Machine Learning avec Spark

11 janvier 2023

Léo Géré

leo.gere@inrae.fr

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

INRAE

lipme

cnrs
dépasser les frontières

Plan de la présentation

I. Bases et principes du machine learning

- Objectif
- Évaluation d'un modèle

II. Exemples de modèles de machine learning

- Arbre de décision
- Forêt aléatoire
- SVM

III. Machine Learning avec Spark

- SparkML
- Transformer
- Estimator
- Algorithmes de machine learning
- Évaluation d'un modèle de machine learning

Plan de la présentation

I. Bases et principes du machine learning

- Objectif
- Évaluation d'un modèle

II. Exemples de modèles de machine learning

- Arbre de décision
- Forêt aléatoire
- SVM

III. Machine Learning avec Spark

- SparkML
- Transformer
- Estimator
- Algorithmes de machine learning
- Évaluation d'un modèle de machine learning

I. Bases et principes du machine learning

Objectif

- Machine Learning : apprentissage automatique
- Principe :
  - On a des données historiques sur un problème
  - On veut un modèle qui permet d'«apprendre» à partir de ces données
- Type d'apprentissage le plus classique : l'apprentissage supervisé
  - On a des données associées à un label à prédire, et on cherche à prédire ce label pour de nouvelles données
  - Problème de régression (prédire une valeur continue) VS problème de classification (prédire une classe parmi n)
- Autres types d'apprentissage :
  - Non-supervisé, auto-supervisé, par renforcement…

I. Bases et principes du machine learning

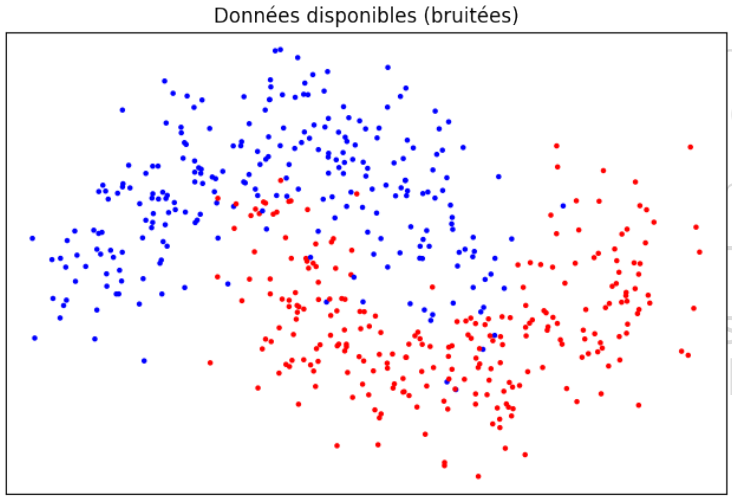
Objectif

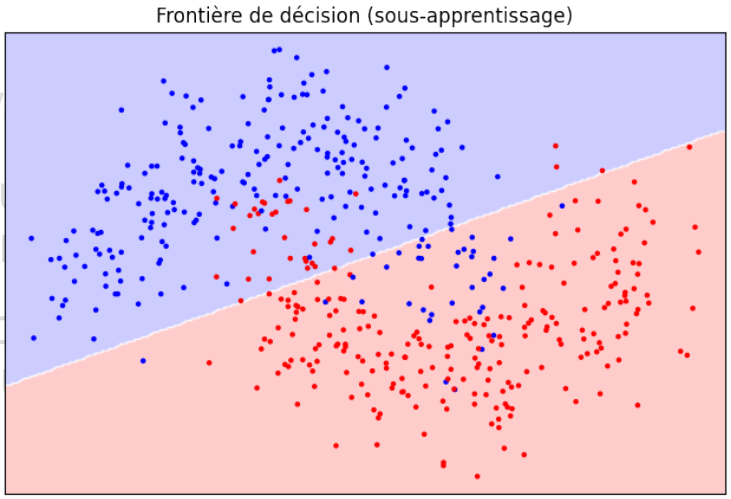
- On cherche une **bonne capacité de généralisation**
  - Capacité à prédire correctement le label de nouvelles données ne faisant pas partie des données d'entraînement
  - Un modèle qui a appris « par cœur » les données d'apprentissage est un mauvais modèle : **sur-apprentissage**
  - Attention également au **sous-apprentissage** : modèle pas assez complexe pour saisir toutes les subtilités des données étudiées

