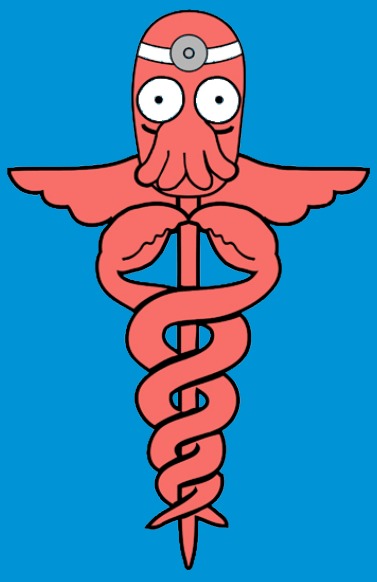
{EPITECH}

ZOIDBERG2.0

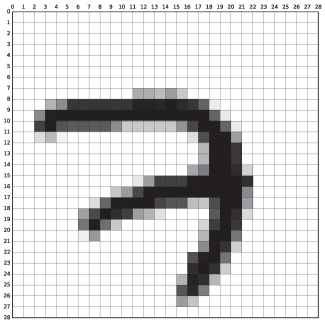
BOOSTRAP



ZOIDBERG2.0

Imports

Get your hands on a popular handwritten digits datababase called **MNIST**.
It should contain 4 datasets.



(a) MNIST sample belonging to the digit '7'.

(b) 100 samples from the MNIST training set.

The `training set images` is of shape [60000,28,28]: find what that means.

Each number represents a shade of grey for a specific pixel: coordinates [10,12,14] represent this value for the pixel in line twelve, fourteenth column of the tenth image.

The `training set labels` contains one label for each one of the previous images.

Now you are introduced with MNIST, let's play with it:

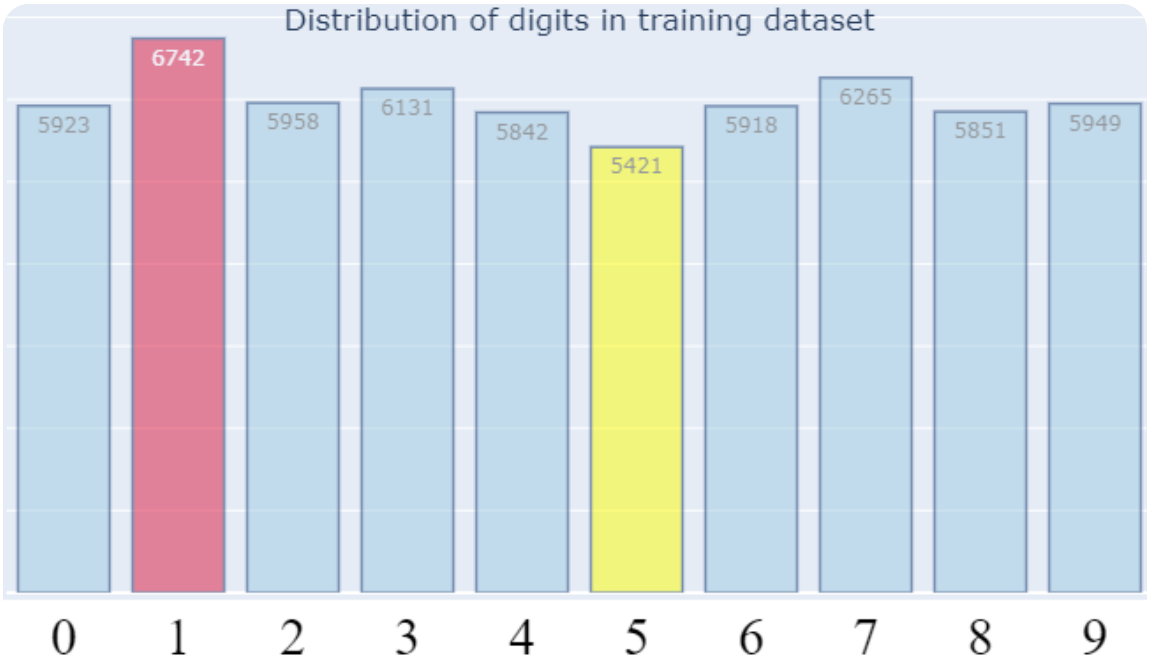
- ✓ Create a class, import and load the datasets in four matrix **with correct shapes** ;
- ✓ Create a method to display basics statistics, like the distributions of each digit ;
- ✓ Create a method to display one desired image from a chosen dataset ;
- ✓ Each digit example has its own specificities: find a way to display the *mean* of each digit ;
- ✓ Use your class to create arrays and reshape the images datasets to size [n,28x28]=[n,784].

