SABD-PROJECT1

Introduzione

L'obbiettivo di questo report è quello di presentare l'implementazione ed i risultati ottenuti nella realizzazione del progetto di Sistemi e Architetture per i Big Data dell'anno 2020-2021.

Per lo sviluppo di questo progetto è stato utilizzato il framework Spark per analizzare i dati presi dalla repository github delle vaccinazioni in Italia contro il Covid-19, in modo tale da poter rispondere alle seguenti domande:

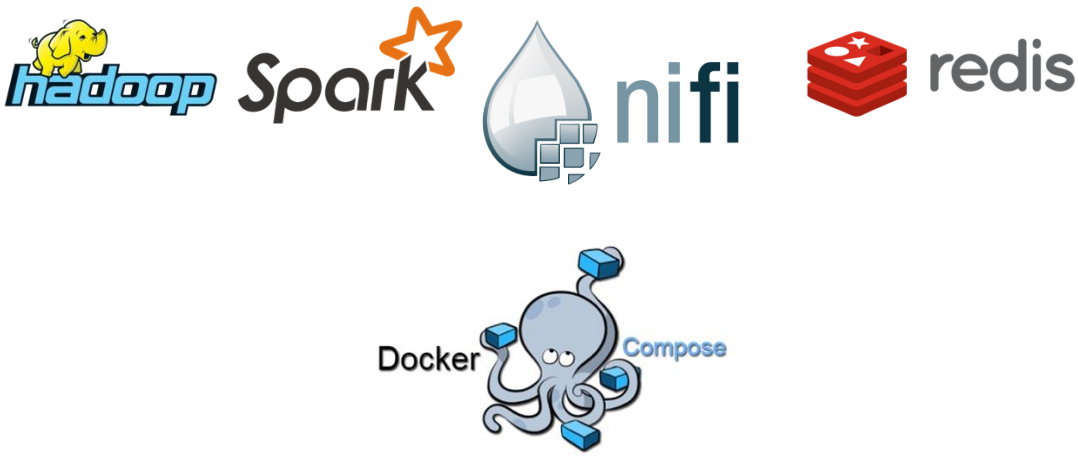
- Query1: calcolare la media delle vaccinazioni per un generico centro per ciascuna regione e ciascun mese

- Query2: individuare per ciascun mese e ciascuna fascia d'età le prime cinque regioni per le quali si prevede il maggior numero di vaccinati di sesso femminile

- Query3: stimare per ciascuna regione il numero totale di somministrazioni effettuate il 1° giugno 2021 e classificare tali risultati attraverso un algoritmo di clustering K-means e Bisecting K-means

Il codice che è stato realizzato può essere reperito in due versioni differenti: la prima all'interno del package src/main/java/queries e la seconda all'interno del package src/main/java/sql_queries, nella quale è stata utilizzata la versione SQL di Spark.

Architettura del sistema



Docker compose

Docker compose è uno strumento per la definizione e l'esecuzione di applicazioni Docker multi-containers. Si utilizza un file "docker-compose.yml" per configurare i servizi necessari all'esecuzione del progetto in modo tale che questi possano eseguire contemporaneamente e in maniera isolata. Con un solo comando si è in grado di creare e avviare tutti i servizi specificati all'interno della configurazione, avendo in più la possibilità di specificare il numero di spark-worker e datanode che si vuole andare ad utilizzare.

In questo caso particolare, i containers che vengono istanziati sono:

- Spark-master/spark-worker
- Namenode/Datanode
- Redis
- Nifi

Apache Spark

Apache Spark è un framework open source per il processamento dei Big Data, che mette a disposizione API di alto livello per numerosi linguaggi di programmazione e tool come SparkSQL.

La scelta di utilizzare tale framework rispetto a MapReduce è stata dettata dalla dimensione del dataset, poiché essendo abbastanza piccolo è possibile contenerlo tutto in memoria traendo dei vantaggi nel processamento rispetto al salvataggio su disco. Inoltre, si è deciso di utilizzarlo per il maggior numero di funzioni messe a disposizione rispetto alle sole map e reduce.

Apache Nifi

Apache NiFi è un tool affidabile e potente progettato per automatizzare il flusso di dati tra i sistemi software. Per tali ragioni, in questo progetto è stato utilizzato per andare a gestire il routing dei dati fra i conteiners istanziati.

