

# Chapitre 4

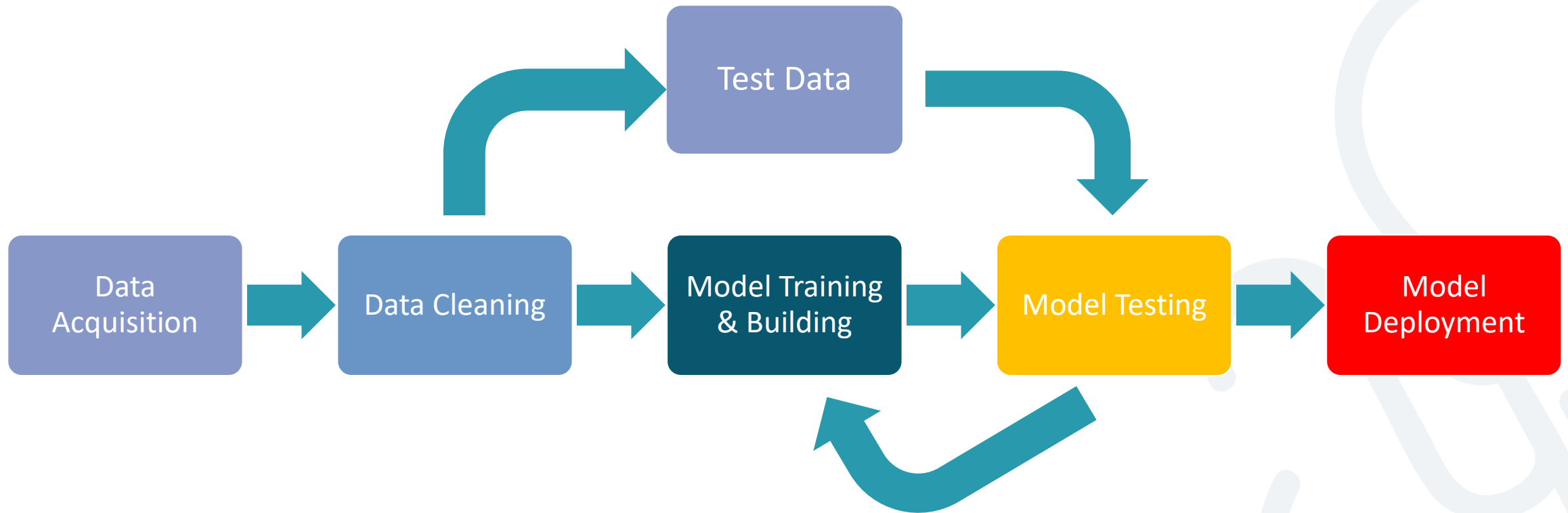
## Machine Learning

- Introduction
- Régression
- Clustering
- Classification
- NLP

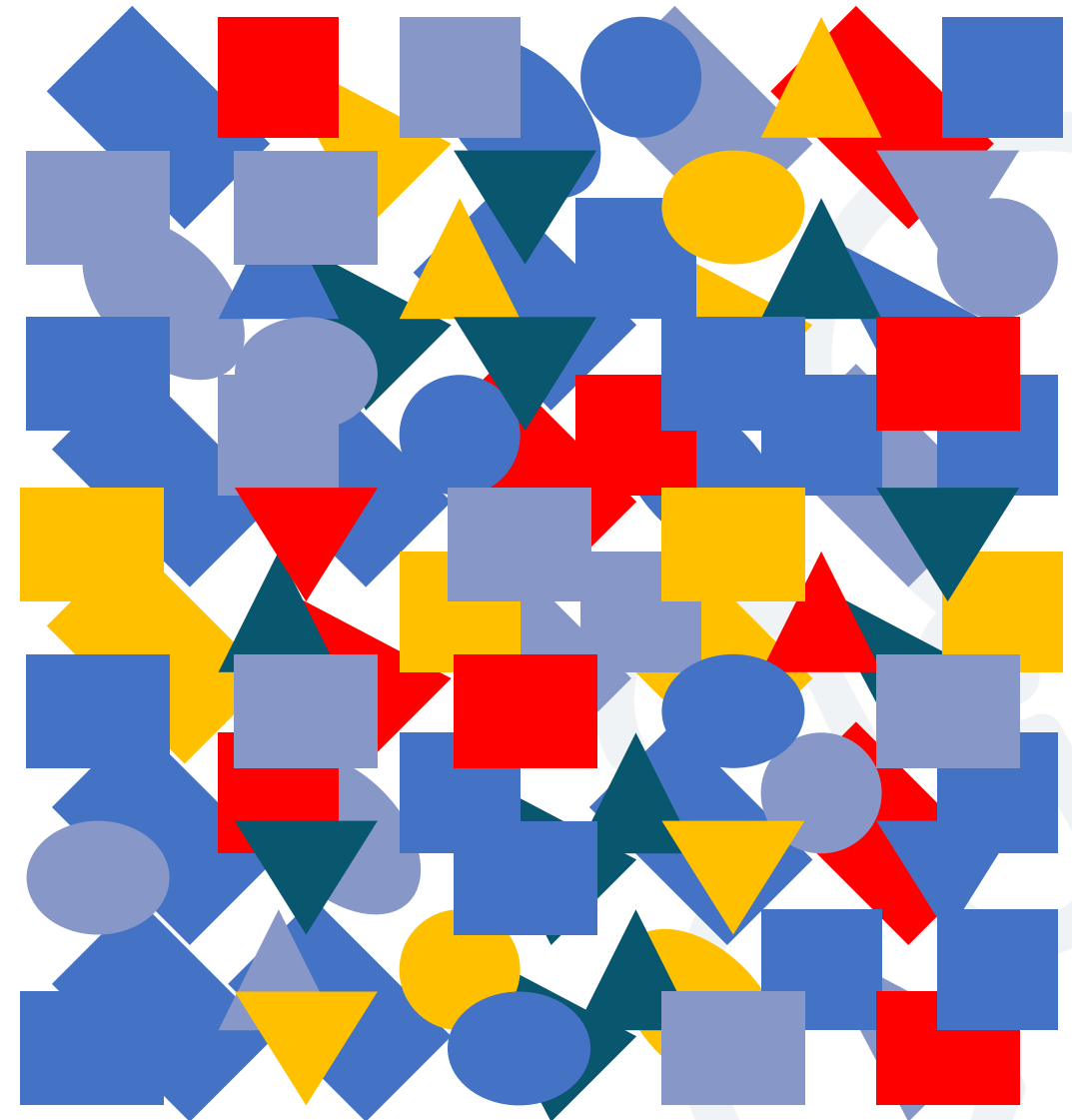
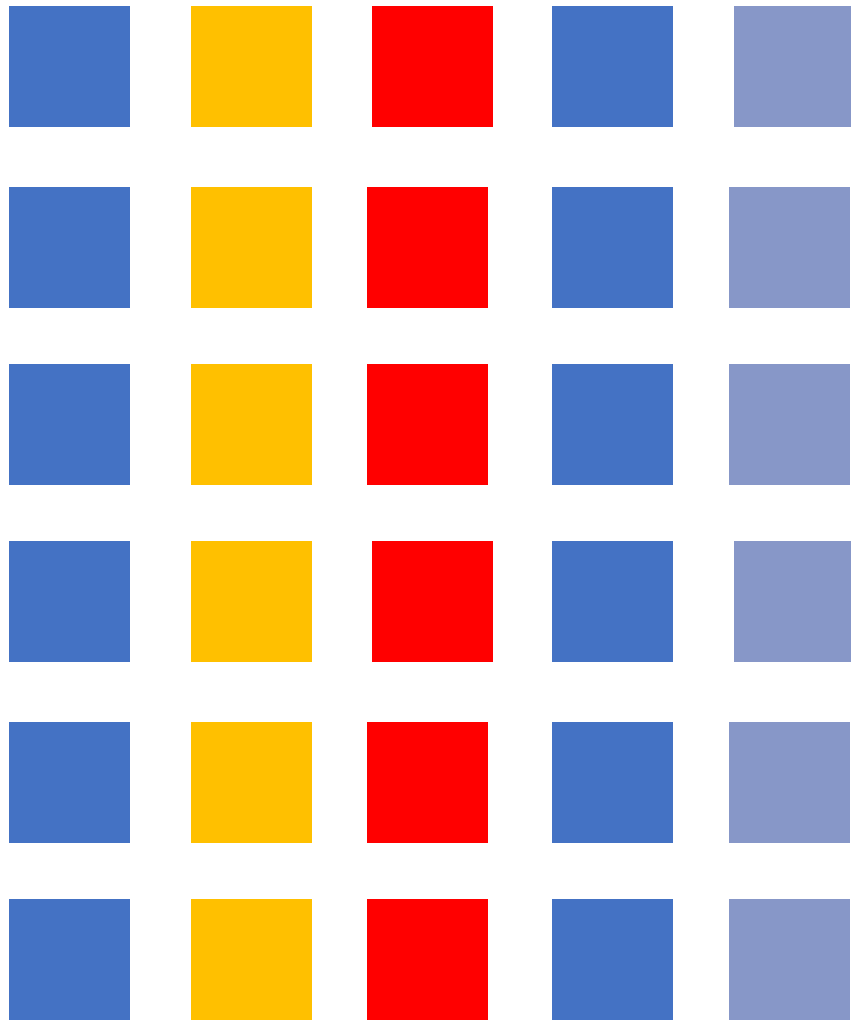
### Machine Learning :

- L'apprentissage automatique est une méthode d'analyse de données qui automatise la construction de modèles analytiques.
- À l'aide d'algorithmes qui apprennent de manière itérative à partir de données, l'apprentissage automatique permet aux ordinateurs de trouver des informations cachées sans être **explicitement** programmés où les chercher.

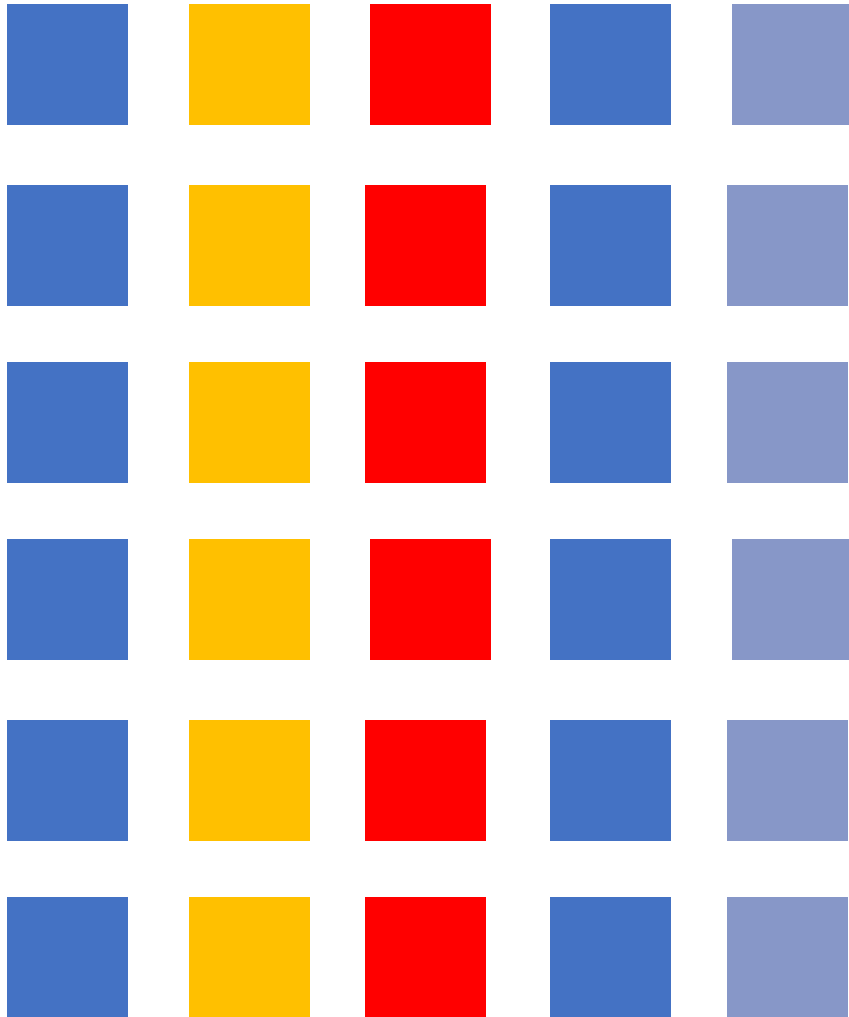
## Machine Learning



## Types des données



## Types des données



Blue	Yellow	Red	Blue	Light Blue
Blue	Yellow	Red	Blue	Light Blue
Blue	Yellow	Red	Blue	Light Blue
Blue	Yellow	Red	Blue	Light Blue
Blue	Yellow	Red	Blue	Light Blue
Blue	Yellow	Red	Blue	Light Blue

