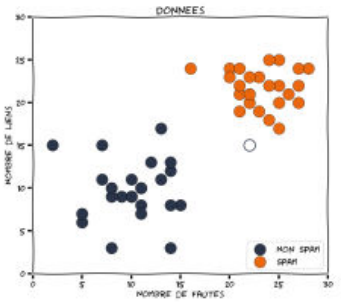
KNN

Qu'est-ce que KNN ?

- **KNN** est un type d'algorithme de ML supervisé qui peut être utilisé à la fois pour les problèmes prédictifs de classification et de régression. Cependant, elle est principalement utilisée pour **des problèmes prédictifs de classification** dans l'industrie.
- Il s'agit d'un algorithme d'apprentissage paresseux (**Lazy Learning Algorithm**) car il n'apprend pas immédiatement de l'ensemble de formation, mais il stocke l'ensemble de données et au moment de la classification, il effectue une action sur l'ensemble de données.

Modèle du Nearest Neighbors

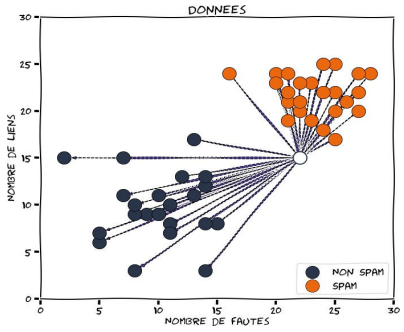
L'algorithme du *Nearest Neighbors* est particulier, car il nécessite de garder en mémoire tous les points du jeu de données pour calculer la distance entre ces points et un nouveau point dont on souhaite faire une prédiction.



Par exemple, voici un point dont on veut connaître la classe.

Modèle du Nearest Neighbors

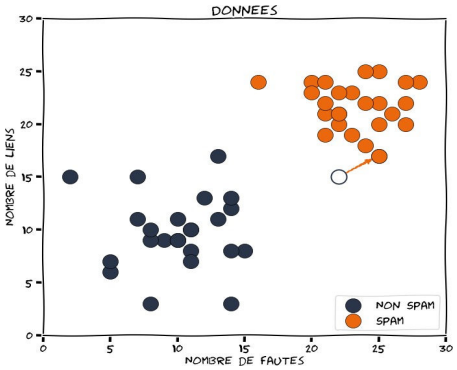
- Pour cela, on calcule la distance entre ce point et tous les autres points du jeu de données.



- Il existe plusieurs méthodes pour calculer la distance entre deux points, mais bien souvent, on utilise la distance euclidienne.

Modèle du Nearest Neighbors

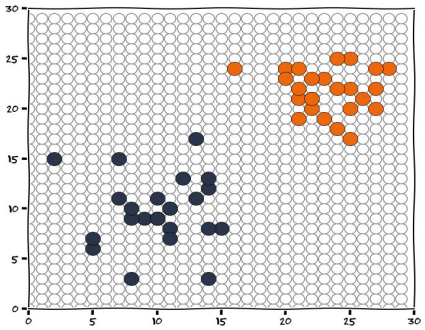
- En classant chaque distance de la plus petite à la plus grande, on identifie le point du jeu de données le plus proche.



- Dès lors, on attribue notre point à la même classe que son plus proche voisin.

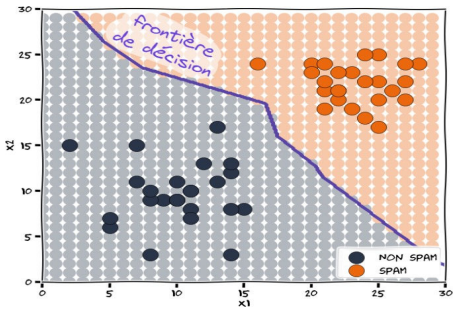
Modèle du Nearest Neighbors

- Nous pouvons généraliser cette approche en appliquant cette opération à tous les points de notre espace...



Modèle du Nearest Neighbors

Ce qui permet de mettre en avant la frontière de décision, c'est-à-dire la frontière qui sépare les deux classes.



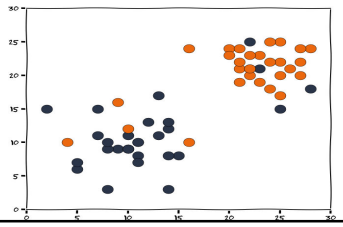
Et voilà, vous savez désormais comment fonctionne l'algorithme du plus proche voisin.

Modèle du Nearest Neighbors

Malgré sa simplicité, l'algorithme KNN est encore largement utilisé aujourd'hui dans de nombreuses applications, y compris chez Spotify, Netflix, Amazon, et bien d'autres.

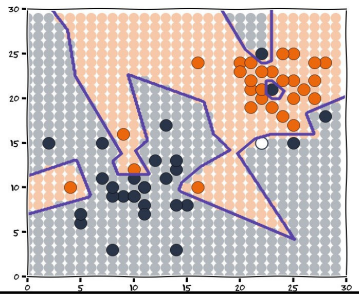
Généralisation: Le modèle du K-Nearest Neighbors

- Dans sa forme la plus simple, l'algorithme du plus proche voisin, dans sa version à 1 voisin, n'est pas très robuste.
- En effet, voyons ce qu'il se produit lorsqu'on fait face à un jeu de données un peu plus complexe. Ci-dessous, quelques valeurs aberrantes se sont glissées dans chaque classes. Cependant, à vue d'œil, cela ne change pas vraiment la tendance globale qui suggère que les emails spams sont situés en haut à droite, et les non-spams en bas à gauche.



