Mini Projets 2019 (Info 232)

Isabelle Guyon
info232@chalearn.org

1. Visualisation

We are visualizing the famous Iris dataset (https://archive.ics.uci.edu/ml/datasets/iris).

Instructions

Ce TP vaut 5 points. Repondez a toutes les questions.

- Pour creer une nouvelle cellule, allez dans le menu "Insert".
- Pour transformer une cellule en commentaire texte, allez dans Cell + Cell Type + Markdown.
- Pour executer une cellule: SHIFT+RETURN

Les cellules doivent entre executee dans l'ordre. Pour plus d'information CONSULTEZ la DOCUMENTATION. ()

Question 0: "Markdown" cells

Creez une nouvelle cellule de type Markdown en dessous de celle-ci. Recopiez dedans le paragraphe sur k-fold cross validation que vous trouverez sur Wikipedia. Essayez de colorer la cellule en VERT!

Question 1: "Code" cells

Maintenant vous allez executer la cellule ci-dessous apres avoir replace la reponse par 1.

In [1]:

```
code_dir = 'code/'
from sys import path; path.append(code_dir)
%matplotlib inline
%load_ext autoreload
%autoreload 2
from checker import check
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
question = 1
answer = 1  # Replace by 1
score = 0
score += check(answer, question)
```

```
1
1

CORRECT
:-)
```

Question 2: AutoML format

Les donnees que vous aurez a analyser dans votre projet seront au format AutoML (https://github.com/codalab/chalab/wiki/Help:-Wizard-%E2%80%90-Challenge-%E2%80%90-Data): . En utilisant les moyens que vous voulez, remplacez les valeurs des variables de dimension des donnees par leur valeurs correctes dans la deuxieme cellule avant de l'executer.

In [2]:

```
data_dir = 'data'
data_name = 'iris'
!ls $data_dir*
!cat $data_dir/iris_public.info
```

```
iris_feat.name      iris_public.info    iris_train.data     iris_vali
d.solution
iris_label.name     iris_test.data      iris_train.solution
iris_private.info   iris_test.solution  iris_valid.data
usage = 'Sample dataset Iris data'
name = 'iris'
task = 'multiclass.classification'
target_type = 'Numerical'
feat_type = 'Numerical'
metric = 'bac_metric'
time_budget =  1200
feat_num =     4
target_num =     3
label_num =     3
train_num =   35
valid_num =    35
test_num =    35
has_categorical =     0
has_missing =     0
is_sparse =     0
```

In [3]:

```
feature_number = 4
training_sample_number = 35
validation_sample_number = 35
test_sample_number = 35
question = 2
reponse = feature_number*(training_sample_number+validation_sample_number+test_s
score += check(reponse, question)
```

```
420
420
```

CORRECT
:-)

Question 3: Pandas

En Anglais pour changer :-)

This time we are going to do simple "exploratory data analysis". To simplify, we lump all the data together in one big data structure calle a "pandas" data frame. This will allow us to use the rich libraries "pandas" and "seaborn" to explore the data.

In the next cell, replace the "head" function, which just shows the first few rows of the dataset, by a padas function providing descriptive statistics. To that end, you may want to check the Pandas DataFrame reference page (https://pandas.pydata.org/pandas-docs/stable/reference/frame.html). Then, in the following cell, replace the variables with their correct values and execute it.

In [4]:

```
from data_io import read_as_df
data = read_as_df(data_dir  + '/' + data_name)                # The data are loa
data.head()
data.describe()
```

```
Reading data/iris_train from AutoML format
Number of examples = 35
Number of features = 4
        Class
0      setosa
1  versicolor
2   virginica
Number of classes = 3
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
count	35.000000	35.000000	35.000000	35.000000
mean	5.625714	3.005714	3.402857	1.054286
std	0.892565	0.421442	1.962514	0.839588
min	4.300000	2.000000	1.000000	0.100000
25%	4.900000	2.800000	1.450000	0.200000
50%	5.500000	3.000000	3.700000	1.000000
75%	6.300000	3.250000	5.100000	1.800000
max	7.700000	4.000000	6.700000	2.500000

In [5]:

```
std_sepal_length = 0.892565  # Standard deviation of the sepal length
mean_sepal_width = 3.005714   # Mean of the sepal width
min_petal_length = 1  # Minimum value of the petal length
max_petal_width = 2.5    # Maximum value of the petal width
question = 3
reponse = std_sepal_length+mean_sepal_width+min_petal_length+max_petal_width
score += check(reponse, question)
```

