Machine Learning

Lab 1. Introduction to machine learning

The aim of the exercise is to get familiar with and try to understand a complete program containing several basic elements from the field of machine learning.

Iris is a type of plant, encompassing over 300 species. Three of them are the *Iris setosa*, *Iris versicolor*, and *Iris virginica*. Data regarding the shape of flowers of these three species are a classic example of a dataset in the field of **machine learning**. This dataset consists of 150 samples (examples), and each sample comprises four parameters of the flower (its **features**): sepal length, sepal width, petal length, and petal width.

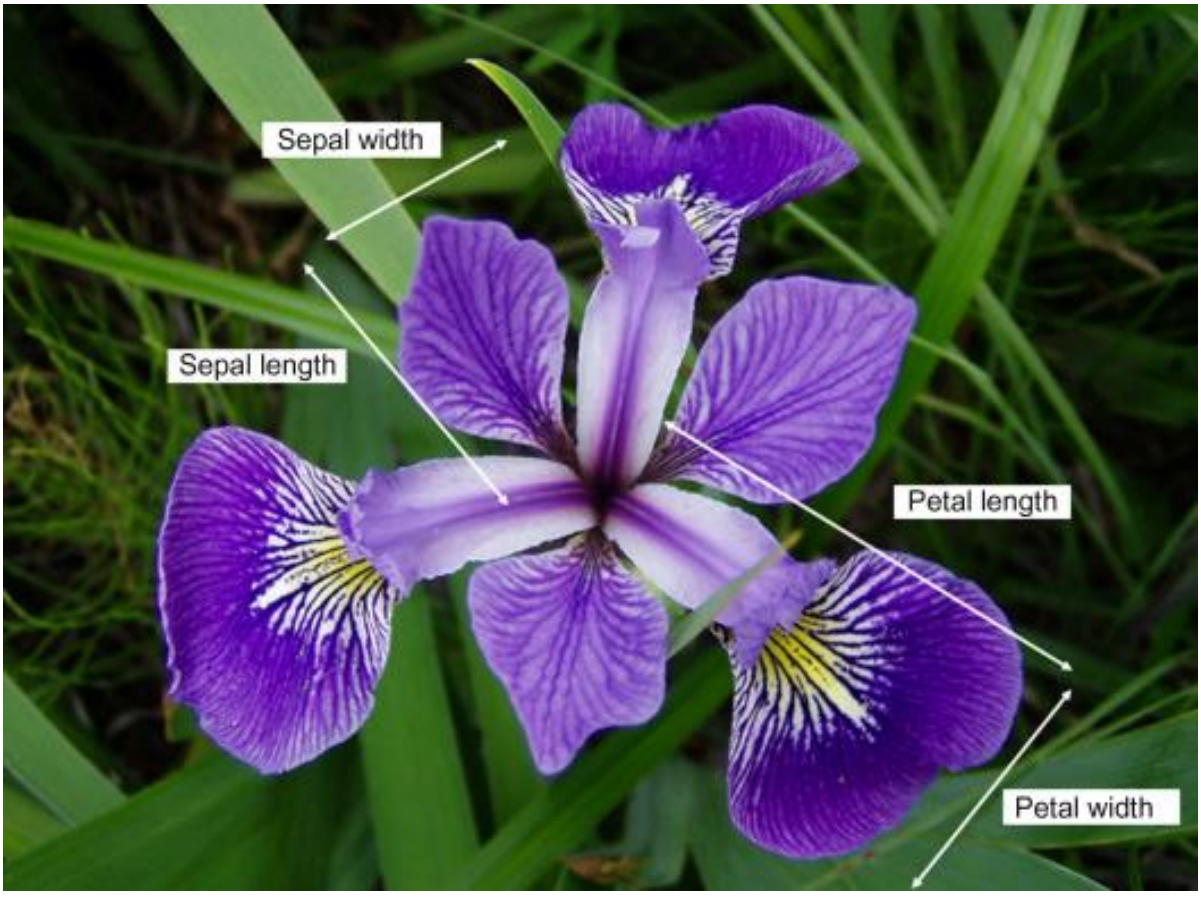

Figure 1. Features of an iris flower (I. versicolor depicted). Vijay Kotu, Bala Deshpande, Data Science, 2nd Ed., 2019, pp. 39-64

These data are available, for example, in the scikit-learn module, which contains many tools used in implementing machine learning methods. In scikit-learn, they are represented as a matrix, whose rows correspond to successive samples (150), and columns to successive features (4), in the order provided above.

