Systemy uczące się
Zadanie 1

Michał Jaroń
mj348711

3 kwietnia 2017

1 Wstępna analiza danych

Przed przystąpieniem do budowania klasyfikatorów zbadałem korelacje pomiędzy binarnymi wartościami oznaczającymi decyzje i poszczególnymi parametrami. Poniżej przedstawiam współczynniki korelacji otrzymane metodą Pearsona. Metody Kendalla oraz Spearmana dają podobne wyniki.

Parametr	Korelacja z decyzją
C1	-0.35193432
C2	-0.10323135
C3	-0.23096990
C4	-0.25713703
C5	-0.06920008
C6	-0.03756056
C7	-0.14201474
C8	-0.06143349
C9	-0.23452781
C10	-0.14630357

Na podstawie tych wartości można stwierdzić, że wartość decyzji nie jest w dużym stopniu zależna od żadnego z parametrów. (maksymalna wartość - 0.35 to nadal mała korelacja, w porównaniu do korelacji uzyskiwanych dla danych wdbc, rzędu 0.80). Co więcej mamy niemal równy rozkład wartości dla wartości decyzji. Tzn. żadna odpowiedź nie występuje częściej. To wszystko sprawia, że ciężko zastosować heurystyki w celu wstępnej obróbki danych, musimy się oprzeć na ulepszaniu samych klasyfikatorów.

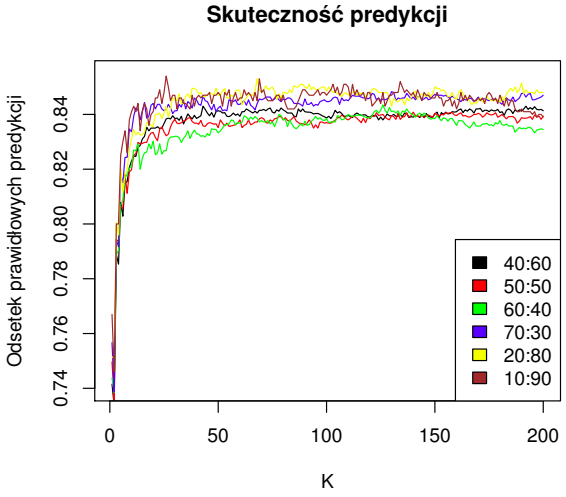
Dane z kolumn nie będących decyzją, mają w przybliżeniu rozkład normalny, z bardzo wyraźnym pikiem dla wartości "średnich"

2 kNN

Pierwszym rozważanym przez mnie klasyfikatorem jest algorytm k-nearest neighbours. Testy przeprowadziłem w środowisku R, używając funkcji knn, z biblioteki class. W celu wykazania różnicy między dwoma metodami oceny klasyfikatora badałem błędy używając sztywnego podziału na dwa zbiory: testujący

i treningowy (testy dla różnych proporcji tych zbiorów), a następnie przeprowadziłem testy metodą walidacji krzyżowej.

Jak wiadomo, kluczowym czynnikiem decydującym o skuteczności kNN jest dobór k, czyli liczby rozważanych sąsiadów, dlatego zbadałem skuteczność algorytmu dla różnych k, poniżej przedstawiam wykres dla różnych proporcji podziału na zbiory testujący/treningowe, oraz kolejnych k.



