LESSON 14

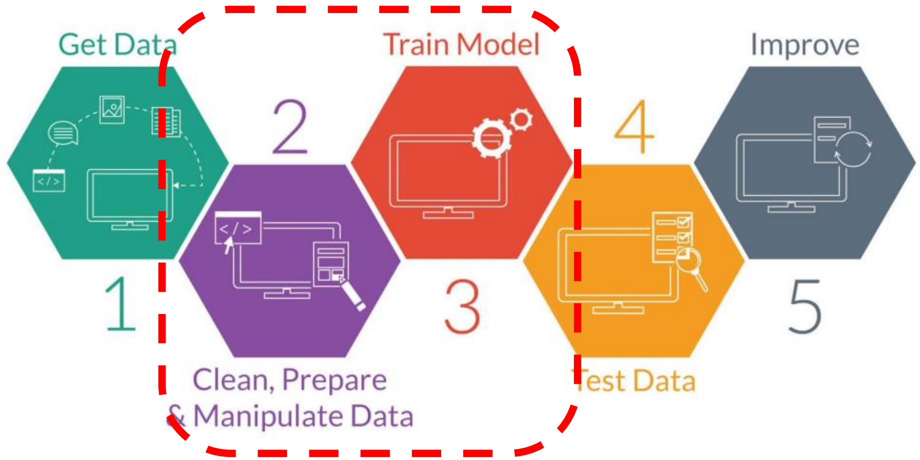
Basic datasets management,
Coding in Matlab and Colab,
Hard encoding




Lesson outline

- Lesson = Theory + LABORATORY
- Using reference dataset
  - The IRIS dataset
- Hard encoding
- **Main points**
- LABORATORY (in a separate file):
  Loading and basic dataset management
  - A Matlab example
  - A Colab example

Step 2 (and ready for Training..)



5.1								
3.5								
1.4								
0.2								

GOLDEN RULE #1: use a simple well-known dataset to start

- To test your
  - knowledge,
  - data processing tools,
  - Models
  - learning methods

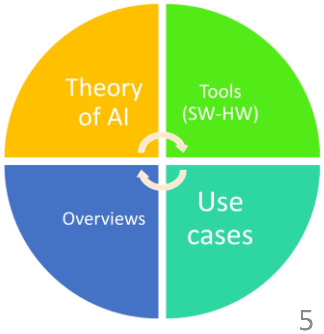
- Do not start with the actual dataset of your application!!

- Start with known and easy dataset with known outputs
  - Plots, accuracy, results are well documented, you can compare your outputs with the literature




Toolboxes
The Iris dataset

Start with something known
every time you test a new tool.



Example: iris flower

- Classify iris flowers into 3 classes:
  1. Setosa
  2. Versicolor
  3. Virginica
- 4 features:
  1. Sepal length in cm
  2. Sepal width in cm
  3. Petal length in cm
  4. Petal width in cm
- 150 samples

"a genius who almost single-handedly created the foundations for modern statistical science"

Ronald Fisher

Data

Iris dataset

- Introduced by the British statistician and biologist Ronald Fisher in his 1936 paper “The use of multiple measurements in taxonomic problems”
- Collected data to quantify the morphologic variation of Iris flowers of three related species

Classes

Iris setosa

Iris versicolor

Iris virginica

Features

Iris dataset

- Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters
- Based on the combination of these four features, Fisher developed a model to distinguish the species from each other.

The iris dataset

Datafile Name: Fisher's Iris

Datafile Subjects: Agriculture , Famous datasets

Story Names: Fisher's Irises

Reference: Fisher, R. A. (1936). The Use of Multiple Measurements in Axonomic Problems. Annals of Eugenics 7, 179-188.

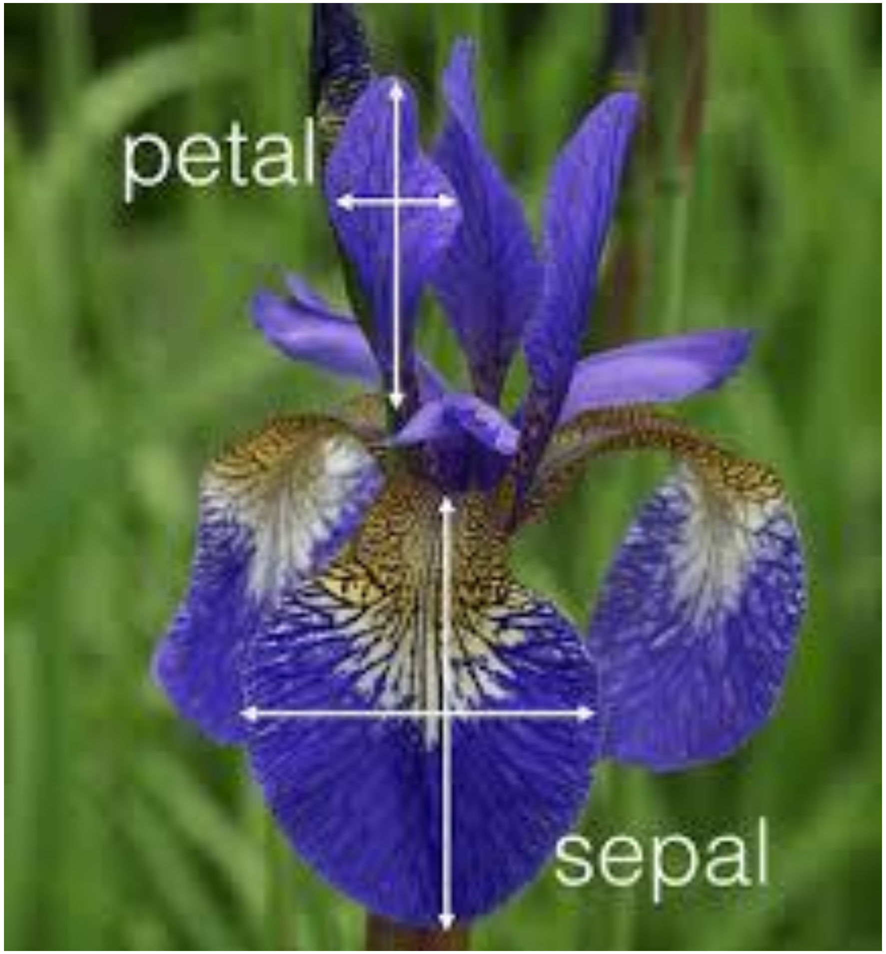
Authorization: free use

Description: This is a dataset made famous by Fisher, who used it to illustrate principles of discriminant analysis. It contains 6 variables with 150 observations.

