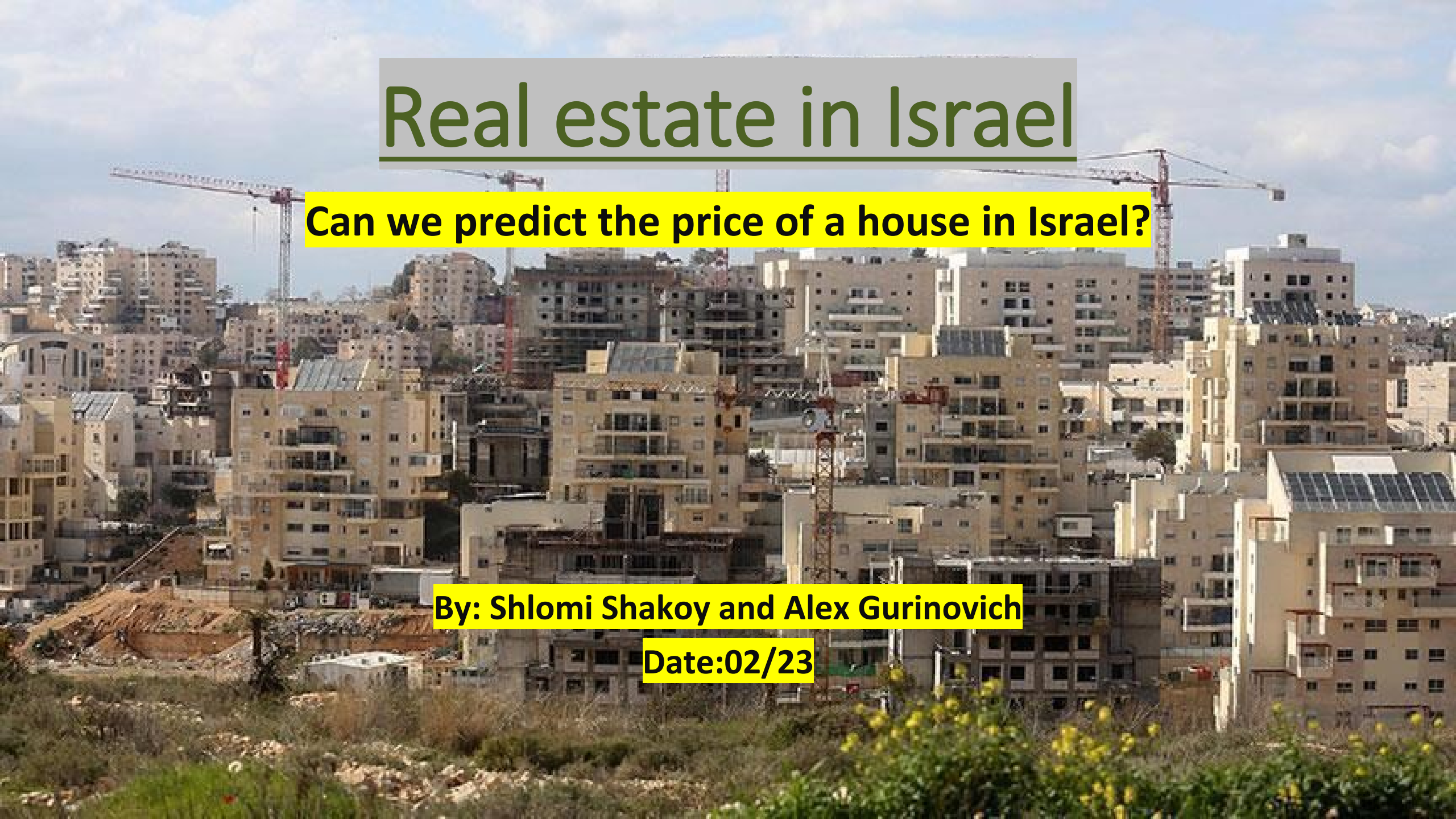
Real estate in Israel

Can we predict the price of a house in Israel?



By: Shlomi Shakoy and Alex Gurinovich

Date:02/23

The Israeli housing prices these days:

A little bit of background:

The Israeli housing market has been in a strange situation in recent months. Far fewer transactions have been recorded, but prices have risen dramatically.

