LESSON 14 LAB

Basic datasets management,

Coding in Colab and Matlab



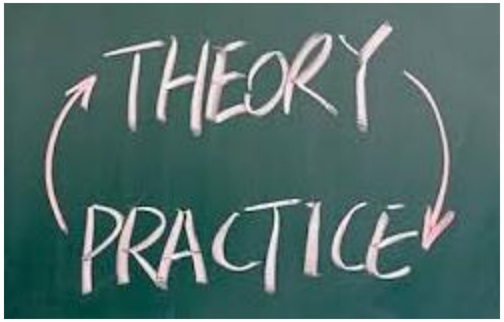

Laboratory outline

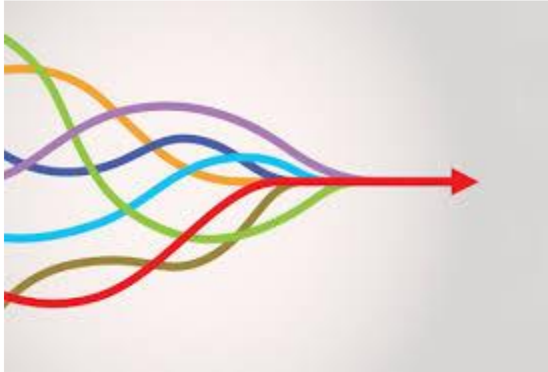
- LABORATORY (in a separate file):
  Loading and basic dataset management
  - Colab example
  - Matalab example

- The laboratory is based on the IRIS dataset
- **Please refer to the Lesson #14 for the description of the IRIS dataset**

What in the exam?

- In the exam there will be no (exact) coding
- Nevertheless, is it important to study the application of the concept of the theory in practical cases and examples in order to better understand the notions





Knowledge

- You will be asked to comment some example of code

LAB CLASS

COLAB

- Environment preparation
- Data loading

MATLAB

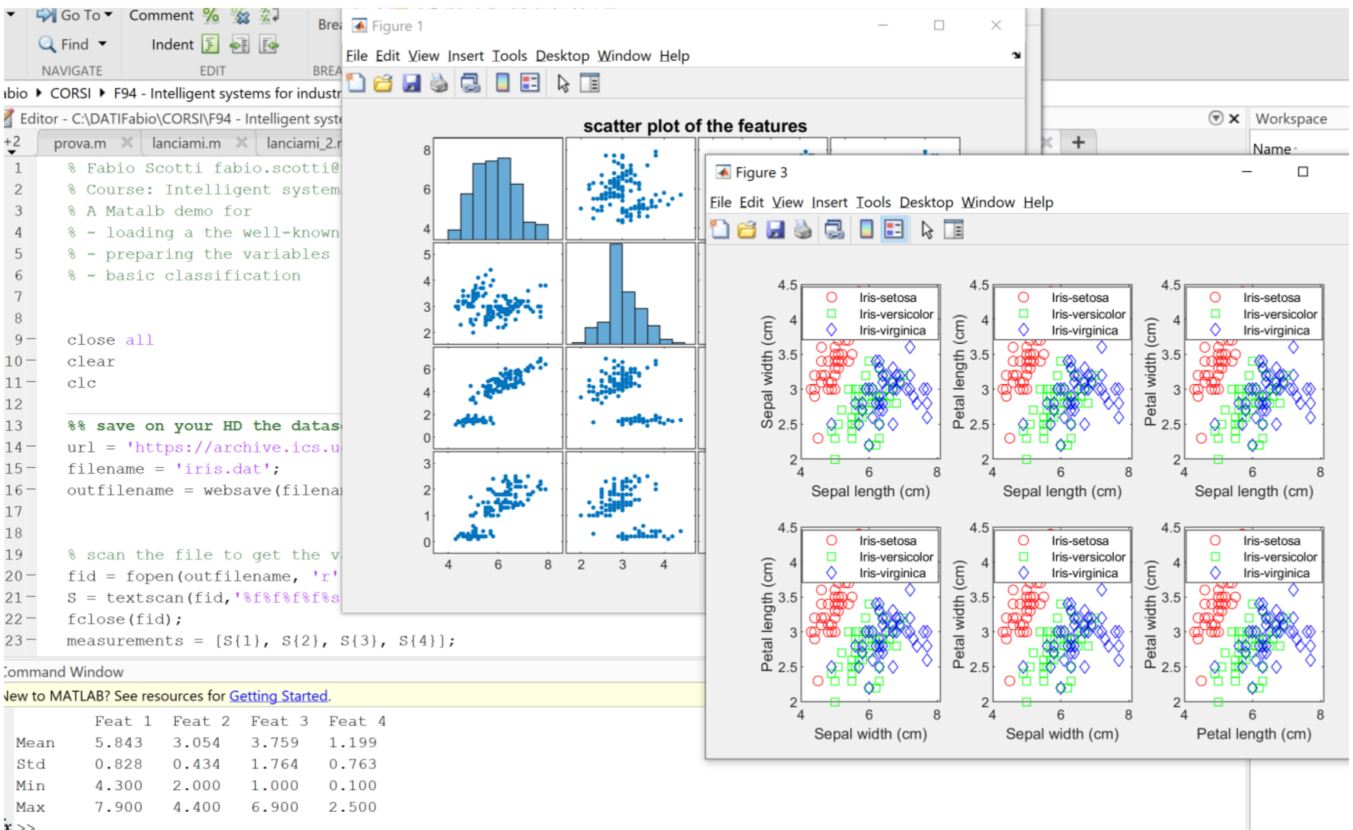
- Load the dataset
- Create the data structures (X and Y)

- Preliminary data analysis
- Plot the data
- Normalization
- Classification example

Best thing to do… (Matlab)

- Download the example and let them run!

laboratory_matlab_IRIS_Dataset.m



The example is commented… explore the code and try to modify it and/or add new plots

Best thing to do… (Colab)

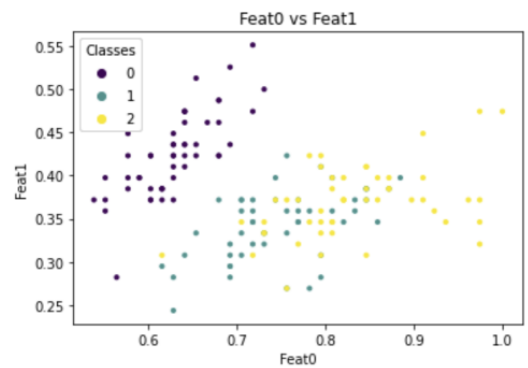
- Download the code laboratory_COLAB_IRIS_Dataset.ipynb
  → upload it to your Gdrive https://colab.research.google.com/
  → open it with Colab and let it run

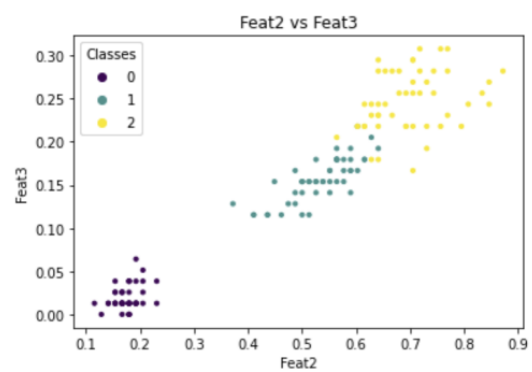
  (use Google Chrome – with login)

Plot features and classes

```
[41] import matplotlib.pyplot as plt
     # select 2 features to plot
     feat0 = xNumpy_Norm[:,0]
     feat1 = xNumpy_Norm[:,1]
     # params
     area = np.pi*3
     # Plot
     fig1, ax1 = plt.subplots()
     scatter1 = ax1.scatter(feat0, feat1, s=area, c=yNumpy)
     plt.title('Feat0 vs Feat1')
     plt.xlabel('Feat0')
     plt.ylabel('Feat1')
     legend1 = ax1.legend(*scatter1.legend_elements(),
                         loc="upper left", title="Classes")
     ax1.add_artist(legend1)
     plt.show()

     # select 2 features to plot
     feat2 = xNumpy_Norm[:,2]
     feat3 = xNumpy_Norm[:,3]
     # params
     area = np.pi*3
     # Plot
     fig2, ax2 = plt.subplots()
     scatter2 = ax2.scatter(feat2, feat3, s=area, c=yNumpy)
```



