FIORINI Aymeric

VANELLE Rémi

# Projet Big Data : Documentation Technique :

## Table des matières :

[Projet Big Data : Documentation Technique : 1](#_Toc72723679)

[Table des matières : 2](#_Toc72723680)

[Copie des fichiers de windows vers le serveur hadoop 3](#_Toc72723681)

[Copie des fichiers du serveur vers HDFS 3](#_Toc72723682)

[Création des dossiers requit sur HDFS : 3](#_Toc72723683)

[Copie des CSV 3](#_Toc72723684)

[Création des tables Hive 3](#_Toc72723685)

[Recherche de solution de stockage AWS : 5](#_Toc72723686)

[Types de stockage : 5](#_Toc72723687)

[Types de stockage par objets : 7](#_Toc72723688)

[Création Amazon S3 : 9](#_Toc72723689)

[Commandes de création du bucket : 9](#_Toc72723690)

[Création de configuration hiérarchisation intelligente pour notre bucket : 9](#_Toc72723691)

[Insertion des fichiers de données dans le bucket : 12](#_Toc72723692)

[Accès partagé d’un bucket S3 : 13](#_Toc72723693)

[Machine Learning : 17](#_Toc72723694)

[Phase de pré-processing : 17](#_Toc72723695)

[Recherche de la meilleure qualité de prédiction : 18](#_Toc72723696)

[Datavisualisation : 19](#_Toc72723697)

[Choix de Tableau Software : 19](#_Toc72723698)

[Taux de prédictions : 19](#_Toc72723699)

[Mots les plus présents : 25](#_Toc72723700)

[Export AWS 28](#_Toc72723701)

# Copie des fichiers de windows vers le serveur hadoop

La première étape de ce projet a consisté en la copie des données fournies sur une plateforme Big Data : le système de fichier HDFS.

Sur Windows, il a fallu installer un utilitaire permettant de copier un fichier de l’ordinateur vers le serveur qui héberge un node hadoop.

J’utilise l’outil pscp en ligne de commande pour transmettre les fichiers depuis ma machine locale vers la node hadoop via SSH.

pscp -P 2222 categories\_string.csv [root@192.168.17.128:/home/hdfs](mailto:root@192.168.17.128:/home/hdfs)

pscp -P 2222 export/data.csv [root@192.168.17.128:/home/hdfs](mailto:root@192.168.17.128:/home/hdfs)

pscp -P 2222 label.csv [root@192.168.17.128:/home/hdfs](mailto:root@192.168.17.128:/home/hdfs)

# Copie des fichiers du serveur vers HDFS

Une fois les fichiers sur la node hadoop, il faut les copier du système de fichier Linux vers le système de fichiers HDFS.

Je crée tout d’abord des dossiers dans HDFS pour le projet big data, ainsi que pour les trois types de données : catégories, labels et enfin les CVs qui seront situés dans le dossier data :

hdfs dfs -mkdir /user/hdfs/projet-big-data

hdfs dfs -mkdir /user/hdfs/projet-big-data/categories

hdfs dfs -mkdir /user/hdfs/projet-big-data/data

hdfs dfs -mkdir /user/hdfs/projet-big-data/labels

Enfin, je copie les fichiers CSV vers HDFS :

hdfs dfs -put data.csv /user/hdfs/projet-big-data/data

hdfs dfs -put label.csv /user/hdfs/projet-big-data/labels

hdfs dfs -put categories\_string.csv /user/hdfs/projet-big-data/categories

# Création des tables Hive

Une fois les CSV présents sur HDFS, nous aurons besoin de les requêter pour utiliser les données associées sur notre machine locale.

Hive permet de requêter nos données en utilisant un langage qui s’apparente au SQL.

Nous pourrons donc utiliser un connecteur vers Hive afin de requêter nos données en langage Java.

CREATE DATABASE projet\_big\_data;

USE projet\_big\_data;

CREATE EXTERNAL TABLE IF NOT EXISTS label(

Id INT, Category INT)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ‘,’

STORED AS TEXTFILE

LOCATION ‘/user/hdfs/projet-big-data/labels’

CREATE EXTERNAL TABLE IF NOT EXISTS categorie(

Id INT, Category INT)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ‘,’

STORED AS TEXTFILE

LOCATION ‘/user/hdfs/projet-big-data/categories’

CREATE EXTERNAL TABLE IF NOT EXISTS cv(

Id INT, Description STRING, gender STRING)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ‘,’

STORED AS TEXTFILE

LOCATION ‘/user/hdfs/projet-big-data/data’

## Recherche de solution de stockage AWS :

### Types de stockage :

Il nous faut d’abord choisir le type de stockage le plus adapté pour notre projet. Voici les 2 types de stockage du service AWS que nous avons étudiés :

* **Stockage par objets :** 3 composants : les données contenues dans l’objet, leurs identifiants et leurs métadonnées (nom, taille, type de données stockées…). Pas de hiérarchie entre les fichiers : deux objets liés ne sont pas forcément stockés aux mêmes emplacements physiques. Idéal quand on veut que les données soient accessibles par plusieurs clients. Conçu pour la lecture de données, et non pas leur écriture.
* **Stockage par blocs :** Utilisés pour stocker les BDD, effectuer des opérations de lecture/écriture sur ces données. Les données sont stockées dans des volumes et des blocs et n’ont pas de métadonnées. Les blocs de données sont de taille égale. On peut leurs associés des systèmes de fichiers pour stocker les données de manière durable. On accède au volume de donnée avec une instance Amazon EC2.