Généralisation: Le modèle du K-Nearest Neighbors

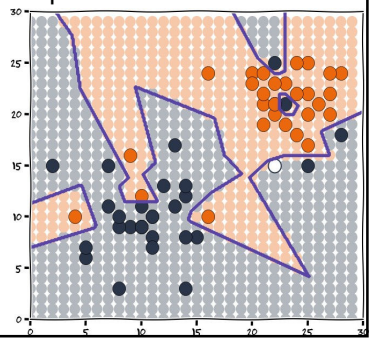
Malheureusement, la frontière de décision obtenue en suivant l'approche du plus proche voisin donne un résultat chaotique, qui ne correspond plus du tout à la tendance globale que nous avions observée précédemment. L'email de départ dont nous voulions connaître la classe est maintenant classé parmi les emails non-spams bien qu'il semble plus proche du groupe des spams.



Généralisation: Le modèle du K-Nearest Neighbors

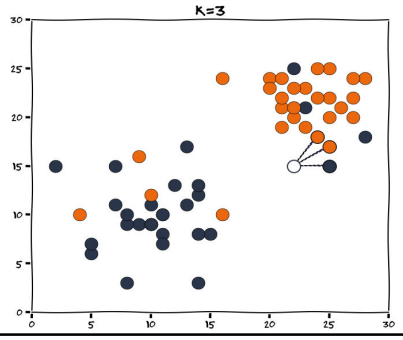
- Malheureusement, la frontière de décision obtenue en suivant l'approche du plus proche voisin donne un résultat chaotique, qui ne correspond plus du tout à la tendance globale que nous avions observée précédemment. L'email de départ dont nous voulions connaître la classe est maintenant classé parmi les emails non-spams bien qu'il semble plus proche du groupe des spams.

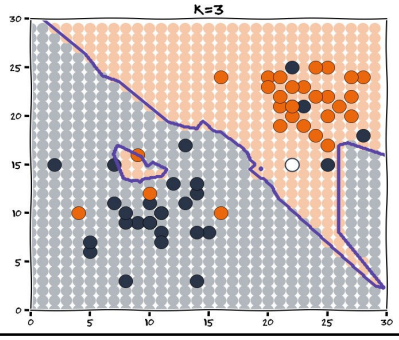
Pour éviter ce problème, on utilise généralement les k-voisins plus proches, pour ensuite assimiler notre point à la classe majoritaire parmi ces k-voisins.



Généralisation: Le modèle du K-Nearest Neighbors

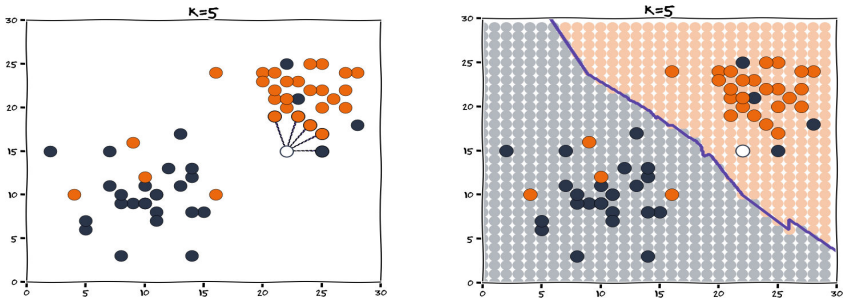
- Voici un exemple avec k=3. Ici, notre point compte parmi ses 3 voisins les plus proches, 2 points de la classe Orange et 1 point de la classe Gris foncé. Il est donc assimilé à la classe Orange. La frontière de décision est ainsi plus cohérente que précédemment.





Généralisation: Le modèle du K-Nearest Neighbors

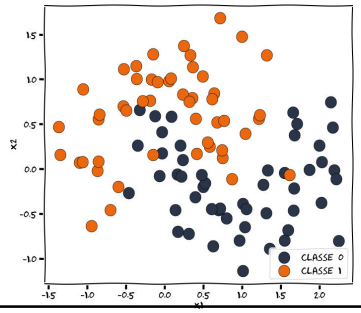
- Et voici un exemple avec K=5.



Comme on peut le constater, plus le nombre de voisins augmente, moins la frontière de décision est sensible aux valeurs aberrantes.

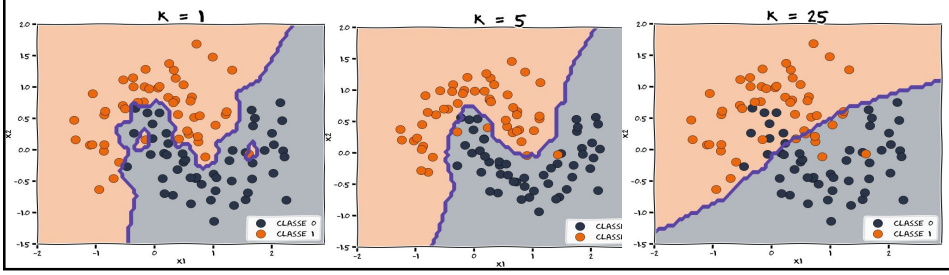
Généralisation: Le modèle du K-Nearest Neighbors

- Cependant, il faut rester vigilant à ne pas trop augmenter le nombre de voisins, au risque de trop lisser la frontière de décision. L'exemple ci-dessous montre un jeu de données dans lequel les deux classes prennent une forme de croissant, avec une légère superposition de certains points



