TP1

November 28, 2019

Apprentissage par la méthode des k-nn

Prendre un échantillon de données appelé data avec une taille de 5000 exemples à l'aide de la fonction np.random.randint(70000, size=5000). Diviser la base de données à 80% pour l'apprentissage (training) et à 20% pour les tests. Entrainez un classifieur k-nn avec k = 10 sur le jeu de données chargé.

[2]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='uniform')
```

Afficher la classe de l'image 9 et sa classe prédite

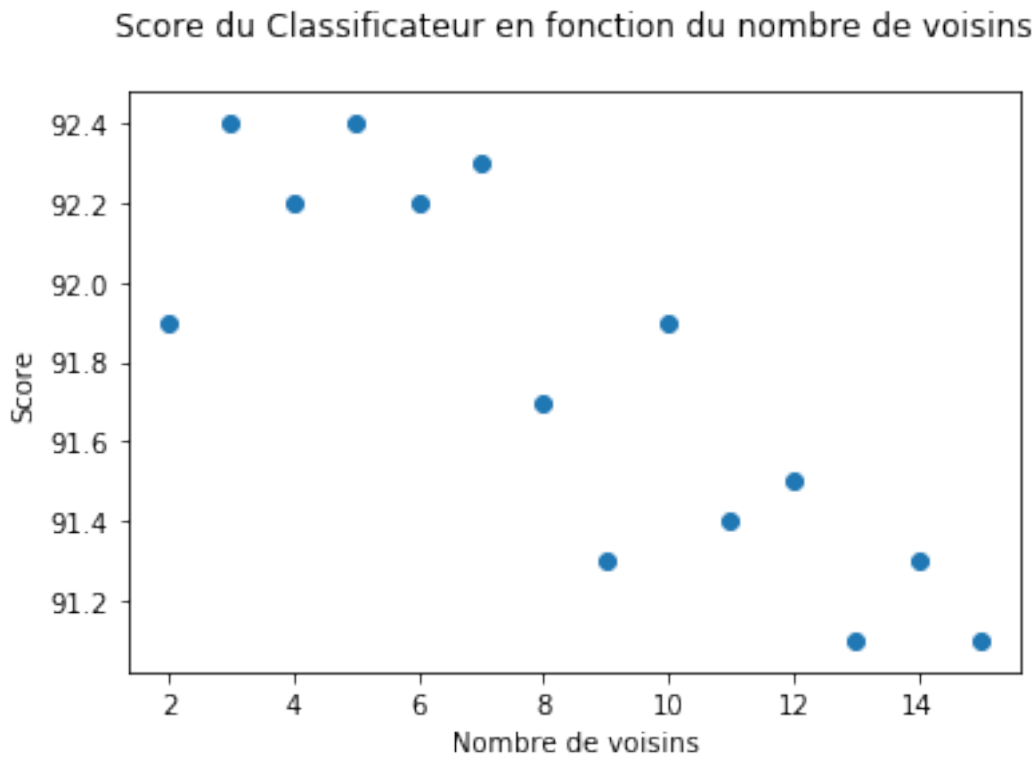
```
Le classificateur prédit: ['9'], et le réel nombre est: 9
```

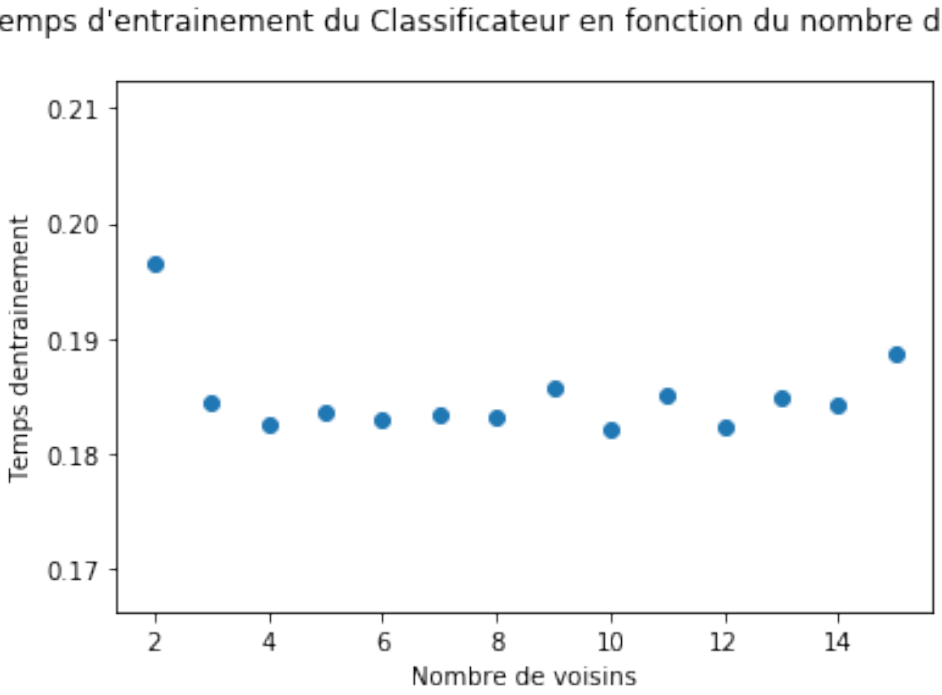
Affiche le score sur l'échantillon de test. Quel est le taux d'erreur sur vos données d'apprentissage ? Est-ce normal ?