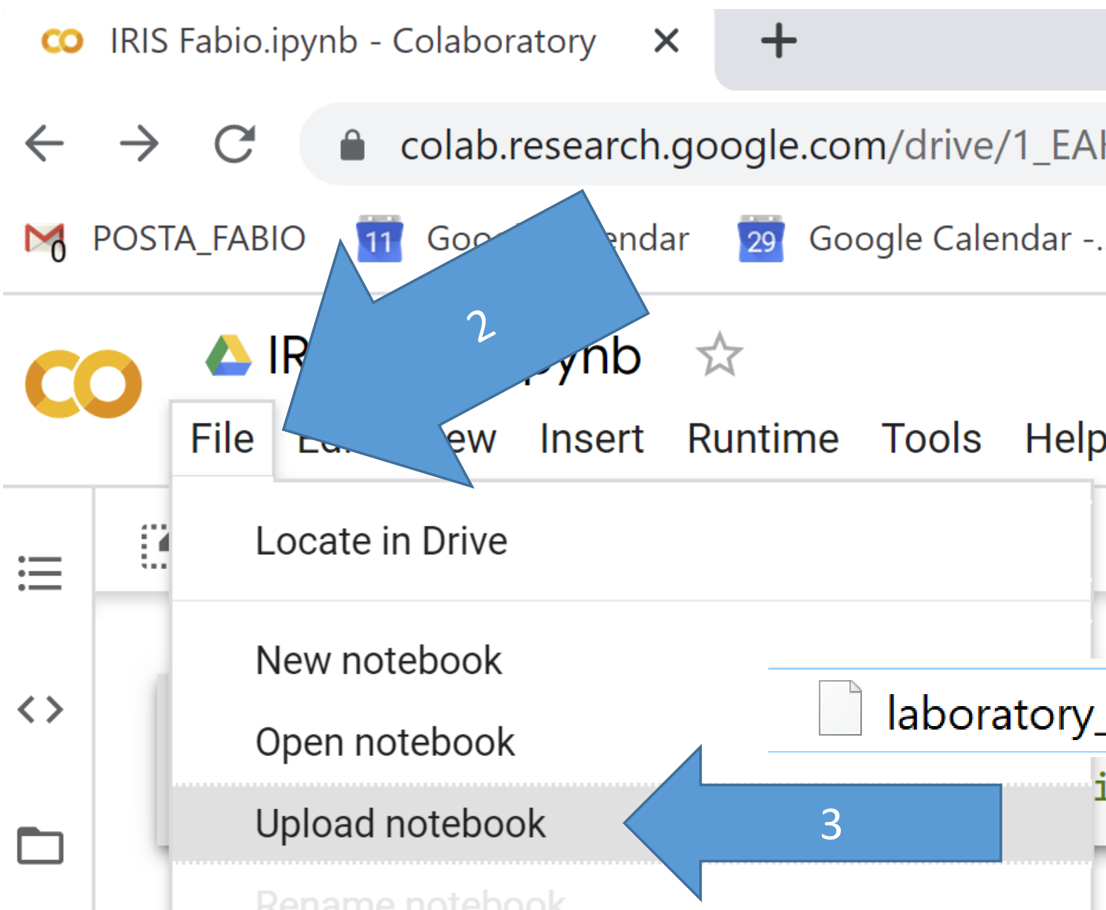


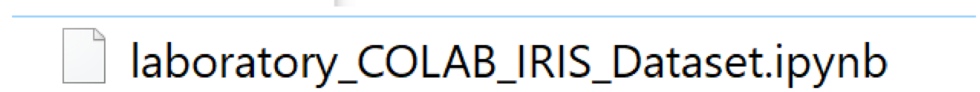
The examples are commented

Explore them and try to modify them or to add new plots

Upload your *.ipynb

https://colab.research.google.com/





An IPYNB file is a notebook document used by Jupyter Notebook, an interactive computational environment designed to help designers to work with the Python language and their data.

Let’s it run

laboratory_COLAB_IRIS_Dataset.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 10:39 PM

+ Code + Text

IRIS data loading, preparation, classification

Fabio Scotti fabio.scotti@unimi.it

Course: Intelligent systems for industry, supply chain and environment

A Matalb demo for

- loading a the IRIS dataset
- preparing the variables

[ ]

Import libraries

1) Read the comments

```
[ ] import pandas as pd #Python Data Analysis Library
    import numpy as np #Python Scientific Library
    from matplotlib import pyplot as plt
```

Load data

1) Read the comments

```
[ ] url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
    new_names = ['sepal_length','sepal_width','petal_length','petal_width','iris_class']
    dataset = pd.read_csv(url, names=new_names, skiprows=0, delimiter=',')
    dataset.info()
```

Import libraries

```
[ ] import pandas
    import numpy a
    from matplotli
```

2

Import libraries

```
[ ] import pandas as pd #Python Data Analysis Library
    import numpy as np #Python Scientific Library
    from matplotlib import pyplot as plt
```

Clik on the [ ]
to run the cell

COLAB_IRIS_Dataset.ipynb ☆

nsert Runtime Tools Help

Run all Ctrl+F9

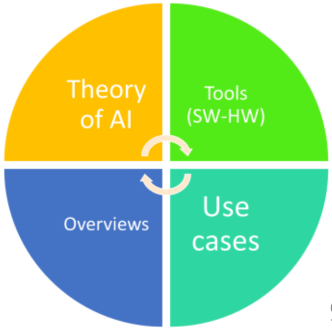
Run before Ctrl+F8

Or «Run all»




Toolboxes Loading, EDA/plotting in Python/Colab

And let’s have a look about **Matplotlib**



Environment preparation

Panda: libraries for dataset preprocessing

Numpy: libraries for array manipulation

Import libraries

```
[74] import pandas as pd #Python Data Analysis Library
     import numpy as np #Python Scientific Library
```

NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/ (NUM-pee)) 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.

Data loading

Iris dataset

- Use the panda library to read the CSV

Load data

```
[75] url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
     new_names = ['sepal_length','sepal_width','petal_length','petal_width','iris_class']
     dataset = pd.read_csv(url, names=new_names, skiprows=0, delimiter=',')
     dataset.info()
```

Data loading

`dataset.info()` gives us organized information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   iris_class    150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Data loading

dataset.head(n) to view the first n samples

	sepal_length	sepal_width	petal_length	petal_width	iris_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Data preprocessing

First, we need to separate the features from the classes

- Different preprocessing for features and classes (Necessary? YES! To avoid Data Leakage!!!)

```
[77] y = dataset['iris_class']
     x = dataset.drop(['iris_class'], axis=1)

     print ("dataset : ",dataset.shape)
     print ("x : ",x.shape)
     print ("y : ",y.shape)
```

As size() in Matlab

Data preprocessing

First, we need to separate the features from the classes

- Feature matrix has 4 columns (the features)
- Target vector has 1 column (the class)

```
dataset :  (150, 5)
x :  (150, 4)
y :  (150,)
```

Data preprocessing

Use the numpy library to convert data into matrix format

- Ease of access to portions of the matrices

```
[78] xNumpy = x.to_numpy()
     yNumpy = y.to_numpy()
```

Data preprocessing

We can access a single feature or a single samples

- Read a specific feature: matrix slicing by column

```
[79] feature_to_extract = 1
     feature = xNumpy[:,feature_to_extract]
     print(feature)
```

Data preprocessing

We can access a single feature or a single samples

- Read a specific feature: matrix slicing by column

```
[79] feature_to_extract = 1
     feature = xNumpy[:,feature_to_extract]
     print(feature)
```

```
[3.5 3.  3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.  3.  4.  4.4 3.9 3.5
 3.8 3.8 3.4 3.7 3.6 3.3 3.4 3.  3.4 3.5 3.4 3.2 3.1 3.4 4.1 4.2 3.1 3.2
 3.5 3.1 3.  3.4 3.5 2.3 3.2 3.5 3.8 3.  3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3
 2.8 2.8 3.3 2.4 2.9 2.7 2.  3.  2.2 2.9 2.9 3.1 3.  2.7 2.2 2.5 3.2 2.8
 2.5 2.8 2.9 3.  2.8 3.  2.9 2.6 2.4 2.4 2.7 2.7 3.  3.4 3.1 2.3 3.  2.5
 2.6 3.  2.6 2.3 2.7 3.  2.9 2.9 2.5 2.8 3.3 2.7 3.  2.9 3.  3.  2.5 2.9
 2.5 3.6 3.2 2.7 3.  2.5 2.8 3.2 3.  3.8 2.6 2.2 3.2 2.8 2.8 2.7 3.3 3.2
 2.8 3.  2.8 3.  2.8 3.8 2.8 2.8 2.6 3.  3.4 3.1 3.  3.1 3.1 3.1 2.7 3.2
 3.3 3.  2.5 3.  3.4 3.  ]
```

Data preprocessing

	sepal_length	sepal_width	petal_length	petal_width	iris_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

We can access
a single feature
or a single samples

- Read a specific feature: matrix slicing by column

```
[79] feature_to_extract = 1
     feature = xNumpy[:,feature_to_extract]
     print(feature)
```

Debugging!!

```
[3.5 3.  3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.  3.  4.  4.4 3.9 3.5
                                                         4.1 4.2 3.1 3.2
 3.5 3.1 3.  3.4 3.5 2.3 3.2 3.5 3.8 3.  3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3
 2.8 2.8 3.3 2.4 2.9 2.7 2.  3.  2.2 2.9 2.9 3.1 3.  2.7 2.2 2.5 3.2 2.8
 2.5 2.8 2.9 3.  2.8 3.  2.9 2.6 2.4 2.4 2.7 2.7 3.  3.4 3.1 2.3 3.  2.5
 2.6 3.  2.6 2.3 2.7 3.  2.9 2.9 2.5 2.8 3.3 2.7 3.  2.9 3.  3.  2.5 2.9
 2.5 3.6 3.2 2.7 3.  2.5 2.8 3.2 3.  3.8 2.6 2.2 3.2 2.8 2.8 2.7 3.3 3.2
 2.8 3.  2.8 3.  2.8 3.8 2.8 2.8 2.6 3.  3.4 3.1 3.  3.1 3.1 3.1 2.7 3.2
 3.3 3.  2.5 3.  3.4 3.  ]
```

	sepal_length	sepal_width	petal_length	petal_width	iris_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Data preprocessing

We can access a single feature or a single samples

- Read a specific sample: matrix slicing by row

```
[80] sample_to_extract = 1
     sample = xNumpy[sample_to_extract,:]
     print(sample)
```

Data preprocessing

We can access a single feature or a single samples

- Read a specific sample: matrix slicing by row

```
[80] sample_to_extract = 1
     sample = xNumpy[sample_to_extract,:]
     print(sample)
```

Debugging!!

```
[4.9 3.  1.4 0.2]
```

	sepal_length	sepal_width	petal_length	petal_width	iris_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Data preprocessin

	sepal_length	sepal_width	petal_length	petal_width	iris_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	[illegible]	[illegible]	[illegible]	[illegible]	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

We can access a
single feature
or a single samples

- Read a specific sample: matrix slicing by row

```
[80] sample_to_extract = 1
     sample = xNumpy[sample_to_extract,:]
     print(sample)
```

```
[4.9 3.  1.4 0.2]
```

