## Objectif

Ce projet est destiné à une compagnie d'assurances qui propose des assurances-vie à ses clients.

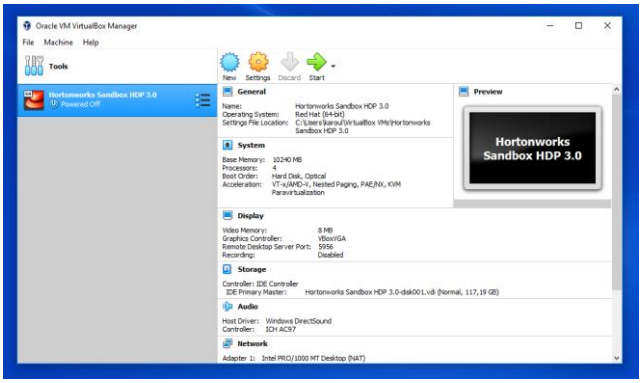
On doit leur fournir un outil permettant de prédire le niveau de risque lié à l'ouverture d'un nouveau contrat client.

On dispose d'un ensemble de données anciennes, où on a pour chaque client le niveau de risque correspondant. En se basant sur ces données ainsi que notre savoir-faire on va pouvoir prédire le niveau de risque associé à chaque nouveau client d'une manière automatique.

## Etapes

On commence par télécharger et installer une machine virtuelle Hadoop « HDP Sandbox 3.0.1 » sur « Oracle VM VirtualBox » local.

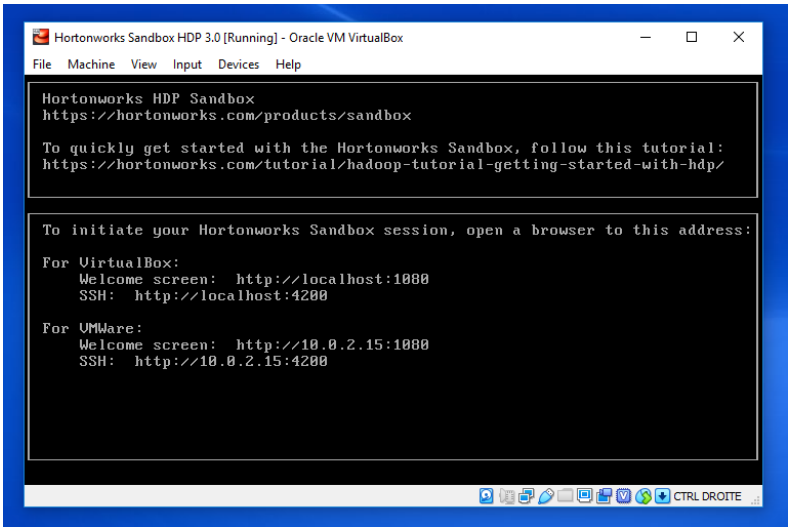
C'est une distribution toute prête d'où son volume de 20Go.




HDP est une plate-forme de données basée sur Hadoop qui contient les systèmes Hadoop Distributed File System (HDFS).