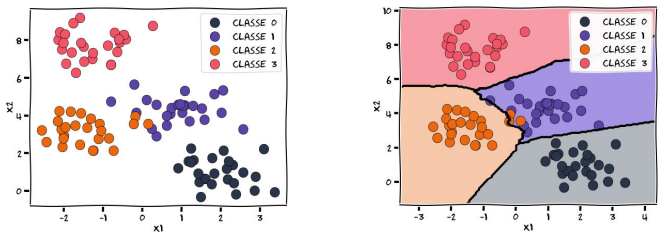
Généralisation: Le modèle du K-Nearest Neighbors

- Ainsi, la situation k = 1 n'est pas idéale, car elle est trop sensible aux valeurs aberrantes, comme nous l'avons vu précédemment. Mais une valeur de k trop élevée à pour effet de tracer une frontière de décision qui ne correspond plus du tout à l'allure générale de notre jeu de données. Il convient donc de trouver le juste milieu.



KNN: Classification multiple

- Les problèmes de classification ne se limitent pas à la classification binaire. En ML, il est possible de travailler avec autant de classes que l'on souhaite. Voici par exemple un jeu de données comprenant quatre classes.



L'algorithme des k-voisins les plus proches fonctionne parfaitement dans ce genre de situation, et nous n'avons rien à modifier dans notre algorithme pour obtenir ces résultats.

Points forts / Points faibles de KNN

Points forts de KNN

- Très simple et intuitif.
- Peut être appliqué aux données de n'importe quelle distribution.
- Bonne classification si le nombre d'échantillons est suffisamment grand.

Points faibles de KNN

- Prend plus de temps pour classer un nouvel exemple : il faut calculer et comparer la distance entre le nouvel exemple et tous les autres exemples.
- Choisir k peut être délicat.
- Besoin d'un grand nombre d'échantillons pour plus de précision.

Comment évaluer les performances d'un algorithme de classification?

L'évaluation de la performance

- L'évaluation de la performance d'un modèle de classification est une étape très importante car elle montre à quel point le classifieur est bon.
- Dans la littérature scientifique, nous trouvons que le plus grand critère utilisé par les chercheurs pour évaluer les classifieurs est le taux de bonne classification (les exemples correctement classés dans la phase de test) ou le taux d'erreur.
- Cependant, la seule utilisation de ce paramètre pour l'évaluation est très insuffisante. Dans la section suivante, nous présenterons un aperçu des différents critères d'évaluation .

Erreur de généralisation

Le taux d'erreur de généralisation (TE) ou de classification (TC) (TE = 100-TC) est la technique la plus utilisée pour évaluer les performances des classifieurs dans lesquelles les classes données par le classifieur sont comparées aux vraies étiquettes de l'ensemble de test. Le TC est un paramètre simple qui est calculé comme suit :

TC = Nombre d'échantillon bien classés / Nombre total d'exemples

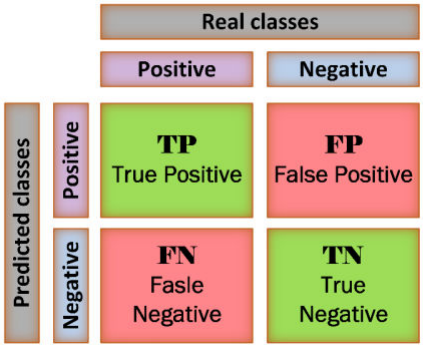
Erreur de généralisation

TC = Nombre d'échantillon bien classés / Nombre total d'exemples

- Si l'on considère par exemple une base de données avec 1000 patients dont seulement 10 sont malades, et que notre classifieur décide que tous les 1000 patients étaient normaux pendant la phase de test, la valeur du TC est égale à 99%.
- Pour un classifieur, les taux d'erreur peuvent signifier que le système fonctionne très bien, ce qui est loin d'être le cas pour l'exemple ci-dessus. Cela signifie que le TC n'est pas suffisant pour évaluer notre système. Ainsi, nous sommes obligés d'ajouter d'autres paramètres pour évaluer les algorithmes proposés

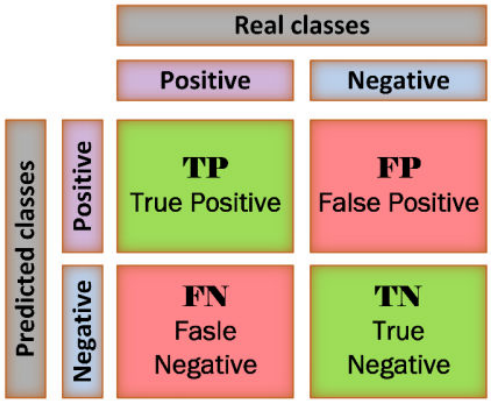
Matrice de confusion

- La matrice de confusion relie les décisions du classifieur et les étiquettes des échantillons.
- C'est un outil pour mesurer la qualité d'un système de classification. Comme le montre la figure ci-dessous, sur la diagonale de la matrice de confusion, nous trouvons les exemples bien classés, les autres sont mal classés.



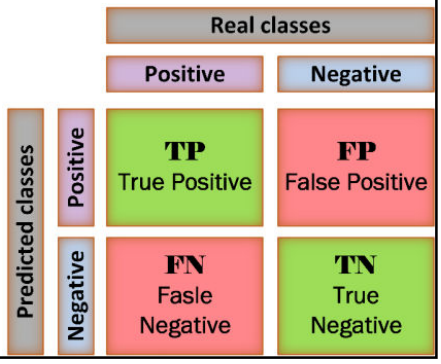
Matrice de confusion

- La matrice est un paramètre d'évaluation qui prend soin de la bonne classification et de la distribution des différentes classes.