**Avantages du stockage par objet :**

* Localisation des objets rapide n’importe où sur le serveur grâce à leur ID : utile dans notre cas, car on voudra chercher des extraits de CV en fonction de leur profession.
* Système de versionnage des objets.
* Pas de limites pour les données stockées, alors qu’on est obligé de réallouer dans le stockage bloc. Très adapté pour des volumes de données très important.

**Avantages du stockage par bloc :**

* Plus facile de modifier les données. Dans le stockage par objets si on veut modifier un objet on est obligé de le recréer.
* Meilleur IOPS :nombre de lectures et d’écritures possible par seconde. Pas trop d’importance dans notre cas car on ne veut pas lire les données, mais les transférer.

Dans notre cas, le système de stockage par objets est donc plus adapté. De plus, il existe un service appelé Amazon Machine Learning qui facilite les prédictions sur les jeux de données importants. Ce service peut créer des algorithmes de ML avec des données stockées sous AmazonS3, raison de plus pour utiliser le stockage par objets.

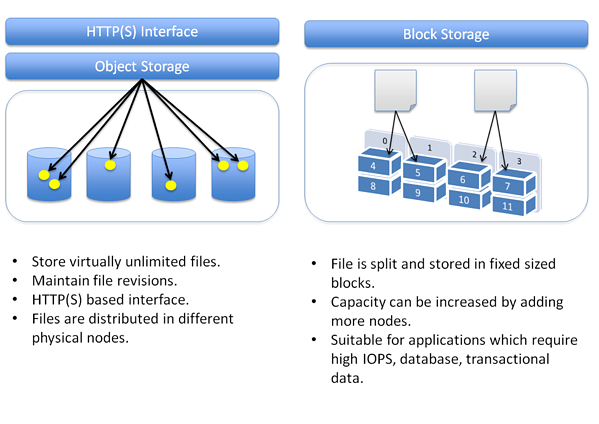


Image récapitulative

[**https://cloud.netapp.com/blog/block-storage-vs-object-storage-cloud**](https://cloud.netapp.com/blog/block-storage-vs-object-storage-cloud)

[**https://www.netapp.com/fr/data-storage/storagegrid/what-is-object-storage/**](https://www.netapp.com/fr/data-storage/storagegrid/what-is-object-storage/)

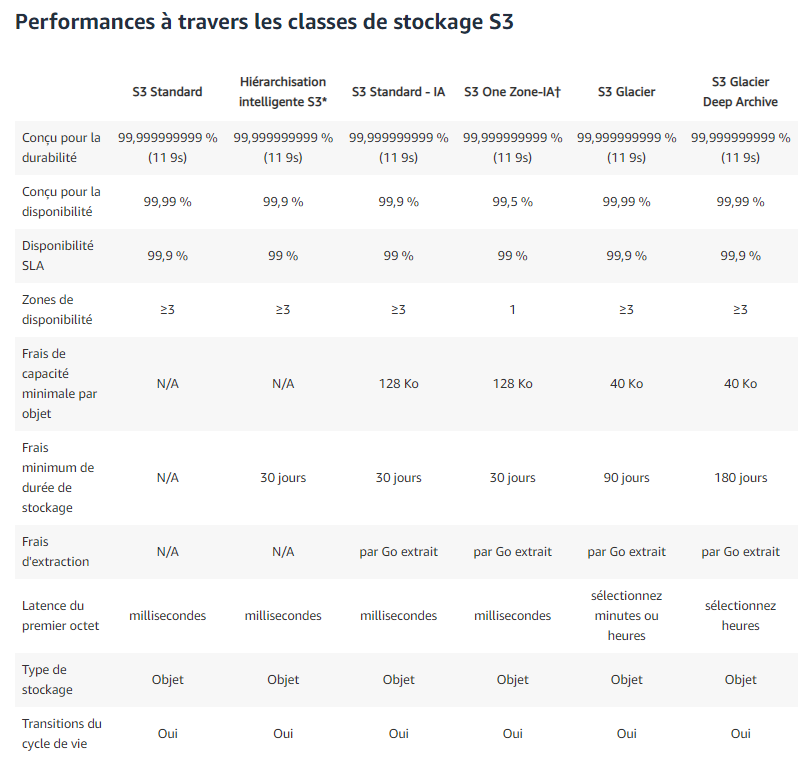
[**https://d1.awsstatic.com/whitepapers/fr\_FR/Big\_Data\_Analytics\_Options\_on\_AWS.pdf**](https://d1.awsstatic.com/whitepapers/fr_FR/Big_Data_Analytics_Options_on_AWS.pdf)

### Types de stockage par objets :

Nous avons ensuite recherché les différentes solutions existantes dans le stockage par objets d’AWS, et choisi la + adapté pour notre projet.

