

2/6/2024

Eindwerk

Huizenprijzen



Mathias Vandeputte

Inhoudstafel

| | |
|---|----|
| Onderzoeksvraag | 3 |
| Hypothese | 3 |
| Dataverzameling | 4 |
| Dataset 1: California Housing prices. | 5 |
| Data opschonen | 5 |
| Data analyse | 8 |
| Factor 1: Woningouderdom | 9 |
| Factor 2: Aantal kamers..... | 9 |
| Factor 3: Aantal slaapkamers | 10 |
| Factor 4: Bevolkingsdichtheid | 11 |
| Factor 5: Aantal huishoudens | 11 |
| Factor 6: Inkomen | 12 |
| Factor 7: Nabijheid oceaan..... | 13 |
| Conclusie | 13 |
| Exporteren | 14 |
| Data visualisatie | 14 |
| Stap 1: Schematiseren | 14 |
| Stap 2: Modelleren in Power BI..... | 15 |
| Stap 3: Visuals maken | 17 |
| Dataset 2: House sales in king county, USA..... | 19 |
| Data opschonen | 19 |
| Data analyse | 24 |
| Factor 1: De datum..... | 24 |
| Factor 2: Aantal slaapkamers | 25 |
| Factor 3: Aantal badkamers | 26 |
| Factor 4: Aantal verdiepingen..... | 26 |
| Factor 5: Oppervlakte | 27 |
| Factor 6: Aanwezigheid waterkant..... | 27 |
| Factor 7: Uitzicht | 28 |
| Factor 8: Huis conditie | 28 |
| Factor 9: Bouwjaar | 29 |
| Exporteren | 30 |

| | |
|--|----|
| Conclusie..... | 30 |
| Data visualisatie | 30 |
| Stap 1: Schematiseren | 30 |
| Stap 2: Modelleren in Power BI..... | 31 |
| Stap 3: Visuals maken | 34 |
| Dataset 3: Housing prices dataset | 36 |
| Data opschonen | 36 |
| Data analyse | 38 |
| Factor 1: Oppervlakte..... | 38 |
| Factor 2: Aantal slaapkamers | 39 |
| Factor 3: Aantal badkamers | 40 |
| Factor 4: Aantal verdiepingen..... | 40 |
| Factor 5: Aanwezigheid hoofdweg..... | 41 |
| Factor 6: Aanwezigheid gastenkamer..... | 41 |
| Factor 7: Aanwezigheid kelder | 42 |
| Factor 8: Aanwezigheid warmwaterverwarming | 42 |
| Factor 9: aanwezigheid airco | 43 |
| Factor 10: Aantal parkeerplaasten | 43 |
| Factor 11: Voorkeursbuurt..... | 44 |
| Factor 12: Bemeubeling..... | 44 |
| Exporteren | 45 |
| Data visualisatie | 45 |
| Stap 1: Schematiseren | 45 |
| Stap 2: Modelleren in Power BI..... | 46 |
| Stap 3: Visuals maken | 46 |
| Besluit | 49 |
| Bron..... | 49 |

Onderzoeksraag

Ik heb niet lang moeten nadenken over welk onderwerp ik mijn eindwerk wou maken. Al snel was ik overtuigd dat ik mijn eindwerk wou maken in thema van de huizenprijzen.

Ik ben zelf van plan om binnen een paar jaar een huis te kopen dus was ik wel nieuwsgierig naar de verschillende factoren die invloed hebben op de valutatie van huizen. Dat is dus ook mijn onderzoeksraag.

“Welke factoren hebben er allemaal invloed op de prijs van huizen?”

Hypothese

Ik heb even gebrainstormd over mijn onderzoeksraag. Ik heb mij afgevraagd welke factoren de grootste impact zouden hebben op de huizenprijs. Na even nadenken ben ik ervan overtuigd dat de factoren die de grootste impact hebben op huizenprijzen de volgende zijn:

De locatie (verstedelijgingsgraad, binnenland/aan zee, werkgelegenheid)

Ik ga ervan uit dat huizen in de mooie groene buurt gemiddeld duurder zijn. Ik denk ook dat in regio's waar het inkomen gemiddeld hoger is de huizenprijzen dat ook zijn. Ik vermoed dat in regio's met meer werkgelegenheid de huizenprijzen hoger zijn. Tenslotte denk ik dat huizen aan de kust gelegen duurder zijn dan in het binnenland omdat velen daar een buitenverblijf willen. Ik verwacht dat huizen nabij een autosnelweg ook een hoger prijskaartje hebben.

De grootte van het huis

Hoe ruimer het huis, hoe meer bouwmateriaal gekocht moeten worden voor de constructie. Hoe meer werkuren er moeten betaald worden. Hoe groter de bouwgrond moet zijn.

De toestand van het huis

Huizen in slechte toestand moeten nog gerenoveerd worden wat ook veel geld kost. Hierdoor blijft er minder geld over voor de aankoop van het huis zelf.

De tijd

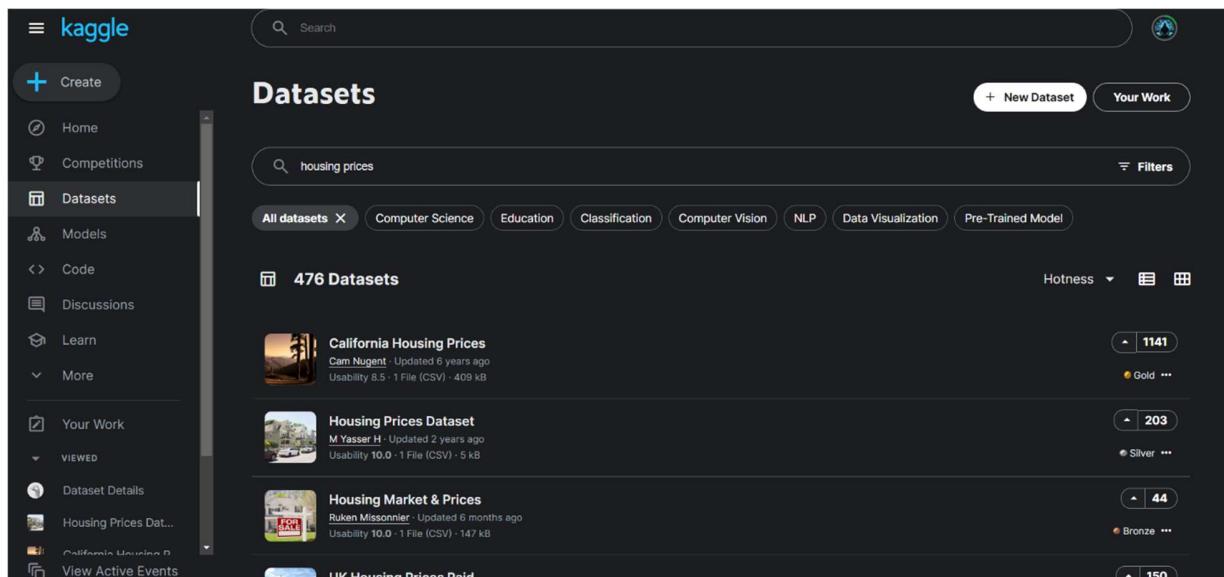
De huizenmarkt schommelt doorheen de tijd. De huizenprijzen zijn in de laatste honderd jaar astronomisch gestegen. Hoewel dat niet in een rechte lijn ging. Er zijn altijd periodes wanneer de huizenmarkt een duik neemt (2008 bijvoorbeeld). Dat brengt mij bij het laatste punt.

De economische factoren

Verschillende economische factoren kunnen een rol spelen. De werkloosheidscijfers, het consumentenvertrouwen, inflatie, rentevoet...

Dataverzameling

Ik ging dus een kijkje nemen wat Kaggle allemaal te bieden had op vlak van huizenprijzen datasets. De eerste dataset die mij aansprak omwille van zijn populariteit en hoge Kaggle bruikbaarheid score was de California housing prices dataset.



Die heb ik dus gedownload.

Dataset 1: California Housing prices.

Data opschonen

Ik ben begonnen met de eerste dataset “California housing prices” [1] in te laden in Jupyter Lab. Vervolgens heb ik de nodige Python bibliotheken geïmporteerd. Dat zijn in dit geval Pandas en Numpy.

```
import pandas as pd  
import numpy as np
```

Daarna heb ik het csv bestand in een dataframe geladen.

```
df = pd.read_csv(r"C:\Users\mvdp1\OneDrive\Documenten\GitHub\seaborn-data\housing.csv")
```

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value | ocean_proximity |
|-------|-----------|----------|--------------------|-------------|----------------|------------|------------|---------------|--------------------|-----------------|
| 0 | -122.23 | 37.88 | 41.0 | 880.0 | 129.0 | 322.0 | 126.0 | 8.3252 | 452600.0 | NEAR BAY |
| 1 | -122.22 | 37.86 | 21.0 | 7099.0 | 1106.0 | 2401.0 | 1138.0 | 8.3014 | 358500.0 | NEAR BAY |
| 2 | -122.24 | 37.85 | 52.0 | 1467.0 | 190.0 | 496.0 | 177.0 | 7.2574 | 352100.0 | NEAR BAY |
| 3 | -122.25 | 37.85 | 52.0 | 1274.0 | 235.0 | 558.0 | 219.0 | 5.6431 | 341300.0 | NEAR BAY |
| 4 | -122.25 | 37.85 | 52.0 | 1627.0 | 280.0 | 565.0 | 259.0 | 3.8462 | 342200.0 | NEAR BAY |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20635 | -121.09 | 39.48 | 25.0 | 1665.0 | 374.0 | 845.0 | 330.0 | 1.5603 | 78100.0 | INLAND |
| 20636 | -121.21 | 39.49 | 18.0 | 697.0 | 150.0 | 356.0 | 114.0 | 2.5568 | 77100.0 | INLAND |
| 20637 | -121.22 | 39.43 | 17.0 | 2254.0 | 485.0 | 1007.0 | 433.0 | 1.7000 | 92300.0 | INLAND |
| 20638 | -121.32 | 39.43 | 18.0 | 1860.0 | 409.0 | 741.0 | 349.0 | 1.8672 | 84700.0 | INLAND |
| 20639 | -121.24 | 39.37 | 16.0 | 2785.0 | 616.0 | 1387.0 | 530.0 | 2.3886 | 89400.0 | INLAND |

20640 rows × 10 columns

Ik heb vervolgens de dataframe informatie geraadpleegd.

```

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   longitude         20640 non-null   float64
 1   latitude          20640 non-null   float64
 2   housing_median_age 20640 non-null   float64
 3   total_rooms        20640 non-null   float64
 4   total_bedrooms     20433 non-null   float64
 5   population         20640 non-null   float64
 6   households         20640 non-null   float64
 7   median_income      20640 non-null   float64
 8   median_house_value 20640 non-null   float64
 9   ocean_proximity    20640 non-null   object  
dtypes: float64(9), object(1)
memory usage: 1.6+ MB

```

Zo ben ik er achter gekomen dat de “total_bedrooms” kolom 207 NaN waarden bevatte. Ik heb dus die rijen met NaN waarden verwijderd.

```

df.dropna(inplace=True)

df.shape[0]

20433

```

Ik heb voor de zekerheid de shape functie gebruikt om te kijken of de rijen effectief verwijderd waren en dat bleek gelukt te zijn. Vervolgens heb ik de tot dan nog naamloze index een naam gegeven. Ik heb hem “Huis_Prijs_ID” genoemd.

```

df = df.rename_axis('Huis_Prijs_ID')

```

Ik heb dan de twee kolommen verwijderd die ik niet van plan was om te analyseren. Dat waren de lengte-en breedtegraad kolom.

```

df.drop(columns= ['longitude', 'latitude'], inplace=True)

```

Een te groot aantal unieke waarden in de meeste kolommen maakte het onmogelijk om op die kolommen te groeperen. Om die reden heb ik besloten om elke kolom (met een te groot aantal unieke waarden) waar ik op wou groeperen in 5 kwantielën op te splitsen. Ik heb hiervoor een for-loop gebruikt in combinatie met de qcut functie.

```

columns_to_exclude = ['median_house_value', 'ocean_proximity']

for col in df.columns:
    if col not in columns_to_exclude:
        df[col] = pd.qcut(df[col], q=5, labels=[0, 0.25, 0.50, 0.75, 1])

```

De kolommen “median_house_value” en “ocean proximity” moesten niet in categorieën geplaatst worden omdat ik de mediaan huis waarde wilde gebruiken om te aggregeren in plaats van te groeperen. De Oceaan nabijheid was al in categorieën verdeeld.

| Huis_Prijs_ID | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value | ocean_proximity |
|---------------|--------------------|-------------|----------------|------------|------------|---------------|--------------------|-----------------|
| 0 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 452600.0 | NEAR BAY |
| 1 | 0.25 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 358500.0 | NEAR BAY |
| 2 | 1.00 | 0.25 | 0.00 | 0.00 | 0.00 | 1.00 | 352100.0 | NEAR BAY |
| 3 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 341300.0 | NEAR BAY |
| 4 | 1.00 | 0.25 | 0.25 | 0.00 | 0.25 | 0.50 | 342200.0 | NEAR BAY |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20635 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | 78100.0 | INLAND |
| 20636 | 0.25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.25 | 77100.0 | INLAND |
| 20637 | 0.00 | 0.50 | 0.50 | 0.25 | 0.50 | 0.00 | 92300.0 | INLAND |
| 20638 | 0.25 | 0.50 | 0.50 | 0.25 | 0.25 | 0.00 | 84700.0 | INLAND |
| 20639 | 0.00 | 0.75 | 0.75 | 0.75 | 0.75 | 0.25 | 89400.0 | INLAND |

20433 rows × 8 columns

Door de qcut functie te gebruiken worden die kolommen automatisch van het float naar het categorie datatype gecast.

| df.dtypes.to_frame() | 0 |
|----------------------|----------|
| housing_median_age | category |
| total_rooms | category |
| total_bedrooms | category |
| population | category |
| households | category |
| median_income | category |
| median_house_value | float64 |
| ocean_proximity | object |

Om de getallen in de mediaan huiswaarde kolom af te ronden heb ik die kolom van float naar integer gecast.

```
df['median_house_value']= df['median_house_value'].astype(int)
```

Om het data cleanen af te ronden heb ik nog eerst de kolom volgorde aangepast en de kolomnamen en waarden vertaald naar het Nederlands.

```

columns = df.columns.tolist()

columns[7], columns[6] = columns[6], columns[7]

df = df[columns]

df.rename(columns={
    'housing_median_age': 'Huis_Ouderdom', 'total_rooms': 'Aantal_Kamers',
    'total_bedrooms': 'Aantal_Slaapkamers', 'population': 'Bevolkingsdichtheid',
    'households': 'Huishoudens', 'median_income': 'Mediaan_Inkommen', 'ocean_proximity': 'Nabijheid_Oceaan',
    'median_house_value': 'Mediaan_Huis_Prijs', 'MHP_category': 'MHP_categorie'
}, inplace=True)

replace_vlaams = {'NEAR BAY': 'NABIJ BAAI',
                  '<1H OCEAN': '<1U OCEAAN',
                  'INLAND': 'BINNENLAND',
                  'NEAR OCEAN': 'NABIJ OCEAAN',
                  'ISLAND': 'EILAND'}

df['Nabijheid_Oceaan'] = df['Nabijheid_Oceaan'].replace(replace_vlaams)

```

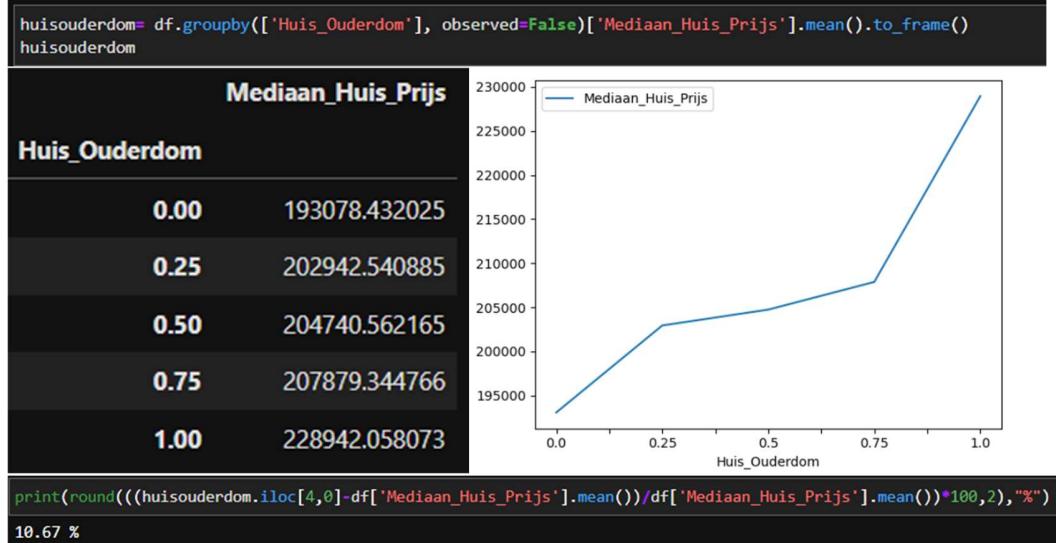
Nadat de data was opgeschoond zag mijn dataframe er zo uit:

| Huis_Prijs_ID | Huis_Ouderdom | Aantal_Kamers | Aantal_Slaapkamers | Bevolkingsdichtheid | Huishoudens | Mediaan_Inkommen | Nabijheid_Oceaan | Mediaan_Huis_Prijs |
|---------------|---------------|---------------|--------------------|---------------------|-------------|------------------|------------------|--------------------|
| 0 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | NABIJ BAAI | 452600 |
| 1 | 0.25 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | NABIJ BAAI | 358500 |
| 2 | 1.00 | 0.25 | 0.00 | 0.00 | 0.00 | 1.00 | NABIJ BAAI | 352100 |
| 3 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | NABIJ BAAI | 341300 |
| 4 | 1.00 | 0.25 | 0.25 | 0.00 | 0.25 | 0.50 | NABIJ BAAI | 342200 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20635 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.00 | BINNENLAND | 78100 |
| 20636 | 0.25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.25 | BINNENLAND | 77100 |
| 20637 | 0.00 | 0.50 | 0.50 | 0.25 | 0.50 | 0.00 | BINNENLAND | 92300 |
| 20638 | 0.25 | 0.50 | 0.50 | 0.25 | 0.25 | 0.00 | BINNENLAND | 84700 |
| 20639 | 0.00 | 0.75 | 0.75 | 0.75 | 0.75 | 0.25 | BINNENLAND | 89400 |

Data analyse

Nadat ik de data opgeschoond had was het tijd om ze onder de loep te nemen. Ik heb nu zeven factoren in mijn kolom titels, die mogelijk de huizenprijs kunnen beïnvloeden. Ik zal individueel groeperen op deze kolommen om er achter te komen welke van deze factoren een significante rol speelt en welke niet. Ik ga ze ook al even snel visualiseren, hoewel ik dat in de volgende stap “data visualisatie” grondiger zal doen met Power BI.

