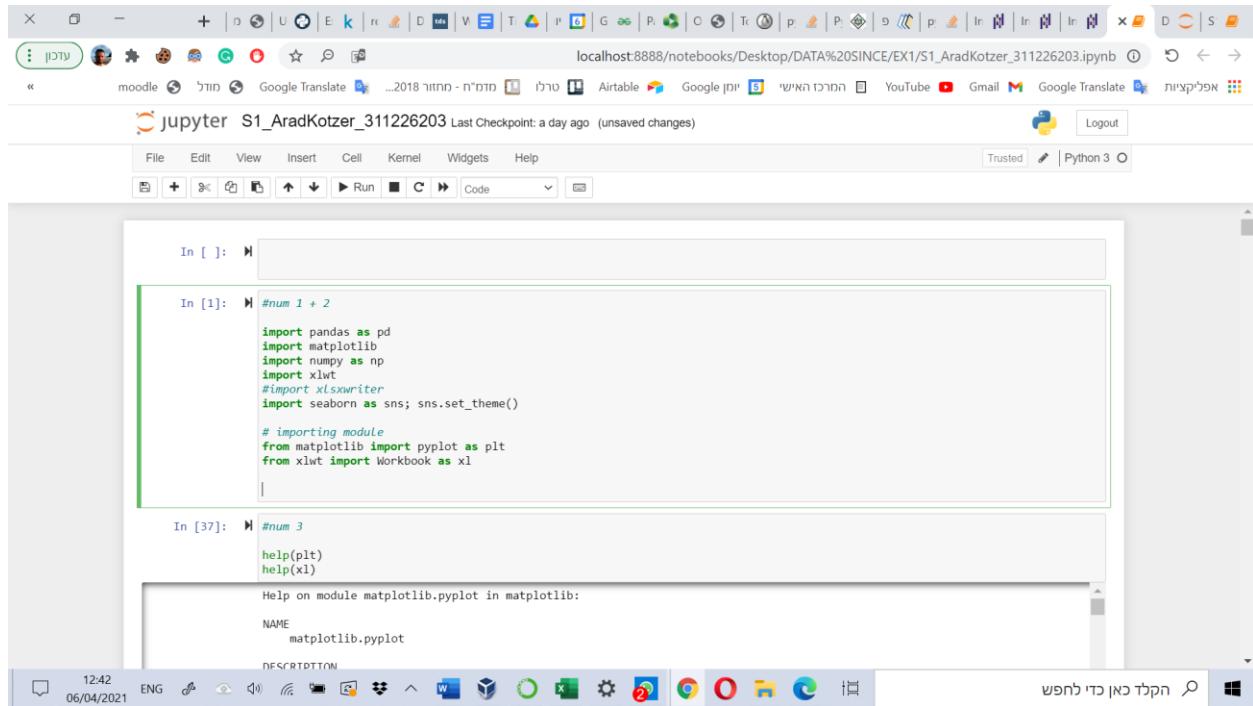


מגיש : ערד קוטר
ת"ז: 311226203
מסלול: מדמ"ח יזמות
 שאלה 1
 $2 +$
יבוא חבילות



The screenshot shows a Jupyter Notebook window titled "jupyter S1_AradKotzer_311226203". The notebook has two cells:

- In [1]:** Contains Python code for importing pandas, matplotlib, numpy, xlwt, and seaborn, followed by a blank line.
- In [37]:** Contains Python code for importing plt and xl from matplotlib, followed by a call to help(plt) and help(xl). The resulting help documentation for matplotlib.pyplot is displayed in the cell output.

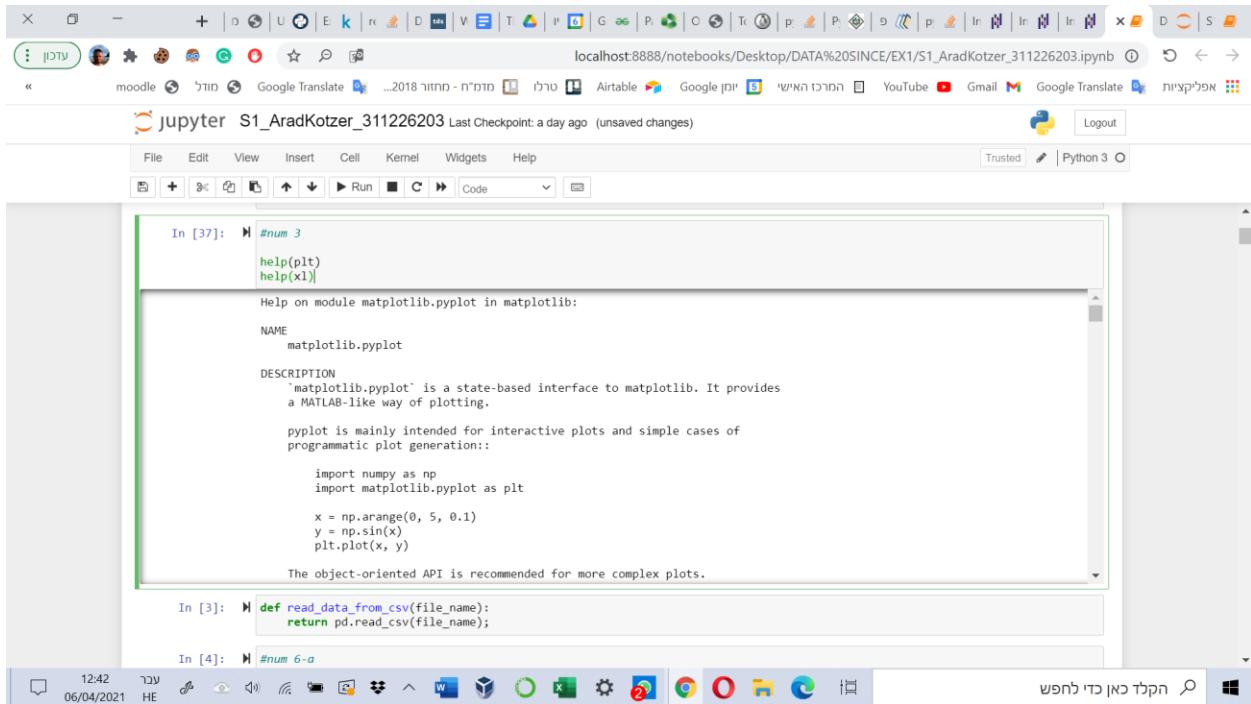
The help documentation for matplotlib.pyplot includes:

- NAME**: matplotlib.pyplot
- DESCRIPTION**: Help on module matplotlib.pyplot in matplotlib:

The operating system taskbar at the bottom shows various open applications, including Microsoft Word, Excel, and Google Chrome.

שאלה 3

הדפיסו למסך את כל הfonקציות והמשתנים של כל אחד משי המודולים אותם ייבאתם.



שאלה 4

וואקס - היא ספרייה לכתיבה נתונים ועיצוב מידע לקבצי Excel ישנים יותר (כולומר: .dax).
זהו ספרייה שפותחים יכולים להשתמש בה כדי ליצור קבצי גליאון אלקטרוני התואם לארסאות Microsoft Excel עד 2003. הchèבילה היא רק פיתון ללא תלות במודולים אחרים.

