

When Data Science is applied in Business Decision

Data Science is a recent umbrella term that includes machine learning (ML) and statistics, certain aspects of computer science including algorithms, data storage, and web application development.

This is my IBM Data Science Capstone Project for the last milestone of IBM program in acquiring ***Professional Certification in Data Science***. In this project/assignment we are asked to build a showcase where we will clearly define a problem and discuss the data that we will be using to solve the problem. The solution should leverage the use of ***Foursquare location data*** to explore, solve or execute the business case. Python codes for Business Case data Analysis & visualization are given in the attached Github link.

1. Introduction

Guaranteed business quality improvement, cost reduction, agility & flexibility to scale up/down the business, and managing customer experience are all key reasons for business owners to outsource some activities (managed service) to focus on their core business. Managed services can be, but not limited to:

- Business quality improvement (KPI – KQI – CEI)
- Preventative Maintenance programs
- Equipment installation
- 24 Hour Emergency Service & Call-Out Support
- Spare Part Management
- Work Force Management (WFM)

Nevertheless, within the restaurant business industry, preventive maintenance of equipment and systems are overlooked. Restaurateurs often choose to maintain their equipment on a reactive maintenance or breakdown model. A good approach to raise some concerns for those who adopt this strategy is to use capital budgeting technique and calculate the ROI, NPV, IRR, PBT...etc. over short and long term. From this perspective, we can evaluate reactive vs. proactive maintenance approach.



1.1.Business Problem

One of my clients has been in the managed services for **Restaurant Preventive Maintenance business** for almost 2 years. His first outsourcing service branch in “Cairo - El Agouza” neighborhood has shown a promising level of financial indicators in terms of ROI and NPV. From this perspective, he believes that establishing a second office branch for restaurant managed service is the way forward to grow the business. My client’s request was to support him in developing the new business plan document and to do the business case model (VBM), which I did. It is worth noting that while working on the business plan for the new branch I had to do **data analysis, visualization, and machine learning (ML) with Python** to answer some of the following questions:

- **First:** Which neighborhood to pick to open the new office for restaurant managed service? The office main goal is to serve venues (restaurants & cafés) in this neighborhood mainly. Selection should consider a number of criteria; the cost of property investment in the selected neighborhood must result in positive NPV, venues are surrounded with particular household income segment, and the neighborhood should be within a maximum geo-location distance of 12km from his first office in “El Agouza” where travelling time plays an important role between his first and second office.
- **Second:** Due to office resources limitation, how to screen or filter the potential venues list (restaurants, Café...etc.) in the neighborhood to identify target venues (customers)? In other words, what are the criteria to be used for screening?
- **Third:** For an efficient Inventory Management for Spare Part Management Service (SPMS) portfolio, how much set of items to order monthly (QTY) based on historical data of “Supply and Demand” pattern reported from his first office in “El Agouza”?
- **Fourth:** For an efficient Inventory Management, how to find out the actual number of each item to invest when there are certain constraints between two items? For example, for every 2 units of the Item (A) consumed, 1 unit of the Item (B) consumed, the total volume (A + B) must not exceed 102 units according to his historical data.

1.2.Target Audience

- An entrepreneur who wants to start his business in managed services. Business location is an element of 4Ps marketing mix, therefore which neighborhood is most likely to have restaurants (Fine Dining, Casual Dining, Family Style, Fast Casual) generating good profit and congested with people should be investigated.
- An Inventory Capacity Planner. Similar to Economic Order Quantity (EOQ) he wants to estimate the order quantity at one time with degree of certainty. With limited data for item order rate, what kind of probability distribution function (pdf) to use with certainty?

2. Input Data

2.1.Household Expenditure at Restaurants

There are many factors that will contribute to restaurant's success including food quality, service quality, marketing, competition, and location. Business location plays an important role in terms of the number of restaurants already exists and category of surrounding households average income.

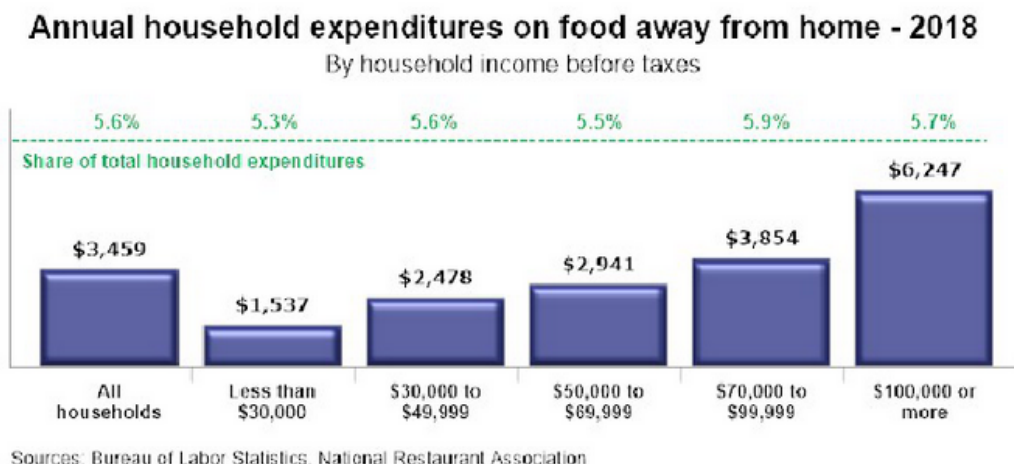
Expenditure Survey data from the Bureau of Labor Statistics in US shows that spending at restaurants and other foodservice outlets on meals, snacks and nonalcoholic beverages represented **5.6 percent** of the average household's total expenditures in 2018. The survey also indicated that spending at restaurants rises along with household owner income. This means we want to look for locations where the median **household income is close to or above certain level in order to address restaurants with successful business (ROI).**

```
import matplotlib.pyplot as plt
from PIL import Image # converting images into arrays

# download image
!wget --quiet https://restaurant.org/Restaurant/media/Research/Econ%20Notebook/Consumer-Expenditures-Income.jpg

# save mask to household_exp
household_exp = np.array(Image.open('Consumer-Expenditures-Income.jpg'))

fig = plt.figure()
fig.set_figwidth(14) # set width
fig.set_figheight(18) # set height
plt.imshow(household_exp)
plt.axis('off')
plt.show()
```



<https://restaurant.org/Articles/News/Households-in-the-West-spent-the-most>

2.2.Average Residential Property Prices

From “Global Property Guide” in Egypt, I will investigate the **Average Residential Property Price** per square meter in every neighborhood to achieve two goals:

- To comply with project accept/reject criteria for the developed business case by identifying the suitable property price/Sqm. For an office space area of 60 to 80 square meters the investment

cost in a condominium **should not exceed (EGP 8k/Sqm)** to get an approved capital budgeting measures (ROI, IRR, NPV, PBT...etc.)