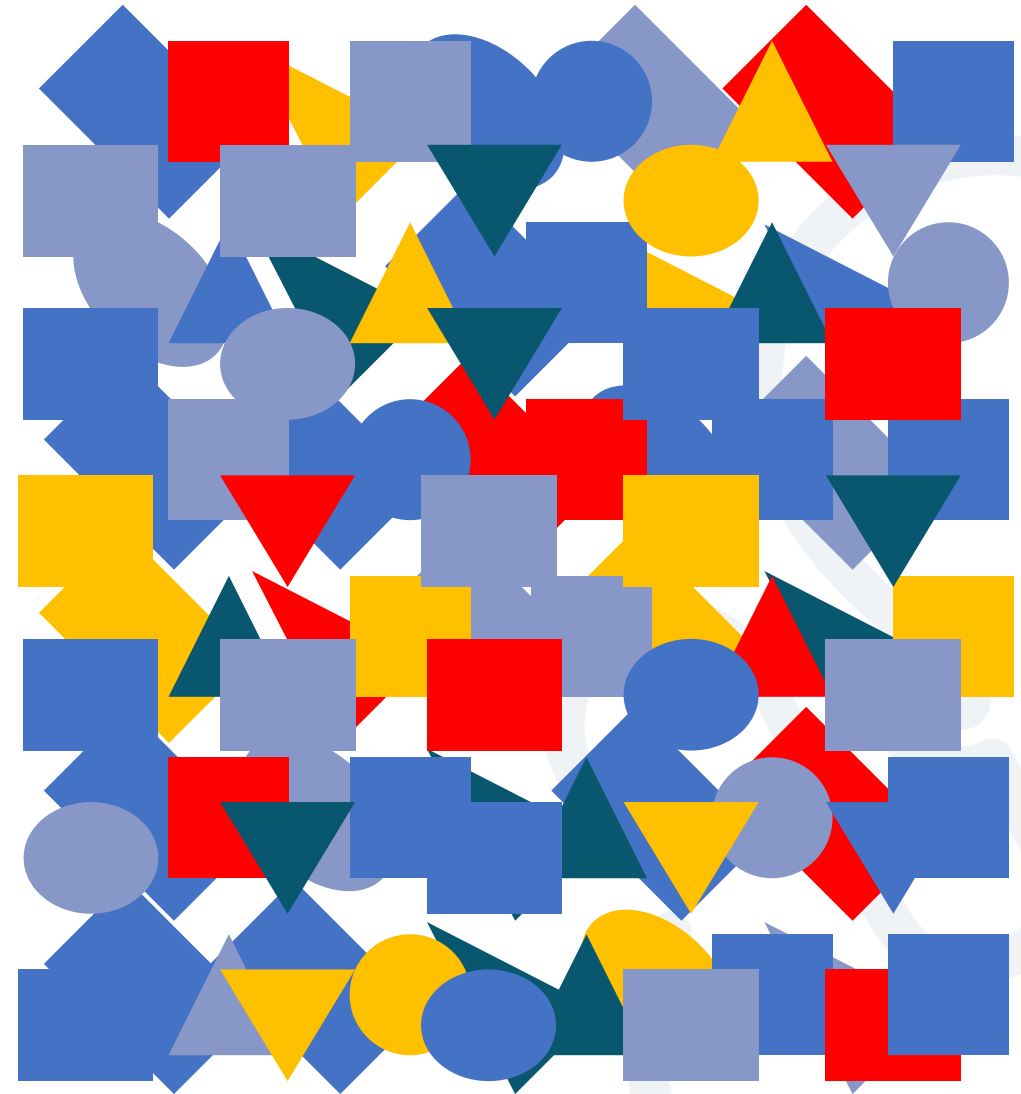
- Organisation logique
- Formats identiques
- Facilité de recherche
- Facilement stockable

Exemple : Données d'un tableau, base de donnée relationnelle

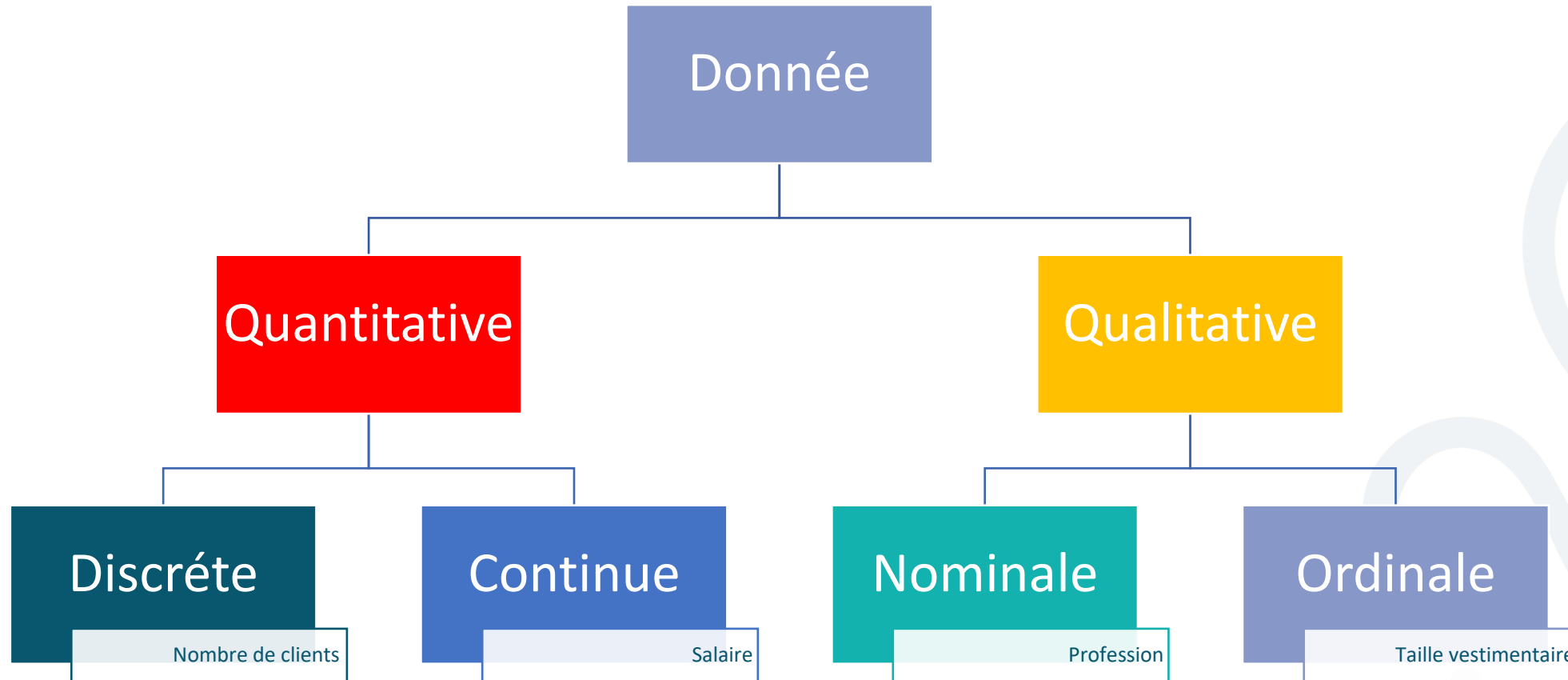
## Types des données

- Organisation complexe
- Données hétérogènes
- Recherche complexe
- Difficilement stockable en l'état (sans transformation)

Exemple : Corps d'un email, donnée multimédia, ...



## Types des données



<https://runawayhorse001.github.io/LearningApacheSpark/clustering.html>



## Machine Learning - apprentissage supervisé

- Les algorithmes d'apprentissage supervisé sont formés en utilisant des exemples étiquetés (annotés) tels qu'une entrée où la sortie désirée est connue.
- Grâce à des méthodes telles que la classification, la régression et le gradient boosting, l'apprentissage supervisé utilise des modèles pour prédire les valeurs de l'étiquette\classe sur des données non étiquetées.
- L'apprentissage supervisé est couramment utilisé dans les applications où les données historiques prédisent des événements futurs probables.

prévoir quand les transactions par carte de crédit sont susceptibles d'être frauduleuses  
ou  
prédire le prix d'une maison en fonction de différentes caractéristiques