Number of cases: 150

Variable Names:

1. Species_No: Flower species as a code
2. Species_Name: Species name
3. **Petal_Width: Petal Width**
4. **Petal_Length: Petal Length**
5. **Sepal_Width: Sepal Width**
6. **Sepal_Length: Sepal Length**



The Iris dataset raw data

Iris Setosa | Iris Virginica | Iris Versicolor

Meaning of the 3 classes

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

⋮

Y = FUNC(X) that’s it!!!

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

- For every dataset in machine learning or toolbox, is all about to create X and Y

Sample, sample, ……

- X

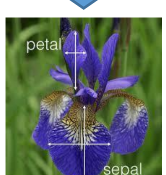
Features

5.1								
3.5								
1.4								
0.2								

Class ID, or value to be learnt

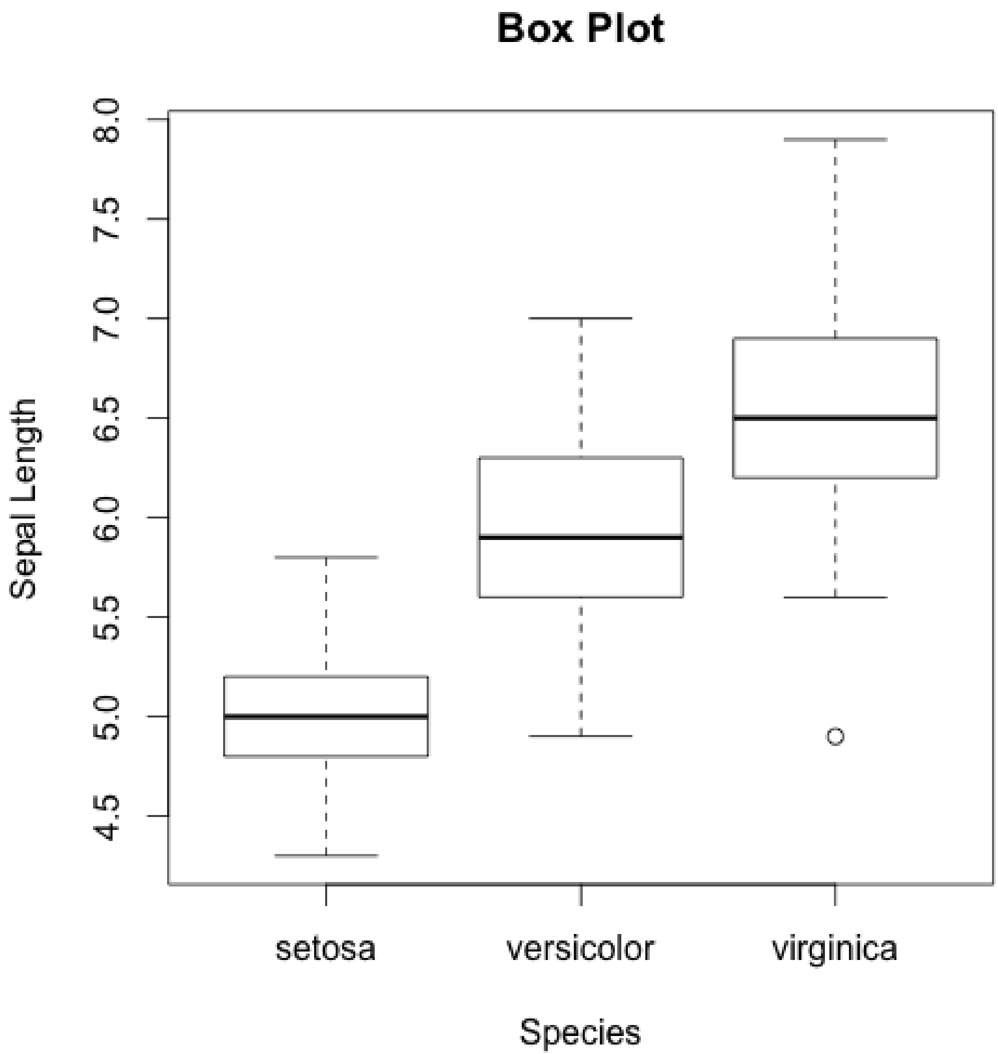
- Y

1	**1**	**3**	**2**					

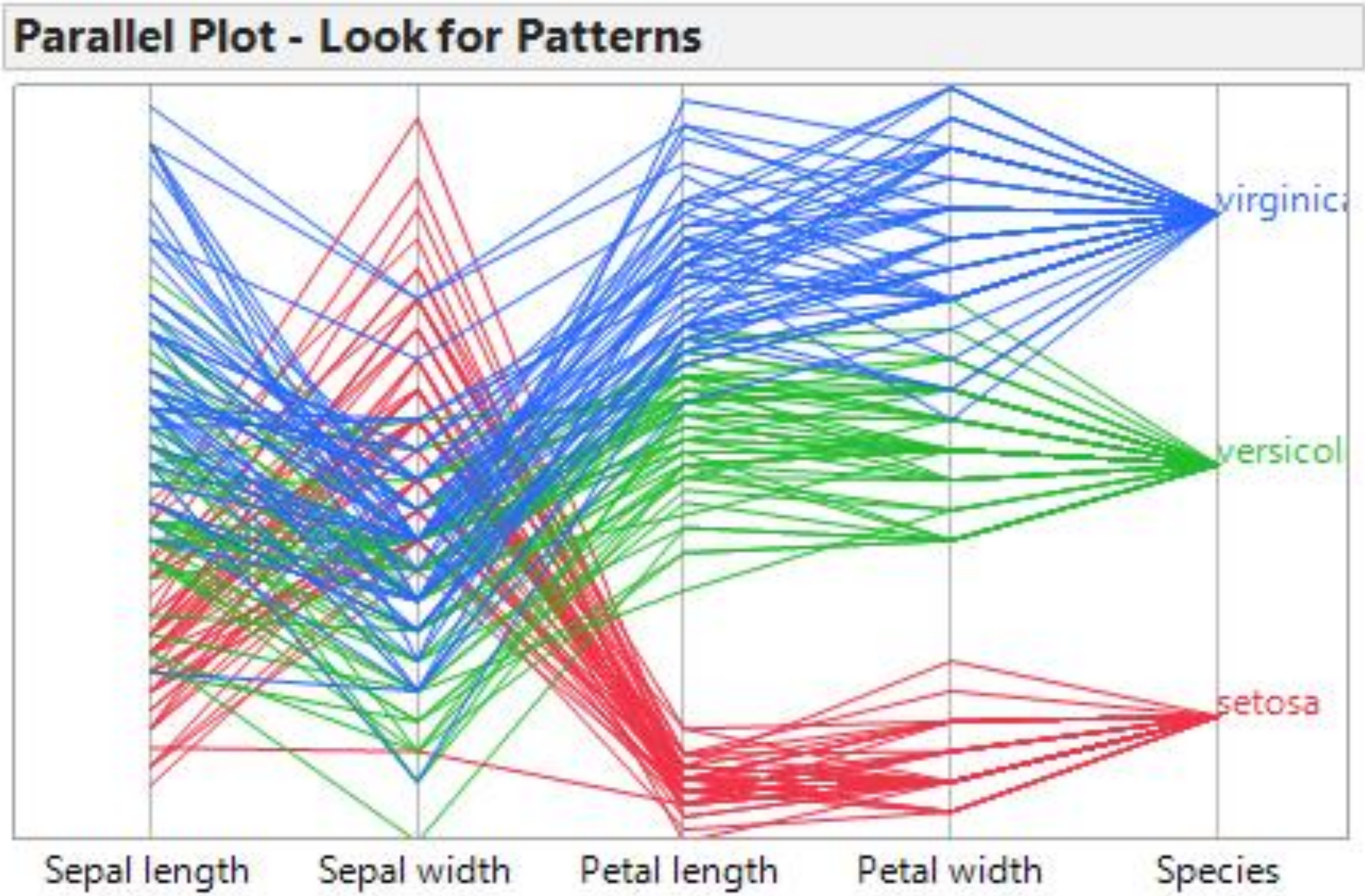


1 = «setosa»

The Fischer Iris boxplots

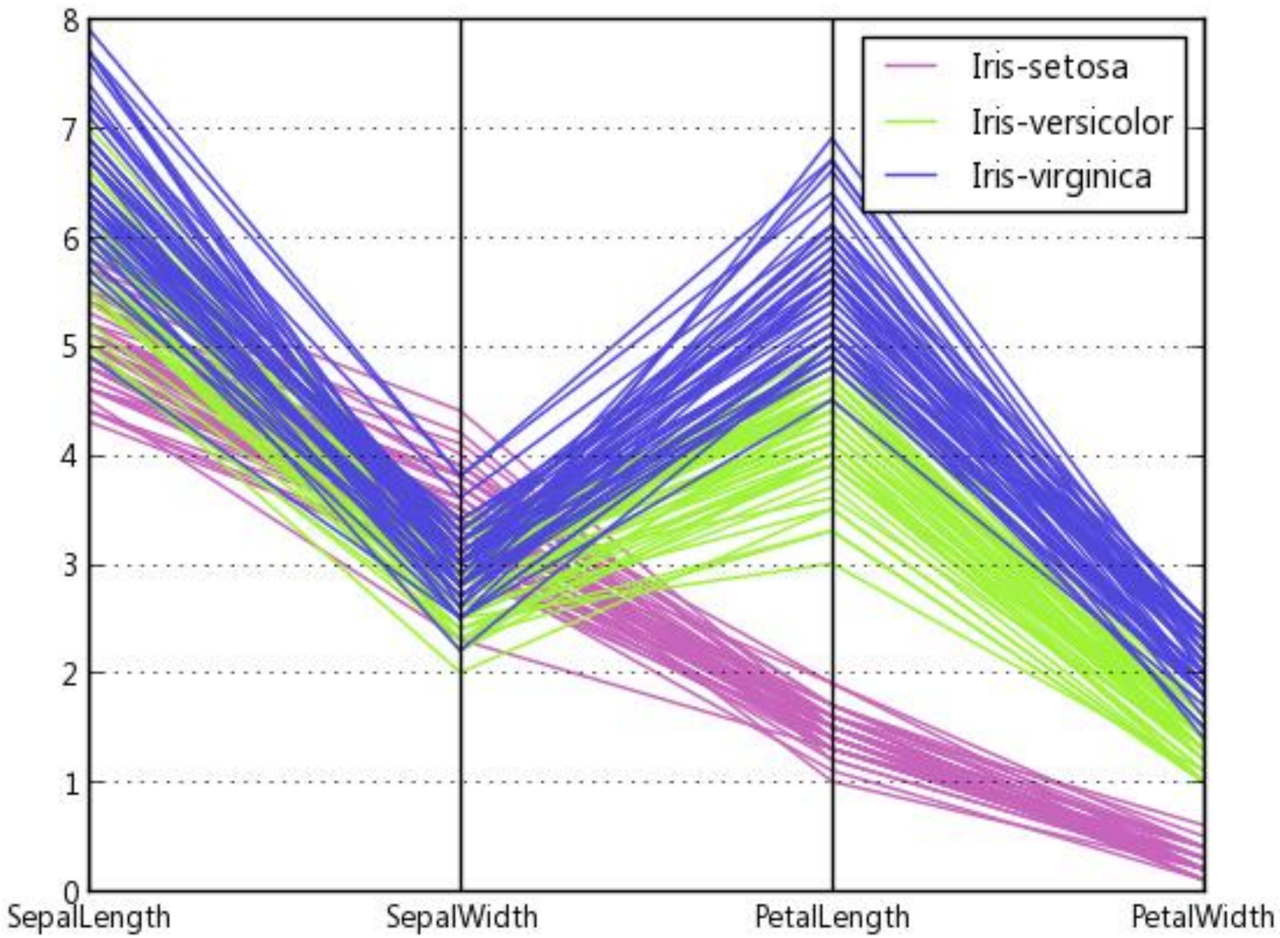


Iris Data – Parallel Plots



The reality….

Note:
you need
same units
or a similar
scale here!!

