



# Intelligent Systems

*Laboratory activity 2016-2017*

Adrian Groza, Anca Marginean and Radu Razvan Slavescu  
Tool:

Name: Cheres Ioana  
Group:30235  
Email: cheresioana@gmail.com

Assoc. Prof. dr. eng. Adrian Groza  
Adrian.Groza@cs.utcluj.ro



# Contents

<b>1</b>	<b>Rules and policies</b>	<b>4</b>
<b>2</b>	<b>Perceptron</b>	<b>9</b>
2.1	Narrative description . . . . .	9
2.2	Facts . . . . .	9
2.3	Specifications . . . . .	9
2.4	Top level design of the scenario . . . . .	10
2.5	Knowledge acquisition . . . . .	10
<b>3</b>	<b>Multi Layer Perceptron</b>	<b>12</b>
3.1	Narrative description . . . . .	12
3.2	Facts . . . . .	12
3.3	Specifications . . . . .	15
3.4	Top level design of the scenario . . . . .	16
3.5	Knowledge acquisition . . . . .	16
<b>4</b>	<b>CNN</b>	<b>18</b>
4.1	Narrative description . . . . .	18
4.2	Facts . . . . .	18
4.3	Specifications . . . . .	19
4.4	Top level design of the scenario . . . . .	19
4.5	Knowledge acquisition . . . . .	19
<b>5</b>	<b>Decision trees</b>	<b>21</b>
5.1	Narrative description . . . . .	21
5.2	Facts . . . . .	21
5.3	Specifications . . . . .	21
5.4	Top level design of the scenario . . . . .	21
5.5	Knowledge acquisition . . . . .	22
<b>6</b>	<b>KNN</b>	<b>23</b>
6.1	Narrative description . . . . .	23
6.2	Facts . . . . .	23
6.3	Specifications . . . . .	24
6.4	Top level design of the scenario . . . . .	25
6.5	Knowledge acquisition . . . . .	25
<b>7</b>	<b>K-means</b>	<b>26</b>
7.1	Narrative description . . . . .	26
7.2	Facts . . . . .	26
7.3	Specifications . . . . .	29

7.4	Top level design of the scenario . . . . .	30
7.5	Knowledge acquisition . . . . .	30
<b>8</b>	<b>Related work and documentation (<math>W_{12}</math>)</b>	<b>31</b>
8.1	Advantages and limitations of your solution . . . . .	31
<b>9</b>	<b>Your original code</b>	<b>32</b>

# Chapter 1

## Rules and policies

### Lab organisation.

1. Laboratory work is 20% from the final grade.
2. There are 4 deliverables in total (see Table 1.1).
3. The scheduling of your work is listed in Table 1.
4. Before each deadline, you have to load your work (latex documentation/code) on moodle.cs.utcluj.ro. Work sent by email will not be graded!

Class: Intelligent Systems 2016-2017  
Enrolment key: Is2016-2017

5. Realistic and original scenarios are encouraged. Well known toy problems (salesmen, map colouring, logistic planning, wumpus, sudoku, queens, missionaries and cannibals, various logic puzzle, etc.) do not worth much for your grade. Your scenario should be realistic and should be business oriented. That means that you should imagine a real client asking you to perform some investigation in the AI domain. Analysing a realistic AI task is complex and can be very demanding but all students who put in the time and effort got there eventually. Note that the focus is both on programming and on modelling the reality into a formal representation. You should be aware that computing interacts with many different domains. Solutions to many AI problems require both computing skills and domain knowledge.
6. *Laptop policy*: you can use your own laptop as long you have Linux. One goal of the laboratory is to increase your competency in Linux. It is **your** task to set static IPs:

IP: 192.168.1.[51..98]  
MASK: 255.255.255.0  
GATEWAY: 192.168.1.2  
DNS: 192.168.1.2  
PROXY 192.168.1.2:3128

Another option is to use EDUROAM or local WIFI

Network: isg  
Password: inteligentaartificiala

7. *Group change policy*. Maximum number of students in a class is 14.

Table 1.1: Deliverables.

Deliverable	Description	Deadline
Proposal	Description of your proposed project	$W_5$
Midway report	Review of related work, details of the proposed method, and preliminary results if available	$W_7$ .
Final report	A full academic paper, including: problem definition and motivation, background and related work, details of the proposed method, details of experiments and results, conclusion and future work, references and appendix	$W_{13}$
Presentation	Present your work to the colleagues, instructors	$W_{14}$

8. *AI learning community.* When classes are not scheduled, the AI laboratory is a dedicated physical space for students to socialize in during non-class time. As an undergraduate you might meet some eager diploma or master students pursuing their path towards junior AI researchers.
9. *For students repeating the class:* For validating the grade obtained in previous years, a discussion is mandatory in the first week. I usually have no problem to validate your previous grades, as long you request this in the first week. Failing to do so, leads to the grade 1 for the laboratory work in the current semester.

**Grading.** Assessment aims to measure your knowledge and skills needed to function in realistic AI-related tasks. Assessment is based on your written report explaining the nature of the project, findings, and recommendations. Meeting the deadlines is also important. Your report is comparable to ones you would write if you were a consultant reporting to a client.

Grade inflation makes difficult to distinguish between students. It also discourages the best students to do their best. In my quest for “optimal ranking of the students“, I do not use the following heuristics:

- “He worked hard at the project“. Our society do not like anymore individuals that are *trying*, but individual that *do* stuff. Such heuristic is not admissible in education, except the primary school.
- “I knew he could do much better“. Such a heuristic is not admissible because it does not encourage you to spread yourself.
- 7 means that you: i) constantly worked during classes, ii) you proved competent to use the tool and its expressivity for a realistic scenario, iii) you understood theoretical concepts on which the tool rely on.
- 8, 9 mean that your code is large enough and the results proved by your experiments are significant.
- 10 means that you did very impressive work or more efficient than I expected or handled a lot of special cases for realistic scenarios.

Table 1.2: Lab scheduling.

<b>Activity</b>	<b>Deadline</b>
<i>Installing the tool.</i>	$W_2$
<i>Running and understanding examples</i> attached to each tool.	$W_3$
<i>Selecting an adequate scenario</i> to be implemented using the tool. <i>Describing the specifications</i> of your own scenario. Describing the top level design of your scenario.	$W_4$
<i>Identifying and describing the knowledge bases</i> (data sets, articles, studies, etc.) planned to be used for realistic modelling of your scenario.	$W_5$
<i>Implementing your scenario.</i> We try to evaluate how much your solution reflects the reality. This is a quantitative criteria: we evaluate how many aspects from real world have been covered by your solution.	$W_{10}$
<i>Exploiting the expressivity of the tool.</i> We try to evaluate how many capabilities provided by the tool have been enacted within your implementation. This is a qualitative criteria: a complex realistic scenario requires more expressivity.	$W_{11}$
<i>Validating the correctness/efficiency</i> of your solution through graphs and/or experiments.	$W_{12}$
<i>Comparing your solution with related work.</i> Illustrate the advantages and weak points of your solution (description and code showing advantages/disadvantages). <i>Demonstrating the running scenario and Latex documentation.</i> You have to show some competency on writing documentation in Latex. For instance, you have to employ various latex elements: lists, citations, footnotes, verbatim, maths, code, etc.	$W_{13}$
<i>Public presentation and feedback</i> provided by the supervisor to clarify the good/bad issues related to student activity/results during the semester.	$W_{14}$

- 5 means that you managed to develop something of your own, functional, with your own piece of code substantially different from the examples available.
- You obtain less than 5 in one of the following situations:
  1. few code written by yourself (i.e 10 formulas in First Order Logic, <10 operators in PDDL).
  2. too much similarity with the provided examples.
  3. non-seriousity (i.e. re-curent late at classes, playing games, worked for other disciplines, poor/unprofessional documentation of your work, etc.)<sup>1</sup>.
- You get 2 if you present the project but fail to submit the documentation or code before the deadline. You get 1 if you do not present your project before the deadline. You get 0 for any line of code taken from other parts that appear in section *My own code*. For information on TUCN's regulations on plagiarism do consult the active norms.

If your grade is 0, 1, or 2, you do not satisfy the preconditions for participating to the written exam. The only possibility to increase your laboratory grade is to take another project in the next year, at the same class, and to make all the steps again.

However, don't forget that focus is on learning, not on grading. Consider this lab as an opportunity to practice your skills on real-world problems. This laboratory best serves as a test to see weather you have at the moment the potential for graduate studies. The lab also provides you with an insight from research methodology. The lab project aims to: 1) develop your critical thinking; 2) enhance your ability to work independently; 3) develop your technical writing and presentation skills.