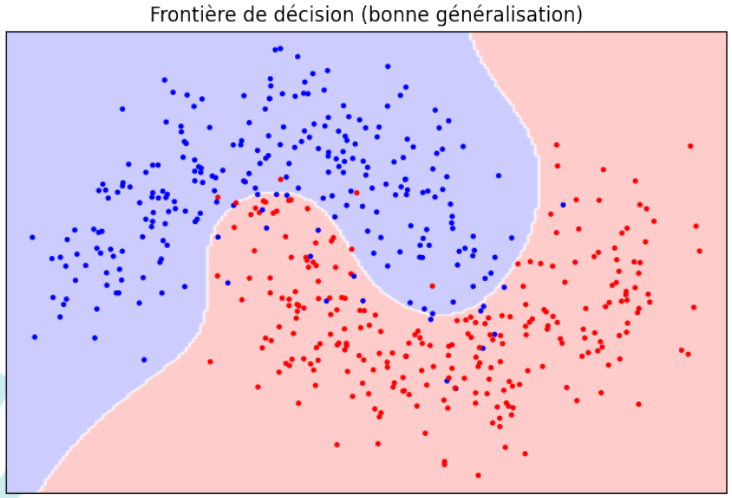
I. Bases et principes du machine learning

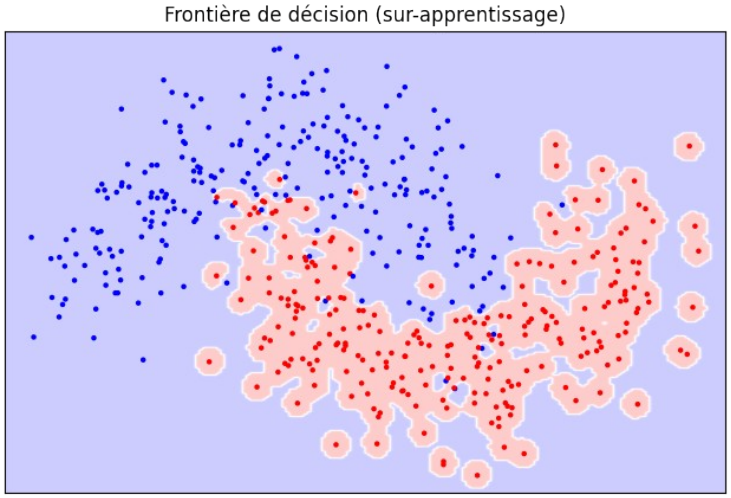
Objectif

- On cherche une **bonne capacité de généralisation**









I. Bases et principes du machine learning

Objectif

- On cherche une **bonne capacité de généralisation**
  - Capacité à prédire correctement le label de nouvelles données ne faisant pas partie des données d'entraînement
  - Un modèle qui a appris par cœur les données d'apprentissage est un mauvais modèle : **sur-apprentissage**
  - Attention également au **sous-apprentissage** : modèle pas assez complexe pour saisir toutes les subtilités des données étudiées
- On joue sur les **hyperparamètres** du modèle
  - **Paramètre ≠ hyperparamètre**
    - Paramètres : déterminés automatiquement à partir des données au cours de l'apprentissage
    - Hyperparamètres : définis par l'utilisateur, impacte l'apprentissage et les performances du modèle

Exemple : régression polynomiale
- Hyperparamètre : degré du polynôme
- Paramètres : coefficients du polynôme

I. Bases et principes du machine learning

Évaluation d'un modèle

- Nécessité d'évaluer le modèle avec de nouvelles données (contrôle du sur/sous-apprentissage)
  - On sépare les données disponibles en deux jeux de données : jeu d'entraînement et jeu de test (par exemple 80 % / 20 %)
  - On entraîne le modèle sur le jeu d'entraînement uniquement
  - On évalue ses capacités sur le jeu de test
    - Différentes métriques d'évaluation en fonction du problème étudié
- Classification binaire

		Réalité	
		1	0
Prédit	1	188 TP	4 FP
	0	12 FN	196 TN
		200	200

- **True positive** : vrai positif
  - Prédit correctement 1
- **True negative** : vrai négatif
  - Prédit correctement 0

} À maximiser

- **False positive** : faux positif
  - Prédit 1 à la place de 0
- **False negative** : faux negatif
  - Prédit 0 à la place de 1