You can use the matplotlib library to show the digit using your favorite colormap.

Here, an example of the distribution of digits in both train and test

digits	train	test
0	5923	980
1	6742	1135
2	5958	1032
3	6131	1010
4	5842	982
5	5421	892
6	5918	958
7	6265	1028
8	5851	974
9	5949	1009

Once you feel comfortable with this part, you may want to display your statistics with nicer libraries like seaborn or plotly



Predictions

Learning algorithm understands bidimensional matrix: each line represents an “individual” and columns represent its variables.

In our study case, one variable is a specific pixel coded with its shade of black and white. This matrix is X.

On the other side, the labels that you have to predict are ordered within a column called Y.

Using your data X, you have to find a function F to predict the labels Y.

The appearance of your function will strongly depend of the algorithm you used.

Reshape the arrays for images so they have a standard matrix format (2 dimensions).
The labels part are called output, they are the numbers you need to predict.

We are in a situation of supervised learning for multiclass predictions, meaning you have to predict 3 or more (among 10 different classes):

- ✓ choose (or create) a ML algorithm and a library to use it
- ✓ train your algorithm with `training sets images` and `training sets labels`
- ✓ try your algorithm with `test set images` and compare its results with `test set labels`

You have to define a measurement of quality for your predictions. You might use accuracy, or any other metrics that you like, **as long as you justify your choice**.

ML libraries like sklearn may be useful, but be sure to understand the methods you used. The different steps like preprocessing, testing, training and evaluating should be explicit in your project; parameters you have chosen and metrics you used should also be justified.

`KNN` is a basic and easy algorithm to understand.

Questions

Ask yourselves some questions, and try to find answers:

- ✓ Why use a separate dataset to measure the performance of an algorithm?
  What are your results when you test your algorithm on the same dataset used in training?

> It is easier to prevent bad habits than to break them.
>
> **— Benjamin Franklin —**

- ✓ What are bias and variance?
  What do they measure?
  Which values should they take?
- ✓ What is cross validation?
  What are the main advantages?
  When can I use it?
- ✓ Can you explain why it's important to normalize (scale) the data when using algos like `KNN`?
  Is it necessary in our specific study case?

Try to preprocess your data, then build and test your algorithm again to see if there is any improvement.

- ✓ When you reshaped your image, do you think the order of the columns (that means the order of the pixels) had an importance for the performance of your algorithm?

Try to apply the same pixels permutation in all the images, then compare your score with previous one(s).If you look closely at Knn's working, the answer should be clear.

- ✓ Which metrics measure performance?
  What does accuracy tell you?
  Does accuracy penalize more one mistake over another?
- ✓ Can you give an explicit example where accuracy would not be a relevant metric?
  In that extreme case, can you propose a more suitable metric?
- ✓ Do confusion matrix display all informations of algorithms' learning?
- ✓ Do you want to have a walk in the `forest` and observe `tree` growth?

Visualizations

Now you have one algorithm and its mesure of performance, let's produce some visualizations and statistics:

- ✓ represent the percentage of misclassified examples for each digit ;

Think of every possible way to display your results

- ✓ show misclassifications made by your algorithm for each digit ;
- ✓ plot a confusion matrix of your predictions ;
- ✓ elaborate on which class your algorithm performs well or poorly.

Here is an example of the confusion matrix for our algorithm (*accuracy=67.2%*):

obs/pred	0	1	2	3	4	5	6	7	8	9
0	873	0	45	1	9	20	9	1	20	2
1	0	1095	5	6	2	3	5	1	18	0
2	59	6	711	78	15	27	105	5	22	4
3	9	13	63	698	8	91	3	12	103	10
4	15	9	7	8	673	31	15	62	8	154
5	47	4	39	138	36	402	11	26	169	20
6	26	3	97	1	21	5	794	0	11	0
7	9	22	8	13	102	26	2	642	26	178
8	78	26	36	148	24	214	11	34	385	18
9	12	3	1	9	252	19	2	249	14	448

Unsupervised learning

UL is a subsection of *ML*. It uses algos to explain the dynamics in some **unlabelled data**.

As you might have noticed, it's not the case in this project.
Yet, you can use UL to visualize your data in a clear way.
It's often useful when you don't have a clear goal.
It can be used to find clusters, reduce dimensions…

v 2.4