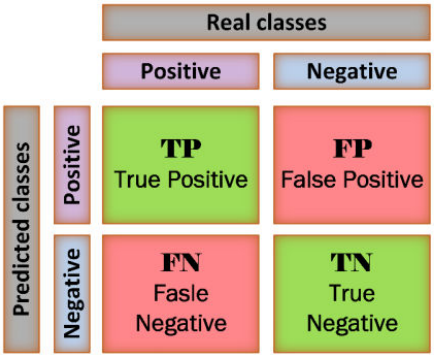
Matrice de confusion

- **True Positive (TP):** Nombre de cas positifs correctement prédits.
- **True Negative (TN):** Nombre de cas négatifs correctement prédits.
- **False Positive (FP):** Nombre de cas négatifs incorrectement prédits comme positifs (faux positif).
- **False Negative (FN):** Nombre de cas positifs incorrectement prédits comme négatifs (faux négatif).



Matrice de confusion

```
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(actual, predicted)
```



Exemple: Matrice de confusion

- Le modèle utilise quatre variables pour faire une prédiction **(Input)** :
  - **Heure (X1) :** L'heure de la transaction ou de l'événement.
  - **Montant (X2) :** Une valeur numérique, probablement le montant d'une transaction.
  - **Long (X3) :** Une longitude.
  - **Lat (X4) :** Une latitude.

Input				Output	
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction
12:20	4500	-1.2	3.2	0	0
01:45	1,60	0.6	4.3	1	1
10:08	100	-2.45	1.3	0	1
11:55	80	-1.2	3.2	1	0
00:00	3200	-1	3	0	0
				0	0
				1	1

- Le résultat de la classification **(Output)** est comparé à la réalité :
  - **Réel :** La valeur réelle (0 ou 1) de la classe.
  - **Prédiction :** La valeur prédite (0 ou 1) par le modèle

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output	
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction
12:20	4500	-1.2	3.2	0	0
01:45	1,60	0.6	4.3	1	1
10:08	100	-2.45	1.3	0	1
11:55	80	-1.2	3.2	1	0
00:00	3200	-1	3	0	0
				0	0
				1	1

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	
10:08	100	-2.45	1.3	0	1	
11:55	80	-1.2	3.2	1	0	
00:00	3200	-1	3	0	0	
				0	0	
				1	1	

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	
11:55	80	-1.2	3.2	1	0	
00:00	3200	-1	3	0	0	
				0	0	
				1	1	

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	
00:00	3200	-1	3	0	0	
				0	0	
				1	1	

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	
				0	0	
				1	1	

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	
				1	1	

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	

Exemple: Matrice de confusion

Comparons les outputs du modèle avec la réalité:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

Exemple: Matrice de confusion

On peut maintenant construire la matrice de confusion qui va contenir le résumé de cette comparaison

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

Matrice de confusion :

		Actual Values	
		Positive (1)	Negative (0)
Predicted	Positive (1)	(TP)	(FP)
	Negative (0)	(FN)	(TN)

Exemple: Matrice de confusion

On peut maintenant construire la matrice de confusion qui va contenir le résumé de cette comparaison

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

Matrice de confusion :

		Actual Values	
		Positive (1)	Negative (0)
Predicted	Positive (1)	2 (TP)	(FP)
	Negative (0)	(FN)	(TN)

Exemple: Matrice de confusion

On peut maintenant construire la matrice de confusion qui va contenir le résumé de cette comparaison

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

Matrice de confusion :

Predicted \ Actual Values	Positive (1)	Negative (0)
Positive (1)	2 (TP)	1 (FP)
Negative (0)	(FN)	(TN)

Exemple: Matrice de confusion

On peut maintenant construire la matrice de confusion qui va contenir le résumé de cette comparaison

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

Matrice de confusion :

Predicted \ Actual Values	Positive (1)	Negative (0)
Positive (1)	2 (TP)	1 (FP)
Negative (0)	1 (FN)	(TN)

Exemple: Matrice de confusion

On peut maintenant construire la matrice de confusion qui va contenir le résumé de cette comparaison

	Input				Output		
	Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
	12:20	4500	-1.2	3.2	0	0	TN
	01:45	1,60	0.6	4.3	1	1	TP
	10:08	100	-2.45	1.3	0	1	FP
	11:55	80	-1.2	3.2	1	0	FN
	00:00	3200	-1	3	0	0	TN
					0	0	TN
					1	1	TP

Matrice de confusion :

		Actual Values	
		Positive (1)	Negative (0)
Predicted	Positive (1)	2 (TP)	1 (FP)
	Negative (0)	1 (FN)	3 (TN)

Matrice de confusion

- La matrice de confusion est utilisée pour mesurer et évaluer les performances d'un algorithme de classification.
- Elle permet de calculer les métriques suivantes :
  - **Accuracy (Exactitude)**
  - **Precision (Précision)**
  - ***Recall (Rappel)***
  - ***F1 Score***

Matrice de confusion

1. L' accuracy:

L' accuracy, appelé exactitude ou justesse ou taux de réussite ou taux de bonne bonne prédiction en français , est une métrique fréquemment utilisée pour évaluer les modèles de classification car :

- elle est facile à calculer et à interpréter, c' est la proportion d'individus correctement prédits.
- elle résume la performance du modèle avec une valeur unique.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