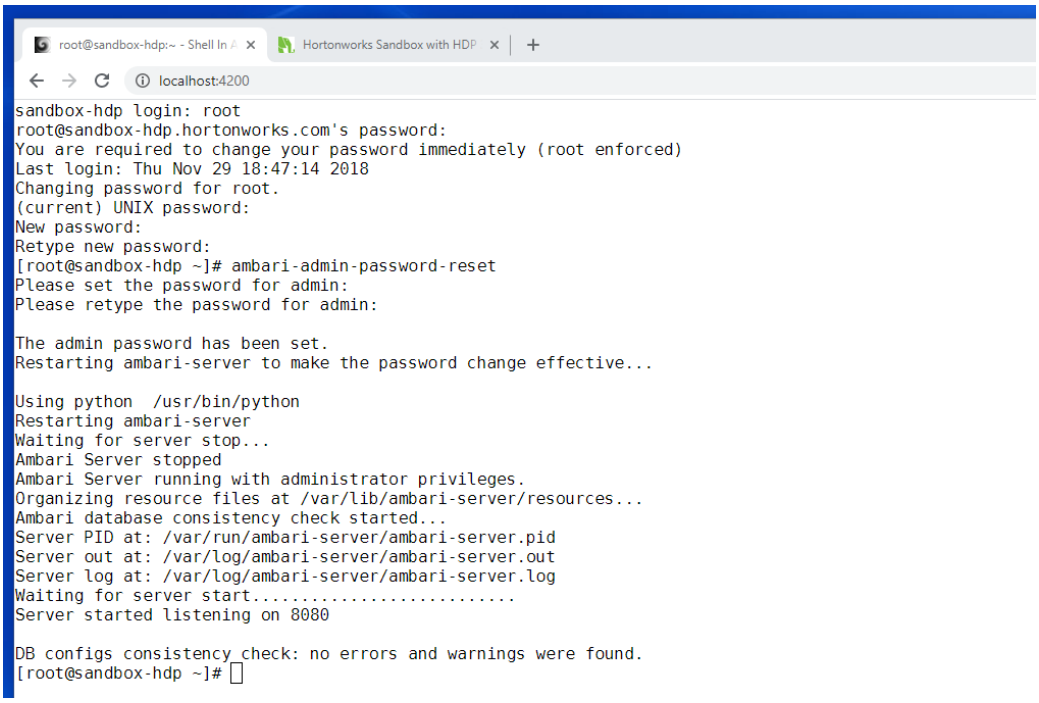
On va se servir de ce système de stockage HDFS pour stocker nos données.

Après exécution de la machine on se trouve face à cette fenêtre.



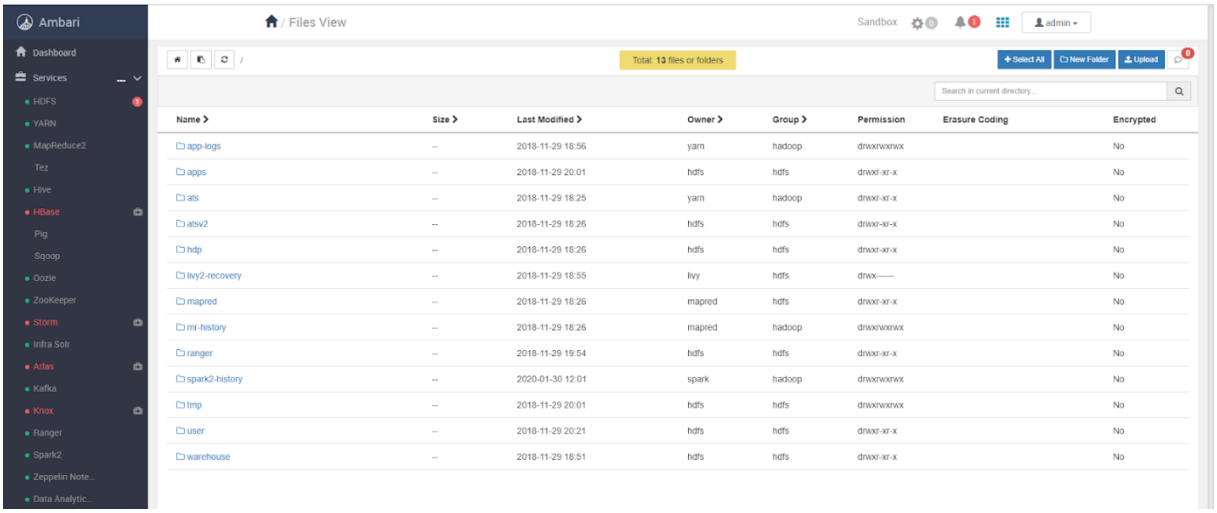
On accède à « localhost:4200 » pour faire les configurations nécessaires, notamment changer le mot de passe du compte administrateur.

Cela se fait la première fois uniquement et en exécutant la commande « ambari-admin-password-reset ».