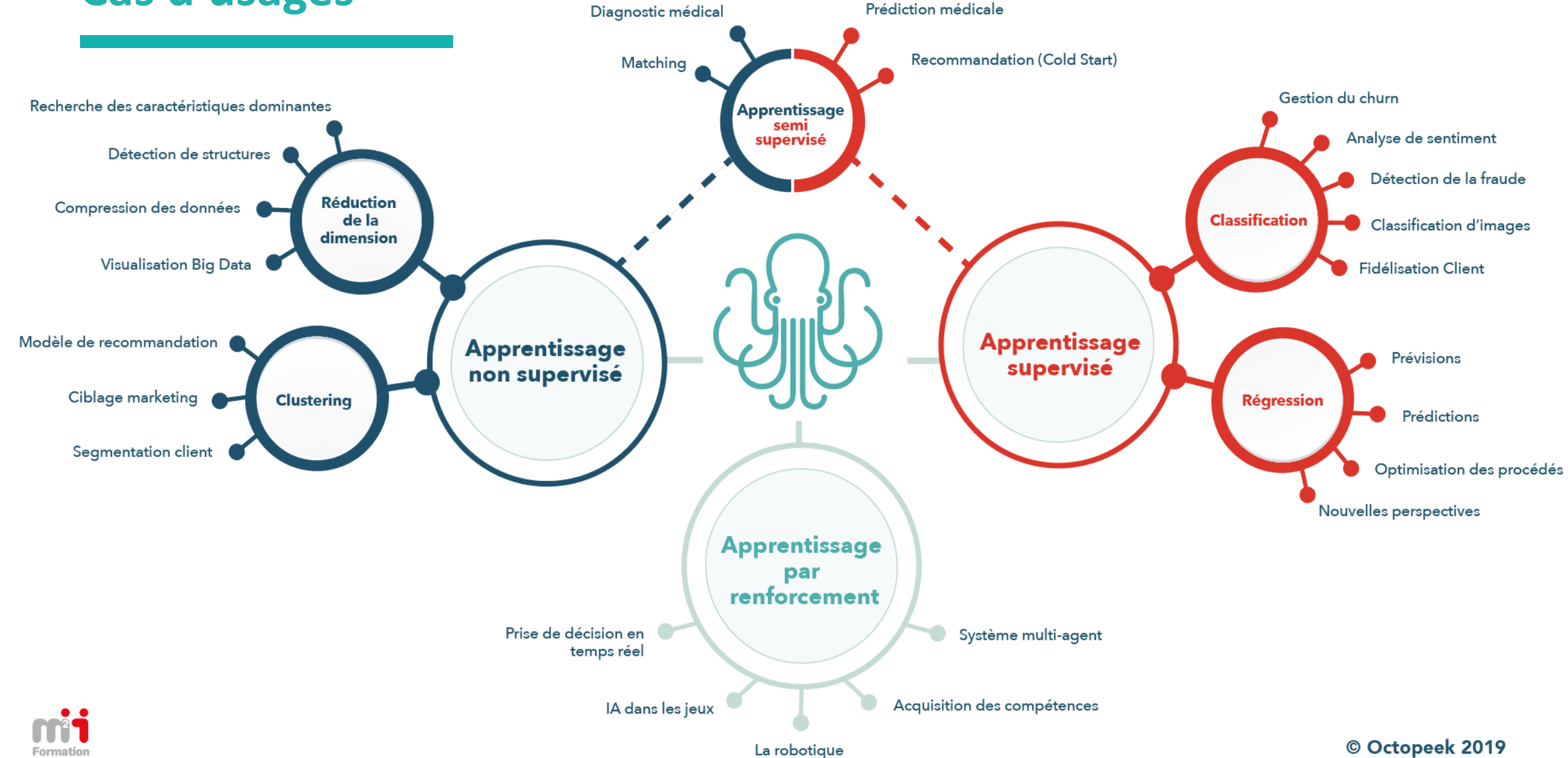
## Machine Learning - apprentissage non supervisé

- L'apprentissage non supervisé est utilisé contre des données sans étiquettes .
- Le modèle maintenant n'est pas dit la "bonne réponse". L'algorithme doit comprendre la donnée comme elle est.
- L'objectif est d'explorer les données et de trouver une structure à l'intérieur.

Par exemple, un algo de clustering peut trouver les attributs principaux qui séparent les segments de clientèle les uns des autres.

- Les techniques populaires comprennent les cartes auto-organisatrices, la cartographie du plus proche voisin, le clustering k-means.

## Cas d'usages





- Spark a sa propre librairie (MLlib) pour l'apprentissage automatique.
- MLlib de Spark est principalement conçu pour les tâches d'apprentissage supervisé et non supervisé.
- MLlib  $\geq$  Spark 2.0 utilise la syntaxe DataFrame.
- Pour pouvoir utiliser MLlib il faut formater vos données pour qu'elles finissent par avoir une ou deux colonnes:
  - Features + Labels (supervisées)
  - Features (non supervisées)
- La documentation Spark est très riche, n'hésitez pas aller sur le site pour la lire :  
<https://spark.apache.org/docs/latest/>

## Machine Learning avec Spark

Les Algorithmes de machine Learning qu'on étudie à cette formation:

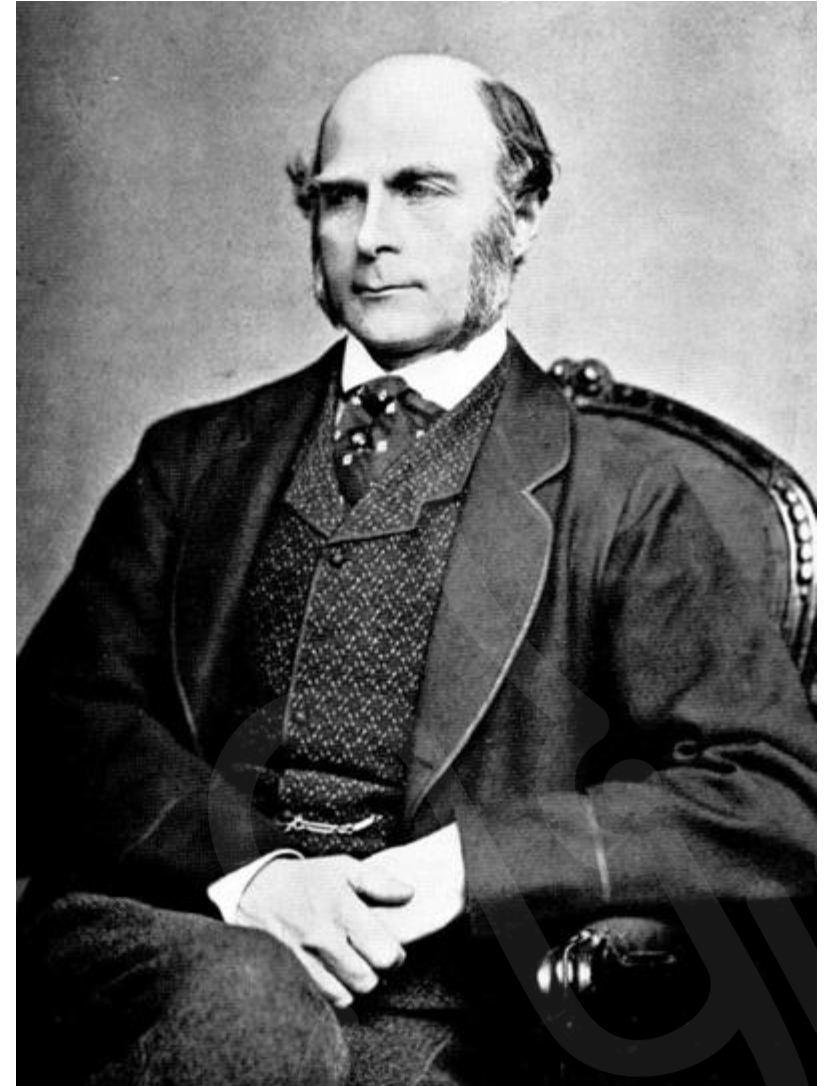
- La régression linéaire
- La régression logistique
- Les K-means
- Les SVMs
- Les arbres de décisions
- Les NLP

## Régression Linéaire :

Tout a commencé dans les années 1800 avec **Francis Galton**.

Galton étudiait la relation entre les parents et leurs enfants. Il voulait comprendre la relation entre la taille des pères et leurs fils. Ce qu'il a découvert, c'est que les fils ont tendance à être aussi grand que leur père.

Cependant, il a aussi déduit que la taille du fils avait tendance à être plus proche de la taille moyenne globale de toutes les personnes.



Prenons l'exemple d'un joueur de Basketball J1.

J1 mesure : 2,2 mètres.

Si J1 a un fils, il y a des chances qu'il soit assez grand aussi.

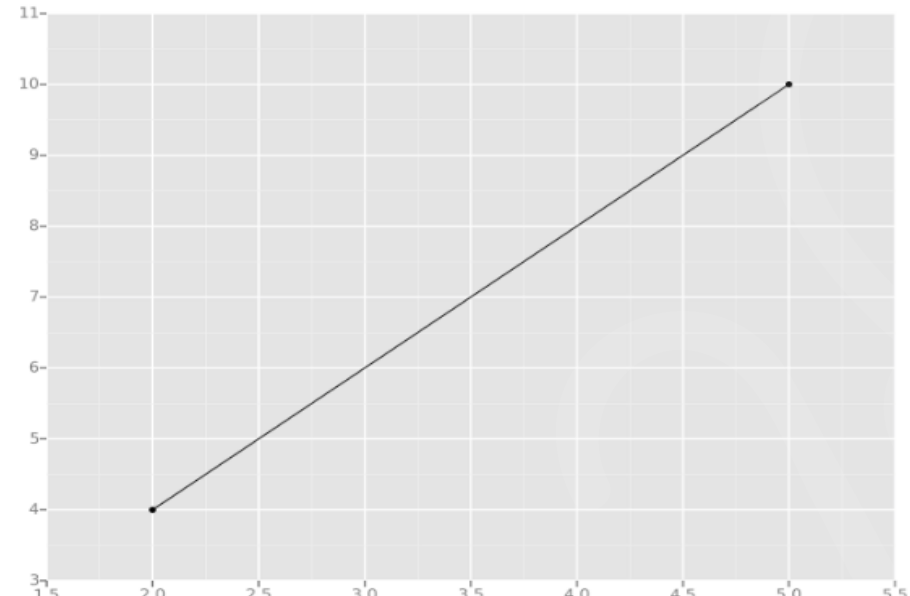
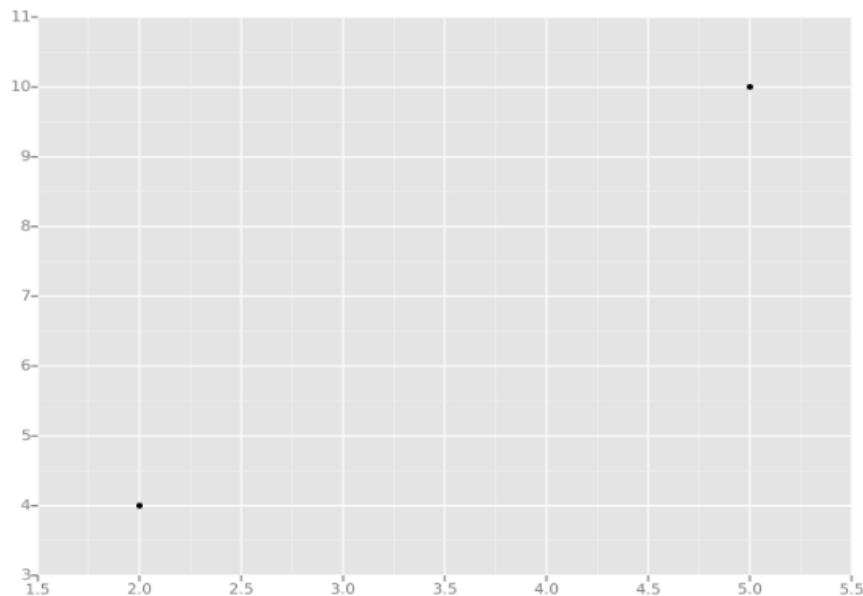
Cependant, J1 peut avoir une anomalie telle qu'il y a aussi de très bonnes chances que son fils ne soit pas aussi grand que lui.

Il s'avère que c'est le cas: le fils de J1 est assez grand, mais pas aussi grand que son père.

Galton a appelé ce phénomène la régression, comme dans "La taille du fils d'un père tend à régresser (ou dériver vers) la hauteur moyenne (moyenne)".

## Machine Learning avec Spark - Régression Linéaire

Prenons l'exemple le plus simple possible: calculer une régression avec seulement 2 points de données. Tout ce que nous essayons de faire lorsque nous calculons notre droite de régression, c'est de tracer une ligne aussi proche que possible de chaque point.

