Univerzitet u Beogradu

Elektrotehnički fakultet



Pronalaženje skrivenog zanja

Projektni zadatak

|  |  |
| --- | --- |
| **Kandidat:** | **Školska godina:** |
| Nenad Babin 2020/3019 | 2020/2021. |

Beograd, jul 2021.

Sadržaj

[Sadržaj 2](#_Toc76406675)

[1. Zadatak 1: Prikupljanje podataka 3](#_Toc76406676)

[2. Zadatak 2: Analiza podataka 4](#_Toc76406677)

[3. Zadatak 3: Vizuelizacija podataka 5](#_Toc76406678)

[4. Zadatak 4: Implementacija regresije 6](#_Toc76406679)

[5. Zadatak 5: Implementacija klasifikacije 7](#_Toc76406680)

1. Zadatak 1: Prikupljanje podataka

U okviru prvog zadatka su realizovani veb indekser i veb parser koji je prikupio podatke sa sajta *halooglasi.com/nekretnike*. Veb indekser i veb parser su realizovani pomoću Python biblioteke Scrapy.

Za čuvanje podataka je korišćena MySQL baza podataka. Pomoću alata MySQL workbench je kreirana baza. Isti alat je takođe korišćen za pisanje i proveru upita koji su dalje korišćeni u programskom kodu. Kolone koje se koriste u bazi su: id, tip\_ponude (prodaja ili izadavanje), tip\_nekretnine (kuća ili stan), broj\_soba, spratnost, sprat, površina\_placa, grejanje, grad, lokacija, mikrolokacija, kvadratura, parking (da ili ne), uknjiženost (da ili ne), terasa (da ili ne), lift (da ili ne), tip\_objekta (stara gradnja ili novogradnja), cena, x\_pos, y\_pos. x\_pos, y\_pos su koordinate na kojima se objekat nalazi. Prikupljeno je ukupno 36615 zapisa sa sajta.

Realizacija veb indeksera i veb parsera je data u okviru foldera *Scrapy*.

1. Zadatak 2: Analiza podataka

U okviru drugog zadatka je realizovana skripta koja čita podatke iz realizovane baze podataka iz zadataka 1, a zatim generiše *csv* fajlove u kojima se nalaze rezultati upita. *csv* fajlovi se zatim konvertuju u *xlsx* fajlove zbog čitljvosti.

U folderu *data\_interpreter* se nalazi skripta, upiti i generisani *csv* fajlovi. Sami upiti iz sql fajla su poslužili kao pomoć i provera upita pre nego što su preneti u skriptu.

1. Zadatak 3: Vizuelizacija podataka

U okviru trećeg zadataka je realizovana skripta koja radi vizuelizaciju. Skripta koristi Python biblioteke *pandas*, *numpy* i *matplotlib*.

Biblioteke *pandas* i *numpy* su korišćene nad podacima koji se iščitaju iz baze podataka. Odlučeno je da se jednom pročitaju svi podaci iz baze podataka, a da se nadalje rade transformacije nad tim podacima pomoću biblioteke *pandas*. Sama biblioteka *pandas* i njena klasa *DataFrame* omogućavaju pisanje svojevrsnih upita sličnim onim u *SQL* jeziku.

Realizacije su priložene u folderu *visualizer*. U podfolderu *pictures* se nalaze generisani grafikoni.

1. Zadatak 4: Implementacija regresije

U okviru četvrtog zadatka je realizovana višestruka linearna regresija. Realizacija se nalazi u okvriru klase *LinearRegressionPSZ*.

Glavna skripta na početku iscrtava raspodele ulaznih odlika. Zatim se radi skaliranje vrednost odlika (kvadratura, broj soba, spratnost, udaljenost od centra, tip objekta) po uzoru na klasu *StandardScaler[[1]](#footnote-1)* iz biblioteke *scikit-learn* tako da srednja vrednost vrednosti određene odlike bude nula. Nakon skaliranja se radi podela podataka na skup za obučavanje i skup za treniranje. 70% podataka je korišćeno za obučavanje, a 20% za testiranje. Zatim se generisani skup za obučavanje prosleđuje glavnoj metodi koja radi treniranje.