} À minimiser

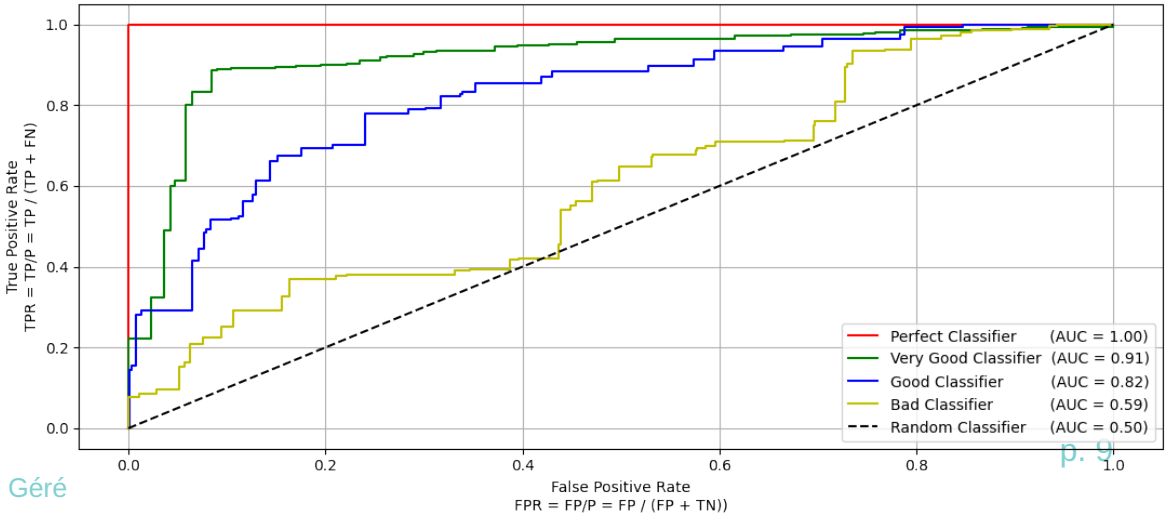
I. Bases et principes du machine learning

Évaluation d'un modèle

- Classification binaire
  - Accuracy : proportion de bien classés
  - Sensibilité: capacité à détecter le maximum de VP
  - Spécificité : capacité à détecter le maximum de VN
  - Aire sous la courbe ROC
    - Plus l'aire est proche de 1, meilleur est le classifieur

Source : https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers

		Predicted condition			
	Total population = P + N	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) = TPR + TNR − 1	Prevalence threshold (PT) = $\frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power = $\frac{TP}{P}$ = 1 − FNR	False negative rate (FNR), miss rate = $\frac{FN}{P}$ = 1 − TPR
Actual condition	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out = $\frac{FP}{N}$ = 1 − TNR	True negative rate (TNR), specificity (SPC), selectivity = $\frac{TN}{N}$ = 1 − FPR
	Prevalence = $\frac{P}{P+N}$	Positive predictive value (PPV), precision = $\frac{TP}{PP}$ = 1 − FDR	False omission rate (FOR) = $\frac{FN}{PN}$ = 1 − NPV	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$	Negative likelihood ratio (LR−) = $\frac{FNR}{TNR}$
	Accuracy (ACC) = $\frac{TP+TN}{P+N}$	False discovery rate (FDR) = $\frac{FP}{PP}$ = 1 − PPV	Negative predictive value (NPV) = $\frac{TN}{PN}$ = 1 − FOR	Markedness (MK), deltaP (Δp) = PPV + NPV − 1	Diagnostic odds ratio (DOR) = $\frac{LR+}{LR-}$
	Balanced accuracy (BA) = $\frac{TPR+TNR}{2}$	F_1 score = $\frac{2PPV \times TPR}{PPV+TPR}$ = $\frac{2TP}{2TP+FP+FN}$	Fowlkes–Mallows index (FM) = $\sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) = $\sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index = $\frac{TP}{TP+FN+FP}$



Plan de la présentation

I. Bases et principes du machine learning
- Objectif
- Évaluation d'un modèle

II. Exemples de modèles de machine learning
- Arbre de décision
- Forêt aléatoire
- SVM

III. Machine Learning avec Spark
- SparkML
- Transformer
- Estimator
- Algorithmes de machine learning
- Évaluation d'un modèle de machine learning