As of the 12-month period, ending September 2022, housing prices increased by 19%. Meanwhile, a review by the Finance Ministry, published earlier this month, shows that in Q2/2022, 27,000 units were purchased on the open market — a 19% drop from the same period in 2021 and the lowest level since Q3/2020.

Buyers are sitting on the sidelines waiting for prices to fall, after the Bank of Israel's interest rate hiker. Moreover, they are hoping that some real estate investors will soon realize their profits, by putting their assets on the market, adding to supply.

Why houses in Israel are so expensive?

There are many parameters that affect the housing prices.

In our project, we were concentrating on the city, size, floor, number of rooms, the type of house(apartment, private house, duplex, penthouse).

Our data relies on “yad2” website, from where we extracted the data.

Those parameters assisted us to predict the price of an average house in Israel.

The importance of the housing price prediction:

The main goal of the final model is to predict the price of an apartment in Israel.

The model can be useful for investors, real estate brokers, etc.

We decided to choose this topic since we were interested about the housing crisis, and we wanted to determine if we were willing to invest in the future.

Our main steps:

1) **Obtaining data:** web crawling with selenium library in order to collect and obtain the data from yad2 website.

2) **Scrubbing data:** cleaning, formatting, and filtering the data.

3) **Exploring data:** visualizing and understanding the data. Furthermore, we reread our data before the modeling test.

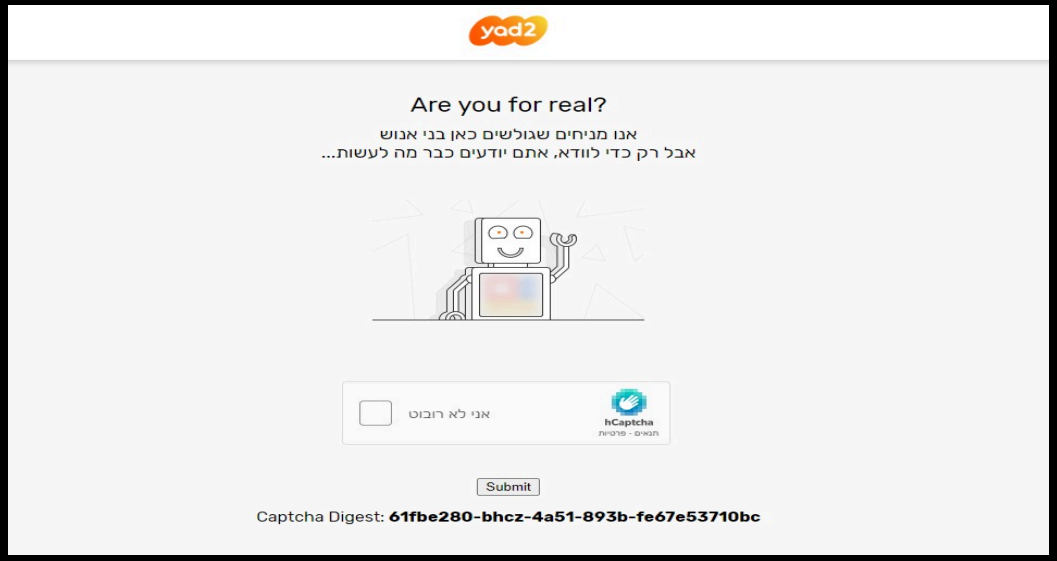
4) **Modeling data:** clustering the data info groups, modeling with unsupervised learning- random forest regressor and supervised learning-linear regression.

Our first step-web crawling:

Main tool:

Selenium library.

We encountered with captcha:



We had to run the program many times, in order to scan and extract the data from the website. In average, every 100 pages the captcha window appeared. It took us more than an hour to retrieve the data.

How was it committed de facto?

We ran the program with two loops- while and for.

The while loop used for iterating through the first 220 pages we extracted data from.

The inner for loop used for iteration in every house ad.