וואקס Workbook זה מודל המציג אפשרות עבודה פונקציות בסיסיות הקשורות לעבודה עם וואקס אבל הי חסרות לה. בעת יצירת קבצי Excel עם וואקס, בדרך כלל נתחיל בהפעלת Workbook כדי לעבוד. למרות שchèבילה זאת כפי שרשמתן יחסית מושנת.

Seaborn היא ספרייה לייצור גרפיקה סטטיסטית בפייטון. זה בניי על גבי matplotlib ומשתלב מקרוב עם מבני נתונים של pandas.

Seaborn עוזר לך לחזור ולהבין את הנתונים שלך. פונקציות ה plotting שלה פועלות על מסגרות נתונים ומערכות המכילים מערכי נתונים שלמים ומבצעות את המיפוי הסמנטי הדרוש כדי ליצור תמונה סטטיסטית שתעזר לך להבין יותר טוב את הנתונים.

NumPy היא חביבת עבודה מערכים למטרות כלליות. היא מספקת אובייקט מערך רב מיידי בעל ביצועים גבוהים וכליים לעבודה עם מערכים אלה.

היא הchèבילה הבסיסית למחשב מדעי עם Python. היא מכילה תוכנות שונות כגון: עבודה עם אובייקטים על מערך N מיידי בצורה טובה, עבודה עם שלל סוגים פונקציות, כלים לשילוב C / C ++ ועוד Fortran ויכולות עבודה על אלגברה לינארית שימושית, כמו טרנספורמציה פוריה, מספרים אקראיים ועוד מלבד השימושים המדעיים הברורים שלו, NumPy יכול לשמש גם לעבודה על נתונים גנריים רב-מדדים בצורה יעילה.

Matplotlib היא ספרייה מקיפה לייצרת ויזואלייזציות סטטיות, מונפשות או אינטראקטיביות בפייתון. היא ספריית plotting לשפת התכנות של Python ולהרחבת מתמטיקה המספקת שפה Cython. הוא מספק ממשק API מונחה עצמים להטמעת מגרשים ביישומים באמצעות ערכות כלים ממשק משתמש כללי כמו Tkinter, Qt wxPython או GTK. יש גם ממשק "pylab" פרוצדורלי המבוסס על מכונת מדינה (כמו OpenGL), שנודעה להידמות באופן הדוק לזה של MATLAB, אם כי השימוש בו אינו מיושם.

מודול matplotlib.pyplot הוא אוסף של פונקציות שגורמות לו-plotting matplotlib לעבד קצת כמו MATLAB. כל פונקציה pyplot גורמת לשינויו כלשהו ונוננת עוד אופציות מיוחדות לubahה על נתונים: למשל, יוצרת תרחיש, יוצרת אזור plotting בתרחיש, מאפשרת הוספה של כמה קוים באזורי plotting וכו' .

pandas היא ספרייה תוכנה שנכתבה עבור שפת התכנות Python לצורכי מניפולציה וניתוח נתונים. בפרט, הוא מציע מבני נתונים ופעולות לתמരון טבלאות מספריות וסדרות זמן. זהה תוכנה חופשית המשוחררת תחת רישיון BSD בן שלושה סעיפים. [2] השם נגזר מהמונה "panel data", מונח אקונומטרי עבור מערכי נתונים הכוללים ציפויות לאורך פרקי זמן מרובים עבור אוטם אנשים. נחשבת לאחת מחבילות העבודה על נתונים הפופולריות ביותר אם לא הכי פופולריות בימנו, Pandas עובדת היטב עם מודולים רבים אחרים של מדעי הנתונים בתוך האקוּסיטם של פיתון ונכללת בדרך כלל בכל הפקת פיתון, מלאה שmagics עם מערכת ההפעלה ועד להפצתו של ספקים מסחריים.

שאלה 4

Supervised learning היא צורה שהמאפשרת לפתח מערכת – יכול להיות מחשב ואפשר להסתכל על זה כמו תהילך למידה של תינוק, שלווה לפטור בעיות על בסיס מאגר גדול של דוגמאות "פתרונות". הלמידה עצמה נעשית באמצעות חיפוש חוזתיות על הדוגמאות שניתן לצפות אותן מהתוצאות שקיבלו בצורה הנconaה ביוטר ולאחר מכן תוכל לחזות בצורה נכונה את התשובה גם עבור דוגמאות שטרם נראו על ידי המערכת. supervised Learning unsupervised Learning שהיא יכולה למדוד את התוכנות והמבנה של אוסף דוגמאות נתונים כאשר הנתונים זמינים כפי שהם ללא תוספת תיוגים.

הבדל ביןיהם הוא שב supervised learning אני צריך שהמידע יהיה מתייג או לפחות מתייג בצורה טובה לאומת supervised learning שבו אני לא צריך שהמידע יהיה מתייג לי ואני יכול לדעת מהם מחלוקת השקילותות שקיימות ואז לפעמים יספיק לנו אפילו לתיג בצורה ידנית רק איבר מרכז מוכל מחלוקת שקיימות בכך לתייג את כל הדטה סט שלנו – כמובן כולל זאת ברגע שאנו רוצח להכניס דגימה חדשה. כמו כן ניתן להשתמש supervised learning ובכך לקבל מידע חדש על הדטה שלנו גם אם הוא מתייג – כאמור להרייך בדיקה ולגלוות שקיימות ל毛病ות שקיימות בהזנחה ולא לקחתי בחשבון. לאומת למידה מונחת שבה מחלוקת השקילותות של מוגדרות להיות רק בגודל קבוצת התיוגים של'. כמובן שיש עוד הבדלים אבל זה בקצרה ☺.

שאלה 6 a

The image shows two vertically stacked screenshots of a Jupyter Notebook interface running on a Windows desktop. Both screenshots have identical window titles: "localhost:8888/notebooks/Desktop/DATA%20SINCE/EX1/S1_AradKotzer_311226203.ipynb". The top screenshot shows the first five cells of the notebook:

```
In [3]: def read_data_from_csv(file_name):
    return pd.read_csv(file_name);

In [4]: #num 6-a
data1 = read_data_from_csv("Mobileye_risk_Barcelona.csv")
data2 = read_data_from_csv("Mobileye_risk_TelAviv.csv")

In [5]: data1 = read_data_from_csv("Mobileye_risk_Barcelona.csv")

In [6]: #num 6-a
rows1,cols1 = data1.shape
rows2,cols2 = data2.shape
print(list(data1.columns))
print(list(data2.columns))

['Unnamed: 0', 'geometry', 'month_timestamp', 'detection_drives_count', 'avg_speed', 'near_miss_pedestrian_ratio', 'near_mis
s_bicycle_ratio', 'near_miss_vehicle_ratio', 'avg_pedestrian_on_road_volume', 'avg_bicycle_on_road_volume', 'braking_count',
'cornering_count', 'harsh_braking_ratio', 'harsh_cornering_ratio']
['Unnamed: 0', 'geometry', 'month_timestamp', 'detection_drives_count', 'avg_speed', 'near_miss_pedestrian_ratio', 'near_mis
s_bicycle_ratio', 'near_miss_vehicle_ratio', 'avg_pedestrian_on_road_volume', 'avg_bicycle_on_road_volume', 'braking_count',
'cornering_count', 'harsh_braking_ratio', 'harsh_cornering_ratio']

In [7]: #num 6-a
print(data1["Unnamed: 0"].describe())
print(data2["Unnamed: 0"].describe())
```

