

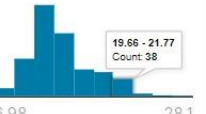
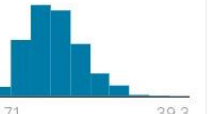
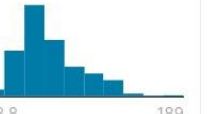

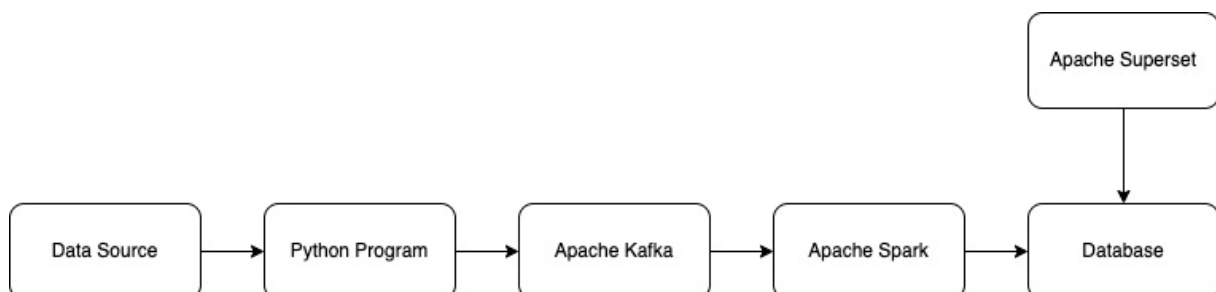


Projet Data Mining

Le projet que nous devons réaliser se base sur un fichier csv récupéré sur Kaggle contenant des données de patients atteint de cancer bénin et malin. Notre but est de faire en sorte que le médecin puisse avoir de manière rapide accès aux informations qui l'intéresse sans avoir à parcourir tout le fichier.

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
ID number	The diagnosis of breast tissues (M = malignant, B = benign)	mean of distances from center to points on the perimeter	standard deviation of gray-scale values	mean size of the core tumor	mean area of the core tumor
					
8670	B	6.98	9.71	43.8	144
842302	M	17.99	10.38	122.8	100
842517	M	20.57	17.77	132.9	132
84300903	M	19.69	21.25	130	120
84348301	M	11.42	20.38	77.58	386
84358402	M	20.29	14.34	135.1	129
843786	M	12.45	15.7	82.57	477
844359	M	18.25	19.98	119.6	104
84458202	M	13.71	20.83	90.2	577
844981	M	13	21.82	87.5	519

Le projet que nous devons réaliser consiste a mettre en œuvre la pipeline de traitement de data suivante :



Ainsi, pour tester le fonctionnement de notre pipeline, nous avons choisi comme source de données un fichier csv test contenant des données utilisateurs comme le nom, l'âge, la date de naissance ou encore la ville d'un utilisateur.

Aminata FADIGA
Nadeesha HATHARASINGHA
ITS2

```
Andrea,35,F,Paris  
Alicia,22,F,Marseille  
Julien,15,M,Nice  
Laurent,37,M,Bordeaux  
Sabrina,27,F,Paris  
Kevin,15,M,Bordeaux  
Lea,21,F,Nice  
Michael,29,M,Paris  
Andy,36,M,Grenoble  
Justin,16,M,Marseille  
Michèle,22,F,Paris  
Andrea,16,F,Toulouse  
Justine,17,F,Avignon  
Mike,26,M,Paris  
Nelson,30,M,Lyon  
Joan,18,M, Paris  
Michaela,24,F,Nantes  
Anne,14,F,Montpellier  
Justine,19,F,Strasbourg  
Leo,39,M,Rennes
```

Nous ouvrirons ce fichier à travers un programme Python « Producer » qui nous permettra de les envoyer dans un bus Kafka.

Ensuite, dans un programme Spark, nous nous connecterons à Kafka afin de récupérer les données reçues dans le bus Kafka, ligne par ligne. Nous effectuerons ensuite plusieurs transformations sur les données et enfin, ces données modifiées finaux seront ensuite sauvegardées dans une table d' une base de données MySQL puis affichées à l'aide de l'outil de visualisation Superset.

