DATA SCIENCE AND ARTIFICIAL INTELLIGENCE DIVISION

Quantitative Strategy Case Study Interview

Section 1
*Please answer **ALL** the questions in this section.*

Question 1 (Prediction)

Download the passing rate of annual vehicle inspection records from https://data.gov.sg/dataset/annual-motor-vehicle-inspection-passing-rate-of-motor-vehicles-on-first-inspection, and focus on inspections for motorcycles.

Task 1: What's the average passing rate on first inspection each year, taking into account motorcycles of all age groups?

Task 2: For motorcycles of each age, estimate their passing rate next year.

Task 3: Assuming your estimated rates are true, can you suggest a sensible range of possible passing ranges for motorcycles in the 5-year age group next year (2017), with at least 95% possibility of including the actual passing rate? If you can come up with multiple ranges that meet this criteria, use the one with the narrowest range. You may assume the number of motorcycles is the same as the number in the 4-year age group in the previous year.

~

Question 2 (Association)

Download the list of transactions by each property agent from https://data.gov.sg/dataset/cea-salesperson-residential-transaction-record, and focus on HDB resale flat transactions where a property agent represented the seller.

Task 1: Based on the dataset, how many sales would you expect an agent to close each year? How much variation is there among agents?

Task 2: Examine the distribution for number of sales closed by an agent in a year & suggest a probability distribution that may be suitable for modelling this set of values. What are some ways in which your suggested distribution is appropriate? What are some of its limitations?

Task 3: Property agents tend to specialise in one or more specific geographical areas, rather than ply their trade equally island-wide. Given a property agent who has closed sales in Sembawang and Yishun during a given year, which other town is he/she most likely to be active in that year?
(Note: you may wish to use association rules for this task.)

~

Question 3 (Classification)

Download the Wireless@SG hotspots file from https://data.gov.sg/dataset/wireless-hotspots (in either KML or geoJSON format), and extract the data associated with it. You should obtain a table with over 1600 rows and several columns, where each row corresponding to a different WiFi hotspot in Singapore.

Task 1: From the table, what are some of the information you can deduce for each hotspot?

Task 2: Due to a system error, the location type column for the last 200 rows of the dataset has become garbled. Using all earlier rows as well as all other columns in the dataset, build a classification model to predict the location type for these hotspots. You may treat the three rarest location types as one category.
(Note: you may wish to create some additional features based on available ones.)

Task 3: The information has now been recovered from a backup copy of the file. Compared to the true location types, how good was your model? Be prepared to explain the metrics you use to evaluate your model.

~

Question 4 (Data Visualisation)

A colleague is working with a salary dataset based on recent poly graduates in a specific course of study highly subsidised by the government, to compare whether the career choices made by students from Group X are different from those from Group Y in any manner. She has already produced the following summary table and listed out the main insight she wishes to highlight, as well as pertinent observations on the dataset's characteristics, but is struggling to come up with a good way to communicate the insight to her audience in one visualisation while also accurately reflecting the dataset's characteristics.

Summary table:

		Student Group X		Student Group Y	
Job Nature	Industry	Median Salary	Count	Median Salary	Count
Closely related to course of study	A	3150	83	3000	23
	B	3300	53	3100	9
	C	2650	47	2600	32
	D	2400	12	2400	15
Somewhat related to course of study	E	4100	30	3900	3
	F	3400	23	3150	7
	G	2800	12	2600	22
	H	2300	8	2200	11
Unrelated to course of study	Others	2900	21	1900	28

Main insight:

It may be worth reviewing the policy behind subsidising this course of study, as a considerable proportion of students from each group do not go on to work in industries closely related to it. For Group X, this may be partially due to higher / comparable salaries offered by other industries. For Group Y, non-salary factors may play a more prominent role.

Data characteristics:

1. There are considerably more students from Group X than Group Y in this course of study.
2. Proportionately more students from Group Y are in jobs unrelated to their course of study.
3. The distribution of students among various industries is considerably different between the two student groups.
4. Students from Group X tend to command higher salaries, for the same type of job & industry.
5. The salary differential between the two student groups differs by job nature and industry.

