# Formatowanie spamu
# Data Mining

Michał Maj i Anna Mieszkalska
album 256556 i 255699

6 maja 2023

## Spis treści

# 1 Analiza opisowa i wizualizacja

## 1.1 Wstęp

W naszym projekcie będziemy analizować dane o nazwie *Spambase* z biblioteki *kernlab*. Zestaw danych *spambase* jest zbiorem wiadomości e-mail, które zostały przeanalizowane i sklasyfikwane jako spam lub non-spam. Celem tego zbioru danych jest dostarczenie użytecznych materiałów potrzebnych do analiz i eksploracji w tym zakresie. W tym projekcie użyjemy różnych metod i technik pozyskiwania wiedzy, aby przeanalizować dane *spambase* w celu opracowania modelu klasyfikującego wiadomości e-mail jako spam lub non-spam. Modele opracowane w tym projekcie mogą być przydatne w rzeczywistych serwisach poczt e-mailowych, gdzie problem dostarczania niechcianych wiadomości jest nam powszechnie znany.

## 1.2 Opis danych

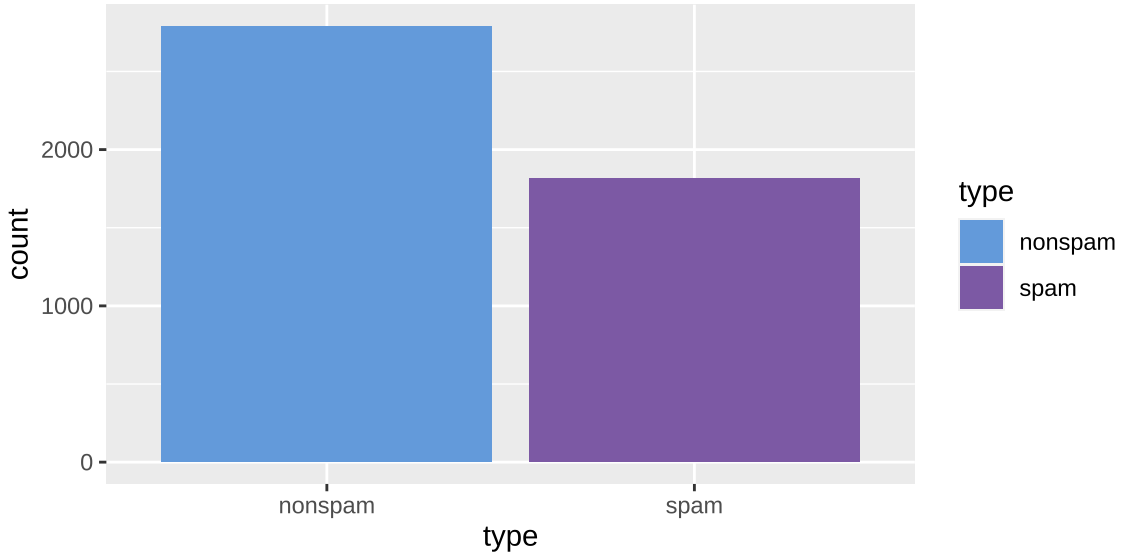
Zbiór danych *spambase* wyodrębnia 58 cech, które oznaczają częstość występowania danego znaku bądź słowa w jednym e-mailu. Pierwsze 48 zmiennych dotyczy występowania konkretnych słów, następne 6 występowania znaków, a kolumny 55-57 dotyczą średniej, najdłuższej i całkowitej długości wielkich liter. Ostatnia zmienna *type* odpowiada za określenie typu e-maila jako spam lub non-spam, zatem będziemy rozważać dwie klasy. Zbiór ten składa się z 4601 obserwacji (wiadomości e-mail). Z powodu dużej obszerności danych przedstawiamy tylko część zmiennych (tabela 1), następnie pokazujemy rozkład klas wykorzystując różne metody wizualizacji (2, 1 i 3).

| | make | address | all | num3d | our | over | remove | internet | order | mail | „(” | type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.000 | 0.640 | 0.640 | 0.000 | 0.320 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | spam |
| 2 | 0.210 | 0.280 | 0.500 | 0.000 | 0.140 | 0.280 | 0.210 | 0.070 | 0.000 | 0.940 | 0.132 | spam |
| 3 | 0.060 | 0.000 | 0.710 | 0.000 | 1.230 | 0.190 | 0.190 | 0.120 | 0.640 | 0.250 | 0.143 | spam |
| 4 | 0.000 | 0.000 | 0.000 | 0.000 | 0.630 | 0.000 | 0.310 | 0.630 | 0.310 | 0.630 | 0.137 | spam |
| 5 | 0.000 | 0.000 | 0.000 | 0.000 | 0.630 | 0.000 | 0.310 | 0.630 | 0.310 | 0.630 | 0.135 | spam |
| 6 | 0.000 | 0.000 | 0.000 | 0.000 | 1.850 | 0.000 | 0.000 | 1.850 | 0.000 | 0.000 | 0.223 | spam |
| 7 | 0.000 | 0.000 | 0.000 | 0.000 | 1.920 | 0.000 | 0.000 | 0.000 | 0.000 | 0.640 | 0.054 | spam |
| 8 | 0.000 | 0.000 | 0.000 | 0.000 | 1.880 | 0.000 | 0.000 | 1.880 | 0.000 | 0.000 | 0.206 | spam |
| 9 | 0.150 | 0.000 | 0.460 | 0.000 | 0.610 | 0.000 | 0.300 | 0.000 | 0.920 | 0.760 | 0.271 | spam |
| 10 | 0.060 | 0.120 | 0.770 | 0.000 | 0.190 | 0.320 | 0.380 | 0.000 | 0.060 | 0.000 | 0.030 | spam |
| 11 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.960 | 0.000 | 0.000 | 1.920 | 0.000 | spam |
| 12 | 0.000 | 0.000 | 0.250 | 0.000 | 0.380 | 0.250 | 0.250 | 0.000 | 0.000 | 0.000 | 0.044 | spam |
| 13 | 0.000 | 0.690 | 0.340 | 0.000 | 0.340 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.056 | spam |
| 14 | 0.000 | 0.000 | 0.000 | 0.000 | 0.900 | 0.000 | 0.900 | 0.000 | 0.000 | 0.900 | 0.000 | spam |

Tabela 1: Spambase - pierwsze 14 rekordów dla wybranych zmiennych.

Tabela 2: Rozkład klas.

| type | ilość |
|---|---|
| nonspam | 2788 |
| spam | 1813 |




Rysunek 1: Wykres rozkładu klas.

Tabela 3: Rozkład klas (procentowo).

| type | procent |
|---|---|
| nonspam | 60.6 |
| spam | 39.4 |

Ilość e-maili, które zostały sklasyfikowane jako spam wynosi 1813 (tabela 2), a ilość tych, które nie są spamem wynosi 2733, zatem klasa spam stanowi prawie 40% całości (tabela 3), więc dane są dość zbalansowane co potwierdza rysunek 1. Za pomocą funkcji `str` mamy pokazane, że wszystkie zmienne są typu *numeric*, oczywiście oprócz zmiennej *type*, która jest typu *factor*. Możemy również sprawdzić typ zmiennych za pomocą funkcji `sapply` i zliczyć ich ilość. Otrzymamy wynik 57 bez zmiennej *type*.

```
sum(sapply(spam,is.numeric))

## [1] 57
```

Patrząc do tabeli 1 widzimy, że wszystkie typy zmiennych zostały określone prawidłowo. Funkcja `is.na()` mówi nam, że nasze dane nie posiadają żadnych wartości NA, należy jednak sprawdzić, czy w tym przypadku nie są one kodowane inaczej. Możemy użyć funkcji `complete.cases`, żeby usunąć wszelkie brakujące wartości.

```
sum(!complete.cases(spam))

## [1] 0
```

Wynik wyszedł 0, zatem nasze dane nie mają żadnych brakujących wartości oraz niestandardowego kodowania.

Podsumowując, mamy:

- $n = 4601$ (liczba przypadków),
- $p = 58$ (liczba cech),
- $K = 2$ (licza klas, „spam" i „nonspam"),
- 0 wartości brakujących.

