Data-intensive Programming

Programming Assignment

Version 0.2
Changes to the version 0.1 are highlighted with yellow

Description

The course has a compulsory programming assignment, which is done in groups of three students. Grouping is done in Moodle. Code repositories for the groups will be generated by the course staff. The repositories are created in the Gitlab. The data and some helper application are published in Gitlab project *assignment* (https://course-gitlab.tut.fi/dip-18/assignment), which should be pulled regularly.

For the basic level, you are given publicly available data of road traffic accidents in Finland during 2015 [https://www.avoindata.fi/data/en_GB/dataset/tieliikenneonnettomuudet]. The data is in csv format and it has 101 columns like: *Onnett_id;Tienpit;Tienpitsel;Tie;Aosa;Aet;Ajr;… x, y, …* The dataset is available in your group's repository in the directory *data*. The coordinate system used in the dataset is *ETRS-TM35FIN*. Group's repository will contain a coordinate converter from ETRS-TM35FIN to WGS84, which can be used with Google map, for instance.

Here are the first few lines and columns of the dataset:

```
Onnett_id;Tienpit;Tienpitsel;Tie;Aosa;Aet;Ajr;Vuosi;Kk;…
6891249;1;Liikennevirasto;1;2;2554;1;2015;9;…
6681113;1;Liikennevirasto;1;3;6;2;2015;1;…
6840692;1;Liikennevirasto;1;3;8;1;2015;7;…
…
```

Even if the amount of data is not actually big, you should write your implementation as if it was. Moreover, Spark should be programmed with Scala.

The minimum amount of points that a group must earn is 7 and maximum is 15 points. The points are direct points in the exam for each group member. The group should commit a working project in the group's repository with a document describing which tasks they have completed and the amount of work for each group member.

Task #1: Basic (2 points)

The task is to use implement *k-means clustering* with Apache Spark for clustering the data based on the accident locations (x, y). The solution computes, with DataFrames and MLlib, a reasonable number (chosen by you) of cluster means based on the x and y coordinates. The result is written in csv format to the file *results/basic.csv*.

The result file should have a header line with columns x and y. Moreover, each line has mean coordinates. Here is an example of results/basic.csv:

```
x,y
322134.957143, 6675603.585714
283447.728395, 6970046.308642
229626.707317, 6990844.967480
```

The program should be able to handle dirty and erroneous data. It should be well-written, easy to understand, be reasonably commented.

Task #2: Third Dimension (3 points)

Do accidents happen in different locations, if the day of the week is used as a third dimension? This question can be studied by adding a third dimension - *dayOfWeek* - to the data. The day of the week is in the column *Vkpv*.
The result is written in csv format to the file *results/task2.csv*.

The result file should have a header line with columns x, y and dow. Moreover, each line has mean coordinates. Here is an example of results/task2.csv:

```
x,y
322134.957143, 6675603.585714, 1
283447.728395, 6970046.308642, 1
229626.707317, 6990844.967480, 2
```

Here are translations from Finnish weekdays to English ones:

Number in csv	Finnish	English
1	Maanantai	Monday
2	Tiistai	Tuesday
3	Keskiviikko	Wednesday
4	Torstai	Thursday
5	Perjantai	Friday
6	Lauantai	Saturday
7	Sunnuntai	Sunday

Task #3: Streaming Data (2 points)

Unfortunately, implementing streaming K-Means seems not be doable using Datasets/DataFrames with any existing version MLlib. Therefore, this task is removed as such. These two points can be earned by sketching the streaming implementation in the document. Sorry, for the inconvenience.

~~Note: There might be problem with this task when using the Spark version, we have been using. This task will be revised in near future.~~

~~Change the basic solution to work with streaming DataFrames. The data is read as files from the directory *streamingData*. This time the data is not written into a file, but instead streaming action is used to output result to stdout.~~

Task #4: Elbow Method (3 points)

Find an optimal number of the clusters with the Elbow method, or justify why it does not exist. Include a graph, that shows the found elbow, to your submission. Find the elbow for

1. Basic case (#1)
2. Third dimension case (#2)
3. Streaming Data (#3) (Elbow should, also be computed in a streaming way)

For the Basic case and Third dimension case document the computation of elbow into the document you submit. In the streaming case, the streaming app should output "Elbow = n" standard output, where n is the elbow point.

Task #5: K-Means Clustering with RDDs (3 points)

Implement the whole clustering algorithm yourself (without MLlib) for three dimensions (like in the Task #2) using the lower-level API.

The result is written to results/task5.csv similarly to Task #1.

Task #6: Elbow Method with RDDs (2 points)

Add computing the elbow for the Task #5. The elbow results are documented similarly as in Task #4.

Google Maps Hint

You can visualize the data by importing it to Google Maps: menu->"Your Places", then go to the *Maps* tab and choose "CREATE MAP". By adding new layers to the newly-created map, you can import CSV files and visualize them on the map.

There is a helper program convert.py for converting the ETRS-TM35FIN coordinates to WGS-84. The program usage is

```
utilities/convert.py inFile.csv outFile.csv
```

where inFile.csv is a csv file with three columns: x, y and label. The column label is meant for your own purposes and it is optional.

Here is an example of inFile.csv:

```
X,Y,LABEL
322134.957143, 6675603.585714, 'size=70'
283447.728395, 6970046.308642, 'Size=81'
229626.707317, 6990844.967480, 'Size=123'
```

and here the corresponding outFile.csv:

LONGITUDE,LATITUDE,LABEL
23.7930521547,60.1784348268,'size=70'
22.7520307619,62.7963721062,'Size=81'
21.6674284839,62.9464984208,'Size=123'

The outFile.csv can be uploaded to Google Maps.

Submission

The assignment is submitted by commiting the source code and other additional files to the group's repository and returning the commit hash to Moodle.

Spark Program

The submission should include spark App with name Assignment, that takes the following command line arguments:
`-task n`
where n is 1, 2, 3, 5 or 6.
E.g. executing the application with arguments
`-task 5`
runs K-Means clustering with RDDs as described in Section #5. If the group has not implemented task n then the program print "Not implemented!".

Document

The group must also return a document describing which tasks they have completed and the amount of work for each group member. Moreover, if the group has done Task #6, then the result with images should be in the document. The document has to be in PDF format and it should be placed in docs directory.

The deadline is Oct 19.