MS987 Optimization for Analytics

2023-24 Second Semester

Group Project

Due 28th March 2024 (submit before 5pm)

You are recruited for an analytics project by *Governytics*, a data consultancy company developing data-driven predictive analytics and decision making tools for a broad range of clients from local and national governments worldwide, as well as for partner companies. *Governytics* do not only provide specific analysis or solutions to a client's urgent needs, but they also carry in-house research to investigate opportunities that may give them the competitive advantage of first-entry into market. The company houses a number of technical experts, including data scientists, optimization experts and statisticians, and they also employ management consultants and a sales team for the interface with their clients.

Governytics have been getting heavily more involved in optimization-related projects than usual, and although they are in the process of recruiting more optimization experts in-house, they have a number of urgent projects with due dates approaching and hence they need your expertise in this area. The two specific projects they require your help are: 1) an in-house project to develop a predictive tool for 'air quality', and 2) a consultancy project for an external client to help them make various operational and strategic decisions. These two projects are described in detail in the following two parts, including background and expectations for each part.

PART 1: Using Air Quality Estimators for Prediction

(40% marks of the total project mark + 5% bonus marks)

With many environmental concerns including air quality and predicted rising temperatures, many cities and local governments working with *Governytics* are exploring possibilities for better predictive capabilities to improve their services. There are various branches of ongoing research in industry and academia trying to establish relationships between different characteristics of air, such as levels of key pollutants and naturally occurring gases, temperature, humidity and air pressure, and such research is expected to help many cities to manage their populations' wellbeing to increasing levels of quality.

Debasis Krishnan and Sarah Wu, two data scientists with optimization expertise from *Governytics*, are currently working on developing a generic predictive tool for air quality attributes, independent of ongoing research elsewhere. More specifically, they would like to build a tool that enables the user to input data that involves measurements of various attributes of interest with dates and time stamps (or attributes a city is capable of measuring, such as due to budget limitations), then specify which attribute is of particular interest for prediction, and then the tool should calculate the best possible function estimating the level of this attribute, given the levels of all or some of other attributes (they would like to give the user the control on choosing these attributes.)

The idea they have in mind for this tool is similar to multiple linear regression, albeit with some differences. More specifically, assume there is data for 11 attributes in total, with the levels of 10 independent attributes used to calculate the level of 11th (and dependent) attribute, as follows:

$$f(x) = \alpha x + \beta = \sum_{i=1}^{10} \alpha_i x_i + \beta = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \cdots + \alpha_{10} x_{10} + \beta$$

If the actual value of the 11th attribute is *y*, then in the classical multiple linear regression, one would simply calculate the error (or distance) of this value to the estimate, that is, $\varepsilon = |y - f(x)|$, and then attempt to minimize the sum of all such errors to identify the best fitting line. However, Debasis and Sarah are considering to define a *tolerance* μ, which would dictate whether the estimation by $f(x)$ is sufficiently good (and hence the error should be simply zero). For example, if the tolerance is set to 0.05 (or 5%), and if the function $f(x)$ estimates a value of 100 for a given data point, then they want the error to be zero if the actual value *y* is between 105 and 95 (and if *y* is outside of this range, for example, say, 109, then the error ε should be just 4, not 9). With this new error definition, they consider the classical approach of minimizing sum of all errors as a first step, but they also consider minimizing the maximum error (that is, the highest ε value among all data points). They believe this idea can be handled using linear programming and implemented in FICO Xpress Mosel, which they have access to, but they are limited with their time and need your help.

Debasis and Sarah found a real dataset[1] that contains measurements of various air quality measurements, as it can be seen in the data file called "AirQualityUCI.txt" (see "Data File for Part 1" on MyPlace). This file contains 9,358 data points for hourly averaged responses over a year from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device, which was located in a significantly polluted area in an Italian city. Note that in case a data value is missing, it is tagged with the value of -200. The data has the following 15 specific attributes (including date and time) for each measurement, in this given order:

1. Date (DD/MM/YYYY)
2. Time (HH.MM.SS)
3. True hourly averaged concentration CO in mg/m^3 (reference analyzer)
4. PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)
5. True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)
6. True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)
7. PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)
8. True hourly averaged NOx concentration in ppb (reference analyzer)
9. PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)
10. True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)
11. PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO2 targeted)
12. PT08.S5 (indium oxide) hourly averaged sensor response (nominally O3 targeted)
13. Temperature in °C
14. Relative Humidity (%)
15. AH Absolute Humidity

[1] Air Quality Dataset from: https://archive.ics.uci.edu/ml/datasets/Air+Quality
Also used in: S. De Vito, E. Massera, M. Piga, L. Martinotto, G. Di Francia, On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario, Sensors and Actuators B: Chemical, Volume 129, Issue 2, 22 February 2008, Pages 750-757, ISSN 0925-4005.