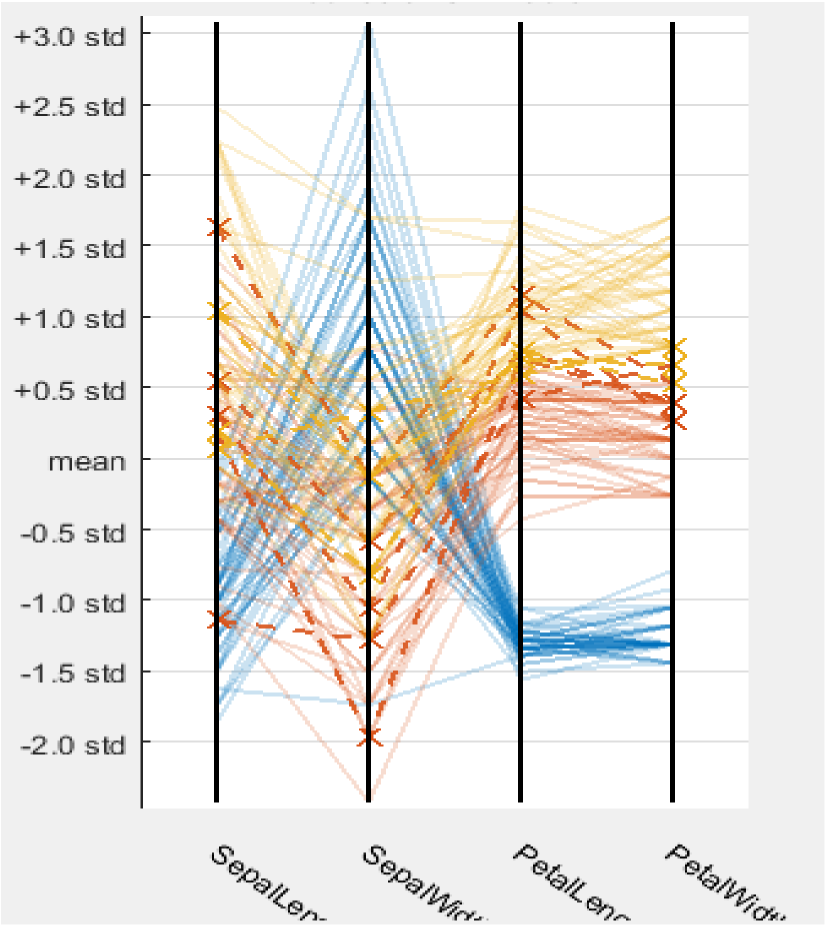


Harmonization

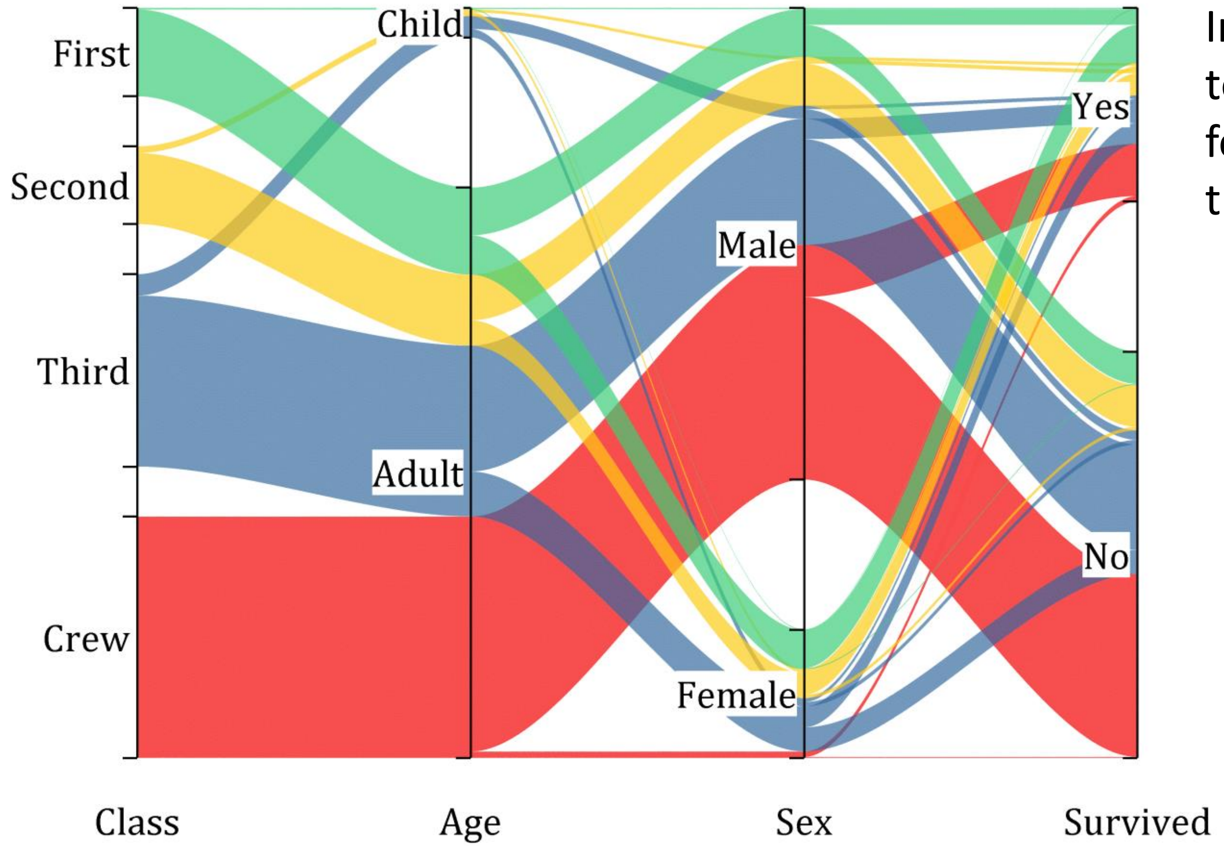
Using the **mean** and the **std** fractions you can harmonize the values

