# Introduction

L’objectif de cette étude est de prédire à partir d’un ensemble de caractéristiques dans un jeu de données, des chiffres manuscrits de «0» à «9».

Le jeu de données décrit les caractéristiques de chiffres manuscrits («0» - «9») extraits d'une collection de cartes utilitaires. 200 motifs par classe (pour un total de 2 000 motifs) ont été numérisés en images binaires. Ces chiffres sont représentés par les six groupes de variables comme décrit ci-dessous :   
  
1. mfeat-fou: 76 coefficients de Fourier des formes des caractères;  
2. mfeat-fac: 216 corrélations de profils;  
3. mfeat-kar: 64 coefficients Karhunen-Love;  
4. mfeat-pix: 240 pixels en moyenne dans 2 x 3 fenêtres;  
5. mfeat-zer: 47 moments Zernike;  
6. mfeat-mor: 6 caractéristiques morphologiques.  
  
Dans chaque fichier, les 2000 motifs sont stockés en ASCI sur 2000 lignes. Les 200 premiers modèles sont de classe «0», suivis par des ensembles de 200 modèles pour chacune des classes «1» - «9». Les modèles correspondants dans différents jeux de fonctions (fichiers) correspondent au même caractère d'origine.

Afin de déterminer la classe d’appartenance de chaque chiffre manuscrit, nous allons chercher une fonction discriminante qui permettra de prédire si un chiffre manuscrit est un 0, un 1… ou un 9, ceci tout en minimisant le taux d’erreur.

S’agissant d’une analyse supervisée, plusieurs méthodes peuvent être utilisées afin de prédire la classe d’appartenance à partir des caractéristiques : Régression logistique multinomiale, analyse discriminante, arbre de décision, Gradient Boosting, SVM, réseaux de neurones, classification naïve bayésienne…

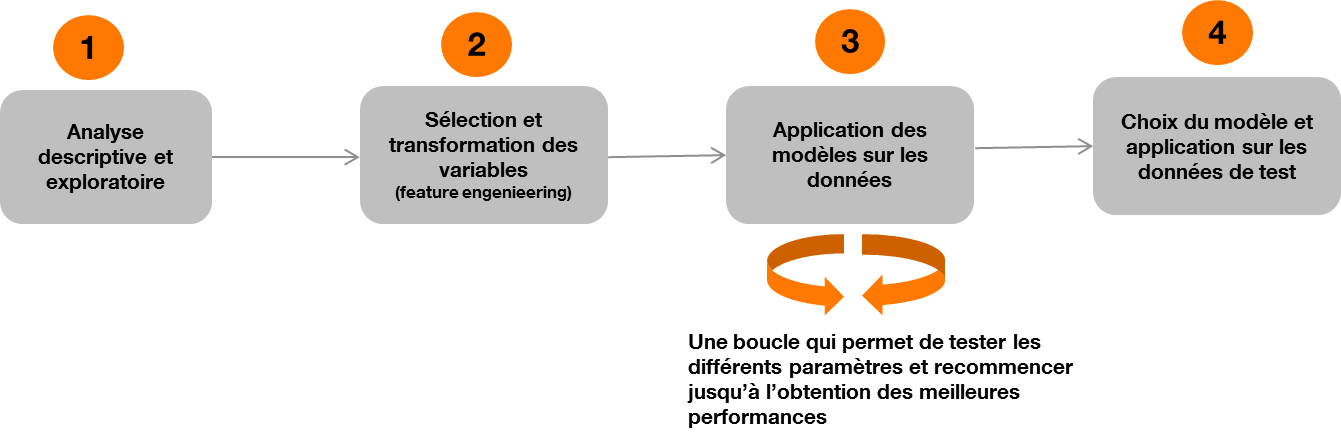
Nous allons en particulier tester deux méthodes vues en cours qui sont le SVM et les réseaux de neurones.

**Méthodologie de traitement**

Afin de procéder à cette analyse, nous allons tout d’abord nous intéresser à une étude exploratoire /descriptive des données, ce qui va nous permettre de mieux comprendre leurs structures.

