**Andmete “koorimine” kv.ee-st**

Daniel Henri Trump

**Lahendatav probleem**

Luua ülevaatlik kaart üürikorterite hindadest Tartus KV.ee andmete põhjal. Lisaks soovisin teada, kas korteri pindala ja asukoha vahel on mingi seos.

**Ettevalmistus**

Vaja läks Pythonit ja ka Micromambat.

Mõistlik on ka luua uus Pythoni environment, selleks et vältida oma teiste Pythoni enviromentide rikkumist.

Pythoni lisad, mida kasutasin tööks:

playwright

ipykernel

jupyterlab

folium

PowerShellis Pythoni lisade installimine:

# uue enviromenti loomine ja lisade installimine

micromamba create -n envnimi python=3.11

micromamba activate envnimi

pip install playwright

pip install jupyterlab

pip install ipykernel

Pip install folium

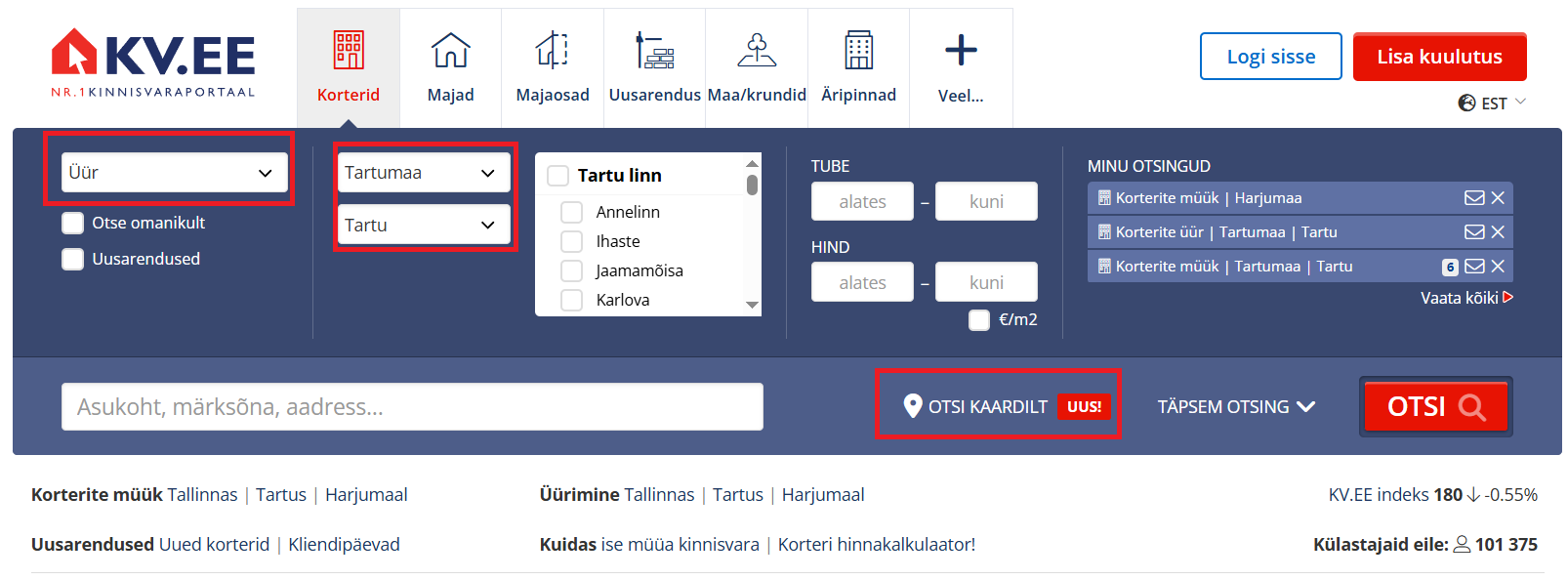
# uue kerneli loomine

python -m ipykernel install --user --name envnimi --display-name "envnimi"