(z score)

mean of all of the values is 0 and the standard deviation is 1

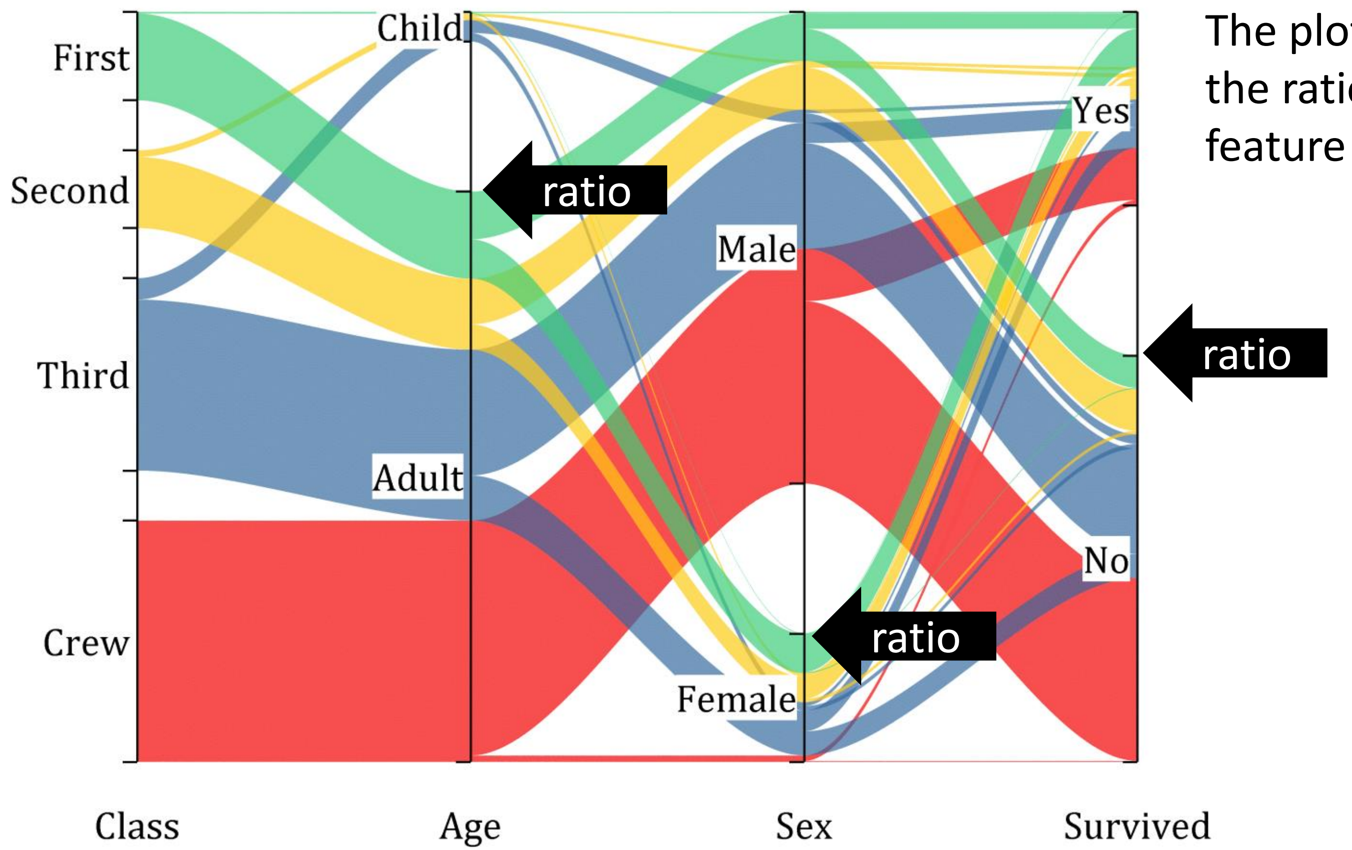


EDA, Parallel Plot: a more complex case



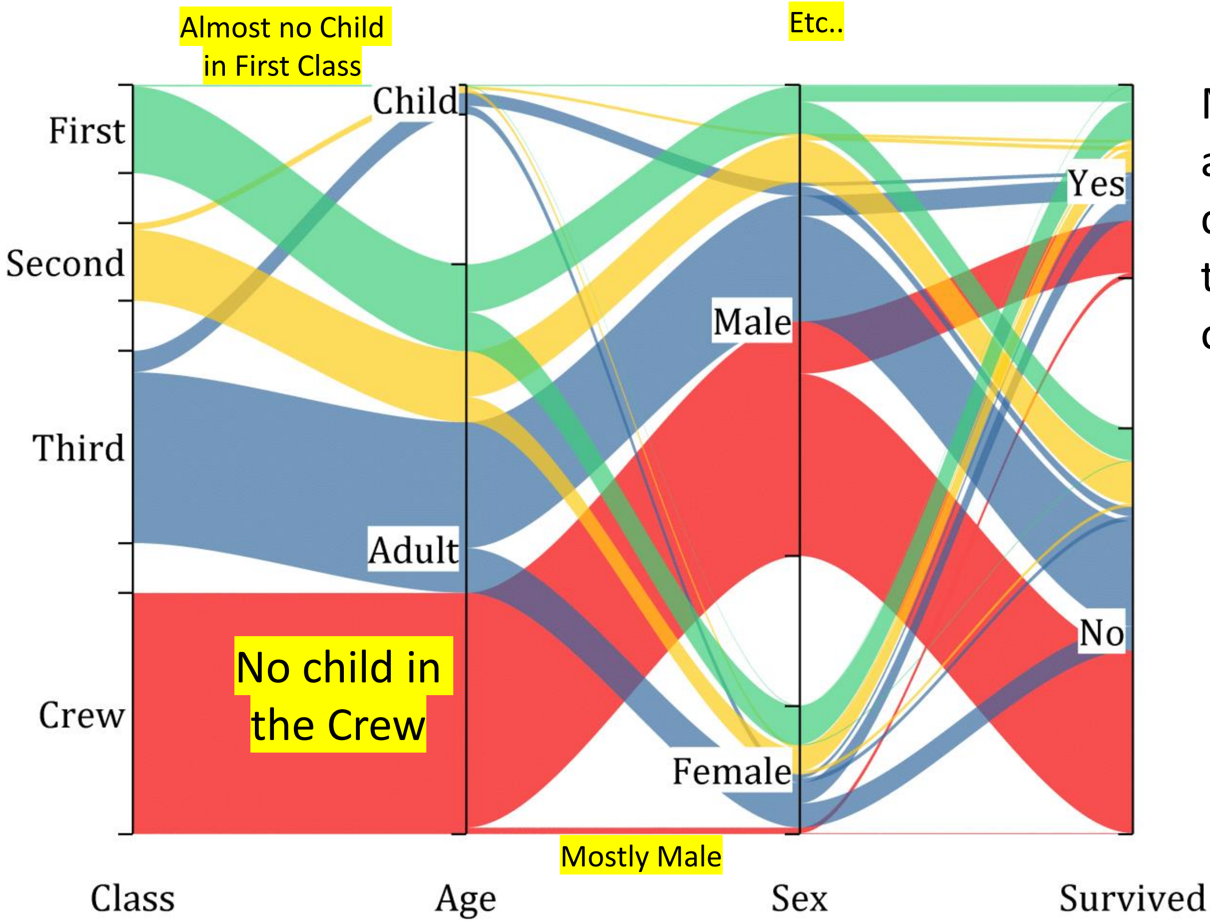
In this case it is harder to understand which feature will better help the classification

