# Filtrowanie spamu

Adrian Bukowski Anna Mikołajczyk

14 czerwca 2018

## Spis treści

**1 Wstęp oraz analiza opisowa** **3**

**2 Metody** **11**

**3 Klasyfikacja** **12**
3.1 Regresja Logistyczna . . . 13
3.2 LDA . . . 13
3.3 Klasyfikator Naiwny Baysa . . . 13
3.4 Metoda k-Najbliższych Sąsiadów . . . 14
3.5 Lasy losowe . . . 14
3.6 Maszyna wektorów nośnych . . . 15
3.7 XGBoost . . . 16
3.8 Sieci neuronowe . . . 17
3.9 Porównanie . . . 17

**4 Redukcja wymiaru** **18**
4.1 PCA . . . 18

**5 Klasyfikacja z redukcją wymiaru** **20**
5.1 Regresja Logistyczna . . . 20
5.2 LDA . . . 21
5.3 Klasyfikator Naiwny Bayesa . . . 22
5.4 Metoda k-Najbliższych Sąsiadów . . . 23
5.5 Lasy losowe . . . 24
5.6 Maszyna wektorów nośnych . . . 25
5.7 XGBoost . . . 26
5.8 Sieci neuronowe . . . 27
5.9 Porównanie . . . 28

**6 Analiza skupień** **29**
6.1 Metody grupujące . . . 29
6.1.1 K-means . . . 29
6.1.2 PAM . . . 30
6.2 Metody chierarchiczne . . . 32
6.2.1 Wskaźniki wewnętrzne . . . 33
6.2.2 Wskaźniki zewnętrzne . . . 35

**7 Podsumowanie** **37**

# 1 Wstęp oraz analiza opisowa

Przedmiotem naszych rozważań są dane dotyczące zawartości 4601 wiadomości e-mail. Celem analiz jest filtrowanie spamu, tzn. rozróżnienie spamu od pożadanych wiadomości. Mamy do czynienia z 57 zmiennymi, spośród których:

- 48 przyjmuje wartości od 0 do 100 i oznacza procentowy udział danego słowa w mailu (za słowo uznajemy dowolny ciąg liter i cyfr)
- 6 przyjmuje wartości od 0 do 100 i oznacza procentowy udział danego znaku w mailu
- 1 przyjmuje wartości dodatnie i oznacza średnią długość nieprzerywanych ciągów wielkich liter
- 1 przyjmuje wartości naturalne i oznacza długość najdłuższego nieprzerwanego ciągu wielkich liter
- 1 przyjmuje wartości naturalne i oznacza łączna ilość wielkich liter w mailu

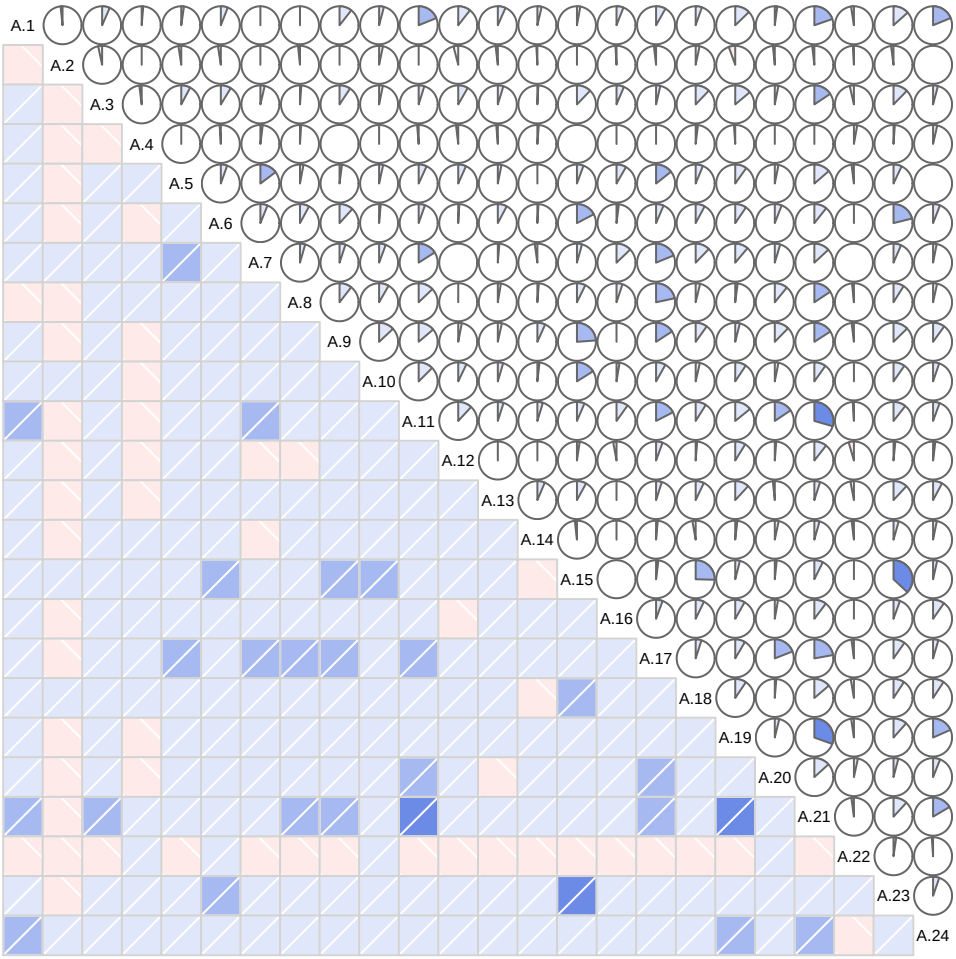
Oprócz tego dane zawierają kolumnę spam/mail oznaczającą czy dany mail jest spamem. Mamy więc do czynienia z zadaniem klasyfikacji. Przyjrzyjmy się danym.

```
data("spam")
dim(spam)

## [1] 4601   58

X<-spam[,1:(dim(spam)[2]-1)]
y<-spam[,dim(spam)[2]]
```

