SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

VARAŽDIN

Sandra Sačarić

Razvoj deskriptivnog i prediktivnog modela

Zamjenski zadatak kolokvija iz kolegija Uvod u umjetnu inteligenciju

Mentori: Izv. prof. dr. sc. Dijana Oreški

Prof. dr. sc. Markus Schatten

Doc. dr. sc. Bogdan Okreša Đurić

Dunja Višnjić, mag. oec.

Varaždin, 2025.

Izjavljujem da je rješenje zadatka izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Sadržaj

Table of Contents

Opis implementacije

1. Uvoz i pripremanje podataka

Prije svega, potrebno je uvesti podatke i pripremiti iste kako bi se stvorio deskriptivni model. Za uvoz se koristila metoda iz pandas biblioteke, a kasnije se analizirao set podataka korištenjem klasičnih metoda poput *info* kako bi se vidjele vrste podataka, odnosno količinu, *describe* da se prikažu kvartili i središnje vrijednosti, i *head* za prikaz prvih nekoliko redaka.

Obavila se provjera mogućih dupliciranih vrijednosti, tj. redaka preko metode *duplicated*, koja uz pomoć *sum* metode prebrojava koliko se puta pojavljuju duplicirani retci, odnosno instance objekta. Provjerom je ustanovljeno da dupliciranih redaka nema.

Nakon toga su istaknuti i opisani primjeri kategorijskih podataka preko najučestalijih vrijednosti i slično. Kako bi se set podataka pripremio za razvoj deskriptivnog modela, za početak je potrebno eliminirati kategorijske varijable. Atributi i njihove vrijednosti podijeljeni su u numeričku, odnosno kategorijsku skupinu. Naravno, numeričke podatke je potrebno normalizirati jer u ovom slučaju postoji razlika u rasponu vrijednosti, a to se postiže putem inicijalizacije objekta *StandardScaler*, nakon čega se poziva metoda *fit_transform* koja zatim skalira vrijednosti.

Zatim se prijelazi na kategorijske varijable. Njih i njihove vrijednosti je potrebno pretvoriti u numeričke, a jedan od načina za postizanje toga je korištenje metode *get_dummies*; metoda koja pretvara kategorijske podatke u numeričke vrijednosti.

Jednom kada se postigne homogeniji set podataka, ta dva seta preko metode *concat* kombinirat će se u jedan set. S obzirom da se skup podataka sastoji od jako puno vrijednosti, za lakšu analizu i ubrzanje procesa, smanjit će se dobiveni skup na manji uzorak tako da se ostavi 5% nasumično odabranih vrijednosti.

Sa time, navedeni su svi koraci kroz koje se prošlo kako bi se podaci adekvatno pripremili. U daljnjem koraku će se nad istim podacima primjeniti K-means algoritam kako bi se odredio optimalan broj klastera, odnosno kako bi se podaci grupirali prema sličnosti, što će pomoći kod daljnjeg stvaranja modela za predviđanje.

2. Primjena K-means algoritma

Sljedeći primjere laboratorijskih vježbi sa kolegija, primjenjena je metoda lakta za vizualizaciju K vrijednosti, gdje optimalna K vrijednost predstavlja optimalan broj klastera.

Prije svega, inicijaliziran je prazak skup u koji bi se kasnije spremale vrijednosti inercije, odnosno mjeru uspješnosti K-means algoritma kod određivanja optimalnog broja klastera, za svaki pojedinačni klaster kroz koji se iterira.

Određen je raspon K vrijednosti koje se žele testirati, u ovom slučaju od 1 do 10 (u Pythonu postavljeno kao 1-11). Zatim je napravljena petlja koja će za svaku instancu (inkrementalna vrijednost broja klastera u definiranom rasponu) K stvoriti model kojem su postavljeni parametri *n_clusters*, koji predstavljaju broj klastera koji se provjerava, i *random_state* koji predstavlja *seed* po kojem algoritam osigurava da se za svako pokretanje odabrane iste vrijednosti za uzorkovanje. Nadalje, preko *fit* metode model uči pozicije centroida, koordinate točaka, inerciju, i tako dalje. To je sve zatim prikazano u formi dijagrama koji prikazuje inerciju na y osi i K vrijednosti na x osi.