**Plagiarism.** Most of you consider plagiarism only a minor form of cheating. This is far from accurate. Plagiarism is passing off the work of others as your own to gain unfair advantage.

During your project presentation and documentation, I must not be left with doubts on which parts of your project are your work or not. Always identify both: 1) who you worked with and 2) where you got your part of the code or solution.

Describe clearly the starting point of your solution. List explicitly any code re-used in your project. List explicitly any code adapted from other sources. List explicitly any help (including debugging help, design discussions) provided by others (including colleagues or teaching assistant). Keep in mind that it is your own project and not the teaching assistant's project. Learning by collaborating does remain an effective method. You can use it, but don't forget to mention any kind of support. Learning by exploiting various knowledge-bases developed by your elder colleagues remain also an effective method for "learning by example". When comparing samples of good and poor assignments submitted by your colleagues in earlier years try to identify which is better and why. You can use this repository of previous assignments, but don't forget to mention any kind of inspiration source.

The assignment is designed to be individual and to pose you some difficulties (both technological and scientific) for which you should identify a working solution by the end of the semester. Each semester, a distinct AI tool is assigned to a small group of students. Your are strongly encouraged to collaborate, especially during the installation phase and example understanding phase ( $W_1$ - $W_4$ ). The quicker you get throughout these preparatory stages, the more time you have for your own project.

---

<sup>1</sup>Consider non-seriousity as a immutable boolean value that is unconsciously activated in my brain when one of the above conditions occurs for the first time.

**Class attendance.** We are all grown-ups, when and whether you attend lecture is up to you<sup>2</sup>. Keep in mind the exam can include any topic that was covered during class, explained on the board, or which emerged from discussions among participants.

First class is the most important one. It is the class in which we clarify the rules. Future mails or requests for such clarifications are not welcome. After the first class, the necessary presumption is that these rules are common knowledge. You should be aware that "Ignorantia legis neminem excusat".

Instead, attendance to laboratory classes is mandatory. Missing lab assignments or midterm leads to minimum grade for that part. You are free to manage your laboratory classes - meaning that you can submit the project earlier or send your work earlier - as long as you meet all the constraints and deadlines. However, it is mandatory to participate at the final public presentation of your project.

**Re-assessment of your work.** Your project is developed based on your interaction with instructor, interaction with similar examples, interaction with scientific literature, but also on your scientific interests<sup>3</sup> and your own deliberation. All these processes require some time steps and scheduling. That's why, developing another project in 3-4 days of the re-examination period does not provide the skills that I am interested you to have for AI. Consequently, the project will be assessed only once, in week 13. Consider that i) project assessment, ii) midterm, iii) kahoot points, iv) lecture quizzes, v) other assignments are activities that take place within the allocated 14 weeks of the semester. Asking for a project re-assessment is similar to asking for an extra kahoot game only for you.

---

<sup>2</sup>However, you should be aware that when signing the study contract, you have some (moral) obligations towards the people that are paying your classes through the Ministry of Education budget. These people asked you to spend 8 hours daily for learning. By signed the university contract, you practically agree to study 8 hours/day.

<sup>3</sup>I am forced to assume that, by studying computer science you do have such scientific curiosity.



# Chapter 2

## Perceptron

### 2.1 Narrative description

A perceptron is a simple algorithm that represents the smallest unit of a neural network. It consisting of n inputs, n weights 1 bias 1 activation function and one output.

The function implemented by the perceptron is

$$sum = inputs * weights + bias$$

$$if sum > 0 \Rightarrow output = 1$$

$$if sum < 0 \Rightarrow output = 0$$

### 2.2 Facts

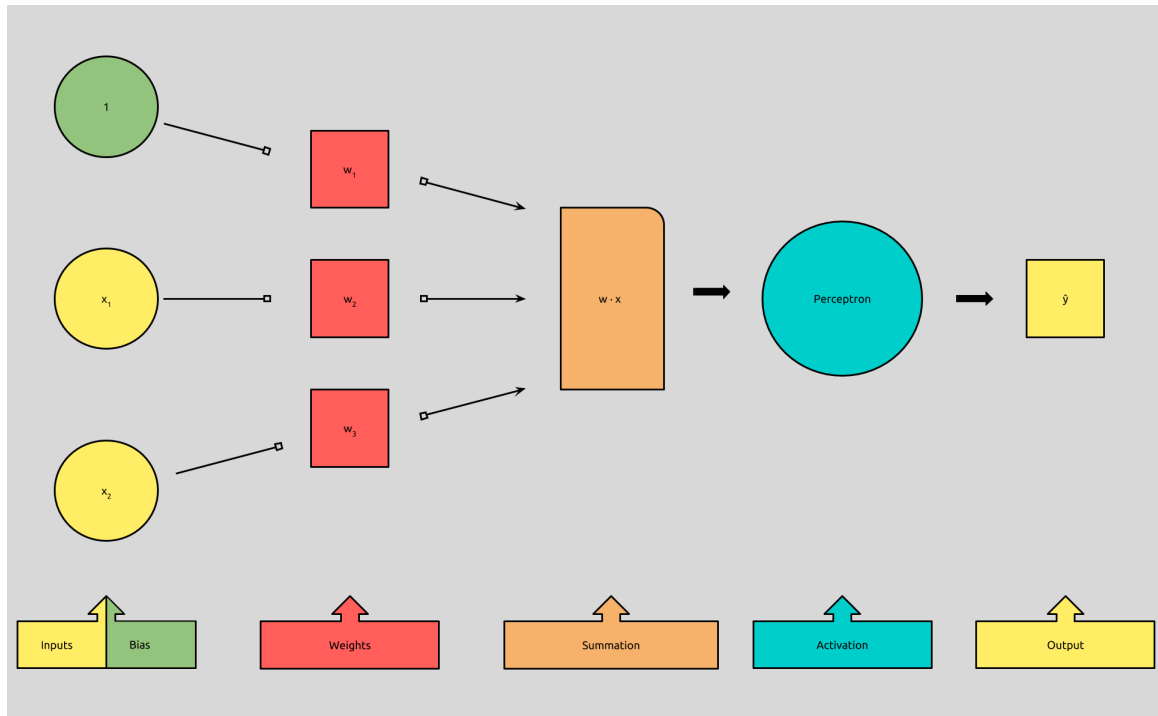
For the function I chose to implement logical "or" function for 3 elements.

TRUTH TABLE			
INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

### 2.3 Specifications

For implementing the project i used python and numpy. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

## 2.4 Top level design of the scenario



The perceptron is composed by  $n$  inputs,  $n$  weights 1 bias 1 activation function and one output.

## 2.5 Knowledge acquisition

**How do represent knowledge?** Each input node is one value and one parameter in perceptron equation. The weights represent how important is each input to the final result. The weights are stored in an array. The bias is the bias term of the equation. In order to have a single data structure for "internal" values, the bias is stored in the weights array at index 0.

**Where are you getting the required knowledge/data** The output is computed by the perceptrons function.

$$sum = inputs * weights + bias$$

The perceptron is train on the input dataset.

```
training_inputs.append(np.array([1, 1, 1]))
training_inputs.append(np.array([1, 1, 0]))
training_inputs.append(np.array([1,0, 1]))
training_inputs.append(np.array([1, 0, 0]))
training_inputs.append(np.array([0, 1, 1]))
training_inputs.append(np.array([0, 1, 0]))
training_inputs.append(np.array([0,0, 1]))
training_inputs.append(np.array([0, 0, 0]))
```

```
labels = np.array([1, 1, 1, 1,1,1, 1, 0])
```

After the weights are computed, the perceptron can compute the output of or function. The computed weights are

```
Computed weights[0.  0.01 0.01 0.01]
```

The user can verify the way perceptron computes the output by multiplying the weights with the inputs and adding the bias.

Predicted value for 1 or 1 or 1 = 1

Computing  $0.0 + 0.01 * 1 + 0.01 * 1 + 0.01 * 1 = 0.03$

Predicted value for 0 or 1 or 0 = 1

Computing  $0.0 + 0.01 * 0 + 0.01 * 1 + 0.01 * 1 = 0.01$

Predicted value for 1 or 0 or 0 = 1

Computing  $0.0 + 0.01 * 1 + 0.01 * 0 + 0.01 * 0 = 0.01$

Predicted value for 0 or 0 or 0 = 0

Computing  $0.0 + 0.01 * 0 + 0.01 * 0 + 0.01 * 0 = 0.0$

# Chapter 3

## Multi Layer Perceptron

### 3.1 Narrative description

A multi layer perceptron (MLP) is a class of feedforward artificial neural network. It is composed of multiple layers of perceptrons (with threshold activation).

