Primjena web scraping tehnika za prikupljanje i obradu podataka s platformi za posredovanje u prometu nekretninama

Kancijanić, Aljoša

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

ALJOŠA KANCIJANIĆ

PRIMJENA WEB SCRAPING TEHNIKA ZA PRIKUPLJANJE I OBRADU PODATAKA S PLATFORMI ZA POSREDOVANJE U PROMETU NEKRETNINAMA

Diplomski rad

Pula, srpanj, 2023. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

ALJOŠA KANCIJANIĆ

PRIMJENA WEB SCRAPING TEHNIKA ZA PRIKUPLJANJE I OBRADU PODATAKA S PLATFORMI ZA POSREDOVANJE U PROMETU NEKRETNINAMA

Diplomski rad

JMBAG: 0246058749, redoviti student
Studijski smjer: Informatika
Predmet: IT Management
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijsko i programsko inženjerstvo

Mentor: izv. prof. dr. sc. Darko Etinger

Pula, srpanj, 2023. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Aljoša Kancijanić, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, srpanj, 2023. godine

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Aljoša Kancijanić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom „Primjena Web Scraping tehnika za prikupljanje i obradu podataka s platformi za posredovanje u prometu nekretninama“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, srpanj, 2023. godine

Potpis

_____

Sadržaj

1. Uvod

S obzirom na današnji brzi napredak digitalne ekonomije, sposobnost efikasnog prikupljanja i obrade podataka postala je ključni faktor za donošenje informiranih odluka u svakom poslovnom okruženju a i u privatnom životu. Kroz procese prikupljanja, obrade i analize podataka, otvaramo vrata za dublje razumijevanje fenomena, otkrivanje uzoraka, predviđanje trendova i donošenje informiranih odluka. Međutim, prikupljanje tih podataka može biti izazovno, posebno s obzirom na velike količine informacija dostupnih na internetu.

Web scraping je praksa prikupljanja podataka putem bilo kojeg sredstva osim programa koji komunicira s API sučeljem (ili, naravno, kroz ljudsku interakciju s web preglednikom). Ovo se najčešće postiže pisanjem automatiziranog programa koji komunicira s web poslužiteljem, zahtijeva podatke (obično u obliku HTML-a i drugih datoteka koje čine web stranice), a zatim analizira te podatke kako bi izvukao potrebne informacije (Mitchell, 2018).

U ovom diplomskom radu, istražit ćemo upotrebu web scraping tehnika u kontekstu prikupljanja i obrade podataka s platformi za posredovanje u prometu nekretninama. Posebnu pažnju posvetit ćemo analizi Njuškalo.hr, jedne od najpopularnijih web platformi za oglašavanje u Hrvatskoj. Cilj je detaljno proučiti i primijeniti web scraping tehnike na ovom specifičnom primjeru kako bismo dobili uvide u tržište nekretnina.

Kako bi to postigli, koristit ćemo Scrapy okvir, alat za izvlačenje podataka s web stranica, koji nam omogućuje da prolazimo kroz web stranice, prikupljamo potrebne podatke i obrađujemo ih za daljnju analizu. Kao rezultat ovog procesa, prikupljeni podaci bit će pohranjeni u privremenu bazu podataka.

Za daljnju analizu ovih podataka iskoristiti ćemo Matplotlib i Seaborn, biblioteke za vizualizaciju podataka u programskom jeziku Python. One pružaju širok raspon mogućnosti za stvaranje grafova, dijagrama i drugih vrsta vizualnih prikaza. Ove biblioteke će nam omogućiti dublji uvid u prikupljene podatke.

Cilj ovog rada je istražiti potencijal web scraping-a kao tehnike za učinkovito prikupljanje podataka s platformi za posredovanje u prometu nekretninama. Analizirat ćemo kako se prikupljeni podaci mogu koristiti za dobivanje uvida u samo tržište nekretnina i prikazati dobivene rezultate.

2. Prikupljanje podataka

Prikupljanje podataka igra ključnu ulogu u gotovo svakoj industriji. Podaci su postali dragocjen resurs koji omogućuje donošenje informiranih odluka, oblikovanje strategija i poticanje inovacija. Obrađeni podaci, informacije, korisni su za poboljšanje procesa i donošenje odluka. Formalni proces prikupljanja podataka je nužan jer osigurava da prikupljeni podaci budu jasno definirani i točni te da su kasnije odluke temeljene na analizi podataka valjane (Sapsford, 2006). Bez obzira na industriju ili poslovnu funkciju, prikupljanje podataka predstavlja temeljni korak u procesu analize i interpretacije podataka.

U ovom poglavlju posvetit ćemo se različitim metodama prikupljanja podataka koje su dostupne istraživačima. Važno je identificirati relevantne izvore podataka, skupiti podatke iz tih izvora te pripremiti podatke za daljnju obradu i analizu. Metode prikupljanja podataka variraju ovisno o prirodi podataka, dostupnosti izvora te specifičnim potrebama i ciljevima istraživanja. Istražit ćemo različite tehnike prikupljanja podataka, uključujući ručno prikupljanje podataka, korištenje API-a (engl. Application Programming Interface), gotove skupove podataka te web scraping. Svaka od ovih tehnika pruža svoje prednosti i izazove, a njihova primjena može ovisiti o kontekstu istraživanja i specifičnim zahtjevima projekta.

Prikupljanje podataka je ključno za informirano donošenje odluka i napredak u suvremenom poslovanju. U nastavku poglavlja, istražit ćemo svaku od navedenih metoda prikupljanja podataka kako bismo stvorili jasnu sliku o njihovoj primjeni i koristima.

2.1. Ručno prikupljanje podataka

Ručno prikupljanje podataka je tradicionalna metoda koja uključuje direktno prikupljanje podataka putem istraživanja, upitnika, anketa ili drugih izvora. Ova tehnika može biti korisna iz nekoliko razloga, ali također nosi i određene prednosti i nedostatke.

Ručno prikupljanje podataka omogućuje veću kontrolu kvalitete prikupljenih podataka. Ljudi koji vrše prikupljanje podataka mogu koristiti svoju osobnu procjenu nad podacima, provjeriti informacije i ispraviti pogreške ili nepodudarnosti odmah. To može osigurati točnost i pouzdanost skupa podataka. Ovaj način prikupljanja podataka često je prikladniji za složene podataka koje nije lako automatizirati. To uključuje prikupljanje podataka putem intervjua, anketa ili promatranja, gdje je ljudska interakcija i prosudba ključna za prikupljanje značajnih podataka. Isto, ako se radi o podacima iz jedinstvenih ili nekonvencionalnih izvora koji možda nisu dostupni automatiziranim metodama. To može uključivati povijesne arhive koje još nisu digitalizirane i potrebno je imati osobu koja će prolaziti kroz te dokumente, arhivske zapise ili knjige.

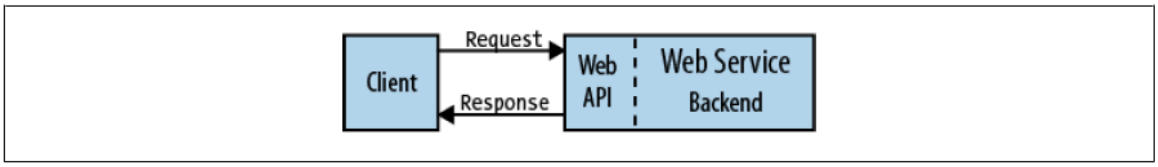
Ova metoda također ima i određene nedostatke poput vremenska zahtjevnosti. Taj proces može biti dugotrajan, pogotovo ako je potrebno prikupiti velike količine podataka ili ako se provode ponavljajuća istraživanja. Ovaj pristup zahtijeva vrijeme i trud istraživača ili osoblja koje sudjeluje u prikupljanju podataka. Ako se radi o prikupljanju velikog skupa podataka, ljudska greška je neizbježna. Postoji rizik od nenamjernih pogrešaka pri unosu podataka. Ove pogreške mogu utjecati na točnost i pouzdanost prikupljenih podataka.

Unatoč tim nedostacima, ručno prikupljanje podataka i dalje ima svoje mjesto u istraživanju, pogotovo u situacijama gdje su potrebni specifični i dubinski uvidi ili kada nema drugih izvora podataka koji su dostupni. Kombinacija ručnog prikupljanja podataka s drugim metodama prikupljanja može pružiti sveobuhvatan i pouzdan skup podataka za daljnju analizu i donošenje odluka.

2.2. API

API (Application Programming Interface) je skup definiranih metoda, funkcija i protokola koji omogućuju komunikaciju između različitih softverskih aplikacija. API omogućuje programerima pristup podacima ili funkcionalnostima druge aplikacije ili servisa, bez potrebe za pristupom izvornom kodu ili detaljima implementacije. API funkcionira kao posrednik između aplikacije koja šalje, klijenta, i aplikacije ili servisa koji pruža podatke, servera.

Slika 1. Primjer WEB API strukture



Izvor: (Masse, 2011.)

Klijent šalje zahtjev prema API sučelju u obliku standardiziranih HTTP[1] metoda poput GET, POST, PUT, PATCH ili DELETE. API tada obrađuje zahtjev, komunicira s odgovarajućom aplikacijom ili servisom te vraća rezultat ili odgovor na zahtjev nazad klijentu.

[1] Hypertext Transfer Protocol (HTTP) - protokol koji omogućuje komunikaciju i prijenos podataka između web poslužitelja i klijenata putem interneta.

Tablica 1. Popis HTTP metoda

HTTP metoda	Primjer	Upotreba
GET	GET /api/korisnici	Koristi se za čitanje ili dohvaćanje podataka s poslužitelja. Višestruki identični zahtjevi proizvode isti rezultat).
POST	POST /api/korisnici	Koristi se za stvaranje novih resursa na poslužitelju ili slanje podataka na obradu. Višestruki identični zahtjevi mogu proizvesti različite rezultate.
PUT	PUT /api/korisnici/1	Koristi se za ažuriranje postojećeg resursa na poslužitelju. Potpuno zamjenjuje postojeći prikaz novim podacima navedenim u zahtjevu.
DELETE	DELETE /api/korisnici/1	Koristi se za brisanje postojećeg resursa s poslužitelja. Uklanja navedeni resurs.
PATCH	PATCH /api/korisnici/1	Koristi se za djelomično ažuriranje postojećeg resursa na poslužitelju. Ona modificira samo određena polja ili atribute, ostavljajući ostatak nepromijenjenim

Izvor: izrada autora

2.2.1. Web API

Za dohvaćanje već obrađenih i formatiranih podataka, može se koristiti vrsta API-a zvana Web API. Web API omogućuje pristup podacima i funkcionalnostima putem interneta. Ova vrsta API-a koristi HTTP protokol za komunikaciju i često koriste formate poput JSON[2] ili XML[3] za prijenos podataka. Primjeri popularnih web API-a uključuju Twitter API, koji omogućuje pristup osnovnim elementima poput Tweet-ova, izravnih poruka, lista korisnika i još mnogo toga (Twitter, 2023). Google Maps API za razne geografske podatke i funkcionalnosti i Facebook Graph API koji omogućuje pristup podacima i funkcionalnostima povezanim s Facebook platformom.

Korištenjem Web API-a možemo na brz način pristupiti podacima iz različitih izvora, s time imamo pristup velikoj količini podataka u relativno kratkom vremenskom razdoblju, što nam omogućuje da te podatke integriramo s drugim aplikacijama i metodama koje koristimo za obradu podataka. Konzistentnost ovog formata smanjuje greške prilikom obrade i eliminira potrebu za ručnim unosom podataka.

Međutim, postoje i neki izazovi koji se mogu javiti prilikom korištenja API-a. Ne postoje API izvori za sve vrste podataka na kojima se rade istraživanja što ga čini korisnim samo u nekim situacijama. Učestali problem koji se isto javlja je ako dođe do promjene strukture podataka na API-u , postojeći procesi koji koriste taj API se moraju doraditi kako bi se prilagodili novoj strukturi. Nestabilnost API-a također može predstavljati izazov jer može doći do prekida ili nedostupnosti usluge, što može utjecati na prikupljanje podataka.

---

[2] JavaScript Object Notation (JSON) - format za razmjenu podataka koji se koristi za strukturiranje i prijenos informacija u čitljivom obliku koji je pogodan za programiranje.

[3] eXtensible Markup Language (XML) - jezik za označavanje podataka koji se koristi za strukturiranje i organiziranje podataka u obliku teksta radi lakše razmjene i obrade podataka između različitih aplikacija i platformi.

2.3. Web scraping

Web scraping je tehnika koja se koristi za dohvaćanje podataka s web stranica. Koristi se automatizacija za imitiranje ljudskih aktivnosti pregledavanja i dohvaćanja informacija s web stranica. Iako je širenje web usluga smanjilo potrebu za web scraper-ima kao metodom prikupljanja podataka, još uvijek postoje scenariji u kojima su oni korisni, konkretno, kad god programska sučelja nisu dostupna, npr. još uvijek postoje domene s malo relevantnih web usluga (Glez-Peña et al., 2014)

Web scraping tehnike mogu biti jednostavne, kao što je dohvaćanje podataka iz HTML[4] koda stranice, ili složene, koristeći napredne algoritme za interakciju s web stranicama na isti način kao što bi to činio ljudski korisnik.

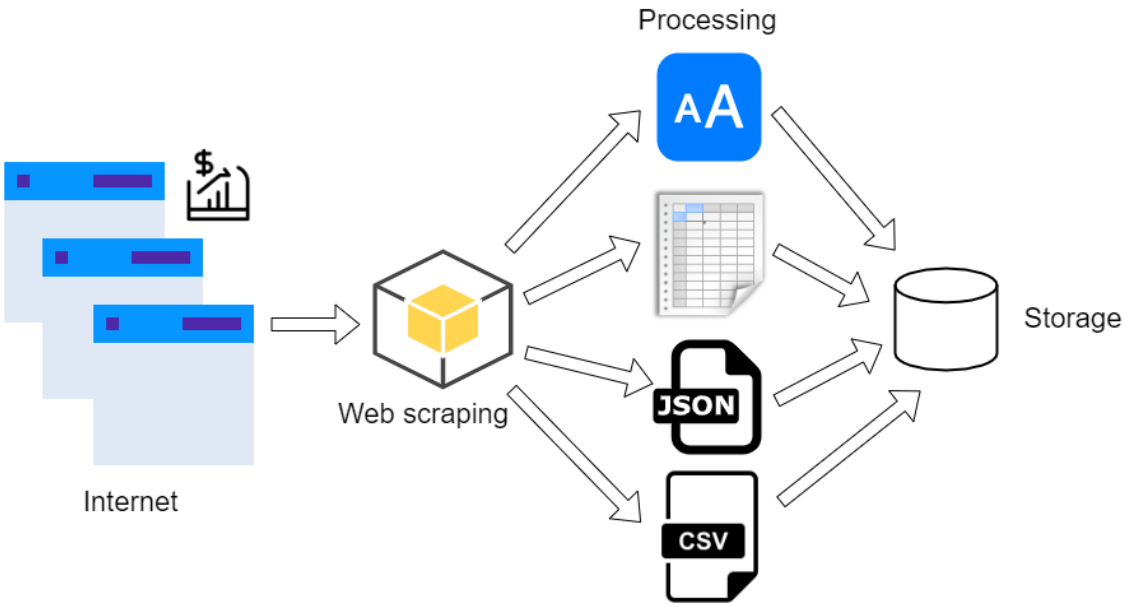
Da bi se prilagodile različitim scenarijima, trenutne tehnike web scraping-a postale su prilagođene, od manjih ad hoc[5] procedura uz pomoć ljudi do korištenja potpuno automatiziranih sustava koji su sposobni pretvoriti cijele web stranice u dobro organizirane skupove podataka. Najnoviji alati za web scraping ne samo da su sposobni obraditi markup jezike[6] ili JSON datoteke, već se također mogu integrirati s vizualnom analitikom računala (Butler, 2007).

[4] HTML (HyperText Markup Language) - jezik za označavanje koji se koristi za strukturiranje i prikazivanje sadržaja na web stranicama, sastoji se od elemenata koji opisuju semantičku strukturu i stilizaciju web sadržaja.

[5] Ad hoc - zahtjev koji se izvodi ili stvara prema potrebi ili u odgovoru na određenu situaciju, bez unaprijed definirane strukture ili plana.

[6] Markup jezici - jezici koji se koriste za označavanje ili strukturiranje sadržaja unutar elektroničkih dokumenata, pružajući smjernice o tome kako prikazati ili interpretirati određene dijelove sadržaja

Slika 2. Web scraping proces



Izvor: https://www.megatrend.com/en/all-you-need-to-know-to-make-your-first-web-scraper/

Dva glavna pojma u području ekstrakcije sadržaja s weba su web scraping (pretraživanje weba) i web crawling (skupljanje podataka s weba). Web scraping je sistematizirano izvlačenje teksta ili medijskog sadržaja s web stranica, postignuto korištenjem alata nazvanih web scraper-i. Koncept web scraping-a temelji se na korištenju metoda web crawling-a, automatiziranom sustavnom pregledavanju interneta putem praćenja veza na web stranicama pomoću web crawler-a. Procesi web scraping-a i web crawling-a tvore kontinuirani ciklus. Crawling vodi do HTML dokumenata iza web stranica, iz kojih izvlačimo željeni sadržaj i veze prema drugim web stranicama pomoću scraping-a, te dalje skupljamo podatke po prikupljenim poveznicama (Marić, 2022).

2.3.1. Web scraping alati

Web scraping alati su softverski alati ili biblioteke koje se koriste za automatizirano prikupljanje podataka s web stranica. Oni omogućuju izvlačenje informacija iz HTML, XML ili drugih oblika web sadržaja na strukturiran način, što olakšava daljnju analizu, obradu ili pohranu podataka. Ovi alati omogućuju programerima ili istraživačima da definiraju obrasce, pravila ili selektore kako bi odabrali specifične elemente i podatke koji su od interesa na web stranici. Alati mogu pružiti razne funkcionalnosti kao što su navigacija po stranicama, pretraga, filtriranje, raspakiravanje podataka i automatsko spremanje u određenim formatima poput CSV-a ili Excel datoteka.

Web scraping alati mogu biti biblioteke programskih jezika, kao što su Python biblioteke BeautifulSoup, Selenium i Scrapy okvir, koje pružaju fleksibilnost i kontrolu nad procesom scraping-a. Ovi alati igraju ključnu ulogu u prikupljanju podataka s weba za istraživačke, analitičke ili poslovne svrhe. Omogućuju korisnicima da automatiziraju procese prikupljanja podataka koji bi inače bili vremenski zahtjevni ili nepraktični.

Neki od tih alatu su :

Beautiful Soup - Python biblioteka koja se koristi za izdvajanje podataka iz HTML i XML dokumenata na jednostavan i intuitivan način. Prednosti ove biblioteke leže u njenom jednostavnom korištenju, podršci za CSS[7] selektore i mogućnosti navigacije po dokumentu. Također pruža metode za izvlačenje željenih podataka, poput pretraživanja po imenima oznaka, klasama ili atributima. Biblioteka također ima dobru toleranciju na pogreške i sposobnost obrade nepotpunih ili nevaljanih HTML dokumenata. Međutim, neke od mana uključuju slabije performanse u usporedbi s drugim bibliotekama, te ovisnost o vanjskim implementacijama za pretvaranje HTML koda u strukturu stabla.

