Raport 3

Karol Pustelnik
album 249828

20 maja 2020

Spis treści

1 Krótki opis zagadnienia

W raporcie przedstawię różne algorytmy/metody klasyfikacji zmiennych. Do analizy użyję danych iris, które zawieraja informacje o cechach trzech gatunków kwiatów oraz danych spam, które zawieraja informacje o mailach chcianych i niechcianych wraz z ich własnościami. Spróbuję odpowiedzieć na pytania:

- Jaka metoda jest najlepsza dla konkretnego zbioru danych?
- Jakie są różnice pomiędzy metodami?

2 Opis eksperymentów

Do analizy użyję metod:

- k-Nearest Neighbors,
- classification trees,
- Klasyfikacja z wykorzystaniem naiwnego klasyfikatora bayesowskiego,
- regresje logistyczną,

3 Klasyfikacja na bazie modelu regresji liniowej.

```
library(MASS)
library("datasets")
library(ElemStatLearn)
library("HDclassif")
library(class)
library(ipred)
library(MASS)
library(rpart)
library(rpart.plot)
library(e1071)
set.seed(66)
data(iris)
data<-iris
a<-as.factor(data$Species)
data$Species<-as.numeric(a) #przekształcam nazwy gatunków na cyfry
dt = sample(nrow(data), nrow(data)*0.7) #dzielę zbiór danych
train<-data[dt,]
test<-data[-dt,]
model.train<-lm(train$Species~. , data = train)
class.table1<-table(train$Species,round(model.train$fitted.values))
class.table1

##
##      1  2  3
##   1 37  0  0
##   2  0 33  2
##   3  0  1 32

class.table2<-table(test$Species,round(predict(model.train, test)))
class.table2

##
##      1  2  3
##   1 13  0  0
##   2  0 15  0
##   3  0  3 14
```

Nie występuje problem maskowania klas. Model regresji liniowej poradził sobie bardzo dobrze z klasyfikacją.

```
err1<-(class.table1[1,2]+class.table1[1,3]+class.table1[2,1]
       +class.table1[2,3]+class.table1[3,1]+class.table1[3,2])/length(train$Species)
err1 #błąd klasyfikacji dla zbioru uczącego

## [1] 0.02857143

err2<-(class.table2[1,2]+class.table2[1,3]+class.table2[2,1]
       +class.table2[2,3]+class.table2[3,1]+class.table2[3,2])/length(test$Species)
err2 #Błąd dla zbioru testowego

## [1] 0.06666667
```

Jak widać błąd dla zbioru testowego jest większy niż dla zbioru treningowego. Zbudujmy teraz model liniowy dla rozszerzonej przestrzeni cech.

```
PL<-data$Petal.Length
PW<-data$Petal.Width
SL<-data$Sepal.Length
SW<-data$Sepal.Width
data<-cbind(data,PL^2,PW^2,SL^2,SW^2,PL*PW,PL*SW,PL*SL,PW*SL,PW*SW,SL*SW)

dt = sample(nrow(data), nrow(data)*0.7)
train<-data[dt,]
test<-data[-dt,]


model.train<-lm(train$Species~. , data = train)

class.table1<-table(train$Species,round(model.train$fitted.values))
test.classification<-predict(model.train, test)
class.table2<-table(test$Species,round(test.classification))
class.table1

##
##      1  2  3
##   1 39  0  0
##   2  0 30  1
##   3  0  0 35

class.table2

##
##      1  2  3
##   1 11  0  0
##   2  0 17  2
##   3  0  1 14
```

```
err1<-(class.table1[1,2]+class.table1[1,3]+class.table1[2,1]
       +class.table1[2,3]+class.table1[3,1]+class.table1[3,2])/length(train$Species)
err1 #błąd klasyfikacja dla zbioru uczącego

## [1] 0.00952381

err2<-(class.table2[1,2]+class.table2[1,3]+class.table2[2,1]
       +class.table2[2,3]+class.table2[3,1]+class.table2[3,2])/length(test$Species)
err2 #błąd klasyfikacja dla zbioru testowego

## [1] 0.06666667
```

Dla rozszerzonej przestrzeni cech otrzymialiśmy dokładniejsze wyniki. Jest tak dlatego, ponieważ modelowi jest łatwiej odróżnić gatunki, gdy mają wiecej cech. Tak samo np. łatwiej jest odróżnić mały zielony sześcian wykonany z plastiku od dużego kolorowego ostrosłupa zrobionego z metalu, niż małą piłeczkę od średniej piłeczki.