|  |  |  |
| --- | --- | --- |
| **Classe de stockage** | **Avantages** | **Inconvénients** |
| Amazon S3 Standard | Faible latence,  Débit élevé  adapté à de nombreux cas d’utilisation, notamment les analyses Big Data | Coût non indiqué |
| Hiérarchisation intelligente S3 | Déplace automatiquement les objets vers la zone d’accès la plus économique.  4 niveaux d’accès : fréquents, peu fréquents, archivage et archivage profond | Déplace les objets qui n’ont pas été consultés pendant 30 jours en archive. D’un autre côté, ils peuvent être récupéré facilement. |
| S3 Standard – IA | Permet d’accéder rapidement aux données peu fréquentes.  Faible coût + performances élevées : idéal pour le stockage des données à long terme. | Prévu pour l’accès peu fréquent des données |
| S3 One Zone-IA+ |  | Données sauvegardées dans 1 seule zone de sauvegarde /!\ |
| S3 Glacier | Faible coût | Délai d’extraction de plusieurs heures /!\ |
| S3 Glacier Deep Archive | Même chose que S3 Glacier en exagérer | Délai d’extraction plus important que S3 Glacier /!\ |

La solution de stockage par objet la plus adaptée à notre projet est donc la solution Amazon S3 Hiérarchisation intelligente. Il y a peu de différences entre Amazon S3 Standard, S3 Standard – IA et Hiérarchisation intelligente S3. Néanmoins, le fait qu’il déplace automatiquement les données non utilisées dans des zones non utilisées (vu qu’il y aura forcément des données que nous n’allons pas utilisées tout de suite, séparation des données d’entraînement et des données de tests) le rend plus adapté.

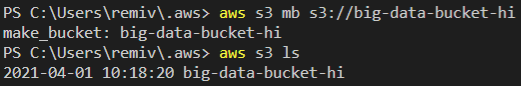


<https://aws.amazon.com/fr/s3/storage-classes/>

### Création Amazon S3 :

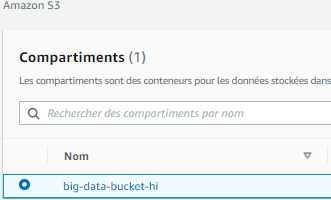
Je vais maintenant vous expliquer comment nous avons mis en place le bucket de notre solution.

### Commandes de création du bucket :

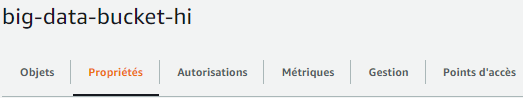


Création de configuration hiérarchisation intelligente pour notre bucket :

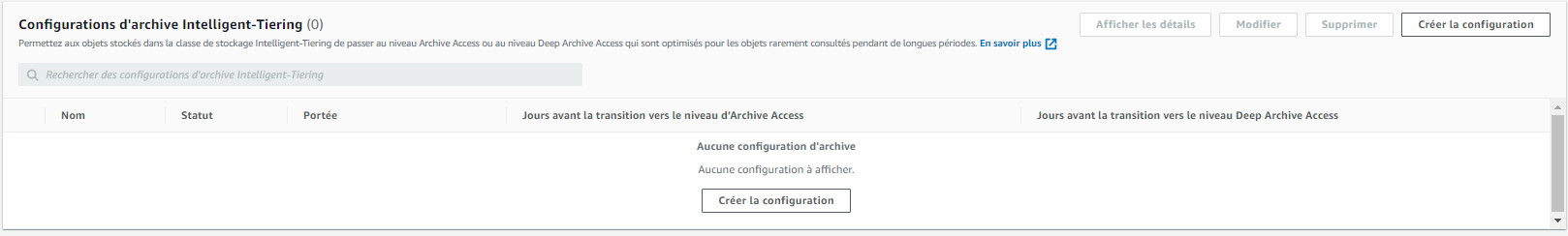
Une fois le bucket créé, cliquer sur le nom de ce dernier.



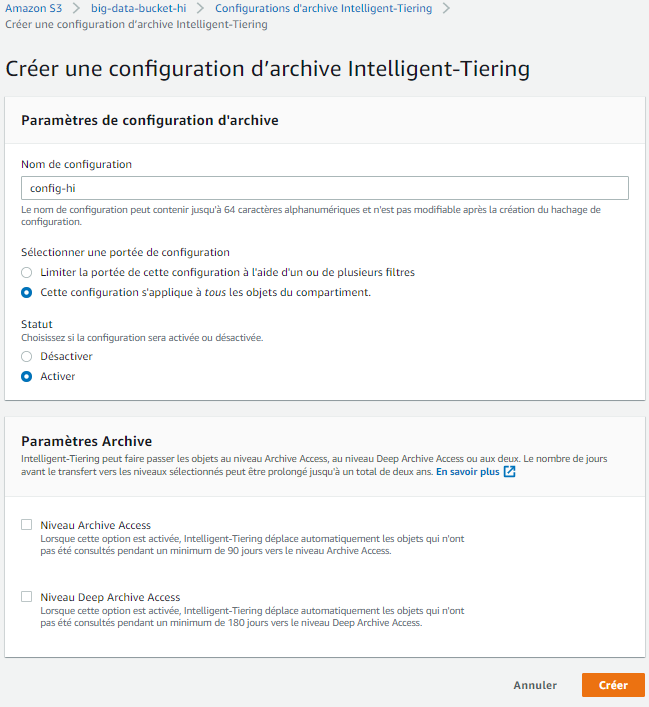
Cliquer sur l’onglet “Propriété” :

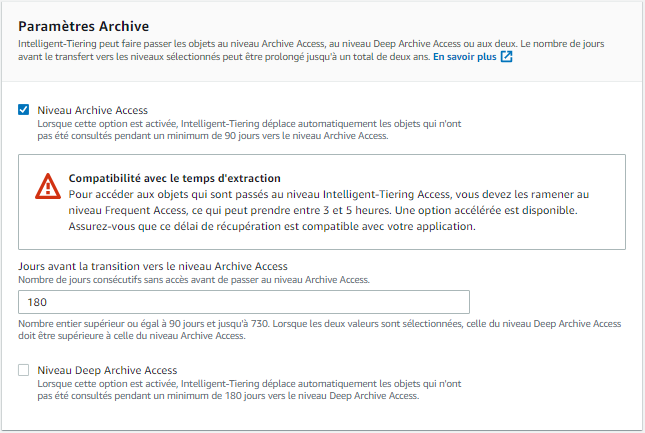


Descendre jusqu’à arriver à la section “Configurations d’archive Intelligent-Tiering", puis cliquer sur “Créer la configuration”.