Etant donné que le jeu de données est composé d’un groupe de variables décrivant des classes d’individus, nous allons :

* Analyser la signification de chaque groupe variable pour avoir une compréhension métier fine des variables traitées.
* Faire une analyse uni-variée et bi-variée des différentes variables afin de détecter les valeurs aberrantes, celles avec peu de variance, les variables très corrélées…
* Faire une classification de variables pour détecter celles qui portent la même information.
* Faire une AFM sur les groupes de variables pour détecter par groupe, les composantes qui apportent le maximum d’inertie par groupe (cercle de corrélation…),
* Traiter chaque groupe de classes séparément, ce qui permettra d’identifier pour chacune d’elle, les variables discriminantes,
* Faire une régression régularisée (LASSO, RIDGE, NET) pour classer les variables par ordre d’importance (p-value).
* Faire un partitionnement K-means pour vérifier si on peut reconstituer les différentes classes à partir des variables discriminantes identifiées ,
* Transformation des données à partir des résultats des analyses exploratoires (suppression ou création des nouvelles variables, normalisation, discrétisation….)
* Diviser les données en 2 parties : Apprentissage et test.
* Appliquer les modèles SVM et CNN avec une validation croisée,
* Benchmark des deux modèles et synthèse.



## Analyse exploratoire des données

Analyse uni-variée

Notre jeu de donnée d’apprentissage est composé de 1500 individus et 650 variables.

La variable d’intérêt est la variable ‘class’ qui prend 10 valeurs différentes.

Une description globale nous donne un aperçu général des données, la moyenne, leurs variances et les déciles à 25%,50% et à 75%.



Suite à cette analyse, nous pouvons constater :

* Absence des valeurs manquantes dans le jeu de données
* Absence des variables avec une variance à 0

Nous avons ensuite fait une étude des différentes variables en étudiant leurs moyennes et leurs écarts type par classe :



Vu le nombre élevés des variables, une représentation graphiques des distributions s’est avérée difficile. Néanmoins, nous allons faire une description analytique des différentes caractéristiques de chaque variable :



Nous allons nous baser sur les valeurs des variances, asymétrie et principalement du kurtosis pour détecter les variables qui peuvent contenir des valeurs extrêmes.

Notre objectif par la suite et de réduire le nombre de dimension par groupe de variable, mais avant de procéder à cette transformation, il est primordial de traiter les valeurs extrêmes, car la méthode de réduction basée sur l’analyse de composante principale et la méthode d’analyse SVM y sont très sensibles.

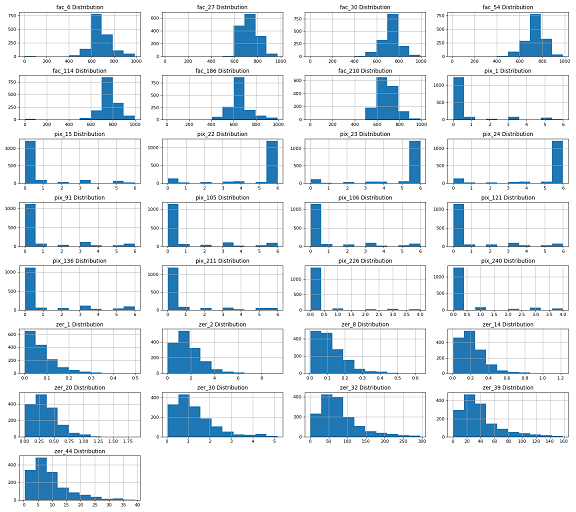
Avant de commencer les transformations des variables, nous allons garder 20% de notre échantillon pour tester les performances de notre modèle.

### Traitement des valeurs extrêmes

Comme indiqué dans le paragraphe précédent, un kurtosis grand indique la présence probable des valeurs extrêmes.

Plusieurs méthodes de détection sont possibles tel que : IQR ou z-score. Cette dernière est à écarter car nos données ne suivent pas une distribution normale (cf test shapiro).

Nous allons par la suite visualiser les variables avec un |Kurtosis|>2 qui sont :

fac\_6,fac\_27,fac\_30,fac\_54,fac\_114,fac\_186,fac\_210,pix\_1,pix\_15,pix\_22,pix\_23,pix\_24,pix\_91,pix\_105,pix\_106,pix\_121,pix\_136,pix\_211,pix\_226,pix\_240,zer\_1,zer\_2,zer\_8,zer\_14,zer\_20,zer\_30,zer\_32,zer\_39,zer\_44,

A travers les histogrammes, nous constatons :

* La présence des valeurs extrêmes pour les variables : 'fac\_6','fac\_27','fac\_30', 'fac\_54', 'fac\_114','fac\_186','fac\_210',’zer\_20’
* Une distribution particulière pour les variables 'pix\_1', 'pix\_15','pix\_22', 'pix\_23', 'pix\_24','pix\_91','pix\_105','pix\_106', 'pix\_121','pix\_136','pix\_211','pix\_226','pix\_240'.
* La distribution très asymétrique des variables 'zer\_1','zer\_2','zer\_8','zer\_14', 'zer\_30', 'zer\_32','zer\_39','zer\_44'.