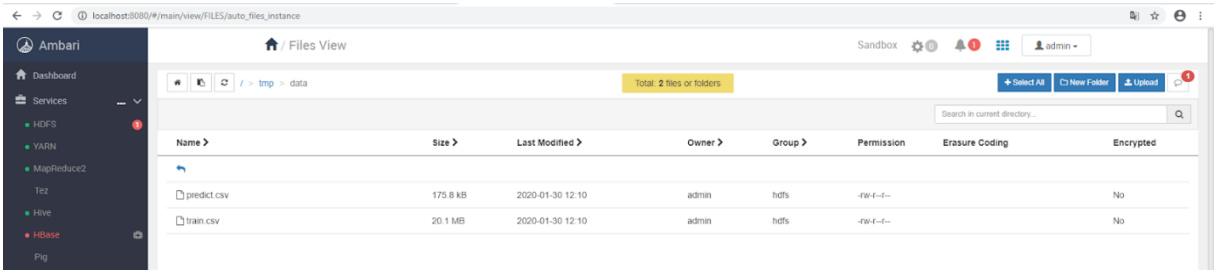



On peut maintenant accéder à « localhost:1080 » et se connecter à « Ambari ».

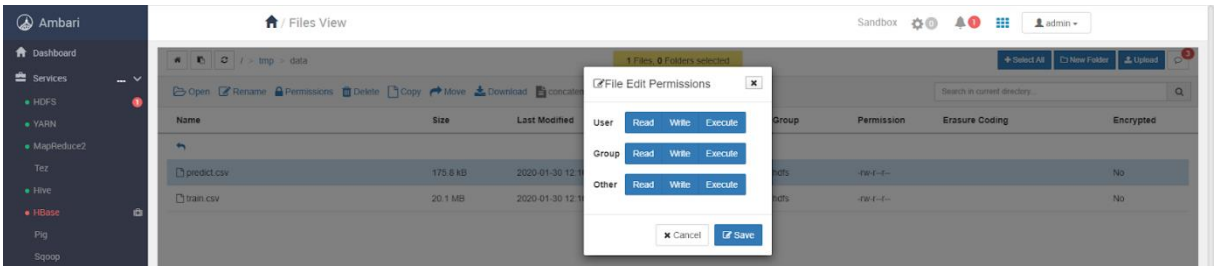
« Ambari » est une interface qui va nous permettre d'interagir avec le système de fichiers.



On va créer un dossier nommé « data » sous le dossier « tmp » où on va importer nos fichiers : « train.csv » et « predict.csv ».




Ces fichiers auront toutes les permissions nécessaires pour assurer leur manipulation par la suite.



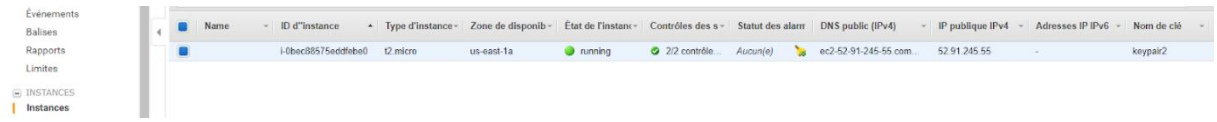

Maintenant que nos fichiers sont stockés sur HDFS ; distribués sur les machines de cluster. On peut les récupérés, à tout moment, vers notre machine locale Windows.

Pour ce faire, il suffit d'utiliser la commande suivante en précisant le chemin complet des fichiers depuis la racine.

```
[root@sandbox-hdp ~]# hadoop fs -get /tmp/data/* /root
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg  predict.csv  train.csv
```

Ces données récupérées vont être transférés à une machine virtuelle Linux dans le cloud AWS.

On va utiliser Amazon EC2 pour créer notre instance.




Pendant la création de cette machine virtuelle Linux, on spécifie les groupes de sécurité ainsi que la paire de clés qui va nous permettre d'y accéder d'une manière sécurisée.

Ce chiffrement est mis en place suite au caractère sensible des données manipulées.

Il faut importer la clé privée sur HDFS puis notre local de la même manière que les fichiers « train.csv » et « predict.csv » mais cette fois en limitant les permissions accordées.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'keypair2.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "keypair2.pem": bad permissions
Permission denied (publickey).
lost connection
[root@sandbox-hdp ~]# sudo chmod 400 keypair2.pem
```

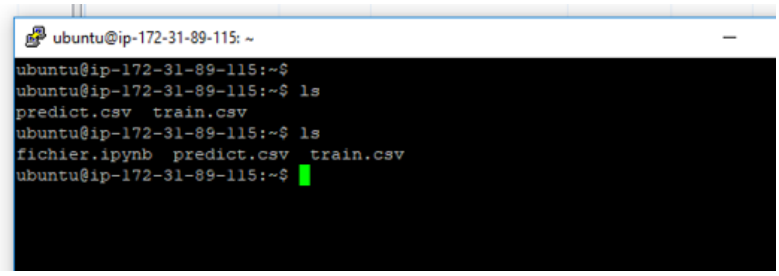
On a bien tous nos fichiers déposés localement.