Dans la nouvelle fenêtre, choisir le nom de votre configuration, puis choisissez les paramètres ci-dessous (il faut obligatoirement choisir un niveau de paramètres d’archive) :

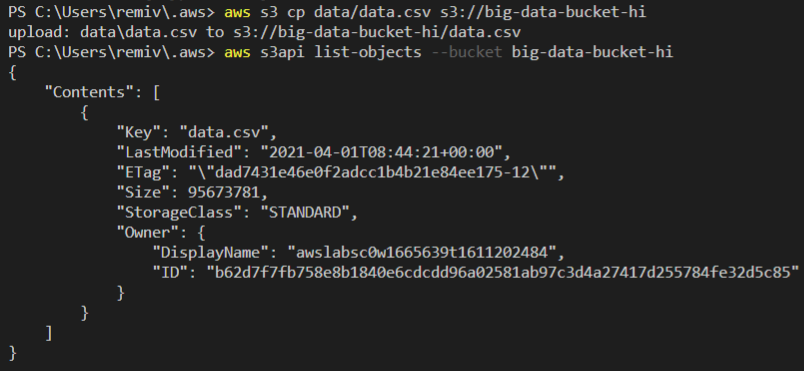




Lien pour les commandes

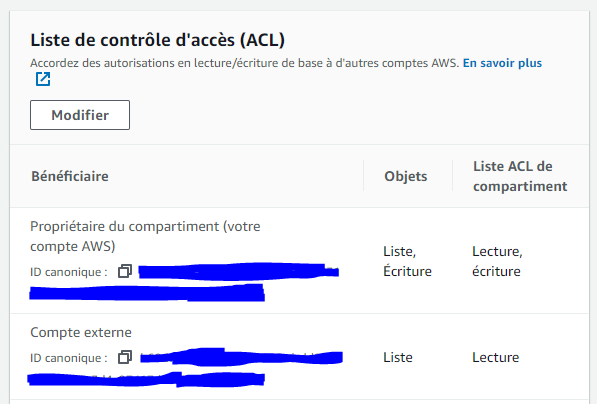
### Insertion des fichiers de données dans le bucket :





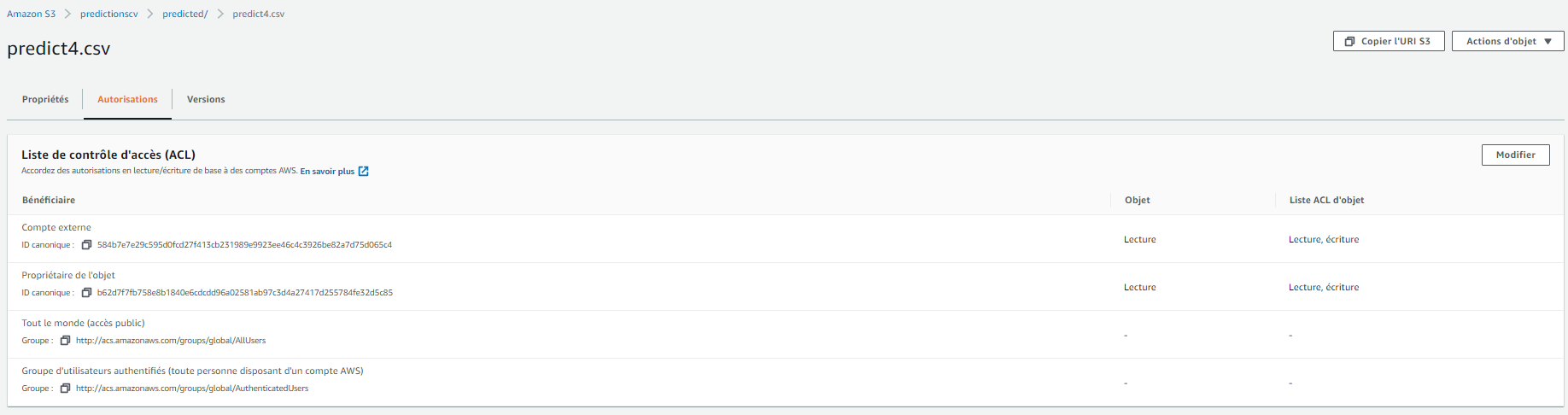
### Accès partagé d’un bucket S3 :

Depuis la console AWS : S3 -> sur les paramètres du bucket ajouter un bénéficiaire depuis l’onglet Autorisations