## 1.3 Przygotowanie danych do analizy

Przed analizą poszczególnych zmiennych należy znormalizować dane, ponieważ są nierównomiernie rozłożone. Zostały one znormalizowane przy użyciu własnoręcznie zaimplementowanej funkcji `normalize`, tak aby wszystkie wartości liczbowe mieściły się w przedziale od 0 do 1.

```
normalize <- function(x) {
        return( (x - min(x)) / (max(x) - min(x)))
}
NormalizeSpam <- as.data.frame(lapply(spam[1:57], normalize))

#Cleaning Data - dodanie kolumny type (naszej klasy)
CleanSpam <- cbind(spam[, 58], NormalizeSpam)
names(CleanSpam)[names(CleanSpam) == "spam[, 58]"] <- "class"
CleanSpam$class <- as.character(CleanSpam$class)

#usytuowanie zmiennej class na sam koniec
CleanSpam <- CleanSpam %>% relocate(class, .after=capitalTotal)
```

W poprzedniej sekcji pokazaliśmy, że nie ma żadnych brakujących wartości. Musimy również usunąć obserwacje zduplikowane, za pomocą funkcji `distinct` z pakietu *dplyr*.

```
## Liczba usuniętych wierszy: 391
```

Ilość obserwacji zmniejszyła się i wynosi aktualnie 4210.
Obliczając jeden ze wskaźników sumarycznych otrzymujemy, że większość zmiennych cechuję się bardzo niską wariancją (tabela 4), co świadczy o bardzo małej zmienności, zatem ciężko będzie nam wybrać zmienne, które powinny zostać usunięte (biorąc pod uwagę tylko ten aspekt).

| | make | address | all | num3d | our | over | remove | internet | order | mail | „(" |
|---|---|---|---|---|---|---|---|---|---|---|---|
| make | 0.00437 | 0.00007 | 0.00042 | 0.00001 | 0.00010 | 0.00017 | 0.00004 | -0.00001 | 0.00037 | 0.00010 | -0.00004 |
| address | 0.00007 | 0.00101 | 0.00009 | -0.00001 | 0.00008 | 0.00001 | 0.00013 | 0.00002 | 0.00009 | 0.00021 | -0.00003 |
| all | 0.00042 | 0.00009 | 0.01023 | -0.00006 | 0.00046 | 0.00032 | 0.00016 | 0.00003 | 0.00042 | 0.00009 | -0.00007 |
| num3d | 0.00001 | -0.00001 | -0.00006 | 0.00100 | 0.00000 | -0.00001 | 0.00002 | 0.00000 | -0.00000 | -0.00000 | -0.00001 |
| our | 0.00010 | 0.00008 | 0.00046 | 0.00000 | 0.00473 | 0.00013 | 0.00051 | 0.00006 | 0.00005 | 0.00007 | -0.00011 |
| over | 0.00017 | 0.00001 | 0.00032 | -0.00001 | 0.00013 | 0.00220 | 0.00012 | 0.00014 | 0.00024 | 0.00002 | -0.00002 |
| remove | 0.00004 | 0.00013 | 0.00016 | 0.00002 | 0.00051 | 0.00012 | 0.00299 | 0.00007 | 0.00014 | 0.00011 | -0.00009 |
| internet | -0.00001 | 0.00002 | 0.00003 | 0.00000 | 0.00006 | 0.00014 | 0.00007 | 0.00136 | 0.00021 | 0.00011 | -0.00004 |
| order | 0.00037 | 0.00009 | 0.00042 | -0.00000 | 0.00005 | 0.00024 | 0.00014 | 0.00021 | 0.00288 | 0.00024 | -0.00006 |
| mail | 0.00010 | 0.00021 | 0.00009 | -0.00000 | 0.00007 | 0.00002 | 0.00011 | 0.00011 | 0.00024 | 0.00130 | -0.00001 |
| „(" | -0.00004 | -0.00003 | -0.00007 | -0.00001 | -0.00011 | -0.00002 | -0.00009 | -0.00004 | -0.00006 | -0.00001 | 0.00079 |

Tabela 4: Wariancja poszczególnych zmiennych.

## 1.4 Analiza poszczególnych zmiennych

W tej sekcji zajmiemy się pozostałymi wskaźnikami sumarycznymi czyli między innymi miarami położenia i rozrzutu, wyznaczonymi dla wszystkich danych.
Wiemy, że wszystkie zmienne oprócz *type* są ilościowe zatem możemy użyć funkcji `describe` z pakietu *dlookr*, żeby zobaczyć statystyki opisowe.

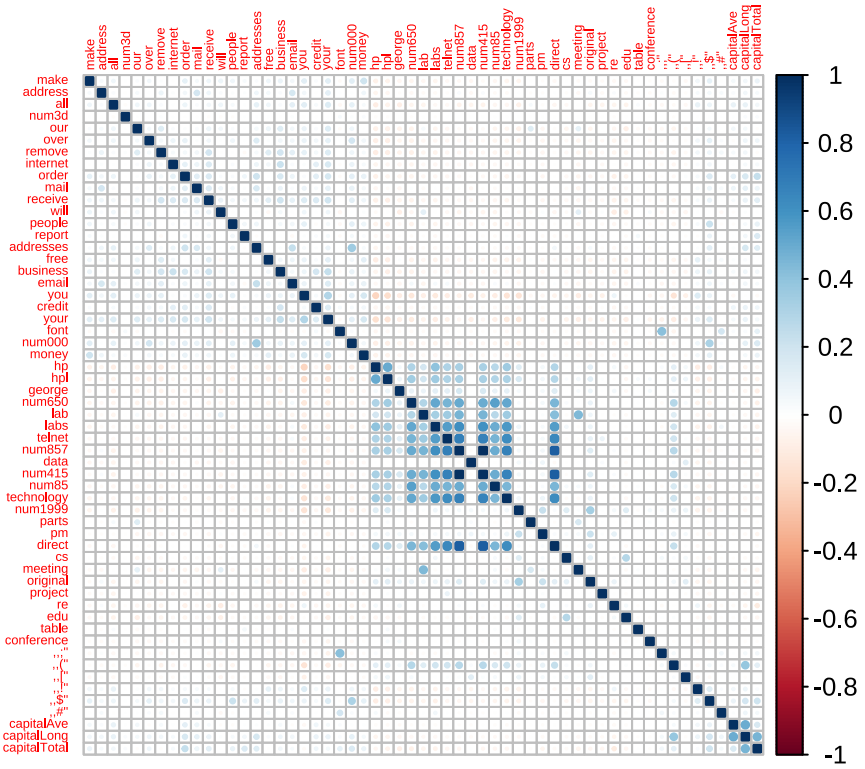
Tabela 5: Wskaźniki sumaryczne dla wszystkich zmiennych.