```
[root@sandbox-hdp /]# hadoop fs -get /tmp/data/keypair2.pem /root
[root@sandbox-hdp /]# ls
apps  boot          core.16406  dev  hadoop  kafka-logs  lib64  mnt                          opt           proc  run      sandbox-flavour  srv  tmp  var
bin   cgroups_test  core.20215  etc  home    lib         media  mysql-connector-java-5.1.45  packer-files  root  sandbox  sbin             sys  usr
[root@sandbox-hdp /]# ls -l /root
total 20756
-rw------- 1 root root     3302 May 31  2018 anaconda-ks.cfg
-rw-r--r-- 1 root root     1670 Jan 30 12:48 keypair2.pem
-rw-r--r-- 1 root root   179969 Jan 30 12:21 predict.csv
-rw-r--r-- 1 root root 21063261 Jan 30 12:21 train.csv
[root@sandbox-hdp /]#
```

On passe maintenant au transfert de données vers AWS. Pour ce faire, on utilise la commande suivante :

```
[root@sandbox-hdp ~]# sudo scp -i keypair2.pem train.csv ubuntu@ec2-52-91-245-55.compute-1.amazonaws.com:~
train.csv                                     100%   20MB   5.4MB/s   00:03
[root@sandbox-hdp ~]# sudo scp -i keypair2.pem predict.csv ubuntu@ec2-52-91-245-55.compute-1.amazonaws.com:~
predict.csv                                   100%  176KB 353.6KB/s   00:00
[root@sandbox-hdp ~]#
```

SCP va permettre de transférer d'une manière sécurisée nos fichiers via une connexion SSH.




## Traitement de données

En se basant sur les données fournies et les algorithmes de Machine Learning, on va tester puis sélectionner un modèle d'apprentissage approprié.

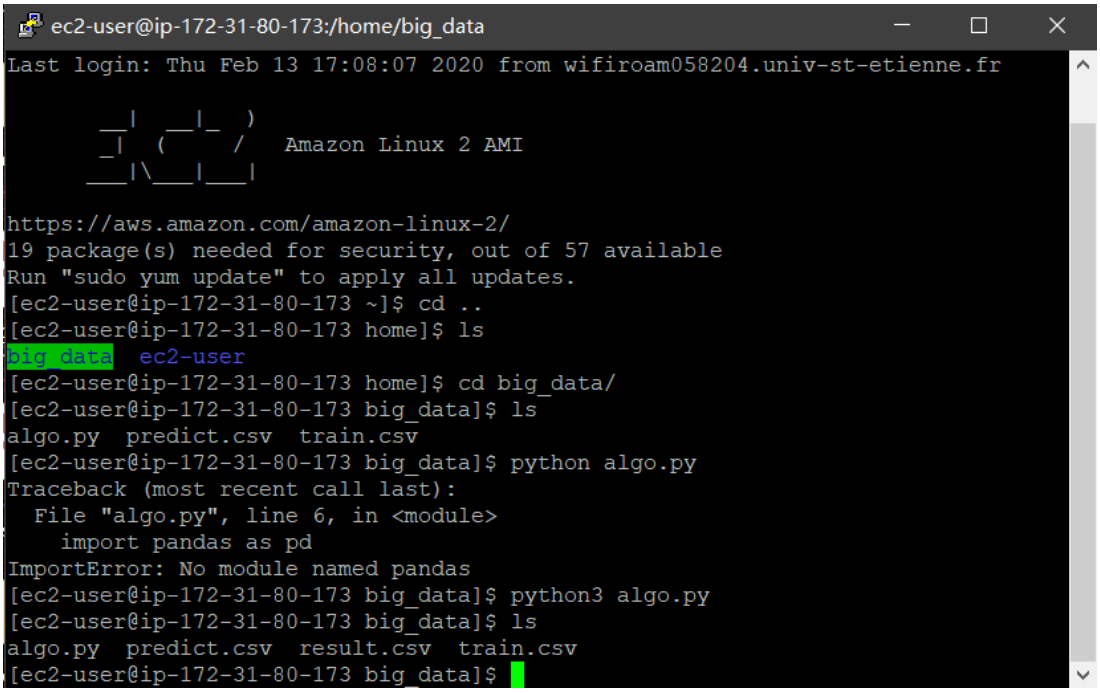
On va développer ce modèle en utilisant Python et les bibliothèques nécessaires.

