Poročilo - SEMINARSKA NALOGA 1 - DN3, NAL2 - Izbrane teme iz analize podatkov

Matej Novoselec

2. avgust 2024

Koda za izpis izračunov in izris grafov je na voljo v datoteki *seminarska1_dn3.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 zamujanja s plačilom obroka posojila

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

1. OPIS IN OBDELAVA PODATKOVNE MNOŽICE

Podatkovna množica ima 11 napovednih in eno ciljno binarno spremenljivko, ki kodira dogodek, ali je posameznik s plačilom posojila zamudil za več kot 90 dni ali ne.

Hiter izpis podatkovne množice nakazuje manjkajoče vrednosti tako v učni, kot tudi testni množici. Te se pojavijo predvsem pri napovedni spremenljivki, ki sem jo poimenoval *monthly_income*. Po štetju manjkajočih vrednosti opazimo, da jih v tem stolpcu manjka skoraj 20%. Pri drugih napovednih spremenljivkah, z izjemo še okoli 2.5% pri spremenljivki *number_of_dependents*, manjkajočih vrednosti ni zaznati.

Pred nadaljnjo analizo zato najprej "luknje"ustrezno zapolnimo. To storimo z uporabo algoritma *KNNImputer*. Ker je algoritem z izbiro 5-ih najbližjih sosedov kar časovno potraten, zapolnjeni podatkovni množici shranimo v datoteki *0x_train.csv*, ter *0x_test.csv*.

Sedaj, za boljšo predstavo o podatkovni množici, s histogramom in škatlo z brki še izrišimo podatke vsake napovedne spremenljivke posebej (tu za lepši prikaz prilagodimo število "škatel"vsake izmed napovednih spremenljivk). Na histogramih opazimo kar veliko razlik med porazdelitvami posameznih napovednih spremenljivk, škatle z brki pa pri večini napovednih spremenljivk nakazujejo na veliko število osamelcev ("outlier-jev").

Ker ekstremne vrednosti velikokrat negativno vplivajo na natančnost našega napovednega modela, se lotimo odstranjevanja le-teh. Tega sem se tu lotil z izračunom *Z-score-a* podatkov, ter izločil podatke, ki presežejo (standardna izbira) vrednost 3. Opazimo, da je tako od začetnih 12000 podatkov v učni množici, ostalo 115723 podatkov (gre torej za "izgubo"okoli 3.5% podatkov).

Raznolikost porazdelitve in vrednosti napovednih spremenljivk se izraža tudi v razponu vrednosti, ki jih je zaznati pri vsaki napovedni spremenljivki. Pred nadaljevajem zato podatkovno množico ustrezno skaliramo - to sem v tem primeru storil z *MinMaxScaler*, ki vrednosti napovednih spremenljivk spravi v domeno $[0,\ 1]$.

Imamo kar veliko število napovednih spremenljivk, zato se že na tem mestu lahko bojimo preprileganja kasneje izbranega napovednega modela. Sledečega se z mislijo na nadaljevanje lotimo na dva načina - uporabo algoritma *PCA*, ki temelji na iskanju glavnih komponent (torej tistih, ki bi načeloma na vrednost ciljne spremenljivke imeli največji vpliv), ter nekoliko bolj primitivno, hitrim izračunom koreliranosti med spremenljivkami, pri čemer se osredotočimo na koreliranost napovednih spremenljivk s ciljno spremenljivko. Uporaba *PCA* po testu rezultatov ni izboljšala (ter že tako ali tako časovno zahteven postopek le podaljšala), zato sem se v nadaljevanju osredotočil le na drugo omenjeno metodo. Rezultate dobro predstavi izris grafa *heatmap*. Opazimo, da so z zamuditvijo za več kot 90 dni najbolj korelirane napovedne spremenljivke, ki kodirajo pretekle zamude (napovedne spremenljivke, poimenovane *late_3059*, *late_ 90* in *late_6089*). Te vzemimo kot prvi seznam napovednih spremenljivk po pomembnosti. Drugi seznam tvorimo z dodajo še nekaterih napovednih spremenljivk, pri čemer seveda upoštevamo absolutne vrednosti izpisanih koreliranosti s ciljno spremenljivko - tu sedaj izpustimo le prvi dve napovedni spremenljivki, ker imata izredno majhno vrednost koreliranosti s ciljno spremenljivko. Tretji seznam pa vključuje kar vse napovedne spremenljivke.

