Séance 1

Question 1:

150 entries
variable cible : Species
3 classes : setosa, versicolor, virginica
5 attributs :

- id : identifiant de la fleur (pas utile)
- SepalLengthCm : la longueur de la sépale (utile)
- SepalWidthCm : la largeur de la sépale (utile)
- PetalLengthCm : la longueur de la pétale (utile)
- PetalWidthCm : la largeur de la pétale (utile)

répartition :

- Iris-setosa 50
- Iris-versicolor 50
- Iris-virginica 50

Question 2 :

La première est à ignorer car on classe en fonction de l'id.
sepalwidth en fonction de sepallength : tout est mélangé, du coup pas utile
sinon les autres sont bien séparés
personnellement le petalwidth en fonction du petallength me paraît être le mieux.

Question 3

a)

⅔ pour l'apprentissage, du coup 100 (150 échantillons en tout)

b)

'Iris-setosa' : 50
Iris-versicolor' : 50
'Iris-virginica' : 0

on va s'entraîner sur les setosa et les versicolor et tester sur les virginica, ce qui est pas pertinent du tout

c)

'Iris-setosa' : 33
Iris-versicolor' : 31
'Iris-virginica' : 36
c’est plus pertinent car c’est mieux réparti entre les 3 espèces

Question 4

le classifieur se trompe 4 fois

```
for i in range(len(y_pred)):
    if y_pred[i] != y_test[i]:
        print(y_pred[i],y_test[i])
```

qui me rend

Iris-setosa Iris-versicolor
Iris-virginica Iris-versicolor
Iris-virginica Iris-versicolor
Iris-versicolor Iris-virginica

Question 5

Accuracy: 0.92 = TP + TN / (TP + TN + FP + FN)

Question 6

erreur réelle 0.07999999999999996 (erreur entre la valeur de test et sa prédiction)
erreur empirique 0.06999999999999995 (erreur entre la valeur d'entraînement et sa prédiction)

savoir si notre entraînement donne des résultats corrects. Si les 2 erreurs sont similaires et petites, notre modèle est bien.

Question 7

[17 0 0]
[1 16 2]
[0 1 13]

	Prédites setosa	Prédites versicolor	Prédites virginica
Réelles setosa	setosa	setosa prédite en versicolor	setosa prédite en virginica

Réelles versicolor	versicolor prédite en setosa	versicolor	versicolor prédite en virginica
Réelles virginica	virginica prédite en setosa	virginica prédite en versicolor	virginica

Question 8

- Retrouvez la valeur de l'accuracy à partir de cette matrice

sommeDiagonale / total = (16+17+13)/(17+16+1+2+1+13) = 0.92

- Quelle est la précision pour la classe 'Iris-virginica' ?

TP/(TP+FP) = 13/(13+2+0) = 0,8666667

- Quel est le rappel pour la classe 'Iris-virginica' ?

TP/(TP+FN) = 13/(13+1+0) = 0,9285714

- Quelle est la classe qui a le meilleur rappel ? Qu'est-ce que cela signifie ?

Rappel(Virginica) = 0,93

Rappel(versicolor) = 16/(16+2+1) = 0,84

Rappel(setosa) = 17/(17+0+0) = 1

La setosa a le meilleur rappel, cela veut dire que la plus grande proportion de setosa a été prédit en tant que setosa, ici elles le sont toutes car égale à 1

- Quelle est la classe qui a la plus faible précision ? Qu'est-ce que cela signifie ?

Précision(Virginica) = 0,87

Précision(versicolor) = 16/(16+1+0) = 0,94

Précision(setosa) = 17/(17+1+0) = 0,94

La virginica est la moins précise, elle a beaucoup de versicolor et setosa considéré comme virginica.

Question 9

13

Séance 2

Question 10

0 : SepalLengthCm
1 : SepalWidthCm
2 : PetalLengthCm
3 : PetalWidthCm

	0	1	2	3
0	0,68	0,74	0.9	0,9
1	0,74	0,56	0,96	0,92
2	0,9	0,96	0,96	0,98
3	0,92	0,92	0,98	0,96

en rouge résultat pas logique, ou différent par rapport aux autres, je sais pas pourquoi (ils ont 0,02 en plus)

petallength et petalwidth
ce qui est cohérent avec ce que j'avais dis

DataToy

1
small data toy: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
train_set1 : [1, 4, 2, 8, 9, 6]
test_set1 : [3, 10, 7, 5]
20
small data toy: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
train_set1 : [1, 3, 7, 10, 5, 4]
test_set1 : [8, 2, 9, 6]
100
small data toy: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
train_set1 : [5, 3, 1, 4, 10, 9]
test_set1 : [8, 7, 2, 6]

Question 11

1 Accuracy on test1: 0.92
20 Accuracy on test2: 0.9
34 Accuracy on test3: 0.88
45 Accuracy on test4: 0.86
103 Accuracy on test5: 0.74

plus on l'augmente, moins c'est accurate, du coup moins performant

Question 12

On fait k sous échantillons de notre ensemble, on crée un modèle sur k-1 de ces ensembles, et on teste sur le k-ieme. On teste avec tous les échantillons pour obtenir un score pour chaque échantillon. Cela permet de mieux évaluer le biais et la variance de notre prédiction.

Question 13

[0.9 0.93333333 0.86666667 0.93333333 0.93333333]
0.91 accuracy with a standard deviation of 0.03

Il renvoie les scores/accuracy de chacun des plis. Il y a 5 plis.

Question 14

- On binarise les données : inférieur à 0.5 on met à 0 et supérieur à 0.5 à 1
- On enlève la moyenne et l'écart type des données d'entrées
- On définit le min à 0 et le max à 1 pour chaque ligne, et les autres on les re-calcule en fonction du min et max.
- On normalise les données

Question 15

TODO: Sélectionner et adapter ici les 6 à 8 lignes de code permettant de calculer l'accuracy sur le jeu de donnée en lui appliquant un pré-traitement

```
Xplot1=Xplot
Xplot2=preprocessing.scale(Xplot)
Xplot3=data_scaler_minmax.fit_transform(Xplot)
Xplot4=preprocessing.Normalizer().fit_transform(Xplot)

Xplot_train, Xplot_test, yplot_train, yplot_test = train_test_split(Xplot1, yv, test_size = 1./3,
random_state = 1)
yplot_pred = classifier_plot.predict(Xplot_test)
print("Accuracy", metrics.accuracy_score(yplot_test, yplot_pred))

Xplot_train, Xplot_test, yplot_train, yplot_test = train_test_split(Xplot2, yv, test_size = 1./3,
random_state = 1)
yplot_pred = classifier_plot.predict(Xplot_test)
print("Accuracy", metrics.accuracy_score(yplot_test, yplot_pred))

Xplot_train, Xplot_test, yplot_train, yplot_test = train_test_split(Xplot3, yv, test_size = 1./3,
random_state = 1)
yplot_pred = classifier_plot.predict(Xplot_test)
print("Accuracy", metrics.accuracy_score(yplot_test, yplot_pred))

Xplot_train, Xplot_test, yplot_train, yplot_test = train_test_split(Xplot4, yv, test_size = 1./3,
random_state = 1)
yplot_pred = classifier_plot.predict(Xplot_test)
print("Accuracy", metrics.accuracy_score(yplot_test, yplot_pred))
```