[7] CSS (Cascading Style Sheets) - jezik za stiliziranje web stranica koji se koristi za definiranje izgleda, dizajna i prezentacije elemenata na web stranici, kao što su boje, fontovi, pozicioniranje i ostali vizualni efekti.

Scrapy - Python bazirani web scraping okvir (engl. Framework) koji omogućuje automatizaciju procesa prikupljanja podataka s web stranica. Scrapy je dizajniran s fokusom na efikasnost i skalabilnost. Omogućuje paralelno slanje zahtjeva i asinkrono procesiranje što rezultira visokom skalabilnošću. Također, dolazi s ugrađenim funkcionalnostima kao što su automatsko otvaranje linkova, definiranje pravila za izdvajanje podataka i podrška za ekstrakciju podataka iz različitih formata. Fleksibilnost je još jedna prednost ovog alata, jer je modularan i omogućuje prilagodbu i proširenje funkcionalnosti prema potrebama programera.

Unatoč prednostima, Scrapy također ima nekoliko mana. Prva mana je sama kompliciranost sustava, koja može biti izazovna za početnike ili one bez prethodnog iskustva s prikupljanjem podataka s web stranica. Osim toga, Scrapy je osjetljiv na promjene u strukturi web stranica, pa zahtijeva redovito održavanje i ažuriranje koda kako bi se prilagodili tim promjenama. Također, Scrapy je specifičan za Python i koristi svoje biblioteke i alate, što može uzrokovati probleme ako se koristi u kombinaciji s drugim tehnologijama.

Selenium – Python biblioteka za automatizaciju preglednika, koji se izvorno koristio za testiranje web aplikacija. Međutim, u kontekstu web scraping-a, Selenium se koristi za interakciju s web stranicama kako bi se prikupili podaci koji nisu lako dostupni preko jednostavnih HTTP zahtjeva. Arhitektura Selenium biblioteke omogućuje korisnicima da simuliraju interakcije korisnika s web stranicama, poput klika na gumbe ili popunjavanja obrazaca. To je osobito korisno prilikom izvlačenja podataka s dinamičkih web stranica koje koriste Javascript[8] za učitavanje ili prikazivanje informacija. Selenium se obično koristi u kombinaciji s drugim alatima za ekstrakciju podataka, poput BeautifulSoup-a ili Scrapy-a, za kompleksnije zadatke prikupljanja podataka. Međutim, samo korištenje ove biblioteke može biti dovoljno za jednostavne zadatke. Jedan od glavnih izazova ovog alata je njegova složenost i sporo vrijeme izvršavanja u usporedbi s drugim alatima, posebno kada se koristi za velike scraping projekte.

[8] Javascript - programski jezik koji se široko koristi za razvoj interaktivnih web stranica i aplikacija putem izvođenja skripti na strani klijenta u web pregledniku.

2.3.2. Izazovi

Iako je web scraping koristan način za prikupljanje podataka, postoje brojni izazovi s kojima se korisnici mogu suočiti tijekom njegove primjene. Neki od problema koji se mogu pojaviti tijekom ovog procesa su promjena strukture web stranice s koje se prikupljaju podaci, otežani pristup podacima ili razne mjere ograničenja koje vlasnici stranica mogu postaviti kako bi onemogućili pristup podacima s njihove stranice.

Promjene strukture web stranice - Struktura web stranice odnosi se na način na koji su elementi na stranici organizirani i povezani. Kada se struktura web stranice mijenja, može utjecati na sposobnost skripta za prikupljanje podataka da uspješno pristupe prethodno dostupnim podacima. Na primjer, ako skripta očekuje da će određena informacija uvijek biti u određenom HTML elementu, a taj element se promijeni ili premjesti, skripta može prestati raditi. Održavanje skripti koje se koriste za web scraping tako da prate promjene na web stranicama može biti vremenski zahtjevno i tehnički složeno. To također zahtijeva kontinuirano praćenje i testiranje kako bi se osiguralo da skripte za web scraping još uvijek pravilno funkcioniraju nakon svake promjene strukture web stranice.

Pristup podacima - Web stranice se sve više koriste napredne tehnologije kao što je JavaScript za prikaz sadržaja. To može stvoriti izazove za web scraping, jer ovi elementi često zahtijevaju interakciju korisnika da bi se podaci učitali. Standardni web scraping alati obično izvlače podatke iz statičkog HTML koda stranice, a ne mogu lako pristupiti podacima koji se dinamički učitavaju. Za rješavanje ovog problema, istraživači često koriste alate kao što je Selenium, koji može automatizirati interakcije s web preglednikom i simulirati korisničke radnje kako bi se pristupilo dinamički učitanim podacima.

Blokiranje i ograničenja - Web stranice koriste različite metode za zaštitu od web scraping tehnika. Captcha (Completely Automated Public Turing Test to Tell Computers and Humans Apart) sustav sprječava različite web stranice ili se koristi na webu kako bi se spriječio pristup raznim bot-ovima[9] koji napadaju mrežne resurse (Kaur and Behal, 2014). Ovo može otežati ili čak onemogućiti web scraping. Neke web stranice također provjeravaju IP adrese koje šalju zahtjeve, te mogu blokirati one koje šalju previše zahtjeva u kratkom vremenskom razdoblju. Ovo je poznato kao ograničavanje broja pristupa (engl. Rate limiting) i može usporiti ili čak potpuno zaustaviti proces web scraping-a. Postoje metode za zaobilaženje ovih ograničenja, kao što su korištenje proxy[10] servera za promjenu IP adrese ili programiranje pauza između zahtjeva, ali ove metode mogu dodati dodatnu složenost procesu prikupljanja podataka.

Postavke pravila za web scraping - *robots.txt* je standardna datoteka koju web administratori koriste za davanje uputa web robotima, poznati kao pauci, o tome kako da se ponašaju na njihovoj stranici. Smještena je u korijenskom direktoriju web stranice. Svrha te datoteke je da vlasnicima stranica omogući kontrolu nad načinom na koji njihove stranice indeksiraju i pregledavaju web roboti. Pravila u *robots.txt* datoteci nisu pravno obvezujuća, ali općenito se smatra dobrom praksom da ih se pridržavaju svi roboti, uključujući one koji se koriste za web scraping. Poštivanje *robots.txt* datoteke pomaže očuvanju resursa web stranice. Web roboti, ako nisu pravilno kontrolirani, mogu preopteretiti server stranice sa zahtjevima, što može negativno utjecati na performanse stranice. Poštivanje *robots.txt* datoteke i postavke za niski broj zahtjeva prema stranici koja se pristupa pomaže u sprečavanju ovog problema.

[9] Bot - skraćeni naziv za robota, u značenju računalnog programa koji se izvršava samostalno.

[10] Proxy - posrednik između korisnika i interneta koji omogućuje anonimnost i zaštitu privatnosti skrivanjem stvarne IP adrese korisnika i omogućuje pristup web stranicama putem posrednika.

Iako napredak alata i tehnologija za web scraping olakšava pristupanje web podacima za razne korisnike, zakonitost aktivnosti web scraping-a je složena i često se zanemaruje (Murray State University *et al.*, 2020). Podaci objavljeni na internetu često su zaštićeni autorskim pravima. Iako je većina podataka na web stranicama javno dostupna, to ne znači da ih se može slobodno kopirati i koristiti u bilo koje svrhe. Upotreba ovih podataka može biti ograničena zakonima o autorskim pravima, pogotovo ako se podaci koriste komercijalno ili distribuiraju na način koji šteti vlasniku autorskih prava.

Mnoge web stranice imaju uvjete korištenja koji izričito zabranjuju korištenje i objavu podataka koji su prikupljeni s njihovih stranica. Prije kretanje procesa prikupljanja podataka, preporučljivo je pročitati uvjete korištenja stranice s koje se skupljaju podaci. Ako stranica zadržava sva prava na korištenje podataka koji se prikupljaju pomoću web scraping ili drugih tehnika, potrebno je zatražiti pisano dopuštenje od vlasnika stranice za uporabu tih istih podataka

3. Scrapy Framework

Kao što je objašnjeno u Scrapy 2.9 dokumentaciji (Scrapy, 2023) , Scrapy je aplikacijski okvir koji se koristi za pretraživanje web stranica i izvlačenje strukturiranih podataka, što se može koristiti za širok raspon korisnih aplikacija poput rudarenja podataka, obrade informacija ili povijesnog arhiviranja. Scrapy je asinkroni okvir, što omogućuje istovremeno izvršavanje više zadataka, što poboljšava brzinu i efikasnost pri dohvatu podataka.

Scrapy je izuzetno prilagodljiv i može se koristiti za različite vrste web scraping zadataka, uključujući istraživanje, razvoj proizvoda, analizu podataka i mnoge druge primjene. Omogućuje izvlačenje podataka iz HTML stranica, API-a i sadržaja generiranih putem JavaScript-a, te nudi mogućnost praćenja linkova i navigacije kroz web stranice na način koji imitira ljudskog korisnika. Scrapy pruža sve alate koji su potrebni za učinkovito izvlačenje podataka s web stranica, obradu, pohranu u odabranu strukturu i format podataka za daljnje korištenje.

3.1. Pregled arhitekture

Sljedeći dijagram sa slike 3, koji prikazuje pregled Scrapy arhitekture i tijek kojim se procesi izvršavaju, izdvajamo sljedeće ključne komponente :

Scrapy mašina (engl. Engine) **:** Djeluje kao središnji koordinator i upravljački sustav za cijeli postupak preuzimanja podataka s web stranice. Kontrolira protok podataka između različitih komponenti poput pauka (engl. Spider), planera (engl. Scheduler), preuzimatelja (engl. Downloader), međuslojeva pauka (engl. Spider Middlewares) i cjevovoda stavki (engl. Item Pipelines). Obrađuje zahtjeve koji se šalju web stranicama i odgovore koje dobi od tih zahtjeva od strane pauka i raspoređuje ih u planer. Također dohvaća sljedeće zahtjeve iz planera i prosljeđuje ih preuzimatelju radi preuzimanja web stranica. Nakon preuzimanja stranice, mašina prima odgovor i prosljeđuje ga pauku na obradu i upravlja interakcijom između pauka i ostalih dijelova okvira. Šalje preuzete odgovore pauku radi čitanja i izdvajanja podataka. Također prima prikupljene stavke i nove zahtjeve od pauka i preusmjerava ih odgovarajućim komponentama radi daljnje obrade. Mašina neprekidno komunicira s planerom kako bi odredio sljedeći skup zahtjeva za pretraživanje.

Planer (engl. Scheduler) **:** Upravlja procesom pretraživanja koristeći red čekanja pomoću kojeg odlučuje o prioritetima zahtjeva. On izbjegava duplicirane zahtjeve, raspoređuje zahtjeve za preuzimanje, upravlja greškama i podržava distribuirano pretraživanje. Ukratko, planer osigurava učinkovito upravljanje zahtjevima i koordinaciju tijekom web scraping procesa.

Scrapy preuzimatelj (engl. Downloader) : Odgovoran je za preuzimanje web stranica. On šalje HTTP zahtjeve, rukuje HTTP odgovorima, filtrira zahtjeve, upravlja preusmjeravanjem web zahtjeva i podržava međuslojeve za preuzimanje. Ima ključnu ulogu u upravljanju preuzimanjem web stranica tijekom procesa.

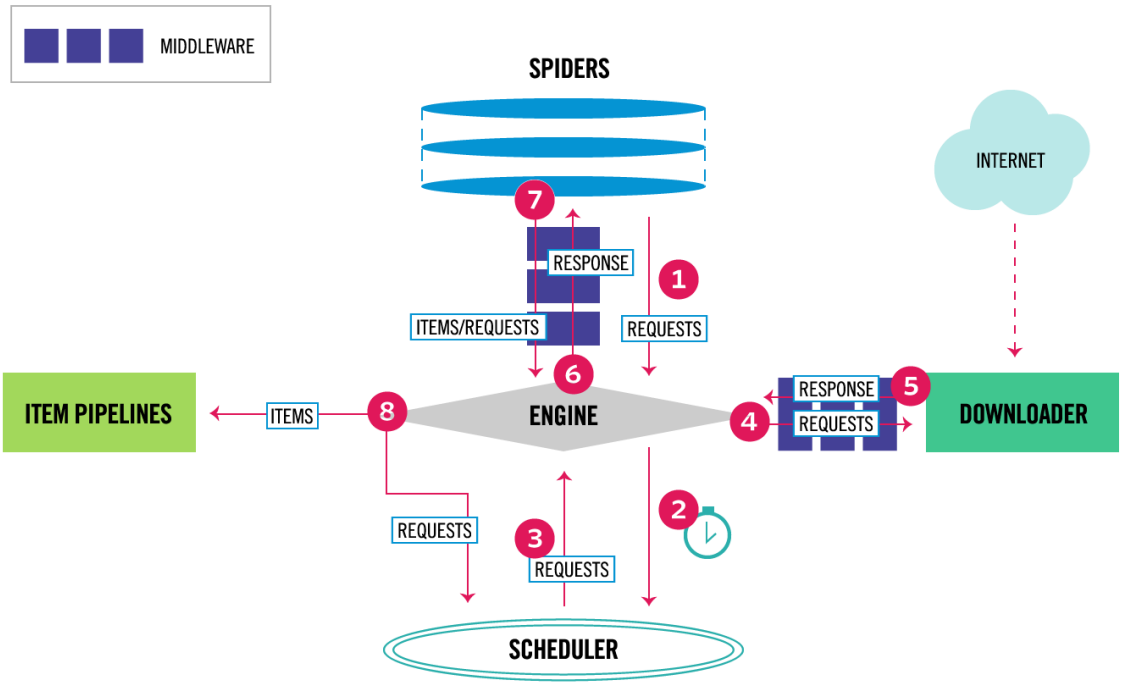
Pauk (engl. Spider) : Python klasa koja definira kako pretraživati i izdvajati podatke s određenih web stranica. To je glavna komponenta odgovorna za logiku web scraping procesa. Pauk započinje slanjem početnih zahtjeva web stranicama, obrađuje odgovore, izdvaja željene podatke i može generirati dodatne zahtjeve za daljnje praćenje poveznica ili navigaciju kroz stranice. On definira strukturu i pravila za proces pretraživanja, specificira kako izdvojiti podatke i rukovati različitim vrstama web stranica.

Cjevovod stavki (engl. Item Pipeline) : Komponenta koja obrađuje i upravlja prikupljenim podacima. Omogućuje transformaciju, provjeru ispravnosti i pohranu stavki koje su izdvojene s web stranica. On prima stavke od pauka, primjenjuje niz prilagodljivih koraka obrade i odlučuje kako postupiti sa svakom stavkom. To može uključivati čišćenje podataka, provjeru valjanosti i pohranu stavki u bazu podataka ili druge sustave za pohranu.

Međuslojevi preuzimatelja (engl. Downloader Middlewares) : Komponente koje omogućavaju prilagodbu i manipulaciju zahtjevima i odgovorima tijekom procesa preuzimanja. Oni presreću zahtjeve prije nego što se pošalju web stranicama i odgovore prije nego što stignu do pauka. Omogućuju različite funkcionalnosti, poput izmjene zaglavlja, dodavanja autentifikacije, upravljanja proxy poslužiteljima i implementacije prilagođene logike preuzimanja.

Međuslojevi pauka (engl. Spider Middleware) : Komponente koje služe za prilagodbu i manipulaciju zahtjevima i odgovorima tijekom procesa pretraživanja. Oni presreću zahtjeve prije nego što stignu do pauka i odgovore prije nego što se obrade. Omogućuju različite funkcionalnosti, poput izmjene zahtjeva, rukovanja greškama, dodavanja ili izmjene specifičnih podataka za pauka i implementacije prilagođene logike ponašanja pauka.

Slika 3. Dijagram Scrapy arhitekture



Izvor: https://docs.scrapy.org/en/latest/topics/architecture.html

Protok podataka unutar Scrapy okvira kontrolira izvršni mehanizam koji slijedi sekvencijalni proces. Na početku, mašina dobiva početne zahtjeve od pauka (1). Ti zahtjevi se zatim raspoređuju u planer (2), a mašina nastavlja tražiti sljedeće zahtjeve za pretraživanje. Planer vraća sljedeći skup zahtjeva mašini (3). Ti zahtjevi se šalju preuzimatelju s mogućnošću prolaska kroz međuslojeve preuzimatelja tijekom poziva metode *process_request()* (4). Nakon što se stranica završi preuzimanje, preuzimatelj generira odgovor koji sadrži preuzetu stranicu. Taj odgovor se zatim šalje natrag mašini, prolazeći kroz preuzimateljeve međuslojeve putem metode *process_response()* (5). Mašina prima odgovor i prosljeđuje ga pauku na daljnju obradu, prolazeći kroz međuslojeve pauka koristeći metodu *process_spider_input()* (6) .Unutar pauka, odgovor se obrađuje nakon čega vraća prikupljene stavke i potencijalno nove zahtjeve za daljnje pretraživanje (7). Obrađene stavke se šalju cjevovodu stavki, dok se obrađeni zahtjevi šalju natrag planeru gdje mašina traži moguće sljedeće zahtjeve za pretraživanje (8). Taj proces se nastavlja ponavljati od koraka tri sve dok planer više ne pruža dodatne zahtjeve.

3.2. Scrapy pauk

Scrapy pauk je klasa u sklopu Scrapy projekta koja nasljeđuje osnovnu Scrapy pauk klasu, odnosno osnovna jedinica za izvršavanje web scraping procesa, u kojoj se definiraju koraci kako će se prikupljati podaci s odabrane web stranice i kako će pauk pratiti poveznice.

Prvi korak je davanje imena pauku kako bi se mogao identificirati u samom scrapy projektu. Ime mora biti jedinstveno, najčešće u string[11] obliku, te se definira kao atribut[12] u pauk klasi.

Kako bi pauk znao kojim stranicama će pristupati, definira se početna web stranica od koje proces kreće. Kao i za ime, definira se atribut koji sadrži poveznicu na koju se šalje HTTP zahtjev kako bi se dohvatili podaci s web stranice.

Za obradu dohvaćenih podataka koji su poslani kao odgovor na HTTP zahtjev definira se metoda, funkcija koja je vezana za određeni objekt i omogućuje izvršavanje određenih operacija na tom objektu, *parse()* unutar pauk klase. To je *callback* metoda[13] i koristi se za obradu odgovora nakon što se dovrši preuzimanje web stranice. Ova metoda je ključna u procesu ekstrakcije podataka jer omogućuje pristup sadržaju odgovora, ekstrakciju podataka, slanje dodatnih zahtjeva, navigaciju na druge stranice i daljnju obradu podataka.

[11] String - tip podataka u programiranju koji se koristi za pohranjivanje i manipulaciju niza karaktera.

[12] Atribut - varijabla koja je povezana s objektom i sadrži informacije ili karakteristike koje opisuju taj objekt.

[13] Callback metoda - metoda koja se predaje ili registrira kao argument funkciji ili klasi kako bi se pozvala kasnije, obično u odgovarajućem trenutku ili događaju

Metoda prima dva parametra, *self* i *response*. U kontekstu Scrapy pauka, *self* se odnosi na samu instancu pauka što omogućuje pristup i manipulaciju atributa i metoda klase unutar njenih drugih metoda. *Response* zato zadrži sve podatke koji su dohvaćeni s dane poveznice.