Za treniranje se koristi stohastički gradijentni spust zbog njegove brzine u odnosu na grupni gradijentni spust. Interfejs klase nudi da korisnik odabere brzinu obučavanja i broj iteracija. Obučavanje se zaustavlja kada algoritam konvergira ili kada se izvrši zadati broj iteracija. Na kraju se iscrtava grafik koji prikazuje kako se smanjivala funkcija greške kroz iteracije.

Treba napomenuti da je izvršeno čišćenje podatak u cilju otklanjanja outlier-a. Kako bi se uočili outlier-i, korišćen je sajt *tableau.com* uz pomoć kog su vizuelizovani podaci i uočeni outlier-i. Takođe je korišćena *matplotlib* biblioteka za vizuelizaciju, ali je *tableau* bilo bolje rešenje. Rezultati su se poboljšali nakon otklanjanja podataka koji odskaču od ostatka. Skaliranje odlika je pomoglo da funkcija greške nakon svake iteracije opadne. Pre nego što su skalirane odlike, funckija greške nije konstantno opadala, već je dolazilo i do porasta u nekim iteracijama, a zatim do ponovnog opadanja u narednim iteracijama.

U okviru oplikacije korisnik može da istrenira model zadavajući proizvoljan broj iteracija, brzinu obučavanja i procenat podataka koji će se koristiti u skupu za testiranje. Korisnik takođe bira koje odlike će model koristiti - kvadratura, broj soba, spratnost, udaljenost od centra i tip objekta klase. Rezulat pokretanja je funkcija greške po iteracijama i koren srednje kvadratne greške na skupovima za testiranje i obučavanje. U okviru aplikacije, korisnik sam nakon treniranja modela može da za vrednosti svih odlika unese željene vrednosti i da mu aplikacija ispiše cenu stana.

Implementacija se nalazi u folderu *linear\_regression*.

1. Zadatak 5: Implementacija klasifikacije

U okviru četvrtog zadataka je realizovan algoritam K najbližih suseda. Realizacije se nalazi u okviru klase *KNN*.

Skripta učitava podatke i dodaje dodatnu kolonu koja se formira na osnovu opsega u koji upada cena nekretnine. Nova kolona predstavlja klasu podatka (0 - 4) koja odgovara opsegu iz zadataka (0 – cena nekretnike manja od 49 999, 4 – cena nekretnine veća od 200 000). Ta kolona se koristi za klasifikaciju. Skripta zatim radi stratifikovanu podelu na skup za obučavanje i testiranje. Zatim se radi inicijalizacija klase koja implementira algoritam. Interfejs klase omogućava da korisnik prosledi vrednost K. Ukoliko to ne korisnik ne uradi, implementacija sama računa vrednost K. Nakon inicijalizacije algoritma, algoritam se pokreće za sve podatke iz skupa za testiranje. Tokom faze testiranja se popunjava i matrica zabune iz koje se kasnije računa preciznost modela (accuracy).

U okviru aplikacije korisnik može da učita podatke u zavisnosti od unetog parametra za procenat podatak u skupu za testiranje, pri čemu mu aplikacija ispisuje vrednost parametra K. Pre pokretanja algoritma korisnik može da promeni vrednost ovog parametra. Korisnik takođe odabira da li će koristiti Euklidsku ili Menhetn razdaljinu. Nakon pokretanja algoritma, algoritam se pokreće za sve podatke iz skupa za testiranje, ispisuju se prediktovane i stvarne klase, i na kraju preciznost modela (accuracy). U okviru aplikacije, korisnik sam nakon treniranja modela može da za vrednosti svih odlika unese željene vrednosti i da mu aplikacija ispiše klasu kojoj stan pripada.

Implementacija se nalazi u folderu *k\_nearest\_neighbors*.

1. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [↑](#footnote-ref-1)