Andrzej Cieślik **12.07.2023**

1.Describe a machine learning basicals (single work).

2. Supervised Machine learning with IRIS dataset and SUM (single work).

3. IRIS dataset , try to clusterite (group work).

1. Describe a machine learning basicals (single work).

1.1. Machine learning.

Machine Learning (ML) is a field of Artificial Intelligence (AI) that focuses on creating algorithms and models without the need for direct programming.

Traditionally, computer programs were created in a deterministic manner, where the programmer would establish rules and instructions for the computer to follow. In the case of machine learning, the computer learns from data and examples to independently recognize patterns and make decisions based on those patterns. This means that instead of directly programming explicit rules, the programmer creates algorithms and models that learn from the available data.

In machine learning, there are two main types: supervised learning and unsupervised learning.

1.2. Supervised learning.

Supervised learning involves providing the computer with input data along with desired labels or answers. Based on this data, the computer learns to create rules and patterns that allow it to predict labels for new input data.

1.3. Unsupervised learning.

Unsupervised learning does not require labeled data. The computer analyzes the available data and independently discovers patterns by grouping similar data together. An example of this is data clustering, where the computer groups data points into clusters based on their similarities.

2. Supervised Machine learning with IRIS dataset and SUM (single work).

2.1. IRIS data set.

The Iris dataset is a famous and widely used dataset in machine learning and statistics for classification tasks. It was first introduced by the British statistician and biologist Ronald Fisher in 1936. The dataset consists of 150 samples of iris flowers, each with four features or attributes: sepal length, sepal width, petal length, and petal width.

The Iris dataset contains 4 measurements of its petals (width and length) and the corresponding class it belongs to. Here are a few selected records as examples:

sepal length, sepal width, petal length, petal width, class

5.1, 3.5, 1.4, 0.2, Iris-setosa

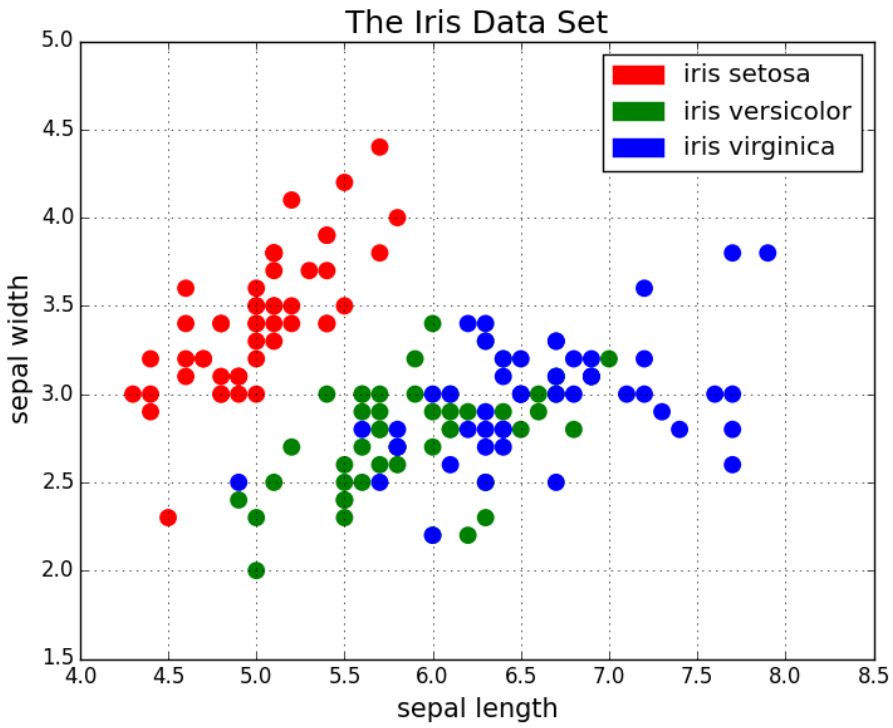
4.9, 3.0, 1.4, 0.2, Iris-setosa

4.7, 3.2, 1.3, 0.2, Iris-setosa

7.0, 3.2, 4.7, 1.4, Iris-versicolor

6.4, 3.2, 4.5, 1.5, Iris-versicolor

The Iris dataset is commonly used to demonstrate various classification algorithms, especially for solving problems involving multiple classes. The goal is to assign iris flowers to one of three species: setosa, versicolor, and virginica, based on their measurements. Example visualization of the dataset with class divisions:



The Iris dataset is widely used for educational purposes, algorithm demonstrations, and benchmarking in the field of machine learning.

One of the simplest examples of using the Iris dataset is the classification of iris species based on their morphological features. The Iris dataset contains information about four features: sepal length, sepal width, petal length, and petal width for three different iris species: Setosa, Versicolor, and Virginica. To do this, we can use a machine learning algorithm such as Support Vector Machine (SVM) classifier or k-Nearest Neighbors algorithm.