Analiza wykresu Jest jasne, że im większy zbiór treningowy, tym większa skuteczność klasyfikatora. Przeznaczając jednak zbyt duża ilość danych do treningu tracimy dokładność w szacowaniu błędów, jakie popełnia nasz klasyfikator. Większa ilość danych przeznaczona do treningu daje lepszą skuteczność. Na wykresie widać wskazaną prawidłowość, wykresy przedstawiające skuteczność dla małych rozmiarów zbiorów testowych, znajdują się nad wykresami, przedstawiającymi klasyfikatory karmione mniejszą ilością danych treningowych(kolory żółty, niebieski, brązowy górują nad pozostałymi wykresami). Oczywiście testowanie ma w sobie czynnik losowości, ponieważ za każdym razem losuje nowy podział na zbiory. Widać to dla klasyfikatorów, zbudowanych na prawie równych zbiorach, klasyfikator (kolor czarny), klasyfikuje opierając się na próbie wielkości 40 procent z całego zbioru danych, a daje lepszą skuteczność od prób wielkości 50:50 procent. Oczywiście, te skuteczności są bardzo do siebie zbliżone, a różnice są wręcz pomijalne. Bardziej wiarygodne oszacowanie da dopiero ocena walidatorem krzyżowym.

Dobór K Analizując zachowanie klasyfikatora w zależności od parametru K, widać gwałtowny przyrost skuteczności dla małych K. W okolicach k = 39, mamy już małe wahania skuteczności algorytmu. Nie jest to specjalnie zadziwiające. Przy 10 parametrach, które tworzą 10-wymiarową przestrzeń w algorytmie kNN, małe wartości K, okazują się nieskuteczne. K dające maksymalną skuteczność klasyfikatora różni się w zależności od podziału zbioru, jednak prawie maksymalną skuteczność można już osiągnąć dla K = 40, potem przyrosty są marginalne, a koszt algorytmu dla większej ilości K rośnie. Jednocześnie zbyt

duża wartość K, może dawać dobre wyniki dla danych testowych, ale może okazać się algorytmem zbyt mało "generalizującym" i w konsekwencji dawać gorsze wyniki dla innych danych.

Poniżej przedstawiam K, dla których osiągnięto maksymalną skuteczność w zależności od wielkości zbioru treningowego.

trening:test	K
40:60	43
50:50	140
60:40	126
30:70	130
20:80	68
10:90	26

Po ustaleniu, że powyżej pewnej wartości K, nie mamy dużych zmian w skuteczności, przedstawiam skuteczność klasyfikacji, tylko dla K = 50, oraz skupię się na podziałach zbioru danych w proporcjach 60:40, 70:30, 80:20 oraz 10:90 (trening:test).

Skuteczność klasyfikatora

	60:40	70:30	80:20	10:90
Mean	0.83325	0.8423333	0.847	0.848
BAC	0.8333178	0.8422715	0.8469836	0.8487981

Udoskonalenia Po przeprowadzonej analizie, podjąłem próbę ulepszenia klasyfikatora. Mianowicie, wybrałem podzbiór 3 najbardziej skorelowanych z wynikiem parametrów. Podobnie jak dla danych analizowanych na zajęciach, skuteczność takiego klasyfikatora jest mniejsza niż w przypadku predykcji dla całego zbioru parametrów i wynosi tylko około 73 procent (dla podziału danych w proporcji 80:20 (trening:test)).
Zdecydowanie lepszy efekt uzyskałem, łącząc odpowiedzi wszystkich klasyfikatorów dla k z przedziału 1-200. Wyjaśniając: odpowiedzi klasyfikatorów dla każdego K z przedziału 1-200 zsumowałem i jeżeli mniej niż 180 klasyfikatorów dało jednoznaczną odpowiedź, to wtedy udzielałem końcowej odpowiedzi "nie wiadomo". Dla losowego zbioru testowego o rozmiarze 2000 wierszy 201 razy wygenerowałem odpowiedź "nie wiadomo", jednakże, w przypadku odpowiedzi, kiedy ponad 180 klasyfikatorów dało taką samą odpowiedź, uzyskana skuteczność to aż 87 procent. Okazuję się, więc, że takie "głosowanie" kiedy jest naprawdę większościowe udziela poprawnej odpowiedzi z bardzo duża skutecznością, jednocześnie mało przypadków odrzuca, z powodu "niezdecydowania".