Dakle, korištenjem K-means algoritma započelo se s analizom podataka kako bi se odredio optimalan broj klastera koji najbolje grupira podatke prema njihovim sličnostima. Za odabir broja klastera definirane su četiri opcije: 3, 4, 5 i 6 (u ovom slučaju najbolji kandidati ako se koristi metoda lakta), a kako bi se procijenila kvaliteta klasteriranja, koristi se Silhouette Score, metrika koja mjeri povezanost ili „blizinu“ instanci unutar klastera, odnosno udaljenost između svakog pojedinačnog klastera. Za svaku od predviđenih vrijednosti broja klastera inicijaliziran je KMeans model, gdje je *random_*state opet postavljen na 42 radi reproducibilnosti podataka. Pripadnosti klasterima su zatim dobivene metodom *fit_predict*, nakon čega je izračunat *silhouette_score* na temelju prethodno prilagođenog seta podataka *sampled_*df. Same dobivene vrijednosti su pohranjene radi usporedbe.

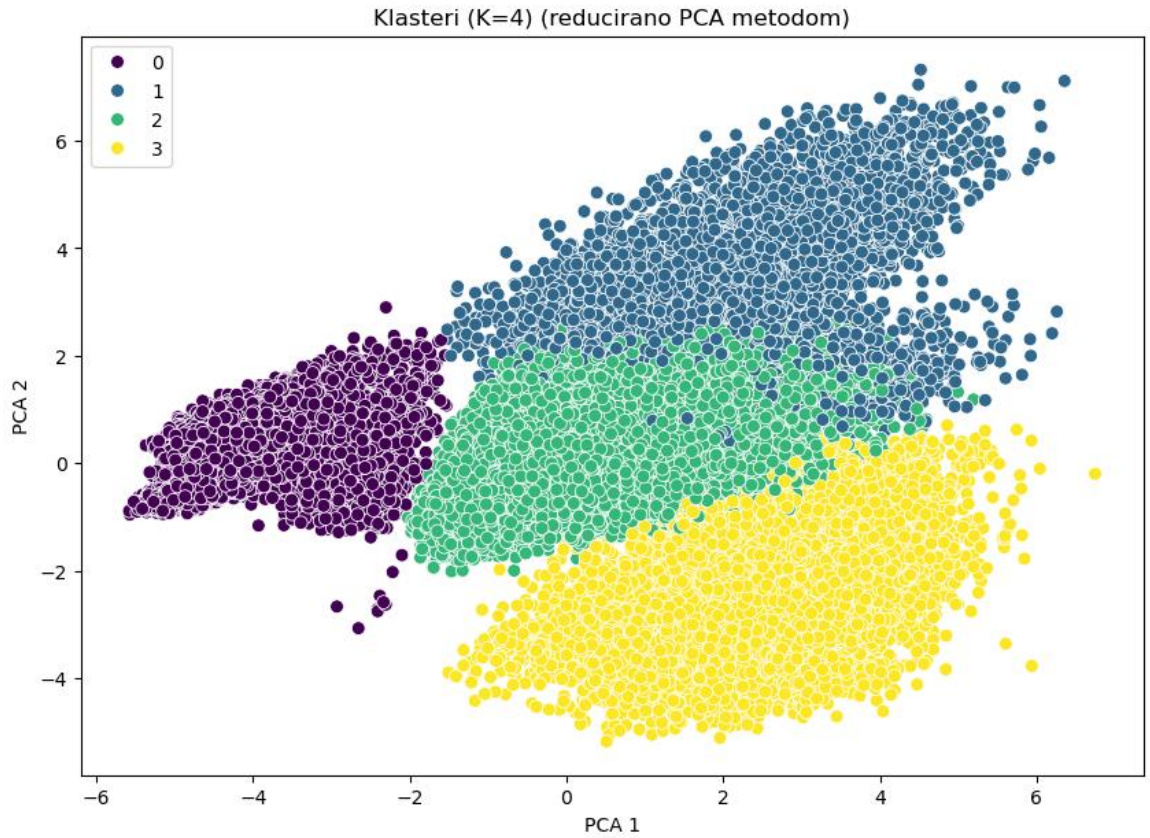
Nakon što su rezultati za sve opcije izračunati, identificiran je optimalan broj klastera kao onaj s najvećim rezultatom. Optimalni broj klastera pokazao se kao K = *best_k* s rezultatom *best_score*.