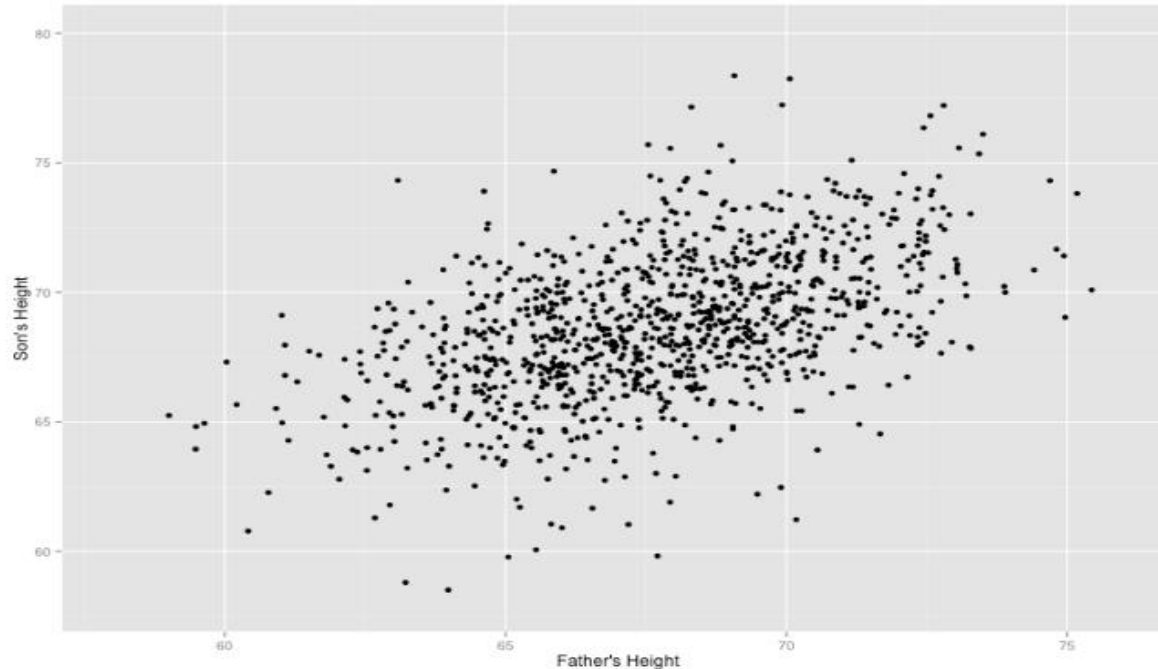




## Machine Learning avec Spark - Régression Linéaire

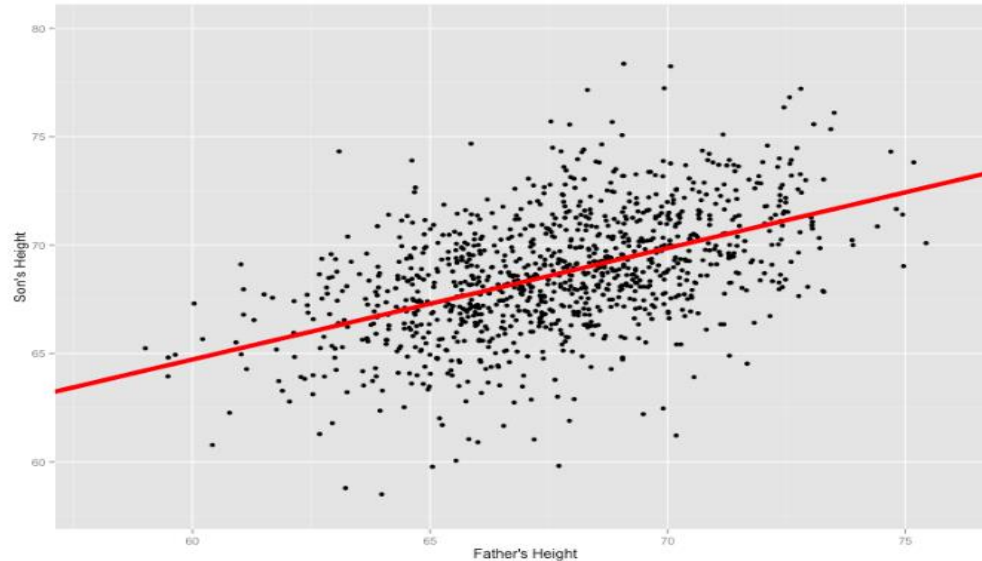
Cela peut nous être très utile quand on applique ce même concept à un graphique avec plus que deux points de données.

En faisant cela, nous pourrions prendre plusieurs hommes et les tailles de leurs fils et faire des choses comme dire à un homme combien mesure son fils ... avant même qu'il ait un fils! ➔ **Prédiction**



## Machine Learning avec Spark - Régression Linéaire

- Notre objectif avec la régression linéaire est de minimiser la distance verticale entre tous les points de données et notre ligne.
- Donc, en déterminant la meilleure ligne, nous essayons de minimiser la distance entre tous les points et leur distance à notre ligne.



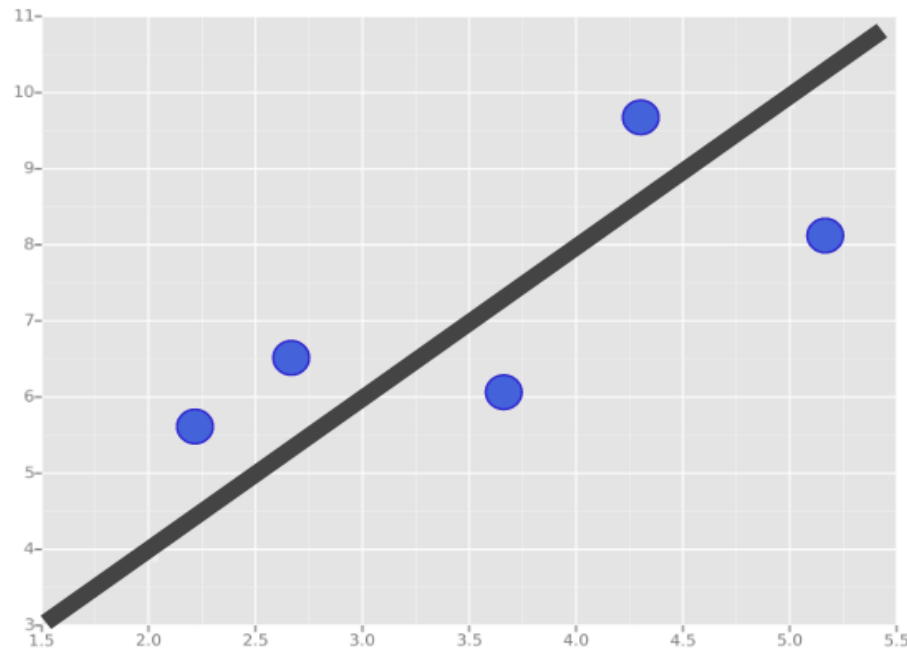
Il y a plusieurs façons de minimiser cela : **somme des erreurs au carré**, **somme des erreurs absolues**, etc...

## Machine Learning avec Spark - Régression Linéaire

Par exemple, l'une des méthodes les plus populaires est la méthode des **moindres carrés (RMSE)**.

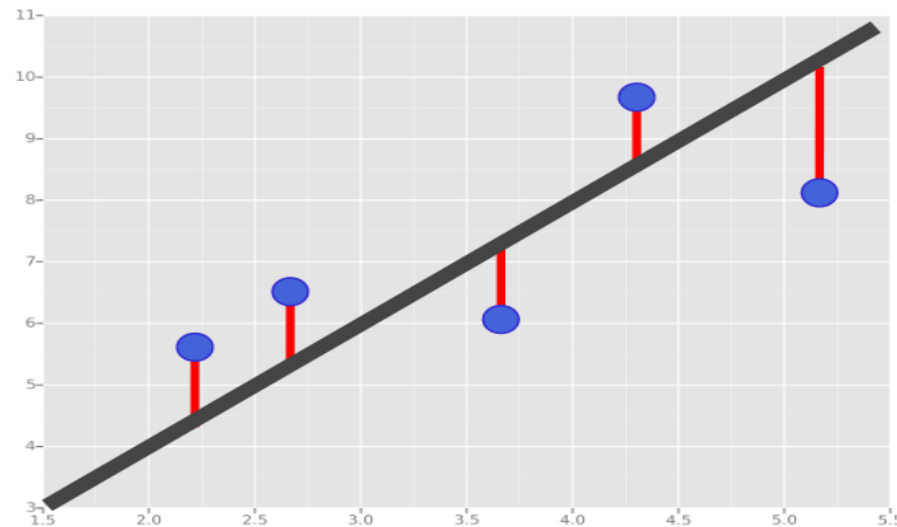
Maintenant, nous voulons ajuster une ligne de régression linéaire.

La question est, comment pouvons-nous décider quelle ligne est la plus appropriée?



## Machine Learning avec Spark - Régression Linéaire

Nous utiliserons la méthode des moindres carrés, qui est ajustée en minimisant la somme des carrés des résidus. Les résidus pour une observation est la différence entre l'observation (la valeur  $y$ ) et la ligne ajustée.



## Spark Machine Learning

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('rL').getOrCreate()
from pyspark.ml.regression import LinearRegression
training = spark.read.format('libsvm').load('sample_linear_regression_data.txt',
header = True)
training.show()
lr = LinearRegression (featuresCol='features', labelCol='label',
predictionCol='prediction')
lrModel = lr.fit(training)
lrModel.coefficients
training_summary = lrModel.summary
```

Sauf qu'ici, on n'a travaillé qu'avec les données training

### Séparations de train / test :

En apprentissage automatique séparé l'ensemble de données en un ensemble d'apprentissage et de test. Jusqu'à ici, nous avons entraînés sur TOUTES les données, ce que nous voulons généralement éviter de faire.

N'obtiendrons pas une évaluation juste de notre modèle en jugeant à quel point il fonctionne correctement sur les mêmes données!

Heureusement, Spark DataFrames a une méthode presque trop pratique pour diviser les données!

## Spark Machine Learning

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('rL').getOrCreate()
from pyspark.ml.regression import LinearRegression
all_data = spark.read.format('libsvm').load('sample_linear_regression_data.txt')
train_d, test_d = all_data.randomSplit([0.7, 0.3])
train_d.describe().show()
model = lr.fit(train_d)

print("Coefficients: " + str(lrModel.coefficients))
print("RMSE: %f" % training_summary.rootMeanSquaredError)
print("R2: %f" % training_summary.r2)
```

## Spark Machine Learning

```
test_results = model.evaluate(test_d)  
unlabeled_d = test_d.select('features')  
  
predict = model.transform(unlabeled_d)  
predict.show()
```

Maintenant on va prédire sur la donnée où on ne connaît pas le prix de l'immobilier



