

PROJET BIG DATA ANALYTICS

ANALYSE DE LA CLIENTELE D'UN CONCESSIONNAIRE AUTOMOBILE POUR LA RECOMMANDATION DE MODELES DE VEHICULES

Réalisé par le Groupe 4

NITEKA Lys Ciella
MAMANE LAWEL Sadio
ABDALLAH Swabahadine
DIALLO Elhadj Mamadou Foula
PAJANY CARPIN CAOUNDIN Allan

Master 2 : Mobilité, Base de Données & Big Data, et Intégration de Systèmes

[Lien Github du projet](#)

Responsables

Mr. MOPOLLO MOKE Gabriel
Mr. SIMONIAN Sergio

Gestion des données

Mme. MELIN Aline
Mr. WINKLER Marco

Visualisation des données

Mr. PASKIER Nicolas

Analyse des données



SOMMAIRE

LE PROJET	3
1. GESTION DE PROJET	4
1.1. AGILE SCRUM	4
1.2 GITHUB	4
1.3. REPARTITION DU TRAVAIL	5
2. ARCHITECTURE DU PROJET BIG DATA VOITURES	6
2.1. ARCHITECTURE 1 : DATA LAKE HADOOP AVEC TABLES EXTERNES	6
2.1.1. SGBD NoSQL	6
2.1.2. Hadoop HIVE	15
2.1.3 Oracle	17
2.2. ARCHITECTURE 2 : HDFS HADOOP UNIQUEMENT SANS TABLES EXTERNES	20
2.2.1. Nettoyage et préparation des données	20
2.2.2. Adaptation et intégration du fichier C02	21
3. TECHNIQUES DE DATA VISUALISATION	22
3.1. CHAÎNE DE TRAITEMENT	22
3.2. LES ACTEURS VISES ET OBJECTIFS DE VISUALISATION	29
4. ANALYSE DES DONNÉES PAR LES TECHNIQUES DE DATA MINING, MACHINE LEARNING ET DEEP LEARNING	30
4.1. ANALYSE EXPLORATOIRE DES DONNÉES	30
4.2. DÉTECTION DES TYPES DES VOITURES	31
4.2.1. Première approche	31
4.2.2. Clustering avec l'algorithme des K-means	32
4.2.3. Matrices de confusion du modèle KNeighbors	36
5. PROBLÈMES ET DIFFICULTÉS RENCONTRÉS	38
5.1 MONGO ET DYNAMODB	38
5.2 SOLUTIONS	38
6. PERSPECTIVES ET RÉFLEXIONS PERSONNELLES	38
CONCLUSION	39
REFERENCES	40
1. GESTION DES DONNÉES	40
2. VISUALISATION DES DONNÉES	40



LE PROJET

Ayant été contacté par un concessionnaire automobile afin de l'aider à mieux cibler les véhicules susceptibles d'intéresser ses clients. Cinq fichiers de données nous ont été mis à disposition :

- **Catalogue.csv** : Catalogue de véhicules

Catalogue								
Marque	Nom	Puissance	Longueur	NbPlaces	NbPortes	Couleur	Occasion	Prix

- **Client_N.csv** : Fichier clients concernant les achats de l'année en cours

Clients_N						
Age	Sexe	Taux	SituationFamiliale	NbEnfantsACharge	2eme voiture	Immatriculation

- **Immatriculations.csv** : Informations sur les immatriculations effectuées cette année

Immatriculation									
Immatriculation	Marque	Nom	Puissance	Longueur	NbPlaces	NbPortes	Couleur	Occasion	Prix

- **Marketing.csv** : Clients sélectionnés par le service marketing

Marketing					
Age	Sexe	Taux	SituationFamiliale	NbEnfantsACharge	2eme voiture

Le but du projet est de permettre au concessionnaire d'évaluer sans trop de difficulté, la catégorie de véhicule susceptible d'intéresser un client qui viendrait se présenter à la concession et envoyer par la suite documentation une documentation détaillée du véhicule.

Nous devons pour cela mettre en place une chaîne de processus dans lequel nous implanterons différentes architectures de stockages de données, (interconnexion de différents SGBD, récupération des données, nettoyage, insertion...), de visualisation et d'analyse.



1. GESTION DE PROJET

1.1. AGILE SCRUM

Afin d'avoir une meilleure organisation du projet, nous avons utilisé les acquis du cours de gestion de projet. Nous avons appliqué la méthode Agile et donc définie des sprints de rendu toutes les semaines.

1.2 GITHUB

Pour la mise en œuvre de cette méthode, le projet a été hébergé sur Github et nous avons ensuite découpé nos exigences de manière simple et conventionnelle avec :

- EPIC : Regroupe toutes les exigences décomposées en plusieurs story.
- Story : Définie les exigences d'un besoin au point de vu utilisateur.
- Task : Plus technique, identifie les tâches à effectuer au niveau du développeur.

Ci-dessous le planning succinct des exigences sur lesquelles nous avons travaillé :

EXIGENCES	NOVEMBRE				DECEMBRE				JANVIER			
Réunion : Choix des environnements techniques, programmation et SGBD NoSQL												
Définition, organisation et répartition de la charge de travail (Github, Agile-Scrum)												
Nettoyage des données												
Mise en place de l'architecture 1 et 2												
Mise en place des connexions entre les bases de données NoSql, Hive et Oracle (A1)												
Implémentation des algorithmes pour la partie HADOOP-MAPREDUCE (A2)												
Définition des variables et choix des types de visualisations												
Mise en place des visualisations dans R												
Migration des visualisations de R dans D3JS												
Réalisations de la partie analyse des données.												
Rédaction du rapport												

1.3. REPARTITION DU TRAVAIL

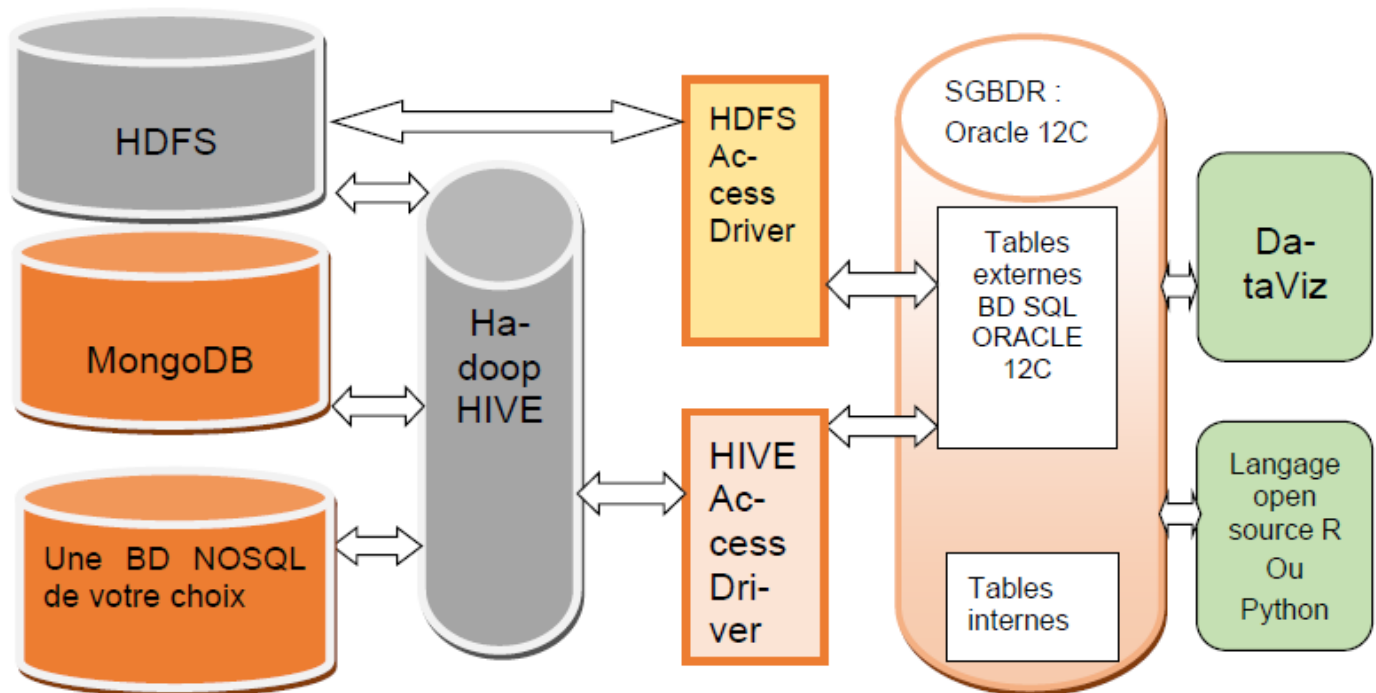
Le travail à été répartie comme suit :

- ✓ Architecture 1 :
 - SGBD NoSQL : **DIALLO Elhadj Mamadou Foula**
 - HADOOP HIVE : **MAMANE LAWEL Sadio**
 - ORACLE : **NITEKA LYS Ciella**
- ✓ Architecture 2 :
 - INSERTION ET PREPATION DES DONNES : **PAJANY CARPIN CAOOUNDIN Allan**
 - ADAPTION ET INTEGRATION DES DONNES : **ABDALLAH Swabahadine**
- ✓ Visualisation et analyse : Tous le monde

Cependant, vu la diversité des concepts à mettre en œuvre dans le projet, nous avons travailler tous ensemble en parfaite synergie.

2. ARCHITECTURE DU PROJET BIG DATA VOITURES

2.1. ARCHITECTURE 1 : DATA LAKE HADOOP AVEC TABLES EXTERNES



2.1.1. SGBD NoSQL

Pour nos bases de données NoSql aux choix, nous avons utilisé des solutions cloud : **MongoDB Atlas** et **DynamoDB**.

a. MongoDB Atlas :

Alternative de MongoDB, Atlas MongoDB est une solution multicloud pour MongoDB adapté pour équipe agiles et facile d'utilisation disponible sur les plateformes AWS, Google Cloud et Azure.

Dans cette base de données, nous avons chargé les données du fichier clients11.csv.

Pourquoi MongoDB Atlas ? :

Nous l'avons choisi, car il donne efficacement une solution d'utilisation de cluster à distance très adaptée pour la mise en place de notre lac de données.

Installations effectuées en locale :

- Mongo Shell
- MongoDB Compass

Etapes de création de la base de données :

Afin de pouvoir utiliser Atlas MongoDB, nous avons dans un premier temps créer un compte sur le site : [Atlas MongoDB](#), ensuite effectués les étapes suivantes:

- Création du projet : TPA_MBDS_G4

The screenshot shows the 'Create a Project' page in the MongoDB Cloud console. The left sidebar contains navigation links: ORGANIZATION, Projects (selected), Alerts, Activity Feed, Settings, Access Manager, Billing, and Support. The main content area has a 'Name Your Project' section with a text input field containing 'TPA_MBDS_G4' and a 'Next' button. Below the input field, there is a 'Cancel' button and a 'Next' button.

Une fois le projet créé, nous avons créé et configuré un cluster pour contenir nos données.

- Création du cluster : pour créer un cluster, nous nous sommes rendus dans l'onglet **Cluster** du projet, puis sur « **Build A Cluster** ».

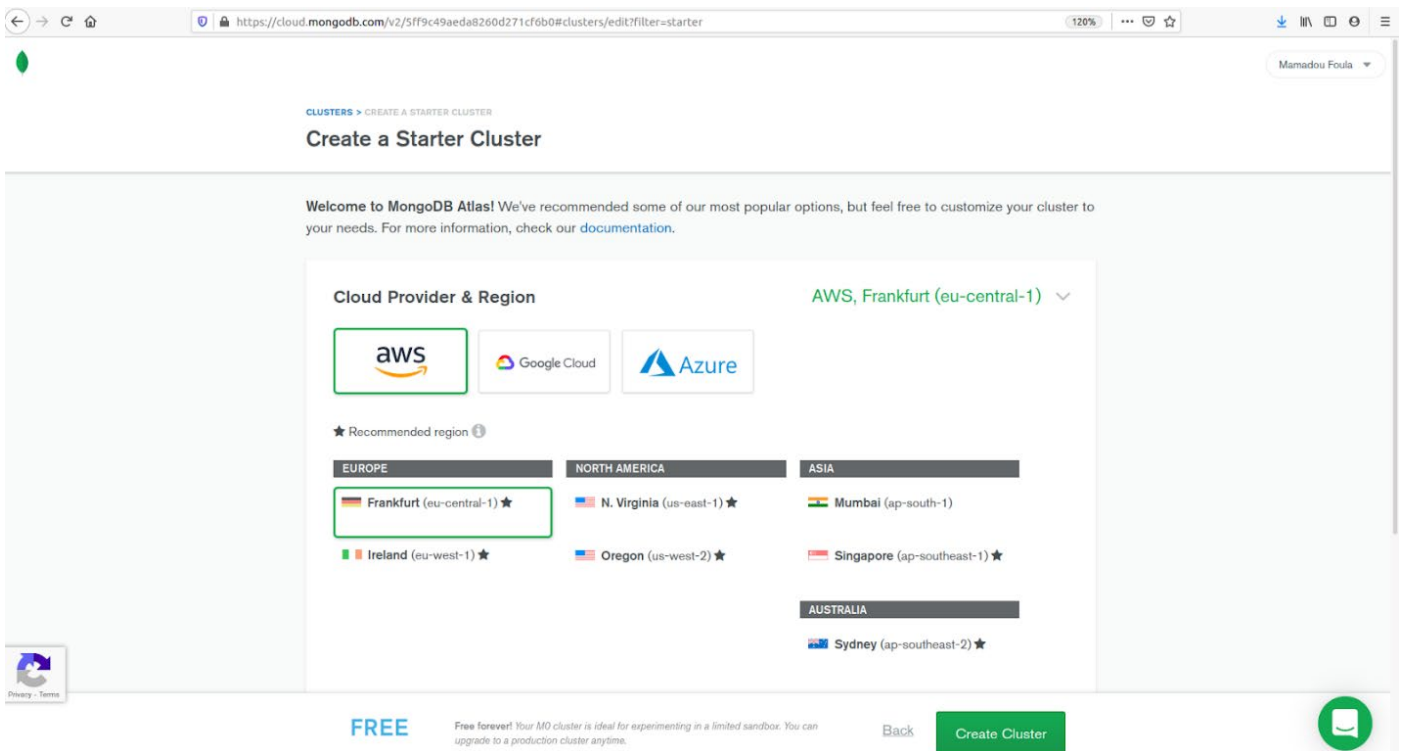
The screenshot shows the 'Create a cluster' page in the MongoDB Cloud console. The left sidebar contains navigation links: DATA STORAGE, Clusters (selected), Triggers, Data Lake, SECURITY, Database Access, Network Access, and Advanced. The main content area has a 'Create a cluster' section with a 'Build a Cluster' button. Below the button, there is a note: 'Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our Live Migration Service.'

- Choisir l'option “**FREE**” sur la prochaine fenêtre :

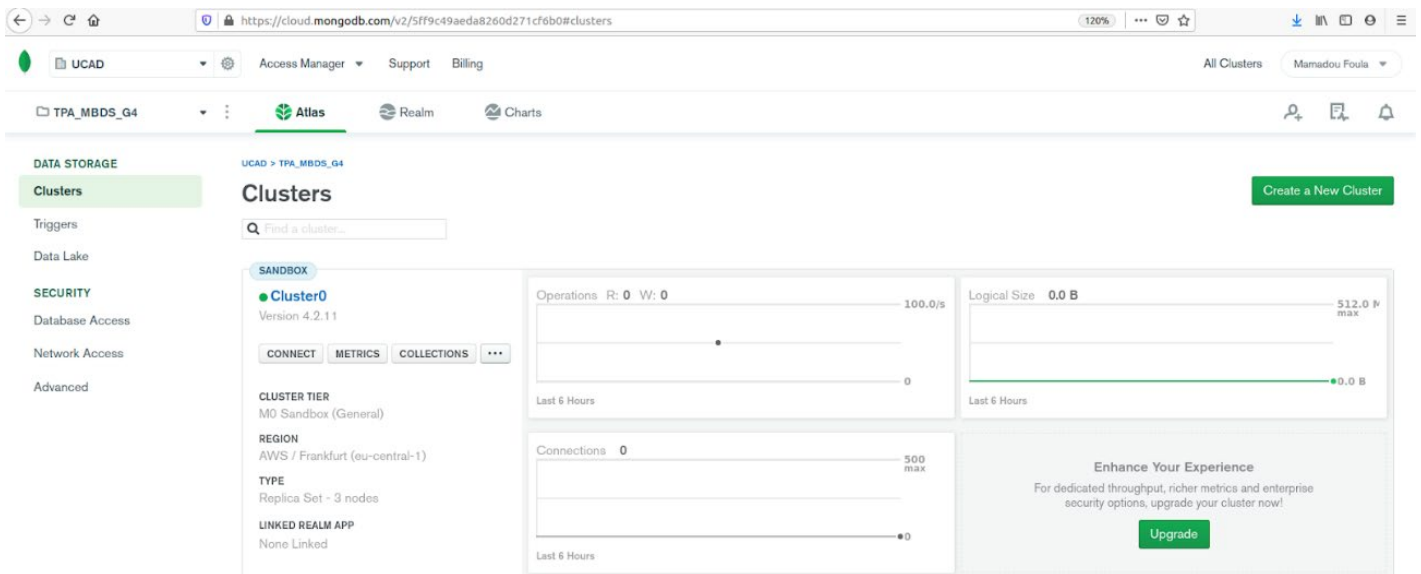
The screenshot shows the 'Choose a path. Adjust anytime.' page in the MongoDB Cloud console. The page displays three options for creating a cluster: Dedicated Multi-Cloud & Multi-Region Clusters, Dedicated Clusters, and Shared Clusters. Each option has a 'Create a cluster' button and a starting price. The Shared Clusters option is highlighted with a green border and a 'FREE' label.