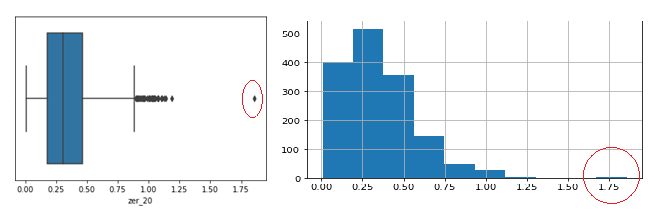
Suite à cette analyse, nous allons :

* Traiter les valeurs extrêmes constatées pour les variables fac\_\* et ’zer\_20’,
* Laisser les autres valeurs telles qu’elles sont, car et vu leurs distributions particulières, ces valeurs ne sont pas considérés comme extrêmes.

## Traitement des valeurs extrêmes

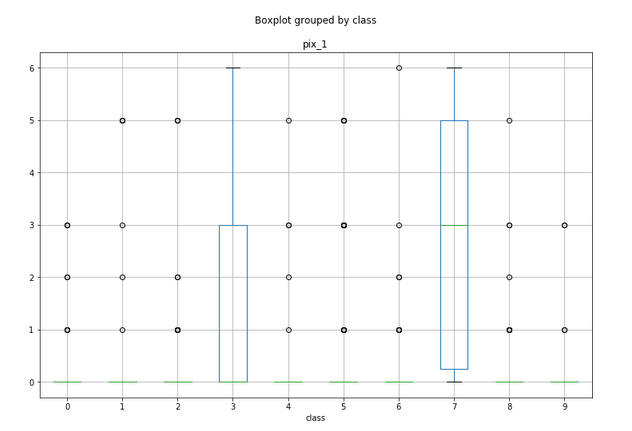
En couplant la visualisation des histogrammes avec les box-plot et les tests analytiques des IQR (écart interquartile), nous avons constaté que la valeur 3 de l’IQR représente un bon seuil de coupure pour les valeurs aberrantes.

En effet, nous prenons en pratique la valeur 1,5 pour les valeurs qu’on appelle « soft outliers » et 3 pour les « extreme outliers ».

Avec cette méthode, nous avons détecté 25 points extrêmes, ce qui représente 1,6% des données.

Plusieurs méthodes de traitement sont possibles : winsorisation, transformation des données ou la suppression…

Vu le nombre limité des données détectées, nous allons les supprimer de notre échantillon.

Nous avons aussi effectué une analyse de variance sur ces variables par classe, et nous avons détecté très peu de variances pour certaines variables comme le montre la figure xxx :

Néanmoins, en l’absence d’une connaissance métiers des variables et leurs méthodes de construction, nous avons préféré de les garder dans un premier temps et faire une ACP qui pourrait mieux résumer leurs inerties.

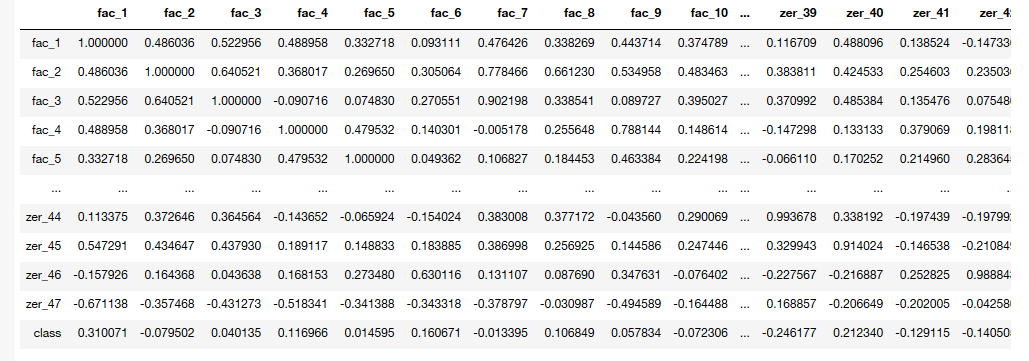
Nous allons par la suite construire des modèles avec et sans ces variables.