As a result, we succeeded in obtaining our data with crawling.

```
#First loop for iterating through the first 220 pages.
#Second loop to iterating through each apartment ad.
i = 0
j = 0
while(j < 220):
    i = 0
    for element in content:
        city = element.find_element(By.CLASS_NAME,'subtitle')

        street = element.find_element(By.CLASS_NAME,'title')

        floor = element.find_element(By.ID,'data_floor_{}'.format(i))

        NumOfRooms = element.find_element(By.ID,'data_rooms_{}'.format(i))

        size = element.find_element(By.ID,'data_SquareMeter_{}'.format(i))

        type_ = city.text.split(',')[0]

        price = element.find_element(By.ID,'feed_item_{}_price'.format(i))

        apartment = [city.text ,street.text ,floor.text ,NumOfRooms.text ,size.text ,type_ ,price.text]
        raw_df.loc[len(raw_df.index)] = apartment

        i += 1

    j += 1
    driver.get('https://www.yad2.co.il/realestate/forsale?propertyGroup=apartments&priceOnly=1&page={}'.format(j))
    element = WebDriverWait(driver, 5).until(EC.presence_of_element_located((By.CLASS_NAME, 'feeditem')))
    content = driver.find_elements(By.CLASS_NAME, "feeditem")
```

Scrubbing data

In this process, we encountered and dealt with some problems. On the one hand, we hadn't encountered missing values. While on the other hand, we had to make many changes in our data:

Dealing with duplicates- we had around 20 duplicates and we removed the duplicates, using drop_duplicates() function and we kept the first ones.

```
#Dropping duplicates
df.drop_duplicates(keep='first', inplace=True)
```

Replacing data, unnecessary signs and formatting by categories:

Price- we removed the sign of NIS: ₪, in order to commit conversion from string to integer.

As a result, we could choose houses that their price was less than 100m NIS.

```
#Removing the ₪ symbol from the price, as well as the commas and changing the price to integer
i = 0
j = 0
for price in df['Price(₪)']:
    x = price.split('₪')
    y = x[0].split(',')
    z = ''.join(map(str,y))
    z = z[:-1]
    df['Price(₪)'][i] = int(z)
    i += 1
```

Size- there were some houses that their size wasn’t mentioned and we had to remove it. Furthermore, we removed houses with a size above 1000sqm. Moreover, we converted the size from string to integer.

Number of rooms- we have removed houses with zero number of rooms.

```
#Dropping apartments with no rooms
noRooms = df[df['NumOfRooms']=='-'].index
df.drop(noRooms, inplace=True)

#Dropping apartments with no size
noSize = df[df['Size(SQM)']=='לא צוין'].index
df.drop(noSize, inplace=True)
```

Floor- we replaced some cells which were exhibited as ‘floor’ to 0. In addition, we converted from string to integer.

```
#Changing the ground floor to 0 for simpler calculations
groudFloor = df[df['Floor']=='קרקע'].index
for floor in groudFloor:
    df['Floor'][int(floor)] = 0
```

```
#Changing the attributes Floor, NumOfRooms and Size(SQM) to Integer and Double since it was a String
df['Floor'] = df['Floor'].astype('int')
df['NumOfRooms'] = df['NumOfRooms'].astype('double')
df['Size(SQM)'] = df['Size(SQM)'].astype('int')
```

City- we used split function in order to acquire only the name of the city.

```
#Removing unnecessary commas and words from the City label
i = 0
for city in df['City']:
    x = city.rsplit(',',1)
    df['City'][i] = x[1]
    i+=1
```

An example of our dataframe after the process:

	City	Street	Floor	NumOfRooms	Size(SQM)	Type	Price(₪)
0	גבעת שמואל	רמת הדר	15	5.5	150	דירה	3895000
1	קרית גת	כרמי גת	1	3.0	80	דירה	1650000
2	חיפה	שדרות הצבי 37	1	5.0	130	דירה	3200000
3	אילת	משעול המערב 6	2	4.0	99	דירה	1650000
5	שדרות	משעול פינס 5	0	4.0	270	דירת גן	1550000
...	...	...	...	...	...	...	...
8495	קרית מוצקין	יהודה הלוי	0	3.0	85	דירת גן	1190000
8496	ראשון לציון	נווה חוף	22	6.0	230	גג/פנטהאוז	5420000
8497	ירושלים	השיירות	2	3.5	75	דירה	3150000
8499	הרצליה	עזרא הסופר	4	4.0	108	דירה	4090000
8500	חיפה	חיפה	4	5.0	190	דירה	5200000

8150 rows × 7 columns

Eventually, we were left with 57050 pieces of information.