To load the "Iris" dataset, the following instructions should be executed:

```
from sklearn import datasets

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

Check the dimensions of the X and y arrays. The X array contains data for 150 samples, while the y array stores the labels (0 - setosa, 1 - versicolor, 2 - virginica) of the three flower species. In machine learning terminology, these distinct groups or types of objects are called **classes**, and the task of recognizing the class of a given object is called **classification**. Classification is one of the basic tasks in machine learning. In the case of the Iris dataset, the aim is to predict the species of a flower based on its features.

Features are a crucial element in machine learning. Determining their values is called **feature extraction**. In the discussed case, feature extraction requires physical action - a measurement made by a human using an appropriate tool. We also need to know which features we want to determine (with some exceptions for certain neural networks) and how to measure their value for a given object. In many cases, feature extraction is a mathematical operation because the objects of our interest are digital (e.g., medical images, bioelectric or acoustic signals). In such a case, we also need to know which feature we want to measure (e.g., the average brightness of an image) and how to do it (add up the brightness of all pixels and divide the result by their number).

We will return to the Iris dataset to examine the features of the Iris flowers for the 150 available samples. The code required to generate the chart is provided. The above two-dimensional chart shows 150 samples. Samples belonging to the same species of flower have the same color. Two out of the four available features were selected here: sepal length and sepal width. Visualization methods for features are one of the research directions in machine learning. Analyze the chart. Do the two visualized features allow for unambiguous classification of the Iris species based on them? Modify the chart so that it includes two different features.

```
import matplotlib.pyplot as plt

x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5

fig, ax = plt.subplots()
scatter = ax.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("sepal length")
plt.ylabel("sepal width")

legend1 = ax.legend(*scatter.legend_elements(),loc="lower right", title="Class label")
ax.add_artist(legend1)

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
```

Analyze the above code and try to understand it. The result looks as follows:

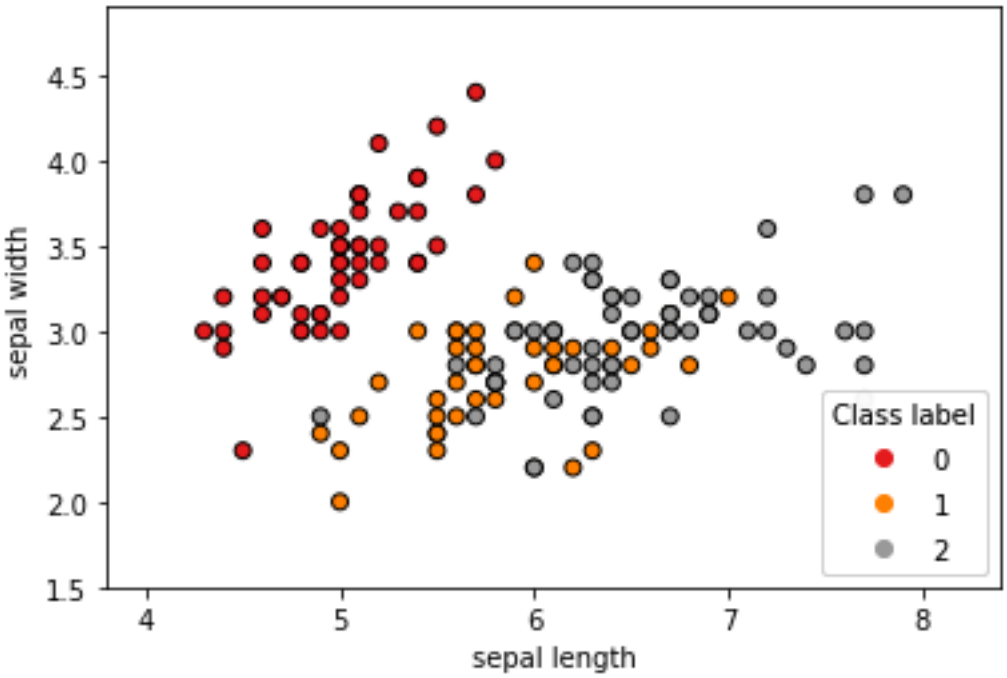

Figure 2

The above two-dimensional plot shows 150 samples. Samples belonging to the same species of flower have the same color. Out of four available features, two were selected here: length and width of sepals. Visualization of features is one of the research directions in machine learning. Analyze the plot. Do the two visualized features allow for unambiguous classification of the iris species based on them? **Modify the code to display other combinations of features (with 4 features, there are 6 different pairs).**

Classification consists in finding a mathematical function that relates the features of an object to its class. In other words, if we give this function (classifier) the values of the features of an object, such as an iris, we will get the class, such as versicolor. Typically, the engineer defines **the model** in advance, i.e., the type of function. This model always has some parameters. For example, if the model is a linear function of one variable, $y = ax + b$, it has two parameters - a and b. Machine learning involves searching for optimal values of these parameters, which give the smallest classification error. Thus, a **learning algorithm** is needed to define the learning process. Also, some measure of error (called **objective function**, **loss function**, **cost function**) is needed, which tells us what error the current parameter values give and whether that error can still be reduced.

Below is an example of iris data classification. First, from the original set of 150 samples with 4 features, representing three species, we will discard one of them (virginica, 50 samples). The task will, therefore, be to assign samples to one of two classes - this is **binary classification**. Secondly, we will perform manual **feature selection** - we will limit ourselves only to the first two (length and width of sepals). After these operations, we will have to deal with a limited set of 100 samples with 2 features. Finally, we will also divide the **data set** into two separate sets: a **training set** and a **test set**, in a 4:1 ratio - the training set will have 80 samples, and the test set 20 samples, with an equal number of examples from both classes in each of them. Samples of each class will be randomly divided between the sets to avoid the possibility of selecting a subset poorly representing the entire class population.