| described_variables | n | na | mean | sd | se_mean | IQR | skewness | kurtosis | p25 | p50 | p75 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| make | 4210 | 0 | 0.0230 | 0.0661 | 0.0010 | 0.0000 | 5.8699 | 54.3514 | 0.0000 | 0.0000 | 0.0000 |
| address | 4210 | 0 | 0.0079 | 0.0318 | 0.0005 | 0.0000 | 13.4647 | 306.6860 | 0.0000 | 0.0000 | 0.0000 |
| all | 4210 | 0 | 0.0572 | 0.1011 | 0.0016 | 0.0863 | 2.9806 | 12.9414 | 0.0000 | 0.0000 | 0.0863 |
| num3d | 4210 | 0 | 0.0015 | 0.0316 | 0.0005 | 0.0000 | 27.2095 | 788.3701 | 0.0000 | 0.0000 | 0.0000 |
| our | 4210 | 0 | 0.0325 | 0.0688 | 0.0011 | 0.0410 | 4.6969 | 37.0648 | 0.0000 | 0.0000 | 0.0410 |
| over | 4210 | 0 | 0.0164 | 0.0469 | 0.0007 | 0.0000 | 6.1227 | 71.6033 | 0.0000 | 0.0000 | 0.0000 |
| remove | 4210 | 0 | 0.0162 | 0.0546 | 0.0008 | 0.0000 | 6.8087 | 76.3816 | 0.0000 | 0.0000 | 0.0000 |
| internet | 4210 | 0 | 0.0097 | 0.0369 | 0.0006 | 0.0000 | 9.7342 | 167.5502 | 0.0000 | 0.0000 | 0.0000 |
| order | 4210 | 0 | 0.0175 | 0.0536 | 0.0008 | 0.0000 | 5.2843 | 48.0114 | 0.0000 | 0.0000 | 0.0000 |
| mail | 4210 | 0 | 0.0137 | 0.0361 | 0.0006 | 0.0105 | 8.5686 | 162.5390 | 0.0000 | 0.0000 | 0.0105 |
| receive | 4210 | 0 | 0.0217 | 0.0706 | 0.0011 | 0.0000 | 5.0783 | 34.6561 | 0.0000 | 0.0000 | 0.0000 |
| will | 4210 | 0 | 0.0584 | 0.0913 | 0.0014 | 0.0858 | 2.8280 | 12.0961 | 0.0000 | 0.0196 | 0.0858 |
| people | 4210 | 0 | 0.0176 | 0.0557 | 0.0009 | 0.0000 | 6.8900 | 82.5963 | 0.0000 | 0.0000 | 0.0000 |
| report | 4210 | 0 | 0.0061 | 0.0346 | 0.0005 | 0.0000 | 11.5641 | 219.6452 | 0.0000 | 0.0000 | 0.0000 |
| addresses | 4210 | 0 | 0.0102 | 0.0549 | 0.0008 | 0.0000 | 7.5134 | 69.4938 | 0.0000 | 0.0000 | 0.0000 |
| free | 4210 | 0 | 0.0127 | 0.0399 | 0.0006 | 0.0064 | 9.3877 | 157.6917 | 0.0000 | 0.0000 | 0.0064 |
| business | 4210 | 0 | 0.0209 | 0.0641 | 0.0010 | 0.0000 | 5.5677 | 43.5142 | 0.0000 | 0.0000 | 0.0000 |
| email | 4210 | 0 | 0.0208 | 0.0595 | 0.0009 | 0.0000 | 5.4539 | 48.1575 | 0.0000 | 0.0000 | 0.0000 |
| you | 4210 | 0 | 0.0916 | 0.0943 | 0.0015 | 0.1451 | 1.5243 | 4.9981 | 0.0000 | 0.0725 | 0.1451 |
| credit | 4210 | 0 | 0.0047 | 0.0278 | 0.0004 | 0.0000 | 15.4302 | 424.6858 | 0.0000 | 0.0000 | 0.0000 |
| your | 4210 | 0 | 0.0729 | 0.1035 | 0.0016 | 0.1152 | 2.2457 | 8.0942 | 0.0000 | 0.0261 | 0.1152 |
| font | 4210 | 0 | 0.0077 | 0.0626 | 0.0010 | 0.0000 | 9.5415 | 99.6661 | 0.0000 | 0.0000 | 0.0000 |
| num000 | 4210 | 0 | 0.0185 | 0.0645 | 0.0010 | 0.0000 | 5.8595 | 49.1199 | 0.0000 | 0.0000 | 0.0000 |
| money | 4210 | 0 | 0.0073 | 0.0345 | 0.0005 | 0.0000 | 15.5567 | 340.1758 | 0.0000 | 0.0000 | 0.0000 |
| hp | 4210 | 0 | 0.0276 | 0.0805 | 0.0012 | 0.0047 | 5.6388 | 43.1824 | 0.0000 | 0.0000 | 0.0047 |
| hpl | 4210 | 0 | 0.0168 | 0.0544 | 0.0008 | 0.0000 | 6.2975 | 63.0729 | 0.0000 | 0.0000 | 0.0000 |
| george | 4210 | 0 | 0.0111 | 0.0533 | 0.0008 | 0.0000 | 9.4816 | 111.0998 | 0.0000 | 0.0000 | 0.0000 |
| num650 | 4210 | 0 | 0.0141 | 0.0589 | 0.0009 | 0.0000 | 6.5180 | 58.6692 | 0.0000 | 0.0000 | 0.0000 |
| lab | 4210 | 0 | 0.0069 | 0.0404 | 0.0006 | 0.0000 | 11.6836 | 189.4087 | 0.0000 | 0.0000 | 0.0000 |
| labs | 4210 | 0 | 0.0176 | 0.0748 | 0.0012 | 0.0000 | 6.3808 | 49.1053 | 0.0000 | 0.0000 | 0.0000 |
| telnet | 4210 | 0 | 0.0051 | 0.0310 | 0.0005 | 0.0000 | 13.5149 | 302.3699 | 0.0000 | 0.0000 | 0.0000 |
| num857 | 4210 | 0 | 0.0094 | 0.0632 | 0.0010 | 0.0000 | 10.4325 | 128.2958 | 0.0000 | 0.0000 | 0.0000 |
| data | 4210 | 0 | 0.0056 | 0.0313 | 0.0005 | 0.0000 | 13.0694 | 290.7362 | 0.0000 | 0.0000 | 0.0000 |
| num415 | 4210 | 0 | 0.0096 | 0.0634 | 0.0010 | 0.0000 | 10.3398 | 126.4371 | 0.0000 | 0.0000 | 0.0000 |
| num85 | 4210 | 0 | 0.0054 | 0.0265 | 0.0004 | 0.0000 | 16.0018 | 491.5416 | 0.0000 | 0.0000 | 0.0000 |
| technology | 4210 | 0 | 0.0129 | 0.0503 | 0.0008 | 0.0000 | 7.3588 | 79.9880 | 0.0000 | 0.0000 | 0.0000 |
| num1999 | 4210 | 0 | 0.0208 | 0.0616 | 0.0009 | 0.0000 | 5.1266 | 40.8880 | 0.0000 | 0.0000 | 0.0000 |
| parts | 4210 | 0 | 0.0017 | 0.0277 | 0.0004 | 0.0000 | 27.0787 | 836.2293 | 0.0000 | 0.0000 | 0.0000 |
| pm | 4210 | 0 | 0.0076 | 0.0403 | 0.0006 | 0.0000 | 11.7623 | 205.9592 | 0.0000 | 0.0000 | 0.0000 |
| direct | 4210 | 0 | 0.0129 | 0.0671 | 0.0010 | 0.0000 | 9.1018 | 101.1202 | 0.0000 | 0.0000 | 0.0000 |
| cs | 4210 | 0 | 0.0058 | 0.0446 | 0.0007 | 0.0000 | 11.4688 | 166.2617 | 0.0000 | 0.0000 | 0.0000 |
| meeting | 4210 | 0 | 0.0099 | 0.0556 | 0.0009 | 0.0000 | 9.1852 | 108.9947 | 0.0000 | 0.0000 | 0.0000 |
| original | 4210 | 0 | 0.0137 | 0.0649 | 0.0010 | 0.0000 | 7.4082 | 73.7954 | 0.0000 | 0.0000 | 0.0000 |
| project | 4210 | 0 | 0.0043 | 0.0323 | 0.0005 | 0.0000 | 18.1968 | 449.0177 | 0.0000 | 0.0000 | 0.0000 |
| re | 4210 | 0 | 0.0149 | 0.0488 | 0.0008 | 0.0079 | 8.9179 | 122.0675 | 0.0000 | 0.0000 | 0.0079 |
| edu | 4210 | 0 | 0.0086 | 0.0421 | 0.0006 | 0.0000 | 10.0770 | 150.1195 | 0.0000 | 0.0000 | 0.0000 |
| table | 4210 | 0 | 0.0027 | 0.0365 | 0.0006 | 0.0000 | 19.2985 | 431.3092 | 0.0000 | 0.0000 | 0.0000 |
| conference | 4210 | 0 | 0.0035 | 0.0299 | 0.0005 | 0.0000 | 18.8718 | 492.1469 | 0.0000 | 0.0000 | 0.0000 |
| „;” | 4210 | 0 | 0.0092 | 0.0576 | 0.0009 | 0.0000 | 13.3559 | 200.4498 | 0.0000 | 0.0000 | 0.0000 |
| „(” | 4210 | 0 | 0.0148 | 0.0281 | 0.0004 | 0.0199 | 14.0027 | 404.3701 | 0.0000 | 0.0075 | 0.0199 |
| „[” | 4210 | 0 | 0.0043 | 0.0259 | 0.0004 | 0.0000 | 21.2720 | 664.6204 | 0.0000 | 0.0000 | 0.0000 |
| „!” | 4210 | 0 | 0.0087 | 0.0260 | 0.0004 | 0.0102 | 18.3373 | 579.4385 | 0.0000 | 0.0005 | 0.0102 |
| „$” | 4210 | 0 | 0.0127 | 0.0399 | 0.0006 | 0.0088 | 10.6090 | 188.4627 | 0.0000 | 0.0000 | 0.0088 |
| „#” | 4210 | 0 | 0.0023 | 0.0220 | 0.0003 | 0.0000 | 31.5002 | 1238.3210 | 0.0000 | 0.0000 | 0.0000 |
| capitalAve | 4210 | 0 | 0.0040 | 0.0301 | 0.0005 | 0.0019 | 22.7607 | 614.3999 | 0.0006 | 0.0012 | 0.0025 |
| capitalLong | 4210 | 0 | 0.0051 | 0.0200 | 0.0003 | 0.0037 | 31.2055 | 1471.2156 | 0.0006 | 0.0014 | 0.0043 |
| capitalTotal | 4210 | 0 | 0.0183 | 0.0391 | 0.0006 | 0.0148 | 8.8016 | 146.1084 | 0.0025 | 0.0063 | 0.0172 |

W naszych danych występuje 4210 obserwacji oraz 0 wartości brakujących (tabela 5). Jeżeli chodzi o średnią, odchylenie standardowe, błąd standardowy średniej czy rozstęp międzykwartylowy (IQR) to osiągane są dość niskie wyniki dla każdej zmiennej. Podobnie możemy powiedzieć o wartościach osiąganych przez kwartyl pierwszy, drugi i trzeci. Inaczej to się ma jeśli chodzi o skośność gdzie osiągane są wysokie wyniki powyżej zera, co świadczy o prawostronnej asymetrii rozkładu zmiennych. Jeżeli chodzi o wartości kurtozy to są one największe spośród wszystkich wskaźników sumarycznych wymienionych w tabeli 5 i mówią o tym, że rozkład zmiennych jest bardziej wysmukły niż normalny (rozkład leptokurtyczny), czyli mają większe skupienie wartości wokół średniej.