2.2 Support Vector Machine example.

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

iris = datasets.load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

classifier = SVC()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

for pred, true in zip(y_pred, y_test):
        print(f"Predicted: {iris.target_names[pred]}, True: {iris.target_names[true]}")
```

2.3. Training parameters.

2.3.1. Test size.

Test_size is the parameter specifies the proportion of the dataset that should be allocated for testing. In the example, test_size=0.2 means that 20% of the data will be used for testing, while the remaining 80% will be used for training. Adjusting this value allows you to control the size of the test set relative to the training set.

2.3.2. Random state.

This parameter is used to set the random seed for reproducibility. The random seed determines the random permutation applied to the data before splitting. By setting a specific value for random_state (e.g., random_state=42), you ensure that the data is split in the same way every time you run the code. This is useful for obtaining consistent results and making your experiments reproducible.

2.2. Wine dataset and K-nearest neighbours.

The wine dataset contains information about various attributes of different wines, such as their chemical composition. The dataset is commonly used to classify wines into different classes based on their features.

K-Nearest Neighbors (K-NN) is a simple yet powerful machine learning algorithm used for both classification and regression tasks. It is a non-parametric method, meaning it does not make any assumptions about the underlying data distribution.

In K-NN, the "K" refers to the number of nearest neighbors used for decision-making. Given a new, unlabeled data point, the algorithm finds the K closest labeled data points (neighbors) in the training dataset based on a distance metric (e.g., Euclidean distance). The majority class or the average value of the K neighbors is then assigned to the new data point for classification or regression, respectively.

In the example the value of the K was checked from 1 to the sqrt of number of samples.

```
import numpy as np
from sklearn.datasets import load_wine
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import math

data = load_wine()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

num_samples = X_train.shape[0]
k_values = range(1, int(math.sqrt(num_samples)) + 1)

best_accuracy = 0
best_k = 0

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    y_pred = knn.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy for k =", k, ":", accuracy)

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k

print("Best K:", best_k)
print("Best Accuracy:", best_accuracy)

new_wine = np.random.rand(1, 13)


best_knn = KNeighborsClassifier(n_neighbors=best_k)
best_knn.fit(X, y)
predicted_class = best_knn.predict(new_wine)

class_labels = data.target_names
predicted_class_name = class_labels[predicted_class[0]]