Slika 4. Primjer *parse()* metode

```
def parse(self, response):

    # Initialize a scrapy.Item class
    item = MyItem()
    # Extract data using css and xpath selectors
    item['headline'] = response.css('h1::text').get()
    item['text'] = response.xpath('//p/text()').get()


    # Yield the data as an item
    yield item
```

Izvor: izrada autora

Tablica 2. Popis statusnih kodova

Statusni kod	Naziv odgovora	Opis
200	OK	Zahtjev je uspješno izvršen, a tijelo odgovora sadrži traženi resurs.
201	Kreirano	Zahtjev je uspješno izvršen, te je kao rezultat stvoren novi resurs.
204	Nema sadržaja	Poslužitelj je uspješno obradio zahtjev, ali nije potrebno vratiti sadržaj.
301	Trajno premješteno	Traženi resurs trajno je premješten na novu adresu
400	Pogrešan zahtjev	Poslužitelj ne može razumjeti zahtjev zbog neispravne sintakse ili nedostajućih parametara.
401	Neautorizirano	Zahtjev zahtijeva autentifikaciju korisnika
403	Zabranjeno	Poslužitelj razumije zahtjev, ali odbija ga autorizirati. Klijent nema dovoljne ovlasti.
404	Nije pronađeno	Traženi resurs nije pronađen na poslužitelju.
500	Interna pogreška	Došlo je do opće pogreške sa strane poslužitelja.

Izvor: izrada autora

U tablici 2 su prikazani neki od najčešće korištenih statusnih kodova.

Kako bi se dohvatili podaci iz *response* atributa, Scrapy dolazi s vlastitim mehanizmom za izvlačenje podataka koji se zovu selektori (engl. Selectors), oni mogu izdvajati određene elemente HTML objekta koristeći XPath i CSS izraze (Yin,2021). CSS selektori su način odabira određenih elemenata unutar HTML dokumenta. Bazirani su na CSS sintaksi i omogućuju ciljanje elemenata na temelju njihovih oznaka, klasa, identifikatora, atributa ili njihovih odnosa unutar strukture dokumenta. CSS selektori pružaju preciznu i jednostavnu metodu za izdvajanje željenih elemenata iz HTML dokumenata.

S druge strane, XPath selektori su oblici upita za odabir čvorova u XML ili HTML dokumentima. Oni pružaju naprednu i više fleksibilniju metodu za navigaciju kroz strukturu dokumenta. Pomoću XPath-a je moguće odabrati elemente na temelju njihovih oznaka, atributa, tekstualnog sadržaja ili čak njihove pozicije unutar dokumenta.

Tablica 3. Popis selektora

Vrsta Selektora	CSS	XPath
Vrsta elementa	div	//div
Klasa	.classname	//*[contains(@class, 'classname')]
ID	#id	//*[@id='id']
Atribut	[atribut=value]	//*[@atribut=value]
Hijerarhija	parent > child	//parent/child
Pozicija	:nth-child(n)	(//element)[n]
Tekst	:contains(text)	//*[contains(text(),'text')]

Izvor: izrada autora

Podatke koji su dohvaćeni iz *response* atributa spremamo u Scrapy stavku (engl. Item). Scrapy stavke su jednostavni Python objekti koji se koriste za prikazivanje i pohranu podataka tijekom web scraping-a. Djeluju kao spremnici za prikupljene informacije, omogućavajući jednostavno organiziranje i manipuliranje podacima. Detaljnije ćemo ih obraditi u sljedećem poglavlju.

Na kraju ovog procesa koristi se ključna riječ *yield.* Ključna riječ *yield* koristi se unutar *callback* funkcije Scrapy pauka kako bi se vratili podaci ili daljnji zahtjevi za obradu. Kada se *yield* koristi s atributom stavke, šalje se u Scrapy mašinu koja ju zatim dalje obrađuje. To omogućava jednostavno i učinkovito upravljanje sakupljenim podacima. Osim toga, *yield* se može koristiti za generiranje novih zahtjeva unutar povratne funkcije. Kroz vraćanje novog objekta *request* s određenom web adresom, Scrapy će poslati taj zahtjev i pozvati odgovarajuću povratnu funkciju za obradu odgovora. To omogućava sekvencijalno pretraživanje više stranica ili praćenje veza sa stranice koju obrađujemo.

Korištenje *yield* umjesto izravnog vraćanja vrijednosti omogućava da asinkrono obrađujemo zahtjeve, čime se omogućuje učinkovito upravljanje velikim web scraping projektima.

Slika 5. Primjer vraćanja stavke

```
import scrapy

class MySpider(scrapy.Spider):
    name = 'example_spider'
    start_urls = ['http://www.example.com']

    def parse(self, response):
         item = {
         'headline': response.css('h1::text').get()
         ,'text': response.xpath('//p/text()').get()
        }
      yield item
```

Izvor: izrada autora

3.3. Scrapy Stavka

Glavni cilj pri web scraping-u je izvući strukturirane podatke iz neuređenih izvora, obično web stranica. Pauci mogu vratiti izvučene podatke kao stavke (engl. Item), Python objekte koji definiraju podatke u obliku parova ključ-vrijednost (Scrapy, 2023).

Scrapy stavka je jednostavan spremnik koji se koristi za pohranu izvučenih podataka s web stranica. Djeluje kao objekt sličan rječniku, omogućavajući definiciju polja i njihove odgovarajuće vrijednosti koje dohvaćamo. Korištenje Scrapy stavki nam dopušta organizaciju i strukturiranje izvučenih podataka u standardiziranom formatu, što olakšava kasniju obradu i analizu.

Za definiranje Scrapy stavke potrebno je stvoriti Python klasu koja je pod klasa klase *scrapy.Item*. Svaki atribut klase predstavlja polje u stavci, a njegova vrijednost će sadržavati izvučene podatke za to polje

Slika 6. Primjer Scrapy stavka klase

```
import scrapy

class ExampleItem(scrapy.Item):
    name = scrapy.Field()
    number = scrapy.Field()
    long text = scrapy.Field()
    codename = scrapy.Field()
```

Izvor: izrada autora

U gore navedenom primjeru smo napravili Scrapy stavku naziva *ExampleItem* koji sadrži četiri polja: *name* , *number* , *long_text* i *codename*. Dio *.Field()* u definiciji Scrapy atributa, recimo *name = scrapy.Field()*, služi kao oznaka polja. Ona ukazuje da je atribut polje koje može sadržavati izvučene podatke za to određeno polje. Korištenjem *.Field()* indikatora stvara se instanca klase *scrapy.Field* i povezuje se s atributom. To omogućava da Scrapy prepozna i obradi atribut kao polje prilikom obrade izvučenih podataka. Ova oznaka služi kao oblik dokumentacije, ukazujući da je atribut namijenjen za korištenje kao polje za pohranjivanje izvučenih podataka.

To pomaže poboljšati čitljivost i održavanje koda, posebno u većim Scrapy projektima.

Nakon stvaranja klase, možemo popuniti njene atribute vrijednostima koje se dohvaćaju prilikom web scraping procesa. Ovaj korak se inače odrađuje unutar pauk klase koja šalje zahtjeve web stranicama, dohvaća rezultate zahtjeva i sadrži programski kod koji sprema dohvaćene podatke u atribute stavka klase.

Slika 7. Primjer spremanja atributa u stavku

```
import scrapy

class ExampleSpider(scrapy.Spider):
    name = 'example_spider'
    start_urls = ['http://www.example.com']

    def parse(self, response):
        #Create an instance of the ExampleItem class
        item = ExampleItem()

        item['name']=response.css('h3.title::text').get()
        item['number']=response.css('p.price::text').get()
        item['long_text']=response.css('div.txt::text').get()
        item['codename']=response.xpath('//p/text()').get()}

        yield item
```

Izvor: izrada autora

Na slici 7 prikazan je primjer kako spremiti podatke u Scrapy stavku. Unutar metode *parse()* u pauku, stvara se instanca *ExampleItem* klase, a njezina polja (*name, number, long_text i codename*) se popunjavaju podacima tako da se koriste kombinacija CSS i XPath selektora kako bi se izvukli određeni podaci iz rezultata poslanog zahtjeva koji se nalaze u *response* atributu. Konačno, popunjena stavka se vraća za daljnju obradu u procesu. Nakon što Scrapy pauk vrati stavku, on prolazi kroz cjevovod (engl. Pipeline) radi daljnje obrade. Ti cjevovodi stavki su odgovorni za primjenu različitih operacija kao što su provjera ispravnosti, čišćenje i spremanje. Cjevovod stavki ćemo detaljnije obraditi u sljedećem poglavlju.

3.4. Cjevovod stavki

Nakon što je stavka izvučena od strane pauka, šalje se u cjevovod stavki koji je obrađuje kroz nekoliko komponenata koje se izvršavaju sekvencijalno. Svaka komponenta cjevovoda stavki je Python klasa koja implementira jednostavnu metodu. One primaju stavku i obavljaju radnju nad njom, također odlučujući hoće li stavka nastaviti kroz cjevovod ili biti odbačena i više ne obrađivana (Scrapy, 2023).

Nakon što pauk vrati stavku , ona prolazi kroz cjevovod stavki koji izvršava različite operacije nad tom stavkom, kao što su provjera ispravnosti, čišćenje i spremanje. Taj cjevovod omogućuje kontrolu i manipulaciju izvučenih podataka prije nego što se spreme ili dalje obrade.

Slika 8. Primjer cjevovoda stavki

```
class ExampleItemPipeline:
    def process_item(self, item, spider):
        item['name'] = self.clean_name(item['name'])
        item['number'] = self.transform_number(item['number'])
        item['long text'] = self.clean long text(item['long text'])
        item['codename'] = self.generate_codename(item['codename'])
        return item

    def clean_name(self, name):
        cleaned_name = name.strip().title()
        return cleaned_name
    def transform_number(self, number):
        transformed_number = int(float(number))
        return transformed_number
    def clean long text(self, long text):
        cleaned_long_text = long_text.replace('\n', ' ').strip()
        return cleaned_long_text
    def generate_codename(self, codename):
        generated_codename = codename.upper().replace(" ", "_")+ "_CODE"
        return generated_codename
```

Izvor: izrada autora

Na slici 8 prikazan je jednostavan primjer čišćenja stavki koje pauk prosljeđuje cjevovodu stavki. Klasa *ExampleItemPipeline* implementira metodu *process_item()* koja je osnovna metoda cjevovoda stavki. Metoda prima stavku izvučenu od strane pauka i vrši transformacije i čišćenje na određenim poljima. Očišćene i transformirane vrijednosti se zatim ažuriraju u stavci prije nego što se vrati.

clean_name(self, name) : funkcija vrši čišćenje polja *name*. Uklanja početne i završne razmake koristeći metodu *strip()* i velika slova prve riječi svake riječi koristeći metodu *title().*

transform_number(self, number): Pretvara polje u cjelobrojni broj koristeći funkciju *int()* kako bi se izostavila decimalna mjesta broja.

clean_long_text(self, long_text): Zamjenjuje znakove za novi red `*'\n'*` razmacima koristeći metodu *replace()* Dodatno, uklanjaju se početni i završni razmaci koristeći metodu *strip().* Ovakva metoda je korisna kako bi se veći tekstovi bolje formatirali prilikom daljnjeg spremanja.

generate_codename(self, codename): metoda *upper()* se koristi za pretvaranje polja *codename* u velika slova. Zatim, metoda *replace()* se primjenjuje za zamjenu svih razmaka s donjim crtama. Na kraju, nastavak "_CODE" se dodaje transformiranom kodnom imenu. Ova promjena osigurava da generirano kodno ime bude velikim slovima, zamjenjuje razmake donjim crtama i dodaje nastavak "_CODE" na kraju atributa.

4. Prikupljanje i obrada podataka

Web stranice su ocean neograničenih informacija kojima svatko može pristupiti. Novi trend tehnologije natjerao nas je da promijenimo način poslovanja. Internet je sada novo mjesto za poslovanje (Henrys, 2020).

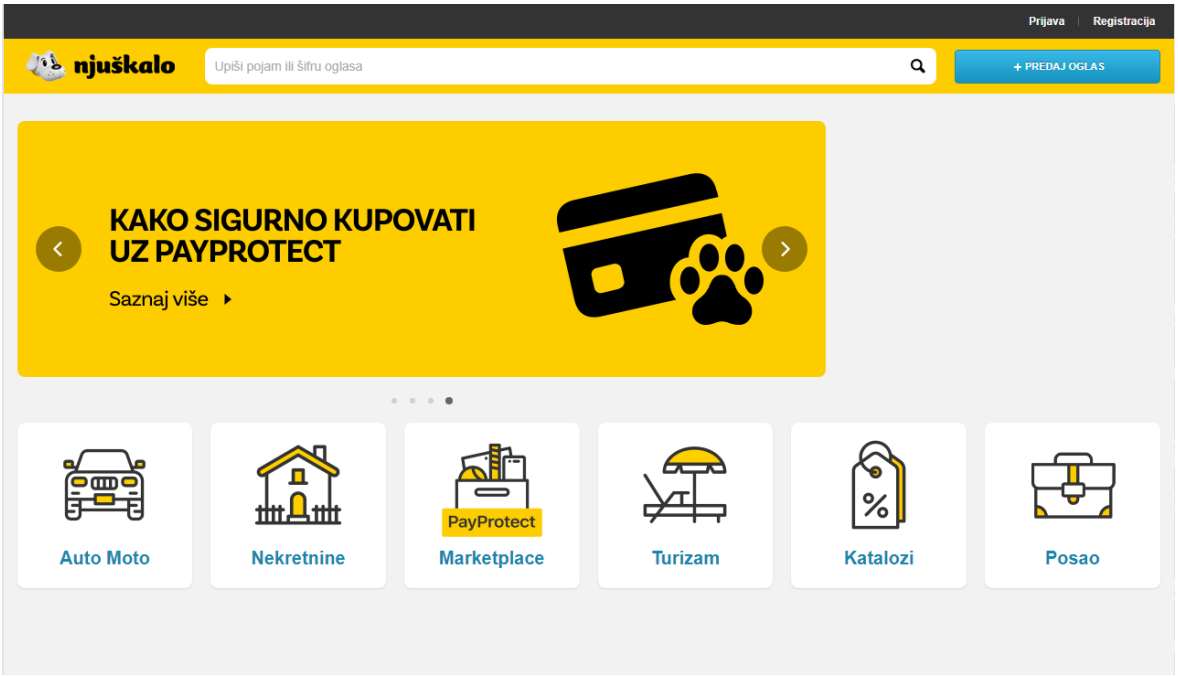
Elektronička trgovina uključuje digitalno omogućene komercijalne transakcije između organizacija i pojedinaca (Laudon & Traver, 2009). Uz sve veći porast elektroničkog poslovanja, E-tržišta su postala neizostavan dio suvremenog načina kupovine. Jedna od ključnih usluga koje se pružaju internetskim posjetiteljima ili kupcima na E-tržištima jest mogućnost naručivanja proizvoda putem interneta. Kupovina putem interneta često se smatra najbržim i najekonomičnijim načinom za nabavu različitih proizvoda. Trgovina na internetu također se ističe kao najprofitabilniji oblik trgovine zahvaljujući svojoj jednostavnosti i niskim troškovima. U Hrvatskoj, jedna od vodećih web stranica koja pruža sve te mogućnosti je Njuškalo.hr.

Njuškalo.hr je popularna platforma i oglasna stranica koja omogućuje korisnicima da prodaju i kupuju razne proizvode i usluge. Osnovana je 2007. godine, stranica nudi širok spektar kategorija, uključujući vozila, elektroniku, odjeću, kućanske aparate te nekretnine koje su nam od posebnog interesa u ovom radu. Njuškalo.hr je postao sinonim za online oglašavanje i trgovinu u Hrvatskoj, privlačeći veliki broj korisnika svakodnevno. Koristeći mogućnosti detaljnog pretraživanje oglasa putem naprednih filtera, kao što su kategorije proizvoda, cijena, geografska lokacija i drugih ,iskoristit ćemo ovu stranicu kako bi prikupili podatke o oglasima vezanim za nekretnine na području grada Pule. Prikupljanje ovih podataka odrađeno je u suglasnosti s Njuškalo.hr.

4.1.1. Struktura stranice

Za prikupljanje podataka o nekretninama sa stranice Njuskalo.hr , od interesa će nam biti dvije web poveznice koje ćemo koristiti kao našu početnu točku. Prilikom pristupanja stranici, prvo smo usmjereni na njenu početnu stranicu (engl. Homepage).

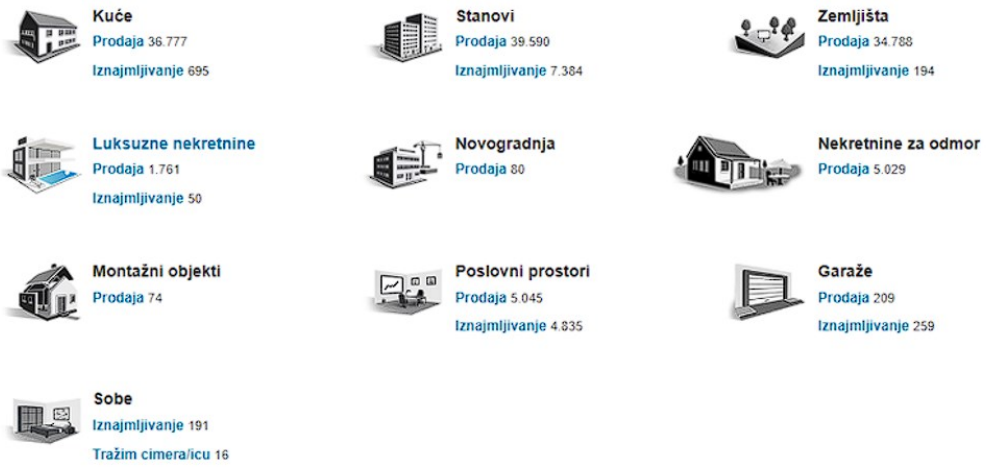
Slika 9. Početna stranica Njuškalo.hr



Izvor: https://www.njuskalo.hr/

Na početnoj stranici postoji kategorizacija oglasa koja korisnicima omogućava da lako pristupe različitim dijelovima web stranice. U postavljanu samog procesa skupljanja podataka s ove stranice, moguće je krenuti proces i s ove točke ali s obzirom da su nam potrebni podaci samo o nekretninama, nastavljamo na sljedeću poveznicu koja nas vodi na oglase o nekretninama.