- To address a particular household owner segment as there is a positive correlation between property value and annual household owner income. Moreover, a study showed that household spending at restaurants rises along with household owner income. Therefore to cope with the household spending considered in the business case, the Residential Property Average Price/Sqm **should be greater than (EGP 6k/Sqm)**.

To scrap the “Residential Property Prices” site I will use Request library and BeautifulSoup to fetch and parse the data. Then I will sort the parsed table content with price per square meter.

<https://www.globalpropertyguide.com/Middle-East/Egypt/Price-History>

2.3.Geographical Coordinates of Cairo Neighborhoods

One more important aspect for the new neighborhood selection is how far it is from the first office branch in “El Agouza”? Because the Service Level Agreement (SLA) has strict resolution time for Emergency & Call-out support it is important that the new office is relatively close to first office in case of support is needed in either branch. This will help to avoid penalty or contract breach. The geo-location distance from the first office should not be greater than 12km.

Working with “geopy.geocoder to import Nominatim” will provide the geo-location of neighborhoods (Latitude & Longitude). Then using “geopy.distance” will help to calculate the geodesic distance from the first office in "El Agouza" to each neighborhood in the list. The geodesic distance is not the road/route distance but it is the shortest distance on the surface of an ellipsoidal of the earth.

Scraping the web site of “neighborhood property price” will give us Cairo neighborhood names to get its geo-location

<https://www.globalpropertyguide.com/Middle-East/Egypt/Price-History>

2.4.Foursquare API for Venues Exploration

Once the target neighborhood for the second office meets the three selection criteria (Capital budget, the particular household owner segment, location distance), foursquare API will get the top 100 venues that are within a radius of 1,500 meters. Data is taken from JSON file under ['response']['groups'][0]['items'] to be structured into a pandas data frame.

```
url =  
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius=  
={}&limit={}'.format(
```

2.5.Municipal Divisions of Cairo Governorate with Population

Just to have the feeling of how condensed the selected neighborhood, a list of all the municipal divisions in Cairo city with an estimated population for each division from wikipedia. Scraping web table using “pandas.read_html” function will manifest all Cairo neighborhoods with population in Data-Frame.

https://en.wikipedia.org/wiki/Cairo_Governorate

2.6.Inventory Historical Data

This is last year monthly warehouse report “Supply & Demand” items. The report listed quantity of combined items only with no show to each item quantity separately (see the table below). This is due to the nature of equipment where some spare part items are not replaced alone or rarely are replaced separately.

	(2a + b)	(a + b)
Month		
1	70	60
2	85	80
3	85	80
4	90	80
5	85	65
6	90	82
7	85	78
8	90	85
9	95	87
10	85	77
11	85	80
12	120	90

When applying (solving with) linear programming technique (LP) on the historical data you can get the ***minimum quantity for each item*** to minimize inventory cost (objective) while meeting the demand of spare part items (constraints) according to reported data (explained in details in later sections). Since the spare part items’ quantity varied over the last 12 months with no clear sign which distribution it followed, Triangular probability distribution will be used to set the quantity for the above constraints with likelihood of occurrence (CDF) of not less than 80%.

Using python, the following is applied “from scipy.stats import triang” and “import pulp”

3. Exploratory Data Analysis (EDA) & Visualization

The objective in this section is to deep dive into data analysis to find out the proper neighborhood for the new office establishment, identify target venues (restaurants & cafés) from potential list to meet the resources’ capacity limitation, and to define the items’ order quantity for the warehouse for an efficient inventory management.

3.1.Office Location Address (neighborhood selection)

According to new ***Consumer Expenditure Survey data in US*** from the Bureau of Labor Statistics, Household expenditures at restaurants rose faster than most other spending categories during the last 5 years. Not surprisingly, spending at restaurants rises along with household owner income.

As mentioned earlier, the most suitable neighborhood to place the new managed service branch should meet 3 key success factors (KSF):

- The cost of investment in the selected neighborhood should result in positive NPV and an acceptable ROI. After developing the business case model, the average price in a condominium should not be greater than **EGP 8k/Sqm**.
- Address particular household income segment. After careful study of the correlation between average residential property price/Sqm and employee annual income, the neighborhood's property price/Sqm should not be less than **EGP 6k/Sqm**
- The office location should be within a maximum location distance of **12km from the first office** in "El Agouza" for emergency and call-out support.

Here is the final list of neighborhoods' latitude and longitude. Because python geocoder failed to find six neighborhoods locations due to name mismatch, I had to add them (df_no_loc) to the generated list by geocoder (neighbrs_Geo) with some code manipulation.

```
# Due to six names mismatch, searching for the Location and developing dataframe table for them
add_loc = [[30.026300, 31.496773], [30.090984, 31.322708], [29.992201, 31.317760], [30.096263, 31.303912],
           [30.229736, 31.479492], [30.013109, 31.208802]]
```

```
Latitudes = [add_loc[i][0] for i in range(0,6)]
Longitudes = [add_loc[i][1] for i in range(0,6)]
df_no_loc["Latitude"] = Latitudes
df_no_loc["Longitude"] = Longitudes
```

```
df_no_loc.rename(columns={0:'Neighborhood'}, inplace = True)
df_no_loc
```