Afin de réaliser ce projet, nous avons suivi les étapes suivantes :

-Nous avons commencer par installer et initialiser trois machines virtuelles à l'aide de vagrant en utilisant les fichiers d'installation donnés et en utilisant la commande « vagrant up » :

- 1^{ère} machine : Spark, on attribue l' adresse IP : 192.168.33.12
- 2^{ème} machine : Kafka, on attribue l'adresse IP : 192.168.33.13. Pour démarrer Kafka, on lance les deux commandes suivantes dans deux sessions de terminaux différents :

- \$ bin/zookeeper-server-start.sh config/zookeeper.properties
- \$ bin/kafka-server-start.sh config/server.properties
-

On y crée notre sujet « projet_datamining » avec la commande suivante :

- \$ bin/kafka-topics.sh --create --topic projet_datamining --bootstrap-server 192.168.33.13:9092

Pour tester le fonctionnement directement dans un terminal, nous pouvons écrire dans le topic en tant que producer en lançant cette commande :

- \$ bin/kafka-console-producer.sh --topic projet_datamining --bootstrap-server 192.168.33.13:9092

Et pour lire les données en tant que consumer, on lance cette commande :

Aminata FADIGA
Nadeesha HATHARASINGHA
ITS2

- \$ bin/kafka-console-consumer.sh --topic projet_datamining --from-beginning --bootstrap-server 192.168.33.13:9092
- 3^{ème} machine : on y installe MySQL en suivant les étapes de ce lien : <https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04-fr> et on attribue l'adresse IP : 192.168.33.14. On configure les préférences de sécurité puis nous créons l'utilisateur « nadeesha » et on donne à l'utilisateur le droit de tout faire sur toutes les tables :

```
mysql> CREATE USER 'nadeesha'@'%' IDENTIFIED BY 'nadeesha';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'nadeesha'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Puis on se connecte en tant que nadeesha en utilisant la commande : « mysql -u nadeesha -p » avec comme mot de passe : « nadeesha » :

On crée maintenant la base de données « projet_datamining » :

```
vagrant@vagrant:~$ mysql -u nadeesha -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 189
Server version: 5.7.36-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> CREATE DATABASE projet_datamining;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| projet_datamining |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> USE projet_datamining;
Database changed
mysql>
```

Nous installerons également dans cette même machine virtuelle Apache Superset.

➤ Lien entre les programmes

- Nous avons ensuite lier Spark et Kafka. Pour cela, on suit les commandes suivantes dans la machine virtuelle Spark:

On désactive la variable d'environnement PYSPARK_DRIVER_PYTHON :

```
root@vagrant:/home/vagrant# unset PYSPARK_DRIVER_PYTHON
```

On télécharge la librairie correspondante contenue dans le jar suivant :

```
root@vagrant:/home/vagrant# wget https://repol.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8-assembly_2.11/2.4.7/spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar
--2022-01-06 23:06:43-- https://repol.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8-assembly_2.11/2.4.7/spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar
Resolving repol.maven.org (repol.maven.org)... 151.101.120.209
Connecting to repol.maven.org (repol.maven.org)|151.101.120.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11709363 (11M) [application/java-archive]
Saving to: 'spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar'

spark-streaming-kafk 100%[=====>] 11.17M 22.4MB/s in 0.5s

2022-01-06 23:06:44 (22.4 MB/s) - 'spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar' saved [11709363/11709363]
```

Puis on le déplace dans le dossier local où se trouve les dépendances de spark :

```
root@vagrant:/home/vagrant# sudo mv spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar /usr/local/spark/jars/
```

- On suit les mêmes étapes pour la librairie mysql-connector-java-5.1.36.jar permettant de lier mysql et spark.

```
vagrant@vagrant:~$ ls
book.txt                               projet_data_mining
checkpoint-1641523700000               Quick_Start_Nadeesha_HATHARASINGHA_ITS2.ipynb
checkpoint-1641523700000.bk            rdd.ipynb
Consumer.py                            README.md
dataframe-1.ipynb                      spark-2.4.8-bin-hadoop2.7.tgz
dataframe.ipynb                        spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar.1
data.json                              streaming-wordcount-kafka.py
exemple.py.ipynb                       'TP Spark.ipynb'
kafka_2.13-3.0.0                       wordcount2mysql.py
kafka_2.13-3.0.0.tgz                   wordcount_kafka2.py
mysql-connector-java-5.1.36.jar         wordcount_kafka.py
path