EDA, Parallel Plot: a more complex case



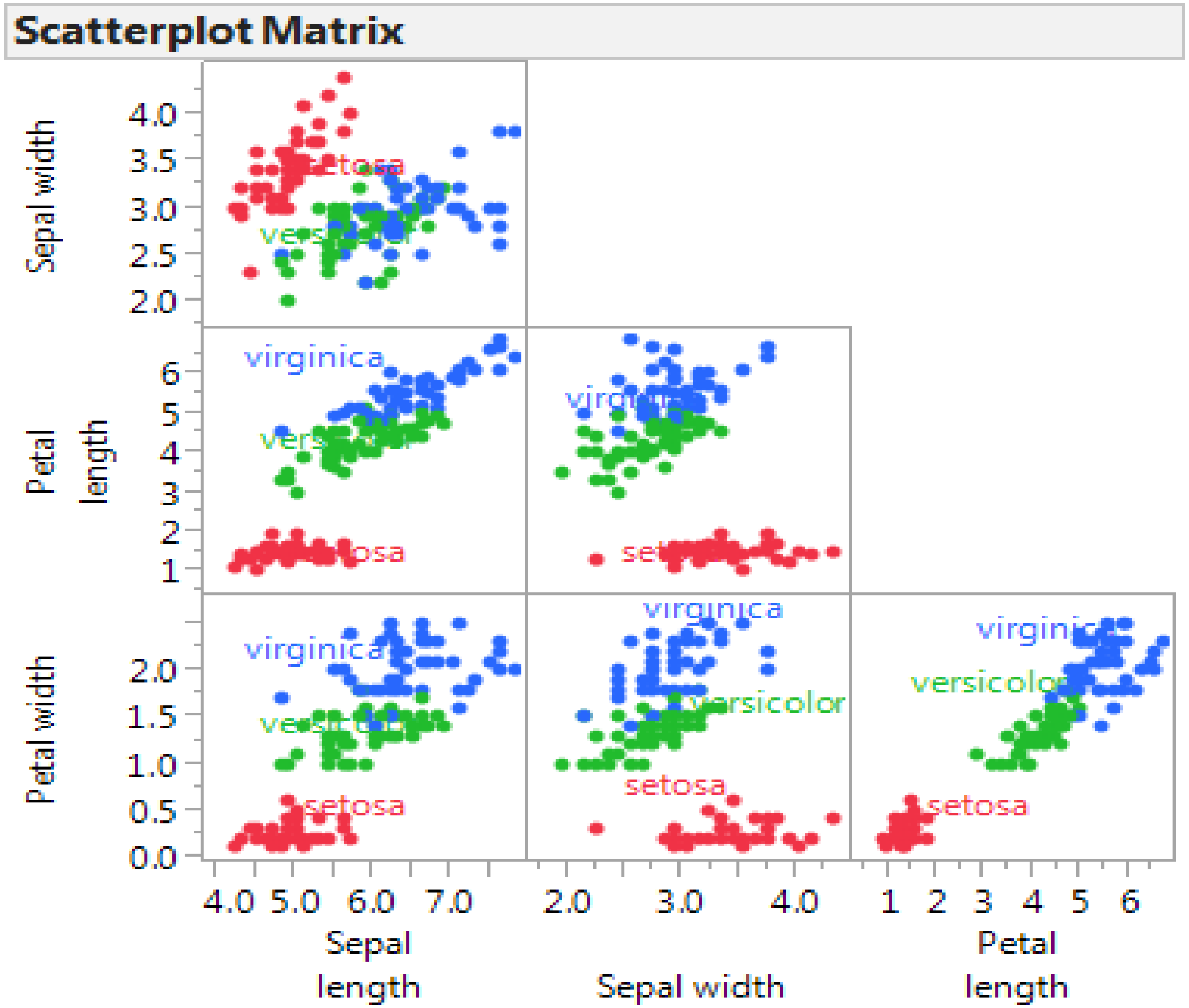
The plot is showing also the ratio between the feature (crisp) values

EDA, Parallel Plot: a more complex case



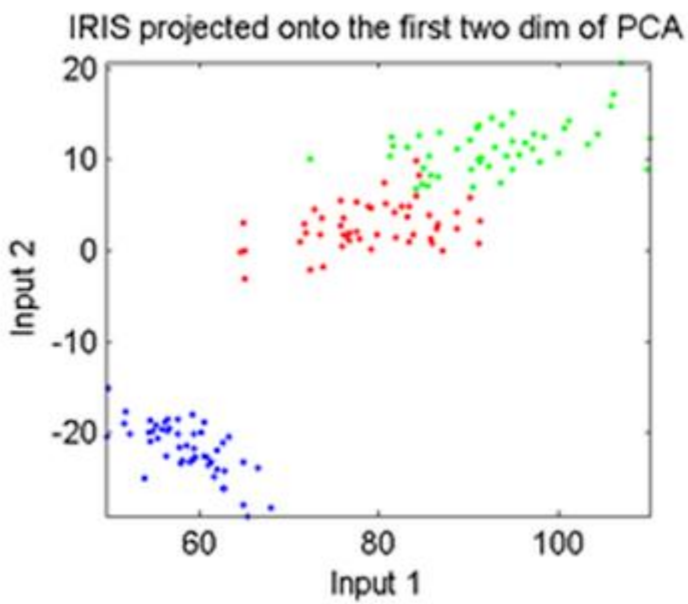
Meaningful information and valuable insights can be extracted from this plot just by observation