Task: Help your colleague present the insight in an intuitive manner that is easily understood by a non-technical audience, and that reflects as many characteristics in the list as possible. Be prepared to justify any and every aspect of your visualisation (e.g. chart choice, colour palette, labels, orientation, etc.).

End of Section 1

Section 2

*Please pick **ONE** of the hypothetical scenarios in this section, and prepare a deck to brief the relevant hypothetical senior management about your findings.*

Points to note in the hypothetical scenario:

1. *The deck will be used for a face-to-face briefing with management.*
2. *The deck will be circulated to the management for reading beforehand.*
3. *The briefing is expected to take 30 minutes, excluding Q&A.*
4. *Management includes a mix of directors from technical as well as non-technical backgrounds.*
5. *Management is not available for further clarification. If you find the scenario to be coached in overly broad terms, you may make reasonable assumptions to narrow things down, but be prepared to explain / justify them.*
6. *Management is interested to hear what can be done to further improve the analysis, given more time / resources / etc., if the problem described in the scenario surfaces again in the future.*

A list of suggested data sources is included below. You may use them (or any other publicly available data source) for your analysis.

Suggested data sources for Section 2

Data.gov.sg	
Resale flat prices	https://data.gov.sg/dataset/resale-flat-prices
NEA licensed eating establishments	https://data.gov.sg/dataset/list-of-nea-licensed-eating-establishments-with-grades-demerit-points-and-suspension-history
National map line	https://data.gov.sg/dataset/national-map-line
Singapore Master Plan 2019	https://data.gov.sg/dataset/master-plan-2019-region-boundary-no-sea https://data.gov.sg/dataset/master-plan-2019-planning-area-boundary-no-sea https://data.gov.sg/dataset/master-plan-2019-subzone-boundary-no-sea
LTA DataMall	
Bus stop locations / train station locations	https://www.mytransport.sg/content/mytransport/home/dataMall/static-data.html#Whole%20Island
Bus routes	https://www.mytransport.sg/content/mytransport/home/dataMall/dynamic-data.html#Public%20Transport
OneMap	
Geocode API	https://docs.onemap.sg/#onemap-rest-apis
Parliament	
Official reports – parliamentary debates (Hansard)	https://sprs.parl.gov.sg/search/home

Scenario 1

Some forum posters have complained that the value of their HDB flats suffer because they are near expressways, which are very noisy. Others say expressway proximity is good, due to the unblocked view (at least for higher floors).

The Housing and Development Board has tasked your team to analyse whether there is merit to either view, based on transaction prices for resale HDB flats in recent years.

~

Scenario 2

There is concern that the experience of the current public transport infrastructure varies significantly across HDB towns.

The Land Transport Authority has tasked your team to assess the current infrastructure, and review if there is merit to accelerate the construction of any specific phase of any MRT line that is due to open before 2030.

~

Scenario 3

The Singapore Food Agency is planning to conduct a 12-hour inspection operation on food establishments, using 5 teams of inspectors.

It has tasked your team to design an inspection schedule for each inspection team. Combined, the schedules should enable the inspectors to have a good geographical coverage of food establishments island-wide, with focus on those who have performed poorly in the past[1]. You may assume that each inspection takes 30 minutes on average, and that all food establishments are open for business during the inspection period.

(For the purpose of this question, you may limit your analysis to mainland food establishments that belong to one of the following two categories: 1) located within food centres, or 2) include postcodes in their addresses. I.e. excluding temporary canteens at construction sites, offshore hawker centre on Jurong Island, etc.)

~

Scenario 4

The National Archives of Singapore is preparing an exhibition about the evolution of parliamentary debate in Singapore, and wants to do a feature on Committee of Supply (COS) debates[2] in recent years.

It has tasked your team to summarise common topics from COS debates that took place within the last Parliament (13th), as well as identify MPs who asked questions frequently about each topic.

(Sample codes written in R / Python are provided at this end of this document to demonstrate the process of scraping text from the Parliament website. You may use them—with any necessary modification for your use case—or write your own code in your preferred language for the task.)

End of Section 2

---

[1] Based on inspection records from Jun 2015 to Sep 2016 available here: https://data.gov.sg/dataset/list-of-nea-licensed-eating-establishments-with-grades-demerit-points-and-suspension-history.