```
7.3982790000000005
7.3982790000000005
```

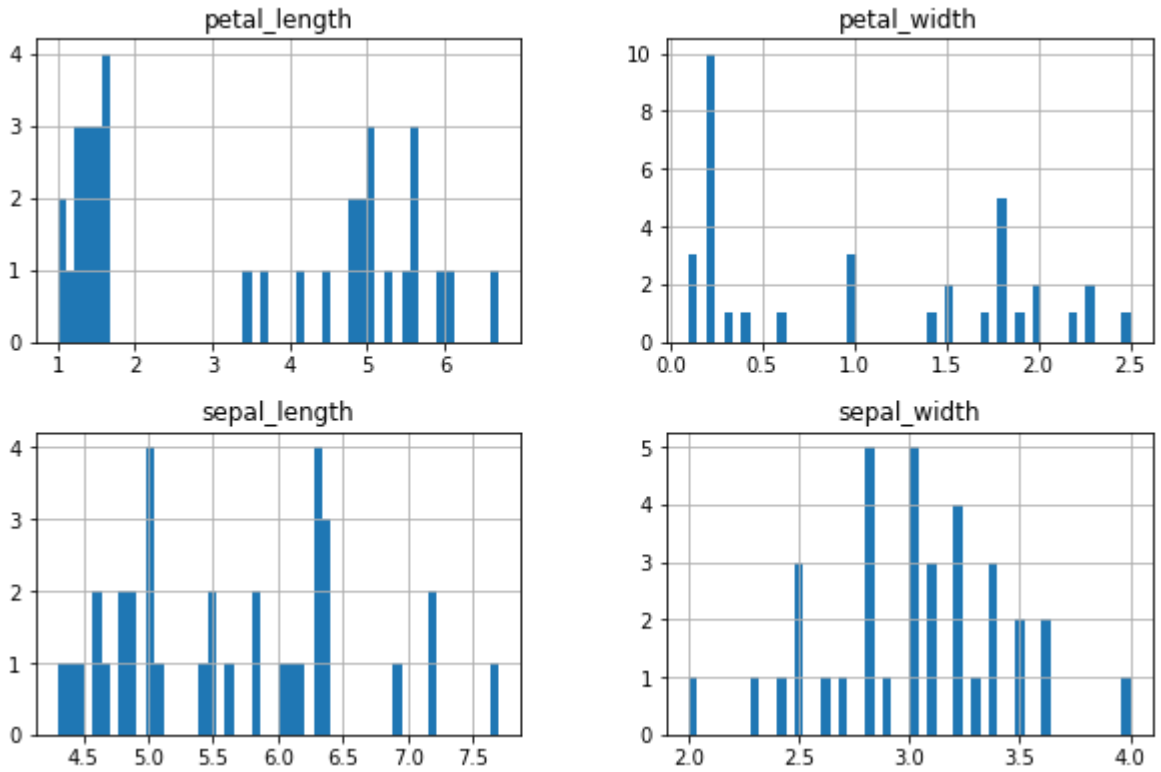
CORRECT
:-)

Question 4: Histograms

Un truc sympa de Pandas c'est que ca permet aussi de faire des graphes pour visualiser les donnees. Remplacez le nombre de bins par un plus petit nombre. Repondez en changeant la variable answer dans la deuxieme cellule: answer=1 si la hauteur de la plus grande barre dimininue et answer=0 sinon. Comprenez-vous pourquoi?

In [6]:

```
data.hist(figsize=(10, 10), bins=50, layout=(3, 2));
```



In [7]:

```
question = 4
answer = 1            # one if the maximum bar height increases when the bin numb
score += check(answer, question)
```

1
1

CORRECT
:-)

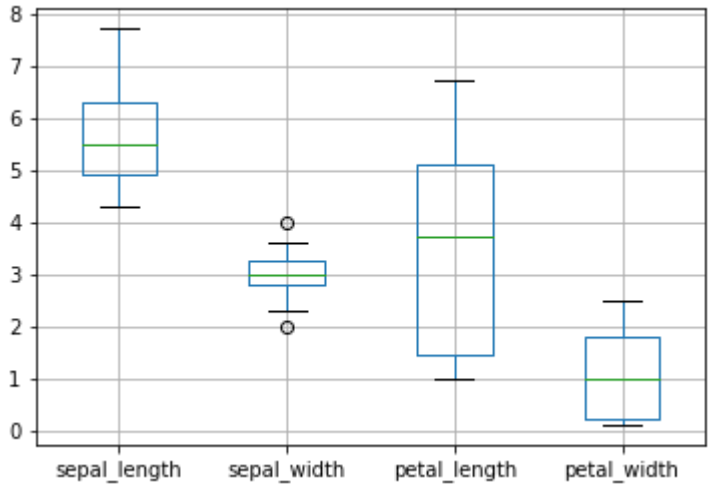
Have fun with the Pandas DataFrame reference page (https://pandas.pydata.org/pandas-docs/stable/reference/frame.html). There are lots of other plotting functions. Try some of them in the cell below.

In [8]:

```
# What's this one for instance? Change it to something else.
data.boxplot()
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d303780>
```



Question 5: Pair plots

Seaborn (sns) is a package of data visualization functions: https://seaborn.pydata.org/ (https://seaborn.pydata.org/). Quite useful! It is convenient to visualize data in 2 dimensions. One way of doing that is to plot a variable (feature) against another one, one point representing a sample (a flower). The pairplot function shows all the possibilities (off-diagonal graphs).

On the diagonal, what do you see? Compare with the histograms of the previous question. Then add another argument to the pairplot function

```
hue="target"
```

(if you do not understand, consult the documentation https://seaborn.pydata.org/generated/seaborn.pairplot.html) (https://seaborn.pydata.org/generated/seaborn.pairplot.html)). After executing the next cell again, in the folowing cell, answer the questions:

What is the color of the class, which is best separated from all others?