```
Score: 91.9
Taux d'erreur: 8.099999999999994

A première vue, le taux d'erreur peut paraitre anormalement grand (environ 8%)
malgré le fait qu'on ait un grand  nombre de voisins (k=10). Mais en réalité, ce
score est tout à fait normal car quand il y a trop de voisins, cela diminue la
précision. On verra cela plus tard, pour détérminer à quel niveau environ se
situe ce seuil.
```

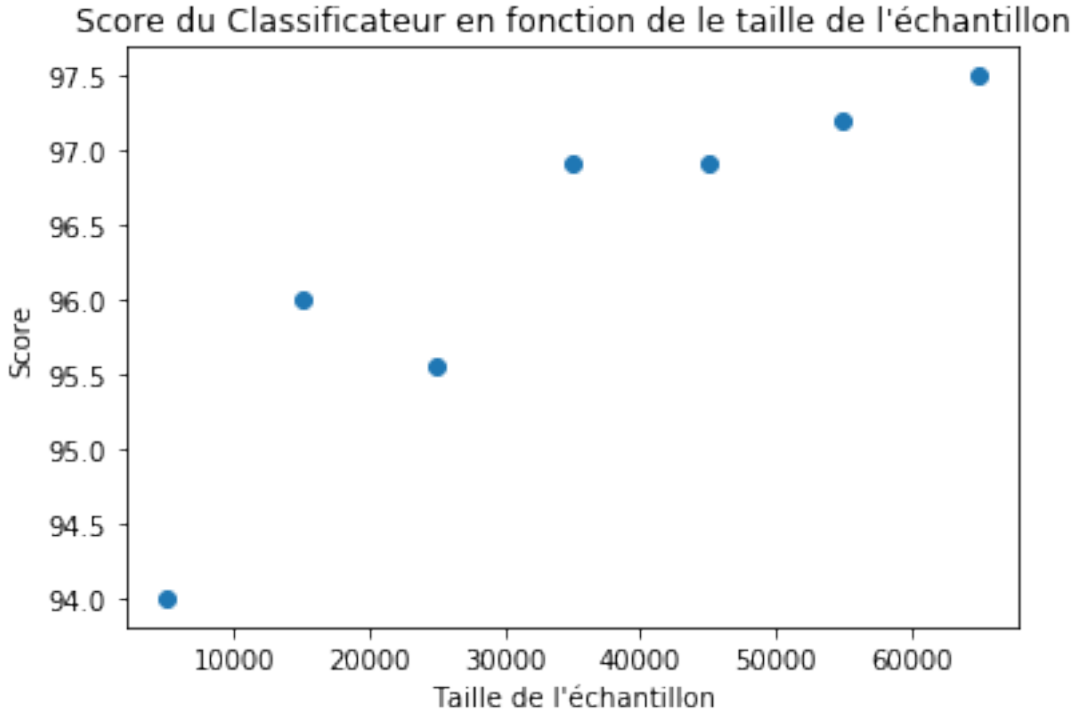
Faites varier le nombre de voisins (k) de 2 jusqu'à 15 et afficher le score. Quel est le k optimal ?

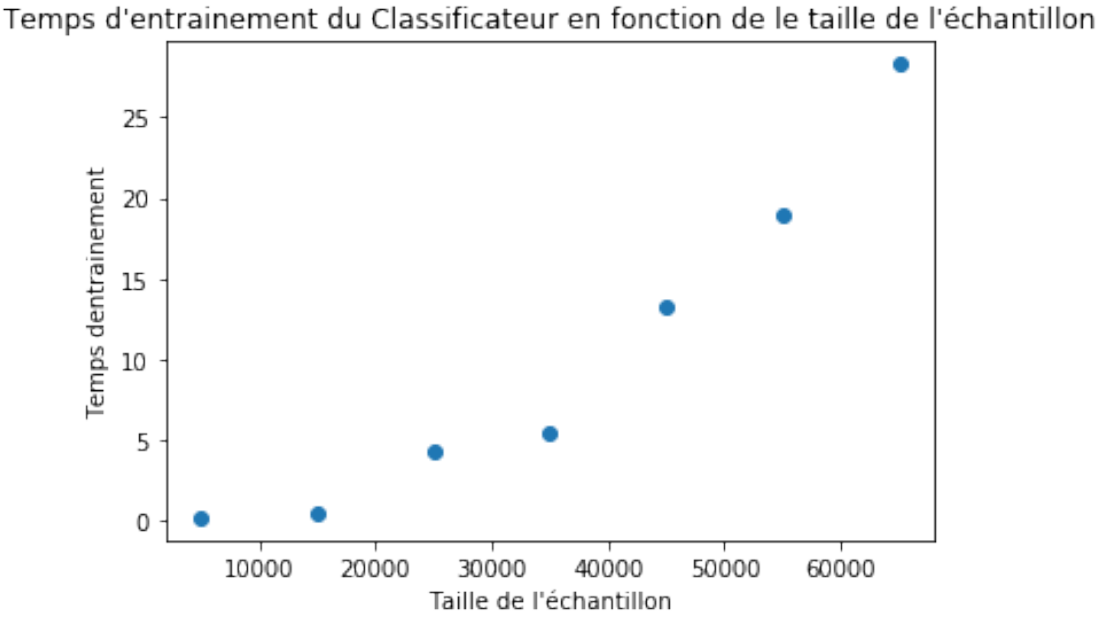

Score du Classificateur en fonction du nombre de voisins
Score
92.4
92.2
92.0
91.8
91.6
91.4
91.2
2
4
6
8
10
12
14
Nombre de voisins


Temps d'entrainement du Classificateur en fonction du nombre de voisins
Temps dentrainement
0.21
0.20
0.19
0.18
0.17
2
4
6
8
10
12
14
Nombre de voisins

`k est optimal pour les valeurs 4 et 5.`

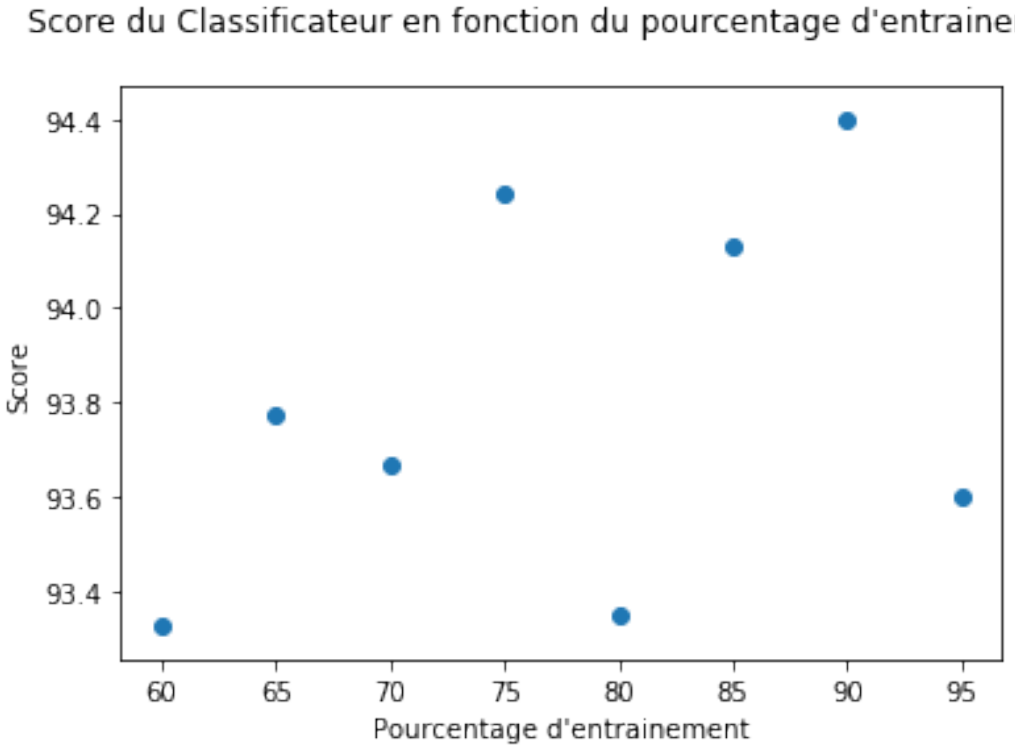
Faites varier le pourcentage des échantillons (training et test) et afficher le score. Quel est le pourcentage remarquable ?

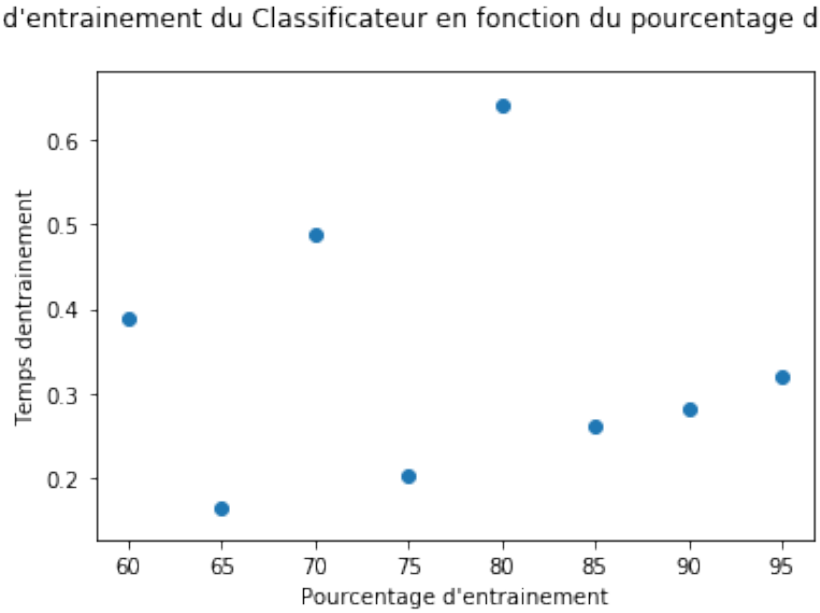