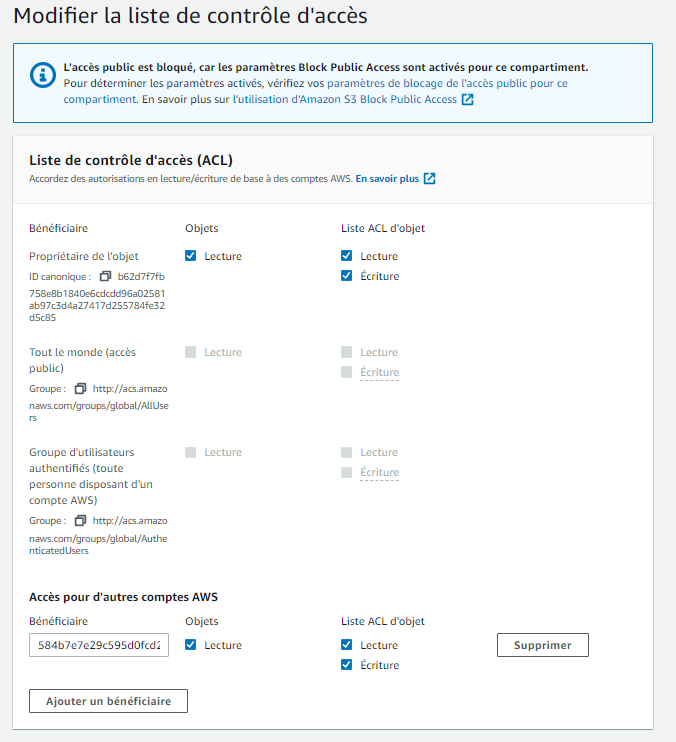


Pour récupérer l’ID canonique d’un autre utilisateur, il faut le récupérer depuis cet onglet (sur un bucket qu’il aurait créé par exemple avec son compte)

Il peut également être nécessaire d’accorder les droits pour les objets du bucket. Dans ce cas, il faut se rendre dans l’onglet autorisation de l’objet. Cliquer sur le bouton “Modifier” de la liste de contrôle d’accès.



Dans la fenêtre des ACL, Ajouter un bénéficiaire en indiquant l’ID canonique du membre de votre équipe que vous voulez ajoutez, et cochez les droits qu’il doit posséder sur l’objet.



## Machine Learning :

Cette étape consiste à trouver mettre au point un algorithme de prédictions de métier à partir des extraits de CV.

<https://pandas.pydata.org/docs/reference/api/pandas.read_json.html>

<https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0>

### Phase de pré-processing :

Cette phase consiste à préparer nos données pour la phase d’apprentissage. Il y a quatre étapes de pré-processing :

1. Mettre tous nos extraits de CV en majuscule/ou en minuscule afin que les mots en début de phrase et au milieu de la phrase ne soient pas comptés comme des mots différents.
2. Supprimer la ponctuation des phrases (virgules, points...).
3. La phase de tokenisation (division des extraits de CV : on passe d'une seule colonne à une colonne pour chaque mot).
4. La suppression des stop-words, ces mots qui sont présents dans tous les extraits de CV car ils sont présents dans toutes les phrases (articles, verbe être, adjectifs possessifs...) mais qui ne sont pas pertinents pour nous aider à classer les extraits de CV.

Or la phase d’apprentissage ne pouvait accepter qu’une seule colonne pour l’axe X, donc nous nous sommes contentés des 2 premières phases de pré-processing, et nous avons appliqués les 2 dernières plus tard.

### Recherche de la meilleure qualité de prédiction :

Nos données sont prêtes à être traitées. On passe donc notre colonne description en paramètre de la fonction train\_test\_split, qui va séparer nos données en données d’apprentissage (pour déterminer le poids des entrées offrant la meilleure prédiction possible) et de tests (pour tester l’algorithme avec les valeurs des poids déterminées à la fin de la phase précédente). On a choisi 30% des données pour servir de tests puisqu’il s’agit de valeurs normales souvent utilisées.

Cette séparation terminée, on cherche un algorithme d’apprentissage. Nous choisissons pour commencer la régression logistique car il s’agit de l’algorithme le plus utilisé en Machine Learning pour prédire la probabilité qu’un évènement se produise ou non lorsque qu’on a des valeurs Y aléatoires et des valeurs X non aléatoires. Dans notre cas nous prédisons la probabilité que la prédiction soit correcte en utilisant comme paramètre X la description de nos extraits de CV et comme paramètre Y les identifiants associés aux métiers réels des personnes à qui appartiennent ces extraits de CV. Elle attribue une valeur de 1 si l’évènement s’est produit (si la prédiction correspond à la profession réelle) et 0 s’il ne s’est pas produit (si la prédiction est différente).

Une fois l’algorithme exécuté, nous regardons son paramètre f1 score, qui est la somme de la précision de nos tests. Il est la moyenne des valeurs booléennes retournées par la régression logistique. Plus ce paramètre est proche de 1 plus la précision est précise. Nous avons pour valeur environ 0,715, un ordre de gardeur habituel pour un premier résultat. Nous avons ensuite cherché à améliorer cette valeur en agissant sur deux points :

* Les valeurs des paramètres de la régression logistique
* Le choix de l’algorithme de prédiction de probabilité utilisé.