### RL Evaluation :

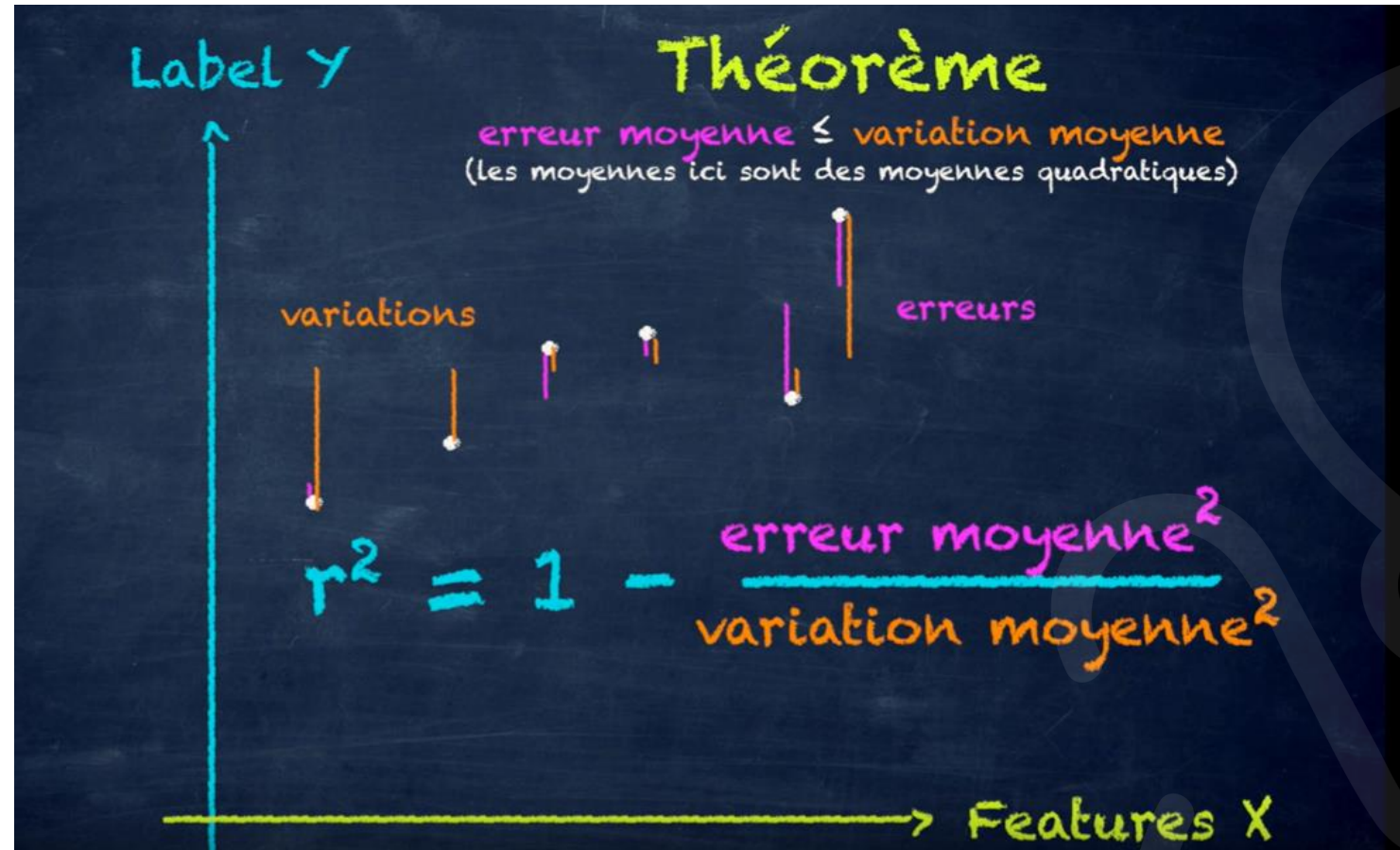
Après l'étape de l'élaboration d'un modèle, l'étape qui suit est de l'évaluer, et tester si la prédiction (succès) de notre modèle est bonne.

Il existe plusieurs types d'évaluation de modèles, dans notre cas il s'agit de prédire une valeur continue (régression linéaire), parmi les exemples d'évaluations on trouve :

- Mean Absolute Error
- Mean Squared Error
- Root Mean Square Error
- R Squared Values

RL Evaluation :

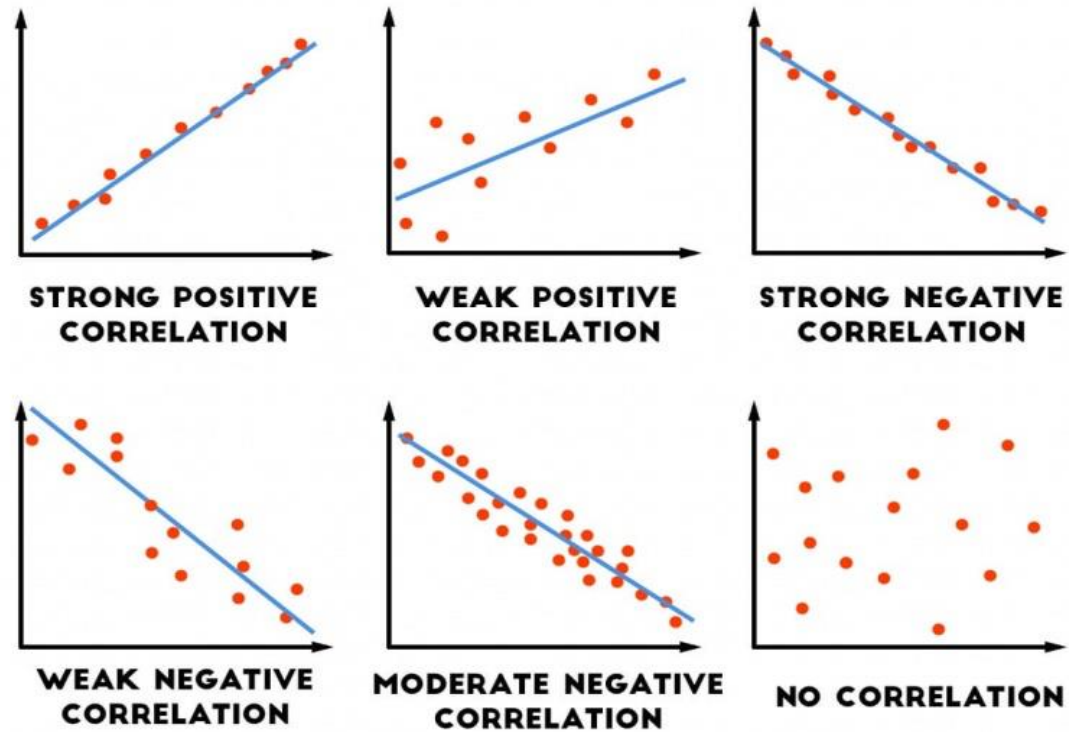
R-squared



## Spark Machine Learning

RL Evaluation :

R-squared



RL Evaluation :

Erreur absolue moyenne (MAE)

C'est la moyenne de la valeur absolue des erreurs.

Facile à comprendre, juste une erreur moyenne!

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

# Metrics d'évaluation d'un modèle

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

## Régression

- RMSE
- MSAE
- R Square
- Adjusted R Square

## Classification

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

## Non- supervisé

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

## Autre

- CV error
- Méthodes heuristiques pour trouver k
- BLEU Score (NLP)

# Metrics d'évaluation d'un modèle

$$\hat{R}^2 = 1 - \frac{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}{\frac{1}{n} \sum_{j=1}^n (y_j - \bar{y})^2}$$

$$R_{adj}^2 = 1 - \left[ \frac{(1 - \hat{R}^2)(n - 1)}{n - k - 1} \right]$$

## Régression

- RMSE
- MSAE
- R Square
- Adjusted R Square

## Classification

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

## Non- supervisé

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

## Autre

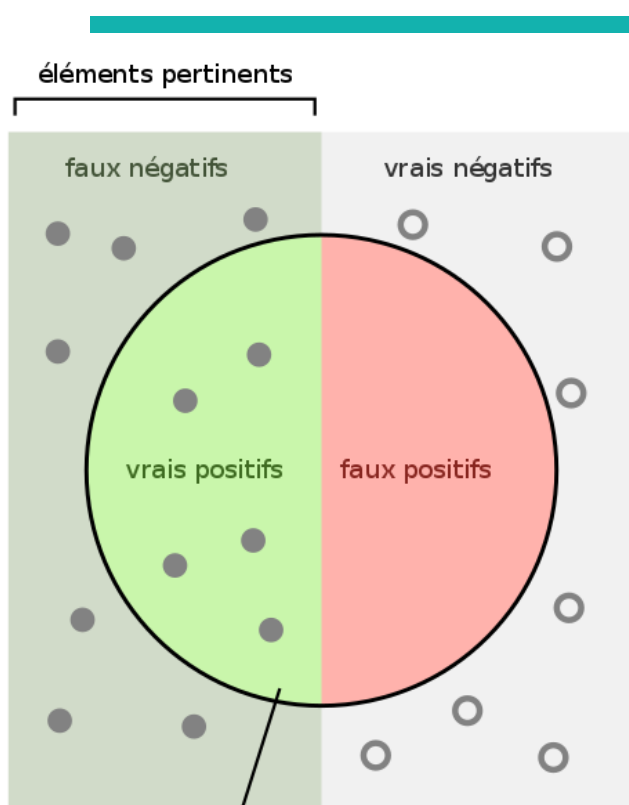
- CV error
- Méthodes heuristiques pour trouver k
- BLEU Score (NLP)

# Metrics d'évaluation d'un modèle

Case 1		Case 2			Case 3		
Var1	Y	Var1	Var2	Y	Var1	Var2	Y
x1	y1	x1	$2 * x1$	y1	x1	$2 * x1 + 0.1$	y1
x2	y2	x2	$2 * x2$	y2	x2	$2 * x2$	y2
x3	y3	x3	$2 * x3$	y3	x3	$2 * x3 + 0.1$	y3
x4	y4	x4	$2 * x4$	y4	x4	$2 * x4$	y4
x5	y5	x5	$2 * x5$	y5	x5	$2 * x5 + 0.1$	y5

	Case 1	Case 2	Case 3
R_squared	0.985	0.985	0.987
Adj_R_squared	0.981	0.971	0.975

# Metrics d'évaluation d'un modèle



éléments sélectionnés

Combien de candidats sélectionnés sont pertinents ?

$$\text{Précision} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux positifs}}$$

Combien d'éléments pertinents sont sélectionnés ?

$$\text{Rappel} = \frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux négatifs}}$$

## Régression

- RMSE
- MSAE
- R Square
- Adjusted R Square

## Classification

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

## Non- supervisé

- Davies-Bouldin Index
- Dunn Index
- Silhouette Coefficient

## Autre

- CV error
- Méthodes heuristiques pour trouver k
- BLEU Score (NLP)



# Metrics d'évaluation d'un modèle

## – Matrice de Confusion

	Mail	Spam
Prédit Mail	TP	FP
Prédit SPAM	FN	TN

$$\text{Précision} = \frac{TP}{TP+FN}$$

$$\text{Rappel} = \frac{TN}{TN+FP}$$

### Régression

- RMSE
- MSAE
- R Square
- Adjusted R Square

### Classification

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

### Non- supervisé

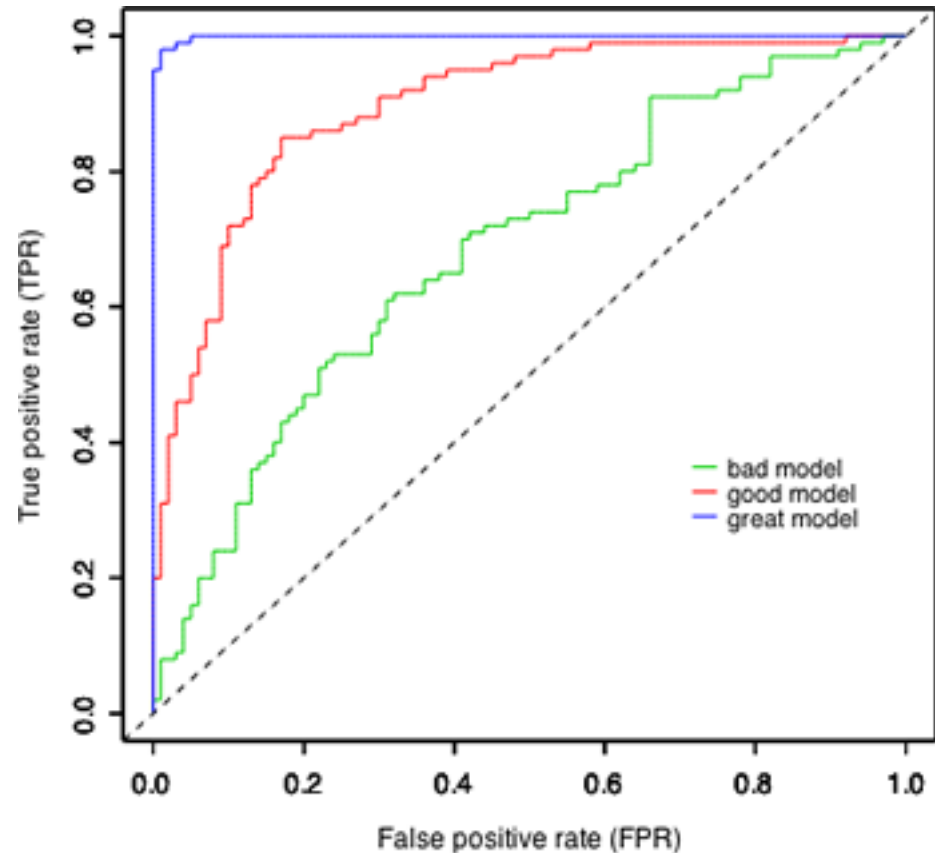
- Davies-Bouldin Index
- Dunn Index
- Silhouette Coefficient