	Neighborhood	Latitude	Longitude
0	New Cairo – Fifth Settlement, Cairo, Egypt	30.026300	31.496773
1	Heliopolis – Masr El Gedida, Cairo, Egypt	30.090984	31.322708
2	El Hadabah El Wosta, Cairo, Egypt	29.992201	31.317760
3	El Koba Gardens, Cairo, Egypt	30.096263	31.303912
4	El Oubour, Cairo, Egypt	30.229736	31.479492
5	El Jizah District, Cairo, Egypt	30.013109	31.208802

```
# Working with geocoder generated table, and combining it with the missing Location in new dataframe table
neighbrs_Geo = []
```

```
for i in range(0,len(neighbrs_loc)):
    neighbr_Geo = {"Neighborhood":neighbrs_loc[i][0], "Latitude":neighbrs_loc[i][1], "Longitude":neighbrs_loc[i][2]}
    neighbrs_Geo.append(neighbr_Geo)
```

```
neighbrs_Geo = pd.DataFrame(neighbrs_Geo)
neighbrs_Geo.set_index(['Neighborhood'], inplace = True)
```

```
#Removing neighborhoods with no Location, by dropping the row if it contains any the following neighborhood's name (no_loc)
neighbrs_Geo.drop(no_loc, inplace = True)
```

```
# Reset index
neighbrs_Geo.reset_index(inplace = True)
```

```
# We append rows of the six missing neighborhood Locations from one DataFrame table to another.
neighbrs_Geo = pd.concat([neighbrs_Geo, df_no_loc], axis=0)
neighbrs_Geo.reset_index(drop = True, inplace = True)
```

neighbors_Geo

	Neighborhood	Latitude	Longitude
0	New Administrative Capital, Cairo, Egypt	30.024871	31.776803
1	El Mohandeseen, Cairo, Egypt	30.076218	31.350072
2	El Sheikh Zayed City, Cairo, Egypt	30.048347	30.983224
3	El Agouza, Cairo, Egypt	30.054944	31.212728
4	El Maadi, Cairo, Egypt	29.960331	31.263055
5	Nasr City, Cairo, Egypt	30.052118	31.342205
6	Shoubra, Cairo, Egypt	30.078500	31.243305
7	El Abbasiya, Cairo, Egypt	30.066667	31.283333
8	Mokattam, Cairo, Egypt	30.033333	31.283333
9	New Heliopolis, Cairo, Egypt	30.118097	31.692560
10	6th of October, Cairo, Egypt	29.972346	30.940921
11	Downtown, Cairo, Egypt	30.088761	31.289820
12	15th of May, Cairo, Egypt	29.832042	31.364768
13	Hadayek El Ahram, Cairo, Egypt	29.970055	31.097298
14	Badr City, Cairo, Egypt	30.142806	31.742824
15	Helwan, Cairo, Egypt	29.850000	31.333333
16	El Zamalek, Cairo, Egypt	30.064972	31.219594
17	10th of Ramadan, Cairo, Egypt	30.290010	31.749342
18	El Haram, Cairo, Egypt	29.550000	31.133333
19	Ain Shams, Cairo, Egypt	30.130476	31.316749
20	Garden City, Cairo, Egypt	30.034722	31.231188
21	Manial, Cairo, Egypt	29.952956	31.239584

Calculating distances to every neighborhood


```

import geopy.distance

Dist = []

for i in range(0, len(neighbrs_Geo)):
    coords_ref = (neighbrs_Geo.iloc[3,1], neighbrs_Geo.iloc[3,2])
    coords_nei = (neighbrs_Geo.iloc[i,1], neighbrs_Geo.iloc[i,2])
    D = geopy.distance.distance(coords_ref, coords_nei).km
    Dist.append(D)

neighbrs_Geo['Distance'] = Dist
neighbrs_Geo

```

	Neighborhood	Latitude	Longitude	Distance
0	New Administrative Capital, Cairo, Egypt	30.024871	31.776803	54.505700
1	El Mohandeseen, Cairo, Egypt	30.076218	31.350072	13.451469
2	El Sheikh Zayed City, Cairo, Egypt	30.048347	30.983224	22.144643
3	El Agouza, Cairo, Egypt	30.054944	31.212728	0.000000
4	El Maadi, Cairo, Egypt	29.960331	31.263055	11.557422
5	Nasr City, Cairo, Egypt	30.052118	31.347205	12.489914

Adding two columns describing neighborhood's distance and property price/Sqm

Removing "Cairo, Egypt" from Neighborhood column, adding price/Sqm column, and sorting

```
# Removing "Cairo, Egypt" from Neighborhood column
neighbors_Geo['Neighborhood'] = neighbors_Geo['Neighborhood'].str.split(',').str[0]

# Combining two DataFrame tables, Table(neighbors_geoloc) & table (Neighbors_pr)
neighbors_geoloc = pd.merge(left=neighbors_Geo, right=Neighbors_pr, how='left', left_on='Neighborhood', right_on='Neighborhood')

# Sorting the table according to price/square meter.
neighbors_geoloc.sort_values(['Price_Sqm'], ascending=False, axis=0, inplace = True)

# Set "Neighborhood" index
neighbors_geoloc.set_index(['Neighborhood'], inplace = True)

neighbors_geoloc
```

	Latitude	Longitude	Distance	Price_Sqm
Neighborhood				
New Administrative Capital	30.024871	31.776803	54.505700	9,400
El Mohandeseen	30.076218	31.350072	13.451469	9,200
New Cairo – Fifth Settlement	30.026300	31.496773	27.578656	9,150
El Sheikh Zayed City	30.048347	30.983224	22.144643	8,850
El Agouza	30.054944	31.212728	0.000000	8,200
Heliopolis – Masr El Gedida	30.090984	31.322708	11.331458	6,800
El Maadi	29.960331	31.263055	11.557422	6,050
Nasr City	30.052118	31.342205	12.489914	5,850
Shoubra	30.078500	31.243305	3.938406	5,600
El Abbasiya	30.066667	31.283333	6.931173	5,550
Mokattam	30.033333	31.283333	7.218492	5,450
6th of October	29.972346	30.940921	27.774707	5,100
New Heliopolis	30.118097	31.692560	46.783742	5,100
El Hadabah El Wosta	29.992201	31.317760	12.289294	5,000
El Koba Gardens	30.096263	31.303912	9.912998	5,000
El Oubour	30.229736	31.479492	32.187832	4,400
Downtown	30.088761	31.289820	8.324758	4,200
15th of May	29.832042	31.364768	28.739847	3,600
Hadayek El Ahrām	29.970055	31.097298	14.579479	3,500
Badr City	30.142806	31.742824	52.016173	3,350
Helwan	29.850000	31.333333	25.527703	NaN
El Zamalek	30.064972	31.219594	1.293941	NaN
10th of Ramadan	30.290010	31.749342	57.883318	NaN
El Haram	29.550000	31.133333	56.496379	NaN
Ain Shams	30.130476	31.316749	13.063480	NaN
Garden City	30.034722	31.231188	2.862579	NaN
Manial	29.952956	31.239584	11.598762	NaN
Dokki	30.038895	31.212556	1.779050	NaN
El Jizah District	30.013109	31.208802	4.652916	NaN

From the table, meeting the 3 KSF will lead to two neighborhoods namely “Heliopolis” & “El Maadi”. I will choose “El Maadi” for my second office establishment.