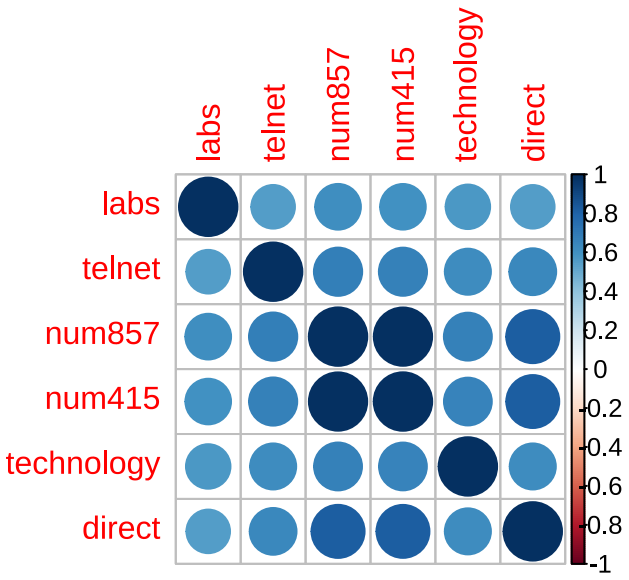
## 1.5 Analiza zależności (korelacji) między zmiennymi

W tej sekcji zajmiemy się rysowaniem podstawowych wykresów takich jak histogramy, wykresy rozrzutu, wykresy pudełkowe, macierze korelacji czy estymatory jądrowe gęstości.
Na początek tworzymy macierz korelacji, żeby zobaczyć jak prezentują się zależności między zmiennymi.



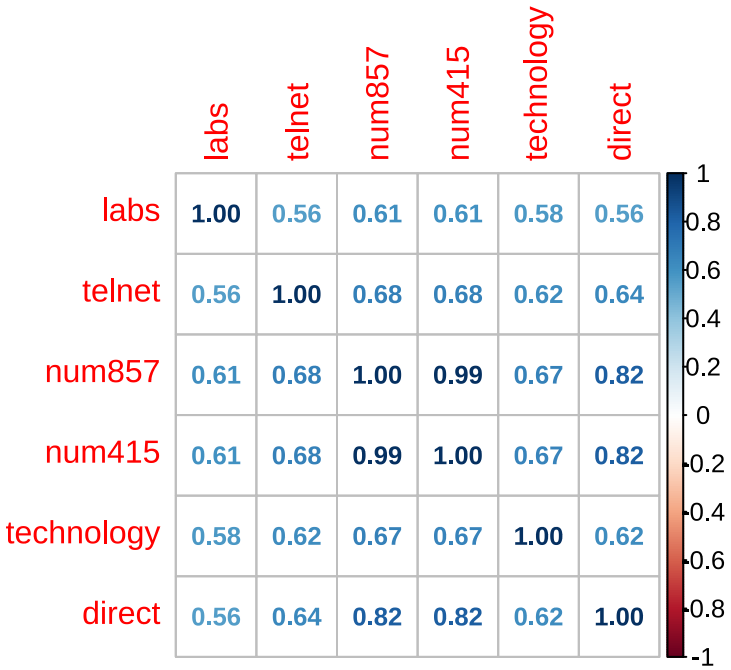

Rysunek 2: Macierz korelacji zmiennych.

Zmienne w znacznej większości nie są ze sobą powiązane (tabela 2), jednak wyodrębnimy te, dla których zależność jest wysoka. Wybierzemy wartości korelacji zmiennych, dla których wartość bezwzględna współczynnika korelacji jest wyższa niż 0.55. Wtedy otrzymamy lepszy i wyrazistszy wgląd na zmienne oraz będziemy mogli przejść do dalszej analizy.




Rysunek 3: Macierz korelacji poszczególnych zmiennych.

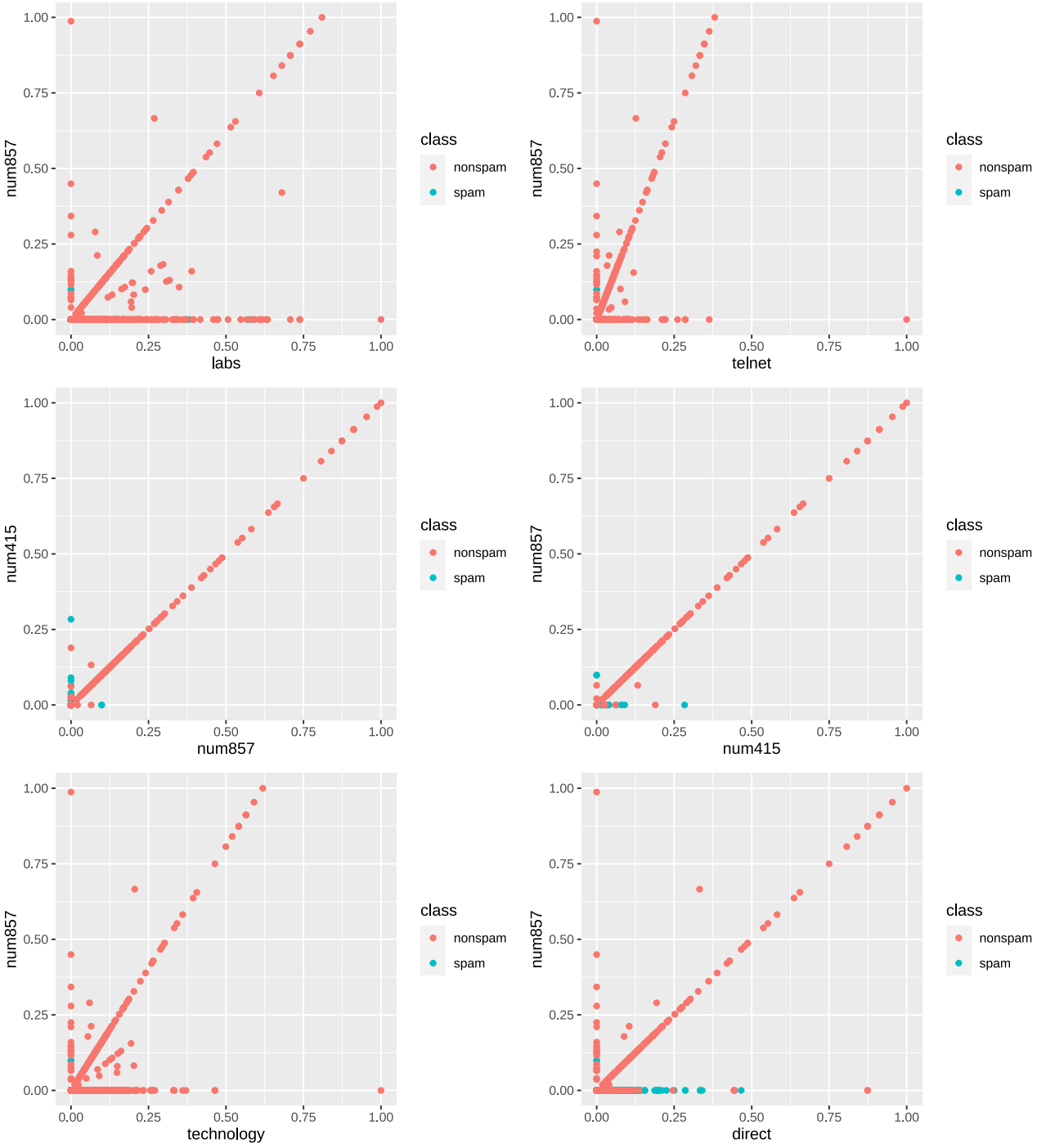
Dla lepszego zobrazowania na rysunku 4 przedstawione są dokładne wartości współczynnika korelacji, które są osiągane na rysunku 3.