The bottom screenshot shows the last three cells of the notebook, which were run after a break:

```
In [7]: #num 6-a
print(data1["Unnamed: 0"].describe())
print(data2["Unnamed: 0"].describe())

count      42948.000000
mean     21473.500000
std      12398.164017
min       0.000000
25%    10736.750000
50%    21473.500000
75%    32210.250000
max     42947.000000
Name: Unnamed: 0, dtype: float64
count      9266.000000
mean     4632.500000
std      2675.008131
min       0.000000
25%    2316.250000
50%    4632.500000
75%    6948.750000
max     9265.000000
Name: Unnamed: 0, dtype: float64

print(data1["Unnamed: 0"].mean())

In [8]: #num 6-a
data1.drop('Unnamed: 0', inplace=True, axis=1)
data2.drop('Unnamed: 0', inplace=True, axis=1)
```

שאלה b.6

In [10]:

```
#num 6-b
data1[“City”] = “Barcelona”
data2[“City”] = “Tel-Aviv”
data2.head()
```

Out[10]:

	geometry	month_timestamp	detection_drives_count	avg_speed	near_miss_pedestrian_ratio	near_miss_bicycle_ratio	near_miss_vehicle_ratio	avg_per
0	[34.772099, 32.054769], [34.772205, 32.054769]	2021-01-01	298.0	30.062881	0.000000	0.0	0.0	
1	[34.774698, 32.089434], [34.774468, 32.089478]	2021-01-01	126.0	26.210692	0.000000	0.0	0.0	
2	[34.786151, 32.074455], [34.786218, 32.074516]	2021-01-01	581.0	32.475986	0.001721	0.0	0.0	
3	[34.777766, 32.038728], [34.777856, 32.038799]	2021-01-01	294.0	34.194698	0.003401	0.0	0.0	
4	[34.763697, 32.052037], [34.763973, 32.052226]	2021-01-01	75.0	30.859624	0.000000	0.0	0.0	

שאלה c.6

In [11]:

```
#num 6-c
Dmerge = pd.concat([data1, data2], ignore_index = True)
rows3,cols3 = Dmerge.shape
print ( “There are”, rows3 ,”rows in our data.” )
print ( “There are”, cols3 ,”cols in our data.” )
```

There are 52214 rows in our data.
There are 14 cols in our data.

In [12]:

```
print ( “There are”, rows3 ,”rows in our data.” )
print ( “There are”, cols3 ,”cols in our data.” )
```

There are 52214 rows in our data.
There are 14 cols in our data.

In [13]:

```
Dmerge
```

Out[13]:

	geometry	month_timestamp	detection_drives_count	avg_speed	near_miss_pedestrian_ratio	near_miss_bicycle_ratio	near_miss_vehicle_ratio	avg_per
0	[202711, 41.399904], [2027, 41.399886]	2021-01-01	53.0	16.677624	0.000000	0.0	0.000000	
1	[2116737,							

שאלה 6 d+e.6

The screenshot shows a Jupyter Notebook interface with two code cells and their outputs.

In [14]:

```
#num 6-d
city = {'Tel-Aviv': 1, 'Barcelona': 0}
Dmerge['City'] = Dmerge['City'].apply(lambda x: city[x])
```

In [15]:

```
#num 6-e
print(Dmerge.iloc[list(range(10))])
```

Output of In [15]:

```
geometry month_timestamp \
0 [[2.202711, 41.399986], [2.2027, 41.399886]] 2021-01-01
1 [[2.116737, 41.367855], [2.116674, 41.368069]] 2021-01-01
2 [[2.134581, 41.383152], [2.132885, 41.382576]] 2021-01-01
3 [[2.171651, 41.382927], [2.171646, 41.382905]] 2021-01-01
4 [[2.131998, 41.359155], [2.131434, 41.358718]] 2021-01-01
5 [[2.213811, 41.437025], [2.213706, 41.43708]] 2021-01-01
6 [[2.215053, 41.439242], [2.21528, 41.439474]] 2021-01-01
7 [[2.213761, 41.436599], [2.213849, 41.436662]] 2021-01-01
8 [[2.213849, 41.436662], [2.213904, 41.436744]] 2021-01-01
9 [[2.156947, 41.37398], [2.156686, 41.373625]] 2021-01-01

detection_drives_count avg_speed near_miss_pedestrian_ratio \
0 53.0 16.677624 0.000000
1 78.0 20.446870 0.000000
2 108.0 19.869511 0.046296
3 266.0 29.255693 0.000000
4 86.0 51.434637 0.000000
5 126.0 27.616541 0.000000
6 52.0 19.130008 0.000000
```

שאלה 7

The screenshot shows a Jupyter Notebook interface with three code cells and their outputs.

In [16]:

```
#num 7
print(list(Dmerge.columns))
```

Output of In [16]:

```
['geometry', 'month_timestamp', 'detection_drives_count', 'avg_speed', 'near_miss_pedestrian_ratio', 'near_miss_bicycle_ratio', 'near_miss_vehicle_ratio', 'avg_pedestrian_on_road_volume', 'avg_bicycle_on_road_volume', 'braking_count', 'cornering_count', 'harsh_braking_ratio', 'harsh_cornering_ratio', 'City']
```

In [17]:

```
#num 8
temp = Dmerge[Dmerge.near_miss_pedestrian_ratio > 0.2]
(temp)
```

Out[17]:

	geometry	month_timestamp	detection_drives_count	avg_speed	near_miss_pedestrian_ratio	near_miss_bicycle_ratio
851	[2.211271, 41.42111], [2.211579,	2021-01-01	98.0	22.076602	0.244898	0.0001

שאלה 8

The screenshot shows two Jupyter Notebook sessions. The left session (In [22]) displays a code snippet for calculating counts of specific events across cities and then sorting them by average speed. The right session (In [59]) shows a similar process for calculating near-miss ratios and sorting the results.

```

In [22]: #num 8
temp[temp['City' == 'Barcelona']].value_counts()
num_of_Tel_Aviv = (temp['City' == 'Tel Aviv']).value_counts()
print('num of total 0.2 is - ', len(temp), "num of 0.2 in Barcelona", num_of_Barcelona, "num of 0.2 in Tel_Aviv", num_of_Tel_Aviv)
print('num of total 0.2 is - ', len(temp), "num of 0.2 in Barcelona", num_of_Barcelona, "num of 0.2 in Tel_Aviv", num_of_Tel_Aviv)

In [23]: #num 9
dfMerge['near_miss_vehicle_ratio'].tail(10)
dfMerge['near_miss_pedestrian_ratio'].tail(10)
Out[23]: geometry month_timestamp detection_drives_count avg_speed near_miss_bicycle_ratio near_miss_pedestrian_ratio avg

```

```

In [59]: #num 8
temp = dfMerge[dfMerge['near_miss_pedestrian_ratio' > 0.2] | temp['City'].value_counts()]
Out[59]: 0    113
1     6
Name: City, dtype: int64

```

```

In [18]: temp.sort_values(by='avg_speed')

```

אפשר לראות שיש 113 ערכים שגדולים מ 0.2 מטל אביב ו 6 ערכים שגדולים מ 0.2 מרצלונה, מה שניתן להגיד על מקטעים אלה הוא שהיחס המרחק מההنجשות מכך שהוא גדול ממשועותיה בברצלונה, דבר שהוא הגיוני במידת מסוימת שכן כמות הדגימות מרצלונה הרבה יותר גדולה. בתל אביב הוא סיבי 9000 בעוד שברצלונה הוא 46 אלף, ועודין בכך יחסית יש יותר התקראבות בצורה יחסית בברצלונה מאשר בתל אביב. לכן אפשר להסיק שבברצלונה פחות שומרים מרחוק.