3.2.My Customer List (customer screening)

Since “El Maadi” neighborhood is my target neighborhood, I will work with **Foursquare API** to get the top 100 venues that are in "El Maadi" within a radius of 1,500 meters. Data is taken from json file with “requests.get” under ['response']['groups'][0]['items'] to be structured into a pandas dataframe

```

LIMIT = 100 # Limit of number of venues returned by Foursquare API
radius = 1500 # define radius

url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
url

```

Clean the json, get the information under the *items* key, and structure it into a pandas dataframe.

```

# flatten JSON
nearby_venues = pd.json_normalize(results)

# filter columns for venue: name, categories, Lat, lng
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues = nearby_venues.loc[:, filtered_columns]
#nearby_venues = pd.DataFrame(nearby_venues)
nearby_venues.head()

```

	venue.name	venue.categories	venue.location.lat	venue.location.lng
0	El Qahwa Khan	[{'id': '4bf58dd8d48988d16d941735', 'name': 'C...	29.960822	31.264985
1	Gold's Gym	[{'id': '4bf58dd8d48988d175941735', 'name': 'G...	29.959940	31.259918
2	NOLA Cupcakes (نولا)	[{'id': '4bf58dd8d48988d1bc941735', 'name': 'C...	29.958510	31.259526
3	Villa Belle Epoque (Villa Belle Epoque فيلا ...	[{'id': '4bf58dd8d48988d1f8931735', 'name': 'B...	29.959463	31.263092
4	Maadi Club (نادي المعادي الرياضي)	[{'id': '52e81612bcbcb57f1066b7a2e', 'name': 'S...	29.964020	31.263031

```

# filter the column category for each row by extracting only the category "name" for each venue.
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns (by keeping only the last section name for each column name)
nearby_venues.columns = [col.split(".")[1] for col in nearby_venues.columns]

# Rename columns name
nearby_venues.rename(columns={'name': 'Venue', 'categories': 'Category', 'lat': 'Latitude', 'lng': 'Longitude'}, inplace = True)

print('\n', "{}" venues were returned by Foursquare as follows:'.format(nearby_venues.shape[0]), '\n')

nearby_venues.head()

```

"100" venues were returned by Foursquare as follows:

	Venue	Category	Latitude	Longitude
0	El Qahwa Khan	Café	29.960822	31.264985
1	Gold's Gym	Gym / Fitness Center	29.959940	31.259918
2	NOLA Cupcakes (نولا)	Cupcake Shop	29.958510	31.259526
3	Villa Belle Epoque (Villa Belle Epoque فيلا ...)	Bed & Breakfast	29.959463	31.263092
4	Maadi Club (نادي المادي الرياضي)	Sports Club	29.964020	31.263031

The number of unique categories and the number of each category

```
print('There are {} uniques categories.'.format(len(nearby_venues['Category'].unique())))
```

There are 49 uniques categories.

```
print(len(nearby_venues), '\n')
venues_count = pd.DataFrame(nearby_venues['Category'].value_counts())
venues_count = venues_count.reset_index().rename(columns={'index': 'venue', 'Category': 'count' })
venues_count.head()
```

100

	venue	count
0	Café	11
1	Restaurant	7
2	Italian Restaurant	7
3	Coffee Shop	6
4	Bookstore	4

For better illustration for the top venue category numbers, I am going to use wordcloud

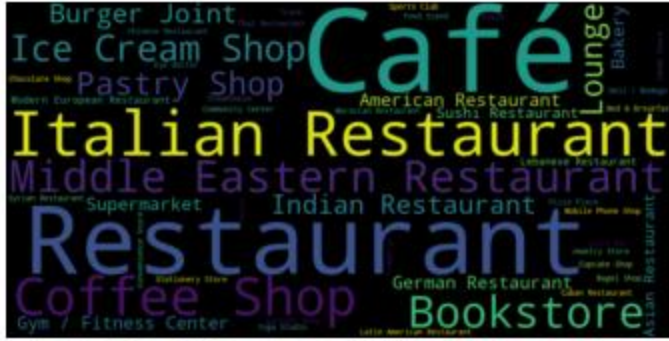
```
# Convert the word & count columns to a dict
venues_count = dict(zip(venues_count['venue'].tolist(), venues_count['count'].tolist()))
print(venues_count)
```

```
{'Café': 11, 'Restaurant': 7, 'Italian Restaurant': 7, 'Coffee Shop': 6, 'Bookstore': 4, 'Middle Eastern Restaurar
ger Joint': 2, 'Gym / Fitness Center': 2, 'Supermarket': 2, 'German Restaurant': 2, 'Bakery': 2, 'American Restaur
'Lebanese Restaurant': 2, 'Latin American Restaurant': 1, 'Gourmet Shop': 1, 'Sports Club': 1, 'Plaza': 1, 'Track'
za Place': 1, 'Moroccan Restaurant': 1, 'Eye Doctor': 1, 'Food Stand': 1, 'Bed & Breakfast': 1, 'Syrian Restaurant
': 1, 'Creperie': 1, 'Candy Store': 1, 'Convenience Store': 1, 'Cupcake Shop': 1, 'Chocolate Shop': 1, 'Jewelry St
```

```
#install and import wordcloud
!pip install wordcloud
from wordcloud import WordCloud
import imageio

venues_cloud = WordCloud(width=1200, height=600).generate_from_frequencies(venues_count)

# Visualize the word cloud
import matplotlib.pyplot as plt
plt.imshow(venues_cloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



For a startup business, my client has limited resource capacity that needs careful planning for the number of served venues/customers and the contracted SLA. After careful calculations the office can work on average of **3 sites a day and 22 working days a month**, this will lead to an average of 66 sites preventive maintenance support per month.

In order to build a customer loyalty, it is important that the agreed SLA not to be broken i.e. not breaching the SLA resolution time in case of emergency or call-out support while maintaining your restaurant preventive maintenance schedule as planned. To do so, it is wise to pick venues/customers that are close to each other as this will help to minimize the travelling time to and between venue sites.

Scikit library for machine learning algorithms is used to explore customers' data to find a pattern for grouping (clustering). Since traveling time (distance) is my main concern, ML algorithm will be applied on latitude and longitude. Let's start first plotting the table of venues to get a sense of what it is looks like. An important step before plotting is to do data pre-processing is to transform data (latitude & longitude) to have a mean of zero and a standard deviation of one and this can be achieved using python **"StandardScaler"**.

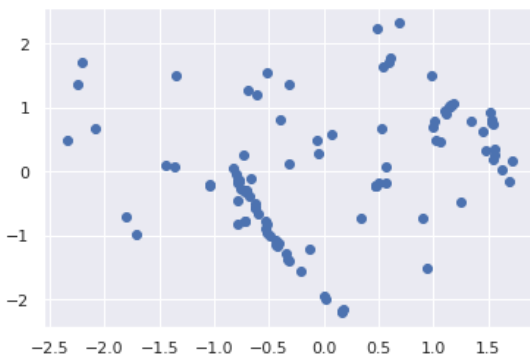
```
# Creating table for venue coordination of (Latitude & Longitude) and plot it to get a sense of what it looks like
coords = nearby_venues.values[:,2:] # Latitude in column(2) & Longitude in column(3)