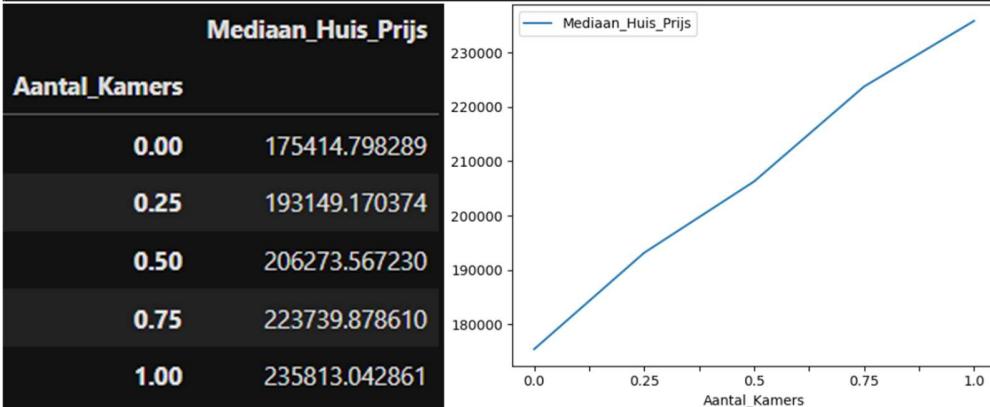
Factor 1: Woningouderdom



Het resultaat van deze analyse toont aan dat volgens deze dataset de huizenouderdom positief gecorreleerd is met de huizenprijs. De buurten met de hoogste huisouderdom waren 10% duurder dan het gemiddelde. Dat was niet in lijn met mijn verwachtingen. Een mogelijke verklaring hiervoor zou kunnen zijn dat die oudere huizen ook groter zijn.

Factor 2: Aantal kamers

```
kamers= df.groupby(['Aantal_Kamers'], observed=True)[['Mediaan_Huis_Prijs']].mean().to_frame()
kamers
```

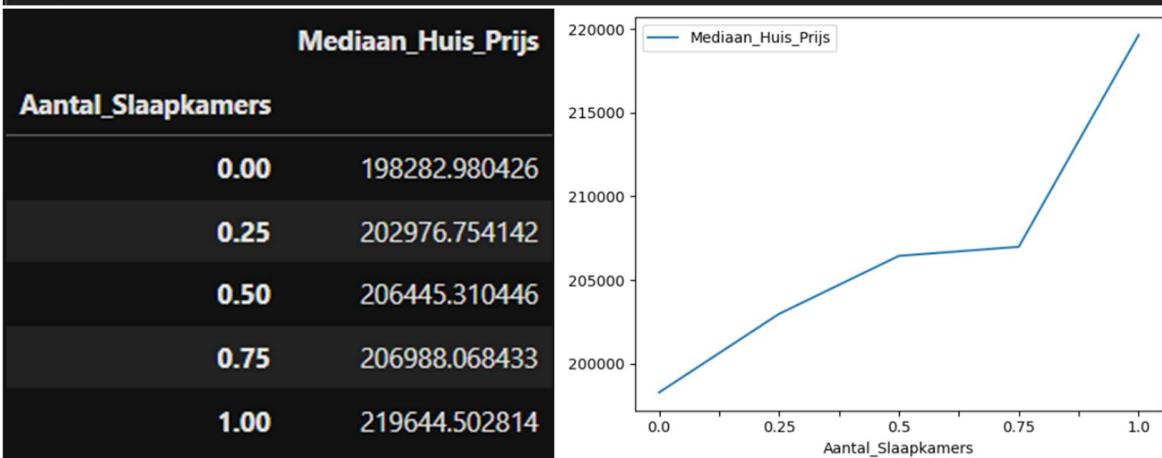


```
print(round(((kamers.iloc[4,0]-df['Mediaan_Huis_Prijs'].mean())/df['Mediaan_Huis_Prijs'].mean())*100,2),"%")
13.99 %
```

Het totaal aantal kamers heeft een positieve correlatie met de waarde van huizen. In deze dataset zijn de huizen met de meeste kamers 14% duurder dan de gemiddelde huisprijs. Dit is in lijn met mijn verwachtingen. Gemiddeld gezien hebben huizen met meer kamers een grotere oppervlakte. Dit zeggen dat er voor de constructie meer bouwmaterialen nodig zijn. Die bouwmaterialen kosten geld. Ook moeten er meer werkuren voor betaald worden.

Factor 3: Aantal slaapkamers

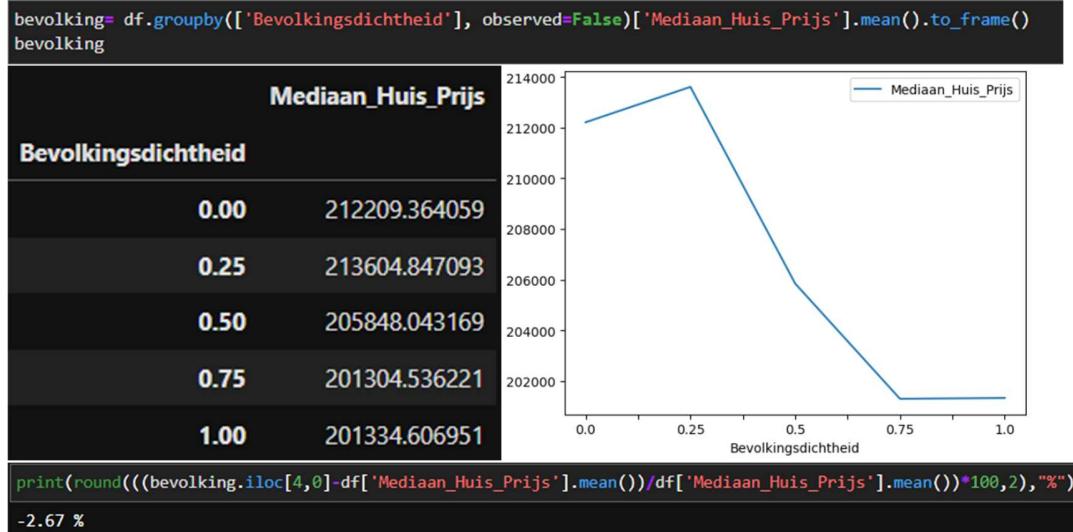
```
slaapkamers= df.groupby(['Aantal_Slaapkamers'], observed=True)[['Mediaan_Huis_Prijs']].mean().to_frame()
slaapkamers
```



```
print(round(((slaapkamers.iloc[4,0]-df['Mediaan_Huis_Prijs'].mean())/df['Mediaan_Huis_Prijs'].mean())*100,2),"%")
6.18 %
```

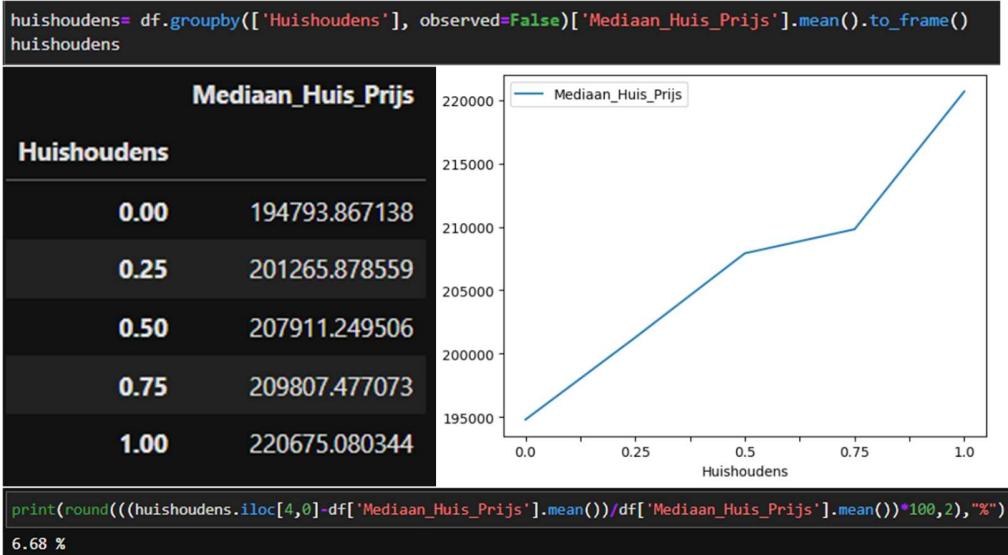
Volgens deze dataset is een hoger aantal slaapkamers gecorreleerd met hogere huizenprijzen, al is het verschil met amper 6% niet groot.

Factor 4: Bevolkingsdichtheid



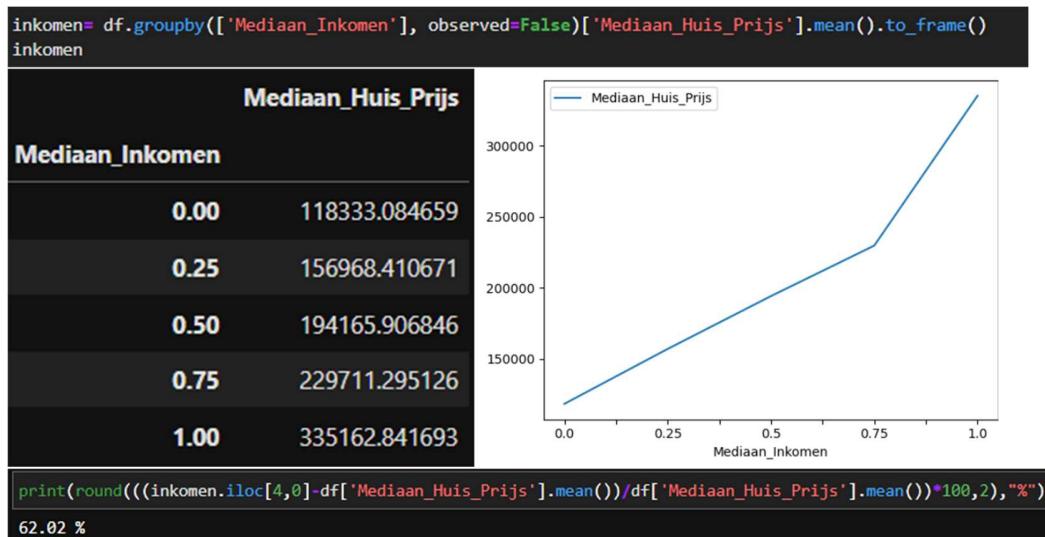
De bevolkingsdichtheid is volgens deze dataset geassocieerd met lagere huizenprijzen. De huizenprijzen zijn een kleine 3% goedkoper in de dichtbevolkte gebieden. Dit is niet significant. Andere factoren, zoals het feit dat in dunbevolkte gebieden tuinen vaak groter zijn En er vaak meer groen aanwezig is, kan ertoe bijdragen dat kopers ook vaker bereid zijn om er meer voor te betalen.

Factor 5: Aantal huishoudens



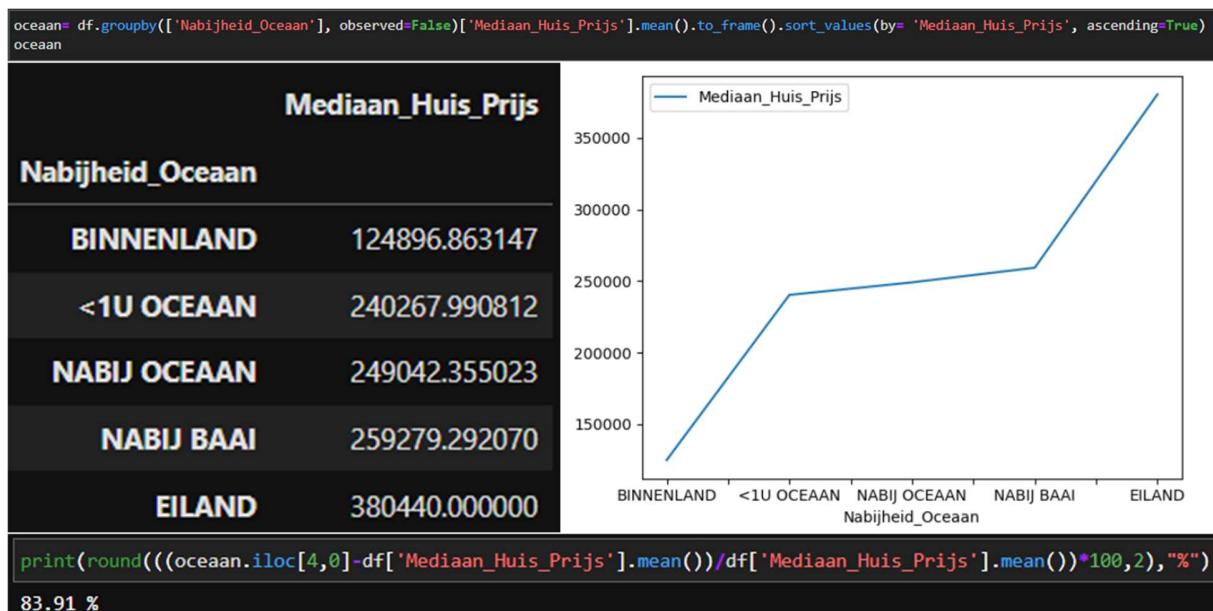
Ondanks het feit dat bevolkingsdichtheid niet positief gecorreleerd is met de huizenprijzen, toont de dataset een positief verband tussen het aantal huishoudens en de huizenprijzen: de gemiddelde huizenprijs in de buurten met het hoogst aantal huishoudens is een kleine 7% hoger dan gemiddeld. Beide vaststellingen samen genomen, kunnen we besluiten dat de huishoudens kleiner zijn in duurdere buurten. Tenminste als we deze dataset mogen geloven.

Factor 6: Inkomen



Van alle factoren tot nu toe, is dit veruit de meest significante factor. Buurten met de hoogste inkomens zijn gemiddeld 62% duurder volgens deze dataset. Dit is ook heel logisch, want hoe meer mensen verdienen, hoe meer ze bereid zijn om uit te geven om in de meest aantrekkelijke buurt te wonen. Bij hogere huizenprijzen zal de vraag niet snel dalen omdat meer vermogende buurten het zich kunnen permitteren. In buurten met een lager inkomen veronderstel ik dat dit vaak wel het geval is.

Factor 7: Nabijheid oceaan



Ook de nabijheid van de oceaan is een zeer significante factor. De prijzen van huizen die het dichtst aan de oceaan zijn gelegen, zijn veruit de duurste. Vooral huizen gelegen op eilanden zijn heel duur. Die huizen kosten gemiddeld 84% meer dan gemiddeld. Huizen gelegen op eilanden zijn gemiddeld maar liefst 3 keer zo duur als huizen in het binnenland.

Conclusie

Van alle factoren zijn er 2 die een heel duidelijke impact hebben op de huizenprijs. Dat zijn het mediaan inkomen per buurt en de nabijheid van de oceaan. Deze twee factoren kwamen aan bod in mijn hypothese en dat was dus in lijn met mijn verwachtingen.