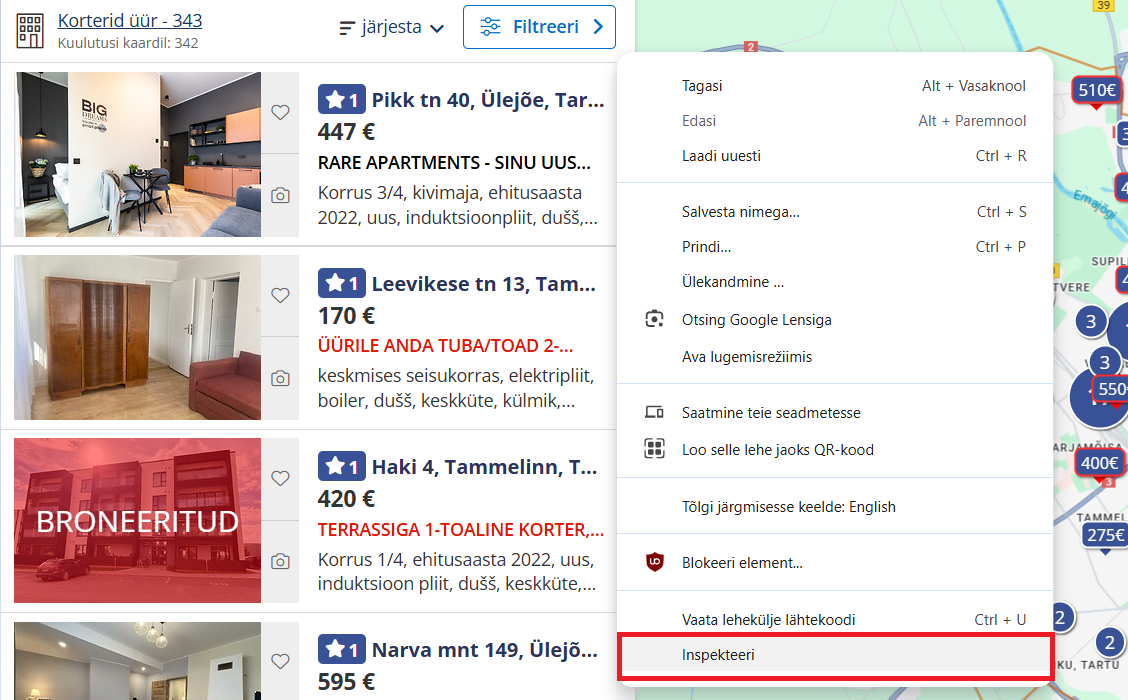
Nüüd peaks Pythoni keskkond olema tööks valmis.

**KV.ee leheküljelt info saamine**

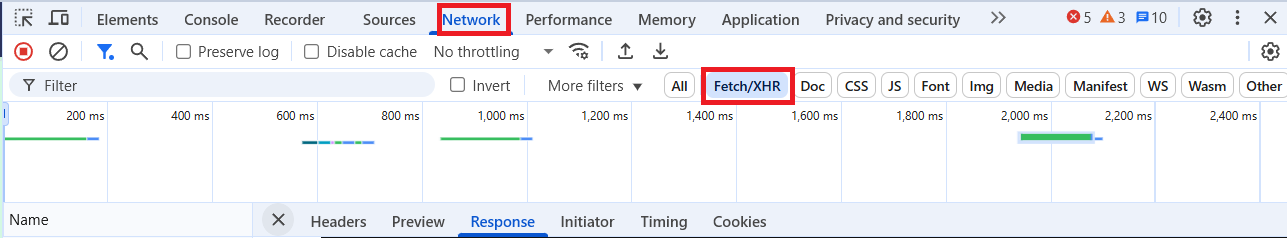
Valisin kinnisvara liigiks üür ja piirkonnaks Tartumaa, Tartu, sest siis saan Tartu linna üürikorterid. Valik ,,Otsi kaardilt’’ avab vaate koos filtritega kaardist.