Exploring and visualizing the data

Our examples:

In this example, we created a new dataframe of a 4-room apartment. As a result, we could visualize the average price for a 4-room apartment in the central cities:

	City	Street	Floor	NumOfRooms	Size(SQM)	Type	Price(₪)
12	תל אביב יפו	שקד 15	7	3.0	90	דירה	2590000
22	תל אביב יפו	אלנבי	2	3.0	70	דירה	3000000
32	תל אביב יפו	הצפון הישן - צפון	7	5.5	195	גג/פנטהאוז	10390000
33	תל אביב יפו	יהודה הימית	3	3.5	120	דירה	3950000
79	תל אביב יפו	נאות אפקה ב'	8	5.0	240	גג/פנטהאוז	9625000
...	...	...	...	...	...	...	...
8402	באר שבע	הנדיב 10	1	2.5	45	דירה	740000
8420	באר שבע	הנרי קנדל 14	12	4.0	148	דירה	1650000
8442	באר שבע	אריה דולצ'ין 11	5	4.0	96	דירה	1050000
8445	באר שבע	אברהם אבינו	2	3.0	67	דירה	910000
8473	באר שבע	שרה ואהרון אהרונסון 26	2	4.0	148	דופלקס	1050000

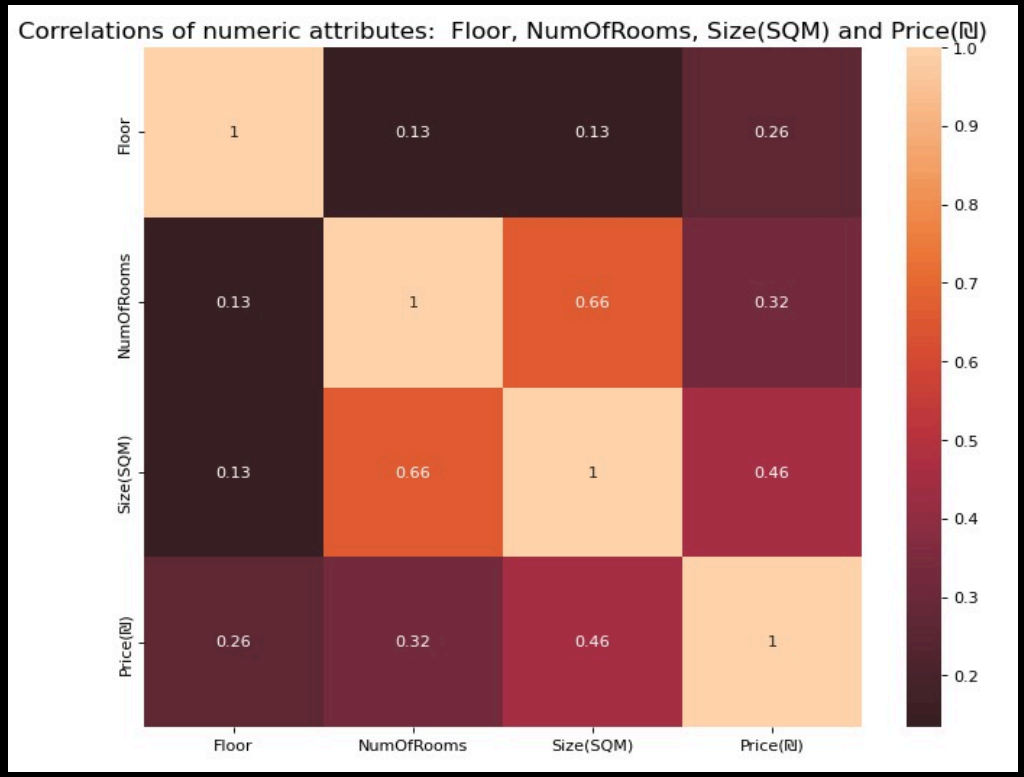
2589 rows × 7 columns

First, the Hebrew letters were opposite. In order to flip them, we used bidialg.get_display() function from bidi library

