SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT337

End-of-Course Assessment - July Semester 2023

Big Data Computing in the Cloud

INSTRUCTIONS TO STUDENTS:

1. This End-of-Course Assessment paper comprises **7** pages (including the cover page).

2. You are to include the following particulars in your submission: Course Code, Title of the ECA, SUSS PI No., Your Name, and Submission Date.

3. Late submission will be subjected to the marks deduction scheme. Please refer to the Student Handbook for details.

IMPORTANT NOTE

ECA Submission Deadline:
Tuesday, 07 November 2023 12:00 pm

<u>ECA Submission Guidelines</u>

Please follow the submission instructions stated below:

This ECA carries 70% of the course marks and is a compulsory component. It is to be done individually and not collaboratively with other students.

Submission

You are to submit the ECA assignment in exactly the same manner as your tutor-marked assignments (TMA), i.e. using Canvas. Submission in any other manner like hardcopy or any other means will not be accepted.

Electronic transmission is not immediate. It is possible that the network traffic may be particularly heavy on the cut-off date and connections to the system cannot be guaranteed. Hence, you are advised to submit your assignment the day before the cut-off date in order to make sure that the submission is accepted and in good time.
Once you have submitted your ECA assignment, the status is displayed on the computer screen. You will only receive a successful assignment submission message if you had applied for the e-mail notification option.

ECA Marks Deduction Scheme

Please note the following:

(a) Submission Cut-off Time – Unless otherwise advised, the cut-off time for ECA submission will be at 12:00 noon on the day of the deadline. All submission timings will be based on the time recorded by Canvas.

(b) Start Time for Deduction – Students are given a grace period of 12hours. Hence calculation of late submissions of ECAs will begin at 00:00 hrs the following day (this applies even if it is a holiday or weekend) after the deadline.

(c) How the Scheme Works – From 00:00 hrs the following day after the deadline, 10 marks will be deducted for each 24-hour block. Submissions that are subject to more than 50 marks deduction will be assigned zero mark. For examples on how the scheme works, please refer to Section 5.2 Para 1.7.3 of the Student Handbook.
Any extra files, missing appendices or corrections received after the cut-off date will also not be considered in the grading of your ECA assignment.

Plagiarism and Collusion

Plagiarism and collusion are forms of cheating and are not acceptable in any form of a student's work, including this ECA assignment. You can avoid plagiarism by giving appropriate references when you use some other people's ideas, words or pictures (including diagrams). Refer to the American Psychological Association (APA) Manual if you need reminding about quoting and referencing. You can avoid collusion by ensuring that your submission is based on your own individual effort.
The electronic submission of your ECA assignment will be screened through a plagiarism detecting software. For more information about plagiarism and cheating, you should refer to the Student Handbook. SUSS takes a tough stance against plagiarism and collusion. Serious cases will normally result in the student being referred to SUSS's Student Disciplinary Group.

(Full marks: 100)

Question 1

Question 1a

Use a table to highlight the differences between Apache Hadoop versus Apache Spark.

(5 marks)