שאלה 9

The screenshot shows two Jupyter Notebook sessions. The left session (In [21]) displays a code snippet for calculating the number of collisions per hour and then sorting them by average speed. The right session (In [19]) shows a similar process for calculating near-miss ratios and sorting the results.

```

In [21]: #num 8
dfMerge['hour'].value_counts().sort_index().tail(10)
dfMerge['hour'].value_counts().tail(10)
Out[21]: hour
0000    121224
0100    121224
0200    121224
0300    121224
0400    121224
0500    121224
0600    121224
0700    121224
0800    121224
0900    121224

```

```

In [19]: #num 9
dfMerge['near_miss_vehicle_ratio'].tail(10)
dfMerge['near_miss_pedestrian_ratio'].tail(10)
Out[19]: geometry month_timestamp detection_drives_count avg_speed near_miss_bicycle_ratio near_miss_pedestrian_ratio avg

```

מספר התוצאות הינו מ-121266, בהתחלה חשבתי שזה פניות לריגול של מוביל או אז נראתה פשוט הסטטוס תקול – שכן ערך צזה גובה אין אפשרי, או מכוניות שנסע רק במעגלים – לאחר שהבנתי שזה סטיות מהנתיב הבנתי שכנראה עדין יש בעיה בחישון, למורות שנראה גם שאין קורלציה מספקת טוביה בין בין הנהגים האחרים שסתומים הרבה נתיבים וכן לווחצים גם הרבה על הבלתי בעוד הנהג הזה לחץ רק פעם אחת. שורה תחתונה תקלה בחישון או נהג שיכור שמיליכך לנו את הדטה סט.

שאלה 10

```
In [63]: # num 20
T = Dmerge.City.value_counts(normalize=True).to_frame()
describe = Dmerge.describe().T
TEL = T.loc[1,:]
TEL['1'] = 1
describe['Tel_Aviv'] = TEL
describe[['Barcelona']] = B
describe
```

	count	mean	std	min	25%	50%	75%	max	Tel-Aviv	Barcelona
detection_drives_count	52214.0	296.992255	621.326235	50.000000	81.000000	143.000000	306.000000	25047.000000	0.177462	0.822538
avg_speed	52214.0	33.524120	14.434483	8.274068	23.952468	29.538498	38.080639	98.826219	0.177462	0.822538
near_miss_pedestrian_ratio	52214.0	0.005985	0.021517	0.000000	0.000000	0.000000	0.000000	0.540230	0.177462	0.822538
near_miss_bicycle_ratio	52214.0	0.006934	0.033778	0.000000	0.000000	0.000000	0.007407	0.357895	0.177462	0.822538
near_miss_vehicle_ratio	52214.0	0.00639	0.015728	0.000000	0.000000	0.000000	0.007407	0.357895	0.177462	0.822538
avg_pedestrian_on_road_volume	52214.0	0.02917	0.102580	0.000000	0.000000	0.019320	0.079847	1.610309	0.177462	0.822538
avg_bicycle_on_road_volume	52214.0	0.01736	0.045937	0.000000	0.000000	0.015317	0.061316	0.177462	0.177462	0.822538
braking_count	52214.0	35.759808	75.027213	0.000000	4.000000	14.000000	38.000000	2715.000000	0.177462	0.822538
coming_count	52214.0	149.194222	309.350413	0.000000	35.000000	70.000000	155.000000	7130.000000	0.177462	0.822538
harsh_braking_ratio	52214.0	0.00000	0.002600	0.000000	0.000000	0.000000	0.000000	0.250000	0.177462	0.822538
harsh_coming_ratio	52214.0	0.02587	0.070260	0.000000	0.000000	0.01089	0.019625	1.000000	0.177462	0.822538
City	52214.0	0.177462	0.382065	0.000000	0.000000	0.000000	0.000000	1.000000	0.177462	0.822538

ניתן לראות ש אחוז הדגימות מטל אביב הוא 17.7% וברצלונה הוא 82.25%

שאלה 11

```
In [65]: # num 11,
# We can understand that in TEL AVIV there is a more detection_drives_count, That means drivers drive a lot more
# It could be understood from the count and the std for each of the city's
Dmerge['detection_drives_count'].describe()
detection_Tel_Aviv = Dmerge[Dmerge.City == 1]
describe_1 = detection_Tel_Aviv['detection_drives_count'].describe()
detection_Barca = Dmerge[Dmerge.City == 0]
describe_2 = detection_Barca['detection_drives_count'].describe()

describe_3 = pd.concat([describe_1, describe_2], axis=1, ignore_index=True)
describe_3.columns = ['Tel_Aviv', 'Barcelona']
describe_3
```

	Tel_Aviv	Barcelona
count	9269.000000	42948.000000
mean	557.363048	239.723270
std	1303.459900	291.389532
min	50.000000	50.000000
25%	93.000000	79.000000
50%	188.000000	136.000000
75%	481.000000	279.000000
max	25047.000000	9325.000000

```
In [66]: # num 12
# 1) It can be seen that there is a relative correlation between the speed of travel (avg_speed) and the length of travel (de
```

ניתן להבין מההשוואה שבתאל אביב מtbיצעות מסוימות יותר מאשר במדינת ישראל. דבר זה נובע dara מכיוון שהנהג הספרדי מזיז פחות את הרכב ופחות תלי בהוא, בעוד הנהיג התל אביב איבר תלי ברכב לנסיעות יומיומיות לעובדה וחזרה. יכול להיות עקב בערים בתחרות ציבורית.

