2. Domača naloga
Učenje iz podatkovnih tokov

Tine Markočič

Junij 2023

1 Regresija

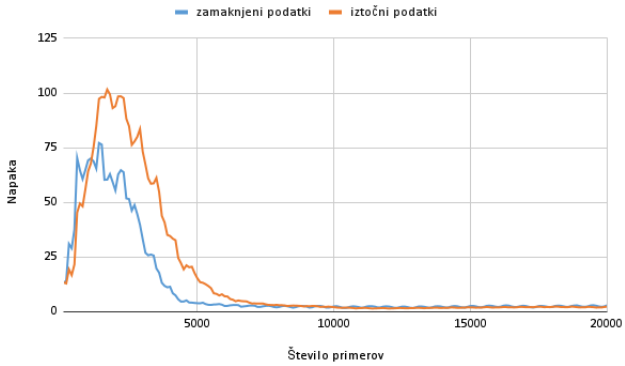
1.1 Podatkovni nabor

V podatkovnem naboru imamo napovedni spremenljivki `x` in `y` ter odvisno spremenljivko `c`, vseh primerov pa je 20.000. Ob izrisu grafov posameznih spremenljivk opazimo nekakšno periodičnost, hkrati pa se ob vsaki periodi razpon vrednosti povečuje. Posebej zanimivo je, da spremenljivka `c` zavzame bistveno večje vrednosti kot napovedni spremenljivki. Sklepamo, da je v spremenljivki prisotna avtokorelacija, torej da je odvisna od svojih prejšnjih vrednosti. Odločimo se pripraviti podatkovni nabor v katerega dodamo spremenljivke `c-3`, `c-2` in `c-1`, zadnje tri vrednosti spremenljivke c. Ker za prve tri primere nimamo podatkov, te vrstice izbrišemo. Zamaknjeni podatkovni nabor shranimo v datoteko `dn2a_lag.arff`.

1.2 Napaka in model

Na zamaknjenem naboru izberemo nalogo `EvaluatePrequentialRegression` in poženemo privzeto metodo FIMT-DD za regresijo z delitvenim kriterijem `VarianceReductionSplitCriterion`. Nastavimo še `sampleFrequency=100`, tako da metoda izvede 200 meritev. Po približno 5.000 primerih dobimo zadovoljivo majhno napako. Za primerjavo enako storimo še na izhodiščnem naboru `dn2a.arff` in napaka je astronomsko večja, model se v bistvu ne nauči nič pametnega. Nekaj presenetljivega pa se zgodi, ko pri modelu na izhodiščnem naboru izberemo kriterij delitve `GiniSplitCriterion`. Napaka se obnaša zelo podobno kot pri privzeti metodi na zamaknjenem naboru (kar je prikazano na sliki 1), modelu torej uspe prepoznati dinamiko sistema.

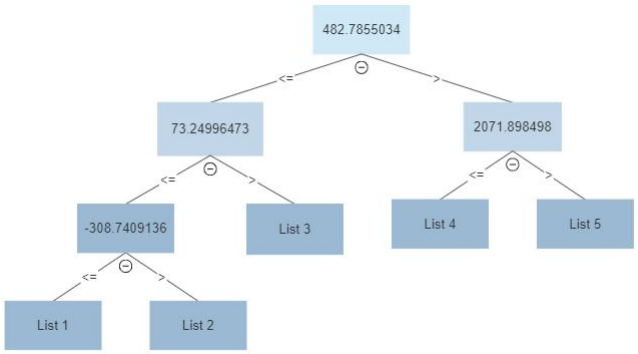
Prednost uporabe izhodiščnega nabora je, da je učenje na njem časovno manj zahtevno. Ker pa je zamaknjeni nabor bolj univerzalno dober (na njem dobro delujejo vse metode, ki sem jih preizkusil), se odločimo, da bomo na njem izvajali nadaljnjo analizo.



Slika 1: Primerjava napak za izhodiščne in zamaknjene podatke na različnih modelih

Na izbranem naboru naučimo model s privzeto metodo FIMT-DD, tako da izberemo nalogo `LearnModelRegression` in shranimo zapisano drevo.

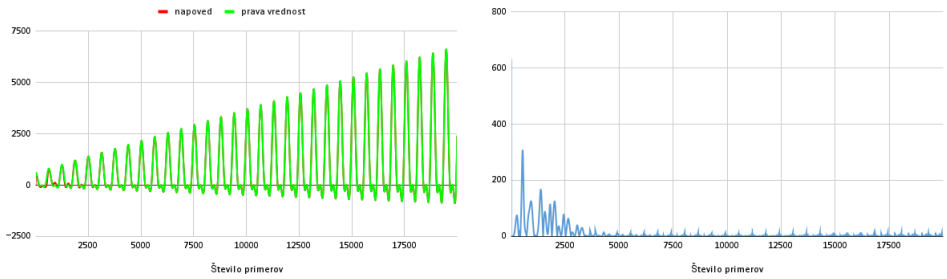
Dobimo drevo s petimi listi, ki se odloča le na podlagi spremenljivke `c-1`, v listih pa ima zapisane enačbe za napoved. Točen model je shranjen v datoteki `dn2a_tree.txt`.



Slika 2: Drevo modela na zamaknjenih podatkih

1.3 Napoved

Preizkusimo še napovedno točnost na zamaknjenem naboru, tako da pri evaluaciji modela napovedi shranimo v datoteko `dn2a_lag.pred`. Največja odstopanja od pravih vrednosti so vidna na začetku, ko se model še uči. Zaradi velikih vrednosti spremenljivke `c` in prekrivanja z napovedmi narišemo še graf absolutne razlike med napovedjo in pravo vrednostjo (slika 3, desni graf). Vidimo, da



Slika 3: Primerjava napovedi in prave vrednosti

modelu povzročajo težave prehodi iz negativnih vrednosti spremenljivke `c` v pozitivne, zato so na grafu "otočki", kjer je napaka nekoliko večja.