Slika 10. Podjela kategorija oglasa za nekretnine



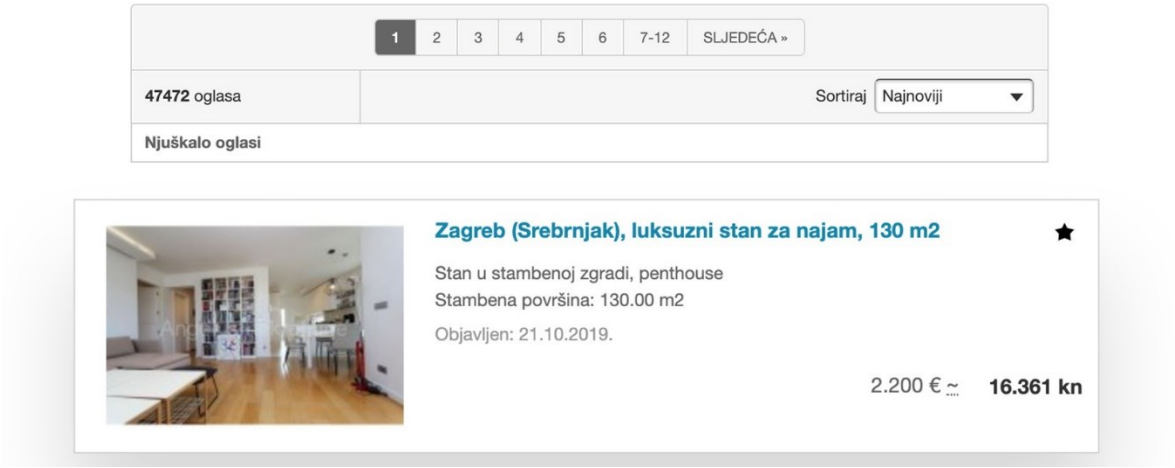
Izvor: https://www.njuskalo.hr/nekretnine

Oglasi za nekretnine su podijeljeni u kategorije prikazano na slici 10. Od interesa su nam kategorije *Kuće* i *Stanovi*. Podaci koje želimo dohvatiti nalaze se u njihovim potkategorijama iliti pod *Prodaja*. Iako se potkategorija zove isto za oglase kuća i stanova, bitno je naglasiti da vode na različite stranice. Prateći poveznicu za prodaju kuća stižemo na stranicu *njuskalo.hr/prodaja-kuca* , dok za stanove dolazimo na *njuskalo.hr/prodaja-stanova*. Ove dvije stranice su strukturom jednake s nekoliko manjih iznimaka u mogućnosti filtriranja, te razlike ćemo detaljnije opisati u budućim poglavljima. Postoje dvije stavke filtera koje ćemo iskoristiti kako bi dobili početne poveznice koje će služiti kao početna točka u Scrapy projektu. Te stavke su *Županija i Grad/Općina* koje postavljamo na *Istarska* i *Pula*. Nakon primjene filtera dobivamo konačnu poveznicu s koje će se podaci dohvaćati, *njuskalo.hr/prodaja-stanova/pula* i *njuskalo.hr/prodaja-kuca/pula*.

4.1.2. Ključni web elementi

Prolaskom kroz web stranice *njuskalo.hr/prodaja-stanova/pula* i *njuskalo.hr/prodaja-kuca/pula* , ističu nam se tri elementa od interesa.

Slika 11. Primjer Njuškalo oglasa



Izvor: https://www.njuskalo.hr/?ctl=help&content_id=49

Prvo su sami oglasi koji su strukturirani kao kartice. Svaka od kartica sadrži ime oglasa, koje je i poveznica na stranicu s dodatnim detaljima, cijenu nekretnine, kratki opis oglasa i datum objave. Cilj nam je pristupiti svakoj od ovih kartica, proći i skupiti podatke s nje i nastaviti na sljedeći oglas. Sljedeći element, potreban za prelazak na novu stranicu kartica s oglasima, je traka za navigaciju stranica. Nalazi se dnu i vrhu stranice. Svaki od elemenata trake sadrži poveznicu koja vodi na sljedeću listu oglasa dok gumb *Sljedeća* vodi na sljedeću stranicu ovisno o trenutnoj lokaciji. Ovu funkcionalnost ćemo iskoristiti kako bi prošli sve stranice s oglasima i dohvatili svaku od kartica na njima. Zadnji element je sam filter oglasa. Dok iz njega samog nećemo dohvaćati podatke, daje nam uvid u elemente koji će se prikazivati na stranicama s dodatnim detaljima što će nam olakšati posao u mapiranju vrijednosti koje ćemo dohvaćati.

Kako bi istražili strukturu ovih elemenata koristit ćemo alat koji je ugrađen u Chrome[14] pregledniku zvan Chrome alati za programere (engl. Devtools). To je kolekcija naprednih alata koji se koriste za analizu, testiranje i optimizaciju web stranica i aplikacija. Uključeni alati pružaju programerima mogućnost istraživanja strukture i sadržaja web stranica te manipulaciju podacima unutar preglednika. Jedan od ključnih alata je inspektor (engl. Inspector) koji omogućava vizualnu inspekciju HTML i CSS koda web stranice. Pomoću njega moguće je jednostavno pregledavati elemente, identificirati njihove specifične selektore i ciljati ih prilikom razvoja skripti za web scraping.

Tablica 4. Popis ključnih web elemenata

Element	Opis	Atribut	Upotreba
<article>	Blok koji sadrži oglas	class = "entity-body cf"	Grupiranje oglasa
<h3>	Naslovi oglasa u kojoj se nalaze poveznice za detaljan opis oglasa	class="entity-title"	Ekstrakcija naziva nekretnine
<a>	Poveznice do drugih stranica ili do detaljnijih informacija o oglasu	href	Navigacija do drugih stranica ili do detaljnijih informacija o oglasu

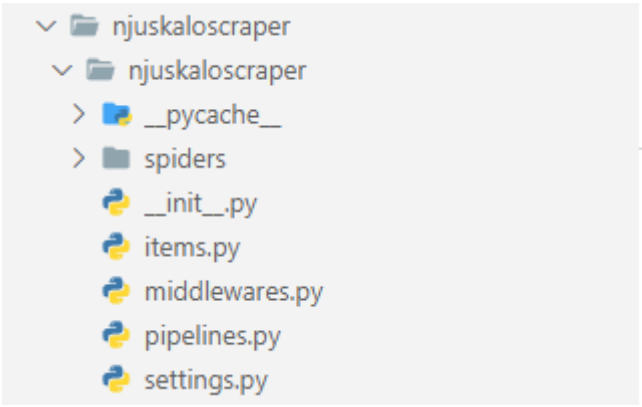
Izvor: izrada autora

[14] Chrome - popularni web preglednik razvijen od strane Googlea koji omogućava korisnicima pregledavanje interneta i pristup raznim web stranicama i aplikacijama.

4.2. Postavljanje projekta prikupljanja podataka

Kako bi krenuli s projektom, prvo moramo instalirati Scrapy okvir. Za razvoj procesa korišten je Visual Studio Code, univerzalni tekstualni editor. Pokretanjem terminala u okruženju editora možemo iskoristiti pip[15] paketni upravitelj kako bi instalirali Scrapy. Nakon uspješne instalacije paketa potrebno je stvoriti Scrapy projekt. Scrapy ima ugrađenu naredbu za stvaranje osnovne strukture projekta s osnovnim datotekama i mapama. Izvršavanjem naredbe *scrapy startproject njuskaloscraper* stvaramo naš projekt.

Slika 12. Struktura projekta



Izvor: izrada autora

U *setting.py* datoteci postavljamo nekoliko vrijednosti kako naše prikupljanje podataka ne bi naškodilo performansama stranice Njuskalo.hr. Varijablu *ROBOTSTXT_OBEY* stavljamo na *True* vrijednost kako bi pratili pravila pristupanja podacima sa stranice kako je vlasnik odredio te varijable *CONCURRENT_REQUESTS* i *DOWNLOAD_DELAY* postavljamo na 1 kako bi prikupljali podatke bez nepotrebnog opterećenja web poslužitelja.

[15] pip - alat za upravljanje paketima u programskom jeziku Python koji omogućuje jednostavno instaliranje, ažuriranje i uklanjanje Python paketa i njihovih ovisnosti.

4.3. Stvaranje Pauka

S obzirom da dohvaćamo oglase dvaju različitih tipova, kuće i apartmani, stvaramo dva različita pauka, to su Python datoteke *njuskalo_kuce.py* i *njuskalo_stanovi.py* . Ove datoteke će sadržavati logiku po kojoj će pauk pristupati stranicama, dohvaćati odabrane podatke i spremati ih u stavke koje će se kasnije prosljeđivati cjevovodu za daljnje čišćenje podataka te na kraju spremati u bazu podataka. Prve stavke koje moramo definirati su ime pauka, dopuštene domene na koje pauk može pristupati i početna stranica s koje pauk kreće dohvaćati podatke. Da pauk može obraditi podatke s početne stranice koje smo mu zadali moramo definirati *parse()* metodu u toj klasi. Ta metoda prima odgovor na zahtjev pristupa početnoj stranici koja sprema HTML kod stranice u objekt koji sadrži korisne metode i atribute za obradu odgovora.

Tablica 5. Popis atributa i metoda *response* objekta

Atribut/Metoda	Opis
body	Sadrži HTML sadržaj odgovora.
text	Pruža dekodirani tekstualni sadržaj odgovora.
status	Predstavlja HTTP statusni kod odgovora.
headers	Sadrži zaglavlja odgovora kao rječnik.
url	Sadrži URL odgovora.
request	Pruža pristup odgovarajućem Request objektu za trenutni odgovor.
xpath()	Omogućava izvođenje XPath upita na odgovoru kako bi se izdvojili određeni elementi ili podaci.
css()	Omogućava korištenje CSS selektora za izdvajanje podataka.
follow()	Omogućava praćenje veza unutar odgovora i generiranje daljnjih zahtjeva.

Izvor: izrada autora

Koristeći ove atribute i metode dohvatiti ćemo sve kartice sa stranice oglasa i na kojoj se trenutno stranici nalazimo.

Kartica oglasa u imenu sadrže poveznicu na detaljan opis nekretnine, definirati ćemo zasebnu metodu, *parse_property_page()*, koja će kao parametar primati tu poveznicu i u njoj napraviti logiku koja će prolaziti kroz sva bitna polja oglasa i spremati ih u prikladnu stavku. Za kuće će to biti *NjuskaloIKuceItemKeyFields* dok za stanove *NjuskaloStanoviItemKeyFields*. Te stavke će se onda proslijediti cjevovodu za obradu podataka *NjuskaloscraperPipeline* koji će podatke dodatno obraditi kako bi bili u formatu prikladnom za spremanje u bazu podataka čiji će proces biti definiran u cjevovodu *SaveAddsToMySQLPipeline*. Ove procese ćemo detaljnije obraditi u budućim poglavljima.

4.4. Definiranje stavki

Da bi definirali koje stavke želimo dohvatiti iz detaljnog opisa oglasa potrebno je ručno mapirati vrijednosti. Struktura svih oglasa za prodaju kuća i stanova je identična što nam olakšava stvaranje stavke jer uvijek možemo očekivati ista polja na oglasu. Filter oglasa na stranici Njuskalo.hr sadrži sva moguća polja koje se mogu pojaviti u oglasu. Na temelju njega stvoriti ćemo strukturu naših stavki. U datoteci *items.py* stvoriti ćemo dvije klase stavki koje smo prethodno spomenuli, *NjuskaloIKuceItemKeyFields i NjuskaloStanoviItemKeyFields*. Te stavke će imati sljedeće atribute prikazane u tablicama koje slijede.

Tablica 6. Osnovne informacije stavke Kuća

Atribut	Opis
posting_code	Jedinstven kod oglasa
posting_name	Ime oglasa
posting_price	Cijena oglasa u eurima
posting_link	Poveznica na oglas
location	Kvart u gradu gdje se oglas nalazi
city	Grad gdje se oglas nalazi
house_type	Koliko katova ima kuća
number_of_rooms	Broj soba
size_of_house	Veličina kuće u metrima kvadratnim
size_of_garden	Veličina okućnice u metrima kvadratnim
house_code	Šifra objekta
year_built	Godina izgradnje objekta
year_of_last_renovation	Zadnja godina renovacije
furniture_status	Kakav namještaj se nalazi u kući
energy_certificate	Razina energetskog certifikata
outer_part_of_house	Ima li objekt Balkon, Lođu ili Terasu

Izvor: izrada autora

Ovo su informacije koje se prve ističu na oglasu, neke od njih ćemo i iskoristiti kasnije u analizi oglasa. Nakon njih slijede opis i dodatne informacije o oglasu koje spremamo u sljedeće atribute.

Tablica 7. Dodatne informacije stavke Kuća

Atribut	Opis
date_of_posting	Datum objave
date_of_expiration	Datum isteka oglasa
number_of_views	Broj pregleda oglasa

Izvor: izrada autora

Struktura klase u koju će se spremati podaci o stanovima je slična prethodnoj, razlikuju se po nekim atributima koji su specifični za stanove za osnovne informacije dok su dodatne informacije identične. Slijedi tablice atributa za stavku stanova.

Tablica 8. Osnovne informacije stavke Stan

Atribut	Opis
posting_code	Jedinstven kod oglasa
posting_name	Ime oglasa
posting_price	Cijena oglasa u eurima
posting_link	Poveznica na oglas
location	Kvart u gradu gdje se oglas nalazi
city	Grad gdje se oglas nalazi
floor_location	Na kojem je katu stan
apartment_type	Tip stana
multiple_floors	Na koliko je etaža stan
number_of_rooms	Broj soba u stanu
size_of_living_area	Veličina stana u metrima kvadratnim
house_code	Šifra objekta
year_built	Godina izgradnje objekta
year_of_last_renovation	Zadnja godina renovacije
status_of_furniture	Kakav namještaj se nalazi u kući
energy_certificate	Razina energetskog certifikata
outer_part_of_apartment	Ima li objekt Balkon, Lođu ili Terasu

Izvor: izrada autora

4.5. Prikupljanje podataka s poveznica

Prije nego se podaci mogu spremiti u stavke, moramo dohvatiti sve poveznice oglasa. Logika za prolazak poveznica prikazana je u sljedećem kodu.

Slika 13. Definicija *parse()* metode

```
def parse(self, response):
  list_tag = 'article.entity-body.cf'
  listings = response.css(list_tag)
  page_tag = 'li.Pagination-item.Pagination-item-next.Pagination-item--nextSolo'
  next_page = response.css().get(page_tag)
  home tag = 'li.Pagination-item.Pagination-item--next'
  next_page_after_homepage = response.css(home_tag).get()
   for property_link in listings:
      property url = property link.css('h3 a').attrib['href']
      property_page = 'https://www.njuskalo.hr'+property_url
      if 'nekretnine' in property_page:
       match = re.search(r'oglas-(\d+)$', property_page)
       if match:
        oglas_number = match.group(1)
        value = oglas_number
        if self.is_duplicate(value):
          self.logger.info("Skipping duplicate page: %s", property_page)
          continue
        rand_index = random.randint(0,len(self.fake_user_agents_list)-1)
        yield response.follow(url=property_page
                        ,callback=self.parse_property_page
                        ,headers={"UserAgent":self.fake_user_agents_list[rand_index]})
      else:
        continue
```

Izvor: izrada autora

U *listings* atribut spremamo glavno tijelo stranice gdje se nalaze svi oglasi dok u atribute *next_page* i *next_page_after_homepage* spremamo elemente koji sadrže poveznice na sljedeće stranice oglasa. One se koriste kao provjera ako smo prošli sve stranice oglasa kako bi pauk mogao zaustaviti pretraživanje. Koristeći *for* petlju prolazimo kroz sve kartice oglasa na stranici i stvaramo poveznice za daljnje pristupanje stranicama s detaljima oglasa. U atributu *property_url* spremamo dio poveznice koji izvlačimo iz kartice oglasa, ta poveznica je u formatu */nekretnine/ime_oglasa*. Dodavanjem ovog nastavka na *property_page* atribut dobivamo punu poveznicu za stranicu koju želimo obraditi. Prije nego nastavimo na poveznicu radimo provjeru ako je ona tipa nekretnine. Ovo radimo iz razloga što na svim stranicama postoje promovirani oglasi s različitim stavkama koje nisu nužno nekretnine. Ovim korakom odbacujemo sve poveznice koje nisu vezane za nekretnine. Koristeći regularne izraze[16] (engl. Regex) obrađujemo poveznicu kako bi iz nje izvukli kod oglasa, taj kod šaljemo metodi *is_duplicate()* .

[16] Regularni izrazi - specifični obrasci ili uzorci koji se koriste za pretragu, provjeru ili manipulaciju teksta, omogućujući programerima da izvrše složene operacije pretraživanja i obrade teksta temeljene na tim uzorcima.

Slika 14. Implementacija *is_duplicate()* metode

```
def is_duplicate(self, posting_code):
  # Establish a connection
  conn = mysql.connector.connect(
            host = 'localhost',
            user = 'root',
            password ='admin',
            database ='njuskalo'
        )
   # Create a cursor object
   cur = conn.cursor()
   # Query database
   cur.execute("SELECT COUNT(*) FROM njuskalo_kuce WHERE posting_code = %s"
               ,(posting_code,))
   result = cur.fetchone()
   count = result[0]
   return count > 0
```

Izvor: izrada autora

Metoda radi jednostavan upit s kojim provjerava ako zapis već postoji u bazi, vraća *True* vrijednost ako je upit vratio zapis ili vraća *False* vrijednost ako zapis nije pronađen u bazi. Ako kod oglasa postoji u bazi podataka, preskačemo poveznicu i nastavljamo istu provjeru za sljedeću poveznicu. U slučaju da zapis ne postoji u bazi podataka, vraćamo poveznicu na novi oglas i obrađujemo je u metodi *parse_property_page()*. Isto dodajemo novi parametar, *fake_user_agents_list* .

Lažni korisnički agenti (engl. Fake User Agents) se odnose na izmišljene ili promijenjene informacije o identitetu i karakteristikama internetskog korisnika, poput preglednika ili uređaja. Korisnički agenti obično su dostupni web preglednicima te web stranicama omogućuju prilagođavanje prikaza sadržaja prema sposobnostima korisnikovog uređaja ili preglednika. Međutim, lažni korisnički agenti namjerno iskrivljuju te informacije kako bi prikrili pravu identifikaciju korisnika ili zaobišli određene restrikcije ili sigurnosne mjere. Mogu se koristiti u različite svrhe, uključujući anonimnost, zaobilazak detekcijskih mehanizama ili emulaciju određenih korisničkih okruženja.