שאלה 12

The screenshot shows a Jupyter Notebook interface with the following details:

- Code Cell (In [27]):**

```
# num 12
corr = Dmerge.corr(method ='pearson')
corr_Barca = detection_Barca.corr(method ='pearson')
corr_Tel = detection_Tel_avliv.corr(method ='pearson')
corr3 = pd.concat([corr_Barca, corr_Tel] )
corr3
```
- Output Cell (Out[27]):**

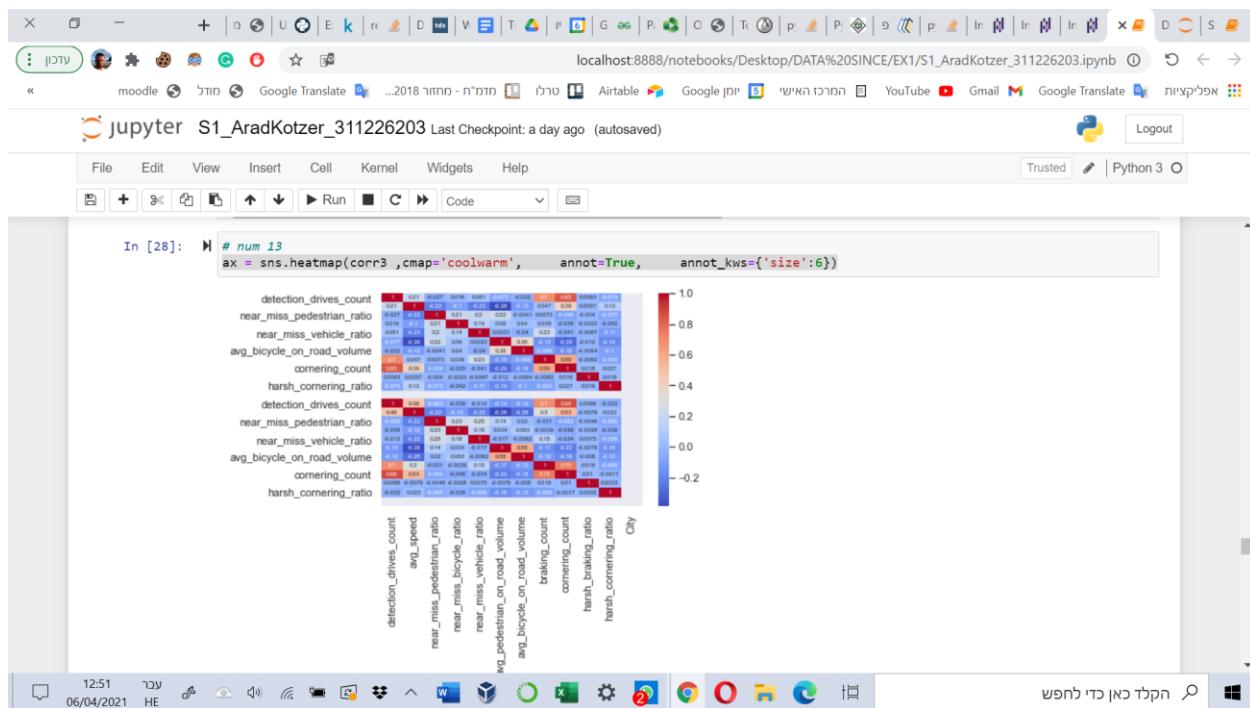
	detection_drives_count	avg_speed	near_miss_pedestrian_ratio	near_miss_bicycle_ratio	near_miss_vehicle_ratio	avg_pede
detection_drives_count	1.000000	0.214400	-0.026649	0.015973	0.060964	
avg_speed	0.214400	1.000000	-0.222009	-0.102783	-0.231479	
near_miss_pedestrian_ratio	-0.026649	-0.222009	1.000000	0.208737	0.199944	
near_miss_bicycle_ratio	0.015973	-0.102783	0.208737	1.000000	0.136843	
near_miss_vehicle_ratio	0.060964	-0.231479	0.199944	0.136843	1.000000	
avg_pedestrian_on_road_volume	-0.076556	-0.379642	0.222858	0.060239	0.002339	
avg_bicycle_on_road_volume	-0.031981	-0.119431	-0.004090	0.040312	-0.040209	
braking_count	0.695882	0.046934	0.007332	0.038526	0.231210	
cornering_count	0.847205	0.389987	-0.098872	-0.034980	-0.041361	
harsh_braking_ratio	0.008301	0.005729	-0.003976	-0.002319	-0.006677	
harsh_cornering_ratio	-0.074068	0.131263	-0.073202	-0.042094	-0.111498	
City	NaN	NaN	NaN	NaN	NaN	
detection_drives_count	1.000000	0.484662	-0.062983	-0.037985	-0.011663	
avg_speed	0.484662	1.000000	-0.220746	-0.119433	-0.217547	

מו 1') ניתן לראות כי יש מתאם יחס' בין מהירות הנסיעה (avg_speed) לבין כמות הנסיעות (detection_drives_count) כך שבין תושבי תל אביב (0.484662) לעומת מתאם פחות חזק בברצלונה (0.214400)

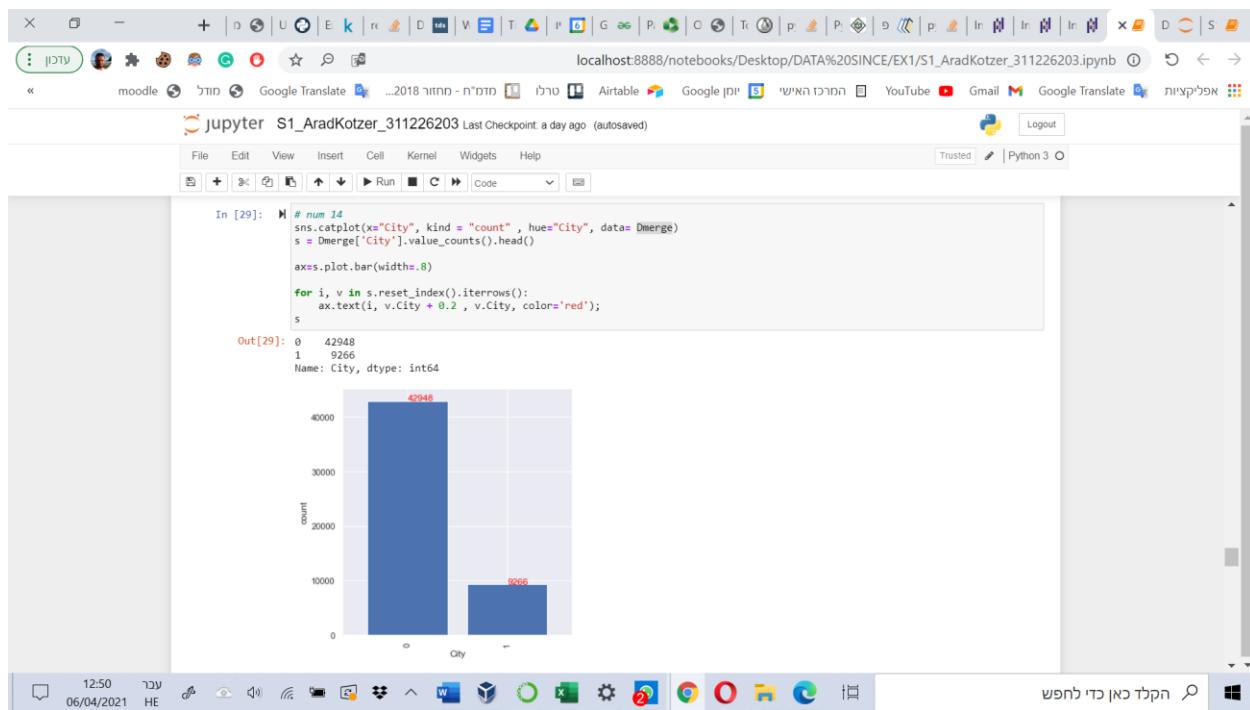
דבר זה יכול להיות בגלוי יותר פקקים בברצלונה או תחבורה ציבורית יותר טובה ובכך הם פחות מוצאים את האוטו.