2 Klasifikacija

2.1 Podatkovni nabor

Ponovno imamo podatkovni nabor z 20.000 primeri, na katerem izvajamo binarno klasifikacijo. Pri analizi značilk ugotovimo naslednje: značilka `f1` zavzame vrednosti med -1 in 1, `f2` je strogo naraščajoča, zato je neprimerna za učenje modela, pri značilki `f3` pride na neki točki do spremembe, značilki `f4` in `f5` zavzameta omejeno število vrednosti, a ju vseeno pustimo v numerični obliki, saj ni videti, da bi bile vrednosti med seboj posebno odvisne.

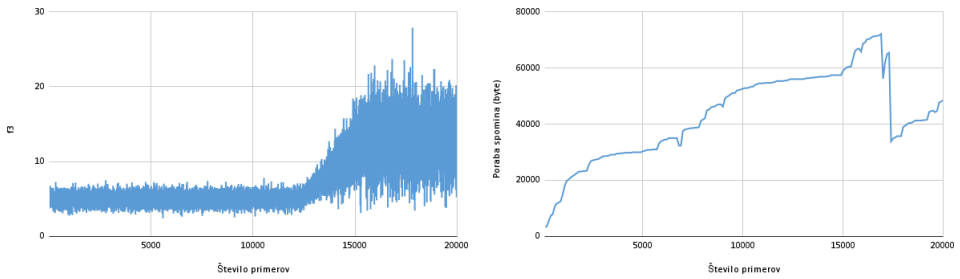
Pripravimo podatkovni nabor brez značilke `f2` in ga shranimo v datoteko `dn2b_-f2.arff`.

2.2 Spremembe v naboru

Prvo spremembo smo opazili že pri pripravi nabora, in sicer se pri značilki `f3` začnejo vrednosti po približno 12.000 primerih povečevati, opazna pa je tudi večja variacija vrednosti. Pride torej do spremembe porazdelitve na vhodnem prostoru, kar klasificiramo za navidezno spremembo. Po časovnem profilu pa gre za koračno spremembo, saj se zgodi postopoma (slika 4, levi graf).

Za odkrivanje drugih sprememb je potrebno na naboru naučiti model. To storimo s klasifikacijsko nalogo `EvaluatePrequential` in metodo `HoeffdingAdaptiveTree`. Poimenujmo model `Adaptive_Tree`. Najprej pogledamo, ali pride na kateri točki do velikega padca točnosti modela, vendar je ta dokaj konstantno okoli 94%, zato o bistvenih spremembah ne moremo govoriti.

Naslednje si lahko ogledamo porabo spomina, saj nam ta nakazuje, kdaj pride do sprememb modela (slika 4, desni graf). Tu opazimo velik padec po približno 17.000 primerih, torej je bil narejen nov model in je bilo veliko starega pozabljenega. Prišlo je do spremembe odvisnosti ciljnih vrednosti od napovednih, to je prava sprememba, pa tudi nenadna, saj se zgodi takoj.

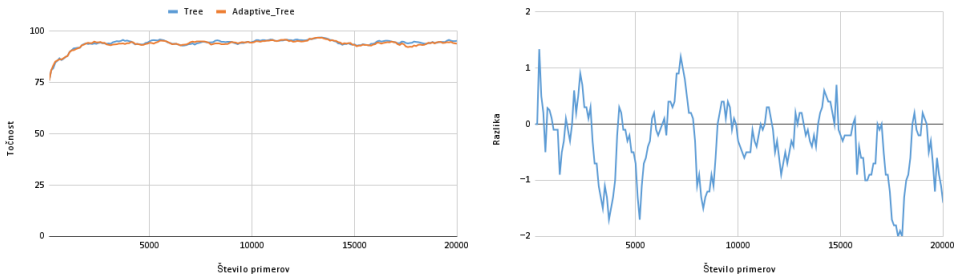


Slika 4: Spremembe podatkov in modela

Manjša padca porabe se pojavita tudi po približno 7.000 in 9.000 primerih, torej imamo tudi tu dve manjši pravi nenadni spremembi.

2.3 Primerjava modelov

Metoda, ki smo jo uporabili na prejšnji točki, je nastavljena tako, da prepoznava spremembe in se nanje prilagaja. Preizkusimo kakšno točnost dobimo, če naučimo model z metodo brez zaznavanja sprememb, `HoeffdingTree`. Poimenujmo model `Tree`.



Slika 5: Primerjava napovednih točnosti

Opazimo, da imata oba modela zelo podobno točnost, zato imamo na sliki 5, desni graf, prikazano razliko točnost(Adaptive_Tree) − točnost(Tree).

Osredotočimo se na opažene spremembe. Po navidezni spremembi se točnost obeh modelov zmanjša, na kar se bolje odzove model `Tree`. Razlika v točnosti skoraj konstantno pada, vse dokler v modelu `Adaptive_Tree` ne pride do prave spremembe. Graf porabe spomina je za model `Tree` nepadajoč, torej v njem ne moremo opaziti pravih sprememb.

Sklepamo torej, da se po pravih spremembah model `Adaptive_Tree` dobro prilagodi, poveča se njegova točnost, je pa v celoti nekoliko slabši od modela `Tree`.