L'algorithme développé sera exécuté sur l'instance AWS vue précédemment.

- Pour consulter l'analyse détaillée, vous pouvez consulter le repo. GitHub :

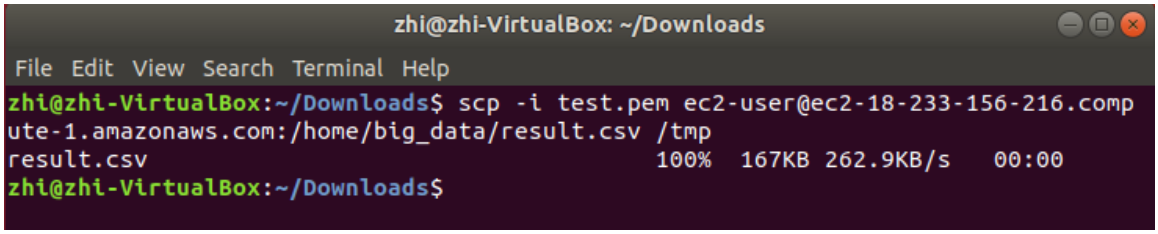
  https://github.com/WOQUQ/BigDataProjet/tree/master/Data_Process

Nous pouvons voir qu'après l'exécution du fichier python, un nouveau fichier est généré.




Ce fichier nommé « result.csv » fait une correspondance entre chaque ligne du fichier « predict.csv » et la prédiction du modèle choisie. Il est stocké actuellement dans le système de fichiers de la machine virtuelle AWS.

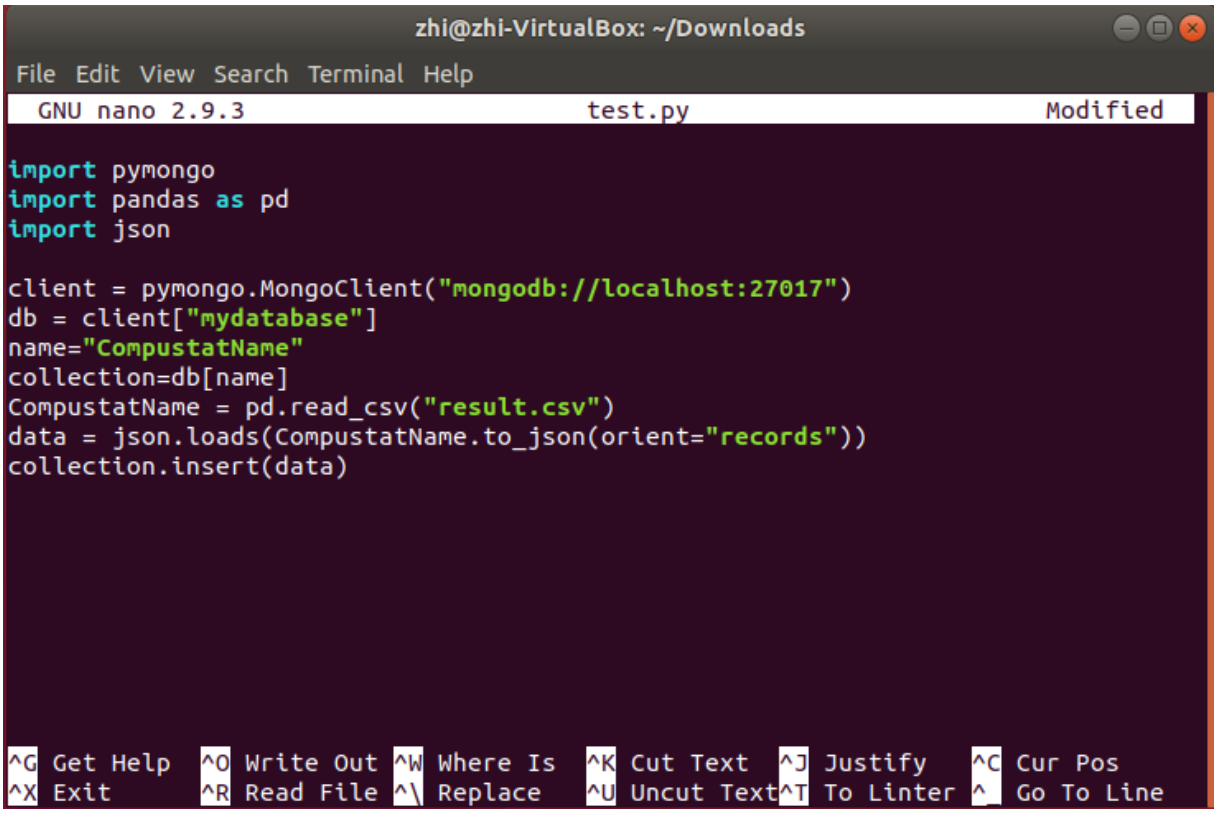
Nous allons utiliser la même instruction SCP vue précédemment pour récupérer ce fichier depuis le serveur distant.



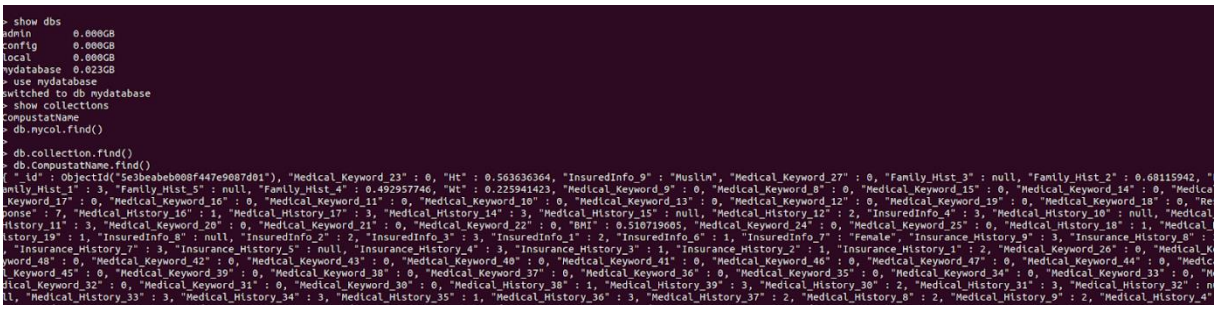

Ensuite, nous devons stocker les données dans « MongoDB ».