On remarque que plus l'échantillon est grand, plus le classificateur met de temps, surtout à partir des échantillons supérieurs à 2500. On remarque aussi que plus la taille de l'échantillon est grande, plus le score est meilleure, et on se rapproche des 100%

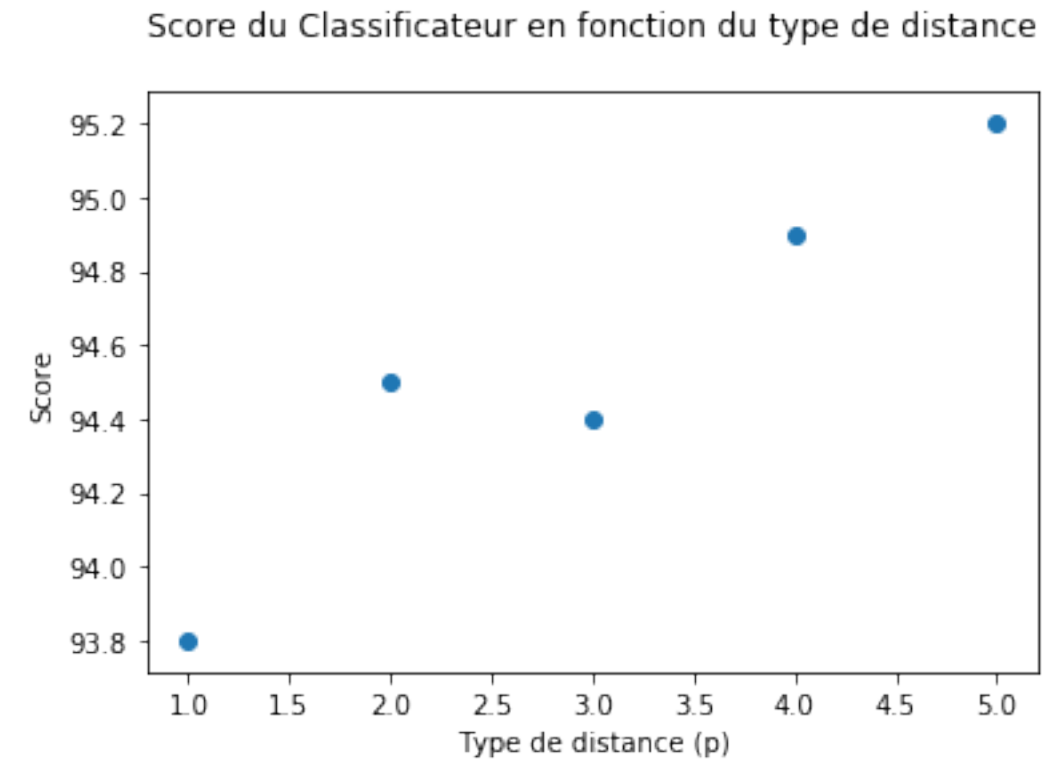
Faites varier la taille de l'échantillon training et afficher la précision. Qu'est-ce que vous remarquez ?

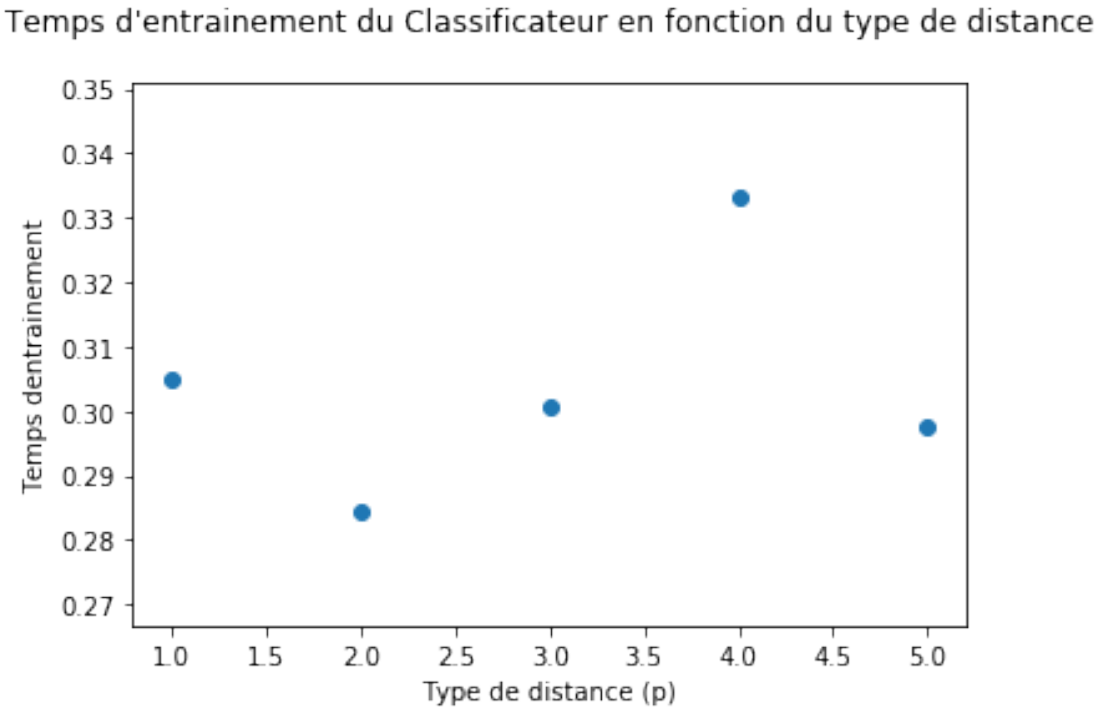




On remarque le max est atteint avec 90% d'entraînement et 10% de test et que au delà le score diminue, certainement car il n'y a quasiment plus de part de test

Faites varier les types de distances (p). Quelle est la meilleure distance ?


Score du Classificateur en fonction du type de distance
Score
95.2
95.0
94.8
94.6
94.4
94.2
94.0
93.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Type de distance (p)


Temps d'entrainement du Classificateur en fonction du type de distance
Temps dentrainement
0.35
0.34
0.33
0.32
0.31
0.30
0.29
0.28
0.27
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Type de distance (p)

La mesure Euclidienne apporte de meilleures résultat que la distance de Manhattan. Les métriques suppérieurs à 3 offrent de meilleurs résultats que les 2 précédentes mais on ne connaît pas leurs natures.