Option	Starting Price
Dedicated Multi-Cloud & Multi-Region Clusters	\$0.13/hr*
Dedicated Clusters	\$0.08/hr*
Shared Clusters	FREE

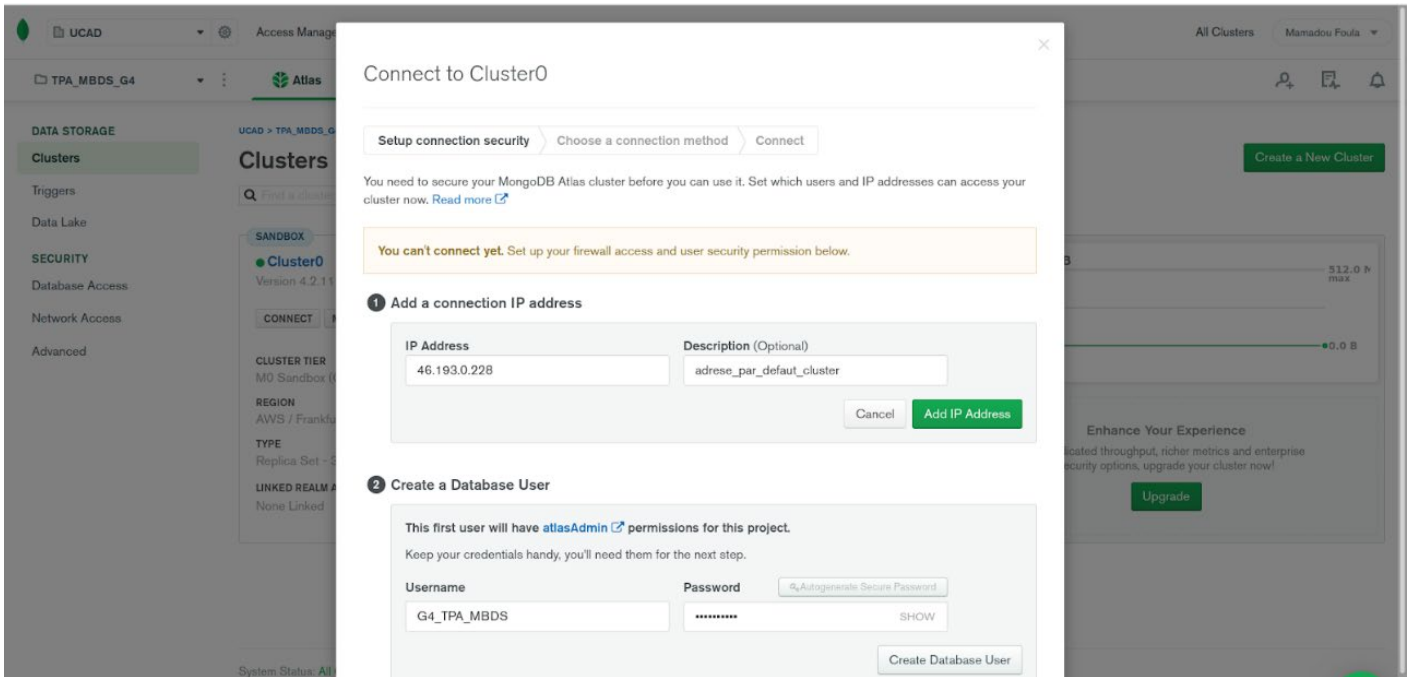
- Nous avons utilisé **AWS** comme plateforme pour héberger notre cluster et **Frankfort** pour la région, puis cliquez sur « **Create Cluster** » afin de valider la création du cluster.



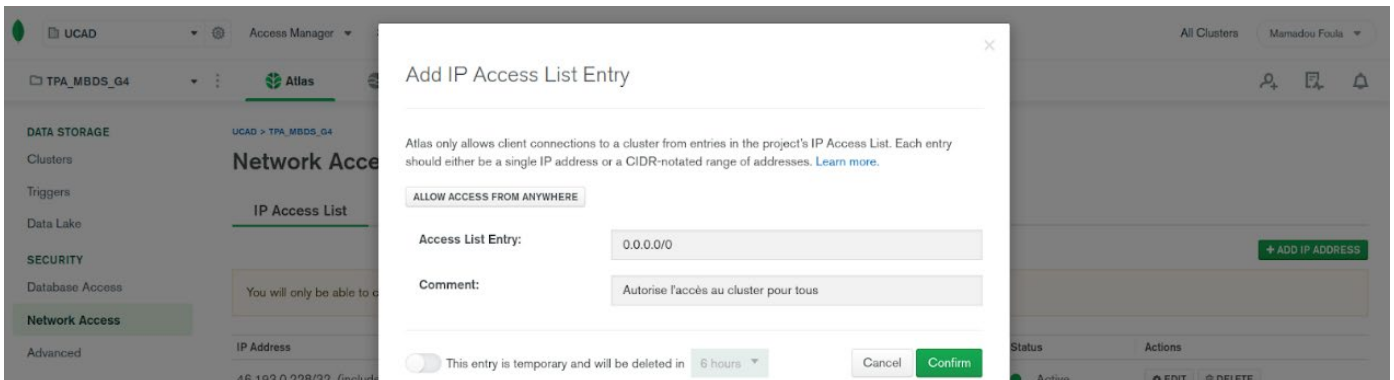
- Configuration du cluster - connexion et des accès utilisateurs : Pour permettre à notre cluster d'être utilisé par des applications externes, nous avons configuré une connexion, en ajoutant une adresse IP et éventuellement un ou plusieurs utilisateurs. Pour cela, nous nous sommes rendus dans l'onglet **Cluster** et avons choisi l'option **CONNECT**.



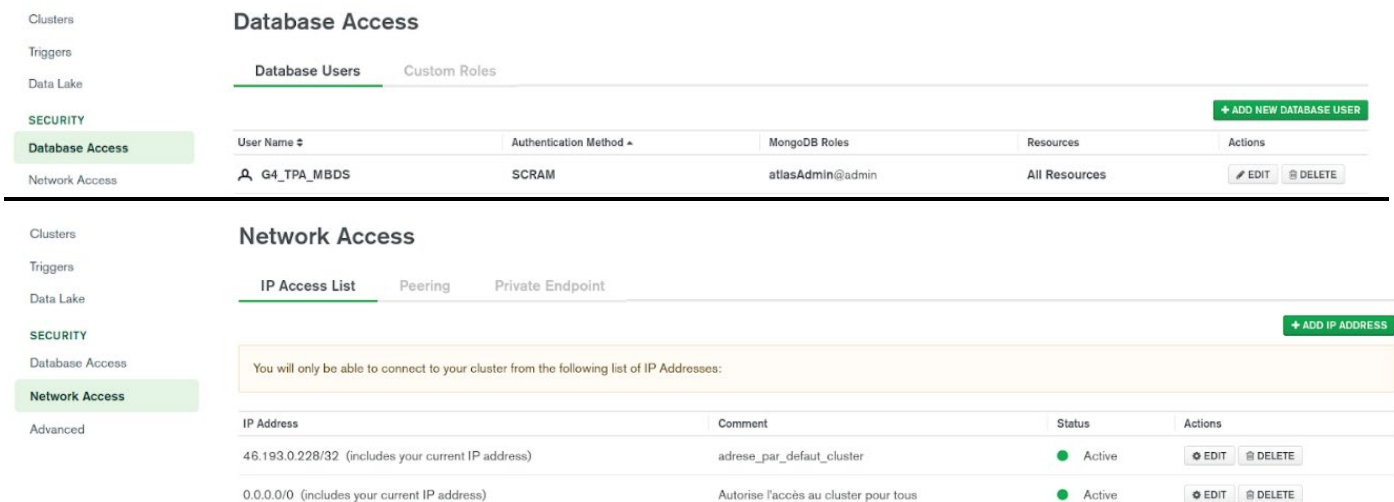
- Une nouvelle fenêtre s'ouvre en nous proposant d'utiliser l'adresse IP de la machine courante comme adresse du cluster ainsi que de remplir les champs **User** et **Password** afin de définir un utilisateur pour le cluster.



- Par ailleurs, pour permettre l'accès du cluster pour tout le monde, nous avons configuré l'adresse **0.0.0.0/0** sur le cluster en utilisant l'onglet **Database Access** comme suit:



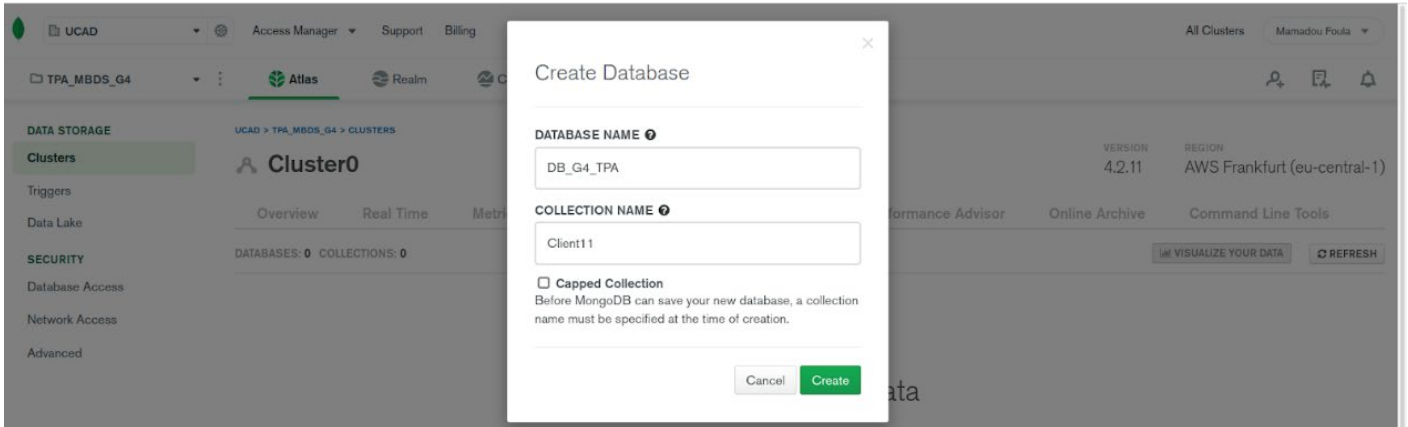
- Une fois l'adresse et l'utilisateur créé, nous pouvons les visualiser, modifier, supprimer, ou même en rajouter respectivement à partir des onglets Database Access et Network Access :



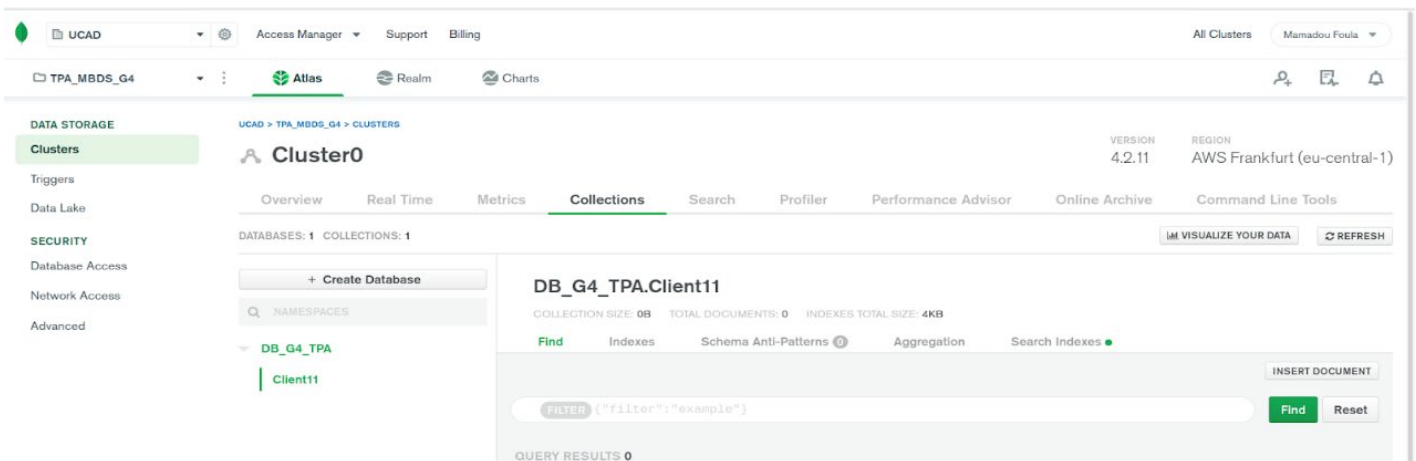
Après cette configuration, notre cluster est maintenant prêt pour recevoir des données.

Une fois notre cluster créé et configuré, nous pouvons ainsi créer une base de données et y insérer des collections, en procédant comme suit :

- Création de la base données : A partir de l'onglet Collection du cluster, cliquer sur l'option "ADD MY OWN DATA", remplir le nom de la base de données et celui du cluster



- A la fin de cette opération, nous obtenons ainsi une base de données nommée DB_G4_TPA et une collection Client11 prête à recevoir les données de notre fichier client11.csv.