vagrant@vagrant:~$ unset PYSPARK_DRIVER_PYTHON
vagrant@vagrant:~$ sudo mv mysql-connector-java-5.1.36.jar /usr/local/spark/jars/
```

➤ Création du programme 'Producer' Python :

Nous avons ensuite créer un programme Python « Producer » qui lit un fichier csv ligne par ligne et l'envoie dans le bus Kafka toutes les 10 secondes. Pour cela, nous nous sommes connecter à Kafka en indiquant l'adresse IP de la VM correspondante ainsi que le port et nous avons utiliser le code suivant :

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 7 01:15:44 2022

@author: nadee
"""

#Envoie un fichier json ligne ligne par ligne dans le bus Kafka
import json
from kafka import KafkaProducer
import time
#On se connecte à la machine Kafka
producer = KafkaProducer(bootstrap_servers='192.168.33.13:9092', value_serializer=lambda v: json.dumps(v).encode('utf-8'))

#On récupère chaque ligne du fichier json dans une liste
with open('data.csv', 'r') as f:
    listusers= f.readlines()

for i in listusers:
    user=i.strip()
    print(user)
    producer.send('projet_datamining', user)
    time.sleep(10)
```

On vérifie avec la commande Consumer pour lire les données citée ci-dessus :

```
vagrant@vagrant:~/kafka_2.13-3.0.0$ bin/kafka-console-consumer.sh --topic projet_datamining --from-beginning --bootstrap-server 192.168.33.13:9092
"nom,age,genre,ville"
"Andrea,35,F,Paris"
"Alicia,22,F,Marseille"
"Julien,15,M,Nice"
"Laurent,37,M,Bordeaux"
"Sabrina,27,F,Paris"
"Kevin,15,M,Bordeaux"
"Lea,21,F,Nice"
"Michael,29,M,Paris"
"Andy,36,M,Grenoble"
"Justin,16,M,Marseille"
"Mich\u00c3\u00a8le,22,F,Paris"
"Andrea,16,F,Toulouse"
"Justine,17,F,Avignon"
"Mike,26,M,Paris"
"Nelson,30,M,Lyon"
"Joan,18,M, Paris"
"Michaela,24,F,Nantes"
"Anne,14,F,Montpellier"
"Justine,19,F,Strasbourg"
"Leo,39,M,Rennes"
```

➤ Création du programme 'Consumer' Spark :

Nous avons ensuite créer le programme Consumer Spark.

Ce programme consiste à se connecter à Kafka pour récupérer les données du fichier source csv qui ont été envoyés dans le bus Kafka puis à appliquer des transformations afin de trier et organiser les données reçues et finalement les transmettre dans notre base de données MySQL afin de minimiser les tâches du médecin.

Voici quelques exemples des transformations que nous allons effectuer :

- **Ligne.split(',') :** Pour transformer les chaînes de caractères en liste
- **Filter :** Pour afficher seulement les personnes dont le diagnostic est Malin

Nous afficherons finalement la moyenne de chaque donnée du diagnostic pour toutes les personnes atteintes d'un cancer du sein. Le médecin pourra ensuite se servir de cette moyenne pour déterminer le diagnostic de ses nouveaux patients.

Pour cela, nous avons d'abord indiqué l'adresse IP de la machine virtuelle Kafka correspondante (192.168.33.13), ainsi que notre nom de topic « projet_datamining ». Pour chaque ligne de mots envoyé qui sont donc séparés par des virgules car ils proviennent d'un fichier.csv, on transforme la chaîne de caractères en liste de mots avec la commande : `ligne.split(',')`.

Enfin, on sauvegarde les données dans une nouvelle table « testuser » dans la base de données MySQL « projet_datamining » créée précédemment à l'aide du code suivant :

Aminata FADIGA
Nadeesha HATHARASINGHA
ITS2

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession, Row
from pyspark.streaming.kafka import KafkaUtils
from pyspark.sql import SparkSession

def getSparkSessionInstance(sparkConf):
    if ("sparkSessionSingletonInstance" not in globals()):
        globals()["sparkSessionSingletonInstance"] = SparkSession \
            .builder \
            .config(conf=sparkConf) \
            .getOrCreate()
    return globals()["sparkSessionSingletonInstance"]

def process(rdd):
    # Get the singleton instance of SparkSession
    spark = getSparkSessionInstance(rdd.context.getConf())
    #rowRdd = rdd.map(lambda v: Row(val=v))
    df=spark.createDataFrame(rdd,schema= ['name', 'age', 'genre', "ville"])
    df.printSchema()
    df.write.format('jdbc').options(
        url='jdbc:mysql://192.168.33.14/projet_datamining',
        dbtable='testuser',
        user='nadeesha',
        password='nadeesha').mode('append').save()