# Change dtype from object to float
coords = coords.astype("float")

# normalizing the data
from sklearn.preprocessing import StandardScaler
coords = StandardScaler().fit_transform(coords)

#ax = nearby_venues.plot(kind='scatter', x='Longitude', y='Latitude', alpha=0.8, linewidth=0)
plt.scatter(coords[:,1], coords[:,0])
```

<matplotlib.collections.PathCollection at 0x7f951319bcf8>



Density-based clustering algorithms (DBSCAN) is more suitable for this case than other clustering algorithms e.g. k-means, hierarchical and fuzzy clustering for the following reasons:

- From the figure, venues geographical distribution has an arbitrary shape with no clear separation between possible clusters
- No idea for the right number of clusters, if k-means will be used for clustering
- The objective is to locate clusters of high density, and separates outliers
- DBSCAN is the most common clustering algorithms which works based on density of object

DBSCAN algorithm works based on two parameters: **Epsilon** (which is the radius) and **Minimum Points** to define a cluster. For my case the minimum neighborhood should not be less than 5 in a cluster, but what is the best epsilon? Meaning, what is the optimum radius that if includes enough number of points within, we call it dense area?

To find a suitable value for epsilon we calculate the distance to the nearest n points (venue) for each point, sorting and plotting the results. The optimal value for epsilon will be found at the point of maximum curvature.

```

neigh = NearestNeighbors(n_neighbors=2) # two means the point itself plus one nearest point
nbrs = neigh.fit(coords)
distances, indices = nbrs.kneighbors(coords)

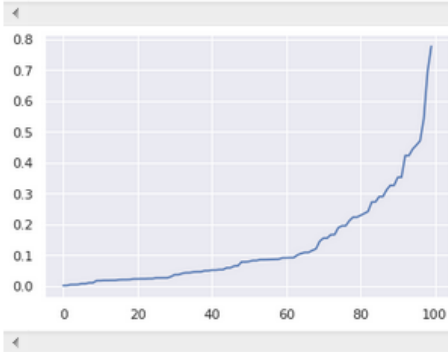
```

```

distances = np.sort(distances, axis=0)
distances = distances[:,1] # First column is the point itself, where the second column is the second nearest point
plt.plot(distances)

```

[<matplotlib.lines.Line2D at 0x7f12965fa5f8>]



The best epsilon is between (0.45-0.5)

```

epsilon = 0.45

# Compute DBSCAN
db = DBSCAN(eps=epsilon ,min_samples =5).fit(coords)
cluster_labels = db.labels_
unique_labels = set(cluster_labels)

# get the number of clusters
num_clusters = len(set(unique_labels))
print('Number of clusters: {}'.format(num_clusters))
print('Clusters label are: {}'.format(unique_labels))

```

Number of clusters: 4
Clusters label are: {0, 1, 2, -1}

```

nearby_venues["Clus_Db"] = cluster_labels
venue_cluster = nearby_venues[["Category", 'Clus_Db']]

# Counting each cluster number
venue_cluster = venue_cluster.groupby('Clus_Db').count()
venue_cluster.columns = ['venues_count']
venue_cluster

```

	venues_count
Clus_Db	
-1	25
0	47
1	6
2	22

We can recognize 3 clusters (0, 1, 2) and 1 outlier (-1). I need to select clusters that are suitable for my office service capability of **66 venues per month**. This means cluster 0 & 2 with total venues of (47+22 = 69). Excluding 5 shops from the list (3 bookstores, 1 mobile shop, 1 GYM) the remaining is 64 venues.

Visualizing the result and finding those venues on Folium map (red & cyan circles)

```
# Matplotlib and associated plotting modules
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors

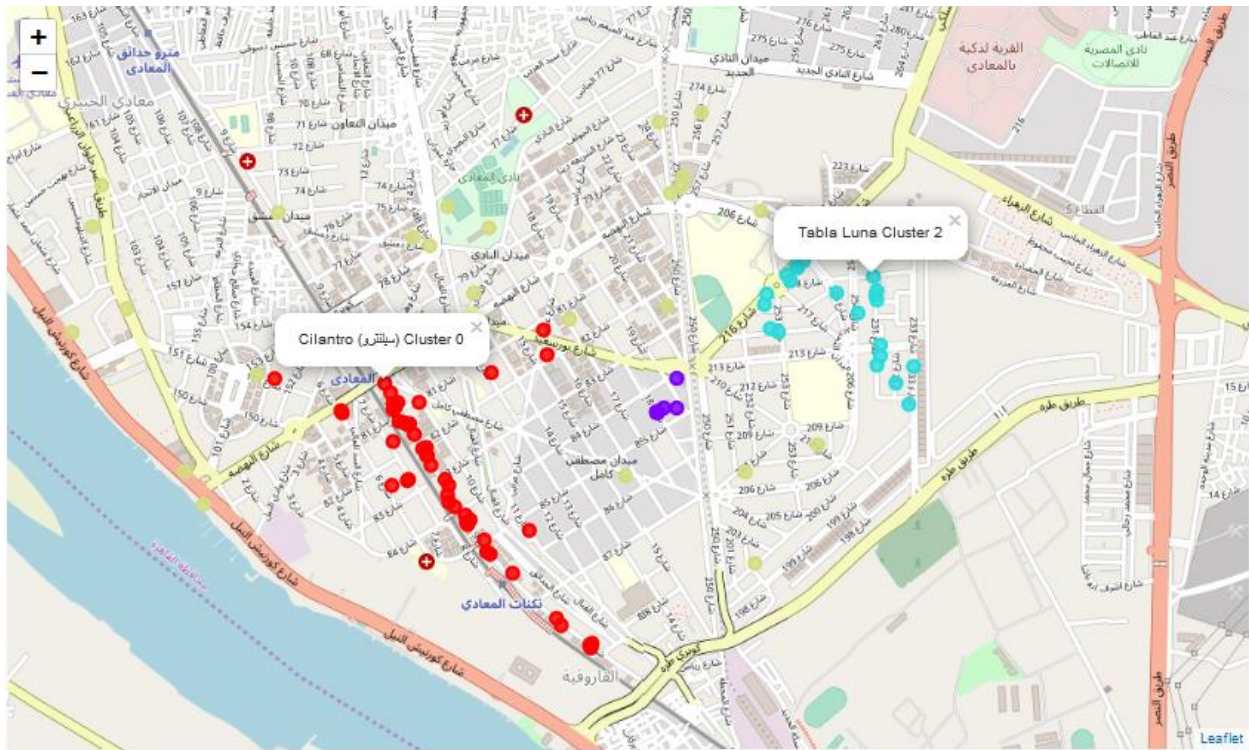
# create map
maadi_clusters = folium.Map(location=[Latitude, Longitude], zoom_start=14)
print(Latitude, Longitude)