[2] A brief explanation on what the COS debates are can be found here: https://www.singaporebudget.gov.sg/budget_2019/about-budget/budget-features/how-singapore-s-finances-are-managed

Boiler plate code in R for Section 2 Scenario 4

```
if(!require(RSelenium)){
  install.packages(RSelenium)
  library(RSelenium)
}

rD <- rsDriver(browser = "chrome", check = FALSE)
# If you encounter error message with the above line requiring your chrome driver to be one or more specific
# versions, try the following:
# 1. rerun the line with check = TRUE once to download the latest chrome webdriver versions (or other
# browsers if you don't use chrome);
# 2. go to your browser & check its version (should be found under settings or similar);
# 3. enter binman::list_versions("chromedriver") (or other browsers) to confirm you now have the webdriver
# for the version checked in step 2;
# 4. rerun the line once again with check = FALSE & your preferred version specified
# Please note that you may need to quit & restart your R session for everything to work smoothly.

driver <- rD[["client"]]
driver$open()
driver$navigate("https://sprs.parl.gov.sg/search/home")
Sys.sleep(2)

# Get search box and fill it up
search <- driver$findElement("css", "#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(1) > input")
search$sendKeysToElement(list("enter search term here"))

# Uncomment the following lines to only search in titles
# checkbox <- driver$findElement("css", "#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(2) > label >
input")
# checkbox$clickElement()

# This will select the 13th parliament
session <- driver$findElement("css", "#divmpscreen2 > div.row > div:nth-child(1) > div > div.form-group.byParText >
select > option:nth-child(14)")
session$clickElement()

# Find submit element and click
submit <- driver$findElement("css", "#divmpscreen2 > div.row > div.col-sm-12.text-right.pull-right > div > button:nth-
child(2)")
submit$clickElement()

print("Search parameters submitted.")
suppressWarnings(rm(search, checkbox, session, exact, submit))

# Sleep momentarily for result to load
Sys.sleep(2)

# helper code for switching windows (from https://github.com/ropensci/RSelenium/issues/143)
myswitch <- function (remDr, windowId) {
  qpath <- sprintf("%s/session/%s/window", remDr$serverURL, remDr$sessionInfo[["id"]])
  remDr$queryRD(qpath, "POST", qdata = list(handle = windowId))
}

# Switch window and check for number of search results
myswitch(driver, driver$getWindowHandles()[[2]])
num_results <- driver$findElement("class", "showingResults")$getElementText()
res <- rev(strsplit(num_results[[1]], " ")[[1]])
num_pages <- ceiling(as.integer(res[1]) / as.integer(res[3]))

# Create empty list to store results, & tracker for current result under examination
res_list <- vector("list", as.integer(res[1]))
current_res <- 0
print(paste("There are", res[1], "results in", num_pages, "pages to click through."))
rm(num_results, res)

# Nested for loop to click through all search results
for(click in seq(1, num_pages)){
  print(paste("Search result page:", click, "of", num_pages))

  # This assumes that 20 search results are returned
  for(item in seq(1, 20)){
    # Switch to search results tab
    myswitch(driver, driver$getWindowHandles()[[2]])

    # stop if all results have been examined (this handles the exception on the last
    # page, which may have fewer than 20 search results)
    current_res <- current_res + 1
    if(current_res > length(res_list)) break

    # Get element to click on, to see each individual page with content
    elem <- driver$findElement("xpath",
                               paste0("//*[@id='searchResults']/table/tbody[",
                                      item,
                                      "]/tr[1]/td[2]/a"))
    elem$clickElement()

    # Sleep momentarily for result to load (if your results contain nothing but
    # html tags, increase sleep time)
    Sys.sleep(2)

    # Switch to page with content and get URL name
    myswitch(driver, driver$getWindowHandles()[[3]])
    item_key = driver$getCurrentUrl()[[1]]
    item_key = gsub("\\?", "_", rev(strsplit(item_key, "/")[[1]])[[1]])

    # Append result to list for later processing
    res_list[[current_res]] <- driver$getPageSource()[[1]]
    names(res_list[[current_res]]) <- item_key

    # Write out each page source as a file
```

```
    writeLines(capture.output(XML::htmlParse(res_list[[current_res]])),
               con = paste0(item_key, ".txt"))

    # Close tab
    driver$closeWindow()
  }

  # Switch back to search results tab
  myswitch(driver, driver$getWindowHandles()[[2]])

  # Once 20 results have been saved, stop if on last page, click on next page otherwise
  # (Next page button's relative location changes after first 20 results are shown,
  # hence the need for alternative xpaths)
  if(click == num_pages) break
  next_page <- if(click == 1){
    driver$findElement("xpath", "//*[@id='searchResults']/div[3]/section/ul/li[1]/a/em")
  } else {
    driver$findElement("xpath", "//*[@id='searchResults']/div[3]/section/ul/li[3]/a/em")
  }
  next_page$clickElement()

  # Sleep momentarily because next page takes a while to load
  Sys.sleep(2)
}

print(paste("Search completed.", num_pages, "pages clicked through for", current_res - 1, "results."))

driver$close()
rD[["server"]]$stop()
rm(driver, rD, click, current_res, elem, item, item_key, next_page, num_pages, myswitch)
```

