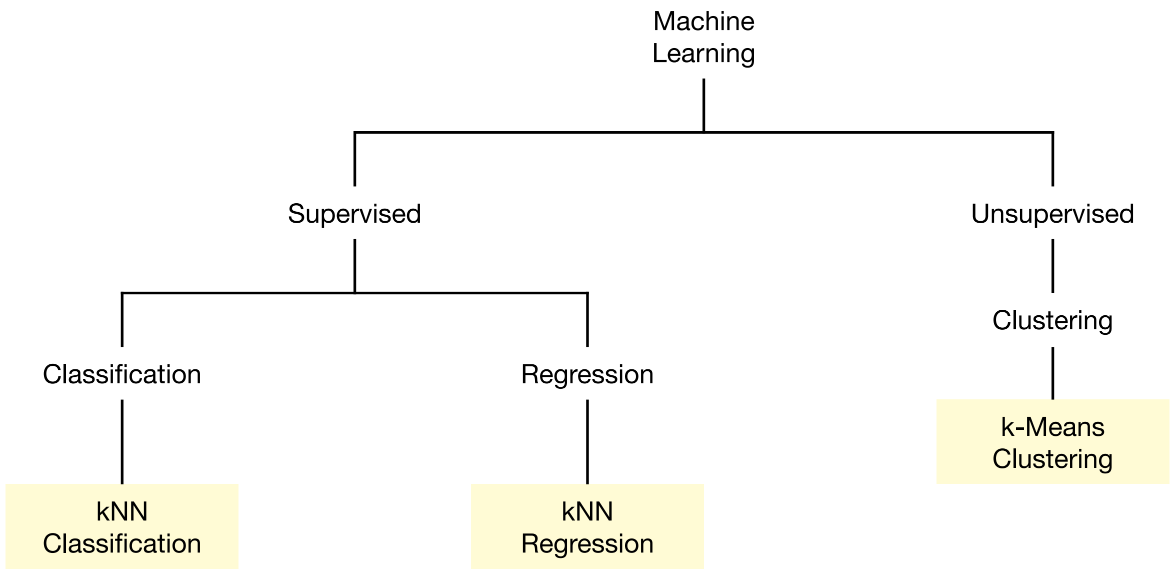
A Taste of Machine Learning



For this set of activities, we'll be performing several machine-learning tasks on a (famous) data set containing sepal and petal measurements and species information about a sample of iris flowers. For more information about this data set, see the *Iris Flower Data Set* Wikipedia article.

Iris setosa

Iris versicolor

Iris virginica

Images taken from the Wikipedia Iris Flower Data Set article.
https://en.wikipedia.org/wiki/Iris_flower_data_set

Supervised Learning → Classification→ kNN Classification

Predicting the species of a flower

In this task, we are going use an algorithm called **kNN (k nearest neighbors) classification** to predict the species of an iris flower based on its petal and sepal measurements.

1. Design a function that takes in (1) a new flower, (2) a data set of flowers and (3) an integer k. It then finds the k neighbors in the data set that are nearest — using Euclidean distance between the sepal and petal measurements — to the new flower. After that, it returns *the most common species* among these neighbors as the prediction of the species of the new flower.

However, there is a problem. In part 1, we need to specify k. If we choose a value that is too small, then we risk making predictions that are incorrect because of *over-fitting* our data. However, if we choose a value that is too large, then we risk making predictions that are incorrect because of *over-generalizing* from our data. So how can we choose an appropriate value of k? In part 2, we'll use a method called **leave-one-out cross validation**.

2. Design a function that takes in (1) a value of k and (2) a data set of flowers. For each flower in the data set, the function predicts the species of the flower removed (for example, by using the function from part 1) using the k nearest neighbors in the flowers that are still in the data set. The function then checks *whether the prediction is correct or not* by looking at the actual species. After going through all the flowers, the function returns the proportion of the predictions that were incorrect; in other words, the function returns the leave-one-out cross validation **error rate**.

3. Plot the error rate against k for a range of values, say k from 1 to 50. Use this plot to choose an appropriate value of k based on leave-one-out-cross validation.

Supervised Learning → Regression→ kNN Regression

Predicting the petal length of a flower

In this task, we are going use an algorithm called **kNN (k nearest neighbors) regression** to predict the length of an iris petal based on its sepal measurements and species.

1. Design a function that takes in (1) a new flower, (2) a data set of flowers and (3) an integer k. It then finds the k neighbors in the data set that are nearest — using Euclidean distance on sepal width and length and a species indicator (whether flowers are of different species) — to the new flower. After that, it returns *the average petal length* among these neighbors as the prediction of the petal length of the new flower.

As in the previous activity, there is a problem. In part 1, we need to specify k. If we choose a value that is too small, then we risk making predictions that are incorrect because of *over-fitting* our data. However, if we choose a value that is too large, then we risk making predictions that are incorrect because of *over-generalizing* from our data. So how can we choose an appropriate value of k? As in the previous activity, in part 2, we'll use a method called **leave-one-out cross validation**.

2. Design a function that takes in (1) a value of k and (2) a data set of flowers. For each flower in the data set, the function predicts the petal length of the flower removed (for example, by using the function from part 1) using the k nearest neighbors in the flowers that are still in the data set. The function then calculates the *squared deviation of the prediction from the actual petal length*. After going through all the flowers, the function returns *the square root of the average squared deviation*; in other words, the function returns the leave-one-out cross validation **root-mean-squared error (RMSE)**.

3. Plot the RMSE against k for a range of values, say k from 1 to 50. Use this plot to choose an appropriate value of k based on leave-one-out-cross validation.

Unsupervised Learning → Clustering→ k-Means Clustering

Creating clusters of similar flowers

In this task, we are going use an algorithm called **k-means clustering (KMC)** to assign the iris flowers in out data set to clusters of similar flowers. Because we want to visualize the clusters at the end of this activity, we'll restrict out data to only two dimensions, namely the *sepal width* and the *petal width*.

KMC is an iterative algorithm. The way it works is:

A. **Randomly place k clusters in space.**
   This can be done in different ways. One idea is to select k random flowers in the data and put the cluster centers directly on those flowers. Another idea is to look at the range of the data, and then place the cluster centers randomly in that range. (Other ideas that make the centers more likely to be spread out also exist.)

B. **Assign each point in the data to the nearest cluster.**
   In other words, go through each point in the data, find the cluster that it is nearest to the point, and then mark that point as belonging to that cluster.

C. **Shift each cluster center.**
   For each cluster, identify all the points that belong to it and then shift the cluster to the mean position of its points.

D. **Repeat until the clusters settle down.**
   Practically, we can set a minimum threshold distance for cluster center movement: if no clusters move more than the threshold distance, we break out of the algorithm; otherwise we continue iteration from step B.

1. Design a function that takes in (1) an integer k, which can be from 1 to 5, and (2) a set of flowers, and then performs KMC until the flowers have been placed in stable clusters. At each step in the algorithm, the function should plot the the clusters as large **x** marks with different colors together with the data points as small **o** marks colored according to which cluster they belong to. (Ideally, the function should save the plots as image files so that we can view the evolution of the clusters as an animation afterwards.)

2. Using k = 3, compare the clusters formed to the species of each data point. (For example, plot the data using point color to identify the cluster a point belongs to, and point shape to identify the species it represents.)