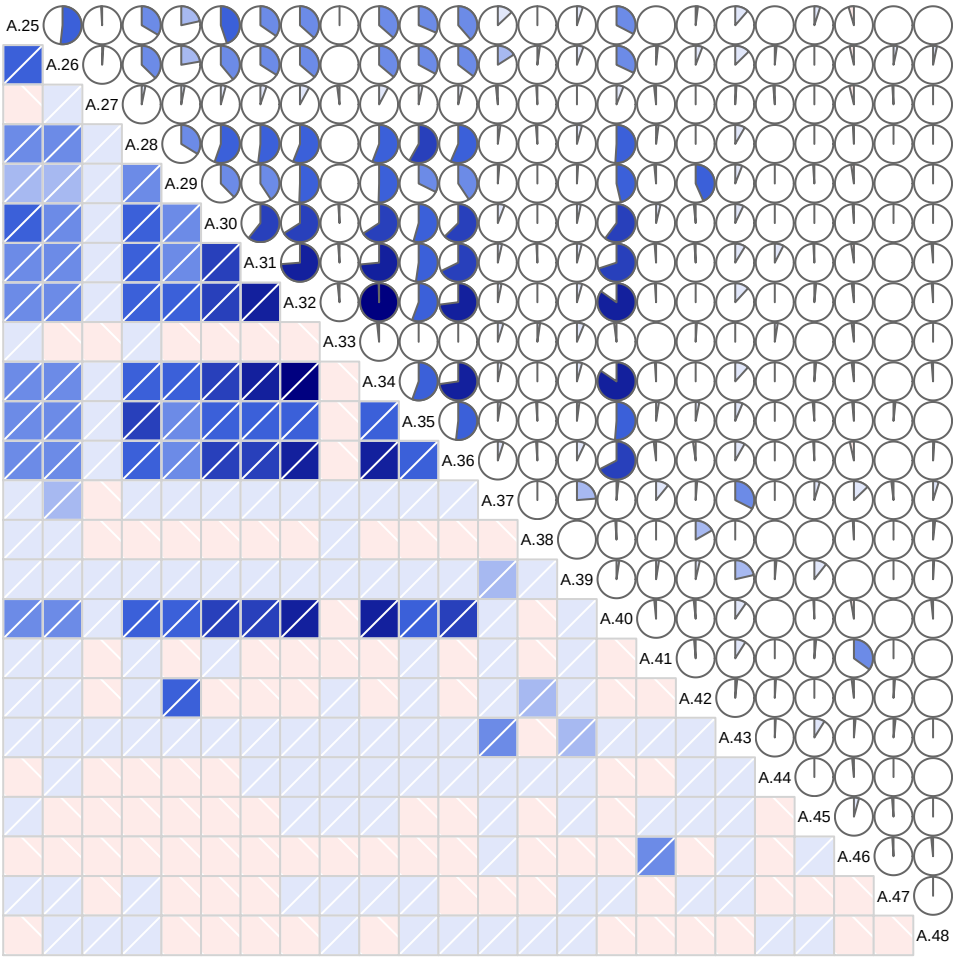
Przed wykonaniem analiz sprawdziliśmy czy dane są dobrze wczytane, zmienne okazały się zgodne z opisem. Upewniliśmy się również, że nie występują brakujące dane.

Na początku analiz przyjrzeliśmy się poszczególnym zmiennym, wyznaczając podstawowe statystyki opisowe (w związku z dużą ilością zmiennych nie umieszczamy ich w raporcie) i kowariancje pomiędzy nimi (prezentujemy wynik w postaci tzw. heatmap, czyli mapy ciepła).

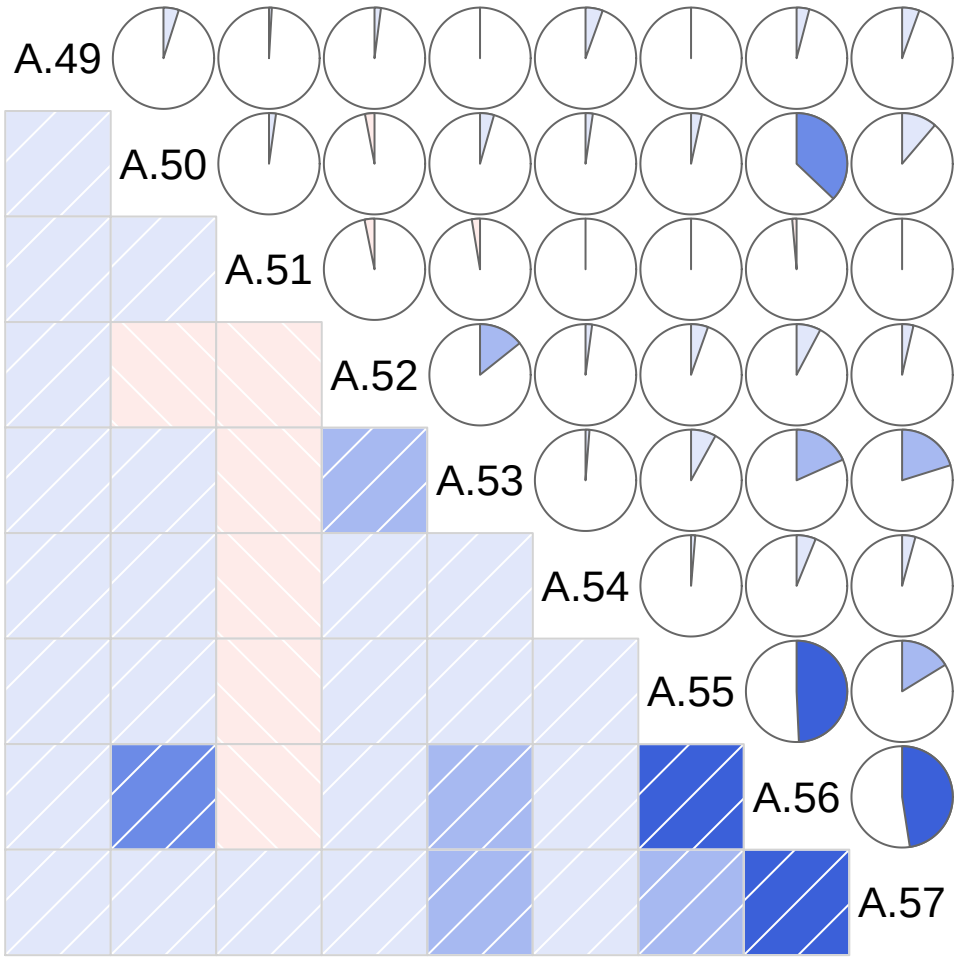
Na początku analiz przyjrzeliśmy się poszczególnym zmiennym, wyznaczając podstawowe statystyki opisowe (w związku z dużą ilością zmiennych nie umieszczamy ich w raporcie) i korelacje pomiędzy nimi. Najpierw na zbiorze wszystkich zmiennych, jednak tak duży wykres korelacji okazał się być bardzo nieczytelny, dlatego też zrobiliśmy trzy osobne zestawienia dotyczące 3 podgrup, kierując się opisem poszczególnych zmiennych. Na poniższych wykresach im głębszy kolor, tym silniejsza korelacja między zmiennymi.




Rysunek 1: Korrelogram dla zmiennych związanych z występowaniem słów




Rysunek 2: Korrelogram dla zmiennych związanych z występowaniem słów C.D.




Rysunek 3: Korrelogram zmiennych związanych z występowaniem znaków i wielkich liter

Analiza wieloczynnikowa pokazuje, że prawie żadne zmienne nie są ze sobą skorelowane. Pojawiają się jednak pary, jak na przykład A34 i A32, dla których korelacja przekracza nawet 75%. Zweryfikowaliśmy czy usunięcie tych pojedynczych zmiennych wpływa istotnie na wyniki. Ponieważ nie zaobserwowaliśmy poprawy, postanowiliśmy analizować kompletny zbiór danych "spam".