****

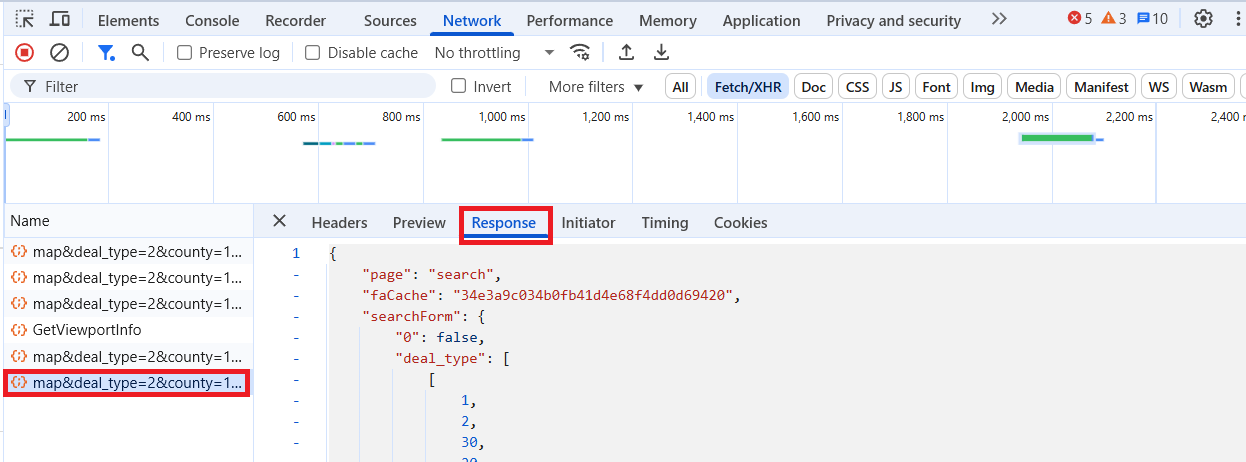
Avanenud kaardiaknas tuleb teha paremklikk ja valida **inspekteeri.**



Pärast seda avaneb “DevTools” aken. Sealt tuleb valida **network,** pärast seda **Fetch/XHR**.



Pärast seda näeb päringuid ja siis tuleks valida päring, mis näeb välja nagu **“map&deal\_type=2…..”** ja siis tuleks valida ka **response**. Seda tehes näeme päringu tulemust .json formaadis. **NB! Kõige alumine päring on kõige värskem.**

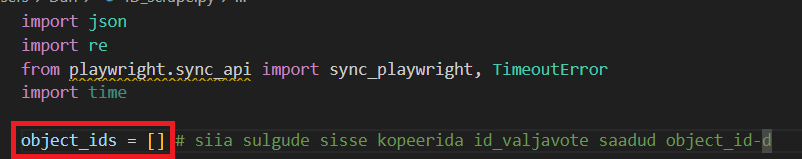


Selleks, et päring leheküljelt kätte saada, on mitu võimalust, millest mina valisin manuaalse variandi. Kui vajutada paremal all oleva teksti pihta ning siis klahvikombinatsiooni crtl + a, valitakse kogu tekst, mis lehekülg on vastuseks saatnud. See tekst tuleks kopeerida värskelt loodud .json faili.

*Kirjutasin algselt ka koodi, et sama informatsioon automaatselt .json faili saada, aga kuna tegemist pole täielikult automatiseeritud töövooga, siis leian, et kopeerides on lihtsam.*