		Réalité	
		Negative : 0	Positive : 1
Prédiction	Negative : 0	True Negative : TN	False Negative : FN
	Positive : 1	False Positive : FP	True Positive : TP

Pour calculer l'accuracy d'un modèle avec scikit-learn, il suffit d'utiliser la fonction **accuracy_score**

Matrice de confusion

2. rappel, recall ou sensibilité

- Le rappel ("recall" en anglais), ou sensibilité ("sensitivity" en anglais), est le taux de vrais positifs, c' est à dire la proportion de positifs que l' on a correctement prédites parmi toutes les observations réellement positives.
- Il mesure la capacité du modèle à détecter l' ensemble des individus positifs.

$$Rappel = \frac{TP}{TP + FN}$$

		Réalité	
		Negative : 0	Positive : 1
Prédiction	Negative : 0	True Negative : TN	False Negative : FN
	Positive : 1	False Positive : FP	True Positive : TP

Avec notre exemple sur les transactions frauduleuses, c' est la capacité de notre modèle à détecter toutes les transactions frauduleuses.

Un rappel élevé indique que le modèle est capable de détecter la plupart des exemples positifs

Matrice de confusion

3. Précision (precision)

- La précision est la proportion de predictions correctes parmi les points que l'on a prédits positifs.
- Proportion de vrais positifs parmi toutes les prédictions positives, reflétant la **validité** des prédictions.

$$Précision = \frac{TP}{TP + FP}$$

Prédiction \ Réalité	Negative : 0	Positive : 1
Negative : 0	True Negative : TN	False Negative : FN
Positive : 1	False Positive : FP	True Positive : TP

Avec notre exemple, c' est la capacité de notre modèle à ne déclencher une notification que pour une vraie transaction frauduleuse.

une précision élevée indique que le modèle fait peu de prédictions fausses pour la classe positive, ce qui est souhaitable dans de nombreuses applications

Matrice de confusion

4. F1-score

- Le F1-score évalue la capacité d' un modèle de classification à prédire efficacement les individus positifs, en faisant un compromis entre la précision et le rappel.
- Il est particulièrement utile pour les problèmes utilisant des données déséquilibrées.
- Le F 1 - score mesure la moyenne harmonique de la précision et du recall.
- Le F1-score appartient à la famille plus large des F-beta scores.

$$\text{F1-score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

$$= 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Prédiction \ Réalité	Negative : 0	Positive : 1
Negative : 0	True Negative : TN	False Negative : FN
Positive : 1	False Positive : FP	True Positive : TP

Matrice de confusion

4. F1-score

$$\text{F1-score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

$$= 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

		Réalité	
		Negative : 0	Positive : 1
Prédiction	Negative : 0	True Negative : TN	False Negative : FN
	Positive : 1	False Positive : FP	True Positive : TP

- Le F1-score atteint sa meilleure valeur à 1 et sa pire valeur à 0.
- Un F1-score élevé indique à la fois une bonne précision et un bon rappel, ce qui signifie que le modèle a bien classé à la fois les exemples positifs et négatifs.

Remarque

La précision seule peut être trompeuse dans les cas où les classes sont déséquilibrées. Par exemple, dans un ensemble de données où la classe négative est dominante, un modèle peut obtenir une précision élevée simplement en prédisant constamment la classe majoritaire.

Alors Il est particulièrement utile pour les problèmes utilisant des données déséquilibrées de calculer plusieurs métriques afin de pouvoir évaluer bien les performances d'un modèle donné.

Remarque

La mesure et l'évaluation de la performance d'un modèle de classification se fait toujours sur l'échantillon de test. Il faut tester la performance de modèle sur des données qui n'ont pas été utilisées pour construire le modèle de classification.

Application

Calculer les métriques suivantes : **Accuracy (Exactitude),Precision (Précision), *Recall (Rappel)*** , ***F1 Score***

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

Application:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Rappel = \frac{TP}{TP + FN} \qquad Précision = \frac{TP}{TP + FP}$$

$$\text{F1-score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

$$= 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Application:

Input				Output		
Heure X1	Montant X2	Long X3	Lat X4	Réel	Prédiction	
12:20	4500	-1.2	3.2	0	0	TN
01:45	1,60	0.6	4.3	1	1	TP
10:08	100	-2.45	1.3	0	1	FP
11:55	80	-1.2	3.2	1	0	FN
00:00	3200	-1	3	0	0	TN
				0	0	TN
				1	1	TP

$$Accauracy = \frac{TP+TN}{TP+FP+FN+TN} = \frac{2+3}{2+1+1+3} = \frac{5}{7} = 0,71 = 71\%$$

$$Precision = \frac{TP}{TP+FP} = \frac{2}{2+1} = \frac{2}{3} = 0,66 = 66\%$$

$$Recall = \frac{TP}{TP+FN} = \frac{2}{2+1} = \frac{2}{3} = 0,66 = 66\%$$

$$\text{f1Score} = \frac{2 \cdot precision \cdot recall}{precision + recall} = 0,68 = 68\%$$

Apprentissage non supervisé

Apprentissage non supervisé

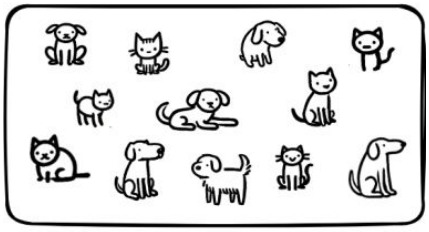
- L'apprentissage non supervisé est la seconde branche très connue du machine learning.
- Contrairement à L'apprentissage supervisé , celle-ci consiste à laisser la machine apprendre par elle-même certaines structures présentes dans les données, sans la contraindre à apprendre une simple relation d'entrée / sortie $X \rightarrow y$.