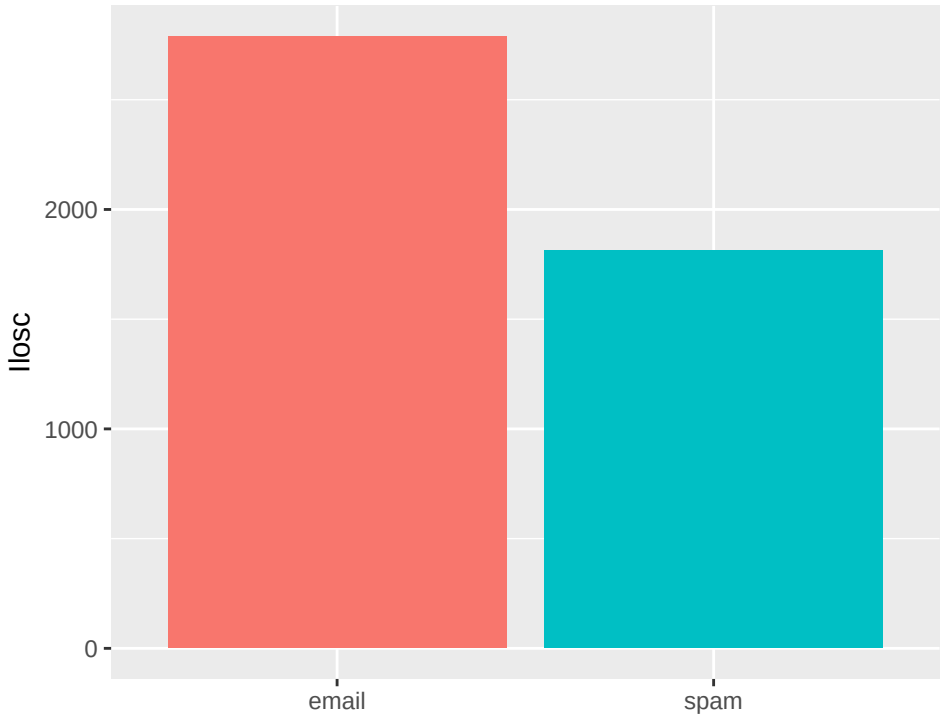
Następnie skupiliśmy sie na zmiennej objaśnianej

```
class(y)

## [1] "factor"

levels(y)

## [1] "email" "spam"
```

Jest to zmienna typu "factor" o dwóch poziomach - "email" i "spam".

W kolejnym kroku, zbadaliśmy jak dzielą się dane, którymi dysponujemy, ze względu na zmienną objaśnianą. Rozkład możemy obserwować na poniższym wykresie słupkowym.

```
## y
## email  spam
##  2788  1813
##       [,1]
## email 2788
## spam  1813
```



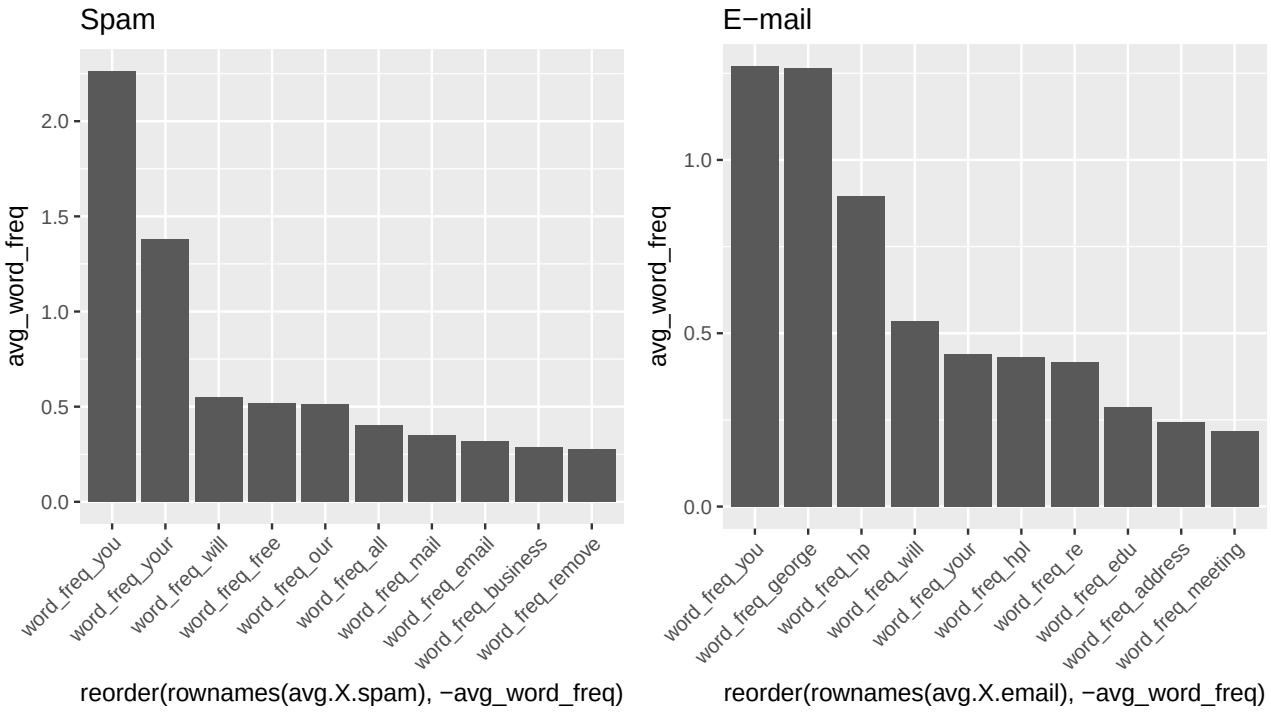

Rysunek 4: Rozkład wiadomości na spam i pożądane e-maile

```
## y
##     email      spam
## 0.6059552 0.3940448
```

Okazało się, że około 60% obserwacji stanowią pożądane e-maile, a 40% spam.