```
color_best_separated = 1 if blue; 2 if orange; 3 if green.
```

Which iris type does this correspond to?

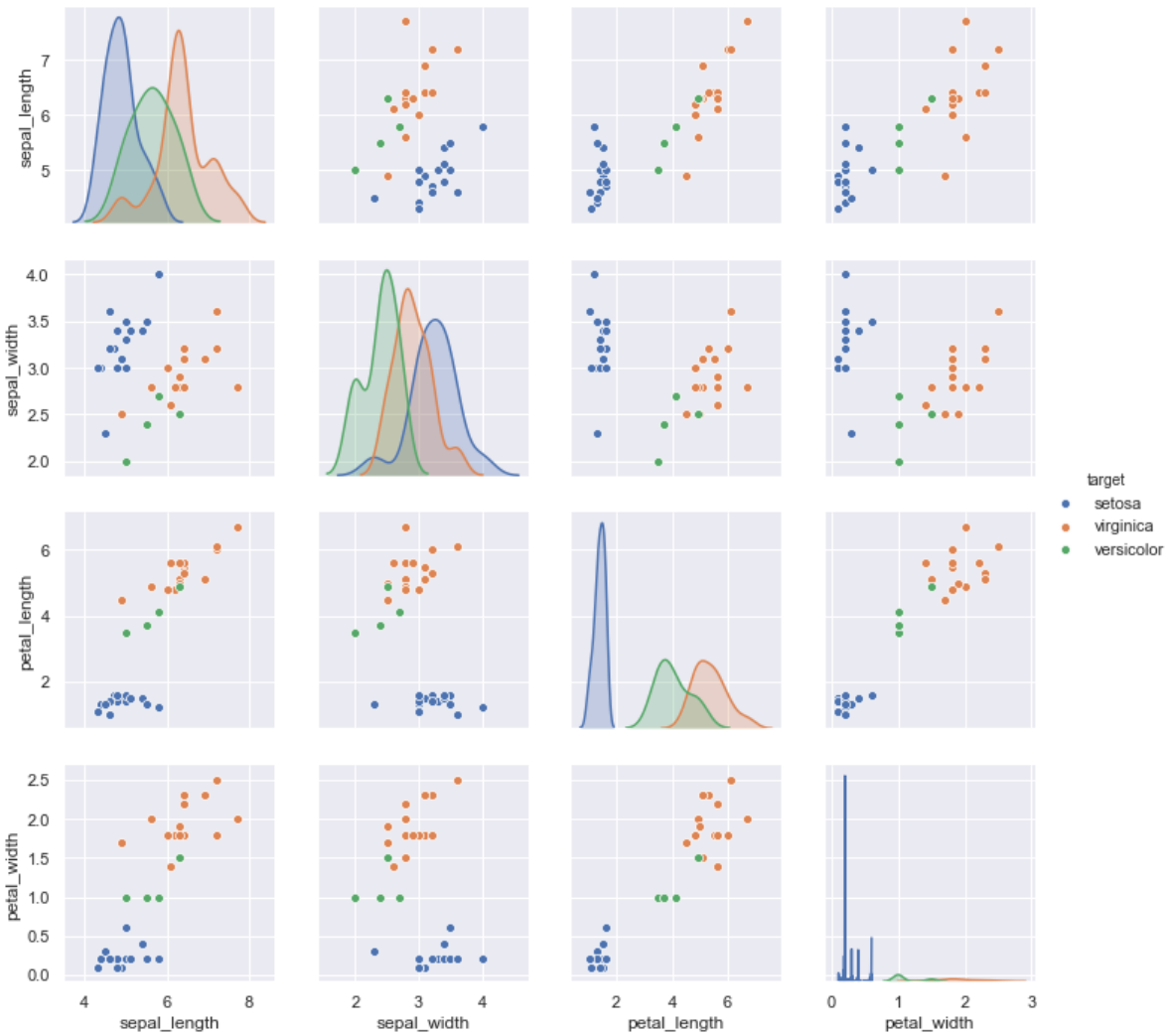
```
iris_best_separated = 1 if virginica; 2 if versocolor; 3 if setosa.
```

In [9]:

```
1 import seaborn as sns; sns.set()
2 sns.pairplot(data, hue="target"),
```

Out[9]:

```
(<seaborn.axisgrid.PairGrid at 0x1a1d7054a8>,)
```



In [10]:

```
question = 5
color_best_separated = 1
iris_best_separated = 3
score += check(color_best_separated*iris_best_separated, question)
```