Nous utilisons :

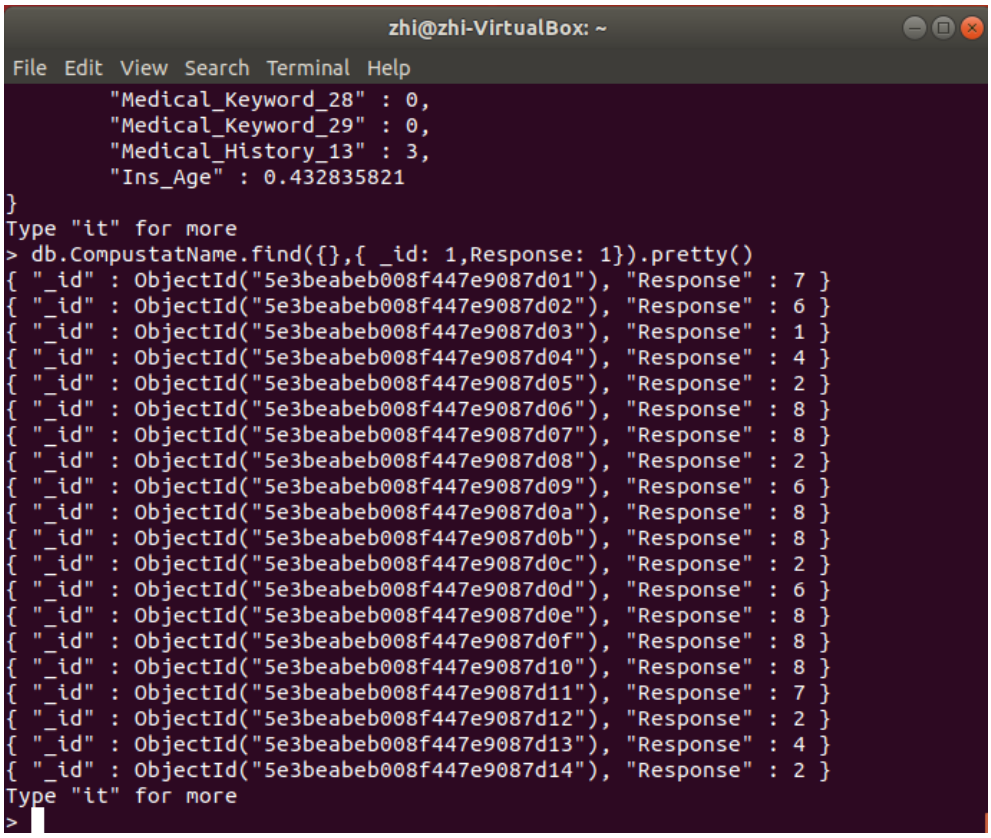
- Le package *panda* pour lire le fichier csv.
- Le package *json* pour convertir les données au format json.
- *pymongo* pour nous connecter à la base de données et stocker les données.



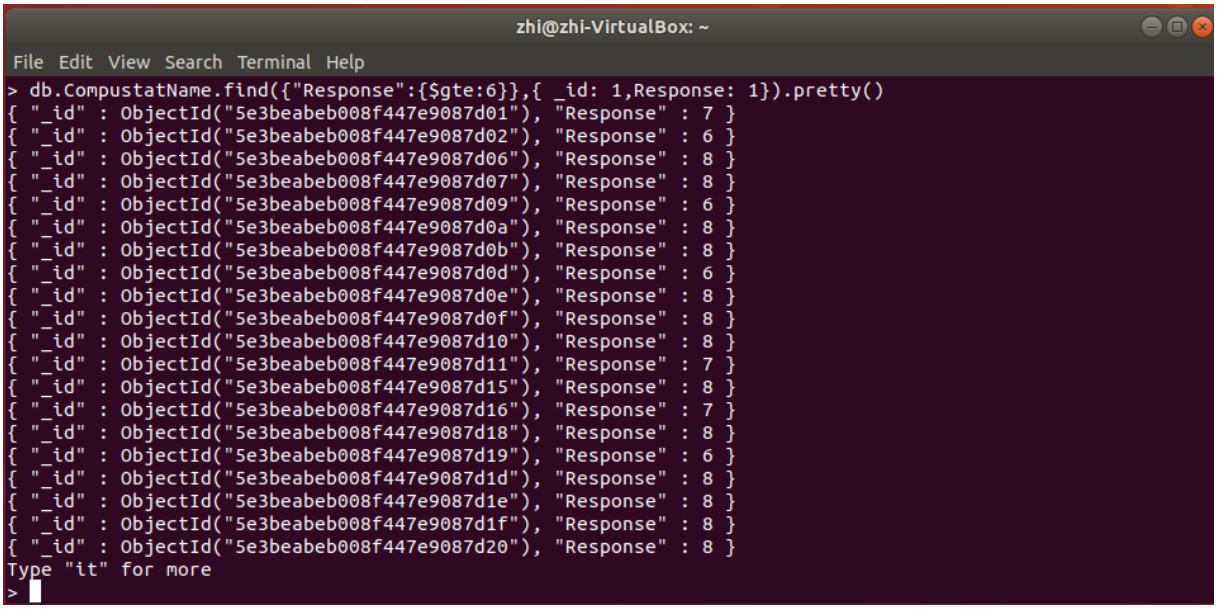

Nous pouvons voir que les données ont été stockées dans la base de données au format json.




Ensuite, nous pouvons interroger les données dans la base de données. Par exemple, imprimez l'id et la réponse correspondante :




Il est tout de même possible de définir des conditions de requête. Par exemple, pour afficher les données avec un résultat supérieur ou égal à 6 :




Donc on a nos données stockées localement dans une base de données et sont consultables avec un format approprié.