Ovaj korak omogućuje bolje grupiranje podataka u daljnjem radu i pruža temelje za kvalitetniji deskriptivni model. Radi lakše vizualizacije, kasnije su korišteni stupčasti dijagrami kako bi se prikazala distribucija podataka po klasterima.

Kod implementacije K-means algoritma i izračuna Silhouette vrijednosti koristile su se laboratorijske vježbe s kolegija.

3. PCA metoda za smanjenje dimenzionalnosti

S obzirom na veliku količinu značajki, odnosno nemogućnost prikazivanje klastera putem *scatterplota,* ukazala se potreba za korištenjem PCA metode. To je metoda koja omogućava da se višedimenzionalan skup podataka reducira na manje dimenzija, u ovom slučaju na dvije.



Slika 1 Scatterplot, samostalna izrada

Za vrijeme rješavanja zadatka se set podataka prilagođavao više puta i ovisno o tome bilo je varijacija na optimalan broj klastera. Prilikom inicijalnog postavljanja zadatka, intuicija je navodila da bi optimalan broj klastera mogao biti 4, no kasnijom analizom se uspostavilo drugčije. Najčešći optimalan broj klastera koji bi se dobio kao izlaz je 3 klastera, zatim 5, a najrjeđe 4, dok ostali brojevi nisu ni jednom bili označeni za optimalan broj. Specifično, kada je skup podataka reduciran na 5%, optimalan broj klastera je 4. Ako se uzme veći uzorak, na primjer 10%, onda je optimalan broj 3. To je utjecalo na odluku da se podaci ipak svedu na 5% od ukupnih radi prividno bolje podjele podataka.

Kao pomoć kod korištenja PCA metode koristili se se izvori s interneta (365DataScience, StackExchange, i KDnuggets).

4. Treniranje modela

Time je utvrđen optimalan broj klastera i razvijen deskriptivni model. Sada će se prijeći na razvoj prediktivnog modela koji će koristiti normalizirani set podataka.

Opet, zbog višedimenzionalnosti ovog skupa podataka, bilo je potrebno uvesti dodatne korake koji osiguravaju kvalitetno treniranje i testiranje modela. Prvo je određena ciljana varijabla, u ovom slučaju *Energy_Use_Intensity*, koja će se koristiti za predviđanje. Nadalje, podaci su podijeljeni prema pripadnosti klasterima, pri čemu je za svaki klaster formiran podskup podataka kako bi se omogućila prilagodba modela specifičnostima pojedinog klastera.

Kroz iteraciju nad jedinstvenim vrijednostima klastera, podaci su filtrirani tako da uključuju samo redove koji pripadaju trenutnom klasteru. Unutar svakog podskupa podaci su podijeljeni na značajke i ciljnu varijablu. Značajke su definirane izbacivanjem stupaca koji nisu relevantni za treniranje, poput informacija o klasterima ili komponentama PCA analize, dok je ciljana varijabla izolirana za predviđanje. Nakon toga podaci su podijeljeni na skupove za treniranje i testiranje pomoću metode *train_test_split*, pri čemu je 20% podataka ostavljeno za testiranje, a 80% za treniranje. Ova podjela osigurava da se model može testirati na podacima koji nisu korišteni tijekom procesa učenja.

Za svaki klaster podaci su pohranjeni u definirani format, gdje se za svaki klaster bilježe pripadajuće značajke i ciljne varijable za oba skupa (trening i test). Na kraju, za svaki klaster ispisana je veličina skupa podataka kako bi se potvrdila pravilna podjela i osigurala ravnoteža podataka između različitih skupova.

Ovim postupkom uspostavljen je temelj za treniranje prediktivnog modela koji uzima u obzir specifičnosti pojedinih klastera.

Za taj specifičan dio koda korištena je umjetna inteligencija (Anaconda AI Assistant). Trenutno nije omogućeno da se konverzacija priloži preko linka i slično. Osim toga točan kôd koji je generirala umjetna inteligencija direktno je kopirana i postavljena u blok koda u notebooku.