V nadaljevanju bomo izbrane modele evalvirali pri vseh treh seznamih, ter analizirali, pri katerem dobimo najbolj ugodne vrednosti in se tako potencialno izognili preprileganju.

Veliko je bilo govora o porazdelitvah napovednih spremenljivk, pa si nekoliko oglejmo še porazdelitev ciljne spremenljivke.

Ta je, kot že omenjeno, binarna, vendar pa porazdelitev ni enakomerna - po izpisu opazimo, da ciljna spremenljivka vrednost 1 v učni množici zavzame le 8120-krat, medtem ko vrednost 0 zavzame kar 111880-krat. Vrednost 1 torej zavzamemo le v okoli 6.8% primerov. Želeli bi si, da model z učenjem dobi enako število pozitivnih kot tudi negativnih primerov (ker je drugače model nagnjen k napovedovanju vrednosti 0). To storimo z funkcijo *resample_data*, ki ponovno ("pravično") vzorči iz podatkovne množice, ter tako poskrbi za konstrukcijo novih testnih množic. Ponovno vzorčenje je storjeno na tri nove načine (*oversampling* z *RandomOverSampler*, *undersampling* z *RandomUnderSampler*, ter *smote* s *SMOTE*), ki skupaj s prvotno učno podatkovno množico ustvarijo štiri različne podatkovne množice na katerih bomo učili posamezen model.

Zaradi časovne zahtevnosti vsako izmed podatkovnih množic shranimo v svojo smiselno poimenovano *.csv* datoteko.

2. KONSTRUKCIJA, EVALVACIJA IN IZBIRA NAPOVEDNEGA MODELA

Kot omenjeno, bomo za vsako vrsto modela imeli 12 verzij - glede nato "na kateri"učni množici je natreniran (3 izbire seznama napovednih spremenljivk, ki jih bomo uporabili, ter 4 učne podatkovne množice - {normalna/prvotna, undersampling, oversampling, smote}).

Gre za nalogo binarne klasifikacije, zato bomo uporabili (standardne) klasifikacijske modele *DecisionTreeClassifier*, *RandomForestClassifier*, *KNeighborsClassifier*, *LogisticRegression* in *GradientBoostingClassifier*, kot tudi nekaj samostojno zgrajenih nevronskih mrež (Opomba: tu sem v kodi pustil tri (le eno pri zagonu koda evalvira, ostale sem zakomentiral), ki se med seboj vidno razlikujejo (po aktivacijskih funkcijah, strukturi plasti, nekatere imajo večje število vmesnih normalizacij, nekatere večje verjetnosti "dropout", ...), preizkusil sem jih še več, a se mi je za prikaz raznolikosti zdelo smiselno ohraniti te tri - v izpis/graf sem zaradi časovne zahtevnosti potem dodal le najboljšo).

Komentirajmo še evalvacija napovednih modelov. Kot že omenjeno, ciljna binarna spremenljivka ni enakomerno porazdeljena, zato vrednost "osnovne"metrike *accuracy* ne oriše nujno korektno cele zgodbe/ne moremo samo prek vrednosti te metrike sklepati o kvaliteti napovednega modela. Zato modele ovrednotimo tudi z metriko *f1_score*, ter *recall* in *precision* (ki v obzir vzamejo tudi porazdelitev ciljne spremenljivke po razredih). Omenimo tudi, katere vrednosti metrik šmatramo za dobre". Prav pri vseh štirih omenjenih metrikah, se vrednosti nahajajo med 0 in 1, za "dobre vrednosti", pa smatramo tiste, ki so bližje 1.