En faisant des recherches, nous avons vu deux paramètres possédant une influence non négligeable sur la précision :

* Le paramètre class\_weight. Par défaut à None (tous les mots en entrée ont le même poids), nous faisant passer sa valeur à balanced (ajuste automatiquement le poids des entrées afin qu’il soit inversement proportionnel à la fréquence de la classe dans les données d’entrées). Les entrées peuvent avoir des valeurs de poids différentes : en offrant plus de liberté, il est logique que l’on améliore notre précision.
* Le paramètre C. Paramètre qui s’assure d’un compromis au niveau de la complexité du modèle. Plus la valeur est élevée, plus le modèle pourra augmenter sa complexité, et vice-versa. Augmenter la complexité du modèle augmente la précision de la prédiction, mais trop l’augmenter provoque des risques de sur-apprentissage. Par défaut la valeur était à 1. Après des tests, nous avons déterminé que la valeur 10 était celle qui nous offrait la meilleure valeur du f1 score.

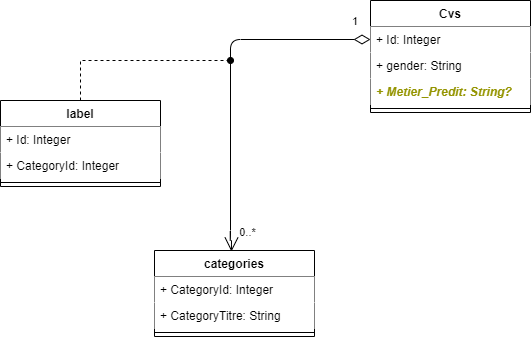
Après modification de ces paramètres, nous obtenons un f1 score de valeur de 0,745.

Nous avons ensuite, pour les mêmes paramètres, comparés la valeur du f1 score avec un autre algorithme, le svm. La valeur de f1 score obtenu étant inférieur, nous avons donc décidé de conserver la régression logistique.

Nous avons à présent exporté les résultats sous forme de fichier csv. Ce fichier est celui que nous avons utilisé pour la phase de data-visualisation.

# Récupération des données prédites

Afin de transférer les données prédites par AWS vers une datavisualisation locale, j’ai créé un script qui permet de réaliser la transmission de ces données dans une base de données de type NoSQL : il s’agit de MongoDB.

Notre script Java permettant de récupérer les données depuis S3.

Pour cela nous utilisons la bibliothèque S3, combiné à la bibliothèque OpenCSV.

La bibliothèque S3 renvoie les données du fichier sous la forme d’un flux de données (InputStream), qu’il faut consommer via un lecteur de flux. (InputStreamReader)

La bibliothèque OpenCSV récupère l’objet BufferedReader lors de la création de l’objet document CSV et va créer un objet de type CSVReader qui va posséder des méthodes permettant de lire un fichier CSV délimité par des virgules.

Je peux retourner une liste de chaines de caractères (String) qui correspondent aux champs du fichier CSV que je veux récupérer.

Afin de simplifier le programme, cette fonction de récupération est localisé dans une classe Service afin de l’utiliser facilement.

Dans notre script le chargement du CSV est réalisé, puis l’insertion des données dans la base MongoDB est réalisée via une connexion JDBC.

Ci-contre le schéma représentant les trois tables sous MongoDB. Nous utilisons la flexibilité de MongoDB pour afficher la colonne Metier\_Predit uniquement lorsqu’une prédiction est présente sur AWS.

## Datavisualisation :

A présent que nous avons nos prédictions, nous allons pouvoir les utiliser pour vous transmettre des informations.

### Choix de Tableau Software :

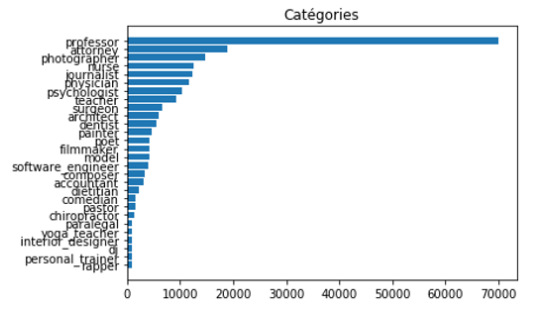
Il y a plusieurs technologies différentes pour effectuer de l’analyse visuelle de données. Nous avons acquis, grâce à notre formation académique et entreprise, de l’expérience avec D3.js et Tableau Software. D3.js est une bibliothèque graphique javascript qui permet d’afficher des données numériques sous forme graphique à l’intérieur de nos pages web. Tableau Software est un outil de BI (Business Intelligence) qui permet de créer des tableaux de bords consultables, eux-mêmes contenant des feuilles de calculs. Chaque feuille de calcul contient une figure différente. Tous deux peuvent traiter des fichiers au format csv.

Nous avons choisi Tableau Software car nous avons une plus grande expérience avec cet outil, ce qui nous a permis de gagner du temps, l’une des grandes contraintes de ce projet. Nous avons ainsi pu réaliser des visualisations de meilleures qualités que si nous avions choisi D3.js. Tableau est également un outil spécialement conçu pour traiter de gros volumes de données, ce qui le rend donc plus adapter à notre cas d’utilisation que D3.js. Enfin, il offre un grand niveau d’interaction possible avec les figures réalisées.

### Taux de prédictions :

Nous avons pensé qu’une information intéressante serait de pouvoir comparer la qualité de nos prédictions, à la fois sur l’ensemble de nos extraits de CV et pour des prédictions particulières. En comparant ces dernières avec le nombre d’extraits de CV pour chaque profession, nous pourrons en déduire si le nombre d’occurrence à une influence sur la précision de la prédiction.