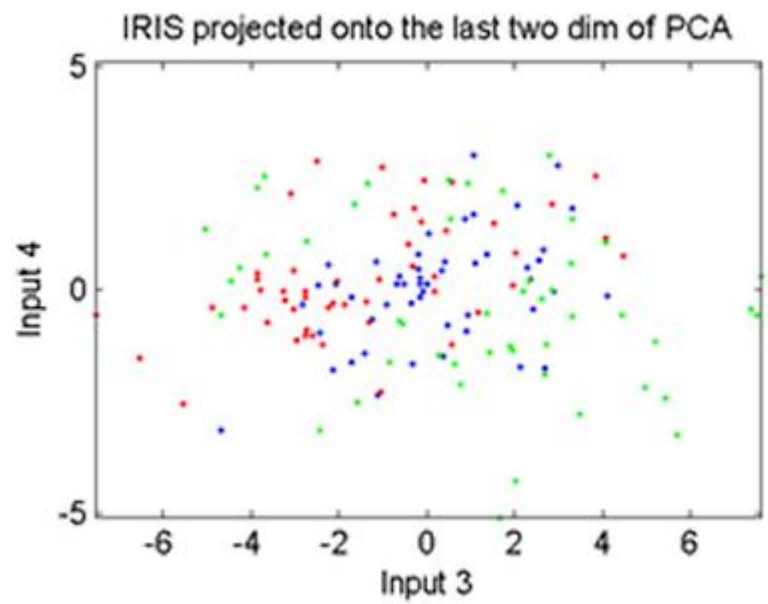
Iris Data - Correlation Plots



Principal Component Analysis on the Iris Dataset

The PCA is an effective method for **reducing** a dataset's **dimensionality** while keeping spatial characteristics as much as possible (no need for labeled data)





Data harmonization

- No missing data
- No labelling errors
- No scaling needed

- WE WILL USE THE FISCHER IRIS DATESET AS A DEBUG TOOLS!!

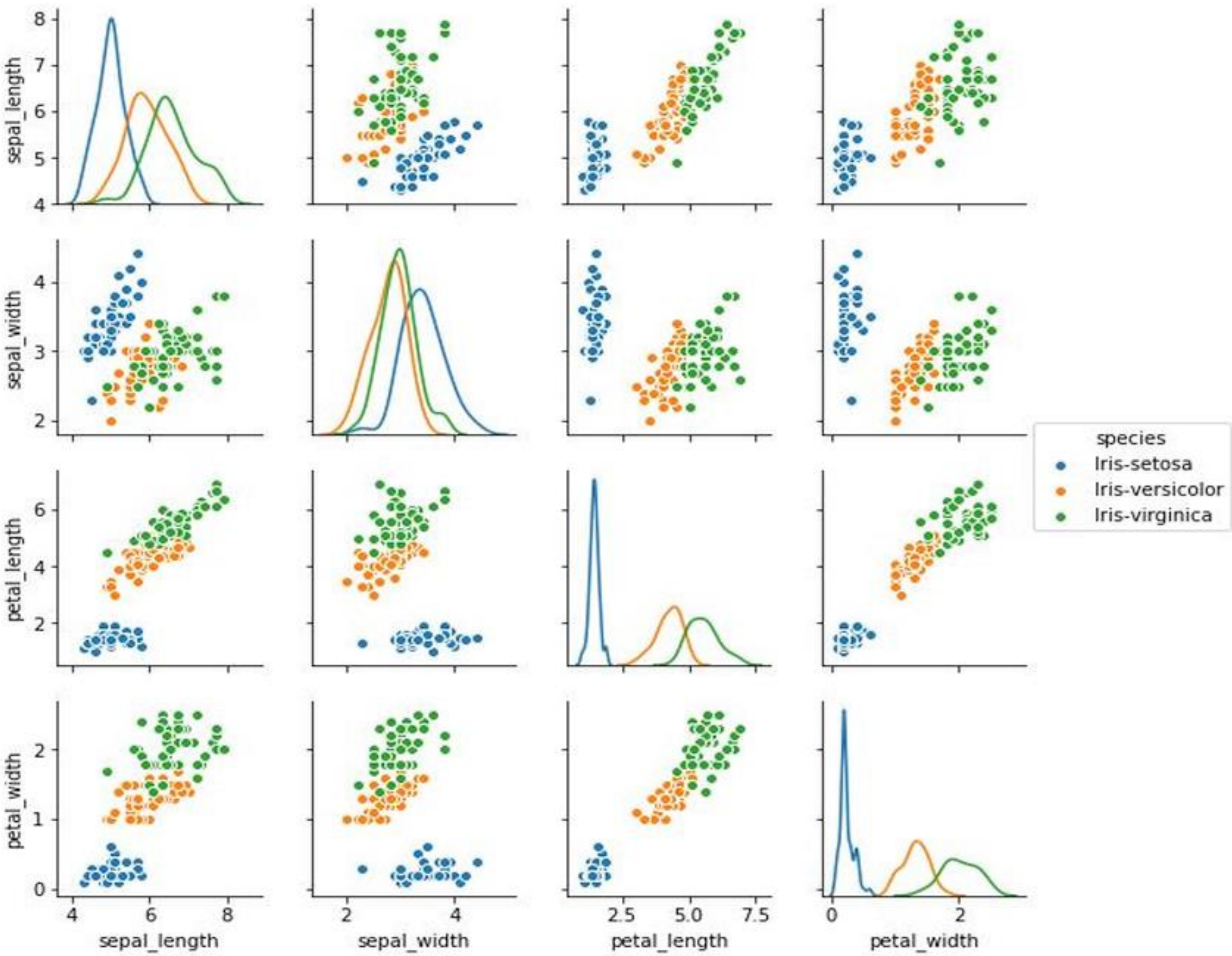
- Let’s use the Google **TensorFlow** notation in the next examples

- **TensorFlow** is a free and open-source software library for dataflow and differentiable programming across a range of tasks from Google.

```
#Plot the dataset seaborn.pairplot(dataset, hue="species", size=2, diag_kind="kde") plt.show()
```