מו 2') ניתן לראות כי יש מתאם גובה יחס' בין מספר הסטיות מהנתיב שbowצעו לבין מספר הלחיצות על הבלמים. בתל אביב (0.754685) ניתן לראות מתאם חזק יותר מאשר בברצלונה (0.594944) דבר זה יכול להיות משום שבישראל יותר אנשים מאטים לפני שהם עוברים נתיב או שאפשר לקטול אותם בתור נהגים עם אופי נהיגה שונה, مثل הנהגים הספרדים. ושונה בין הנהגים לבין עצםם. ככלומר קיימים מתאם יותר חזק בין אנשים שעוברים יותר נתיבים לבין לוחצים על בלם. וההבדל יכול להיות שהספרדים בכלל שהם נסעים פחות ידעים נהוג פחות טוב ולשלב בין מעבר נתיבים לבין לחיצה על הבלם.

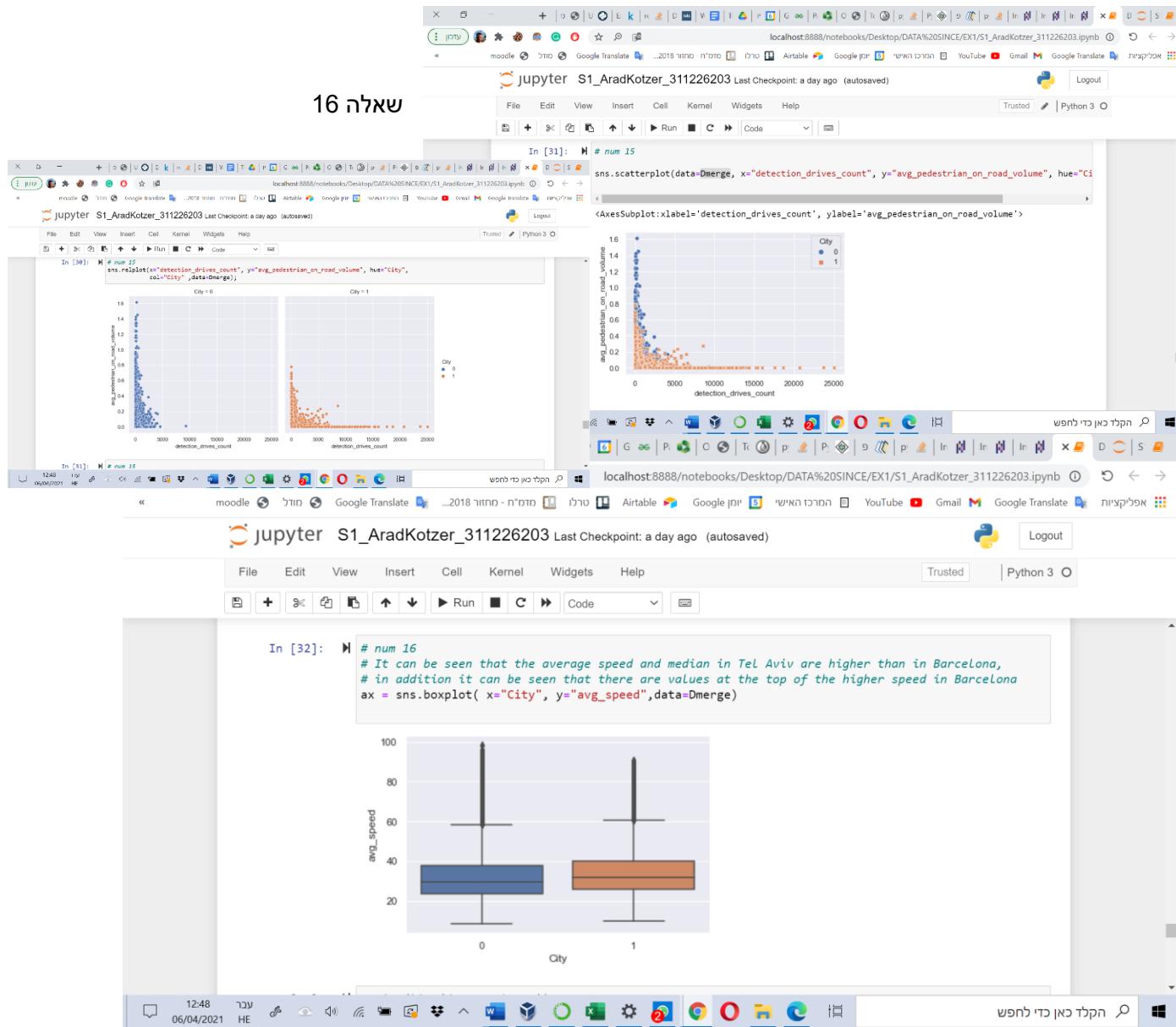
שאלה 13



שאלה 14



שאלה 15



ניתן לראות כי המהירות הממוצעת והחציון בתל אביב גבוהים יותר מאשר בברצלונה, בנוסף ניתן לראות כי ישנו ערכיים בראש המהירות הגבוהה יותר בברצלונה וגם המהירות הנמוכה, אפשר להסיק שדבר זה יכול לחזק את ההנחה של שבספרד יש יותר פקקים ותחבורה – דבר שימושי יותר נרחב בתחום ציבוריות מהסעיפים הקודמים, וגם לירידה במהירות הנסיעה אצל הנהגים. כמובן שגם להיות שזה דבר תרבותי או יותר אכיפת מהירות של המשטרה המקומיית אך לדעתו הדבר נובע מריבוי פקקי תנועה.

שאלה 21+22

The screenshot shows a Jupyter Notebook interface with the following code:

```
In [35]: # num 21  
#make a Dataframe  
  
N_data1 = data1[['detection_drives_count', 'avg_speed', 'braking_count', 'cornering_count']].copy()  
  
N_data1.columns  
  
Out[35]: Index(['detection_drives_count', 'avg_speed', 'braking_count',  
                 'cornering_count'],  
                dtype='object')  
  
In [38]: # create a new XLSX workbook  
workbook = xlswriter.Workbook('311226203_Arad_kotzer1.xlsx')  
worksheet = workbook.add_worksheet()  
  
In [ ]: #num 23  
  
worksheet.write(0, 0, 'Index')  
worksheet.write(0, 1, 'detection_drives_count')  
worksheet.write(0, 2, 'avg_speed')  
worksheet.write(0, 3, 'braking_count')  
worksheet.write(0, 4, 'cornering_count')
```

The notebook is titled "jupyter S1_AradKotzer_311226203" and shows code for creating an XLSX file from a DataFrame. The code includes writing the column names to the first row of the sheet.