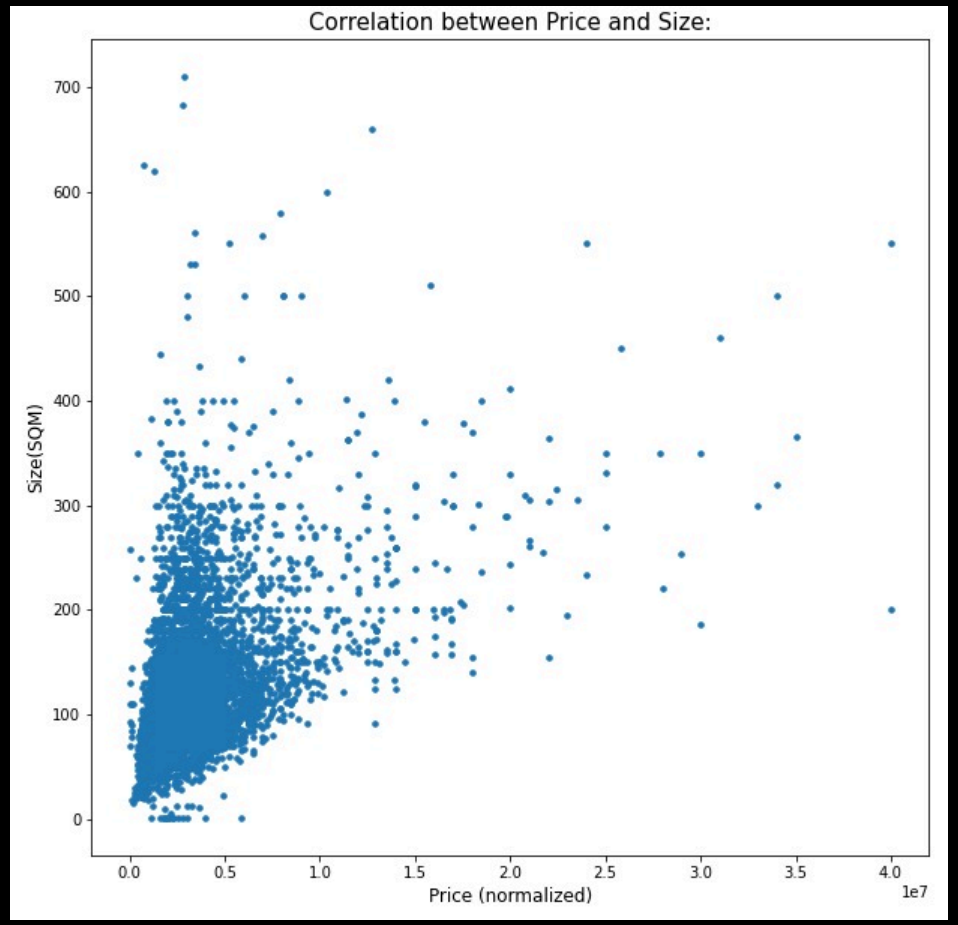
The correlations between Floor, rooms number, size and price. For instance, as we can see in the matrix the highest correlation is between size and rooms number(around 0.66).

```
dfNum = df[['Floor', 'NumOfRooms', 'Size(SQM)', 'Price(₪)']]
dfNum.corr()
```

