Treći domaći zadatak iz Mašinskog Učenja

–Univerzitet u Novom Sadu, Prirodno-Matematički fakultet–

Nikola Pujaz

–21m/19–

29. april 2020.

1 Zadatak - WEKA alat - *autos.arff*

Koristeći WEKA Experimenter možemo uspešno porediti različite algoritamske šeme ne bi li utvrdili koja daje najbolje rezultate na određenom skupu podataka. U ovom slučaju zadatak je bio da testiramo K - Nearest Neighbour klasifikator na autos.arff skupu podataka sa tri različite funkcije rastojanja:

1. Euklidsko rastojanje;
2. Manhattan rastojanje;
3. Chebyshev rastojanje.

Prethodno definisani eksperiment je dao sledeće rezultate:

1.1 Rangiranje algoritama

U Setup tabu WEKA alata, odabrana su tri K - Nearest Neighbour klasifikatora sa različitim funkcijama rastojanja. Za sva tri klasifikatora vrednost K = 1. Nakon pokretenja eksperimenta, izvršen je test, pri čemu je kao Test Base odabrana opcija za rangiranje algoritama i dobijeni su sledeći rezultati:

```
>-<   >   < Resultset
  1   1   0 lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
  1   1   0 lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance
 -2   0   2 lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ChebyshevDistance
```

Ranking Test rangira algoritme na osnovu ukupnog broja značajnih pogodaka (znak > u tabeli) i značajnih promašaja (znak < u tabeli) u odnosu na ostale algoritamske šeme. Prva kolona (>-<) predstavlja razliku između broja pogodaka i broja promašaja. Ova razlika se koristi za generisanje rangiranja. Iz priložene tabele možemo videti da su klasifikatori rangirani po sledećem redosledu, na osnovu funkcije rastojanja:

1. Klasifikator sa Manhattan rastojanjem kao funkcijom rastojanja;
2. Klasifikator sa Euklidskim rastojanjem kao funkcijom rastojanja;
3. Klasifikator sa Chebyshev rastojanjem kao funkcijom rastojanja;

Pri čemu možemo primetiti da se na osnovu razlike broja pogodaka i broja promašaja prva dva klasifikatora gotovo isto rangiraju, dok je poslednji klasifikator značajno lošiji od prethodna dva na osnovu rangiranja. Detalji tačnosti klasifikatora su opisani u nastavku.

1.2 Tačnost klasifikacije

Tabelom u nastavku je prikazana tačnost klasifikacije za sva tri prethodno opisana klasifikatora, pri čemu je za Test Base izabran klasifikator koji kao funkciju rastojanja koristi Manhattan rastojanje.

```
Dataset                   (2) lazy.IBk '-K 1 | (1) lazy.IBk '- (3) lazy.IBk '-
------------------------------------------------------------------------------
autos                   (100)   78.06(8.40) |   74.55(9.40)     58.52(8.93) *
------------------------------------------------------------------------------
                                    (v/ /*) |         (0/1/0)         (0/0/1)


Key:
(1) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
(3) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ChebyshevDistance
```

Ovi rezultati nam pokazuju da klasifikator sa Chebyshev rastojanjem kao funkcijom rastojanja ima značajno lošiju tačnost klasifikacije u odnosu na klasifikator sa Manhattan rastojanjem kao funkcijom rastojanja (Manhattan - 78.06%, Euclidean - 74.55%, Chebyshev - 58.52%). U slučaju u kom poredimo tačnost klasifikacije sa Manhattan i Euklidskim rastojanjem zbog približnih rezultata ne možemo doneti zaključak da li je bolji rezultat klasifikatora sa Manhattan rastojanjem statistički značajan u odnosu na klasifikator sa Euklidskim rastojanjem na 5% nivou statističkog značaja, što nam pokazuje i WEKA Experimenter u (x/y/z) vrednostima koji se nalaze u poslednjem redu tabele, koje označavaju koliko puta je taj klasifikator bio bolji (x), isti (y), ili lošiji (z) u odnosu na odabrani Test Base (u ovom slučaju klasifikator sa Manhattan rastojanjem kao funkcijom rastojanja). Ovo formira *null hypothesis* - nultu hipotezu sa kojom pretpostavljamo da je su ova dva klasifikatora jednaka po performansama. Uz pretpostavku da su dobijeni rezultati manje verovatni ako je nulta hipoteza istinita, možemo odbaciti nultu hipotezu na 5% nivou statističkog značaja i zaključiti da najbolje performanse nad ovim skupom podataka ima klasifikator koji ima Manhattan rastojanje kao funkciju rastojanja.

