

Sprawozdanie IV

Reprezentacja niepewności

Prawdopodobieństwo warunkowe – proste przeliczenie

Szacuje się, że 0,05% populacji USA ma HIV. Istnieje test na HIV: (a) jeśli badany ma HIV, test ma 98% szansy na pozytywny wynik; (b) jeżeli osoba nie ma HIV, test ma 3% szansy na pozytywny wynik. Tomek ma wynik pozytywny. Jakie jest prawdopodobieństwo, że ma HIV?

$$\begin{aligned}
 P(HIV) &= 0,05 \\
 P(T_{pos} | HIV) &= 0,98 \\
 P(F_{pos} | !HIV) &= 0,03 \\
 P(Tomek\ HIV | Pos) &= \frac{P(Pos | Tomek\ HIV) \cdot P(Tomek\ HIV)}{P(Pos)} = \\
 &= \frac{P(T_{pos} | HIV) \cdot P(HIV)}{P(Pos | Tomek\ HIV) \cdot P(Tomek\ HIV) + P(Pos | !Tomek\ HIV) \cdot P(!Tomek\ HIV)} = \\
 &= \frac{0,98 \cdot 0,05}{0,98 \cdot 0,05 + 0,03 \cdot 0,95} \approx 0,63
 \end{aligned}$$

Klasyfikator Naive Bayes - ręcznie

Rozważmy taki prosty zbiór treningowy, w którym każdy przykład ma cztery binarne atrybuty i przydzieloną jedną z dwóch klas (+/-):

Przykład	Atrybut_1	Atrybut_2	Atrybut_3	Atrybut_4	Klasa
x1	1	1	1	1	+
x2	1	1	0	1	+
x3	0	1	1	0	+
x4	1	0	0	1	+
x5	1	0	0	0	+
x6	1	0	1	0	-
x7	0	1	0	0	-
x8	0	0	1	0	-

W jaki sposób naiwny klasyfikator Bayesowski, wyuczony na powyższym zbiorze treningowym, zaklasyfikuje poniższy przykład? Policz ręcznie :)

Przykład	Atrybut_1	Atrybut_2	Atrybut_3	Atrybut_4	Klasa
x9	1	1	0	0	-

	a_1	a_2	a_3	a_4	c
x_1	1	1	1	1	+
x_2	1	1	0	1	+
x_3	0	1	1	0	+
x_4	1	0	0	1	+
x_5	1	0	0	0	+
x_6	1	0	1	0	-
x_7	0	1	0	0	-
x_8	0	0	1	0	-
x_9	1	1	0	0	?

$$P(+|a_i) = P(+). \prod_i P(a_i|+) =$$

$$= \frac{5}{8} \cdot (P(a_1|+) \cdot P(a_2|+) \cdot P(a_3|+) \cdot P(a_4|+)) =$$

$$= \frac{5}{8} \cdot \left(\frac{4}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{2}{5} \right) = \frac{9}{125}$$

$$P(-) \cdot \prod_i (P(a_i|-)) = \frac{3}{5} \cdot (P(a_1|-) \cdot P(a_2|-) \cdot P(a_3|-) \cdot P(a_4|-))$$

$$= \frac{3}{5} \cdot \left(\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot 1 \right) = \frac{1}{45}$$

Zastosowanie klasyfikatora Naive Bayes w Wece

1. Włącz Wekę i wczytaj plik `weather.numeric.arff` ze znanego Ci już zbioru danych: data.tar.gz
2. Przejrzyj ten zbiór danych i przypomnij sobie czego on dotyczy.
3. Przejdź na zakładkę **Classify**. Weka udostępnia dwie wersje Naive Bayes. Zapoznaj się z ich opisami:
 - NaiveBayes
 - NaiveBayesUpdateable
4. Przetestuj obydwie wersje algorytmu? Jakie są różnice? Podczas testowania nie zauważyłem żadnych różnic w wartościach wyjściowych. W dokumentacji tak samo jest nie wiele napisane na temat tych metod i nie można wywnioskować jakie są różnice. Jedynie jest wspomniane o tym, że jest to wersja updateable, zatem mogę przypuszczać, że w trakcie predykcji po podaniu prawdziwej odpowiedzi metoda może zaktualizować swoje parametry
5. Następnie przetestuj te same algorytmy na pliku `weather.nominal.arff`. Jakie różnice występują teraz? Co jest przyczyną występowania różnic - odpowiedz korzystając z opisów algorytmów.