# Create a Local StreamingContext with two working thread and batch interval of 1 second
sc = SparkContext("local[2]", "Kafka Stroke")
sc.setLogLevel("ERROR")
ssc = StreamingContext(sc, 10)

# Create a DStream that will connect to hostname:port, Like localhost:9999
kds = KafkaUtils.createDirectStream(ssc, ["projet_datamining"], {"metadata.broker.list": "192.168.33.13:9092"})

lines = kds.map(lambda x: x[1])

#Transforme la première ligne (string) en liste de mots
rdd = lines.map(lambda s: s.split(','))

#Il sélectionne les personnes atteintes du cancer du sein (diagnostic=Malin)
#rdd2=rdd.filter(lambda x: x[1] == 'M')

#Affiche les 10 premiers éléments de chaque RDD généré
rdd.pprint()

rdd.foreachRDD(process)

ssc.start() # Start the computation
ssc.awaitTermination() # Wait for the computation to terminate
```

On teste ensuite dans Spark avec la commande suivante dans un terminal :
/usr/local/spark/bin/spark-submit Consumer.py et on a le résultat suivant :

Aminata FADIGA
Nadeesha HATHARASINGHA
ITS2

```
22/02/09 00:37:09 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.2.15, 34613, None)
22/02/09 00:37:09 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.2.15:34613 with 413.9 MB RAM, BlockManagerId(
iver, 10.0.2.15, 34613, None)
22/02/09 00:37:09 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.2.15, 34613, None)
22/02/09 00:37:09 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.2.15, 34613, None)
-----
Time: 2022-02-09 00:37:20
-----
[["Alicia", '22', 'F', 'Marseille']]

root
|-- name: string (nullable = true)
|-- age: string (nullable = true)
|-- genre: string (nullable = true)
|-- ville: string (nullable = true)
-----
Time: 2022-02-09 00:37:30
-----
[["Julien", '15', 'M', 'Nice']]

root
|-- name: string (nullable = true)
|-- age: string (nullable = true)
|-- genre: string (nullable = true)
|-- ville: string (nullable = true)
-----
Time: 2022-02-09 00:37:40
-----
[["Laurent", '37', 'M', 'Bordeaux']]

root
|-- name: string (nullable = true)
|-- age: string (nullable = true)
|-- genre: string (nullable = true)
|-- ville: string (nullable = true)
```

On vérifie maintenant que les entrées se sont bien ajoutés dans notre table « testuser » :

```
This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
Last login: Sat Jan 22 15:20:45 2022 from 10.0.2.2
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Feb  8 22:13:40 UTC 2022

System load:  0.62           Processes:      104
Usage of /:   3.1% of 61.80GB Users logged in:  0
Memory usage: 28%           IP address for eth0: 10.0.2.15
Swap usage:   0%            IP address for eth1: 192.168.33.14

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
Last login: Sat Jan 22 15:20:45 2022 from 10.0.2.2
vagrant@vagrant:~$ mysql -u nadeesha -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.36-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use projet_datamining;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```


Aminata FADIGA
Nadeesha HATHARASINGHA
ITS2

```
mysql> show tables;
+-----+
| Tables_in_projet_datamining |
+-----+
| test                          |
| testuser                     |
| users                        |
| words                        |
+-----+
4 rows in set (0.00 sec)

mysql> select * from testuser;
+-----+-----+-----+-----+
| name  | age | genre | ville  |
+-----+-----+-----+-----+
| "Alicia" | 22 | F     | "Marseille" |
| "Julien" | 15 | M     | "Nice"      |
| "Laurent" | 37 | M     | "Bordeaux"  |
| "Sabrina" | 27 | F     | "Paris"     |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

On voit ainsi que les données se sont bien ajoutés en temps réel dans la base de données « projet_datamining » dans notre table « testuser ».

On teste maintenant l’affichage dans Apache Superset :