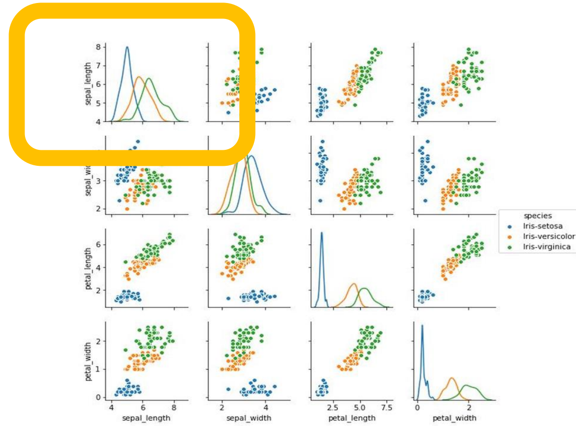
Data preprocessing

Data normalization:
ensure that the data is in the range [0-1]

- Min-max normalization:
  (x-min(x)) / (max(x)-min(x))

```
[81] xNumpy_Norm = (xNumpy - np.min(xNumpy)) / (np.max(xNumpy) - np.min(xNumpy))
```

Data preprocessing

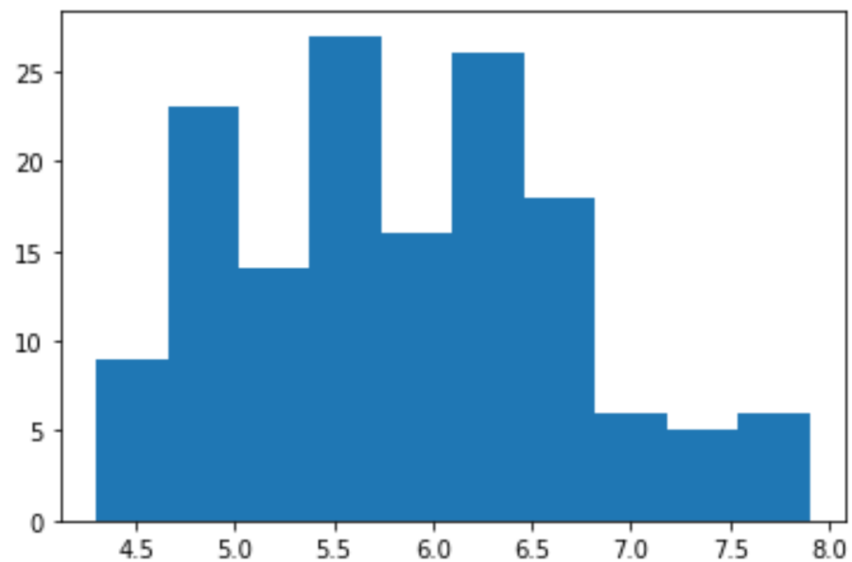


Data normalization:
ensure that the data is in the range [0-1]

- Non-normalized data

```
plt.subplots()
plt.hist(xNumpy[:,0])
```

	sepal_length	sepal_width	petal_length	petal_width	iris_class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa



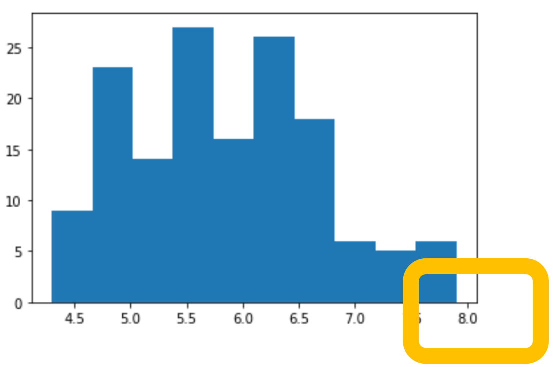
```
[81] xNumpy_Norm = (xNumpy - np.min(xNumpy)) / (np.max(xNumpy) - np.min(xNumpy))
```

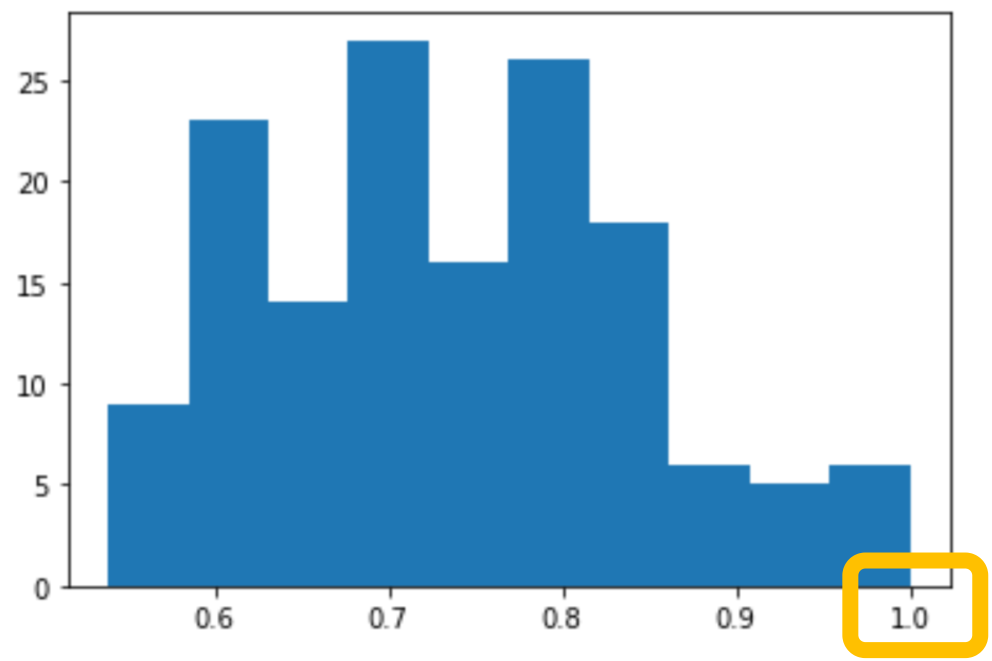
Data preprocessing

Data normalization:
ensure that the data is in the range [0-1]

- Normalized data

```
plt.subplots()
plt.hist(xNumpy_Norm[:,0])
```





Data preprocessing

Data conversion: from categorical to decimal

- Class data is in categorical format

iris_class
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa

Data preprocessing

Data conversion: from categorical to decimal

- Conversion to decimal for easier manipulation

```
yNumpy, dummy = pd.factorize(yNumpy)
print(yNumpy)
```

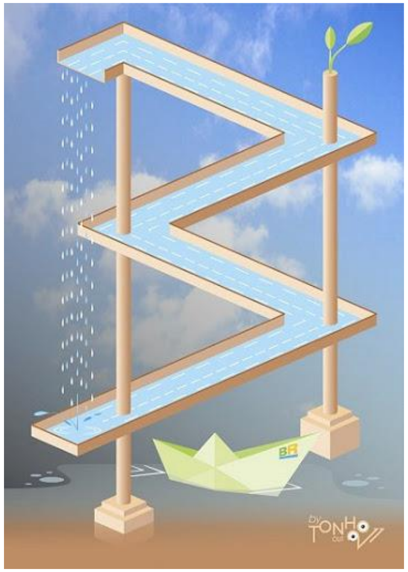
```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

iris_class
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa

Data visualization

Visualizing the features and the classes allows us to have a first visual check on the discriminatory capability

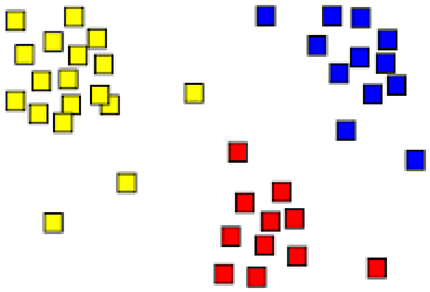
- Drawback: visualization is limited to 2 or 3 dimensions



Data visualization

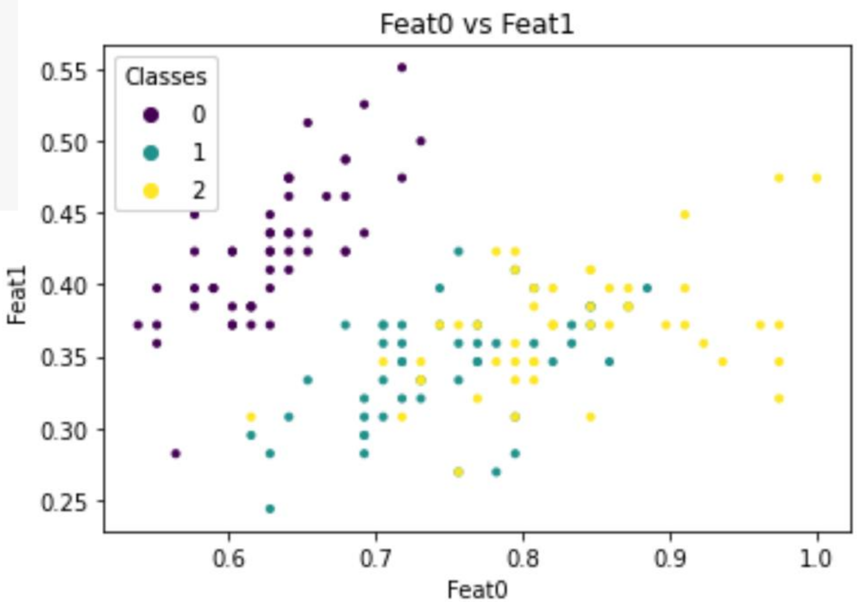
Visualizing the features and the classes allows us to have a first visual check on the discriminatory capability

1. Select features to plot
2. Assign color based on the class



Data visualization

