The ministry of Higher education and scientific research

Control engineering department
3^{rd} year

ADC 3.0 Unilever Challenge Solution Proposition

Dauntless Team

School Year : 2023/2024

Contents

I. Introduction

This is our proposed solution for the challenge given by Unilever, a British multinational fast-moving consumer goods company that gave us a quite interesting optimisation problem to overcome. We will address this problem in two parts, the first one is by regrouping the data on the basis of costumers location and products used overtime during a precise day in the week. The organised data will then be used to perform the method that we will define next in this document.

II. Problem definition

The Problem as described in the paper presented by the company is to predict the optimal number of vendors and thus the trucks needed to do the most efficient distribution in the Algiers area. The paper also mentions that we could eventually reduce the dimensions of the problem by targeting specific geographical areas, thus making the challenge a prediction based one. Of course, the prediction term comes from the presence of historical data of sales in the Algiers area.

	MONTH	SALESMAN_CODE	channel	subchannel	CUSTOMER_CODE	ORDERNO	CITYNAMEE	AREANAMEE	TOT_CATEGORY_DESC	CATEGORY_DESC	BRAND_DESC	Chiffre d'affaire		KD	Active	Customer Address	Lattitude	Longitude	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Selling Type
85456	3.0	3501084.0	20.0	201.0	350110245.0	3,501,368,317.00	Baraki	Baraki	Home Care	OMO	OMO HS	2,308.35	ALGER/BOUMERDES	1.00		Cité amirat baraki	36.64	3.11	0.0	0.0	0.0	0.0	0.0	3.0	0.0	HPC
66421	2.0	3501084.0	20.0	201.0	350110245.0	3,501,364,276.00	Baraki	Baraki	Home Care	OMO	OMO HS	2,296.33	ALGER/BOUMERDES	1.00		Cité amirat baraki	36.64	3.11	0.0	0.0	0.0	0.0	0.0	3.0	0.0	HPC
85457	3.0	3501084.0	20.0	201.0	350110245.0	3,501,368,317.00	Baraki	Baraki	Home Care	SURF	SURF HS	1,808.47	ALGER/BOUMERDES	1.00		Cité amirat baraki	36.64	3.11	0.0	0.0	0.0	0.0	0.0	3.0	0.0	HPC
85455	3.0	3501084.0	20.0	201.0	350110245.0	3,501,368,317.00	Baraki	Baraki	Home Care	OMO	OMO HS	1,951.33	ALGER/BOUMERDES	1.00		Cité amirat baraki	36.64	3.11	0.0	0.0	0.0	0.0	0.0	3.0	0.0	HPC
66420	2.0	3501084.0	20.0	201.0	350110245.0	3,501,364,276.00	Baraki	Baraki	Home Care	OMO	OMO HS	2,331.33	ALGER/BOUMERDES	1.00		Cité amirat baraki	36.64	3.11	0.0	0.0	0.0	0.0	0.0	3.0	0.0	HPC

Figure 1: Dataframe output of the head function

The precedent figure shows some of the present features in the dataset provided by the company. Some features are apparent and distinctive from the others, the ones worth mentioning are indeed the codes for both the salesman and the customer, the order number of that precise delivery and the detailed region to where it's supposed to be headed. Details about the products are also present as well as other transaction details that are coded a numerical format from 1 to 7 as they represent occurrences that could happen on one day of the week i.e. for example, 0 on a Saturday means that there wouldn't be any delivery/transaction happening on that day, 3 on a Wednesday means that there will be a delivery for every Wednesday during the occurring month. Two other variables have the same coding process (channel and sub-channel), this will add even more complexity to the problem as the data representation isn't suited to direct usage for a prediction model training for example.

In fact, when we see the problem as a resource optimisation one, the problem is even more apparent. Thus, a preparation of the data must be done to assure the best data setup for the model that we're going to build further in this document.

