CentraleSupélec

MS SIO

Introduction au Machine Learning

Compte rendu de Travaux Pratiques

Jérémy CAO

William AFONSO

Code source accessible à :
https://github.com/will-afs/IML

Plan de présentation

01 Analyse descriptive de notre ensemble de données

02 Entraînement d'un modèle via la méthode KNN

03 Entraînement d'un arbre de décisions

04 Conclusion

CentraleSupélec

1. Analyse descriptive de notre ensemble de données

1. Analyse descriptive de notre ensemble de données

1. Présentation des données brutes

Accéléromètre (x, y, z)

1 - Working at Computer
2 - Standing up
3 - Standing
4 - Walking
5 - Going up/down stairs
6 - Walking + talking with someone
7 - Talking while standing

Activités

Mesure n°	Accélération x	Accélération y	Accélération z	N° activité
1	1502	2215	2153	1
...	-	-	-	-
53440	1878	2277	1988	3
...	-	-	-	
162501	1550	1600	1455	7

15 individus

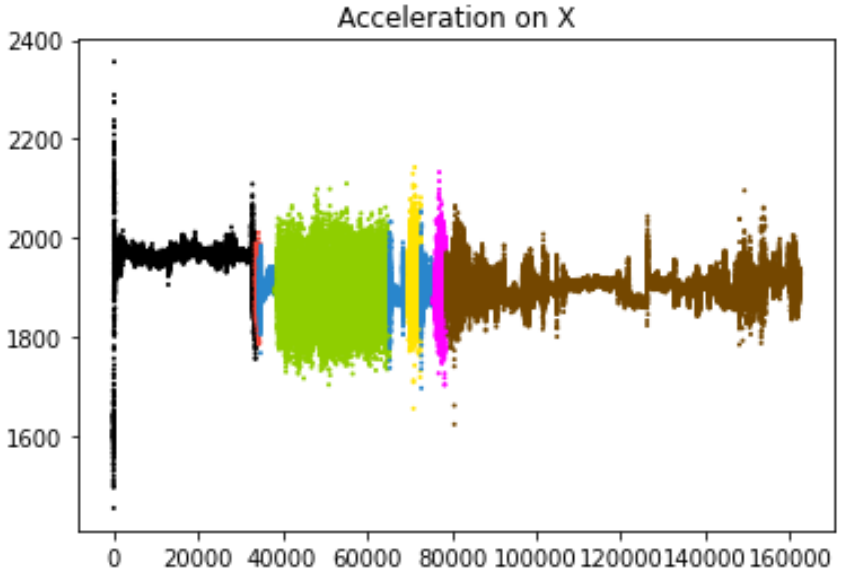
Jeu de données brut

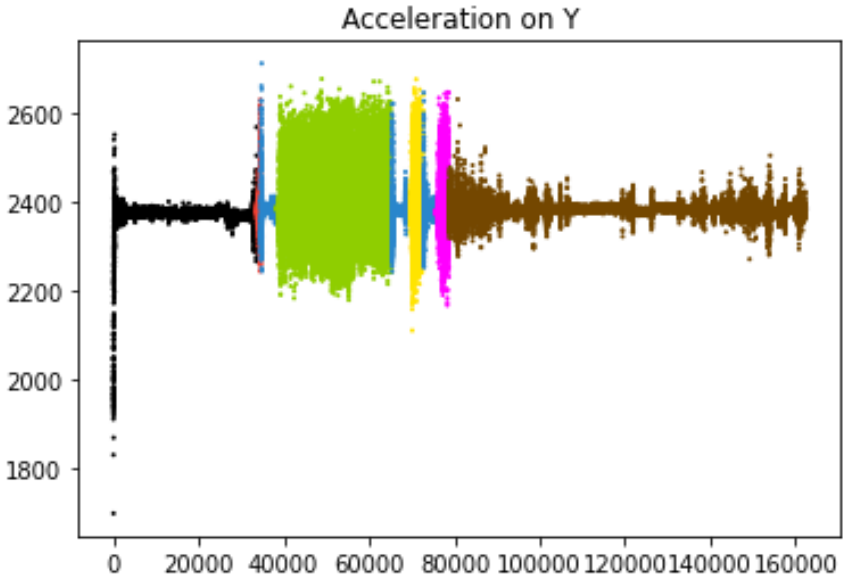
Objectif :

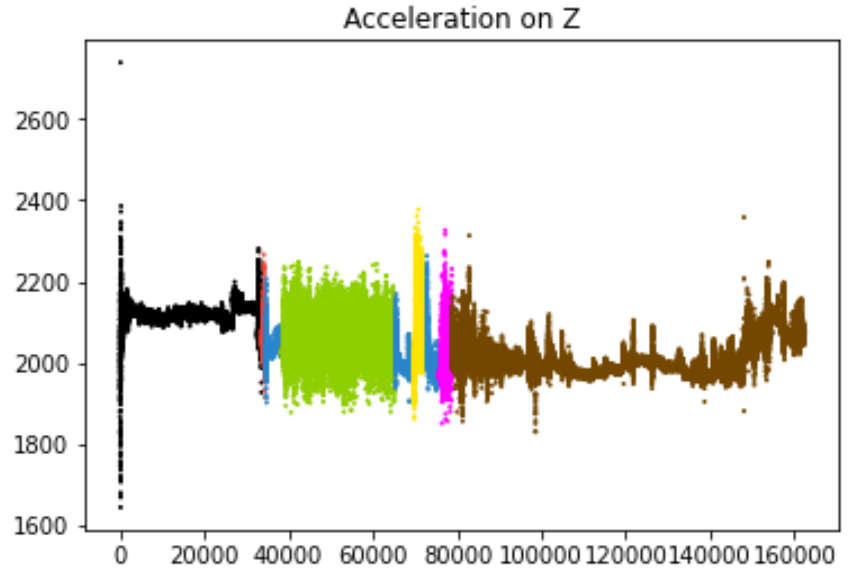
Raffiner nos données pour les rendre exploitables

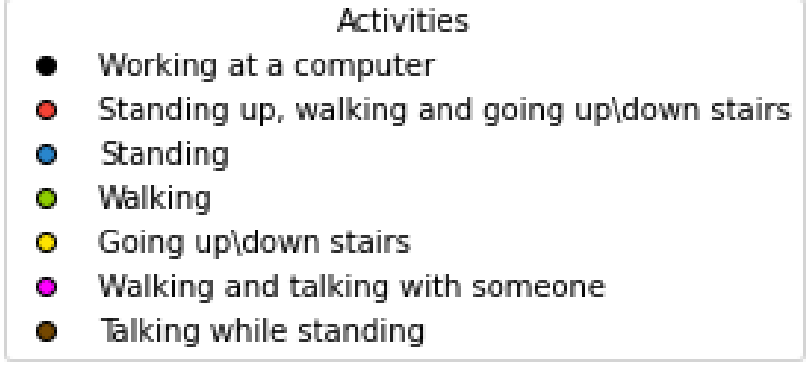
Trouver l'algorithme le plus approprié pour déduire l'activité à partir d'accélérations (x, y, z)

1. Analyse descriptive de notre ensemble de données
2. Analyse des données brutes et commentaires

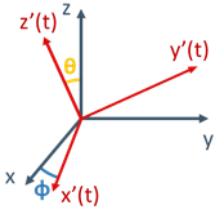








3,7 g au repos?!