Fixer n_job à 1 puis à -1 et calculer le temps dans chacun

n_jobs= -1
score: 92.60000000000001
error: 7.3999999999999915

n_jobs= 1
score: 92.60000000000001
error: 7.3999999999999915

A votre avis, quels sont les avantages et les inconvénients des k-nn : optimalité ? temps de calcul ? passage à l'échelle ?

Avantages:

- Possibilité de réaliser des prédictions
- Algorithme simple
- Adapté aux modèles avec des frontières sont irrégulières, comme des chiffres dans notre cas

Inconvénients:

- Non adapté aux données massives
- Obligation de stocker tous l'apprentissage en mémoire
- Prédiction plutôt lente, car nécessité de calculer les distances pour chaque points

TP2

November 28, 2019

Apprentissage par perceptron multi-couche

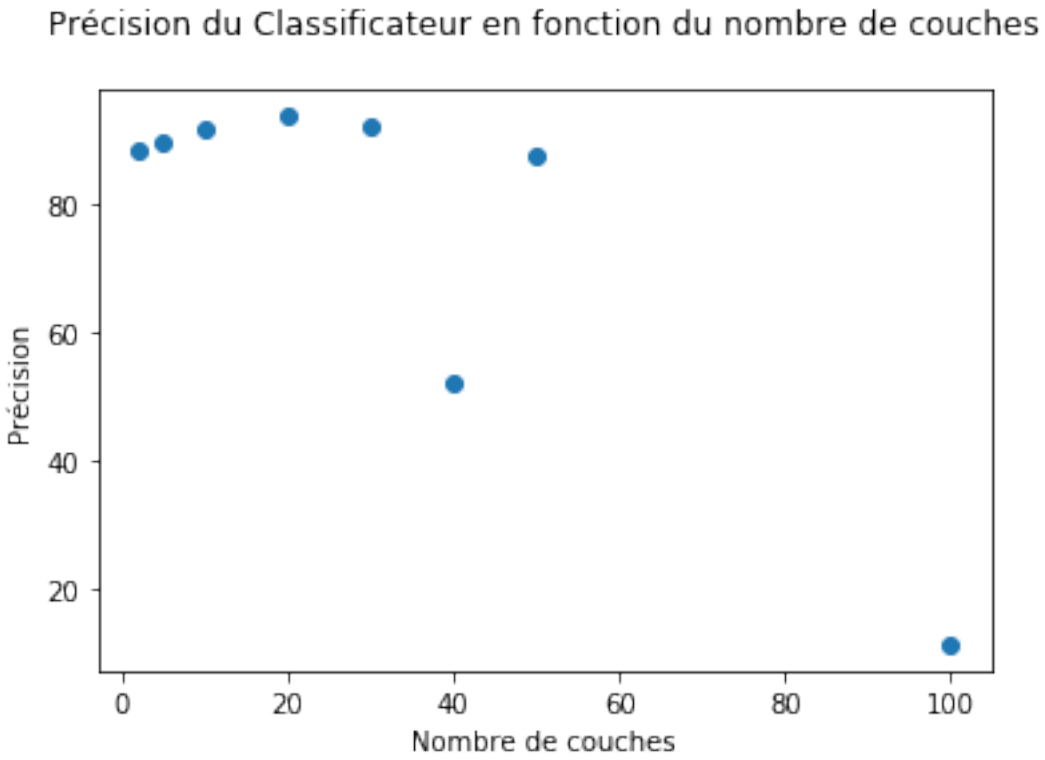
Construire un modèle de classification ayant comme paramètre : hidden_layer_sizes = (50), puis calculez la précession du classifieur

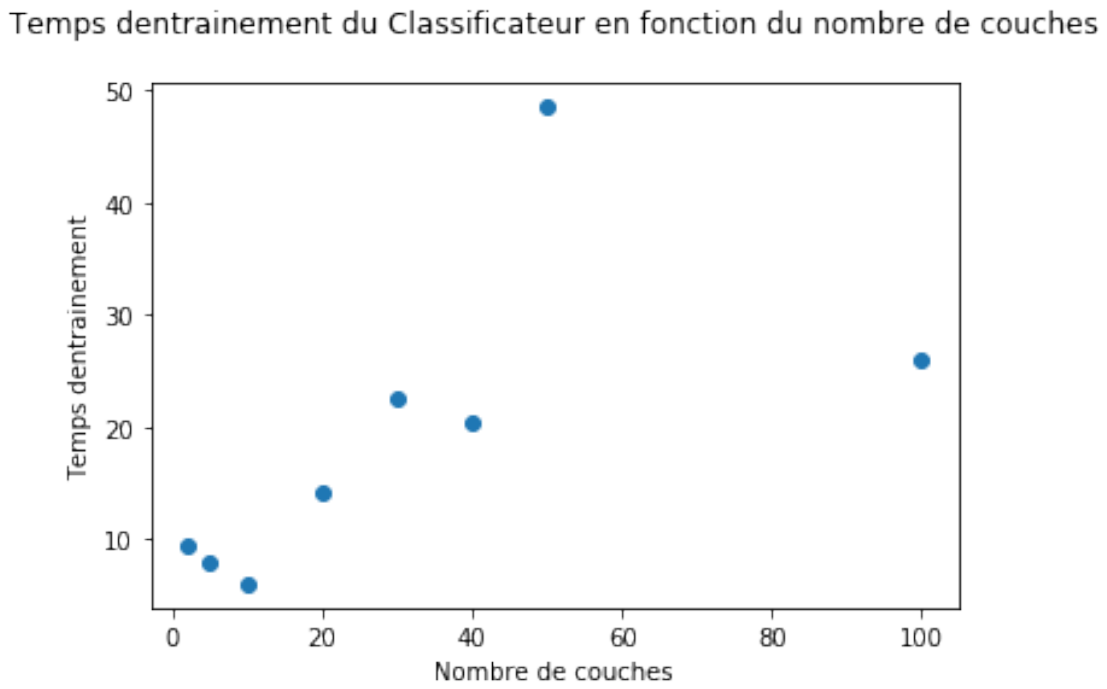
```
Score: 88.23809523809524
Error: 11.76190476190476
```

Afficher la classe de l'image 9 et sa classe prédite