```
3
3
```

CORRECT
:-)

Question 6: Feature correlation

Le variables (features) peuvent etre redondantes (c'est a dire capturer des informations similaires). Le coefficient de correlation de Pearson (voir page Wikipedia (https://en.wikipedia.org/wiki/Pearson_correlation_coefficient)) permet de detecter la correlation (c'est a dire la similarite au sens d'une dependance lineaire).

En regardant les "pair plots" ci-dessus, a votre avis, quelle paire de variable est la plus correlee? Reperez la paire par son numero de ligne et de colonne pour repondre. Verifiez votre intuition en executant la cellule suivante qui represente graphiquement la matrice de correlation. Changez la methode 'pearson' pour d'autres coefficients de correlation (voir documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html)). Est-ce que ca change quelle paire est la plus correlee? Regardez les definitions de https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient (Kandall tau) et Spearman correlation coefficient (https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient). Essayez de comprendre la difference entre correlation et dependence (https://en.wikipedia.org/wiki/Correlation_and_dependence).

In [11]:

```
question = 6
numero_ligne = 3              # Lignes numerotee de 0 a 3
numero_colonne = 2            # Colonnes numerotee de 0 a 3
score += check(numero_ligne*numero_colonne, question)
```

```
6
6
```

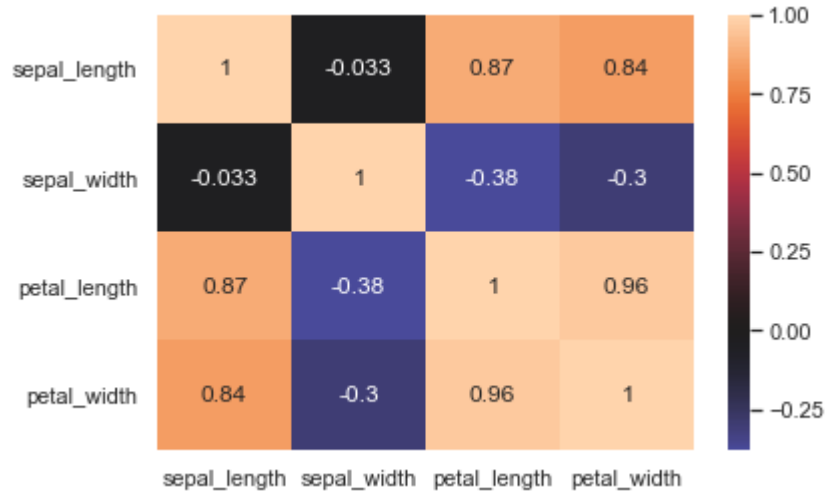
CORRECT
:-)

In [12]:

```
corr_mat = data.corr(method='pearson')
sns.heatmap(corr_mat, annot=True, center=0)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d14f710>
```



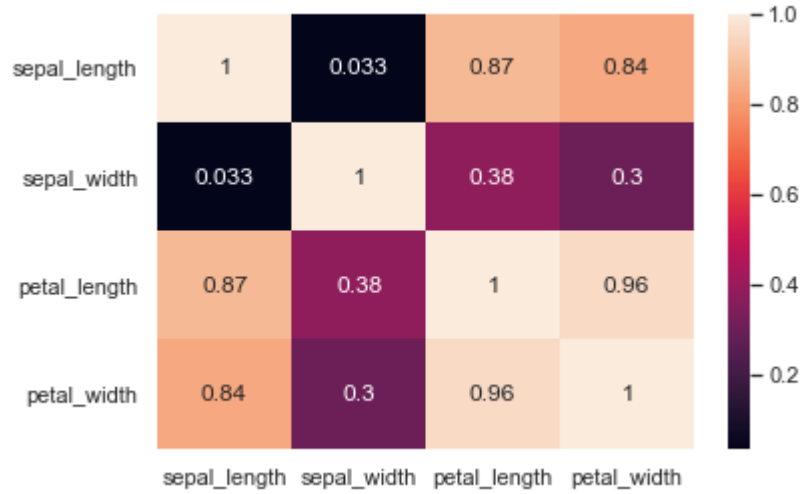
Remarquez que les variables peuvent etre correlee ou anti-corelees. Vous voulez donc peut-etre vous interesser a la valeur absolue du coefficient de correlation. Cela change-t-il votre reponse? Peut-etre pas, mais ca pourrait!

In [13]:

```
sns.heatmap(abs(corr_mat), annot=True)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d0d5be0>
```



Notez que la matrice est symetrique. Si vous voulez vous amuser, effacez les valeurs au dessus de la diagonale (voir en bas de cette page (https://seaborn.pydata.org/generated/seaborn.heatmap.html)).

Question 7: Feature selection

Representing a matrix of coefficients with colors seems to be pretty convenient for visualization purposes. We would like to do that also for the data matrix itself. Note that, since the last column (target) contains strings ("categorical variables"), we first need to convert them to numbers.

Observing the heatmap, which column is most correlated with the target? Insert another cell in which you plot the correlation matrix of data_new (inspiring yourself from the previous question), then confirm your intuition and anwer the question.

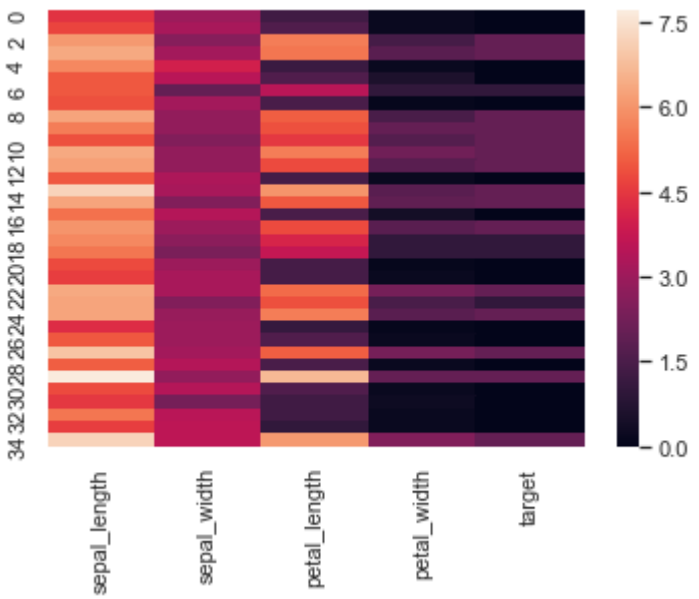
In [14]:

```
print(data.head())
data_num = data.copy()  # If you don't use "copy", any change in data_num will a
data_num['target']= data_num['target'].astype('category')
data_num['target'] = data_num['target'].cat.codes
print(data_num.head())
sns.heatmap(data_num)
```

```
   sepal_length  sepal_width  petal_length  petal_width     target
0           4.4          3.0           1.3          0.2     setosa
1           4.7          3.2           1.6          0.2     setosa
2           6.1          2.6           5.6          1.4  virginica
3           6.4          3.1           5.5          1.8  virginica
4           5.8          4.0           1.2          0.2     setosa
   sepal_length  sepal_width  petal_length  petal_width  target
0           4.4          3.0           1.3          0.2       0
1           4.7          3.2           1.6          0.2       0
2           6.1          2.6           5.6          1.4       2
3           6.4          3.1           5.5          1.8       2
4           5.8          4.0           1.2          0.2       0
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1e8966d8>
```

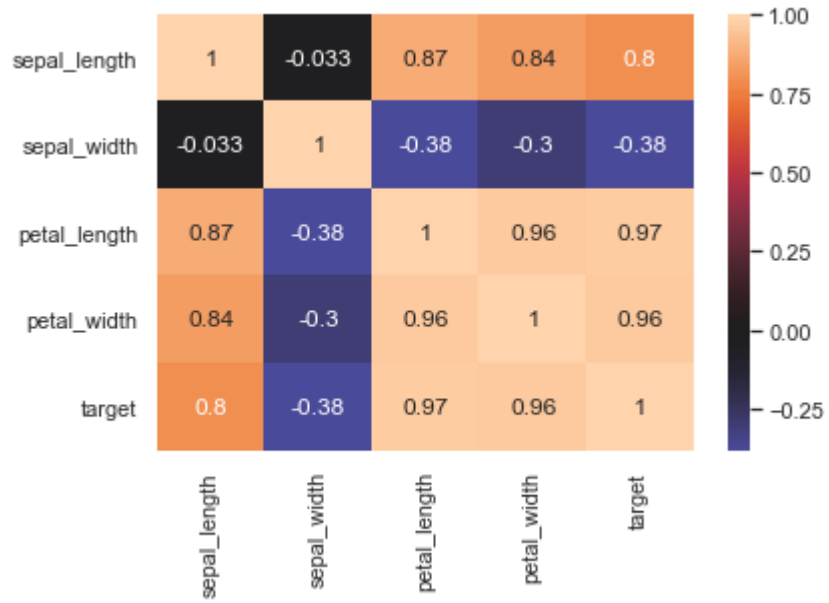


In [15]:

```
# Mettez ici votre code montant la matrice de correlation de data_num
corr_mat = data_num.corr(method='pearson')
sns.heatmap(corr_mat, annot=True, center=0)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1e896278>
```



La question 7 est ici:

Quelle est la variable (feature) la plus correlee avec la colonne "target"? Quelle est la valeur du coefficient de correlation de Pearson correspondant?

In [16]:

```
question = 7
numero_variable = 2             # Variables numerotees de 0 a 3
pearson_correlation = 0.97
score += check(numero_variable+pearson_correlation, question)
```

```
2.9699999999999998
2.97
```

CORRECT
:-)

In [ ]:

```

```

Question 8: One-Rule classifier

In this section, we show how we can create a very simple classifier based on just ONE rule to separate the 3 types of flowers. That rule classifies irises on the basis of their petal length only. Check the code to see whether you understand it.

This classifier respects the structure of scikit-learn (https://scikit-learn.org/stable/) learning machines. To make it compatible with other scikit-learn tools, we derive it from the base class BaseEstimator and overload 2 methods: "fit" and "predict". Then we use the Iris data to train and test a model (in this case we lumped all the data into a single matrix and use it as training data). After that we compute the training error.

Scikit-learn allows you to compute a lot of other metrics (https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics). To answer this question, you will have to compute the BAC (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn.metrics.balanced_a that is the Balanced ACcuracy score.

In [17]:

```
import numpy as np
from sklearn.base import BaseEstimator

class oneR(BaseEstimator):
    ''' One Rule classifier '''
    def __init__(self):
        ''' The "constructor" initializes the parameters '''
        self.selected_feat = 0  # The chosen variable/feature
        self.theta1 = 0         # The first threshold
        self.theta2 = 0         # The second threshold

    def fit(self, X, Y, F=[]):
        ''' The method "fit" trains a super-simple classifier '''
        if not F: F=[str(item) for item in range(X.shape[1])]
        # First it selects the feature most correlated to the target
        correlations = np.corrcoef(X, Y, rowvar=0)
        self.selected_feat = np.argmax(correlations[0:-1, -1])
        best_feat = X[:, self.selected_feat]
        print('Feature selected = ' +  F[self.selected_feat])
        # Then it computes the average values of the 3 classes
        mu0 = np.median(best_feat[Y==0])
        mu1 = np.median(best_feat[Y==1])
        mu2 = np.median(best_feat[Y==2])
        # Finally is sets two decision thresholds
        self.theta1 = (mu0+mu1)/2.
        self.theta2 = (mu1+mu2)/2.

    def predict(self, X):
        ''' The method "predict" classifies new test examples '''
        # Select the values of the correct feature
        best_feat = X[:, self.selected_feat]
        # Initialize an array to hold the predicted values
        Yhat = np.copy(best_feat)               # By copying best_fit we get an
        # then classify using the selected feature according to the cutoff thres
        Yhat[best_feat<self.theta1] = 0
        Yhat[np.all([self.theta1<=best_feat, best_feat<=self.theta2], 0)] = 1
        Yhat[best_feat>self.theta2] = 2
        return Yhat
```

In [18]:

```
XY=data_num.as_matrix()             # On transforme le Pandas dataframe en un Nu
X = XY[:,0:4]                       # On recupere X (num_exemples x num_features
Y = XY[:,4]                         # et Y (num_exemples x 1) ==> les valeurs de
feature_names = list(data_num)[:-1] # On recupere aussi les noms des features
my_model = oneR()
my_model.fit(X, Y, feature_names)   # Le nom des features est "optionnel", il pe
Yhat = my_model.predict(X)
```

```
Feature selected = petal_length
```

In [19]:

```
def error_rate(solution, prediction):
    return np.mean(solution!=prediction)

errate = error_rate(Y, Yhat)
print('Training error = %5.2f' % errate)
Yperm = np.random.permutation(Y)
print('Random permutation error (for comparison)= %5.2f' % error_rate(Y, Yperm))
print('Ideal error rate (for comparison)= %5.2f' % error_rate(Y, Y))
```

```
Training error =  0.06
Random permutation error (for comparison)=  0.63
Ideal error rate (for comparison)=  0.00
```

La question 8 est ici:

Remplacez xxx par la bonne function qui calcule le BAC (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn.metrics.balanced_a i.e. le Balanced ACcuracy score, et enlevez les # de commentaire. Observez que error_rate est different de Balanced Error Rate BER=1-BAC.

In [20]:

```
BER = errate
#from sklearn.metrics import xxx
#BAC = xxx(Y, Yhat)
#print('BAC = %5.2f' % BAC)
#BER = 1 - BAC
from sklearn.metrics import balanced_accuracy_score
BAC = balanced_accuracy_score(Y, Yhat)
print('BAC = %5.2f' % BAC)
BER = 1 - BAC
print('BER = %5.2f' % BER)
question = 8
score += check(BER, question)
```