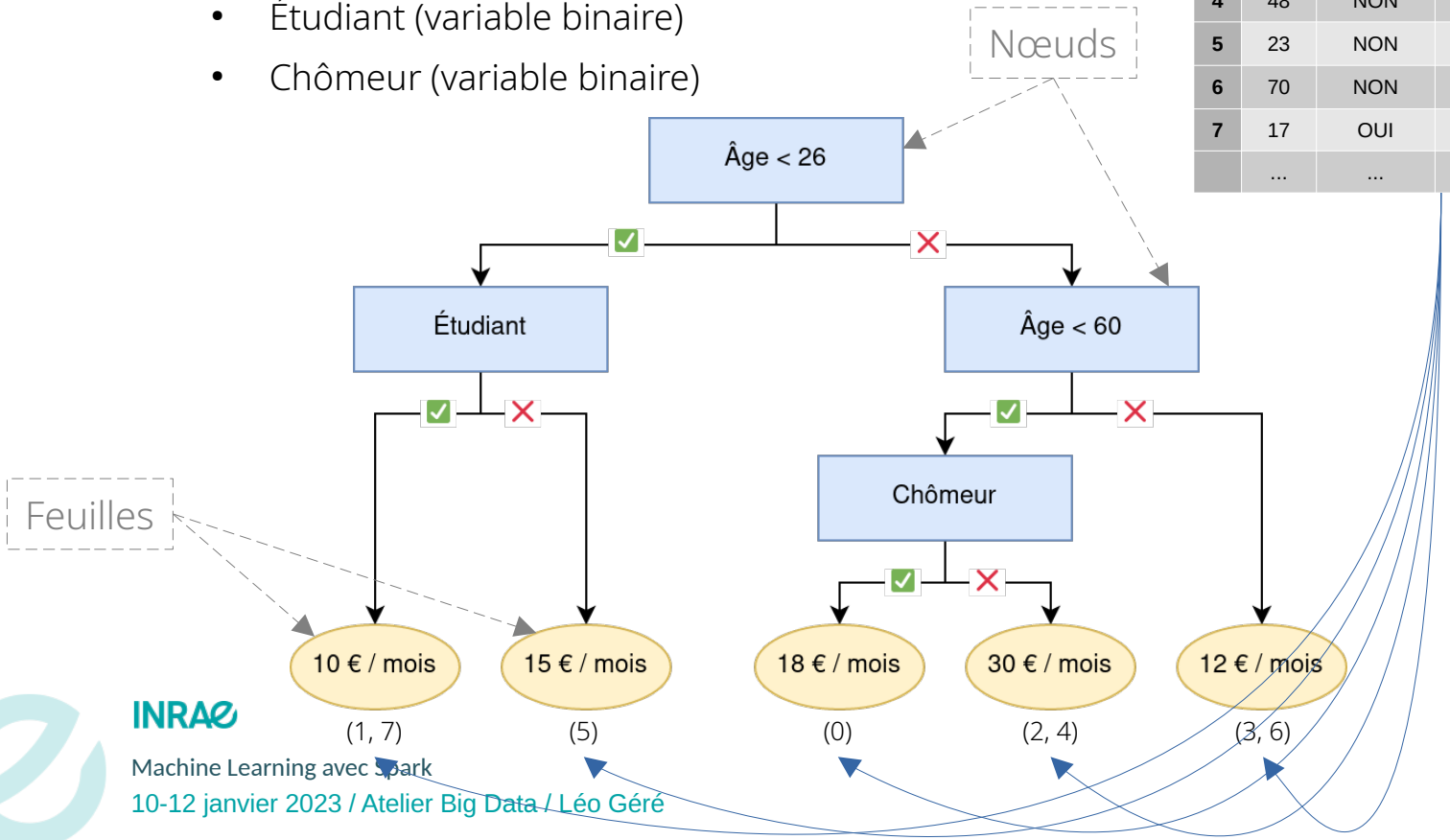
II. Exemples de modèles de machine learning

Arbre de décision

- Exemple : abonnement à un service
  - Variables disponibles :
    - Âge (variable numérique)
    - Étudiant (variable binaire)
    - Chômeur (variable binaire)

Données

	Âge	Étudiant	Chômeur	Tarif (€ / mois)
0	35	NON	OUI	18
1	21	OUI	NON	10
2	29	OUI	NON	30
3	63	NON	OUI	12
4	48	NON	NON	30
5	23	NON	OUI	15
6	70	NON	NON	12
7	17	OUI	NON	10
	...	...	...	...