```
Prediction = ['4'] et nombre réel = 9
```

Varier le nombre de couches de 1 entre (2 et 100) couches, et recalculer la précision du classifieur





Construire 2 modèles de classification des données mnist, avec des réseaux qui ont respectivement :

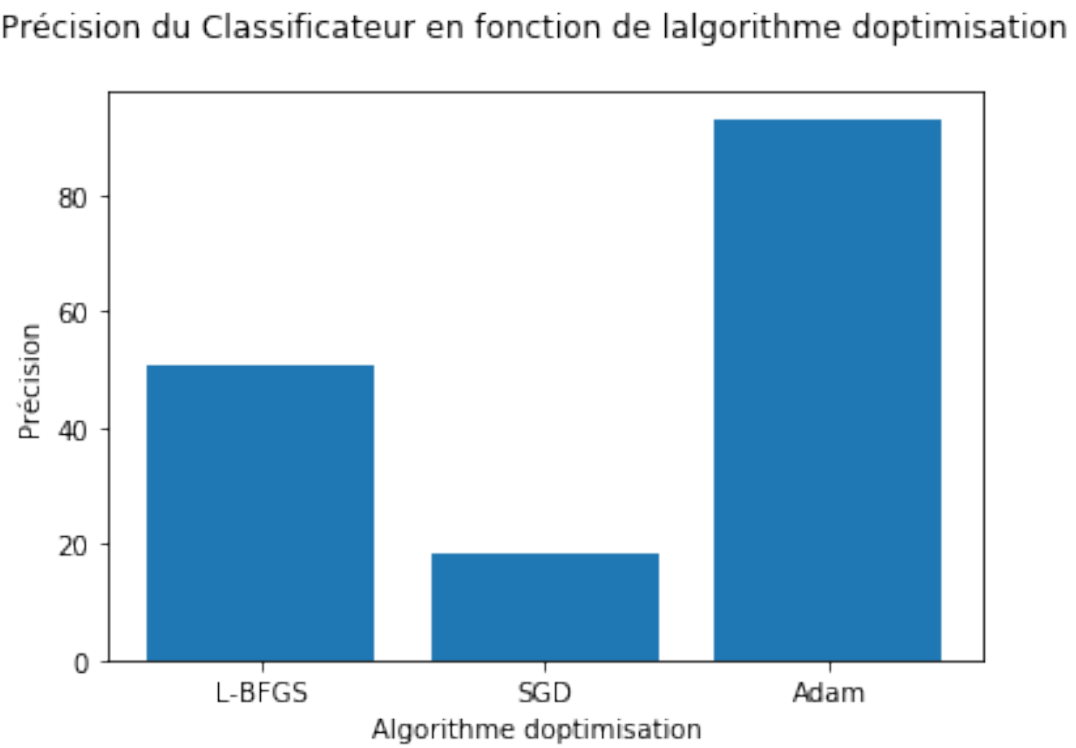
- Modèle1: 50 couches cachées, et des tailles de couches de manière regressives de 60 à 11 par pas de -1
- Modèle2: 20 couches cachées, et des tailles de couches de manière regressives de 60 à 33 par pas de -3, puis -2

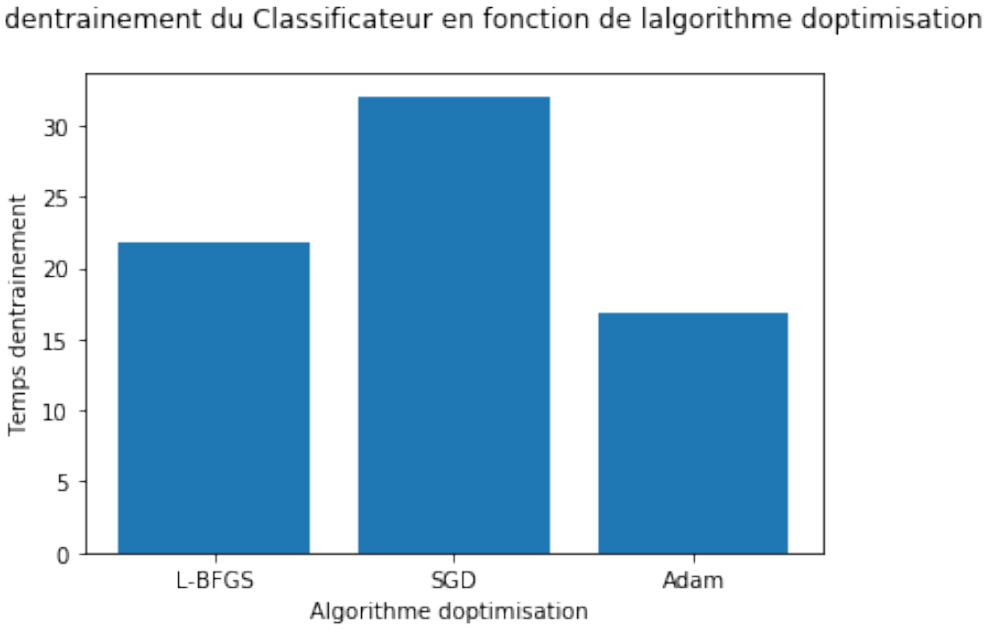
Quelles sont les performances en taux de bonne classification et en temps d'apprentissage obtenus pour chaque modèle ? Utilisez la fonction time() du package time pour mesurer le temps d'apprentissage d'un modèle.