Each perceptron is computing a small function (as described in the previous chapter). The perceptrons are arranged in layers, the inputs of each perceptron from the first layers are the features. The outputs of the first layer of perceptrons are the inputs of the 2nd layer of perceptrons and so on. At the end, the output of the final layer represents the output of the neural network.

The weights are computed using a process called backpropagation. The "backwards" part of the name comes from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately

The dataset that I used contained the information from the people that were on Titanic, along with a column that tells if they survived or not. The model tries to predict if a person survived, given a set of features like: age fare parch class sex number of siblings family size and title

### 3.2 Facts

The titanic dataset is the dataset that contains all the informations about the people that were on Titanic. I processed the data and I generated some statistics

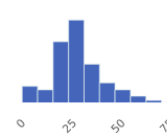
# Variables

## Age

Real number ( $\mathbb{R}_{\geq 0}$ )

Distinct count	89
Unique (%)	10.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	29.44519640852974
Minimum	0.42
Maximum	80.0
Zeros	0
Zeros (%)	0.0%
Memory size	7.1 KiB



Toggle details

## Cabin

Categorical

HIGH CARDINALITY  
MISSING

Distinct count	147
Unique (%)	72.1%
Missing	687
Missing (%)	77.1%
Memory size	7.1 KiB

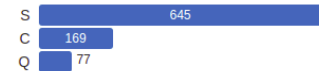


Toggle details

## Embarked

Categorical

Distinct count	3
Unique (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB



Toggle details

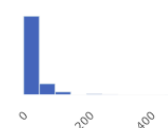
## Fare

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	248
Unique (%)	27.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	32.204207968574636
Minimum	0.0
Maximum	512.3292
Zeros	15
Zeros (%)	1.7%
Memory size	7.1 KiB



Toggle details

## Name

Categorical

HIGH CARDINALITY  
UNIQUE

Distinct count	891
Unique (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB



Toggle details

## Parch

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	7
Unique (%)	0.8%
Missing	0
Missing (%)	0.0%

Mean	0.38159371492704824
Minimum	0
Maximum	6
Zeros	678



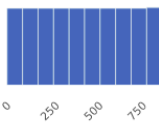
PassengerId

Real number ( $\mathbb{R}_{\geq 0}$ )

UNIQUE

Distinct count	891
Unique (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	446.0
Minimum	1
Maximum	891
Zeros	0
Zeros (%)	0.0%
Memory size	7.1 KiB



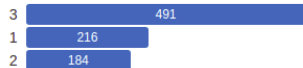
Toggle details

Pclass

Categorical

Distinct count	3
Unique (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

3	491
1	216
2	184



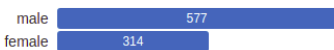
Toggle details

Sex

Categorical

Distinct count	2
Unique (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

male	577
female	314



Toggle details


SibSp

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	7
Unique (%)	0.8%
Missing	0
Missing (%)	0.0%
Infinite	0

Mean	0.5230078563411896
Minimum	0
Maximum	8
Zeros	608
Zeros (%)	68.2%



Survived

Boolean

Distinct count	2
Unique (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

0549

1342

Toggle details

Ticket

Categorical

HIGH CARDINALITY

Distinct count	681
Unique (%)	76.4%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

16017

3470827

CA. 23437

3470886

CA 21446

Other values (676)858

Toggle details

Title

Categorical

Distinct count	6
Unique (%)	0.7%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

Mr525

Miss185

Mrs128

Master40

Dr7

Toggle details


Family\_Size

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	9
Unique (%)	1.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	0.9046015712682379
Minimum	0
Maximum	10
Zeros	537
Zeros (%)	60.3%
Memory size	7.1 KiB



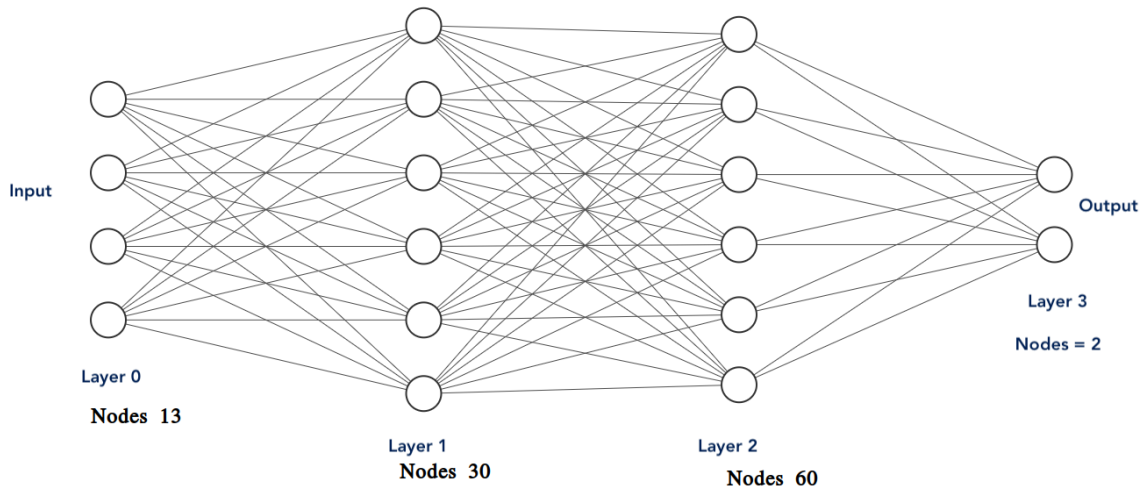
### 3.3 Specifications

For implementing the project i used python, numpy and pandas.

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas is a software library written for the Python programming language for data manipulation and analysis

### 3.4 Top level design of the scenario



The neural network has 4 layers. First is the input layer, then there are 2 hidden layers and one output layer.

The first hidden layer has a sigmoid activation function and the 2nd one relu activation.

### 3.5 Knowledge acquisition

**How do represent knowledge?** The data is represented in a dataframe with the features age fare parch class sex number of siblings family size and title. The title feature is a string, so it has to be binarized, this is why it is represented as a 6 features. For it's transformation I used LabelBinarizer. The sex feature is also a string, so I used number 1 for females and 0 for males.

All the data is scaled between -1 and 1 using a Min Max Scaler. The min max scaler apply the function written below on each feature.

$$X_{std} = (X - X_{min}) / (X_{max} - X_{min})$$

$$X_{scaled} = X_{std} * (max - min) + min$$

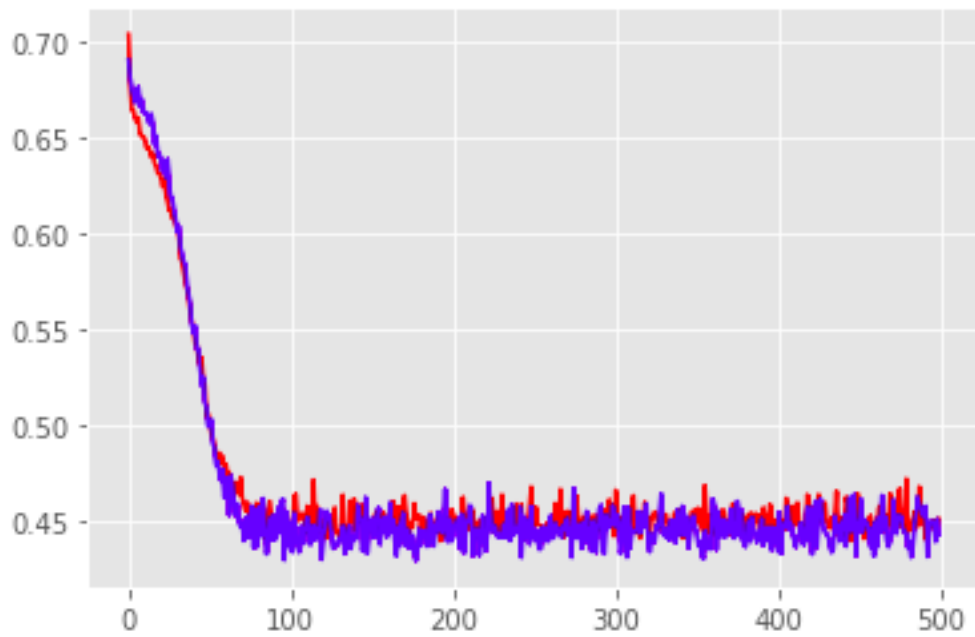
The dataset is then split 80% for training ad 20% for validation

**Where are you getting the required knowledge/data** The algorithm computes the output for each minibatch. It then updates the weights for each layer using the backpropagation. After one epoch, the validation set is tested to compute the accuracy and the loss. In order to avoid overfit, underfit or the divergence of the algorithm, I made some plots. At the end of the algorithm we can check the loss value on the test and validation set, to see how well the algorithms fits the data.

```
[TRAIN] Epoch: 499 Loss:0.4515246956244759
[Validation set] Loss: 0.44316211839516956
```

The chart for losses over epochs.