Slika 15. Primjer polja korisničkog agenta

User Agent String explained :

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/114.0.0.0 Safari/537.36
```

Copy/paste any user agent string in this field and click 'Analyze' — Analyze

Chrome 114.0.0.0	
Mozilla	MozillaProductSlice. Claims to be a Mozilla based user agent, which is only true for Gecko browsers like Firefox and Netscape. For all other user agents it means 'Mozilla-compatible'. In modern browsers, this is only used for historical reasons. It has no real meaning anymore
5.0	Mozilla version
Windows NT 10.0	Operating System: Windows 10
Win64	(Win32 for 64-Bit-Windows) API implemented on 64-bit platforms of the Windows architecture - currently AMD64 and IA64
x64	64-bit windows version
AppleWebKit	The Web Kit provides a set of core classes to display web content in windows
537.36	Web Kit build
KHTML	Open Source HTML layout engine developed by the KDE project
like Gecko	like Gecko...
Chrome	Name : Chrome
114.0.0.0	Chrome version
Safari	Based on Safari
537.36	Safari build
Description:	Free open-source web browser developed by Google. Chromium is the name of the open source project behind Google Chrome, released under the BSD license.

Izvor: https://useragentstring.com/

Prilikom svakog novog zahtjeva za pristupanje poveznici, nasumično koristimo drugog korisničkog agenta kako bi izbjegli blokiranje pristupa na stranicu.

Slika 16. Implementacija automatske navigacije stranicama

```
if next_page is not None or next_page_after_homepage is not None:
 next_page_url ='https://www.njuskalo.hr/prodajakuca/pula?page='+str(self.starting_page)
 self.starting_page = self.starting_page+1
 rand_index = random.randint(0,len(self.fake_user_agents_list)-1
 yield response.follow(
        url=next_page_url
        ,callback=self.parse
        ,headers={"User-Agent":self.fake_user_agents_list[rand_index]}
        )
```

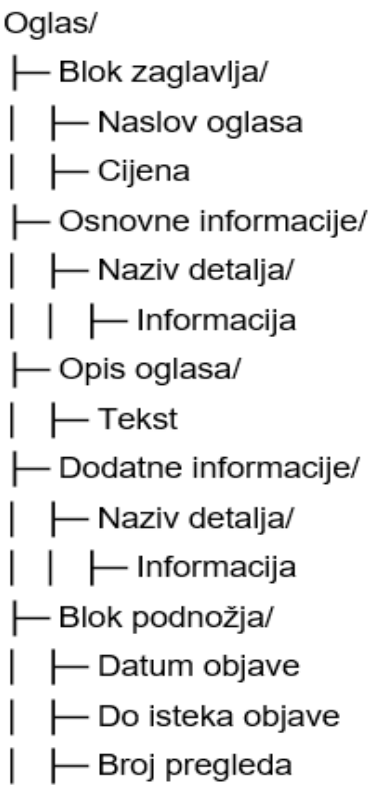
Izvor: izrada autora

Sada kada imamo sve potrebne poveznice s prve stranice oglasa, potrebno je dodati logiku za navigaciju po svim ostalim stranicama objava. Provjeravamo ako postoje elementi trake za navigaciju stranicama, dok god postoji jedan od ovih elemenata rekurzivno pozivamo *parse()* metodu dok god postoji novih stranica za obraditi.

4.6. Spremanje podataka u stavke

Nakon što je *parse()* metoda proslijedila objekt stranice s detaljima o oglasu možemo krenuti s procesom izvlačenja i spremanja tih detalja u stavke koje smo prethodno definirali. Struktura stranice podijeljena je u pet sekcija prikazanih na slici 17.

Slika 17. Struktura stranice s detaljnim opisom oglasa



Izvor: izrada autora

Postoje slučajevi gdje neke od ovih sekcija nisu navedene u oglasu, kako bi dinamički dohvaćali podatke kako prolazimo kroz stranicu napraviti ćemo rječnik *desc_value_pair*. U tom rječniku će ključna vrijednost biti *string* koji predstavlja element sa stranice a vrijednost će biti podatak koji je vezan za taj element. Prvu vrijednost koju spremamo je sam kod oglasa i njegovu poveznicu, kako bi dohvatili te vrijednosti možemo iskoristiti objekt *response* koji smo predali funkciji i na njemu pozvati metodu *response.url*

Slika 18. Spremanje koda i poveznice oglasa

```
# Get the code of the property from url
posting_link = response.url
match = re.search(r'oglas-(\d+)$', posting_link)
posting_number = match.group(1)
key = 'Kod oglasa'
value = posting_number
pair = {key: value}
desc_value_pair.append(pair)
# Get visited link
key = 'Weblink_oglasa'
value = posting_link
pair = {key: value}
desc_value_pair.append(pair)
```

Izvor: izrada autora

Uporabom regularnih izraza izvlačimo kod oglasa iz poveznice te odmah nakon toga uzimamo samu adresu poveznice i spremamo ih u rječnik.

Prateći strukturu stranice, sljedeći elementi koje dohvaćamo su sam naslov oglasa i cijena pod kojom je naveden. Koristeći sljedeći kod pristupamo tim podacima.

Slika 19. Spremanje cijene i naziva oglasa

```
# Get the price of the property posting
price_tag = '//div[contains(@class,"ClassifiedDetailSummary-pricesBlock")]/dl/dd'
posting_price = response.xpath(path)
key = 'Cijena_oglasa'
value = posting_price.css('dd ::text').get()
pair = {key: value}
desc_value_pair.append(pair)
# Get the name of the property posting
name_tag = '//div[contains(@class,"ClassifiedDetailSummary-titleRow")]'
posting_name = response.xpath(tag)
key = 'Naziv_oglasa'
value = posting_name.css('h1 ::text').get()
pair = {key: value}
desc_value_pair.append(pair)
```

Izvor: izrada autora

Pomoću XPath selektora izoliramo elemente na stranici koji sadrže potrebne podatke i kao prije spremamo ih u rječnik. Na ovim podacima još nismo radili dodatno čišćenje, taj korak će biti odrađen u cjevovodu stavki.

Dio stranice s osnovnim informacijama ima složeniju strukturu za razliku od prijašnjih elemenata. Kako bi dohvatili te podatke upotrijebiti ćemo kombinaciju CSS i XPath selektora kako bi spremili veće blokove HTML strukture.

Slika 20. Spremanje osnovnih informacija s oglasa

```
desc = '//div[contains(@class,"BlockStandardClassifiedDetailBasicDetails")]/dl/dt/span'
table_desc = response.xpath(desc)

val = '//div[contains(@class,"BlockStandard ClassifiedDetailBasicDetails")]/dl/dd/span'
table_values = response.xpath(val)

# Get values from Osnovne informacije table
for x in range(len(table_desc)):
     # Cleaning up key value
     key = table_desc[x].css('span ::text').get()
     key = re.sub(r"\s", "_", key)
     key = unidecode(key)
     value = table_values[x].css('span ::text').get()
     pair = {key: value}

     desc_value_pair.append(pair)
```

Izvor: izrada autora

Koristeći XPath selektor u varijablu *table_desc* spremamo listu imena za osnove informacije iz oglasa dok u *table_values* vrijednosti koje su navedene za te informacije. Pomoću *for* petlje prolazimo svako od ovih polja kako bi u rječnik dodali te vrijednosti. Prije nego dodamo ključ napravit ćemo standardiziranje naziva za ove informacije uporabom regularnih izraza. S ovim izrazom se iz imena informacije svi razmaci zamjenjuju podvlakama i odmah nakon toga koristimo *unicode()* funkciju kako bi zamijenili sve posebne hrvatske znakova za slova. Tako ime poput *Površina okućnice* postaje *Povrsina_okucnice*.

Opis oglasa je proizvoljno polje teksta gdje osoba koja je objavila oglas može navesti dodatne informacije o samoj nekretnini koje nisu predefinirane i kontakt informacije.

U procesu prikupljanja podataka s web stranice, posebna pažnja posvećena je poštivanju Opće uredbe o zaštiti podataka (GDPR). GDPR, koju je uvela Europska unija 2018. godine, postavlja stroge smjernice za prikupljanje, obradu i čuvanje osobnih podataka građana Europske unije. Prema ovom pravnom okviru, osobni podaci uključuju bilo kakve informacije koje mogu identificirati pojedinca, uključujući, ali ne ograničavajući se na, imena, e-mail adrese i telefonske brojeve. Unatoč tome što bi prikupljanje i analiza kontaktnih informacija pružila dodatni sloj detalja u našem istraživanju, odlučeno je da se takvi podaci neće prikupljati kako bi se izbjegle potencijalne komplikacije vezane uz GDPR. Bez odgovarajućeg pristanka od strane subjekata podataka, prikupljanje ovakvih podataka može biti u suprotnosti s odredbama GDPR-a. Također, zaštita tih podataka zahtijeva odgovarajuće tehničke i organizacijske mjere koje osiguravaju njihovu sigurnost od neovlaštenog pristupa, gubitka ili uništenja. Zbog tih razloga, a s obzirom na etičke smjernice ovog istraživanja, odlučeno je izbjeći prikupljanje bilo kakvih osobnih podataka tijekom web scraping procesa. Ova odluka omogućava nam da se usredotočimo na prikupljanje i analizu podataka koji ne uključuju osobne informacije, a istovremeno osiguravamo poštovanje prava i privatnosti pojedinaca.

Od informacija s oglasa nam preostaje blok podnožja. Tu možemo pronaći tri informacije, kada je oglas objavljen, koliko vremena je preostalo do isteka oglasa te koliko puta je sama objava bila pregledana. Koristeći sličnu metodu kao što je upotrijebljena za blok s osnovnim informacijama dohvaćamo zadnje vrijednosti s oglasa.

Slika 21. Spremanje informacija iz podnožja oglasa

```
date_and_view_table_desc = response.xpath(name_tag)


value_tag = '//div[contains(@class,"BlockStandard ClassifiedDetailSystemDetails cf")]/dl/dd'
date_and_view_table_value = response.xpath(value_tag)
  for x in range(len(date_and_view_table_desc)):
    key = date_and_view_table_desc[x].css('dt ::text').get()

    # Cleaning up our key values
    key = key.replace("\n", "").strip()
    key = re.sub(r"\s", "_", key)
    key = unidecode(key)

    value = date_and_view_table_value[x].css('dd ::text').get()
    pair = {key: value}
    desc_value_pair.append(pair)
```

Izvor: izrada autora

Nakon što imamo sve informacije oglasa u našem rječniku, kreirati ćemo Scrapy stavku koja će onda biti proslijeđena cjevovodu stavki za konačnu obradu prije nego podatak o oglasu spremimo u bazu.

Slika 22. Mapiranje prikupljenih vrijednosti u stavku

```
# Extract the unique field names from the list
field_names = set().union(*(item.keys() for item in desc_value_pair))
# Create a dynamic Scrapy Item class
fields = {field: scrapy.Field() for field in field_names}
ItemClass = type('MyItem', (scrapy.Item,), fields)
# Instantiate an item object
item = ItemClass()
# Assign values to the item fields
for pair in desc_value_pair:
    for field, value in pair.items():
        item[field] = value
items_to_extract = { ... }
njuskalo_item = {}
for key, value in items_to_extract.items():
    try:
      njuskalo_item[key] = item[value]
    except KeyError:
      njuskalo_item[key] = None
yield njuskalo_item
```

Izvor: izrada autora

U ovom kodu, prvo se iz rječnika *desc_value_pair* izvlače jedinstvena imena polja i spremaju u skup *field_names*. Zatim se stvara rječnik *fields* koji mapira svako ime polja na *scrapy.Field()* objekt. Na temelju ovog rječnika, dinamički se stvara klasa *ItemClass* koja nasljeđuje *scrapy.Item* i ima ta polja. Nakon toga, stvara se instanca objekta stavke pripadajuće klase *ItemClass*. Vrijednosti polja se dodjeljuju prolaskom kroz vrijednosti *desc_value_pair* rječnika koristeći njegove ključeve i vrijednosti. Zatim se koristi rječnik *items_to_extract* koji sadrži mapiranje koje povezuje polja stavke s poljima koje je pauk prikupio sa stranice. Stvara se prazan rječnik *njuskalo_item*, te se iterira kroz parove ključeva i vrijednosti u *items_to_extract*. Pokušava se pristupiti vrijednosti objekta *item* koristeći *value* kao ključ i onda se ta vrijednost dodjeljuje *njuskalo_item* stavci pod ključem *key*. Ako ključ ne postoji u objektu *item*, postavlja se vrijednost *None* za taj ključ u *njuskalo_item* što znači da na stranici nije bilo podatka o tom polju. Na kraju vraćamo *njuskalo_item* stavku koja se prosljeđuje na daljnju obradu cjevovodu stavki.

4.7. Dodatno čišćenje stavki

Postoji mnogo razloga zbog kojih se izvor podataka ne može koristiti izravno. Na primjer, ako postoji previše deskriptivnih varijabli, potrebno je primijeniti algoritme za odabir značajki ili smanjenje dimenzionalnosti. Ako su podaci preveliki, mogu se koristiti tehnike pod uzorkovanja. Kod neuravnoteženih skupova podataka, prekomjerno uzorkovanje ili pod uzorkovanje mogu pomoći. Jedan od najčešćih razloga za izmjenu sirovih podataka su nedostajuće ili netočne vrijednosti. Najčešći pristupi za suočavanje s tim problemom su odbacivanje redaka s nedostajućim ili netočnim podacima ili imputacija, tj. zamjena nedostajućih vrijednosti procijenjenim vrijednostima na temelju dostupnih podataka (Munappy, Bosch & Olsson, 2020).

Kako bi dodatno uredili podatke koje smo dohvatili s oglasa, koristimo cjevovod stavki koji pruža mogućnosti za obradu, transformaciju i pohranu podataka prikupljenih tijekom procesa. Njegove prednosti uključuju fleksibilnost, modularnost, mogućnost složene obrade, provjeru kvalitete podataka, učinkovito rukovanje velikim količinama podataka i mogućnost automatske pohrane u različite izvore ili formate. U Scrapy projektu unutar *pipelines.py* datoteke definiramo klasu *NjuskaloscraperPipeline* koja će sadržati logiku za daljnje čišćenje podataka.
Unutar te klase definirana je metoda *process_item()* koja kao vrijednosti prima stavku s podacima i objekt pauka koji je prikupio podatke. Pomoću proslijeđenog objekta pauka možemo utvrditi ako je pauk skupljao podatke za stanove ili kuće te ovisno o tome koristimo definirane korake za obradu podataka.

Slika 23. Izdvajanje kvarta i grada iz polja Lokacija

```
if spider.name == 'njuskalo_kuce':
    adapter = ItemAdapter(item)
    # Get the neighbourhood and city name from Lokacija
    # location example : Istarska, Pula, Štinjan
    neighbourhood_name = adapter.get('location')
    parts = neighbourhood_name.split(",")
    neighbourhood_name_last = parts[-1].strip()
    city_name = parts[1].strip()
    adapter['location'] = neighbourhood_name_last
    adapter['city'] = city_name
```

Izvor: izrada autora

U ovom bloku koda, stvara se instanca klase *ItemAdapter* s argumentom *item*, koji predstavlja trenutnu stavku koju pauk prikuplja. Nakon toga, koriste se metode *ItemAdapter* klase za pristup i manipulaciju podacima unutar stavke što nam olakšava pisanje, čitanje i izmjenu vrijednosti polja unutar stavke. Ovaj kod izvlači podatke iz polja *location* iz stavke prikupljene od strane pauka. Zatim dijeli vrijednost na dijelove koristeći zarez kao separator i izvlači ime susjedstva i ime grada iz te vrijednosti. Naposljetku, ažurira vrijednosti polja *location* i *city* u stavci s dobivenim podacima. Kod obrade podataka za stanove proces je isti ali je odvojen u zasebni blok zbog obrade polja koje su specifične za stanove.

Slika 24. Čišćenje polja za cijenu i veličinu nekretnina

```
# Extract the first number using regular expression
price_value = adapter.get('posting_price')
matches = re.findall(r'\d{1,3}(?:[.,]\d{3})*', price_value)
first_number = int(matches[0].replace('.', ''))
adapter['posting_price'] = first_number

# Extract the property size and garden size
house_size_value = adapter.get('size_of_house')
property_size_value = adapter.get('size_of_garden')
matches = re.findall(r"\d+,\d+", house_size_value)
extracted_size = float(matches[0].replace(",", "."))
adapter['size_of_house'] = extracted_size

matches = re.findall(r"\d+,\d+", property_size_value)
extracted_size_property = float(matches[0].replace(",", "."))
adapter['size_of_garden'] = extracted_size_property
```

Izvor: izrada autora

Ovaj kod koristi regularne izraze za ekstrakciju određenih brojevnih vrijednosti i ažuriranje odgovarajućih polja u stavci. Prvo dohvaćamo vrijednost za cijenu oglasa iz stavke koju je napravio pauk. Zatim se koristi *re.findall()* funkcija s regularnim izrazom kako bi se pronašle sve brojne vrijednosti u varijabli. Rezultat se sprema u varijablu *matches*. Ako je dužina ove varijable veća od nula, to znači da je pronađen barem jedan broj. U tom slučaju, prva pronađena vrijednost se izdvaja, uklanjaju se eventualne točke kao tisućini separatori pomoću *replace()* metode i pretvara se u cjelobrojnu vrijednost. Dobivena vrijednost se sprema u varijablu *first_number* i polje *posting_price* u stavci se ažurira na tu vrijednost. Slično tome, koristi se *re.findall()* za pronalaženje brojnih vrijednosti u varijablama *house_size_value* i *property_size_value*, koje predstavljaju vrijednosti polja *size_of_house* i *size_of_garden* u stavci. Ako je pronađena barem jedna brojna vrijednost, prva pronađena vrijednost se izdvaja, zamjenjuju se zarezi s točkama kao decimalnim separatorima i pretvara se u brojčanu vrijednost kako bi ju kasnije mogli koristiti za analizu. Dobivena vrijednost se sprema u varijable *extracted_size* i *extracted_size_property* . Naposljetku se ta polja u stavci ažuriraju na odgovarajuće dobivene vrijednosti.

4.8. Spremanje u bazu podataka

MySQL je popularni otvoreni relacijski sustav upravljanja bazama podataka (engl. Reletional Database Management System) koji je distribuiran, razvijen i podržan od strane tvrtke Oracle Corporation. Relacijski sustavi poput MySQL-a pohranjuju podatke u tabličnom obliku i koriste strukturirani upitni jezik (engl. Structured Query Language) za pristup podacima (Damodaran B, Salim & Vargese, 2016).

Razvijen je za pohranu, upravljanje i manipulaciju strukturiranim podacima. Koristi se u širokom rasponu aplikacija i sustava, uključujući web aplikacije, poslovne sustave, e-trgovinu, mobilne aplikacije i još mnogo toga. Svojom popularnošću, pouzdanošću i skalabilnošću, MySQL je postao jedan od najčešće korištenih sustava upravljanja bazama podataka u industriji. Može efikasno upravljati velikim količinama podataka i pruža optimizaciju upita radi bržeg izvršavanja. Sustav podržava indekse koji ubrzavaju pretraživanje i izvlačenje podataka iz baze. MySQL baza podataka pruža pouzdanu i skalabilnu platformu za upravljanje podacima.

Kako bi spremali podatke s Njuškala koje smo prikupili koristeći Scrapy, stvoriti ćemo lokalno MySQL bazu podataka nazvanu *njuskalo* koja će sadržavati dvije tablice, *njuskalo_kuce* i *njuskalo_stanovi*. Samu izradu tablice napraviti ćemo kroz kod koristeći novu klasu cjevovoda stavki nazvanu *SaveAddsToMySQLPipeline*.

Slika 25. Spajanje na MySQL bazu i kreiranje tablica

```
import logging
import mysql.connector
class SaveAddsToMySQLPipeline:
    logger = logging.getLogger(__name__)

    def __init__(self):
        self.conn = mysql.connector.connect(
            host = 'localhost',
            user = 'root',
            password ='admin',
            database ='njuskalo'
        )

        self.cur = self.conn.cursor()
    # Create njuskalo kuce table if it does not exist
        self.cur.execute("""
            CREATE TABLE IF NOT EXISTS njuskalo_kuce ...
            """)
    # Create njuskalo stanovi table if it does not exist
        self.cur.execute("""
            CREATE TABLE IF NOT EXISTS njuskalo_stanovi ...
               """)
```

Izvor: izrada autora

Ovaj kod predstavlja klasu *SaveAddsToMySQLPipeline* koja se koristi kao cjevovod u Scrapy procesu za spremanje podataka u MySQL bazu. U dijelu konstruktora [17] inicijalizira se veza s MySQL bazom podataka pomoću *mysql.connector* biblioteke. Specifične informacije o vezi koje su potrebne za spajanje na bazu podataka su definirane u konfiguraciji. Nakon uspostave veze, dobiva se kursor (engl. Cursor) koji se koristi za izvršavanje SQL naredbi. Zatim se izvršavaju SQL naredbe za stvaranje tablica u bazi podataka. Ova naredba provjerava postoji li već tablica s tim imenom, a ako ne postoji, stvara je.

[17] Konstruktor – posebna metoda u Pythonu koja se automatski poziva prilikom stvaranja nove instance klase i koristi se za inicijalizaciju objekta postavljanjem početnih vrijednosti atributa.

Na slici 26 prikazan je dijagram tablica u njuskalo bazi.

Slika 26. Tablice u MySQL bazi

Izvor: izrada autora

Tablice su jednostavnog dizajna s obzirom da prikupljanje podataka radimo samo na području grada Pule. Odvajamo oglase po tipu nekretnine i spremamo prikladne stavke u njih.

Nakon što su tablice kreirane, slijedi metoda *process_item()* koja će se spajati na bazu podataka i spremati stavke u prikladnu tablicu.

Slika 27. Unos podataka u MySQL bazu

