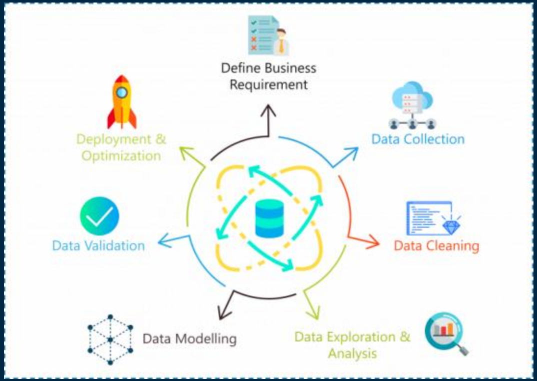
Rising apartment prices in Gush Dan

Oded Saban
Chen Yank
[Git Hub]

Research questions

1. Has there indeed been an increase in apartment prices as noticeable
2. Given the current price of an apartment, is it possible to predict its price in about 10 years



Data sources

- **Crawling** – [ad – Information on real estate transactions]

1. First, we used the ‘BeautifulSoup’ library to crawl the relevant data from ad site.
2. Second, we add 2 columns, the first is DealYear, and the second is DealMonth.
3. Finally, we saved the data in the allDf.csv file

מידע על עסקות נדלן

סינון

מיון תאריך - מהחדש לישן

2,281,064 מודעות - עמוד 1 מתוך 45,622

אזור

- ירושלים ומעלה אדומים 250,309
- באר שבע והסביבה 248,677
- אשדוד - אשקלון 242,384
- ראשל"צ והסביבה 125,200
- רמת גן - גבעתיים 122,835
- חיפה וחוף הכרמל 118,760
- נס ציונה - רחובות 113,504
- תל אביב 103,351
- גליל ועמקים 71,219
- נתניה והסביבה 67,598

בחירה מרובה

תאריך	ישוב	רחוב	חד'	שטח	קומה	מחיר	מחיר למר	בניה
28/1/2021	ירושלים	רמת מוצא 27	5	134	1	3,675,000	27,425	2000
28/1/2021	גבעתיים	המבוא 9	2	72	3	1,961,800	27,247	1960
28/1/2021	גבעתיים	המרי 26	2	72		1,961,800	27,247	1960
28/1/2021	ירושלים	רמת מוצא 27	5	134		3,675,000	27,425	2000
27/1/2021	בת ים	הרצל 39	2	41		1,380,000	33,658	1960
27/1/2021	אור עקיבא	אור עקיבא 3	5	120	8	1,665,000	13,875	2017
27/1/2021	חולון	עמק יזרעאל 17	4	80		1,855,000	23,187	1970

City	Street	Rooms	Surface	Floor	Price	PricePerSq	BuildYear	DealYear	DealMonth
ראשון לציון	NaN	5.0	110	8.0	2,035,000	18500	2011	2010.0	12.0
ראשון לציון	NaN	3.0	80	7.0	1,363,000	17037	2010	2010.0	12.0
ראשון לציון	בורג יוסף	5.0	100	5.0	1,460,000	14600	2010	2010.0	12.0
ראשון לציון	ראובן 11	5.0	133	8.0	1,970,000	14812	2010	2010.0	12.0
ראשון לציון	נהריים 16	4.0	100	18.0	1,223,300	12233	2011	2010.0	12.0
...	...	...	...	...	...	...	...	...	...
רחובות	NaN	4.0	110	NaN	1,900,000	17272	2012	2020.0	1.0
רחובות	NaN	4.0	105	NaN	1,840,000	17523	1992	2020.0	1.0
רחובות	הרימון 3	3.0	99	1.0	1,667,500	16843	1960	2020.0	1.0
רחובות	דרך בן ארי יצחק 2	5.0	130	9.0	2,220,000	17076	2014	2020.0	1.0
רחובות	NaN	5.0	125	3.0	4,004,614	32036	2018	2020.0	1.0

Data cleaning

1. NaN handling
2. Convert variables to uniform formats
3. Outliners handling

Data cleaning – NaN handling

- We removed the data that appears as NaN in the PricePerSq column, because with this column we will analyze the data and it will not be possible if we have NaN values there.

```
In [182]: df.isna().sum()

Out[182]: Unnamed: 0        0
          Date              0
          City              0
          Street         2541
          Rooms             2
          Surface           0
          Floor          4113
          Price             6
          PricePerSq       46
          BuildYear         0
          DealYear          0
          DealMonth         0
          dtype: int64
```

```
In [244]: df = df[df['PricePerSq'].notna()]

In [245]: df.isna().sum()

Out[245]: Unnamed: 0        0
          Date              0
          City              0
          Street         2532
          Rooms             2
          Surface           0
          Floor          4099
          Price             0
          PricePerSq        0
          BuildYear         0
          DealYear          0
          DealMonth         0
          dtype: int64
```

Data cleaning – Convert variables to uniform formats

- We converted the values of the columns: Price, PricePerSq, DealYear, DealMonth from float64 and Object to int32.
  This is so that we can handle the data optimally when all the variables are of the same type.