# set color scheme for the clusters
x = np.arange(num_clusters)
ys = [i + x + (i*x)**2 for i in range(num_clusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, ven, cluster in zip(nearby_venues['Latitude'], nearby_venues['Longitude'], nearby_venues['Venue'], nearby_venues['Clus_Db']):

    label = folium.Popup(str(ven) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.8,
        parse_html=False).add_to(maadi_clusters)

maadi_clusters
```



My customer list from cluster 0 & 2

```
nearby_venues.loc[nearby_venues['Clus_Db'] == 0,:]
```

	Venue	Category	Latitude	Longitude	Clus_Db
0	Kazouza (كازوزة)	Middle Eastern Restaurant	29.950858	31.266704	0
1	Desoky & Soda	Restaurant	29.950922	31.266674	0
2	IL Pennello Ceramic Café	Café	29.950988	31.266752	0
3	Barista	Italian Restaurant	29.951569	31.265620	0
4	Ralph's German Bakery	Bakery	29.951771	31.265444	0
5	Wienerwald (وينروالد)	German Restaurant	29.953197	31.263859	0
7	Khayrat El Sham (خيرات الشام)	Syrian Restaurant	29.953793	31.263039	0
8	Union Lounge	American Restaurant	29.953875	31.262927	0
9	Man'oucheh (منقوشة)	Lebanese Restaurant	29.954246	31.262852	0
10	Ma7ali	Deli / Bodega	29.954512	31.264467	0
11	Brazilian Coffee Stores	Coffee Shop	29.954708	31.262242	0
12	Lucille's	American Restaurant	29.954776	31.262209	0
13	Villa 55 (فيللا ٥٥)	Restaurant	29.954871	31.262337	0
14	Shobak Habibi (شباك حبيبي)	Restaurant	29.955000	31.262175	0
15	The Bakery Shop (TBS) (ذا بيكيري شوب)	Bakery	29.955261	31.261787	0
17	The Backyard	Burger Joint	29.955416	31.261557	0
18	Caffé Greco (كافيه جريكو)	Café	29.955658	31.261502	0
19	Sugar Spell	Candy Store	29.955938	31.259510	0

20	Caracas Lebanese Cuisine & Cafe	Middle Eastern Restaurant	29.955949	31.261572	0
21	Cuba Cabana (كوبا كابانا)	Cuban Restaurant	29.956101	31.260071	0
22	Tanoureen (تنورين)	Lebanese Restaurant	29.956126	31.260106	0
23	Cold Stone Creamery	Ice Cream Shop	29.956136	31.261443	0
27	Mori Sushi	Sushi Restaurant	29.956573	31.260918	0
28	Koueider (قويدر)	Pastry Shop	29.956882	31.260778	0
29	Baskin-Robbins	Ice Cream Shop	29.956898	31.260760	0
30	Fat Cow	Burger Joint	29.957140	31.260723	0
32	Abou El Sid (ابو السيد)	Middle Eastern Restaurant	29.957307	31.259536	0
33	Auntie Anne's (أنتي آنز)	German Restaurant	29.957522	31.260322	0
34	Gusto	Restaurant	29.957817	31.260086	0
35	Mermaid (ميرمايد)	Italian Restaurant	29.957883	31.260123	0
36	Zöoba (زوبوا)	Middle Eastern Restaurant	29.957894	31.259910	0
37	Grapes Restaurant & Lounge	Restaurant	29.957948	31.259746	0
40	Mahraja	Indian Restaurant	29.958192	31.257675	0
42	Maharani (ماهاراني)	Indian Restaurant	29.958269	31.257651	0
44	Dragon House	Asian Restaurant	29.958318	31.259552	0
46	Papa John's	Pizza Place	29.958401	31.259616	0
48	NOLA Cupcakes (نولا)	Cupcake Shop	29.958510	31.259526	0
49	Alef Bookstore	Bookstore	29.958540	31.259690	0
50	Beano's Cafe (بينوس كافيه)	Café	29.958543	31.260475	0
51	Vodafone (فودافون)	Mobile Phone Shop	29.958837	31.259455	0
53	Cilantro (سيلنترو)	Coffee Shop	29.959139	31.259236	0
55	Bua Khao (بوا خاو)	Thai Restaurant	29.959284	31.255255	0
57	Villa Belle Epoque (Villa Belle Epoque فيلا ...	Bed & Breakfast	29.959463	31.263092	0
60	Gold's Gym	Gym / Fitness Center	29.959940	31.259918	0
62	BCA Maadi	Lounge	29.960039	31.265096	0
68	El Qahwa Khan	Café	29.960822	31.264985	0

