

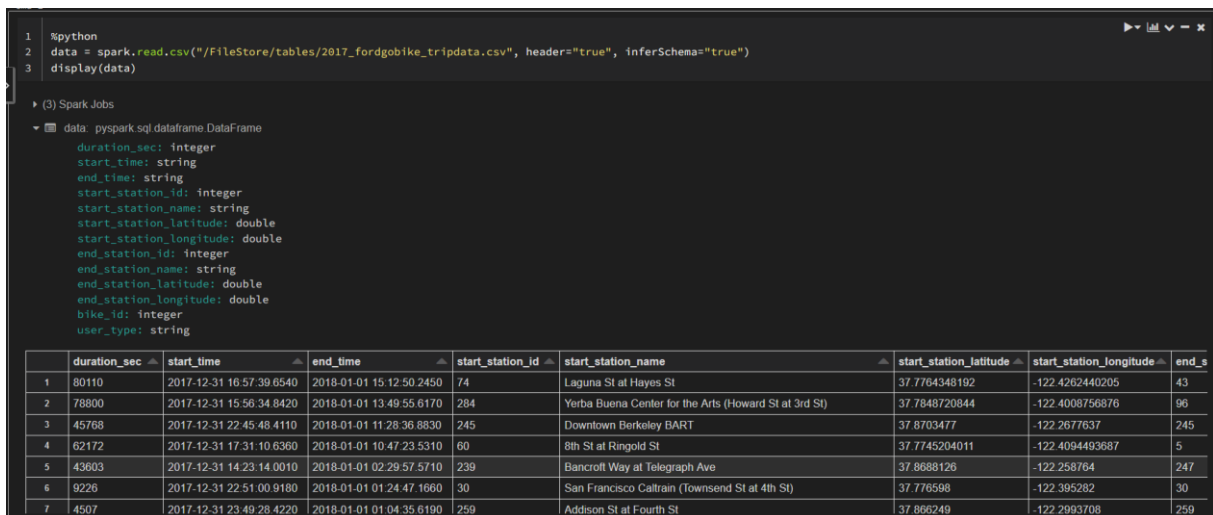
Découverte et mise en oeuvre de Spark

Dans ce projet d'initialisation au Spark, j'ai commencé par créer un compte Databricks , puis j'ai fait l'ingestion de données.

Le dataset utilisé est sous forme d'un fichier CSV qui regroupe des données sur les déplacements à vélos, vous trouverez le fichier ici :

<https://s3.amazonaws.com/baywheels-data/index.html>

j'ai choisi par hasard le fichier 2017_fordgobike_tripdata.csv.



```
1 %python
2 data = spark.read.csv("/FileStore/tables/2017_fordgobike_tripdata.csv", headers="true", inferSchema="true")
3 display(data)
```

