Pandas: Tabellen combineren

Dr. Cornelis Stal

April 28, 2022

[ ]:
```
import pandas as pd
```

1 Tabellen combineren

Deze tutorial is een vertaling van de *Pandas Tutorial* op https://pandas.pydata.org/pandas-docs/stable/getting_started/.

Data: voor deze tutorial zullen we gebruikmaken van de jaarlijkse vastgoedcijfers die bijgehouden en beschikbaar gemaakt worden door Statbel via deze link. We richten ons hierbij meer bepaald op de cijfers van verkoop van onroerende goederen (N) per jaar 2010-2021 voor de individuele gemeenten. Aanvullend maken we gebruik van de combinatie van de Vlaamse gemeentegrenzen, waarvoor enkel de attributen geëxporteerd zijn uit de gehele dataset. De oorspronkelijke data zijn te vinden op geopunt. Aan iedere gemeente in deze dataset is eveneens het aantal inwoners toegekend, zoals beschikbaar gesteld door StatBel. Voor een omschrijving van de verschillende velden verwijzen we naar de tutorial over data lezen en schrijven met Pandas.

In deze tutorial zullen we de hierboven omschreven data met elkaar combineren. Het is hierbij de bedoeling om voor iedere Vlaamse gemeente het de mediaanprijs van residentieel vastgoed te vergelijken met de oppervlakte en het aantal inwoners van iedere gemeente.

Klik hier om de data met vastgoedgegevens te downloaden.

Klik hier om de data te downloaden met het aantal inwoners en oppervlakte van de Vlaamse gemeentes.

1.1 Data voorbereiden

De data met Belgische vastgoedgegevens bereiden we voor zoals we in eerdere tutorials ook gedaan hebben:

[ ]:
```
statbel = 'https://statbel.fgov.be/sites/default/files/files/\
    documents/Bouwen%20%26%20wonen/2.1%20Vastgoedprijzen/NL_immo_jaar.xlsx'

# Data importeren met selectie van de gewenste velden
vastgoed = pd.read_excel(statbel, sheet_name='Per gemeente', skiprows=2,
    usecols=['refnis', 'lokaliteit', 'jaar', 'aantal transacties',
    'mediaan prijs(€)', 'eerste kwartiel prijs(€)','derde kwartiel prijs(€)'])
# Kolomhoofdingen hernoemen
vastgoed = vastgoed.rename(
```

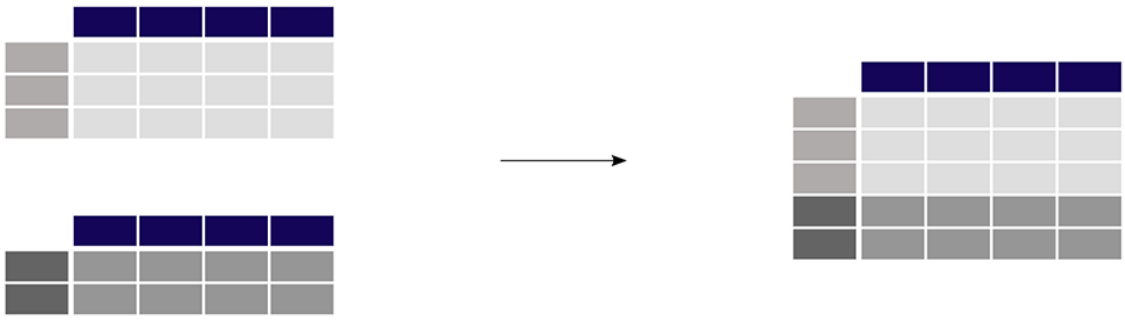
```
    columns={'refnis': 'NIS', 'lokaliteit': 'NAAM', 'jaar': 'JAAR',
    'aantal transacties': 'AANTAL', 'mediaan prijs(€)': 'MEDIAAN',
    'eerste kwartiel prijs(€)': 'Q1', 'derde kwartiel prijs(€)': 'Q3'},
    errors="raise"
)
# We zetten enkel de eerste letter van iedere gemeente in hoofdletters
vastgoed['NAAM'] = vastgoed['NAAM'].str.capitalize()
vastgoed.head()
```

De dataset met Vlaamse gemeenten kunnen we gebruiken zoals deze aangeboden worden:

[ ]:
```
refgem = pd.read_csv("data/refgem.csv")
refgem.head()
```

2 Tabellen samenvoegen

De meest eenvoudige manier om data samen te voegen is simpelweg de rijen uit een bepaalde dataset toe voegen aan een reeds bestaande tabel.



