Rīgas 64. Vidusskola

Rīgas apkaimju vidējā mēneša bruto darba samaksas noteikšana, izmantojot nekustamo īpašumu un vakanču sludinājumu tendences

Zinātniskās pētniecības darbs programmēšanā

Darba autors:
Teo Justs Holcmanis
12.INZ klases skolnieks
Darba vadītājs:
Edvards Bukovskis
Programēšanas skolotājs

Rīga 2024

Anotācija

Annotation

Saturs

Ievads.....4
1. Datu skrāpēšana.....5
1.1. Ieskats datu skrāpēšana.....5
1.2. Datu skrāpēšanas process.....5
1.3. Datu skrāpēšana izmantojot selenium.....5
1.4. Datu skrāpēšana izmantojot requests.....5
1.5. Datu skrāpēšanas nepieciešamība.....6
1.6. Datu skrāpēšana no juridiskā un ētiskā skatapunkta.....6
2. Datu vizualizācija.....7
2.1. Numpy.....7
2.2. Pandas.....7
2.3. Mathplot un Seaborn.....7
2.4. Sklearn.....7
3. Datu iegūšanas un apstrādes gaita.....8
3.1 Vakanču sludinājumu datu ievākšana un apstrāde no tīmekļalapas nva.gov.lv.....8
3.2 Vakanču sludinājumu datu ievākšana un apstrāde no tīmekļalapas ss.lv.....9
3.3 Nekustamo īpašumu sludinājumu datu ievākšana un apstrāde no tīmekļalapas ss.lv.....10
Izmantotie literatūras avoti.....11
Pielikums.....12

Ievads

Pilsētas apkaimju iedzīvotāju ienākumu sadalījums palīdz noteikt idzīvotāju labklājību. Nesen tika publicēta oficiāla informācija tīmekļvietnē stat.gov.lv par katras Rīgas apkaimes vidējo mēneša bruto darba samaksu. Ņemot vērā minēto, man rodas jautājums: vai šo informāciju var prognozēt? Šī pētījuma mērķis ir pārbaudīt, vai ir iespējams noteikt Rīgas apkaimju iedzīvotāju vidējo mēneša bruto darba samaksu, izmantojot publiski pieejamus nekustamo īpašumu un vakanču datus. Analīzē ir iekļauta datu ievākšana, apstrāde un salīdzināšana ar oficiālajiem statistikas datiem. Datu ievākšana, izmantojot kodētu programmu, ir ātrs un efektīvs veids, kā apkopot un atjaunot datus, jo tie var mainīties dažādu apstākļu dēļ. Pētījums var noteikt, cik precīzi šāda metode spēj atspoguļot patieso situāciju un vai šādu pieeju var izmantot, lai atjaunotu vai papildinātu esošos datus.

Pētījuma mērķis:
Noskaidrot, vai ir iespējams prognozēt Rīgas apkaimju iedzīvotāju vidējo mēneša bruto darba samaksu, izmantojot nekustamo īpašumu un vakanču sludinājumus.

Pētījuma uzdevumi:

1. Apkopot pieejamo informāciju par datu skrāpēšanu un datu apstrādi;
2. Izveidot programmu, kas ievāc šobrīd pieejamos datus no sludinājuma vietnēm ss.lv un nva.gov.lv;
3. Apkopot datus;
4. Veikt analīzi;
5. Izdarīt secinājumus.

Pētījumā izmantotās metodes:

1. Literatūras analīze - apkopot informāciju par datu vākšanas, apstrādes un analīzes metodēm;
2. Koda izstrāde - tīmekļvietnes skrāpēšana. Izveidotais kods izmantos *requests, selenium* un *lxml* bibliotēkas, kas ļauj automatizēti iegūt datus no portāliem ss.lv un nva.gov.lv. Tas nodrošinās strukturētu datu ievākšanu no vairākām lapām, ļaujot apkopot plašu informācijas apjomu;
3. Datu eksportēšana - saglabāt visus apkopotos datus CSV vai JSON, vai Excel formātā, lai dati būtu pārskatāmāki un atvieglotu tālāku apstrādi citās programmās;
4. Datu analīze - datu vizualizācijas rīki. Izmantojot *matplotlib*, *seaborn* vai *Plotly*, veido histogrammas, kastes-diagrammas un kartes, lai ilustrētu datu sadalījumu;
5. Rezultātu salīdzināšana: salīdzināt iegūtos prognozētās algas līmeņus ar oficiālajiem datiem, novērtējot metodes precizitāti.