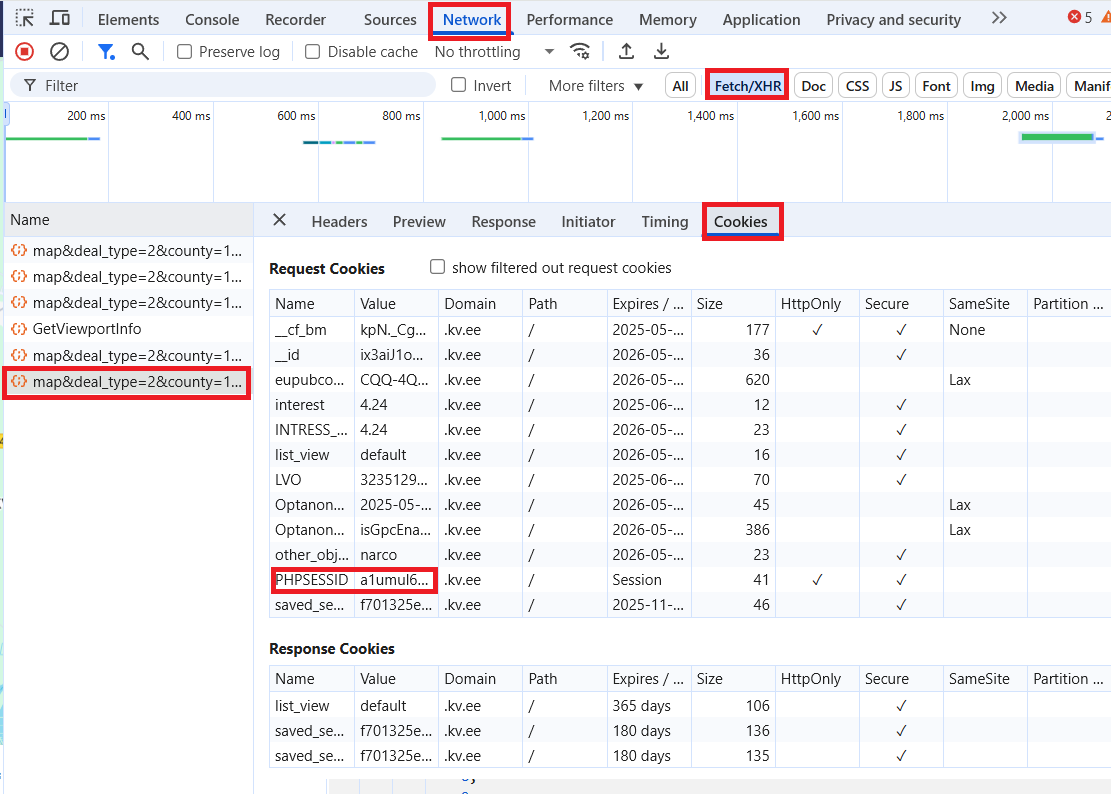
Edasi tuleks avada JupyterLab notebook **“id\_valjavote”,** selleks et .json failist kätte saada kõik kaardil olnud ID-d. Põhjus, miks andmeid otse json failist kätte ei saa, on see, et andmed on klastrite põhjal grupeeritud.

Pärast unikaalsete ID-d leidmist tuleks need ID-d kopeerida **ID\_scrape.py** skripti. Väärtused tuleb kopeerida “**object-ids = []**” kantsulgude vahele.

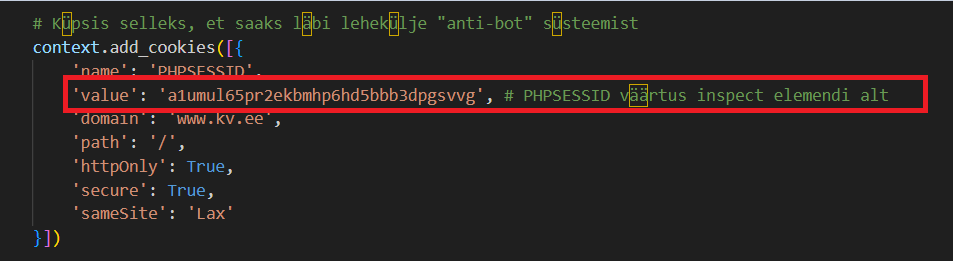


Peale seda tuleks minna tagasi KV.ee lehe **“DevTools”** juurde, et saada andmete koorimiseks vajalik unikaalne küpsise kood.  
**Network → Fetch/XHR → map&…..(päring) → Cookies → PHPSESSID**

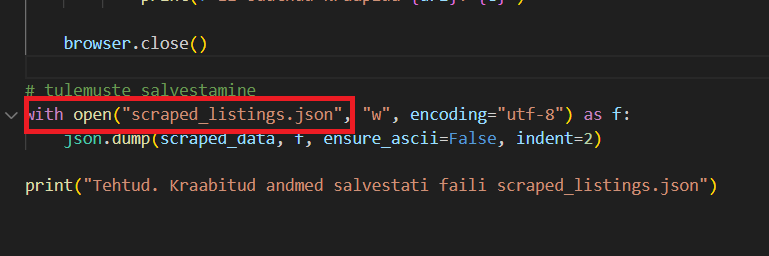
**PHPSESSID** kõrval oleva **“value”** peale tuleb vajutada topeltklikk ja siis tuleb see väärtus kopeerida.