Exporteren

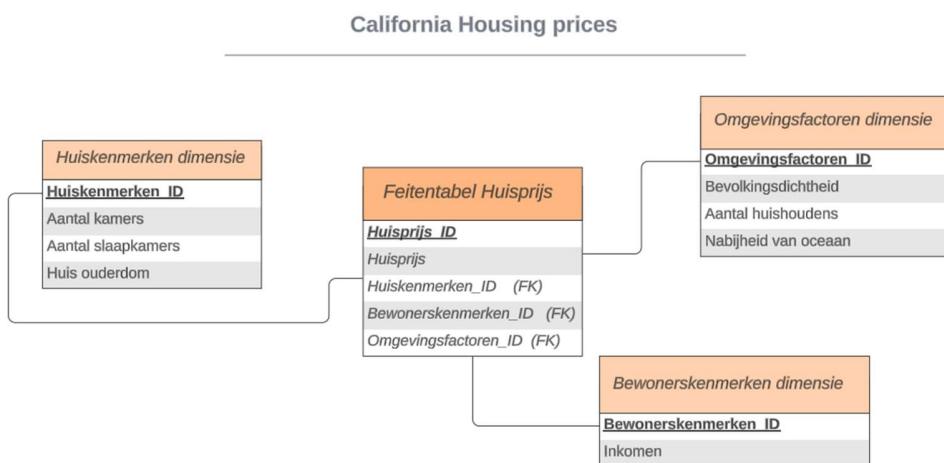
Ik heb mijn opgeschoonde dataframe tenslotte nog terug naar csv geëxporteerd zodat ik die in Power BI kon importeren om mooie visualisaties te maken.

```
df.to_csv('dataset_1.csv', index=True)
```

Data visualisatie

Stap 1: Schematiseren

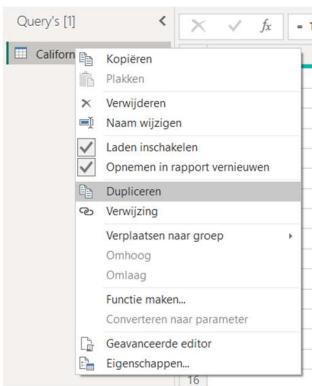
Voor ik kon beginnen te modelleren in Power BI, heb ik het sterschema eerst ontwerpen met Lucid Chart. Ik ben begonnen met een feitentabel te maken: ‘de ‘Feitentabel Huisprijs’. Ik heb de kolommen uit de tabel die bij elkaar hoorden elk toegewezen aan een dimensie. Als resultaat had ik een feitentabel die gelinkt was aan 3 dimensies. De ‘Huiskenmerkendimensie’, de ‘Omgevingsfactorendimensie’ en de ‘Bewonerskenmerken-dimensie’. Zowel de feitentabel als de dimensies hebben elk een primary key die ze uniek identificeert. De primary keys uit de dimensies zijn ook terug te vinden als foreign keys in de feitentabel.



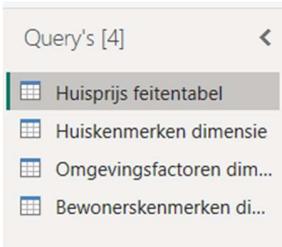
Stap 2: Modelleren in Power BI

Ik heb de opgeschoonde California housing prices csv file in Power BI geladen. Ik ben direct begonnen met mijn gegevens te transformeren in Power Query. De data had ik al opgeschoond met Pandas. Ik had dus alleen maar Power Query nodig om mijn tabel in een sterschema om te zetten.

Eerst en vooral ben ik begonnen met mijn tabel drie keer te dupliceren.



Vervolgens gaf ik elke query een andere naam. De eerste query noemde ik "Huisprijs feitentabel". De tweede "Huiskenmerken dimensie". De derde query "Omgevingsfactoren dimensie". Tenslotte noemde ik de laatste query "Bewonerskenmerken dimensie".



Ik heb dan voor elke dimensie de juiste kolommen toegevoegd en nieuwe index kolom aangemaakt met een surrogate key. Die surrogate key is nodig om al mijn rijen uniek te identificeren en dient dus als primary key.

Screenshot of Power BI Data Editor showing the 'Kolommen kiezen' (Select Columns) dialog. The dialog lists columns from a table named 'Table.ReplaceValue(#"Waarde vervangen2", 7)' and allows selecting which columns to keep. The columns listed are Huis_Prijs_ID, Huis_Ouderdom, Aantal, and Aantal_Slaapkamers. Other columns like Bevolkingsdichtheid, Huisoudhouders, etc., are shown but not selected.

Screenshot of Power BI Data Editor showing the result of the query. The table has been modified to include an index column ('Index') and a dimension key column ('Huiskenmerken_ID'). The data rows now correspond to the indices 1 through 28.

| | Huis_Prijs_ID | Huis_Ouderdom | Aantal | Aantal_Slaapkamers | Huiskenmerken_ID |
|----|---------------|---------------|--------|--------------------|------------------|
| 1 | 0 | 2 | 0 | 0 | 1 |
| 2 | 1 | 0,25 | 1 | 1 | 2 |
| 3 | 2 | 1 | 0,25 | 0 | 3 |
| 4 | 3 | 1 | 0 | 0 | 4 |
| 5 | 4 | 1 | 0,25 | 0,25 | 5 |
| 6 | 5 | 1 | 0 | 0 | 6 |
| 7 | 6 | 1 | 0,75 | 0,5 | 7 |
| 8 | 7 | 1 | 0,75 | 0,75 | 8 |
| 9 | 8 | 1 | 0,75 | 0,75 | 9 |
| 10 | 9 | 1 | 0,75 | 0,75 | 10 |
| 11 | 10 | 1 | 1 | 0,75 | 11 |
| 12 | 11 | 1 | 0,5 | 0,5 | 12 |
| 13 | 12 | 1 | 0,75 | 0,5 | 13 |
| 14 | 13 | 1 | 0 | 0 | 14 |
| 15 | 14 | 1 | 0,75 | 0,75 | 15 |
| 16 | 15 | 1 | 0 | 0,25 | 16 |
| 17 | 16 | 1 | 0,5 | 0,25 | 17 |
| 18 | 17 | 1 | 0 | 0,25 | 18 |
| 19 | 18 | 1 | 0,5 | 0,5 | 19 |
| 20 | 19 | 1 | 0,25 | 0,25 | 20 |
| 21 | 20 | 0,75 | 0 | 0 | 21 |
| 22 | 21 | 1 | 0,25 | 0,25 | 22 |
| 23 | 22 | 1 | 0,5 | 0,75 | 23 |
| 24 | 23 | 1 | 0,25 | 0,25 | 24 |
| 25 | 24 | 1 | 0,5 | 0,5 | 25 |
| 26 | 25 | 1 | 0 | 0 | 26 |
| 27 | 26 | 1 | 0 | 0 | 27 |
| 28 | 27 | 1 | 0,5 | 0,5 | 28 |

Eens elke dimensie zijn eigen unieke sleutel had heb ik mijn feitentabel samengevoegd aan elke dimensie via de huis_prijs_ID kolom.

Samenvoegen

Selecteer een tabel en overeenkomende kolommen om een samengevoegde tabel te maken.

Huisprijs feitentabel

| Huis_Prijs_ID | Huis_Ouderdom | Aantal_Kamers | Aantal_Slaapkamers | Bevolkingsdichtheid | Huishoudens |
|---------------|---------------|---------------|--------------------|---------------------|-------------|
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0,25 | 1 | 1 | 1 | 1 |
| 2 | 1 | 0,25 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0,25 | 0,25 | - | - |

Huiskenmerken dimensie

| Huis_Prijs_ID | Huis_Ouderdom | Aantal_Kamers | Aantal_Slaapkamers | Huiskenmerken_ID |
|---------------|---------------|---------------|--------------------|------------------|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0,25 | 1 | 1 | 2 |
| 2 | 1 | 0,25 | 0 | 3 |
| 3 | 1 | 0 | 0 | 4 |
| 4 | 1 | 0,25 | 0,25 | 5 |

Type join

Left outer (alle uit de eerste, overeenkomende uit de...)

Gebruik fuzzy overeenkomst om de samenvoeging uit te voeren

Opties voor fuzzy overeenkomsten

✓ De selectie komt overeen met 2043 van 2043 rijen in de eerste tabel.

OK **Annuleren**

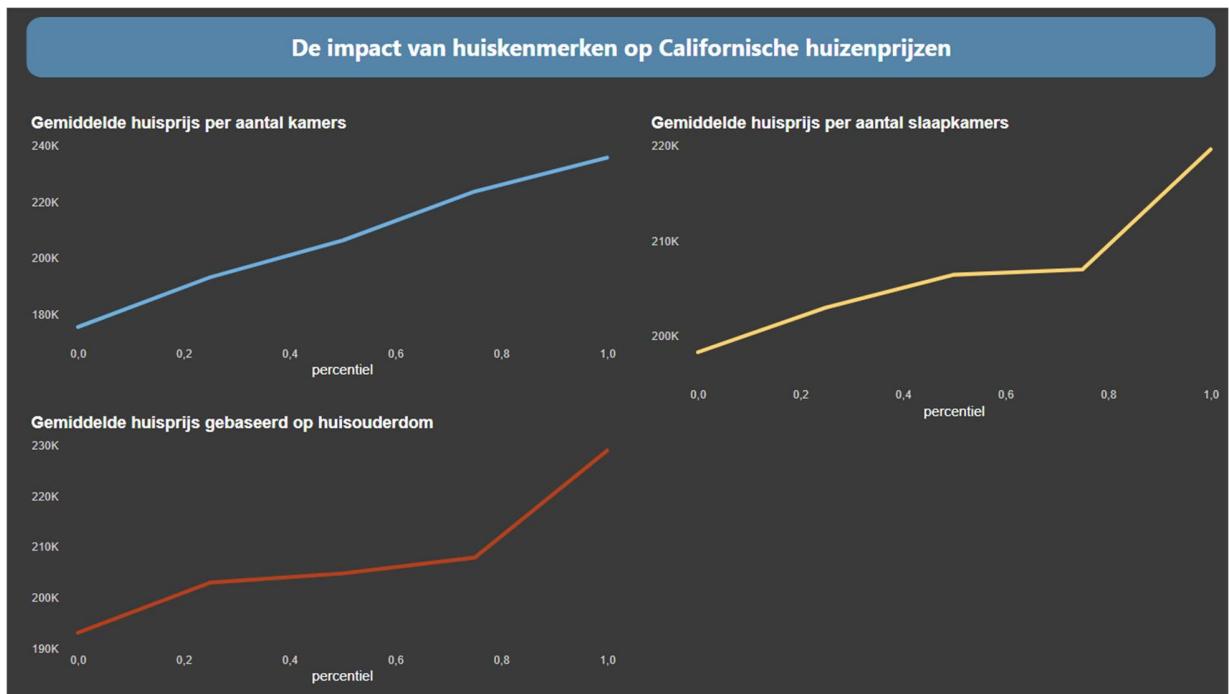
Ik heb de indexkolom van de samengevoegde tabellen uitgevouwd en daarna kon ik alle kolommen verwijderen die bij de andere dimensies horen.

Dat was het dan voor het modelleren. Ik heb power query gesloten en was nu klaar voor de volgende stap.

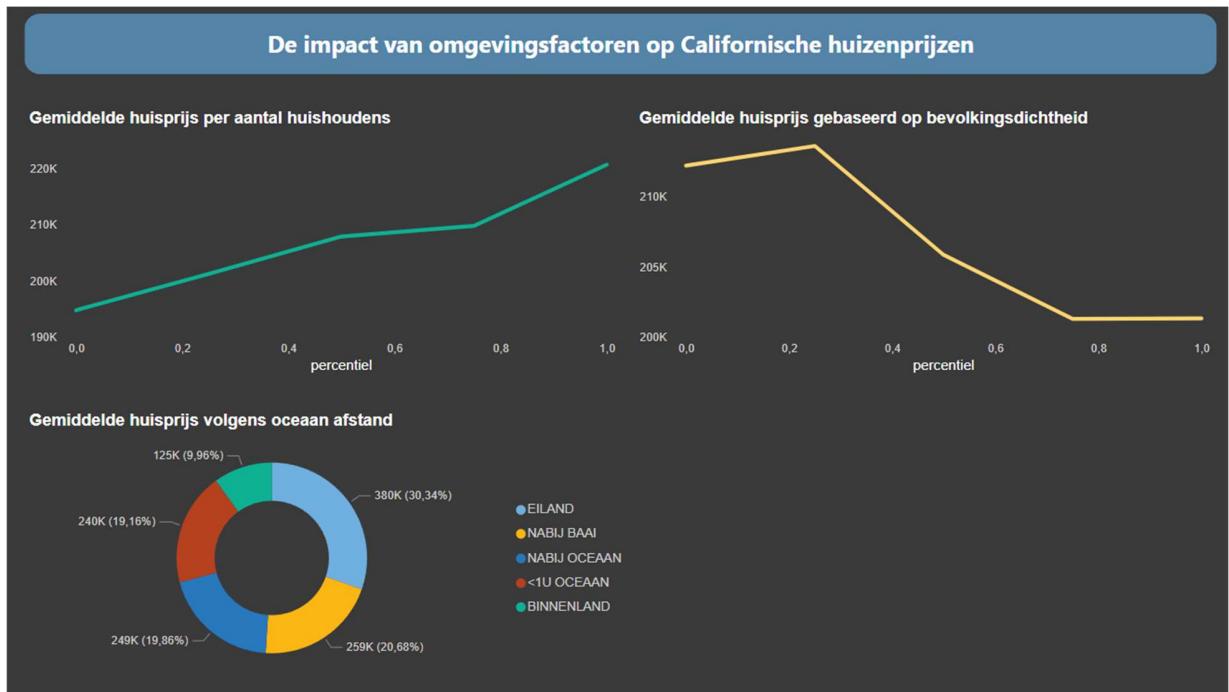
Stap 3: Visuals maken

Ik heb dan voor elke dimensie een eigen dashboard gemaakt

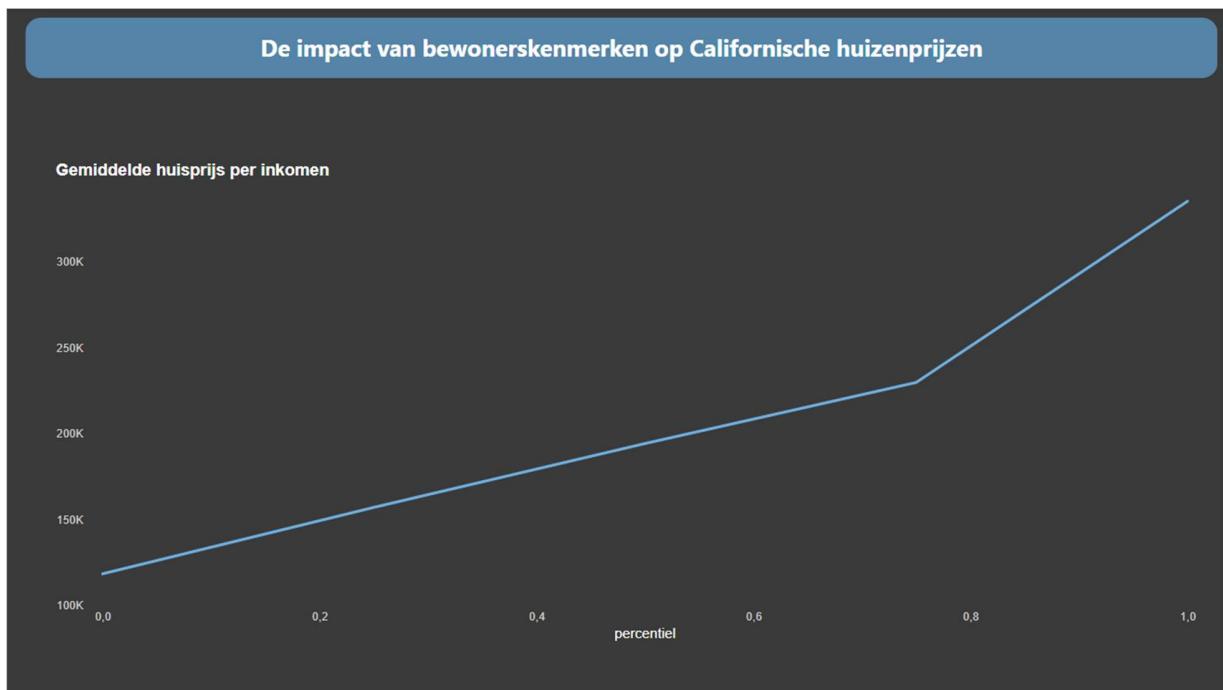
Huiskenmerken dimensie



Omgevingsfactoren dimensie



Bewonerskenmerken dimensie



Dataset 2: House sales in king county, USA

Op Kaggle heb ik nog een interessante dataset [2] gevonden in verband met huizenprijzen. Deze dataset bevat een lijst met huis verkopen in King County in de Verenigde Staten. Ik heb de file dus gedownload. Vervolgens heb ik een nieuw notebook geopend en ben ik weer begonnen met de nodige bibliotheken te importeren.

```
import pandas as pd  
import numpy as np
```

Vervolgens heb ik mijn Kaggle dataset file in een dataframe geladen.

```
df= pd.read_csv(r"C:\Users\mvdp1\OneDrive\Documenten\sample_data\hous_price.csv")
```

Data opschonen

Als ik mijn dataframe oproep zag ik meteen dat er nog heel wat opschoonwerk was.

| | id | date | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | ... | grade | sqft_above | sqft_basement | yr_built | yr_renovated | zipcode | lat | long |
|-------|------------|-------------------|--------|---------------|-----------|-------------|----------|--------|------------|------|-----|-------|------------|---------------|----------|--------------|---------|-----|------|
| 0 | 7129300520 | "20141013T000000" | 221900 | 3,1,1180,5,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 1 | 6414100192 | "20141209T000000" | 538000 | 3,2.5,257,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 5631500400 | "20150225T000000" | 180000 | 2,1,770,10,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 3 | 2487200875 | "20141209T000000" | 604000 | 4,3,1960,5,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 4 | 1954400510 | "20150218T000000" | 510000 | 3,2,1680,8,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. | ... | ... | ... | ... | ... | ... | ... | ... |
| 21608 | 0263000018 | "20140521T000000" | 360000 | 3,2.5,1530,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 21609 | 6600060120 | "20150223T000000" | 400000 | 4,2.5,2310,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 21610 | 1523300141 | "20140623T000000" | 402101 | 2,0.75,102,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 21611 | 0291310100 | "20150116T000000" | 400000 | 3,2.5,1600,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 21612 | 1523300157 | "20141015T000000" | 325000 | 2,0.75,102,.. | NaN | NaN | NaN | NaN | NaN | NaN | .. | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 21613 | rows | x 21 columns | | | | | | | | | | | | | | | | | |