### Autre

- CV error
- Méthodes heuristiques pour trouver k
- BLEU Score (NLP)

# Metrics d'évaluation d'un modèle

## – ROC - AUC



### Régression

- RMSE
- MSAE
- R Square
- Adjusted R Square

### Classification

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

### Non- supervisé

- Davies-Bouldin Index
- Dunn Index
- Silhouette Coefficient

### Autre

- CV error
- Méthodes heuristiques pour trouver k
- BLEU Score (NLP)

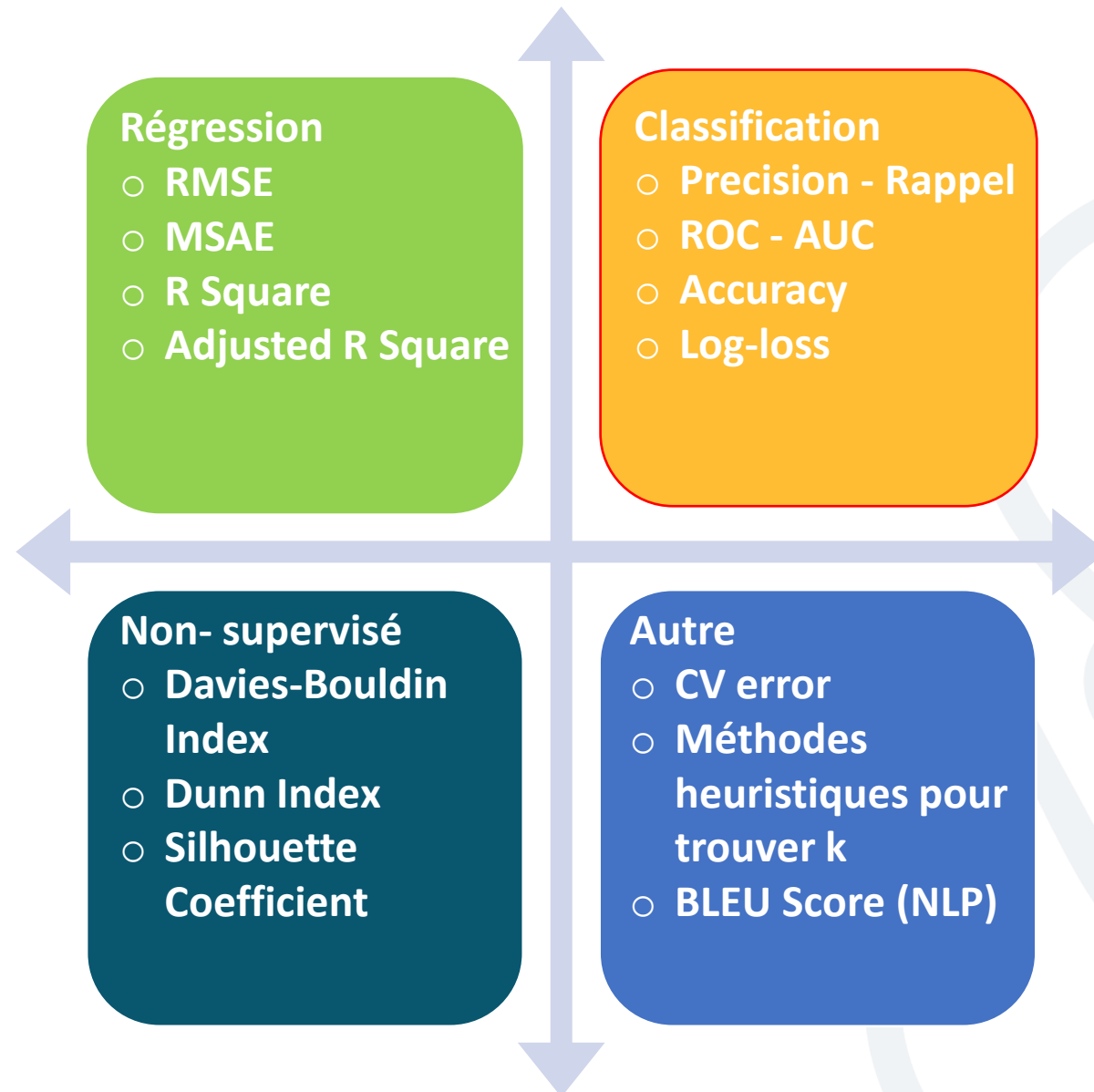
# Metrics d'évaluation d'un modèle

– Accuracy, F1 et log-loss

$$Accuracy = \frac{TP+TN}{P+N}$$

$$F1 = \frac{2*Précision*Rappel}{Précision+Rappel}$$

$$\log - loss = (1 - y)\log(1 - p) - y\log(p)$$



# Metrics d'évaluation d'un modèle

S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Actual (Balanced)	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Predicted (Model 1)	0.1	0.1	0.1	0.1	0.1	0.1	0.6	0.6	0.5	0.5	0.9	0.9	0.9	0.9	0.9	0.9
Predicted (Model 2)	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.8	0.8	0.8	0.8

	F1 (threshold=0.5)	F1 (Threshold which maximize score)	ROC-AUC	Log-Loss
<b>Model 1</b>	0.88	0.88	0.94	0.28
<b>Model 2</b>	0.67	1	1	0.6

# Metrics d'évaluation d'un modèle

S.No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Actual (Imbalanced)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Predicted (Model 1)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.9
Predicted (Model 2)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.9	0.9	0.9	0.9

	F1 (threshold=0.5)	ROC-AUC	Log-Loss
<b>Model 1</b>	0.8	0.83	0.24
<b>Model 2</b>	0.86	0.96	0.24

# Metrics d'évaluation d'un modèle

$$DB = \frac{1}{n} \sum_{i=1}^n \max_{i \neq j} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

$$D = \frac{\min_{1 \leq i < j \leq n} d(i, j)}{\max_{1 \leq k \leq n} d'(k)}$$

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

## Régression

- RMSE
- MSAE
- R Square
- Adjusted R Square

## Classification

- Precision - Rappel
- ROC - AUC
- Accuracy
- Log-loss

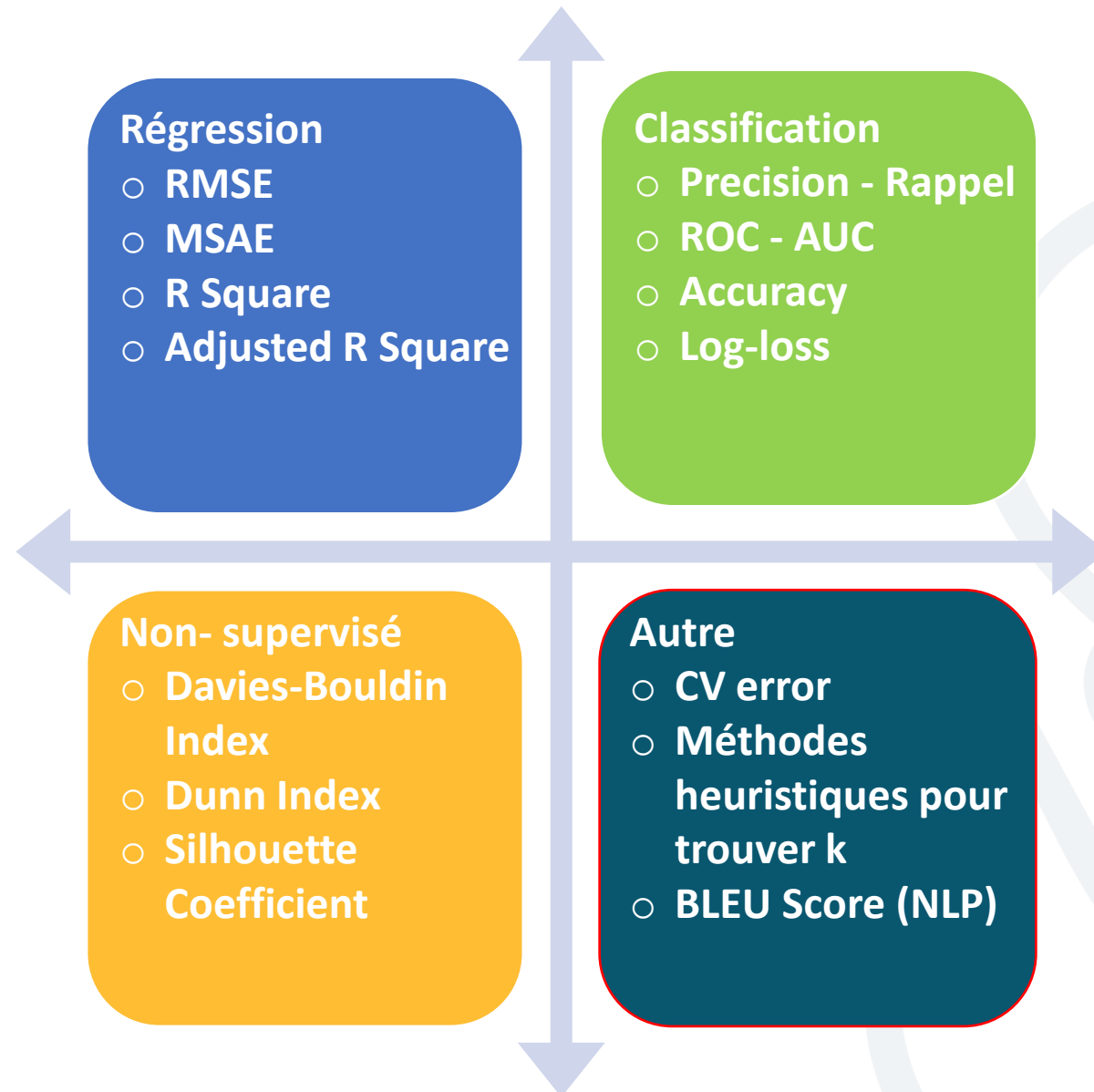
## Non- supervisé

- Davies-Bouldin Index
- Dunn Index
- Silhouette Coefficient

## Autre

- CV error
- Méthodes heuristiques pour trouver k
- BLEU Score (NLP)

# Metrics d'évaluation d'un modèle



## Spark Machine Learning - RL

Exemple réel : L'exercice est sur : Mlib\_RL

Data : Données client e-commerce pour les sites Web et les applications mobiles des entreprises.