(3) Spark Jobs

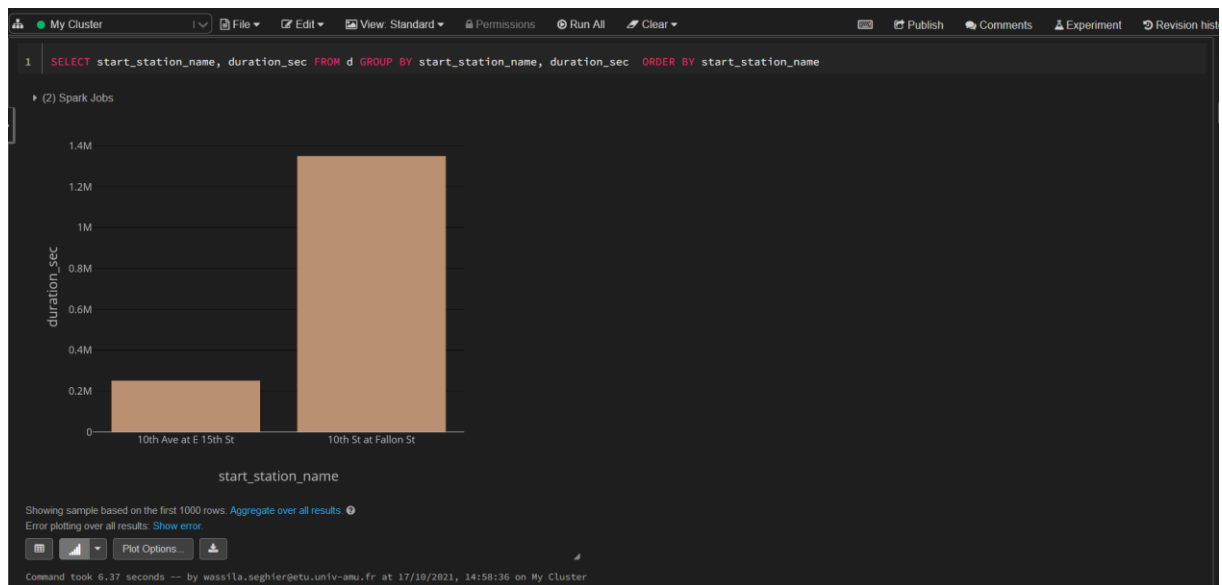
data pyspark.sql.dataframe.DataFrame

- duration_sec: integer
- start_time: string
- end_time: string
- start_station_id: integer
- start_station_name: string
- start_station_latitude: double
- start_station_longitude: double
- end_station_id: integer
- end_station_name: string
- end_station_latitude: double
- end_station_longitude: double
- bike_id: integer
- user_type: string

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_longitude	end_s
1	80110	2017-12-31 16:57:39.6540	2018-01-01 15:12:50.2450	74	Laguna St at Hayes St	37.7764348192	-122.4262440205	43
2	78800	2017-12-31 15:56:34.8420	2018-01-01 13:49:55.6170	284	Yerba Buena Center for the Arts (Howard St at 3rd St)	37.7848720844	-122.4008756876	96
3	45768	2017-12-31 22:45:48.4110	2018-01-01 11:28:36.8830	245	Downtown Berkeley BART	37.8703477	-122.2677637	245
4	62172	2017-12-31 17:31:10.6360	2018-01-01 10:47:23.5310	60	8th St at Ringold St	37.7745204011	-122.4094493687	5
5	43603	2017-12-31 14:23:14.0010	2018-01-01 02:29:57.5710	239	Bancroft Way at Telegraph Ave	37.8688126	-122.258764	247
6	9226	2017-12-31 22:51:00.9180	2018-01-01 01:24:47.1660	30	San Francisco Caltrain (Townsend St at 4th St)	37.776598	-122.395282	30
7	4507	2017-12-31 23:49:28.4220	2018-01-01 01:04:35.6190	259	Addison St at Fourth St	37.866249	-122.2993708	259

Après la phase d'importation des données, j'ai pu analyser celles-ci avec quelques requêtes SQL avec SparkSQL.

Par exemple la longitude des station et la latitude selon le nom de la station, j'ai compter le nombre de stations sur la base de données qui est de 272 et le type d'utilisateur client et abonné.



Cmd 7

```
1 SELECT distinct start_station_name, start_station_latitude, start_station_longitude, user_type from d where start_station_name == "Addison St at Fourth St"
```

(2) Spark Jobs

	start_station_name	start_station_latitude	start_station_longitude	user_type
1	Addison St at Fourth St	37.866249	-122.2993708	Customer
2	Addison St at Fourth St	37.866249	-122.2993708	Subscriber