```
import matplotlib.pyplot as plt
# select 2 features to plot
feat0 = xNumpy_Norm[:,0]
feat1 = xNumpy_Norm[:,1]
# params
area = np.pi*3
# Plot
fig1, ax1 = plt.subplots()
scatter1 = ax1.scatter(feat0, feat1, s=area, c=yNumpy)
plt.title('Feat0 vs Feat1')
plt.xlabel('Feat0')
plt.ylabel('Feat1')
legend1 = ax1.legend(*scatter1.legend_elements(),
                    loc="upper left", title="Classes")
ax1.add_artist(legend1)
plt.show()
```

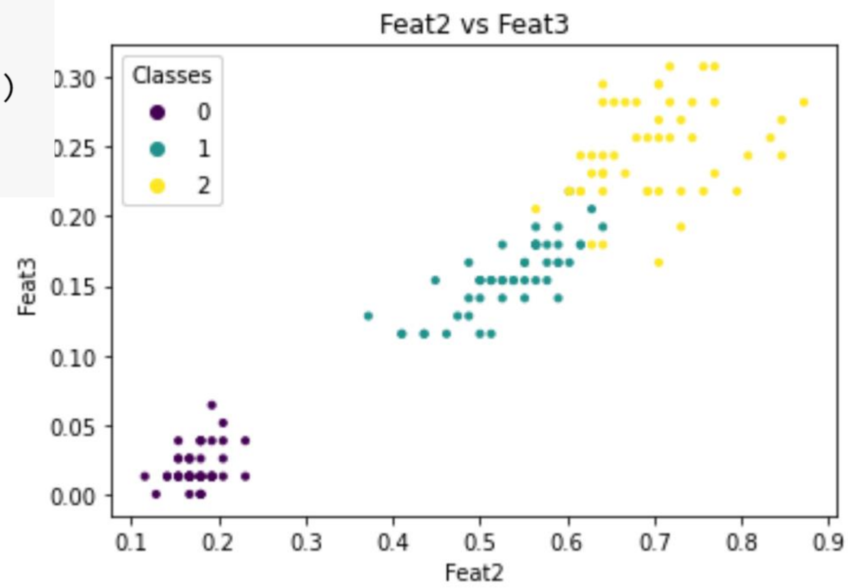


Feat0 + Feat1:

- Good discrimination for class 0
- Bad discrimination for class 1 and 2 (classes are mixed)

Data visualization

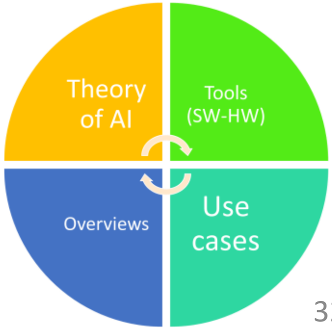
```
# select 2 features to plot
feat2 = xNumpy_Norm[:,2]
feat3 = xNumpy_Norm[:,3]
# params
area = np.pi*3
# Plot
fig2, ax2 = plt.subplots()
scatter2 = ax2.scatter(feat2, feat3, s=area, c=yNumpy)
plt.title('Feat2 vs Feat3')
plt.xlabel('Feat2')
plt.ylabel('Feat3')
legend2 = ax2.legend(*scatter2.legend_elements(),
                    loc="upper left", title="Classes")
ax2.add_artist(legend2)
plt.show()
```



Feat2 + Feat3:

- Better discrimination for all classes

Toolboxes Loading, EDA/plotting in Matlab



Load the dataset (1/2)

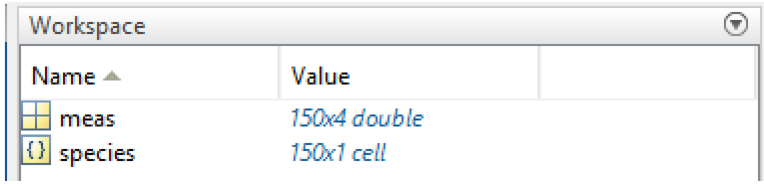
```
% Load the dataset and cast data
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data';
filename = 'iris.dat';
outfilename = websave(filename,url);

fid = fopen(outfilename, 'r');
S = textscan(fid,'%f%f%f%s' , 'delimiter' ,  ','    ) ;
fclose(fid);
meas = [S{1}, S{2}, S{3}, S{4}];
species = S{5};
% Equivalent to "load fisheriris"
```

5.1000	3.5000	1.4000	0.2000
4.9000	3	1.4000	0.2000
4.7000	3.2000	1.3000	0.2000
4.6000	3.1000	1.5000	0.2000
5	3.6000	1.4000	0.2000
5.4000	3.9000	1.7000	0.4000
4.6000	3.4000	1.4000	0.3000
5	3.4000	1.5000	0.2000
4.4000	2.9000	1.4000	0.2000
4.9000	3.1000	1.5000	0.1000
5.4000	3.7000	1.5000	0.2000
4.8000	3.4000	1.6000	0.2000
4.8000	3	1.4000	0.1000
4.3000	3	1.1000	0.1000

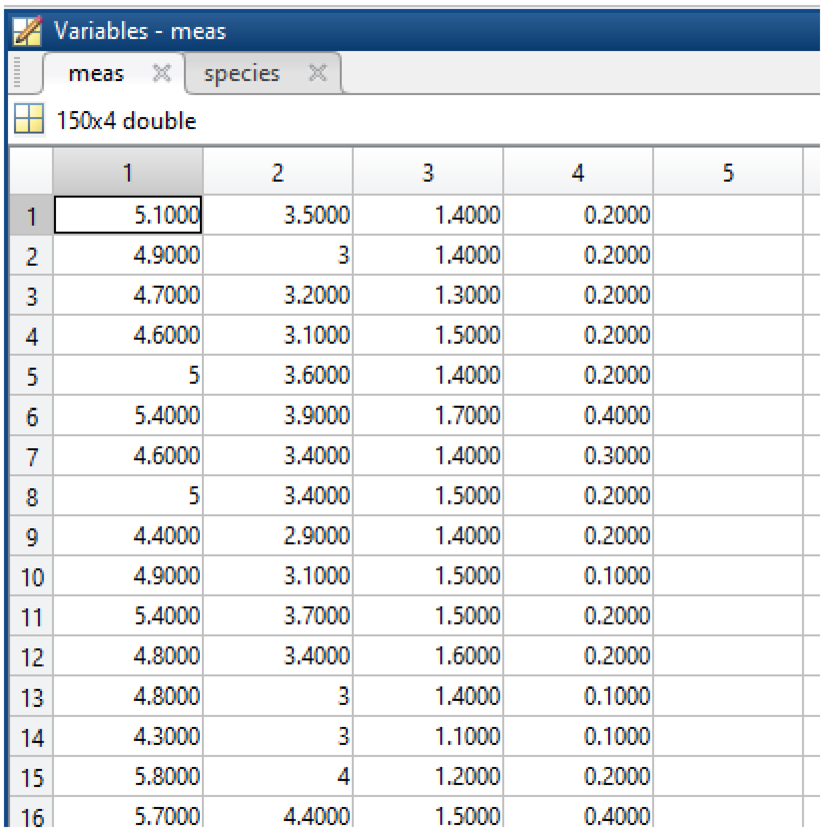
...

Load the dataset (2/2)



Workspace

Name	Value
meas	*150x4 double*
species	*150x1 cell*

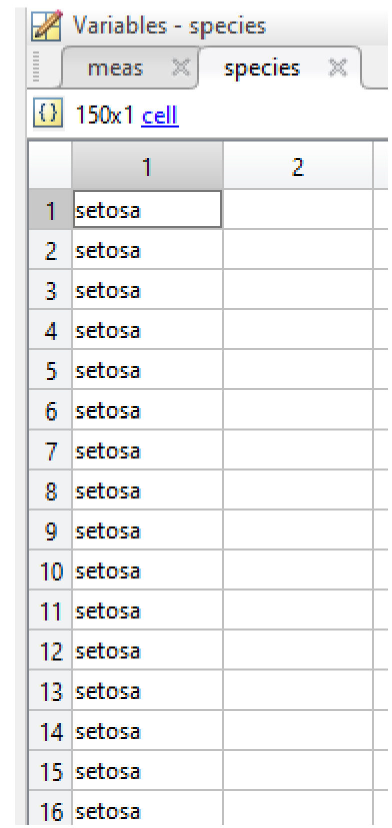


Variables - meas

meas | species

150x4 double

	1	2	3	4	5
1	5.1000	3.5000	1.4000	0.2000	
2	4.9000	3	1.4000	0.2000	
3	4.7000	3.2000	1.3000	0.2000	
4	4.6000	3.1000	1.5000	0.2000	
5	5	3.6000	1.4000	0.2000	
6	5.4000	3.9000	1.7000	0.4000	
7	4.6000	3.4000	1.4000	0.3000	
8	5	3.4000	1.5000	0.2000	
9	4.4000	2.9000	1.4000	0.2000	
10	4.9000	3.1000	1.5000	0.1000	
11	5.4000	3.7000	1.5000	0.2000	
12	4.8000	3.4000	1.6000	0.2000	
13	4.8000	3	1.4000	0.1000	
14	4.3000	3	1.1000	0.1000	
15	5.8000	4	1.2000	0.2000	
16	5.7000	4.4000	1.5000	0.4000	



Variables - species

meas | species

150x1 cell

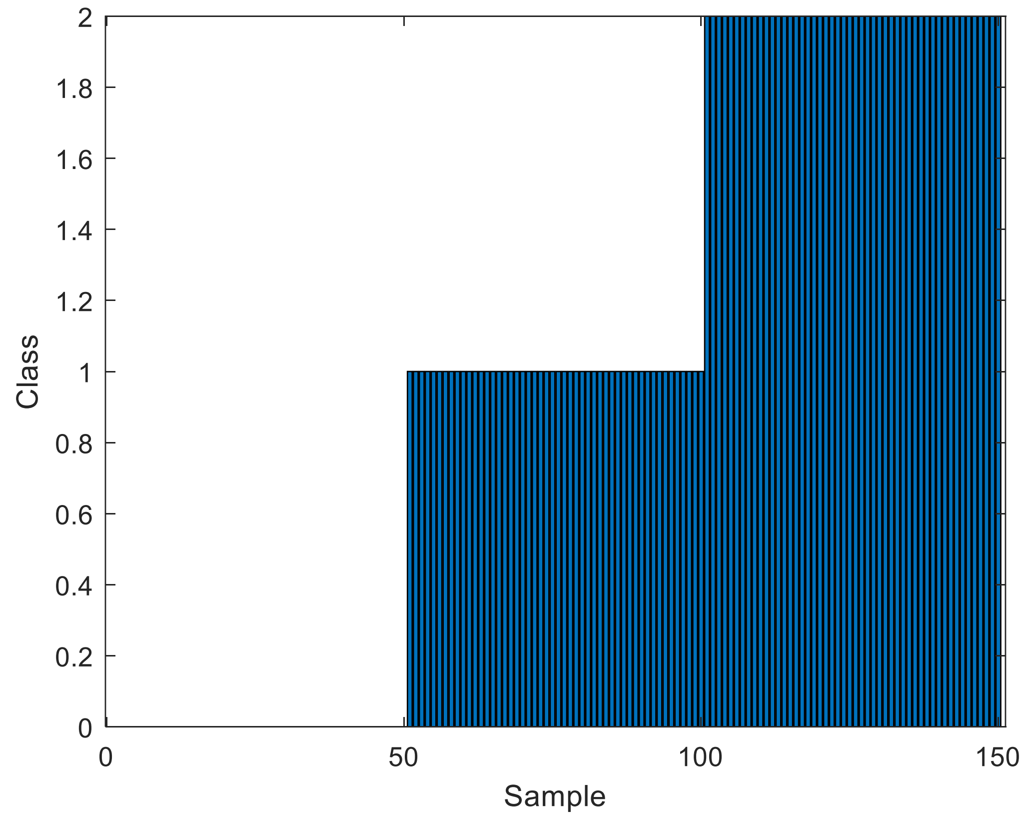
	1	2
1	setosa	
2	setosa	
3	setosa	
4	setosa	
5	setosa	
6	setosa	
7	setosa	
8	setosa	
9	setosa	
10	setosa	
11	setosa	
12	setosa	
13	setosa	
14	setosa	
15	setosa	
16	setosa	

Create the data structures