```
 - Modèle 1 :
Score = 11.333333333333332
Temps d'entrainement = 13.117745876312256

 - Modèle 2 :
Score = 92.61904761904762
Temps d'entrainement = 20.750596523284912
```

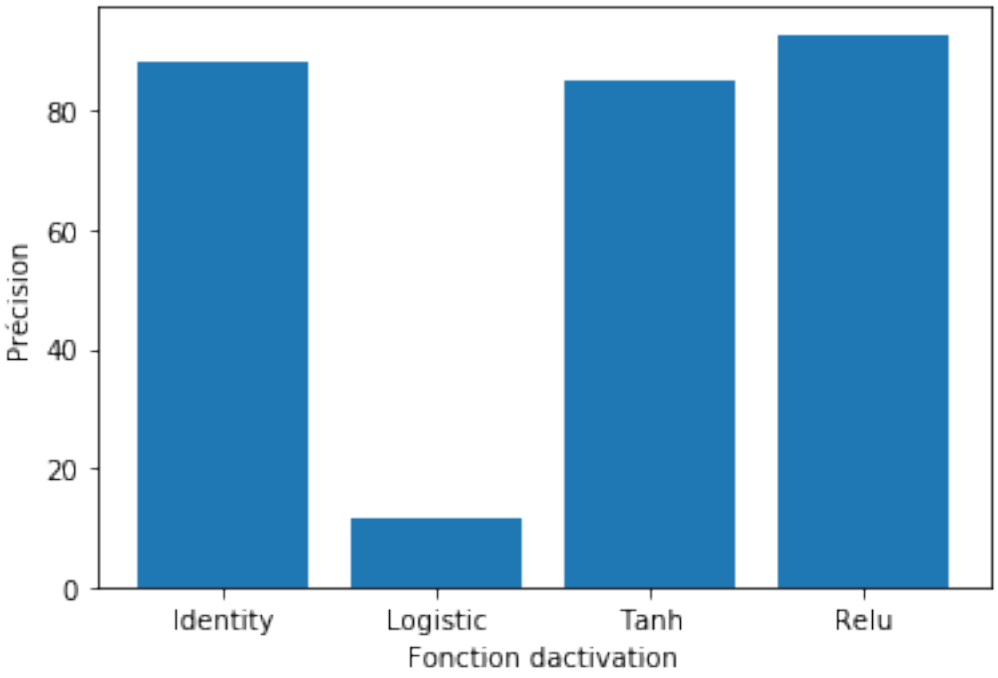
Etudier la convergence des algorithmes d'optimisation disponibles : L-BFGS, SGD et Adam



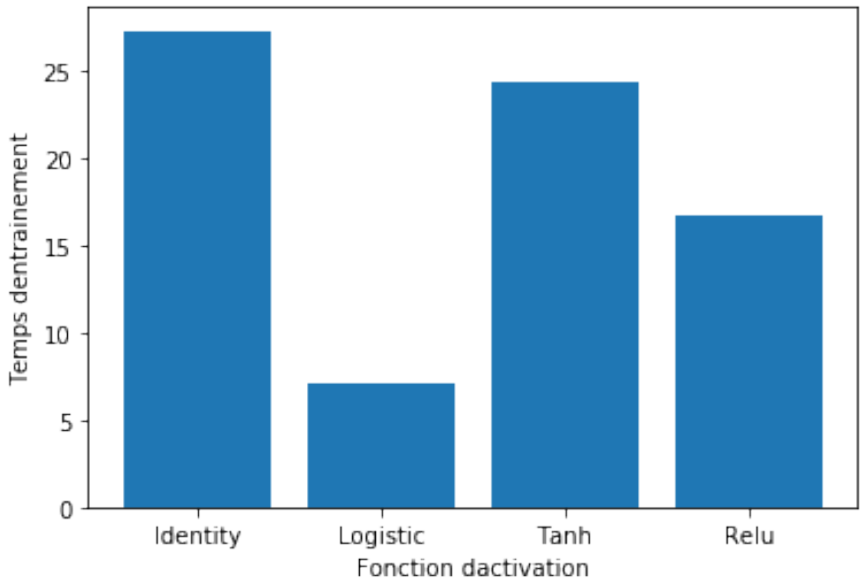


Varier les fonctions d'activation {'identity', 'logistic', 'tanh', 'relu'}

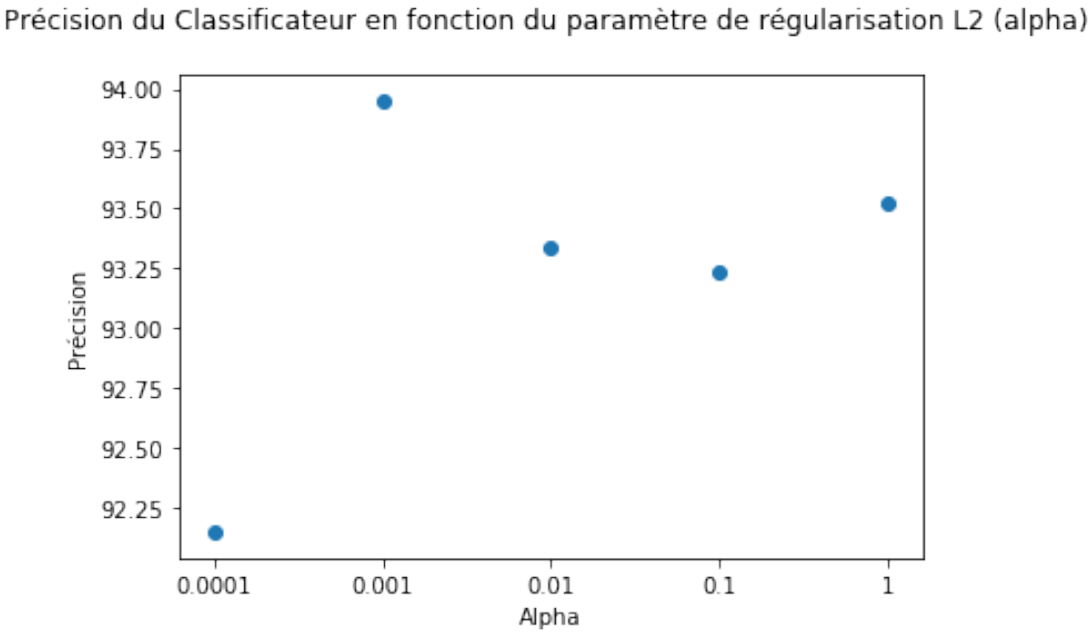
Précision du Classificateur en fonction de la fonction dactivation

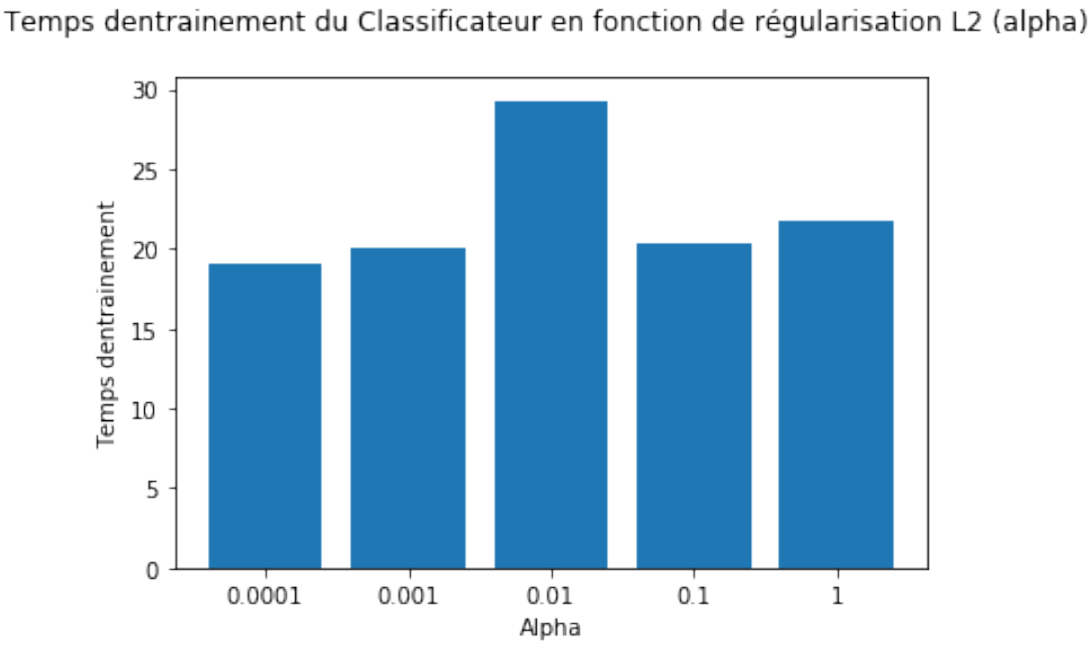


Temps dentrainement du Classificateur en fonction de la fonction dactivation



Changer la valeur de la régularisation L2 (paramètre)





Choisissez le modèle qui propose de meilleurs résultats Au vu de toutes les expérimentations vues jusqu'à présent, le modèle qui propose de meilleurs résultats est un modèle ayant comme

paramètres: - Un nombre de couches compris aux alentours de 20 (entre 15 et 25 environ, selon le 1er graphique), en méthode régréssive avec un nombre de neuronnes compris entre 60 et 10 meme avec un