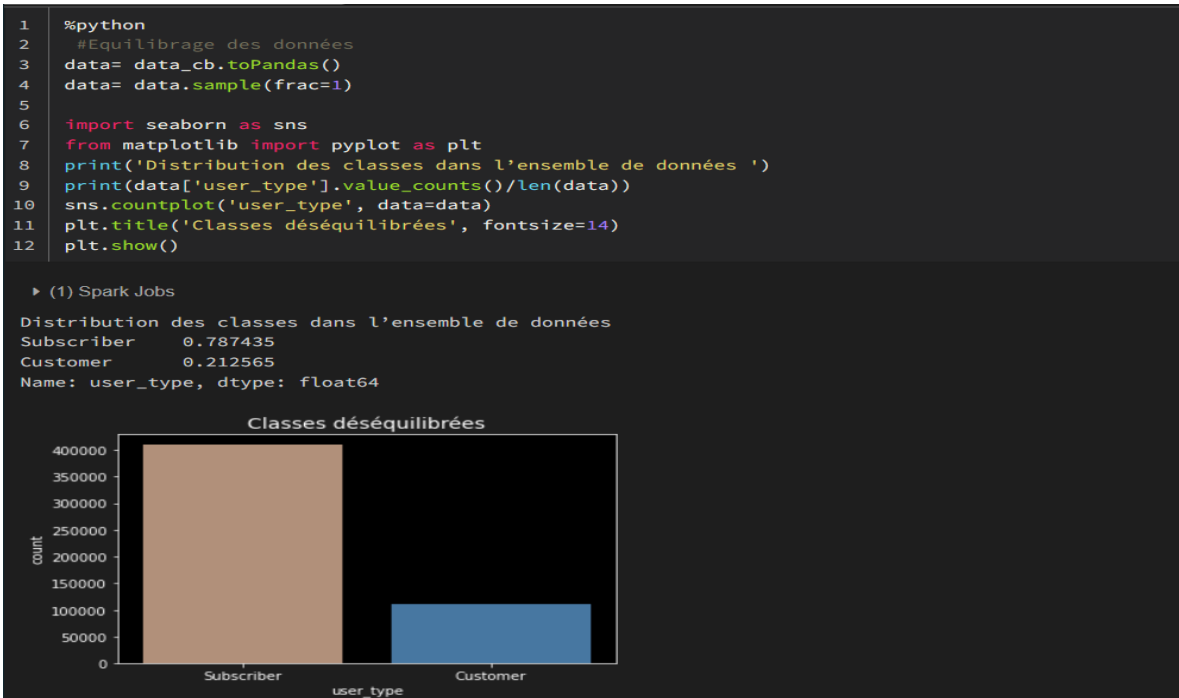
Showing all 2 rows.

Command took 3.17 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 14:59:03 on My Cluster

Puis j'ai attaqué la partie Machine learning, où j'ai construit un modèle pyspark pour détecter si l'utilisateur est un client ou un abonné.

```
1 %python
2 from pyspark.sql import SparkSession
3 from pyspark.sql.functions import split
4 from pyspark.ml import Pipeline
5 from pyspark.ml.classification import DecisionTreeClassifier
6 from pyspark.ml.feature import IndexToString, StringIndexer, VectorIndexer
7 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
8
9 if __name__ == "__main__":
10     spark = SparkSession\
11         .builder\
12         .appName("Python tp1")\
13         .getOrCreate()
14
15 Command took 8.07 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:06:30 on My Cluster
```

Il a fallu d'abord équilibrer les données pour obtenir un bon modèle et cela en réduisant le nombre de Suscriber.



J'ai utilisé les Random Forest, decision trees et Gradient boosted trees comme algorithmes d'apprentissage (supervisé), j'ai divisé le corpus en un ensemble d'apprentissage et un ensemble de test, et cela après une phase de transformation des données pour cela j'ai utilisé un vecteur assembleur pour combiner les caractéristiques que j'ai jugé utiles pour l'apprentissage.

```

1 %python
2 from pyspark.ml import Pipeline
3 from pyspark.ml.classification import GBTCClassifier
4 from pyspark.ml.feature import VectorIndexer, VectorAssembler
5 from pyspark.ml.evaluation import BinaryClassificationEvaluator
6 from pyspark.ml.linalg import DenseVector
7
8 from pyspark.ml.feature import VectorAssembler
9
10 df = sqlContext.createDataFrame(new_df)
11 numericCols = ['duration_sec', 'start_station_latitude', 'start_station_longitude', 'end_station_latitude', 'end_station_longitude']
12 assembler = VectorAssembler(inputCols=numericCols, outputCol="features")
13 df = assembler.transform(df)
14 df.show()