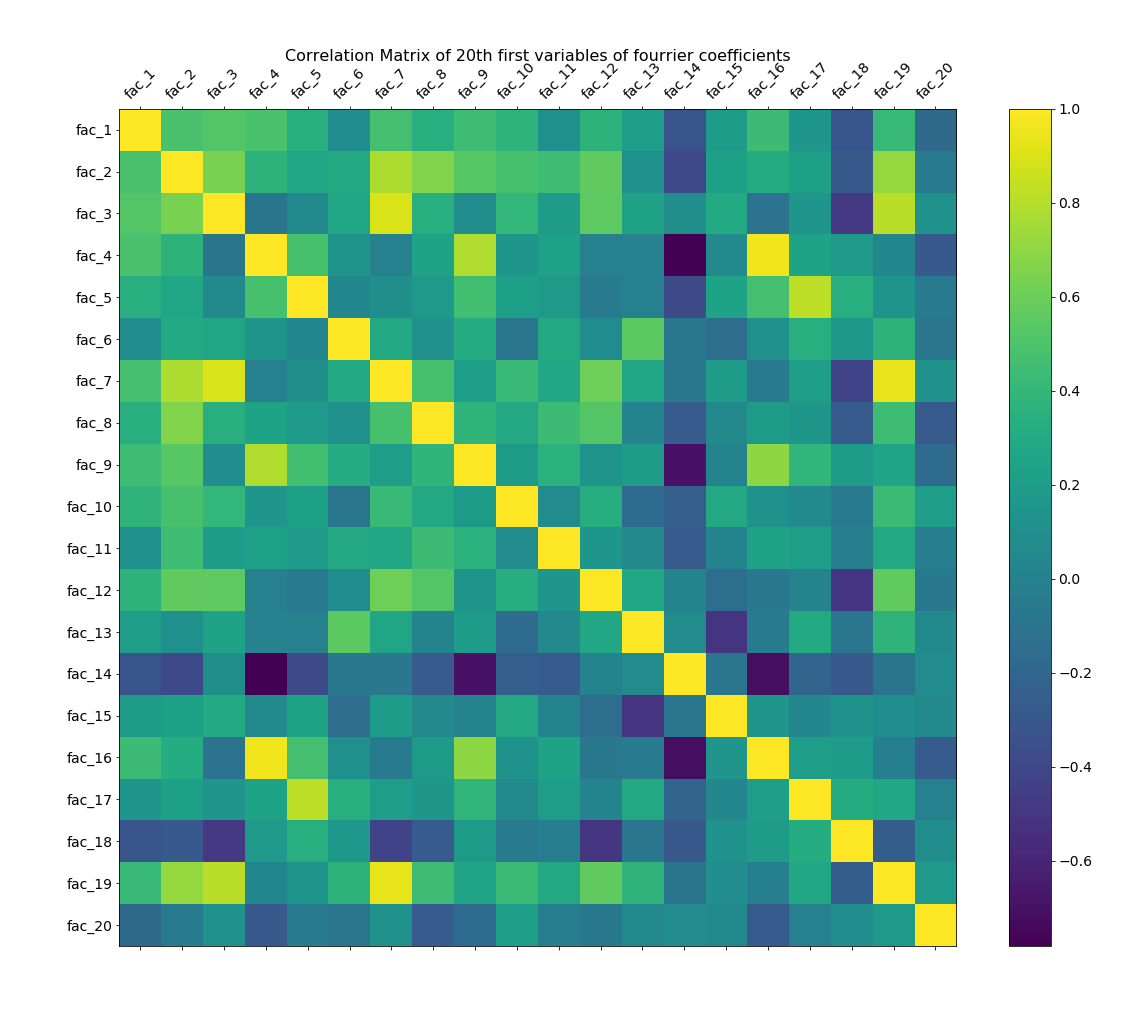
### Analyse bi variée

L’analyse bi-variée représente une étape importante dans le prétraitement des données.

En effet, nous avons des méthodes d’analyse qui sont sensibles aux corrélations entre variables, et des variables très corrélés peuvent impacté négativement les modèles (matrice de variance covariance non inversible, sur-apprentissage…)

La méthode SVM avec un kernel RBF en est un exemple car elle se base sur les distances entre les individus. Ainsi, si nous avons par exemple 11 attributs, mais l'un d'eux est répété 10 fois, ce dernier contribuera alors 10 fois plus à la distance que tout autre attribut, et le modèle final sera très impacté.

Vu le nombre élevé des variables, nous allons nous baser dans un premier temps sur une table analytique des corrélations comme le montre la figure ci-dessous :

Nous allons par la suite construire une table de corrélation par groupe de variable :

Nous remarquons que certaines variables sont très corrélées entre elle comme le montre la matrice de corrélation sur la figure xxx :

### 

Nous allons par la suite supprimer les composantes très fortement corrélé, en fixant un seuil de corrélation linéaire de 0.95 🡺 81 variables supprimées.

### Analyse AFM des données

L’Analyse Factorielle Multiple (Escofier and Pagès, 1990, 1994) apporte une solution très satisfaisante au problème de l’équilibre des groupes. Elle pourrait être vue comme une ACP dans laquelle l’influence des groupes de variables est équilibrée. C’est dans cet esprit que nous effectuons l’analyse de nos données.