II. Exemples de modèles de machine learning

Arbre de décision

- Construction automatique d'un arbre possible à partir de données d'entraînement
  - Dans le cas de classification binaire, on cherche parmi les variables disponibles la condition qui permet de discriminer au mieux nos données par rapport au label étudié, puis on réitère sur les deux branches crées
- Hyperparamètres du modèle
  - Profondeur maximale de l'arbre
  - Critère d'hétérogénéité (comment quantifier la qualité d'une séparation pendant la construction de l'arbre)
  - Nombre de feuilles maximum de l'arbre
  - Etc…

II. Exemples de modèles de machine learning

Forêt aléatoire

- Utilisation de plusieurs arbres de décision pour obtenir de meilleurs performances
  - Chaque arbre est entraîné avec un sous-ensemble des données et un sous-ensemble des variables tirées aléatoirement
  - La prédiction de la forêt correspond à la prédiction majoritaire de tous les arbres qui la composent

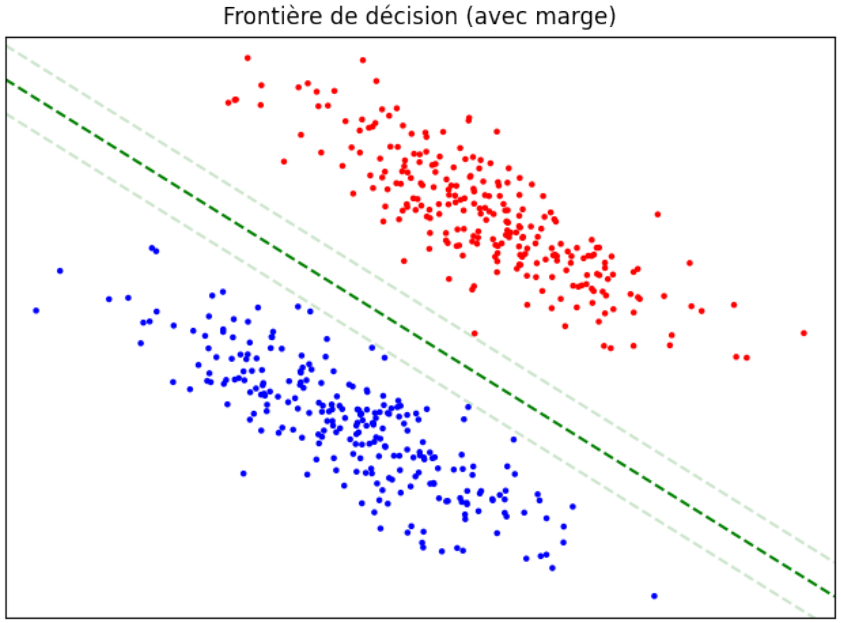
Note : en régression, on prend la moyenne des prédictions

- Hyperparamètres du modèle
  - Nombre d'arbres dans la forêt
  - Nombre de données et de variables utilisées pour chaque arbre
  - Tous les paramètres disponibles pour les arbres de décision

II. Exemples de modèles de machine learning

SVM

- Support Vector Machine (machine à vecteur de support, ou séparateur à vaste marge)
  - Principalement pour de la classification
- Idée : trouver un hyperplan qui sépare les données avec **la plus grande marge possible**



Quand pas de séparation linéaire possible, on utilise une « marge souple » qui permet de tolérer des erreurs.

Un hyperparamètre (C) permet de chercher un compromis entre le taux de mal-classés et la largeur de la marge

II. Exemples de modèles de machine learning

SVM

- Support Vector Machine (machine à vecteur de support, ou séparateur à vaste marge)
  - Principalement pour de la classification
- Idée : trouver un hyperplan qui sépare les données avec **la plus grande marge possible**
- Seconde idée : “projeter” les données dans un **espace de plus grande dimension** (SVM non-linéaires)
  - Dans un espace de plus grande dimension, il est plus probable que les données soient linéairement séparables
  - **Pas disponible dans Spark pour le moment** (difficile à paralléliser) → uniquement SVM linéaire

Plan de la présentation

I. Bases et principes du machine learning
- Objectif
- Évaluation d'un modèle

II. Exemples de modèles de machine learning
- Arbre de décision
- Forêt aléatoire
- SVM

III. Machine Learning avec Spark
- SparkML
- Transformer
- Estimator
- Algorithmes de machine learning
- Évaluation d'un modèle de machine learning