Rysunek 4: Macierz korelacji poszczególnych zmiennych z widocznymi wartościami współczynnika korelacji.

Największe powiązanie występuje między zmiennymi *num857* i *num415* gdzie współczynnik korelacji wynosi prawie 1. Może to być związane z numerem telefonu, który składa się z takich cyfr, wtedy numery te tworzyłyby całość co jest równoznaczne z tym, że będą występowały razem w danym mailu lub nie. Zauważalna jest również jedynie korelacja dodatnia oznaczająca, że wraz ze wzrostem wartości jednej cechy następuje wzrost wartości drugiej.

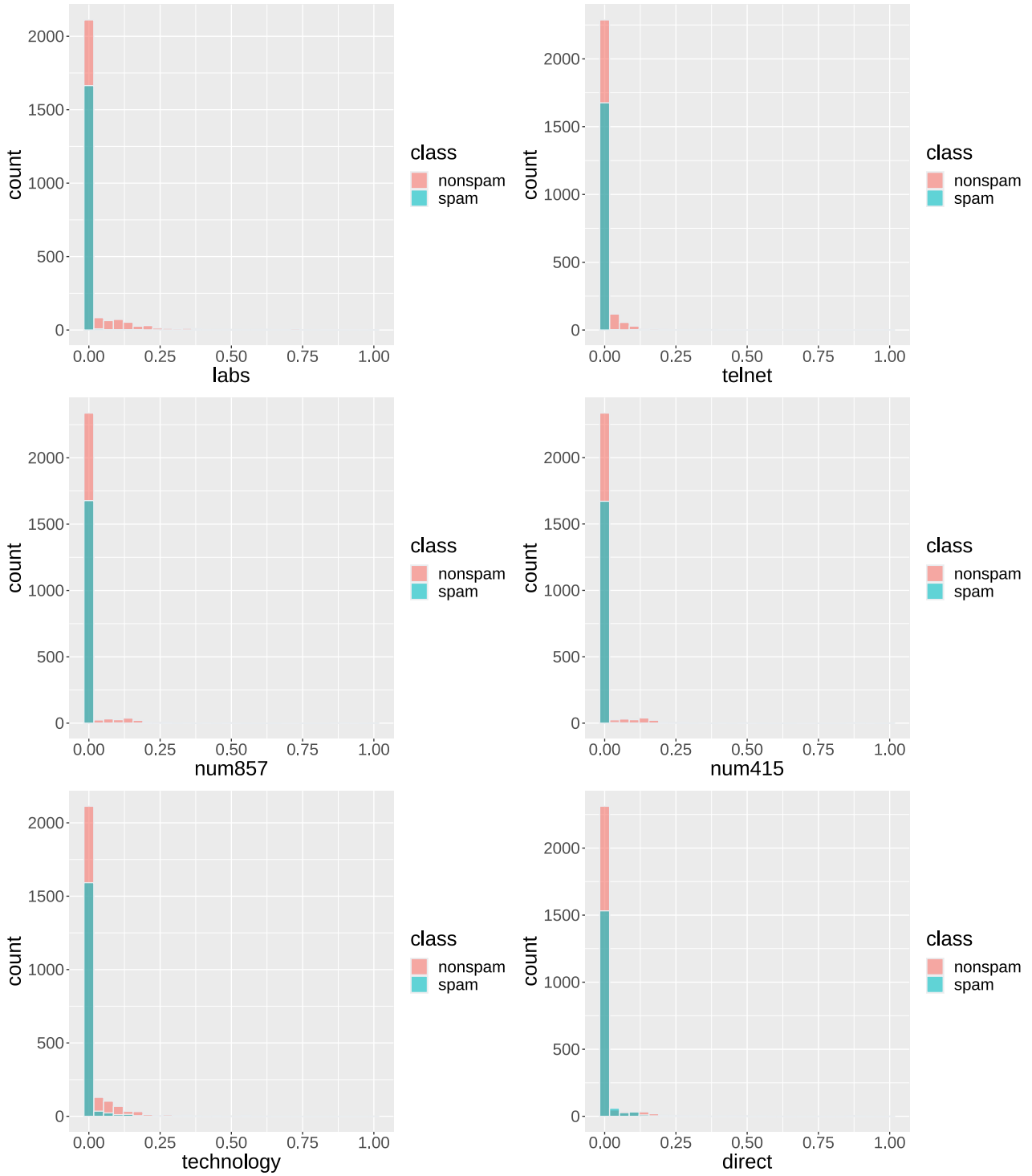
Zobaczmy jak zmienne prezentują się na skategoryzowanych wykresach rozrzutu.




Rysunek 5: Skategoryzowane wykresy rozrzutu poszczególnych zmiennych.

Widzimy na wykresach 5, że wraz ze wzrostem jednej obserwacji równomiernie rośnie druga obserwacja, co świadczy o dużej zależności, ale punkty też układają się pionowo, kiedy wartości obserwacji na osi $x$ osiągają 0 lub poziomo, kiedy wartości obserwacji na osi $y$ osiągają 0.

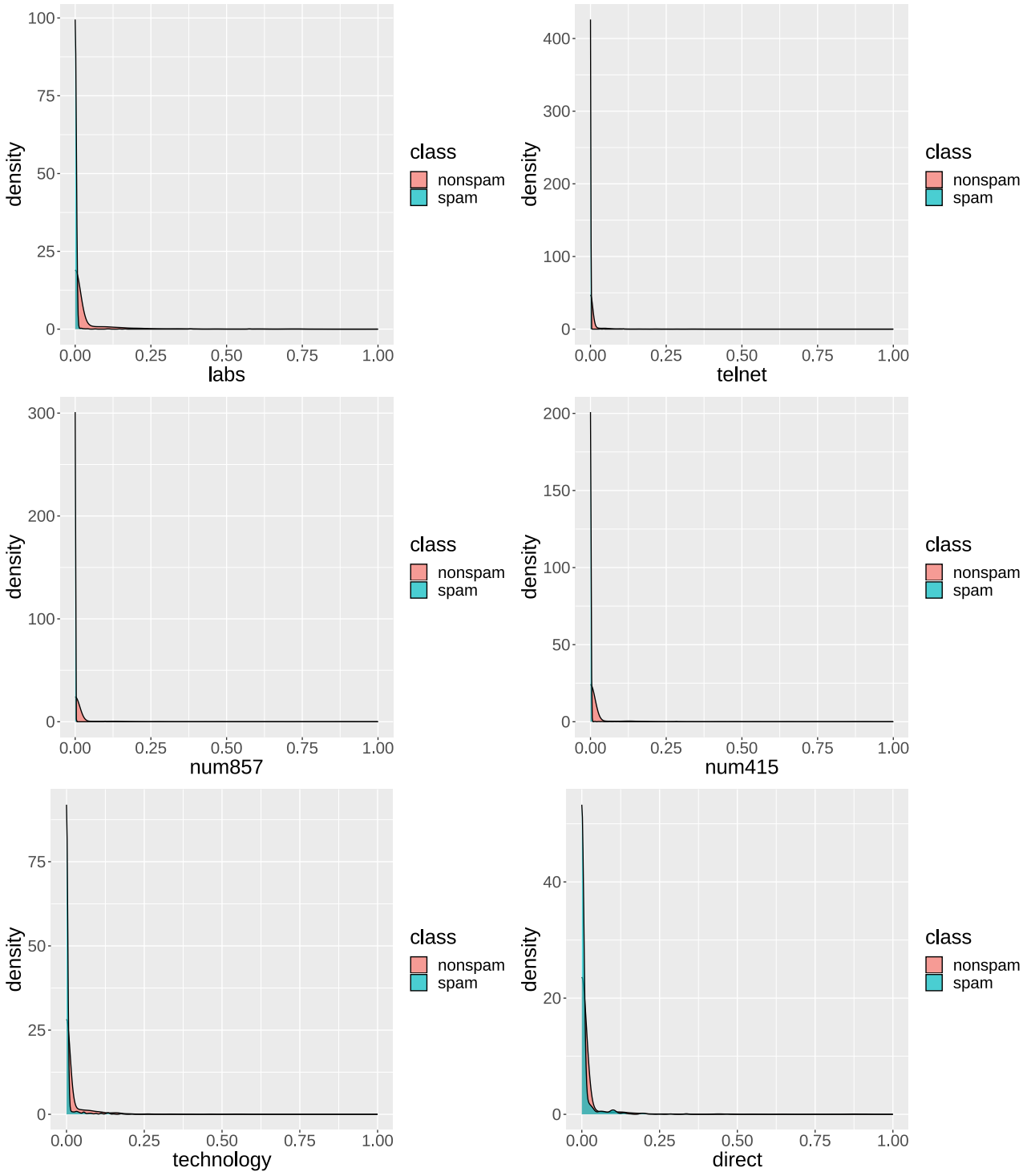
Następnie tworzymy histogramy wraz z estymatorami jądrowych gęstości dla zmiennych z najwyższym współczynnikiem korelacji, gdzie zmienną grupującą jest cecha *class*.




