Druga domača naloga

Učenje iz podatkovnih tokov

Napredno strojno učenje 2023

Dejan Perić

Vpisna številka: 27222015

6. junij 2023

Prva naloga-regresija

1.0 Priprava podatkovnega nabora

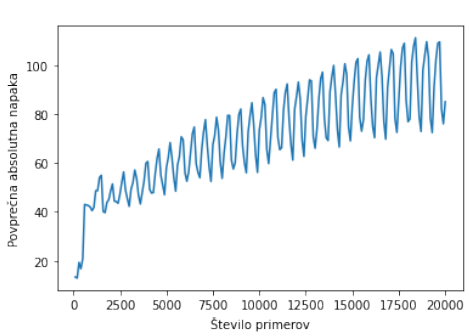
Najprej si je treba ustrezno pripraviti podatkovni nabor, ki mora biti shranjen v .arff datoteki. Za pretvorbo sem najprej csv datoteko odprl kot pandas tabelo, potem pa sem to s funkcijo dump iz knjižnice arff zapisal v arff datoteko `dn2a.arff`. Na podoben način sem zapisal tudi datoteko `dn2a-window3.arff`, le da sem vmes drugače pripravil tabelo.

Ker je pri učenju iz podatkovnih tokov pomembna tudi zgodovina, sem drugo tabelo pripravil tako, da sem za vsako ciljno vrednost c dodal vrednosti `x`, `y` in `c` treh prejšnjih vrstic. Na primer, `x` sem zamenjal s stolpci `x-3`, `x-2` in `x-1`. Vsaka vrstica te tabele ima torej vključno z ciljno vrednostjo `c` 10 različnih atributov. V tabelo ne dodamo prvih treh ciljnih vrednosti `c`, saj so vrstice za te nepopolne in zaradi veliko primerov ne bodo bistveno vplivali na rezultat.

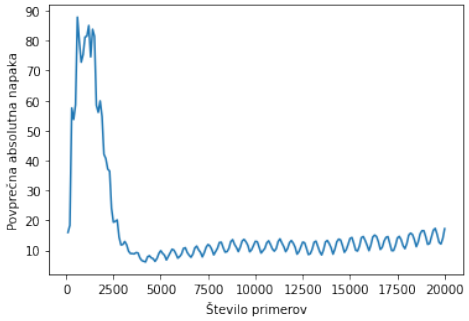
1.1 Učenje modela

V programu MOA naučimo drevesa na pripravljenih podatkovnih naborih. V grafičnem vmesniku MOA izberemo regresijo, kjer ustrezno nastavimo metodo FIMT-DD kot model za učenje, eno izmed zgornjih dveh `.arff` datotek kot podatkovni tok. Spremenimo še dva izmed parametrov, `instanceLimit=`

`100.000` in `sampleFrequency=100`. Za metodo evaluacije pa izberemo evaluacijo regresije z uporabo oken, `WindowRegressionPerformanceEvaluator`.