La scelta di tale sistema è stata dettata dall'interfaccia grafica semplice e intuitiva e dalla buona documentazione reperibile online.

Apache Hadoop

Apache Hadoop è un framework open source che permette il processamento distribuito di grandi quantità di dati all'interno di cluster di computer utilizzando un modello di programmazione semplice.

In questo progetto viene utilizzato per memorizzare sia i dati di input alle query eseguite su Spark, sia i risultati ottenuti.

Redis

Redis è un key-value store open source residente in memoria, adatto per la memorizzazione veloce dei dati ed è stato utilizzato per esportare tutti i dati delle query.

Implementazione

Query

Come già detto nell'introduzione, le query sono state realizzate in due versioni differenti, una in cui viene ad essere utilizzata la versione SQL di Spark e in una in cui viene utilizzata l'api RDD. Tale scelta è stata dettata dal tentativo di andare a effettuare le predizioni, richieste nella query2 e nella query3, grazie all'uso della Linear Regression messa a disposizione dalla libreria mllib. Nonostante ciò, questa alla fine non è stata utilizzata in quanto ha portato notevoli limitazioni:

- Necessità di utilizzo di un ciclo for
- Degrado delle prestazioni e tempi di computazione esponenzialmente più lunghi
- Obbligo di utilizzo di un dataset, che ha portato a un aumento delle operazioni necessarie nel caso delle query2 e query3, dovuto alla necessità di riconvertire i JavaPairRDD in dataset. Per ovviare a tale problema si è tentato di utilizzare SQL Spark, andando quindi a lavorare direttamente sui dataset, ma le operazioni necessarie al processamento dei dati non hanno portato miglioramenti

Per tali motivi si è scelto di andare a utilizzare all'interno del codice, per effettuare le predizioni dei valori, la Simple Regression, ovvero una implementazione della regressione lineare della libreria Math3 di Apache.

Inoltre, un'ulteriore decisione che si è presa in merito all'implementazione della singola query 3 è stata quella di lasciarla "aperta": non è stata impostata in modo tale da effettuare una previsione solamente al 1 Giugno 2021, ma è stato impostato un calcolo delle previsioni al giorno successivo.

Nifi

Per l'ingestion dei dati sono stati utilizzati due templates:

input.xml

Questo template si occupa di prendere i dati dalla repository github e inserirli in hdfs. È stato creato un process group per ogni singolo file da importare in cui viene creato un flow file per convertirlo da formato csv a formato parquet e sul quale viene ad essere realizzata una "query" per effettuare il pre-processamento dei dati.

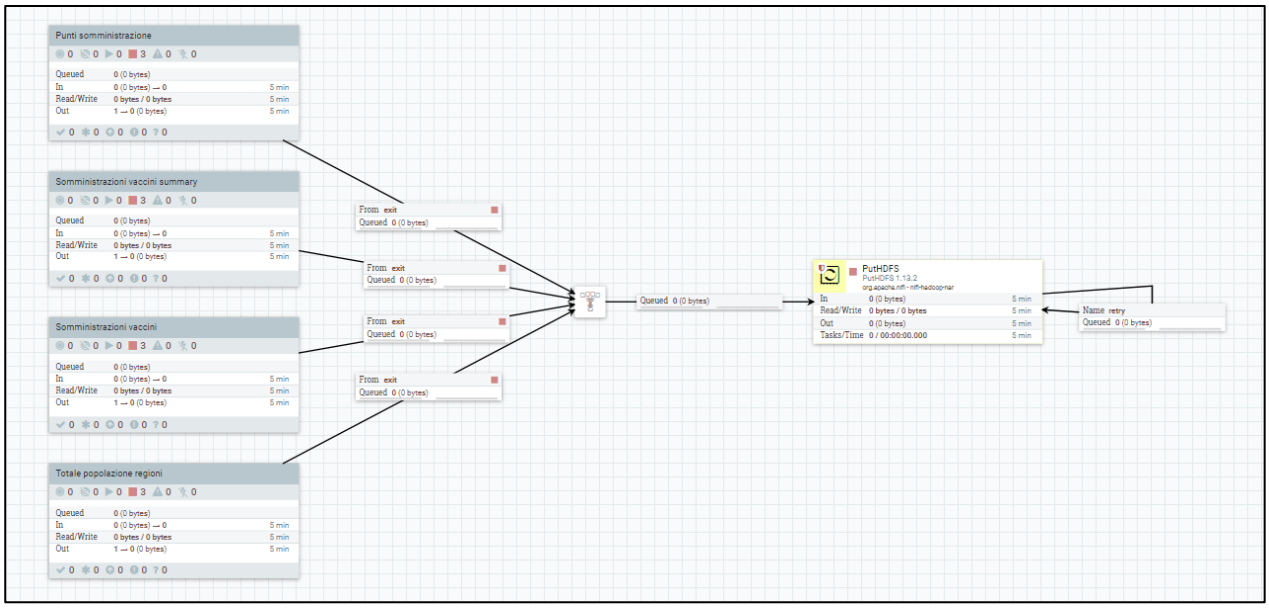
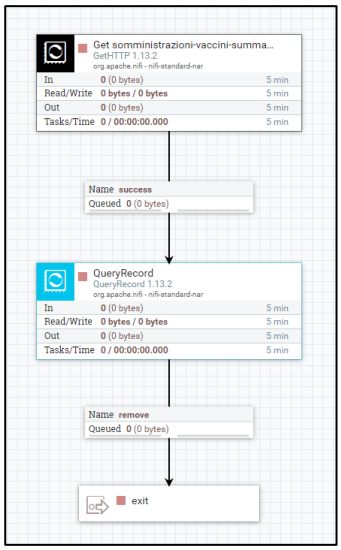