Ci-dessous les étapes clés pour conduire une analyse AFM :

* Faire une analyse ACP de chaque groupe séparément,
* Equilibré chaque groupe en divisant toutes ses variables par la plus grande valeur propre.
* Faire une ACP globale sur l’ensemble des variables équilibrées.

Pour faire l’exercice, nous allons nous baser sur un module python, assez complet, pour les analyses factorielles : PRINCE [référence]

Une revue du code a été effectuée pour vérifier que l’implémentation technique de l’AFM dans le module est bien conforme à la théorie.

Lors de la construction de la méthode, nous devons choisir le nombre de composantes principales à retenir suite à cette analyse.

Etant donné que nous n’avons pas à priori cette information, nous allons effectuer cette analyse deux fois : la première fois en fixant le nombre de composantes assez élevé pour avoir une information sur la répartition d’inertie entre les composantes, et une deuxième fois en fixant le nombre de composante qui sera déterminé via la règle de Kaiser.

Suite à la première analyse, nous remarquons que l’inertie est répartie entre plusieurs composantes, et que la valeur des 8 premières valeurs propres est supérieure à 0,06 (I/P car l’implémentation de l’ACP dans le module PRINCE n’est pas normée).

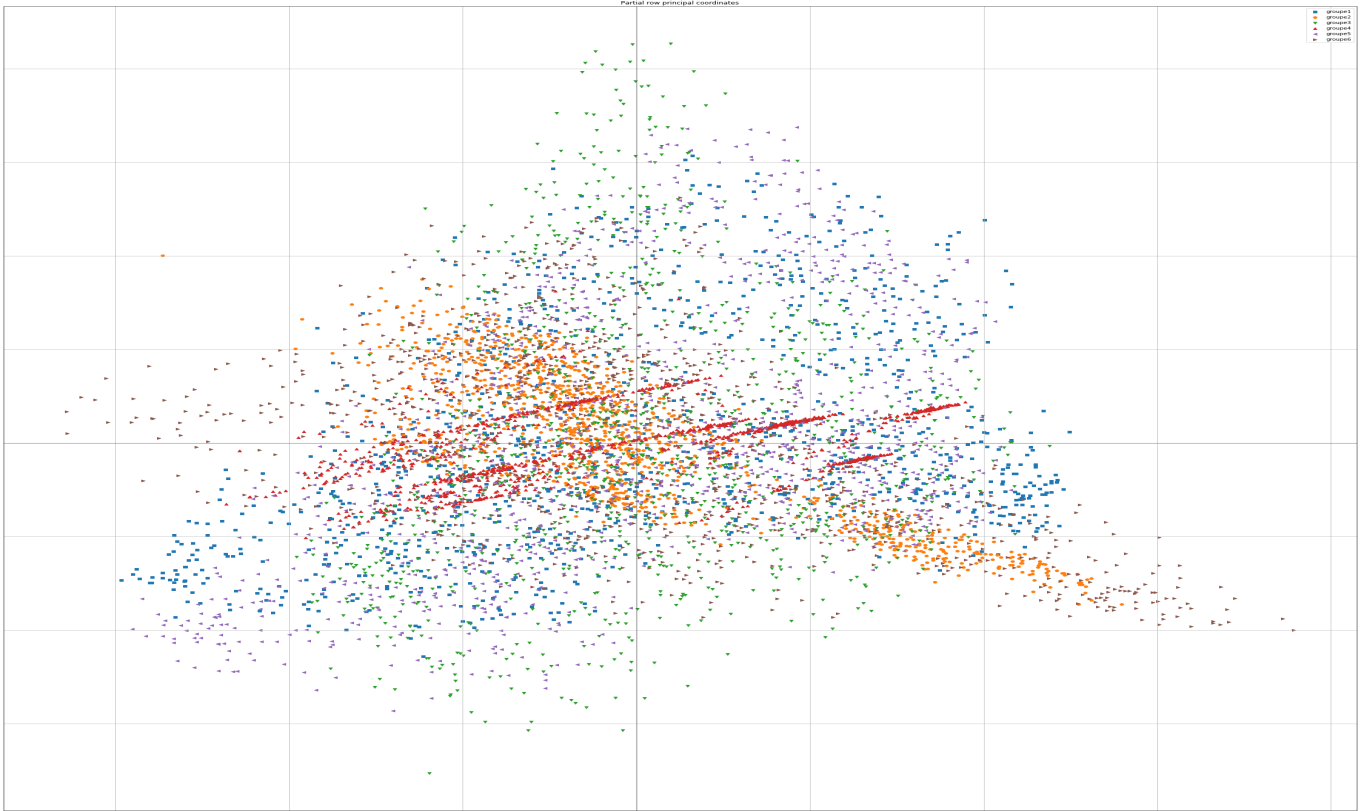
Nous allons garder les 91 premières valeurs propres pour réduire la dimension des données.