- It can be seen that during the conversion process of the Price and PricePerSq columns we deleted the character – , because an int32 variable could not contain the character – , .

```python
#handle columns types
df_basic['PricePerSq'] = df_basic['PricePerSq'].astype(int).str.replace(',', '').astype(int)
df_basic['DealYear'] = df_basic['DealYear'].astype(int)
df_basic['DealMonth'] = df_basic['DealMonth'].astype(int)
df_basic['Price'] = df_basic['Price'].astype(str).str.replace(',', '').astype(int)
```

```
df_basic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27382 entries, 0 to 27381
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   27382 non-null  int64
 1   Date         27382 non-null  object
 2   City         27382 non-null  object
 3   Street       21116 non-null  object
 4   Rooms        27375 non-null  float64
 5   Surface      27382 non-null  int64
 6   Floor        18205 non-null  float64
 7   Price        27359 non-null  object
 8   PricePerSq   27231 non-null  object
 9   BuildYear    27381 non-null  float64
 10  DealYear     27382 non-null  float64
 11  DealMonth    27382 non-null  float64
dtypes: float64(5), int64(2), object(5)
memory usage: 2.5+ MB
```

```
df_basic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27231 entries, 0 to 27381
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   27231 non-null  int64
 1   Date         27231 non-null  object
 2   City         27231 non-null  object
 3   Street       21009 non-null  object
 4   Rooms        27224 non-null  float64
 5   Surface      27231 non-null  int64
 6   Floor        18106 non-null  float64
 7   Price        27231 non-null  int32
 8   PricePerSq   27231 non-null  int32
 9   BuildYear    27230 non-null  float64
 10  DealYear     27231 non-null  int32
 11  DealMonth    27231 non-null  int32
dtypes: float64(3), int32(4), int64(2), object(3)
memory usage: 2.3+ MB
```

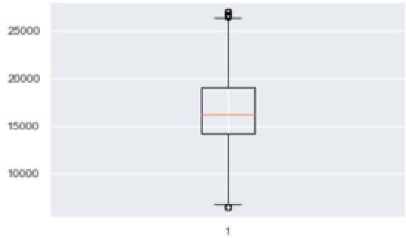
Data cleaning – Outliners

- We handled outliners with percentages and during the process we used the boxplot graph belonging to the matplotlib library to analyze the efficiency of the handled.

```python
def remove_outlier(df_in, col_name):
    q1 = df_in[col_name].quantile(0.25)
    q3 = df_in[col_name].quantile(0.75)
    iqr = q3-q1 #Interquartile range
    fence_low  = q1-1.5*iqr
    fence_high = q3+1.5*iqr
    df_out = df_in.loc[(df_in[col_name] > fence_low) & (df_in[col_name] < fence_high)]
    return df_out
```

```python
plt.boxplot(df_basic.loc[df_basic['City'] == 'holon']["PricePerSq"])
plt.show()
```


```python
df_test = df_clear.loc[df_clear['City'] == 'holon']["PricePerSq"]
plt.boxplot(df_test)
plt.show()
```



Exploring & visualizing the Data (EDA)