Quand envisager l'apprentissage non supervisé ?

- L'apprentissage non supervisé est utile lorsqu'on souhaite explorer des données mais que on n'a pas encore d'objectif spécifique ou qu'on ne sait pas quelles informations contiennent les données.
- C'est également un bon moyen de réduire les dimensions des données.

Exemple: Apprentissage non supervisé

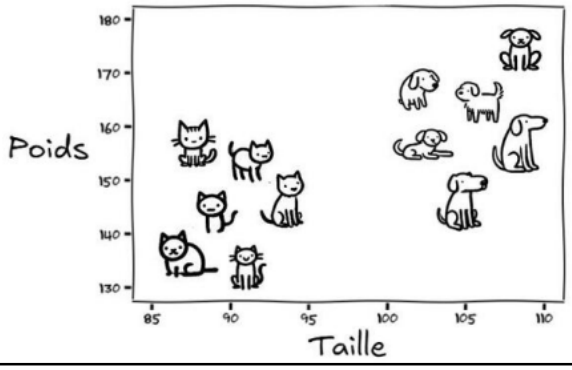
Nous pouvons présenter à la machine des animaux sans préciser s'il s'agit de chats ou de chiens,



et lui demander simplement de les regrouper selon leur ressemblance. Autrement dit, nous fournissons uniquement les attributs *X*, sans indiquer la sortie *y* attendue.

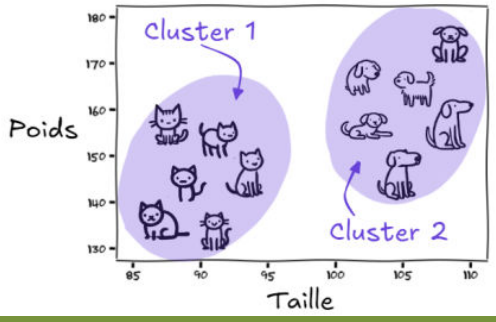
Exemple: Apprentissage non supervisé

En examinant simplement les attributs *X* (tels que la taille, le poids, l'apparence, etc) la machine remarquera d'elle-même que certains animaux se ressemblent par leurs caractéristiques.



Exemple: Apprentissage non supervisé

Ainsi, elle sera capable de les regrouper, sans pour autant savoir s'il s'agit de chats ou de chiens. En fait, elle n'en aura pas la moindre idée, car nous ne l'aurons pas contrainte à apprendre ce genre de choses.



Ce type d'application porte de le nom de **clustering**.

Clustering

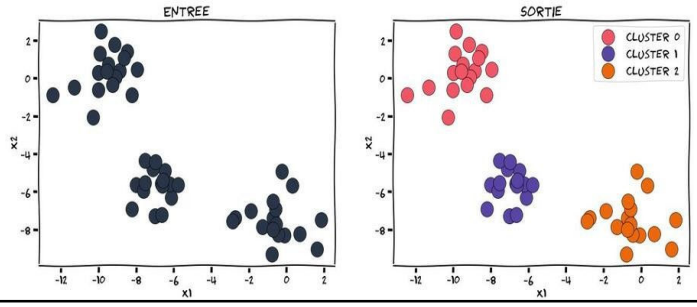
Clustering

- Le clustering est utilisé lorsque l'on souhaite classer nos données de manière non supervisée.
- le clustering consiste donc à regrouper nos données en nous basant uniquement sur leur ressemblance, sans se préoccuper d'une quelconque classe "correcte" ou "incorrecte".
- Le clustering est donc une méthode utilisé lorsqu'on souhaite laisser à la machine le pouvoir de proposer sa propre solution, et ainsi découvrir une approche différente de la nôtre.

Clustering

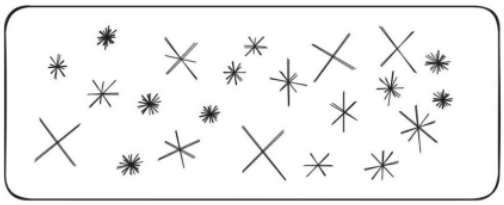
Par exemple :

- Classer des images ou des objets selon leur ressemblance
- Segmenter une base de données de clients, selon leurs habitudes de consommation
- Regrouper des documents selon leur contenu.
- etc.



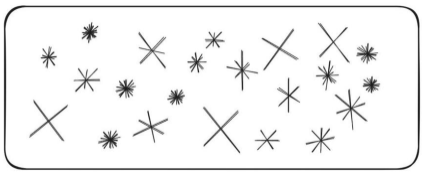
Clustering

Observez attentivement les formes ci-dessous. Elles ressemblent peut-être à des étoiles, mais ne représentent en réalité rien de concret.

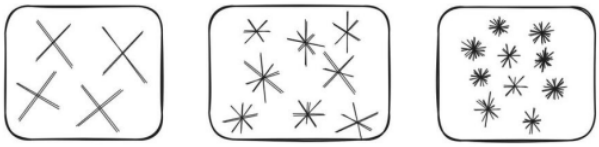


Clustering

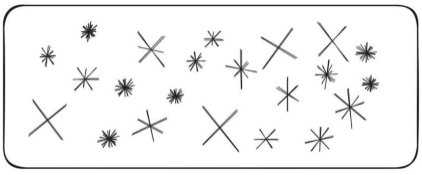
Observez attentivement les formes ci-dessous. Elles ressemblent peut-être à des étoiles, mais ne représentent en réalité rien de concret.