Fabio Scotti - Università degli Studi di Milano

The (pair)plot



Getting ready to start learning

1. Encoding
   1. Integer encoding
   2. One-Hot encoding

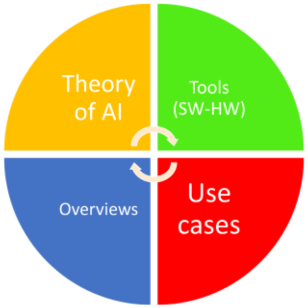
2. Prepare the Input Features
3. Split Train Test
4. **TensorFlow** Neural Network




THEORY

Integer encoding
one-hot encoding

Learning can be
affected by the way you <u>list the classes</u>



Integer encoding

- Each unique category value is assigned an integer value. For example,
  - "*red*" is 1,
  - "*green*" is 2,
  - and "*blue*" is 3.
- This is called a label encoding or integer encoding and is easily reversible.
- For some variables, this may be enough.
  - The integer values have a natural ordered relationship between each other, and machine learning algorithms may be able to understand and harness this relationship.

One-Hot Encoding

- For categorical variables where no such ordinal relationship exists, the integer encoding is not enough.
- Using integer encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

INTEGER ENCODING

"red" → 1
"green" → 2
and *"blue"* → 3

ONE-HOT ENCODING

1	red,	green,	blue
2	1,	0,	0
3	0,	1,	0
4	0,	0,	1

green-red = 1 blu-red=2 (!)

One Hot Encoding

```
from sklearn.preprocessing import LabelBinarizer
species_lb = LabelBinarizer() Y =
species_lb.fit_transform(dataset.species.values)
```

LabelBinarizer

- We must now one-hot encode the species column from text into a vector that our machine learning algorithm will understand.
- This is why we use one hot encoder to perform “binarization” of the category and include it as a feature to train the model.

CompanyName	Categoricalvalue	Price
VW	1	20
Acura	2	10011
Honda	3	50000
Honda	3	10000

One hot encoding

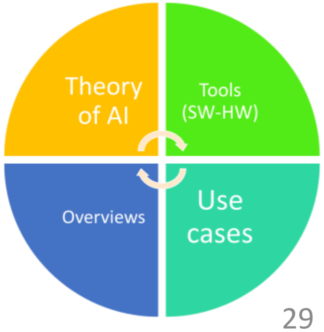
VW	Acura	Honda	Price
1	0	0	20000
0	1	0	10011
0	0	1	50000
0	0	1	10000




Toolboxes

Coding Normalization and Train/Test partitions

The basic tools to start



One Hot Encoding in Matlab

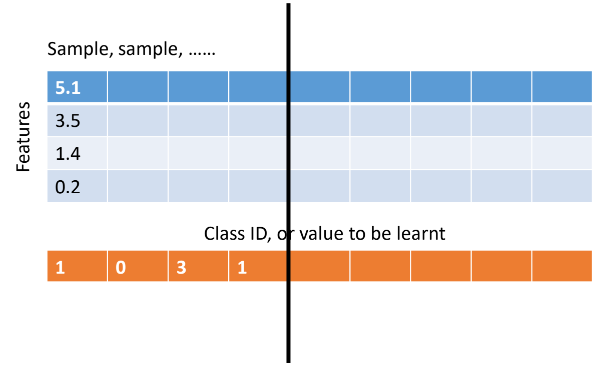
```
labels = ["red"; "blue"; "red"; "green"; "yellow"; "blue"];  % 6 elements
labels = categorical(labels);    % 4 only different labels
categories(labels)
ans = 4x1 cell
    {'blue'  }
    {'green' }
    {'red'   }
    {'yellow'}
labels = onehotencode(labels,2)
labels = 6×4

     0     0     1     0
     1     0     0     0
     0     0     1     0
     0     1     0     0
     0     0     0     1
     1     0     0     0
```

Prepare the Input Features

- To improve gradient descent, we will normalize the values utilizing the normalize class
- X_data variable will contain our normalized features we will use to train our neural network.

```
from sklearn.preprocessing import normalize
FEATURES = dataset.columns[0:4]
X_data = dataset[FEATURES].as_matrix()  #for compatib.
X_data = normalize(X_data)
```

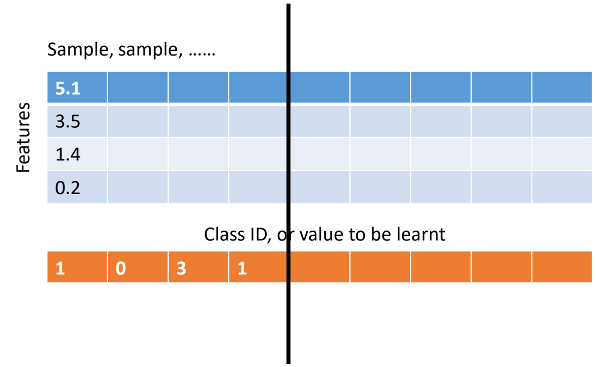
Split Train Test



- The 2 vectors, X_data and y, contains the data needed to train a neural network with the engine (like Tensorflow).
- Now split this data into a training and a test set in order to prevent overfitting and be able to obtain a better benchmark of our network's performance.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, Y, test_size=0.3,
random_state=1)
```

Split Train Test




```
import numpy as np
from sklearn.model_selection import train_test_split
```

```
[53] X1 = np.arange(10)
     X = np.reshape(X1, (5, 2))
     X
```

```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

```
[54] y = range(5)
     list(y)
```

```
[0, 1, 2, 3, 4]
```

```
[55] X_train, X_test, y_train, y_test = train_test_split(
...      X, y, test_size=0.33, random_state=1)
```

```
[56] X_train
```

```
array([[8, 9],
       [0, 1],
       [6, 7]])
```

```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

```
[57] y_train
```

```
[4, 0, 3]
```

```
X_test
```

```
array([[4, 5],
       [2, 3]])
```

Next step is learning

- As you can see in the exercitation in Maltlab and Colab we are now ready to create the first classifiers!

Main points

- The FISHER IRIS dataset as “debug tool”
- One-hot encoding
- Creating the matrices for learning
- Statistical analysis of the data
- Example relevant of plots and charts for ML
- Statistical analysis in needed before to start processing
- Data Visualization is quite useful….