File : Ecommerce\_Customers.csv

But : L'idée est de tenter de prédire la dépense totale d'un client (une valeur monétaire continue).

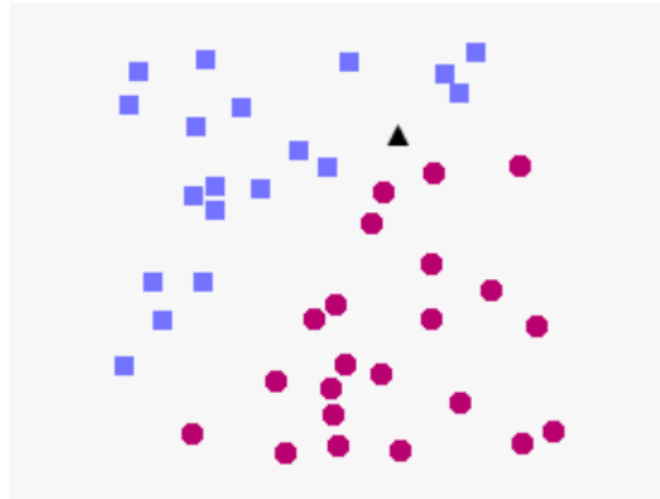




### Classification:

Considérons l'exemple suivant. On se place dans le plan, et l'on dispose de deux catégories : les ronds rouges et les carrés bleus. Cependant, la frontière entre ces deux régions n'est pas connue. Ce que l'on veut, c'est que quand on lui présentera un nouveau point dont on ne connaît que la position dans le plan, l'algorithme de classification sera capable de prédire si ce nouveau point est un carré rouge ou un rond bleu.

Autrement dit, il faut être capable de trouver la frontière entre les différentes catégories. Si on connaît la frontière, savoir de quel côté de la frontière appartient le point, et donc à quelle catégorie il appartient.



## Spark Machine Learning – SVM

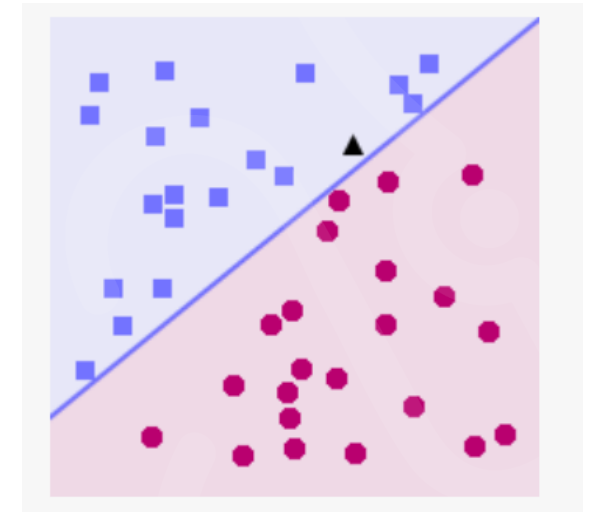
### Classification:

L'acronyme de *Support Vector Machines*, soit *machines à vecteurs support* en français, parfois traduit par *séparateur à vaste marges*

Un algorithme d'apprentissage automatique, et qui est très efficace dans les problèmes de classification

Pour que le SVM puisse trouver cette frontière, il est nécessaire de lui donner des **données d'entraînement**

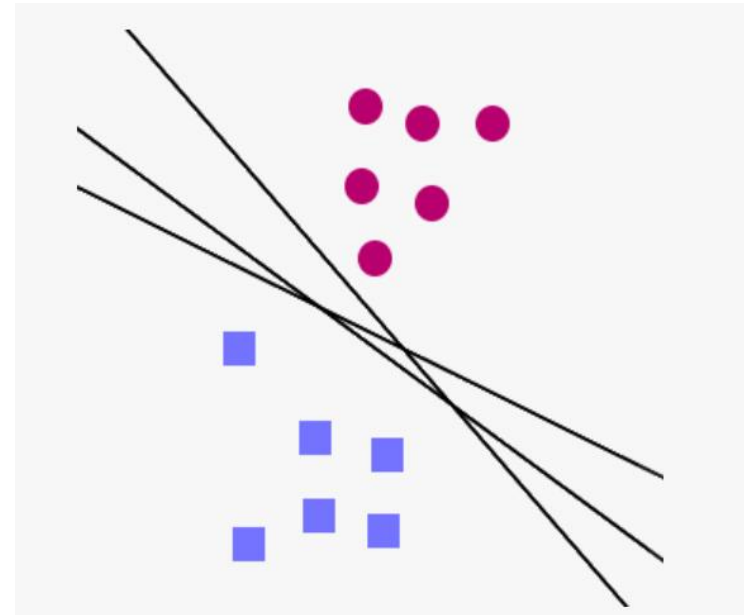
Une fois la phase d'entraînement terminée, le SVM a ainsi trouvé, à partir de données d'entraînement, l'emplacement supposé de la frontière. En quelque sorte, il a « appris » l'emplacement de la frontière grâce aux données d'entraînement



## Spark Machine Learning – SVM

Classification:

laquelle choisir ?



## Spark Machine Learning – SVM

### Classification:

c'est bien la frontière la plus éloignée de tous les points d'entraînement qui est optimale, on dit qu'elle a la meilleure **capacité de généralisation**



## Spark Machine Learning – SVM

### Avantages SVM:

- Les SVMs sont efficaces lorsque le nombre de fonctionnalités est assez grand.
- Il fonctionne efficacement même si le nombre d'entités est supérieur au nombre d'échantillons.
- Les données non linéaires peuvent également être classées à l'aide d'hyperplans personnalisés construits en utilisant l'astuce du noyau.
- Il s'agit d'un modèle robuste pour résoudre les problèmes de prédiction puisqu'il maximise la marge.

## Spark Machine Learning – SVM

### Inconvénients SVM:

- La plus grande limitation de support vector machine est le choix du noyau. Le mauvais choix du noyau peut conduire à une augmentation du pourcentage d'erreur.
- Avec un plus grand nombre d'échantillons, il commence à donner des performances médiocres.
- Les SVMs ont une bonne performance de généralisation, mais ils peuvent être extrêmement lents dans la phase de test.

### Applications SVM:

- Classification d'expression faciale: les SVMs peuvent être utilisés pour classer les expressions faciales. Il utilise des modèles statistiques de forme et de SVMs.
- Reconnaissance vocale: les SVMs sont utilisés pour accepter les mots-clés et rejeter les non-Mots-clés et construire un modèle pour reconnaître la parole.
- Reconnaissance manuscrite des chiffres: les classificateurs vectoriels de soutien peuvent être appliqués à la reconnaissance des chiffres manuscrits isolés optiquement numérisés.
- Catégorisation de texte: dans la récupération d'informations, puis catégorisation des données à l'aide d'étiquettes peut être fait par SVM.

## Spark Machine Learning – SVM

Exercice:

Explorer les étapes d'apprentissage d'un SVM et essayer d'améliorer les scores d'évaluation (Split, outil de comparaison, hyperparametres)

Pyspark\_SVM\_exo



[doc](#)



## Spark Machine Learning – apprentissage non supervisé

Nous avons vu comment gérer les données étiquetées, mais qu'en est-il des données non étiquetées ?

Souvent, vous essayez de créer des groupes à partir de données, au lieu d'essayer de prévoir des classes ou des valeurs.

Ce type de problème est connu sous le nom de clustering, vous pouvez le considérer comme une tentative de création d'étiquettes.

Vous entrez des données non étiquetées et l'algorithme d'apprentissage non supervisé renvoie les clusters possibles des données.

## Spark Machine Learning – apprentissage non supervisé

En raison de la nature de ce problème, il peut être difficile d'évaluer les groupes pour déterminer s'ils sont corrects. ( connaissance métier nécessaire)

Par exemple, vous pourriez regrouper les tumeurs en deux groupes, en espérant séparer les tumeurs bénignes des tumeurs malignes.

Mais il n'y a aucune garantie que les clusters tomberont dans ce sens, elles se diviseront simplement en deux groupes les plus séparables (décider à l'avance le nombre de clusters que vous prévoyez de créer)

## Spark Machine Learning – apprentissage non supervisé

Beaucoup de problèmes de clustering n'ont pas une approche ou une réponse correcte à 100%, c'est la nature de l'apprentissage non supervisé!

Continuons le clustering avec l'algorithme le plus utilisé: les K-means.

## Spark Machine Learning – K-Means

**K Means** Clustering est un algorithme d'apprentissage non supervisé itératif qui peut être défini comme la tâche d'identifier des sous-groupes non superposés dans les données de telle sorte que les points de données dans le même sous-groupe (cluster) sont très similaires

Alors, à quoi ressemble un problème de clustering typique?

- Documents similaires

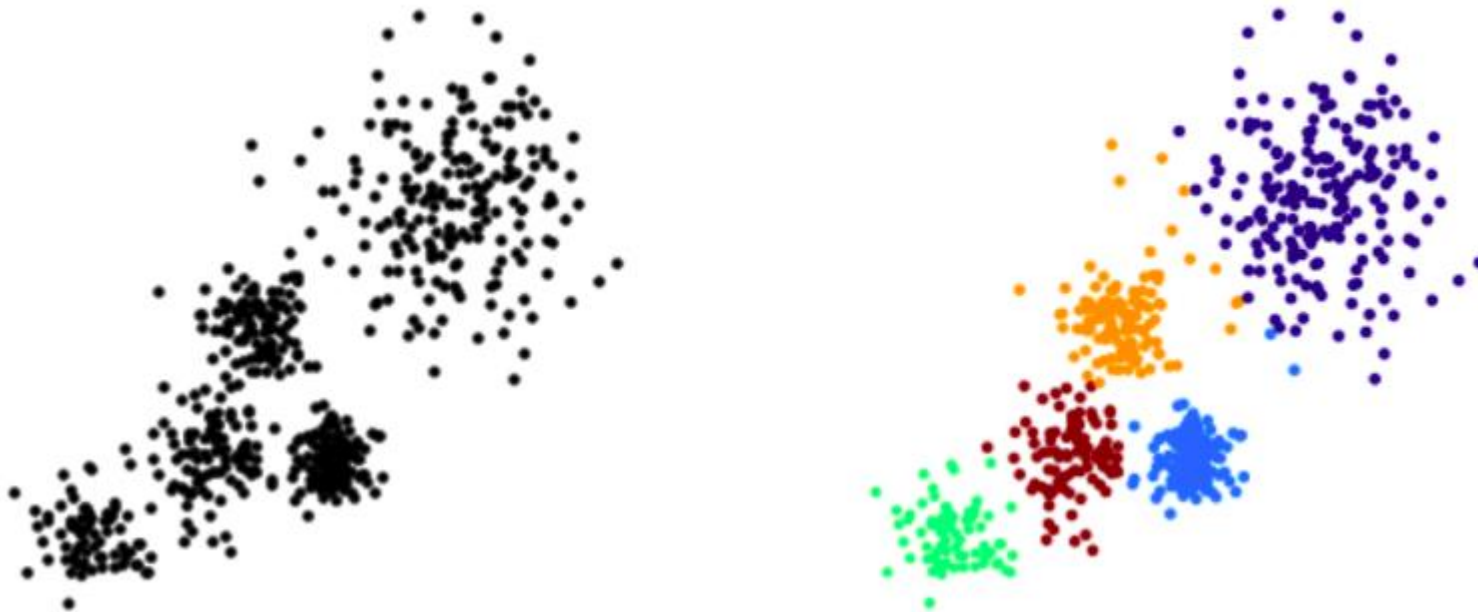
- Clients de cluster basés sur les fonctionnalités