Rysunek 6: Histogramy dla poszczególnych zmiennych.

Widzimy, na histogramach 6, że głównie osiągane są wartości równe 0 oraz większą liczbę stanowi klasa *nonspam*.

Przechodzimy do tworzenia estymatorów jądrowych gęstości.




Rysunek 7: Estymatory jądrowe gęstości dla poszczególnych zmiennych.

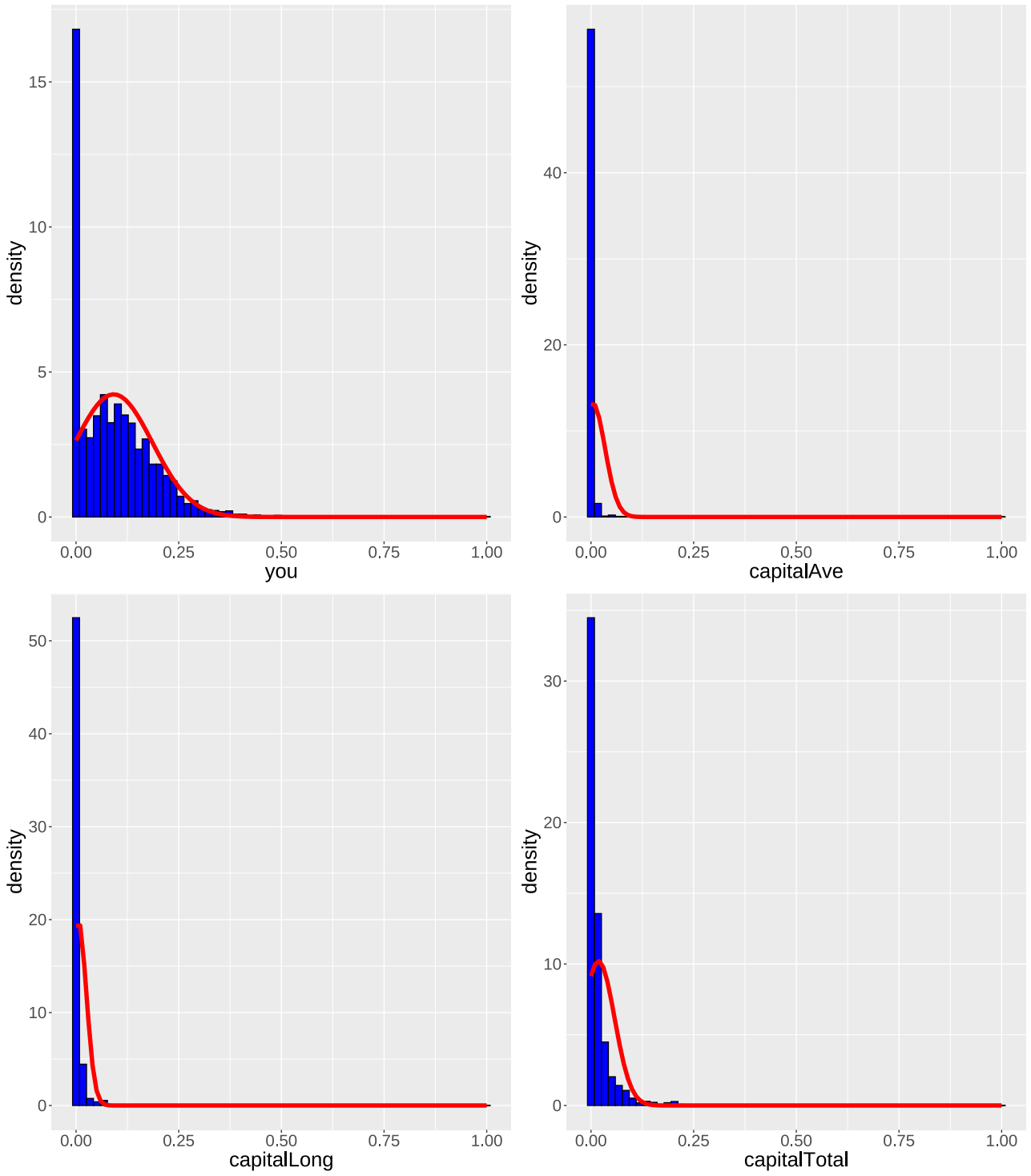
Wnioski podobne jak w przypadku histogramów, dlatego też wyniki na wykresach 6, 7 nie są zróżnicowane (kolejny raz wartości 0 stanowią większość).

Stworzymy tabelę obrazującą jak często pojawia się wartość 0 dla każdej zmiennej w zbiorze danych `CleanSpam`.

Tabela 6: Ilość występowania wartości 0 dla każdej zmiennej.

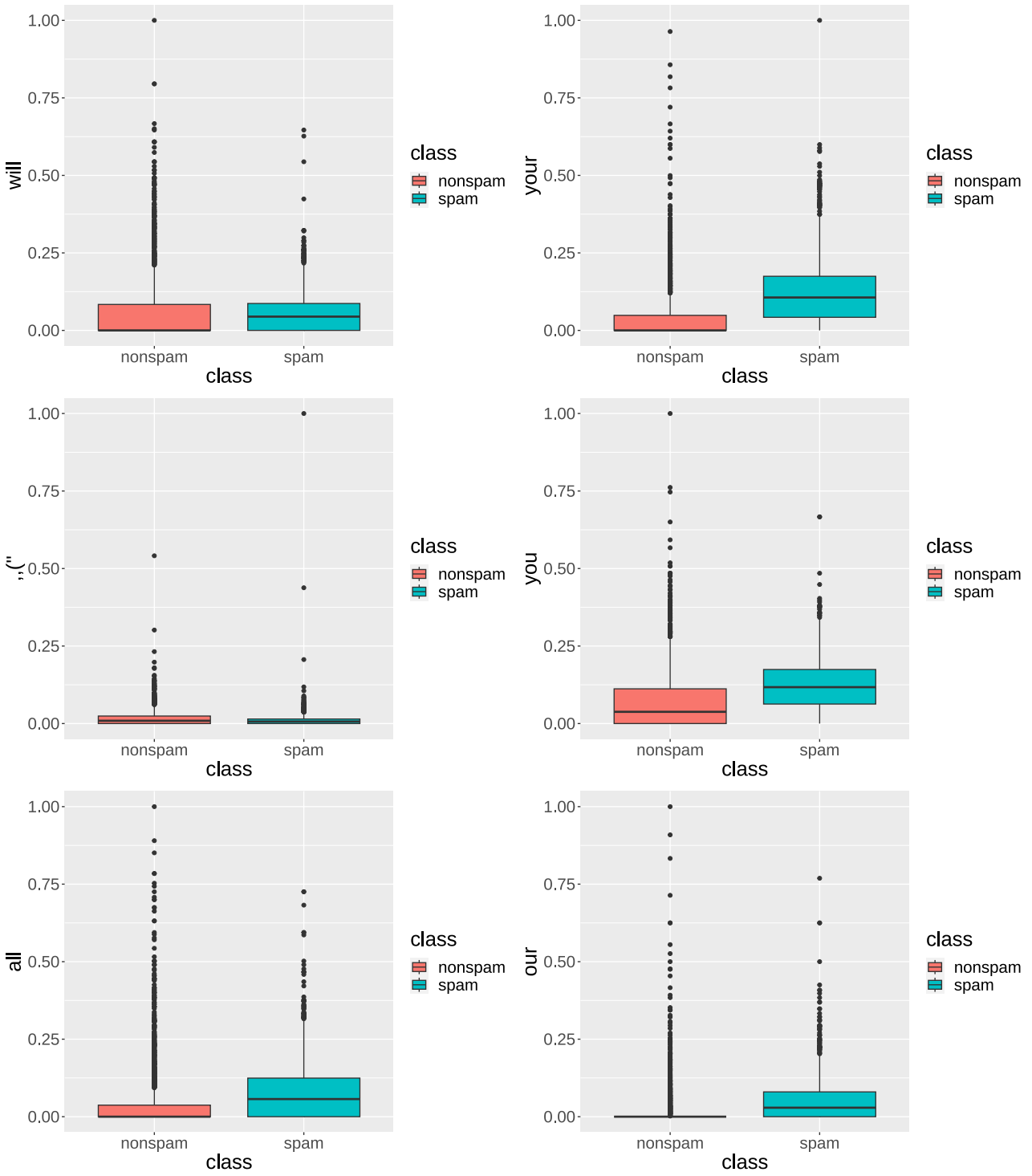
| | 0 |
|---|---|
| make | 3228 |
| address | 3399 |
| all | 2426 |
| num3d | 4164 |
| our | 2551 |
| over | 3277 |
| remove | 3448 |
| internet | 3443 |
| order | 3488 |
| mail | 2981 |
| receive | 3559 |
| will | 2009 |
| people | 3404 |
| report | 3874 |
| addresses | 3905 |
| free | 3054 |
| business | 3304 |
| email | 3238 |
| you | 1146 |
| credit | 3821 |
| your | 1925 |
| font | 4098 |
| num000 | 3591 |
| money | 3551 |
| hp | 3146 |
| hpl | 3421 |
| george | 3543 |
| num650 | 3758 |
| lab | 3851 |
| labs | 3754 |
| telnet | 3931 |
| num857 | 4014 |
| data | 3813 |
| num415 | 4004 |
| num85 | 3740 |
| technology | 3628 |
| num1999 | 3401 |
| parts | 4132 |
| pm | 3832 |
| direct | 3788 |
| cs | 4067 |
| meeting | 3875 |
| original | 3857 |
| project | 3893 |
| re | 2962 |
| edu | 3710 |
| table | 4150 |
| conference | 4009 |
| „;” | 3460 |
| „(” | 1628 |
| „[” | 3705 |
| „!” | 2068 |
| „$” | 2912 |
| „#” | 3508 |
| capitalAve | 216 |
| capitalLong | 216 |
| capitalTotal | 7 |