1.3 K-Nearest Neighbour sa Manhattan rastojanjem kao funkcijom rastojanja gde je K = 1,2,3,4,5

Klasifikator sa Manhattan rastojanjem kao funkcijom rastojanja se pokazao najboljim pod prethodno definisanim kriterijumima za koje je K = 1. U nastavku ćemo ispitati da li promena vrednosti K dovodi do poboljšanja tačnosti ovog klasifikatora. Vrednost K je pozitivan ceo broj koji u K - Nearest Neighbour klasifikaciji određuje broj najbližih suseda, koje uzimamo u obzir, instance skupa podataka koju pokušavamo da klasifikujemo. Posmatrajući klasu K - najbližih instanci instance koju klasifikujemo, određujemo klasu željene instance na osnovu dominantne klase među K - najbližim susedima. U slučaju u kom je K = 1, instancu klasifikujemo na osnovu saamo jednog najbližeg suseda, dodeljujući tu instancu klasi najbližeg suseda. Povećavanjem K vrednosti dolazi i do promene performansi klasifikatora. U ovom slučaju smo odabrali K vrednosti od 1 do 5 i dobili sledeće rezultate rangiranja:

```
>-<   >   < Resultset
  4   4   0 lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
 -1   0   1 lazy.IBk '-K 5 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
 -1   0   1 lazy.IBk '-K 4 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
 -1   0   1 lazy.IBk '-K 3 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
 -1   0   1 lazy.IBk '-K 2 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance
```

Primećujemo da se najbolje rangira klasifikator za koji je K = 1. Poređenjem tačnosti klasifikacije, gde je za Test Base stavljen klasifikator sa K = 1, primećujemo da su klasifikatori sa K vrednošću od 2 do 5 značajno lošiji od klasifikatora gde je K = 1 (K1 - 78.06%, K2 - 69.51%, K3 - 69.33%, K4 - 66.53%, K5 - 64.29%). Ukoliko bi testiranje izveli u odnosu na neki klasifikator sa K vrednošću od 2 do 5 videli bi da su performanse klasifikatora sa K vrednošću od 2 do 5 iste u 5% nivou statističkog značaja. Detalje tačnosti klasifikacije se mogu videti u sledećoj tabeli:

```
Dataset                   (1) lazy.IBk '-K 1 | (2) lazy.IBk '- (3) lazy.IBk '-K (4) lazy.IBk '-K (5) lazy.IBk '-K
-------------------------------------------------------------------------------------------------------------------
autos                    (100)   78.06(8.40) |   69.51(9.29) *   69.33(10.15) *   66.53(10.29) *   64.29(10.11) *
-------------------------------------------------------------------------------------------------------------------
                                        (v/ /*) |         (0/0/1)          (0/0/1)          (0/0/1)          (0/0/1)


Key:
(1) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance -R first-l
(2) lazy.IBk '-K 2 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance -R first-l
(3) lazy.IBk '-K 3 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance -R first-l
(4) lazy.IBk '-K 4 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance -R first-l
(5) lazy.IBk '-K 5 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.ManhattanDistance -R first-l
```

1.4 Ocena atributa po *information gain* i *gain ratio* meri

Koristićemo AttributeSelectedClassifier kako bi izvršili odabir atributa samo na osnovu trening skupa, a nakon toga istrenirali klasifikator i testirali ga sa podacima iz testnog skupa. Kao klasifikator smo odabrali K-Nearest Neighbour sa Manhattan rastojanjem kao funkcijom rastojanja, gde je K = 1. U prvom slučaju smo koristili Information Gain meru za evaluaciju atributa, a kao metodu pretrage smo odabrali rangiranje atributa, što se može videti u tabeli u nastavku:

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 26 symboling):
        Information Gain Ranking Filter

Ranked attributes:
 1.1896   10 length
 0.9211    2 make
 0.8889    9 wheel-base
 0.8158   12 height
 0.5233    1 normalized-losses
 0.4428   11 width
 0.4323    5 num-of-doors
 0.4201    6 body-style
 0.3591   17 fuel-system
 0.3164   16 engine-size
 0.2935   21 horsepower
 0.2737   13 curb-weight
 0.2689   25 price
 0.2688   14 engine-type
 0.2247   18 bore
 0.2033   24 highway-mpg
 0.1866   23 city-mpg
 0.1724   15 num-of-cylinders
 0.1492    7 drive-wheels
 0.1268   22 peak-rpm
 0.0626    3 fuel-type
 0.0439    8 engine-location
 0.041     4 aspiration
 0        20 compression-ratio
 0        19 stroke
```

U drugom slučaju smo koristili Gain Ratio meru za evaluaciju atributa, a kao metodu pretrage smo odabrali rangiranje atributa, što se može videti u tabeli u nastavku:

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 26 symboling):
        Gain Ratio feature evaluator

Ranked attributes:
 0.4371    5 num-of-doors
 0.3982    8 engine-location
 0.3569    9 wheel-base
 0.345    10 length
 0.331    12 height
 0.3161   13 curb-weight
 0.3002   16 engine-size
 0.287    25 price
 0.2501   11 width
 0.2409    6 body-style
 0.2352   23 city-mpg
 0.2268   18 bore
 0.2237    2 make
 0.2133   21 horsepower
 0.2052   24 highway-mpg
 0.2033    1 normalized-losses
 0.1834   17 fuel-system
 0.1798   14 engine-type
 0.145    15 num-of-cylinders
 0.1371   22 peak-rpm
 0.1357    3 fuel-type
 0.1263    7 drive-wheels
 0.0602    4 aspiration
 0        19 stroke
 0        20 compression-ratio
```