1. Datu skrāpēšana

1.1. Ieskats datu skrāpēšana

Datu skrāpēšana jeb automatizēta informācijas vākšana no interneta vietnēm dod lietotājam iespēju, izmantojot vienkāršu, automatizētu programmu, piekļūt interneta serverim, pieprasīt datus un apstrādāt tos, lai iegūtu nepieciešamo informāciju. Šis paņēmiens ir būtisks mūsdienu digitālajā laikmetā, kad tiešsaistē ir pieejams milzīgs datu apjoms, bet ne vienmēr viegli pieejamā vai analizējamā formātā.[1]

1.2. Datu skrāpēšanas process

Pirmais solis datu skrāpēšana ir lejuplādēt interneta pārlūka lapas, kas satur datus, ko lietotājs vēlas apstrādāt. Šis ir iespējams ar dažādām programmatūras valodām un rīkiem, kā, piemēram, *Python* vai specializēta datu skrāpēšanas programmatūra.[2]

Nākamais solis ir analizēt *HTML* saturu, lai atrastu specifiskos datu elementus, kurus ir nepieciešams iegūt. Vispopulārāka metode ir dokumentu objektu modeļa (DOM) koka izveide, kā mērķis ir attēlot *HTML* struktūru.[3] Taču ir dažas ērtākas un efektīvākas metodes, kā, piemēram, *UzunExt*, kas atvieglo processu, ļaujot apiet (DOM) koka izveidi, izmantojot virkņu metodes.[3] F

Pēc *HTML* analīzes tiek iegūti konkrēti datu elementi izmantojot metodes, kā, piemēram, sakarību meklēšana, *HTML* tagu skaitīšana, kas apzīmē dokumenta daļas, piemēram, virsrakstus, attēlus, paragrāfus un citus elementus.[3] Turpmākai analīzei iegūtie dati tiek attīrīti un pārveidoti strukturētā un pārlasāmā formātā, piemēram, izklājlapā, tabulā vai datu bāzē.[1]

Pēdējais solis ir iegūto datu strukturēta saglabāšana. Šis ir ērti panākams izmantojot JSON failus. JSON fails ir viegls, cilvēkiem viegli pārlasāms un strukturēts datu formāts, ko bieži izmanto datu pārnešanai starp programmām, kā arī JSON failus var lasīt un uzrākstīt vairākās programmēšanas valodās. Strukturētos datus pēc tam var dažādi pielietot, piemēram, statistikas analīzei, mašīnmācīšanai un vizualizēšanai.[1]

1.3. Datu skrāpēšana izmantojot selenium

1.4. Datu skrāpēšana izmantojot requests

1.5. Datu skrāpēšanas nepieciešamība

Atklājumi mūsdienu datorikā dod iespēju lietotājam automatizēti vākt un analizēt lielus datu apjomus, kas agrāk nebija iespējams. Datu skrāpēšana tikai dažu stundu laikā ir spējīga ievākt apjomīgas datu kopas ar desmitiem tūkstošu mainīgo vai pat vairāk. Šo metodi var izmantot zinātniskām izpētes problēmam, ļaujot pārbaudīt izpētes jautājumus, kuriem nepieciešamas lielas datu kopas. Šādu datu izvēle notiek pēc konkrētām hipotēzēm.[4] Bez programmatūras palīdzības šis process aizņemtu dienas, ja ne nedēļas, taču ar datu skrāpēšanas palīdzību lietotājs ir spējīgs ietaupīt laiku un pūles.

Datu skrāpēšana ir itīpaši noderīgi tirgus izpētei, cenu salīdzināšanai un patērētāju tendenču analīzei. Lietotāji izmanto iegūtos datus, lai pieņemtu stratēģiskus lēmumus, piemēram, produktu cenu noteikšanu, izstrādi un pozicionēšanu tirgū. Datu skrāpēšana apkopo reāla laika datus, kas ir ļoti noderīgi uzņēmumiem, kuriem ir nepieciešams ātri reaģēt uz tirgus izmaiņām, kā arī tas ir noderīgi konkurentu darbību uzraudzībai un stratēģiju pielāgošanai.[5]

Mašīnmācīšanās ir mākslīgā intelekta atzars, kas mācās no datiem un pieņem lēmumus ar minimālu cilvēka iejaukšanos. Algoritms mācās no liela apjoma datiem - novēro, secina sakarības un veic prognozes vai lēmumus patstāvīgi. Šis process ļauj veikt tādus uzdevumus kā attēlu atpazīšana, runas analīze un klasifikācija, kas tradicionāli prasīja cilvēka intelektu. Mašīnmācīšanās algoritmiem nepieciešams liels datu apjoms, lai veiktu precīzas prognozes, dažkārt arī nepieciešami vairāku apvienotu avotu dati, kas ļauj veikt vēl plašāku analīzi.[6] Datu skrāpēšana risina šo problēmu, automatizējot datu iegūšanas processu un nodrošinot nepārtrauktu, aktuālu un efektīvu datu plūsmu.