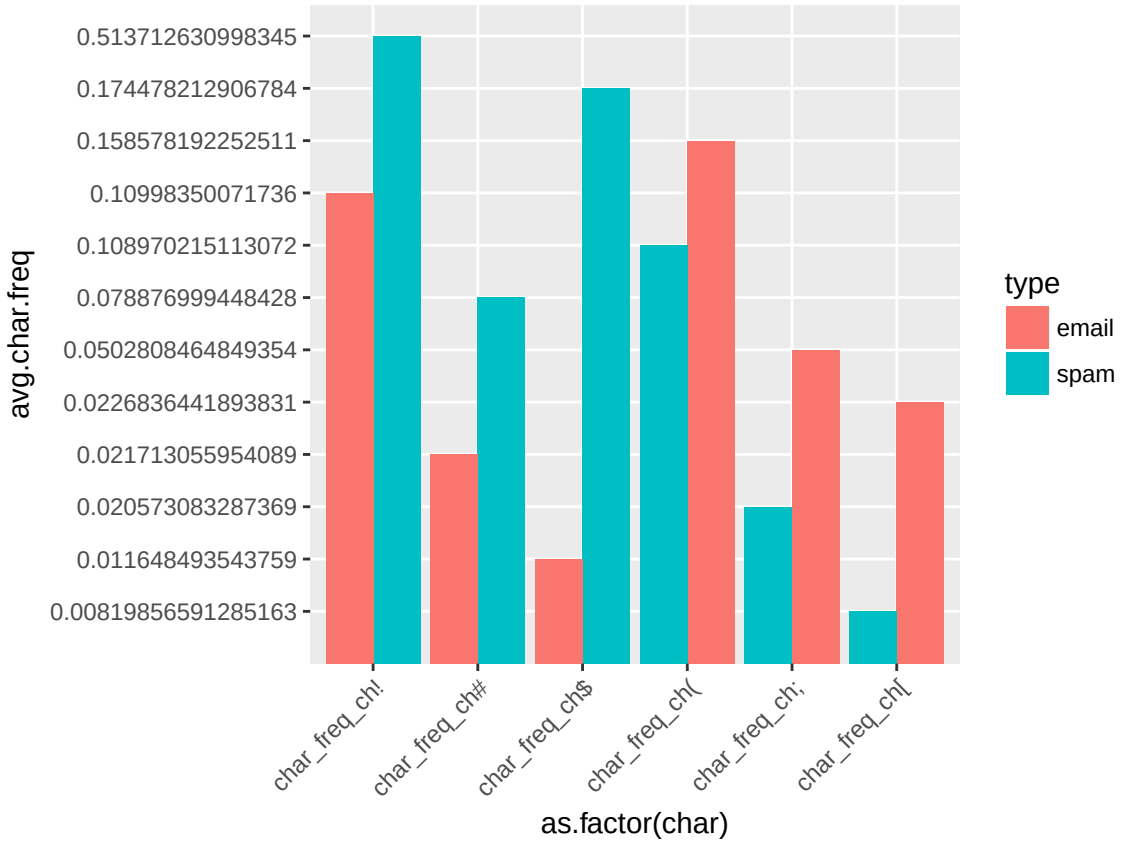
Posiłkując się dokumentacją danych, aby nasze analizy były bardziej świadome, dodaliśmy nazwy kolumn.

W celu wyłapania różnic pomiędzy obserwacjami które są spamem, a tymi które nie są, podzieliliśmy całą próbkę na dwie rozłączne. Chcieliśmy również sprawdzić czym wyróżniają się wiadomości będące spamem, a czym pozostałe. Poniższe wykresy przedstawiają, które słowa pojawiają się z największą średnią częstotliwością we wspomnianych grupach.



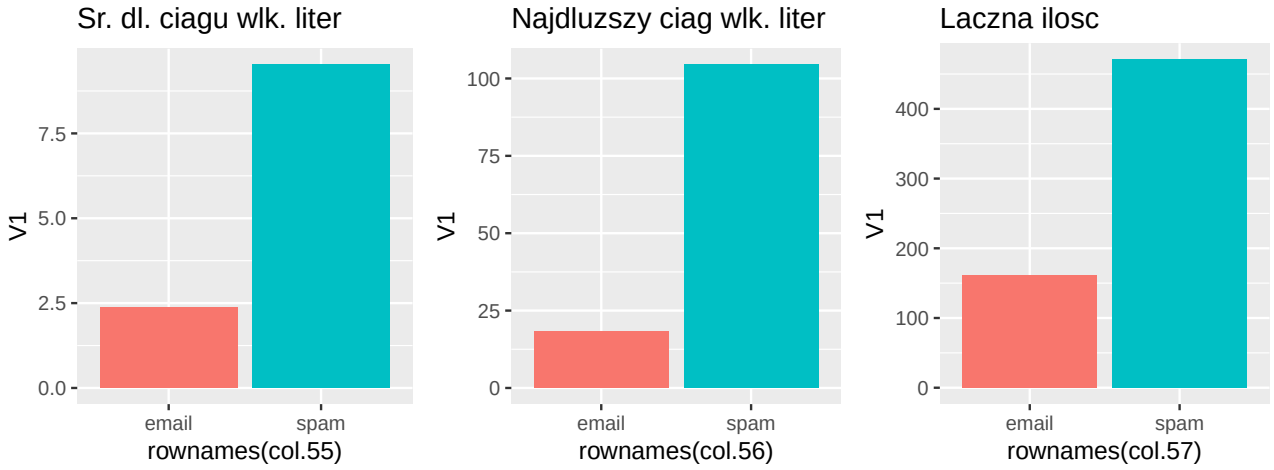

Rysunek 5: Największe średnie częstotliwości pojawiania się słów wśród spamu i wiadomości pożądanych

Łatwo zauważyć, że najczęściej mamy do czynienia ze słowem "you", jednak wystepuje ono w obu grupach, zatem prawdopodobnie, nie różnicuje ich we właściwy sposób. Drugie w kolejności, jeśli chodzi o średni procentowy udział słowa we wiadomości, w przypadku spamu, jest słowo "your", a w przypadku e-maili słowa "George" i "hp". Z kolejnych wykresów, możemy odczytać jakie znaki stanowią średnio największy procentowy udział w wiadomości, w przypadkach spamu i zwykłych e-maili.




Rysunek 6: Największe średnie częstotliwości pojawiania się znaków w spamie i pożądanych wiadomościach