```
nearby_venues.loc[nearby_venues['Clus_Db'] == 2, :]
```

	Venue	Category	Latitude	Longitude	Clus_Db
47	Wanted	Coffee Shop	29.958430	31.278135	2
52	Akasya	Modern European Restaurant	29.959109	31.277646	2
58	Micah's Grill & Jared's Bagels	Bagel Shop	29.959635	31.278293	2
59	La Rosa	Italian Restaurant	29.959662	31.276967	2
61	Sizzler	Steakhouse	29.959963	31.277094	2
63	Diwan Bookstore	Bookstore	29.960223	31.276470	2
64	Crave (كريف)	Restaurant	29.960285	31.277105	2
65	Seoudi (سعودي)	Supermarket	29.960688	31.273391	2
67	Dunes	Moroccan Restaurant	29.960807	31.273110	2
70	Al Kotob Khan (كتب خان)	Bookstore	29.961293	31.276283	2
73	Capricci	Italian Restaurant	29.961538	31.272894	2
74	Vittorio's (فيتوريوز)	Italian Restaurant	29.961713	31.276955	2
75	Sakura Sushi	Sushi Restaurant	29.961848	31.276900	2
76	Esta Bene	Café	29.961875	31.272955	2
77	Dishes	Café	29.961917	31.275503	2
79	Stavolta	Ice Cream Shop	29.962007	31.276953	2
80	Bakier Stationery	Stationery Store	29.962305	31.273833	2
81	Tabla Luna	Latin American Restaurant	29.962431	31.276850	2
82	Gourmet Egypt	Gourmet Shop	29.962465	31.273706	2
83	The White Owl	Restaurant	29.962710	31.274079	2
84	The Four Fat Ladies	Pastry Shop	29.962786	31.274033	2
85	Cilantro	Coffee Shop	29.962804	31.274105	2
86	Volume One	Bookstore	29.962934	31.274320	2

3.3.Inventory Management (items order QTY)

To forecast the quantity of items with certainty of not less than 80%, I will consider:

- The items list in the historical data for the last 12 months,

- Triangular CDF (the S-curve) for the combined items (2a+b) & (a+b),
- Applying linear programming (LP) to minimize the spare part cost (objective) with the specified constraints.

The result will give a minimum QTY for (2a + b) = 102 AND for (a + b) = 80

	(2a + b)	(a + b)
Month		
1	70	60
2	85	80
3	85	80
4	90	80
5	85	65
6	90	82
7	85	78
8	90	85
9	95	87
10	85	77
11	85	80
12	120	90

```
import pandas as pd
import numpy as np
from scipy.stats import triang
import matplotlib.pyplot as plt
```

Calculating the amount of combined spare part of items a&b using Triangular probability distribution

```
# Defining the required percentage for Likelihood of occurrence for number of spare parts a & b

# For the case of (2a + b)
m = 70      # minimum
n = 85      # mode
k = 120     # maximum
prob_ab = 0.8 # for minimum Likelihood of occurrence

x=np.linspace(m,k,(k-m)*2)
x=np.nan_to_num(x)
x=x.tolist()

c = (n-m)/(k-m)
loc = m
scale = k-m
y = triang.cdf(x, c, loc, scale)
y = y.tolist()
cdf = pd.DataFrame({'Item_ab':x, 'CDF':y})
cdf.head()

Item_ab = cdf.loc[cdf['CDF'] > prob_ab].iloc[0,0]
print('For {} likelihood of occurrence: '.format(prob_ab))
print('The quantity of combined items of (2a + b) >= {} item'.format(Item_ab))

For 0.8 likelihood of occurrence:
The quantity of combined items of (2a + b) >= 101.31313131313132 item
```

The combined items (2a+b) >= 102



Where the combined items (a+b)

```
# For the case of (a + b)
m = 60          # minimum
n = 70          # mode
k = 90          # maximum
prob_ab = 0.8   # for minimum likelihood of occurrence

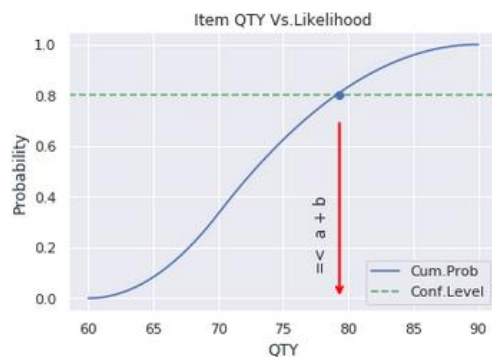
x=np.linspace(m,k,(k-m)*2)
x=np.nan_to_num(x)
x=x.tolist()

c = (n-m)/(k-m)
loc = m
scale = k-m
y = triang.cdf(x, c, loc, scale)
y = y.tolist()
cdf = pd.DataFrame({'Item_ab':x,'CDF':y})
cdf.head()

Item_ab = cdf.loc[cdf['CDF'] > prob_ab].iloc[0,0]
print('For {} likelihood of occurrence: '.format(prob_ab))
print('The quantity of combined items of (a + b) >= {} item'.format(Item_ab))

For 0.8 likelihood of occurrence:
The quantity of combined items of (a + b) >= 79.32203389830508 item
```

The combined items (a+b) >=80



In order to solve for the monthly order quantity to be in the office warehouse for the spare part management, I will use Linear Programming (LP) to get the minimum quantity for each item to minimize inventory cost (objective) while meeting the demand of spare part management (constraints) as follows:

- $3*a + 2*b$ (objective)
- $2*a + b \geq 102$ (constraint_1)
- $a + b \geq 80$ (constraint_2)

- $a \leq 41$ (constraint_3). What I know is that Item(a) should not be greater than 41.

Solving with python linear programming (pulp.LpVariable)

```
!pip install pulp

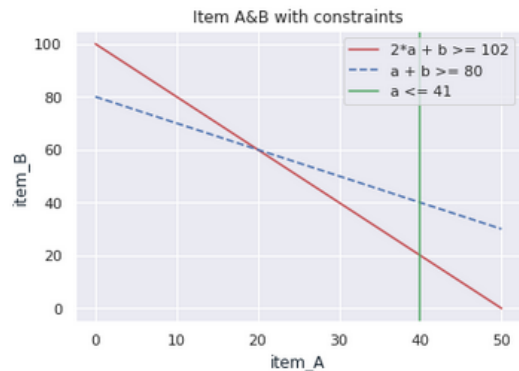
Requirement already satisfied: pulp in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (2.0)
Requirement already satisfied: pyparsing>=2.0.1 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from pulp) (2.4.6)

import pulp
a = pulp.LpVariable('a', lowBound=0)
b = pulp.LpVariable('b', lowBound=0)

problem = pulp.LpProblem('A simple minimization objective',pulp.LpMinimize)
problem += 3*a + 2*b, 'The objective function'
problem += 2*a + b >= 102, '1st constraint'
problem += a + b >= 80, '2nd constraint'
problem += a <= 41, '3rd constraint'
problem.solve()

print("Minimization_Results:")
for variable in problem.variables():
    print(variable.name,'=', variable.varValue)