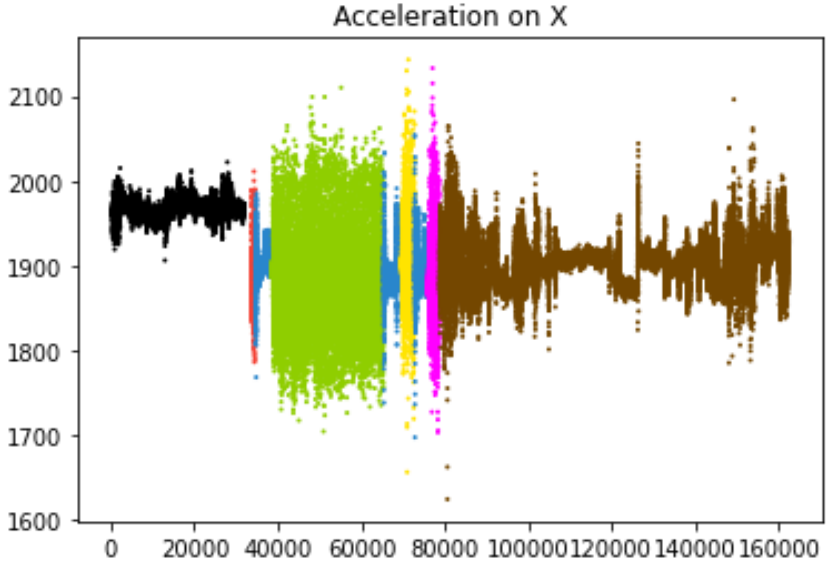
Système désaxé

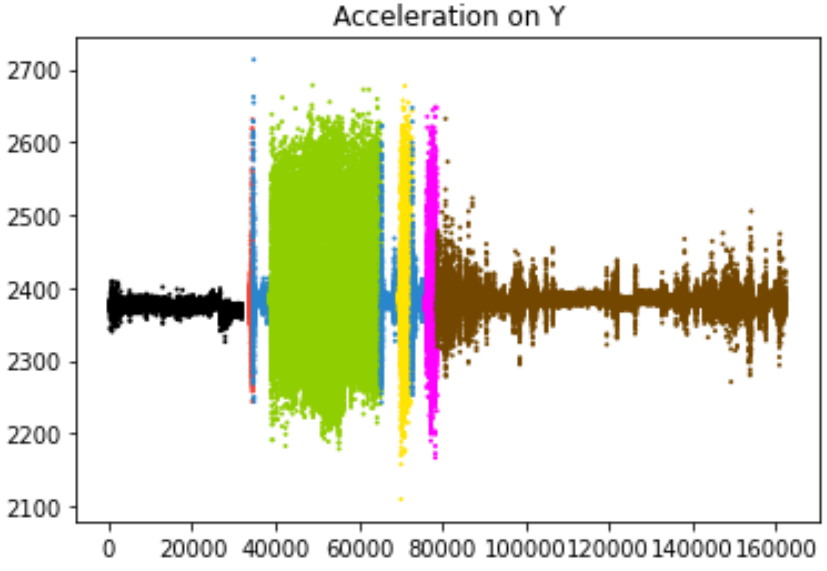
Régimes transitoires

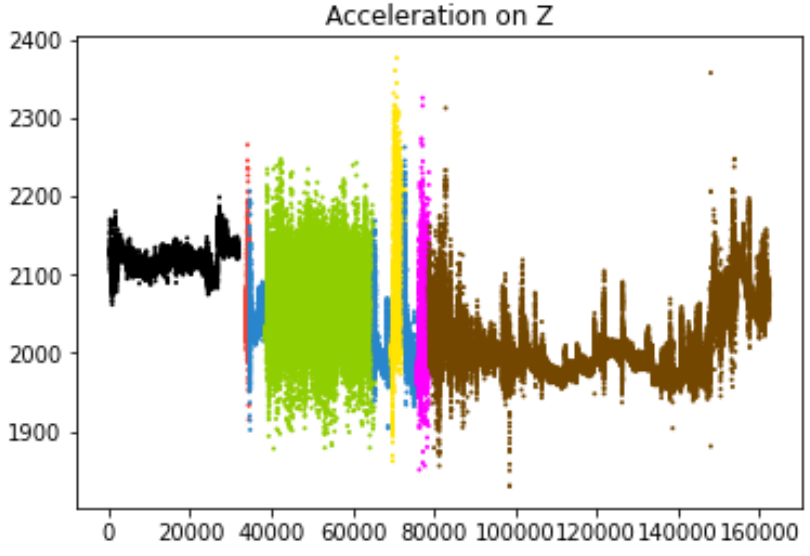
Mauvaise répartition

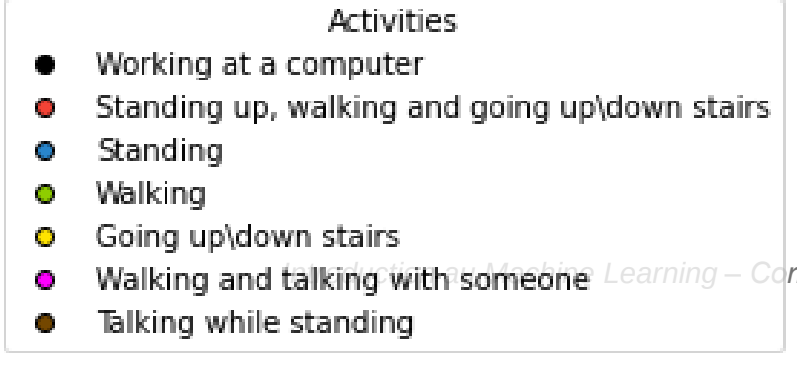
Mouvements chaotiques ?
Périodiques ?

Suppression des transitoires sur l'activité 1









1. Analyse descriptive de notre ensemble de données
4. Augmentation de l'échantillon de données

Mesure n°	Accélération x	Accélération y	Accélération z	N° activité
1	1502	2215	2153	1
…	-	-	-	-
53440	1878	2277	1988	3
…	-	-	-	
162501	1550	1600	1455	7

Individu 1

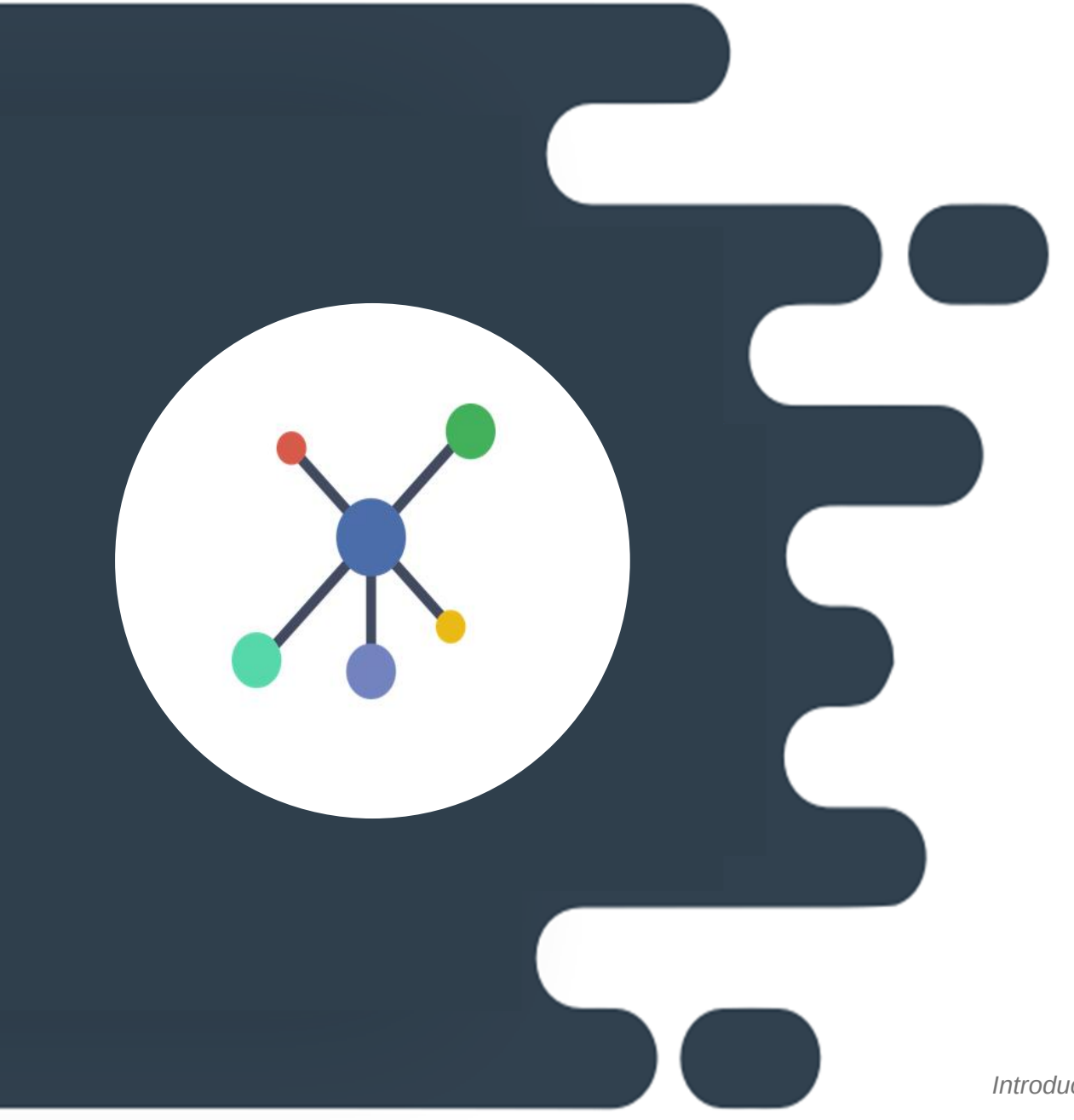
…

Mesure n°	Accélération x	Accélération y	Accélération z	N° activité
1	1897	2198	2110	1
…	-	-	-	-
53440	1877	2455	2322	3
…	-	-	-	
145099	1887	2110	2232	7