```
BAC =  0.89
BER =  0.11
0.10555555555555551
0.10555555555555551
```

CORRECT
:-)

Question 9: Viewing the classification results

Pour se faire une idee de comment marche les classifieurs il est utile de visualiser les regions de decision. Ce petit programme permet de le faire en deux dimensions.

In [21]:

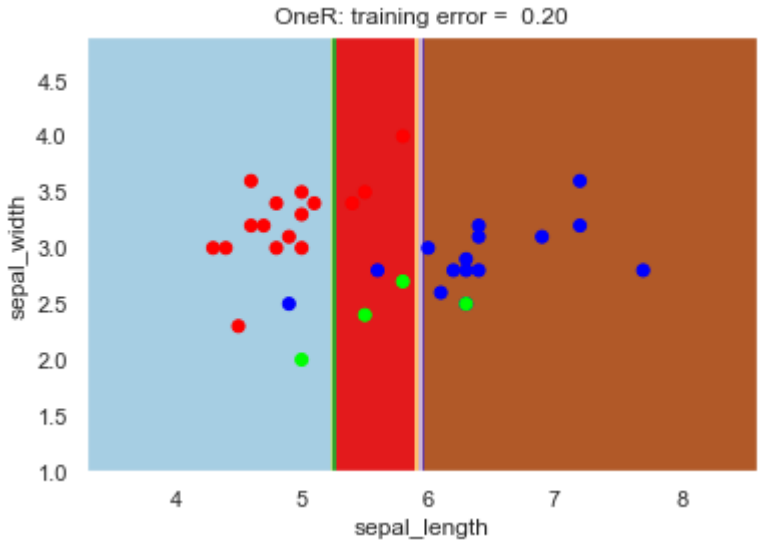
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
colors = [(1, 0, 0), (0, 1, 0), (0, 0, 1)] # Red, lime, blue
cm = LinearSegmentedColormap.from_list('rgb', colors, N=3)

def ClfScatter(clf, X, Y, F, dim1=0, dim2=1, title=''):
    '''clf_scatter(clf, X, Y, F, dim1=0, dim2=1)
    Display decision function and training examples.
    clf: a classifier with at least a fit and a predict method
    like a sckit-learn classifier.
    X: a 2 dimensional data matrix, samples in line and features in columns
    Y: a target vectors of class values 0, 1, 2, etc.
    F: feature names
    dim1 and dim2: chosen features.
    title: Figure title.
    Returns: Predictions on training examples.
    '''
    # Fit model in chosen dimensions
    X2 = X[:,(dim1,dim2)]
    try:
        clf.fit(X2, Y, F=[F[dim1],F[dim2]])
    except:
        clf.fit(X2, Y)
    # Define a mesh
    x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() + 1
    y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() + 1
    h = 0.1 # step
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Xtest = np.c_[xx.ravel(), yy.ravel()]
    # Compute the training error
    Yhat = clf.predict(X2)
    training_error = error_rate(Y, Yhat)
    # Make your predictions on all mesh grid points (test points)
    Yhat = clf.predict(Xtest)
    # Make contour plot for all points in mesh
    Yhat = Yhat.reshape(xx.shape)
    plt.contourf(xx, yy, Yhat, cmap=plt.cm.Paired)
    # Overlay scatter plot of training examples
    plt.scatter(X2[:, 0], X2[:, 1], c=Y, cmap=cm)
    plt.title('{}: training error = {:5.2f}'.format(title, training_error))
    plt.xlabel(F[dim1])
    plt.ylabel(F[dim2])
    plt.show()
    return clf.predict(X2)
```

In [22]:

```
Yhat = ClfScatter(my_model, X, Y, feature_names, dim1=0, dim2=1, title='OneR')
errate = error_rate(Y, Yhat)
print('Training error = %5.2f' % errate)
```

```
Feature selected = sepal_length
```




```
Training error =  0.20
```

La question 9 est ici:

Replacez dim1=2, dim2=3 par dim1=0, dim2=1 dans la cellule du dessus et re-executez la. Vous devriez obtenir de moins bonnes performances. Pensez vous qu'un classifieur qui utiliserait plusieurs regles pourrait obtenir 0 erreurs?

In [23]:

```
question = 9
reponse = 1 # Oui  = 1,  Non  = 0
score += check(errate+reponse, question)
```

```
1.2
1.2
```

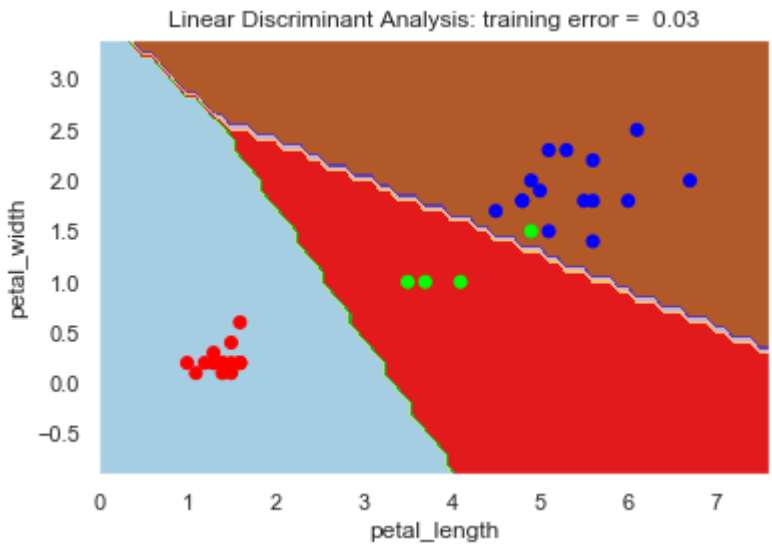
CORRECT
:-)