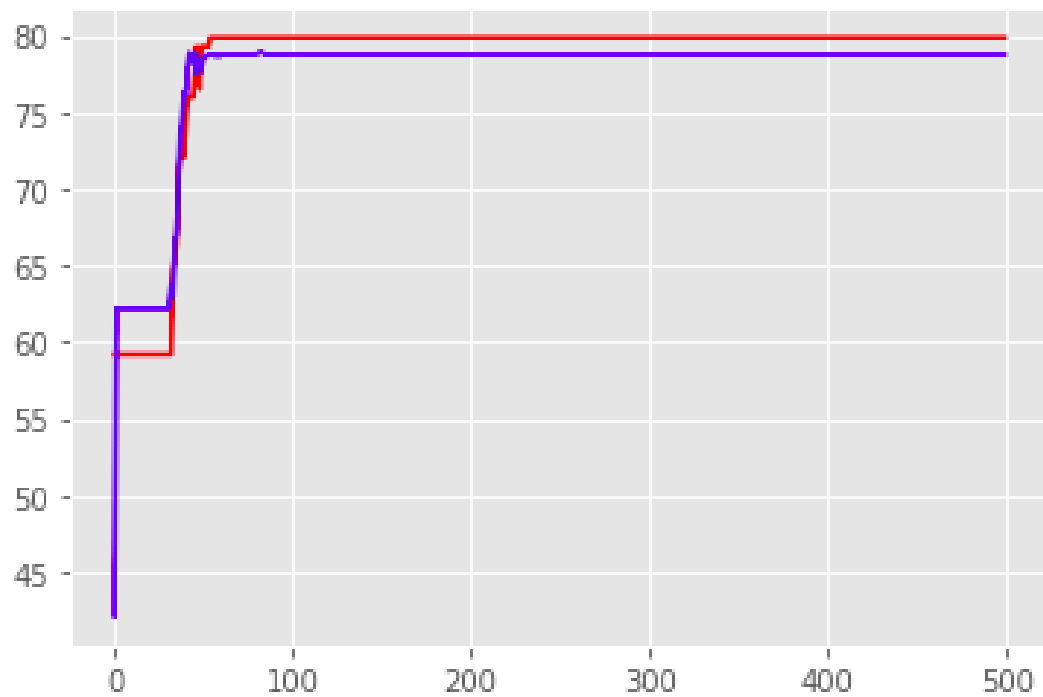




Also at the end of the algorithm we can see the accuracy of predictions on test and validation set.

```
[TRAIN] Accuracy: 78.7921371459961%  
[Validation set] Accuracy: 79.88826751708984%
```

The chart for the accuracy over epochs



# Chapter 4

## CNN

### 4.1 Narrative description

A convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. The neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with the image given as input.

The parameters for the convolution layers are:

in channels (int) — Number of channels in the input image

out channels (int) — Number of channels produced by the convolution

kernel size (int or tuple) — Size of the convolving kernel

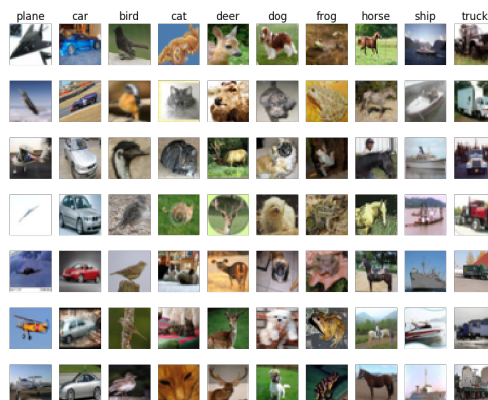
stride (int or tuple, optional) — Stride of the convolution. Default: 1

padding (int or tuple, optional) — Zero-padding added to both sides of the input. Default: 0

The final layer of the net is a fully connected layer.

### 4.2 Facts

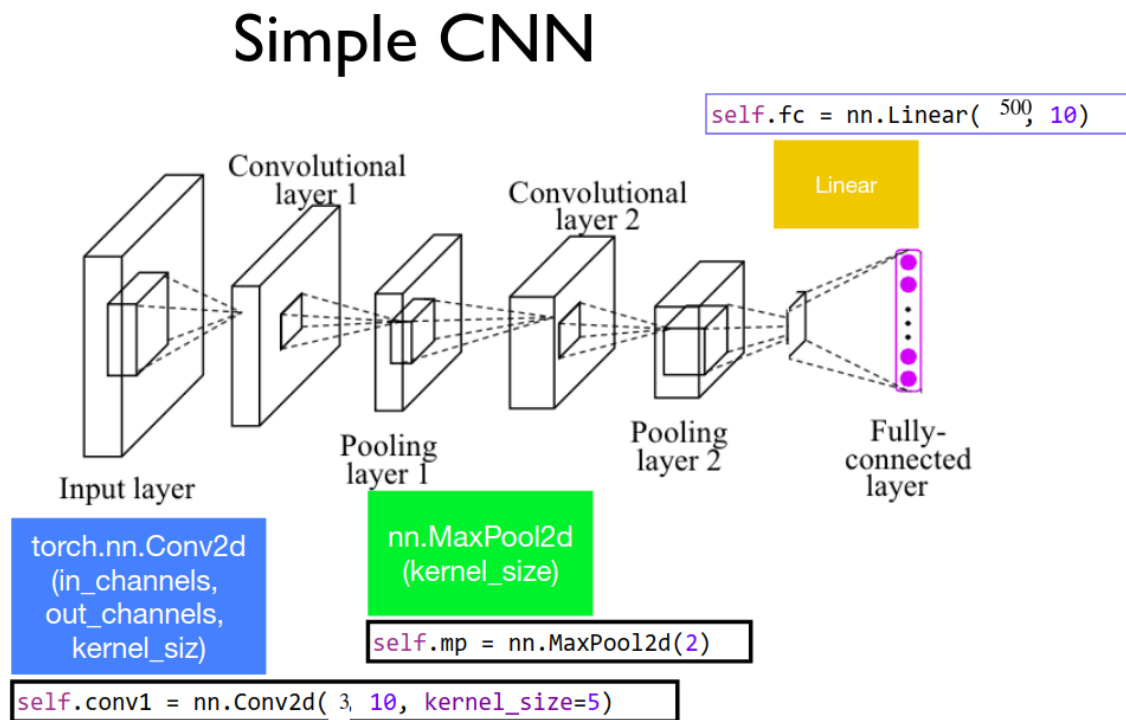
The dataset that I used is CIFAR10 dataset. It consists of photos 10 different type of animals and objects. The dataset is from torchvision datasets.



## 4.3 Specifications

For implementing the project i used python and torchvision. The torchvision package consists of popular datasets, model architectures, and common image transformations for computer vision.

## 4.4 Top level design of the scenario



The cnn has 2 convolutional layers at the begining and 2 fully connected layers. The first convolutional layes has 3 channel inputs because it deals with rgb color, a 10 channel output and a kernel size of 5. The Max pooling window takes the maximum from the specified area. The last 2 layers are 2 fully connected layers just like the ones in MLP.

## 4.5 Knowledge acquisition

**How do represent knowledge?** The inputs are images of type rgb.Each channel is represented as a number and normalized between 0 and 1. After each array of numbers that represent the image, there is a number between 0 an 10 which is the associated class for classification.