- Insertion de clients11.csv : Pour insérer les données contenues dans clients11.csv, nous avons expérimenté deux méthodes :

- **Mongo Shell** : Cette méthode consiste à installer MongoDB en local et utiliser la chaîne de connexion proposée à cet effet par notre cluster. Dans le Shell mongo, la connexion au cluster se fait en utilisant la commande suivante :

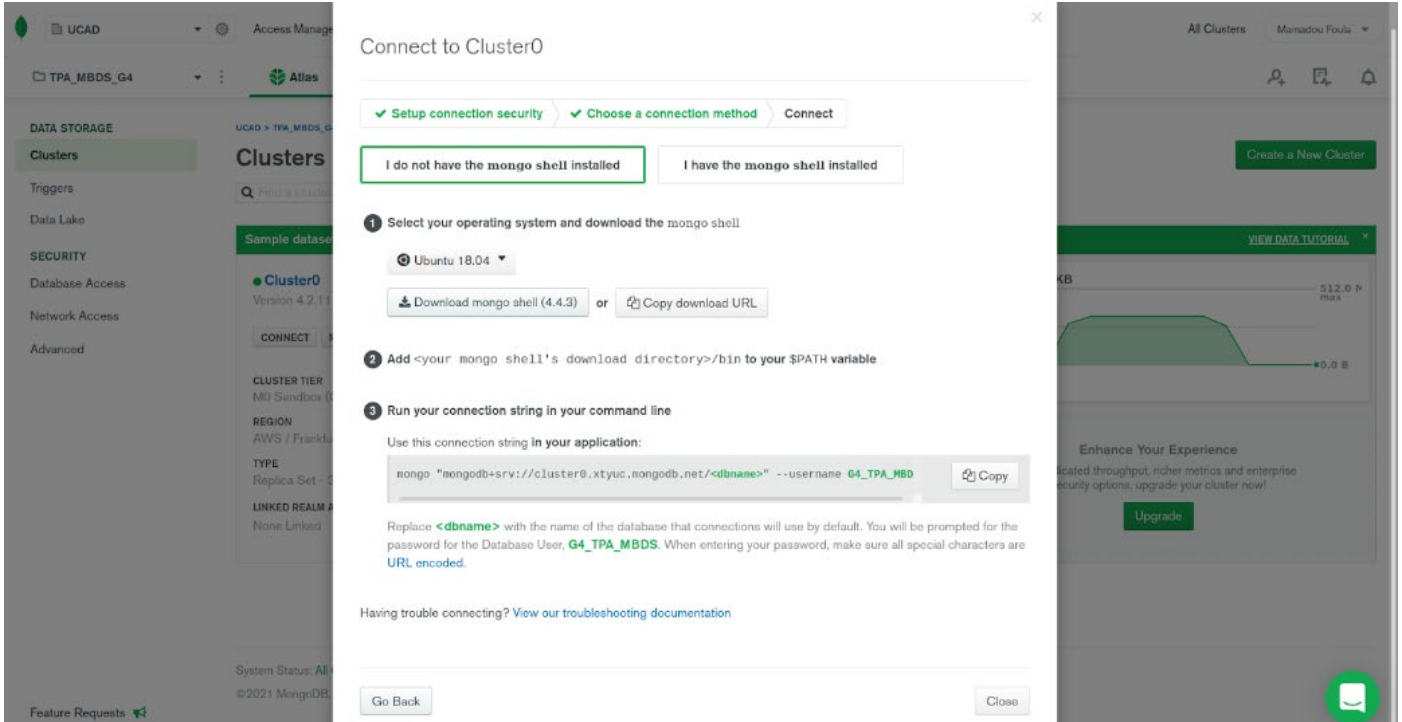
- `mongo "mongodb+srv://cluster0.xtyuc.mongodb.net/DB_G4_TPA" --username G4_TPA_MBDS`

On insère ensuite les données :

- `mongoimport --host=cluster0.xtyuc.mongodb.net --db=DB_G4_TPA --collection=Client11 --type=csv --file=Clients_11.csv --authenticationDatabase=admin --ssl --username=G4_TPA_MBDS --password=DIALLO2B2001 --headerline`

- **Mongo Compass** : Application permettant d'interfacer notre cluster à distance, elle permet également d'insérer directement nos données dans le cluster à partir d'un fichier

- Connexion entre Mongo Compass et notre cluster :
 - mongodb+srv://G4_TPA_MBDS:DIALLO2B2001@cluster0.xtyuc.mongodb.net/DB_G4_TPA
- Une fois la connexion établie : Se connecter sur le cluster -> DB_G4_TPA -> Documents -> ADD DATA -> Import file -> choisir le fichier client11.csv



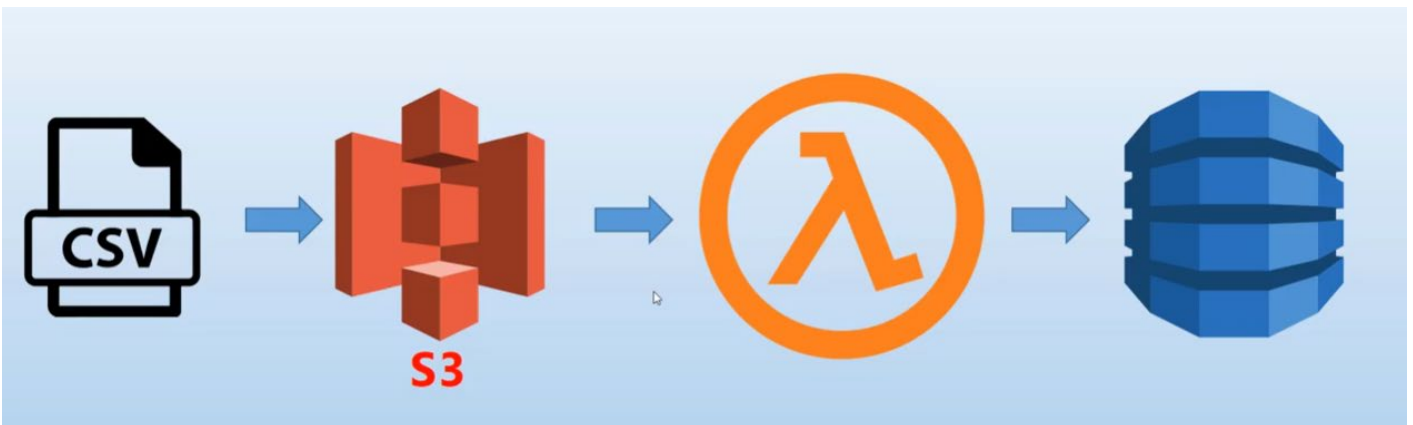
b. Amazon DynamoDB :

Pour notre deuxième base de données NoSql, nous avons choisi Amazon DynamoDB.

C'est un service AWS de base de données NoSql rapide, flexible, orienté document et clé-valeur, il offre de très grandes permanences avec un temps de réponse très court.

Pourquoi DynamoDB ? : Nous avons choisi Amazon DynamoDB d'abord pour découvrir à travers ce TPA l'un des services cloud les plus utilisés actuellement par les entreprises dont entre autres (Airbnb, Samsung, Toyota, ...), mais aussi parce qu'il donne la possibilité grâce à ses nombreux services de déployer une base de données dans le cloud accessible partout et à tout moment.

- Etapes de création de la base de données :

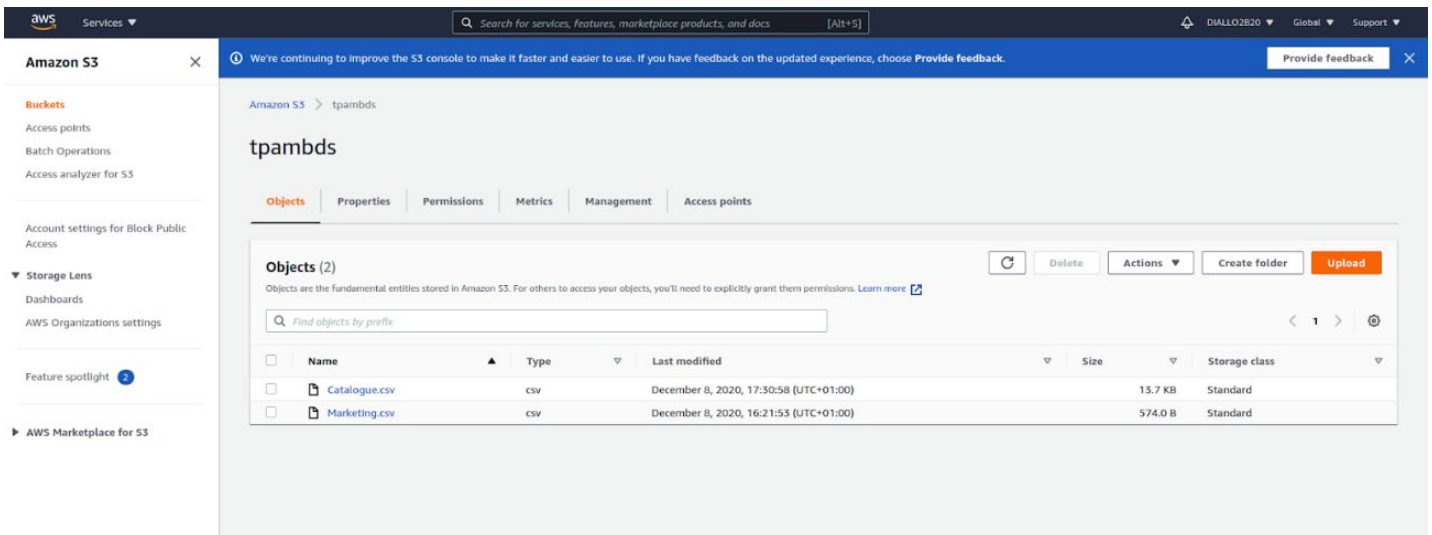


Pour mettre en place notre base de données, nous avons effectués principalement cinq étapes :

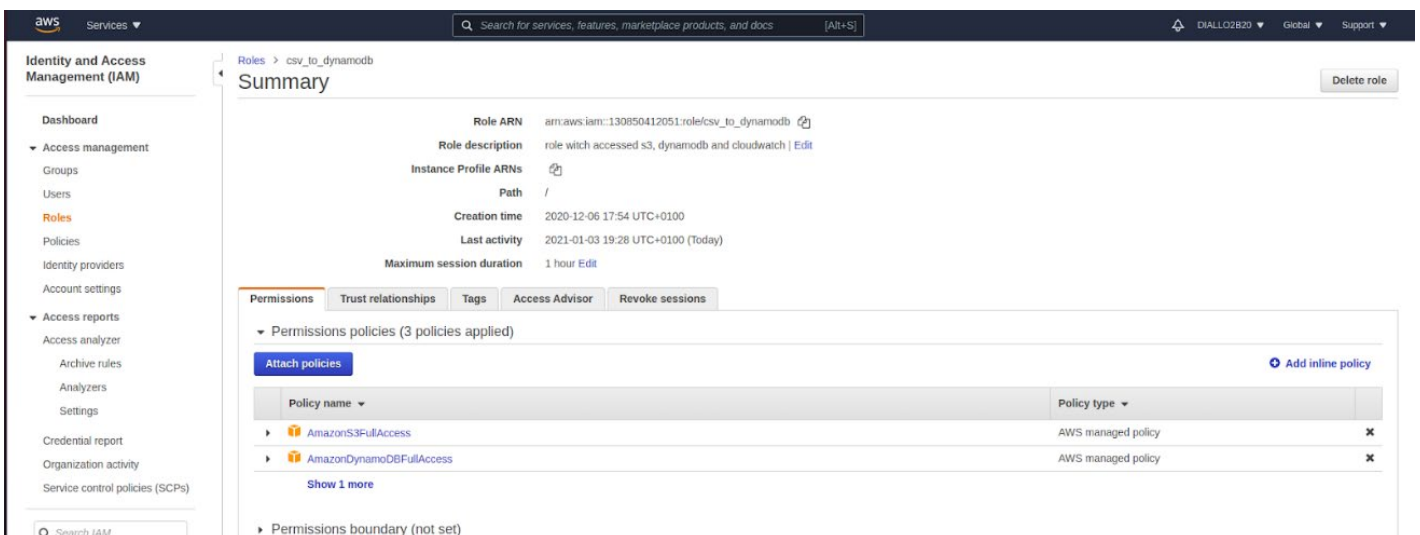
1. **Création d'un compte sur AWS :** Pour pouvoir utiliser Amazon DynamoDB, il faut au préalable créer un compte sur le site [AWS DynamoDB](#)

Nb : Nous avons utilisé l'option **Free Tier** afin d'être dispensé de payer les frais d'hébergement du cluster.

2. **Chargement des données dans S3 Bucket :** une fois le compte créé et s'être connecté à la console, nous avons pour un premier temps chargé les données de nos fichiers catalogue.csv et marketing.csv dans un Bucket Amazon S3. Pour cela, nous avons créé à partir du service **Amazon S3** un Bucket "tpambds".



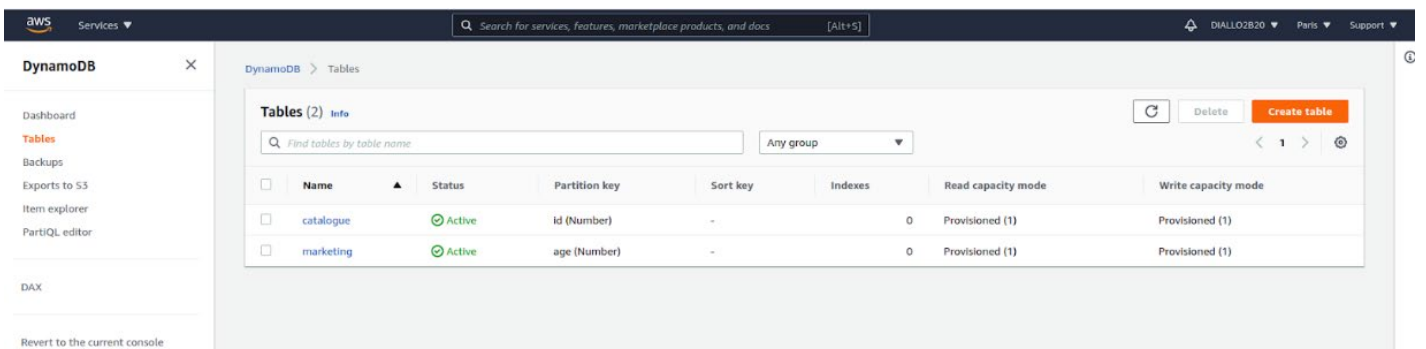
3. **Création de Rôles (Autorisation) avec le service IAM :** Identity Access Manager est un service qui permet de définir les droits d'accès à nos données, ce rôle sera utilisé juste après par notre fonction lambda pour pouvoir accéder aux données du Bucket S3 "tpambds" afin de récupérer les données de nos fichiers et les charger dans la BD.



4. **Ecriture d'une fonction lambda avec le langage python :** Après avoir chargé les données dans le Bucket S3 et mis en place les différentes autorisations d'accès, nous avons créé une fonction lambda en utilisant le service **Lambda**. Cette fonction écrite en python a pour rôle de récupérer les données de S3 et les charger dans les tables de DynamoDB. L'image ci-dessous montre le code de la fonction utilisé pour charger les données du fichier marketing.csv.

```
1 import json
2 import csv
3 import boto3
4
5
6 def lambda_handler(event, context):
7     region = 'eu-west-3'
8     record_list= []
9     try:
10         s3 = boto3.client('s3')
11         dynamodb = boto3.client('dynamodb', region_name = region)
12         bucket= event['Records'][0]['s3']['bucket']['name']
13         key= event['Records'][0]['s3']['object']['key']
14         print('Bucket ', bucket, 'key ', key)
15
16         csv_file = s3.get_object(Bucket = bucket, Key = key)
17         record_list = csv_file['Body'].read().decode('utf-8').split('\n')
18
19         csv_reader = csv.reader(record_list, delimiter=',')
20
21         for row in csv_reader:
22             age=row[0]
23             sexe=row[1]
24             taux=row[2]
25             situationFamilliale=row[3]
26             nbEnfantsAcharge=row[4]
27             deuxiemvoiture=row[5]
28
29             print('Age', age, 'Sexe', sexe, 'taux',
30                   taux, 'situationFamilliale',situationFamilliale,
31                   'nbEnfantsAcharge', nbEnfantsAcharge, 'deuxiemvoiture', deuxiemvoiture)
32
33             add_to_db = dynamodb.put_item(
34                 TableName = 'marketing',
35                 Item = {
36                     'age' : {'N': str(age)},
37                     'sexe' : {'S': str(sexe)},
38                     'taux' : {'N': str(taux)},
39                     'situationFamilliale' : {'S': str(situationFamilliale)},
40                     'deuxiemvoiture' : {'S': str(deuxiemvoiture)}
41                 })
42
43             print('success insert')
44
45     except Exception as e:
46         print(str(e))
47
48     return {
49         'statusCode': 200,
50         'body': json.dumps('csv to DynamoDB success')}
51 }
```

5. Création des tables dans DynamoDB : Après avoir écrit nos fonctions lambda, nous avons créé les tables dans DynamoDB en utilisant le service **Table** de la console **AWS**, et exécuter les fonctions pour charger les données dans nos tables



The screenshot shows the AWS Management Console interface for DynamoDB. The left sidebar contains navigation links: Dashboard, Tables (selected), Backups, Exports to S3, Item explorer, PartiQL editor, DAX, and Revert to the current console. The main content area displays the 'Tables (2)' page. At the top, there's a search bar and a 'Create table' button. Below, a table lists the existing DynamoDB tables:

	Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode
<input type="checkbox"/>	catalogue	Active	Id (Number)	-	0	Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	marketing	Active	age (Number)	-	0	Provisioned (1)	Provisioned (1)

The screenshot shows the AWS DynamoDB console interface. On the left, the 'Table group' is set to 'Any group' and the 'marketing' table is selected. The main panel displays 'General information' for the 'marketing' table. The partition key is 'age (Number)' and the sort key is '-'. The capacity mode is 'Provisioned'. The table status is 'Active' with 'No active alarms'. Below this, the 'Items summary' shows 17 items, a table size of 1,080 bytes, and an average item size of 63.53 bytes. The 'Items preview (17)' section shows a table with columns: age, deuxiem..., situation..., sexe, and taux. The preview shows three items with their respective values.

age	deuxiem...	situation...	sexe	taux
19	False	Celibataire	F	212
21	False	Celibataire	F	1596
22	True	En Couple	M	411

The screenshot shows the AWS DynamoDB console interface for the 'catalogue' table. The partition key is 'id (Number)' and the sort key is '-'. The capacity mode is 'Provisioned'. The table status is 'Active' with 'No active alarms'. The 'Items summary' shows 270 items, a table size of 29,232 bytes, and an average item size of 108.27 bytes. The 'Items preview (20)' section shows a table with columns: id, prix, puissance, longueur, occasion, marque, couleur, nbPortes, nom, and nbPlaces. The preview shows three items with their respective values.

id	prix	puissance	longueur	occasion	marque	couleur	nbPortes	nom	nbPlaces
7	50500	272	tres longue	False	Volvo	bleu	5	S80 T6	5
8	35350	272	tres longue	True	Volvo	rouge	5	S80 T6	5
47	22900	150	moyenne	False	Volkswagen	noir	5	Golf 2.0 FSI	5

2.1.2. Hadoop HIVE

A. INSERTION DES FICHIERS

Nous avons créé dans le système de fichiers HADOOP HDFS deux dossiers qui contiendront les fichiers que nous avons jugé assez volumineux afin de rendre plus rapide leur traitement.