Najdiskriminativniji atribut po Information gain meri je *length*, a po Gain Ratio meri je *num-of-doors*. Nakon ove selekcije atributa klasifikator daje iste rezultate bez obzira na primenjenu meru, što možemo videti u nastavku:

```
Correctly Classified Instances         164               80      %
Incorrectly Classified Instances        41               20      %
Kappa statistic                          0.7422
Mean absolute error                      0.064
Root mean squared error                  0.235
Relative absolute error                 28.9508 %
Root relative squared error             70.8004 %
Total Number of Instances              205


   a  b  c  d  e  f  g   <-- classified as
   0  0  0  0  0  0  0 |  a = -3
   0  2  1  0  0  0  0 |  b = -2
   0  2 17  2  1  0  0 |  c = -1
   0  1  2 56  2  5  1 |  d = 0
   0  0  1  4 42  1  6 |  e = 1
   0  0  0  1  5 25  1 |  f = 2
   0  0  0  0  3  2 22 |  g = 3


Correctly Classified Instances         164               80      %
Incorrectly Classified Instances        41               20      %
Kappa statistic                          0.7422
Mean absolute error                      0.064
Root mean squared error                  0.235
Relative absolute error                 28.9508 %
Root relative squared error             70.8004 %
Total Number of Instances              205
```

```
  a  b  c  d  e  f  g   <-- classified as
  0  0  0  0  0  0  0 |  a = -3
  0  2  1  0  0  0  0 |  b = -2
  0  2 17  2  1  0  0 |  c = -1
  0  1  2 56  2  5  1 |  d = 0
  0  0  1  4 42  1  6 |  e = 1
  0  0  0  1  5 25  1 |  f = 2
  0  0  0  0  3  2 22 |  g = 3
```

1.5 AttributeSelection filter

AttributeSelection filter nam omogućava da izvršimo odabir atributa u fazi pretprocesiranja, u zavisnosti od odabrane *evaluator* i *search* funkcije. Primenom ovog flitera možemo izvršiti odabir bitnih atributa za klasifikaciju na osnovu željene mere pre samog treniranja i testiranja klasifikatora, a rezultat filtriranja može da utiče na rezultate klasifikacije, što se može videti i u nastavku:

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.ManhattanDistance -R first-last\""
Relation:     autos-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSelection.CfsSubsetEval -P 1 -E 1-Sweka.attributeSelection.BestFirst -D 1 -N 5
Instances:    205
Attributes:   6
              normalized-losses
              num-of-doors
              wheel-base
              length
              height
              symboling
Test mode:    10-fold cross-validation
```

Od 26 atributa, filtriranjem je izvodjeno 6 bitnih atributa. Primenom K-Nearest Neighbour sa Manhattan rastojanjem kao funkcijom rastojanja, gde je K = 1, pre filtriranja, dobijeni su sledeći rezultati:

```
Correctly Classified Instances         164               80      %
Incorrectly Classified Instances        41               20      %
Kappa statistic                          0.7422
Mean absolute error                      0.064
Root mean squared error                  0.235
Relative absolute error                 28.9508 %
Root relative squared error             70.8004 %
Total Number of Instances              205
```

```
  a  b  c  d  e  f  g   <-- classified as
  0  0  0  0  0  0  0 |  a = -3
  0  2  1  0  0  0  0 |  b = -2
  0  2 17  2  1  0  0 |  c = -1
  0  1  2 56  2  5  1 |  d = 0
  0  0  1  4 42  1  6 |  e = 1
  0  0  0  1  5 25  1 |  f = 2
  0  0  0  0  3  2 22 |  g = 3
```

Nakon primene filtera je zabeleženo poboljšanje:

```
Correctly Classified Instances         170               82.9268 %
Incorrectly Classified Instances        35               17.0732 %
Kappa statistic                          0.781
Mean absolute error                      0.0535
Root mean squared error                  0.2181
Relative absolute error                 24.1941 %
Root relative squared error             65.7132 %
Total Number of Instances              205
```