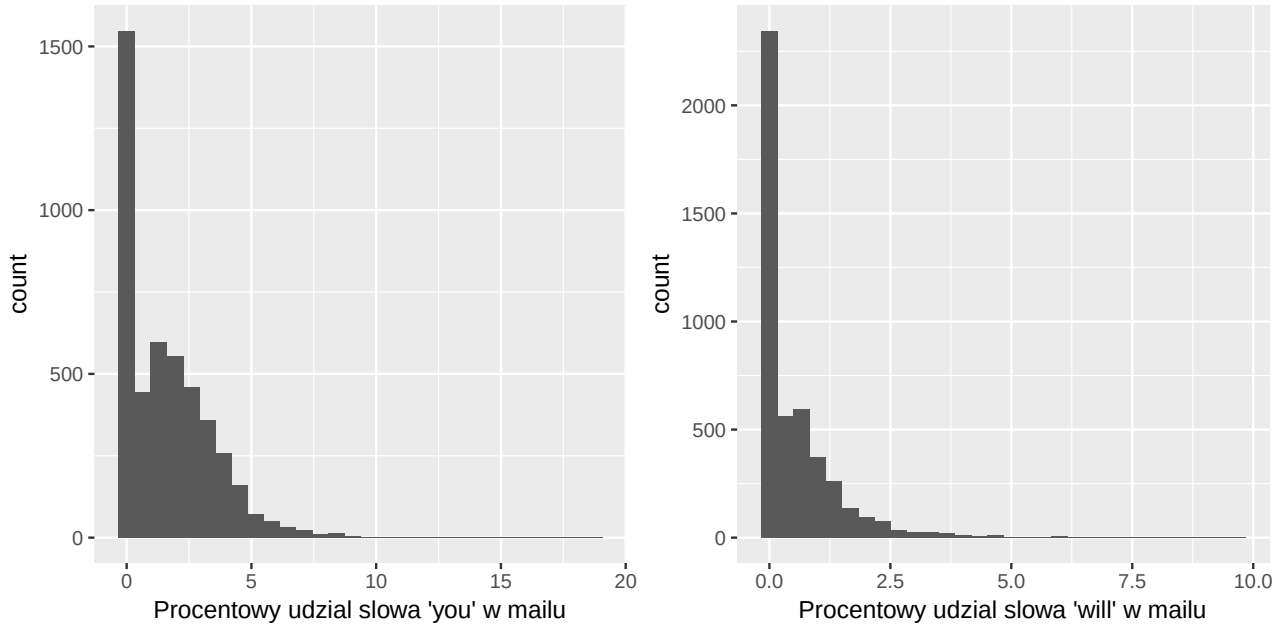
Widzimy, że średnio w spamie, większy procentowy udział mają znaki !, #, $, a w e-mailach ), ], ;. Można przypuszczać, że występowanie znaków $ i ] jest istotne w rozpoznawaniu spamu, ponieważ w ich przypadku obserwujemy największą różnicę pomiędzy średnim procentowym udziałem w spamie i w e-mailu. Kończąc wstępne analizy, zajęliśmy się jeszcze 3 ostatnimi zmiennymi, czyli średnią długością nieprzerwanych ciągów wielkich liter, długością najdłuższego nieprzerywanego ciągu wielkich liter oraz łączną ilością wielkich liter w mailu.




Rysunek 7: Analiza wielkich liter w spamie i w pożądanych wiadomościach

Powyższe wykresy sugerują, że wielkie litery znacznie większą rolę odgrywają w spamie. Przyjrzyjmy się histogramom częstości występowania niektórych słów:

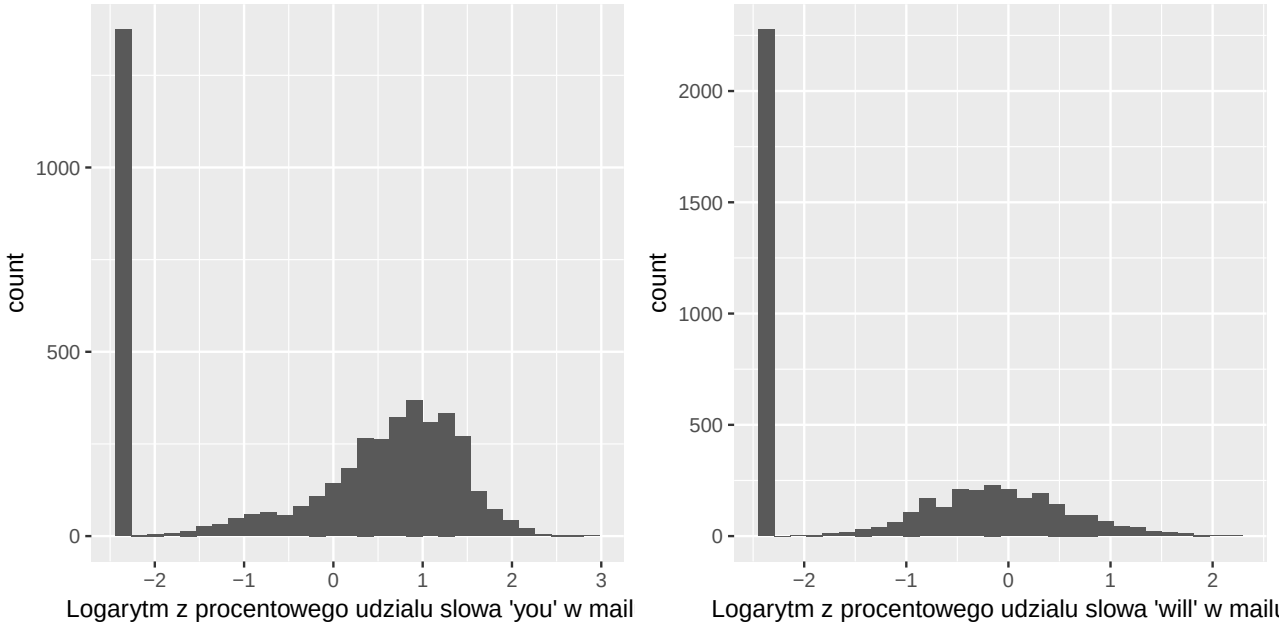
```
q1<-qplot(X$word_freq_you, geom="histogram") +
    xlab("Procentowy udział słowa 'you' w mailu")
q2<-qplot(X$word_freq_will, geom="histogram")+
    xlab("Procentowy udział słowa 'will' w mailu")
multiplot( q1,q2, cols=2)
```




Rysunek 8: Rozkład przykładowych zmiennych

Charakter rozkładów jest dyskretno-ciągły z atomem w 0. Dodatkowo dane nie przypominają rozkładu normalnego i są ciężkoogonowe. Warto zatem rozważyć transformację logarytmiczną. Ze względu na wspomniane wartości zerowe rozważymy jednak transformację $\log(x + 0.1)$.

```
q1<-qplot(log(X$word_freq_you+0.1), geom="histogram") +
    xlab("Logarytm z procentowego udziału słowa 'you' w mailu")
q2<-qplot(log(X$word_freq_will+0.1), geom="histogram") +
    xlab("Logarytm z procentowego udziału słowa 'will' w mailu")
multiplot( q1,q2, cols=2)
```




Rysunek 9: Rozkład przykładowych zmiennych po transformacji

Oczywiście dalej mamy do czynienia z atomem w punkcie log(0.1), jednak pozostałe dane dużo bardziej przypominają rozkład normalny. Transformację zastosujemy do metod LDA i LR (w pozostałych metodach pogorszała ona wyniki). Po wstępnej analizie przechodzimy do bardziej złożonych rozważań związanych z postawionym problemem.