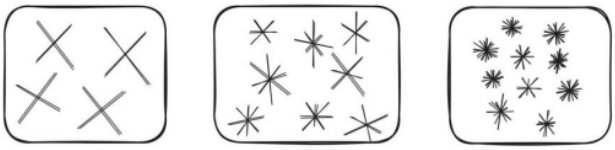
À présent, votre tâche consiste à regrouper ces formes selon leur ressemblance, afin de créer trois groupes distincts. La grande majorité des gens formeraient les trois groupes suivants :



Clustering

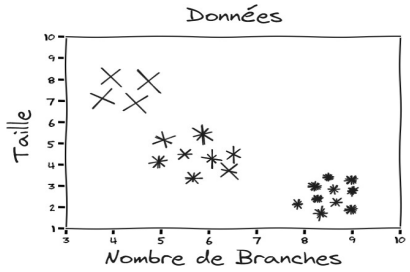


Les caractéristiques qui vous ont probablement permis de trier ces objets sont la taille et le nombre de branches de chaque forme.

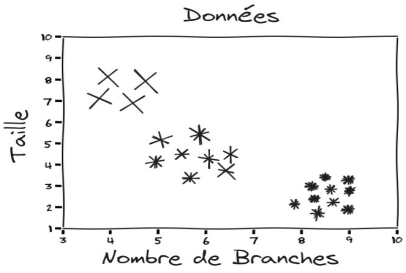


Clustering

Ainsi, en mesurant ces attributs (la taille et le nombre de branches) pour chaque objet et en les plaçant sur un graphique, on obtient le résultat suivant :



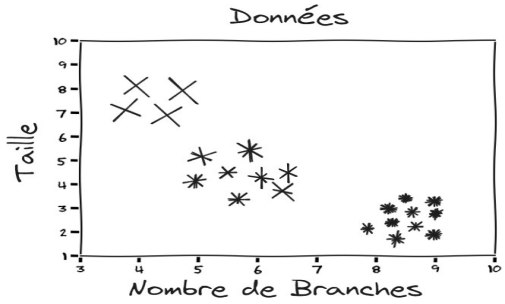
Clustering



Sur ce graphique, on distingue très facilement la présence de trois groupes.

Est-il alors nécessaire de superviser l'apprentissage de la machine en désignant des classes *y* pour que celle-ci puisse classer ces points?

Clustering



Sur ce graphique, on distingue très facilement la présence de trois groupes.

Est-il alors nécessaire de superviser l'apprentissage de la machine en désignant des classes *y* pour que celle-ci puisse classer ces points? **La réponse est non.**

Clustering

- Le principe du clustering est d'analyser les différentes variables *X* qui caractérisent nos données, afin de regrouper les points en clusters, sans pour autant connaître la nature de ces points.
- De nombreux algorithmes permettent de réaliser du clustering :
  - **Le K-Means Clustering**
  - Le Clustering hierarchique
  - DBSCAN
  - OPTICS
  - etc.

Fonctionnement du modèle de K-Means

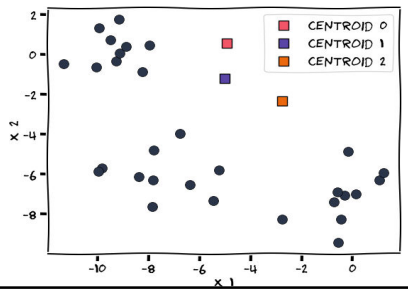
Le K-Means Clustering est un algorithme itératif, dans lequel on demande à notre machine de trouver un nombre K de clusters au sein de notre jeu de données.

Pour cela, elle place au hasard K points dans l'espace, puis déplace ces points pour qu'ils deviennent les barycentres de nos clusters.

Fonctionnement du modèle de K-Means Clustering

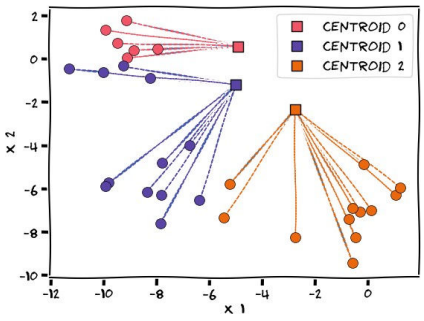
Prenons un exemple avec K=3 clusters.

- Pour commencer, on place trois points aléatoires au sein de nos données.
- Ces points sont appelés centroïdes et sont les futurs centres de masse des clusters que l'on veut former.



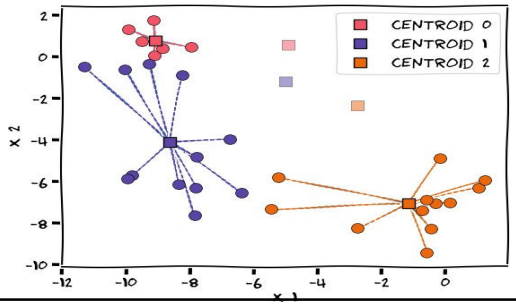
Fonctionnement du modèle de K-Means

- Ensuite, on associe chaque point de notre jeu de données au centroïde dont il est le plus proche, en calculant la distance euclidienne



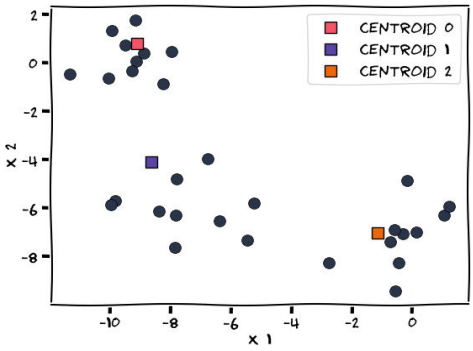
Fonctionnement du modèle de K-Means

- Ensuite, on déplace chaque centroïde au centre de son groupe, ce qui explique son nom de ”centroïde”. Pour ce faire, on calcule la moyenne des points du cluster, et cette moyenne depuis la nouvelle position du centroïde, d’où le nom de ”K-Means”.



