

## Éléments de corrigé


### Corrigé Section 1

Réponse :

```
/spark/bin/spark-shell --conf spark.driver.extraJavaOptions="-Dscala.color"
```

```
val lignes = sc.textFile("lai-eliduc.txt") // or sc.textFile("file:///lai-eliduc.txt")
lignes.count()
lignes.getNumPartitions
```

**Réponse :** `flatMap` parce-que l'on souhaite un décompte au total et non par ligne, donc plutôt que d'avoir une liste par ligne on préfère fusionner toutes les listes en une seule liste de mots.

NB. Si on voulait visualiser le résultat, le mieux est dans doute d'utiliser un nuage de mots ( wordcloud). Il existe pour cela plusieurs applications populaires en python et scala. Pas besoin d'exporter en csv si on reste dans le même langage. De plus ces applications peuvent parfois calculer les décomptes elles-même donc on (en python on peut passer au choix soit le texte original soit un dictionnaire avec les fréquences)

```
val counts = lignes.flatMap(s => s.split("\\W+")).map(word => (word, 1)).reduceByKey(_ + _)
counts.saveAsTextFile("nbmots")
counts.coalesce(1).saveAsTextFile("nbmots2")
```

```
counts.filter(_._2 >= 8).take(10)
// ou
counts.filter(_._2 >= 8).take(10).foreach(println)
```

## Corrigé Section 2

```
# wget -O food.csv https://data.cityofchicago.org/api/views/4ijn-s7e5/rows.csv

# to execute from a notebook (I'm not sure about the settings):
# export PYSPARK_DRIVER_PYTHON='jupyter'
# export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
#
# or otherwise, just launch pyspark from command line

from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer
from pyspark.sql import Row
from pyspark.sql.functions import udf
from pyspark.sql.functions import desc
from pyspark.sql.types import *

df1 = spark.read.load("/home/tp-home002/bgroz/testspark/dataset2/food_inspections.csv",
format="csv",
sep=",",
inferSchema="true",
header="true")

df1.show()
df1.printSchema()

inspections = df1.select('Inspection ID', 'DBA Name', 'results', 'Violations').dropna()

nbre = df1.groupBy("results").count().withColumnRenamed("count", "nb").sort(desc("nb"))
nbre.show()
nbre.explain(True)
```

## Corrigé Section 3 et 4

```
def labelForResults(s):
    if s == 'Fail':
        return 0.0
    elif s == 'Pass w/ Conditions' or s == 'Pass':
        return 1.0
    else:
        return -1.0

monudf = udf(labelForResults, DoubleType())
labeledData = inspections.select(monudf(inspections["results"])
    .alias('label'), "violations")
    .where('label >= 0')

splitdf = labeledData.randomSplit([0.25, 0.75], 105)
training = splitdf[0]
validationDf = splitdf[1]

tokenizer = Tokenizer(inputCol="violations", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

model = pipeline.fit(labeledData)

predictionsDf = model.transform(validationDf)
predictionsDf.registerTempTable('Predictions')
predictionsDf.columns

numSuccesses = predictionsDf.where("""(prediction = 0 AND results = 'Fail') OR
    (prediction = 1 AND (results = 'Pass' OR
        results = 'Pass w/ Conditions'))""").count()
numInspections = predictionsDf.count()
print "There were", numInspections, "inspections and there were", numSuccesses, "successful predictions"
print "This is a", str((float(numSuccesses) / float(numInspections)) * 100) + "%", "success rate"

## Write the python code in simpleapp.py:
#from pyspark.sql import SparkSession
#spark = SparkSession.builder.appName("SimpleApp").getOrCreate()
#
#your code here
#
#spark.stop()

## Type in terminal:
#spark-submit \
# --master local[1] \
# spark-simpleapp.py
```