Individu 15

Datasets concaténés

CentraleSupélec



2. Prédictions avec la méthode KNN

2. Prédictions avec la méthode KNN

1. Comparaison de la précision du modèle avant et après data-processing

Kopt = 9

Suppression des transitoires sur l'activité 1

Avant : 85%

Activités réelles

```
[[ 9382    2    5  217   25    2  386]
 [   23    2    7  133    5    0  117]
 [   11    5  688  267   15    1 2362]
 [  320    3   84 6468  131   13  935]
 [   62    1    4  517  212    0  148]
 [   29    1    9  571   18    8  239]
 [ 1034   13  616  885   12    7 22755]]
```

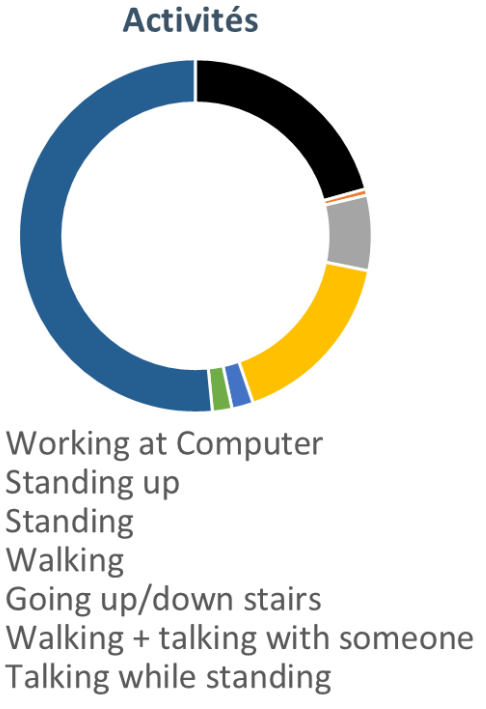
Après : 87%

Activités réelles

```
[[ 9737    3    4  134   14    1  125]
 [   11   23    6  117    6    0   90]
 [   10    3 1110  260   20    8 1964]
 [  184   19  104 6778  148   23  776]
 [   26    2   10  430  308    4  166]
 [   10    2   52  550   19   27  180]
 [  174   28  663  716   34   17 23534]]
```

1 2 3 4 5 6 7

Activités prédites par le modèle



CentraleSupélec

2. Prédictions avec la méthode KNN

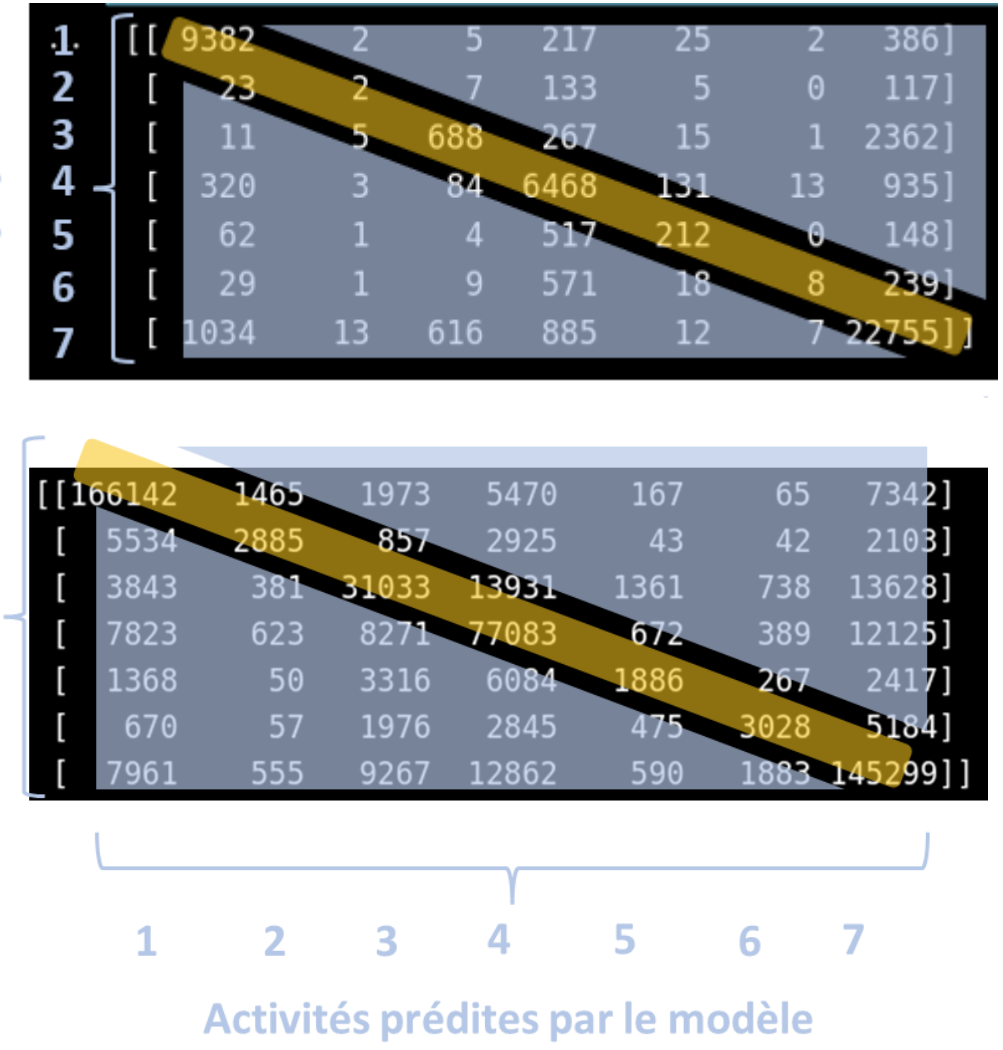
1. Comparaison de la précision du modèle avant et après data-augmentation

Kopt = 9

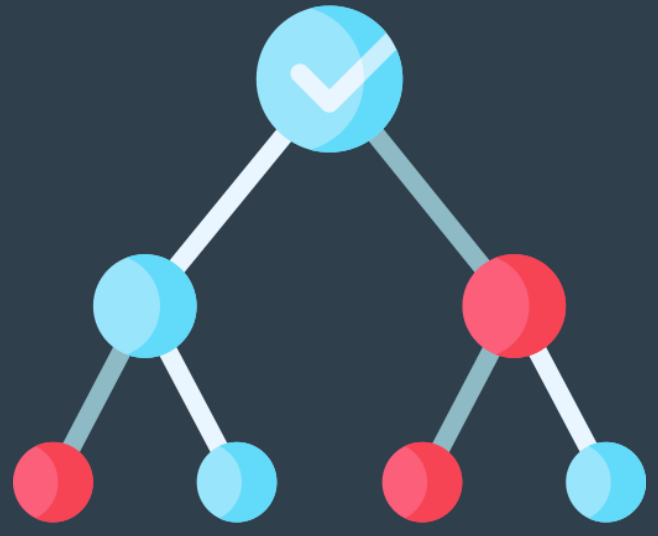
Concaténation des données de chaque individu