The screenshot shows the Weka software interface. On the left, the 'Test options' panel has 'Use training set' selected. Below it, the 'Result list' shows two entries: '11:21:39 - bayes.NaiveBayesUpdateable' and '11:21:44 - bayes.NaiveBayes'. The main 'Classifier output' panel displays the following data:

```
sunny      3.0    4.0
overcast   5.0    1.0
rainy      4.0    3.0
[total]    12.0   8.0

temperature
mean       72.9697 74.8364
std. dev.  5.2304  7.384
weight sum 9      5
precision  1.9091  1.9091

humidity
mean       78.8395 86.1111
std. dev.  9.8023  9.2424
weight sum 9      5
precision  3.4444  3.4444

windy
TRUE       4.0    4.0
FALSE      7.0    3.0
[total]    11.0   7.0
```

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances	13	92.8571 %
Incorrectly Classified Instances	1	7.1429 %
Kappa statistic	0.8372	
Mean absolute error	0.2798	
Root mean squared error	0.3315	
Relative absolute error	60.2576 %	
Root relative squared error	69.1352 %	
Total Number of Instances	14	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.200	0.900	1.000	0.947	0.849	0.911	0.947	yes
	0.800	0.000	1.000	0.800	0.889	0.849	0.911	0.911	no
Weighted Avg.	0.929	0.129	0.936	0.929	0.926	0.849	0.911	0.934	

=== Confusion Matrix ===

```
a b  <-- classified as
9 0 | a = yes
1 4 | b = no
```

The screenshot shows the Weka GUI with the 'Classifier output' tab selected. The 'Test options' section on the left shows 'Use training set' selected. The 'Classifier output' section on the right displays the results of a Naive Bayes classifier.

Test options:

- ☒ Use training set
- ☐ Supplied test set (Set...)
- ☐ Cross-validation (Folds: 5)
- ☐ Percentage split (%: 66)
- More options...

Classifier output:

(Nom) play

Start Stop

Result list (right-click for options)

- 11:21:39 - bayes.NaiveBayesUpdateable
- 11:21:44 - bayes.NaiveBayes

Classifier output details:

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	13	92.8571 %
Incorrectly Classified Instances	1	7.1429 %
Kappa statistic	0.8372	
Mean absolute error	0.2798	
Root mean squared error	0.3315	
Relative absolute error	60.2576 %	
Root relative squared error	69.1352 %	
Total Number of Instances	14	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0.929	0.129	0.936	0.929	0.926	0.849	0.911	0.934	yes
	1.000	0.200	0.900	1.000	0.947	0.849	0.911	0.947	no

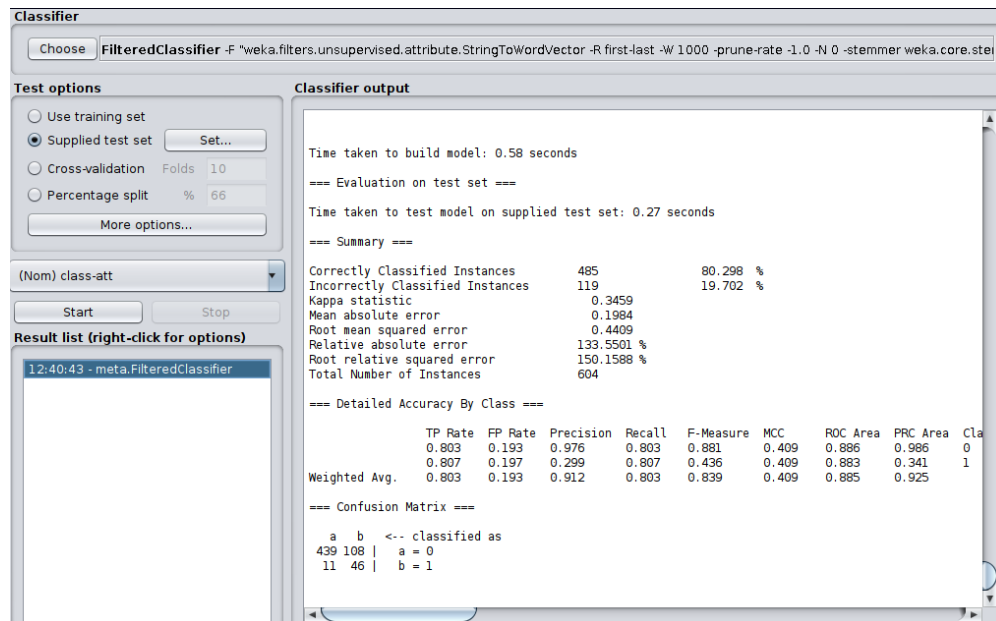
=== Confusion Matrix ===

```
a b  <-- classified as
9 0 | a = yes
1 4 | b = no
```