```
def process item(self,item,spider):
  try:
        if spider.name == 'njuskalo_stanovi':
            self.cur.execute("""
                INSERT INTO njuskalo_stanovi ...
                """
        if spider.name == 'njuskalo_kuce':
            self.cur.execute("""
                INSERT INTO njuskalo_kuce ...
                """
        self.conn.commit()
        self.logger.info("Item inserted into database: %s",
        item['posting_code'])
  except Exception as e:
        self.logger.error("Error during insert: %s", item['posting_code'])
        self.logger.error(traceback.format_exc())

  return item
```

Izvor: izrada autora

Kod prvo provjerava ime pauka kako bi odredio koji tip oglasa se obrađuje, stanovi ili kuće. Ovisno o imenu pauka koji je proslijeđen metodi, izvršava se *SQL INSERT* naredba za umetanje podataka iz stavke u tablicu u bazi podataka. Nakon izvršavanja SQL naredbe za umetanje, poziva se *self.conn.commit()* kako bi se potvrdile promjene u bazi podataka. U bloku *try-except*, hvataju se moguće iznimke koje se mogu dogoditi prilikom izvršavanja SQL naredbe. Ako se dogodi iznimka, koristi se *logger* za zapisivanje pogreške u *log* datoteku. Ovdje se koristi šifra oglasa za identifikaciju stavke koja nije uspješno spremljena.

5. Analiza podataka

U ovom poglavlju provodimo analizu podataka koje smo prikupili putem web scraping-a oglasa za kuće i stanove na području grada Pule. Fokus ove analize je pružiti uvid u prosječne tražene cijene nekretnina ovisno o njihovom tipu, kao i identificirati prosječne cijene po kvartovima. Također ćemo izraditi tablice i grafove koje prikazuju broj oglasa po kvartovima i istražiti kako se kreću cijene i veličine nekretnina ovisno o lokaciji.

5.1. Matplotlib i Seaborn

Matplotlib je Python paket za 2D crtanje grafova koji generira grafikone visoke kvalitete. Podržava interaktivno i ne interaktivno crtanje te može spremiti slike u nekoliko izlaznih formata (PNG, PS i ostale). Može koristiti više prozorskih alata (GTK+, wxWidgets, Qt i drugi) i pruža širok spektar vrsta grafova (linije, stupčasti grafikoni, kružni grafikoni, histogrami i mnoge druge). Osim toga, vrlo je prilagodljiv, fleksibilan i jednostavan za korištenje (Tosi, 2009).

Matplotlib podržava različite vrste grafikona, ova raznolikost omogućava korisnicima da vizualiziraju podatke na način koji najbolje odgovara njihovim specifičnim potrebama istraživanja. Isto tako nudi visoku razinu prilagodljivosti za svaki grafikon, omogućavajući korisnicima da kontroliraju sve aspekte grafikona, uključujući veličinu, boje, stilove linija, oznake osi i ostale detalje s grafikona. To omogućava korisnicima da prilagode svoje grafikone kako bi bili jasni, lako čitljivi, i estetski privlačni.

Seaborn predstavlja Python biblioteku specijaliziranu za generiranje vizualizacija podataka. Razvijena kao nadogradnja nad Matplotlib-om, ova biblioteka omogućuje istraživaćima i analitičarima da istražuju i prezentiraju raznolike aspekte skupova podataka. Seaborn se ističe intuitivnim sučeljem koje omogućuje brzu izradu različitih vrsta grafova i vizualizacija uz minimalno kodiranje. Dodatno, Seaborn uključuje podršku za kompleksne statističke analize, olakšavajući prikaz rezultata istraživanja na vizualno jasan način.

5.2. Pregled prikupljenih podataka

U periodu od 20.06.2023 do 25.06.2023 prikupljeno je ukupno 3273 oglasa za nekretnine na području grada Pule. Kako bi se izbjegao utjecaj ekstremnih vrijednosti na prosjek, sve vrijednosti prikupljenih podataka su medijalne. Također, iz uzorka podataka su maknute sve nekretnine koje su objavljene po cijeni ispod 10 000 eura i s kvadraturom manjom od 15 metara kvadratnih. Od ukupnog broja, 766 oglasa su za kuće dok 2507 oglasa su za stanove. Stanovi čine čak 77% ponude na području Pule. Slijedi tablični prikaz podataka grupiran po tipu nekretnine i kvarta u kojem se nalaze te njihovim prosječnim traženim cijenama, kvadraturi, prosječnoj cijeni po metru kvadratnom i ukupnim brojem oglasa.

Tablica 9. Prikaz prikupljenih podataka o kućama na području grada Pule

Kvart	Cijena	m2	euro/m2	Broj Oglasa
Centar	601977	257	2511	224
Štinjan	714976	333	2233	93
Veli vrh	654662	293	2275	92
Valdebek	497286	290	1944	49
Šijana	452283	265	1744	45
Nova Veruda	684739	393	1805	44
Šikici	546302	202	2924	43
Veruda	656921	366	1856	34
Vidikovac	470979	289	1898	24
Gregovica	613841	315	2101	22
Kaštanjer	521556	271	2006	18
Busoler	469938	233	2169	16
Stoja	756875	269	2596	12
Monte Magno	552650	261	2228	10
Montešerpo Komunal	447740	201	2285	10
Monvidal	758167	250	2451	9
Škatari	540750	236	2751	6
Padulj	565000	167	3605	3
Valmade	177467	149	1114	3
Arsenal	379000	150	2527	2
Sv. Polikarp / Sisplac	1300000	506	2569	1
Valkane	700000	453	1545	1
Monte Zaro	520000	195	2667	1
Pragrande	495000	164	3018	1
Ilirija	450000	200	2250	1
Valsaline	779000	524	1486	1
Monteghiro	420000	290	1448	1

Izvor: https://www.njuskalo.hr/

Tablica 10. Prikaz prikupljenih podataka o stanovima na području grada Pule

Kvart	Cijena	m2	euro/m2	Broj Oglasa
Centar	228241	83	2871	1121
Šijana	180563	76	2470	202
Veruda	223183	76	3206	199
Valdebek	199365	75	2671	183
Veli vrh	193543	77	2639	122
Vidikovac	223568	79	2875	116
Kaštanjer	227832	86	2640	87
Monvidal	148397	57	2636	86
Nova Veruda	264391	75	3808	73
Štinjan	222536	84	2924	66
Stoja	204806	70	2958	62
Gregovica	201076	73	2862	45
Monte Zaro	207380	95	2521	39
Monte Magno	241546	94	2903	35
Valmade	202731	73	2755	16
Busoler	184884	70	2674	14
Sv. Polikarp / Sisplac	274624	71	3815	14
Šikici	191375	81	2473	8
Verudela	252400	57	4293	5
Arsenal	211250	91	2490	4
Monteghiro	228667	83	2718	3
Valkane	374567	88	4488	2
Valsaline	299250	59	5073	2
Padulj	390000	142	2754	1
lirija	232500	100	2325	1
Škatari	192000	108	1770	1

Izvor: https://www.njuskalo.hr/

Tablica 9 i Tablica 10 predstavljaju statističke podatke o oglasima nekretnina u Puli, posebno klasificirane po kvartovima. Prva tablica koncentrira se na kuće, dok druga tablica pruža usporedne informacije o stanovima. Podaci za kuće i stanove uključuju ime kvarta, prosječnu traženu cijenu nekretnine, prosječnu površinu, prosječnu cijenu po metru kvadratnom i broj oglasa. Zbog ovog dizajna, tablice omogućuju direktne usporedbe između broja, veličine i tražene cijene nekretnina, kao i zastupljenost pojedinih tipova nekretnina na tržištu po kvartovima.

Kroz prosječne tražene cijene kuća i stanova po kvartovima, moguće je dobiti uvid u ekonomsku distribuciju nekretnina na području Pule. Prosjeci površina kuća i stanova u svakom kvartu mogu nam pružiti uvid u tipične veličine nekretnina u tom području. Prosjek cijene po metru kvadratnom omogućava analizu stvarne vrijednosti nekretnina u odnosu na njihovu veličinu, što može pomoći u identifikaciji kvartova s boljom vrijednosti za novac. Također, broj oglasa može nam pokazati kvartove s većom ili manjom aktivnošću na tržištu nekretnina.

Proučavanjem tablice 9, koja prikazuje podatke o kućama u različitim kvartovima Pule, možemo dobiti uvid u tržište nekretnina u različitim kvartovima grada.

Na temelju ovih podataka može se uočiti da kvart Centar, s 224 oglasa, dominira tržištem nekretnina, dok su kvartovi kao što su Štinjan i Veli Vrh također relevantni s 93, odnosno 92 oglasa. Ovi kvartovi pokazuju visoku aktivnost na tržištu nekretnina, što ukazuje na visoku razinu interesa za ove lokacije.