Avant : 85 %

Après : 77 %



CentraleSupélec



3. Prédictions avec la méthode d’arbre de décisions

3. Construction d'un arbre de décisions

1. Comparaison de la précision du modèle avant et après data-processing

Kleaf = +- 209

Suppression des transitoires sur l'activité 1

Avant : 85-%

Après : 85+%

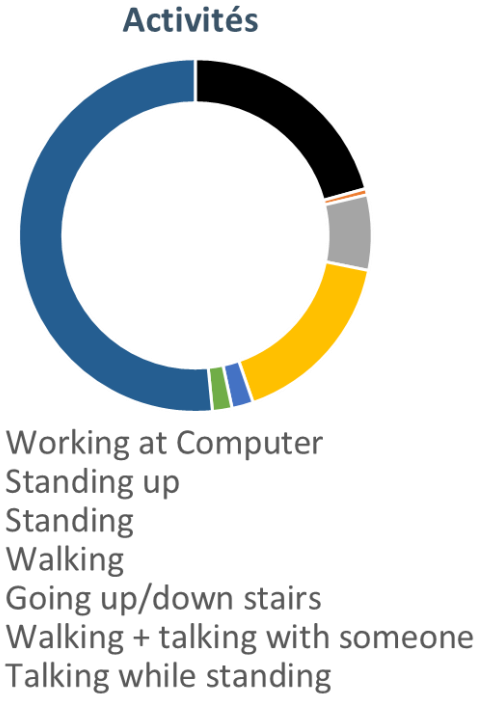
```
*** CONFUSION MATRIX ***
[[ 9738     0     1   166    32     0   203]
 [   12     0     7   119     3     0   120]
 [    7     0   823   238    14     3  2325]
 [  186     0    83  6673   124     5   841]
 [   24     0     0   535   240     4   165]
 [    7     0    16   608    29    14   196]
 [  219     0   274   812    18     9 23857]]
```

```
*** SCORE ***
0.8549493018617021

*** CONFUSION MATRIX ***
[[ 9374     0     0    39     0     0    63]
 [    2     0     6   134    10     0   122]
 [    6     0   823   227    19     1  2224]
 [  166     0   104  6803   176    15   888]
 [   19     0     0   456   313     4   147]
 [    6     0    54   597    28    17   178]
 [  162     0   368   722    30     8 23817]]
```

Activités réelles : 1 2 3 4 5 6 7

Activités prédites par le modèle : 1 2 3 4 5 6 7



CentraleSupélec

3. Construction d'un arbre de décisions

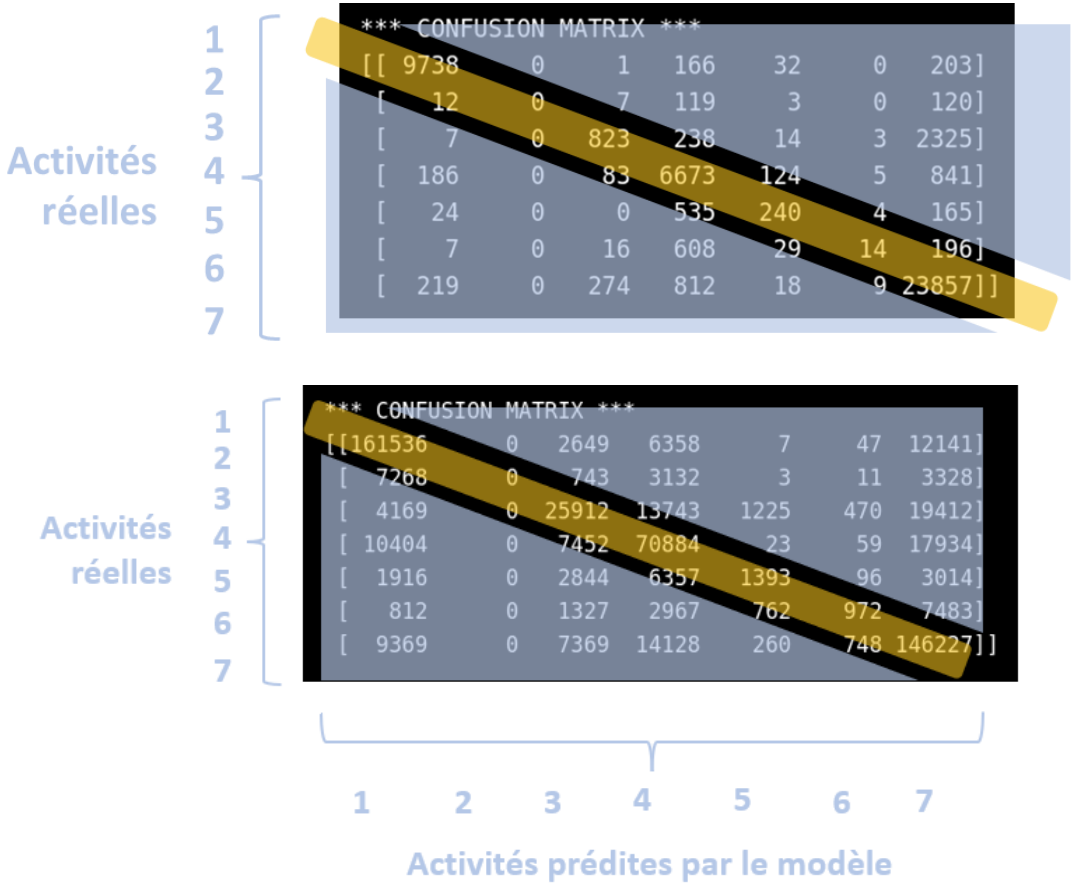
1. Comparaison de la précision du modèle avant et après data-augmentation

Kleaf = +- 209

Concaténation des données de chaque individu

Avant : 85-%

Après : 71%



CentraleSupélec

Conclusion

CentraleSupélec

MERCI !
Des questions?

