Poročilo - SEMINARSKA NALOGA 2, SAMOSTOJNO IZBRANA PODATKOVNA MNOŽICA - Izbrane teme iz analize podatkov

Matej Novoselec

5. avgust 2024

Koda za izpis izračunov in izris grafov je na voljo v datoteki *seminarska2.ipyb*. Pogon kode je časovno zahteven, zato so vmesni rezultati shranjeni v tekstovni in tabelarni obliki. Za izpis glavnih rezultatov, ki nakazujejo evalvacijo modelov, je dovolj pognati le zadnje okno s kodo.

Naloga: Napoved kvalitete vina

Nalogo sem smiselno razdelil na dva dela - opis in obdelava podatkovne množice; konstrukcija, evalvacija in izbira napovednega modela.

1. OPIS IN OBDELAVA PODATKOVNE MNOŽICE

Prvotni podatkovni množici se nahajata v datotekah *winequality-red.csv* in *winequality-white.csv*. Ker podatkovni množici vsebujeta enake spremenljivke, sem za potrebe projekta podatkovni množici združil, pri čemer sem dodal napovedno spremenljivko *type*, ki kodira ali gre za podatek belega (torej iz datoteke *winequality-white.csv*, nastavimo vrednost 0) ali rdečega (torej iz datoteke *winequality-red.csv*, nastavimo vrednost 1) vina. (Opomba: verjetno bi lahko postopali ločeno, ter tako konstruirali dva bolj kvalitetna napovedna modela, a je podatkov tako (sploh za rdeče vino) nekoliko malo, ter tak pristop za potrebe učenja prispeva k zanimivosti naloge)

Po ogledu podatkovne množice opazimo, da se kar nekaj podatkov ponovi, zato se ponovljenih vnosov znebimo, ter novo-dobljeno podatkovno množico pred zapisom v datoteko *winequality.csv* premešamo. Po hitrem izpisu podatkovne množice opazimo, da je podatkov sedaj 5400, spremenljivk pa 13. Zadnjo spremenljivko, označeno z *quality*, razglasimo za ciljno, ter jo ločimo od ostalih, ki jih razglasimo za napovedne spremenljivke.

Po izpisu opazimo, da v podatkovni množici ni manjkajočih podatkov, zato po navodilih asistenta konstruiramo tudi novo podatkovno množico (označimo jo z *X_rem*), kjer naključno izberemo 5 izmed 12-tih napovedni spremenljivk, ter

pri vsaki izbrani, pri naključno izbranih 500 (izmed 5400, kar je okoli 9.3%) podatkih vrednosti nastavimo na "manjkajoče"/"Nan". Tako bomo lahko kakovost napovednega modela na "polni"učni podatkovni množici primerjali s kakovostjo napovednega modela na podatkovni množici z manjkajočimi vrednostmi.

Sedaj se lotimo odpravljanja naslednjih dilem:

1. ne vemo kako se obnašajo/so porazdeljene napovedne spremenljivke in ali lahko kako dodatno predprocesiramo podatke, da takoj iz teoretičnega vidika preprečimo preprileganje ter izboljšamo stabilnost kasneje konstruiranega napovednega modela,
2. ne vemo kakšna je porazdelitev/katere vrednosti zavzame ciljna spremenljivka,
3. v eni verziji podatkovne množice imamo manjkajoče vrednosti,
4. imamo le eno podatkovno množico, ki jo moramo razdeliti na učno in testno podatkovno množico.