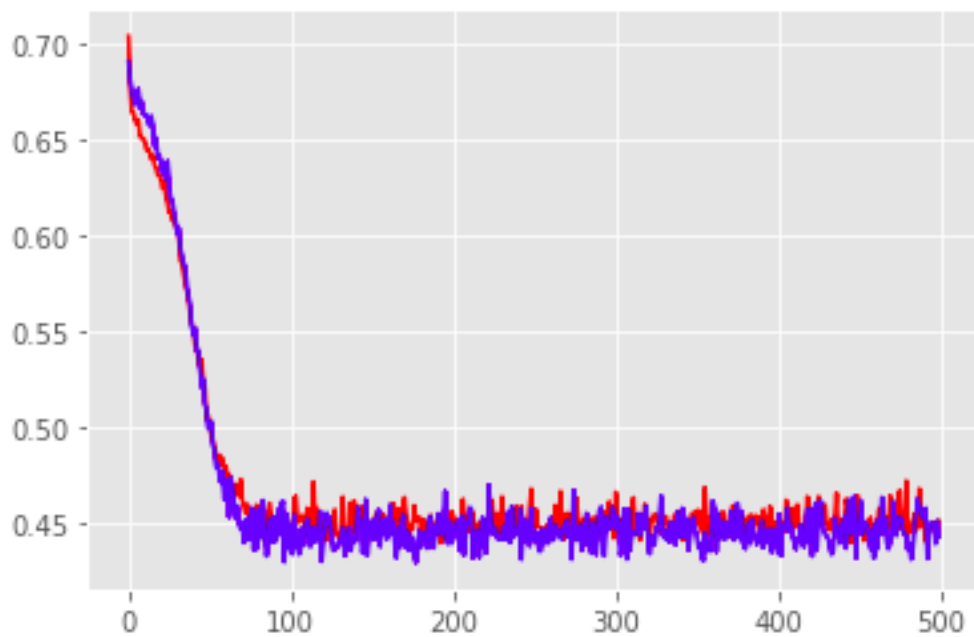
**Where are you getting the required knowledge/data** The last layer is the output of the network. The final layer consists of 10 outputs, each represents the probability of the input image being a part of a specific class. After the training

At the end of the algorithm we can check the loss value on the test and validation set, to see how well the algorithms fits the data.

[TRAIN] Epoch: 499 Loss:0.4515246956244759

[Validation set] Loss: 0.44316211839516956

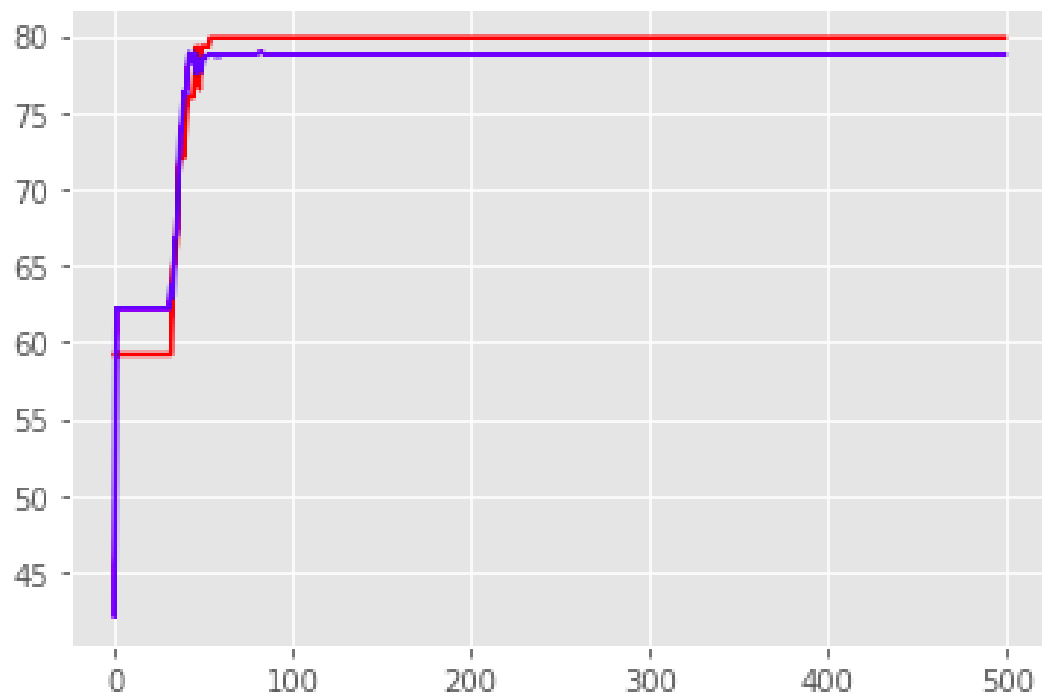
The chart for losses over epochs.



Also at the end of the algorithm we can see the accuracy of predictions on test and validation set.

```
[TRAIN] Accuracy: 78.7921371459961%  
[Validation set] Accuracy: 79.88826751708984%
```

The chart for the accuracy over epochs



# Chapter 5

## Decision trees

### 5.1 Narrative description

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network.

### 5.2 Facts

The titanic dataset is the dataset that contains all the informations about the people that were on Titanic.

### 5.3 Specifications

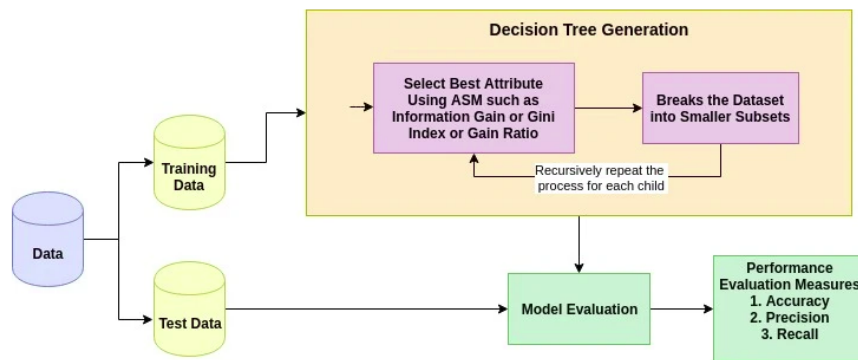
For implementing the project i used python, numpy, pandas and sklearn. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas is a software library written for the Python programming language for data manipulation and analysis.

Scikit-learn is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to inter-operate with the Python numerical and scientific libraries NumPy and SciPy.

### 5.4 Top level design of the scenario

The decision tree is composed by node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. It selects an attribute based on attribute selection measures. It tries to minimize the entropy of smaller sub-datasets by splitting them based on an attribute criteria.



## 5.5 Knowledge acquisition

**How do represent knowledge?** The data is represented in a dataframe with the features age fare parch class sex number of siblings family size and title. The title feature is a string, so it has to be binarized, this is why it is represented as a 6 features. For it's transformation I used LabelBinarizer. The sex feature is also a string, so I used number 1 for females and 0 for males.

All the data is scaled between -1 and 1 using a Min Max Scaler. The min max scaler apply the function written below on each feature.

$$X_{std} = (X - X_{min}) / (X_{max} - X_{min})$$

$$X_{scaled} = X_{std} * (max - min) + min$$

The dataset is then split 80% for training ad 20% for validation

**Where are you getting the required knowledge/data** The output is represented by the final nodes from the tree. Based on the values for certain attributes, the algorithm chooses a path towards the end of the tree.

Accuracy: 0.7798507462686567

31:



# Chapter 6

## KNN

### 6.1 Narrative description

k-nearest neighbors algorithm is a non-parametric method used for classification. The output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If  $k = 1$ , then the object is simply assigned to the class of that single nearest neighbor.

For this project I chose to classify the flowers from iris dataset.

### 6.2 Facts

In the iris dataset are 3 categories of flowers: 'setosa' 'versicolor' 'virginica'. The dataset contains features related to the flowers like 'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'.