LinePlot

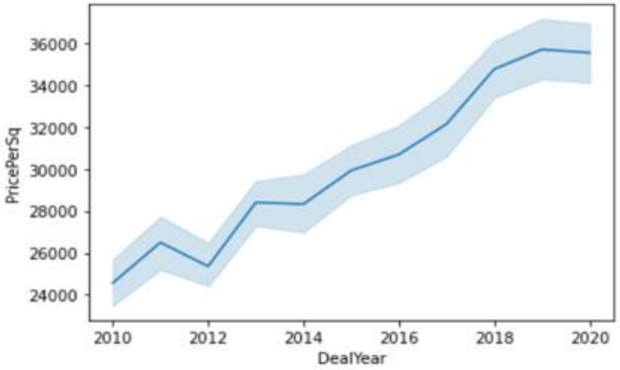
<u>What the graph contain:</u>

The graph contain 2 axes : 1. X-axis is the DealYear
2. Y-axis is the PricePerSq

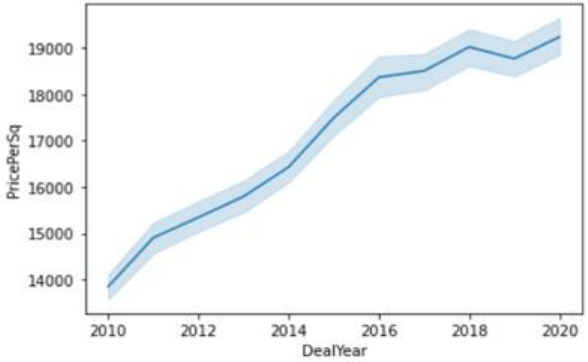
<u>Conclusion:</u>

With the help of the graph, we see that over the years there has been an increase in apartment prices in the various cities in Gush Dan.

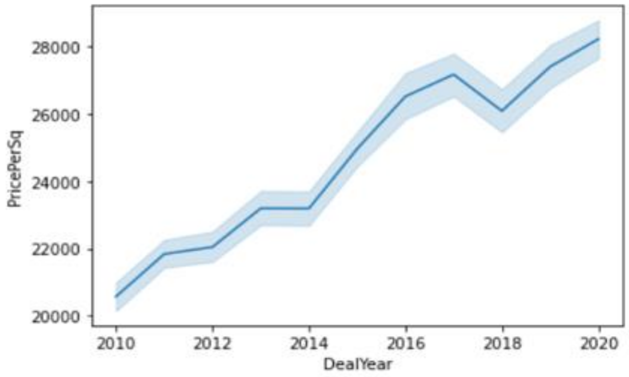
Tel-Aviv



Rishon-LeZion



Givatayim



Exploring & visualizing the Data (EDA)

BarPlot

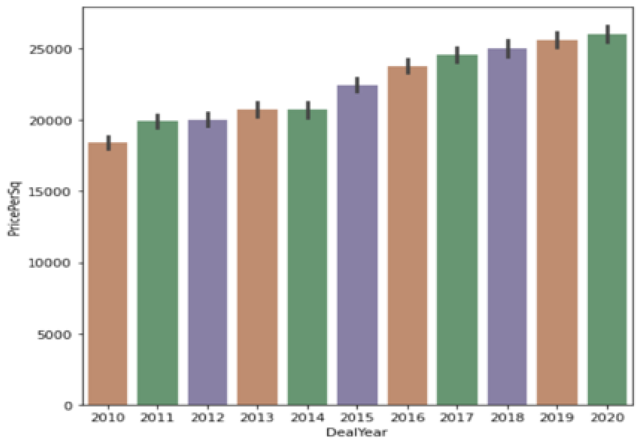
What the graph contain:

The graph contain 2 axes : 1. X-axis is the DealYear
2. Y-axis is the PricePerSq

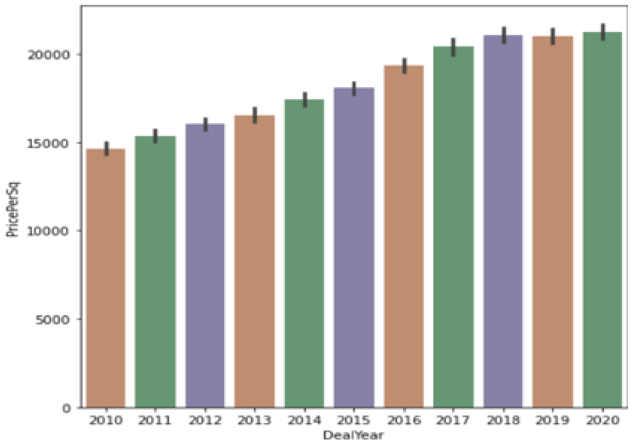
Conclusion:

With the help of the graph, we see that over the years there has been an increase in apartment prices in the various cities in Gush Dan.