Ogólnie do klasyfikacji lepszy jest model regresji logistycznej, a nie liniowej. W kolejnym zadaniu uzwględnię model regresji logistycznej.

4 Porównanie różnych metod klasyfikacji

Porównam teraz różne metody klasyfikacji przy użyciu danych "spam" z biblioteki "ElemStatLearn". Zbiór danych zawiera 57 (ciągłych i dyskretnych) zmiennych objaśniających i jedną zmienna objaśnianą "spam", która informuje nas czy dany mail został sklasyfikowany jako spam czy nie. Nie ma w tym zbiorze brakujących danych. Przygotuję zbiór testowy i treningowy do analizy algorytmów. Dzielę dane w proporcji 30 procent dla zbioru testowego i 70 procent dla zbioru treningowego.

```
porownanie<-matrix(nrow=3, ncol=4) #towrzę macierz do zapisywania wyników
rownames(porownanie) <- c("Czułość", "Specyficzność",
                          "Szansa na sklasyfikowanie")
colnames(porownanie)<-c("Logistyczna", "k-NN",
                        "Drzewa",
                        "Bayes")
data(spam)
data<-spam
spam<-spam$spam
data$spam<-as.numeric(spam)
data$spam[data$spam==1]<-0
data$spam[data$spam==2]<-1
dt = sample(nrow(data), nrow(data)*0.7)
train<-data[dt,]
test<-data[-dt,]
```

4.1 Metoda k-NN

```
model<-ipredknn(train$spam ~ ., data=train, k=6) #Ustalam liczbę sąsiadów na 6.

class.predicted <- predict(model, test,type="class")

results<-table(class.predicted,test$spam)
```

Confusion matrix:

```
results

##
## class.predicted   0   1
##               0 684 137
##               1 149 411
```

Jak widać nasz algorytm poradził sobie dosyć dobrze. Zdecydowana większość emaili została dobrze skategoryzowana. Wyznaczmy pozostałe parametry, które oceniaja poprawność algorytmu, czyli czułość, specyficzność i prawdopodobienstwo poprawnej klasyfikacji.

```
TP<-results[2,2]
TN<-results[1,1]
FP<-results[1,2]
FN<-results[2,1]
sensitivity<-TP/(TP+FN)
specificity<-TN/(TN+FP)
overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.7339286
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.8331303
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.7929037
```

Interpretacja wyników w dużej mierze zależy od typu danych jaki analizujemy. W naszym przypadku chcemy zmaksymalizować specyficzność, ponieważ informuję nas ona czy i jak dużo maili zostało fałszywie sklasyfikowanych jako spam (false positives). Dążymy do tego, aby wybrać taki algorytm, który zminimalizuje ilość "false positives" nawet kosztem pozostałych parametrów. Jasne, że kilka (może nawet kilkadziesiąt) niepotrzebnych maili nam nie zaszkodzi, a w zamian za to nie przepadnie nam żaden ważny list. Z drugiej strony możnaby się spierać, że wśród wielu nieistotnych maili ciężej znaleźć ważną wiadomość, ale przecież można po prostu uważniej przeglądać pocztę.