```

(1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 12 more fields]

[duration_sec]	start_time	end_time	[start_station_id]	start_station_name	[start_station_latitude]	[start_station_longitude]	[end_station_id]	end_station_name
26925	2017-09-19 13:25:...	2017-09-19 20:54:...	36	Folsom St at 3rd St	37.78383	-122.39887	36	Folsom St at 3rd St
37.78383	-122.39887	2543	1	[26925.0, 37.78383...				
322	2017-11-28 17:45:...	2017-11-28 17:50:...	29	O'Farrell St at D...	37.7824046019	-122.4394458532	285	Webster St at O'F...
37.7835208353	-122.4311578274	1837	0	[322.0, 37.7824046...				

```

1 %python
2 pd.DataFrame(df.take(110), columns=df.columns).transpose()
3
4
5 train, test = df.randomSplit([0.7, 0.3], seed = 2018)
6 print("Training Dataset Count: " + str(train.count()))
7 print("Test Dataset Count: " + str(test.count()))

```

(5) Spark Jobs

train: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 13 more fields]

test: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 13 more fields]

Training Dataset Count: 154610

Test Dataset Count: 66330

Command took 8.17 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:12:25 on My Cluster

Je n'ai gardé que les caractéristiques numériques, puis j'ai utilisé un string Indexer pour coder les étiquettes.

```

1 %python
2
3 from pyspark.ml.feature import StringIndexer
4
5 label_stringIdx = StringIndexer(inputCol = 'user_type', outputCol = 'labelIndex')
6 df = label_stringIdx.fit(df).transform(df)
7 df.show()

```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 13 more fields]

[duration_sec]	start_time	end_time	[start_station_id]	start_station_name	[start_station_latitude]	[start_station_longitude]	[end_station_id]	end_station_name
26925	2017-09-19 13:25:...	2017-09-19 20:54:...	36	Folsom St at 3rd St	37.78383	-122.39887	36	Folsom St at 3rd St
37.78383	-122.39887	2543	1	[26925.0, 37.78383...				
322	2017-11-28 17:45:...	2017-11-28 17:50:...	29	O'Farrell St at D...	37.7824046019	-122.4394458532	285	Webster St at O'F...
37.7835208353	-122.4311578274	1837	0	[322.0, 37.7824046...				
415	2017-07-21 19:15:...	2017-07-21 19:22:...	318	San Carlos St at ...	37.330698	-121.888979	314	Santa Clara St at...
37.333988	-121.894902	1755	0	[415.0, 37.330698...				
229	2017-12-28 09:34:...	2017-12-28 09:38:...	137	Jersey St at Cast...	37.750506	-122.4339496	134	Valencia St at 24...
37.7524278	-122.4206278	1501	0	[229.0, 37.750506...				
364	2017-12-20 17:31:...	2017-12-20 17:37:...	28	The Embarcadero a...	37.7871680147	-122.3880979233	15	San Francisco Fer...
37.795392	-122.394203	3306	0	[364.0, 37.787168...				
526	2017-07-07 08:11:...	2017-07-07 08:20:...	30	San Francisco Cal...	37.776598	-122.395282	25	Howard St at 2nd St
37.7875217805	-122.3974049091	601	0	[526.0, 37.776598...				
770	2017-08-19 13:15:...	2017-08-19 13:28:...	197	El Embarcadero at...	37.8088479	-122.2496799	200	2nd Ave at E 18th St
37.800213567	-122.2538101673	1125	1	[770.0, 37.8088479...				