De kolomtitels waren correct gesplitst maar de waarden niet. Die bevonden zich nog steeds in de eerste kolom. Ik ben dus begonnen met die eerste kolom te splitsen en de gesplitste waarden heb ik in een ander dataframe geladen (df2).

| | df2=df['id'].str.split(pat=',', expand=True) | df2 | | | | | | | | | | | | | | | | | | | |
|-------|--|-------------------|--------|-----|------|------|-------|-----|-----|-----|-----|-----|------|-----|------|------|---------|---------|----------|------|------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
| 0 | 7129300520 | "20141013T000000" | 221900 | 3 | 1 | 1180 | 5650 | "1" | 0 | 0 | .. | 7 | 1180 | 0 | 1955 | 0 | "98178" | 47.5112 | -122.257 | 1340 | 5650 |
| 1 | 6414100192 | "20141209T000000" | 538000 | 3 | 2.25 | 2570 | 7242 | "2" | 0 | 0 | .. | 7 | 2170 | 400 | 1951 | 1991 | "98125" | 47.721 | -122.319 | 1690 | 7639 |
| 2 | 5631500400 | "20150225T000000" | 180000 | 2 | 1 | 770 | 10000 | "1" | 0 | 0 | .. | 6 | 770 | 0 | 1933 | 0 | "98028" | 47.7379 | -122.233 | 2720 | 8062 |
| 3 | 2487200875 | "20141209T000000" | 604000 | 4 | 3 | 1960 | 5000 | "1" | 0 | 0 | .. | 7 | 1050 | 910 | 1965 | 0 | "98136" | 47.5208 | -122.393 | 1360 | 5000 |
| 4 | 1954400510 | "20150218T000000" | 510000 | 3 | 2 | 1680 | 8080 | "1" | 0 | 0 | .. | 8 | 1680 | 0 | 1987 | 0 | "98074" | 47.6168 | -122.045 | 1800 | 7503 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 21608 | 0263000018 | "20140521T000000" | 360000 | 3 | 2.5 | 1530 | 1131 | "3" | 0 | 0 | .. | 8 | 1530 | 0 | 2009 | 0 | "98103" | 47.6993 | -122.346 | 1530 | 1509 |
| 21609 | 6600060120 | "20150223T000000" | 400000 | 4 | 2.5 | 2310 | 5813 | "2" | 0 | 0 | .. | 8 | 2310 | 0 | 2014 | 0 | "98146" | 47.5107 | -122.362 | 1830 | 7200 |
| 21610 | 1523300141 | "20140623T000000" | 402101 | 2 | 0.75 | 1020 | 1350 | "2" | 0 | 0 | .. | 7 | 1020 | 0 | 2009 | 0 | "98144" | 47.5944 | -122.299 | 1020 | 2007 |
| 21611 | 0291310100 | "20150116T000000" | 400000 | 3 | 2.5 | 1600 | 2388 | "2" | 0 | 0 | .. | 8 | 1600 | 0 | 2004 | 0 | "98027" | 47.5345 | -122.069 | 1410 | 1287 |
| 21612 | 1523300157 | "20141015T000000" | 325000 | 2 | 0.75 | 1020 | 1076 | "2" | 0 | 0 | .. | 7 | 1020 | 0 | 2008 | 0 | "98144" | 47.5941 | -122.299 | 1020 | 1357 |
| 21613 | rows | x 21 columns | | | | | | | | | | | | | | | | | | | |

Nu had ik een dataframe met correct gesplitste kolomtittels, maar zonder gesplitste waarden in mijn rijen. Ik had ook een dataframe zonder kolomtitels, maar met correct gesplitste waarden. Om dit op te lossen heb ik de kolomnamen van mijn eerste dataframe (df) over gezet naar mijn tweede dataframe (df2).

```
df2.columns = df.columns.tolist()
```

Na deze stap zag mijn dataframe er al beter uit maar er was nog veel werk aan de winkel. Voor het mezelf gemakkelijk te maken had ik mijn tweede dataframe (df2) terug df genoemd aangezien ik mijn oorspronkelijke dataframe niet meer nodig had.

```
df= df2
```

Wat me direct opviel als ik naar het dataframe keek was de aanhalingstekens die in verschillende kolommen aanwezig waren.

| | id | date | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | ... | grade | sqft_above | sqft_basement | yr_built | yr_renovated | zipcode | lat | long | sqft_living15 |
|-------|------------|-------------------|--------|----------|-----------|-------------|----------|--------|------------|------|-----|-------|------------|---------------|----------|--------------|---------|---------|----------|---------------|
| 0 | 7129300520 | "20141013T000000" | 221900 | 3 | 1 | 1180 | 5650 | "1" | 0 | 0 | ... | 7 | 1180 | 0 | 1955 | 0 | "98178" | 47.5112 | -122.257 | 1340 |
| 1 | 6414100192 | "20141209T000000" | 538000 | 3 | 2.25 | 2570 | 7242 | "2" | 0 | 0 | ... | 7 | 2170 | 400 | 1951 | 1991 | "98125" | 47.7212 | -122.319 | 1690 |
| 2 | 5631500400 | "20150225T000000" | 180000 | 2 | 1 | 770 | 10000 | "1" | 0 | 0 | ... | 6 | 770 | 0 | 1933 | 0 | "98028" | 47.7379 | -122.233 | 2720 |
| 3 | 2487200875 | "20141209T000000" | 604000 | 4 | 3 | 1960 | 5000 | "1" | 0 | 0 | ... | 7 | 1050 | 910 | 1965 | 0 | "98136" | 47.5208 | -122.393 | 1360 |
| 4 | 1954400510 | "20150218T000000" | 510000 | 3 | 2 | 1680 | 8080 | "1" | 0 | 0 | ... | 8 | 1680 | 0 | 1987 | 0 | "98074" | 47.6168 | -122.045 | 1800 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 21608 | 0263000018 | "20140521T000000" | 360000 | 3 | 2.5 | 1530 | 1131 | "3" | 0 | 0 | ... | 8 | 1530 | 0 | 2009 | 0 | "98103" | 47.6993 | -122.346 | 1530 |
| 21609 | 6600060120 | "20150223T000000" | 400000 | 4 | 2.5 | 2310 | 5813 | "2" | 0 | 0 | ... | 8 | 2310 | 0 | 2014 | 0 | "98146" | 47.5107 | -122.362 | 1830 |
| 21610 | 1523300141 | "20140623T000000" | 402101 | 2 | 0.75 | 1020 | 1350 | "2" | 0 | 0 | ... | 7 | 1020 | 0 | 2009 | 0 | "98144" | 47.5944 | -122.299 | 1020 |
| 21611 | 0291310100 | "20150116T000000" | 400000 | 3 | 2.5 | 1600 | 2388 | "2" | 0 | 0 | ... | 8 | 1600 | 0 | 2004 | 0 | "98027" | 47.5345 | -122.069 | 1410 |
| 21612 | 1523300157 | "20141015T000000" | 325000 | 2 | 0.75 | 1020 | 1076 | "2" | 0 | 0 | ... | 7 | 1020 | 0 | 2008 | 0 | "98144" | 47.5941 | -122.299 | 1020 |

Die wou ik dus verwijderen. Om dat te bereiken heb ik eerst de kolommen met aanhalingstekens in een lijst gezet en in een variabele opgeslagen. Ik heb vervolgens een for-loop gebruikt die gebruik maakt van die variabele in combinatie met de string replace functie om de aanhalingstekens te vervangen door niets.

```
kolommen = ['floors', 'zipcode', 'date']

for col in kolommen:
    df[col] = df[col].str.replace("'", "")
```

In de date kolom stond er na de datumcijfers nog een T met nullen achter. Ik heb die verwijderd door de split functie te gebruiken.

```
df['date']= df['date'].str.split('T').str[0]
```

Vervolgens heb ik het datatype veranderd van string naar datetime.

```
df['date']= pd.to_datetime(df['date'])
```

Dan werden mijn datumcijfers automatisch opgesplitst door streepjes.

| | id | date | price | bedrooms | bathrooms |
|---|------------|------------|--------|----------|-----------|
| 0 | 7129300520 | 2014-10-13 | 221900 | 3 | 1 |
| 1 | 6414100192 | 2014-12-09 | 538000 | 3 | 2.25 |

Er waren teveel kolommen aanwezig in deze dataset om allemaal te analyseren dus de minst interessante heb ik laten vallen.

```
kolommen_te_verwijderen= ['lat', 'long', 'sqft_living', 'sqft_basement', 'sqft_above', 'yr_renovated',
                           'grade', 'sqft_living15', 'sqft_lot15','zipcode']

df.drop(columns= kolommen_te_verwijderen, inplace= True)
```

Dan heb ik de data frame info opgevraagd om na te gaan of er nergens NaN-waarden verstopt zaten en om na te kijken welke kolommen er nog naar een ander datatype moesten gecast worden.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 12 columns):
 #   Column      Non-Null Count Dtype  
--- 
 0   id          21613 non-null   object  
 1   date        21613 non-null   datetime64[ns]
 2   price       21613 non-null   object  
 3   bedrooms    21613 non-null   object  
 4   bathrooms   21613 non-null   object  
 5   sqft_lot    21613 non-null   object  
 6   floors      21613 non-null   object  
 7   waterfront  21613 non-null   object  
 8   view        21613 non-null   object  
 9   condition   21613 non-null   object  
 10  yr_built    21613 non-null   object  
 11  zipcode     21613 non-null   object  
dtypes: datetime64[ns](1), object(11)
memory usage: 2.0+ MB
```

Het bleek dat er geen NaN-waarden aanwezig waren in de dataset. De meeste kolommen moesten nog wel naar een meer gepaste datatype worden gecast. Ik heb een lijst met de kolommen aangemaakt waar ik vanaf wou. Die lijst heb ik aan een variabele toegewezen.

```
kolommen_naar_integer = ['id', 'price', 'bedrooms', 'bathrooms',
                           'sqft_lot', 'floors', 'view', 'condition', 'yr_built']
```

Ik heb dan die variabele gebruikt om verschillende kolommen in een keer te casten naar het integer datatype.

```
df[kolommen_naar_integer] = df[kolommen_naar_integer].apply(pd.to_numeric, errors='coerce', downcast='integer')
```

Vervolgens wou ik de waterfront kolom omzetten naar boolean. Dat kon ik niet in een stap doen. Ik moest de kolom eerst casten naar integer aangezien de kolom de waarden “0” en “1” bevatte. Had ik die kolom direct naar boolean gecast, dan had die voor alle rijen “True” aangegeven aangezien “0” en “1” geen lege stringen zijn.

```
df['waterfront']= df['waterfront'].astype(int)
df['waterfront']= df['waterfront'].astype(bool)
```

Daarna heb ik de kolomnamen van het Engels naar het Nederlands vertaald.

```
df.rename(columns={'id': 'Huis_Prijs_ID', 'date': 'Datum', 'price': 'Prijs',
                  'bedrooms': 'Slaapkamers', 'bathrooms': 'Badkamers', 'sqft_lot': 'Oppervlakte',
                  'floors': 'Verdiepingen', 'waterfront': 'Waterkant', 'view': 'Uitzicht',
                  'condition': 'Conditie', 'yr_built': 'Bouwjaar'}, inplace= True)
```

Vervolgens heb ik de Huis_Prijs_ID kolom aangewezen als indexkolom.

```
df.set_index('Huis_Prijs_ID', inplace= True)
```

Ik heb de kolom volgorde nog aangepast zodat het beter oogt. Ik heb de oppervlakte kolom naar voor gebracht.

```
oppervlakte_kolom = df.columns[-5]
df.insert(5, oppervlakte_kolom, df.pop(oppervlakte_kolom))
```

De oppervlakte kolom stond uitgedrukt in vierkante voet. Die heb ik met een formule omgezet in vierkante meter. Ik heb dan de vierkante meter waarden afgrond op 2 decimalen.

```
df['Oppervlakte'] = df['Oppervlakte'] * 0.092903
```

```
df['Oppervlakte']= df['Oppervlakte'].round(2)
```

Tenslotte heb ik nog de kolommen met teveel unieke waarden in categorieën geplaatst. Ik heb daarvoor de qcut functie gebruikt. Ik heb ook een nieuwe kolom aangemaakt, de bouwjaar categorie kolom. Die is afgeleid van de bouwjaar kolom, maar in deze kolom zijn de waarden verdeeld in 5 kwantielen.

```
df['Badkamers']= pd.qcut(df['Badkamers'], q=3, labels=['Weinig', 'Gemiddeld', 'Veel'])

df['Bouwjaar_Categorie']= pd.qcut(df['Bouwjaar'], q=5, labels=['Heel Oud', 'Oud', 'Gemiddeld', 'Nieuw', 'Heel Nieuw'])

df['Slaapkamers']= pd.qcut(df['Slaapkamers'], q=3, labels=['Weinig', 'Gemiddeld', 'Veel'])

df['Oppervlakte']= pd.qcut(df['Oppervlakte'], q=5, labels=['Heel Klein', 'Klein', 'Gemiddeld', 'Groot', 'Heel Groot'])
```

Door de qcut functie te gebruiken, wordt het datatype van de kolom ook meteen naar het categorie datatype gecast.

```
Datum           datetime64[ns]
Prijs            int32
Slaapkamers    category
Badkamers       category
Oppervlakte    category
Verdiepingen   float64
Wterkant        bool
Uitzicht        int8
Conditie        int8
Bouwjaar        int16
Bouwjaar_Categorie  category
dtype: object
```

Hiermee was het opschonen van de data afgerond en had ik nu dit onderstaand dataframe als resultaat.

| Huis_Prijs_ID | Datum | Prijs | Slaapkamers | Badkamers | Oppervlakte | Verdiepingen | Waterkant | Uitzicht | Conditie | Bouwjaar | Bouwjaar_Categorie |
|---------------|------------|--------|-------------|-----------|-------------|--------------|-----------|----------|----------|----------|--------------------|
| 7129300520 | 2014-10-13 | 221900 | Weinig | Weinig | Klein | 1.0 | False | 0 | 3 | 1955 | Oud |
| 6414100192 | 2014-12-09 | 538000 | Weinig | Gemiddeld | Gemiddeld | 2.0 | False | 0 | 3 | 1951 | Oud |
| 5631500400 | 2015-02-25 | 180000 | Weinig | Weinig | Groot | 1.0 | False | 0 | 3 | 1933 | Heel Oud |
| 2487200875 | 2014-12-09 | 604000 | Gemiddeld | Veel | Klein | 1.0 | False | 0 | 5 | 1965 | Oud |
| 1954400510 | 2015-02-18 | 510000 | Weinig | Gemiddeld | Gemiddeld | 1.0 | False | 0 | 3 | 1987 | Nieuw |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 263000018 | 2014-05-21 | 360000 | Weinig | Gemiddeld | Heel Klein | 3.0 | False | 0 | 3 | 2009 | Heel Nieuw |
| 6600060120 | 2015-02-23 | 400000 | Gemiddeld | Gemiddeld | Klein | 2.0 | False | 0 | 3 | 2014 | Heel Nieuw |
| 1523300141 | 2014-06-23 | 402101 | Weinig | Weinig | Heel Klein | 2.0 | False | 0 | 3 | 2009 | Heel Nieuw |
| 291310100 | 2015-01-16 | 400000 | Weinig | Gemiddeld | Heel Klein | 2.0 | False | 0 | 3 | 2004 | Heel Nieuw |
| 1523300157 | 2014-10-15 | 325000 | Weinig | Weinig | Heel Klein | 2.0 | False | 0 | 3 | 2008 | Heel Nieuw |

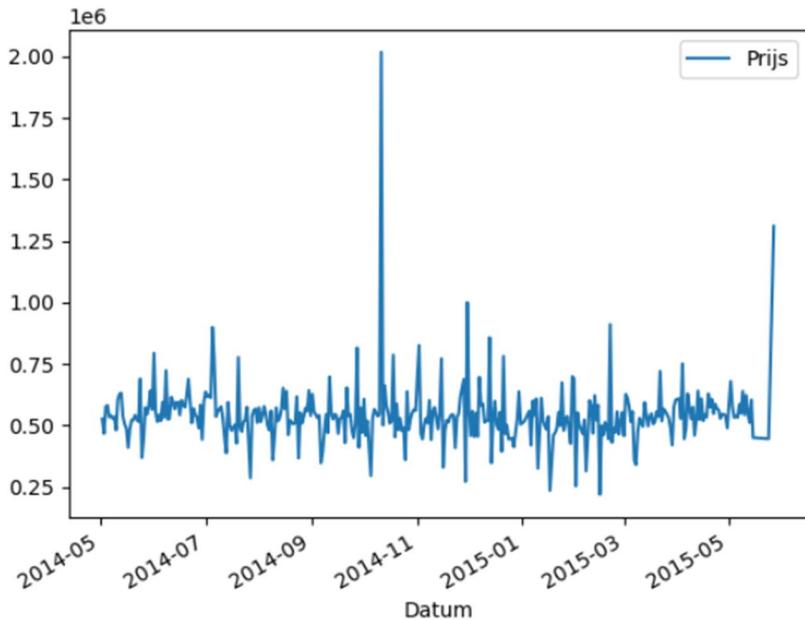
21613 rows x 11 columns

Data analyse

In de kolomtitels van mijn dataframe had ik nu 9 factoren die mogelijk een impact kunnen hebben op de huizenprijs. In het deel data analyse ben ik er proberen achter te komen welke factoren de grootste impact hebben op de huizenprijs.

Factor 1: De datum