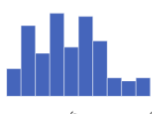
## Variables

sepal length (cm)

Real number ( $\mathbb{R}_{\geq 0}$ )

Distinct count	35
Unique (%)	23.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	5.843333333333334
Minimum	4.3
Maximum	7.9
Zeros	0
Zeros (%)	0.0%
Memory size	1.3 KiB




Toggle details

sepal width (cm)

Real number ( $\mathbb{R}_{\geq 0}$ )

Distinct count	23
Unique (%)	15.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	3.0573333333333337
Minimum	2.0
Maximum	4.4
Zeros	0
Zeros (%)	0.0%
Memory size	1.3 KiB



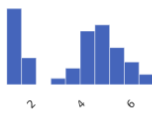
Toggle details

petal length (cm)

Real number ( $\mathbb{R}_{\geq 0}$ )

Distinct count	43
Unique (%)	28.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	3.7580000000000005
Minimum	1.0
Maximum	6.9
Zeros	0
Zeros (%)	0.0%
Memory size	1.3 KiB



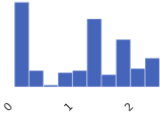
Toggle details

petal width (cm)

Real number ( $\mathbb{R}_{\geq 0}$ )

Distinct count	22
Unique (%)	14.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	1.1993333333333336
Minimum	0.1
Maximum	2.5
Zeros	0
Zeros (%)	0.0%
Memory size	1.3 KiB



Toggle details

target

Categorical

Distinct count	3
Unique (%)	2.0%
Missing	0
Missing (%)	0.0%
Memory size	1.3 KiB

2

50

1

50

0

50

Toggle details

## 6.3 Specifications

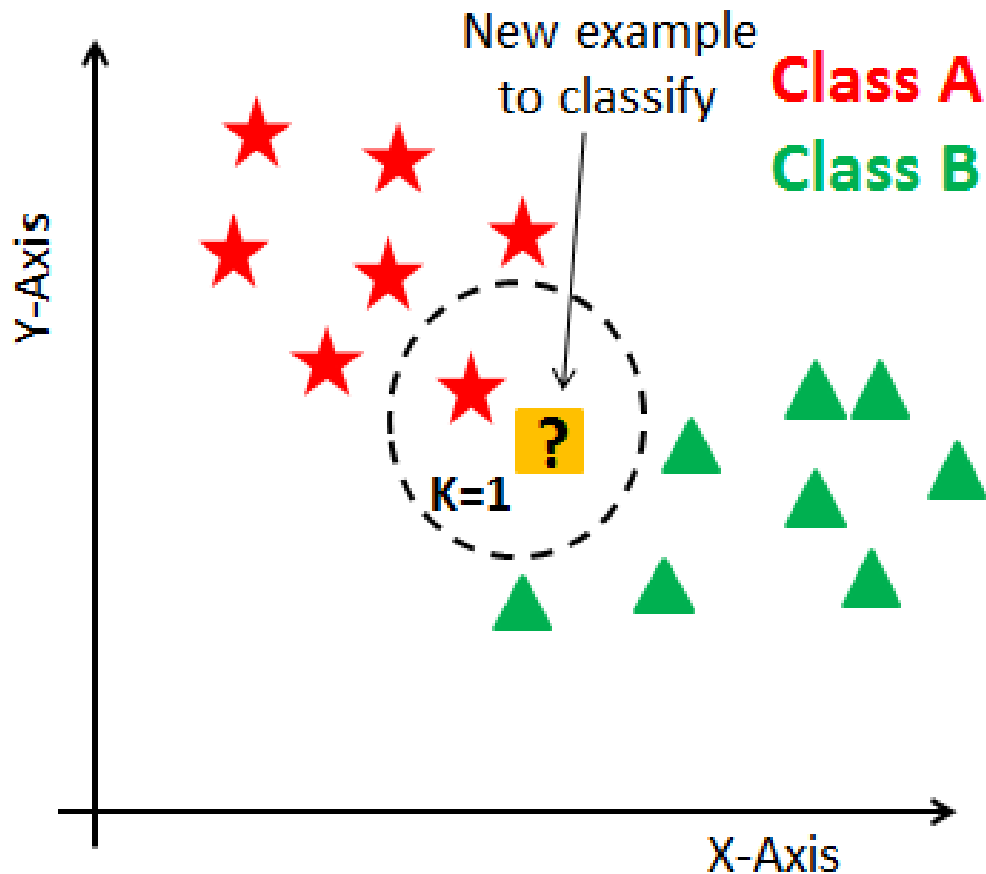
For implementing the project i used python, numpy and sklearn. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Scikit-learn is a free software machine learning library for the Python programming language.



It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

## 6.4 Top level design of the scenario



First the algorithm finds the closest  $K$  neighbours. If the neighbours are not classified, then it classifies them (that is in the lazy scenario). Then the algorithm finds the  $K$  nearest neighbours and then finds the class that is majority from the neighbours. It then classifies the data point according to the majority of the neighbours.

## 6.5 Knowledge acquisition

**How do represent knowledge?** All the data point are represented in  $n$ -dimension space, where  $n$  is number of features. The features from the flowers: 'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)' are numbers, which means that they do not have to be processed. The flower names (labels) are encoded as numbers because they are Strings. The names ['setosa' 'versicolor' 'virginica'] are represented as [0, 1, 2]

**Where are you getting the required knowledge/data** After the data is fitted the algorithm computes  $k$  nearest neighbours and outputs the predicted class for test set. The accuracy for Iris dataset with 7 neighbours was 0.9555555555555556.

# Chapter 7

## K-means

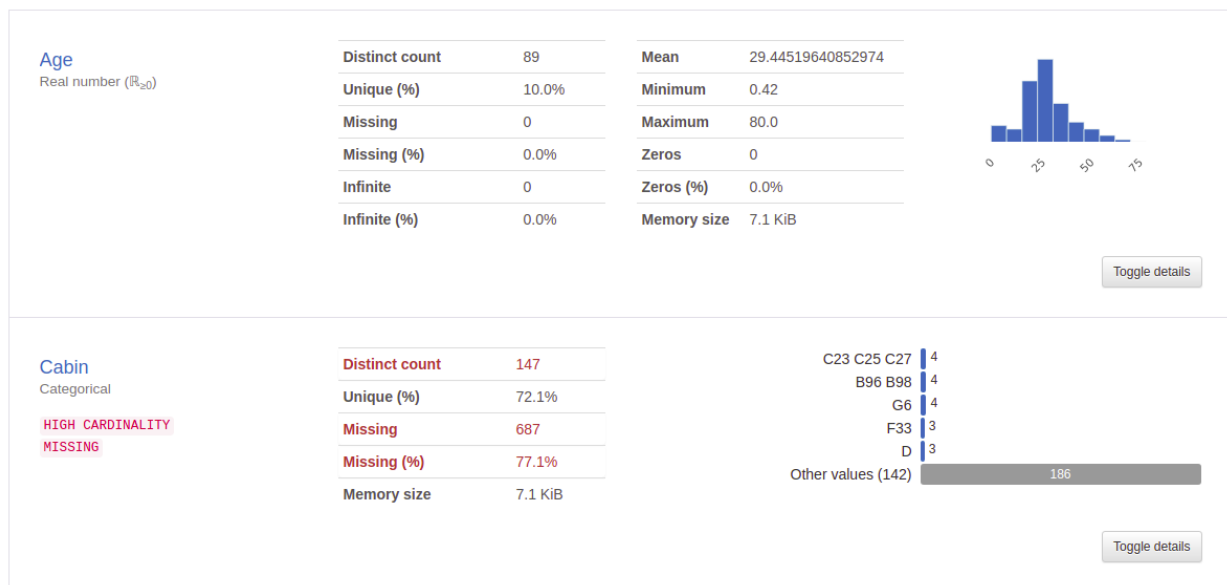
### 7.1 Narrative description

K-means clustering is a method of vector quantization that aims to partition  $n$  observations into  $k$  clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. K-means clustering minimizes within-cluster variances .

### 7.2 Facts

The titanic dataset is the dataset that contains all the informations about the people that were on Titanic. I processed the data and I generated some statistics

#### Variables



Embarked

Categorical

Distinct count	3
Unique (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

S645

C169

Q77

Toggle details

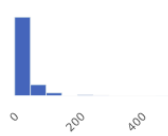
Fare

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	248
Unique (%)	27.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	32.204207968574636
Minimum	0.0
Maximum	512.3292
Zeros	15
Zeros (%)	1.7%
Memory size	7.1 KiB



Toggle details

Name

Categorical

HIGH CARDINALITY

UNIQUE

Distinct count	891
Unique (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

Gaskell, Mr. Alfred1

Johansson, Mr. Erik1

Abbott, Mr. Rossmore Edward1

Landergren, Miss. Aurora Adelia1

Leitch, Miss. Jessie Wills1

Other values (886)

886

Toggle details

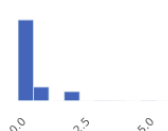
Parch

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	7
Unique (%)	0.8%
Missing	0
Missing (%)	0.0%

Mean	0.38159371492704824
Minimum	0
Maximum	6
Zeros	678



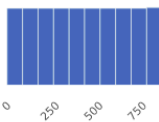
PassengerId

Real number ( $\mathbb{R}_{\geq 0}$ )

UNIQUE

Distinct count	891
Unique (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	446.0
Minimum	1
Maximum	891
Zeros	0
Zeros (%)	0.0%
Memory size	7.1 KiB



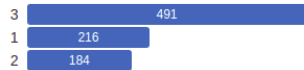
Toggle details

Pclass

Categorical

Distinct count	3
Unique (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

3	491
1	216
2	184




Toggle details

Sex

Categorical

Distinct count	2
Unique (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

male	577
female	314



Toggle details

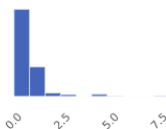
SibSp

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	7
Unique (%)	0.8%
Missing	0
Missing (%)	0.0%
Infinite	0