Da dobimo občutek za porazdelitev napovednih spremenljivk, si za podatke vsake izrišemo histogram in škatlo z brki. Za lepši prikaz, pri vsaki napovedni spremenljivki izberemo smiselno (uporaba fukcije *optimal_bins*) število "košev"na histogramu. Porazdelitve napovednih spremenljivk so vidno različne, vsaka napovedna spremenljivka pa zavzame tudi drugačen razpon vrednosti. Zapisano o porazdelitvah dodatno potrdi izris škatel z brki, na katerih opazimo, da imajo nekatere napovedne spremenljivke kar veliko osamelcev ("outlier-jev"). S standardizacijo/skaliranjem napovednih spremenljivk, ter odstranitvijo ekstremnih vrednosti (to v nadaljevanju naredimo s funkcijo *clean_outliers*, ki z uporabo *Z-score-a* odstrani podatke, katerih vrednost le-tega po absolutni vrednosti presega mejo 3) lahko v teorijo prispevamo k stabilnosti napovednega modela, zato bomo v nadaljevanju storili prav to. Skaliranje podatkov bo tu narejeno z uporabo transformacije *MinMaxScaler*, ki vrednosti vsake napovedne spremenljivke podatkovne množice spravi v razpon $[0, 1]$. Vendar z izvedbo zapisanega še nekoliko počakajmo.

Posvetimo se sedaj drugi alineji. Izpis podatkov za napovedno spremenljivko nakazuje, da je le-ta diskretna, ter zavzame vrednosti med 3 in 9. Porazdelitev očitno ni enakomerna, vrednost 6 in 5 v podatkovni množici namreč zavzamemo kar 2323-krat oziroma 1752-krat, vrednosti 3 in 9 pa srečamo le 30-krat oziroma 9-krat. Porazdelitev ciljne spremenljvke bomo v nadaljevanju smiselno upoštevali pri razdelitvi podatkovne množice na učno in testno množico.

Problem iz tretje alineje, za primer podatkovne množice z manjkajočimi vrednostmi, rešimo z uporabo vnašalnika ("imputer-ja"). To je v tem primeru storjeno z uporabo transformacije, ki temelji na *KNNImputer*. Ta manjkajoče vrednosti dopolni po principu najbližjih sosedov (tu sem jih izbral 5).

Sedaj zgoraj zapisano upoštevajmo pri razdelitvi podatkovne množice na učni in testni del. Uporabili bomo prečno preverjanje, ter podatkovno množico razdelili na 5 "fold-ov", pri čemer bomo vsakič model natrenirali na 4/5 podatkov, ter zadnjo 1/5 uporabili za testno množico. V duhu druge alineje,

bomo pri razdelitvi na odseke upoštevali porazdelitev ciljne spremenljivke, ter razdelitev naredili s pomočjo *StratifiedKFold*, ki mu bomo za ponovljivost dodali *random_state*. Ko konstruiramo odseke, bomo na vsakem odseku podatke v duhu tretje alineje (če je to potrebno) dopolnili, ter se znebili manjkajočih vrednosti. Nato bomo po zgornjem komentarju, ki se nanaša na prvo alinejo, podatkom odstranili ekstremne vrednosti, ter jih nato še skalirali. Tako dobimo seznam (v resnici sezname) petih odsekov podatkovne množice, s katerimi bomo kasneje konstruirali, natrenirali, ter evalvirali izbrane napovedne modele. Vsa zapisana "mašinerija"je skrito izvedena pod funkcijo *stratifeid_k_fold*, ki nam dobljene sezname, zaradi same časovne zahtevnosti izvedbe, tudi shrani v datoteke. Omenjeno funkcijo tako poženemo za prvotno podatkovno množico (dobimo datoteke *X_train_list.pkl, y_train_list.pkl, X_test_list.pkl*, ter *y_test_list.pkl*), kot tudi tisto, ki smo ji prvotno odstranili nekaj vrednosti, ter smo jih nato tekom funkcije zapolnili (dobimo datoteke *X_train_list_rem.pkl, y_train_list_rem.pkl, X_test_list_rem.pkl*, ter *y_test_list_rem.pkl*)