1.6. Datu skrāpēšana no juridiskā un ētiskā skatapunkta

Datu skrāpēšana ir noderīgs un svarīgs rīks pētniekiem un uzņēmējiem. Tomēr tas var radīt būtiskas juridiskas un ētiskas problēmas. Ir svarīgi ievērot tīmekļalapu un datu avotu privātuma politiku un lietošanas noteikumus, lai izvairītos no juridiskām sekām un tālākām problēmām.

Datu skrāpēšana Latvijā ir atļauta, ja tiek ievēroti juridiskie un ētiskie nosacījumi. Tā ir pieļaujama tikai tad, ja dati tiek iegūti no publiski pieejamiem avotiem, nepārkāpjot privātuma politikas, autortiesības vai tīmekļalapas lietošanas noteikumus. Regulā (GDPR) tiek uzsvērts, ka datu apstrādei nepieciešams skaidrs juridiskais pamats un jānodrošina personas datu aizsardzība. Lai gan autortiesības ne vienmēr aizsargā datus, pastāv citi juridiskie likumi, piemēram, komercnoslēpumu aizsardzība un datu neatļautas iegūšanas likums, piemēram, *Computer Fraud and Abuse Act* (CFAA), kas var tikt izmantoti, lai limitētu negodprātīgu datu ievākšanu un izmantošanu. [7]

1

Pētniekiem un uzņēmējiem būtu jāiegūst skaidra piekrišana par datu ievākšanu, tādējādi saglabājot uzticību un uzturot ētikas standartus. Ir būtiski, ka datu skrāpēšana nerada kaitējumu un neapdraud datu avotu darbību, integritāti vai reputāciju, kā arī jāizvairās no pārmērīgas slodzes uz serveriem.[8]

2. Datu vizualizācija

2.1. Numpy

2.2. Pandas

2.3. Mathplot un Seaborn

2.4. Sklearn

3. Datu iegūšanas un apstrādes gaita

Veidots izmantojot draw.io

3.1 Vakanču sludinājumu datu ievākšana un apstrāde no tīmekļalapas nva.gov.lv

Latvijā ir daudzas un dažādas tīmekļalapas, kur tiek publicēti darba vakanču sludinājumi. Valsts piederošajai nva.gov.lv tīmekļalapā ir pieejamas darba vakances, ko nodrošina valsts.

Šai tīmekļa lapai ir unikāls lapas izklājums - ar katru peles kustību tīmekļalapa izveido jaunu tīmekļalapas adresi un izvēlētie filtri netiek saglabāti tīmekļalapas adreses saturā. Šis padara tīmekļalapas datu ievākšanu sarežģitu un kompleksu, tādēļ tiek pielietots *Selenium* metode datu ievākšanas procesā.
Procesu atvieglo tas, ka nepieciešamā informācija ir pieejama vienā tīmekļalapā, tas nozīmē, ka nav nepieciešamas papildus darbības, lai piekļūtu specifiskajiem datiem.

No tīmekļalapas tiek iegūti visas Latvijas darba vakanču attalgojumi, jeb bruto alga, un attiecīgā vakanču adrese. Programma izfiltrē adreses un saglabā tikai vakances, kas atrodās Rīgā. Dati tiek saglabāti JSON failā. Tiek izveidota papildus programma, kas spēj noteikt Rīgas apkaimi no piedāvātās adreses. Programma ņem vērā arī ielas, kas plešas garūma caur vairākām Rīgas apkaimēm, tādēļ specifiski izvēlētām ielām arī tiek ievākts ielas numurs. Tālāk katrai specifiskajai ielai tiek piešķirti intervāli, kas nosaka Rīgas apkaimi, lai precīzi noteiktu adreses atrašanos vietu. Pēc tam bruto algas tiek sadalītas attiecīgajās apkaimēs.

3.2 Vakanču sludinājumu datu ievākšana un apstrāde no tīmekļalapas ss.lv

Latvijas populārākā sludinājuma tīmekļalapa ir ss.lv. Tīmekļalapas uzbūve ir īpatnēja, jo pēc likuma darba devējam ir jānorāda attalgojums, taču tīmekļalapā nav specifiska lauciņa, kur, veidojot sludinājumu, būtu jāievada attalgojums. Tādēļ daudzos sludinājumos pat nav norādīts attalgojums, vai arī tas tiek rakstīts neregulāros formātos.