Väärtus tuleb kopeerida **ID\_scrape.py** scripti sisse **‘value’** väärtuse asemele



Pärast seda tuleks skript salvestada selleks, et muutused säiliksid.



Tulemuste salvestamise asukohta saab muuta skripti lõpust.

Edasiseks tegevuseks on vaja sulgeda **JupyterLab ja PowerShell**. Pärast seda tuleks **PowerShellis** uuesti käivitada oma python eviroment ja käivitada **PowerShellis ID\_scrape.py** skript. Skripti ei saa windowsis otse pythonis käivitada.

See on oluline, kuna skript emuleerib veebilehe külastamist info kogumiseks.

PowerShell:

python ID\_scrape.py

Pärast skripti käivitamist hakkab skript ükshaaval läbi käima kõiki KV.ee objekte (kortereid), mille ID-d skripti sisestatud said. See võtab natuke aega olenevalt objektide hulgast.

**Kui andmed on kraabitud annab PowerShell teavituse:**“Tehtud. Kraabitud andmed salvestati faili ……”

**Andmete korrastamine**

Objektide info salvestatakse eraldi .json faili (näiteks “scraped\_listings.json”), mis tuleks muuta .csv failiks selleks, et seda GIS-is edasi töödelda.

Seda tegin skriptiga **json\_to\_csv.py.**

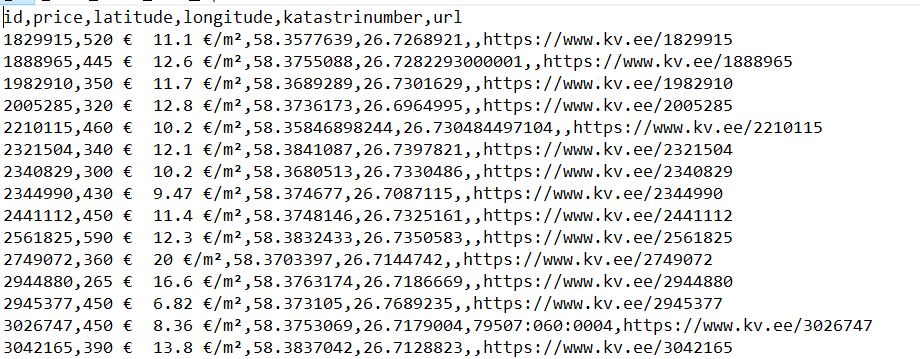
Punasega märgitud koodis saab muuta sissevõetavat faili.

Sinisega märgitud koodis saab muuta .csv faili nime.

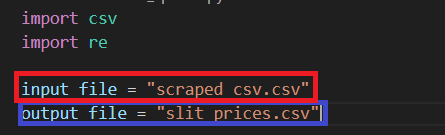
Seda skripti ei pea käivitama PowerShellis.

****

Tulemuseks sain .csv faili. Failiga on aga probleem. **“Price”** väärtuste all on lisaks hinnale ka hind ruutmeetri kohta.



Selle probleemi lahendamiseks on skript **csv\_splitter.py**. Skript loob eraldi välja ruutmeetrihinna jaoks. Sisendfaili saab muuta punasega märgitud koodilõigus ja sinisega märgitud lõigus saab muuta väljundfaili.