```
% Problem
P = meas;

% Targets
T = zeros(size(species));
for i = 1 : size(T,1)
    switch species{i}
        case 'setosa'
            T(i) = 0;
        case 'versicolor'
            T(i) = 1;
        case 'virginica'
            T(i) = 2;
    end
end

figure
bar(T)
xlabel('Sample')
ylabel('Class')
```



Plotmatrix() → Feat$_i$ VS Feat$_j$

%% creating the feature matrix Problem (X in the course)

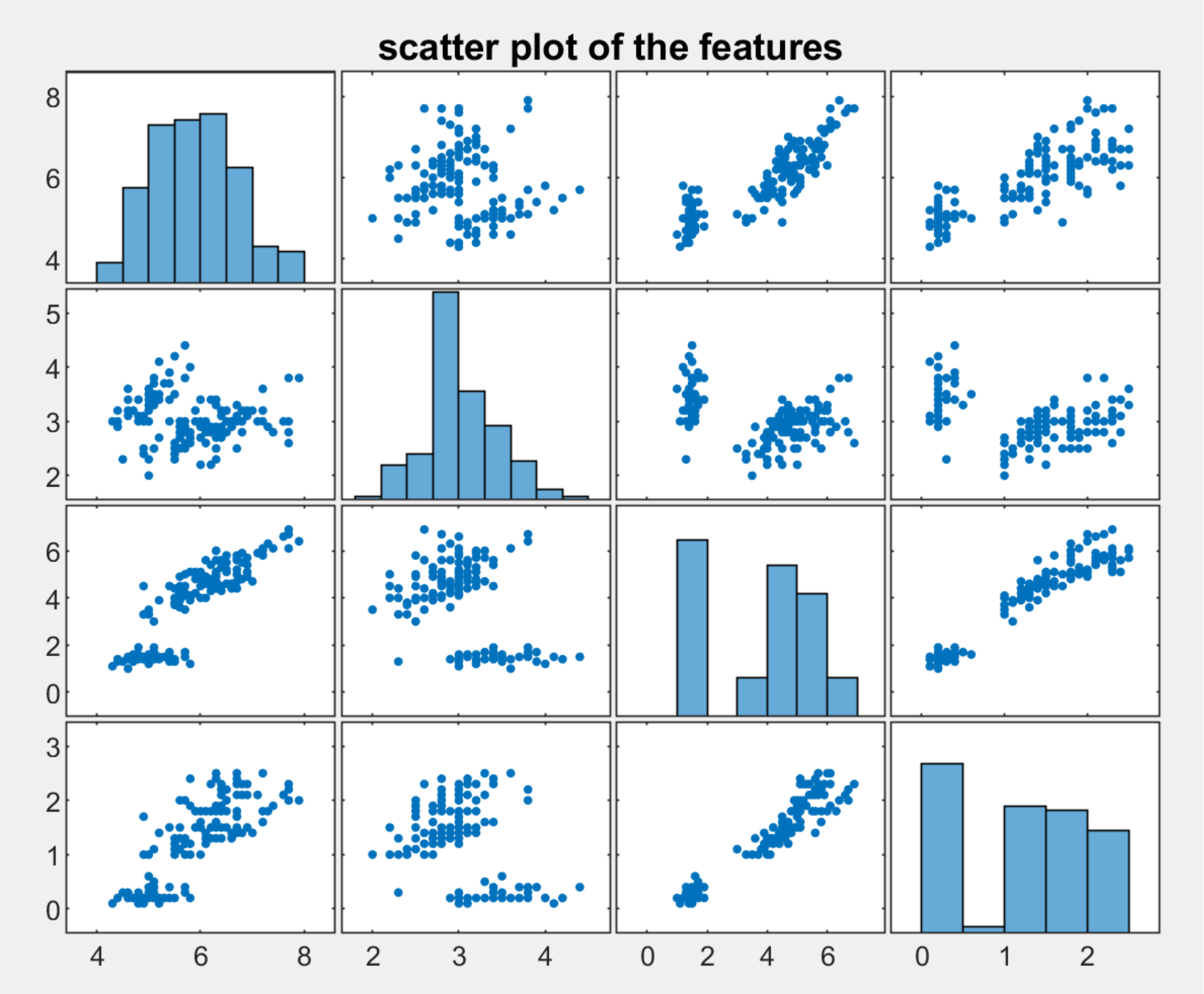
P = meas;

% plot the feature matrix

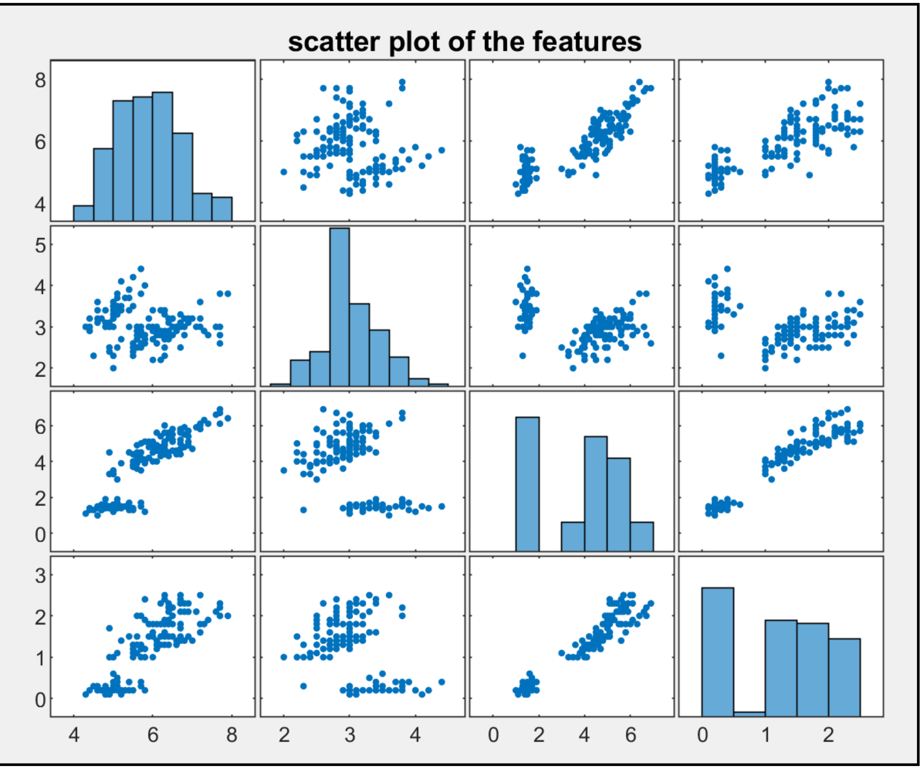
figure

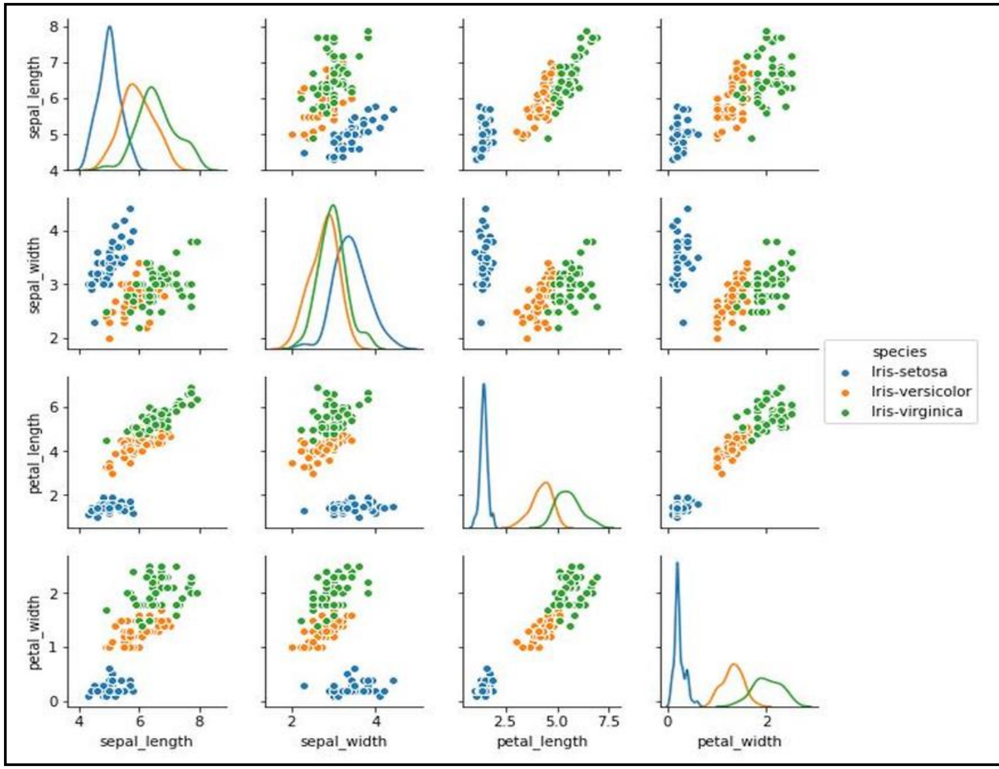
plotmatrix(P)

