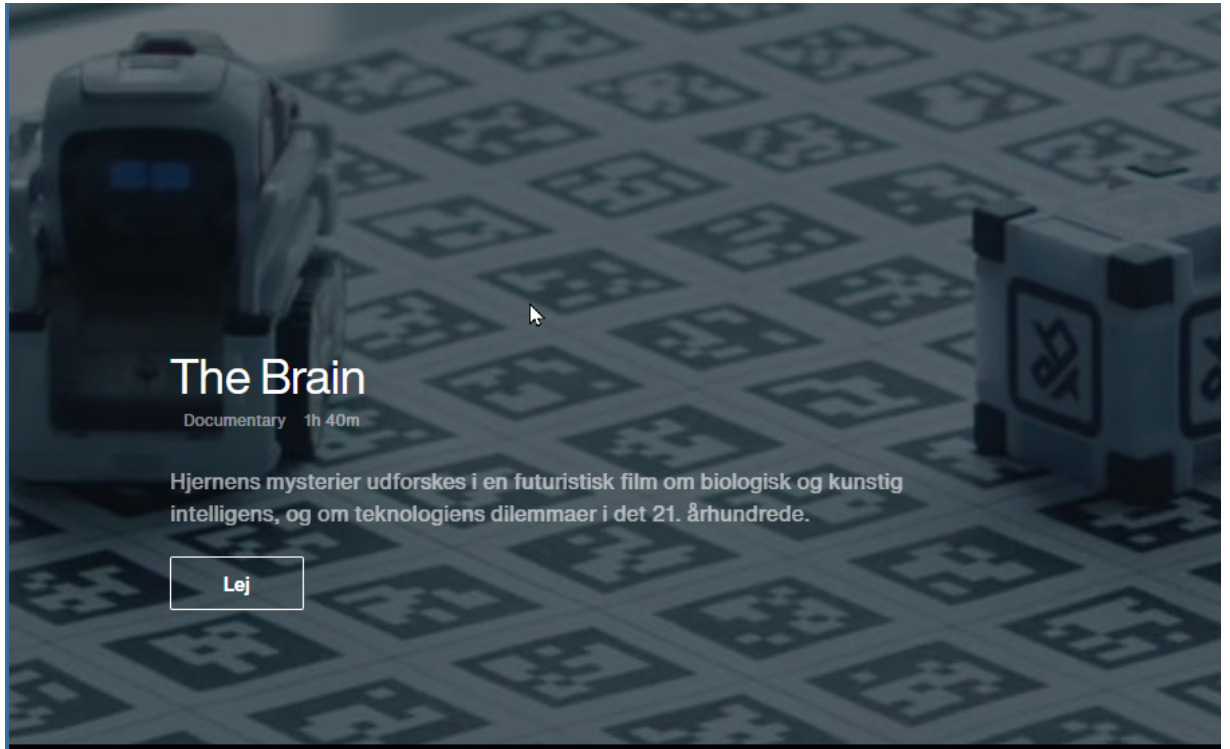
Machine Learning 25-04-2022

Agenda

- Unsupervised Learning
- Unsupervised Learning - Example - Iris dataset
  - Clustering - K-means
  - Hierarchical clustering
  - t-SNE clustering
  - DBSCAN clustering

The Brain



Hjernens mysterier udforskes i en futuristisk film om biologisk og kunstig intelligens, og om teknologiens dilemmaer i det 21. århundrede.

Kapløbet mellem menneskelig og kunstig intelligens er i gang. Mens forskere langsomt kortlægger den menneskelige hjernes mysterier, sker der dramatiske fremskridt inden for AI. Vi følger den tavse krig, der raser mellem feltets førende forskere og banebrydende laboratorier. Mens nogle af dem dedikerer deres liv til forståelsen af den menneskelige hjerne, har andre til formål at bygge intelligente maskiner, der overhaler den i mental båndbredde. Instruktør Jean-Stéphane Brons nye film er en futuristisk rejse fuld af etiske dilemmaer og eksistentielle grundspørgsmål, der udfordrer vores mest grundlæggende viden om verden - og om os selv.

The mysteries of the brain are explored in a futuristic film about biological and artificial intelligence and the dilemmas of technology in the 21st century.

The race between human and artificial intelligence is underway. While scientists are slowly mapping the mysteries of the human brain, dramatic advances are happening within the field of AI. We follow the silent war raging between the leading scientists and pioneering laboratories of the field. While some of them are dedicating their lives to the understanding of the human brain, others aim to build intelligent machines that can surpass it in mental bandwidth. The director Jean-Stéphane Bron's latest film is a futuristic journey full of ethical dilemmas and existential questions, which challenge our most basic understanding of the world – and ourselves.

https://www.doxonline.dk/film/the-brain

Unsupervised Learning

Unsupervised Learning

Unsupervised learning is a class of machine learning (ML) techniques used to find patterns in data.

