SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT233

End-of-Course Assessment - January Semester 2023

Data Programming

INSTRUCTIONS TO STUDENTS:

1. This End-of-Course Assessment paper comprises **8** 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: Sunday, 14 May 2023 12:00 pm

ECA Submission Guidelines

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. For other cases, significant marking penalties or expulsion from the course will be imposed.

(Full marks: 100)

Question 1

Objectives:

- Understand dataset with Data Scientist mindset.
- Exposure to real-world dataset analysis.
- Understand and design computation logic and routines in Python.
- Assess use of Pandas and Dataframes to perform extract, load, transformation and calculation operations.
- Assess the Design and use of Database method to perform create and load operations.
- Conduct visualization in an appropriate way.
- Structure code in appropriate methods (functions), looping and conditions.

The dataset ***titanic.csv*** is from the Kaggle competition (https://www.kaggle.com/competitions/titanic/data). The dataset contains a set of on-board passengers. Each row represents one passenger with the below information in Table 1:

Variable	Definition
Survived	Whether the passenger survived or not: 0 = No, 1 = Yes
Pclass	Ticket class: 1 = 1st, 2 = 2nd, 3 = 3rd A proxy for socio-economic status
Sex	Sex: male, female
Age	Age in years
SibSp	The number of siblings / spouses onboard
Parch	The number of parents / children onboard
Ticket	Ticket number
Fare	Passenger fare
Cabin	Cabin number
Embarked	Port of embarkation: C = Cherbourg, Q = Queenstown, S = Southampton
Name	Passenger name

Table 1: Data dictionary of titanic.csv

Question 1a

Load the dataset into a dataframe and identify fields contain missing data. Drop fields with more than 50% rows of missing values.

(3 marks)

Question 1b

Compute the average survival rate.

(2 marks)

Question 1c

Perform visualisation `Sex` and `Pclass` using 2 separate pie charts.

(4 marks)

Question 1d

Compute the number of passengers who travelled alone without siblings / spouses or children / parents.

(2 marks)

Question 1e

Design a histogram plot to visualise the data distribution of `Fare`. Share **ONE (1)** insight which you can draw from the visualisation.

(3 marks)

Question 1f

Design a boxplot to visualise `Fare` to identify outliners.

(2 marks)

Question 1g

Perform the required task in Q1(e) and Q1(f) for `Age`.

(2 marks)

Question 1h

Design and perform visualisation to identify if there is correlation exists between `Pclass` and `Survived`.

(3 marks)

Question 1i

Perform the required task in Q1(h) for `Sex` and `Survived`

(2 marks)

Question 1j

Create a new ordinal discrete field called `AgeBand` from the continuous `Age`. There can be 5 age bands: [0, 16], (16, 32], (32, 48], (48, 64] and (64, 80]. For example, a passenger with `Age` of 22 has the `AgeBand` of (16, 32].

(4 marks)

Question 1k

Perform the required task in Q1(h) for `AgeBand` and `Survived`.

(2 marks)

Question 1l

Create a new field called `Title` which is extracted from the `Name` field. For example, `Mr` is extracted from the name `Braund, Mr. Owen Harris`. The extraction rule is that the word ending with `.` within a name is the title of that name. Standardise uncommon `Title` values by following the mapping shown in Table 2. Compute and display the count per title.

From title	To title
Mlle, Ms	Miss
Rev, Major, Col, Don, Sir, Capt, Jonkheer, Lady, Mme, Countess	Uncommon

Table 2: Mapping of values to standardise uncommon `Title` values

(4 marks)

Question 1m

Create a new field `Alone` for passengers having no siblings / spouses and no parents / children: use **1** for alone passengers and **0** for passengers travelling with someone. Justify if there is any correlation between `Alone` and `Survived`.

(2 marks)

Question 1n

Create a new field `TravelWith` where `TravelWith` = `AibSp` + `Parch`. Justify if there is any correlation between `TravelWith` and `Survived`.

(2 marks)

Question 1o

Divide the `ln(Fare + 1)` into 5 even bands between its min and max values to create a new field called `LogFareBand`. Justify if there is any correlation between `LogFareBand` and `Survived`. Note that ln is natural log.

(3 marks)

Question 1p

Conduct data imputation to fill missing values for `Age` by the steps below:

(i) Design a visualisation to prove the correlation between `Age` and the pair of (`Pclass`,`Sex`).

(ii) Group by (`Pclass`, `Sex`) and compute the mean `Age` per group.

(iii) For each row with missing `Age` value, set its `Age` with the mean `Age` of its (`Pclass`,`Sex`).

(iv) Create a new field called `ImputedAge` and set the value to 1 for imputed `Age` and 0 for non-missing `Age`.

