Assignment 5

Return your solutions via Moodle page. We do not count belated returnings. You have to use .pdf file type for any written answers, and .scala for your code. Do not return any project or object files, etc. Mark your **full name and student number** clearly to all your solution files. Each exercise will be graded pass/fail. The purpose of this exercise set is to use spark cluster and learn to use spark advanced methods to optimize the performance of spark applications.

Exercise 1 (3 Points)

Data can be in binary format that needs to be first processed before the actual analysis can be performed. In this exercise, you need to download an image dataset, convert images to RDD, and then analyze the RDD data.

Download the flower image data set from
http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html

Write a function that takes the flower pictures to RDD, one picture as an element. You can use image processing tools in Java, e.g. https://docs.oracle.com/javase/tutorial/2d/images/ when the RDD should look like RDD[BufferedImage]. Your classes for the image reading and processing can be made by Java (which is also a valid Scala). Read also how to separate the RGB colors from the image:
http://stackoverflow.com/questions/2615522/java-bufferedimage-getting-red-green-and-blue-individually
For Python,
http://stackoverflow.com/questions/138250/how-can-i-read-the-rgb-value-of-a-given-pixel-in-python

Then implement (using Spark) a function that performs 1NN classifying, e.g., find the nearest neighbor for each flower. As neighbors, use references, or plain color images from http://en.wikipedia.org/wiki/List_of_colors:_A%E2%80%93F (use about 10-30 different colors). As an output, you should have an information, that which is the most dominant color in the figure, based on the comparison between the flower images and the references. Perform the comparison using RGB values. Use RDD and distribution as efficiently you can.

Tip: You can start with an RDD of file names, and do the image loading in a map() call. Avoid loading all of the files at the main program level, this way you may run out of memory.

Tip2: Choose reference colors that distribute the flower set well, e.g. different blues, yellows and reds. I you have a reference of grey, you probably end up having the grey as a closest neighbor for almost every flower.

Return your code with clarifying comments as a solution (.scala and/or .python files).

Exercise 2 Collaborative Filtering Summary (2 Points)

Movielens dataset https://grouplens.org/datasets/movielens/
A Sample Scala implementation of collaborative filtering is provided in the following github url,
https://github.com/amplab/training/blob/6fcefb16eb5f91d4a6e161724e0acd0d1a96f541/machine-learning/scala/solution/MovieLensALS.scala
A detailed documentation for the code is provided at,
http://ampcamp.berkeley.edu/big-data-mini-course/movie-recommendation-with-mllib.html

Copy the example code in a scala file and add comments according to the following Explain the usage of various Spark actions and transformations in ***context of the algorithm*** *and the return the code with comments.*

Optional Exercise (3 Points)

The following section is optional and carries an additional 3 points. If you have already received, full score in all your assignments, these will be added into your exam points. *Implement the Matrix Factorization Model for the movielens data set. And, report the final accuracy of your model.* Is your accuracy greater than 88%(Read the code documentation in Exercise 2 for more attributes wise comparison).

1. Code with proper documentation,
2. Report with a short note on Matrix Factorization, detailed accuracy analysis and, detailed notes on model parameters and their calibration.

Note: If you prefer, you are also allowed to perform Collaborative Filtering through different methods instead of using Matrix Factorization Model.

References:
https://en.wikipedia.org/wiki/Collaborative_filtering
https://spark.apache.org/docs/2.1.0/mllib-collaborative-filtering.html
https://link.springer.com/chapter/10.1007%2F978-3-540-68880-8_32