שאלה 24-23

The screenshot shows a Jupyter Notebook interface with the following code:

```
In [ ]: # create a new XLSX WORKBOOK  
workbook = xlswriter.Workbook('311226203_Arad_kotzer1.xlsx')  
worksheet = workbook.add_worksheet()  
  
In [ ]: #num 23  
  
worksheet.write(0, 0, 'Index')  
worksheet.write(0, 1, 'detection_drives_count')  
worksheet.write(0, 2, 'avg_speed')  
worksheet.write(0, 3, 'braking_count')  
worksheet.write(0, 4, 'cornering_count')  
  
In [ ]: #num 23 - 24  
# Loop through indices and rows in the dataframe using iterrows  
  
for index, row in N_data1.iterrows():  
    worksheet.write(index + 1, 0, index + 1)  
    worksheet.write(index + 1, 1, N_data1.iloc[index][0])  
    worksheet.write(index + 1, 2, N_data1.iloc[index][1])  
    worksheet.write(index + 1, 3, N_data1.iloc[index][2])  
    worksheet.write(index + 1, 4, N_data1.iloc[index][3])  
  
workbook.close()
```

The notebook is titled "jupyter S1_AradKotzer_311226203" and shows code for creating an XLSX file from a DataFrame using `iterrows`. The code loops through each row of the DataFrame and writes its values to the corresponding row in the XLSX file. The first row contains the column indices (0-4) and the second row contains the actual data values.

פתרונות בדף יותר עיליה.

localhost:8888/notebooks/Desktop/DATA%20SINCE/EX1/S1_AradKotzer_311226203.ipynb

jupyter S1_AradKotzer_311226203 Last Checkpoint: a day ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
worksheet.write(0, 4, 'cornering_count')

In [ ]: #num | 24
# Loop through indices and rows in the dataframe using iterrows

for index, row in N_data1.iterrows():
    worksheet.write(index + 1, 0, index + 1)
    worksheet.write(index + 1, 1, N_data1.iloc[index][0])
    worksheet.write(index + 1, 2, N_data1.iloc[index][1])
    worksheet.write(index + 1, 3, N_data1.iloc[index][2])
    worksheet.write(index + 1, 4, N_data1.iloc[index][3])
workbook.close()

In [ ]: #num 23 - 24
# Another way to solve the problem more efficiently, then using iterrows
N_data1.to_excel("311226203_Arad_kotzer3.xlsx", header=True, index=True, index_label = 'Index')
worksheet.write('A1', 'Index')

In [ ]: index = pd.DataFrame()
detection_drives_count = pd.DataFrame()
avg_speed = pd.DataFrame()
```

הקלד כאן כדי לחפש

שאלה 25

localhost:8888/notebooks/Desktop/DATA%20SINCE/EX1/S1_AradKotzer_311226203.ipynb

jupyter S1_AradKotzer_311226203 Last Checkpoint: a day ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
names = [index, detection_drives_count, avg_speed, braking_count, cornering_count]

for i in range(len(frames)):
    filename = '311226203_Arad_kotzer1'+ names[i] + '.xlsx'
    writer = ExcelWriter(311226203_Arad_kotzer1)
    print(311226203_Arad_kotzer1)
    frames[i].to_excel(writer, 'Sheet1', index=False)
    writer.save()

In [ ]: # num 25
N_data1.to_csv("311226203_Arad_kotzer2.csv", header=True, index=True, index_label = 'Index' )
```

הקלד כאן כדי לחפש

שאלה 26

Excel Sheet: Mobileye_risk_Barcelonah

Cell F2 contains the formula: =ROUND(E2,-1)

P	O	M	K	J	H	G	F	E	D	C	B	A
ash_conharsh_brai	cornering_braking_ccavg	bicyclavg_pednear_miss	near_miss	near_miss	avg	specavg	speed	detection	detection_drives_count_group	month	tim	geometry
0.03333	0	30	2	0	0.018668	0	0	0	16.67762	20	53	=ROUND(E2,-1)
0	0	51	41	0	0	0.064103	0	0	20.44687	20	78	80 #####
0.007344	0	19	25	0.009259	0.083333	0	0	0.046296	19.86951	20	108	110 #####
0.009009	0	185	16	0.030075	0.022556	0	0	0	29.25569	30	266	270 #####
0	0	56	14	0	0.011628	0	0	0	51.43464	50	86	90 #####
0.044071	0	95	10	0	0	0.02381	0	0	27.61654	30	126	130 #####
0	0	20	15	0	0	0.076923	0	0	19.13001	20	52	50 #####
0.038738	0	92	1	0	0	0	0	0	28.55917	30	107	110 #####
0.065657	0	92	3	0	0	0	0	0	29.10925	30	106	110 #####
0	0	0	0	0	0.026667	0.306667	0	0	15.24046	20	75	80 #####
0	0	81	28	0.013699	0.027397	0.006849	0	0	27.36015	30	146	150 #####
0.008511	0	47	20	0	0.016129	0	0	0	42.08545	40	62	60 #####
0	0	39	7	0	0.110092	0	0	0	29.53365	30	109	110 #####
0	0	63	23	0.004545	0.072727	0.004545	0	0.009091	28.39392	30	220	220 #####
0.014036	0	54	10	0.015038	0.045113	0	0	0	29.33303	30	133	130 #####
0	0	36	30	0.010417	0.09375	0.072917	0	0.010417	19.76953	20	96	100 #####
0.036364	0	33	7	0	0.101852	0	0	0.009259	23.09466	20	108	110 #####
0.016056	0	116	39	0.001634	0.045752	0.004902	0	0.001634	28.95406	30	612	610 #####
0	0	10	18	0.010638	0.212766	0	0	0.010638	28.29307	30	94	90 #####
0	0	232	16	0	0.002865	0	0	0.002865	30.43566	30	349	350 #####
0.17826	0	101	26	0	0.0625	0	0	0	26.44466	30	176	180 #####

שאלה 27

Excel Sheet: Mobileye_risk_Barcelonah

Cell E2 contains the formula: =ROUND(G2,-1)