Nous remarquons aussi que les deux premières composantes résument uniquement 19% de l’inertie globale.

Ceci indique que la représentation graphique sur un plan factorielle serait partielle, mais elle permet néanmoins de visualiser la répartition des différentes classes sur le plan (fig xxx).

A travers cette présentation factorielle, nous constatons que :

* Hormis les classes 5 et 9 au centre qui sont superposées, les autres sont moyennement séparées.
* Le premier axe factoriel est caractérisé les groupes variables 4 et 2 (fou\_\* et mor\_\*).
* Le deuxième axe factoriel est caractérisé par le groupe de variable 3 (kar\_\*).
* Le premier axe caractérise les classes 8 et 7
* Le deuxième axe caractérise les classes 4,6 et 1.
* La présence de quelques points aberrants, non détectés lors de la première analyse. Vu la dimension réduite et le peu d’inertie, nous allons les laisser dans un premier temps et valider, s’il le faut, un autre modèles sans ces données.



A travers cette analyse, nous avons pu synthétiser les données et visualiser sur un plan factoriel les différentes classes.

Nous avons pu aussi à travers toute cette analyse exploratoire, mettre en évidence les différentes transformations, celles-ci seront effectuées uniquement sur les données d’apprentissage.

## Classification des individus via une méthode géométrique non supervisée

Afin de vérifier si les différentes classes peuvent être reconstruites d’une manière automatique, nous allons implémenter la méthode K-means.

K-Means est un algorithme d'apprentissage automatique non supervisé qui regroupe les données en k clusters (k entier strictement positif). Le nombre de clusters est défini par l'utilisateur et l'algorithme essaiera de regrouper les données même si ce nombre n'est pas optimal pour le cas spécifique.  
  
Afin de trouver le K optimal, nous allons nous appuyer sur la méthode du coude, en exécutant un clustering k-means pour une gamme de clusters k (de 3 à 15) et pour chaque valeur, nous calculons la somme des distances au carré de chaque point à son centre assigné (distorsions).

Nous allons effectuer cette classification en deux étapes :

* En utilisant un seul groupe de variables, en l’occurrence, le groupe 1 qui caractérise le premier axe factoriel.
* En utilisant l’ensemble des individus.

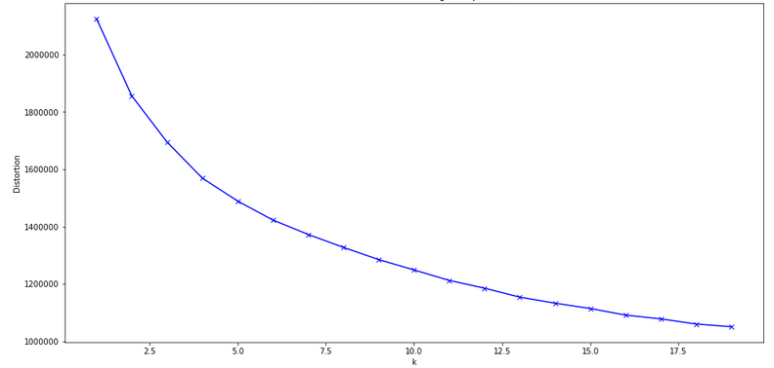
### Classification avec le groupe 1

Le groupe 1 est constitué des facteurs de transformation de fourrier.

Nous allons nous baser sur ces variables pour faire une classification des différents groupes.

#### Trouver le nombre de groupe optimal

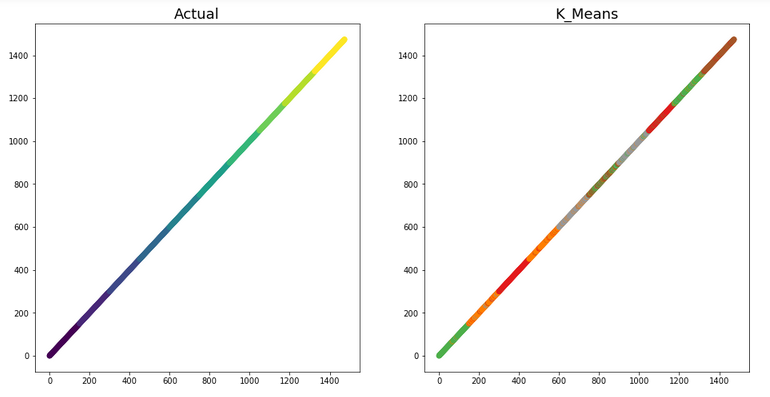
Afin de trouver le nombre de K optimal, nous allons visualiser la variation de la distorsion des différents groupes avec K :



Nous remarquons que la valeur optimale se situe entre 4 et 5.