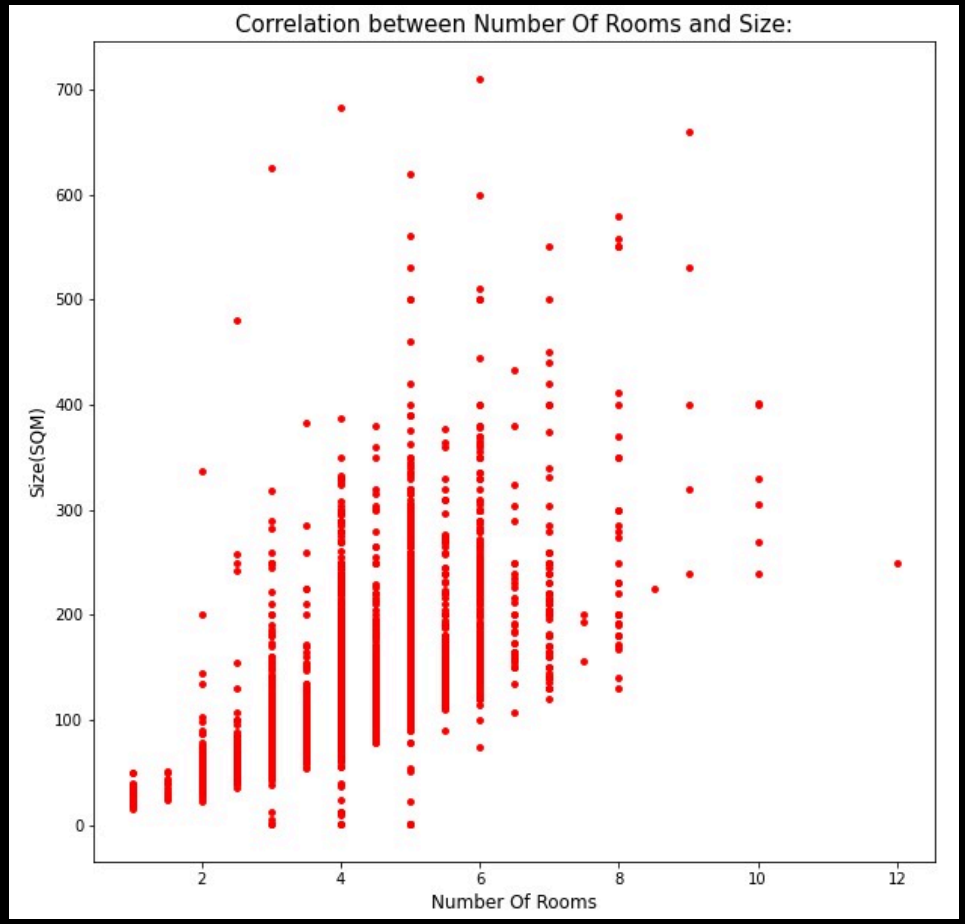
	Floor	NumOfRooms	Size(SQM)	Price(₪)
Floor	1.000000	0.134090	0.133974	0.260507
NumOfRooms	0.134090	1.000000	0.657972	0.321772
Size(SQM)	0.133974	0.657972	1.000000	0.458290
Price(₪)	0.260507	0.321772	0.458290	1.000000



The correlation between price and size:
As we can see, for the most part, the bigger the apartment, the more expensive it is.



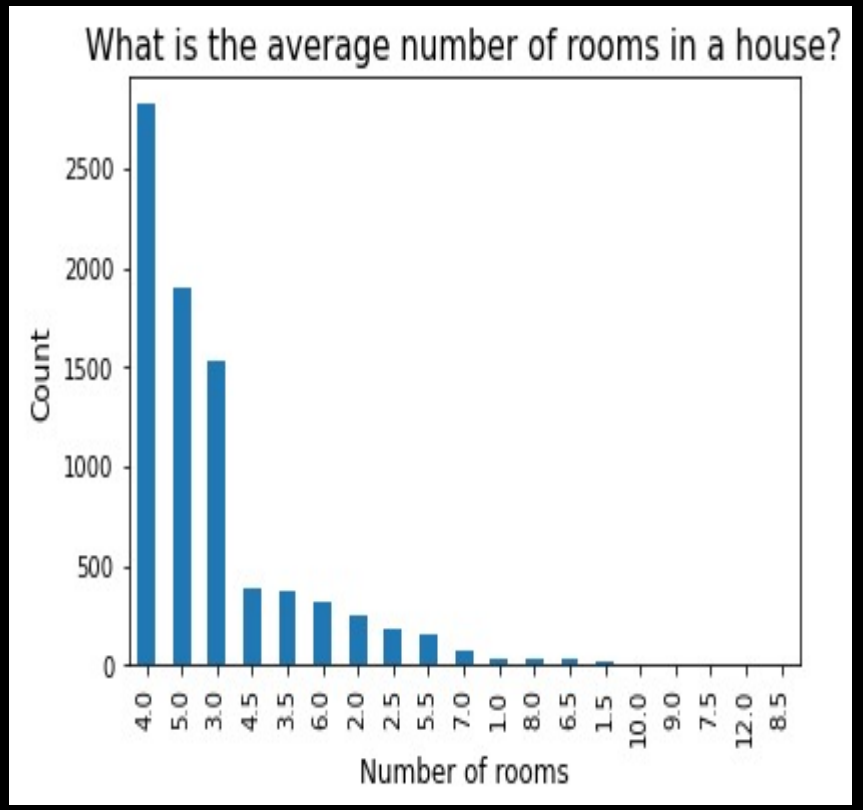
Correlation between rooms number and size:
The more rooms there are, the bigger the apartment.