Ensuite j'ai entraîné mes modèles, le Random Forest a obtenu environ 72% d'accuracy, j'ai à la fin visualisé la matrice de confusion du modèle.

```

1 %python
2
3 # RANDOM FOREST
4 from pyspark.ml.classification import RandomForestClassifier
5
6 rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'labelIndex')
7 rfModel = rf.fit(train)
8 predictions = rfModel.transform(test)
9 predictions.select('duration_sec', 'start_station_latitude', 'start_station_longitude', 'end_station_latitude', 'end_station_longitude', 'labelIndex',
10 'rawPrediction', 'prediction', 'probability').show(25)
18

```

► (9) Spark Jobs

► predictions: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 16 more fields]

duration_sec	start_station_latitude	start_station_longitude	end_station_latitude	end_station_longitude	labelIndex	rawPrediction	prediction	probability
61	37.774814	-122.418954	37.776619	-122.417385	0.0	[14.1353981589351...	0.0	[0.70676990794675...
68	37.7630152	-122.4264968	37.7630152	-122.4264968	1.0	[14.1526588714784...	0.0	[0.70763294357392...
70	37.7753058	-122.39738	37.7766392	-122.3955263	0.0	[15.3368050797556...	0.0	[0.76684025398778...
70	37.7662185	-122.4310597	37.7662185	-122.4310597	1.0	[14.1526588714784...	0.0	[0.70763294357392...

```

6
7 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
8
9 evaluator = MulticlassClassificationEvaluator(labelCol="labelIndex", predictionCol="prediction")
10 accuracy = evaluator.evaluate(predictions)
11 print("Accuracy = %s" % (accuracy))
12 print("Test Error = %s" % (1.0 - accuracy))

```

► (2) Spark Jobs

```

+-----+-----+
|labelIndex|prediction|
+-----+-----+
|      0.0|      0.0|
|      1.0|      0.0|
|      0.0|      0.0|
|      1.0|      0.0|
|      0.0|      0.0|
|      0.0|      0.0|
|      0.0|      0.0|
|      1.0|      0.0|
|      0.0|      0.0|
|      0.0|      0.0|
+-----+-----+
only showing top 10 rows

```

Accuracy = 0.71908030458591
Test Error = 0.28091969541409

Command took 5.91 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:14:11 on My Cluster

```

1 %python
2
3 from pyspark.mllib.evaluation import MulticlassMetrics
4 from pyspark.sql.types import FloatType
5 import pyspark.sql.functions as F
6
7 preds_and_labels = predictions.select(['prediction', 'labelIndex']).withColumn('labelIndex', F.col('labelIndex').cast(FloatType())).orderBy('prediction')
8 preds_and_labels = preds_and_labels.select(['prediction', 'labelIndex'])
9 metrics = MulticlassMetrics(preds_and_labels.rdd.map(tuple))
10 print(metrics.confusionMatrix().toArray())

```

► (4) Spark Jobs

► preds_and_labels: pyspark.sql.dataframe.DataFrame = [prediction: double, labelIndex: float]

```

[[27143, 6059,]
 [12402, 20726,]]

```

Command took 10.22 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:14:39 on My Cluster

Les arbres de décision ont obtenu environ 73% d'accuracy, j'ai à la fin visualisé la matrice de confusion du modèle.

```
1 %python
2 # DECISION TREES
3 from pyspark.ml.classification import DecisionTreeClassifier
4 DT = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'labelIndex')
5 model = DT.fit(train)
6 predictions = model.transform(test)
7 predictions.groupBy("prediction").count().show()
```

► (10) Spark Jobs

► predictions: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 16 more fields]

```
+-----+-----+
|prediction|count|
+-----+-----+
|      0.0|39249|
|      1.0|27081|
+-----+-----+
```

Command took 15.03 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:17:38 on My Cluster

```
7 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
8
9 evaluator = MulticlassClassificationEvaluator(labelCol="labelIndex", predictionCol="prediction")
10 accuracy = evaluator.evaluate(predictions)
11 print("Accuracy = %s" % (accuracy))
12 print("Test Error = %s" % (1.0 - accuracy))
```