Debasis and Sarah would like you to model and implement their ideas using this dataset, after successfully reading and cleansing data (that is, not taking on board any data points that has a missing value for any of the attributes). If time permits, they want you also test your regression function on some of the remaining data (but this is optional). More specifically, they expect the following actions for this project:

a) You should read the data from the input file and cleanse it to obtain the **first 500 valid data points** (that is, data points with no missing values) using FICO Xpress Mosel. When doing so, ensure you record the measurements for 13 air quality related attributes for each data point (you do **not** need to keep track of date or time). While obtaining this data, also count how many data points were not valid and hence cleansed (**only** until you obtain the first 500 valid data points), and print this value out. Note that you may be cleansing data as you go, that is, you record data *only* after you confirm it does not have any missing values. ***(12 marks)***

b) Using the data you read, build an LP model as described above in FICO Xpress Mosel, where you first minimize sum of all errors and then minimize the maximum error, in order to find the best values for α and β in both cases. Your model should use all 10 attributes from attribute 3 to 12 inclusive (that is, measurements of gas levels) as independent attributes in order to predict the value of the dependent attribute (for which you can choose *either* temperature or relative humidity), where you set the tolerance μ = 0.05. Ensure your Mosel code is as generic as possible (for example, by setting any parameters in the declarations section, rather than hard coding them inside your code), and comment throughout your file where appropriate. ***(16 marks)***

c) Ensure your FICO Xpress model outputs into an output file the values of α and β in both cases, as well as the maximum error value in both cases (note that you need to *find* this value in the first case, that is, when you minimize sum of errors). In order to avoid overwriting an existing output file, you should use "F_OUTPUT+F_APPEND" instead of "F_OUTPUT" when opening/closing the file a second time. ***(4 marks)***

d) Write a memo to Sarah and Debasis (at most 2 pages with standard margins, 11pt Arial or Calibri, 1.5 line spacing), briefly explaining key aspects of your FICO Xpress model (such as sets you built, how you structured constraints correctly, etc.) and any key messages regarding model outputs. It is perfectly fine to refer to your FICO Xpress model or output file when writing your memo. (*Note:* A memo is in a way an 'informal letter' written to someone to give key messages; also remember Debasis and Sarah are technical experts so you can certainly use technical language if you want.) ***(8 marks)***

e) ***BONUS:*** If time permits, Sarah and Debasis would like you to test either of your regression functions (that is, when you minimize either sum of errors, or the maximum error) on a test set with **further 100 valid data points** from the input data. This will require you to read and cleanse data *beyond* the first 500 valid data points (hence you may do this right after your initial data reading of part a) but recording it separately), and then plug in the optimal values of α and β to calculate error values for the selected attribute and given tolerance μ. These error values should be also written into the output file (*preferably* only if they are not zero). Note that you will **not** resolve the LP for this part. Also write a single page memo addressed to Sarah and Debasis with your key findings (same format as in part d). ***(5 marks)***

Deliverables: Submit a **single** Mosel file (for a, b, c and e), **two** output files (for c and e) and a **single** Word/PDF file (for d and e). You can name the files as you wish, but they should include your group number and should be **not** longer than 12 characters. If you prefer, you can submit a single zip file for Part 1 containing all these files.

PART 2: Improving Operations and Strategic Decision Making for City Councils

(60% marks of the total project mark + 12% bonus marks)

Governytics is closely working with a number of city councils in Scotland and England, in order to develop analytical solution methods for a number of challenging decision-making problems stemming from daily operations. A team led by Shona MacDonald and Dmitri Zotov require your help by addressing some of these challenges, including enabling them to build some generic tools they will likely employ in upcoming projects from their broad range of clientele. While Shona and Dmitri have quantitative backgrounds (mathematics and computer engineering), their knowledge of optimization is limited, and hence, your guidance is also essential in that respect.