Preden se osredotočimo na konstrukcijo in izbiro napovednih modelov, je smiselno še nekoliko pridobiti občutek, katere napovedne spremenljivke so s ciljno spremenljivko najbolj korelirane (s tem se namreč vsaj v teoriji ponudi možnost, da bi upoštevali le-te, ter se tako potencialno izognili preprileganju napovednega modela, ter izboljšali "kvaliteto"njegove napovedi na testni množici). To sem poskušal narediti na dva načina. Prvi, nekoliko bolj učen, z uporabo *PCA*, ni pozitivno prispeval k napovedi modelov (ter je le bistevno prispeval k časovni zahtevnosti, zato sem ga na koncu izpustil), drugi, nekoliko bolj primitiven in "neprecizen", pa je uporabljen v nadaljevanju. Tu sem naključno izbral eno izmed učnih množic (za vsako verzijo - najprej odstranjeno in dopolnjeno, ter "navadno"), ter si ogledal, koliko so posamezne napovedne spremenljivke s ciljno spremenljivko korelirane. Nato sem po absolutni vrednosti koreliranosti okvirno skonstruiral tri sezname, ki bodo ponudili tri možne načine izbire napovednih spremenljivk za napovedne modele.

2. KONSTRUKCIJA, EVALVACIJA IN IZBIRA NAPOVEDNEGA MODELA

Kot omenjeno, bomo vsako vrsto napovednega modela konstruirali za tri podatkovne množice (določene s seznamom napovednih spremenljivk, ki jih bomo uporabili), ter potem naredili prečno preverjanje na petih različnih odsekih. Iz ogleda ciljne spremenljivke je jasno, da gre za nalogo klasifikacije, pri kateri podatkom za vino določimo njegovo kvaliteto v razponu od 3 do 9.

Ogledali si bomo torej (standardne) klasifikacijske modele *DecisionTreeClassifier*, *RandomForestClassifier*, *KNeighborsClassifier*, *LogisticRegression* in *GradientBoostingClassifier*, dodal pa sem še tri nekoliko različne nevronske mreže. (Opomba: preizkusil sem jih več (testiral sem različno postavitev plasti, ter aktivacijskih funkcij, dodal sem različne vmesne normalizacije in "dropout-te), te sem pustil, ker se mi zdi da smiselno zavzamejo raznolikost te metode).

Za konstrukcijo, trening in evalvacijo (standardnih - torej vseh razen nevronskih mrež) modelov poskrbi funckija *check_model*, ki za vhod dobi seznam

odsekov podatkovne množice, ter sistematično evalvira vrsto modela na vsakem izmed odsekov. Vrednosti (kasneje opisanih in komentiranih) metrik na vsakem odseku so shranjene v seznam, nato pa fukcija vrne povprečje vsake izmed metrik, kar nakaže kvaliteto napovedi vsakega izmed modelov.

S konstrukcijo in evalvacijo modelov nevronskih mrež imamo nekoliko več težav. Konstrukcija poteka z uporabo funkcij *create_nn_model#*, ki je skupaj s samim treningov in evalvacijo izvedena v funkciji *make_check_nn_model*. Tu je vredno omeniti, da pred treningom nevronskih mrež ciljno spremenljivko učne množice zakodiramo z uporabo kodirnika *OneHotEncoder*, ki nam eno stolpično diskretno spremenljivko z vrednostmi od 3 do 9 spremeni v 7 stolpično tabelo, pri katerem binarna vrednosti v i-tem stolpcu nakazuje, ali je podatek v tej vrstici pripadal $(3 + i)$-ti vrednosti prvotne ciljne spremenljivke. Slednje potrebujemo zato, ker je na zadnjem nivoju nevronskih mrež (s 7-mi vozlišči) uporabljena aktivacijska funkcija *softmax*, ter si lahko izhod na j-tem vozlišču predstavljamo kot verjetnost, da gre za podatek, kvalitete $j+3$ (izhod je torej 7-stolpičen). Ker za napoved potrebujemo eno diskretno vrednost (med 3 in 9), vrnemo indeks tistega vozlišča, ki ima največjo verjetost, ter ga nato povečamo za 3. Podobno kot zgoraj, nato na vsakem odseku izračunamo vrednosti vsake izmed metrik, ter vrnemo vrednosti njihovega povprečja po odsekih.