Multinomial Naive Bayes w Wece

1. Wczytaj w Wece plik ReutersGrain-train.arff i zapoznaj się z jego budową. Jakie są atrybuty? Jakiej przyjmują wartości?
Jest to zbiór zawierający tekst oraz przypisaną do nich etykietę. Liczba rekordów wynosi 1554. Etykiety to 0 lub 1 przy czym elementów oznaczonych jako jest 14 razy więcej. Po przefiltrowaniu text używając StringToWordVector pojawiły się klasy zawierające każde słowo/liczbę/zbiór znaków z oryginalnego tekstu.
2. Przejdź do zakładki **Classify**, z gałęzi meta wybierz **FilteredClassifier**:
 - a. W ustawieniach wybierz classifier **NaiveBayes** oraz filter **StringToWordVector**.
 - b. W polu Test options wybierz **Supplied test set** i wskaż plik ReutersGrain-test.arff.
 - c. Uruchom klasyfikację wciskając **Start**. Zapoznaj się z otrzymanymi wynikami.

3. Wykonaj klasyfikację w analogiczny sposób korzystając z **NaiveBayesMultinomial** oraz algorytmu tworzenia drzewa decyzyjnego **J48**. Porównaj wyniki.



The screenshot shows the Weka Classifier window with the 'FilteredClassifier' selected. The 'Test options' section shows 'Supplied test set' selected. The 'Classifier output' section displays the following results:

```

Time taken to build model: 0.58 seconds

=== Evaluation on test set ===

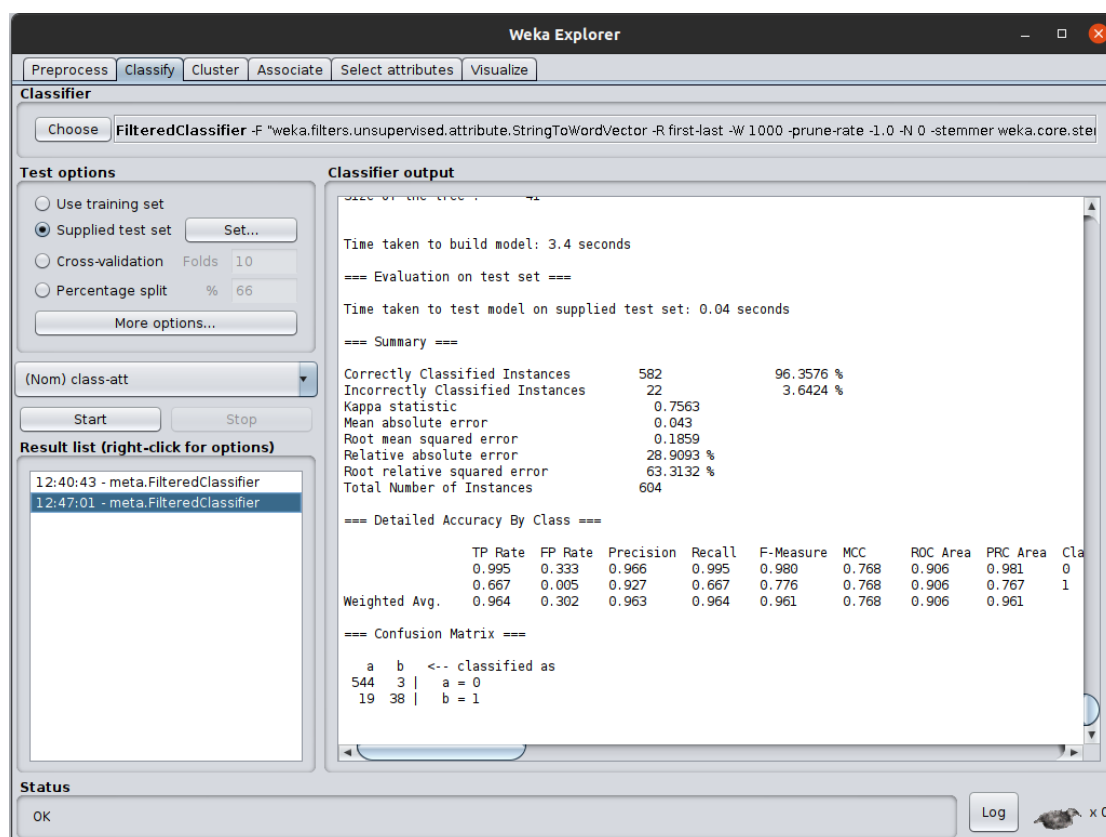
Time taken to test model on supplied test set: 0.27 seconds

=== Summary ===
Correctly Classified Instances      485          80.298 %
Incorrectly Classified Instances    119          19.702 %
Kappa statistic                    0.3459
Mean absolute error                 0.1984
Root mean squared error             0.4409
Relative absolute error             133.5501 %
Root relative squared error         150.1588 %
Total Number of Instances          604

=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Cla
Weighted Avg.   0.803   0.193   0.299     0.803   0.436     0.409  0.883    0.341    1
               0.803   0.193   0.912     0.803   0.839     0.409  0.885    0.925

=== Confusion Matrix ===
  a  b  <-- classified as
439 108 | a = 0
 11  46 | b = 1
    
```