Mean	0.5230078563411896
Minimum	0
Maximum	8
Zeros	608
Zeros (%)	68.2%



Survived

Boolean

Distinct count	2
Unique (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

0549

1342

Toggle details

Ticket

Categorical

HIGH CARDINALITY

Distinct count	681
Unique (%)	76.4%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

16017

3470827

CA. 23437

3470886

CA 21446

Other values (676)858

Toggle details

Title

Categorical

Distinct count	6
Unique (%)	0.7%
Missing	0
Missing (%)	0.0%
Memory size	7.1 KiB

Mr525

Miss185

Mrs128

Master40

Dr7

Toggle details

Family\_Size

Real number ( $\mathbb{R}_{\geq 0}$ )

ZEROS

Distinct count	9
Unique (%)	1.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	0.9046015712682379
Minimum	0
Maximum	10
Zeros	537
Zeros (%)	60.3%
Memory size	7.1 KiB

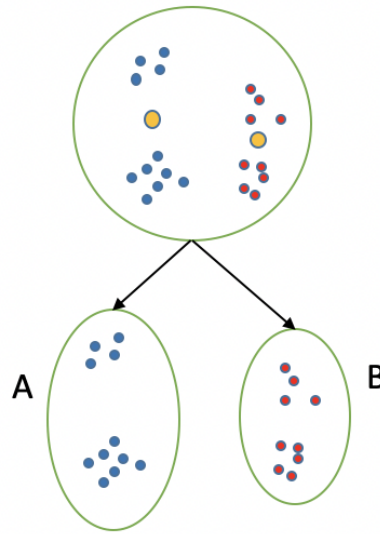
## 7.3 Specifications

For implementing the project i used python, numpy, pandas and sklearn. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas is a software library written for the Python programming language for data manipulation and analysis.

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

## 7.4 Top level design of the scenario



The Kmeans algorithm repeats the same algorithm for n times, where n is the number of epochs.

Step 1: chooses K random data points that become the centroids

Step 2: computes the distance to other data points and assign data points to clusters.

Step 3: Define the clusters based on the variance from inside the cluster.

Step 4: Compute the new centroid.

## 7.5 Knowledge acquisition

**How do represent knowledge?** The data is represented in a dataframe with the features age fare parch class id sex number of siblings family size . The sex feature is also a string, so I used number 1 for females and 0 for males.

All the data is scaled between -1 and 1 using a Min Max Scaler. The min max scaler apply the function written below on each feature.

$$X_{std} = (X - X_{min}) / (X_{max} - X_{min})$$

$$X_{scaled} = X_{std} * (max - min) + min$$

**Where are you getting the required knowledge/data** After the data is fitt, I test the clustering. The accuracy is 0.6262626262626263.

# Chapter 8

## Related work and documentation ( $W_{12}$ )

### 8.1 Advantages and limitations of your solution

Each algorithm is specialized on computing a certain task, so it's rather complicated to talk about advantages and disadvantages together.

#### 1. Perceptron

Advantages: Simple algorithm, easy to verify.

Disadvantages: It is too simple so it can not accomplish complex tasks.

#### 2. MLP

Advantages: Complex function that is easy to program and can resolve complex problems.

Disadvantages: It can not be verified.

#### 3. CNN

Advantages: Complex function that is easy to program and performs great on images.

Disadvantages: It can not be verified and requires a lot of memory and computing.

#### 4. Decision Trees

Advantages: Easy to follow algorithms that look like human thinking.

Disadvantages: They are rather simple and can't resolve too complex problems.

#### 5. KNN

Advantages: Easy to follow algorithm. Because it's lazy approach, it computes only what is really necessary, saving a lot of time and resources

Disadvantages: They are rather simple and can't resolve complex problems.

#### 6. KMeans

Advantages: They perform a task that all the others can't do.

Disadvantages: I think that they can converge in a way that you don't want them to. All the others algorithms "care" about what you want from them, but this one doesn't.

# Chapter 9

## Your original code

```
from sklearn.preprocessing import LabelBinarizer
df = pd.read_csv("csv/titanic.csv")
df["Sex"] = np.where(df["Sex"] == "female", 1, 0)
X = df.drop(["Cabin", "Survived", "Name", "Title", "Embarked", "Ticket", "PassengerId"],
encoder = LabelBinarizer()
title = encoder.fit_transform(df["Title"])
X = pd.concat([X, pd.DataFrame(title)], axis=1)
print(X)
y = df['Survived']
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler((-1, 1))
X = sc.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)
X_train = torch.tensor(X_train).float()
X_test = torch.tensor(X_test).float()
y_train = torch.tensor(y_train.values).long()
y_test = torch.tensor(y_test.values).long()
trainDataset=Dataset(X_train, y_train)
trainLoader=DataLoader(dataset=trainDataset,
                        batch_size=32,
                        shuffle=True,
                        num_workers=1)
validationDataset=Dataset(X_test, y_test)
validationLoader=DataLoader(dataset=validationDataset,
                            batch_size=32,
                            shuffle=True,
                            num_workers=1)

class TitanicNN(nn.Module):
    def __init__(self):
        super(TitanicNN, self).__init__()

        #Sequential oferă o alternativă mai estetică a codului
        #Rețeaua noastră are 2 neuroni pentru output.
        #Unul va prezice probabilitatea pentru cazul afirmativ iar celălalt va prezice p
        self.sequential= nn.Sequential(
```



```

        nn.Linear(13,30),
        nn.Sigmoid(),
        nn.Linear(30, 60),
        nn.ReLU(),
        nn.Linear(60, 2)
    )

    def forward(self, x):
        return self.sequential(x)
optimizer = optim.SGD(net.parameters(), lr=0.01)
criterion = nn.CrossEntropyLoss()
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=5, verbose=True)
def train(epoch):
    net.train()
    losses=[]
    correct = 0
    for batch_idx, data in enumerate(trainLoader, 0):
        inputs, labels =data
        outputs = net(inputs)
        # Compute and print loss
        loss = criterion(outputs, labels)
        losses.append(loss.item())
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        pred = outputs.data.max(1, keepdim=True)[1]
        correct += pred.eq(labels.data.view_as(pred)).sum()
        #print(f"[Train Epoch: {epoch}, Batch: {batch_idx+1}, Loss: {loss.item()}")
    mean_loss=sum(losses)/len(losses)
    accuracy = 100. * correct/len(trainLoader.dataset)
    scheduler.step(mean_loss)
    train_losses.append(mean_loss)
    acc_train.append(accuracy)
    print(f"[TRAIN] Epoch: {epoch} Loss:{mean_loss}, Accuracy: {accuracy}%")
def validation():
    #Pune pe off flagurile setate in model.train()
    net.eval()
    test_loss=[]
    correct = 0

    with torch.no_grad():
        for batch_idx, data in enumerate(validationLoader, 0):
            inputs, labels = data

            output=net(inputs)
            loss= criterion(output, labels)
            test_loss.append(loss.item())

            pred = output.data.max(1, keepdim=True)[1]

```

```

        #Verificăm câte predicții sunt corecte și le însumăm numărul pentru a afla tot
        correct += pred.eq(labels.data.view_as(pred)).sum()
        current_correct=pred.eq(labels.data.view_as(pred)).sum()
    mean_loss=sum(test_loss)/len(test_loss)
    test_losses.append(mean_loss)
    accuracy = 100. * correct/len(validationLoader.dataset)
    print(f"[Validation set] Loss: {mean_loss}, Accuracy: {accuracy}%")
    accuracies.append(accuracy)
for epoch in range(500):
    train(epoch)
    validation()

train_url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url)
train = train.drop(['Name', 'Ticket', 'Cabin', 'Embarked'], axis=1)
abelEncoder = LabelEncoder()
labelEncoder.fit(train['Sex'])
train['Sex'] = labelEncoder.transform(train['Sex'])
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
kmeans = kmeans = KMeans(n_clusters=2, max_iter=1200, algorithm = 'full')
kmeans.fit(X_scaled)
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1

print(correct/len(X))

train_dataset = datasets.CIFAR10(root='./data/',
                                train=True, # represents the training section of the data
                                transform=transforms.ToTensor(),
                                download=True)
test_dataset = datasets.CIFAR10(root='./data/',
                                train=False, #validation set
                                transform=transforms.ToTensor())

print(test_dataset[0])
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                           batch_size=batch_size,
                                           shuffle=False)

def plot_images(images, labels):
    # normalise=True below shifts [-1,1] to [0,1]