- **Création des dossiers architecture 1 et 2 dans hadoop hdfs du serveur distant.**

- **Connexion au serveur**

```
ssh Nom_utilisateur@134.59.152.111 -p 443  
Mot_de_passe_utilisateur
```

- **Création du dossier BigDataProject2020Groupe4**

```
hdfs dfs -mkdir /bigDataProject2020Groupe4
```

- **Création du dossier Architecture1/ Architecture2 dans BigDataProject2020Groupe4**

```
hdfs dfs -mkdir /bigDataProject2020Groupe4/architecture1  
hdfs dfs -mkdir /bigDataProject2020Groupe4/architecture2.
```

Après avoir créé ces dossiers, nous avons procédé au déplacement des fichiers dans les dossiers du HDFS.

- **Déplacement des fichiers.csv du serveur vers les dossiers Architectures du BigDataProject2020Groupe4**

- **Fichier CO2.csv**

```
hdfs dfs -put CO2.csv /bigDataProject2020Groupe4/architecture2/
```

- **Fichier CLIENTS_11.csv**

```
hdfs dfs -put CLIENTS_11.csv /bigDataProject2020Groupe4/architecture1/
```

- **Fichier CLIENTS_3.csv**

```
hdfs dfs -put CLIENTS_3.csv /bigDataProject2020Groupe4/architecture1/
```

- **Fichier IMMATRICULATION.csv**

```
hdfs dfs -put IMMATRICULATION.csv /bigDataProject2020Groupe4/architecture1/
```

- **Fichier Catalogue.csv**

```
hdfs dfs -put catalogue.csv /bigDataProject2020Groupe4/architecture1/
```

B. CREATION DES TABLES EXTERNE APACHE HIVE POINTANT SUR LES DIFFERENTES TABLES DES BASES DE DONNEES.

Afin de faire communiquer nos tables externes avec nos tables internes, nous avons utilisé APACHE HIVE qui est le logiciel d'entrepôt de données facilitant la lecture, l'écriture et la gestion des grands ensembles de données résidant dans un stockage distribué à l'aide de SQL.

- **Connexion au serveur**

```
ssh Nom_utilisateur@134.59.152.111 -p 443  
Mot de passe utilisateur
```

- **Connexion à Beeline**

```
beeline  
!connect jdbc:hive2://localhost:1000
```

- **Création des différentes tables externes HIVE**



- Table IMMATRICULATION_groupe4_ONS_H_EXT qui pointe sur le fichier hdfs Immatriculation.csv

```
CREATE EXTERNAL TABLE IMMATRICULATION_Groupe4_ONS_H_EXT (  
  IMMATRICULATION      STRING,  
  MARQUE                STRING,  
  NOM                  STRING,  
  PUISSANCE             INT,  
  LONGUEUR              INT,  
  NB_PLACES             INT,  
  NB_PORTES             INT,  
  COULEUR               STRING,  
  OCCASION              BOOLEAN,  
  PRIX                  INT)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
STORED AS TEXTFILE LOCATION 'hdfs://bigDataProject2020Groupe4/a1/immatriculation' ;
```

- Table CLIENTS_11_Groupe4_ONS_H_EXT qui pointe sur la table mongoDB CLIENTS_11_Groupe4

```
drop table CLIENTS_11_Groupe4_ONS_H_EXT;  
CREATE EXTERNAL TABLE CLIENTS_11_Groupe4_ONS_H_EXT (  
  id          INT,  
  age         INT,  
  sexe        STRING,  
  taux        INT,  
  situationFamiliare STRING,  
  nbEnfantsAcharge INT,  
  deuxiemeVoiture STRING,  
  immatriculation STRING)  
STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'  
WITH SERDEPROPERTIES('mongo.columns.mapping'='{ "id": "_id", "age": "age", "sexe": "sexe", "taux": "taux",  
  "situationFamiliare": "situationFamiliare", "nbEnfantsAcharge": "nbEnfantsAcharge",  
  "deuxiemeVoiture": "deuxiemeVoiture", "immatriculation": "immatriculation" }', 'mongo.input.uri' = 'mongodb+srv')  
TBLPROPERTIES('mongo.properties.path' = 'mongodb://Allan:ALLAN2B20@dbclients.uc0w4.mongodb.net/ClientsGroupe4?retryWrites=true&w=majority')  
);  
select * from CLIENTS_11_Groupe4_ONS_H_EXT LIMIT 1;
```

- Table CLIENTS_3_Groupe4_ONS_H_EXT qui pointe sur un fichier hdfs CLIENT_3_GROUPE4

```
drop table CLIENTS_3_Groupe4_ONS_H_EXT;  
  
CREATE TABLE CLIENTS_3_Groupe4_ONS_H_EXT  
(  
  age NUMBER(11),  
  sexe varchar2(50),  
  taux number(11),  
  situationFamiliare varchar2(50),  
  nbEnfantsAcharge NUMBER(11),  
  deuxiemeVoiture varchar2(50),  
  immatriculation varchar2(50)  
)  
ORGANIZATION EXTERNAL (  
  TYPE ORACLE_HDFS DEFAULT DIRECTORY  
  ORACLE_BIGDATA_CONFIG  
  ACCESS PARAMETERS  
(  
    com.oracle.bigdata.fileformat:TEXTFILE  
    com.oracle.bigdata.overflow:{"action": "truncate"}  
    com.oracle.bigdata.erroropt:{"action": "setnull"}  
  )  
  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
  LOCATION ('hdfs://bigDataProject2020Groupe4/a2/Clients_3.csv')  
);
```


- Table CATALOGUE_Groupe4_ONS_H_EXT qui pointe sur la table MongoDB CATALOGUE _Groupe4

```
drop table CATALOGUE_Groupe4_ONS_H_EXT;
CREATE EXTERNAL TABLE CATALOGUE_Groupe4_ONS_H_EXT (
  marque      STRING,
  nom          STRING,
  puissance    INT,
  longueur     STRING,
  nbPlaces     INT,
  nbPortes     INT,
  couleur      STRING,
  occasion     STRING,
  prix         INT)
STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
WITH SERDEPROPERTIES('mongo.columns.mapping'='{ "marque": "marque", "nom": "nom", "puissance": "puissance",
"longueur": "longueur", "nbPlaces": "nbPlaces", "nbPortes": "nbPortes",
"couleur": "couleur", "occasion": "occasion", "prix": "prix" }')
TBLPROPERTIES('mongo.uri'='mongodb://127.0.0.1:27017/DB_Groupe4.Groupe4_Catalogue'
);
select * from CATALOGUE_Groupe4_ONS_H_EXT;
```

2.1.3 Oracle

Oracle Databases est un système de gestion de base de données relationnelle (SGBDR) qui depuis l'introduction du support du modèle objet dans sa version 8 peut être aussi qualifié de système de gestion de base de données relationnel-objet (SGBDRO).

Le but de notre travail consistait à utiliser des tables externes pour accéder aux données de sources hétérogènes (MongoDB, une 2ème BD NoSQL de votre choix, Hadoop HDFS, Oracle SQL).

Dans l'organisation de nos données, on avait décidé de mettre le fichier Clients 3.csv dans Oracle et d'importer les données dans la table interne. Le langage utilisé pour manipuler ces données dans Oracle est SQL3.

Clients_3.csv concerne les achats de l'années en cours

Attribut	Type	Description	Domaine de valeurs
Age	Numérique	Age en années du clients	[18, 84]
Sexe	Catégoriel	Genre de la personne	M, F
Taux	Numérique	Capacité d'endettement du client en euros (30% du salaire)	[544, 74185]
SituationFamiliiale	Catégoriel	Situation familiale du client	Célibataire, Divorcée, En Couple, Marié(e), Seul, Seule
NbEnfantsACharge	Numérique	Nombre d'enfants à charge	[0, 4]
2eme voiture	Booléen	Le client possède déjà un véhicule principal ?	true, false
Immatriculation	Caractères	Numéro unique d'immatriculation du véhicule	Texte au format « 9999 AA 99 »

a. CREATION DES TABLES ET INSERTIONS DES DONNEES DANS ORACLE

Pour pouvoir créer une table dans Oracle, il fallait au préalable :

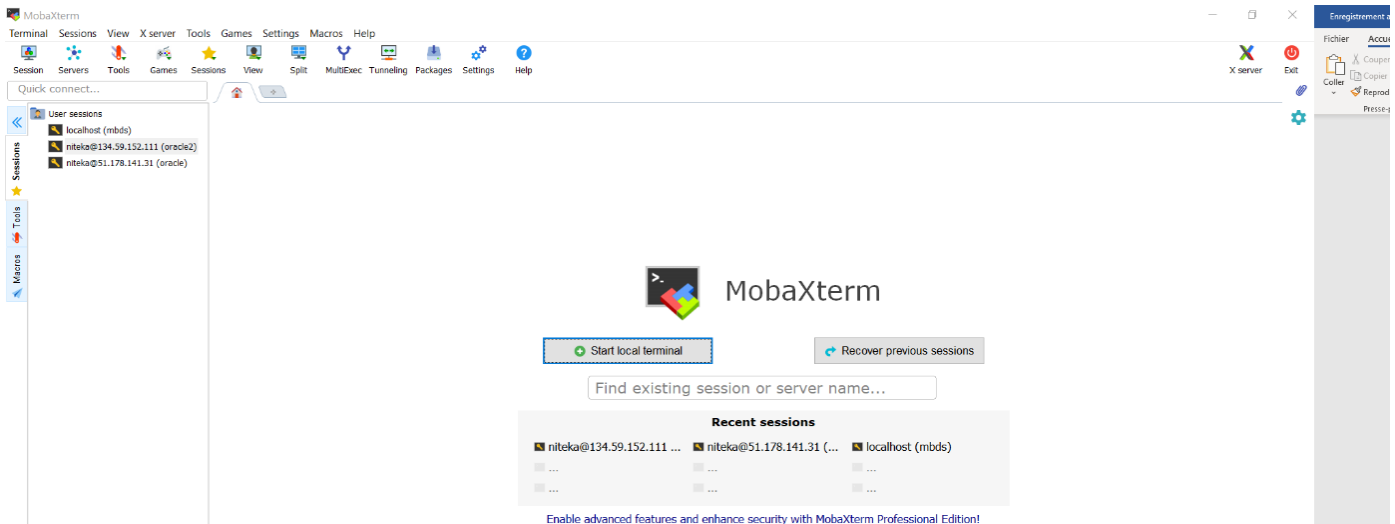
- Avoir une machine Oracle sur lequel on va travailler
- Et se connecter sur la machine pour créer les tables

Nous avons à notre disposition une machine Oracle installé sur un serveur distant de l'université où l'on pouvait se connecter via l'IP **134.59.152.111** pour créer et importer les données. Chaque étudiant avait ces propres identifiants qui lui permettait de se connecter et accéder sur cette machine.

b. CONNECTION SUR LA MACHINE

Ayant l'IP de la machine et les identifiants pour y accéder, nous avons utilisé **MobaXterm** qui est un émulateur de terminal Linux pour Windows. Cela nous a offert la possibilité de nous connecter à notre serveur distant via la commande ssh comme pour le logiciel PuTTY ; nous pouvons entrer des commandes UNIX dans un terminal sur notre session de travail.

Voici la page d'accueil de MobaXterm. En haut à gauche, nous avons la possibilité de créer une session ssh en spécifiant l'IP de la machine sur laquelle nous aimerions nous connecter et vous avez la possibilité de l'enregistrer.



Pour créer et insérer les données du fichiers Clients3 dans Oracle, il fallait :

1. Lancer 2 terminales dans MobaXterm
2. **Dans le premier terminal**, se connecter sur la machine : IP = 134.59.152.111
3. **Dans le premier terminal**, créer un dossier bigDataProject2020Groupe4/
`mkdir /bigDataProject2020Groupe4/`
4. Charger le fichier client_3.csv dans le dossier bigDataProject2020Groupe4/
5. Dans le 2ème terminal, Lancer les commandes
`sql> sqlplus username2B20@orcl/username2B2001`
`sql> connect username@orcl/motDepasse (exemple connect niteka@orcl/niteka14)`
6. **Dans le 2ème terminal**, Lancer la commande
`sql> CREATE TABLE CLIENTS_3_Groupe4 (`

<code>age</code>	<code>number(5),</code>
<code>sexe</code>	<code>varchar2(3),</code>
<code>taux</code>	<code>number(6),</code>
<code>situationFamiliare</code>	<code>varchar2(30),</code>
<code>nbEnfantsAcharge</code>	<code>number(10),</code>
<code>deuxiemeVoiture</code>	<code>varchar2(5),</code>
<code>immatriculation</code>	<code>varchar2(30)</code>
<code>);</code>	

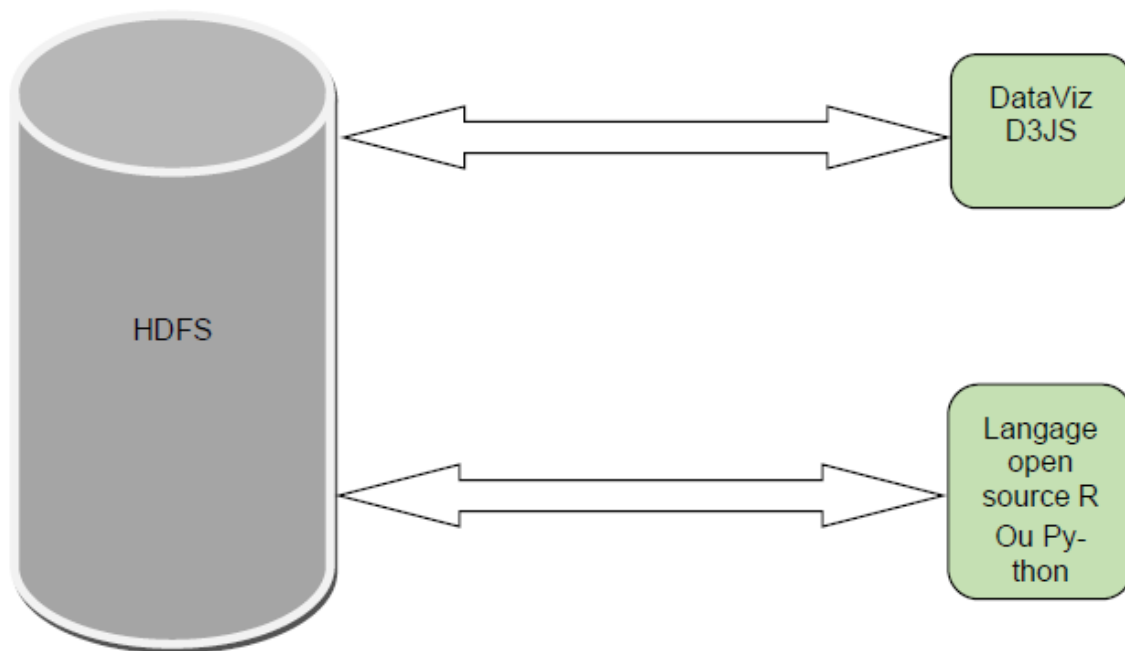
7. Dans le premier terminal, créer un fichier client_3.ctl, copier le script suivant :

```
LOAD DATA
INFILE 'Clients_3.csv'
INTO TABLE CLIENTS_3_Groupe4
FIELDS TERMINATED BY ','
(age,sexe,taux,situationFamiliare,nbEnfantsAcharge,deuxiemeVoiture,immatriculation)
```

8. Dans le premier terminal, lancer la commande pour importer les données
[niteka@bigdatalite bigDataProject2020Groupe4/]\$ *sqlldr NITEKA2B20@ORCL/NITEKA2B2001*
CONTROL=client_3.ctl LOG=client_3.log BAD=clients_3.bad

Une fois les données insérées, on peut faire nos requêtes SQL.