Slika 1: Graf povprečne absolutne napake za osnovni podatkovni nabor



Slika 2: Graf povprečne absolutne napake za razširjen podatkovni nabor

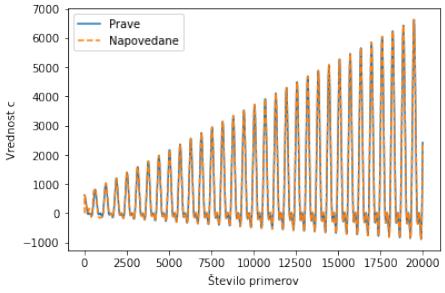
Iz grafa povprečne absolutne napake dveh modelov razberemo, da je sicer za prvih 1000 primerov boljši podatkovni nabor samo s tremi stolpci, vendar se po 2000 primerih model z drugim podatkovnim modelom močno izboljša in tudi napaka se omeji. V pogledu napake je ta podatkovni nabor torej boljši. Oba grafa imata kasneje čik-cak"obliko grafa, kar je po mojem mnenju posledica specifičnega spreminjanja vrednosti c (glej sliko 4). Drugi podatkovni nabor je boljši tudi glede na število listov v naučenem drevesu; to naučeno drevo jih ima namreč le 5. Model, naučen na osnovnem podatkovnem naboru, ima 65 listov, kar je že dosti. Je pa učenje na prvem podatkovnem naboru, glede na rezultate, približno štirikrat manj časovno zahtevno.
Na sliki jjj vidimo izpisano drevo za model naučen na razširjenem podatkovnem naboru. Drevo se razveja glede na prejšnjo vrednost `c`.

```
if [att 9:c-1] <= 482.7855034440994:
  if [att 9:c-1] <= 73.24996473188253:
    if [att 9:c-1] <= -308.7409135716775:
      Leaf [class:c] = 0,2453 * [att 1:x-3] -0,3463 * [att 2:x-2] -0,7127 * [att 3:x-1]
      -0,2360 * [att 4:y-3] + 1,0372 * [att 5:y-2] -0,8829 * [att 6:y-1] + 0,6130 * [att 7:c-3]
      + 0,5675 * [att 8:c-2] + 0,6101 * [att 9:c-1] + -0.0027357801879556868
    if [att 9:c-1] > -308.7409135716775:
      Leaf [class:c] = 0,2461 * [att 1:x-3] -0,3454 * [att 2:x-2] -0,7118 * [att 3:x-1]
      -0,2359 * [att 4:y-3] + 1,0374 * [att 5:y-2] -0,8827 * [att 6:y-1] + 0,6138 * [att 7:c-3]
      + 0,5683 * [att 8:c-2] + 0,6109 * [att 9:c-1] + -0.0042448037757051625
  if [att 9:c-1] > 73.24996473188253:
    Leaf [class:c] = 0,2487 * [att 1:x-3] -0,3430 * [att 2:x-2] -0,7097 * [att 3:x-1]
    -0,2426 * [att 4:y-3] + 1,0301 * [att 5:y-2] -0,8905 * [att 6:y-1] + 0,6158 * [att 7:c-3]
    + 0,5700 * [att 8:c-2] + 0,6123 * [att 9:c-1] + 0.017072775891599987
if [att 9:c-1] > 482.7855034440994:
  if [att 9:c-1] <= 2071.8984977530863:
    Leaf [class:c] = 0,2460 * [att 1:x-3] -0,3460 * [att 2:x-2] -0,7128 * [att 3:x-1]
    -0,2450 * [att 4:y-3] + 1,0271 * [att 5:y-2] -0,8942 * [att 6:y-1] + 0,6127 * [att 7:c-3]
    + 0,5667 * [att 8:c-2] + 0,6087 * [att 9:c-1] + 0.022369597933943403
  if [att 9:c-1] > 2071.8984977530863:
    Leaf [class:c] = 0,2427 * [att 1:x-3] -0,3490 * [att 2:x-2] -0,7156 * [att 3:x-1]
    -0,2496 * [att 4:y-3] + 1,0224 * [att 5:y-2] -0,8989 * [att 6:y-1] + 0,6089 * [att 7:c-3]
    + 0,5631 * [att 8:c-2] + 0,6054 * [att 9:c-1] + 0.03651148213470908
```

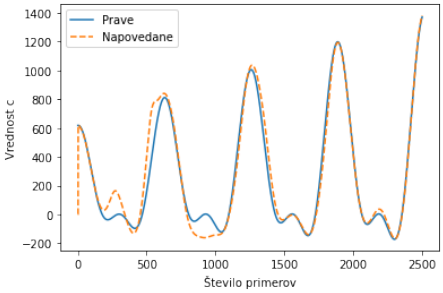
Slika 3: Drevo modela naučenega na razširjenem podatkovnem naboru

1.2 Graf ujemanja

Izrišimo si še ujemanje pravih in napovedanih vrednosti modela. Za napovedno množico vzamemo tisto, ki je natrenirana na razširjenem podatkovnem naboru. Vidimo, da se napovedane vrednosti grafično dobro ujemajo s pravimi vrednosti `c`. Poglejmo si še ujemanje na prvih 2500 primerih, kjer je povprečje absolutne napake največje. Kot lahko razberemo iz grafa povprečne absolutne napake, lahko vidimo, da se model pri prvih 2500 primerih še malce "lovi"in na parih mestih malo odstopa od pravih vrednosti.