Spróbujmy znaleźć lepsze k, tzn, dla którego specyficzność będzie największa.

```
lista<-list()
for (i in 1:15)
{
  model<-ipredknn(spam ~ ., data=train, k=i)
  class.table<-table(test$spam,predict(model, test,type="class"))
  TP<-class.table[2,2]
  TN<-class.table[1,1]
  FP<-class.table[1,2]
  FN<-class.table[2,1]
  lista[i]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
}
lista

## [[1]]
## [1] 0.8095583
##
## [[2]]
## [1] 0.7929037
##
## [[3]]
## [1] 0.8081101
##
## [[4]]
## [1] 0.7965243
##
## [[5]]
## [1] 0.7994207
##
## [[6]]
## [1] 0.7842143
##
## [[7]]
## [1] 0.7878349
```

```
##
## [[8]]
## [1] 0.782042
##
## [[9]]
## [1] 0.7958001
##
## [[10]]
## [1] 0.7871108
##
## [[11]]
## [1] 0.7827661
##
## [[12]]
## [1] 0.7798697
##
## [[13]]
## [1] 0.7849385
##
## [[14]]
## [1] 0.782042
##
## [[15]]
## [1] 0.7834902
```

```
model<-ipredknn(spam ~ ., data=train, k=1)
class.table<-table(test$spam,predict(model, test,type="class"))
TP<-class.table[2,2]
TN<-class.table[1,1]
FP<-class.table[1,2]
FN<-class.table[2,1]
porownanie[1,2]<-sensitivity<-TP/(TP+FN)
porownanie[2,2]<-specificity<-TN/(TN+FP)
porownanie[3,2]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Największą specyficzność otrzymujemy dla k=1.

4.2 Regresja Logistyczna

Teraz zastosuję regresję logistyczna do klasyfikacji maili.

```
model.logistic<-glm(train$spam~., data = train, family="binomial")
class.table1<-table(test$spam,predict(model.logistic,test)>0.5)
```

Na poziomie odcięcia pi = 0.5 confusion matrix wygląda nastepująco:

```
class.table1

##
##     FALSE TRUE
##   0   805   28
##   1    65  483
```

Wyznaczmy pozostałe parametry.
Czułość jest na poziomie:

```
sensitivity

## [1] 0.8813869
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.9663866
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.9326575
```

Ten model jest zatem o niebo lepszy bo specyficzność jest znacznie większa w porównaniu z modelem wykorzystującym k-NN. Pozostałe parametry też przemawiają za modelem logistycznym. Okazuje się, że nasz model możemy usprawnić wybierając inny poziom odcięcia. Tak jak wcześniej wspomniałem, w tej analizie interesuje nas model, który zmaksymalizuje specyficzność. Zatem sprawdźmy czy istnieje taki poziom odcięcia, dla którego będzie ona lepsza.

```
lista<-list()
for (i in 1:9)
{
  class.table<-table(test$spam,predict(model.logistic,test)>i/10)
  TP<-class.table[2,2]
  TN<-class.table[1,1]
  FP<-class.table[1,2]
  FN<-class.table[2,1]
  lista[i]<-specificity<-TN/(TN+FP)
}
lista
```

```
## [[1]]
## [1] 0.9603842
##
## [[2]]
## [1] 0.9627851
##
## [[3]]
## [1] 0.9627851
##
## [[4]]
## [1] 0.9651861
##
## [[5]]
## [1] 0.9663866
##
## [[6]]
## [1] 0.9711885
##
## [[7]]
## [1] 0.972389
##
## [[8]]
## [1] 0.9759904
##
## [[9]]
## [1] 0.9759904
```

Po przejrzeniu listy, najlepszym poziomem odcięcia jest p = 0.9. Sprawdźmy wartości pozostałych parametrów.

Confusion matrix:

```
class.table1

##
##     FALSE TRUE
##   0   813   20
##   1    84  464
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.8467153
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.9759904
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.9246923
```

4.3 Algorytm drzew klasyfikacyjnych

Zobaczmy teraz jak poradzi sobie algorytm drzew klasyfikacyjnych.

Confusion matrix:

```
class.table

##
##         email spam
##   email   909   41
##   spam     92  523
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.8504065
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.9568421
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.915016
```

4.4 Naiwny klasyfikator bayesowski

Teraz sprawdzmy jak poradzi sobie naiwny klasyfikator bayesowski.