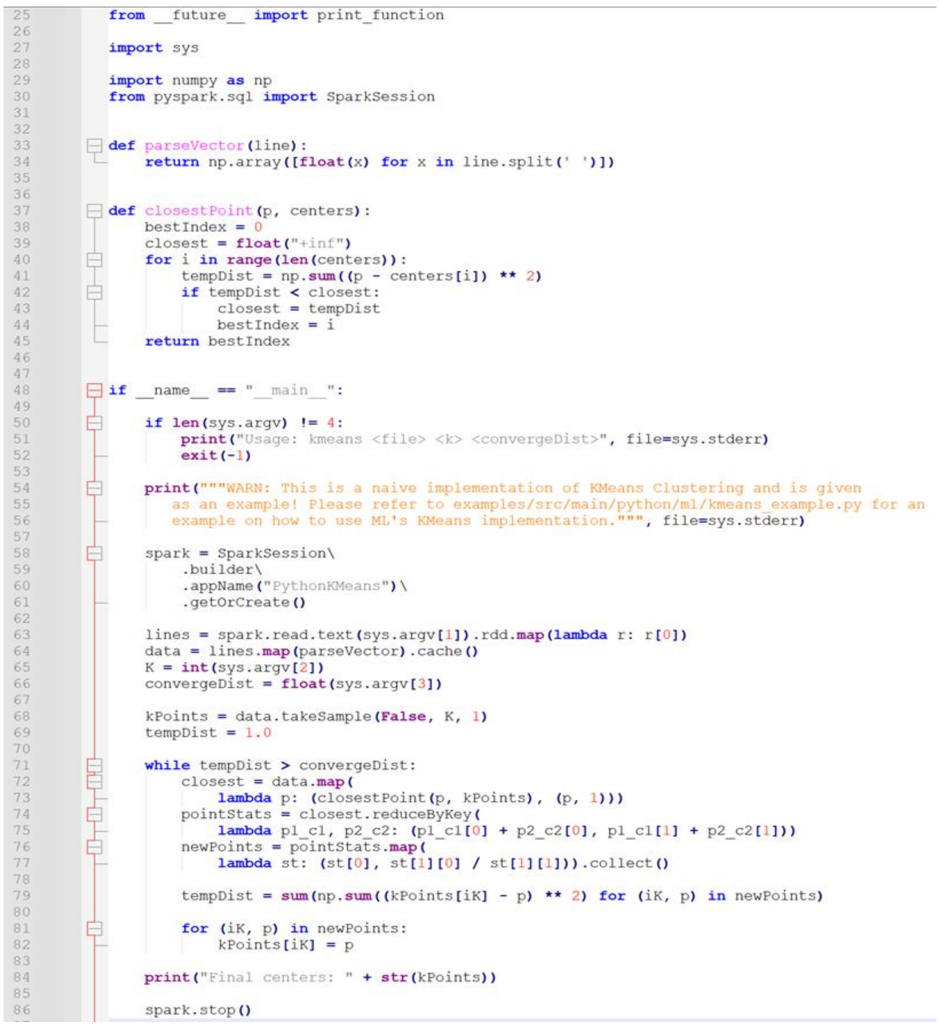
Question 1b

Explain in details on the Spark job execution process in a cluster computing setup.

(10 marks)

Question 2

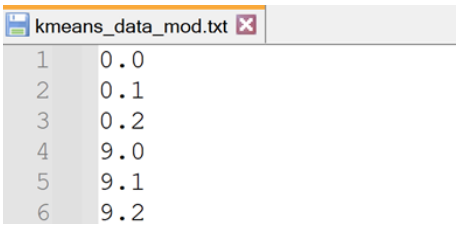
K-Means algorithm is one of the popular unsupervised learning approach to perform data point clustering. Explain the key logic of kmeans.py built-in PySpark program, as shown in Figure 1.



```python
from __future__ import print_function

import sys

import numpy as np
from pyspark.sql import SparkSession


def parseVector(line):
    return np.array([float(x) for x in line.split(' ')])


def closestPoint(p, centers):
    bestIndex = 0
    closest = float("+inf")
    for i in range(len(centers)):
        tempDist = np.sum((p - centers[i]) ** 2)
        if tempDist < closest:
            closest = tempDist
            bestIndex = i
    return bestIndex


if __name__ == "__main__":

    if len(sys.argv) != 4:
        print("Usage: kmeans <file> <k> <convergeDist>", file=sys.stderr)
        exit(-1)

    print("""WARN: This is a naive implementation of KMeans Clustering and is given
      as an example! Please refer to examples/src/main/python/ml/kmeans_example.py for an
      example on how to use ML's KMeans implementation.""", file=sys.stderr)

    spark = SparkSession\
        .builder\
        .appName("PythonKMeans")\
        .getOrCreate()

    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
    data = lines.map(parseVector).cache()
    K = int(sys.argv[2])
    convergeDist = float(sys.argv[3])

    kPoints = data.takeSample(False, K, 1)
    tempDist = 1.0

    while tempDist > convergeDist:
        closest = data.map(
            lambda p: (closestPoint(p, kPoints), (p, 1)))
        pointStats = closest.reduceByKey(
            lambda p1_c1, p2_c2: (p1_c1[0] + p2_c2[0], p1_c1[1] + p2_c2[1]))
        newPoints = pointStats.map(
            lambda st: (st[0], st[1][0] / st[1][1])).collect()

        tempDist = sum(np.sum((kPoints[iK] - p) ** 2) for (iK, p) in newPoints)

        for (iK, p) in newPoints:
            kPoints[iK] = p

    print("Final centers: " + str(kPoints))

    spark.stop()
```

Figure 1: Snap-shot of kmeans.py built-in PySpark example

To facilitate your program testing, you may create an input text file with the content as shown in Figure 2. The number of cluster, parameter *K* is set to 2 and convergeDist is set to 0.1.



```
0.0
0.1
0.2
9.0
9.1
9.2
```

Figure 2: Example input to test kmeans.py

(15 marks)

Question 3

In your local machine's Spark setup, develop a PySpark program using **Spark DataFrame APIs** to perform the following tasks. Show your full PySpark program and provide screenshots for all key steps where applicable.

Data sources used in this question are: (i) vehicle_mpg.tsv, (ii) vehicle_manufacturers.csv. Note that these data files can be downloaded from ICT337 Canvas webpage.

Question 3a

Read the "vehicle_mpg.tsv" file and store the content with Spark DataFrame. Show the content, number of occurrences, schema and DataFrame dimension.

(3 marks)

Question 3b

Find any missing data from the DataFrame and drop the corresponding rows. Show the content and the number of occurrences. Also, perform statistical profiling for all numerical columns.

(4 marks)

Question 3c

Perform the following tasks and show the results in each step (i.e., DataFrame content and its number of occurrences):

- Create a new column called "manufacturer". Populate the vehicle manufacturer data by implementing a function to extract the first word (i.e., manufacturer) from the column "carname".
- Modify the data by appending a prefix of "19" in the data of "modelyear" column.