title('scatter plot of the features')



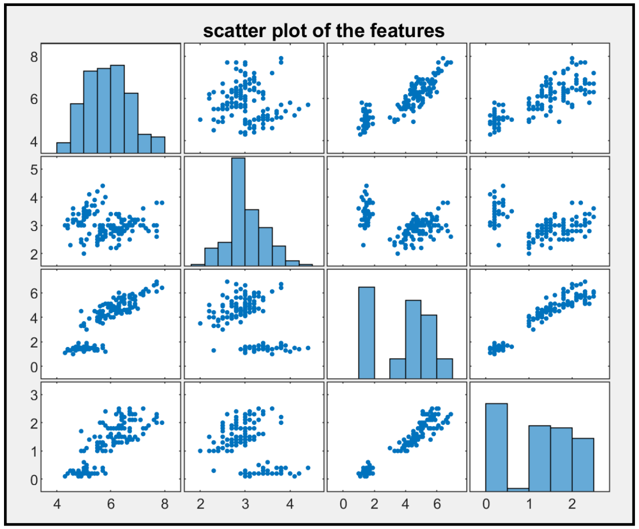
plotmatrix() VS pairplot ()

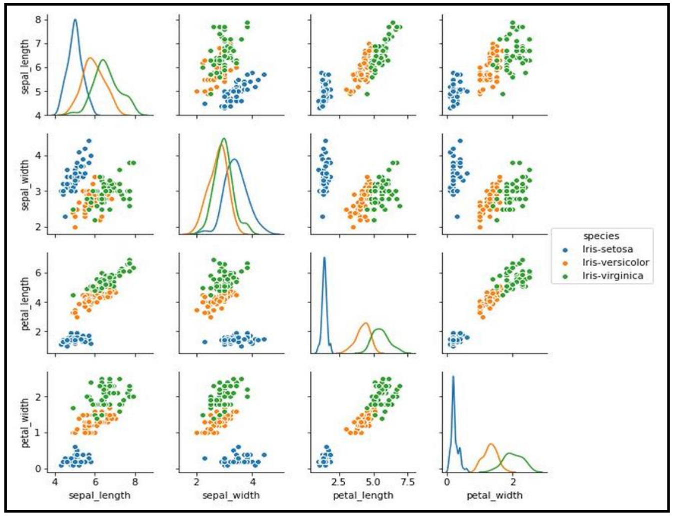


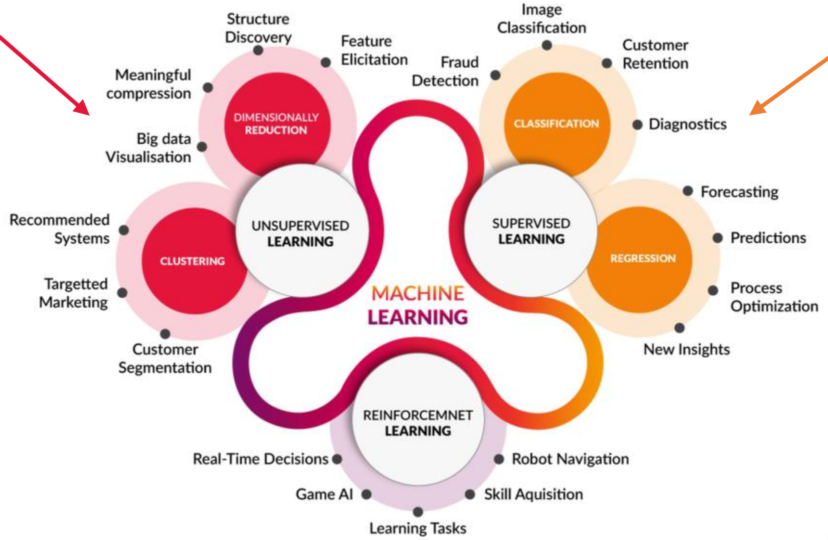


Unsupervised VS Supervised

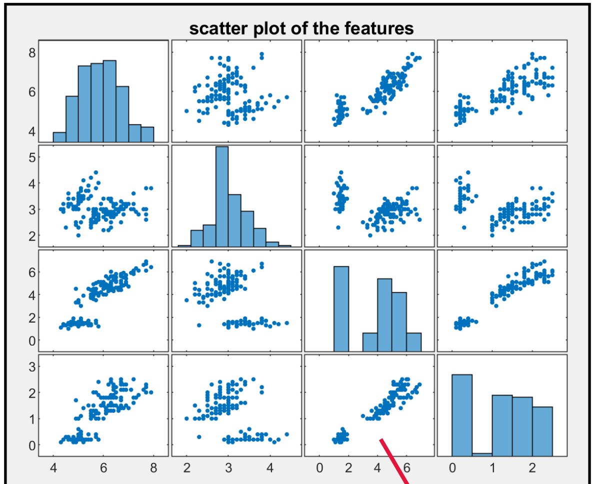
Unsupervised VS Supervised

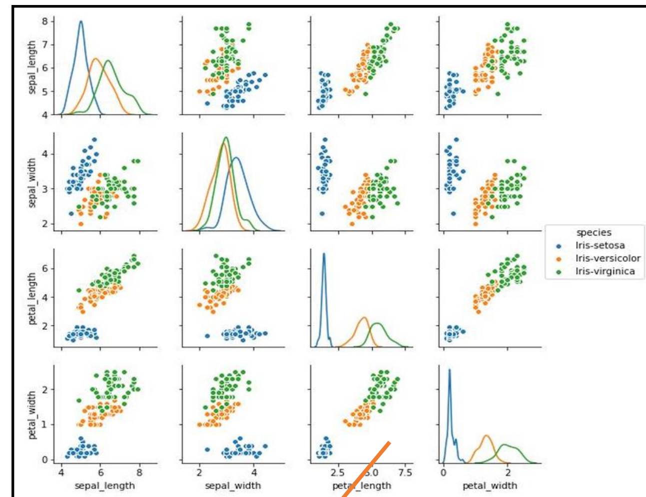




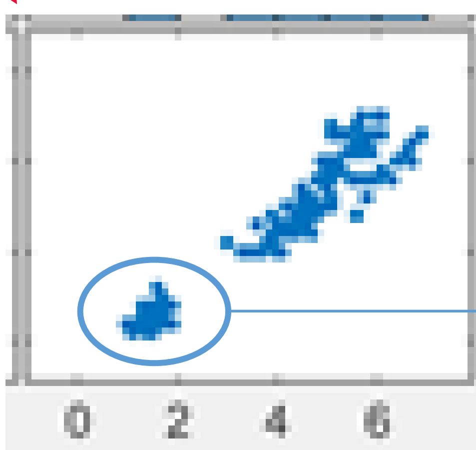


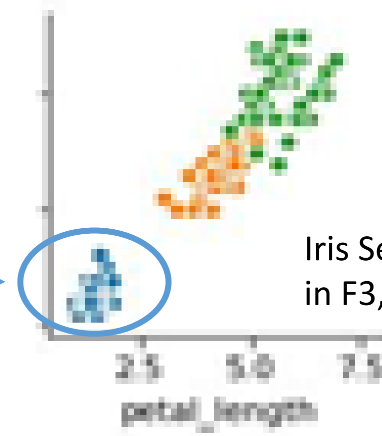
Clusters in the unsupervised analysis





F4 VS F3





Petal with VS Petal length

Iris Setosa Cluster!
in F3,F4 subspace is well separate

Preliminary data analysis