```
model<-naiveBayes(train$spam~., data = train)
class.table <- table(test$spam,predict(model, test))
TP<-class.table[2,2]
TN<-class.table[1,1]
FP<-class.table[1,2]
FN<-class.table[2,1]
porownanie[1,4]<-sensitivity<-TP/(TP+FN)
porownanie[2,4]<-specificity<-TN/(TN+FP)
porownanie[3,4]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Confusion matrix:

```
class.table

##
##         email spam
##   email   513  437
##   spam     40  575
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.9349593
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.54
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.6952077
```

Ten algorytm poradził sobie zdecydowanie najgorzej w porównaniu z pozostałymi.

```
porownanie

##                           Logistyczna      k-NN    Drzewa     Bayes
## Czułość                     0.8467153 0.7846715 0.8504065 0.9349593
## Specyficzność               0.9759904 0.8259304 0.9568421 0.5400000
## Szansa na sklasyfikowanie   0.9246923 0.8095583 0.9150160 0.6952077
```

Najlepiej wypadła regresja logistyczna.

5 Analiza metod dla wybranych zmiennych objaśniających

Nasze dane zawierają wiele oberwacji, a przede wszystkim dużo zmiennych objaśniających. Być może część z nich jest niepotrzebna, a nawet negtywnie wpływa na zdolności klasyfikacyjne przedstawionych wyżej metod. Przeprowadzę teraz ponowną analizę, ale uwzględnie tylko zmienne, które są statystycznie znaczące. Wykorzystam do tego wyniki p-value dla modelu regresji logistycznej.

```r
porownanie2<-matrix(nrow=3, ncol=4) #tabelka porownawcza
rownames(porownanie2) <- c("Czułość", "Specyficzność",
                           "Szansa na sklasyfikowanie")
colnames(porownanie2)<-c("Logistyczna", "k-NN",
                         "Drzewa",
                         "Bayes")
spam.col<-spam$spam
data$spam<-as.numeric(spam.col)
data$spam[data$spam==1]<-0
data$spam[data$spam==2]<-1
dt = sample(nrow(data), nrow(data)*0.66)
train<-data[dt,]
test<-data[-dt,]
model.logistic<-glm(train$spam~., data = train, family="binomial")
summary(model.logistic)
```

```
##
## Call:
## glm(formula = train$spam ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0752  -0.1709   0.0000   0.1114   3.9675
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.702e+00  1.986e-01  -8.573  < 2e-16 ***
## A.1         -2.184e-01  2.640e-01  -0.827 0.408075
## A.2         -1.432e-01  7.793e-02  -1.837 0.066203 .
## A.3          2.839e-02  1.487e-01   0.191 0.848618
## A.4          1.772e+00  1.639e+00   1.081 0.279693
## A.5          5.821e-01  1.254e-01   4.640 3.48e-06 ***
## A.6          6.719e-01  2.975e-01   2.258 0.023925 *
## A.7          2.442e+00  4.260e-01   5.733 9.85e-09 ***
## A.8          5.255e-01  2.290e-01   2.294 0.021766 *
## A.9          1.246e+00  4.078e-01   3.057 0.002236 **
## A.10         1.942e-03  7.889e-02   0.025 0.980358
## A.11        -7.654e-01  3.763e-01  -2.034 0.041945 *
## A.12        -1.868e-01  1.012e-01  -1.847 0.064815 .