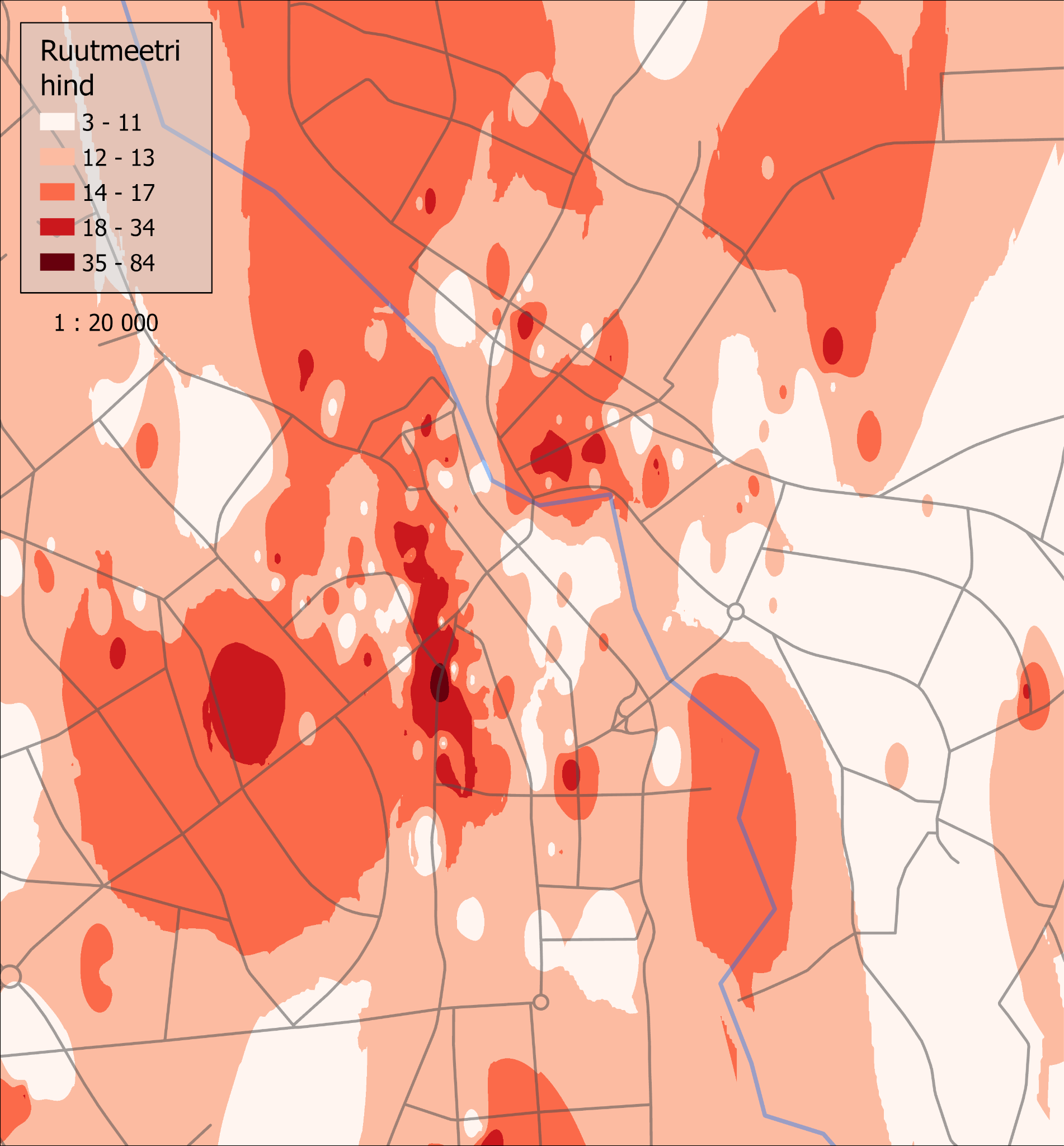


csv\_splitter.py skripti väljund fail on nii öelda lõpplik fail. Seda kasutan edasistes etappides.

**ArcGIS Pro**

Selleks, et andmed kaardile kuvada, peaks kasutama tööriista XY Table to Point.