CentraleSupélec

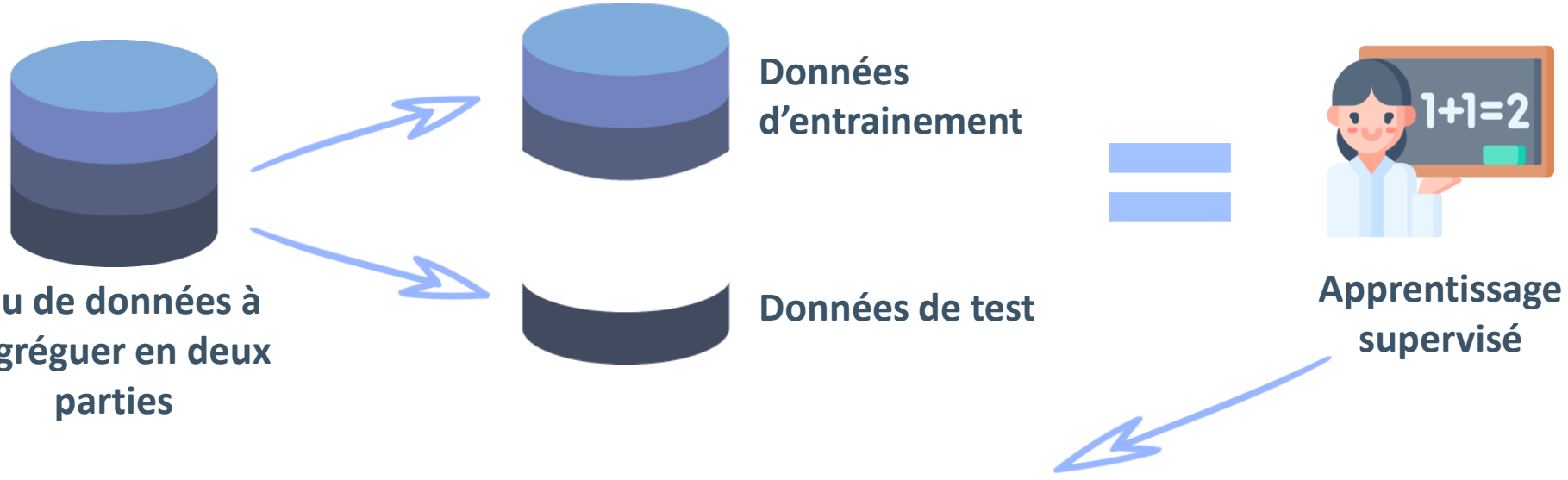
ANNEXES

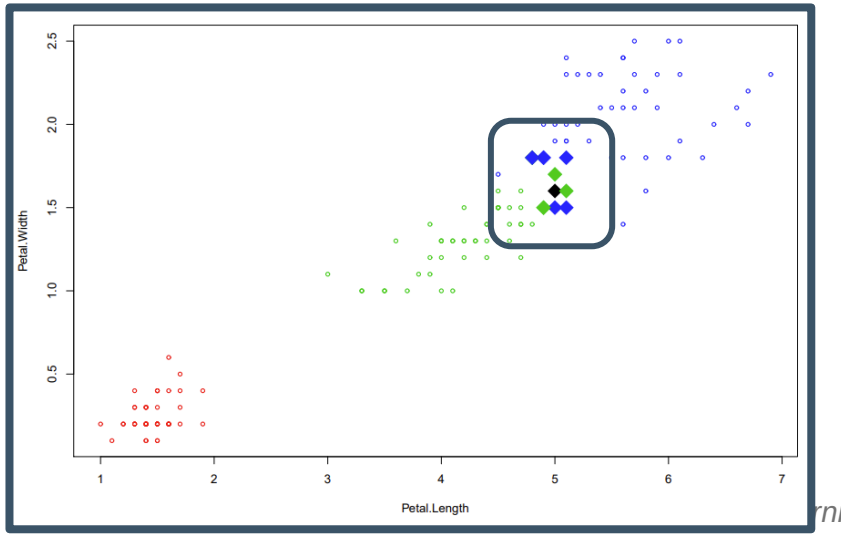
CentraleSupélec

Synthèse du cours

Synthèse du cours

1. Méthode KNN





On prédit la caractéristique du point à partir de ses K voisins les plus proches

Attention aux données et au paramétrage !

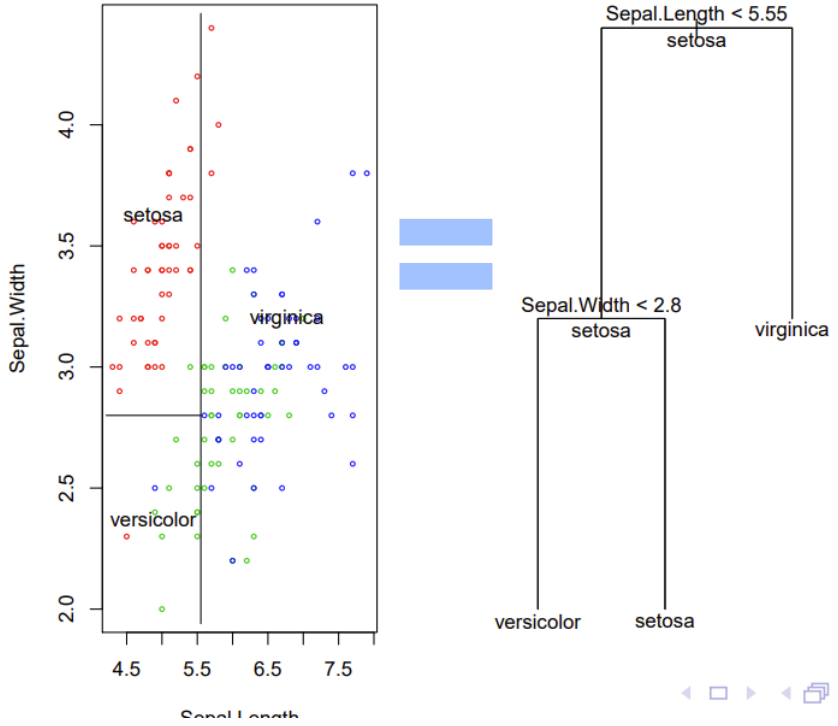
CentraleSupélec

Synthèse du cours

2. Arbres de décision

Apprentissage supervisé (comme Knn)

Segmentation de l'espace en sous-espaces où l'entropie est minimale

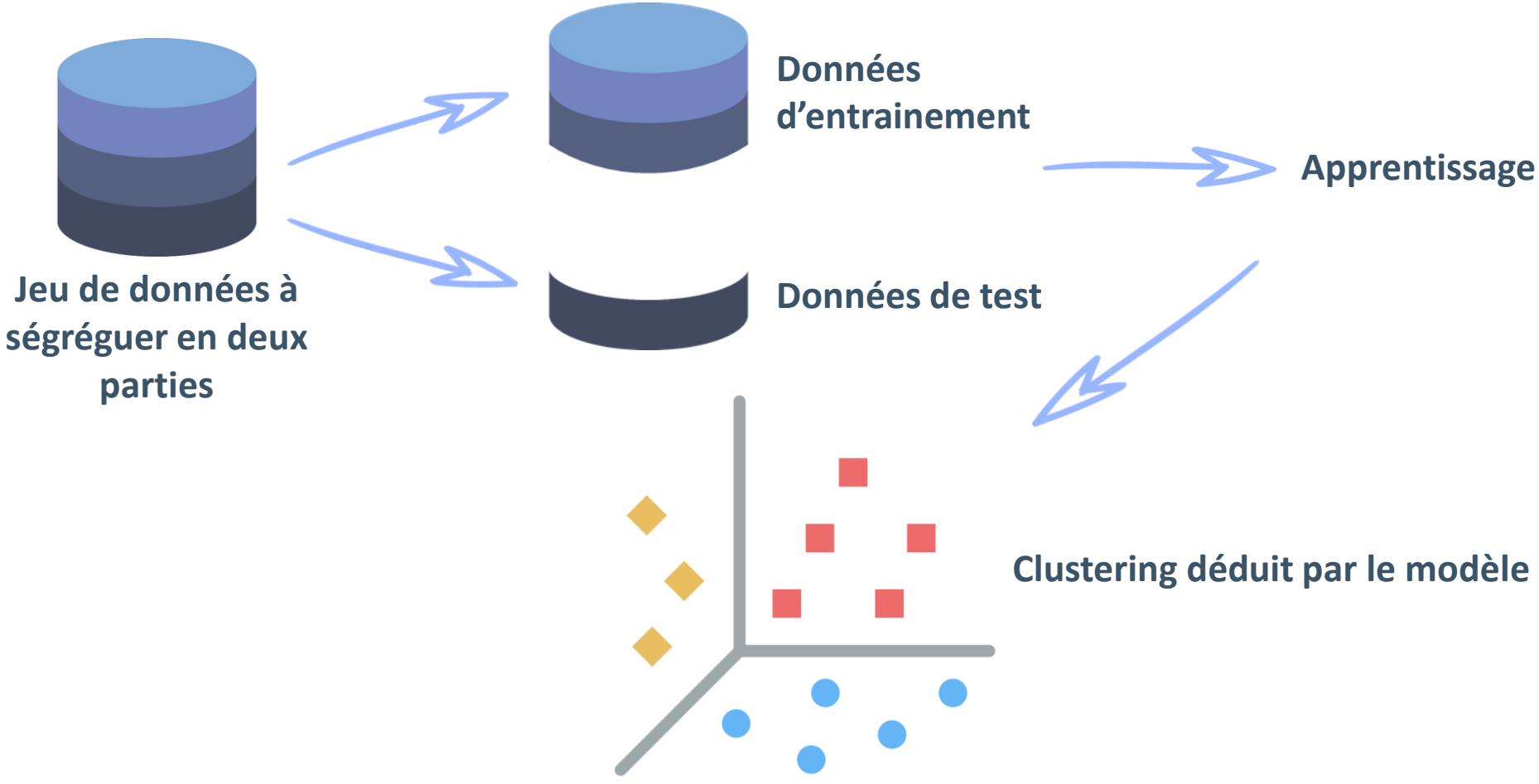


Puis, à partir des caractéristiques de l'échantillon étudié, l'algorithme pourra prédire sa classe, en s'orientant grâce à l'arbre

Il est possible d'élaguer des branches de l'arbre pour éliminer les moins pertinentes et ainsi améliorer la précision de l'arbre