Walidacja krzyżowa Na koniec przeprowadziłem ocenę klasyfikatora dla K=50, metodą walidacji krzyżowej, z podziałem na 10 zbiorów. Przedstawiam szacowania skuteczności dla kolejnych zbiorów, będących testowymi:

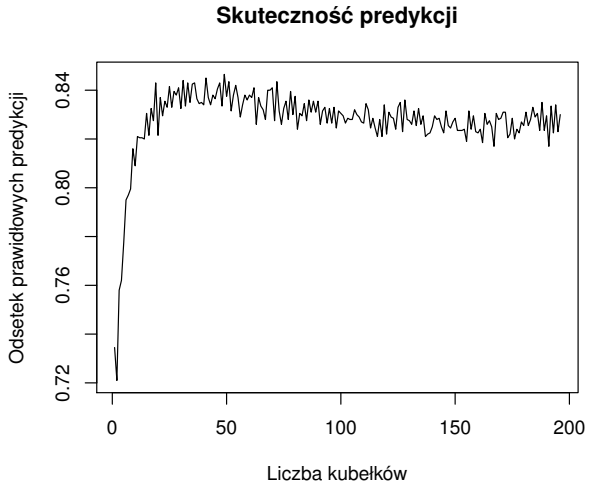
Iteracja	Mean	BAC
1	0.877	0.8771826
2	0.828	0.8280223
3	0.835	0.8359496
4	0.856	0.856378
5	0.829	0.8293947
6	0.842	0.8418572
7	0.844	0.8411062
8	0.85	0.8504542
9	0.823	0.8225063
10	0.851	0.8510134

Średnia skuteczność obliczona na podstawie iteracji walidacji krzyżowej, to 0.8435, a więc skuteczność podobna jak przy ocenie klasyfikatora, metodą sztywnego podziału na zbiory testowy i treningowy. Widzimy jednak, jakie znacznie ma zbiór, na którym testujemy, dla niektórych zbiorów, skuteczność to prawie 86 procent, co daje 2 procentową różnicę, między zbiorami gdzie ta skuteczność jest najmniejsza.

Wnioski i pomysły Kończąc analizę kNN warto wspomnieć, że użyty przeze mnie wariant kNN, to tylko jedna z opcji. Możemy badać zachowanie klasyfikatora dla innych metryk lub obliczać "głosowanie" stosując średnią ważoną, tzn. głos punktów bliższych, jest ważniejszy. Być może któreś z udoskonaleń da jeszcze lepszą skuteczność, skoro jednak raport miał być krótki, pozostanę przy tradycyjnej wersji kNN. Podsumowując, udało uzyskać się dobrą skuteczność klasyfikatora, oraz potwierdzić to "dobre zachowanie" kilkoma metodami oceny klasyfikatora. kNN choć bardzo prosty w swoim działaniu, to jednak klasyfikator skuteczny. Na podstawie analizy, mogę stwierdzić, że warto używać go do budowania modeli, ponieważ nie wymaga dużych nakładów "technicznych" oraz jest łatwy w analizie, a i poprawianie skuteczności (nauka = dobór K), jest szybka i przejrzysta. Warto również odnotować, że kNN ma wiele implementacji w najpopularniejszych środowiskach. W mojej analizie wybrałem język R, ponieważ chciałem w pełni kontrolować to, co dzieje się "pod spodem".

3 Naive Bayes

Klasyfikator opierający się na prawdopodobieństwie również badałem w R(w zadaniu można wysłać tylko jeden plik raport, tak więc kodów nie załączam, mogę je oczywiście pokazać). Tym razem jednak nie korzystałem z żadnych gotowych bibliotek i samodzielnie napisałem klasyfikator Bayesa, zgodnie z opisem ze slajdów. W przeciwieństwie do kNN konstrukcja Naive Bayes wymaga dyskretyzacji danych. Tak więc pierwszym etapem przed badaniem zachowania klasyfikatora, był podział danych na kubełki. Każdą kolumnę danych dzielę na kubełki w sposób niezależny. Jedynym miejsce, w którym możemy mówić o nauce klasyfikatora, jest dobór liczby kubełków. Faktycznie liczba kubełków ma kluczowe znaczenie dla skuteczności Bayesa jednak podobnie jak dla kNN, powyżej pewnej liczby kubełków klasyfikator nie poprawia już swojej skuteczności. W celu znalezienia najlepszych punktów cięć badałem skuteczność klasyfikatora, w zależności od liczby cięć. Poniżej przedstawiam wykres.