2.2. ARCHITECTURE 2 : HDFS HADOOP UNIQUEMENT SANS TABLES EXTERNES



Nous avons dans cette partie utiliser un script Shell permettant la création et l’insertion de nos données dans l’HDFS.

Chemin du script : `Projet_BIGDATA_S1_MBDS2020\Architecture2\Scripts\script1_insertion_hdfs.sh`

2.2.1. Nettoyage et préparation des données

Nous devons ici :

- ✓ Intégrer les informations du fichier « C02.csv » ("Bonus / Malus", "Rejets CO2 g/km", "Cout Energie") dans le fichier « Catalogue.csv »
- ✓ Calculer une valeur moyenne de ces 3 colonnes selon les marques des véhicules

Afin de pouvoir travailler correctement avec le fichier « C02.csv », nous avons homogénéisé les données qui y étaient stockés à l’aide d’un script Python en utilisant la bibliothèque **Pyspark**. Ce script effectue les opérations suivantes :

- ✓ Suppression du modèle dans la colonne "Marque / Modele"
- ✓ Suppression de caractères inutiles et conversion en entier des valeurs des colonnes "Bonus / Malus", "Rejets CO2 g/km", "Cout energie".

Chemin du script : `Projet_BIGDATA_S1_MBDS2020\Architecture2\Scripts\NettoyageCO2.py`

2.2.2. Adaptation et intégration du fichier C02

Pour permettre l'intégration du fichier « C02.csv » dans notre fichier « Catalogue.csv », nous avons décidé d'utiliser l'écosystème Spark (PySpark).

Pourquoi PySpark ? : Tout d'abord, ses outils SQL intégrés qui nous ont permis d'effectuer des requêtes simples afin de pouvoir manipuler nos dataframes (gestion de plusieurs de colonnes).

Ensuite, le projet nécessitant d'effectuer énormément de tests afin d'affiner nos résultats, sa rapidité a été un réel gain de temps.

Enfin, le côté fonctionnel de python nous a permis mettre en place un script demandant beaucoup moins de lignes de codes.

Etapes d'adaptation et d'intégration :

- Création et nettoyage de dataframe du fichier « CO2.csv »
- Réduction des marques équivalentes de manière en une seule marque en effectuant la moyennes de leur colonnes « Bonus / Malus », « Rejet CO2 » et « Cout Energie »
- Jointure des dataframes des fichier « Catalogue.csv » avec « CO2.csv » par la marque.
- Calcul des moyennes globales de nos colonnes « Bonus / Malus », « Rejet CO2 » et « Cout Energie »
- Remplissage des colonnes « Bonus / Malus », « Rejet CO2 » et « Cout Energie » du fichier « Catalogue.csv »

Chemin du script : `Projet_BIGDATA_S1_MBDS2020\Architecture2\Scripts\ MapReduce.py`

Chemin du fichier de sortie : `Projet_BIGDATA_S1_MBDS2020\Architecture2\Scripts\`

`Catalogue_co2.csv`



3. TECHNIQUES DE DATA VISUALISATION

3.1. CHAÎNE DE TRAITEMENT

Nous avons pu atteindre nos représentations finales grâce au pipeline suivant :

- Etape 1 : Nettoyage des données

Les données brutes ne respectant pas toujours le format usuel des spécifications des données (voir le sujet du projet), nous avons décidé d'appliquer les corrections suivantes afin d'obtenir des résultats harmonieux et d'éviter toute erreur qui serait liée à la syntaxe (dans notre cas, les caractères spéciaux) lors de nos requêtes effectuées à partir de la base de données.

- Liste des valeurs non définies :
 - - ? " " N/D - -1 = NaN
- Catlogue.csv et Immatriculation.csv :
 - Colonne "longueur" : très longue = tres longue
- Clients_n.csv :
 - Renommage nom de colonne "2eme voiture" en "deuxiemeVoiture"
 - Colonne "Sexe" : [Féminin, Femme] = F
[Masculin, Homme] = M
 - Colonne "situationFamiliiale" : [Seule, Seul, Célibataire] = Celibataire
Divorcée = Divorce
Marié(e) = Marie
- Marketing.csv :
 - Renommage nom de colonne "2eme voiture" en "deuxiemeVoiture"
 - Colonne "situationFamiliiale" : Célibataire = Celibataire

Ce nettoyage s'effectue grâce à un script python « **Nettoyage.py** » en utilisant la bibliothèque de manipulation de donnée, Pandas.

Exécution du script dans un Shell : **sh script_1_init.sh**

- Etape 2 : Traitement et filtrage des données

Nous effectuons ensuite dans R un filtrage afin de récupérer les variables que nous voulons représenter pour nos différentes visualisations.

Les variables internes qui nous avaient semblé intéressantes sont :

- Age, Situation Familiale, Taux nous permettant de savoir quel véhicule sera acheté
- Marque, Prix qui sont les résultantes des variables ci-dessus

- **Etape 3 : Choix du type de visualisation selon l'ensemble des variables**

TYPE DE VISUALISATION	VARIABLES	REPRESENTATION
Histogramme	situationFamiliale marque Achats	Le Nombre de voiture achetées par marques selon la situation familiale
Camembert	Sexe nbPortes Achats	Le nombre d'achat de véhicule à 5 portes selon le sexe
Ligne	age NombreClients PrixMoyen PrixMoyenMarque	Le prix moyen des véhicules achetés par les clients selon l'âge
Nuage de point	age PrixMoyen NombreClients	Le nombre de clients total et prix moyens des véhicules achetés selon l'âge
Nuage de texte	marque TauxMoyen NombreMarque	Le nombre de voiture vendu par prix selon le taux moyen du client

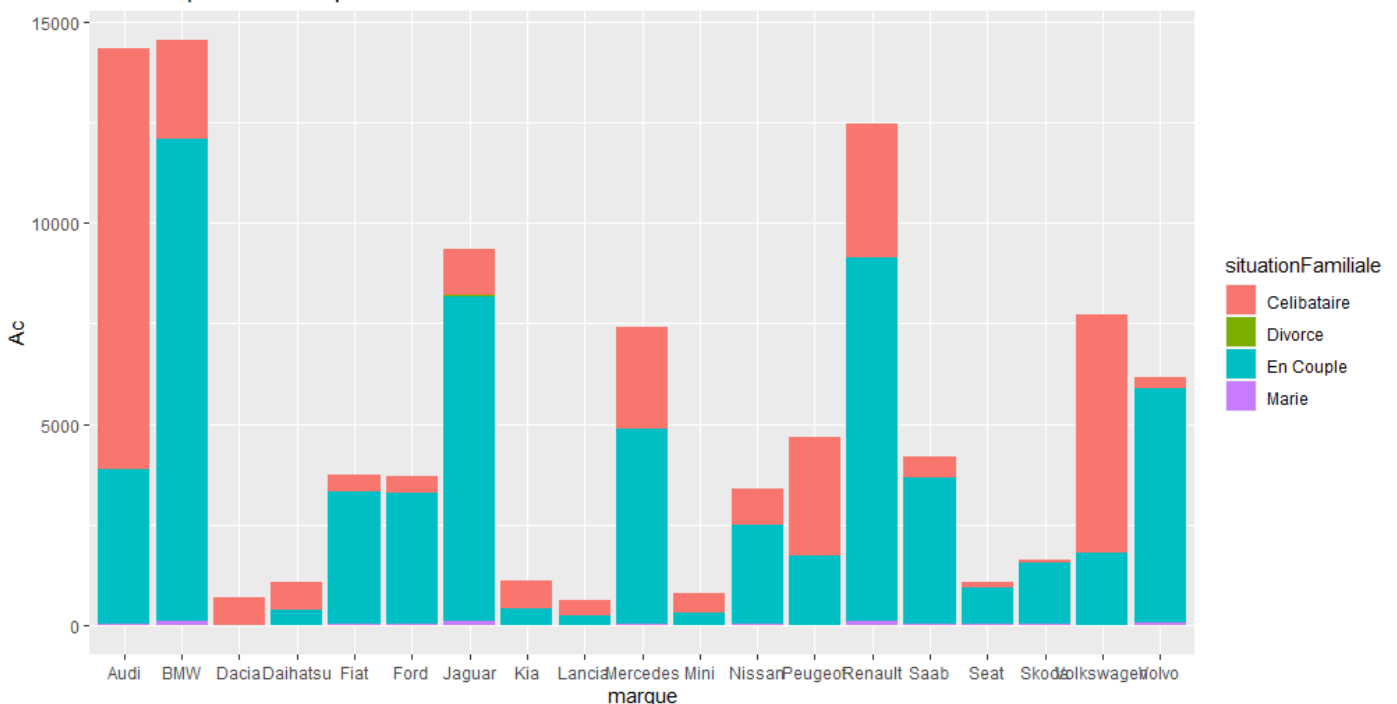
Parmi ces 5 types de visualisations, le nuage de points ainsi que l'histogramme nous ont semblé les plus appropriés afin d'avoir un visuel direct sur les informations pertinentes que l'on veut extraire pour les utilisateurs.

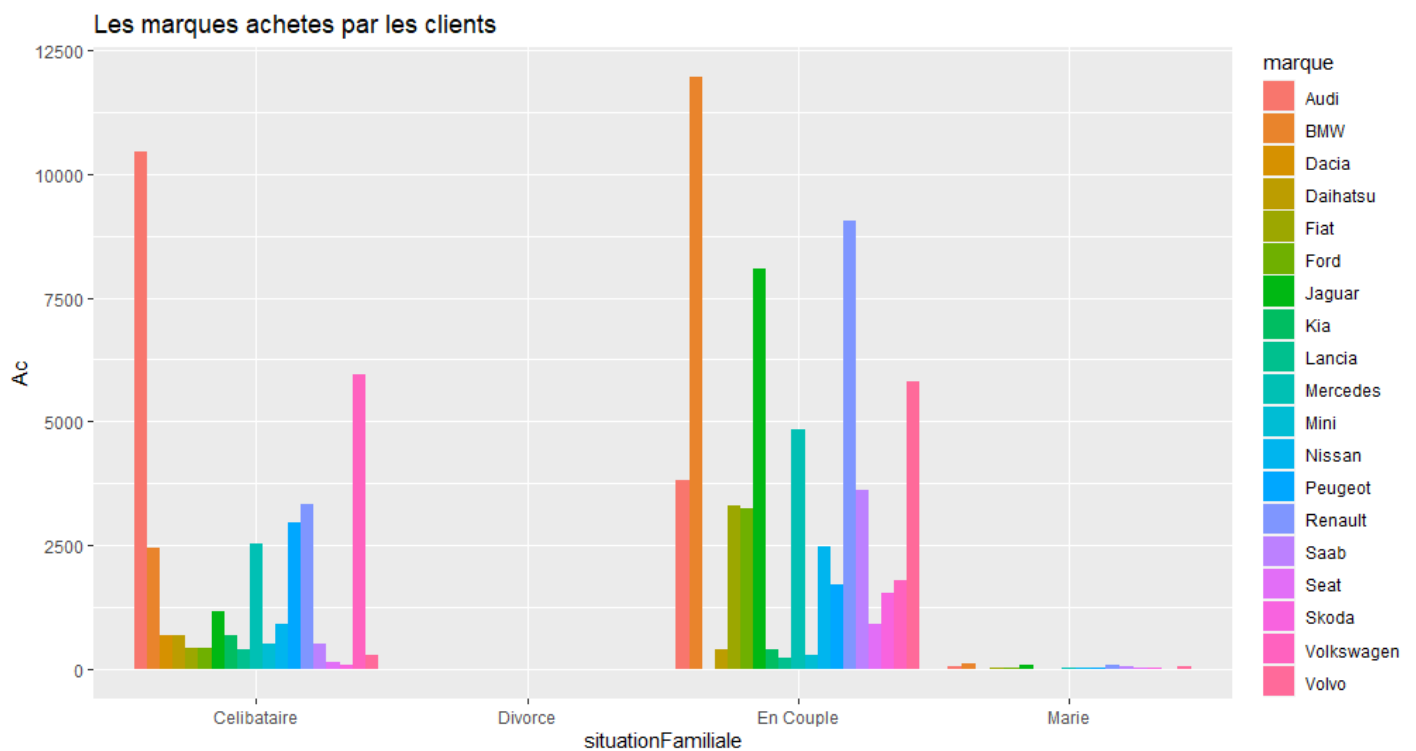
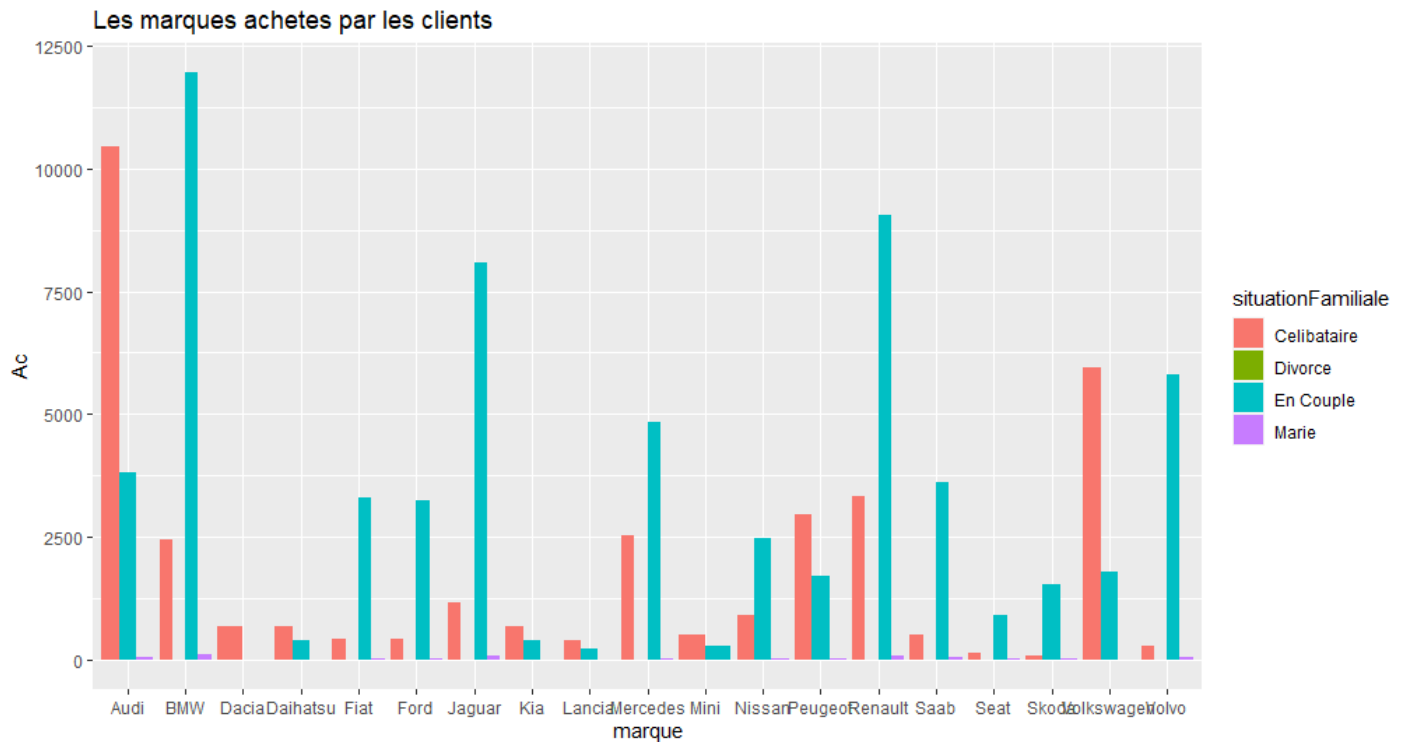
- **Etape 4 : Rendu**