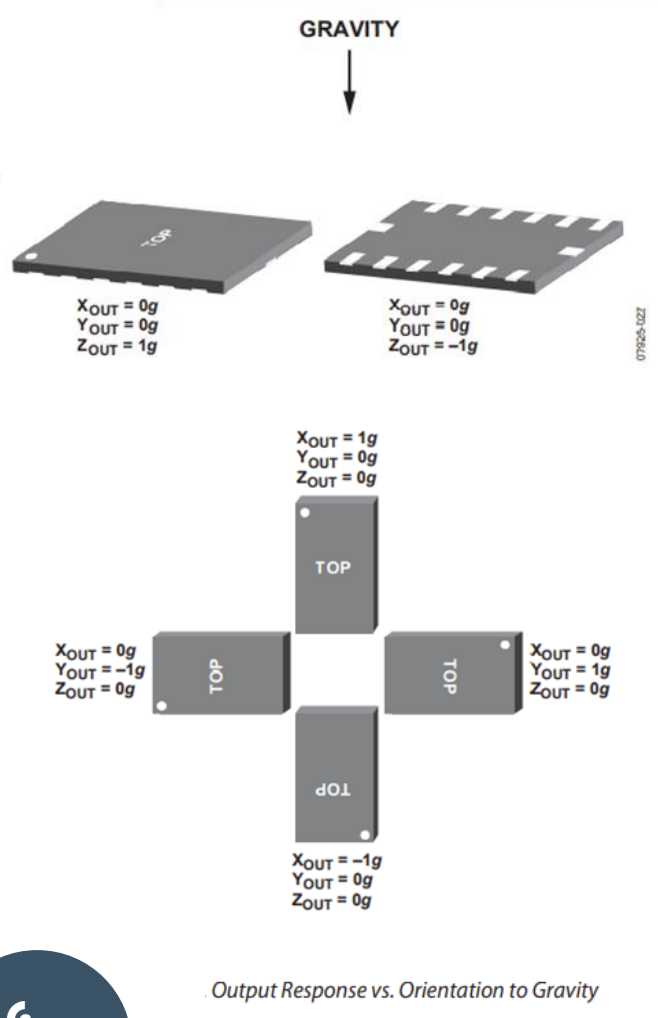
CentraleSupélec

3. Kmeans


Jeu de données à ségréguer en deux parties
Données d'entrainement
Données de test
Apprentissage non supervisé
Clustering déduit par le modèle

Analyse des données (sources)

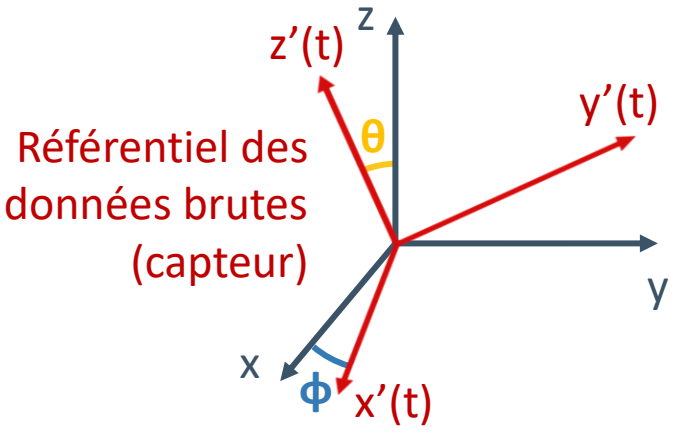
Analyse descriptive de notre ensemble de données
Calcul de l'angle Thêta pour corriger les valeurs



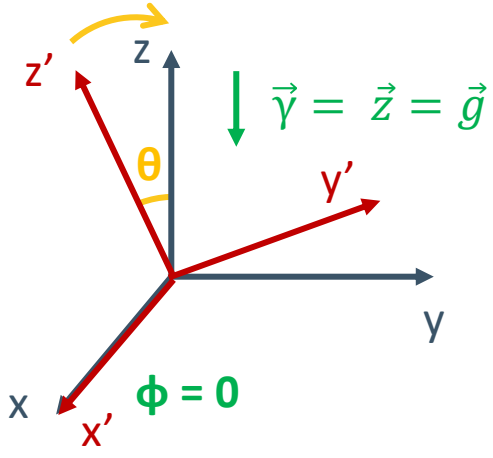
Output Response vs. Orientation to Gravity

Un problème de repère ?

Repère terrestre



Référentiel des données brutes (capteur)



Working at a computer

(1) $$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

(2) $$z = \sqrt{x'^2 + y'^2 + z'^2} = g$$

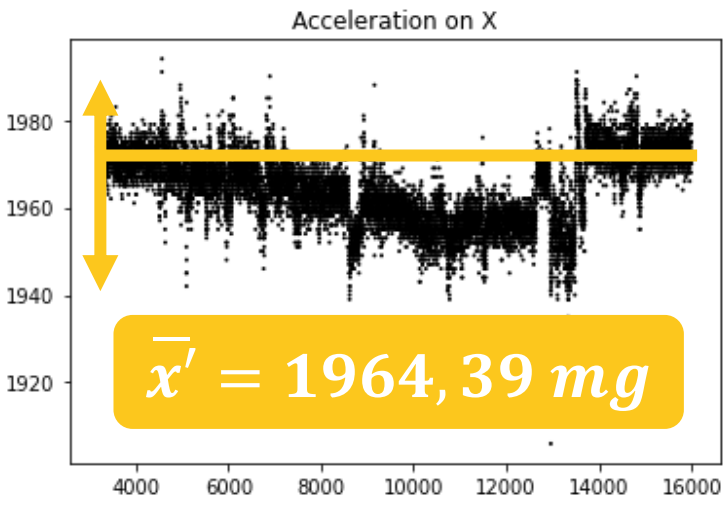
(1)+(2) ➔ Θ

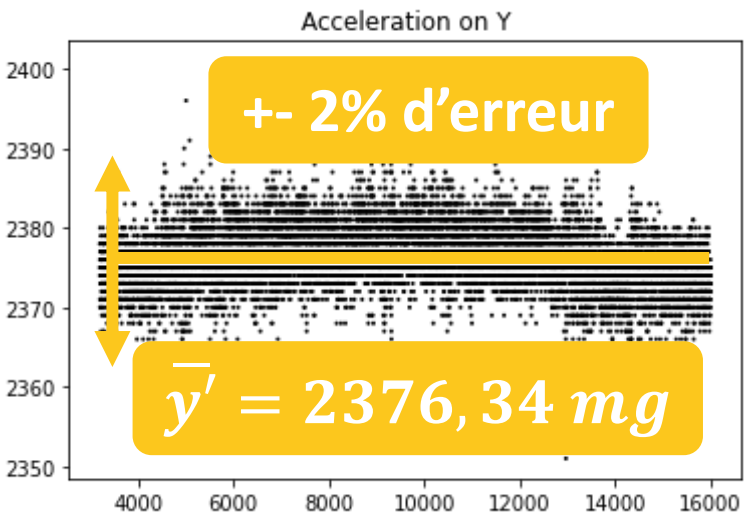
sympy

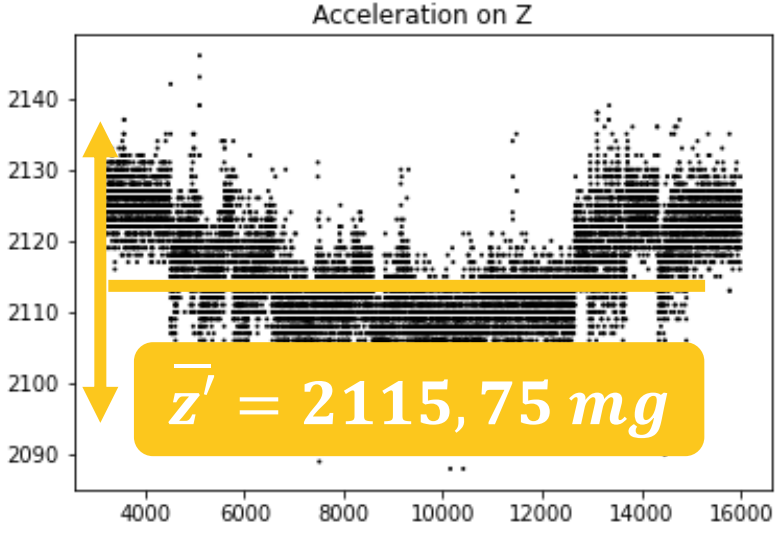
CentraleSupélec

Analyse descriptive de notre ensemble de données
Calcul de l'angle Thêta pour corriger les valeurs