```
statisticsP = zeros(4, size(P,2));
statisticNames{1} = 'Mean';
statisticNames{2} = 'Std ';
statisticNames{3} = 'Min ';
statisticNames{4} = 'Max ';
statisticsP(1,:) = mean(P);
statisticsP(2,:) = std(P);
statisticsP(3,:) = min(P);
statisticsP(4,:) = max(P);

fprintf('\t\tFeat 1\tFeat 2\tFeat 3\tFeat 4\n');
for i = 1 : 4
    fprintf('%s\t', statisticNames{i});
    for j = 1 : size(P,2)
        fprintf('%03.03f\t', statisticsP(i,j));
    end
    fprintf('\n')
end
```

Matlab is natively working on columns

5.1000	3.5000	1.4000	0.2000
4.9000	3	1.4000	0.2000
4.7000	3.2000	1.3000	0.2000
4.6000	3.1000	1.5000	0.2000
5	3.6000	1.4000	0.2000
5.4000	3.9000	1.7000	0.4000
4.6000	3.4000	1.4000	0.3000
5	3.4000	1.5000	0.2000
4.4000	2.9000	1.4000	0.2000
4.9000	3.1000	1.5000	0.1000
5.4000	3.7000	1.5000	0.2000
4.8000	3.4000	1.6000	0.2000
4.8000	3	1.4000	0.1000
4.3000	3	1.1000	0.1000

mean(P) → [5.843 3.057 3.758 1.199]

Command Window

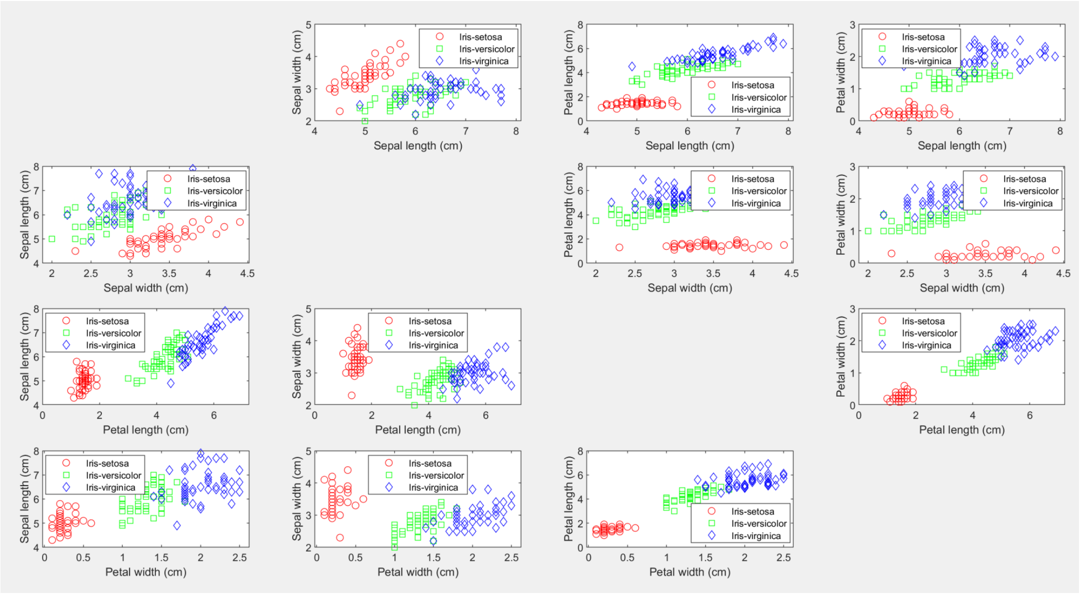
New to MATLAB? See resources for Getting Started.

	Feat 1	Feat 2	Feat 3	Feat 4
Mean	5.843	3.057	3.758	1.199
Std	0.828	0.436	1.765	0.762
Min	4.300	2.000	1.000	0.100
Max	7.900	4.400	6.900	2.500

fx >>

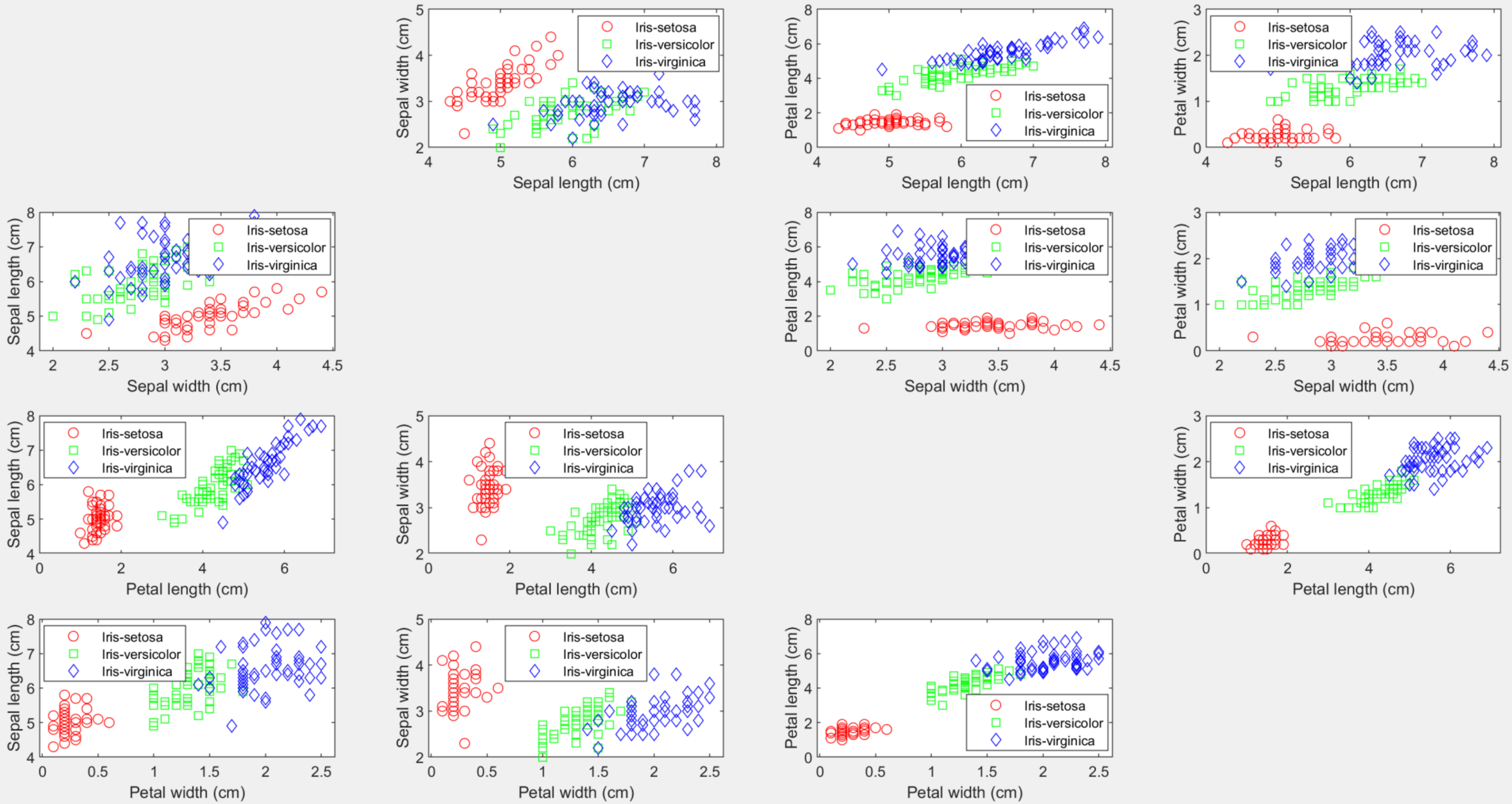
Plot the data (1/2)

```
% Plot the dataset
variableNames{1} = 'Sepal length (cm)';
variableNames{2} = 'Sepal width (cm)';
variableNames{3} = 'Petal length (cm)';
variableNames{4} = 'Petal width (cm)';
iCount = 1;
figure
for i = 1 : 4
  for j = 1 : 4
    if (i~=j)
      subplot(4, 4, iCount)
      gscatter(P(:,i), P(:,j), species,'rgb','osd');
      xlabel(variableNames{i})
      ylabel(variableNames{j})
    end
    iCount = iCount + 1;
  end
end
```



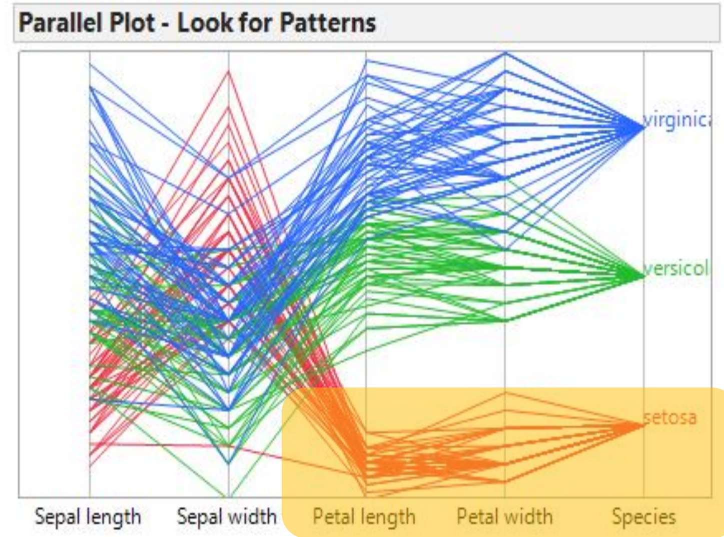
You can create your own visualization plots and dashboard! This is a valuable part of the job…

Plot the data (2/2)

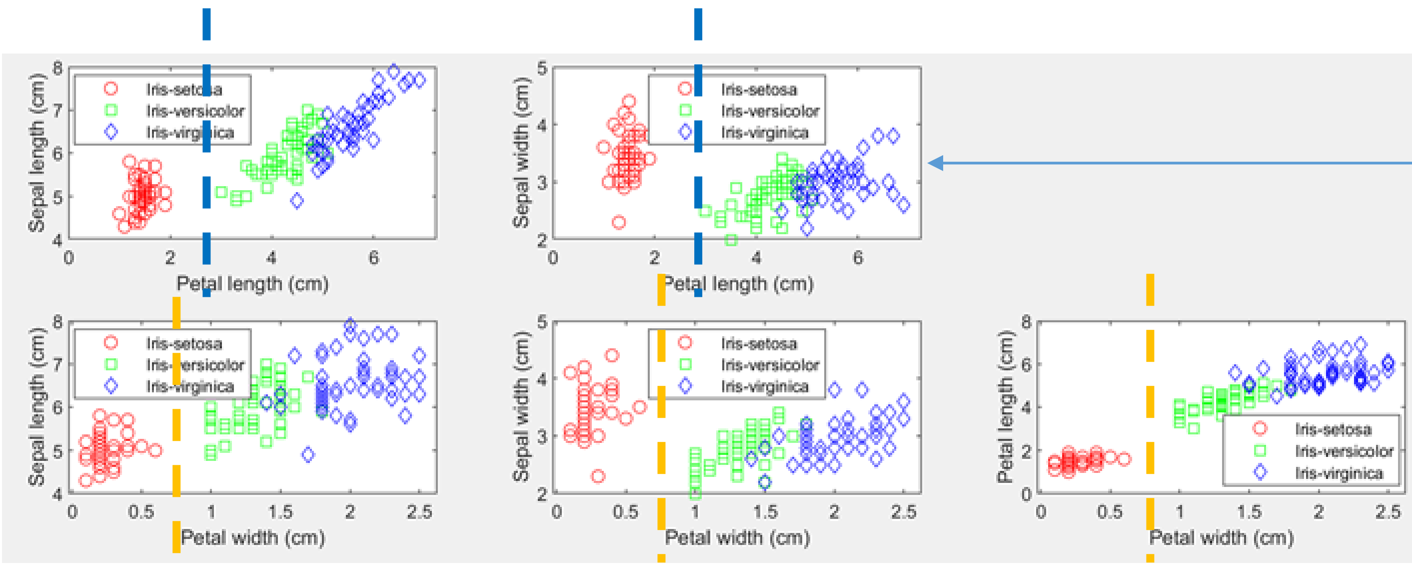


EDA: application

From the Parallel Plot we already verified the separation at least of the Setosa Class
Just using the Petal length or Petal width



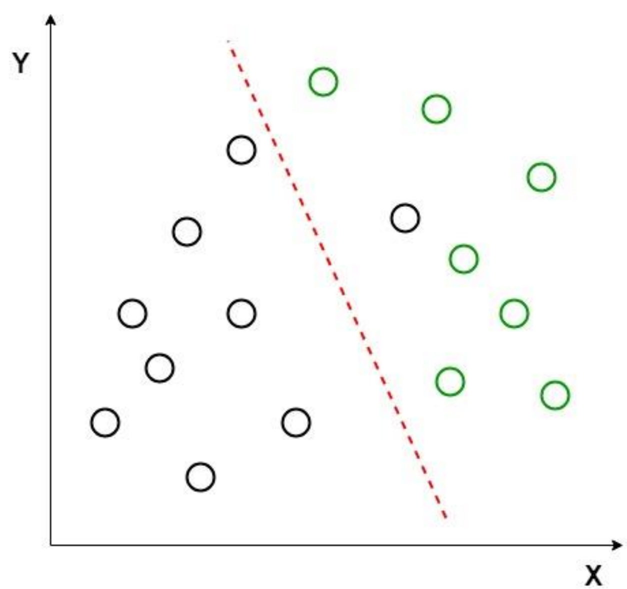
This approach give EXPLANABILITY to your models!



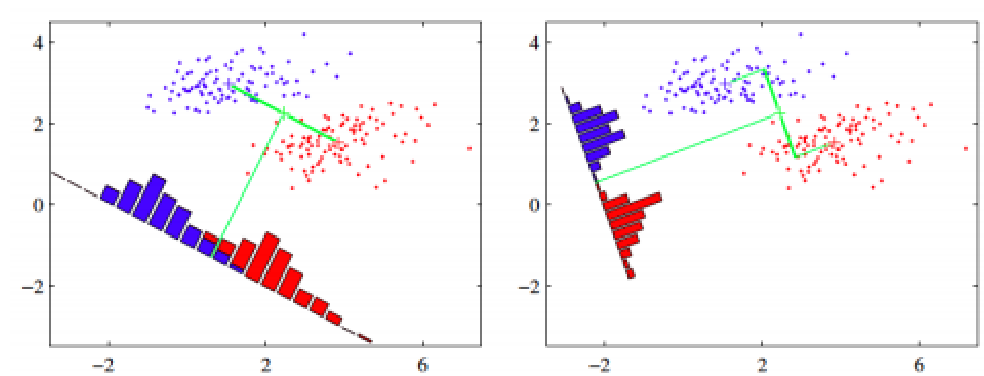
Classification