Komentirajmo sedaj še metrike, ki jih bomo uporabili pri evalvaciji napovednih modelov. Za začetek izrčunamo *accuracy*, ker pa imamo nenakomerno porazdeljeno diskretno ciljno spremenljivko, bomo več pozornosti posvetili vrednosti *f1_score-a* vsakega napovednega modela. Za polnost analize evalvaciji vsakega modela dodamo še metriki *precission* in *recall*. (Opomba: omenimo, da so za izračun zadnjih treh izbrali obtežene različice povprečja, ki upoštevajo neenakost razlike med razredi ciljne spremenljivke) Omenimo tudi, katere vrednosti metrik smatramo za dobre. Prav pri vseh štirih omenjenih metrikah, se vrednosti nahajajo med 0 in 1, za "dobre vrednosti", pa smatramo tiste, ki so bližje vrednosti 1.

Sedaj tako za prvotno podatkovno množico, kot tudi tisto, ki smo jo dobili z odstranitvijo in dopolnitvijo manjkajočih podatkov, konstruiramo po tri sezname odsekov podatkovnih množic, pri čemer upoštevamo različno število (po koreliranosti, kot sem komentiral) napovednih spremenljivk. Vsak napovedni model na njej ovrednotimo, ter slovar z rezultati shranimo v tekstovno datoteko *res_{ime modela}_{seznam nap spr po vrsti}_{_ ali rem}.txt*

Da bomo napovedne modele med seboj lahko primerjali, narišemo dve vrstici (vsaka za eno podatkovno množico) s tremi (za vsako seznam uporabljenih spremenljivk) grafi *heatmap*, ki prikazujejo vrednosti vseh metrik vsakega modela.

Najprej komentirajmo razliko med prvotno podatkovno množico, ter tisto, ki smo ji najprej nekaj vrednosti odstranili in jih nato dopolnili. Opazimo, da so vrednosti vseh metrik zelo blizu, le na določenih mestih je opazna manjša razlika, kar je nekoliko presenetljivo (da je razlika tako majhna, sam bi pričakoval nekoliko večjo). Celostno gledano je opazno, da so povsod, razen pri napovednem modelu *Neural Network2*, praviloma vrednosti prvotne podatkovne množice malenkost višje od tiste z odstranjenimi vrednostmi. Zanimivo je, da

je pri omenjenem modelu *Neural Network2* praviloma ravno obratno. Predvidevam, da bi bilo večjo razliko možno opaziti, če bi odstranili še več podatkov. V nadaljevanju analize zato rezultate obeh množic obravnavajmo kar skupaj.

Opaziti gre, da vrednosti vseh metrik v splošnem rastejo, ko modelu dodamo več napovednih spremenljivk. Zato je smotrno pri podatkovni množici upoštevati podatke vseh napovednih spremenljivk, ter se iz tega vidika ne rabimo ukvarjati s preprileganjem.

Sedaj natančneje primerjajmo modele v primeru, ko upoštevamo vse napovedne spremljivke. Opazimo, da so prav pri vseh metrikah največje vrednosti dosežene pri modelu *Random Forest Classifier*. Ta doseže povprečen *f1_score-a* kar 0.54, vrednosti preostalih metrik pa so še nekoliko višje. Torej je po videnem ravno ta najbolj primeren za to podatkovno množico. Po kvaliteti napovedi mu sledijo *Gradient Boosting Classifier*, *Logistic Regression*, ter *K-Nearest Neighbors Classifier*, ki še dosegajo vrednosti metrik nekoliko čez 0.5, napovedi ostalih modelov pa po evalvaciji metrik že kar vidno zaostajajo. Nekoliko presenetjivo se vrednosti metrik konstruiranih nevronskih mrež (prve in tretje) gibljejo le okoli 0.45, kar predvidevam, da bi se dalo z večjim časovnim obdobjem učenja (vsaj nekoliko) izboljšati. Evalvacija metrik pri drugi nevronski mreže nakazuje, da njena struktura za to podatkovno množico ni primerna, ter so ostali napovedni modeli veliko bolj primerni.