# 2 Metody

Projekt skupia się na zagadnieniach klasyfikacji i analizy skupień na przykładzie zbioru danych "spam". Celem projektu jest porównanie skuteczności metod klasyfikacji oraz jakości metod analizy skupień. W raporcie rozpatrzyliśmy te same metody z uwzględnieniem redukcji wymiaru za pomocą metody PCA. W przypadku klasyfikacji wykorzystaliśmy następujące metody:

- Regresja Logistyczna
- LDA
- Klasyfikator Naiwny Bayesa
- Metoda k Najbliższych Sąsiadów
- Lasy Losowe
- SVM
- XGBoost

Do porównania skuteczności klasyfikacji wykorzystaliśmy wskaźniki:

- Skuteczność (accuracy)
- ROC

- AUC
- Czułość
- Specyficzność

Do analizy skupień zastosowaliśmy:

- k-Means
- PAM
- AGNES
- DIANA

Do analizy jakości grupowania wykorzystaliśmy wskaźniki zarówno wewnętrzne jak i zewnętrzne:

- Silhouette
- wskaźnik Dunn'a
- wskaźnik Randa
- wskaźnik Jaccarda
- wskaźnik Fowlkesa-Mallowsa

# 3 Klasyfikacja

Zajmiemy się teraz zagadnieniem klasyfikacji. Wykorzystamy pakiet mlr, który umożliwia zastosowanie różnych metod w szybki sposób. Zaczynamy od stworzenia zadania klasyfikacji. Dostrajanie parametróW i badanie skuteczności zostanie wykonane za pomocą walidacji krzyżowej na 80% danych, a następnie na pozostałych 20% sprawdzimy uzyskaną skuteczność.

```
spam.log<-spam
spam.log[,-dim(spam)[2]]<-log(spam[,-dim(spam)[2]]+0.1)
smp<- sample(dim(spam)[1],dim(spam)[1]*0.2)
test.log<- spam.log[smp,]
train.log<-spam.log[-smp,]
test <-spam[smp,]
train<-spam[-smp,]
task.log <- makeClassifTask(data = train.log, target = "spam")
task<-makeClassifTask(data=train, target = "spam")
data("spam")
```

W kolejnym roku napisaliśmy funkcję do rysowania krzywej ROC.

```
ROC<-function(pred.prob,true.labels){
  pred.ROCR <- ROCR::prediction(pred.prob, true.labels)
  perf.ROCR <- ROCR::performance(pred.ROCR, "tpr", "fpr")
  plot(perf.ROCR, print.cutoffs.at=seq(0.1,1,0.1), colorize=TRUE, lwd=2)
}
```

W kolejnych podrozdziałach wyznaczymy klasyfikatory za pomocą regresji logistycznej (RL), LDA, klasyfikatora Naiwnego Bayes'a, metody kNN, lasów losowych, SVM oraz XGBoost. Dla wszystkich klasyfikatorów zastosowana zostanie walidacja krzyżowa z 5 podzbiorami. W celu porównania dopasowania porównamy dokładność i AUC.

## 3.1 Regresja Logistyczna

Zaczynamy od jednej z najpopularniejszych metod klasyfikacji, czyli regresji logistycznej. Wykorzystamy dane po przekształceniu.

```
logistic.learner <- makeLearner("classif.logreg",predict.type = "prob")
cv.logistic <- crossval(learner = logistic.learner, task = task.log, iters = 5,
                        stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                        show.info = F)
cv.logistic$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9386057     0.9813923     0.9551515     0.9125408
```

Jak widać już zwykła regresja logistyczna dobrze odróżnia nasze dane - dokładność wynosi prawie 94%. Po rozważeniu wszystkich metod, przedstawimy dla nich krzywe ROC na jednym wykresie.

## 3.2 LDA

Kolejnym klasyfikator otrzymamy za pomocą metody LDA. Podobnie jak w regresjii logistycznej wykorzystamy dane po przekształceniu.

```
lda.learner <- makeLearner("classif.lda",predict.type = "prob")
cv.lda <- crossval(learner = lda.learner, task = task.log, iters = 5,
                   stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                   show.info = F)
cv.lda$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9348039     0.9774768     0.9666972     0.8845442
```

Otrzymujemy niewiele gorszy wynik niż przy użyciu regresji logistycznej.

## 3.3 Klasyfikator Naiwny Baysa

Następnym klasyfikatorem będzie klasyfikator naiwny Bayesa.

```
bayes.learner <- makeLearner("classif.naiveBayes", predict.type = 'prob')
cv.bayes <- crossval(learner = bayes.learner, task = task, iters = 5,
                     stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                     show.info = F)
cv.bayes$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.6957392     0.8897204     0.5395289     0.9419311
```

Klasyfikator Bayesa poradził sobie z tym zadaniem najgorzej z dotychczasowych klasyfikatorów.

## 3.4 Metoda k-Najbliższych Sąsiadów

Metoda k-Najbliższych Sąsiadów jest następną z rozważanych metod. W celu doboru liczby sąsiadów wykorzystamy metodę 'Grid search' na siatce od 1 do 10 sąsiadów.

```
knn_params <- makeParamSet(
  makeDiscreteParam("k", values = c(1,2,3,4,5,6,7,8,9,10))
)
ctrl = makeTuneControlGrid()
rdesc = makeResampleDesc("CV", iters = 3L)
tuned_params = tuneParams("classif.kknn", task = task, resampling = rdesc,
  par.set = knn_params, control = ctrl)
knn.learner <- makeLearner("classif.kknn", predict.type = 'prob',
                           par.vals = tuned_params$x)
cv.knn <- crossval(learner = knn.learner, task = task, iters = 5,
                   stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                   show.info = F)
cv.knn$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9043766     0.9558139     0.9285124     0.8663379
```

Algorytm wybrał model z 8 sąsiadami. Uzyskuje ona dokładność na poziomie niecałych 92%, zatem wynik niewiele gorszy od regresji logistycznej. Ponieważ wykorzystaliśmy te same dane do strojenia parametru oraz ewaluacji modelu, powinniśmy zatem sprawdzić jeszcze jak model radzi sobie na nowych danych (najlepiej wykorzystując walidację krzyżową), ponieważ jednak 'podwójna' walidacja krzyżowa jest bardzo kosztowna obliczeniowo (zwłaszcza dla metod typu lasy losowe/xgboost/sieci neuronowe), zatem dla metody kNN jak i następnych sprawdzimy skuteczność na jednym zbiorze testowym. Wynik na zbiorze testowym:

```
## [1] 0.9130435
```

Jest on podobny do wyniku uzyskanego metodą walidacji krzyżowej na zbiorze treningowym, co czyni model wiarygodnym.

## 3.5 Lasy losowe

Przyjrzyjmy się również klasyfikacji za pomocą lasów losowych. W celu doboru parametrów: liczby drzew, liczby zmiennych branych pod uwagę przy każdym podziale, maksymalnej liczby liści i minimalnej liczby zmiennych w liściu, wykorzystamy metodę 'Random search' z odpowiednimi przedziałami początkowymi.

```
randForest_params <- makeParamSet(
  makeIntegerParam("ntree", lower = 100, upper = 500),
  makeIntegerParam("mtry", lower = 1, upper = 10),
  makeIntegerParam("maxnodes", lower = 5, upper = 50),
```

```
  makeIntegerParam("nodesize", lower = 1, upper = 15)
)
ctrl = makeTuneControlRandom(maxit = 10)
rdesc = makeResampleDesc("CV", iters = 3L)
tuned_params = tuneParams("classif.randomForest", task = task,
                          resampling = rdesc, par.set = randForest_params,
                          control = ctrl)

randForest.learner <- makeLearner("classif.randomForest",
                                  predict.type = 'prob',
                                  par.vals = tuned_params$x)
cv.randForest <- crossval(learner = randForest.learner, task = task, iters = 5,
                          stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                          show.info = F)
cv.randForest$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9315402     0.9719327     0.9689116     0.8726365
```

Po wybraniu odpowiednich parametrów model uzyskał skuteczność ponad 93%. Zobaczmy jak wygląda wynik na zbiorze testowym:

```
mod <- train(randForest.learner, task= task)
test.pred<-predict(mod,newdata = test)
randForest.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
randForest.test.acc

## [1] 0.9402174
```

Wynik na zbiorze testowym wynosi 0.9402174, zatem jest on podobny jak w zbiorze treningowym. Poniżej przedstawiono wykres krzywej ROC.