► (2) Spark Jobs

```
+-----+-----+
|labelIndex|prediction|
+-----+-----+
|      0.0|      0.0|
|      1.0|      0.0|
|      0.0|      0.0|
|      1.0|      0.0|
|      0.0|      0.0|
|      0.0|      0.0|
|      0.0|      0.0|
|      1.0|      0.0|
|      0.0|      0.0|
|      0.0|      0.0|
+-----+-----+
```

only showing top 10 rows

Accuracy = 0.7276193212548361
Test Error = 0.2723806787451639

Command took 3.56 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:18:30 on My Cluster

```

1 %python
2
3 from pyspark.mllib.evaluation import MulticlassMetrics
4 from pyspark.sql.types import FloatType
5 import pyspark.sql.functions as F
6
7 preds_and_labels = predictions.select(['prediction', 'labelIndex']).withColumn('labelIndex', F.col('labelIndex').cast(FloatType())).orderBy('prediction')
8 preds_and_labels = preds_and_labels.select(['prediction', 'labelIndex'])
9 metrics = MulticlassMetrics(preds_and_labels.rdd.map(tuple))
10 print(metrics.confusionMatrix().toArray())

```

► (4) Spark Jobs

► preds_and_labels: pyspark.sql.dataframe.DataFrame = [prediction: double, labelIndex: float]

[[27268. 5934.]

[11981. 21147.]]

Command took 6.43 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:20:20 on My Cluster

Les GBT ont obtenu environ 75% d'accuracy (le meilleur modèle), j'ai à la fin visualisé la matrice de confusion du modèle.

```

1 %python
2 # GRADIENT BOOSTED TREES
3 from pyspark.ml.classification import GBTClassifier
4
5 gbt = GBTClassifier(featuresCol="features", labelCol = 'labelIndex')
6 model = gbt.fit(train)
7 predictions = model.transform(test)
8 predictions.groupBy("prediction").count().show()

```

► (5) Spark Jobs

► predictions: pyspark.sql.dataframe.DataFrame = [duration_sec: integer, start_time: string ... 16 more fields]

```

+-----+-----+
|prediction|count|
+-----+-----+
| 0.0|36615|
| 1.0|29715|
+-----+-----+

```

Command took 1.57 minutes -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:21:54 on My Cluster

```

1 %python
2
3 predictions.select("labelIndex", "prediction").show(10)
4
5
6
7 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
8
9 evaluator = MulticlassClassificationEvaluator(labelCol="labelIndex", predictionCol="prediction")
10 accuracy = evaluator.evaluate(predictions)
11 print("Accuracy = %s" % (accuracy))
12 print("Test Error = %s" % (1.0 - accuracy))

```

► (2) Spark Jobs

```

+-----+-----+
|labelIndex|prediction|
+-----+-----+
| 0.0| 0.0|
| 1.0| 0.0|
| 0.0| 0.0|
| 1.0| 0.0|
| 0.0| 0.0|
| 0.0| 0.0|
| 0.0| 0.0|
| 1.0| 0.0|
| 0.0| 0.0|
| 0.0| 0.0|
+-----+-----+

```

only showing top 10 rows


Accuracy = 0.7451117819355649

Test Error = 0.25488821806443507

Command took 5.68 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:23:36 on My Cluster

```
1 %python
2
3 from pyspark.mllib.evaluation import MulticlassMetrics
4 from pyspark.sql.types import FloatType
5 import pyspark.sql.functions as F
6
7 preds_and_labels = predictions.select(['prediction','labelIndex']).withColumn('labelIndex', F.col('labelIndex').cast(FloatType())).orderBy('prediction')
8 preds_and_labels = preds_and_labels.select(['prediction','labelIndex'])
9 metrics = MulticlassMetrics(preds_and_labels.rdd.map(tuple))
10 print(metrics.confusionMatrix().toArray())
```

▶ (4) Spark Jobs

▶  preds_and_labels: pyspark.sql.dataframe.DataFrame = [prediction: double, labelIndex: float]

[[26478. 6724.]

[10137. 22991.]]

Command took 8.02 seconds -- by wassila.seghier@etu.univ-amu.fr at 17/10/2021, 15:23:50 on My Cluster