```
round(df.groupby(['Datum'])['Prijs'].mean().to_frame(),2).plot()
```



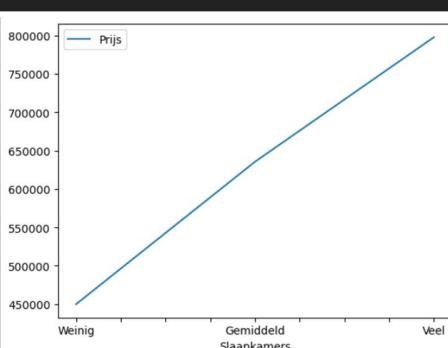
Ik zag geen duidelijke trend als ik groepeerde op de datum. Waarschijnlijk omdat de data niet heel ver uit elkaar lagen. Elke huis verkoop uit de dataset vond plaats in 2014 of 2015.

Factor 2: Aantal slaapkamers

```
slaapkamers= round(df.groupby(['Slaapkamers'])['Prijs'].mean().to_frame(),2)
```

Slaapkamers

| | Prijs |
|------------------|-----------|
| Weinig | 449873.95 |
| Gemiddeld | 635419.50 |
| Veel | 797612.90 |



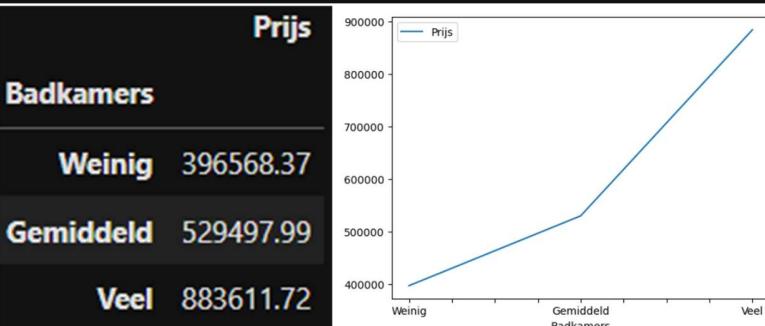
```
print(round(((slaapkamers.iloc[2,0]-df['Prijs'].mean())/df['Prijs'].mean()*100,2), "%")
```

47.68 %

Er is een duidelijke positieve correlatie tussen het aantal slaapkamers en de huizenprijs volgens deze dataset. De groep huizen met de meeste slaapkamers zijn bijna 48% duurder dan de gemiddelde huisprijs.

Factor 3: Aantal badkamers

```
badkamers= round(df.groupby(['Badkamers'])['Prijs'].mean().to_frame(),2)
```

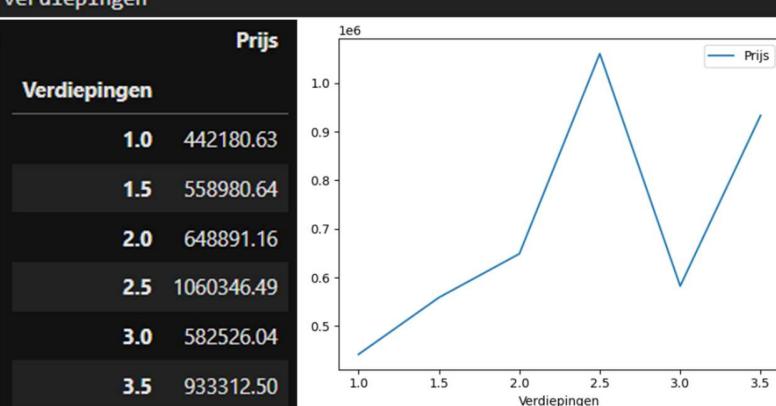


```
print(round(((badkamers.iloc[2,0]-df['Prijs'].mean())/df['Prijs'].mean())*100,2),"%")  
63.61 %
```

Ook het aantal badkamers is blijkbaar geassocieerd met hogere huizenprijzen. Het verschil is hier nog duidelijker dan bij de slaapkamers. Niet heel verbazend want een groter aantal badkamers is vaak een teken van luxe.

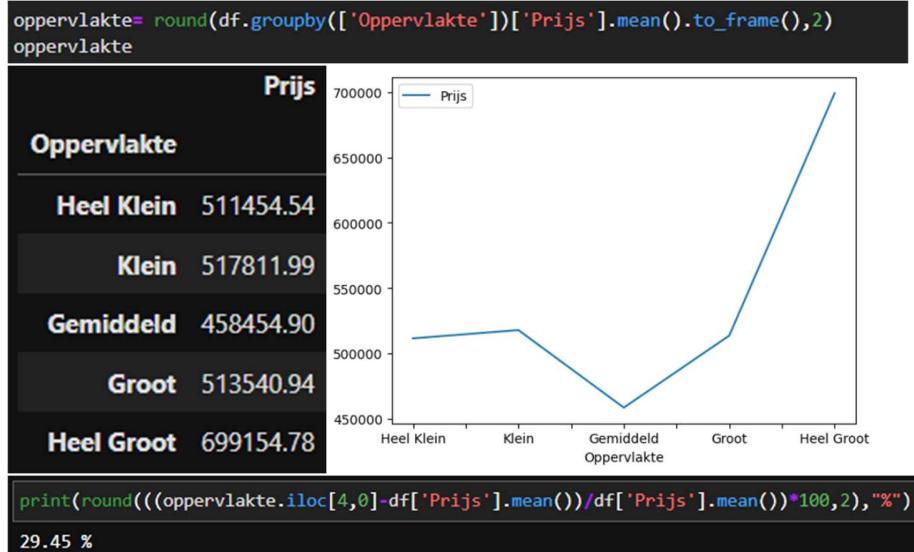
Factor 4: Aantal verdiepingen

```
verdiepingen= round(df.groupby(['Verdiepingen'])['Prijs'].mean().to_frame(),2)
```



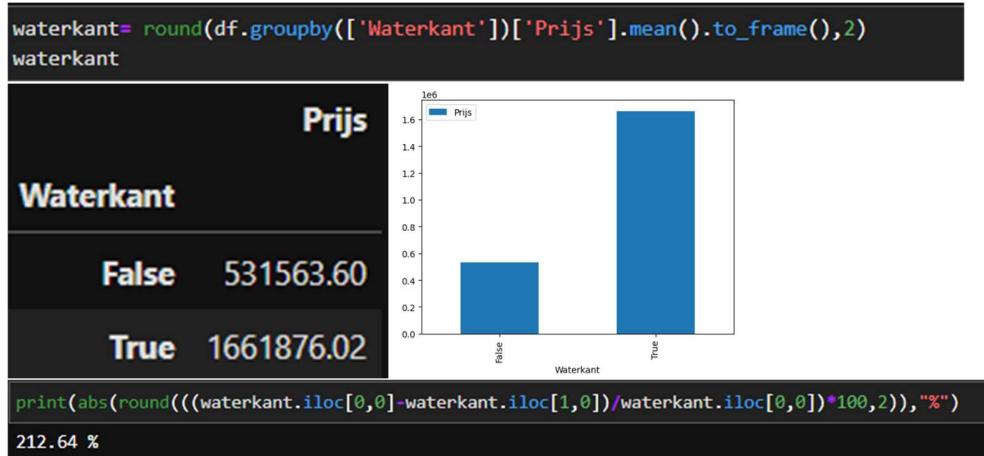
Het aantal verdiepingen lijkt positief gecorreleerd te zijn met hogere huizenprijzen, hoewel de trend niet in rechte lijn stijgt. Het is wel duidelijk dat de huizen met slechts één verdieping het goedkoopst zijn.

Factor 5: Oppervlakte



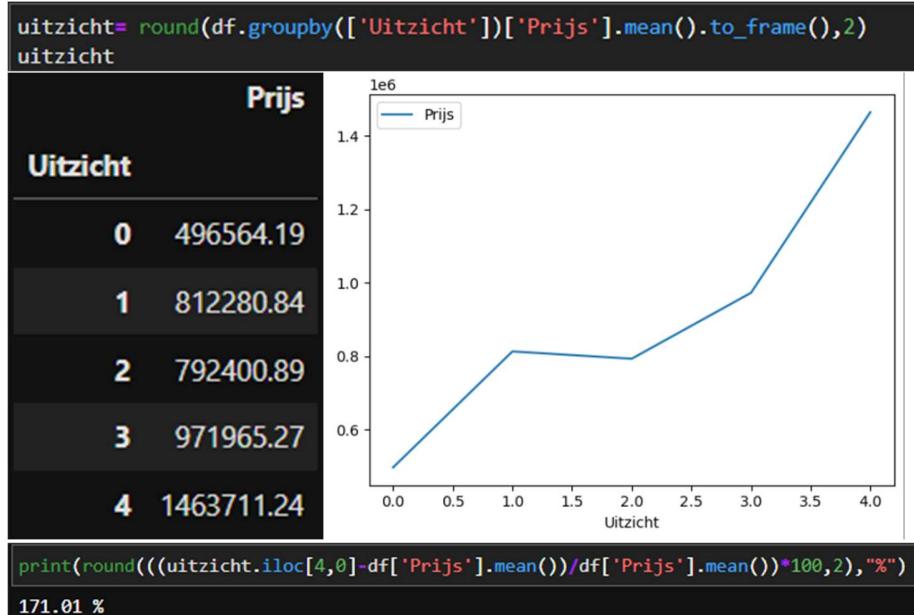
Uit deze grafiek kunnen we concluderen dat het vooral heel grote huizen zijn die duurder zijn dan de rest. Op de rest van de maten zit niet veel verschil qua prijs. Ten minste volgens deze dataset. De grootste huizen zijn 29% duurder dan de gemiddelde huisprijs.

Factor 6: Aanwezigheid waterkant



De aanwezigheid van een waterkant is bijzonder sterk gecoreleerd met hogere huizenprijzen. De huizen gelegen aan een waterkant zijn maar liefst 212% duurder dan die elders gelegen zijn volgens de dataset.

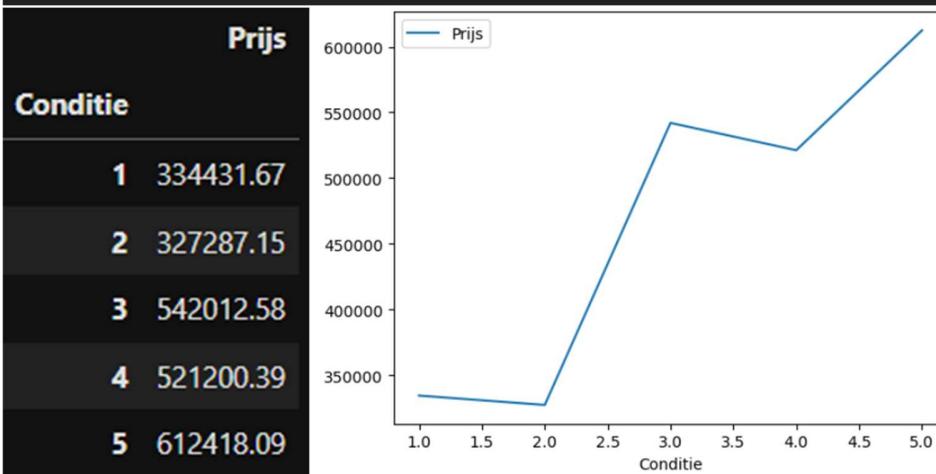
Factor 7: Uitzicht



Hoe mooier het uitzicht, hoe duurder de huizen. De huizen met de hoogste score op vlak van uitzicht zijn 171% duurder dan het gemiddelde huis uit de dataset.

Factor 8: Huis conditie

```
conditie= round(df.groupby(['Conditie'])['Prijs'].mean().to_frame(),2)
conditie
```



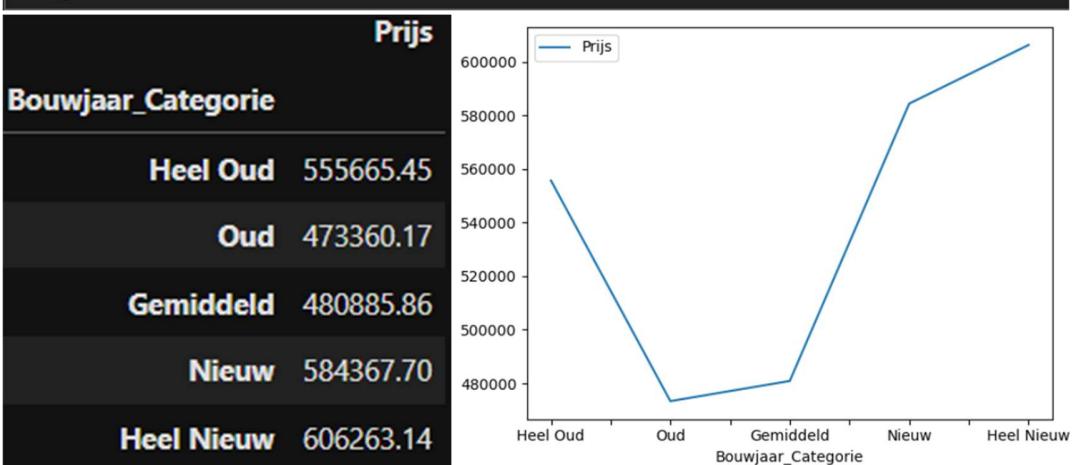
```
print(round(((conditie.iloc[4,0]-df['Prijs'].mean())/df['Prijs'].mean())*100,2), "%")
```

13.39 %

De staat waarin het huis zich bevindt is ook een belangrijke factor die een rol speelt in de valuatie van huizen. Huizen met een hogere conditie score zijn duurder dan huizen met een lagere score. Het verschil mag dan wel niet zo groot zijn als bij andere factoren die ik reeds besproken heb maar er is toch een duidelijke trend te bespeuren. De huizen met de hoogste conditie score zijn 13% duurder dan gemiddeld.

Factor 9: Bouwjaar

```
bouwjaar= round(df.groupby(['Bouwjaar_Categorie'])['Prijs'].mean().to_frame(),2)
bouwjaar
```



Qua bouwjaar is er geen duidelijke link te bespeuren volgens deze dataset.

Exporteren

Ik heb dan mijn opgeschoonde dataframe naar csv geëxporteerd zodat ik die later in Power BI kon importeren.