Nous avons effectué différents rendus avec R et D3JS

R

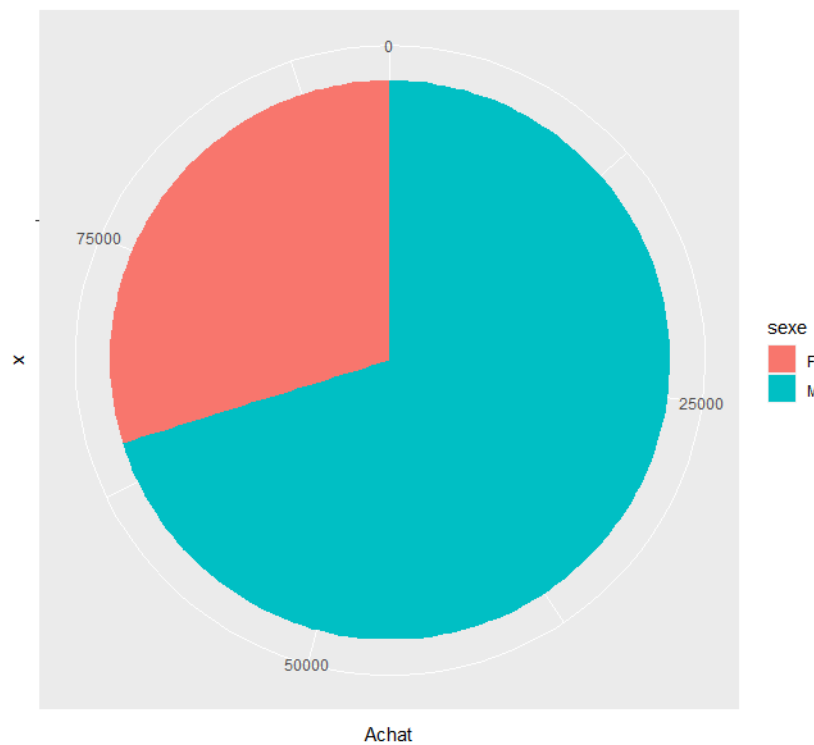
- **Histogramme : Nombre de marque de véhicules acheté par les clients selon la situation familiale**
Les marques achetées par les clients



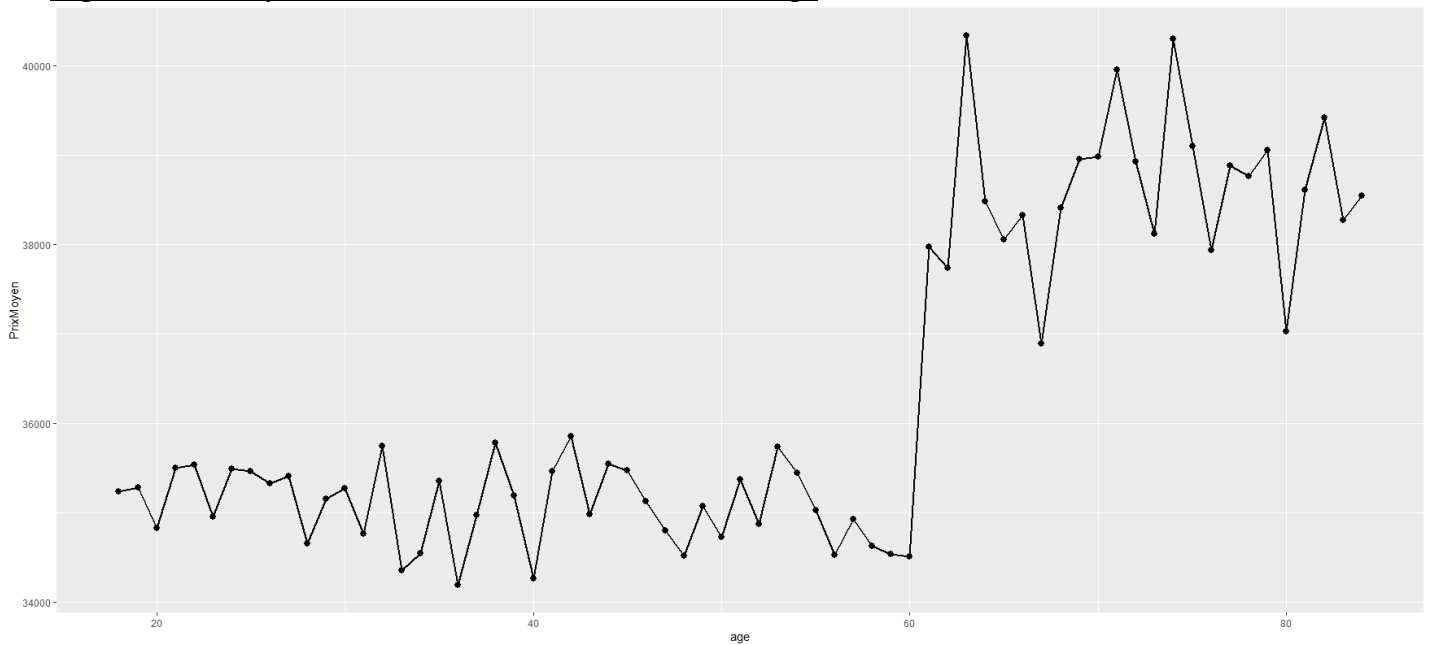


Les 2 premiers histogrammes restent les plus faciles à visualiser

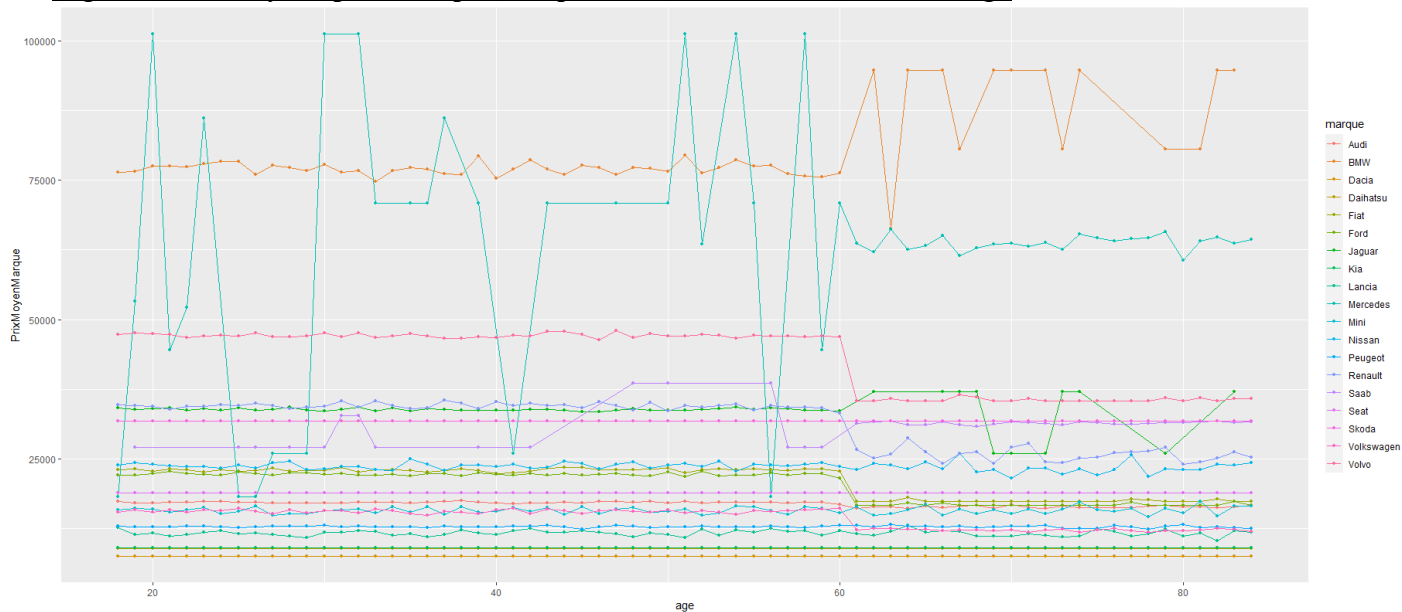
- Camembert : Proportions des véhicules à 5 portes achetés selon le sexe



Lignes : Prix moyen total des véhicules achetés selon l'âge

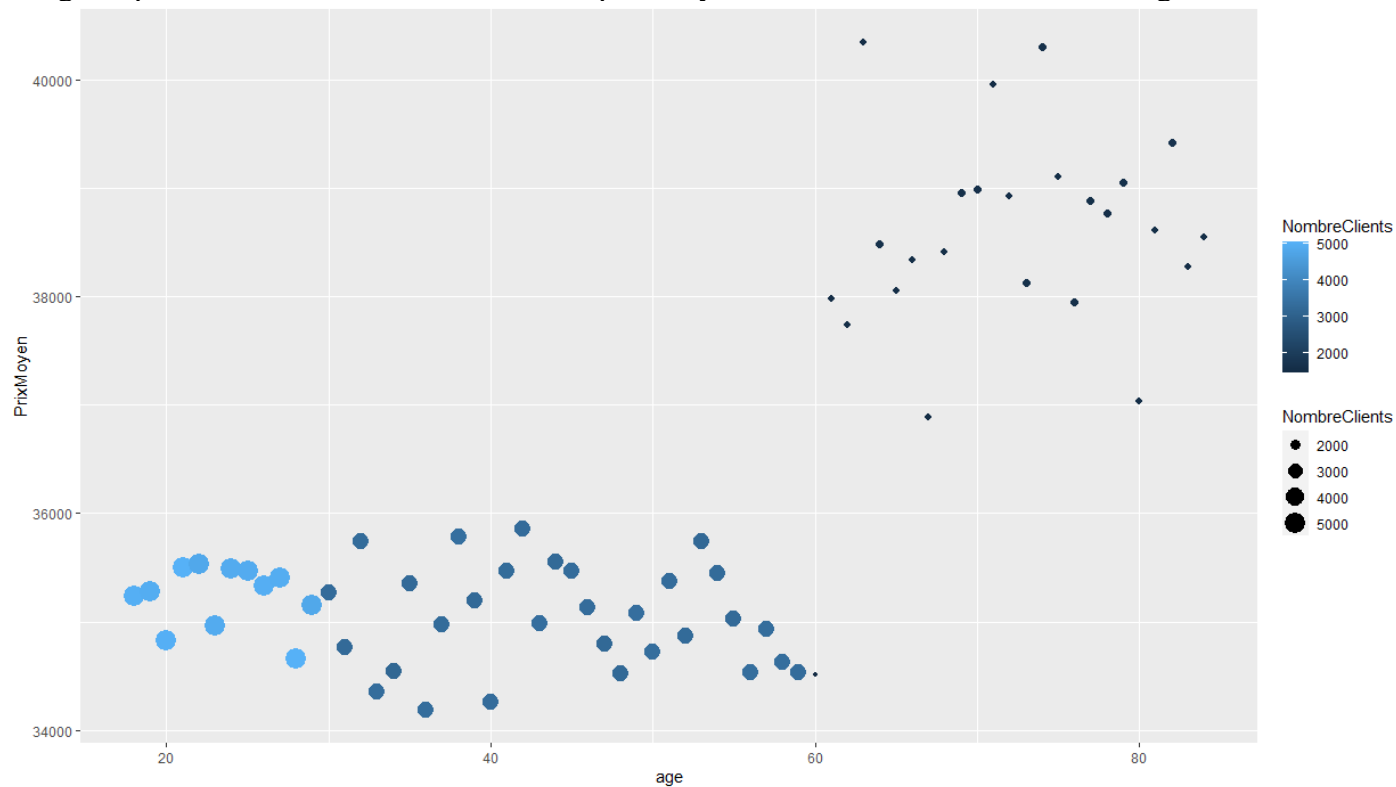


Lignes : Prix moyens pour chaque marque de véhicules achetés selon l'âge



Avec R, ce graphe devient illisible, nous verrons par la suite une implémentation dans D3JS améliorant ce rendu et les informations pouvant en être retirés.

Nuage de point : Le nombre de clients total et prix moyens des véhicules achetés selon l'âge

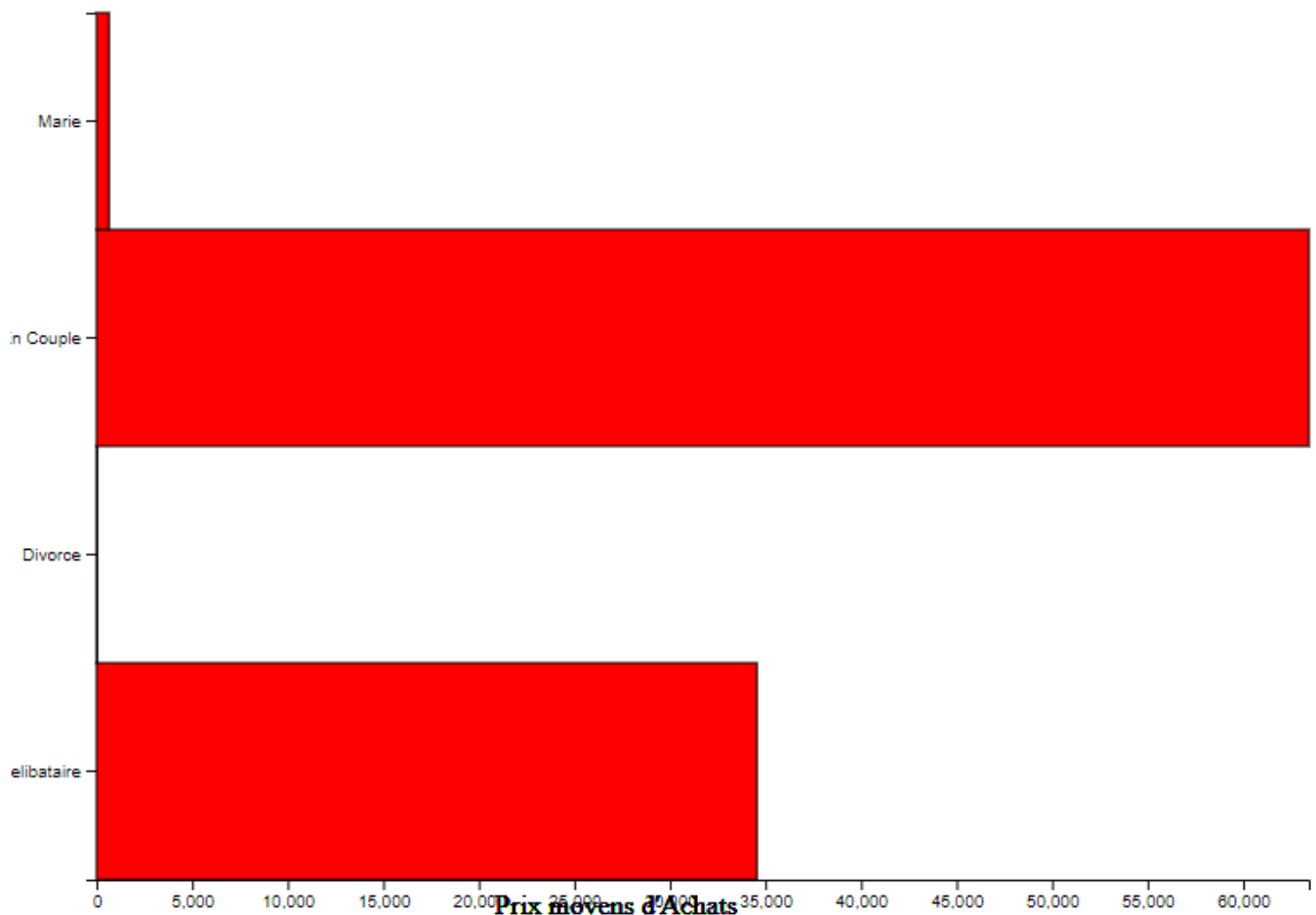


Nuage de texte : Le nombre de voiture vendu par prix selon le taux moyen du client



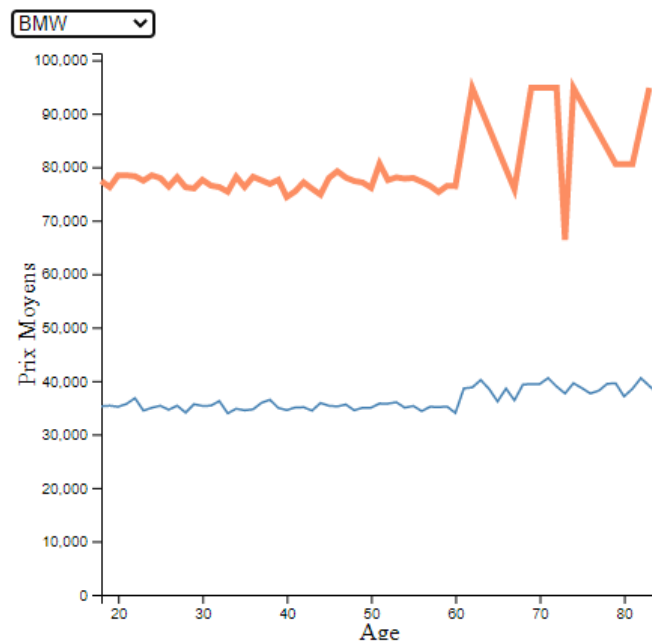
D3JS : Avant eu des petits soucis sur certain graphe en locale, nous vous fournissons les liens pour la visualisation sur JsFiddle