Slika 4: Ujemanje vrednosti spremenljivke c



Slika 5: Ujemanje vrednosti spremenljivke c za prvih 2500 primerov

Druga naloga-klasifikacija

2.0 Priprava podatkovnega nabora

Podatkovni nabor tokrat pripravimo s spletnim pretvornikom iz `.csv` v `.arff` datoteko. Tam sem lahko tudi označil, ali so spremenljivke numerične ali nominalne. Za `y` sem označil, da je nominalna, ostale pa, da so numerične, saj v navodilu piše, da so te spremenljivke zvezne (`f4` in `f5` bi načeloma lahko tudi označil kot nominalne). Podatkovni nabor imamo shranjen v `dn2b.arff`.

2.1 Opažanje sprememb v naboru

Spremembe v podatkovnem naboru poskušamo analizirati z uporabo Page-Hinkleyevega testa. V `Python` uvozim knjižnico `river.drift.PageHinkley`, kateri nastavimo parameter minimalne spremembe ($\alpha = 0,9999$) in parameter dovoljene pojavnosti lažnih signalov ($\delta = 0,005$). Najprej si poglejmo, kje test zazna spremembe pri ciljni spremenljivki `y`.

```
ph = drift.PageHinkley(alpha=0.9999,delta=0.005)
x=0
for i, y in enumerate(data["y"]):
    _ = ph.update(y)
    if ph.drift_detected:
        x+=1
        print(f"Change detected at index {i}, input value: {y}")

print(x)
```

Slika 6: Python program za Page-Hinkleyev test

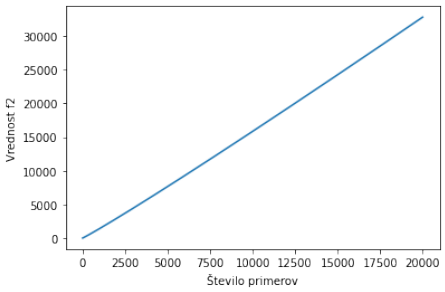
```
Change detected at index 9698, input value:  1
Change detected at index 15051, input value:  1
Change detected at index 17567, input value:  0
3
```

Spremenljivka `f1` ima vrednosti na intervalu $[-1, 1]$, test tu zazna naslednje spremembe:

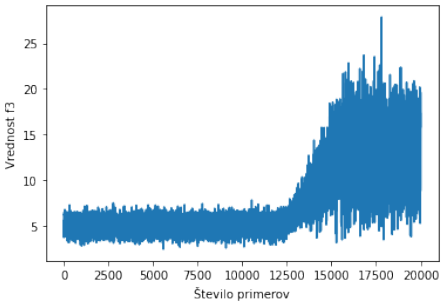
```
Change detected at index 8990, input value:  0.5586849269821885
Change detected at index 15445, input value:  0.7606470945539483
Change detected at index 17739, input value:  -0.7032297954539686
```

```
3
```

Pozicije sprememb vrednosti `f1` in `y` se dokaj ujemajo, tako da ocenimo, da so spremembe spremenljivke `f1` prave spremembe. Za drugo spremenljivko iz grafa ugotovimo, da se približno linearno dviga in tako bo tudi naš test zaznal spremembo na 30 primerov, saj je to prednastavljena vrednost število primerov, katerih spremembe se ignorira po določeni zaznani spremembi. Spremembo spremenljivke `f2` označimo kot navidezno koračno spremembo. Kot koračno spremembo označimo tudi spremembo spremenljivke `f3`, katere vrednost se začne zviševati nekje pri primeru 12500, kar nam pove test in je razvidno tudi iz grafa.