III. Machine Learning avec Spark

SparkML

- Librarie **SparkML** : algorithmes interfacés pour fonctionner directement avec les DataFrame de Spark
- Utilisation de **modules** qui vont permettrent de construire des **pipelines** pour traiter nos données
  - Parmi ces modules on retrouve de notamment des **algorithmes de machine learning**, mais également de nombreux utilitaires pour **pré-traiter les données** afin qu'elles aient le format requis par l'algorithme qui nous intéresse
  - Possibilité de créer des modules personnalisés
  - Deux types de modules :
    - Les "transformeurs" (**Transformer**)
    - Les estimateurs (**Estimator**)

III. Machine Learning avec Spark

Transformer

- Un **Transformer** est une simple opérateur qui prend un Dataframe en entrée, et donne un nouveau Dataframe en sortie
  - Généralement un Transformer se contente d'ajouter une ou plusieurs colonnes au Dataframe initial
  - Exemple : Tokenizer (permet de transformer une chaîne de caractères en une liste de mots)
- Utilisation avec Spark
  - On instancie le Transformer en spécifiant les colonnes d'entrée et sortie (+ éventuellement d'autres paramètres)
  - On l'applique à un Dataframe avec la méthode `.transform`

III. Machine Learning avec Spark

Transformer

- Exemple d'utilisation du Transformer Tokenizer

```scala
import org.apache.spark.ml.feature.Tokenizer

val tokenizer = new Tokenizer().setInputCol("myStringColumn").setOutputCol("tokenized")
val newDf = tokenizer.transform(df)
```

```python
from pyspark.ml.feature import Tokenizer

tokenizer = Tokenizer(inputCol="myStringColumn", outputCol="tokenized")
newDf = tokenizer.transform(df)
```

Dataframe initial

```
+--------------------------+
|myStringColumn            |
+--------------------------+
|Hello, here is some text.  |
|I've got something to say  |
|There are forty-two of them|
+--------------------------+
```

Dataframe final

```
+--------------------------+--------------------------------+
|myStringColumn            |tokenized                       |
+--------------------------+--------------------------------+
|Hello, here is some text.  |[hello,, here, is, some, text.]  |
|I've got something to say  |[i've, got, something, to, say]  |
|There are forty-two of them|[there, are, forty-two, of, them]|
+--------------------------+--------------------------------+
```

Note : Le **Tokenizer** sépare le texte à chaque espace, il reste donc la ponctuation. Pour éviter de la conserver, on peut soit la retirer avant, soit utiliser le **RegexTokenizer** avec un séparateur adéquat.

III. Machine Learning avec Spark

Estimator

- Un **Estimator** est un opérateur qui a besoin de faire deux passes sur les données : une première passe pour estimer certains paramètres, et une seconde pour effectuer la transformation
  - Après la première passe, notre Estimator n'est plus qu'un simple Transformer
  - En Spark, si un Estimator s'appelle `SomeName`, le Transformer associé après passage sur un Dataframe s'appelle généralement `SomeNameModel`
  - Exemple : `StandardScaler` (normalise une colonne numérique pour avoir $\bar{x} = 0$ et $\sigma = 1$)
    - Première passe sur les données pour estimer $\bar{x}$ et σ puis retourne un `StandardScalerModel` (qui est un Transformer car $\bar{x}$ et σ sont connus)
    - Ce `StandardScalerModel` peut être appliqué à ce Dataframe (ou à un autre) afin d'effectivement réaliser la normalisation

III. Machine Learning avec Spark

Estimator

- Utilisation avec Spark
  - On instancie l'Estimator en spécifiant les colonnes d'entrée et sortie (+ éventuellement ses hyperparamètres)
  - On « ajuste » (fit) l'Estimator sur un Dataframe avec la méthode `.fit` qui nous renvoie le Transformer associé
  - On utilise le modèle récupéré avec la méthode `.transform` comme précédemment

III. Machine Learning avec Spark

Estimator

- Exemple d'utilisation de l'Estimator StandardScaler

```scala
import org.apache.spark.ml.feature.StandardScaler

val scaler = new StandardScaler().setInputCol("vector").setOutputCol("normalized").setWithMean(true)
val fittedScaler = scaler.fit(df)
val normalizedDf = fittedScaler.transform(df)  // we can use it on another dataframe if we want
```

```python
from pyspark.ml.feature import StandardScaler

scaler = StandardScaler(inputCol="vector", outputCol="normalized", withMean=True)
fittedScaler = scaler.fit(df)
normalizedDf = fittedScaler.transform(df)  # we can use it on another dataframe if we want
```