Najmniej razy wartość 0 występuje dla zmiennych $you = 1146$, $capitalAve = 216$, $capitalLong = 216$ i $capitalTotal = 7$ (tabela 6). Wykonamy dla tych zmiennych histogramy wraz z dopasowaną gęstością normalną.




Rysunek 8: Histogramy wraz z dopasowaną gęstością normalną.

Widzimy, że wartości wciąż są bliskie zeru i są prawoskośnie asymetryczne. Zobaczymy jeszcze czy występują jakiekolwiek wartości odstające stosując wykresy pudełkowe dla zmiennych z najmniejszą liczbą zer, między innymi *will*, *„(”*, *your*, *you*, *all* i *our*,.




Rysunek 9: Wykresy pudełkowe dla poszczególnych zmiennych.

Na rysunku 9 widać, że istnieje dużo wartości odstających i występują one głównie dla klasy *nonspam*. Tylko dla powyższych zmiennych wykresy pudełkowe są „widoczne", a dla pozostałych kontury pudełek wysmuklają się (mniej więcej tak jak w przypadku wykresu pudełkowego dla zmiennej „(", tylko oczywiście w mniejszym stopniu) na poziomie równym 0 i nie są widoczne.

Na sam koniec przejdziemy do usunięcia z naszych danych zmiennych które są ze sobą najbardziej skorelowane, czyli *labs, telnet, num857, num415, technology* i *direct*.

```
CleanSpam <- CleanSpam[,-c(30:32,34,36,40)]
colnames(CleanSpam)
```

```
##  [1] "make"          "address"       "all"           "num3d"         "our"
##  [6] "over"          "remove"        "internet"      "order"         "mail"
## [11] "receive"       "will"          "people"        "report"        "addresses"
## [16] "free"          "business"      "email"         "you"           "credit"
## [21] "your"          "font"          "num000"        "money"         "hp"
## [26] "hpl"           "george"        "num650"        "lab"           "data"
## [31] "num85"         "num1999"       "parts"         "pm"            "cs"
## [36] "meeting"       "original"      "project"       "re"            "edu"
## [41] "table"         "conference"    ",,;''"         ",,('"          ",,['"
## [46] ",,!''"         ",,$''"         ",,#''"         "capitalAve"    "capitalLong"
## [51] "capitalTotal" "class"
```

# 2 Klasyfikacja

Klasyfikacja to proces przypisywania obiektów do jednej z klas na podstawie ich cech. W naszym przypadku klasyfikacja będzie polegać na przewidywaniu, czy dany e-mail jest spamem czy nie. Do tego celu wykorzystamy parę metod klasyfikacji takich jak regresja liniowa, analiza dyskryminacyjna czy algorytm kNN. Naszym celem będzie wybranie metody, która radzi sobie najlepiej z ocenianiem prawdopodobieństwa przynależności do danej klasy. Aby to ocenić użyjemy różnych miar jakości takich jak dokładność, czy krzywa ROC. Pierwszym krokiem będzie podział danych na zbiór uczący i testowy w proporcji 70:30, ponieważ będzie nam to potrzebne do metod klasyfikacji.

```
set.seed(123456)
classify_data <- CleanSpam

train_indices <- sample(nrow(classify_data), 0.7 * nrow(classify_data))
train_data <- classify_data[train_indices, ]
test_data <- classify_data[-train_indices, ]
```

## 2.1 Klasyfikacja oparta na regresji liniowej

$$Y = r(x) + \varepsilon = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \varepsilon, \tag{1}$$

W naszym przypadku użyjemy modelu regresji liniowej 1, aby wyznaczyć klasyfikator postaci

$$\hat{G} = \begin{cases} \text{SPAM} \quad \text{jeżeli} \quad \hat{Y} > 0.5, \\ \text{NONSPAM} \quad \text{jeżeli} \quad \hat{Y} \leqslant 0.5, \end{cases}$$

gdzie $\hat{Y} = r(X) = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$ oznacza prognozę zmiennej zależnej $Y$, która jest wektorem zawierającym etykietki klas. Dla uproszczenia obliczeń użyjemy wbudowanej funkcji `lm()` z pakietu `stats`. W tym celu zmienimy etykietki naszych klas na 0 oraz 1, ponieważ mamy do czynienia z klasyfikacją binarną ($K = 2$).

```
train_data_lm$class <- ifelse(train_data$class == "spam", 1, 0)
test_data_lm$class <- ifelse(test_data$class == "spam", 1, 0)
model <- lm(class ~ ., data = train_data_lm)
predictions <- predict(model, newdata = test_data_lm)
threshold <- 0.5
predicted_labels <- ifelse(predictions > threshold, 1, 0)
```

Tabela 7: Macierz kotyngencji dla regresji liniowej.

| | nonspam | spam |
|---|---|---|
| nonspam | 722 | 124 |
| spam | 34 | 383 |

```
## Accuracy: 0.874901
```

## 2.2 Klasyfikacja oparta na metodzie LDA (liniowa analiza dyskryminacyjna)

Do przeprowadzenia liniowej analizy dyskryminacyjnej użyjemy funkcji `lda()` z pakietu `MASS`. Biorąc pod uwagę model składający się ze wszystkich zmiennych oczekujemy, że dostaniemy równoważny wynik jak dla regresji liniowej, ponieważ mamy klasyfikację binarną.