pas d'environ -2 ou -3 - Adam, comme comme algo d'optimisation - Relu, comme fonction d'activation. Les autres sont assez proches en terme de résultat, à l'expection de 'logistic' qui n'est pas précise - Un paramètre de regulation le plus petit possible (environ 0.0001 ou moins)

A votre avis, quels sont les avantages et les inconvénients des A-nn : optimalité ? temps de calcul ? passage à l'échelle ?

Avantages:

- Tolérance aux fautes: Meme si un ou plusieurs neuronnes sont corrumpus, cela n'empeche pas la géné
- Capacité à faire du calcul en parallèle
- Capacité à générer des estimations pertinentes à partir de peu d'exemples

Inconvénients:

- Architecture: Il n'existe pas vraiment de moyens permettant de définir l'architecture optimale du
- Décisions prises qui sont difficiles à analyser ou expliquer
- Difficultés à estimer les temps de calcul du A-nn

TP3

November 28, 2019

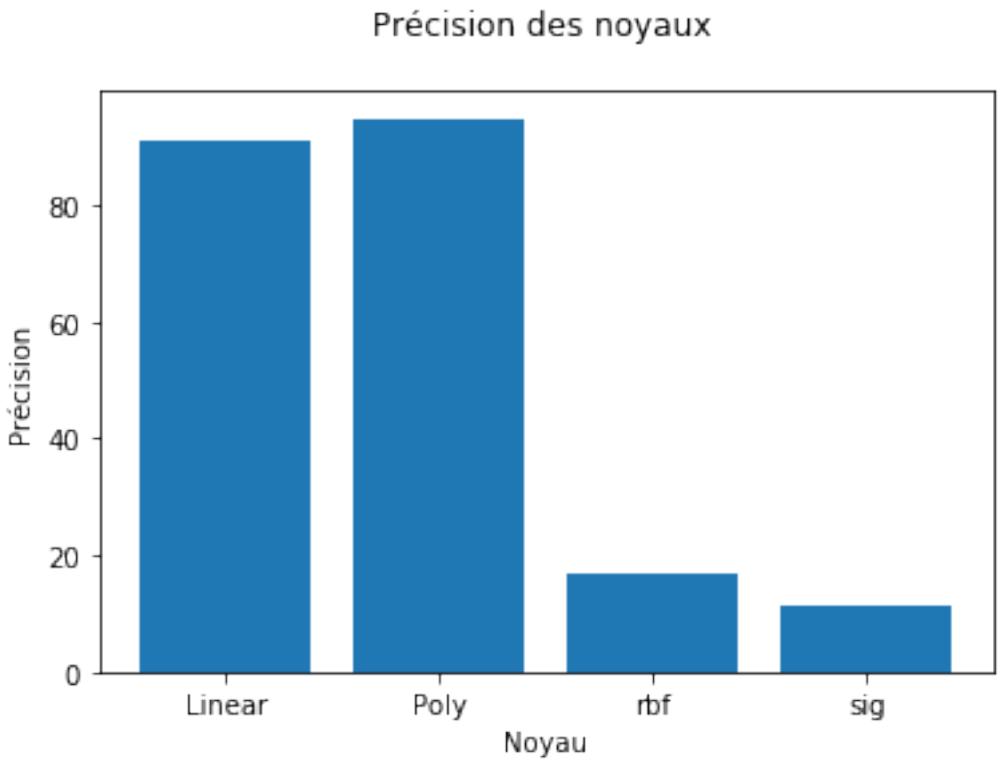
Apprentissage par Machines à Vecteurs de Support

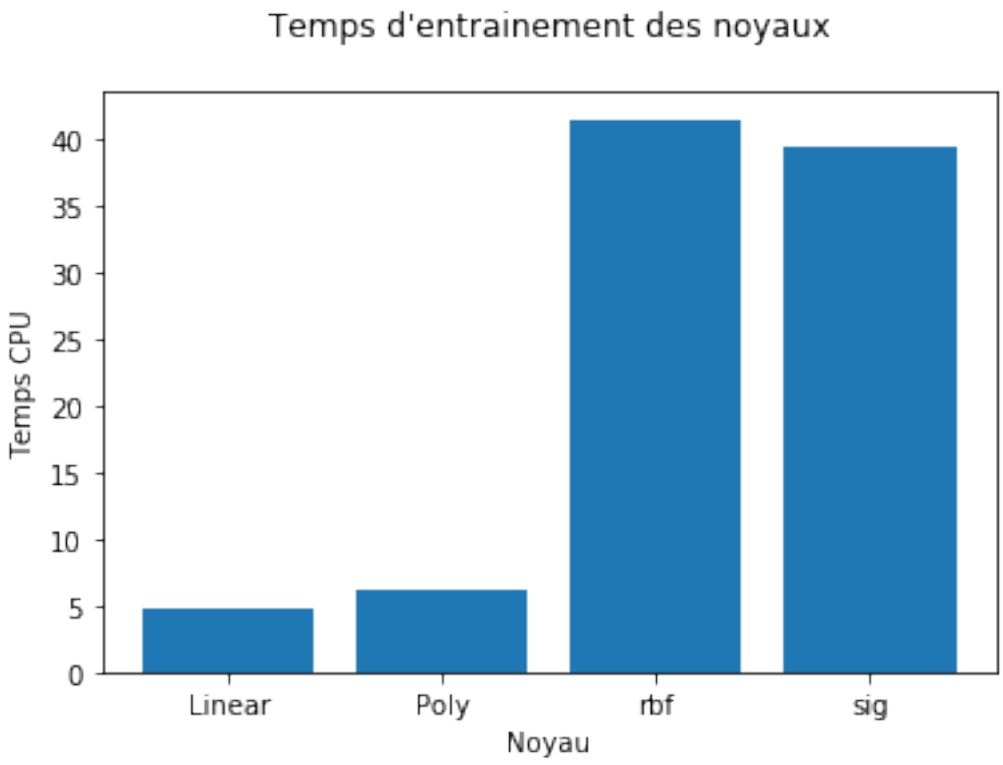
Construire un modèle de classification ayant comme paramètres un noyau linear