The data given to unsupervised algorithms is not labelled, which means only the input variables (x) are given with no corresponding output variables.

In unsupervised learning, the algorithms are left to discover interesting structures in the data on their own.

Supervised vs. Unsupervised learning

In **supervised** learning, the system tries to **learn from the previous** examples given.

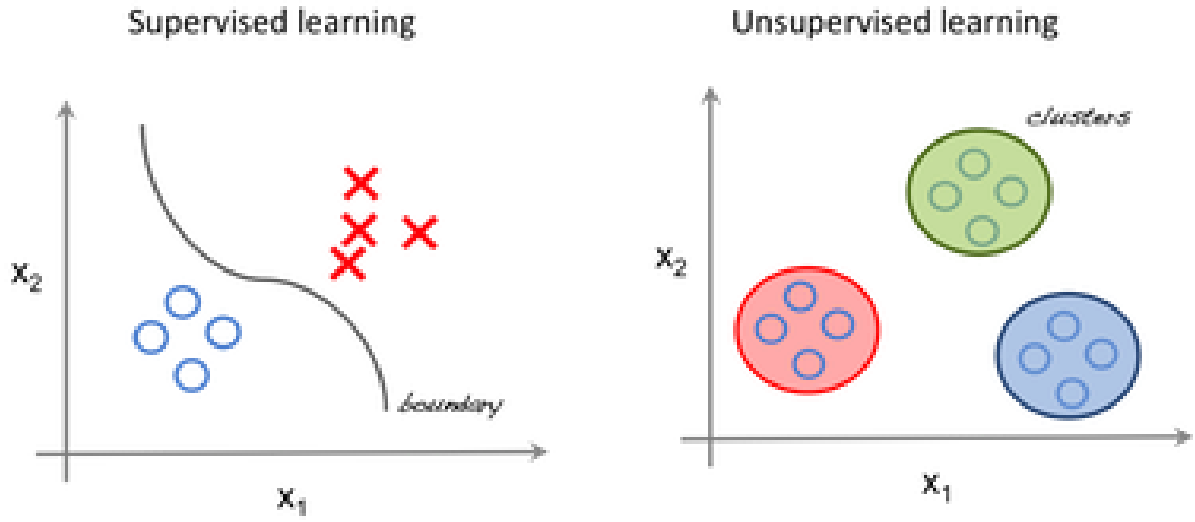
In **unsupervised** learning, the system **attempts to find the patterns** directly from the example given.

*If the dataset is **labeled** it is a **supervised** problem, and if the dataset is **unlabelled** then it is an **unsupervised** problem.*

Supervised vs. Unsupervised learning

Left image is an example of **supervised learning** (*regression techniques used to find the best fit line between the features*).

In **unsupervised learning** the inputs are segregated based on features and the prediction is based on which cluster it belonged to.



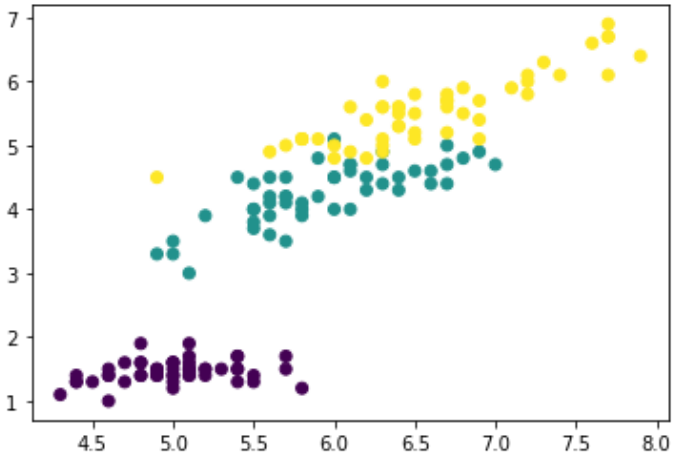
Unsupervised Learning - Examples

- Customer segmentation, or understanding different customer groups around which to build marketing or other business strategies
- Recommender systems, which involve grouping together users with similar viewing patterns in order to recommend similar content.
- Anomaly detection, including fraud detection or detecting defective mechanical parts - *predictive maintenance*

Example - Iris dataset

In this example, we'll use the Iris dataset to make predictions.

The dataset contains a set of 150 records under four attributes — **petal length**, **petal width**, **sepal length**, **sepal width**, and **three iris classes**: **setosa**, **virginica** and **versicolor**.