Pour rappel, voici nos extraits de CV classés par profession de la plus présente à la moins présente :



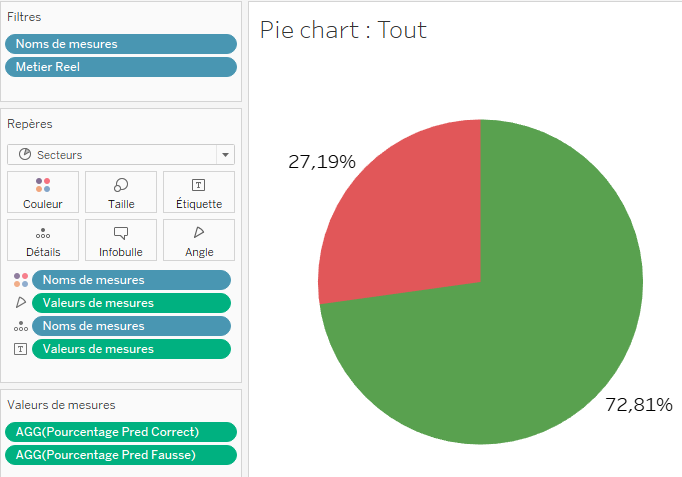
Un bon moyen de représenter la précision de la prédiction pour une profession est de représenter le nombre de prédictions correctes sur le nombre de prédictions total, soit le taux de prédictions correctes. Une représentation parfaitement adaptée à l’affichage des taux est le diagramme circulaire. Il s’agit d’un cercle divisé en segments d’arc de cercles et de couleur différentes, ce qui les rend facilement distinguable les uns des autres. Chaque segment représente les différentes valeurs d’une variable, dans notre cas prédictions juste et prédictions fausses. Si le nombre de variables est trop élevée, alors la représentation deviendra surchargée, mais comme nous n’en avons que deux, c’est parfaitement adapté.

Pour avoir le nombre de prédictions correctes, nous avons utilisé un champ calculé, un paramètre créé à partir d’opérations appliquées à des mesures (variables numériques) de Tableau. Nous l’avons créé en faisant la somme des prédictions correctes (booléen obtenue en comparant la profession prédite et la profession réelle) divisé par la somme du nombre de données, puis en multipliant le tout par 100 :

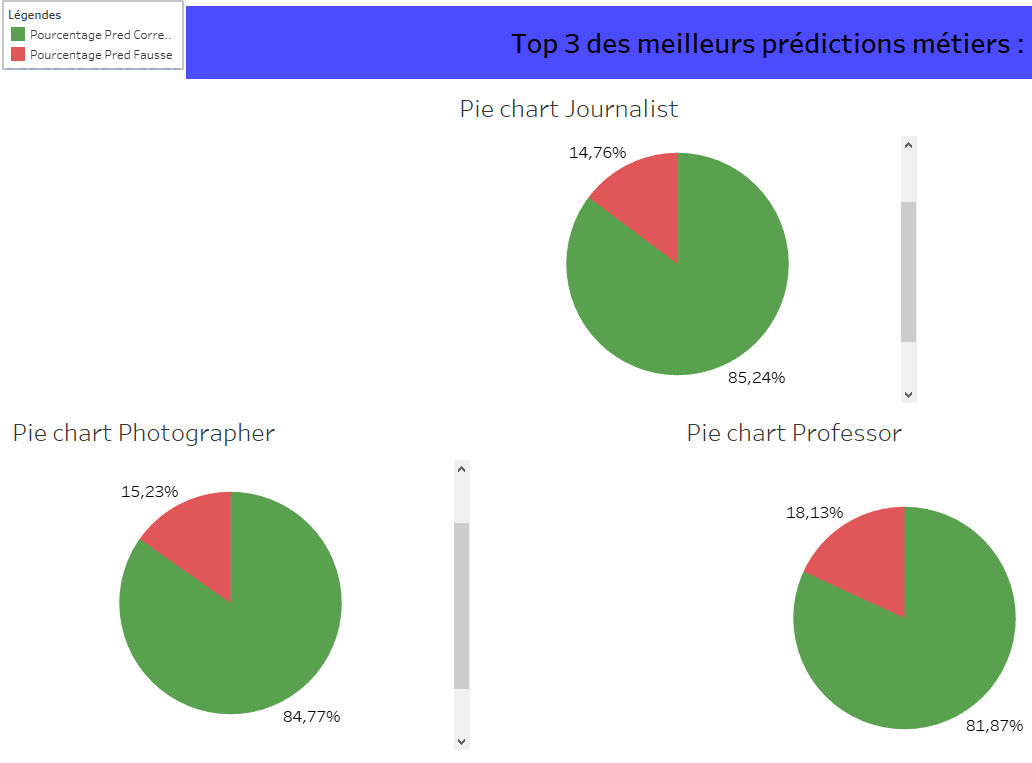
*SUM([Pred Correct])/SUM([Nombre d'enregistrements])\*100*

Le taux de prédictions négatif est lui avec un champ calculé qui fait la différence entre 100 et le précédent champ calculé.