```
df.to_csv('dataset_2.csv',index=True)
```

Conclusie

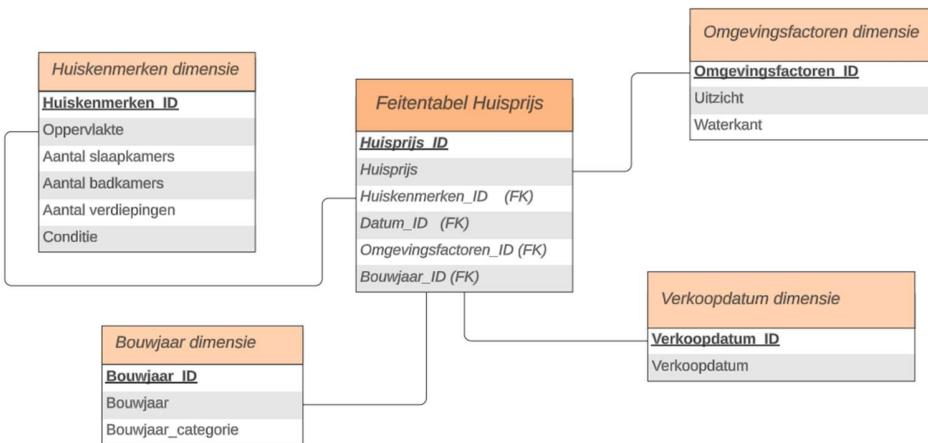
Volgens deze dataset heeft de locatie van het huis de grootste invloed op de huizenprijs. De huizen gelegen aan een waterkant waren 212% duurder dan de huizen die niet aan een waterkant gelegen zijn. Huizen met een hoge score op vlak van uitzicht waren 171% duurder dan gemiddeld. Verder was in mindere mate een hoog aantal slaap-en badkamers geassocieerd met hogere huizenprijzen.

Data visualisatie

Om mijn model in Power BI te maken heb ik eerst een sterschema gemaakt in Lucidchart. Ik ben begonnen met een huizenprijs feitentabel te maken die verbonden is aan vier dimensies door middel van foreign keys. Deze dimensies zijn: de huiskenmerken dimensie, de omgevingsfactoren dimensie, de verkoopdatum dimensie en de bouwjaar dimensie.

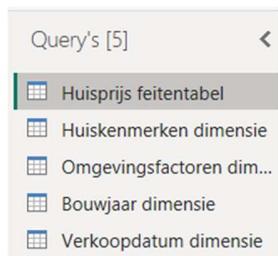
Stap 1: Schematiseren

House sales in king County, USA



Stap 2: Modelleren in Power BI

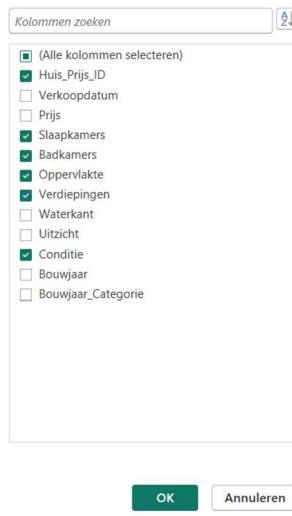
Nu ik mijn sterschema had, kon ik beginnen mijn model te bouwen in Power BI. Ik heb mijn geëxporteerde csv file (mijn pandas dataframe) in Power BI geladen. Dan heb ik op “gegevens transformeren” geklikt om power query te openen. De data was ondertussen opgeschoond, dus moest ik alleen nog mijn model bouwen. Om te beginnen heb ik mijn query 4 keer geduplicateerd zodat ik 5 tabellen had. Vervolgens heb ik de query’s elk een nieuwe naam gegeven. De zelfde namen als in mijn sterschema.



Voor elke dimensie heb ik enkel de relevante kolommen gekozen en weer een nieuwe indexkolom toegevoegd met mijn surrogate key. Die surrogate key werk als een primary key en krijgt dus ook de naam van de primary key in mijn sterschema schets.

Kolommen kiezen

De kolommen kiezen die u wilt behouden



OK

Annuleren

Query's [5] < = Table.RenameColumns#"Index toegevoegd", {"Index", "Huiskenmerken_ID"}#

| | A ^B _C Slaapkamers | A ^B _C Badkamers | A ^B _C Oppervlakte | A ^B ₃ Verdiepingen | A ^B ₃ Conditie | A ^B ₃ Huiskenmerken_ID | |
|----|---|---------------------------------------|---|--|--------------------------------------|--|--|
| 1 | 7129300520 Weinig | Weinig | Klein | 10 | 3 | 1 | |
| 2 | 6414100192 Weinig | Gemiddeld | Gemiddeld | 20 | 3 | 2 | |
| 3 | 5631500400 Weinig | Weinig | Groot | 10 | 3 | 3 | |
| 4 | 2487200875 Gemiddeld | Veel | Klein | 10 | 5 | 4 | |
| 5 | 1954400510 Weinig | Gemiddeld | Gemiddeld | 10 | 3 | 5 | |
| 6 | 7237550310 Gemiddeld | Veel | Heel Groot | 10 | 3 | 6 | |
| 7 | 1321400060 Weinig | Gemiddeld | Gemiddeld | 20 | 3 | 7 | |
| 8 | 2008000270 Weinig | Weinig | Groot | 10 | 3 | 8 | |
| 9 | 2414600126 Weinig | Weinig | Gemiddeld | 10 | 3 | 9 | |
| 10 | 3793500160 Weinig | Gemiddeld | Klein | 20 | 3 | 10 | |
| 11 | 1736800520 Weinig | Gemiddeld | Groot | 10 | 3 | 11 | |
| 12 | 9212900260 Weinig | Weinig | Klein | 10 | 4 | 12 | |
| 13 | 114101516 Weinig | Weinig | Heel Groot | 15 | 4 | 13 | |
| 14 | 6054650070 Weinig | Weinig | Groot | 10 | 4 | 14 | |
| 15 | 1175000570 Veel | Gemiddeld | Klein | 15 | 3 | 15 | |
| 16 | 9297300055 Weinig | Veel | Klein | 20 | 3 | 16 | |
| 17 | 1875500060 Weinig | Gemiddeld | Heel Groot | 20 | 3 | 17 | |
| 18 | 6865200140 Gemiddeld | Weinig | Heel Klein | 15 | 4 | 18 | |
| 19 | 16000397 Weinig | Weinig | Groot | 10 | 4 | 19 | |
| 20 | 7983200060 Weinig | Weinig | Groot | 10 | 4 | 20 | |
| 21 | 6300500875 Gemiddeld | Weinig | Klein | 10 | 4 | 21 | |

Vervolgens heb ik mijn dimensies samengevoegd aan mijn feitentabel via de Huis_Prijs_ID kolom.

Samenvoegen

Selecteer een tabel en overeenkomende kolommen om een samengevoegde tabel te maken.

Huisprijs feitentabel

| Huis_Prijs_ID | Verkoopdatum | Prijs | Slaapkamers | Badkamers | Oppervlakte | Verdiepingen | Waterkan |
|---------------|--------------|--------|-------------|-----------|-------------|--------------|----------------|
| 7129300520 | 13/10/2014 | 221900 | Weinig | Weinig | Klein | 10 | F ₁ |
| 6414100192 | 9/12/2014 | 538000 | Weinig | Gemiddeld | Gemiddeld | 20 | F ₂ |
| 5631500400 | 25/02/2015 | 180000 | Weinig | Weinig | Groot | 10 | F ₂ |
| 2487200875 | 9/12/2014 | 604000 | Gemiddeld | Veel | Klein | 10 | F ₂ |

Huiskenmerken dimensie

| Huis_Prijs_ID | Slaapkamers | Badkamers | Oppervlakte | Verdiepingen | Conditie | Huiskenmerken_ID |
|---------------|-------------|-----------|-------------|--------------|----------|------------------|
| 7129300520 | Weinig | Weinig | Klein | 10 | 3 | 1 |
| 6414100192 | Weinig | Gemiddeld | Gemiddeld | 20 | 3 | 2 |
| 5631500400 | Weinig | Weinig | Groot | 10 | 3 | 3 |
| 2487200875 | Gemiddeld | Veel | Klein | 10 | 5 | 4 |
| 1954400510 | Weinig | Gemiddeld | Gemiddeld | 10 | 3 | 5 |

Type join

Left outer (alle uit de eerste, overeenkomende uit de...)

Gebruik fuzzy overeenkomst om de samenvoeging uit te voeren

Opties voor fuzzy overeenkomsten

De selectie komt overeen met 21613 van 21613 rijen in de eerste tabel.

OK **Annuleren**

Ik heb dan de dimensies opengevouwd in de feitentabel en de surrogate key aangevinkt.

Bouwjaar_Categorie

Bouwjaar dimensie

Kolommen zoeken om uit te breiden

Uitvouwen Samenstellen

(Alle kolommen selecteren)

Huis_Prijs_ID

Bouwjaar

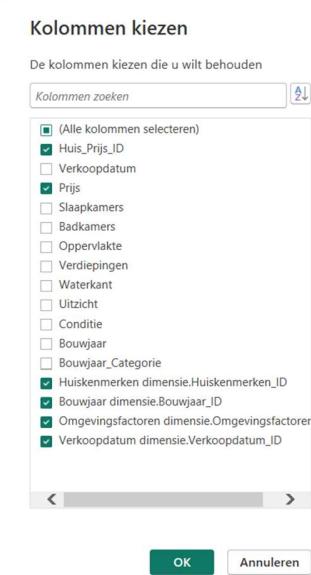
Bouwjaar_Categorie

Bouwjaar_ID

Oorspronkelijke kolomnaam gebruiken als voorvoegsel

OK **Annuleren**

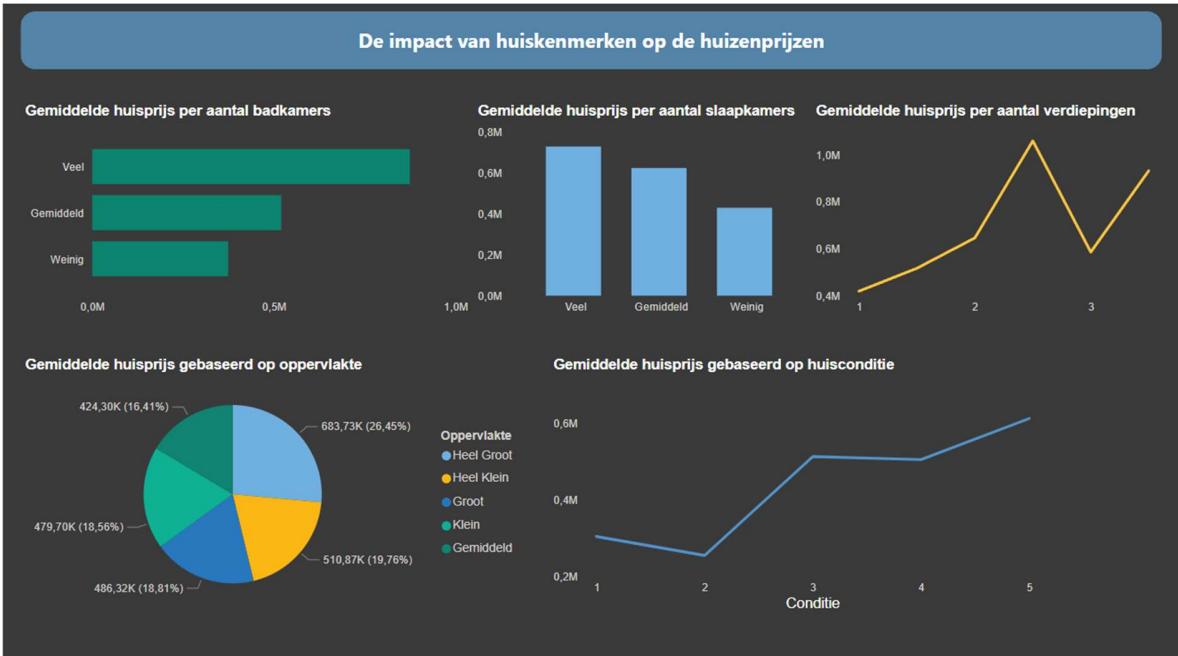
Tenslotte heb ik de niet-sleutel attributen van de dimensies uit mijn feitentabel verwijderd.



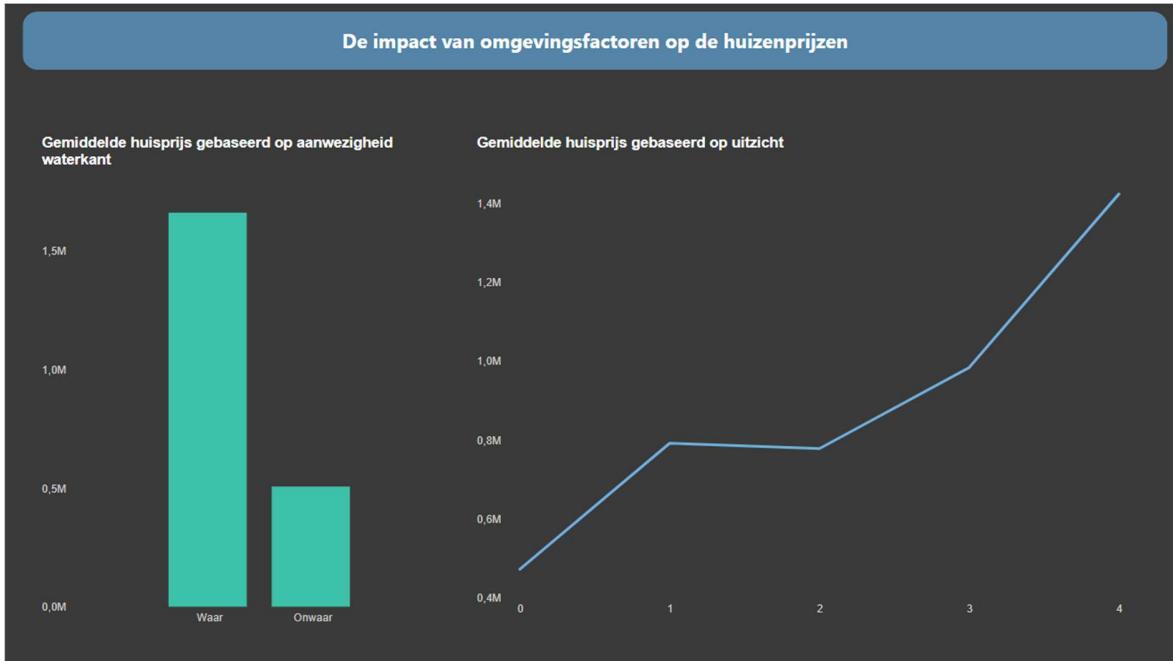
Het model was nu klaar dus heb ik Power Query gesloten.

Stap 3: Visuals maken

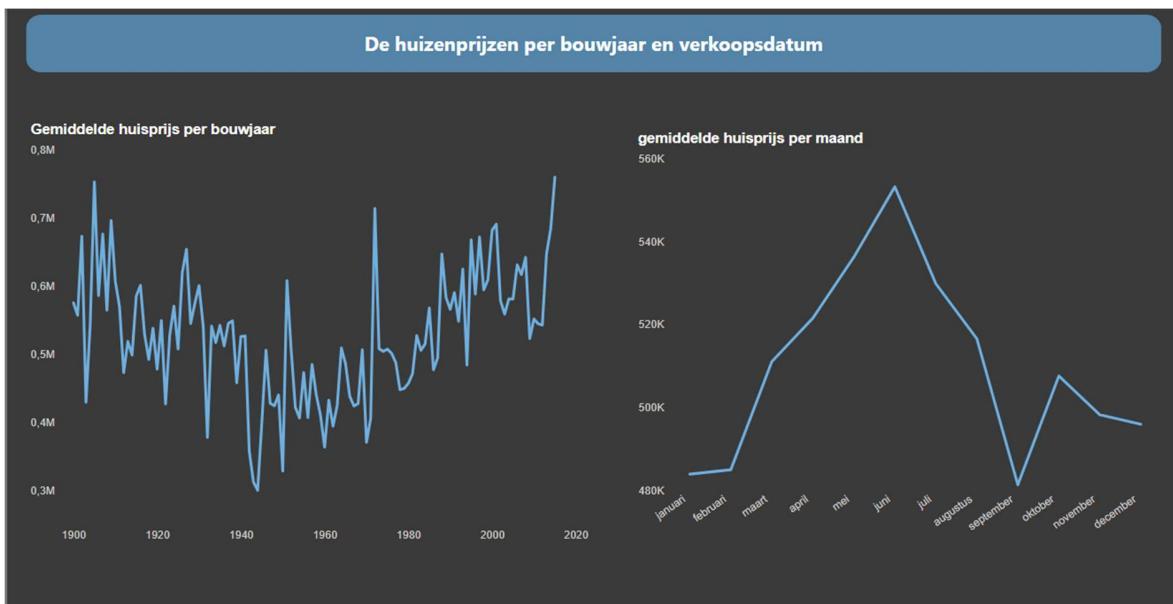
De huiskenmerken dimensie



De omgevingsfactoren dimensie



De bouwjaar en datum dimensies



Dataset 3: Housing prices dataset

Ik heb op Kaggle nog een derde interessante dataset [3] gevonden met een kleinere steekproef maar met een groter aanbod aan factoren. Ik heb de file gedownload en in de cloud opgeslagen.

Vervolgens heb ik weer de nodige bibliotheken geïmporteerd en mijn csv file in een dataframe geladen.

```
import pandas as pd
import numpy as np

df = pd.read_csv(r"C:\Users\mvdp1\OneDrive\Documenten\GitHub\Pandas-Oefeningen-Data\Housing.csv")
```

Voor ik kon beginnen met de data op te schonen heb ik mijn dataframe onder de loep genomen.

| | price | area | bedrooms | bathrooms | stories | mainroad | guestroom | basement | hotwaterheating | airconditioning | parking | prefarea | furnishingstatus |
|-----|----------|------|----------|-----------|---------|----------|-----------|----------|-----------------|-----------------|---------|----------|------------------|
| 0 | 13300000 | 7420 | 4 | 2 | 3 | yes | no | no | no | yes | 2 | yes | furnished |
| 1 | 12250000 | 8960 | 4 | 4 | 4 | yes | no | no | no | yes | 3 | no | furnished |
| 2 | 12250000 | 9960 | 3 | 2 | 2 | yes | no | yes | no | no | 2 | yes | semi-furnished |
| 3 | 12215000 | 7500 | 4 | 2 | 2 | yes | no | yes | no | yes | 3 | yes | furnished |
| 4 | 11410000 | 7420 | 4 | 1 | 2 | yes | yes | yes | no | yes | 2 | no | furnished |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 540 | 1820000 | 3000 | 2 | 1 | 1 | yes | no | yes | no | no | 2 | no | unfurnished |
| 541 | 1767150 | 2400 | 3 | 1 | 1 | no | no | no | no | no | 0 | no | semi-furnished |
| 542 | 1750000 | 3620 | 2 | 1 | 1 | yes | no | no | no | no | 0 | no | unfurnished |
| 543 | 1750000 | 2910 | 3 | 1 | 1 | no | no | no | no | no | 0 | no | furnished |
| 544 | 1750000 | 3850 | 3 | 1 | 2 | yes | no | no | no | no | 0 | no | unfurnished |

Data opschonen

Als eerste stap heb ik mijn dataframe informatie geraadpleegd.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   price        545 non-null    int64  
 1   area         545 non-null    int64  
 2   bedrooms     545 non-null    int64  
 3   bathrooms    545 non-null    int64  
 4   stories      545 non-null    int64  
 5   mainroad     545 non-null    object  
 6   guestroom    545 non-null    object  
 7   basement     545 non-null    object  
 8   hotwaterheating 545 non-null  object  
 9   airconditioning 545 non-null  object  
 10  parking       545 non-null    int64  
 11  prefarea     545 non-null    object  
 12  furnishingstatus 545 non-null  object  
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

Dan zag ik dat er geen NaN-waarden aanwezig waren.
Daar moest ik dus al geen rekening mee houden.

Ik zag wel dat er verschillende kolommen waren die ik nog moest casten naar een ander datatype.
Al de kolommen met “yes” en “no” als waarden moeten naar boolean geconverteerd worden. Om dit te bereiken moet ik eerst de waarden vervangen met “True” en “False”.