Tražene cijene nekretnina i cijene po kvadratnom metru variraju u ovim kvartovima. Na primjer, prosječna tražena cijena nekretnine u Centru iznosi 601.977 eura, dok je u Štinjanu 714.976 eura, iako je kvadratura nekretnina u Štinjanu veća. Ovo pokazuje kako lokacija može utjecati na traženu cijenu nekretnine. Ovi podaci također otkrivaju da postoji niz kvartova s vrlo malim brojem oglasa, samo jedan oglas po kvartu. Ovi kvartovi uključuju Sv. Polikarp / Sisplac, Valkane, Monte Zaro, Pragrande i Ilirija. Ovo može sugerirati da je aktivnost na tržištu nekretnina u tim kvartovima relativno niska.

Analizom tablice 10, koja prikazuje podatke o stanovima na području Pule, jasno se može primijetiti kako vrijednosti nekretnina i cijene po kvadratnom metru variraju između različitih kvartova. Vidljivo je da kvart Centar, s najvećim brojem oglasa (1121), ima prosječnu traženu cijenu stana od 228,241 eura, s prosječnom veličinom od 83 metra kvadratna, što daje prosječnu cijenu po kvadratnom metru od 2,871 eura. Uočeno je da cijena nije uvijek direktno proporcionalna veličini nekretnine, što ukazuje na druge faktore koji utječu na cijene nekretnina, poput lokacije, kvalitete objekata i opće atraktivnosti određenog kvarta.

Primjerice, kvart Centar, koji je najaktivniji kvart prema broju oglasa, ima prosječnu cijenu nekretnine od 228.241 eura. S druge strane, kvart Šijana, koji također pokazuje visoku aktivnost, ima prosječnu cijenu nekretnine od 180.563 eura, usprkos sličnim kvadraturama nekretnina. Kvart Nova Veruda, iako ima manje oglasa (73 oglasa), pokazuje značajno veću prosječnu traženu cijenu nekretnine od 264.391 eura. Ovaj podatak sugerira da kvart Nova Veruda možda nudi nekretnine bolje kvalitete ili je općenito atraktivnija zbog drugih faktora, poput blizine sadržaja ili prirodnih ljepota.

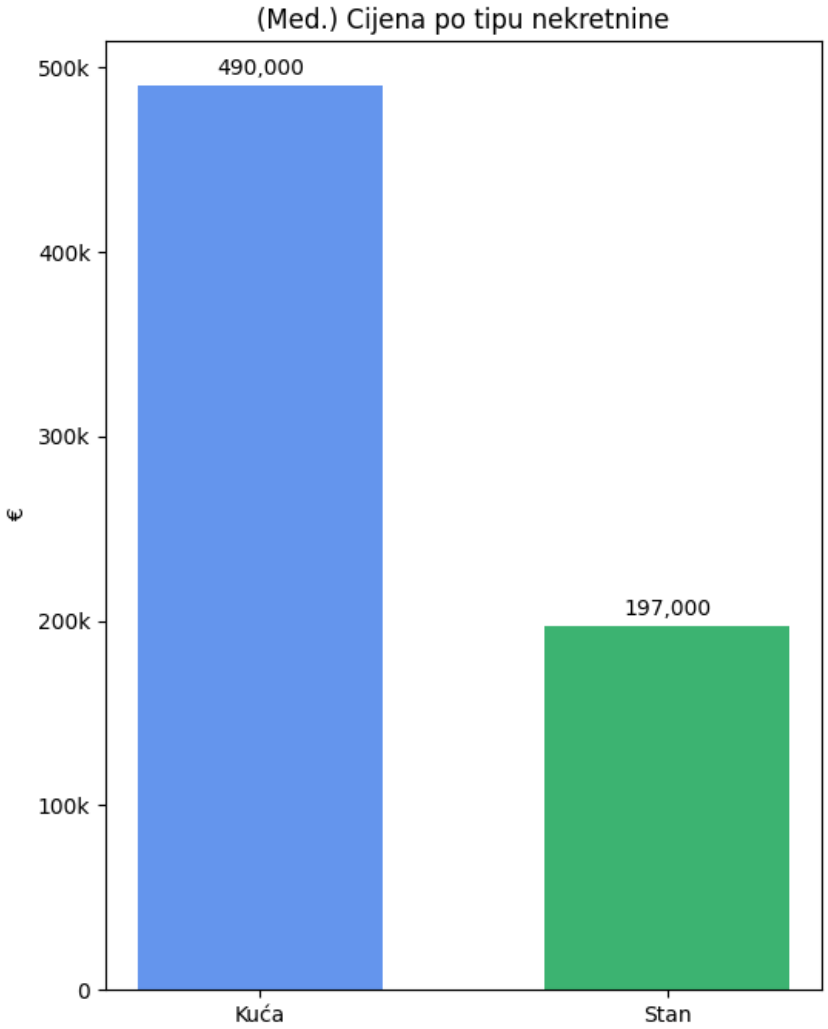
Uz to, možemo usporediti i cijene po kvadratnom metru. Za primjer, u kvartu Veruda, cijena po kvadratnom metru iznosi 3206 eura, što je više u usporedbi s kvartom Centar, gdje je cijena po kvadratnom metru 2871 euro iako je prosječna cijena nekretnine manja. Ovo ukazuje na potencijalno veću gustoću stanovanja ili veću cijenu nekretnine u kvartu Veruda. Na kraju, vrijedno je spomenuti da kvartovi s manje oglasa, kao što su Padulj, Ilirija i Škatari, iako možda nisu predstavljene s velikim brojem nekretnina ne treba zanemariti. S obzirom na njihovu manju količinu oglasa, moguće je da ove kvartovi nude specifične mogućnosti koje se ne mogu naći u drugim kvartovima.

Dok se kvadratura nekretnine sigurno odražava u cijeni nekretnine, drugi čimbenici kao što su lokacija, kvaliteta i atraktivnost kvarta također igraju ključnu ulogu u određivanju cijena na tržištu nekretnina. Daljnje istraživanje o ovim čimbenicima moglo bi omogućiti bolje razumijevanje ovog dinamičnog tržišta tako da se u analizu uključe dodatni čimbenici poput opremljenosti stana, godine renovacije, blizina atraktivnih sadržaja i ostalih.

5.3. Grafički prikaz podataka

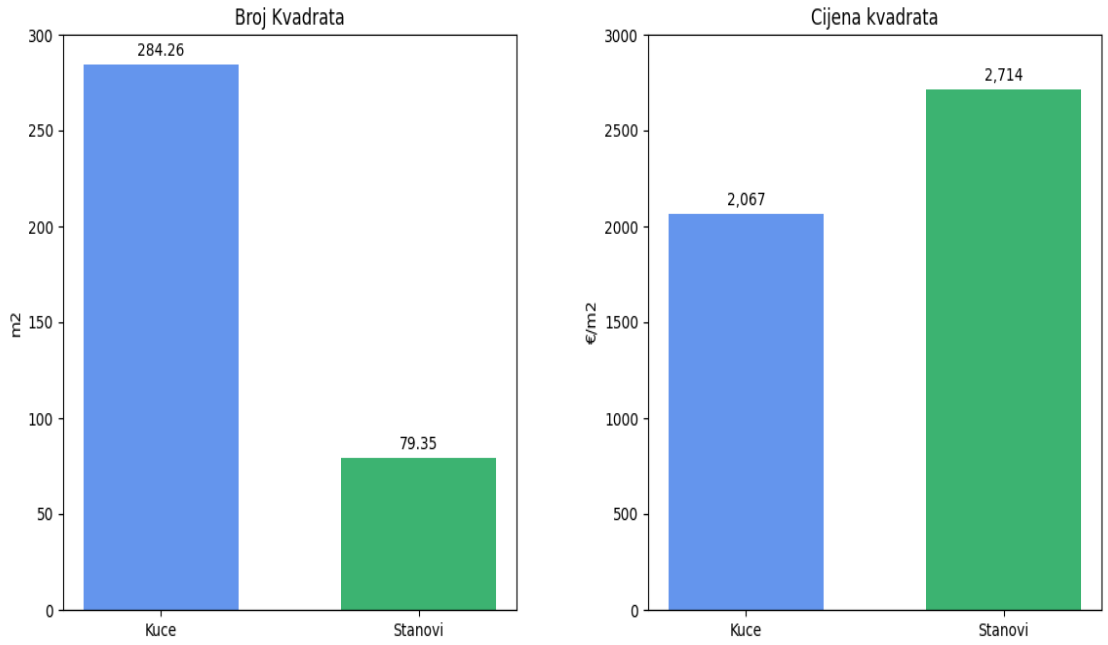
Korištenjem Matplotlib i Seaborn biblioteka izrađeni su grafički prikazi kako bi se bolje vizualizirali prikupljeni podaci.

Slika 28. Grafički prikaz prosječne cijene ovisno o tipu nekretnine



Izvor: https://www.njuskalo.hr/

Slika 29. Grafički prikaz prosjeka broja kvadrata i cijene kvadrata ovisno o tipu nekretnine



Izvor: https://www.njuskalo.hr/

Grafovi na slikama 28 i 29 nam pokazuju prosječnu traženu cijenu, broj kvadrata i cijenu kvadrata za sve prikupljene oglase grupirane po tipu nekretnine. S obzirom na veću kvadraturu kuća, razlika u prosječnim cijenama za kuće i stanove je očekivana. Iz grafa o cijeni kvadrata možemo primijetiti da su prosječne cijene po metru kvadratnom za stanove čak 24% veće od onih za kuće. Stanovi se često nalaze u urbanoj sredini, blizu središta grada ili u gusto naseljenim područjima. Takva urbana okruženja često imaju veću potražnju zbog blizine komercijalnih i društvenih sadržaja, javnog prijevoza, škola, zdravstvenih ustanova i drugih pogodnosti. Kao rezultat, cijena metra kvadratnog za stanove može biti veća zbog atraktivnosti takvih lokacija.

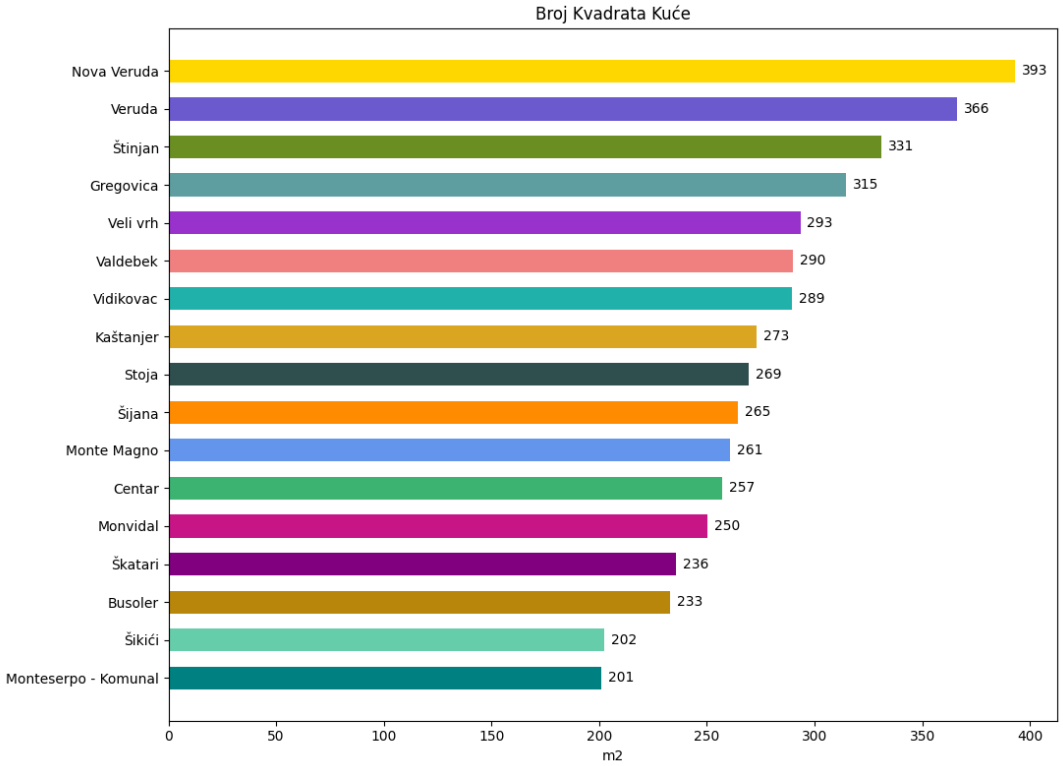
U nastavku poglavlja slijede stupčasti i horizontalni kutijasti dijagrami koji pokazuju usporedbu prosječne tražene cijene, broja kvadrata i cijena kvadrata ovisno o tipu nekretnine.

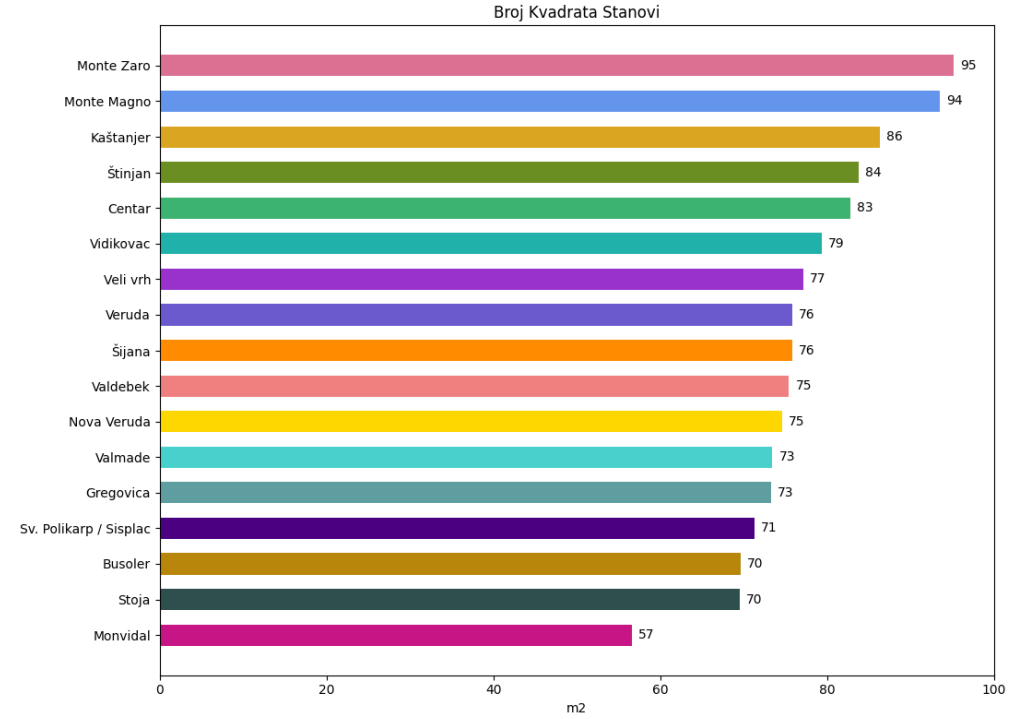
Horizontalni kutijasti dijagrami pružaju vizualni prikaz distribucije, središnje tendencije, raspršenosti te potencijalnih odstupanja u cijenama i veličinama nekretnina smještenih unutar grada Pule. Središnja crta, koja predstavlja medijan, pruža pokazatelj srednjeg vrijednosnog položaja podataka, dok interkvartilni raspon (engl. Interquartile range) obuhvaćen okvirom ističe raspon koji obuhvaća središnjih 50% podataka. Izdvojeni odstupnici, prepoznatljivi kao pojedinačne točke izvan brkova okvira, mogu ukazivati na ekstremne vrijednosti. Duljina brkova ilustrira opseg distribucije podataka unutar 1,5 puta interkvartilnog raspona. Nadalje, koristeći stripplot[18] dopunjujemo ovu reprezentaciju nudeći vizualno razumijevanje gustoće i grupiranja točaka podataka duž x-osi, koja nam pomaže u prepoznavanju potencijalnih grupiranja ili skupina nekretnina sa sličnim cijenama ili veličinama. Ova kombinirana analiza olakšava uvid u karakteristike tržišta nekretnina u gradu Pula.

Za kuće su izbačeni kvartovi koji imaju manje od 5 oglasa, dok za stanove oni koji imaju manje od 10 oglasa.

[18] Stripplot - u Seabornu je vrsta grafa koja prikazuje pojedinačne podatke kao točke raspoređene duž jedne osi, često se koristi za vizualizaciju distribucije i gustoće podataka.

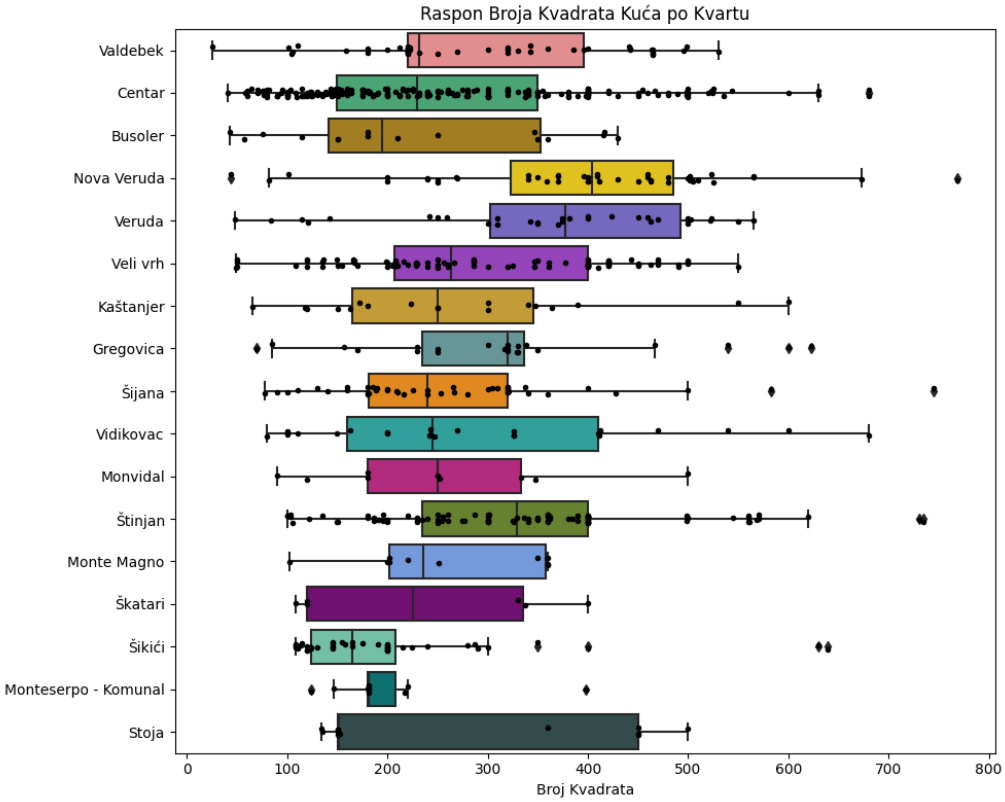
Slika 30. Grafički prikaz prosječne kvadrature nekretnina

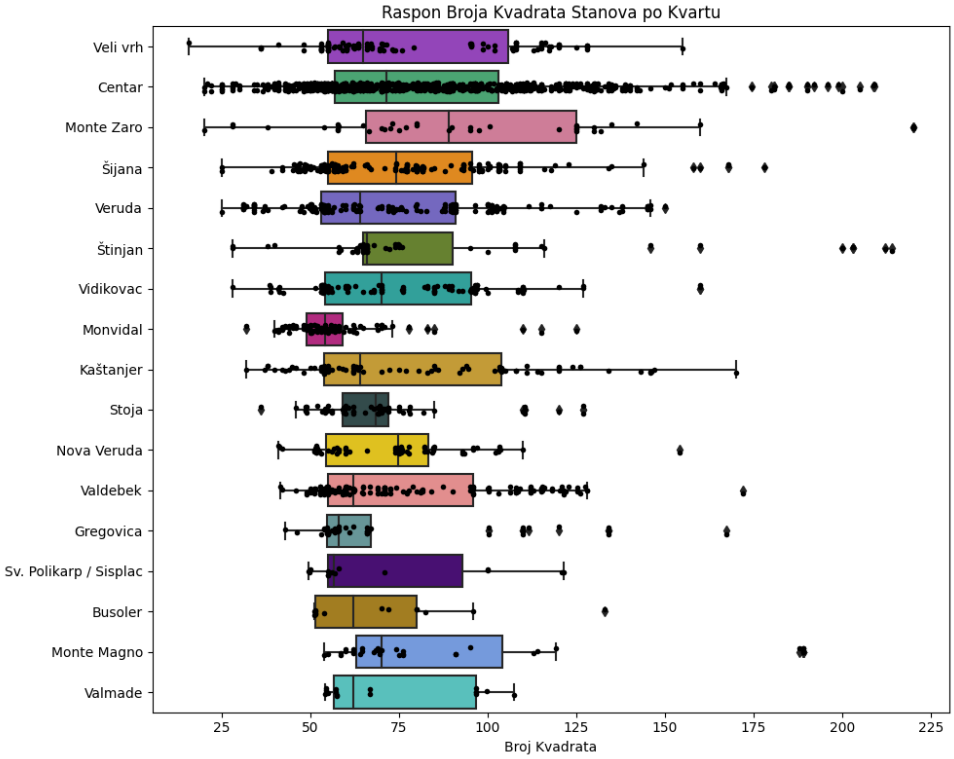




Izvor: https://www.njuskalo.hr/

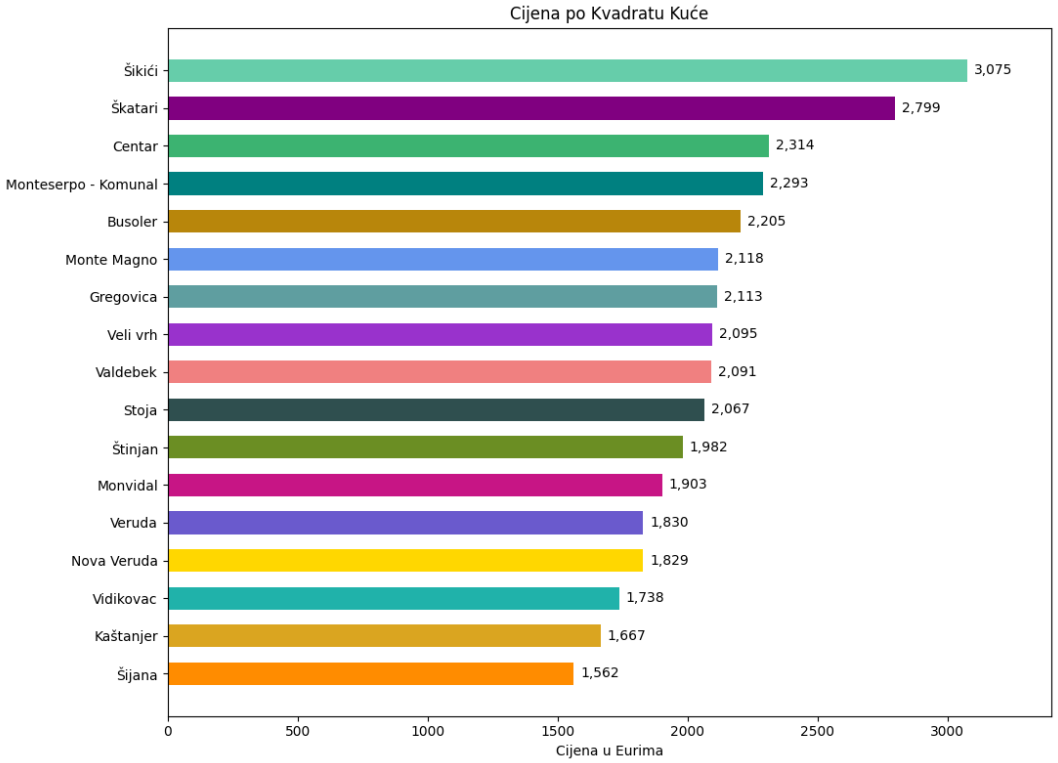
Slika 31. Grafički prikaz raspona broja kvadrata nekretnina

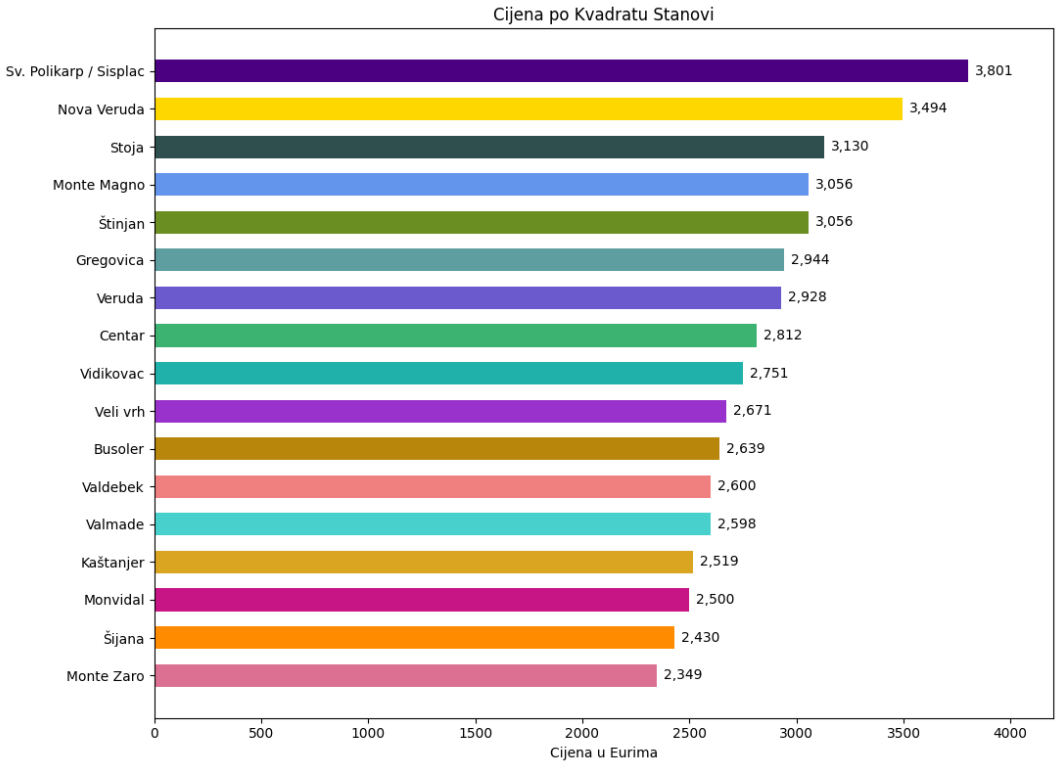




Izvor: https://www.njuskalo.hr/

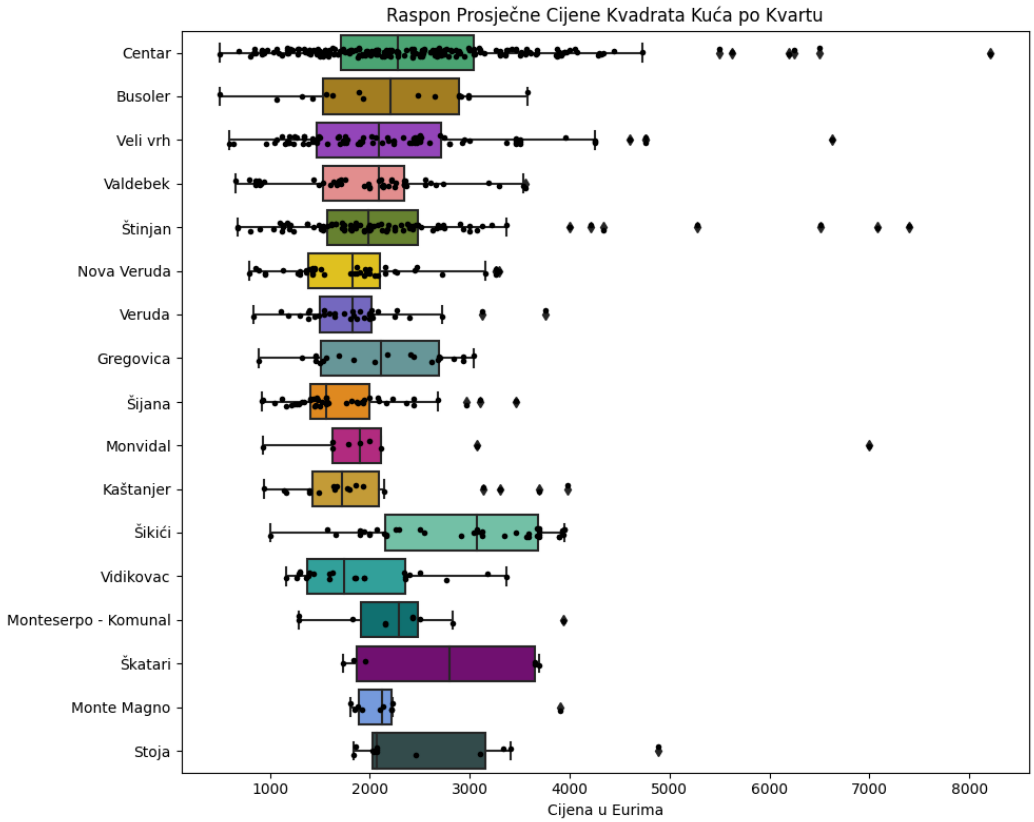
Slika 32. Grafički prikaz prosječne cijene kvadrata nekretnina

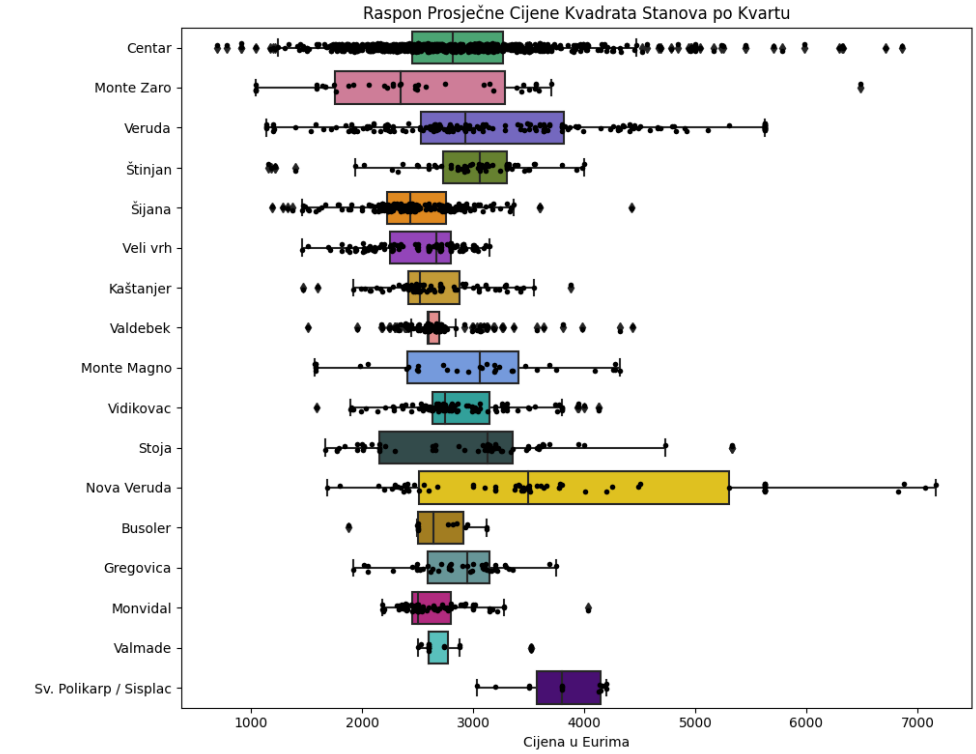




Izvor: https://www.njuskalo.hr/

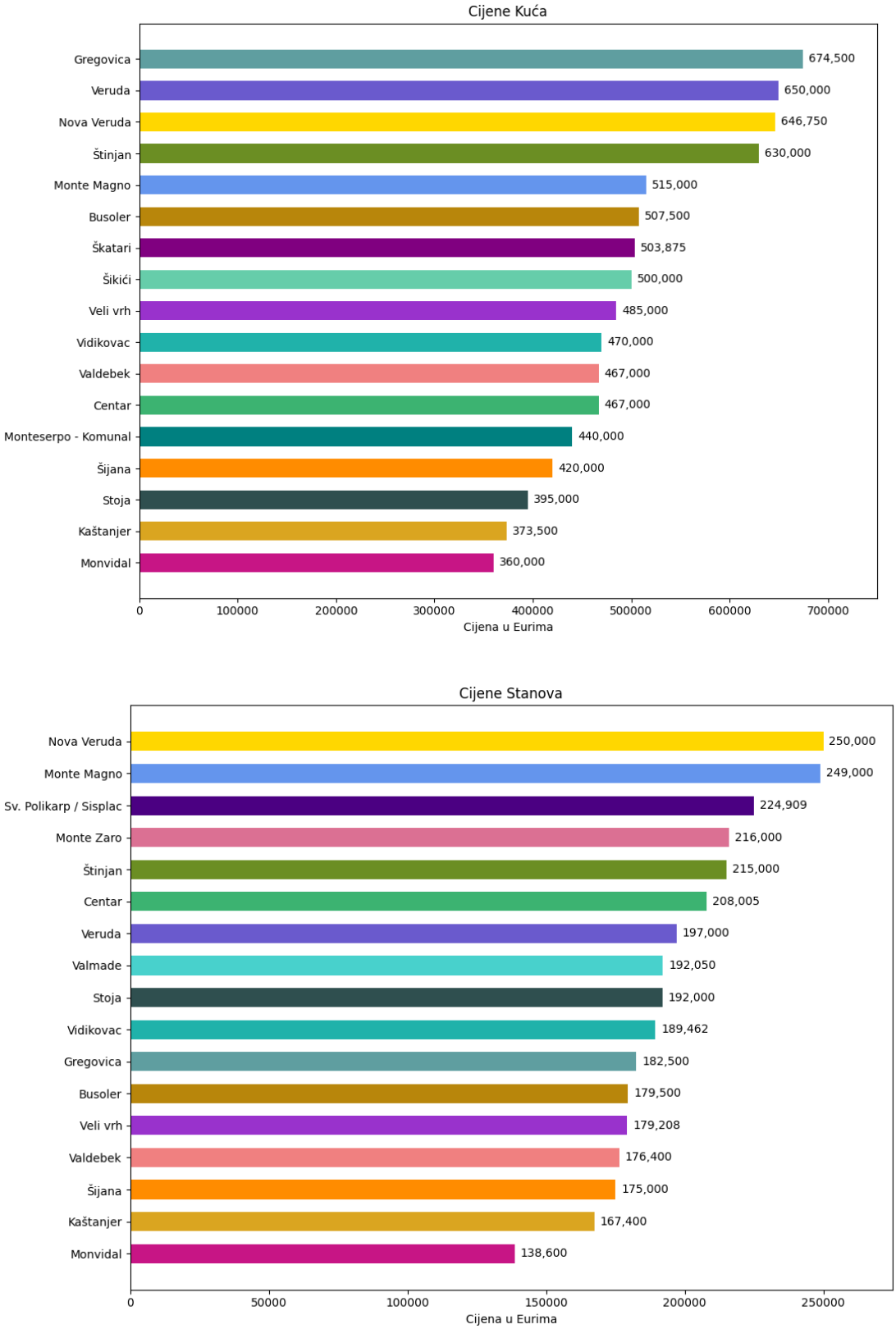
Slika 33. Grafički prikaz raspona prosječne cijene kvadrata nekretnina





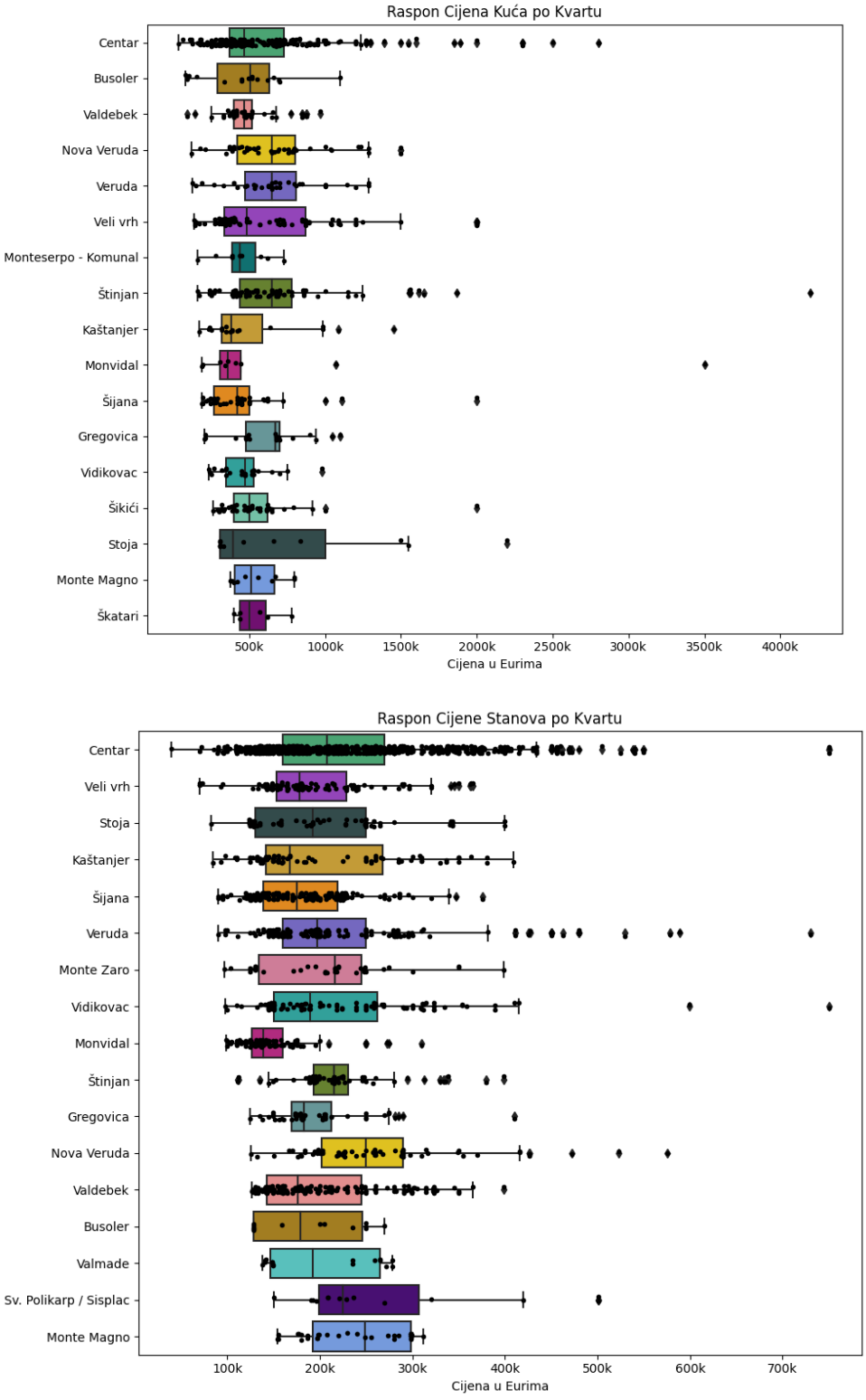
Izvor: https://www.njuskalo.hr/

Slika 34. Grafički prikaz prosječnih cijena nekretnina



Izvor: https://www.njuskalo.hr/

Slika 35. Grafički prikaz raspona cijena nekretnina



Izvor: https://www.njuskalo.hr/

Zaključak

Web scraping je izuzetno koristan za prikupljanje i obradu podataka ako je tražen specifičan skup podataka koji nije prethodno dostupan iz drugih izvora. Kroz ovaj rad smo demonstrirali kako se tehnikama web scraping-a može pristupiti velikom broju podataka dostupnih na Njuškalo.hr, jednoj od najpopularnijih platformi za posredovanje u prometu nekretninama u Hrvatskoj. Proces prikupljanja i obrade podataka koristeći Scrapy, Matplotlib i Seaborn pokazao je kako se web scraping može koristiti za uvid na tržište nekretnina na području grada Pule, te kako prikupljeni podaci mogu pružiti vrijedne uvide u dinamiku tržišta. Prošli smo kroz pitanja vezana uz pravne i etičke aspekte prikupljanja podataka s web stranica, došavši do zaključka da je nužno postupati sa znatnom pažnjom i poštovanjem u slučajevima veće osjetljivosti, uvijek uzimajući u obzir prava vlasnika podataka i njihove dozvole.

Primjenom Scrapy okvira, uspješno smo izvukli i obradili podatke s web stranice Njuskalo.hr. Proces je pokazao kako se web scraping može koristiti za automatizirano prikupljanje podataka na velikoj skali, što bi inače bilo vremenski zahtjevno i nepraktično za obaviti ručnim radom.