Minimization_Results:
a = 22.0
b = 58.0
```



(QTY Item a) >= 22 AND (QTY Item b) >= 58

Same analogy is performed on item c & d. Using Triangular distribution leads to:

$(c + 3d) = 250$ AND for $(c + 2d) = 180$.

Then with linear programming for minimum spare part cost (objective) with the specified constraints we get:


```

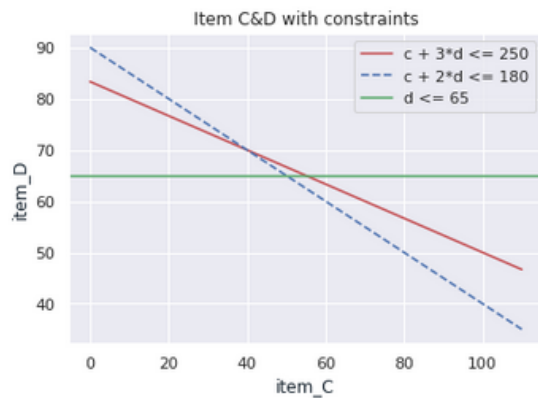
import pulp
c = pulp.LpVariable('c', lowBound=0)
d = pulp.LpVariable('d', lowBound=0)

problem = pulp.LpProblem('A simple minimization objective',pulp.LpMinimize)
problem += 2*c + 5*d, 'The objective function'
problem += c + 3*d >= 250, '1st constraint'
problem += c + 2*d >= 180, '2nd constraint'
problem += d <= 65, '3rd constraint'
problem.solve()

print("Minimization Results:")
for variable in problem.variables():
    print(variable.name, '=', variable.varValue)

```

Minimization Results:
c = 55.0
d = 65.0



(QTY Item c) >= 55 AND (QTY Item d) >= 65

4. Results:

This section provides answers to the questions raised in the business problem part.

4.1.Office location:

El Maadi is the most suitable neighborhood. It addresses the three main critical success factors (CSF); the capital budget, venues surrounded with particular household income segment, and 12km maximum location distance.

4.2.Venue/Customer List (screening)

Based on the machine learning algorithm recommendation for density-based clustering (DBSCAN), venues in cluster 0 and cluster 2 will be my target customer list. Both clusters will provide the suitable list to meet my limited capacity to perform preventive maintenance for 66 venues.

4.3.Combined Items Order Quantity with Confidence Level

Using historical data, statistics, and applying Triangular Probability Distribution to forecast with confidence level not less than 80% we get the following forecasted combined quantity:

- $(2a + b) \geq 102$
- $(a + b) \geq 80$
- $(c + 3d) \geq 250$
- $(c + 2d) \geq 180$
- Where $a \leq 41$ & $d \leq 65$

4.4. Applying Linear Programming (LP)

To resolve for minimum cost to get minimum quantity of items (a, b, c, d) when there are certain constraints, we get:

- Item a = 22
- Item b = 58
- Item c = 55
- Item d = 65

5. Discussion

I would like to shed some light on a number of topics I experienced while working on the project code:

- **Python Try Except:** When working with python geocoder (geopy.geocoder to import Nominatim) to get location, sometimes it fails to provide the latitude and longitude due to name mismatch. To have an instant idea if you have a missing list of geo-location and in order to work on this list later, it is recommended to use python Try Except. With try block it will generate an exception because the neighborhood name does not exist, then with exception block it will work with the error you have defined in the code. In my example, the exception block will create a data frame list for neighborhoods with no location called "no_loc".

```
try:
    location = geolocator.geocode(address)
    lat = location.latitude
    lng = location.longitude
```

```
except:
    no_loc.append(address)
```

- **Geodesic Distance:** Working with geopy.distance.distance will give the shortest line between two points on the earth's surface. This should not be understood as the route distance between the two points as in google map.

- **WordClouds in Python:** Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency. If you have pandas data frame table, you need to convert word & count columns to a dict first before applying “generate_from_frequencies”
- **Why Triangular not Normal Distribution Estimation:** Distribution models are essential in inventory planning as they help to find how many items of each item should be in stock. To estimate uncertainty in demand planning, we use distribution models. Researchers opt to use triangular distribution when the following criteria are met:
 - The lack of a known distribution.
 - Lack of time and money to work on a more elaborate statistical analysis.
 - The upper, lower and the most common outcomes are the only known variables.

I used Triangular cumulative function in quantity forecast because I have the lower, upper, and most common quantity for the combined items **(2a+b) & (a+b)** over the last 12 months. I also do not have clear idea what the distribution should look like. Beside, triangular distribution is widely used in management tools like the Project Evaluation and Review Technique (PERT) and Critical Path Method (CPM) to estimate project completion times based on maximum and minimum values.

Now, let's assume we know it is a normal distribution and the only information we have from the historical data is the **min =70 and max =120**. The mean will be (95). Applying the “**68-95-99.7 rule**” we get a standard deviation **of (120 – 70)/6 = 8.3**. This is because for a normal distribution, **99.73 %** of all samples will fall within 3 Standard Deviations of the mean value. The outcome for a minimum QTY with a confidence level of not less than **80% is (102) item**. This is similar to what we got with triangular distribution. If I assume that the min and max values fall within 2 Standard Deviations of the mean, then the calculated standard deviation will be **(120 – 70)/4 = 12.5**, and the minimum warehouse QTY with a confidence level of not less than **80% is (106) item**.

6. Conclusion

In general not always great idea can be transferred into a successful business. Great idea has to go first through test viability. Then we need to build a business plan to increase the likelihood of success. An important three interrelated components in business plan are:

- Market/customer analysis: I have identified the neighborhood with venues surrounded with particular Household owner income segment. This is based on an existing positive correlation between household income & residential property price.
- Service analysis: For example, the service team capacity and capability for the number of restaurants to work on per day, the SLA resolution time...etc.
- Financial analysis: Because the objective from management team in any profitable organization is to increase the shareholders' wealth, the investment decision to accept or reject a project is based on economic evaluation called capital budgeting techniques. The payback period (PB), internal rate of return (IRR) and net present value (NPV) methods are the most common approaches to project selection. In my case, the investment cost threshold for the average price in a condominium to produce an IRR greater than company's cost of capital and positive NPV should not be greater than EGP 8k/Sqm.

Finally, I capitalized on machine learning (ML), linear programming (LP), and probability distributions for an efficient management decisions

Keywords:

Data Science, Machine Learning (ML), DBSCAN, Statistics, Linear Programming (LP), Python, Management Decision, Managed Services (MS), Restaurant field maintenance, Business Case, Foursquare API