```
te_vervangen_kolommen = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']
df[te_vervangen_kolommen] = df[te_vervangen_kolommen].replace({'yes': True, 'no': False})
```

```
df.dtypes

price          int64
area           int64
bedrooms       int64
bathrooms      int64
stories         int64
mainroad        bool
guestroom       bool
basement        bool
hotwaterheating bool
airconditioning bool
parking         int64
prefarea        bool
furnishingstatus object
dtype: object
```

Ik heb de datatypes functie gebruikt om te zien of de kolommen effectief gecast waren naar boolean en dat bleek gelukt te zijn.

De laatste kolom heb ik gecast naar categorie, dat is het meest gepaste datatype in deze context.

```
df['furnishingstatus']= df['furnishingstatus'].astype('category')
```

Vervolgens heb ik een nieuwe kolom aangemaakt aan de hand van de area kolom. De bedoeling was om de area waarden in categorieën te plaatsen gebaseerd op omvang. Ik heb daarvoor de Pandas qcut functie gebruikt.

```
ruimtecategorie= pd.qcut(df['area'], q=5, labels= ['Heel klein','Klein','medium', 'Groot', 'Heel groot'])
df.insert(2, 'Ruimte_Categorie', ruimtecategorie)
```

Daarna heb ik de waarden in de area kolom omgezet van vierkante voet naar vierkante meter.

```
df['area'] = df.pop('area') * 0.092903
df.insert(1, 'area', df.pop('area'))
```

De nieuwe waarden in vierkante meter hadden nu teveel decimalen. Ik heb ze dus afgerond op 2 decimalen.

```
df['area'] = df['area'].round(2)
```

Ik heb de waarden van de laatste kolom vertaald naar het Nederlands.

```
df.furnishingstatus= df.furnishingstatus.replace({
    'furnished': 'Gemeubeld', 'semi-furnished': 'Deels gemeubeld', 'unfurnished': 'Ongemeubeld'})
```

Ook de kolomnamen heb ik naar het Nederlands vertaald.

```
df.rename(columns={'price': 'Prijs', 'area': 'Oppervlakte', 'bedrooms': 'Slaapkamers', 'bathrooms': 'Badkamers', 'stories': 'Verdiepingen',
'mainroad': 'Hoofdweg', 'guestroom': 'Gastenkamer', 'basement': 'Kelder', 'hotwaterheating': 'Warm_Water_Verwarming',
'airconditioning': 'Airco', 'parking': 'Parking', 'prefarea': 'Voorkeursbuurt', 'furnishingstatus': 'Bemeubelings_Status'}, inplace=True)
```

Tenslotte heb ik de index een nieuwe naam gegeven.

```
df = df.rename_axis('Huis_Prijs_ID')
```

Nu zag mijn getransformeerde dataframe er zo uit:

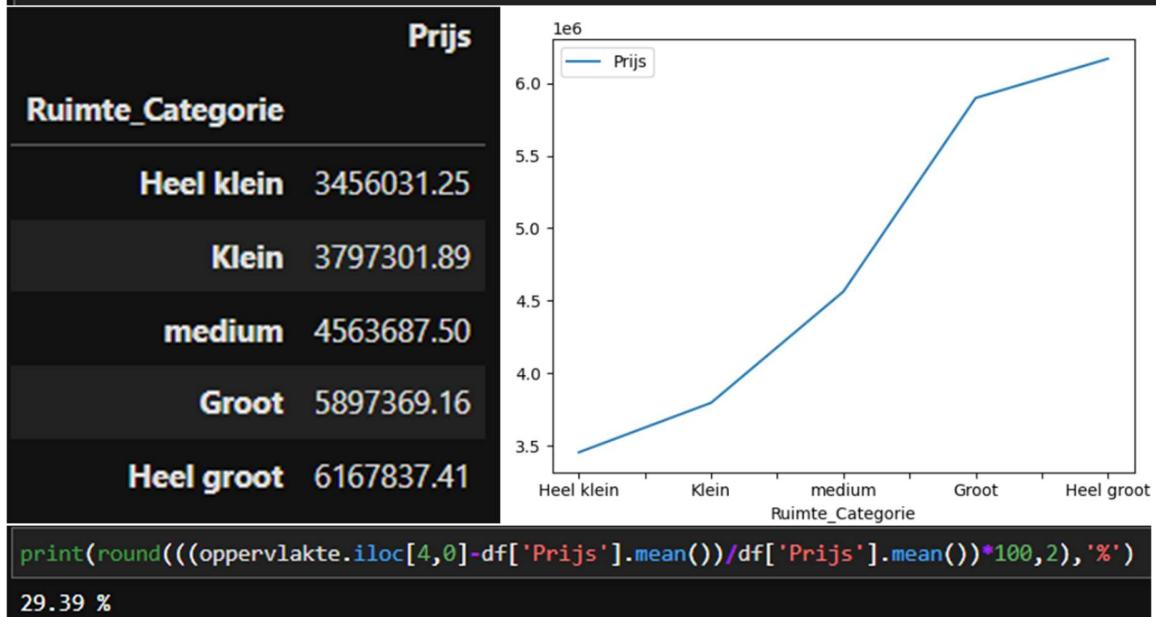
| House_Id | Prijs | Oppervlakte | Ruimte_Categorie | Slaapkamers | Badkamers | Verdiepingen | Hoofdweg | Gastenkamer | Kelder | Warm_Water_Verwarming | Airco | Parking | Voorkeursbuurt | Bemeubelings_Status |
|----------|----------|-------------|------------------|-------------|-----------|--------------|----------|-------------|--------|-----------------------|-------|---------|----------------|---------------------|
| 0 | 13300000 | 689.34 | Heel groot | 4 | 2 | 3 | True | False | False | False | True | 2 | True | Gemeubeld |
| 1 | 12250000 | 832.41 | Heel groot | 4 | 4 | 4 | True | False | False | False | True | 3 | False | Gemeubeld |
| 2 | 12250000 | 925.31 | Heel groot | 3 | 2 | 2 | True | False | True | False | False | 2 | True | Deels gemeubeld |
| 3 | 12215000 | 696.77 | Heel groot | 4 | 2 | 2 | True | False | True | False | True | 3 | True | Gemeubeld |
| 4 | 11410000 | 689.34 | Heel groot | 4 | 1 | 2 | True | True | True | False | True | 2 | False | Gemeubeld |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 540 | 1820000 | 278.71 | Heel klein | 2 | 1 | 1 | True | False | True | False | False | 2 | False | Ongemeubeld |
| 541 | 1767150 | 222.97 | Heel klein | 3 | 1 | 1 | False | False | False | False | False | 0 | False | Deels gemeubeld |
| 542 | 1750000 | 336.31 | Klein | 2 | 1 | 1 | True | False | False | False | False | 0 | False | Ongemeubeld |
| 543 | 1750000 | 270.35 | Heel klein | 3 | 1 | 1 | False | False | False | False | False | 0 | False | Gemeubeld |
| 544 | 1750000 | 357.68 | Klein | 3 | 1 | 2 | True | False | False | False | False | 0 | False | Ongemeubeld |

Data analyse

Na het opruimen was het tijd voor de analyse. Er waren 12 factoren waar ik op gegroepeerd heb.

Factor 1: Oppervlakte

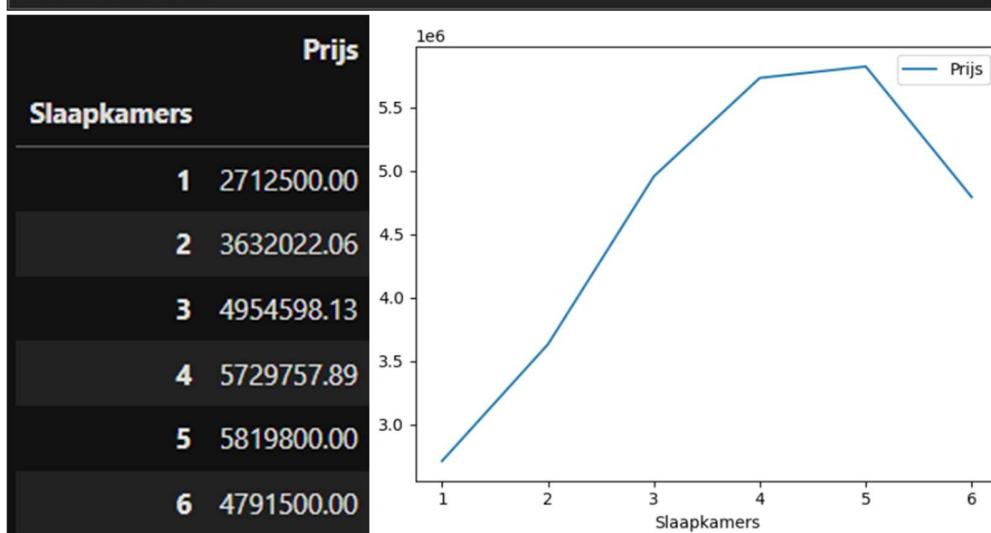
```
oppervlakte= df.groupby(['Ruimte_Categorie'])['Prijs'].mean().round(2).to_frame()  
oppervlakte
```



Volgens deze dataset kosten de hele grote huizen 29% meer dan het gemiddelde huis en bijna het dubbele van de hele kleine huizen.

Factor 2: Aantal slaapkamers

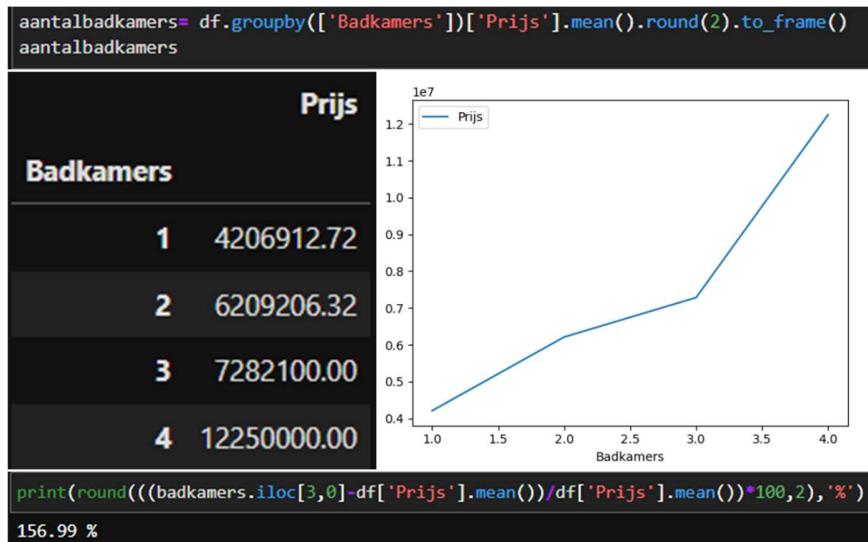
```
aantalslaapkamers= df.groupby(['Slaapkamers'])['Prijs'].mean().round(2).to_frame()  
aantalslaapkamers
```



Het is redelijk duidelijk te zien aan deze grafiek dat een groter aantal slaapkamers geassocieerd is met hogere huizenprijzen.

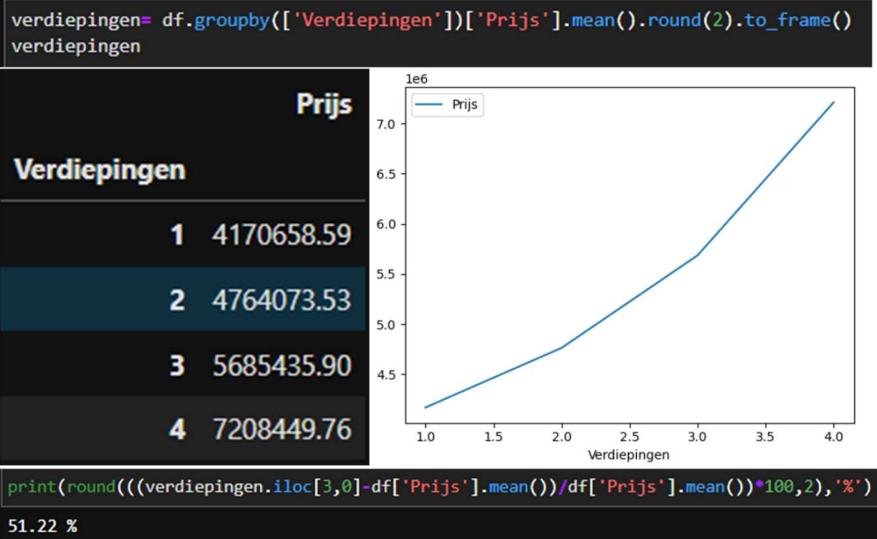
Hoewel er wel een significante prijsverlaging is vast te stellen van 5 naar 6 slaapkamers.

Factor 3: Aantal badkamers



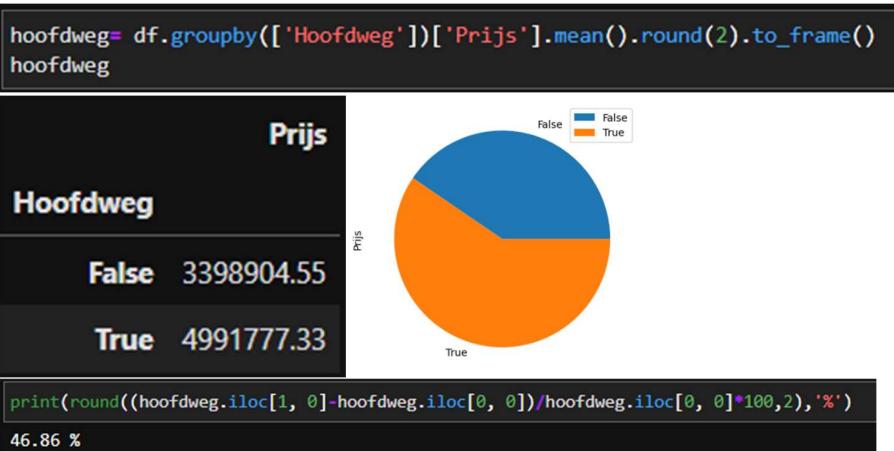
Het aantal badkamers is heel sterk positief gecorreleerd met de huizenprijs. Huizen met 4 badkamers kosten maar liefst 157% meer dan het gemiddelde.

Factor 4: Aantal verdiepingen



De prijs stijgt ook heel sterk naarmate het aantal verdiepingen toeneemt. Huizen met 4 verdiepingen kosten 51% meer dan het gemiddelde huis volgens de data.

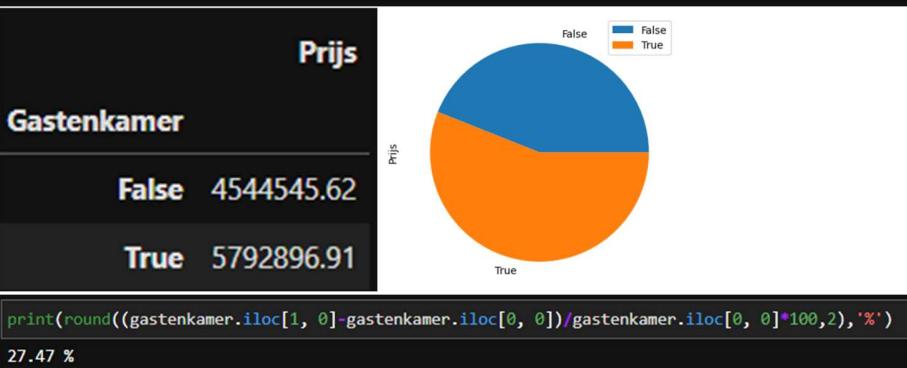
Factor 5: Aanwezigheid hoofdweg



De aanwezigheid van een hoofdweg is duidelijk een voorkeur van de meesten. Huizen die verbonden zijn met een hoofdweg zijn gemiddeld 47% duurder dan huizen die niet verbonden zijn met een hoofdweg.

Factor 6: Aanwezigheid gastenkamer

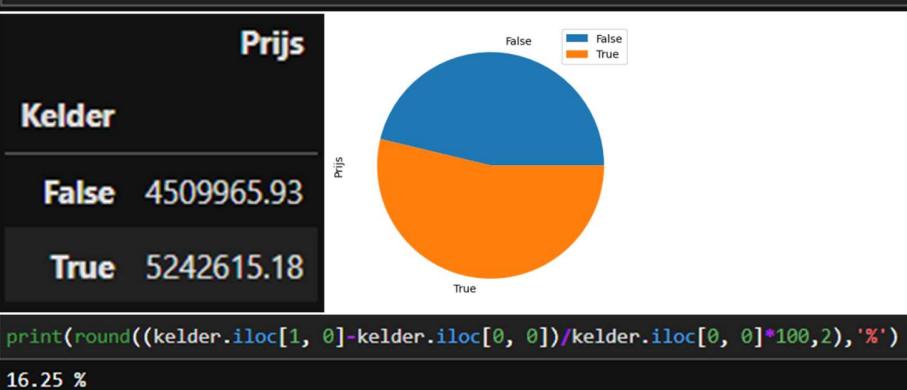
```
gastenkamer= df.groupby(['Gastenkamer'])['Prijs'].mean().round(2).to_frame()  
gastenkamer
```



De aanwezigheid van een gastenkamer is positief gecorreleerd met de huizenprijs volgens deze dataset. Huizen met een gastenkamer zijn gemiddeld gezien 27,5% duurder dan huizen zonder gastenkamer.

Factor 7: Aanwezigheid kelder

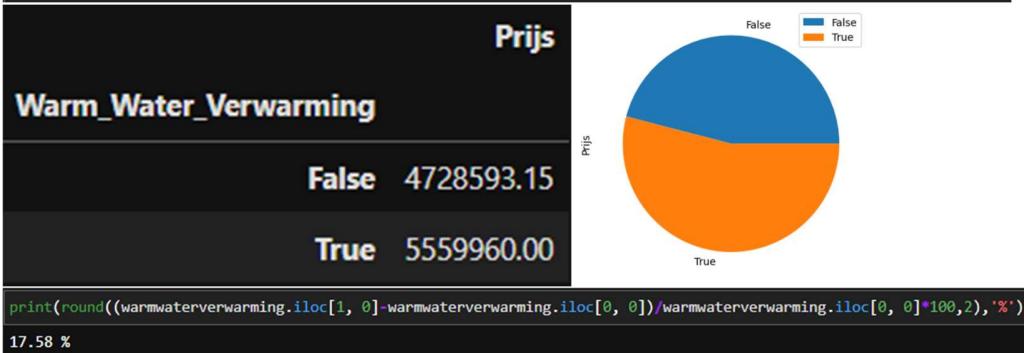
```
kelder= df.groupby(['Kelder'])['Prijs'].mean().round(2).to_frame()  
kelder
```



De aanwezigheid van een kelder is geassocieerd met hogere huizenprijzen. De link is iets minder significant dan de meeste factoren, maar huizen met een kelder zijn gemiddeld toch 16% duurder dan die zonder.

Factor 8: Aanwezigheid warmwaterverwarming

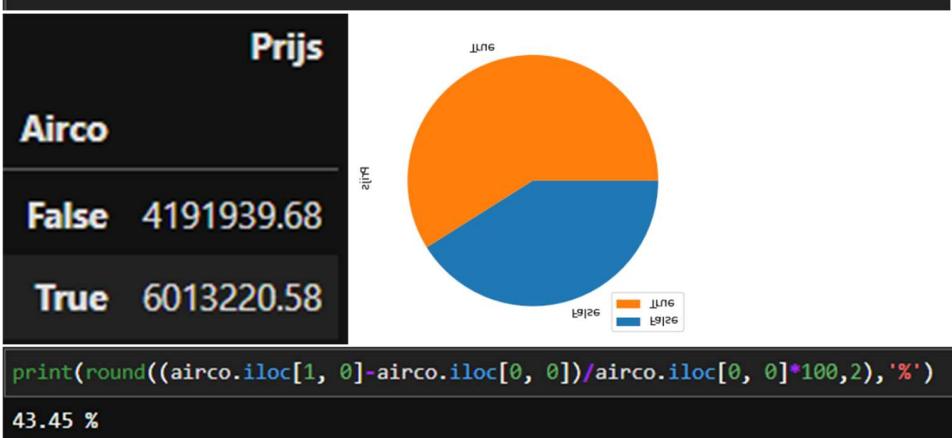
```
warmwaterverwarming= df.groupby(['Warm_Water_Verwarming'])['Prijs'].mean().round(2).to_frame()
warmwaterverwarming
```



De aanwezigheid van warm water verwarming is positief gecorreleerd met de huizenprijs. Huizen met warm water verwarming kosten gemiddeld 17,5 % meer dan de huizen zonder.

Factor 9: aanwezigheid airco

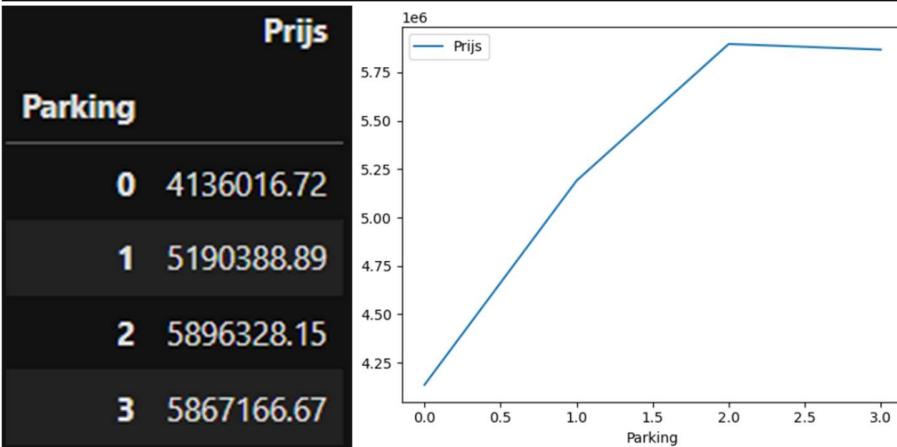
```
airco= df.groupby(['Airco'])['Prijs'].mean().round(2).to_frame()
airco
```



De aanwezigheid van airconditioning installatie heeft een heel significante invloed op de huizenprijs. Huizen met een airco installatie kosten gemiddeld 43% meer dan die zonder.

Factor 10: Aantal parkeerplaosten