5. Evaulacija modela

Nastavljajući razvoj prediktivnog modela, primijenjeni su različiti pristupi za izradu i evaluaciju regresijskih modela, pri čemu se prvo koristila metoda potpornih vektorskih strojeva za regresiju (SVR). Kroz inicijalizaciju SVR objekta i treniranje modela nad skupovima podataka za treniranje (*X_train* i *y_train*), model je naučio uzorke u podacima. Nakon toga, predviđanja su generirana za testni skup podataka (*X_test*), a kvaliteta modela evaluirana je pomoću metrika regresije.

Konkretno, izračunata je srednja kvadratna pogreška (MSE) kao mjera prosječne pogreške između stvarnih i predviđenih vrijednosti te koeficijent determinacije (R^2), koji mjeri koliko dobro model objašnjava varijabilnost podataka. Dobiveni rezultati pokazuju visoku preciznost modela s vrlo niskom pogreškom (MSE) i R^2 vrijednošću bliskom 1, što znači da je velik broj varijabilnosti dobro objašnjen modelom. Osim toga, ima i MAE koji predstavlja prosječnu udaljenost predviđenih vrijednosti od realnih vrijednosti, te RMSE koji označava prosječno odstupanje između stvarnih i predviđenih vrijednosti.

Kako bi se osigurala konzistentnost skale numeričkih značajki, korišten je *StandardScaler* za skaliranje podataka. Standardizacija je obavljena tako da su značajke prilagođene prema srednjoj vrijednosti i standardnoj devijaciji na skupu za treniranje, a isti parametri primijenjeni su na testni skup podataka. Ovaj korak je potreban zbog razlika u rasponima među značajkama pa ih je potrebno svesti na usporedive vrijednosti.

Nakon standardizacije podataka, izrađen je linearni regresijski model kroz inicijalizaciju objekta *LinearRegression* i njegovo treniranje na skaliranim podacima za treniranje (*X_train_scaled* i *y_train*). Ovaj postupak omogućuje daljnje testiranje i evaluaciju modela.

Zatim je prikazana putem metode *intercept* vrijednost zavisne varijable (y) kada su sve nezavisne varijable (x) jednake 0. To samo pomaže oko potencijalnog postavljanja formule, ukoliko bi bilo potrebno računati, tj. ručno predvidjeti neku vrijednost. Osim toga, nedostajao bi još i koeficijent smjera, koji je u ovom slučaju pozitivan (kasnije vidljiv *scatterplotom*).

Prikazani su koeficijenti, a zatim je i napokon iskorišten model za predviđanje. Predviđene vrijednosti su prikazane u odnosu na stvarne vrijednosti potrošnje. S obzirom da predviđene i stvarne vrijednosti okvirno prate pravac x = y, moguće je zaključiti da model dobro predviđa vrijednosti, s relativno vidljivim odstupanjima tek u dijelovima gdje općenito nema puno podataka, što je i logično.

Za implementaciju modela koristile su se laboratorijske vježbe s kolegija, kao i online izvor, stranica GeeksForGeeks.

Literatura

1. FOI, Laboratorijske vježbe s kolegija UUI (2025.). *Vježbe 8 - linearna regresija.* Pristupljeno 13.1.2025.
2. FOI, Laboratorijske vježbe s kolegija UUI (2025.). *Vježbe 10 - k-means algorithm.* Pristupljeno 13.1.2025.
3. FOI, Laboratorijske vježbe s kolegija UUI (2025.). *Vježbe 10 - support vector machines.* Pristupljeno 13.1.2025.
4. KDnuggets (2023.). *Principal Component Analysis (PCA) with Scikit-Learn.* [Link za pristup članku], pristupljeno 13.1.2025.
5. 365DataScience (2024.). *How to Combine PCA and K-means Clustering in Python.* [Link za pristup članku], pristupljeno 13.1.2025.
6. StackExchange (2023.). *Perform k-means clustering over multiple columns.* [Link za pristup forumu], pristupljeno 15.1.2025.
7. GeeksForGeeks (2023.). *Support Vector Regression (SVR) using Linear and Non-Linear Kernels in Scikit Learn.* [Link za pristup članku], pristupljeno 15.1.2025.
8. Anaconda (2025.). *Anaconda AI Assistant.* Chat generiran 15.1.2025.