StandardScaler attend un **vecteur** en colonne d'entrée car il peut normaliser plusieurs variables à la fois. On peut créer une colonne vecteur à partir d'une (ou plusieurs) colonne(s) numérique(s) avec le Transformer VectorAssembler

Dataframe initial

```
+-----+-------+
|input| vector|
+-----+-------+
| 99.2| [99.2]|
| 91.5| [91.5]|
|101.2|[101.2]|
| 98.9| [98.9]|
|105.0|[105.0]|
+-----+-------+
```

Dataframe final

```
+-----+-------+----------------------+
|input|vector |normalized            |
+-----+-------+----------------------+
|99.2 |[99.2] |[0.008122266928344049]|
|91.5 |[91.5] |[-1.5554141167776416] |
|101.2|[101.2]|[0.41423561334548303] |
|98.9 |[98.9] |[-0.05279473503422622]|
|105.0|[105.0]|[1.1858509715380465]  |
+-----+-------+----------------------+
```

III. Machine Learning avec Spark

Algorithmes de machine learning

- Les algorithmes de ML sont des Estimator
  - La méthode `.fit` correspond à l'entraînement du modèle et la méthode `.transform` correspond son utilisation en prédiction
  - Lors de la prédiction, un modèle ajoute une colonne avec la prédiction, mais peut aussi ajouter des colonnes intermédiaires
- Les algorithmes d'apprentissage supervisé prennent en entrée une seule colonne de variables et une colonne de labels (+ d'éventuels hyperparamètres)
  - La colonne de variables (généralement nommée "features") doit être un **vecteur de variables numériques**
    - En cas de variables dans plusieurs colonnes, il faut les rassembler en un seul vecteur avec le VectorAssembler
    - En cas de variables non-numériques, il faut les numériser avec divers Estimator/Transformer existant : StringIndexer, CountVectorizer, OneHotEncoder…

III. Machine Learning avec Spark

Évaluation d'un modèle de machine learning

- Quelques utilitaires proposés par Spark
  - **BinaryClassificationEvaluator**
  - **MulticlassClassificationEvaluator** (certaines métriques utiles en classification binaire uniquement disponibles via cet objet)
- Majorité des métriques faciles à calculer manuellement
  - Majoritairement basées sur le nombre de **VP**, **VN**, **FP**, **FN**
- Plus de détails dans le TP

Plan de la présentation

Liens utiles et références

Pour découvrir et prendre en main SparkML

- Guide officiel de SparkML
  - https://spark.apache.org/docs/3.3.1/ml-guide.html
  - Menu de gauche à explorer → exemple disponibles pour tous les Estimator et Transfomer disponibles (en Scala, Java, Python, R)
  - Parties intéressantes pour le TP :
    - Tous les préprocesseurs de features : https://spark.apache.org/docs/3.3.1/ml-features.html
    - Tous les modèles de classification : https://spark.apache.org/docs/3.3.1/ml-classification-regression.html
- API officielle
  - https://spark.apache.org/docs/3.3.1/api/scala/org/apache/spark/ml/ (Scala)
    https://spark.apache.org/docs/3.3.1/api/python/reference/pyspark.ml.html (Python)
  - Menu de droite à explorer → beaucoup moins clair que le guide, mais référence exhaustivement tous les objets avec leurs méthodes

Liens utiles et références

Formations INRAE/CNRS pour aller plus loin

- DigitBio
  - https://digitbio-ia.github.io/
- Fidle
  - Formation Introduction au Deep Learning
  - https://fidle.cnrs.fr
  - https://www.youtube.com/c/CNRSFormationFIDLE

Questions ?

Léo Géré
leo.gere@inrae.fr

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

INRAE

lipme

cnrs
dépasser les frontières

Travaux pratiques

Traitement des données et utilisation d'outils ML

- Scala ou Python au choix
- Scala
  - Utilisation du Spark-Shell
  - Conseil : fichier texte d'un côté puis copier-coller dans le shell avec la commande `:paste`
- Python
  - Utilisation d'un notebook Jupyter :

```
conda activate pyspark
jupyter-notebook
```

  - À noter : à cause de soucis de versions de Pyspark, on ne pourra pas lancer les calculs sur les clusters