Za konstrukcijo, trening in evalvacijo standardnih modelov (vseh razen nevronskih mrež), poskrbi funkcija *evaluate_model*, ki vrne vrednosti vseh zgoraj zapisanih metrik za posamezen model. Te so potem zbrane (po podatkovni množici na katero so vezani) ter skupaj zapisane v slovar in shranjene v tekstovno obliko (zaporedna številka tu označuje kateri seznam spremenljivk smo vzeli, okrajšava norm, over, under, smote pa način prevzorčenja za podatkovno množico modelov).

Situacija za vse tri nevronske mreže je podobna, le da tu vsako nevronsko mrežo evalviramo nekoliko ločeno (lahko bi seveda tudi skupaj, če bi le to tako zastavil v kodi). Nato vrednosti vseh metrik na enak način zabeležimo v slovar in zapišemo v pripadajočo datoteko.

Sedaj lahko rezultate zberemo, ter izrišemo štiri vrstice (ena za vsako vrsto prevzorčenja) treh (ena za vsak seznam napovednih spremenljivk, ki jih upoštevamo) grafov *heatmap*. Na vsakem grafu (ki pripada podatkovni množici) so za vsak model izpisane vrednosti vsake izmed metrik.

Sedaj komentirajmo dve stvari - najprej izbiro učne podatkovne množice (v duhu vrste prevzorčenja, ter duhu zožitve napovednih spremenljivk, ki jih upoštevamo), ter nato še kvaliteto napovedi vsake vrste modela.

Hitro opazimo, da je prevzorčenje potencialno smiselno le, če gre za prevzorčenje *SMOTE* - hitro namreč opazimo, da so v vrednosti (večinoma vseh) metrih pri vseh modelih nekoliko (in ponekod celo bistveno) nižje, kot vrednosti, ki jih gre zapaziti če se osredotočimo na prevzorčenje *SMOTE*, ali pa prevzor-

čenja ne naredimo. Toda ob natančnejšem ogledu opazimo, da tudi v primeru, ko prevzorčimo z metodo *SMOTE*, se vrednosti metrik ne spremenijo bistveno (vrednosti so med seboj zelo podobne, v nekaterih primerih celo višje/boljše, ko prevzorčenja ne naredimo).

Sedaj se torej osredotočimo le na vrstico, v kateri prevzorčenja nismo storili. Tu so barve na grafih lahko nekoliko zavedljive, saj na prvi pogled deluje, da z izbiro vseh ali le 9-tih najbolj koreliranih napovednih spremenljivk dosežemo občutno višji *precision* in *recall* večinomo pri vseh napovednih modelih, vendar temu ni tako. Vrednosti so pri vseh napovednih modelih, z izjemo modela *Decision Tree Classifier*, več ali manj enake, ali pa se razlikujejo le za 0.01. Tudi vrednosti vseh metrik pri modelu *Decision Tree Classifier* se ne spremenijo bistveno (še največ *accuracy*, za 0.05), vendar so ravno te male razlike pri tem modelu razlog, da se nadaljevanje drugih dveh grafov pri metrikah *precision* in *recall* obarva nekoliko bolj temno, ter nakaže bistveno višjo vrednost. Rahel padec kvalitete napovedi pri omenjenem modelu gre verjetno pripisati preprileganju.

Ker med ostalimi modeli ni bistvenih razlik, ne morem zares trditi, da bi bil kateri izmed njih veliko bolj primeren kot drug - vsi opisani modeli se mi zdijo primerni, če podatkov ne prevzorčimo. Kot omenjeno, zaradi (kot kaže tu hitrejšega) preprileganja, bi se pri tej podatkovni množici izognil le uporabi modela *Decision Tree Classifier*.

Zanimivo je, da kljub nizkemu številu iteracij (zaradi že celotne časovne zahtevnosti) tudi (sicer najboljša glede na moj izbor) nevronska mreža hitro dosega primerljive rezultate z ostalimi modeli. Menim, da bi se s potencialnim dodatnim treningom dalo rezultat verjetno še nekoliko izboljšati.