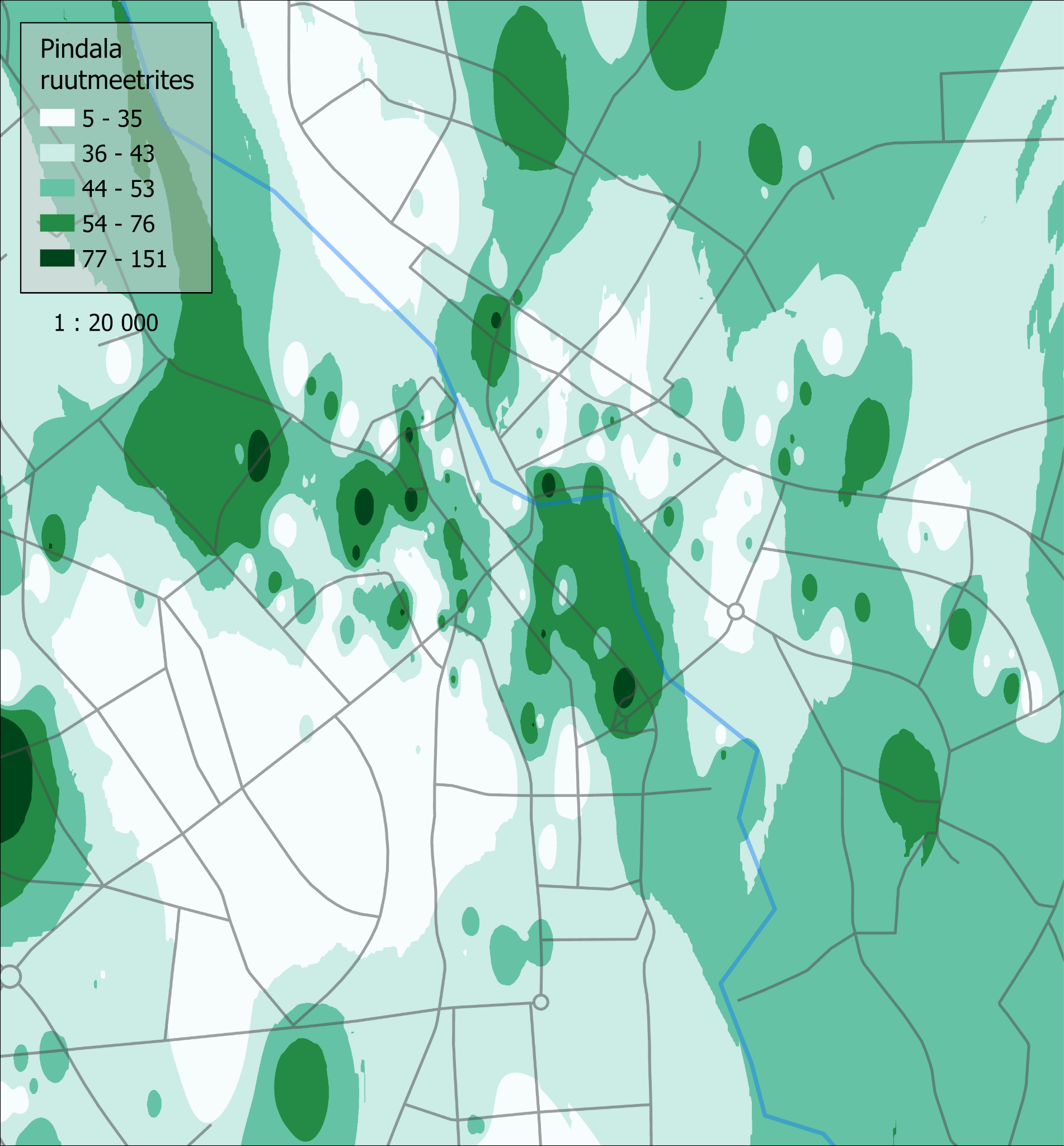
Loodud punktide põhjal interpoleerisin punktidest rastri, kasutades IDW (Inverse distance weighting) meetodit. Kasutasin seda meetodit, kuna ArcGIS Pro jooksis teisi meetodeid kasutades millegipärast kokku. Interpoleerides kasutasin Z-väärtusena ruutmeetri hinda, kuna see annab parema ülevaate kui ainult hind.



Joonis 1. Ruutmeetri hinna jaotumine ruumis

Samuti saaks andmete põhjal vaadata, kas üürikorterite pindala on kuidagi ruumiliselt jaotunud.

Selleks peab tegema uue pindala välja kuhu peab arvutama üürikorteri pindala. Seda saab teha jagades korteri üldhinna ruutmeetrihinnaga.