## A.13        -1.660e-01  2.903e-01  -0.572 0.567521
## A.14         1.448e-01  1.764e-01   0.821 0.411788
## A.15         1.597e+00  9.413e-01   1.697 0.089739 .
## A.16         9.916e-01  1.689e-01   5.872 4.30e-09 ***
## A.17         1.137e+00  2.815e-01   4.039 5.38e-05 ***
## A.18         1.091e-01  1.371e-01   0.796 0.426143
## A.19         9.849e-02  4.527e-02   2.176 0.029583 *
## A.20         6.828e-01  5.259e-01   1.298 0.194221
## A.21         2.398e-01  6.467e-02   3.708 0.000209 ***
```

```
## A.22         3.334e-01  2.287e-01   1.458 0.144909
## A.23         2.259e+00  5.638e-01   4.006 6.17e-05 ***
## A.24         5.267e-01  2.301e-01   2.289 0.022058 *
## A.25        -2.590e+00  4.835e-01  -5.357 8.44e-08 ***
## A.26        -6.801e-01  4.717e-01  -1.442 0.149334
## A.27        -7.198e+00  2.576e+00  -2.794 0.005202 **
## A.28         7.609e-01  3.414e-01   2.229 0.025834 *
## A.29        -2.544e+00  1.706e+00  -1.492 0.135766
## A.30         1.971e-01  3.380e-01   0.583 0.559849
## A.31        -6.953e+00  3.893e+00  -1.786 0.074062 .
## A.32         1.077e+01  2.543e+01   0.423 0.672071
## A.33        -9.516e-01  4.158e-01  -2.289 0.022096 *
## A.34        -1.294e+01  4.838e+00  -2.674 0.007499 **
## A.35        -4.191e+00  1.750e+00  -2.395 0.016608 *
## A.36         1.481e+00  4.172e-01   3.551 0.000384 ***
## A.37         1.067e-01  2.226e-01   0.479 0.631628
## A.38         1.270e+00  1.435e+00   0.885 0.376069
## A.39        -7.757e-01  4.674e-01  -1.659 0.097016 .
## A.40        -4.226e-01  4.440e-01  -0.952 0.341220
## A.41        -3.942e+01  3.569e+01  -1.105 0.269369
## A.42        -4.546e+00  2.049e+00  -2.218 0.026528 *
## A.43        -2.084e+00  1.282e+00  -1.625 0.104056
## A.44        -1.461e+00  6.130e-01  -2.384 0.017141 *
## A.45        -1.053e+00  2.182e-01  -4.825 1.40e-06 ***
## A.46        -1.282e+00  3.180e-01  -4.033 5.52e-05 ***
## A.47        -2.364e+00  1.870e+00  -1.265 0.206018
## A.48        -4.635e+00  2.491e+00  -1.861 0.062793 .
## A.49        -1.506e+00  6.453e-01  -2.334 0.019594 *
## A.50        -5.159e-01  4.234e-01  -1.219 0.223016
## A.51        -1.580e+00  1.541e+00  -1.025 0.305279
## A.52         3.490e-01  9.629e-02   3.624 0.000290 ***
## A.53         4.617e+00  7.800e-01   5.919 3.24e-09 ***
## A.54         1.373e+00  1.310e+00   1.048 0.294501
## A.55         7.708e-02  5.413e-02   1.424 0.154488
## A.56         1.160e-02  3.708e-03   3.130 0.001750 **
## A.57         7.339e-04  2.860e-04   2.566 0.010300 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4070.3  on 3035  degrees of freedom
## Residual deviance: 1160.0  on 2978  degrees of freedom
## AIC: 1276
##
## Number of Fisher Scoring iterations: 13
```

Na poziomie ufności p = 0.05 statystycznie znaczące są zmienne: A.5, A.6, A.7, A.8, A.10, A.12, A.15, A.16, A.17, A.20, A.21, A.23, A.24, A.25, A.26, A.27, A.33, A.36, A.42, A.44, A.45, A.46, A.48, A.49, A.52, A.53, A.56, A.57.