```
precision score linear = 90.85714285714286
```

Tentez d'améliorer les résultats en variant la fonction noyau : 'poly', 'rbf', 'sigmoid', 'precomputed'

```
precision score poly = 94.80952380952381
precision score rbf = 16.952380952380953
precision score sigmoid = 11.285714285714285
```





Faites varier le paramètre de tolérance aux erreurs C

```
precision score for c = 0.2 = 94.80952380952381
precision score for c = 0.4 = 94.80952380952381
precision score for c = 0.6 = 94.80952380952381
precision score for c = 0.8 = 94.80952380952381
precision score for c = 1.0 = 94.80952380952381
```

Tracez la courbe d'erreur de classification sur les données d'entrainement et de test en fonction de C En variant la tolérance aux erreurs C, on obtient toujours les mêmes résultats. Ainsi l'erreur de classification sur les données est la même

Construisez la matrice de confusion en utilisant le package metrics

```
CM linéaire :

[[196   0   0   0   0   2   0   1   1   0]
 [  0 233   1   2   0   0   0   0   1   0]
 [  2   4 187   1   3   1   2   1   1   0]
 [  1   1   6 190   0   9   0   2   4   2]
 [  1   2   3   0 179   0   2   2   0   9]
```

```
 [  4   4   2  10   0 156   2   0   2   0]
 [  1   1   6   0   1   2 196   0   1   0]
 [  0   3   3   2   1   0   0 228   0  10]
 [  3   9   4   7   1  11   4   0 168   5]
 [  0   2   2   3  10   1   0   6   2 175]]
```

CM poly :

```
[[195   1   0   0   0   1   1   1   1   0]
 [  0 232   1   1   1   0   0   0   0   2]
 [  2   1 195   3   0   0   1   0   0   0]
 [  2   1   4 200   1   3   0   1   3   0]
 [  0   1   1   0 186   1   1   0   0   8]
 [  2   5   1   2   0 169   1   0   0   0]
 [  0   0   0   0   0   1 205   0   2   0]
 [  0   4   2   1   0   0   0 234   0   6]
 [  1   3   1   4   1   2   2   0 194   4]
 [  0   3   1   2   4   1   0   8   1 181]]
```

CM rbf :

```
[[  8 192   0   0   0   0   0   0   0   0]
 [  0 237   0   0   0   0   0   0   0   0]
 [  0 186  16   0   0   0   0   0   0   0]
 [  0 201   0  14   0   0   0   0   0   0]
 [  0 182   0   0  16   0   0   0   0   0]
 [  0 168   0   0   0  12   0   0   0   0]
 [  0 194   0   0   0   0  14   0   0   0]
 [  0 236   0   0   0   0   0  11   0   0]
 [  0 199   0   0   0   0   0   0  13   0]
 [  0 186   0   0   0   0   0   0   0  15]]
```

CM sig :

```
[[  0 200   0   0   0   0   0   0   0   0]
 [  0 237   0   0   0   0   0   0   0   0]
 [  0 202   0   0   0   0   0   0   0   0]
 [  0 215   0   0   0   0   0   0   0   0]
 [  0 198   0   0   0   0   0   0   0   0]
 [  0 180   0   0   0   0   0   0   0   0]
 [  0 208   0   0   0   0   0   0   0   0]
 [  0 247   0   0   0   0   0   0   0   0]
 [  0 212   0   0   0   0   0   0   0   0]
 [  0 201   0   0   0   0   0   0   0   0]]
```

A votre avis, quels sont les avantages et les inconvénients du SVM ?

Avantages :

- Il existe plusieurs fonctions de noyaux qui peuvent améliorer les résultats. Cela dépend du nombre des classes dans le jeu de données et de sa taille.
- Le noyau linéaire donne un résultat rapide avec une bonne performance.
- Nécessite moins de jeu de données que le réseau de neurones

Inconvénients :

- Les noyaux rbf et sigmoid ont de faibles performances (du moins dans notre cas)
- Ces deux noyaux nécessitent beaucoup de temps de calcul
- Nécessite plus de jeu de données que kNN

A votre avis, quel est le meilleur classifieur et quels sont ses hyperparamètres finaux ? Selon les cas, un algorithme peut être meilleur qu'un autre.

En effet, les réseaux de neurones sont plus performants que d'autres méthodes si, il existe vraiment beaucoup de données et leur structure est complexe. En particulier, si vous souhaitez résoudre un problème de classification, vous avez besoin de nombreux exemples par classe.

En revanche, le classifieur kNN n'a pas besoin de beaucoup d'exemples par classe. Donc, si vous avez un problème de classification avec quelques exemples d'entraînement par classe, kNN aurait probablement une meilleure performance que d'autres méthodes.

Pour obtenir une bonne performance, SVM nécessite beaucoup plus de données d'entraînement que kNN, mais moins que le réseaux de neurones. SVM linéaire peut être entraîné plus rapidement, mais il est moins précis que les approches non linéaires. Son temps d'entraînement est également plus long que celui du cas linéaire.

Cela dit, les algorithmes d'entraînement pour SVM ont de meilleures garanties. Les réseaux de neurones sont plus difficiles et nécessitent parfois un peu d'entretien.