Isolation des valeurs d'accélération sur l'activité 1 (stable)





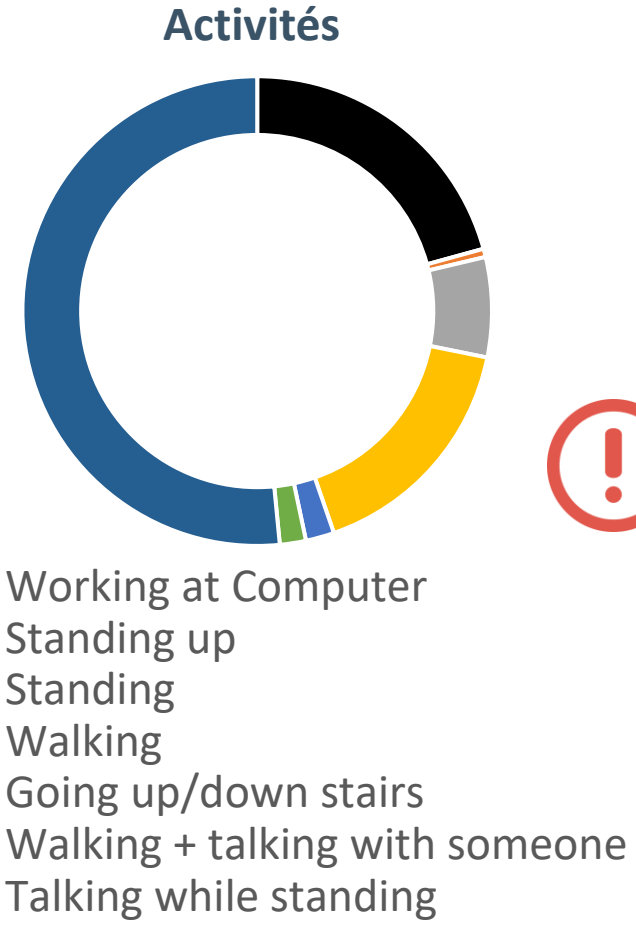


$$3739,28 = 2376,34\ sin(\overline{\theta}) + 2115,75\ cos(\overline{\theta})$$

CentraleSupélec

Analyse descriptive de notre ensemble de données

Analyse des données - Activités dynamiques très peu représentées



- **Les activités « statiques » sont représentées à + de 75%**
- **Certaines données sont très peu représentées**

CentraleSupélec

Analyse descriptive de notre ensemble de données
Analyse des données brutes

	Moyennes		
Activité	Accélération x	Accélération y	Accélération z
1	1965	2373	2119
2	1892	2376	2067
3	1893	2383	2011
4	1884	2381	2051
5	1918	2373	2103
6	1888	2381	2031
7	1900	2383	2008

	Ecart-types		
Activité	Accélération x	Accélération y	Accélération z
1	28	22	24
2	33	50	49
3	20	24	33
4	45	88	49
5	49	89	76
6	46	79	59
7	20	9	39

Pour toutes les activités, c'est l'accélération en y (resp. x) qui est la plus (resp. moins) élevée

Les activités « statiques » (1, 3, 7) ont des accélérations anormalement élevées

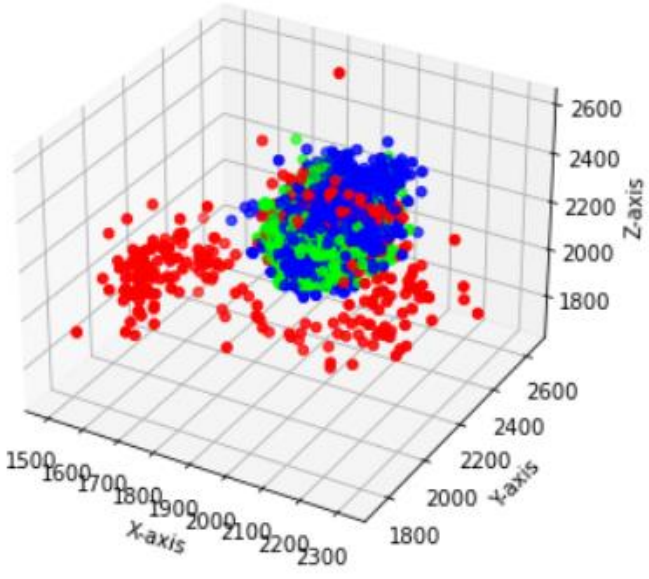
Les activités « dynamiques » (2, 4, 5, 6) – les moins représentées - ont les écarts-types plus élevés

CentraleSupélec

Analyse descriptive de notre ensemble de données
Analyse des données

	Accélération x	Accélération y	Accélération z
Moyenne	1910	2380	2041
Ecart-type	41	42	60
Min	1455	1697	1644
Max	1935	2386	2101

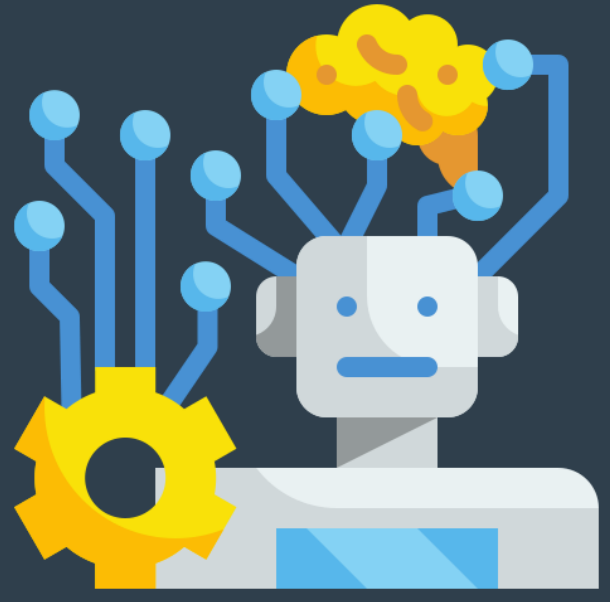
Analyse statistique sur l'ensemble des accélérations, toutes activités confondues



CentraleSupélec

Analyse descriptive de notre ensemble de données
Preprocessing

	Moyennes des accélérations		
Activité	En x	En y	En z
1	1965	2373	2119
2	1892	2376	2067
3	1893	2383	2011
4	1884	2381	2051
5	1918	2373	2103
6	1888	2381	2031
7	1900	2383	2008



Entraînement d’un modèle de ML avec KNN (snippets)

Entraînement d'un modèle de ML avec KNN (snippets)

1. Import des données via Pandas


```
X=pd.read_csv('data/1.csv',header=None,delimiter=',',usecols=[1, 2, 3])

y=pd.read_csv('data/1.csv',header=None,delimiter=',',usecols=[4])



X = X.values
y = y.values[:,0]
```

Entraînement d'un modèle de ML avec KNN (snippets)

2. Entraînement et test avec exactement les mêmes données


```
KNeighborsClassifier(n_neighbors=10)
```

```
# If we were in regresssion
# clf = neighbors.KNeighborsRegressor(n_neighbors=) # je précise la méthode et ses hyperparamètres
```

✓ 0.3s Python

```
# prevision
clf.predict(X)
```

✓ 7.7s Python

```
array([0.0000e+00, 1.0000e+00, 2.0000e+00, ..., 3.6985e+04, 1.5683e+05,
       1.5807e+05])
```

```
clf.score(X,Y) # predit + calcul le score = la précision
```

✓ 7.1s Python

```
0.10551384615384615
```

10% de précision ! C'est très faible ! Overfitting ?