Figura 1 - template input.xml



Come esempio di pre-processamento dei dati si riporta l'immagine relativa al file "somministrazioni-vaccini-summary.csv" nel quale è stato effettuato anche un ordinamento dei dati in base alla data, come richiesto nella specifica del progetto.

Figura 2 - process group somministrazioni vaccini summary

redis.xml

Questo template viene usato per prendere i risultati delle query da hdfs e salvarli su redis.

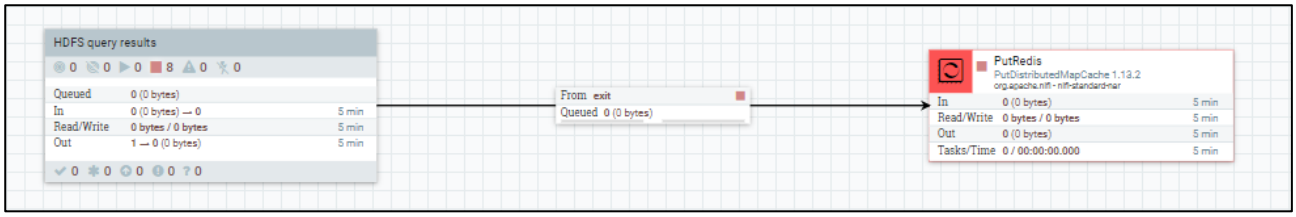


Figura 3- template redis.xml

Prima di essere inseriti nel sistemai file vengono modificati tramite una serie di operazioni.