Stel nu dat de data met Vlaamse gemeenten niet als één dataset geïmporteerd zou zijn, maar als afzonderlijke datasets per provincie. Om dit te reproduceren selecteren bijvoorbeeld alle gemeenten in Oost-Vlaanderen op basis van het veelvoud van 10000 van NIS-code, startend met een waarde 4:

[ ]:
```
oostVlaanderen = refgem[refgem['NISCODE'] // 10000 == 4]
oostVlaanderen = oostVlaanderen.assign(PROV='Oost Vlaanderen')
oostVlaanderen.head()
```

Hetzelfde doen we nu voor de provincie West-Vlaanderen, waar alle NIS-codes starten met een waarde 3:

[ ]:
```
westVlaanderen = refgem[refgem['NISCODE'] // 10000 == 3]
westVlaanderen = westVlaanderen.assign(PROV='West Vlaanderen')
westVlaanderen.head()
```

Deze twee tabelen kunnen we eenvoudig samenbrengen tot een nieuwe tabel:

```
[ ]: vlaanderen = pd.concat([oostVlaanderen, westVlaanderen], axis=0)
     vlaanderen.head()
```

De `concat()`-functie hebben we hier gebruikt om de gewenste operatie uit te voeren. Als attribuut geven we deze functie een lijst mee met verschillende samen te voegen `DataFrames`. Daarnaast wordt een as gedefiniëerd waarmee de richting van de samenvoeging wordt vastgelegd. Dit kan zowel met het samenvoegen in functie van rijen (`axis=0`) als kolommen (`axis=1`). Wanneer er geen `axis`-waarde meegegeven wordt, verloopt de samenvoeging standaard volgens de opeenvolgende rijen, dus met `axis=0`.

Laten we de vorm of 'shape' van de verschillende originele tabellen en de nieuwe tabel eens bekijken om het resultaat te controleren:

```
[ ]: print('Vorm van de tabel ``oostVlaanderen``: ', oostVlaanderen.shape)
```

```
[ ]: print('Vorm van de tabel ``westVlaanderen``: ', westVlaanderen.shape)
```

```
[ ]: print('Vorm van de resulterende tabel ``vastgoedVlaanderen``: '
           , vlaanderen.shape)
```

Zoals we hierboven kunnen zien, beschikt de nieuwe tabel `vlaanderen` over $60 + 64 = 124$ rijen.

> **Opmerking:** het `axis`-argument bepaald in welke richting of over welke as een bepaalde operatie uitgevoerd moet worden. Een `DataFrame`-object heeft twee bijbehorende assen: de eerste doorloopt operaties in verticale richting, dus over de verschillende rijen (`axis=0`). Bij de tweede as worden operaties in horizontale richting overlopen, dus over de verschillende kolommen (`axis=1`). De meeste operaties, zoals het samenvoegen van `DataFrame`-objecten of het berekenen van statistieken, zal standaard verlopen over de verschillende rijen (`axis=0`), maar kunnen eveneens op de kolommen toegepast worden.

Het nut van deze operatie kunnen we illustreren door bijvoorbeeld geïnteresseerd te zijn in de 5 grootste gemeenten in Oost- en West-Vlaanderen. Als we de gecombineerde tabel sorteren in functie van het veld `OPPERVL_HA` in afnemende volgorde, krijgen we het volgende resultaat:

```
[ ]: vlaanderen = vlaanderen.sort_values("OPPERVL_HA", ascending=False)
     vlaanderen[['NAAM', 'PROV', 'OPPERVL_HA']].head(10)
```

> **Opmerking:** om het bovenstaande resultaat te krijgen, had het natuurlijk veel eenvoudiger geweest om de de volgende code te gebruiken:
>
> ```
> x = refgem[(refgem['NISCODE'] // 10000 == 3) | (refgem['NISCODE']
> // 10000 == 4)] prov = (refgem['NISCODE'] // 10000).replace({ 3:
> 'West-Vlaanderen', 4: 'Oost-Vlaanderen'}) x = x.assign(PROV=prov)
> ```
>
> Om het concept van de `concat()`-functie toe te lichten hebben we deze methode gekozen.

In dit specifieke voorbeeld kan onder meer de `NISCODE`-kolom gebruikt worden om iedere entiteit uit de oorspronkelijke tabellen te reconstrueren. Dit is echter niet altijd het geval, en we kunnen dan gebruik maken van he t`key`-argument, waarmee we aanvullende hiërarchische indexering toe kunnen voegen aan de samengevoegde tabel:

[ ]:
```
vlaanderen = pd.concat([oostVlaanderen, westVlaanderen],
    keys=["OOST", "WEST"])
vlaanderen.head()
```

> **Opmerking:** het gelijktijdig bestaan van meerdere rij- of kolomindices the existence of multiple row/column indices zal niet worden behandeld in deze tutorial. Hiërarchische indexering (*Hierarchical indexing* ofwel `MultiIndex`) is een geavanceerde en krachtige functionaliteit van `pandas` of data met hogere dimentionele complexiteit te analyseren. Hoewel hiërarchische indexering buiten de doelstellingen van deze demo ligt, is het belangrijk te onthouden dat de functie `reset_index()` bestaat. Hiermee kan ieder dimentioneel niveau van een index worden toegevoegd aan een kolom, bijvoorbeeld door `vlaanderen.reset_index(level=0)`.
>
> **Gebruikshandleiding:** meer informatie over hiërarchische indexering kan worden teruggevonden in de sectie over geavanceerde indexering in de handleiding.
>
> **Gebruikshandleiding:** er kunnen meer opties gedefinieerd worden om de werking van de `concat()`-functie te bepalen. Zo kunnen we vastleggen of er sprake moet zijn van een rij- of kolomgebaseerde unie of intersectie. Meer informatie hierover kan worden teruggevonden in de sectie over het samenvoegen van objecten.