Analiza podataka korištenjem Matplotlib i Seaborn biblioteka omogućila je stvaranje intuitivnih vizualizacija podataka koje olakšavaju razumijevanje karakteristika tržišta nekretnina na području grada Pule. Slike i tablice dobivene kroz ovu analizu ukazale su na različite trendove i uzorke koji možda prethodno nisu bili vidljivi.

Ovim radom utvrdili smo da je web scraping ne samo tehnički izvediv, nego i vrlo koristan pristup za prikupljanje podataka s interneta. Potencijal web scraping-a seže mnogo dalje od istraženih aspekata u ovom radu, nudeći mogućnosti za primjenu u različitim sektorima digitalne ekonomije. Bez obzira na to radi li se o prikupljanju podataka za istraživačke svrhe, komercijalnu analizu ili osobnu upotrebu. Web scraping se pokazao kao robustan alat za obradu informacija u velikim količinama.

Moguće je izdvojiti nekoliko potencijalnih pravaca za daljnja istraživanja i analize koji bi mogli produbiti naše razumijevanje dinamike tržišta nekretnina temeljenih na podacima prikupljenim putem web scraping-a s platforme Njuškalo.hr. Prvo, kako bi se bolje razumjeli dugoročni trendovi i promjene na tržištu nekretnina, prikupljanje podataka bi se moglo ponoviti nekoliko puta u vremenskim razmacima od 3 mjeseca i proširiti na druge gradove. Ovo bi omogućilo praćenje dinamike promjena cijena i veličina nekretnina te identificiranje sezonskih ili dugoročnih obrazaca koji bi mogli biti korisni za potencijalne investitore, agente za nekretnine i druge sudionike na tržištu. Drugo, kako bi se utvrdila pouzdanost rezultata, prikupljanje podataka bi se moglo napraviti s više različitih izvora, kao što su platforme poput Indeks oglasa ili Facebook Marketplace. Usporedba i analiza podataka s više izvora omogućila bi dublje razumijevanje usklađenosti ili razlike u cijenama i veličinama nekretnina te identificiranje potencijalnih varijacija između različitih platformi. U kontekstu daljnjeg razvoja i unaprjeđenja procesa prikupljanja podataka, moguć je razvoj RPA (Robotic Process Automation) aplikacije koja bi mogla generalizirati i automatizirati pristup prikupljanja podataka s različitih platformi poput Njuškalo.hr, Indeks oglasi, Facebook Marketplace i drugih. Ova aplikacija bi značajno olakšala proces prikupljanja i analize podataka, istovremeno povećavajući pouzdanost i efikasnost istraživanja. RPA robot bi mogao biti programiran da automatski pretražuje, izvlači i obrađuje informacije s različitih web stranica, koristeći unaprijed definirane parametre kao što su cijene, veličine, lokacije i druge relevantne karakteristike nekretnina. Ovaj automatizirani pristup bi omogućio brže i konzistentnije prikupljanje podataka, minimizirajući ljudsku intervenciju i potencijalne pogreške. Nadalje, RPA robot bi također mogao biti konfiguriran za provođenje periodičnih prikupljanja podataka, što bi omogućilo praćenje tržišnih trendova i promjena tijekom vremena. Osim toga, RPA aplikacija bi mogla biti proširena kako bi automatski generirala izvještaje i analize na temelju prikupljenih podataka. To bi moglo uključivati vizualizacije podataka, trendove cijena, geografske rasporedne analize i druge relevantne informacije koje bi olakšale interpretaciju rezultata.

Literatura

Butler, J. (2007). '*Visual web page analytics*' Google Patents.

Damodaran B, D., Salim, S. and Vargese, S.M. (2016) '*Performance Evaluation of MySQL and MongoDB Databases'*, International Journal on Cybernetics & Informatics, 5(2), pp. 387–394. Dostupno na : https://doi.org/10.5121/ijci.2016.5241. 15(5), 788–797. doi:10.1093/bib/bbt026 PMID:23632294

Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2013). '*Web scraping technologies in an API world. Briefings in Bioinformatics'*, 15(5), 788–797. doi:10.1093/bib/bbt026 PMID:23632294

Kasereka, Henrys. (2020). '*Importance of web scraping in e-commerce and e-marketing.'*. SSRN Electronic Journal. 10.6084/m9.figshare.13611395.v1.

Kaur, K. and Behal, S. (2014) *'Captcha and Its Techniques: A Review'*, International Journal of Computer Science and Information Technologies, 5.

Laudon, K., & Traver, C. G. (2009). '*E-commerce'*. Pearson educación.

Marić, D. (2022) '*All you need to know to make your first web scraper*, *Megatrend* ' Dostupno na : https://www.megatrend.com/en/all-you-need-to-know-to-make-your-first-web-scraper/ (Pristupljeno : lipanj 2023).

Masse, M. (2011). '*REST API design rulebook: designing consistent RESTful web service interfaces'*. " O'Reilly Media, Inc.",.

Michael, Y. (2021) '*Scrapy Tutorial #8: Scrapy Selector Guide'* Dostupno na :

https://www.accordbox.com/blog/scrapy-tutorial-8-scrapy-selector-guide/ (Pristupljeno : svibanj 2023)

Mitchell, R. (2018).'*Web scraping with Python: Collecting more data from the modern web*.' " O'Reilly Media, Inc.".

Munappy, A.R., Bosch, J. and Olsson, H.H. (2020) *'Data Pipeline Management in Practice: Challenges and Opportunities*', in M. Morisio, M. Torchiano, and A. Jedlitschka (eds) *Product-Focused Software Process Improvement*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 168–184. Dostupno na : https://doi.org/10.1007/978-3-030-64148-1_11.

Murray State University *et al.* (2020) '*Legality and Ethics of Web Scraping, Communications of the Association for Information Systems'*, 47, pp. 539–563. Dostupno na : https://doi.org/10.17705/1CAIS.04724.

Sapsford, R. (2006) '*Data Collection and Analysis'*, Amazon

Scrapy (2023) '*Items'* Dostupno na : https://docs.scrapy.org/en/latest/topics/items.html (Pristupljeno : svibanj, 2023)

Scrapy (2023) '*Scrapy at a glance'* Dostupno na : https://docs.scrapy.org/en/latest/intro/overview.html (Pristupljeno : lipanj, 2023)

Tosi, S. (2009) '*Matplotlib for python developers: build remarkable publication quality plots the easy way*.' Birmingham: Packt Publ (From technologies to solutions).

Twitter (2023), '*Twitter API'* Dostupno na : https://developer.twitter.com/en/docs/twitter-api (Pristupljeno : svibanj 2023)

Popis slika

Popis tablica

Sažetak

Tema ovog diplomskog rada je primjena web scraping tehnika za prikupljanje i obradu podataka s platformi za posredovanje u prometu nekretninama s naglaskom na tržište nekretnina u gradu Pula. Obrađeni su problemi koji mogu nastati tijekom primjene web scraping tehnika te pravnih i etičkih dilema koje su proizašle iz prikupljanja podataka. Naglasili smo važnost poštivanja vlasničkih prava i dozvola prilikom korištenja ovih tehnika. Prikazan je proces automatiziranog prikupljanja i obrade podataka koristeći Scrapy okvir. Nakon prikupljana podataka, prikazana je analiza i grafički prikaz podataka koje daju uvid u tržišne trendove na području grada Pule.

Ključne riječi : Python, Scrapy, MySQL, matplotlib, analiza podataka, web scraping

Summary

The topic of this master's thesis is the application of web scraping techniques for data collection and processing from real estate brokerage platforms, with a focus on the real estate market in the city of Pula. The thesis addresses the problems that may arise during the application of web scraping techniques, as well as the legal and ethical dilemmas that have emerged from data collection. We emphasized the importance of respecting ownership rights and permissions when using these techniques. The thesis presents the process of automated data collection and processing using the Scrapy framework. After data collection, an analysis and graphical representation of the data were presented, providing insights into market trends in the area of the city of Pula.

Key words : Python, Scrapy , MySQL, matplotlib, data analysis, web scraping