Jak widać od liczby kubełków równej 15 klasyfikator osiąga już niemal maksymalnąskuteczność i wraz ze wzrostem liczby podziałów, nie zmienia już w sposób drastyczny swojej skuteczności. Powyższy wykres przedstawia skuteczność klasyfikator przy podziale danych na zbiory treningowy i testowy w proporcji 80:20. Nauczony doświadczeniem kNN, pozwolę już sobie nie badać innych proporcji podziału danych, różnicę powinny być marginalne. Skuteczność klasyfikatora potwierdza walidacja krzyżowa, z podziałem na 10 zbiorów. Średnia skuteczność to 0.8275893.

Ulepszenia Jako że dane w kolejnych kolumnach mają w przybliżeniu rozkład normalny, metoda kubełkowanie z przedziałami równej długości, niekoniecznie musi okazać się skuteczna. Siłą rzeczy, większość obserwacji zostanie przypisanych do kubełków ze "średnimi" wartościami. Dlatego jedną z metod poprawiania skuteczności może być inny dobór cięć. W swoich testach próbowałem wyznaczać kubełki częściej w "środkowej" 1/3 przedziału, niestety taka metoda uzyskiwała, tylko 54 procent skuteczności, czyli właściwie tyle co klasyfikator losowy.

Pomysły i wnioski Naiwny klasyfikator Bayesa daje bardzo dobre wyniki. Porównując z przaanalizowanym już kNN, wykazuje równie wysoką skuteczność. W mojej opinii Bayes jest jeszcze prostszy ideowo od kNN, jednak przy podziale na większą liczbę kubełków, może okazać się algorytmem zbyt kosztownym czasowo. Oczywiście konieczność dyskretyzacji danych powoduje, że tracimy jakąś informację. Jednocześnie jednak taka generalizacja może okazać się bardzo skuteczna, gdyż pomija nieważne szczegóły. Myślę, że Bayesa warto używać szczególnie dla danych, których parametry pochodzą z dyskretnej dziedziny, w końcu sama dyskretyzacja zajmuje sporo czasu, a i nie do końca wiemy czy nasz podział na kubełki jest "zgodny" z naturą danych. Naive Bayes może być implementowany z kilkoma ulepszeniami. Możemy spróbować, podobnie jak zrobiłem dla kNN, czasem zwracać odpowiedź NIE WIADOMO. Jeżeli wyliczone prawdopodobieństwo zarówno dla 0 jak i dla 1 jest bardzo małe, możemy wtedy

wstrzymywać się od decyzji, zapewne skuteczność klasyfikacji w zbiorze danych, na które udzielimy odpowiedzi, będzie wyższa.

4 Drzewo decyzyjne

Badanie drzew decyzyjnych przeprowadziłem tym razem nie tylko w R, ale również w WEKA.

Weka Pierwszym krokiem była dyskretyzacja danych filtrami Discretize, a następnie Numeric to Nominal. Przeanalizowałem zachowanie klasyfikatora w zależności od liczby kubełków wybranych w Discretize(badałem 10, 20, 40, 80, 150). Samej klasyfikacji dokonałem używając klasyfikatora J48, z parametrami batch =100, confidencefactor = 0.25. Poniżej przedstawiam wyniki klasyfikacji dla 10 kubełków, gdzie do oceny klasyfikacji użyto walidacji krzyżowej z podziałem na 10 zbiorów. Zbudowane drzewo składa się z 710 liści.

```
Correctly Classified Instances        7584    75.84 %
Incorrectly Classified Instances      2416    24.16 %

=== Confusion Matrix ===

    a     b   <-- classified as
 3791  1211 |     a = 0
 1205  3793 |     b = 1
```

Zmiana parametrów J48 nie przyczyniała się do poprawienia skuteczności. Co najwyżej rozmiar drzewa był ograniczany (lepsze przycinanie). Odbywało się to jednak przy skuteczności na poziomie wyżej przedstawionej analizy(a czasem i niższej).