- Segmentation du marché

- Identifier des groupes physiques similaires

## Spark Machine Learning – K-Means

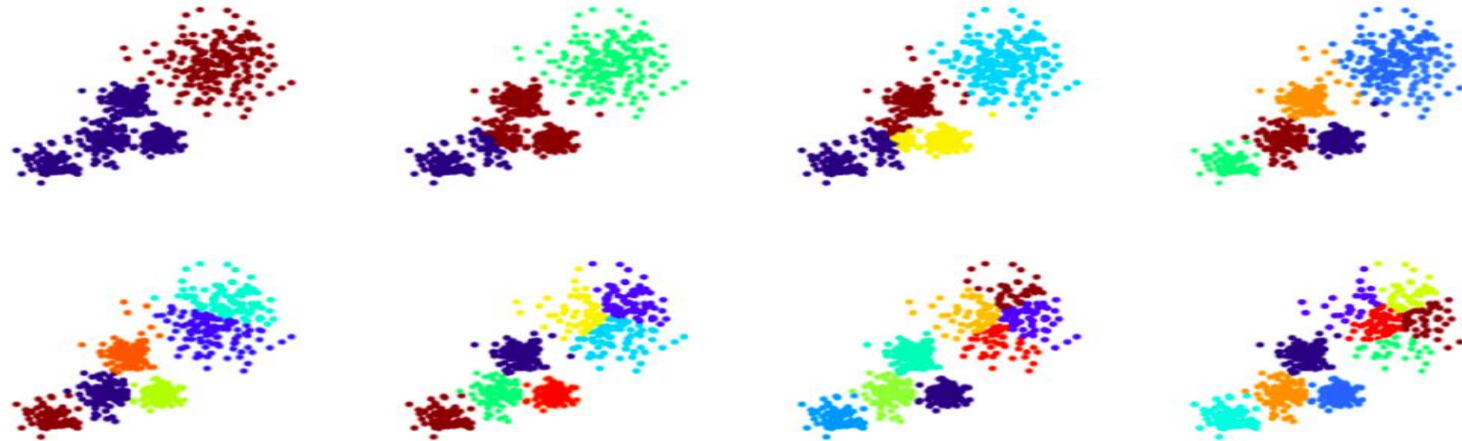
L'objectif global est de diviser les données en groupes distincts de sorte que les observations au sein de chaque groupe soient similaires :



## Spark Machine Learning – K-Means

L'algorithme K Means :

- Choisissez un nombre de groupes "K"
- Attribuez aléatoirement chaque point à un cluster
- Jusqu'à ce que les clusters cessent de changer, répétez les opérations suivantes:
  - Pour chaque grappe, calculez le centroïde du cluster en prenant le vecteur moyen des points dans le cluster
  - Affectez chaque point de données au cluster pour lequel le centroïde est le plus proche

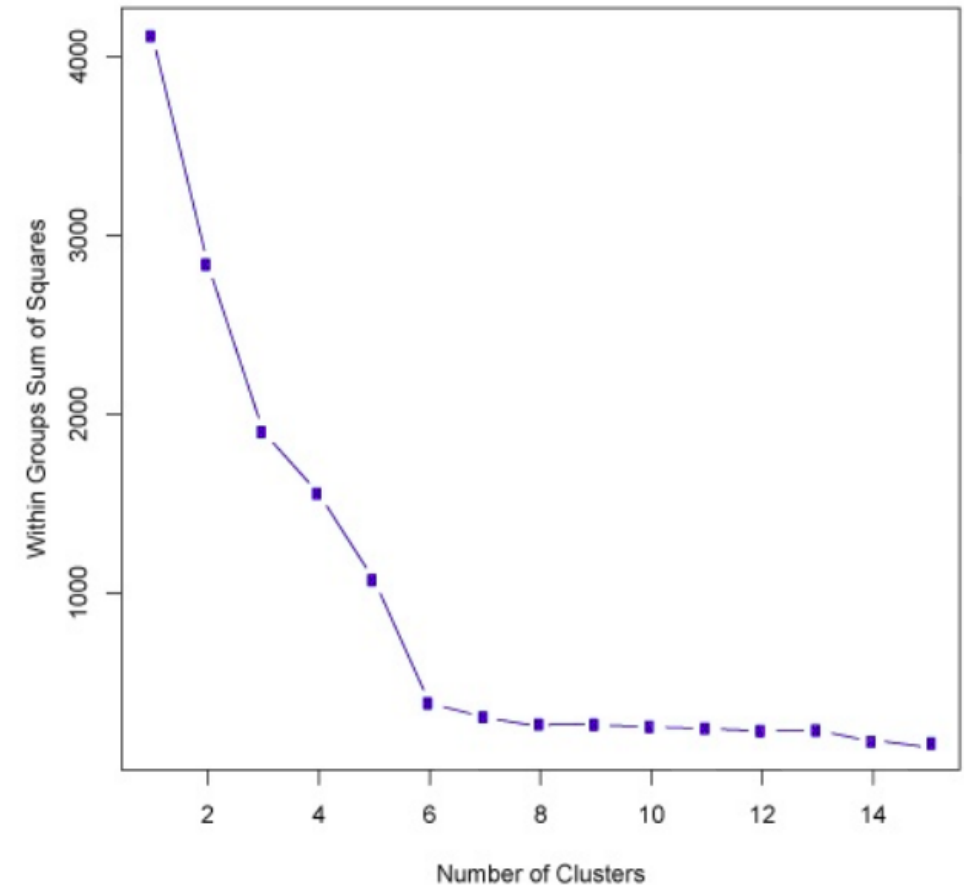


## Spark Machine Learning – K-Means

Tout d'abord, calculez la somme des erreurs au carré (SSE) pour certaines valeurs de K (par exemple 2, 4, 6, 8, etc.).

Le SSE est défini comme la somme de la distance au carré entre chaque membre de la grappe et son centroïde.

Si vous tracez K contre le SSE, vous verrez que l'erreur diminue à mesure que K augmente; c'est parce que quand le nombre de groupes augmente, ils devraient être plus petits, de sorte que la distorsion est également plus petite.



## Spark Machine Learning – K-Means

L'analyse de la silhouette peut être utilisée pour déterminer le degré de séparation entre les clusters. Pour chaque échantillon:

- Calculez la distance moyenne de tous les points de données dans le même cluster ( $a_i$ ).
- Calculez la distance moyenne de tous les points de données dans le cluster le plus proche ( $b_i$ ).
- Calculez le coefficient:

$$\frac{b^i - a^i}{\max(a^i, b^i)}$$

Le coefficient peut prendre des valeurs dans l'intervalle  $[-1, 1]$ .

- Si c'est 0 –> l'échantillon est très proche des grappes voisines.
- Il est 1 –> l'échantillon est loin des grappes voisines.
- Il est -1-> l'échantillon est assigné aux clusters erronés.

Par conséquent, nous voulons que les coefficients soient aussi grands que possible et près de 1



## Spark Machine Learning – K-Means

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('kmeans').getOrCreate()
from pyspark.ml.clustering import Kmeans
dataset = spark.read.format('libsvm').load('sample_kmeans_data.txt')
final_data = dataset.select('features')
final_data.show()
kmeans = Kmeans().setK(2).setSeed(1)
model = kmeans.fit(final_data)
wssse = model.computeCost(final_data)
print(wssse)
centers = model.clusterCenters()
centers
res = model.transform(final_data)
res.show()
```

## Spark Machine Learning – K-Means

Exercice :

Notebook: Clustering Code Clustering\_Exo.ipynb



## Spark Machine Learning – K-Means

Exercice :

Nous allons travailler sur un ensemble de données réelles contenant des données sur trois types de semences distincts.

Notebook: Clustering Code Clustering\_Exo.ipynb



### Natural Language Processing :

C'est le domaine de l'apprentissage automatique qui se concentre sur la création de modèles à partir d'une source de données de texte (directement à partir d'articles de mots).

### Exemples de NLP :

- Clustering des articles de presse/documents juridiques
- Suggérer des livres similaires
- Analyser les commentaires des consommateurs
- Analyser les posts sur tweeter/facebook
- Détecter les courriers indésirables

### Natural Language Processing :

Le process pour faire du NLP sera le suivant :

- Compiler tous les documents (Corpus)
- Convertir les mots en chiffres (caractéristiques)
- Comparer les caractéristiques des documents

Une façon standard de le faire est l'utilisation de ce que l'on appelle les méthodes "TF-IDF\*".

\* : TF-IDF est l'abréviation de 'term frequency-inverse document frequency'

Le TF-IDF est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes. Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus.

Le processus :

Exemple simple :

Vous avez 2 documents:

"Maison bleue"

"Maison Rouge"

Caractériser en fonction du nombre de mots:

"Maison Bleue" -> (rouge, bleu, maison) -> (0,1,1)

"Maison Rouge" -> (rouge, bleu, maison) -> (1,0,1)

Un document représenté comme un vecteur de comptage de mots est appelé un "sac de mots"

Ce sont maintenant des vecteurs dans un espace à N dimensions, on peut comparer des vecteurs avec une similarité de cosinus:

## Spark Machine Learning – NLP

Fréquence des mots : Importance du terme dans ce document

$TF(x, y) = \text{Nombre d'occurrences du terme } x \text{ dans le document } y / \text{Nombre total des termes dans le document}$

Fréquence des documents inversés : Importance du terme dans le corpus

$IDF(t) = \log(N / df_x)$  où

$N$  = nombre total des documents

$df_x$  = nombre des documents contenant le terme  $x$

Voir exemple

## Spark Machine Learning – NLP

TF-IDF peut aussi servir dans la détermination des mots clés qui devraient être idéalement utilisés dans le contenu d'un site web.

Exemple concret de calcul du score TF-IDF

Supposons que vous souhaitiez écrire un post sur la calligraphie. Vous savez que si vous y placez les mots importants, ca vous aidera à mieux vous positionner sur les moteurs de recherche. Vous écrivez donc un **post de 1000 mots**, contenant **20 fois le mot “écriture”** et **15 fois le mot “lettre”**. Nous allons prédire le poids de votre page sur les mots “écriture” et “lettre” en nous basant sur le score TF-IDF. Cela permettra de savoir si vous devez l'inclure plus de fois ou pas sur la page. Supposons donc que nous avons **150 pages sur 200 qui incluent le mot “écriture”** et **seulement 20 sur 200 incluant le mot “lettre”**.

**Calcul du TF-IDF pour le mot “écriture”**

**Calcul du TF-IDF pour le mot “lettre”**



## Spark Machine Learning – NLP

Le mot “écriture” a une plus grande valeur de TF-IDF comparé au mot “lettre” dans notre exemple. En comprenant l’importance relative des expressions au sein d’un document ou d’un corpus de document, vous pourrez adapter le contenu de vos post afin d’améliorer la pertinence d’un point de vue purement sémantique.

Exercice :

Travaillons à la construction d'un filtre de détection de spam en utilisant Python et Spark!

Notre ensemble de données se compose de messages texte volontaires provenant d'une étude à Singapour et de certains textes de spam provenant d'un site de reporting britannique.

Documents PY :

NLP

NLP\_EXO