```
   a   b   c   d   e   f   g   <-- classified as
   0   0   0   0   0   0   0 |  a = -3
   0   3   0   0   0   0   0 |  b = -2
   0   1  19   0   2   0   0 |  c = -1
   0   0   1  51   5   7   3 |  d = 0
   0   0   0   3  46   2   3 |  e = 1
   0   0   0   1   1  28   2 |  f = 2
   0   0   0   0   3   1  23 |  g = 3
```

2 Zadatak - WEKA API - *nursery.arff*

Detalji implementacije se vide iz priloženog Java projekta. Rezultati dobijeni primenom SVM sa *10-fold cross-validation* su sledeći:

```
SVM (10-fold cross-validation) Results:
Correctly Classified Instances        7980               93.3006 %
Incorrectly Classified Instances       573                6.6994 %
Kappa statistic                          0.9017
Mean absolute error                      0.2427
Root mean squared error                  0.3203
Relative absolute error                 88.9284 %
Root relative squared error             86.7122 %
Total Number of Instances             8553

>>>> Overall Confusion Matrix <<<<
    a    b    c    d    e   <-- classified as
 2821    0    0    0    0 |    a = not_recom
    0    0    2    0    0 |    b = recommend
    0    0  152   57    0 |    c = very_recom
    0    0   29 2519  272 |    d = priority
    0    0    0  213 2488 |    e = spec_prior
```

Rezultati dobijeni primenom SVM sa test skupom dobijenim *percentage split* opcijom (66/34%) su sledeći:

```
SVM (Test Set Evaluation) Results:
Correctly Classified Instances        4088               92.7615 %
Incorrectly Classified Instances       319                7.2385 %
Kappa statistic                          0.8939
Mean absolute error                      0.2429
Root mean squared error                  0.3205
Relative absolute error                 88.9332 %
Root relative squared error             86.6955 %
Total Number of Instances             4407

>>>> Overall Confusion Matrix <<<<
    a    b    c    d    e   <-- classified as
 1499    0    0    0    0 |    a = not_recom
    0    0    0    0    0 |    b = recommend
    0    0   87   32    0 |    c = very_recom
    0    0   22 1281  143 |    d = priority
    0    0    0  122 1221 |    e = spec_prior
```

Rezultati dobijeni primenom J48 sa *10-fold cross-validation* su sledeći:

```
J48 (10-fold cross-validation) Results:
Correctly Classified Instances        8233               96.2586 %
Incorrectly Classified Instances       320                3.7414 %
Kappa statistic                          0.945
Mean absolute error                      0.0201
Root mean squared error                  0.1102
Relative absolute error                  7.3598 %
Root relative squared error             29.8425 %
Total Number of Instances             8553

>>>> Overall Confusion Matrix <<<<
    a    b    c    d    e   <-- classified as
 2821    0    0    0    0 |    a = not_recom
    0    0    2    0    0 |    b = recommend
    0    0  113   96    0 |    c = very_recom
    0    0   40 2647  133 |    d = priority
    0    0    0   49 2652 |    e = spec_prior
```

Rezultati dobijeni primenom J48 sa test skupom dobijenim *percentage split* opcijom (66/34%) su sledeći:

```
J48 (Test Set Evaluation) Results:
Correctly Classified Instances        4226               95.8929 %
Incorrectly Classified Instances       181                4.1071 %
Kappa statistic                          0.9399
Mean absolute error                      0.0205
Root mean squared error                  0.1123
Relative absolute error                  7.5119 %
Root relative squared error             30.3833 %
Total Number of Instances             4407

>>>> Overall Confusion Matrix <<<<

    a    b    c    d    e   <-- classified as
 1499    0    0    0    0 |    a = not_recom
    0    0    0    0    0 |    b = recommend
    0    0   85   34    0 |    c = very_recom
    0    0   34 1335   77 |    d = priority
    0    0    0   36 1307 |    e = spec_prior
```

Iz prethodno priloženog možemo primetiti za nijansu bolje rezultate i kod SVM i kod J48 klasifikatora u slučaju korišćenja *10-fold cross-validation* opcije koristeći 66% originalnog skupa kao test skup, nego što je to slučaj sa *percentage split* opcijom.

Metodom *classifyInstances* u odnosu na korisnikov odabir datoteke i klasifikatora, vrši se klasifikacija na osnovu već istreniranih modela.

Metodom *graph* iz WEKA API se generiše graf stabla odlučivanja koji može grafički biti prikazan uz pomoć neke UI biblioteke. U ovom slučaju ovaj graf je ispisan u konzoli.