Slika 7: Graf vrednosti spremenljivke `f2`



Slika 8: Graf vrednosti spremenljivke `f3`

Za vrednost `f4` test ne zazna sprememb. Drugače je za zadnjo spremenljivko `f5`, kjer test zazna 59 sprememb, ki so na videz enakomerno razporejene po naboru. Če povečamo parameter δ, ki določa najmanjšo vrednost, za katero se prepozna sprememba, dobimo za $\delta = 0.3$ naslednje rezultate:

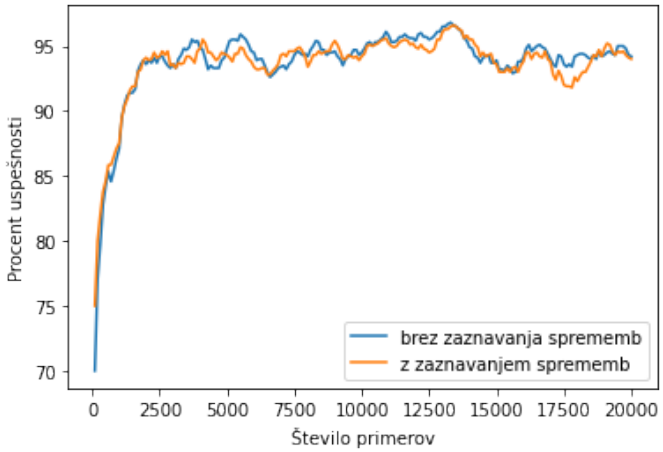
```
Change detected at index 8805, input value:  6
Change detected at index 14514, input value:  2
Change detected at index 17076, input value:  0
3
```

Opazimo, da so te spremembe blizu spremembam spremenljivk `y`. Iz tega razloga sklepamo, da so te spremembe prave.

2.2 Primerjava modelov

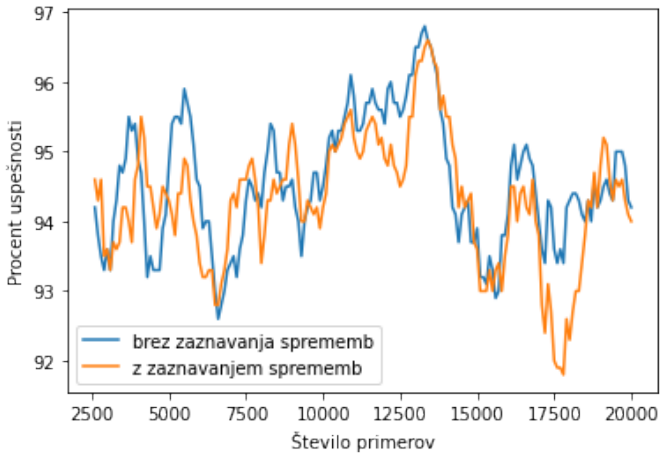
Naj bo model naučen z metodo za učenje Hoeffdingovih dreves z zaznavanjem sprememb model 1, in brez zaznavanja sprememb model 2. Za oba

modela ugotovimo, da dosežeta kar visoko natančnost, in sicer model 1 94% uspešnosti, model 2 pa 94, 2% uspešnosti. Model 2 brez zaznavanja sprememb je po pričakovanju malce hitrejši (0,45s) v primerjavi z modelom 1 (0,67s). Podobno ugotovimo, da ima model 2 manj listov (10 listov) kot model 1 (19 listov).



Slika 9: Graf procenta uspešnosti

Da bomo lažje opazovali vpliv sprememb v podatkovnem naboru, si izrišimo graf uspešnosti od 2500. primera dalje.



Slika 10: Graf procenta uspešnosti primerov od 2500 naprej

Na grafu si poglejmo zgolj, kaj se zgodi po 17 500 primerih, kjer je bila prej zaznana ena od sprememb. Vidimo, da se tam nekje uspešnost modela 1 začne krepko zviševati, vendar je v primerjavi z modelom 2 nižja uspešnost.