Naive Bayes



The screenshot shows the Weka Explorer window with the 'Classify' tab selected. The 'Classifier' section shows 'FilteredClassifier' selected. The 'Test options' section shows 'Supplied test set' selected. The 'Classifier output' section displays the following results:

```

Time taken to build model: 3.4 seconds

=== Evaluation on test set ===

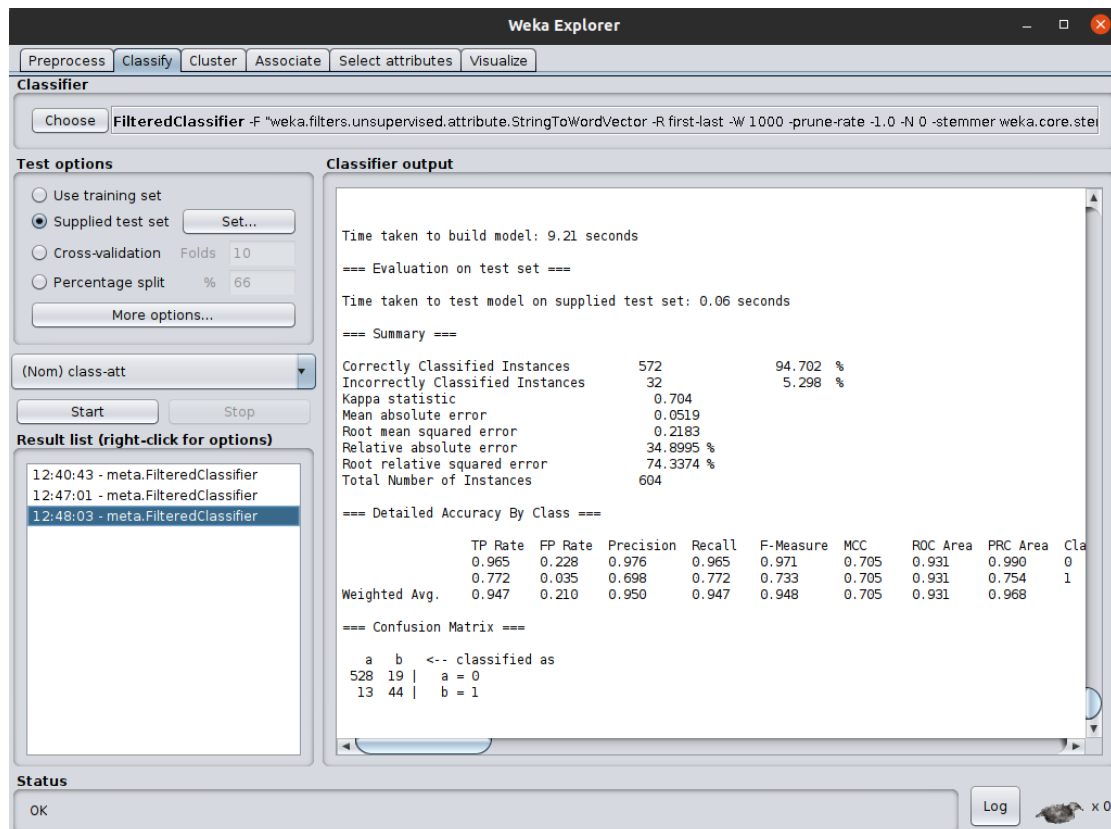
Time taken to test model on supplied test set: 0.04 seconds

=== Summary ===
Correctly Classified Instances      582          96.3576 %
Incorrectly Classified Instances     22           3.6424 %
Kappa statistic                    0.7563
Mean absolute error                 0.043
Root mean squared error             0.1859
Relative absolute error             28.9093 %
Root relative squared error         63.3132 %
Total Number of Instances          604

=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Cla
Weighted Avg.   0.995   0.333   0.966     0.995   0.980     0.768  0.906    0.981    0
               0.667   0.005   0.927     0.667   0.776     0.768  0.906    0.767    1
               0.964   0.302   0.963     0.964   0.961     0.768  0.906    0.961

=== Confusion Matrix ===
  a  b  <-- classified as
544   3 | a = 0
 19  38 | b = 1
    
```

J48



NaiveBayesMultinomial

Jak widać Naive Bayes ma problemy z klasyfikacją dlatego nie poświęcę mu dużo uwagi.