5.1 Regresja logistyczna

Zacznę od regresji logistycznej i odrazu wyznaczę najlepszy poziom odcięcia.

```
spam.column<-data$spam
attach(data)
data.filtered<-as.data.frame(cbind(A.5, A.6, A.7, A.8, A.10, A.12,
                                   A.15, A.16, A.17, A.20, A.21,
                                   A.23, A.24, A.25, A.26, A.27,
                                   A.33, A.36, A.42, A.44, A.45,
                                   A.46, A.48, A.49, A.52, A.53,
                                   A.56, A.57, spam.column))
dt = sample(nrow(data.filtered), nrow(data.filtered)*0.66)
train<-data.filtered[dt,]
test<-data.filtered[-dt,]
model.logistic<-glm(train$spam.column~., data = train, family="binomial")
lista<-list()
for (i in 1:9)
{
  class.table<-table(test$spam.column,predict(model.logistic,test)>i/10)
  TP<-class.table[2,2]
  TN<-class.table[1,1]
  FP<-class.table[1,2]
  FN<-class.table[2,1]
  lista[i]<-specificity<-TN/(TN+FP)
}
lista

## [[1]]
## [1] 0.9526316
##
## [[2]]
## [1] 0.9547368
##
## [[3]]
## [1] 0.9568421
##
## [[4]]
## [1] 0.9589474
##
## [[5]]
## [1] 0.96
##
## [[6]]
```

```
## [1] 0.9621053
##
## [[7]]
## [1] 0.9621053
##
## [[8]]
## [1] 0.9621053
##
## [[9]]
## [1] 0.9631579
```

Najlepszy wynik jest dla poziomu odcięcia pi = 0.9. Pogorszyła się specyficzność, gdy użyliśmy tylko statystycznie znaczących zmiennych. Zobaczmy czy pozostałe parametry się polepszyły.

```
class.table1<-table(test$spam.column,predict(model.logistic,test)>0.9)
class.table1

##
##     FALSE TRUE
##   0   915   35
##   1   121  494

TP<-class.table1[2,2]
TN<-class.table1[1,1]
FP<-class.table1[1,2]
FN<-class.table1[2,1]
porownanie2[1,1]<-sensitivity<-TP/(TP+FN)
porownanie2[2,1]<-specificity<-TN/(TN+FP)
porownanie2[3,1]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.803252
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.9631579
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.9003195
```

Niestety pozostałe parametry tez się pogorszyły.

5.2 Algorytm k-NN

Teraz algorytm k-NN. Spróbujmy odrazu znaleźć najlepsze k, tzn, dla którego specyficzność będzie największa.

```
lista<-list()
for (i in 1:15)
{
  model<-ipredknn(spam.column ~ ., data=train, k=i)
  class.table<-table(test$spam.column,predict(model, test,type="class"))
  TP<-class.table[2,2]
  TN<-class.table[1,1]
  FP<-class.table[1,2]
  FN<-class.table[2,1]
  lista[i]<-specificity<-TN/(TN+FP)
}
lista

## [[1]]
## [1] 0.8221053
##
## [[2]]
## [1] 0.8
##
## [[3]]
## [1] 0.8252632
##
## [[4]]
## [1] 0.8031579
##
## [[5]]
## [1] 0.8063158
##
## [[6]]
## [1] 0.8115789
##
## [[7]]
## [1] 0.8136842
##
## [[8]]
## [1] 0.8084211
##
## [[9]]
## [1] 0.8157895
##
## [[10]]
## [1] 0.8052632
##
## [[11]]
```

```
## [1] 0.8136842
##
## [[12]]
## [1] 0.8
##
## [[13]]
## [1] 0.8094737
##
## [[14]]
## [1] 0.7989474
##
## [[15]]
## [1] 0.8
```

```
model<-ipredknn(spam.column ~ ., data=train, k=3)
class.table<-table(test$spam.column,predict(model, test,type="class"))
TP<-class.table[2,2]
TN<-class.table[1,1]
FP<-class.table[1,2]
FN<-class.table[2,1]
porownanie2[1,2]<-sensitivity<-TP/(TP+FN)
porownanie2[2,2]<-specificity<-TN/(TN+FP)
porownanie2[3,2]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Najlepszą specyficzność dostajemy dla k = 3.

5.3 Algorytm drzew klasyfikacyjnych

Sprawdźmy teraz działanie algorytmu drzew klasyfikacyjnych.