(v) Use imputed `Age` values to compute missing `AgeBand`.

(6 marks)

Question 1q

Fill missing `Embarked` values with the most common `Embarked` value.

(1 mark)

Question 1r

Drop the fields `Ticket`, `PassengerId`, `Name`, `Age`, `Fare`, `SibSp` and `Parch`

(1 mark)

Question 1s

Perform the following tasks:

Question 1s(i)

Run the below code snippet (in Figure 1) to train the Decision Tree model using the pre-processed dataframe.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

titanic = pd.get_dummies(titanic, columns = ['Sex', 'Pclass', 'Embarked',
'AgeBand', 'Title', 'LogFareBand'], drop_first=True)

Xtrain = titanic.drop('Survived', axis=1)
Ytrain = titanic['Survived']

decisionTree = DecisionTreeClassifier()
decisionTree.fit(Xtrain, Ytrain)
acc = round(decisionTree.score(Xtrain, Ytrain) * 100, 2)
print('*** Train accuracy:', acc)

fig = plt.figure(figsize=(25, 20), dpi=600)
tree.plot_tree(decisionTree, filled=True, feature_names=Xtrain.columns,
class_names=['Died', 'Survived'])

fig.savefig("titanic.jpeg")
```

Figure 1: Code snippet to train the Decision Tree model

(1 mark)

Question 1s(ii)
The code snippet saves the decision tree logic into `titanic.jpeg`. Share **ONE (1)** insight from the outputted decision tree logic.

(1 mark)

Question 2

- Manipulate dataset with data scientist mindset.
- Exposure to real-world dataset analysis.
- Design computation logic and routines in Python.
- Structure code in appropriate methods (functions), looping and conditions.
- Design methods to extract and parse information from the internet.
- Assess use of Pandas and Dataframes to perform extract, load, transformation and calculation operations.
- Conduct visualization in an appropriate way.

Question 2a

Apply BeautifulSoup to scrape all Pokémons from the site (https://pokemon.fandom.com/wiki/List_of_Pok%C3%A9mon). The result includes the following fields: `index`, `name`, `url` (to a detailed page), `type1` and `type2`. Compute and display the number of scrapped Pokémons.

(10 marks)

Question 2b

Store the scraped result from the Q2(a) into an SQLite database. Show some results which are directly queried from the database after storing all scraped Pokémons.

(5 marks)

Question 2c

Use SQLite to compose query to define a function which takes a string parameter to partially search by the Pokémon name field. Call the function with the parameter = 'gam' and show its results.

(3 marks)

Question 2d

Use SQLite to query and find the top 3 common pairs of (`type1`, `type2`) from the database.

(4 marks)

Question 2e

Use SQLite to query ALL Pokémons having the longest and shortest name.

(4 marks)

Question 2f

Use SQLite to count the number of Pokémons per type.

(4 marks)

Question 2g

From the dataset scraped in the Q2(a), for each Pokémon, follow the URL in the `url` field to navigate to its detailed page and further scrape the Pokémon name which it is evolved from. Develop a program by filling in the below code (shown in Figure 2).

```
import time
from concurrent.futures import ThreadPoolExecutor
from tqdm import tqdm

# Parameter: `url` is a URL to a detailed pokemon page, for example,
https://pokemon.fandom.com/wiki/Crabominable
# Return: the name of `evolves from` pokemon extracted from the page specified
by the url, for example, 'Crabrawler' for the example URL
def scrape_evolve_from(url):
    evo_from_name = None

    # *** FILL IN YOUR CODE HERE ***

    # *** END ***

    time.sleep(1)
    return evo_from_name

def run(f, my_iter):
    with ThreadPoolExecutor(max_workers=5) as executor:
        results = list(tqdm(executor.map(f, my_iter), total=len(my_iter)))
    return results

evo_from = run(scrape_evolve_from, pd.DataFrame(pokemons)['url'])
evo_from
```

Figure 2: Code to be filled for Q2(g)

(5 marks)

Question 2h

Create and store names scraped from Q2(g) into the new columns called `evolve_from` in the SQLite table created in the Q2(b). Analyse and display some Pokémons after the update of `evolve_from` values.

(5 marks)

Question 2i

Use SQLite to identify the number of Pokémon family trees there are. For example, Charmander, Charmeleon and Charizard are in the same SINGLE family tree because Charmander evolves into Charmeleon, and Charmeleon evolves into Charizard.

(5 marks)

Question 2j

Design a function which takes in a Pokémon name and returns all Pokémons being the same family tree as the provided Pokémon. Call the function with the parameter = 'Flareon' and display the results.

(5 marks)

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