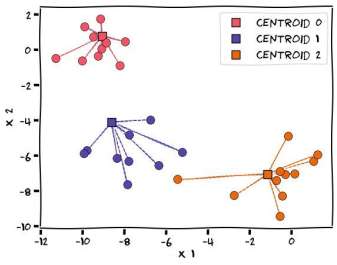
Fonctionnement du modèle de K-Means

Comme la position de chaque centroïde a changé, les centroïdes ont désormais des plus proches voisins différents. Ainsi, nous réitérons notre algorithme...



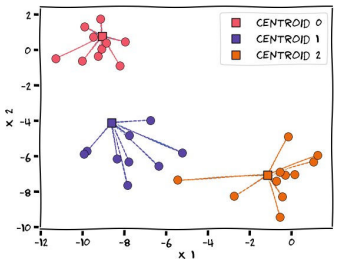
Fonctionnement du modèle de K-Means

- Alors, nous réassignons chaque point à son centroïde le plus proche (notez bien que cela produit des résultats différents de ceux de la première itération!).

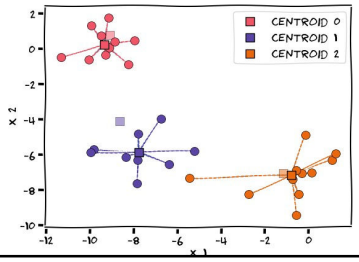


Fonctionnement du modèle de K-Means

- Alors, nous réassignons chaque point à son centroïde le plus proche (notez bien que cela produit des résultats différents de ceux de la première itération!).

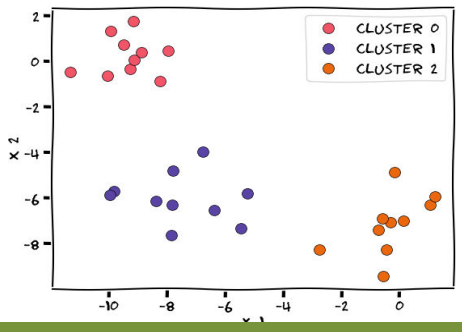


- Puis on déplace chaque centroïde au centre de son groupe.



Fonctionnement du modèle de K-Means

- Et cet algorithme se répète ainsi jusqu'à ce que les centroïdes ne bougent plus. C'est alors que l'algorithme se termine.

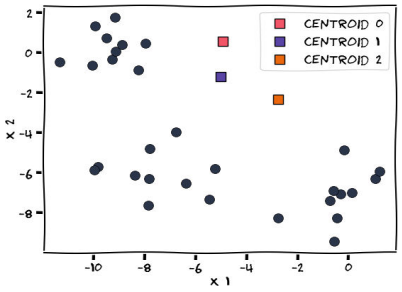


C'est ainsi que fonctionne l'algorithme du K-Means clustering. Il vous suffit de déclarer le nombre de clusters que vous souhaitez obtenir, et l'algorithme s'occupe du reste.

Fonctionnement du modèle de K-Means

Mais comment déterminer le nombre de centroïdes à utiliser dans un projet?

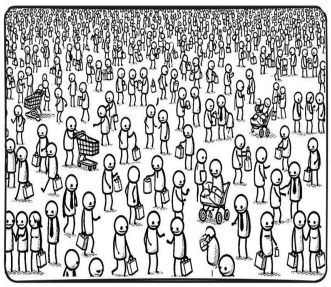
- Dans le cas présent, nous pouvions observer l'allure du jeu de données, et discerner visuellement trois groupes de points avant même l'entraînement du modèle.



Fonctionnement du modèle de K-Means

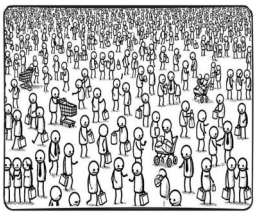
Mais comment déterminer le nombre de centroïdes à utiliser dans un projet?

- Dans la pratique, les choses sont souvent plus complexes. Imaginez devoir segmenter une bases de données clients, comprenant des centaines de variables *X*, tels que leurs habitudes de consommation, leur âge, leur localisation, leur genre, etc.



Fonctionnement du modèle de K-Means

Mais comment déterminer le nombre de centroïdes à utiliser dans un projet?



- Dans la pratique, les choses sont souvent plus complexes. Imaginez devoir segmenter une bases de données clients, comprenant des centaines de variables *X*, tels que leurs habitudes de consommation, leur âge, leur localisation, leur genre, etc.

  Dans ce cas, il n'est pas possible de visualiser les données pour déterminer le nombre optimal de clusters.

Heureusement, il existe une méthode permettant de trouver le nombre optimal de clusters sans avoir à observer directement notre jeu de données. **Cette méthode s'appelle la méthode du Coude.**