We'll feed the four features of our flower to the unsupervised algorithm and it will predict which class the iris belongs to.

We use the scikit-learn library in Python to load the Iris dataset and matplotlib for data visualization.

```
# Importing Modules
from sklearn import datasets
import matplotlib.pyplot as plt

# Loading dataset
iris_df = datasets.load_iris()

# Available methods on dataset
print(dir(iris_df))

# Features
print(iris_df.feature_names)

# Targets
print(iris_df.target)

# Target Names
print(iris_df.target_names)
label = {0: 'red', 1: 'blue', 2: 'green'}

# Dataset Slicing
x_axis = iris_df.data[:, 0]  # Sepal Length
y_axis = iris_df.data[:, 2]  # Sepal Width

# Plotting
plt.scatter(x_axis, y_axis, c=iris_df.target)
plt.show()
```

Clustering - K-means

In clustering, the data is divided into several groups with similar traits.

K-means clustering is an iterative clustering algorithm that aims to find local maxima in each iteration.

In the iris example, we know there are three classes involved, so we program the algorithm to group the data into three classes by passing the parameter **n_clusters** into our k-means model.

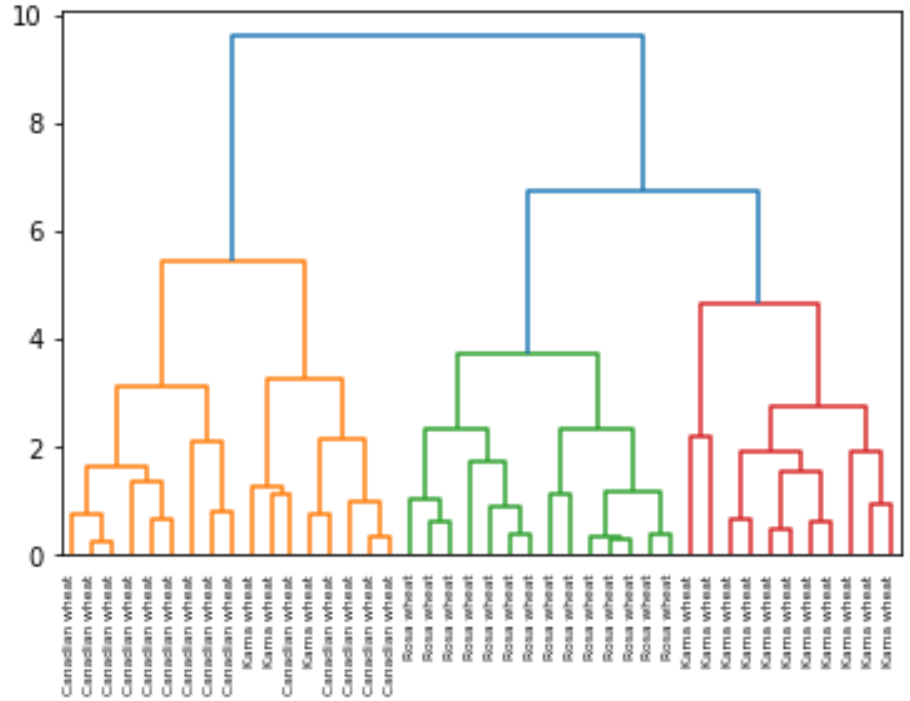
Randomly, three points (inputs) are assigned into three clusters.

```
# Importing Modules
from sklearn import datasets
from sklearn.cluster import KMeans

# Loading dataset
iris_df = datasets.load_iris()

# Declaring Model
model = KMeans(n_clusters=3)

# Fitting Model
model.fit(iris_df.data)

# Predicitng a single input
predicted_label = model.predict([[7.2, 3.5, 0.8, 1.6]])

# Prediction on the entire data
all_predictions = model.predict(iris_df.data)

# Printing Predictions
print(predicted_label)
print(all_predictions)
```

Hierarchical clustering

hierarchical clustering is an algorithm that builds a hierarchy of clusters. This algorithm begins with all the data assigned to a cluster, then the two closest clusters are joined into the same cluster.