Tiek izveidota programma, kas nolasa vakanču ievāktos aprakstus un atrod vārdu kombinācijas, kas varētu apzīmēt jebkāda veida attalgojumu. Atrastie dati par attalgojumu, kā arī vakanču adreses tiek ievāktas. Procesu apgrūtina tas, ka, veicot izmaiņas filtrā, tīmekļalapas adreses saturs netiek atjaunots, tāpēc ir nepieciešams izmantot *Selenium*, lai veiktu filtru izvēli. Nonākot filtrētā darbu vakanču tīmekļalapā, katra darba vakanču sludinājuma nepieciešama informācija atrodas vēl viena datora peles klikšķa attālumā, un programma sistemātiski atlasa katru sludinājumu un izvelk nepieciešamo informāciju.

Tīmekļalapā ss.lv filtros ir iespēja atzīmēt Rīgas apkaimes, taču Centra apkaimē tiek iekļautas gan Avotu, gan Brasas, gan Andrejsalas, gan Skanstes apkaimes, tādēļ ir nepieciešams atlasītos datus no Centra apkaimes sadalīt piecās āttiecīgajās apkaimās. Tiek pielietota programma, kas pēc adreses spēj noteikt Rīgas apkaimi. Pārējās Rīgas apkaimes ir attiecīgi pieejams filtros, tādēļ tām netiek pielietota Rīgas apkaimes noteikšanas programma. Bruto darba samakas datus attiecīgi apkopo pa Rīgas apkaimēm un saglabā JSON faila formātā.

3.3 Nekustamo īpašumu sludinājumu datu ievākšana un apstrāde no tīmekļalapas ss.lv

Visplašāk un visērtākā izmantojamā nekustamo īpašumu sludinājumu tīmekļalapa ir ss.lv. No tīmekļalapas ir ērti un effektīvi ievākt datus, jo tās formāts ir statisks un visa nepieciešamā informācija ir pieejama pirmajā tīmekļalapas lapā. Pēc katras lapas pāršķiršanas kustības tīmekļalapas adreses saturs tiek atjaunots, tāpēc viseffektīvākais veids, kā ievākt datus ir ar requests paņēmienu. Salīdzinot ar Selenium paņēmienu, tas patērē mazāk resursus un aizņem mazāk laiku, jo nav nepieciešams lādēt pilnu tīmekļalapas pārlūka saturu.

Nekustamo īpašumu sludinājuma tīmekļalapa ss.lv ir unikāla ar to, ka tai ir pieejami arhivētie dati, tādejādi apjomīgi papildinot ievākto datu daudzumu. Izveidotā programma ievāc katra aktīvā un arhīvēto sludinājuma īpašuma platību, kopējo cenu un arī aprēķina cenu uz kvadrātmetru. Tīmekļalapa ss.lv apvieno Centra, Avotu, Andrejsalas, Brasas un Skanstes apkaimes vienā Rīgas apkaimē - Centrs, tāpēc ir nepieciešams, izmantojot sludinājumā minētās īpašuma adreses, sadalīt Centra apkaimes datus attiecīgi piecās apkaimēs. Programma apkopo un saglabā JSON failā iegūtos un aprēķinātos datus, un kategorizē Rīgas apkaimēs.

Izmantotie literatūras avoti

1. Kulkarni, S. (2023). Web Scraping: Extracting Insights from the Digital Landscape. *International Journal for Research in Applied Science and Engineering Technology*. https://doi.org/10.22214/ijraset.2023.53467.
2. Bradley, A., & James, R. (2019). Web Scraping Using R. *Advances in Methods and Practices in Psychological Science*, 2, 264 - 270. https://doi.org/10.1177/2515245919859535.
3. Uzun, E. (2020). A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages. *IEEE Access*, 8, 61726-61740. https://doi.org/10.1109/ACCESS.2020.2984503.
4. Landers, R., Brusso, R., Cavanaugh, K., & Collmus, A. (2016). A primer on theory-driven web scraping: Automatic extraction of big data from the Internet for use in psychological research.. *Psychological methods*, 21 4, 475-492 . https://doi.org/10.1037/MET0000081.
5. Henrys, K. (2021). Importance of Web Scraping in E-Commerce and E-Marketing. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.3769593.
6. Sirisuriya, S. (2023). Importance of Web Scraping as a Data Source for Machine Learning Algorithms - Review. *2023 IEEE 17th International Conference on Industrial and Information Systems (ICIIS)*, 134-139. https://doi.org/10.1109/ICIIS58898.2023.10253502.
7. Mancosu, M., & Vegetti, F. (2020). What You Can Scrape and What Is Right to Scrape: A Proposal for a Tool to Collect Public Facebook Data. https://doi.org/10.1177/2056305120940703.
8. Krotov, V., Johnson, L., & Silva, L. (2020). Tutorial: Legality and Ethics of Web Scraping. https://doi.org/10.17705/1cais.04724.
9.

Pielikums

https://github.com/teojusts/ZPD_Pielikums