Histogramme : Prix moyens d'achats de véhicules selon la situation familiale (dynamique) [\[Lien\]](#)

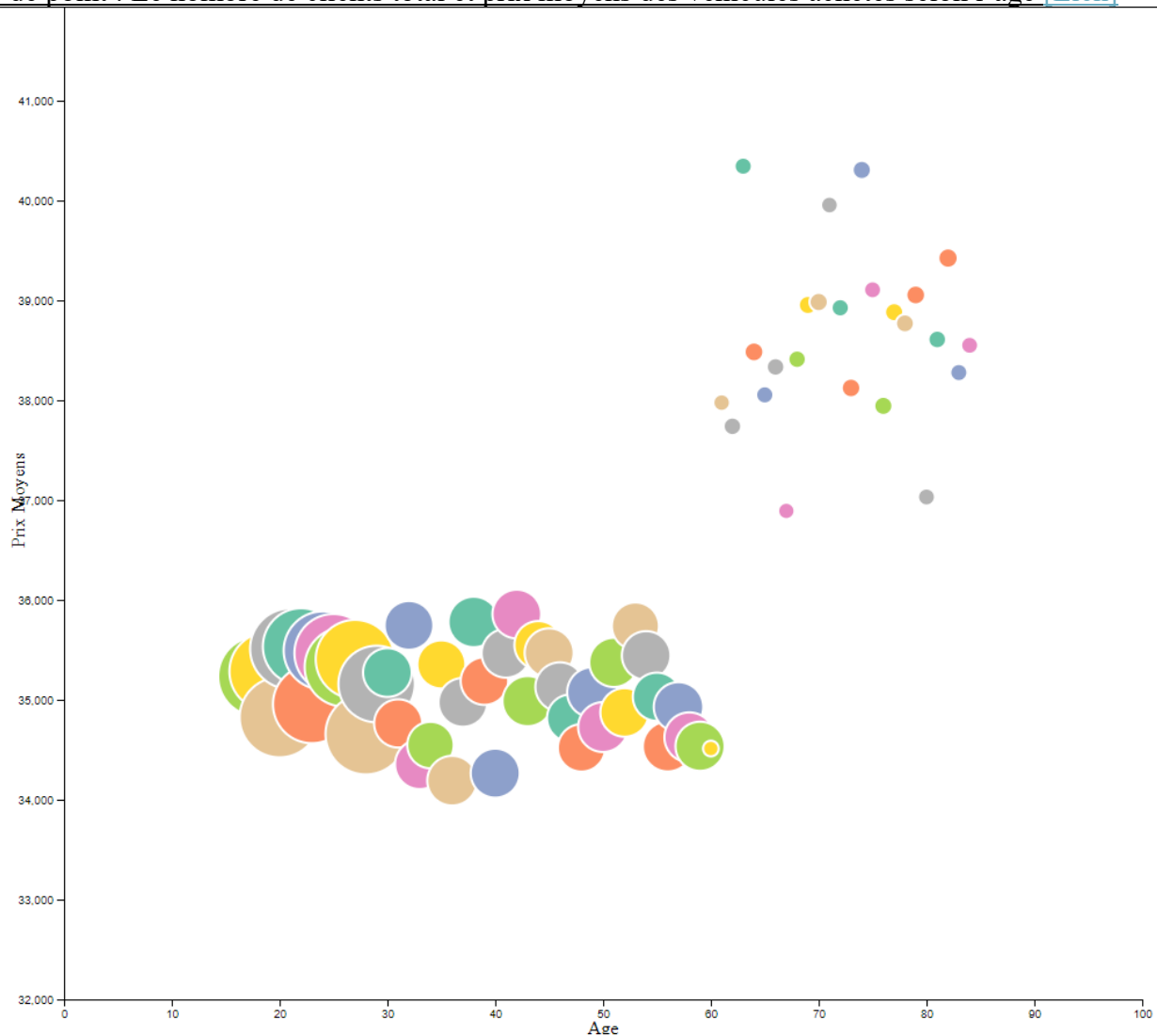


- ☒ Celibataire
- ☒ Divorce
- ☒ En Couple
- ☒ Marie

Lignes : Le prix moyen total des véhicules achetés et prix moyen par marque de véhicules acheté par les clients selon l'âge (dynamique) [\[Lien\]](#)



Nuage de point : Le nombre de clients total et prix moyens des véhicules achetés selon l'âge [\[Lien\]](#)



3.2. LES ACTEURS VISES ET OBJECTIFS DE VISUALISATION

Les acteurs principaux sont les concessionnaires d'automobiles, qui pourraient nous fournir un catalogue de données similaire à ceux utilisés dans notre projet, afin de ressortir les informations les plus pertinentes. Le but étant de pouvoir définir visuellement selon un type de client donné, le nombre de véhicule qui ont été acheté du catalogue. Ceci pourrait par la suite permettre aux concessionnaires de s'orienter vers une clientèle plus précise en fournissant de nouveaux catalogues de véhicule plus susceptible d'être vendu.

Cependant, il peut néanmoins servir à des statisticiens afin d'analyser par exemple la cohérence et la tendance des données d'un concessionnaire à un autre, selon leur pays, le type de ville, la démographie etc...

4. ANALYSE DES DONNEES PAR LES TECHNIQUES DE DATA MINING, MACHINE LEARNING ET DEEP LEARNING

Nous voici dans la dernière partie du projet, l'analyse des données. Dans cette partie nous allons manipuler nos données afin de pouvoir prédire, à partir d'un client donné, le type de voiture le plus susceptible de vendre.

Nous expliquerons les étapes suivies pour arriver à des taux de succès à plus de **90%** et nous parlerons de :

- Nos choix de gestion d'analyse
- Les processus suivis
- Les modèles utilisés
- Les résultats obtenus pour les clients du fichier marketing.csv

Tous les scripts se trouvent dans le dossier Analyse/Code. Ce sont des notebook commentés facilitant la compréhension du code.

Le fichier de prédiction se trouve dans le dossier Analyse/Résultats Prédiction

4.1. ANALYSE EXPLORATOIRE DES DONNEES

Pour réaliser l'analyse, nous disposons des données suivantes :

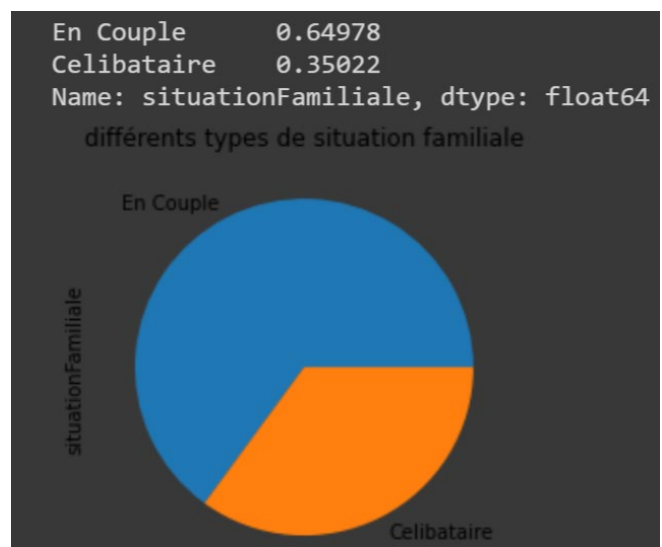
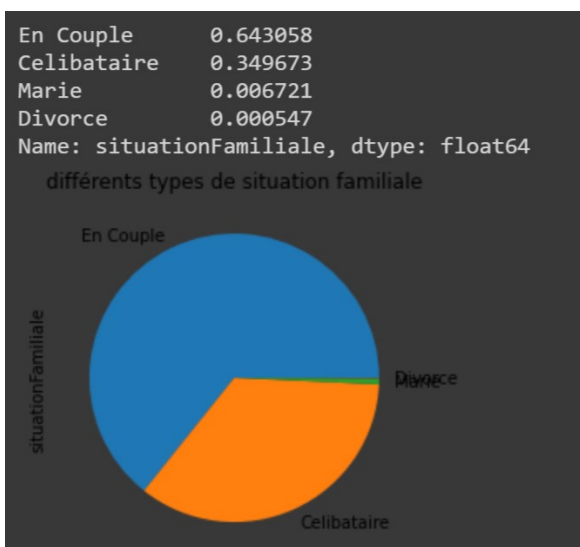
- Catalogue.csv
- Client3.csv
- Client11.csv
- Immatriculation.csv
-

Nous avons effectué une analyse exploratoire de données afin de les rendre propres et utilisables. Celle-ci s'est faite en deux temps :

- a. Nettoyage : (cf. [Etape 1 : Nettoyage des données](#))
- b. Analyse exploratoire : En visualisant les données, on s'est aperçu qu'il manquait encore du nettoyage à réaliser. On s'est aperçu que le fichier "Immatriculation.csv" contenait 3368 duplications, la concaténation des fichiers "Client3" et "Client11" contenait 39 duplications. Nous les avons donc retirées. Nous avons aussi remarqué que le type **Marie** et **Divorce** sont très faiblement utilisés, pour cela nous avons décidé de les transformer respectivement en **En Couple** et **Célibataire**.

Ci-dessous, des illustrations de la proportion de la situation familiale des clients :

- À gauche illustre les différents types de situation familiale AVANT transformation
- À droite illustre les différents types de situation familiale APRÈS transformation





4.2. DÉTECTION DES TYPES DES VOITURES

Une de nos plus grandes difficultés dans la partie analyse était de trouver les bons types de voitures pour définir nos classes de prédictions. Pour cela, nous avons utilisé deux approches :

- La première consistait à définir nous même une classe pour chaque voiture en cherchant les informations sur internet
- La seconde consistait à utiliser un algorithme de clustering (K-means) pour nous aider à mieux classer les voitures ; celle-ci nous donna de meilleurs résultats

4.2.1. Première approche

Sur cette approche, nous avons ajouté un type aux 32 véhicules du catalogue en nous aidant d'internet. Nous avons ajouté les 8 types suivants :

- Berline 5 portes
- Limousine
- Citadines
- Sport
- Break
- Monospaces
- Berline Compacte
- Berline 3 portes

Nous avons alors entraîné nos clients sur ces variables de classe et avons obtenu un taux de succès global de **75%** avec la classification par arbre de décision.

A première vue, les résultats étaient plutôt satisfaisants, mais en analysant de plus près ces résultats, nous nous sommes aperçus que nos résultats n'étaient pas si satisfaisants. En effet, ce score s'explique du fait qu'une classe dominait sur les autres, celle-ci avec un rappel de 98% représentait à elle seule plus de 60% des données de prédiction. Son score influençait beaucoup sur le score total.

Ci-dessous la matrice de confusion avec le rappel de chaque variable (dernière colonne) et la précision de chaque variable (dernière ligne).

	Berline 5 portes	Limousine	Citadines	Sport	Break	Monospaces	Berline Compacte	Berline 3 portes	Rappel (%)
Berline 5 portes	38890.000000	35.000000	380.000000	2.000000	0.0	0.0	0.0	222.000000	98.383465
Limousine	1206.000000	1900.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	61.171925
Citadines	485.000000	0.000000	829.000000	1.000000	0.0	0.0	0.0	0.000000	63.041825
Sport	4739.000000	0.000000	0.000000	2769.000000	0.0	0.0	0.0	0.000000	36.880661
Break	234.000000	0.000000	163.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000
Monospaces	2197.000000	0.000000	397.000000	0.000000	0.0	0.0	0.0	243.000000	0.000000
Berline Compacte	1382.000000	0.000000	159.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000
Berline 3 portes	2130.000000	0.000000	382.000000	0.000000	0.0	0.0	0.0	246.000000	8.919507
Precision (%)	75.863683	98.191214	35.887446	99.891775	0.0	0.0	0.0	34.599156	NaN

Sur cette matrice nous remarquons 2 choses :

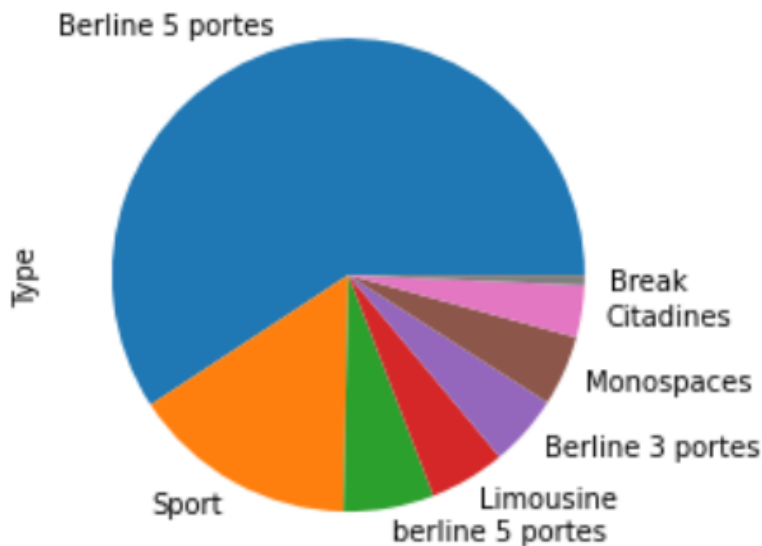
1. Les variables "Break", "Monospaces" et "Berline Compacte" ont un rappel et un score nul ; on peut donc en déduire que ces variables ont été mal choisies ; la classification des classes doit être refaite
2. La première colonne "Berline 5 portes" contient de nombreuses erreurs de prédictions ; quasiment toutes les classes sont prédites dans "Berline 5 portes", pourtant sa précision de 76% ; on en déduit donc que "Berline 5 portes" représente la grande majorité de nos données d'entraînement

Ci-dessous, la répartition du type des voitures achetées par les clients.

Visualisation du type de voiture acheté

```
Berline 5 portes : 1182791 = 59.24 %  
berline 5 portes : 123615 = 6.19 %  
Monospaces : 96037 = 4.81 %  
Sport : 308151 = 15.43 %  
Berline 3 portes : 97691 = 4.89 %  
Citadines : 70810 = 3.55 %  
Limousine : 103583 = 5.19 %  
Break : 13954 = 0.70 %  
<matplotlib.axes._subplots.AxesSubplot at 0x7ff9c4b82518>
```

Proportion du type de voitures achetés



À la suite de ces fortes incohérences, nous avons compris que les variables de prédiction qu'on avait choisies étaient mal distribuées. Pour cela, nous avons décidé de passer à la deuxième approche : l'utilisation d'un algorithme de clustering pour nous aider à mieux trouver nos variables de classe.

4.2.2. Clustering avec l'algorithme des K-means

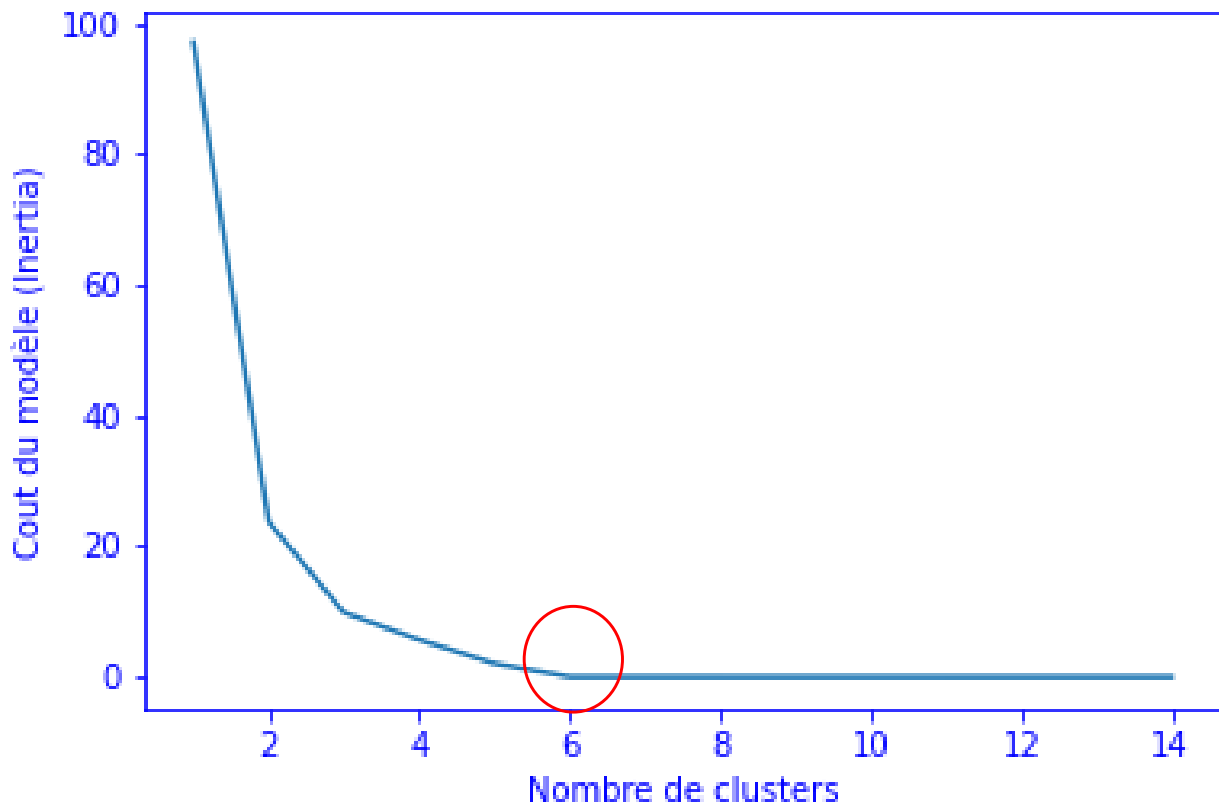
K-means est un algorithme de classification non supervisé par partitionnement pour des données à multiples dimensions. Nous l'avons utilisé pour regrouper les voitures du catalogue en fonction de leur taille, puissance, prix, nombre de place et nombre de portes.