The algorithm ends when only a single cluster is left.



```
# Importing Modules
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import pandas as pd

# Reading the DataFrame
seeds_df = pd.read_csv('seeds-less-rows.csv')

# Remove the grain species from the DataFrame, save for later
varieties = list(seeds_df.pop('grain_variety'))

# Extract the measurements as a NumPy array
samples = seeds_df.values

"""
    Perform hierarchical clustering on samples using the
    linkage() function with the method='complete' keyword argument.
    Assign the result to mergings.
"""
mergings = linkage(samples, method='complete')

"""
Plot a dendrogram using the dendrogram() function on mergings,
specifying the keyword arguments labels=varieties, leaf_rotation=90,
and leaf_font_size=6.
"""
dendrogram(mergings,
        labels=varieties,
        leaf_rotation=90,
        leaf_font_size=6,
        )

plt.show()
```

K-means vs. Hierarchical clustering

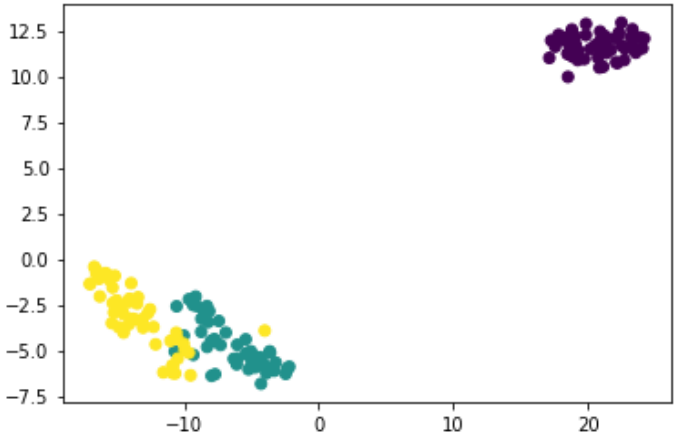
- Hierarchical clustering can’t handle big data very well- k-means clustering can.
- K-means clustering starts with an arbitrary choice of clusters, and the results generated by running the algorithm multiple times might differ. Results are reproducible in hierarchical clustering.
- K-means is found to work well when the shape of the clusters is hyperspherical (like a circle in 2D or a sphere in 3D).
- K-means doesn't allow noisy data, while hierarchical clustering can directly use the noisy dataset for clustering.

t-SNE clustering

A unsupervised learning methods for visualization is **t-distributed stochastic neighbor embedding**, or **t-SNE**.

It maps high-dimensional space into a two or three-dimensional space which can then be visualized.

Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.



https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

t-SNE - Learning rate

The learning rate for t-SNE is usually in the range **[10.0, 1000.0]**.

If the learning rate is **too high**, the data may look like a ‘ball’ with any point approximately equidistant from its nearest neighbors.

If the learning rate is **too low**, most points may look compressed in a dense cloud with few outliers.

If the cost function gets stuck in a bad local minimum increasing the learning rate may help.

```
# Importing Modules
from sklearn import datasets
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Loading dataset
iris_df = datasets.load_iris()

# Defining Model
model = TSNE(learning_rate=100)

# Fitting Model
transformed = model.fit_transform(iris_df.data)

# Plotting 2d t-Sne
x_axis = transformed[:, 0]
y_axis = transformed[:, 1]

plt.scatter(x_axis, y_axis, c=iris_df.target)
plt.show()
```

DBSCAN clustering

Density-based spatial clustering of applications with noise, or DBSCAN, is a popular clustering algorithm used as a replacement for k-means in predictive analytics.

The scikit-learn implementation provides a default for the eps and min_samples parameters, but you’re generally expected to tune those. The eps parameter is the maximum distance between two data points to be considered in the same neighborhood. The min_samples parameter is the minimum amount of data points in a neighborhood to be considered a cluster.

```
# Importing Modules
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA

# Load Dataset
iris = load_iris()

# Declaring Model
dbscan = DBSCAN()

# Fitting
dbscan.fit(iris.data)

# Transoring Using PCA
pca = PCA(n_components=2).fit(iris.data)
pca_2d = pca.transform(iris.data)

# Plot based on Class
for i in range(0, pca_2d.shape[0]):
    if dbscan.labels_[i] == 0:
        c1 = plt.scatter(pca_2d[i, 0], pca_2d[i, 1], c='r', marker='+')
    elif dbscan.labels_[i] == 1:
        c2 = plt.scatter(pca_2d[i, 0], pca_2d[i, 1], c='g', marker='o')
    elif dbscan.labels_[i] == -1:
        c3 = plt.scatter(pca_2d[i, 0], pca_2d[i, 1], c='b', marker='*')

plt.legend([c1, c2, c3], ['Cluster 1', 'Cluster 2', 'Noise'])
plt.title('DBSCAN finds 2 clusters and Noise')
plt.show()
```

Reinforcement Learning

Reinforcement Learning

It is the training of machine learning models to make a sequence of decisions.

The *machine* learns to achieve a goal in an uncertain, potentially complex environment.

The *machine* employs trial and error to come up with a solution to the problem.