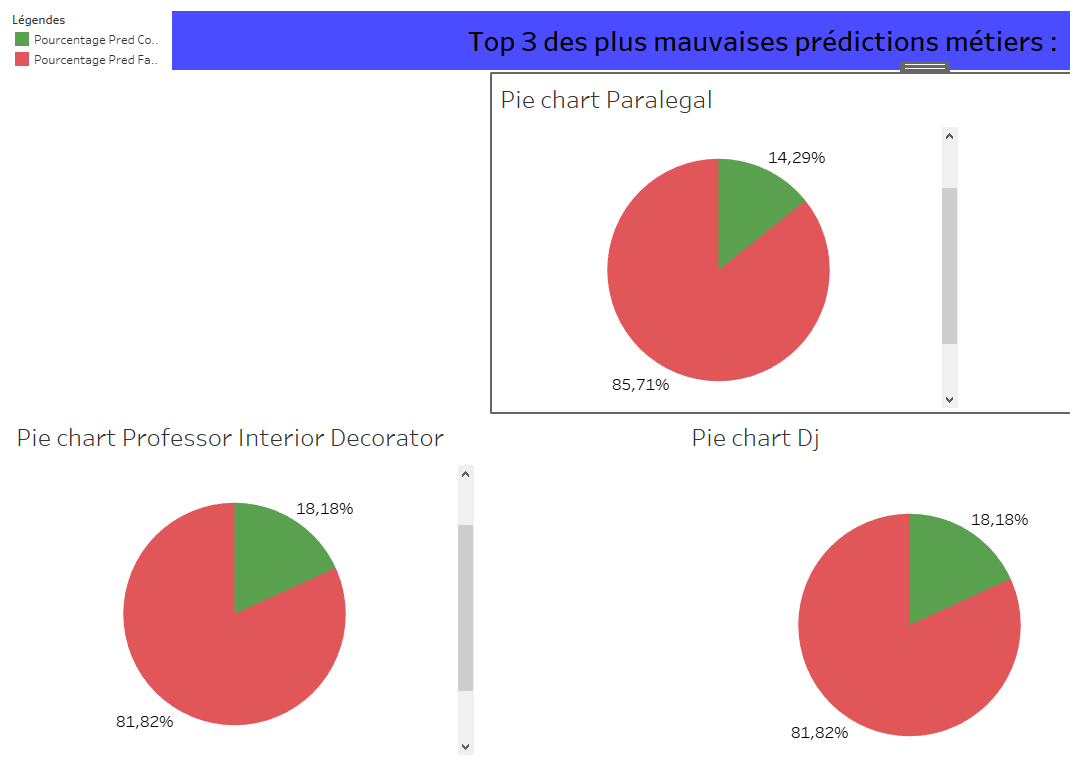
Une fois ces deux champs créés, nous utilisons ensuite les valeurs de ces deux champs calculés pour créer le cercle ci-dessous en les faisant glisser vers les étiquettes couleurs et angle de la fenêtre Repères, après avoir choisi le type de représentation “Secteurs”.



On peut voir que la prédiction pour l’ensemble des métiers est proche de la valeur de notre f1 score. Une fois le diagramme circulaire créé, nous pouvons appliquer comme filtre les métiers réels des extraits de CV pour choisir le métier que l’on veut voir la prédiction, ou si l’on veut voir la prédiction de l’ensemble des métiers. Voici le top 3 des métiers avec le meilleur pourcentage de bonnes prédictions :



Et voici le top 3 des plus mauvaises prédictions :

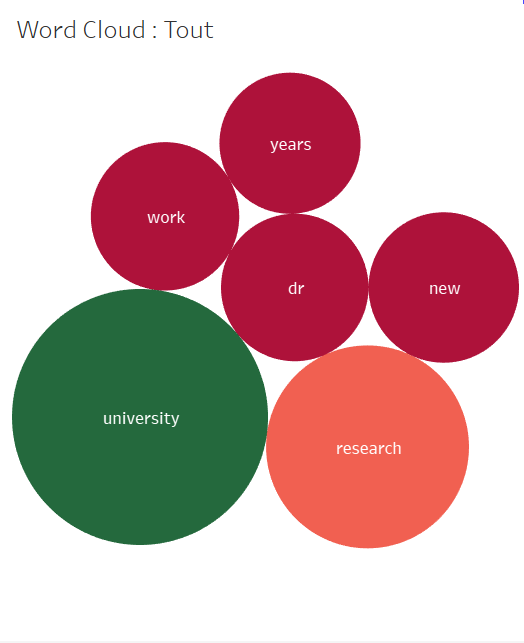


On remarque que les métiers avec les prédictions les moins précises font partie des catégories les moins représentées, et que ceux avec les meilleures prédictions font partie des catégories les plus représentées. On peut donc en déduire que le nombre d’occurrence des catégories de métiers à bien une influence sur la qualité des prédictions.

### Mots les plus présents :

Le second type d’information que l’on veut afficher est de pouvoir voir quels sont les mots qui reviennent les plus souvent pour les prédictions d’un métier donné. Ainsi, quand nous verront de nouveaux extraits de CV, en comparant les mots les plus nombreux de celui-ci avec les mots les plus présents d’une catégorie, on pourra prédire la profession de la personne à qui appartient ce nouveau CV.

Pour cela, il nous faut utiliser un nuage de mots. Nous avons créé cette figure en faisant glisser la somme du nombre d’enregistrements dans les étiquettes couleurs et taille, et les mots séparés des extraits de CV dans l’étiquette Texte, et après avoir supprimer les stops-words des mots filtrés (les articles, les adjectifs possessifs, les modaux, les auxiliaires...) des mots filtrés. On a fait utiliser les mots comme filtres pour cela en excluant les mots indésirables.



Nous avons également créé la possibilité de choisir le nombre de mots que l’on veut afficher grâce à un paramètre et en choisissant sa valeur.

# Export AWS