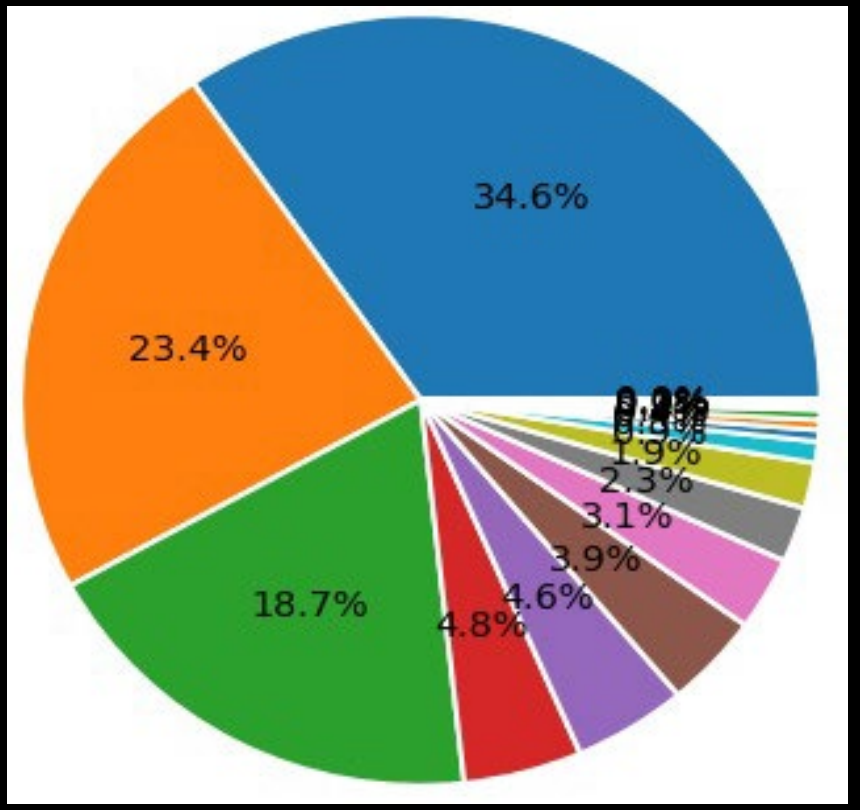


An average price for an apartment, penthouse and ground apartment(in millions):

Since the letters here are in Hebrew too, we used again bidialg.get_display() function from bidi library.

The average number of rooms in a house:
we visualized it with two different graphs: the first is a barplot graph of df['NumOfRooms'].value_counts() function, and the pieplot graph relies on the barplot graph. It can be useful for instance for a contractor who wants to know what is the most profitable number of rooms.
Match exists between the two graphs. For example, the left bar is the highest which matches the biggest part of the pieplot.





Modeling data

We used two different types of learning:

Unsupervised learning- random forest regressor.

Supervised learning- linear regression.

Unsupervised learning- random forest regressor

First, we tried to use random forest classifier model, but it didn't predict the result accurately:

```
Test accuracy: 0.0925
```

so we tried another model…

With the random forest regressor, we succeeded to predict accurately the result, with the parameters: Floor, rooms number, size.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestRegressor(n_estimators=100, random_state=42).fit(X, y)
print("Accuracy:", model.score(X_test, y_test))
```

```
Accuracy: 0.7442004179808843
```

Supervised learning-linear regression

First, we were encountered with inaccurate prediction. We tried to improve our model until eventually we got an accurate score.

```
X = df[['Floor', 'NumOfRooms', 'Size(SQM)']]
y = df['Price(₪)']

X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.25, random_state=50)
reg = LinearRegression().fit(X_train1, y_train1)
print("Accuracy:", model.score(X_test1, y_test1))
```

```
Accuracy: 0.7705894517259481
```

Recap and conclusions:

This was a challenging and a complex project, which made us understand the trend of change in the real estate field.

Collecting the data was tough, since it was a dynamic website. Every reloading, the website’s content had changed.

Finally, we attained our purpose, which was to predict the price of an apartment, with an average accuracy of 75.7%.