Boiler plate code in Python for Section 2 Scenario 4

```
from bs4 import BeautifulSoup as bs # Note: Hint: suggest using bs for subsequent parsing of HTML source
from selenium import webdriver
import time

driver = webdriver.Chrome('input path to chromedriver if not added to PATH')

page_url = 'https://sprs.parl.gov.sg/search/home'
driver.get(page_url)

# Get search box and fill it up
search = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(1) >
input')
search.send_keys('COS')

# Uncomment following two lines to only search in titles
#checkbox = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(2) >
label > input')
#checkbox.click()

# This will select the 13th parliament
session = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div:nth-child(1) > div > div.form-
group.byParText > select > option:nth-child(14)')
session.click()

# Find submit element and click
submit = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div.col-sm-12.text-right.pull-right > div >
button:nth-child(2)')
submit.click()

print('Search parameters submitted.')

# Create empty dictionary to store results
res_dict = {}

# Switch window and check for number of search results
driver.switch_to.window(driver.window_handles[1])
num_results = driver.find_element_by_css_selector('#searchResults > div:nth-child(1) > div')
res = num_results.text.split(' ')
num_clicks = int(res[-1]) // int(res[-3]) + 1

print('There are {} pages to click through.'.format(num_clicks))

# Nested for loop to click through all search results
for click in range(num_clicks):

    # This assumes that 20 search results are returned, which are 1-indexed in the xpaths
     for item in range (1, 21):

        # Switch to search results page
        driver.switch_to.window(driver.window_handles[1])

        # Get element to click on, to see each individual page with content
        # Last page will have fewer than 20 elements, so need to handle this exception
        try:
            elem = driver.find_element_by_xpath('//*[@id="searchResults"]/table/tbody[{}]/tr[1]/td[2]/a'.format(item))
            elem.click()
        except:
            break

        # Switch to page with content and get URL name
        driver.switch_to.window(driver.window_handles[2])
        item_key = driver.current_url.split('/')[-1]
        item_key = item_key.replace('?', '_') # Replace ? because it would be an invalid filename

        # Append result to dictionary for later processing
        res_dict[item_key] = driver.page_source

        # Write out each page source as a file
        with open(item_key + '.txt', encoding = 'utf-8', mode = 'w+') as file:
            file.write(driver.page_source)

        # Close tab
        driver.close()

    # Switch back to search results tab
     driver.switch_to.window(driver.window_handles[1])

    # Click on next page once 20 results have been saved
    # Next page button changes after first 20 results are shown, hence need to enclose the xpath in a try block
    try:
        next_page = driver.find_element_by_xpath('//*[@id="searchResults"]/div[3]/section/ul/li[3]/a/em')
    except:
        next_page = driver.find_element_by_xpath('//*[@id="searchResults"]/div[3]/section/ul/li[1]/a/em')
     next_page.click()

    # Sleep momentarily because next page takes a while to load
    time.sleep(2)

# Check that all results are stored
assert len(res_dict.keys()) == int(res[-1]), "It looks like not all the results were stored!"
```