NaiveBayesMultinomial sprawują się trochę gorzej niż J48 (ma więcej źle zaklasyfikowanych rekordów), ale ma mniej False Negatives(13 do 19). Może zdarzyć się sytuacja w której będziemy chcieli uniknąć takiego błędu (np. badanie ludzi na HIV jest idealnym przykładem, ale nie odnoszącym się do nagłówków z reutersa)

4. Zapoznaj się z opisem filtru **StringToWordVector**. Jak myślisz, które jego opcje mogłyby poprawić klasyfikację? Zwróć uwagę np. na opcje outputWordCounts, lowerCaseTokens, useStoplist. Przetestuj działanie wybranych opcji pojedynczo i w grupach korzystając z algorytmu **NaiveBayesMultinomial**. Jak wpłynęły na jego skuteczność? Użyte pojedynczo opcje ogólnie polepszają wyniki klasyfikatora. Najlepsze poprawki wprowadziła opcja outputWordCounts. Po użyciu jej z lowerCaseTokens wyniki spadły, ale były wciąż lepsze niż bez żadnej opcji. StringToWordVector zwraca wektory w bardzo podstawowej formie, warto zwrócić uwagę na fakt, że np. `he is` oraz `he's` będzie reprezentowane tak samo pomimo takiego samego znaczenia. W tym przypadku pominąłbym preprocessing w weka na rzecz stworzenia własnego filtru i zaimplementowania tam reguł gramatycznych, które mogą znacząco wpłynąć na rozmiar wektora. Tak samo nie jest istotne, że w tekście jest napisana liczb 67, tylko fakt, że jest to liczba.