Shona and Dmitri have recently obtained a partial data file for a setting of a home care provider[2] in Scotland (file named "CareDistances.csv"), which details walking distances (in seconds) between 47 different locations, each location indicated with a UserID. Most city councils they are working with provide essential home care services to elderly, disabled or recovering/sick citizens in their city limits, which range from providing medication and preparing their meal to lifting them from their bed for some basic activities, and they believe that any solution approaches they may develop using this data can help them build customized solutions for their clients in the future. As you can observe, this data file has simply a matrix of distances, where each row and each column provide the information for the associated UserID's (and the value in the matrix simply indicates the walking distance in seconds.) Also note that some of the entries in the matrix are zero, which means there is no information on these distances (or UserID's of the associated row and column are the same) and hence you should assume that there is no connection in between.

Based on several recent or current projects they have been involved in, there are a number of particular scenarios of interest to Shona and Dmitri. First of all, they had a recent project, where a similar network was considered with a number of centres (or 'depots' in the network optimization language), that is, the locations where the home care staff are based at, and the rest of the locations in the network representing the locations of the homes of the citizens needing the home care services (or 'clients'). In such a scenario, a question of interest for the city council was to determine how many clients each centre should serve while keeping the total distance walked to a minimum (and hence enabling the council for subsequent decisions such as capacity planning.) Shona and Dmitri, based on their limited optimization knowledge, believe this can be simply modelled using a minimum cost network flow problem but certainly need your help building this.

In a current and more challenging project regarding daily operations, they faced the issue of deciding how to route each home care staff, where the duration of tasks also vary from one client to another (see the data file "CareDurations.txt" with task durations provided in minutes for the full data set involving 230 client locations, and see the data file "CareDistances-FULL.csv" for the full distance matrix of 236 locations.) Essentially, each staff member will arrive to one of the centres in the morning, and then start their shift of 8 hours work by walking to their first client, carry out the necessary task, then walk to their second client, carry out the necessary task, and so on, until they visit their last client before their 8 hour shift ends (it is allowed that a staff member may arrive to their last client just before their shift ends, even though carrying out this last task goes beyond the length of the shift.) After the task is carried out at the last client, the staff member is officially

[2] Data partly originates from: Dataset of Home Care Scheduling and Routing Problems with Synchronized Visits. University of Strathclyde. 2018. doi: 10.15129/2d4885e1-bc24-414b-83ce-a846fb5c9689, available at: https://doi.org/10.15129/2d4885e1-bc24-414b-83ce-a846fb5c9689

finished with their workday (their travel back to centre is not necessary, as they may head directly to their home.) Shona and Dmitri read about the famous travelling salesman problem (TSP) recently, and they realize that even that problem with a single salesman is very hard to solve, and they are dealing with a situation involving multiple 'salesmen' in this scenario, so they are thinking about developing some heuristic approach and they will share their ideas on this with you later in part c.

Shona and Dmitri expect you to carry out the following actions for this project:

a) Using **only** the file "CareDistances.csv", Shona and Dmitri consider a scenario with six locations (UserID's 71, 142, 280, 3451, 6846, and 7649) being 'centres' (or 'depots'), and the remaining 41 locations indicating where the 'clients' are located. For their first basic modelling approach, Shona and Dmitri assume that all clients need exactly the same services, that is, each of the 41 locations has a demand of one task to be carried. They also assume that the centres with UserID's 142 and 280 have limited capacities, and hence they cannot carry out more than 6 and 7 tasks, respectively. As noted earlier, Shona and Dmitri think they can build a balanced minimum cost network flow model using demands and supplies as well as distances provided, and hence they expect you to help them to build this model in FICO Xpress Mosel. Make sure your model reads all input files, and your network is balanced. Comment throughout your file where appropriate. (*Hint:* Your model may use dummy/artificial nodes and arcs; always think about what the net supplies are at each node. *Note:* You should read the .csv file in the same fashion as demonstrated in the class for reading data files such as .txt.) ***(22 marks)***

b) Ensure your FICO Xpress model from a) outputs into an output file of type .csv (comma-delimited), where the output should include the number of clients each of the six centres is serving (among any other key outputs you would like to output.) Also write a two-page memo to Shona and Dmitri (within standard margins, 11pt Arial or Calibri, 1.5 line spacing) including a brief explanation of your model and how you balanced it, and any key messages regarding model outputs, including whether you obtained integer flows and why. ***(10 marks)***

c) Shona and Dmitri are informed that there is no shortage of home care staff and no capacities in any of the centres, however, the fewer staff are employed for delivering home care services, the more flexibility the city council has for other services. Using the same six centres (depots) you already used in part a) along with the full distance matrix provided in "CareDistances-FULL.csv" as well as the task durations (in minutes) provided in the file "CareDurations.txt", they want to be able to generate valid routes for each staff member, starting at one of the six centres at the start of the workday and finishing at their last client (as noted earlier, they need to arrive to their last client before their 8 hour shift ends.) Since they think this problem will be very complicated to model as an integer programming problem, they have some ideas for building up a heuristic. One idea they have is as follows (see next page):

