Poročilo o izdelavi projekta

Miha Prajs, 89231246

3. avgust 2025

Kazalo vsebine

Razvoj

Pred začetkom izdelave posameznih algoritmov je bilo potrebno vzpostaviti povezavo med programskima jezikoma Java in R, obdelavo podatkov sem namreč naredil v R. Ker sem začel z izdelavo projekta na operacijskem sistemu Windows, to ni bilo mogoče narediti z Oracleovo knjižnico FastR (https://github.com/oracle/fastr), ampak sem uporabil projekt RCaller (https://github.com/jbytecode/rcaller/).

Celotna koda in delujoči algoritem je dostopen na spletnem mestu GitHub (https://github.com/mihaprajs/famnit24-osupr-project). Težave sem odpravljal s pomočjo spletnih orodji ChatGPT v kombinacij z Gemini, za nekaj poenostavitev kode in "debugging" pa je bil v pomoč *IntelliSence* v *IntelliJ Idea* in *Visual Studo Code* programih.

Manipulacija s podatki

Ker je izbrani algoritem k-NN, je bilo potrebno podatke najprej urediti, iz njih odstraniti odvečne argumente - *id* pacienta - in stolpec z diagnoso (stoplec *diagnosis*) spremeniti v numeričnega. Nato sem sortiral argumente po najboljši korelaciji glede na diagnozo in izbral najboljše 4 ter shranil jih v novo datoteko, ki je nato navoljo algoritmu.

Delitev podatkov v Javi

S programskim jezikom Java sem podatke razdelil v dva dela - učne podatke (80 %) in testne podatke (20 %). Razdeljeni so naključno, zato rezultati variirajo v posameznih zagonih kode.

KNN v Javi

V programskem jeziku Java sem najprej implementiral razdalje (evklidsko, manhattensko in diskretno), ki so na voljo za uporabo. Na podlagi definicij posameznih radalj, se izračunajo razdalje ene testne instance z vsemi iz učnih podatkov. Razdalje so nato urejene po velikosti, najbližje 3 pa uporabljene pri določanju razreda instance. Dobljeni rezultati so testne instance so nato evalvirani z uporabo matrike ter nato izračunana natančnost v odstotkih.

KNN v R

Za implementacijo algoritma v R sem glede na podatke priredil algoritem s spletne strani datacamp.com (https://www.datacamp.com/tutorial/k-nearest-neighbors-knn-classification-with-r-tutorial), vendar sem uporabil le del, kjer je predstavljena uporaba R knjižnice *class*. Da je rezultat ponovljiv ob vsakem poizkusu, je določen "seed", vendar to za pravilno delovanje algoritma ni nujno potrebno.

Rezultati

Java

Algoritem napisan v Java programskem jeziku je vedno vsaj 85 % pravilen pri napovedovanju razreda testnih instanc. Če bi določil "seed", bi vedno dobil enak rezultat.

R

Rezultat algoritma napisanega v programskem jeziku R je odličen, kar 97,35 % pravilnih klasifikacij. To sicer lahko tudi kaže, da je uporabljen dataset premajhen in prihaja do "overfitting-a".

Zaključek

Za natančnejše ovrednotenje algoritma bi bilo potrebno rezultate preveriti na večjem datasetu. Za to bi bi bilo sicer potrebno spremeniti R skripto za manipulacijo podatkov, ker je narejena specifično za uporabljeni dataset (https://www.kaggle.com/datasets/erdemtaha/cancer-data/data).

Algoritem in program se lahko v nadaljenvanju uporabi za osnovo aplikacije z grafičnim vmesnikom, ki bi bil uporabniku prijazen in primeren za uporabo v realnem svetu.