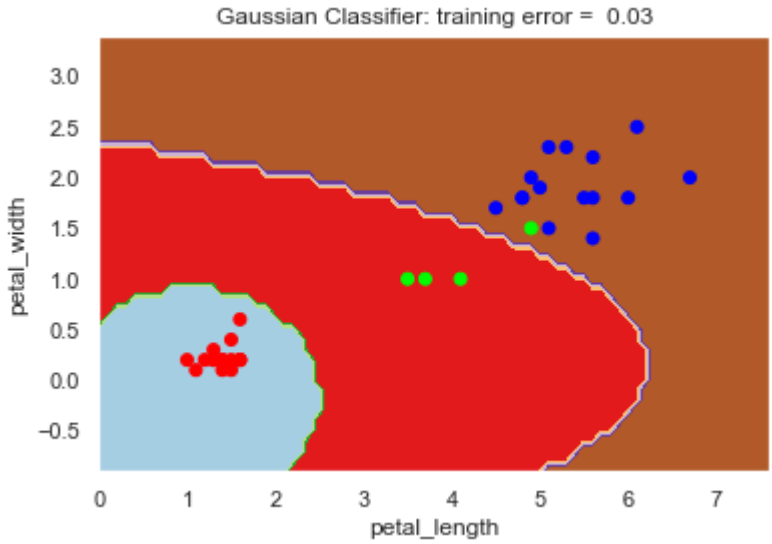
Question 10: More classifiers

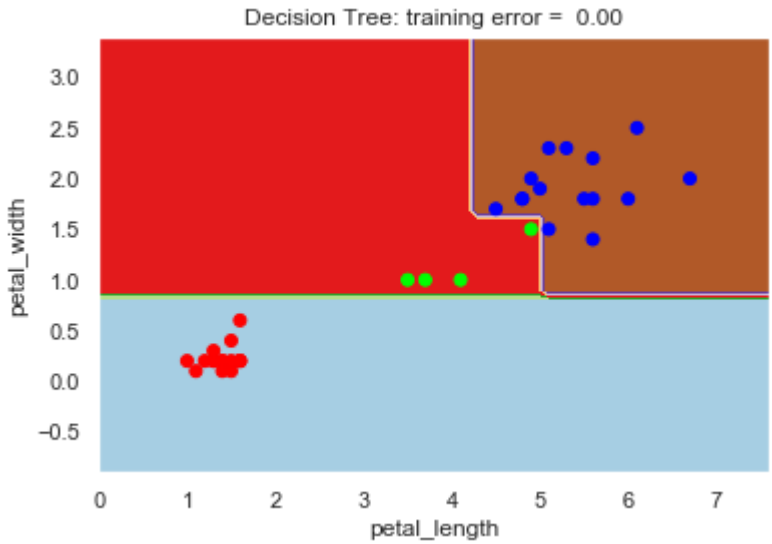
En utilisant le code precedant, on peut comparer plein de classifieurs. Certains obtiennent les memes resultats si on les fait tourner plusieurs fois, d'autres donne toujours la meme chose. Combien d'entre eux (parmi les exemples ci-dessous) donnent toujours la meme chose? Verifiez bien dans la doc de scikit-learn (https://scikit-learn.org/stable/).

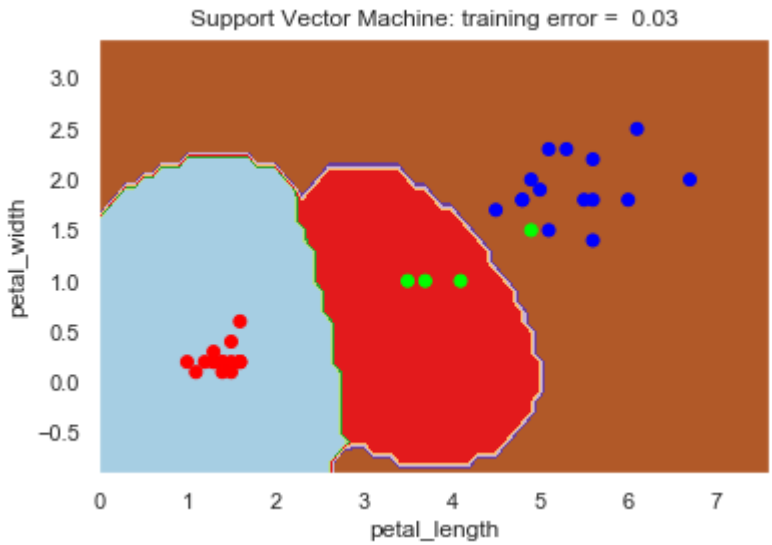
In [24]:

```
import random
from sklearn.linear_model import Perceptron
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
classifiers = [
    Perceptron(random_state=random.randint(1,101)),
    LinearDiscriminantAnalysis(),
    GaussianNB(),
    DecisionTreeClassifier(random_state=random.randint(1,101)),
    SVC()]
names = ['Perceptron',
         'Linear Discriminant Analysis',
         'Gaussian Classifier',
         'Decision Tree',
         'Support Vector Machine']
for clf, name in zip(classifiers, names):
    # This does a two dimensional fit in dim1 and dim2
    ClfScatter(clf, X, Y, feature_names, dim1=2, dim2=3, title=name)
```




Gaussian Classifier: training error = 0.03
petal_width
petal_length
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7


Decision Tree: training error = 0.00
petal_width
petal_length
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7


Support Vector Machine: training error = 0.03
petal_width
petal_length
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7

In [25]:

```
question = 10
reponse = 3 # Nombre de classifieurs qui retournent toujours la meme separation
score += check(reponse, question)
```

3
3

CORRECT
:-)

In [26]:

```
print('Your final score is %d / 10, congratulations!' % score)
```

```
Your final score is 10 / 10, congratulations!
```

In [ ]:

```

```