```
% Linear Discriminant
% Analysis
lda = fitcdiscr(P,T);
ldaClass = resubPredict(lda);
```

Just an example:
we will present this topic
later in the course



Fischer's
Linear Discriminant Analysis Idea in 2D



Linear Classification

	1	2	3	4
1	5.1000	3.5000	1.4000	0.2000
2	4.9000	3	1.4000	0.2000
3	4.7000	3.2000	1.3000	0.2000
4	4.6000	3.1000	1.5000	0.2000
5	5	3.6000	1.4000	0.2000
6	5.4000	3.9000	1.7000	0.4000
7	4.6000	3.4000	1.4000	0.3000
8	5	3.4000	1.5000	0.2000
9	4.4000	2.9000	1.4000	0.2000
10	4.9000	3.1000	1.5000	0.1000
11	5.4000	3.7000	1.5000	0.2000
12	4.8000	3.4000	1.6000	0.2000
13	4.8000	3	1.4000	0.1000
14	4.3000	3	1.1000	0.1000
15	5.8000	4	1.2000	0.2000
16	5.7000	4.4000	1.5000	0.4000

P

[0
0
0
...
1
1
....]

T

% Linear Discriminant Analysis

lda = **fitcdiscr**(P,T);

% predict values

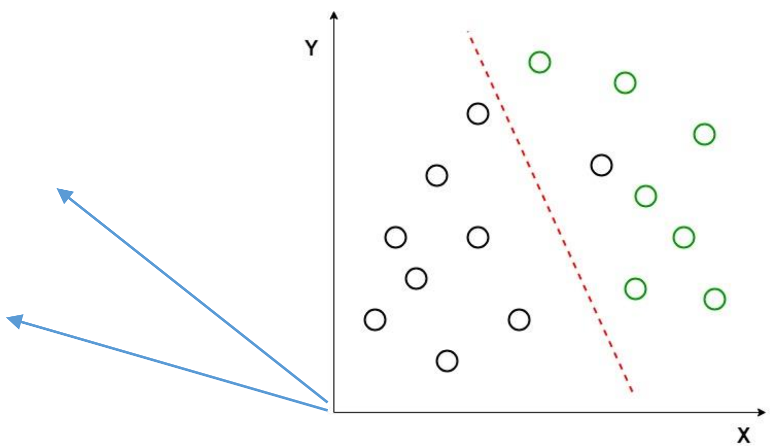
% using the predictor data stored in lda.X.

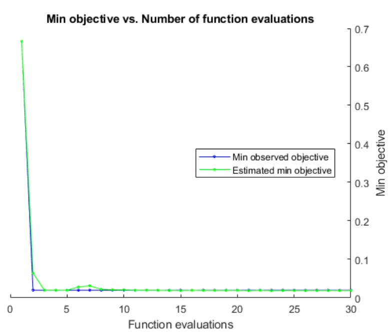
ldaClass = **resubPredict**(lda);

Fischer's

Linear Discriminant Analysis

in 4 dimension

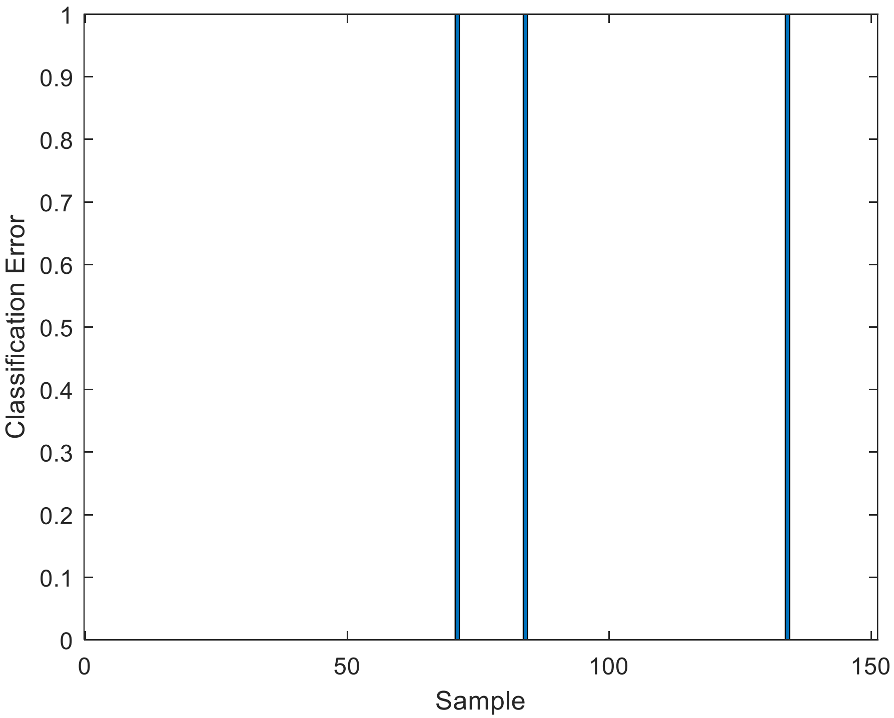




Classification error

```
% Classification error
errorVector = ldaClass ~= T;

figure
bar(errorVector);
xlabel('Sample’)
ylabel('Classification Error')
```



Figures of merit (1/2)

```
Command Window
New to MATLAB? See resources for Getting Started.
Total classification error = 3
Std of the classification error = 0.140
Total classification error = 2.000 %
Std of the classification error = 14.047 %

confusionMatrix =

    50     0     0
     0    48     1
     0     2    49

>>
```

Just an example:
we will present this topic later in the course

Figures of merit

```
% Figures of merit
totalError = sum(errorVector);
stdError = std(errorVector);
totalErrorPerc = mean(errorVector)*100;
stdErrorPerc = stdError * 100;

fprintf('Total classification error = %d \n', totalError);
fprintf('Std of the classification error = %03.03f \n', stdError);
fprintf('Total classification error = %03.03f %% \n', totalErrorPerc);
fprintf('Std of the classification error = %03.03f %% \n', stdErrorPerc);
```

```
Total classification error = 3
Std of the classification error = 0.140
Total classification error = 2.000 %
Std of the classification error = 14.047 %
```

```
%Confusion Matrix
confusionMatrix = confusionmat(ldaClass,T)
```

```
confusionMatrix =

    50     0     0
     0    48     1
     0     2    49
```

Just an example:
we will present this topic later in the course