```

```

img_grid = torchvision.utils.make_grid(images, nrow=4, normalize=True)
np_img = img_grid.numpy().transpose(1,2,0)
plt.imshow(np_img)

d_class2idx = train_dataset.class_to_idx
d_idx2class = dict(zip(d_class2idx.values(),d_class2idx.keys()))
train_iter = iter(train_loader)
images, labels = train_iter.next()
plot_images(images,labels)
print(' '.join('%5s' % d_idx2class[int(labels[j])])for j in range(len(images))))

def __init__(self):
    super(Net, self).__init__()
    '''
    nn.Conv2d(in_channels, out_channels, kernel_size)
    in_channels - number of channels of the input
                  - for greyscale images we have 1, or basically a single channel
                  - for RGB images we have 3 channels and for satellite data number ca

    out_channels - number of different filters applied to the input images and the r
                  - in our case is 10 at first conv, so we will have 10 different ima
    '''
    self.conv1 = nn.Conv2d(3, 10, kernel_size=5)
    self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
    self.mp = nn.MaxPool2d(2) # apply max pool with a window of 2X2
    self.fc = nn.Linear(500, 200) # final fully-connected layer
    self.fc2 = nn.Linear(200, 10) # final fully-connected layer

def forward(self, x):
    in_size = x.size(0)
    x = F.relu(self.mp(self.conv1(x)))
    x = F.relu(self.mp(self.conv2(x)))
    x = x.view(in_size, -1) # flatten the tensor
    x = self.fc(x)
    x = self.fc2(x)
    return x

def train(epoch):
    model.train()
    correct = 0
    losses=[]
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        losses.append(loss.item())
        optimizer.step()
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).cpu().sum()
    #if batch_idx % 10 == 0:

```

```

        #     print('Train Epoch: {} [{} / {}] ({:.0f}%) \t Loss: {:.6f}'.format(
        #         epoch, batch_idx * len(data), len(train_loader.dataset),
        #         100. * batch_idx / len(train_loader), loss.item()))
    loss = sum(losses) / len(losses)
    acc = correct * 100. / len(train_loader.dataset)
    train_loss.append(sum(losses) / len(losses))
    train_accuracy.append(correct * 100. / len(train_loader.dataset))
    print(f'\nTrain set: Average loss: {loss}, Accuracy: {acc}\n')
    model.eval()
    test_loss = 0
    correct = 0
    losses = []
    with torch.no_grad():
        for data, target in test_loader:
            output = model(data)
            # sum up batch loss
            loss = criterion(output, target)
            losses.append(loss.item())
            # get the index of the max log-probability
            pred = output.data.max(1, keepdim=True)[1]
            correct += pred.eq(target.data.view_as(pred)).cpu().sum()

    test_loss = sum(losses) / len(losses)
    validation_loss.append(test_loss)
    validation_accuracy.append(100. * correct / len(test_loader.dataset))
    print('\nTest set: Average loss: {:.4f}, Accuracy: {} / {} ({:.0f}%) \n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))
for epoch in range(1, 20):
    train(epoch)
    test()

import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics # Import scikit-learn metrics module for accuracy calculation
import numpy as np
from sklearn.preprocessing import LabelBinarizer

Read the data
pima = pd.read_csv("./csv/titanic.csv")
pima.head()
Age Cabin Embarked Fare Name Parch PassengerId Pclass Sex SibSp Survived Ticket
Title Family_Size
0 22.0 NaN S 7.2500 Braund, Mr. Owen Harris 0 1 3 male 1 0.0 A/5 21171 Mr 1
1 38.0 C85 C 71.2833 Cumings, Mrs. John Bradley (Florence Briggs Th... 0 2 1 female
1 1.0 PC 17599 Mrs 1
2 26.0 NaN S 7.9250 Heikkinen, Miss. Laina 0 3 3 female 0 1.0 STON/O2. 3101282 Miss
0
3 35.0 C123 S 53.1000 Futrelle, Mrs. Jacques Heath (Lily May Peel) 0 4 1 female
1 1.0 113803 Mrs 1
4 35.0 NaN S 8.0500 Allen, Mr. William Henry 0 5 3 male 0 0.0 373450 Mr 0

```

Feature selection

Un feature relevant este cel de Title.

Acesta are 6 vlaori distincte si este reprezentat printr-un string (Mr, Miss, Mr, Mrs, M

Pentru transformare acestui feature in valori numerice foloses un labelBinarizer. Acesta

La final concatenam matricea in care am encodat feature-ul de Title cu dataframe-ul

```
feature_cols = ['Age', 'Fare', 'Sex', 'Family_Size']
pima["Sex"] = np.where(pima["Sex"] == "female", 1, 0)
encoder = LabelBinarizer()
```

```
title = encoder.fit_transform(pima["Title"])
print(encoder.classes_)
title = pd.DataFrame(title)
title.columns = encoder.classes_
```

```
X= pd.concat([pima[feature_cols], title], axis=1) # Features
```

```
y = pima["Survived"] # Target variable
['Dr' 'Master' 'Miss' 'Mr' 'Mrs' 'Rev']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
clf = DecisionTreeClassifier(criterion="gini", max_depth=5)
```

```
# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)
```

```
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.7798507462686567
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = np.array(X.columns),class_names=
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('titanic.png')
Image(graph.create_png())
import numpy as np
```

```
class Perceptron(object):
```

```

def __init__(self, no_of_inputs, threshold=50, learning_rate=0.01):
    self.threshold = threshold
    self.learning_rate = learning_rate
    self.weights = np.zeros(no_of_inputs + 1)

def predict(self, inputs):
    summation = np.dot(inputs, self.weights[1:]) + self.weights[0]
    if summation > 0:
        activation = 1
    else:
        activation = 0
    return activation

def train(self, training_inputs, labels):
    for _ in range(self.threshold):
        for inputs, label in zip(training_inputs, labels):
            prediction = self.predict(inputs)
            self.weights[1:] += self.learning_rate * (label - prediction) * inputs
            self.weights[0] += self.learning_rate * (label - prediction)
training_inputs.append(np.array([1, 1, 1]))
training_inputs.append(np.array([1, 1, 0]))
training_inputs.append(np.array([1, 0, 1]))
training_inputs.append(np.array([1, 0, 0]))
training_inputs.append(np.array([0, 1, 1]))
training_inputs.append(np.array([0, 1, 0]))
training_inputs.append(np.array([0, 0, 1]))
training_inputs.append(np.array([0, 0, 0]))

labels = np.array([1, 1, 1, 1, 1, 1, 1, 0])

perceptron = Perceptron(3)
perceptron.train(training_inputs, labels)

weights = perceptron.weights
print(f"Computed weights{weights}")
inputs = np.array([1, 1, 1])
print(f"Predicted value for 1 or 1 or 1 = {perceptron.predict(inputs)}")
print(f"Computing {weights[0]} + {weights[1]} * {inputs[0]} + {weights[2]} * {inputs[2]}")

inputs = np.array([0, 1, 0])
print(f"Predicted value for 0 or 1 or 0 = {perceptron.predict(inputs)}")
print(f"Computing {weights[0]} + {weights[1]} * {inputs[0]} + {weights[2]} * {inputs[1]}")

inputs = np.array([1, 0, 0])
print(f"Predicted value for 1 or 0 or 0 = {perceptron.predict(inputs)}")
print(f"Computing {weights[0]} + {weights[1]} * {inputs[0]} + {weights[2]} * {inputs[1]}")

inputs = np.array([0, 0, 0])
print(f"Predicted value for 0 or 0 or 0 = {perceptron.predict(inputs)}")
print(f"Computing {weights[0]} + {weights[1]} * {inputs[0]} + {weights[2]} * {inputs[1]}")

```

# Bibliography

<https://pomegranate.readthedocs.io/en/stable/> - for pomegranate

[https://nces.ed.gov/programs/coe/pdf/coe\\_sa.pdf](https://nces.ed.gov/programs/coe/pdf/coe_sa.pdf) – *for working statistics*

<https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn>

<https://www.datacamp.com/community/tutorials/k-means-clustering-python>

[https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

<https://medium.com/@thomascourtz/19-line-line-by-line-python-perceptron-b6f113b161f3>

[https://github.com/AlexandruGH/AC\\_sisteme\\_inteligente\\_2019-2020](https://github.com/AlexandruGH/AC_sisteme_inteligente_2019-2020)

Intelligent Systems Group