Reinforcement Learning - Example

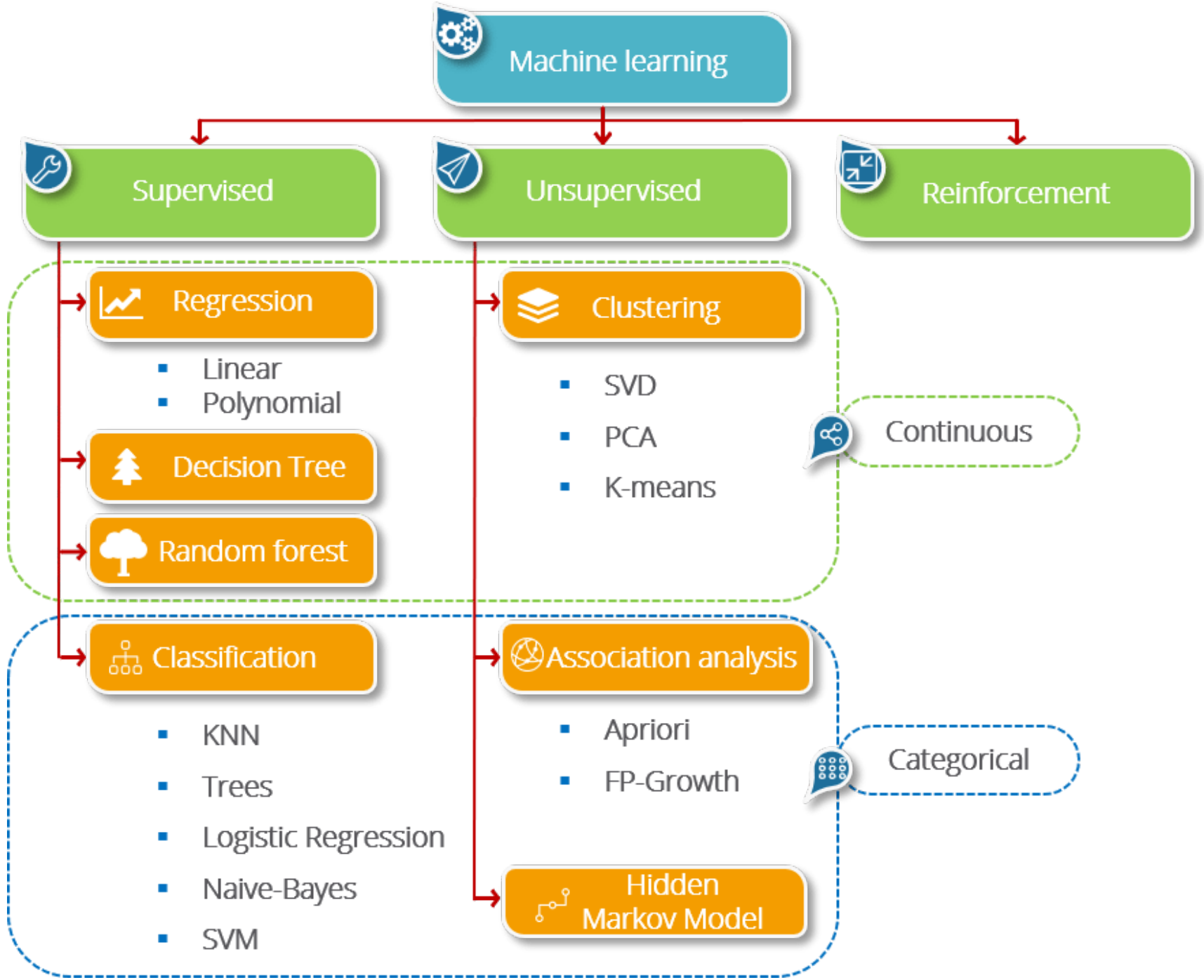
Facebook has developed an open-source reinforcement learning platform—Horizon. The platform uses reinforcement learning to optimize large-scale production systems.

Facebook has used Horizon internally

- To personalize suggestions
- Deliver more meaningful notifications to users
- Optimize video streaming quality

Read more

- https://engineering.fb.com/ml-applications/horizon
- https://research.fb.com/publications/horizon-facebooks-open-source-applied-reinforcement-learning-platform


Machine learning
Supervised
Unsupervised
Reinforcement
Regression
Linear
Polynomial
Decision Tree
Random forest
Clustering
SVD
PCA
K-means
Continuous
Classification
KNN
Trees
Logistic Regression
Naive-Bayes
SVM
Association analysis
Apriori
FP-Growth
Hidden Markov Model
Categorical