```
data(spam)
data<-spam
attach(data)
spam.column<-data$spam
data.filtered<-data.frame(A.5, A.6, A.7, A.8, A.10, A.12,
                          A.15, A.16, A.17, A.20, A.21,
                          A.23, A.24, A.25, A.26, A.27,
                          A.33, A.36, A.42, A.44, A.45,
                          A.46, A.48, A.49, A.52, A.53,
                          A.56, A.57, spam.column)
dt = sample(nrow(data), nrow(data)*0.66)
train<-data.filtered[dt,]
test<-data.filtered[-dt,]
model.tree <- rpart(train$spam.column ~., data=train)
class.table<-table(test$spam.column,predict(model.tree, newdata=test,type="class"))
TP<-class.table[2,2]
TN<-class.table[1,1]
FP<-class.table[1,2]
```

```
FN<-class.table[2,1]
porownanie2[1,3]<-sensitivity<-TP/(TP+FN)
porownanie2[2,3]<-specificity<-TN/(TN+FP)
porownanie2[3,3]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.8216039
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.9392034
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.8932907
```

W porównaniu z modelem uwzględniającym wszystkie zmienne, tutaj pogorszyły się wszystkie parametry.

5.4 Naiwny klasyfikator bayesowski

```
model<-naiveBayes(train$spam.column~., data = train)

class.table <- table(test$spam.column,predict(model, test))
class.table

##
##         email spam
##   email   593  361
##   spam     29  582

TP<-class.table[2,2]
TN<-class.table[1,1]
FP<-class.table[1,2]
FN<-class.table[2,1]
porownanie2[1,4]<-sensitivity<-TP/(TP+FN)
porownanie2[2,4]<-specificity<-TN/(TN+FP)
porownanie2[3,4]<-overall.correct<-(TP+TN)/(TP+TN+FN+FP)
```

Czułość jest na poziomie:

```
sensitivity

## [1] 0.9525368
```

Specyficzność jest na poziomie:

```
specificity

## [1] 0.6215933
```

Szansa na poprawną klasyfikacje wynosi:

```
overall.correct

## [1] 0.7507987
```

Tutaj z kolei wszystkie parametry się polepszyły.

6 Porównanie wyników

Tabelka porownawcza dla wybranych zmiennych:

```
porownanie2

##                            Logistyczna      k-NN    Drzewa     Bayes
## Czułość                      0.8032520 0.7284553 0.8216039 0.9525368
## Specyficzność                0.9631579 0.8252632 0.9392034 0.6215933
## Szansa na sklasyfikowanie    0.9003195 0.7872204 0.8932907 0.7507987
```

Tabelka porownawcza dla wszystkich zmiennych:

```
porownanie

##                            Logistyczna      k-NN    Drzewa     Bayes
## Czułość                      0.8467153 0.7846715 0.8504065 0.9349593
## Specyficzność                0.9759904 0.8259304 0.9568421 0.5400000
## Szansa na sklasyfikowanie    0.9246923 0.8095583 0.9150160 0.6952077
```

7 Podsumowanie

Podsumowując, najlepsza okazała się metoda regresji logistycznej. Zaraz po niej algorytm drzew klasyfikacyjnych. Na trzecim miejscu ląduje algorytm KNN. Najgorzej poradził sobie naiwny klasyfikator bayesowski. Biorąc pod uwagę tylko zmienne statystycznie znaczące nie dostaliśmy znacznie lepszych wyników. Tylko dla klasyfikatora bayesowskiego parametry sie polepszyły.

Literatura

[1] Stackoverflow, https://stackoverflow.com

[2] Strona kursu Eksploracja danych, prowadzonego przez dr. Adama Zagdańskiego, http://prac.im.pwr.wroc.pl/~zagdan/polish_ver/ED2020/index.html