De initiele opzet van deze tutorial is nog steeds het combineren van Belgische vastgoeddata aan de Vlaamse gemeenten. We kunnen een kijken wat er gebeurt als we de de twee `DataFrame`-objecten `vastgoed` en `refgem` gebruiken met de `concat()`-functie:

[ ]:
```
vlaamsVastgoed = pd.concat([refgem, vastgoed], axis=0)
vlaamsVastgoed.head()
```

Het zal duidelijk zijn dat dit niet het gewenste resultaat geeft. We gaan daarom verder met het combineren van data, maar nu op basis van gemeenschappelijke velden.

2.1 Combineren op basis van gemeenschappelijke waarden

Laten we nu inderdaad eens de data met informatie over gemeenten combineren met de vastgoedprijzen op basis van een gemeenschappelijk veld, zodat we per gemeente de mediaanprijzen en dergelijke kunnen achterhalen en verdere berekeningen uit kunnen voeren:

[ ]:
```
vlaamsVastgoed = pd.merge(refgem, vastgoed, how="outer",
    left_on="NISCODE", right_on="NIS")
vlaamsVastgoed.head()
vlaamsVastgoed[(vlaamsVastgoed['NAAM_y'] == 'Gent') |
    (vlaamsVastgoed['NAAM_y'] == 'Brussel') |
    (vlaamsVastgoed['NAAM_y'] == 'Luik')]
```

Met behulp van de `merge()`-functie wordt voor ieder rij in de tabel `refgem` de corresponderende rij gezocht in de tabel `vastgoed`. De velden die voor deze koppeling gemaakt wordt zijn respectievelijk `NISCODE` en `NIS`. Merk op dat de `merge()`-functie beide tabellen in zijn geheel heeft willen samenvoegen, ondanks het feit dat Brusselse en Waalse gemeenten niet in de tabel `refgem` voorkomen. De `merge()`-functie ondersteunt meerdere join-opties, zoals we deze ook tegen zouden komen bij andere database-software. Deze functionaliteit, en meer bepaald het type combinaties kunnen we veranderen om enkel Vlaamse gemeenten in ons resultaat op te nemen.

> **Opmerking:** in dit voorbeeld kiezen we er expliciet voor om de data te combineren met behulp van de NIS-code. Intuïtief zouden we misschien gekozen hebben voor de plaatsnaam van iedere gemeente. Vermits de koppeling plaatsvindt op exact dezelfde waarden, zou dit een slechter keuze zijn, aangezien plaatsnamen niet altijd op dezelfde wijze geschreven worden, in tegenstelling tot de NIS-code.
>
> **Gebruikshandleiding:** naar analogie met veel andere database-systemen, zopals Postgresql, ondersteunt `pandas` eveneens 'inner'-, 'outer'- en 'right'-joins. Voor meer informatie over deze technieken om data te combineren verwijzen we naar de sectie 'database style merging of tables' in de handleiding. De vergelijking tussen `pandas` en SQL wordt gemaakt in de 'comparison with SQL'-pagina.

[ ]:
```
vlaamsVastgoed = pd.merge(refgem, vastgoed, how="left",
    left_on="NISCODE", right_on="NIS")
vlaamsVastgoed.head()
vlaamsVastgoed[(vlaamsVastgoed['NAAM_y'] == 'Gent') |
    (vlaamsVastgoed['NAAM_y'] == 'Brussel') |
    (vlaamsVastgoed['NAAM_y'] == 'Luik')]
```

In vergelijking met het voorgaande voorbeeld hebben we nu enkel informatie over gemeenten in Vlaanderen. Door een `left`-join te kiezen worden enkel elementen uit de linker tabel (`refgem`) genomen. Gemeenten in Brussel en Wallonië zullen in de resulterende tabel niet vermeld worden. Gemeenten in Brussel en Wallonië komen niet voor in de linker tabel (`refgem`) en worden bijgevolg niet meegenomen in de combinatie.

2.2 Te onthouden:

- Met behulp van de `concat()`-code kunnen meerdere tabellen kunnen samengevoegd worden. Deze operatie kunnen we zowel kolom-per-kolom als rij-per-rij toepassen;
- Voor database-achtige combinaties van tabellen (zoals *merging* of *joining*) gebruiken we de `merge()`-functie.

  Gebruikshandleiding: we verwijzen naar de sectie over 'facilities to combine data tables' in de handleiding voor een volledige omschrijving van over verschillende mogelijkheden om tabellen te combineren.