III. Proposed Solution

After analysing the provided dataset and doing a meticulous Exploratory Data analysis, we propose the following solution:

As mentioned before we will start by some feature engineering, first of all through our EDA we noticed that for every single CUSTOMER_CODE There is a single address in the area name feature. This is very important for the next phase as it will allow us to build a monthly planning for every single day of the week for each costumer. It may sound confusing at first, yet this is one of the way to regroup all the data we have and build a solid data that gives us **the number of products** for every **single day in the month** for each mentioned **area** in its respective feature. When doing this we will have a history of the planning that was done in the sales/delivery history of the company over the period of three months (as it is apparent through the data). All of this will be done by this particular part of our code:

```
import pandas as pd
import numpy as np
Data = pd.read_csv('Dataset-Unilever.csv')
# Seeing if every single CUSTOMER_CODE has only one areaname
for i in range(Data[' CUSTOMER_CODE '].unique().shape[0]):
    if (Data[Data[' CUSTOMER_CODE '] ==
    Data[' CUSTOMER_CODE '].unique()[i]][" AREANAMEE "].unique().shape[0] !=1):
        print('The theory is wrong')

# We remove the Year and the district name as they have only a single value
df = Data.sort_values(by = ' CUSTOMER_CODE ')
df.drop(' DISTRICTNAMEE ',axis=1 ,inplace=True)
```

```
df.drop(' YEAR ',axis=1, inplace=True)
# Definning the function to be used to transform the values inside the Days features
def calculate_products(visit_plan_code):
    if visit_plan_code == 0:
        return [0, 0, 0, 0]  # No visit
    elif visit_plan_code == 1:
        return [1, 0, 1, 0]
    elif visit_plan_code == 2:
        return [0, 1, 0, 1]
    elif visit_plan_code == 3:
        return [1, 1, 1, 1]
    elif visit_plan_code == 4:
        return [1, 0, 0, 0]
    elif visit_plan_code == 5:
        return [0, 1, 0, 0]
    elif visit_plan_code == 6:
        return [0, 0, 1, 0]
    elif visit_plan_code == 7:
        return [0, 0, 0, 1]
    else:
        pass

weekdays = [' Sunday ', ' Monday ', ' Tuesday ', ' Wednesday ',
' Thursday ', ' Friday ', ' Saturday ']

for day in weekdays:
    df[day] = df[day].apply(calculate_products)
dft = df.copy()
# Converting the values through the following loop
data_final = pd.DataFrame()
for m in dft[' MONTH '].unique():
    # Preparing the dataset
    df = dft.copy()
    # dft.set_index(' CUSTOMER_CODE ',inplace=True)
    df = df[df[' MONTH ']==m]
    #df.drop(' MONTH ', axis=1, inplace=True)
    df.head()
    for customer_code in df[' CUSTOMER_CODE '].unique():
        df_new = df[df[' CUSTOMER_CODE ']==customer_code]
        for day in weekdays:
            product_per_single_day_and_customer = df_new[day].apply(np.array).sum()
            for id in df[df[' CUSTOMER_CODE ']==customer_code][day].index:
                df[df[' CUSTOMER_CODE ']==customer_code][day][id] = product_per_single_day_and_c
    data_final = pd.concat([data_final, df])
data_final.reset_index(drop=True, inplace=True)

filtered_df = df.loc[:,[" CUSTOMER_CODE "," AREANAMEE ", " Saturday ", " Sunday "," Monday "," T
filtered_df

df.sort_values(by=' AREANAMEE ')[' Longitude '].unique()
df = df.sort_values(by=' AREANAMEE ')
df.loc[df[' Lattitude '] == ' -   ',[' Lattitude ']] = ' 36.79 '
df.loc[df[' Longitude '] == ' -   ',[' Longitude ']] = ' 2.94 '
filtered = df.loc[:, [' MONTH ',' CUSTOMER_CODE ', ' AREANAMEE ',' Lattitude ', ' Longitude ',
filtered = filtered[filtered[' MONTH '] == str(1.0)] # For one month only
import json
columns_to = [' Saturday ', ' Sunday ', ' Monday ', ' Tuesday ', ' Wednesday ', ' Thursday ', '
for clm in columns_to:
    for i in range(len(filtered)):
        array = json.loads(filtered.loc[:, [clm]].values[i][0])
        filtered[str(clm) + ' - First Week'] = array[0]
        filtered[str(clm) + ' - Second Week'] = array[1]
        filtered[str(clm) + ' - Third Week'] = array[2]
        filtered[str(clm) + ' - Forth Week'] = array[3]