## 3.6 Maszyna wektorów nośnych

Kolejnym z rozważanych klasyfikatorów jest SVM z jądrem gaussowskim. W celu odpowiedniego dobrania parametrów $C$ i $\gamma$ zastosujemy metodę 'Random search', czyli losowe przeszukiwanie w którym podajemy odpowiedni zakres do przeszukania.

```
svm_params <- makeParamSet(
  makeNumericParam("C", lower = -1, upper = 10, trafo = function(x) 2^x),
  makeNumericParam("epsilon", lower = -1, upper = 10, trafo = function(x) 2^x)
)
ctrl = makeTuneControlRandom(maxit = 10)
rdesc = makeResampleDesc("CV", iters = 3L)
tuned_params = tuneParams("classif.ksvm", task = task, resampling = rdesc,
  par.set = svm_params, control = ctrl)
svm.learner <- setHyperPars(makeLearner("classif.ksvm", predict.type = 'prob'),
                            par.vals = tuned_params$x)
```

```
cv.svm <- crossval(learner = svm.learner, task = task, iters = 5,
                   stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                   show.info = F)
cv.svm$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9247519     0.9722175     0.9498280     0.8852460
```

Otrzymaliśmy wynik ponad 93%. Dla parametrów $C$ i $\gamma$ odpowiednio 3.8040847 i 0.5861861. Zobaczmy jaki wynik otrzymamy na zbiorze testowym:

```
mod <- train(svm.learner, task= task)
test.pred<-predict(mod,newdata = test)
svm.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
svm.test.acc

## [1] 0.9456522
```

## 3.7 XGBoost

Jednym z najskuteczniejszych algorytmów, wygrywających w konkursach pozyskiwania wiedzy na stronach typu Kaggle jest znany algorytm XGBoost (eXtreme Gradient Boosting). Jest to przykład algorytmu wykorzystującego boosting. Algorytm ze względu na złożoność posiada wiele hiperparametrów, które wybierzemy metodą 'Random search' podając tylko granice poszukiwań.

```
xgb_params <- makeParamSet(
  makeIntegerParam("nrounds", lower = 100, upper = 500),
  makeIntegerParam("max_depth", lower = 1, upper = 10),
  makeNumericParam("eta", lower = .1, upper = .5),
  makeNumericParam("lambda", lower = -1, upper = 0, trafo = function(x) 10^x)
)
control <- makeTuneControlRandom(maxit = 10)
resample_desc <- makeResampleDesc("CV", iters = 3)
tuned_params <- tuneParams('classif.xgboost', task = task,
                           resampling = resample_desc,par.set = xgb_params,
control = control)
xgboost.learner <- makeLearner("classif.xgboost", predict.type = 'prob',
                               par.vals = tuned_params$x)
cv.xgboost <- crossval(learner = xgboost.learner, task = task, iters = 5,
                       stratif=TRUE, measures = list(acc,mlr::auc,tpr,tnr),
                     show.info = F)
cv.xgboost$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9481100     0.9856109     0.9631446     0.9244142
```

Jeden z obecnie najskuteczniejszych algorytmów, podobnie na tym zbiorze uzyskał bardzo dobry wynik ponad 95%. Zobaczmy jaki wynik uzyskał na zbiorze testowym.

```
mod <- train(xgboost.learner, task= task)
test.pred<-predict(mod,newdata = test)
xgboost.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
xgboost.test.acc

## [1] 0.9543478
```

## 3.8 Sieci neuronowe

Ostatnią wśród metod klasyfikacji przedstawionych w projekcie są sieci neuronowe. Charakteryzują się one dużą złożonością obliczeniową i małą interpretowalnością, zazwyczaj jednak dobrą lub bardzo dobrą skutecznością.

```
nn_params <- makeParamSet(
  makeIntegerParam("size", lower = 5, upper = 10),
  makeNumericParam("decay", lower=0,upper=5)
)
control <- makeTuneControlRandom(maxit = 10)
resample_desc <- makeResampleDesc("CV", iters = 3)
nn_learner <- makeLearner("classif.nnet", predict.type = 'prob', trace=FALSE)
tuned_params <- tuneParams(nn_learner, task = task,resampling = resample_desc,par.set =
nn_learner <- makeLearner("classif.nnet", predict.type = 'prob', par.vals = tuned_params
nn_learner<-setHyperPars(nn_learner,trace=FALSE)
cv.nn <- crossval(learner = nn_learner, task = task, iters = 5, stratif=TRUE, measures =

cv.nn$aggr

## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9388745     0.9799757     0.9480473     0.9244215
```

Zobaczmy jak dla optymalnych parametrów, tzn. 9 neuronów nasza sieć sprawdzi się na nowych danych.

```
mod <- train(nn_learner, task= task)
test.pred<-predict(mod,newdata = test)
nn.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
nn.test.acc

## [1] 0.9478261
```