```
parking= df.groupby(['Parking'])['Prijs'].mean().round(2).to_frame()  
parking
```

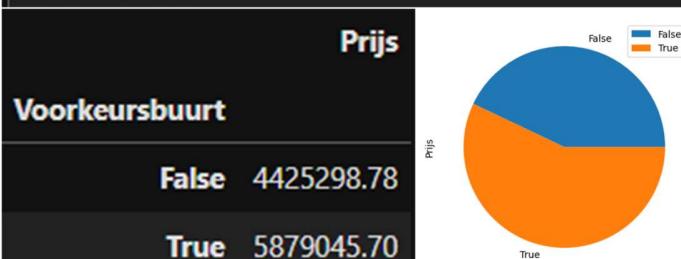


```
print(round(((parking.iloc[2,0]-df['Prijs'].mean())/df['Prijs'].mean())*100,2),'%')  
23.7 %
```

De grafiek toont aan dat het aantal parkeerplaatsen in het huis positief gecorreleerd is met de huizenprijs hoewel de impact stagneert vanaf 2 parkeerplaatsen. Huizen met 2/3 parkeerplaatsen zijn ongeveer 23% duurder dan het gemiddelde.

Factor 11: Voorkeursbuurt

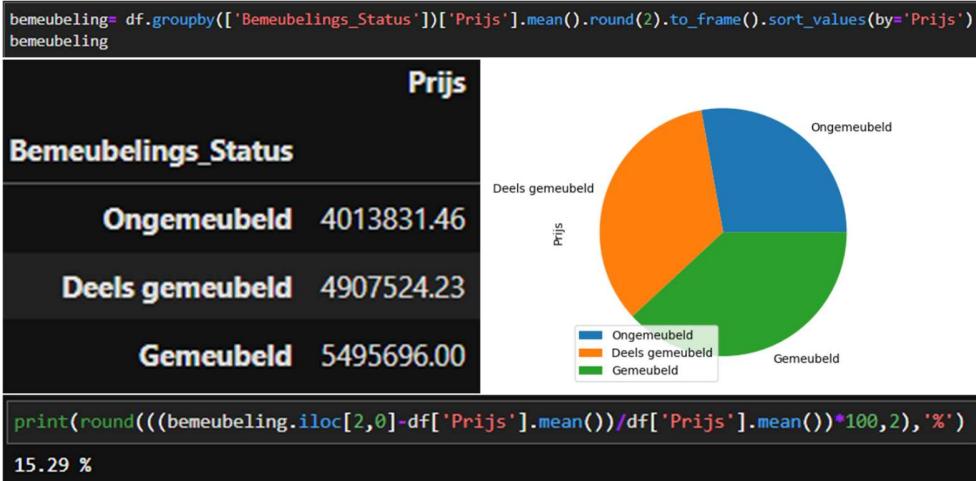
```
voorkeursbuurt= df.groupby(['Voorkeursbuurt'])['Prijs'].mean().round(2).to_frame()  
voorkeursbuurt
```



```
print(round((voorkeursbuurt.iloc[1, 0]-voorkeursbuurt.iloc[0, 0])/voorkeursbuurt.iloc[0, 0]*100,2),'%')  
32.85 %
```

De ligging van het huis speelt ook een significante rol in de huizenwaardering. Volgens deze dataset zijn huizen die gelegen zijn in een voorkeursbuurt gemiddeld 33% duurder dan huizen die daar niet gelegen zijn.

Factor 12: Bemeubeling



Hoe meer bemeubeld het huis is hoe duurder het is volgens deze dataset. Volledig bemeubelde huizen zijn 15% duurder dan de gemiddelde huizenprijs in deze dataset.

Exporteren

Ik heb mijn opgeschoonde dataframe tenslotte nog terug naar csv geëxporteerd zodat ik die in Power BI kon importeren om mooie visualisaties te maken.

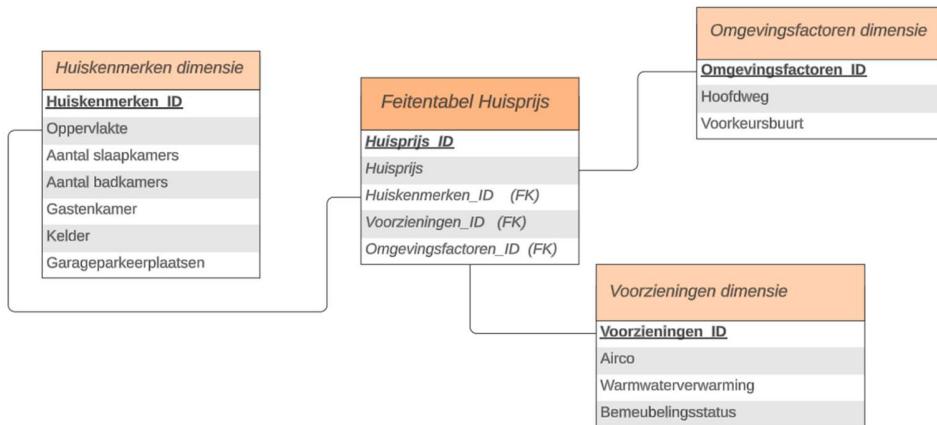
```
df.to_csv('dataset_3.csv',index=True)
```

Data visualisatie

Stap 1: Schematiseren

Om mijn model in Power BI te maken heb ik weer eerst een sterschema gemaakt in Lucidchart. Ik heb weer een huizenprijs feitentabel. Die is verbonden met de dimensies: huiskenmerken, omgevingsfactoren en voorzieningen.

Housing prices dataset



Stap 2: Modelleren in Power BI

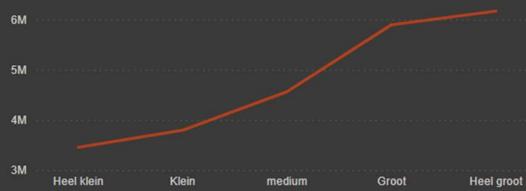
Ik heb net zoals bij de vorige 2 datasets mijn csv file in Power BI geladen en met power query mijn model gemaakt gebaseerd op mijn sterschema.

Stap 3: Visuals maken

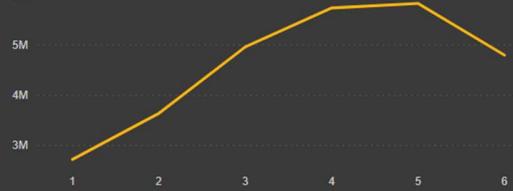
Huiskenmerken dimensie

De impact van huiskenmerken op de huisprijs

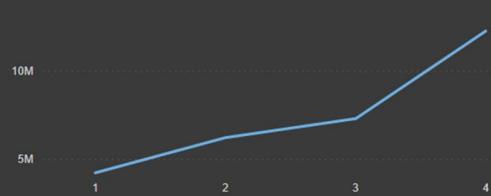
Gemiddelde huisprijs gebaseerd op huisgrootte



Gemiddelde huisprijs per aantal slaapkamers



Gemiddelde huisprijs per aantal badkamers

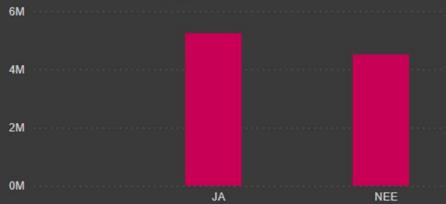


Gemiddelde huisprijs gebaseerd op aanwezigheid gastenkamer

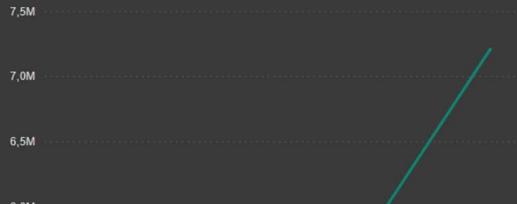


De impact van huiskenmerken op de huisprijs

Gemiddelde huisprijs gebaseerd op aanwezigheid kelder



Gemiddelde huisprijs per aantal verdiepingen



Gemiddelde huisprijs per aantal garage parkeerplaatsen



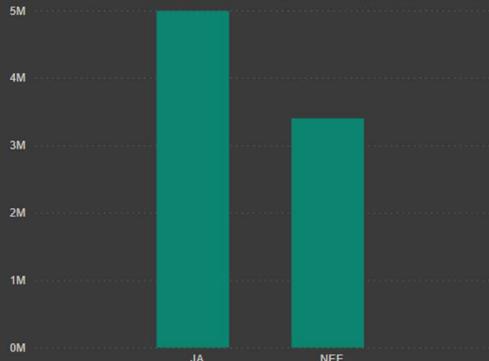
Gemiddelde huisprijs per aantal verdiepingen



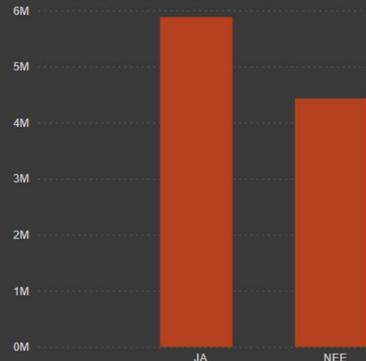
Omgevingsfactoren dimensie

De impact van omgevingsfactoren op de huisprijs

Gemiddelde huisprijs gebaseerd op aanwezigheid hoofdweg



Gemiddelde huisprijs voorkeursbuurt



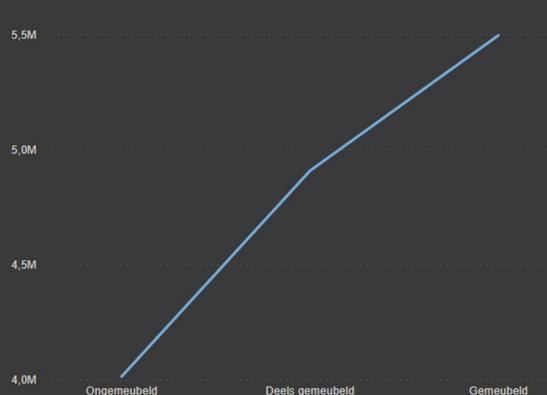
Voorzieningen dimensie

De impact van voorzieningen op de huisprijs

Gemiddelde huisprijs gebaseerd op aanwezigheid airco



Gemiddelde huisprijs gebaseerd op bemeubelingsstatus



Gemiddelde huisprijs gebaseerd op aanwezigheid warm water verwarming



Besluit

Nu ik 3 datasets geanalyseerd heb, weet ik dat er een aantal factoren zijn die consistent een grote impact hebben op de huizenprijs.

De factor die het sterkst geassocieerd was met hogere huizenprijzen was de locatie, meer bepaald de nabijheid van een zee/waterkant en het uitzicht vanuit het huis. De nabijheid van een oceaan was goed voor een stijging van 84% tegenover de gemiddelde huizenprijs in de eerste dataset. In de tweede dataset resulteerde de nabijheid van een waterkant in een stijging van 212% boven de gemiddelde huizenprijs.

Op de tweede plaats stond het aantal badkamers. De huizen met de meeste badkamers waren veel duurder dan de gemiddelde woning in de datasets. De eerste keer dat het aantal badkamers aan bod kwam als factor was het goed voor een stijging van 63% tegenover de gemiddelde huizenprijs. De tweede keer waren de huizen met het hoogst aantal badkamers zelfs 157% duurder dan gemiddeld.

Het inkomen van de bewoners was ook een heel significante factor. Huizen in buurten met het hoogste inkomen waren 62% duurder.

Huizen met een hoog aantal verdiepingen waren 51% duurder dan gemiddeld en huizen gelegen aan een hoofdweg waren 47% duurder.

In beperktere mate waren de rest van de voorzieningen allemaal positief gecorreleerd met hogere huizenprijzen.

Huisouderdom had maar een kleine impact op de huizenprijs de eerste keer dat het aan bod kwam en de tweede keer was er zelfs geen duidelijk verband af te leiden.

Bevolkingsdichtheid was de enige factor die een negatieve impact had op de huizenprijs, hoewel het hier maar over een verschil van 3% ging, wat verwaarloosbaar is.

Bron

[1] Kaggle. (n.d.). *California Housing Prices* - Median house prices for California districts derived from the 1990 census. van <https://www.kaggle.com/c/titanic>

[2] Kaggle. (n.d.). *House Sales in King County, USA* - Predict house price using regression. van <https://www.kaggle.com/datasets/harlfoxem/housesalesprediction>

[3] Kaggle (n.d.). *Housing Prices Dataset* - Housing Prices Prediction/ Regression Problem. van <https://www.kaggle.com/datasets/yasserh/housing-prices-dataset>