Step 0: Label all client nodes as "not served". Go to Step 1.
Step 1: If all client nodes are already labelled "served", then STOP. Otherwise, start at one of the centres (or 'depots') with a new staff member, initialize the work duration of this staff member as zero, and go to Step 2.
Step 2: From the current node, pick the nearest neighbour client node *j* that is "not served". If the current work duration and the walking distance to *j* add up to more than 8 hours, then this staff member has completed their shift, go to Step 1. Otherwise, go to Step 3.
Step 3: Add the walking distance to *j* as well as the task duration at *j* to the current work duration. Update the label of node *j* from "not served" to "served". If the updated work duration is more than 8 hours, then this staff member has completed their shift, go to Step 1. Otherwise, go to Step 2.

Shona and Dmitri realize that Step 1 is a bit tricky, since they do not want to just pick a random centre to start with. They think that a better approach to handle this step would be to evaluate all client nodes that are "not served" and their distances to the six centres, and then identify the smallest distance in these values (and hence pick the centre associated with this smallest distance.) They also think there might be other ideas you have in order to improve this heuristic algorithm (for example, in Step 2, what if there are client nodes that are "not served" yet but they are not a neighbour of the current node?) They would like you to implement this algorithm (with any improvements you might suggest) using Mosel, Python, MATLAB, Java or C++, and also run tests for comparison (in particular, picking a random centre in Step 1 vs. applying their idea of identifying the 'best centre', but you are welcome to test any other ideas you might have too.) Your code should be commented wherever necessary (also ensure to take care of different time units!) Your code should output at least the routes each staff member followed, and the total number of staff used to cover all clients (preferably, they would also like to see the number of staff originating from each centre, but this is not essential.) Also write a memo to Shona and Dmitri (up to three pages within standard margins, 11pt Arial or Calibri, 1.5 line spacing) to discuss any algorithmic details including any improvements you suggest and any key messages regarding model outputs. (*Note:* You will see a similar algorithm in week 9.) ***(28 marks)***

d) **BONUS:** Shona and Dmitri just discovered that there are also so-called improvement heuristics, which are very useful when you already have a valid solution for your problem. More specifically, they want to explore changing the direction of a tour if that can save time/cost. Their idea is as follows: say, we are given a valid route for a staff member, visiting clients in the order of 1-2-3-4. However, you realize that the sum of distances of the two arcs 1-2 and 3-4 is more than the sum of distances of the two arcs 1-3 and 2-4, and hence you can reroute as 1-3-2-4 (where you simply swapped the first two arcs with the latter two arcs, and reversed the direction of the flow on the arc 2-3.) The algorithm essentially compares pairs of existing arcs with their candidate replacement arcs (in the example, observe that the start nodes of two existing arcs become the first candidate arc, and the end nodes of two existing arcs become the second candidate arc, and the original flow between the existing arcs is reversed.)
Shona and Dmitri realize that implementing this idea is challenging (for example, how to pick pairs of existing arcs, both existing in the solution and candidate replacement arcs?), in particular in a generic format and applying it to the output of the previous algorithm (that is,

routes generated for each staff member in part c). They are happy if you pick these arcs in random fashion, but are also interested if you have any other ideas. They are also happy if you set an upper limit on the number of arcs chosen to avoid too long of a computation time. They expect you to implement this algorithm, using Mosel, Python, MATLAB, Java or C++. Make sure your algorithm can read input files as needed (including the output file of part c), and the output of the algorithm should include details on the updated routes. Comment throughout your code where appropriate. ***(12 marks)***

Deliverables: Submit **one** mosel file for a and b, **one** .csv output file for b, **one** Word/PDF file for b, **one** script file (Mosel, Python, Java, MATLAB or C++) and **one** Word/PDF file for c, and **one** script file (Mosel, Python, Java, MATLAB or C++) for d. Make sure when submitting script files other than Mosel, you submit **your source code** only and not e.g. executables (if I cannot read your source code, you get zero). You can name the files as you wish, but they should include your group number and should be **not** longer than 12 characters. If you prefer, you can submit a single zip file for Part 2 containing all these files.