Pour que l'algorithme soit efficace, nous avons uniformisé les colonnes afin qu'aucune d'entre elles ne domine sur une autre. Nous avons utilisé la fonction "MinMaxScaler" qui réduit les valeurs à des valeurs comprises entre 0 et 1 (le minimum à 0 et le maximum à 1).

Notre choix de 6 clusters ?

Pour faire du clustering avec la méthode des K-means, nous devons d'abord définir le nombre de clusters. Pour nous aider dans notre décision, nous avons choisi de visualiser l'inertie du modèle en fonction du nombre de clusters de 1 à 10.

L'idée est de trouver le point ayant le moins de clusters possible avec le moins d'inertie.



Sur la figure ci-dessus on remarque que 6 serait un bon choix pour le nombre de clusters ; nous avons donc appliqué l'algorithme avec le nombre de 6 clusters.

Les 6 clusters par l'algorithme des K-means

K-means nous a proposé un découpage qui nous a donné de bons résultats. Mais pour améliorer nos résultats nous avons changé certaines voitures de classe (voir dans le code) ; ainsi voici nos résultats finaux de groupe de véhicules.

Num Cluster	Type véhicule	puissance	prix neuf	nb voitures
0	Ville / courte - 1ere gamme	55 - 115	7500 - 16450	7
1	Ville / courte - 2eme gamme	90 - 150	18310 - 35800	8
2	Monospace	103 - 150	16000 - 27000	4
3	Route 1ere Gamme	102 - 197	18880 - 27300	7
4	Route 2eme Gamme	193 - 272	30000 - 50500	4
5	Haut de gamme	306 - 507	94800 - 101300	2

Le fichier « Analyse\Code\new_data\catalogue_kmeans_final.csv » contient la colonne des types de véhicule prédits (predicted) par l'algorithme de Kmeans.

Ci-dessous, la nouvelle répartition du type des voitures achetées par les clients.

Visualisation du type de voiture acheté

Haut de gamme : 177150 = 18.02 %

Route 1ere gamme : 134740 = 13.70 %

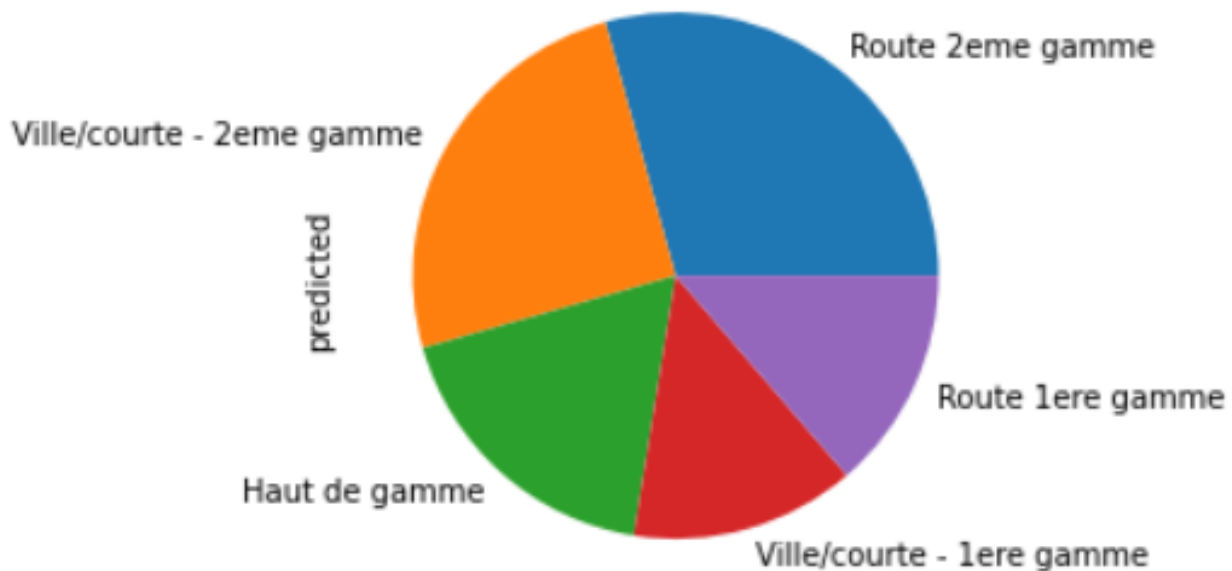
Ville/courte - 2eme gamme : 247015 = 25.12 %

Ville/courte - 1ere gamme : 136090 = 13.84 %

Route 2eme gamme : 288180 = 29.31 %

<matplotlib.axes._subplots.AxesSubplot at 0x7f8c75cc0198>

Proportion du type de voitures achetées



a. LES DONNEES D'ENTRAINEMENT

Le but est d'obtenir pour chaque ligne du fichier Immatriculation.csv, les données clientes de l'acheteur du véhicule (l'âge, le taux, situation familiale ...) ainsi que le type de véhicule qu'on a défini précédemment grâce à K-means, associé au véhicule de l'immatriculation.

Pour cela, nous avons mergé le dataframe du fichier "Catalogue_predicted.csv" (catalogue avec une colonne predicted qui est notre variable de classe avec la méthode des Kmeans) vers Immatriculation ; ainsi Immatriculation contenait la colonne "predicted" en plus qui est en effet le type de véhicule.

Nous avons ensuite fait un left-join entre Client et Immatriculation avec leur colonne commune "immatriculation" et avons obtenu toutes les données des clients avec les données de leur voiture. Notre fichier final contenait 196 635 lignes et 17 colonnes : immatriculation, age, sexe, taux, situationFamiliale, nbEnfantsAcharge, DeuxiemeVoiture, marque, nom, puissance, longueur, nbPlaces, nbPortes, couleur occasion, prix et predicted.

b. UNIFORMISATION DES DONNEES

Nous avons utilisé ici aussi MinMaxScaler pour rendre les données de certaines colonnes numériques uniformes entre elles afin qu'un critère ne prédomine pas sur un autre. Ceci a amélioré notre taux de succès de 4%.

c. ENTRAINEMENTS ET RESULTATS FINAUX

Pour des raisons d'optimisation, nous avons séparé les clients ayant acheté des voitures neuves aux clients ayant acheté des voitures d'occasion. De ce fait nous entraînons séparément ces 2 set de données afin que pour chaque client, l'on puisse savoir le type de voiture neuve et le type de voiture d'occasion qu'il serait susceptible d'acheter.

Nous avons donc atteint un meilleur score de **90.97 %** de taux de succès avec le modèle. Ce choix nous a permis d'augmenter de 8% notre taux de succès.

Ci-dessous les taux de succès des voitures neuves et voitures d'occasions avec les différents modèles utilisés.

Modèle de classification sup	Taux de succès voiture neuve (%)	Taux de succès voiture occasion (%)
Arbre de décision	80.92	80.97
Random Forest	81.06	81.65
Bayes	48.92	51.37
KNeighbors	90.26	90.97
Réseau de neurones	81.38	79.53

Nous avons donc atteint un meilleur score de **90.97 %** de taux de succès avec le modèle **KNeighbors**. Nous avons donc choisi ce modèle pour prédire le fichier marketing.

Ci-dessous, le taux de succès global de toutes les voitures (neuves + occasions). On remarque que les résultats sont inférieurs de 5 à 11%.

Modèle de classification sup	Taux de succès global des voitures neuves et d'occasions (%)
Arbre de décision	72.75
Random Forest	72.79
Bayes	49.59
KNeighbors	84.77



4.2.3. Matrices de confusion du modèle KNeighbors

On a rajouté une colonne rappel et une ligne précision qui indiquent en pourcentage le **rappel** et la **précision** de chaque variable.

Pour les voitures NEUVES

	VilleCourte1G	VilleCourte2G	Route1G	Route2G	Haut de gamme	Rappel (%)
VilleCourte1G	30707.000000	3069.000000	10.000000	13.000000	6.000000	90.835675
VilleCourte2G	2965.000000	43974.000000	14.000000	11.000000	14.000000	93.605517
Route1G	3.000000	14.000000	26885.000000	3887.000000	0.000000	87.320147
Route2G	5.000000	9.000000	4257.000000	54280.000000	2645.000000	88.698608
Haut de gamme	11.000000	2.000000	11.000000	2821.000000	27363.000000	90.581965
Precision (%)	91.143035	93.426532	86.233441	88.966105	91.12495	NaN

Pour les voitures d'OCCASIONS

	VilleCourte1G	VilleCourte2G	Route1G	Route2G	Haut de gamme	Rappel (%)
VilleCourte1G	6106.000000	1006.000000	0.000000	0.000000	0.000000	85.854893
VilleCourte2G	1019.000000	26004.000000	3.000000	3.000000	3.000000	96.197100
Route1G	5.000000	6.000000	8573.000000	965.000000	1.000000	89.769634
Route2G	0.000000	17.000000	1081.000000	22292.000000	1974.000000	87.888346
Haut de gamme	0.000000	9.000000	0.000000	2214.000000	20696.000000	90.300624
Precision (%)	85.638149	96.161527	88.774982	87.508833	91.276352	NaN



Prédiction du fichier marketing

	age	sexe	taux	situationFamiliale	nbEnfantsAcharge	DeuxiemeVoiture	Pred_Voiture_Neuve	Pred_Voiture_Occasion
0	21.0	F	1396.0	Celibataire	0.0	False	Ville/courte - 2eme gamme	Ville/courte - 2eme gamme
1	35.0	M	223.0	Celibataire	0.0	False	Ville/courte - 1ere gamme	Ville/courte - 1ere gamme
2	48.0	M	401.0	Celibataire	0.0	False	Ville/courte - 1ere gamme	Ville/courte - 2eme gamme
3	26.0	F	420.0	En Couple	3.0	True	Route 2eme gamme	Haut de gamme
4	80.0	M	530.0	En Couple	3.0	False	Haut de gamme	Haut de gamme
5	27.0	F	153.0	En Couple	2.0	False	Route 2eme gamme	Route 2eme gamme
6	59.0	F	572.0	En Couple	2.0	False	Route 2eme gamme	Haut de gamme
7	43.0	F	431.0	Celibataire	0.0	False	Ville/courte - 1ere gamme	Ville/courte - 2eme gamme
8	64.0	M	559.0	Celibataire	0.0	False	Ville/courte - 1ere gamme	Ville/courte - 2eme gamme
9	22.0	M	154.0	En Couple	1.0	False	Route 2eme gamme	Route 1ere gamme
10	79.0	F	981.0	En Couple	2.0	False	Haut de gamme	Haut de gamme
11	55.0	M	588.0	Celibataire	0.0	False	Ville/courte - 1ere gamme	Ville/courte - 2eme gamme
12	19.0	F	212.0	Celibataire	0.0	False	Ville/courte - 1ere gamme	Ville/courte - 2eme gamme
13	34.0	F	1112.0	En Couple	0.0	False	Route 2eme gamme	Route 2eme gamme
14	60.0	M	524.0	En Couple	0.0	True	Ville/courte - 1ere gamme	Ville/courte - 2eme gamme
15	22.0	M	411.0	En Couple	3.0	True	Route 2eme gamme	Haut de gamme
16	58.0	M	1192.0	En Couple	0.0	False	Route 2eme gamme	Ville/courte - 2eme gamme
17	54.0	F	452.0	En Couple	3.0	True	Route 2eme gamme	Haut de gamme
18	35.0	M	589.0	Celibataire	0.0	False	Ville/courte - 2eme gamme	Ville/courte - 2eme gamme
19	59.0	M	748.0	En Couple	0.0	True	Ville/courte - 2eme gamme	Ville/courte - 2eme gamme

5. PROBLEMES ET DIFFICULTES RENCONTRES

Durant ce projet, nous avons dû faire face à différents problèmes d'interconnexion notamment dans l'architecture 1.

5.1 MONGO ET DYNAMODB

Suivant le schéma, nous devions utiliser MongoDB comme un de nos SGBD NoSQL, cependant en raison de soucis de connexion externes liées au serveur distant (machine virtuelle fournie par l'université), nous n'avons pas pu mettre en place la connexion entre notre serveur et Mongo Atlas/MongoDB.

Ce problème étant d'ordre interne, la connexion avec DynamoDB, a également été impossible.

5.2 SOLUTIONS

Afin de passer outre ces soucis et d'avoir quand même une architecture respectant le cahier des charges, nous avons décidés de créer les tables et insérer les données qui devaient être initialement fait dans nos SGBD NoSQL, dans HIVE et Oracle.

6. PERSPECTIVES ET REFLEXIONS PERSONNELLES

Pour la continuité de ce projet, nous aurions aimé mettre en place :

Visualisation :

- Intégré du dynamisme et de l'interactivité dans nos visualisations
- Pouvoir échanger les données entre différents graphes
- Mise en place d'une interface web permettant à différents utilisateurs de pouvoir charger leurs propres données et leur proposer un ou plusieurs types de visualisations en fonction de leur critère (générique).

Application :

- Création d'une application de visualisation et d'analyse de données (pour des données sensibles par exemple avec un stockage en local)

CONCLUSION

Faisant en face à une grande émergence de données, beaucoup d'entreprises cherchent plusieurs façons pour assurer le stockage, la manipulation et l'analyse de celles-ci.

Afin de mettre en pratique la gestion et le stockage des données dans différentes base de données SQL et NoSQL, nous avons travaillé sur un projet qui nous a permis de créer notre propre écosystème avec trois bases de données NoSQL (MongoDB, DynamoDB, HIVE) et une base de données Oracle dans le but de pouvoir connecter et accéder via R et python à ces base de données.

Grâce à ce projet, nous avons tous d'abord appris à nous adapter aux différentes circonstances, assurer la mise en œuvre et le bon fonctionnement et la cohérence entre toutes ces bases de données, tout en répondant à la demande du client. Il nous a ensuite permis de définir un cheminement et d'évaluer la qualité des informations renvoyés par des représentations graphiques. Enfin, cela nous a offert l'occasion de bien nous familiariser avec un domaine de la Big Data qui devient de plus en plus présent en raison des nouveaux besoins émergeant et voulant décrire les liens entre les données ainsi que de les prédire, l'analyse des données.

Malgré les difficultés rencontrées, nous avons pu mettre en place d'autres alternatives afin de mener à bien ce projet.

Travailler en équipe a été une expérience enrichissante, car nous avons pu confronter nos différentes compétences, personnalités, idées ..., pour permettre la bonne réalisation du projet.

Cela nous a permis de mettre ensemble nos forces et faisant en sorte, via l'entraide, que nos faiblesses deviennent nos forces.

Comme on dit : "Expliquer à l'autre, t'aidera à mieux comprendre". Ceci était notre slogan.

REFERENCES

1. GESTION DES DONNEES

- <https://www.bmc.com/blogs/mongodb-compass/>
- <https://aws.amazon.com/fr/dynamodb/getting-started/>
- <https://sparkbyexamples.com/>
- <http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html?highlight=withcolumn>

2. VISUALISATION DES DONNEES

- <https://www.d3-graph-gallery.com/>
- <https://sites.google.com/site/rgraphiques/realiser-des-graphiques-avec-le-logiciel-r/les-graphiques>