print("Predicted class for the new wine (using best K):", predicted_class_name)
```

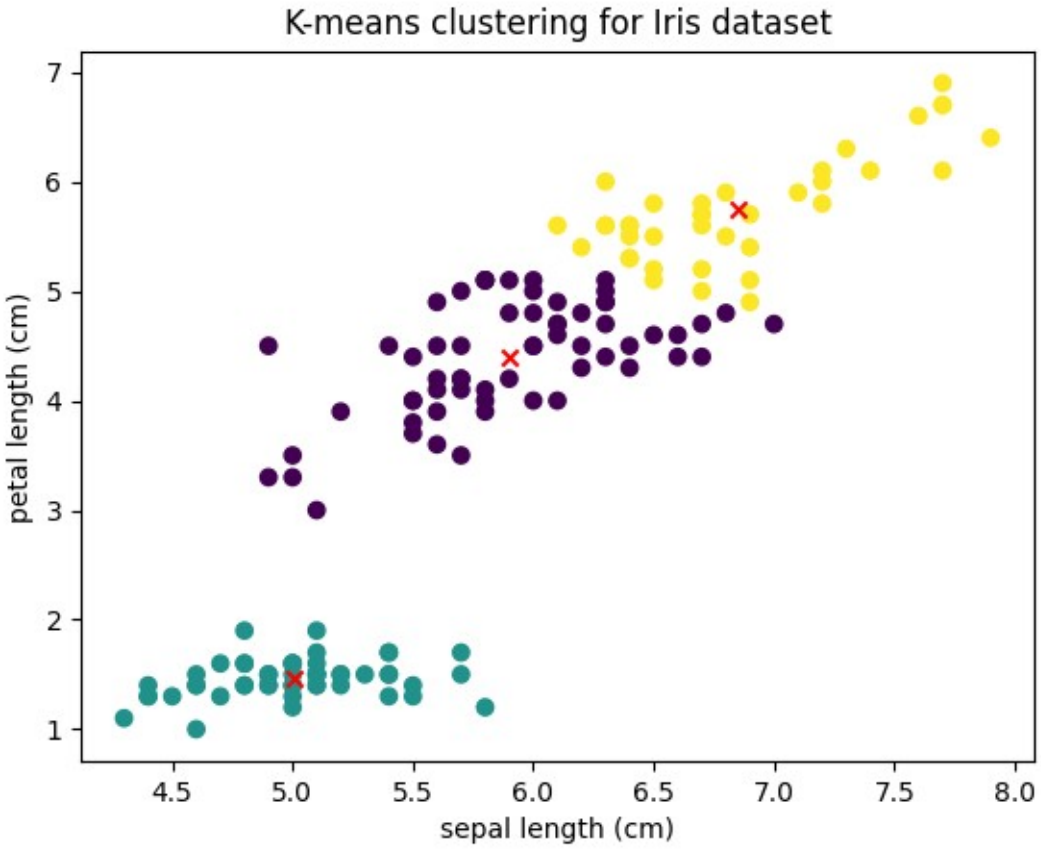
3. Unsupervised machine learning. IRIS dataset , try to clusterite (group work).

3.1. Clustering.

Clustering is an unsupervised machine learning technique that aims to group data points together based on their similarities, without the need for pre-existing labels or categories. It involves finding patterns and structures within a dataset by identifying clusters, which are groups of similar data points. This unsupervised approach allows for the exploration and discovery of inherent organization and relationships within the data, providing valuable insights and understanding without the need for explicit guidance or supervision.

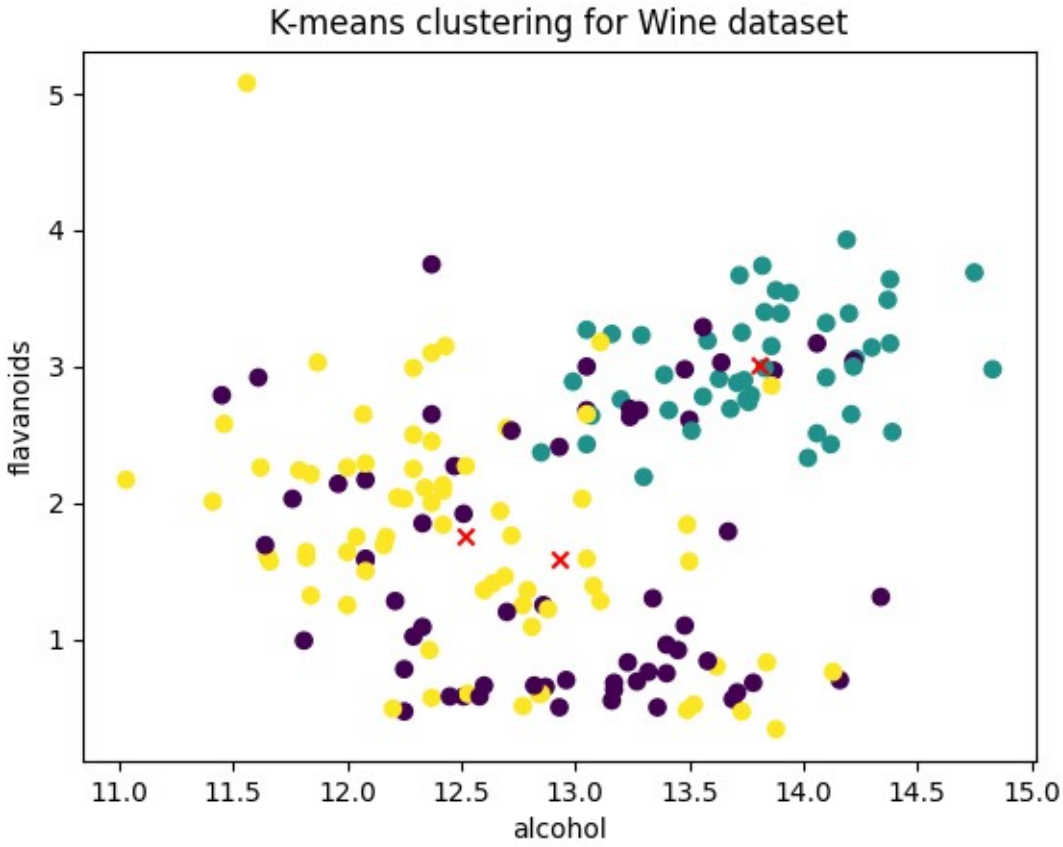
3.2. IRIS clustering.

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

iris_data = load_iris()
X = iris_data.data

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)

kmeans.fit(X)

labels = kmeans.predict(X)

feature1_index = 0
feature2_index = 2

plt.scatter(X[:, feature1_index], X[:, feature2_index], c=labels)
plt.scatter(kmeans.cluster_centers_[:, feature1_index], kmeans.cluster_centers_[:, feature2_index],
marker='x', color='red')
plt.xlabel(iris_data.feature_names[feature1_index])
plt.ylabel(iris_data.feature_names[feature2_index])
plt.title("K-means clustering for Iris dataset")

plt.show()
```



3.3. Wine clustering.

```
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

wine_data = load_wine()
X = wine_data.data

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)  # lub n_init='auto'

kmeans.fit(X)

labels = kmeans.predict(X)

feature1_index = 0
feature2_index = 6

plt.scatter(X[:, feature1_index], X[:, feature2_index], c=labels)
plt.scatter(kmeans.cluster_centers_[:, feature1_index], kmeans.cluster_centers_[:, feature2_index],
marker='x', color='red')
plt.xlabel(wine_data.feature_names[feature1_index])
plt.ylabel(wine_data.feature_names[feature2_index])
plt.title("K-means clustering for Wine dataset")

plt.show()
```


K-means clustering for Wine dataset
flavanoids
alcohol
5
4
3
2
1
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
15.0