```

```
filtered = filtered.drop(columns_to, axis=1)
sorted_df = filtered.sort_values(by=' AREANAMEE ')
x = pd.DataFrame(sorted_df.drop_duplicates(subset=[' AREANAMEE '])[' AREANAMEE '].values, columr
y = sorted_df.drop_duplicates(subset=[' AREANAMEE ']).loc[:, sorted_df.columns[3:]].reset_index
sorted_clean = pd.concat([x, y], axis=1).reset_index(0)
columns_to_put = [' Saturday  - First Week', ' Saturday  - Second Week',
       ' Saturday  - Third Week', ' Saturday  - Forth Week',
       ' Sunday  - First Week', ' Sunday  - Second Week',
       ' Sunday  - Third Week', ' Sunday  - Forth Week',
       ' Monday  - First Week', ' Monday  - Second Week',
       ' Monday  - Third Week', ' Monday  - Forth Week',
       ' Tuesday  - First Week', ' Tuesday  - Second Week',
       ' Tuesday  - Third Week', ' Tuesday  - Forth Week',
       ' Wednesday  - First Week', ' Wednesday  - Second Week',
       ' Wednesday  - Third Week', ' Wednesday  - Forth Week',
       ' Thursday  - First Week', ' Thursday  - Second Week',
       ' Thursday  - Third Week', ' Thursday  - Forth Week',
       ' Friday  - First Week', ' Friday  - Second Week',
       ' Friday  - Third Week', ' Friday  - Forth Week']
for area in sorted_clean['Area Name'].unique():
    for column in columns_to_put:
        sorted_clean.loc[sorted_clean['Area Name'] == area,[column]] = np.sum(sorted_df.loc[sort
sorted_clean.drop(['level_0', 'index','index'], axis=1, inplace=True)
sorted_clean.to_csv('Unilever_cleaned.csv', index=False)
```

As we continue in this road, we will first take the sum of every single delivery done in a day of the week for a given costumer. The new dataframe will then be sorted out according to a single value of the CUSTOMER_CODE and it's corresponding place. We will then sum up the deliveries done in a single month while grouping them by their area name feature. This last task is highly critical for our next phase which solving the optimisation problem.

The final dataset that we obtained has all the days of the week timed four times to represent a single month of transactions i.e. Saturday_first, Saturday_second,etc. We will also have the corresponding coordinates available in the original dataset (longitude and latitude). These two are highly critical for the application of our benchmark metaheuristic genetic algorithm based p-median optimisation done by **mahmoudi2023optimisation**. This algorithm was introduced as a novel optimisation technique for IoT based systems and it's application will be now extended to planning for delivery systems. To our knowledge this might be a first that such technique could be used in this domain.

Without further a do we shall define the optimisation problem mathematically.

$$\min \sum_{1}^{N} x_{ij} d_j$$

as x_{ij} is the coordinate of the location that the delivery vendor for each and every single transaction that he must get to, d_j will be the distance between that coordinate point and our company base supposed to be at the reference for the current application.

$$8 \leq y_j \leq 15$$

as it is one of the constraints imposed on the delivery vendor and how much he will be taking from the deliveries. Particularly, it was imposed in the setup file delivered by Unilever.

$$\sum_{1}^{N} x_{ij} = 1$$

All these constraints could be defined in the chosen algorithm beside the distribution model. As the final output of our data has the following form in the next figure:

	Area Name	Lattitude	Longitude	Saturday - First Week	Saturday - Second Week	Saturday - Third Week	Saturday - Forth Week	Sunday - First Week	Sunday - Second Week	Sunday - Third Week	Sunday - Forth Week	Monday - First Week	Monday - Second Week	Monday - Third Week	Monday - Forth Week	Tuesday - First Week	Tuesday - Second Week	Tuesday - Third Week	Tuesday - Forth Week	Wednesday - First Week	Wednesday - Second Week
0	A?n Benian	36.79	2.92	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	A?n Taya	36.79	3.30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Alger centre	36.77	3.06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Bab El-Oued	36.77	3.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Bab Ezzouar	36.72	3.18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2: Partial Cleaned Dataframe output of the head function

we will be solving the optimisation problem over one day of data points collected (those being number of transactions per area per day in a month). And we will do the solving as it shows in the next figures:

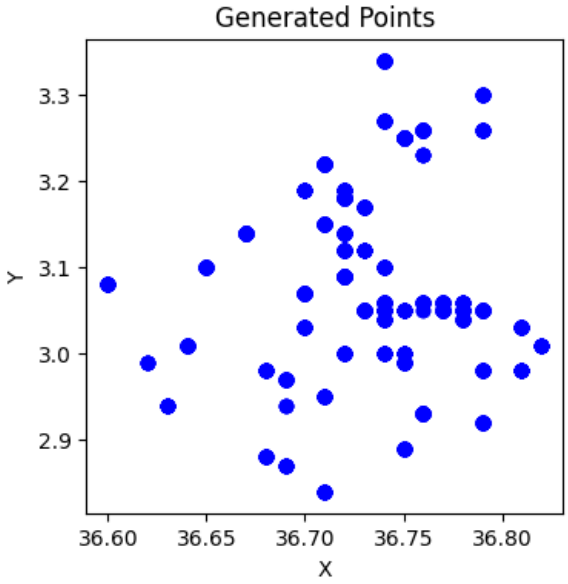

Figure 3: Data points before the GBPMO algorithm

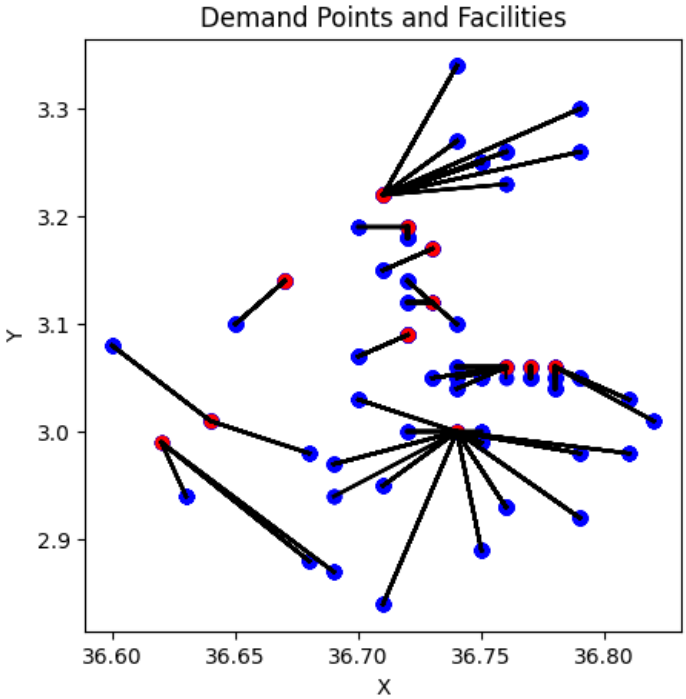

Figure 4: Data points after the GBPMO algorithm

Unfortunately we didn't have enough time to get a more accurate optimisation model as we had to reduce the number of the population size but also the number of iterations. Results could be more accurate if we had spent more time in tunning the parameters.

IV. Perspective about the work & Conclusion

As we lacked very much time in designing our solution, we would like to talk about the must-do thing for this model to give it even more efficiency.

First of all, to avoid using an optimizer and spend a lot of time every time you want to organise a planning for a given day, we can consider doing a Neural Network based model on the selected feature from early-on towards the output of our optimisation algorithm. This is a very hard task to do especially to tune the layer's parameters one by one. The most efficient and state of the art model that could potentially be used is the latest artificial net known as Kolmogorov-Arnold Network as they are well known for their high efficiency in the domain of physical modelling.

Another thing that could be potentially done is to make a predictor on the SALSESMAN_CODE based on the cleaned dataset that we made. We could add even more features to have like a preferred Customer for a given salesman. Using the areas after encoding them could potentially help in doing that. Everything is for sure unclear until we try out these methods.

In the end, we presented a novel optimisation method that was used in IoT systems as a efficient system for planning the best possible solution to do a certain delivery in a given area. The results were pretty satisfying as they show a great potential for future work on this project. In the hope that we will be actually making it a real thing by adding more things like a system that assigns the task to a certain SALESMAN as it still lacks due to time issues unfortunately.