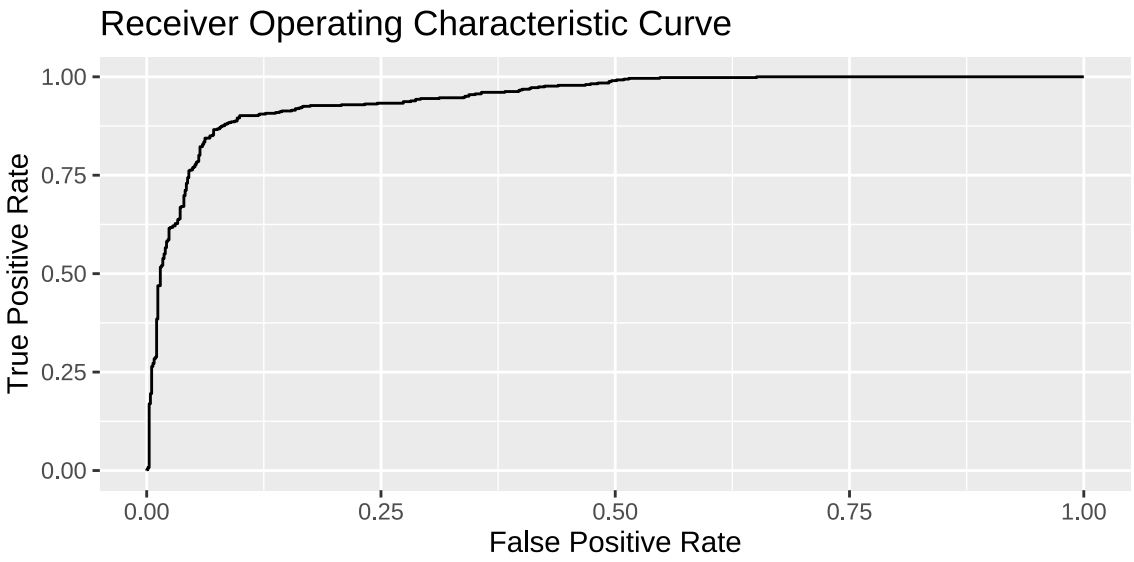
```
lda.model <- lda(class ~ ., data = classify_data, subset=train_indices)
lda.pred <- predict(lda.model, newdata = test_data)
```

Tabela 8: Macierz kontyngencji dla metody lda.

| | nonspam | spam |
|---|---|---|
| nonspam | 722 | 123 |
| spam | 34 | 384 |

```
## Accuracy: 0.8756928
```

Z uwagi na to, że rozkład klas nie jest idealnie równomierny (tabela 3) możemy narysować wykres krzywej ROC umożliwiający ocenę jakości klasyfikacji.




Rysunek 10: Wykres krzywej ROC dla metody lda

Zastosowana klasyfikacja z użyciem metody lda radzi sobie dość dobrze, ponieważ powierzchnia pod krzywą ROC 10 zajmuje duży obszar, co świadczy o wysokiej jakości klasyfikacji.

## 2.3 Klasyfikacja z wykorzystaniem algorytmu kNN (Classification And Regression Training)

Na początek zajmiemy się klasyfikacją z wykorzystaniem algorytmu k najbliższych sąsiadów za pomocą funkcji `knn` wbudowanej w bibliotece *class*. W klasyfikacji wykorzystamy parametr liczby sąsiadów $k = 1$.

```
#etykiety klas
TrainLabels <- train_data[, 52]
TestLabels <- test_data[, 52]

# rzeczywiste klasy spamu
etykietki.rzecz <- test_data$class

#prognoza
etykietki.prog <- knn(train = train_data[, 1:51],
                      test = test_data[, 1:51],
                      cl = TrainLabels, k = 1)
```

Tabela 9: Macierz kontyngencji dla metody knn.

| | nonspam | spam |
|---|---|---|
| nonspam | 678 | 82 |
| spam | 78 | 425 |

```
## Accuracy: 0.8733175
```

Widzimy w macierzy kontyngencji 9, że zmienne zostały przyzwoicie sklasyfikowane o czym świadczy wysoka, prawie 90% dokładność. W tabeli 10 zobaczymy jak przedstawiają się wartości prawdziwie negatywne, pozytywne oraz dokładność jeżeli zwiększymy wartości parametru $k$.

Tabela 10: Wartości prawdziwie negatywne i prawdziwie pozytywne oraz dokładność, dla różnych k.

| k | True Negatives | True Positives | Accuracy |
|---|---|---|---|
| 5 | 703 | 416 | 0.8860 |
| 10 | 713 | 400 | 0.8812 |
| 15 | 708 | 388 | 0.8678 |
| 20 | 708 | 380 | 0.8614 |
| 25 | 709 | 376 | 0.8591 |

Widzimy, że wartości prawdziwie negatywne początkowo wzrastają, a później się stabilizują, natomiast wartości prawdziwie pozytywne i dokładność maleją. Oznacza to, że im większa wartość parametru liczby sąsiadów $k$ tym mniejsza dokładność klasyfikacji.

Przechodzimy do klasyfikacji z wykorzystaniem funkcji `ipredknn` z pakietu *ipred* dla parametru liczby sąsiadów $k = 1$.

```
# budujemy model
model.knn.1 <- ipredknn(class ~ ., data=train_data, k=1)

#jakość modelu
etykietki.prog.ipred <- predict(model.knn.1,test_data,type="class")
```

Tabela 11: Macierz kontyngencji dla metody knn.

| | nonspam | spam |
|---|---|---|
| nonspam | 678 | 82 |
| spam | 78 | 425 |

```
## Accuracy: 0.8733175
```

Po macierzy kontyngencji 11 można zobaczyć że wyniki są bardzo podobne jak w macierzy 9, gdzie wykorzystywaliśmy funkcję `knn` z pakietu *class*. W przypadku dokładności wyniki również są do siebie zbliżone.

## 2.4 Inne (zaawansowane) sposoby oceny jakości klasyfikacji

Często stosuje sie metodę cross-validation polegającą na wielokrotnym losowaniu zbioru uczącego i testowego, budowie klasyfikatora na zbiorze uczącym, sprawdzenia go na testowym oraz uśrednieniu wyników. Można to zrobić oczywiście używając pętli i uśrednić, ale jest jeszcze prostszy sposób.
Można skorzystać z gotowych funkcji ponownie z pakietu *ipred*, ale należy przygotować sobie „wrapper" dostosowujący funkcję `predict` dla naszego modelu do standardu wymaganego przez `errorest`.

```
my.predict  <- function(model, newdata)
  predict(model, newdata=newdata, type="class")
my.ipredknn <- function(formula1, data1, ile.sasiadow)
  ipredknn(formula=formula1,data=data1,k=ile.sasiadow)
```

Przejdziemy do porównania błędów klasyfikacji następujących metod oceny dokładności klasyfikacji:

- CV (Cross-validation)
- boot (bootstrap)
- 632plus

```r
# porownanie błędów klasyfikacji: cv, boot, .632plus
errorest(type ~., spam, model=my.ipredknn, predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10), ile.sasiadow=5)

##
## Call:
## errorest.data.frame(formula = type ~ ., data = spam, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 5)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.1937

errorest(type ~., spam, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=5)

##
## Call:
## errorest.data.frame(formula = type ~ ., data = spam, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##     ile.sasiadow = 5)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.219
## Standard deviation: 0.0012

errorest(type ~., spam, model=my.ipredknn, predict=my.predict, estimator="632plus",
         est.para=control.errorest(nboot = 50), ile.sasiadow=5)

##
## Call:
## errorest.data.frame(formula = type ~ ., data = spam, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot =
##     ile.sasiadow = 5)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.1906
```

W tabeli 12 pokazane są wartości wskaźnika accuracy dla każdej z trzech metod oceny dokładności klasyfikacji.

Tabela 12: Wartości wskaźnika accuracy dla różnych metod oceny dokłądności klasyfikacji.

| Cross-validation | Bootstrap | 632plus |
|---|---|---|
| 0.806 | 0.781 | 0.8094 |

Zważywszy na dość niskie wyniki wartości wskaźnika accuracy, powyższe sposoby oceny dokładności nie będą brane pod uwagę w końcowym porównaniu wszystkich metod klasyfikacji.

**Podsumowanie**

W tym rozdziale przeprowadzono klasyfikację na zbiorze danych przy użyciu różnych metod, takich jak regresja liniowa, LDA i kNN. Wszystkie metody zostały ocenione na podstawie ich skuteczności, mierzonej za pomocą wskaźnika accuracy. Najwyższe wyniki (tabela 13) uzyskano dla metody kNN dla liczby sąsiadów równej 5, osiągając skuteczność na poziomie prawie 90%, co wskazuje na jej wysoką zdolność do poprawnej klasyfikacji obiektów. Tak jak podejrzewaliśmy metody LDA oraz regresji liniowej osiągnęły taki sam poziom accuracy, który był również wysoki.

Tabela 13: Wartości wskaźnika accuracy dla różnych metod klasyfikacji.

| Reg. liniowa | Metoda LDA | Algorytm kNN (k=1) | Algorytm kNN (k=5) | Algorytm kNN (funk. ipredknn) |
|---|---|---|---|---|
| 0.8749 | 0.8757 | 0.8733 | 0.886 | 0.8733 |

Wskaźnik accuracy (tabela 13) dla wszystkich metod jest do siebie zbliżony, dlatego możemy wnioskować, że wszystkie metody klasyfikacji są w stanie skutecznie ocenić klasę danej obserwacji dla danych *Spambase*.