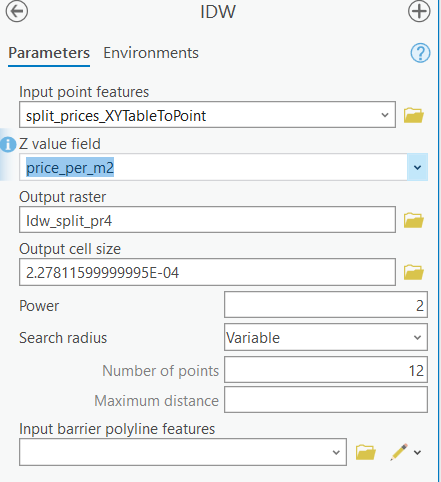


Joonis 2. Üürikorterite pindalade jaotumine ruumis



Joonis 3. Üürikorterite jaotumine ruumis

IDW parameetrid:



Mõlema kaardi tegemiseks kasutasin “Natural Breaks” meetodit ning mõlema kaardil on andmed jaotatud viite klassi.

**Võrgustiku analüüs**

Võrgustiku analüüsi tegemiseks kasutasin “Ruumiandmete analüüs Pythoni ja R-iga” aines loodud pythoni environmenti “geopython”. Skript koostati JupyterLab-is.

Skripti nimi: vorgustikuanaluus.ipynb.

Pythoni lisad, mida kasutasin tööks:

pandas

geopandas

osmnx

networkx

shapely

matplotlib

tqdm

Kasutasin **OpenStreetMap-i** (OSM) kaardistatud Tartu linna teede võrku ja teenuste punkte selleks, et hinnata, kas üürihind sõltub sellest, kui lähedal korterile on mingid teenused. Osa koodist on varasemast projektist võetud.

Kauguste arvutamiseks kasutasin **Dijkstra algorütmi**.

OSM- i teenused grupeerisin liikideks nii:

"food": ["restaurant", "cafe", "fast\_food", "bar", "pub", "ice\_cream"],

"health": ["hospital", "clinic", "doctors", "dentist", "pharmacy", "veterinary"],

"education": ["school", "kindergarten", "university", "college", "library"],

"transport": ["bus\_station", "taxi", "parking", "bicycle\_parking", "car\_rental"],

"services": ["bank", "atm", "post\_office", "hairdresser", "beauty"],

"public": ["townhall", "police", "fire\_station", "courthouse", "prison"],

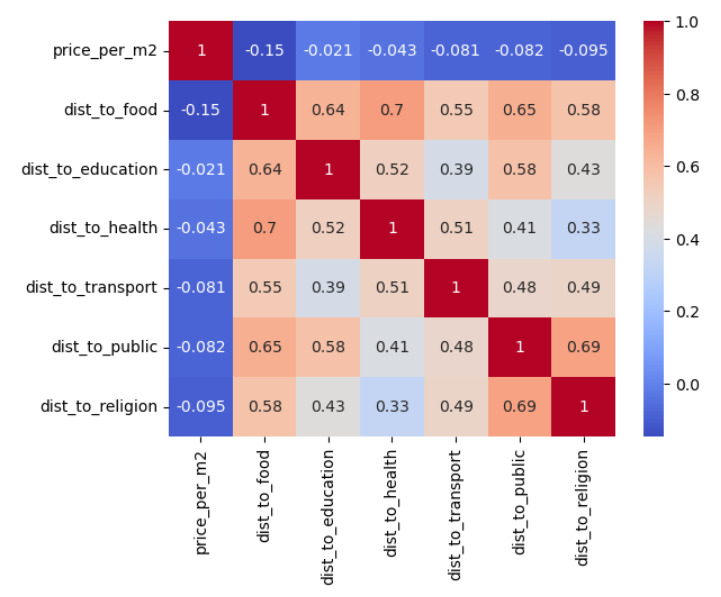
"leisure": ["cinema", "theatre", "community\_centre", "arts\_centre"],

"religion": ["place\_of\_worship", "grave\_yard"]

Arvutatud kaugused lisasin nimedele vastavatesse veergudesse:

'dist\_to\_food', 'dist\_to\_education', 'dist\_to\_health', 'dist\_to\_transport', 'dist\_to\_public', 'dist\_to\_religion', 'dist\_to\_services'

Nende veergude põhjal arvutasin korrelatsiooni maatriksi selleks, et hinnata seoseid ja tulemuste loogilisust.



Joonis 4. Korrelatsioonimaatriks