En prenons 5 classes, nous aurons 5 classes de moins que celle que nous avons dans notre jeu de donnée.

Une visualisation des classes trouvées permet, comme le montre la figure xxx, permet de constater que seule la classe 9 est séparée.



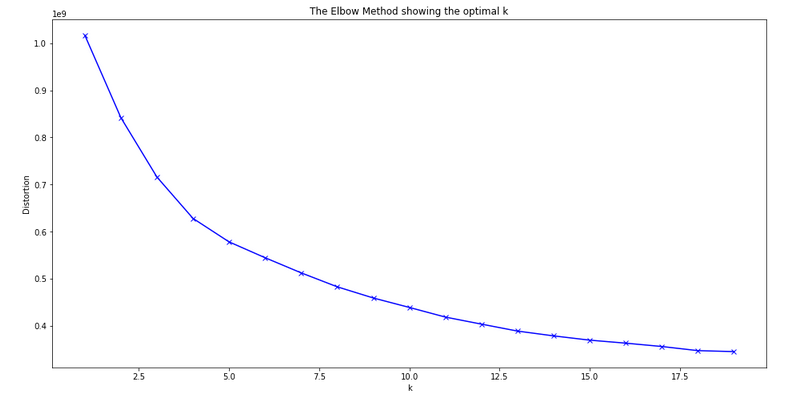
### En passant à K=10, nous remarquons que les classes 4,6 et 7 contiennent beaucoup de points mal classés.

### 

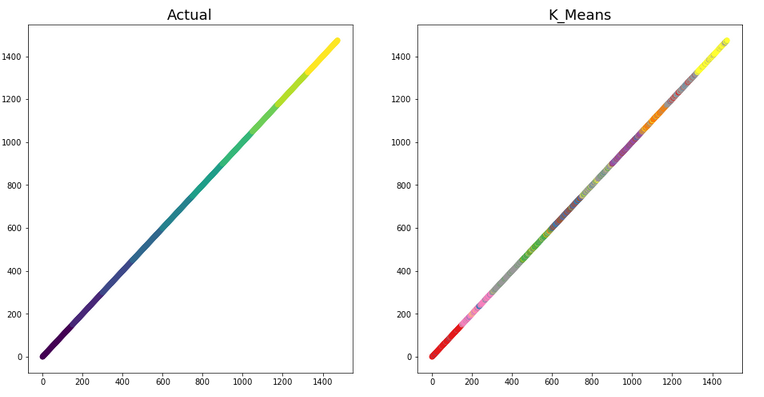
### Nous allons par la suite englober toutes les variables pour étudier si la classification pourrait s’améliorer.

### Classification avec l’ensemble des variables

En prenant en compte toutes les variables, nous remarquons que le niveau de coupure reste le même qu’auparavant : entre 4 et 5.



### En faisant une analyse K-means avec 10 classes, nous remarquons que les classes 4,5,7 et 8 contiennent beaucoup de points mal classés. L’amélioration apportée par l’ajout des autres groupes de variables n’est pas très significative :



Suite à cette analyse non supervisée, nous avons pu conclure qu’une séparation en 10 classes n’était pas possible à travers le modèle K-means.

Ceci prouve aussi que se baser uniquement sur les distances entre les points, n’est pas un critère suffisant pour classifier notre jeu de données en 10 classes.