Entraînement d'un modèle de ML avec KNN (snippets)

2. Méthode KFold pour compléter les données manquantes


```
# Pour voir l'effet du partage ...

from sklearn.model_selection import KFold
kf=KFold(n_splits=4, shuffle=True) # partages de validation
XXX = X_train[:12,:]  # je prends uniquement les 12 premières lignes
#print(XXX)
for learn,test in kf.split(XXX): # boucle sur différents partages de validation
    print("Learn")
    print(learn)
    print("test")
    print(test)
```

[18] ✓ 0.6s Python

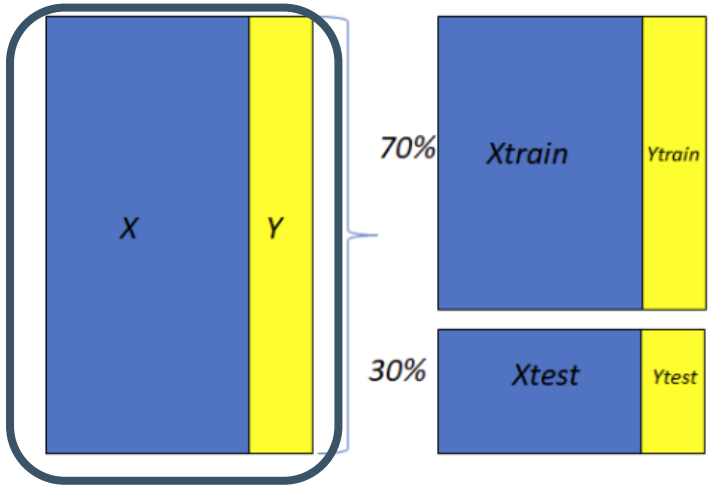
```
Learn
[ 0  2  3  4  5  6  7  9 10]
test
[ 1  8 11]
Learn
[ 1  3  4  5  6  7  8  9 11]
test
[ 0  2 10]
Learn
[ 0  1  2  4  5  7  8 10 11]
```

CentraleSupélec

Entraînement d'un modèle de ML avec KNN (snippets)

3. Préparation des échantillons d'entraînement et de test

Jeu de données initial



Jeu de données ségrégué

```
import random
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,y,test_size=0.3,random_state=random.seed())
print(X_train.shape)
print(X_test.shape)
```

[13] ✓ 0.7s Python

```
(113750, 2)
(48750, 2)
```

On construit cet échantillon sur la base d'un nombre pseudo-aléatoire

70% / 30% classique

Entraînement d'un modèle de ML avec KNN (snippets)

3. Préparation des échantillons d'entraînement et de test

On entraîne notre modèle avec les données splittées en 70/30

```
n_neighbors=9
clf = neighbors.KNeighborsClassifier(n_neighbors) # je précise la méthode et ses hyperparamètres
clf.fit(X_train, Y_train) # je lance l'apprentissage
prev_test = clf.predict(X_test)
sc_train = clf.score(X_train,Y_train)
sc_test = clf.score(X_test,Y_test)
print(sc_train)
print(sc_test)
```

[15] ✓ 5.1s Python

```
0.8255648351648351
0.8105641025641026
```

Le score est intéressant, mais pas non plus imbattable…

Entraînement d'un modèle de ML avec KNN (snippets)

4. Evaluation du modèle avec la matrice de confusion


```
# Compute the confusion matrix between kmeans activity label and accelerations
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(Y_test,prev_test)
# the confusion matrix is difficult to read as labels assigned by kmeans are arbitrary
print(cm)
```

[17] ✓ 0.1s

```
[[9382    2    5  217   25    2  386]
 [  23    2    7  133    5    0  117]
 [  11    5  688  267   15    1 2362]
 [ 320    3   84 6468  131   13  935]
 [  62    1    4  517  212    0  148]
 [  29    1    9  571   18    8  239]
 [1034   13  616  885   12    7 22755]]
```

Activités réelles

1 2 3 4 5 6 7

Activités prédites par le modèle

Activités correctes
(réel correspond à ce qui a été prédit)

Activités incorrectes
(réel correspond à autre chose que ce qui a été prédit)

CentraleSupélec

Entraînement d'un modèle de ML avec KNN (snippets)

1. Recherche de l'hyperparamètre K optimal


```python
from sklearn.model_selection import KFold
kf=KFold(n_splits=3, shuffle=True) # partages de validation


from sklearn import neighbors
scores=[]
for k in range(1,10):  # les différentes valeurs de k à tester
    score=0
    clf=neighbors.KNeighborsClassifier(k)
    for learn,test in kf.split(X_train): # boucle sur différents partages de validation
        X_app=X_train[learn]
        Y_app=Y_train[learn]
        clf.fit(X_app,Y_app)
        X_val=X_train[test]
        Y_val=Y_train[test]
        score+=clf.score(X_val,Y_val)
    scores.append(score)
print(scores)
#plt(scores)
k_opt=scores.index(max(scores)) + 1  # valeur optimale de k
print(k_opt)
```

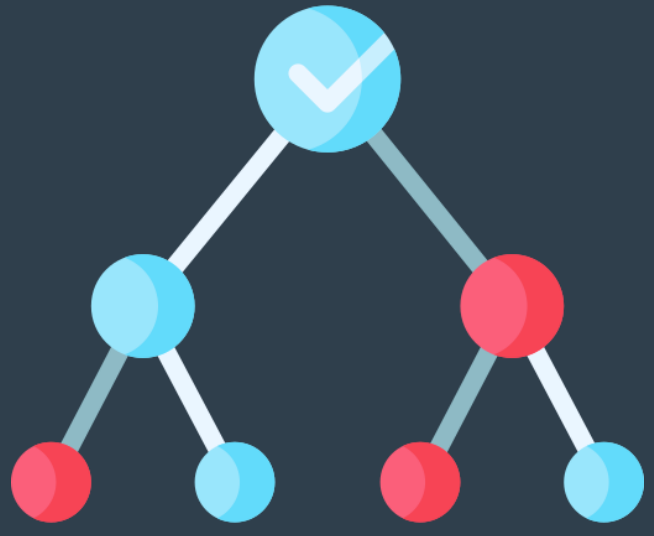
Recherche du meilleur hyperparamètre K

Python

```
[2.430260421697527, 2.4124207978205026, 2.5024149145857715, 2.5292163036099855, 2.5417375292364217,
2.55765088301482, 2.561503642421665, 2.5670314489655865, 2.567910907329197]
9
```

Index du score maximal

Score maximal



Entraînement d’un modèle d’arbre de décisions

Construction d'un arbre de décisions classificateur (snippets)

1. Construction simple

Même chose qu'avec KNN

```
import random
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

X=pd.read_csv('data/3.csv',header=None,delimiter=',',usecols=[1, 2, 3]).values
Y=pd.read_csv('data/3.csv',header=None,delimiter=',',usecols=[4]).values[:,0]

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=random.seed())
clf=tree.DecisionTreeClassifier(max_leaf_nodes=199,criterion='entropy')
clf.fit(X_train,Y_train)
prev_test = clf.predict(X_test)
#print(prev)
scoretree = clf.score(X_test,Y_test)
# Compute the confusion matrix between kmean label and iris types

cm=confusion_matrix(Y_test,prev_test)
# the confusion matrix is difficult to read as labels assigned by kmeans are arbitrary
print(cm)
print(scoretree)
```

On peut construire notre DTC selon différents critères

On configure le nombre de feuilles maximal (ici au doigt mouillé)

Même chose qu'avec KNN

✓ 1.6s Python

Construction d'un arbre de décisions (snippets)

2. Recherche de l'hyperparamètre K (nombre de feuilles) optimal


```
from sklearn import neighbors
scores=[]
for k in range(190,210):  # les différentes valeurs de k à tester
    score=0
    clf=tree.DecisionTreeClassifier(max_leaf_nodes=k,criterion='entropy')
    for learn,test in kf.split(X_train): # boucle sur différents partages de validation
        X_app=X_train[learn]
        Y_app=Y_train[learn]
        clf.fit(X_app,Y_app)
        X_val=X_train[test]
        Y_val=Y_train[test]
        score+=clf.score(X_val,Y_val)
    scores.append(score)
print(scores)
#plt(scores)
k_opt=scores.index(max(scores)) + 1  # valeur optimale de k
print(k_opt)
```

✓ 6.2s Python

```
[2.5648120535533594, 2.563806944251248, 2.560624199841145, 2.5633881329256107, 2.56853901822021, 2.565398323471006,
2.56569139792798 2, 2.5633461639338417, 2.563555691477194, 2.5664033556113415, 2.5655238534058125, 2.56732454095933,
2.5637231097346387, 2.566193915751826, 2.5682039449589613, 2.5657332985263572, 2.5659846898408736,
2.569292811615906, 2.5637232324920105, 2.56481198340629]
18
```

Le nombre de feuilles optimal vaut donc 190+18 = 208