Poniżej skutecznośc klasyfikatorów w zależności od liczby kubełków(walidacja krzyżowa 10 zbiorów):

Liczba kubełków	10	20	40	80	150
Skuteczność	75.84	73.55	70.37	67.76	65.34
Liczba liści	703	1217	1093	1028	895

W przeciwieństwie do Bayesa większa liczba kubełków znacznie pogarsza skuteczność klasyfikatora. Wynika to z innej idei na jakiej opierają się drzewa decyzyjne. Zaskakująca jest również maksymalna skuteczność, jaką udało mi się uzyskać. 75 procent to aż o 10 procent gorsza skuteczność w porównaniu do wcześniej przedstawionych algorytmów. Może wynika to z "magii" jaką robi WEKA?

Rpart Rpart to biblioteka dostarczająca gotową implementacje drzew decyzyjnych w środowisku R. Niestety jest dosyć nieintujcyjna i awaryjna(na podstawie własnych doświadczeń, być może popełniam "gdzieś" błąd). Dlatego zostawiam to tylko jako ciekawostkę "riserczową".

5 Testy WEKA

Na koniec postanowiłem skonfrontować moje analizy dwóch pierwszych klasyfikatorów, przeprowadzone w R, z wynikami uzysknymi w WEKA.

Bayes Dyskretyzacji danych dokonałem tak samo jak w przypadku kubełkowanie danych dla drzew decyzyjnych. Przy podziale na 30 kubełków i wybraniu klasyfikatora Naive Bayes uzyskana skuteczność 83.21 procent(walidacja krzyżowa 10 zbiorów), dla 70 kubełków to 80.97 procent, a dla 10 kubełków 80.23 procent. Jak widzimy uzyskałem podobne wyniki jak dla własnej implementacji.

kNN kNN w WEKA przetestowałem dla danych skubełkowanych oraz czystych. Dla podziału zbioru na 20 kubełków i dla k = 10, uzyskana skutecznośc to: 69.09 procent, z kolei dla k = 40 (dla takich parametrów w R uzyskiwałem najlepsze wyniki) skuteczność to tylko 70.13 procent. Dla danych dyskretnych skuteczność jest niezadawalająca. Spróbujmy, tak jak w mojej orginalnej analizie dla danych "ciągłych"

Niestety dla danych ciągłych WEKA zachowuje się "dziwnie", nie zwraca modeli w pożądanej formie. Na szczęście główną analizę przeprowadziłem na swojej implementacji, więc nie wnikam głębiej w działanie WEKA. Co warte odnotowanie kNN w WEKA to klasyfikator potrzebujący najdłużej czasu na naukę spośród wszystkich zbadanych.

6 Podsumowanie

Tworzenie raportu pozwoliło mi zdobyć cenne doświadczenie, szczególnie, że postanowiłem wybrać własne implementacje klasyfikatorów. Uzyskane skuteczności, na poziomie powyżej 80 procent są zadawalające, szczególnie, że użyte metody są bardzo proste. Być może inne, bardziej zaawansowane metody ML dadzą lepsze wyniki, myślę, jednak, że zawsze warto zaczynać analizę od najprostszych klasyfikatorów. Pomysłem na "przyszłość" jest również połącznie odpowiedzi 3 zbadanych klasyfikatorów i zwrócenie wyniku "głosowania".