**[TODO] : classification des variables**

## Elaboration d’un modèle statistique supervisé

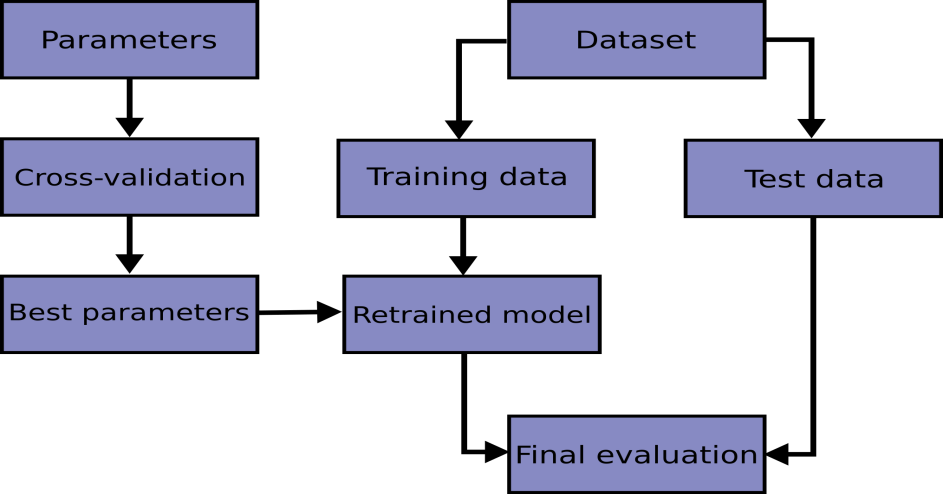
### Modèle SVM

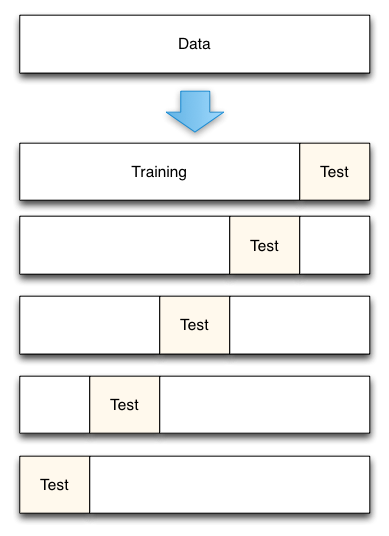
Avant de commencer notre modélisation, nous allons tout d’abord énumérer les méthodes et les hyper-paramètres qu’on souhaite tester pour cette famille de méthode :

* SVM avec séparation linéaire.
* SVM avec séparation RBF avec une variation du paramètre de régularisation C.
* SVM avec séparation polynomiale avec une variation du degré du polynôme.

D’une façon plus globale, la stratégie d’étude des différents modèles qu’on développera par la suite suivra les schémas ci-dessous :

(ref : https://scikit-learn.org/stable/modules/cross\_validation.html)





Ci-dessous les différents résultats obtenus :

|  |  |  |
| --- | --- | --- |
| ***Méthode*** | ***Paramètre*** | ***score de validation croisée (5 folder)*** |
| SVM linéaire | NA | 98,6% |
| SVM RBF | C varie dans [0.1,1,2,5,10,100] | 98,2% |
| SVM polynomial | Degré dans [1,2,3,5,10] | 98 ,3% |

Suite à ces différents tests, nous allons retenir le modèle SVM linéaire avec lequel on obtient le plus grand score via les tests de validations

### Modèle Random Forest

[TODO] : Petit rappel sur la méthode Randomforest en citant ses avantages par rapport à SVM et ANN et ses inconvénients

[TODO] : rappel des paramètres à tunner lors de l’élaboration du modèle

|  |  |
| --- | --- |
| Paramètre | Score |
| n\_estimators=5 | ***93,63%*** |
| n\_estimators=10 | ***94,15%*** |
| n\_estimators=20 | ***94,93%*** |
| n\_estimators=30 | ***94,89%*** |
| n\_estimators=40 | ***94,97%*** |
| n\_estimators=50 | ***95,09%*** |
| n\_estimators=100 | ***95,08%*** |

Nous avons un score de 95% qui maximum avec n\_estimators=50.

### Modèle Réseaux de neurones

[TODO] : Petit rappel sur la en citant ses avantages par rapport à SVM et ANN et ses inconvénients

[TODO] : rappel des paramètres à tunner lors de l’élaboration du modèle

|  |  |
| --- | --- |
| Paramètre | Score |
| n\_hidden\_layer\_sizes=5 | 0.8793766302038897 |
| n\_hidden\_layer\_sizes=10 | 0.95496498803918 |
| n\_hidden\_layer\_sizes=20 | 0.970227147774749 |
| n\_hidden\_layer\_sizes=30 | 0.9711510071408801 |
|  |  |
| n\_hidden\_layer\_sizes=40 | 0.9778045539849852 |
| n\_hidden\_layer\_sizes=50 | 0.9771140449006401 |
| n\_hidden\_layer\_sizes=100 | 0.9779396688656433 |
| n\_hidden\_layer\_sizes=200 | 0.9805044232654291 |

Comparaison des différents modèles à travers les données de test

|  |  |
| --- | --- |
| Modèle | Score |
| Random Forest | 69% |
| SVM linéaire | 99% |
| Réseaux de neurones | 98,7% |

## Synthèse

[TODO]

Commentaire des résultats

Pourquoi le RF n’est pas performant

Pourquoi le SVM est plus performant

Généralisation du modèle

[TODO] : Ajouter références et numéroter les figures