```
import numpy as np

# indices 0-49, 50-99, 100-149 in random order
random0 = np.random.choice(np.arange(0,50),50,replace=False)
random1 = np.random.choice(np.arange(50,100),50,replace=False)
random2 = np.random.choice(np.arange(100,150),50,replace=False)

# take 80% (40 samples) of each class to the training set
X0 = X[random0[:40],:]
X1 = X[random1[:40],:]
X2 = X[random2[:40],:]

# take the corresponding labels
y0 = y[random0[:40]]
y1 = y[random1[:40]]
y2 = y[random2[:40]]

# take 20% (10 samples) of each class to the training set
X0_test = X[random0[40:],:]
X1_test = X[random1[40:],:]
X2_test = X[random2[40:],:]

# take the corresponding labels
y0_test = y[random0[40:]]
y1_test = y[random1[40:]]
y2_test = y[random2[40:]]

# compose the training set - just features 0 and 1 and classes 0 and 1
X01_train = np.concatenate([X0[:,0:2], X1[:,0:2]])
y01_train = np.concatenate([y0, y1])
X01_test = np.concatenate([X0_test[:,0:2], X1_test[:,0:2]])
y01_test = np.concatenate([y0_test, y1_test])
```

The training set, now stored in the variables X01_train and y01_train, contains 80 samples in the end. Classification will be performed using **logistic regression**. The logistic regression model is a generalized linear model. We can describe the linear regression model as follows:

$$h(x) = g(w^T x)$$

where h(x) is the **hypothesis** - the function we are looking for, w is the vector of model parameters, x is the vector of variables, i.e., features, and the function g(z) is the logistic (sigmoidal) function:

$$g(z) = \frac{1}{1 + e^{-z}}$$

```
from sklearn.linear_model import LogisticRegression

# linear binary classification with a logistic regression model
clf = LogisticRegression(random_state=0).fit(X01_train, y01_train)

# read the logistic regression model parameters
b = clf.intercept_[0] #
w1, w2 = clf.coef_.T

# calculate the intercept and gradient of the decision boundary.
c = -b/w2
m = -w1/w2

# plot the data and the classification with the decision boundary.
xmin, xmax = np.min(X01_train,0)[0]-1, np.max(X01_train,0)[0]+1
ymin, ymax = np.min(X01_train,0)[1]-1, np.max(X01_train,0)[1]+1

xd = np.array([xmin, xmax])
yd = m*xd + c

plt.figure()
plt.plot(xd, yd, 'k', lw=1, ls='--')
plt.fill_between(xd, yd, ymin, color='tab:blue', alpha=0.2)
plt.fill_between(xd, yd, ymax, color='tab:orange', alpha=0.2)

plt.scatter(*X01_train.T, c=y01_train, cmap=plt.cm.Set1, edgecolor="k")
plt.scatter(*X01_test.T, c=y01_test, cmap=plt.cm.Set1, edgecolor="b")

plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
plt.ylabel(r'sepal width')
plt.xlabel(r'sepal length')
```

The parameter vector w, in the case of two features, has three elements. In the above code, they are present in the variables b, w1, and w2. We can calculate the **decision boundary** from them - in this case, it is a linear function with parameters m and c, which can be easily presented on the plane:

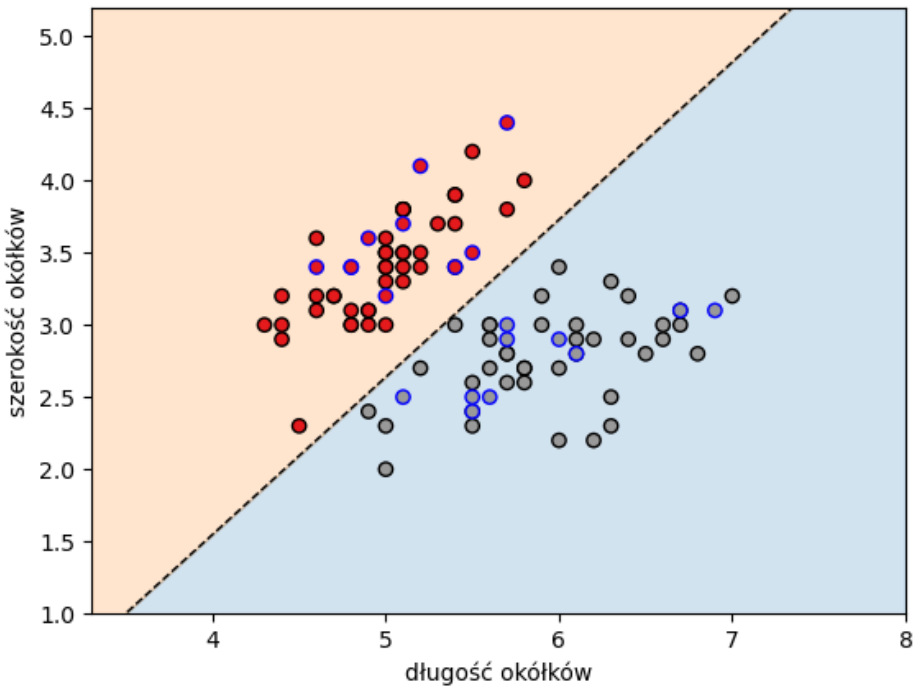

Figure 3

The decision boundary divides the feature space into two zones (marked in pink and blue) corresponding to the classes in our case. Samples from the training set (red and grey points enclosed by black contours) were used to find the parameters of the decision boundary. The learning algorithm, not visible in the above code, tried many decision boundaries and stopped at the best one. The labels from the y01_train vector were also used - they served as a pattern for comparison with classification results, which allowed the learning algorithm to modify the decision boundary correctly. This type of learning, where the training set contains samples and their labels, is called **supervised learning**.

The samples from the test set (enclosed by blue contour) were not used in the learning process. They serve to evaluate the classifier. The red points representing samples from one class should be in the "pink zone", and grey points in the "blue zone". In this case, it happened - on this test set, the classifier achieved 100% accuracy.

However, a poorly used classifier will struggle in classifying samples representing iris versicolor and iris virginica species (classes 1 and 2). **Modify the code accordingly to train a classifier that distinguishes these two classes based on the same features - the length and width of the sepals. Save the resulting plot with samples and decision boundary. Count, for both classes, how many training and test samples were misclassified. After completing this task, run the code again, which will cause a new random split of samples into training and test sets. See if the decision boundary has changed. Do classification results depend on the choice of the training set?**

In this exercise, only the basic components of learning systems and one type of problem that these systems solve - classification - have been outlined. Feature selection was done manually. A linear logistic regression model was chosen.

In the following exercises, knowledge about these basic elements, types of learning, and models will be expanded and deepened. In Exercise 2, we will focus on features, their extraction, dimensionality reduction of the feature space, and ways of visualization. Exercise 3 will present some unsupervised learning methods. In Exercise 4, the distinction between classification and regression problems will be introduced. Different models and measures of their accuracy will be presented. Exercise 5 will introduce artificial neural networks and this topic will be continued within labs of the **deep learning** subject.

Final remarks:

Important concepts were bolded in the text. Using the literature or open courses, for example, familiarize yourself with them to a basic extent. Chosen concepts will be the subject of next labs, but you should know something about all of them.

The following sources were used in the instruction:

https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html?highlight=iris

https://en.wikipedia.org/wiki/Iris_flower_data_set

https://scipython.com/blog/plotting-the-decision-boundary-of-a-logistic-regression-model/

https://www.ibm.com/docs/pl/spss-statistics/saas?topic=statistics-generalized-linear-models

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

This instruction was translated from its Polish version by ChatGPT. *ChatGPT is a large language model created by OpenAI, designed to converse with humans in natural language. It is based on the transformer architecture and has been trained on a vast corpus of text data to generate human-like responses to a wide variety of queries and topics* – as it introduced itself when asked what ChatGPT is in short.