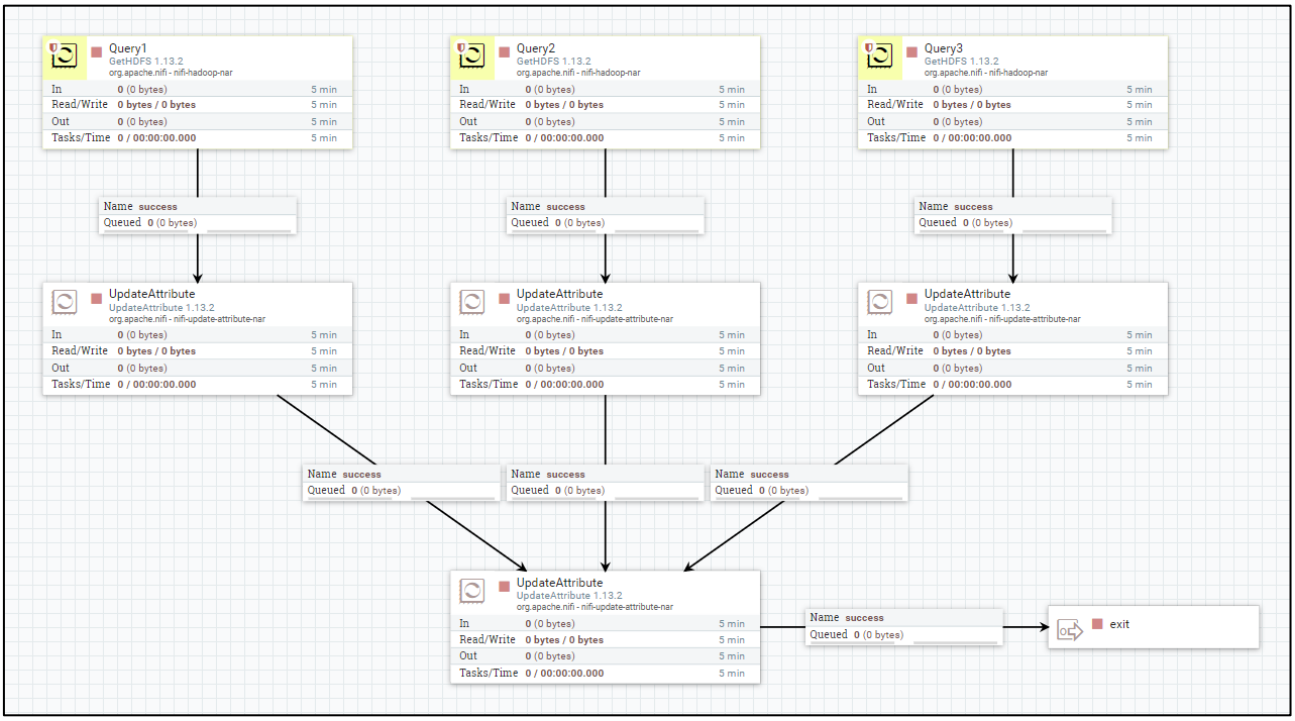


Figura 4 - process group HDFS query results

In particolare, nel process group si va a:

- Prendere i risultati delle query
- Cambiare il nome dei risultati ponendo davanti il numero della query corrispondente
- Mettere davanti al nome anche la data odierna

Deployment

Come specificato precedentemente per sviluppare il progetto è stato utilizzato Docker-compose, è possibile quindi avviare l'applicazione andando ad eseguire contemporaneamente tutti i servizi, in maniera isolata e specificando anche il numero di worker nodes che si vuole istanziare, rendendo il sistema più scalabile. Per fare ciò viene utilizzato il comando:

```
docker-compose up --scale spark-worker=3 --scale datanode=4
```

e.g., cluster con 3 spark worker e 4 hdfs datanode.

Al primo deployment del cluster, inoltre, è necessario, per poter andare ad utilizzare nifi, importare i templates, sopra citati, memorizzati nel folder /nifi/templates.

Submit query

Per effettuare il submit delle query è stato realizzato uno script che prende come parametri il numero della query che si intende andare ad eseguire e, nel caso della query 3 e sql_query3, l'algoritmo da utilizzare ed il numero k di cluster.

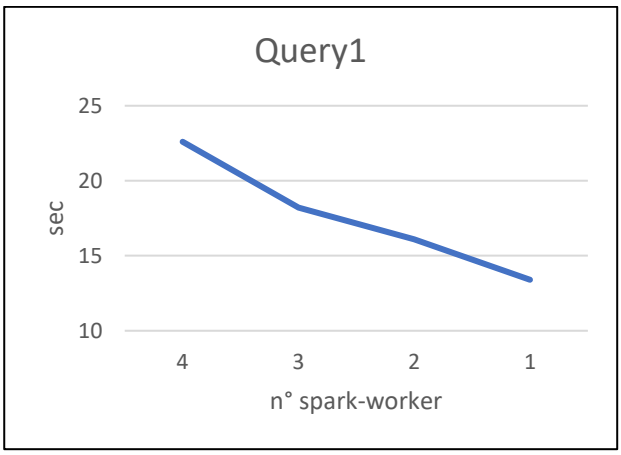
Per tanto è sufficiente eseguire i seguenti comandi:

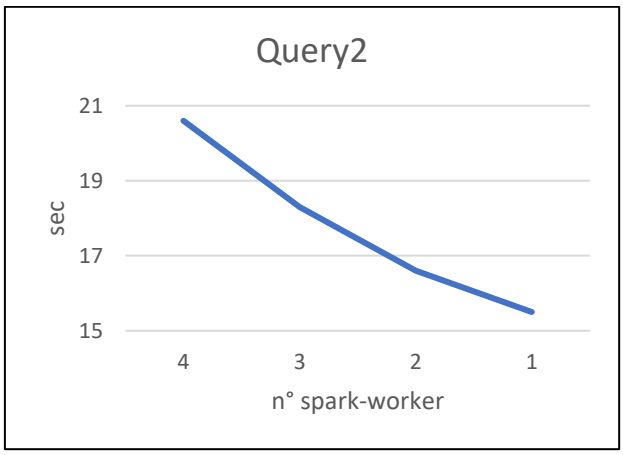
- mvn package: crea il file jar
- sh submit_query.sh <numero della query> <eventuali parametri>

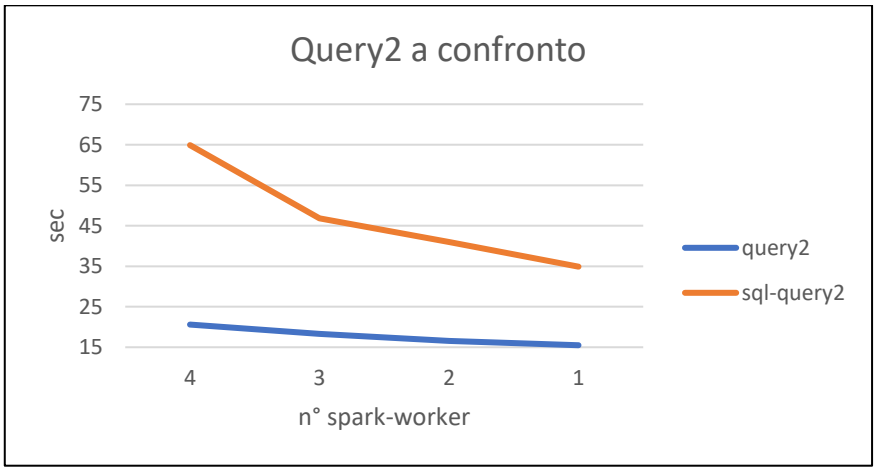
Test

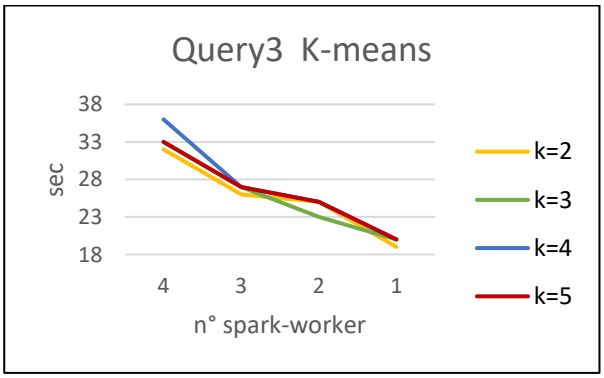
I test sono stati svolti utilizzando un processore intel core i-7 3770. È stata presa in considerazione un'architettura composta da due datanode hdfs e numero variabile di spark worker da 1 a un massimo di 4. Per le due versioni della query 3 è stato inoltre testato anche il caso in cui il numero k di cluster variasse da 2 a 5.

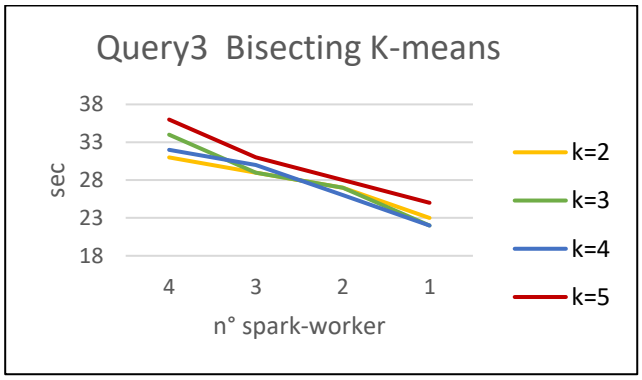
Di seguito vengono riportati i risultati dei test:

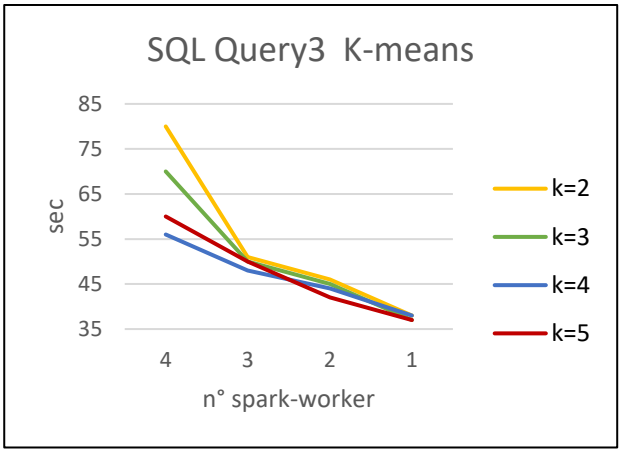


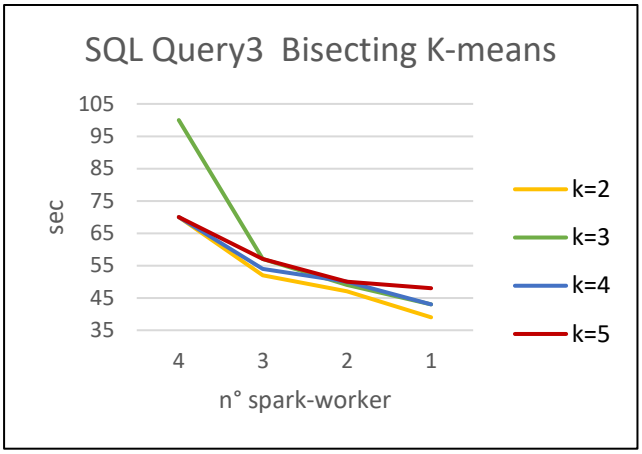


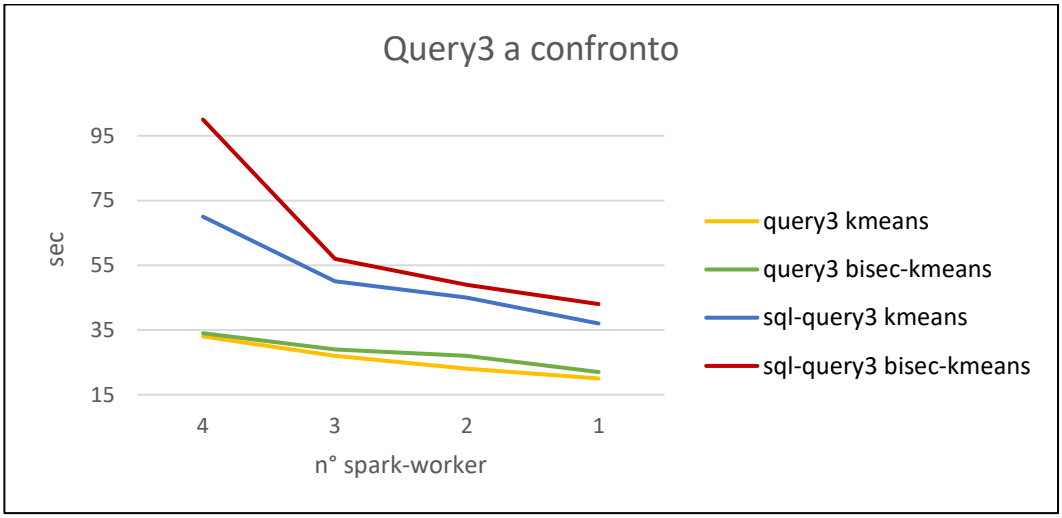












Dai grafici riportati si può vedere che all'aumentare dei nodi worker in esecuzione il tempo di processamento aumenta. Ciò potrebbe essere dovuto al fatto che il dataset analizzato, nonostante sia stato trattato come un dataset di grandi dimensioni, alla fine non risulta essere effettivamente tale.

Conclusione

Si può quindi concludere la discussione dei risultati evidenziando che sono state riscontrate performance migliori nelle query che fanno uso della api RDD, mentre l'utilizzo di Spark SQL ha portato ad un peggioramento dei tempi di processamento. Inoltre, è possibile notare come l'uso dell'algoritmo di Bisecting K-means porta a dei tempi di poco peggiori rispetto al K-means.