N	M	L	K	J	I	H	G	F	E	D	C	B	A	
cornering_braking_ccavg	bicyclavg_pednear_miss	near_miss	near_miss	avg	specavg	speed	group	detection	detection_drives_count_group	month	tim	geometry	1	
30	2	0	0.018668	0	0	0	0.16.67762	=ROUND(G2,-1)	53	50 #####	[[2.202711	0	2	
51	41	0	0	0.064103	0	0	0.20.44687	20	78	80 #####	[[2.116737	1	3	
19	25	0.009259	0.083333	0	0	0	0.046296	19.86951	20	108	110 #####	[[2.134581	2	4
185	16	0.030075	0.022556	0	0	0	0.29.25569	30	266	270 #####	[[2.171651	3	5	
56	14	0	0.011628	0	0	0	0.51.43464	50	86	90 #####	[[2.131998	4	6	
95	10	0	0	0.02381	0	0	0.27.61654	30	126	130 #####	[[2.213811	5	7	
20	15	0	0	0.076923	0	0	0.19.13001	20	52	50 #####	[[2.215053	6	8	
92	1	0	0	0	0	0	0.28.55917	30	107	110 #####	[[2.213761	7	9	
92	3	0	0	0	0	0	0.29.10925	30	106	110 #####	[[2.213849	8	10	
0	0	0.026667	0.306667	0	0	0	0.15.24046	20	75	80 #####	[[2.156947	9	11	
81	28	0.013699	0.027397	0.006849	0	0	0.27.36015	30	146	150 #####	[[2.170887	10	12	
47	20	0	0.016129	0	0	0	0.42.08545	40	62	60 #####	[[2.168726	11	13	
39	7	0	0.110092	0	0	0	0.29.53365	30	109	110 #####	[[2.176831	12	14	
63	23	0.004545	0.072727	0.004545	0	0	0.009091	28.39392	30	220	220 #####	[[2.149416	13	15
54	10	0.015038	0.045113	0	0	0	0.29.33303	30	133	130 #####	[[2.092548	14	16	
36	30	0.010417	0.09375	0.072917	0	0	0.010417	19.76953	20	96	100 #####	[[2.095236	15	17
33	7	0	0.101852	0	0	0	0.009259	23.09466	20	108	110 #####	[[2.089091	16	18
116	39	0.001634	0.045752	0.004902	0	0	0.001634	28.95406	30	612	610 #####	[[2.208435	17	19
10	18	0.010638	0.212766	0	0	0	0.010638	28.29307	30	94	90 #####	[[2.20737,	18	20
232	16	0	0.002865	0	0	0	0.002865	30.43566	30	349	350 #####	[[2.196096	19	21
101	26	0	0.0625	0	0	0	0.26.44466	30	176	180 #####	[[2.205171	20	22	

שאלה 28

מהירות של 30 קמ"ש – הינו ספירות של 16415 לאזרע

The screenshot shows a Microsoft Excel spreadsheet titled "Mobileye_risk_Barcelon". The data is presented in a table with the following columns:

geometry	מהירות שורה	ספירה של
	623	10 4
	12607	20 5
	16415	30 6
	6688	40 7
	2673	50 8
	1387	60 9
	1328	70 10
	905	80 11
	314	90 12
	8	100 13
	42948	סך כל
		15
		16
		17
		18
		19
		20
		21
		22

The total count (sum) is 42948.

שאלה 29

במהירות של 100 קמ"ש – הינו 1144 ספירות של detection_drives_count ולכך הכى פחות שיכחה.

The screenshot shows a Microsoft Excel spreadsheet titled "Mobileye_risk_Barcelon". A PivotTable is displayed, showing the following data:

detection_drives_count	סך כל
74706	10 4
2312462	20 5
3990427	30 6
1604751	40 7
629808	50 8
478163	60 9
649991	70 10
447542	80 11
106641	90 12
1144	100 13
10295635	סך כל

The total count (sum) is 10295635.

שאלה 30

במהירות של 30 קמ"ש – הינו 1925684 ספירות של סטיות מהנתיב ו- 549714 חריגות מהנתיב ולכן
בערך מוחלט הוא הגدول ביותר.

	סימון שורה	סימון של סטיה	סימון של חריגה	סהם כולל
	3	3960	9105	10 4
	2	283389	689869	20 5
	1	549714	1925684	30 6
	2	215837	969480	40 7
	3	78363	443705	50 8
	4	46248	344375	60 9
	5	45169	461192	70 10
	6	24860	311411	80 11
	7	5266	71317	90 12
	8	44	806	100 13
	9			1252850 5226944
	10			
	11			
	12			
	13			
	14			
	15			
	16			
	17			
	18			
	19			
	20			
	21			
	22			

Solver F חלק

שאלה 1

הבנייה לאחר שהכנתי את טבלה שהייתה קיימת עוד טבלה בקובץ להגשה.

Product	Unit Price (\$)	Unit Cost (\$)
Laptop	120	100
Monitor	200	180
Speaker	50	30
Screen	300	250

Component	Motherboard	Working Hours	Cables	Resistors	Meters of Wires
Speaker	1	0.25	3	7	6
Amplifier	1	0.35	26	44	3
Receiver	1	0.38	28	52	3
Screen	1	0.15	10	6	1

שאלה 2

The screenshot shows two Excel windows. The top window displays the Solver Parameters dialog box for a model named 'Q33'. The 'Set Objective' field is set to 'Value Of: \$Q\$33' with the 'Min' radio button selected. The 'By Changing Variable Cells' field is set to '\$B\$25:\$B\$28'. The 'Subject to the Constraints' section contains several linear inequalities involving cells like \$B\$25, \$H\$12, \$I\$12, \$L\$25, \$M\$28, \$S\$105, \$T\$105, \$U\$25, and \$V\$25. Below these are buttons for 'Add', 'Change', 'Delete', 'Reset All', and 'Load/Save'. A checkbox for 'Make Unconstrained Variables Non-Negative' is checked. The 'GRG Nonlinear' tab is selected under 'Select a Solving Method'. The 'Solving Method' dropdown indicates 'Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for Linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.' The bottom window shows the worksheet with data for products Speaker, Amplifier, Receiver, and Screen, their unit costs, and profits. It also includes tables for 'Monetary Data', 'Production Data', 'Components needed for 1 unit', 'Component Inventory', and 'Total Profit'. The total profit is calculated as 24001.7443.

קובץ סופי.

שאלה 3

זהה אילו מהתאים השתנו. מי מבין התאים הללו מייצג את תאית הHELLTAÜ עבורה הבעיה? התאים שהשתנו הינם התאים של כמות היחידות – ונה שמייצג כמה נמכר מכל מוצר (בחירתך לאפשר מספרים עשרוניים כי אולי ניתן לפצל בין חדשניים ואז נגיע לפתרון יותר אידיאלי). כמות היחידות שיצרו משפיעות על החומרים שנדרש ליצור עבורה כל " מוצר", הערכים שנשארו קבועים הינו הצד האפשרי לשימוש – (מה שיש לנו במלאי – כמו אורק כבל כולל, חוטים כולל, שעות עבודה כולל) המחרירים גם של הוצאות וגם של היצור פר מוצר נשארו זהים. הדרישות ליצור כל מוצר נשארו זהים.

שאלה 4

הסולבר מצא שבכדי למסס את הרוחים יש ליצור 0 מסכים, 44,4 מקלטים, 0 מגברים, 1.9 רמקולים. זאת נראה משומ שnitן ליראות שהדרישות של הפריטים ליצור כל מסך נמכים משמעותית ביחס לערכיהם האחרים אך מבחינת מחיר עלות למול רוח הוא לא משמעותית יותר משתלים וכן בחומר עדיין "לנצל" גם יצור של מינורי של מוצרים ולא רק מסכים, בנוסף ניתן ליראות שרמקולים נבחרו יחסית מעט וזאת נראה משומ שנדרש כמות גדולה יחסית של חוטים בצד"י ליצור כל פריט. כמו כן ניסיתי להבין מדוע הסולבר בחר ליצור את רוב המוצרים המיוצרים כמקלטים משומ שnitן ליראות שהרווח הצפוי מהם הוא משמעותית גבוהה ביחס לפריטים אחרים.