- Create a new column called “mpg_class” with the following classification criteria: “low” for mpg <=20, “mid” for 20 < mpg <=30, “high” for 30 < mpg <=40, and “very high” for mpg > 40.

(6 marks)

Question 3d

Read the “vehicle_manufacturers.csv” file and store the content with Spark DataFrame. Join with the DataFrame of Question 3(c) and show its content.

(2 marks)

Question 3e

Perform the following tasks and show the results in each step:

- Find the average, minimum and maximum mpg for a given country. Sort the results by average mpg from highest to lowest.
- Find the average, minimum and maximum mpg for a given cylinder. Sort the results by average mpg from highest to lowest.
- Find the average, minimum and maximum mpg for a given model year. Sort the results by average mpg from highest to lowest.
- Find the average, minimum and maximum mpg for a given manufacturer. Sort the results by average mpg from highest to lowest.
- Find the average mpg for a given car name and manufacturer. Sort the results by average mpg from highest to lowest.

(5 marks)

Question 3f

Repeat the tasks in Q3(e) by using **PySpark SQL approach**.

(5 marks)

Question 4

In your local machine’s Spark setup, develop a PySpark program using **<u>PySpark RDD APIs</u>** to perform the following tasks. Show your full PySpark program and provide screenshots and results for all key steps where applicable.

Data sources used in this question are: (i) mov_rating.dat, (ii) mov_item.dat, (iii) mov_genre.dat, (iv) mov_user.dat, and (v) mov_occupation.dat. Note that these data files can be downloaded from ICT337 Canvas webpage.

Question 4a

Perform the following tasks and show the results in each step:

- Find the total number of user/reviewer rating records.
- Find the unique number of reviewers.
- Find the unique number of movies reviewed.
- Find the **Top TEN** (10) reviewers that review the most movies. Show the reviewer ID and its total reviewed movie counts in descending order (i.e., highest to lowest counts)
- Find the **Top TEN** (10) most popular movies by the number of review counts. Show the movie ID, movie name, reviewed counts in descending order (i.e., highest to lowest counts)

(9 marks)

Question 4b

Perform the following tasks and show the results in each step:

- Based on the movie release date, find the total number of movies released per year. Show the movie released year and the corresponding counts.
- Find the range of movie released year.
- For the above year with the highest counts, find the **Top THREE** (3) most reviewed movies per genre, sorted by average review ratings (i.e., highest to lowest ratings). Note that there are **NINETEEN** (19) genre categories. Show the genre name, movie ID, movie name, and average rating. Save the top three movie results using RDD file saving mechanism and show the content.

(10 marks)

Question 4c

Create an age-group category of: [0,6], (6,12], (12,18], (18,30], (30,50], 50+ and assign corresponding category to user/reviewer based on his/her age. For each age-group, find the **Top THIRTY** (30) most reviewed movies, sorted by average review ratings (i.e., highest to lowest ratings). Show the total movie counts for the occupation category, as well as age-group, movie ID, movie name, and average rating. Save the top thirty movie results using RDD file saving mechanism and show the content.

(6 marks)

Question 4d

We like to analyze the ratings for movies released in Summer, i.e., May, June and July. For each genre category in these three months, find the **Top THREE** (3) most reviewed movies, sorted by average review ratings (i.e., highest to lowest ratings). Show the genre name, movie ID, movie name, and average rating. Save the top three movie results using RDD file saving mechanism and show the content.

(6 marks)

Question 4e

There are **TWENTY-ONE** (21) categories of occupation and **NINETEEN** (19) genre categories. For each occupation and genre categories, find the **Top THREE** (3) most reviewed movies, sorted by average review ratings (i.e., highest to lowest ratings). Display all results by showing the occupation name, genre name, movie ID, movie name, and average rating.

What are the top three rating results for reviewers with "administrator" occupation that review "action" genre?

(8 marks)

Question 4f

We like to build movie recommendation model using the Alternating Least Squares (ALS) algorithm in pyspark.mllib package.

To train our ALS model, perform the following tasks and show the results in each step (i.e., sample RDD content and its total count):

- For the input file used to train the ALS model, we first artificially create a new user profile of user ID = 0 by inserting the following rating records to the front of the "mov_rating.dat":

```
0 50 5 881250949
0 172 5 881250949
0 181 5 881250949
```

Here, we intend to create a fake user that enjoys the movies of Star Wars, Empire Strikes Back and Return of the Jedi.

- To train the ALS model, we use the API: `ALS.train(ratings, rank, numIterations)`. The rank is set to **20** and numIterations is set to **15**. For reference, you may refer to: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.mllib.recommendation.ALS.html
- Use the trained model to find the Top **TEN** (10) movie recommendation for user ID = 0. We can use the API: `model.recommendProducts(user,num)`. For reference, you may refer to: https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.mllib.recommendation.MatrixFactorizationModel.html

(6 marks)

----- END OF ECA PAPER -----