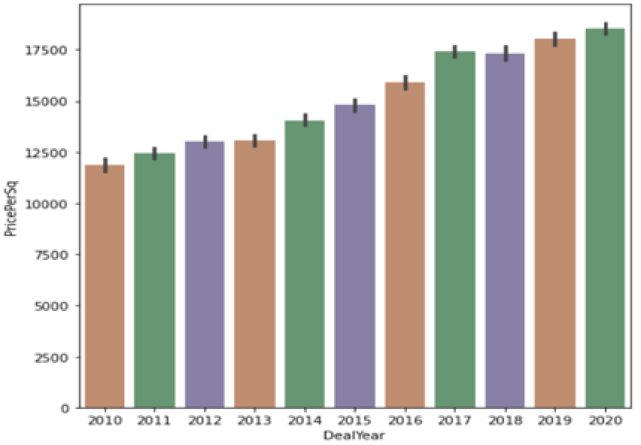
Ramat-Gan



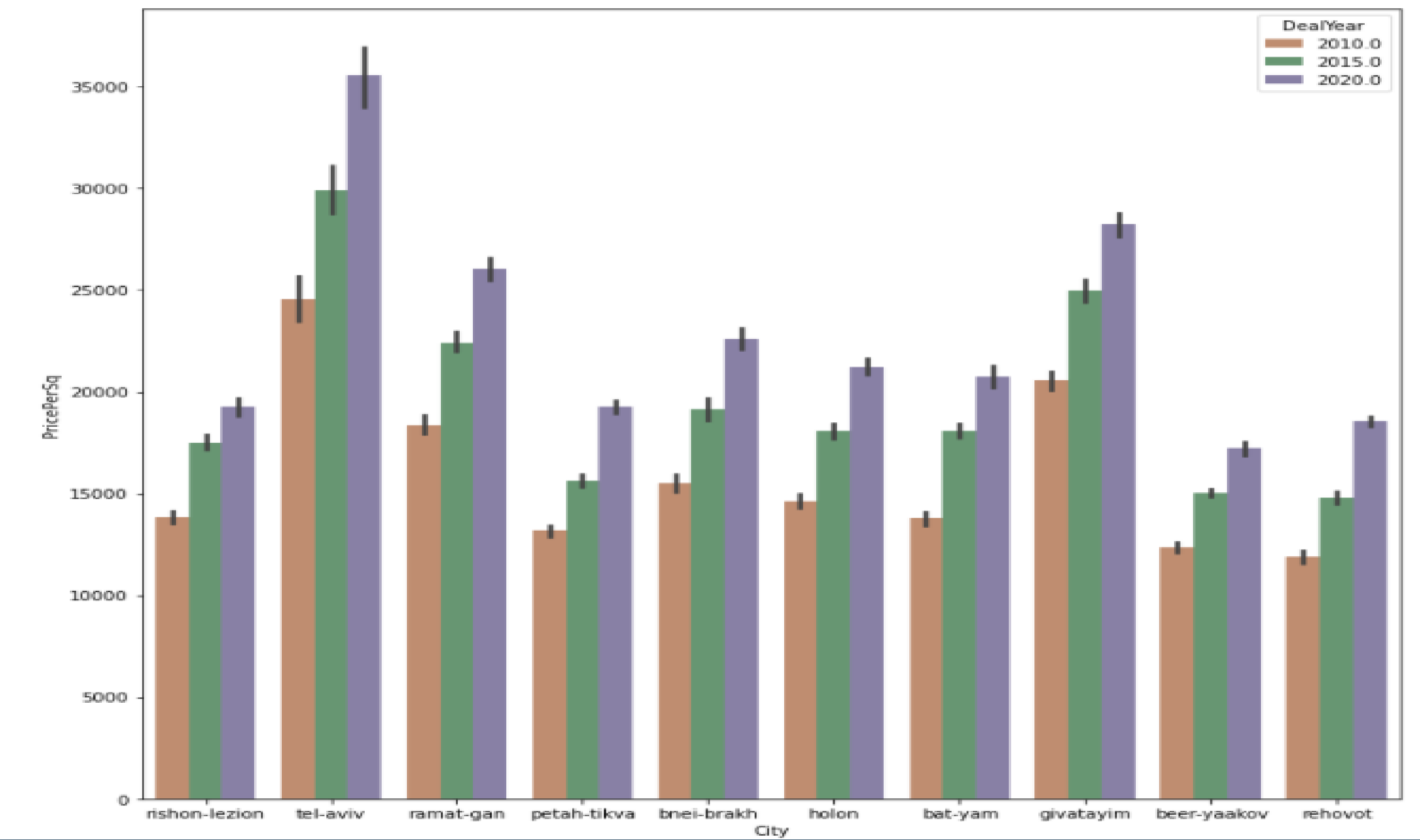
Holon



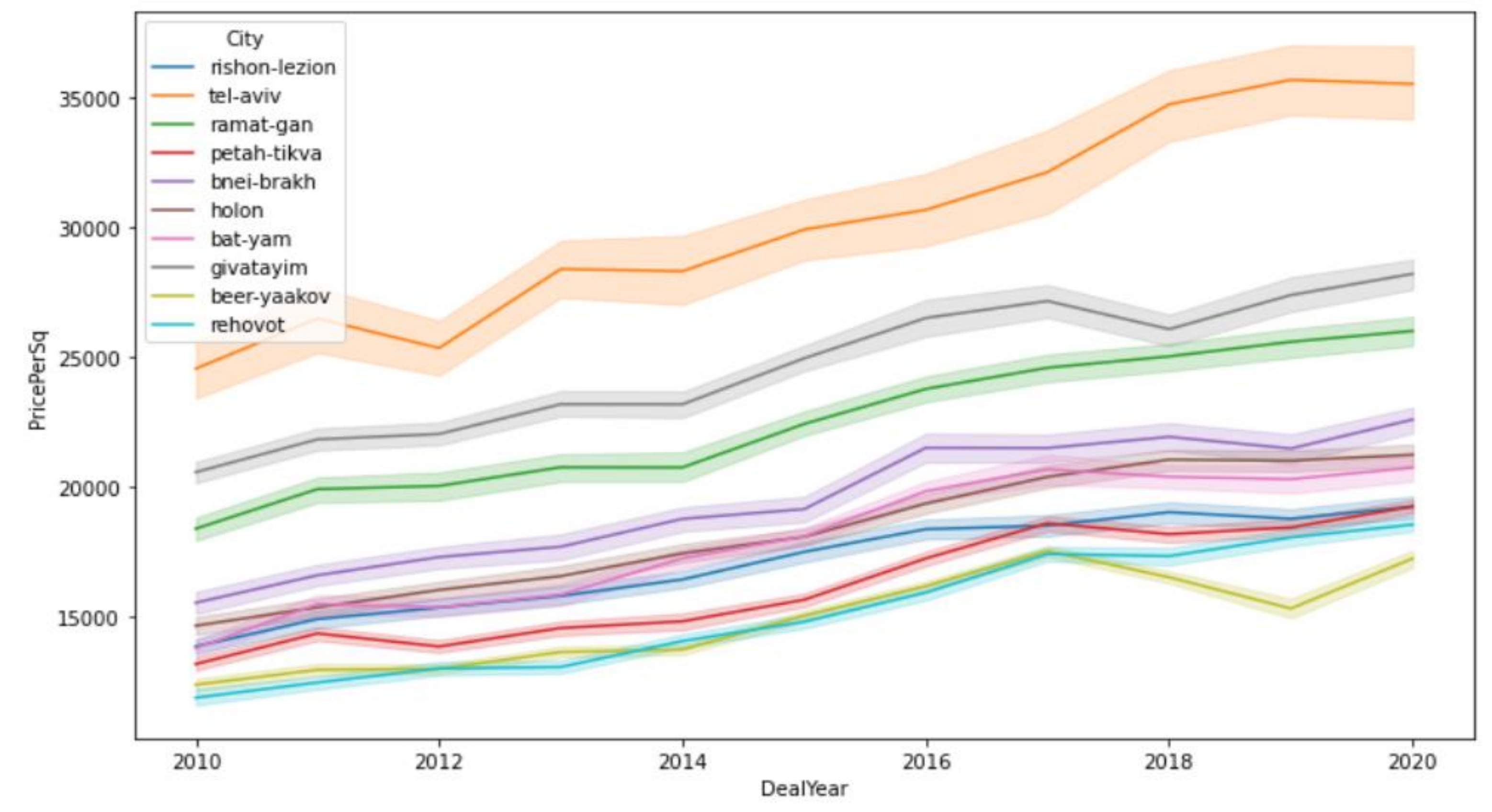
Rehovot



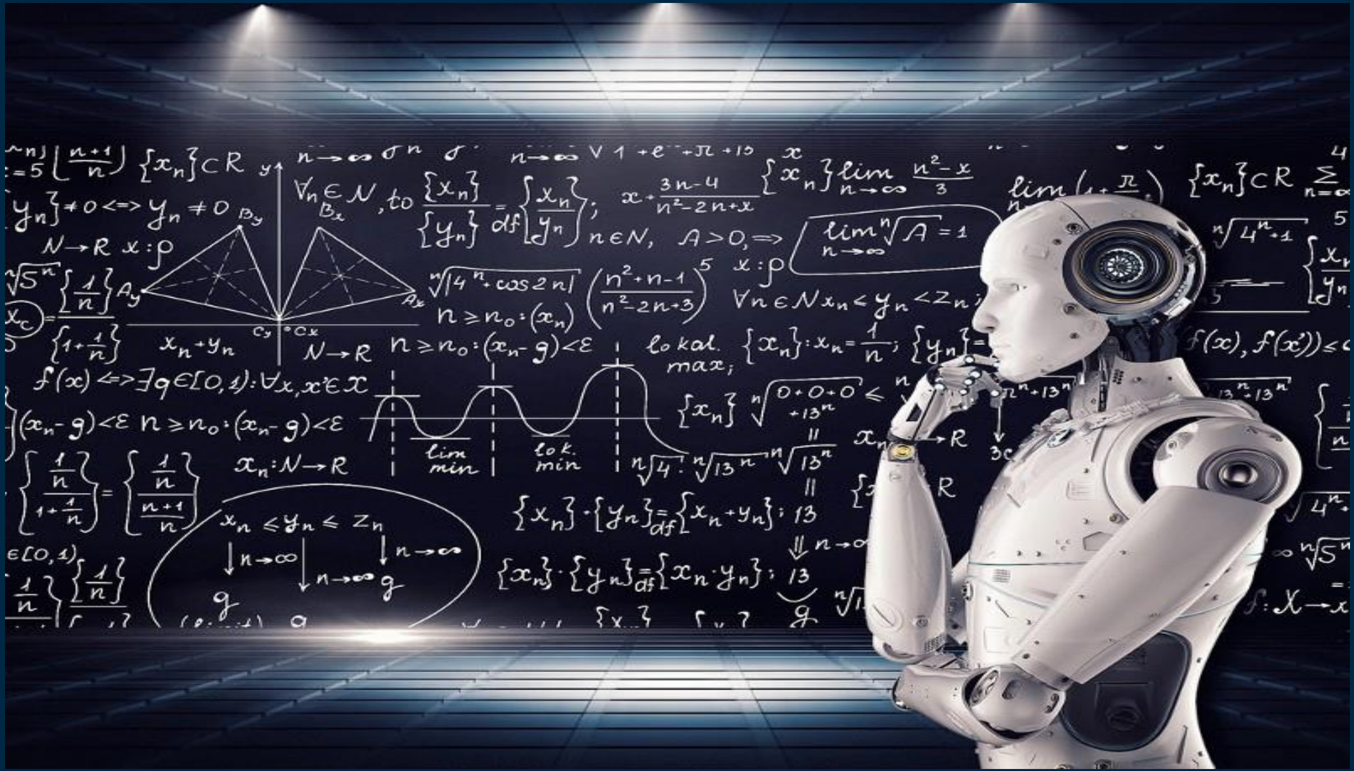
Exploring & visualizing the Data (EDA)



Exploring & visualizing the Data (EDA)



Machine Learning



Machine Learning – linear regression

- We used 'sklearn' library and linear regression.
- With their help we trained a model that can predict the average price per square meter in the city and in the year we chose.

```
x_train, x_test, y_train, y_test = train_test_split(df.DealYear, df.PricePerSq)
```

```
plt.scatter(x_train, y_train, label='Training data', color='r', alpha=.7)
plt.scatter(x_test, y_test, label='Testing data', color='g', alpha=.7)
plt.legend()
plt.title('Test Train Split')
plt.show()
```

```
LR = LinearRegression()
LR.fit(x_train.values.reshape(-1,1),y_train.values)
```

```
LinearRegression()
```

Machine Learning – linear regression

- The program asks the user to enter the year for prediction and prints the model predicted price for that year.

```
year = float(input("Please enter a year: "))
prediction_price = LR.predict(np.array([[year]]))[0]
print("the prediction price in " + choosen_city + " is " + str(prediction_price) + " per square meter")
```

```
Please enter a year: 2030
the prediction price in rehovot is 25698.743137025507 per square meter
```

Conclusions

- In conclusion, according to the findings of the project, we understand that there has indeed been an increase in the prices of apartments in Gush Dan, and moreover, with the help of machine learning, we conclude that in the coming years prices will continue to rise.