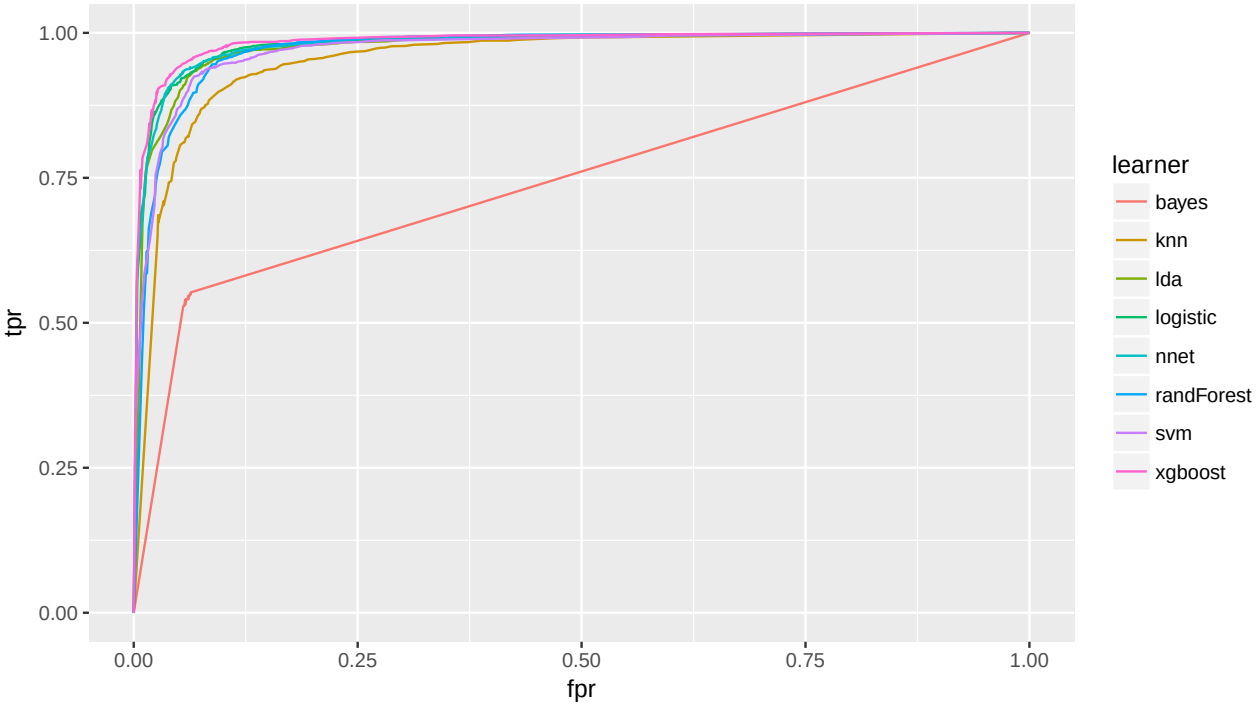
## 3.9 Porównanie

Najskuteczniejszą metodą okazał się XGBoost, jednak znacznie prostsze metody takie jak regresja logistyczna, która ponadto umożliwia interpretację modelu, dają niemalże identyczne wyniki. Najmniej skuteczną metodą jest klasyfikator naiwny Bayesa.

Poniżej przedstawiono porównanie krzywych ROC dla wszystkich klasyfikatorów.

| | Skuteczność | AUC | Czułość | Specyficzność |
|---|---|---|---|---|
| Regresja Logistyczna | 0.94 | 0.98 | 0.96 | 0.91 |
| LDA | 0.93 | 0.98 | 0.97 | 0.88 |
| Naiwny Bayes | 0.70 | 0.89 | 0.54 | 0.94 |
| kNN | 0.90 | 0.96 | 0.93 | 0.87 |
| Lasy Losowe | 0.93 | 0.97 | 0.97 | 0.87 |
| Maszyna Wektorów Nośnych | 0.92 | 0.97 | 0.95 | 0.89 |
| XGBoost | 0.95 | 0.99 | 0.96 | 0.92 |
| Sieci neuronowe | 0.94 | 0.98 | 0.95 | 0.92 |

Tabela 1: Porównanie skuteczności metod klasyfikacji dla danych "spam"



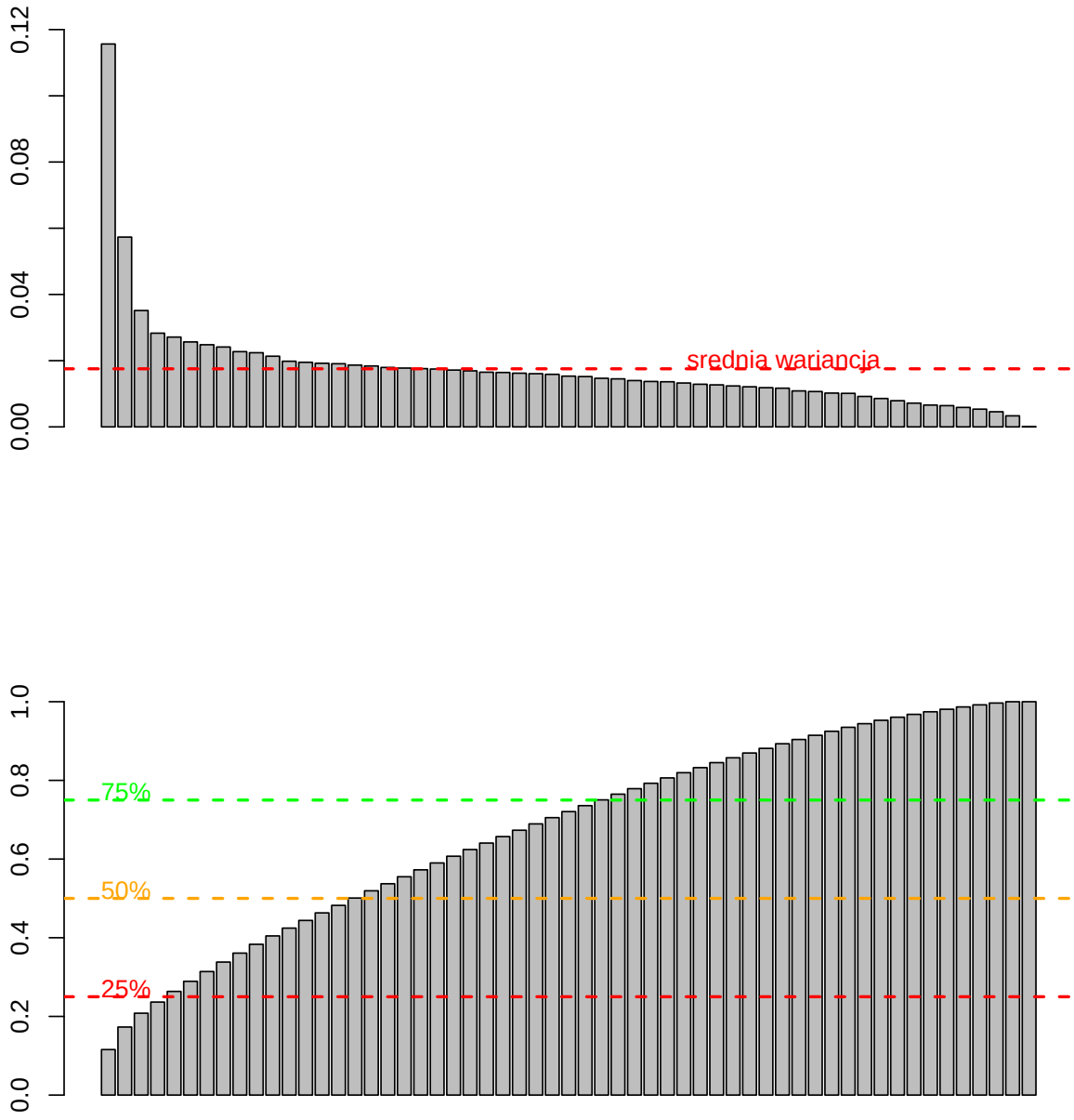

Rysunek 10: Porównanie krzywych ROC dla różnych klasyfikatorów

# 4 Redukcja wymiaru

Ponieważ dysponujemy danymi z 58 zmiennymi, konieczna wydaje się być redukcja wymiaru.

## 4.1 PCA

W tym celu wykorzystamy metodę PCA. Poniżej prezentujemy wykres osypiskowy i wykres skumulowanej wariancji, dzięki którym możemy zaobserwować jaki udział w wyjaśnianiu zmienności, mają kolejne składowe główne.

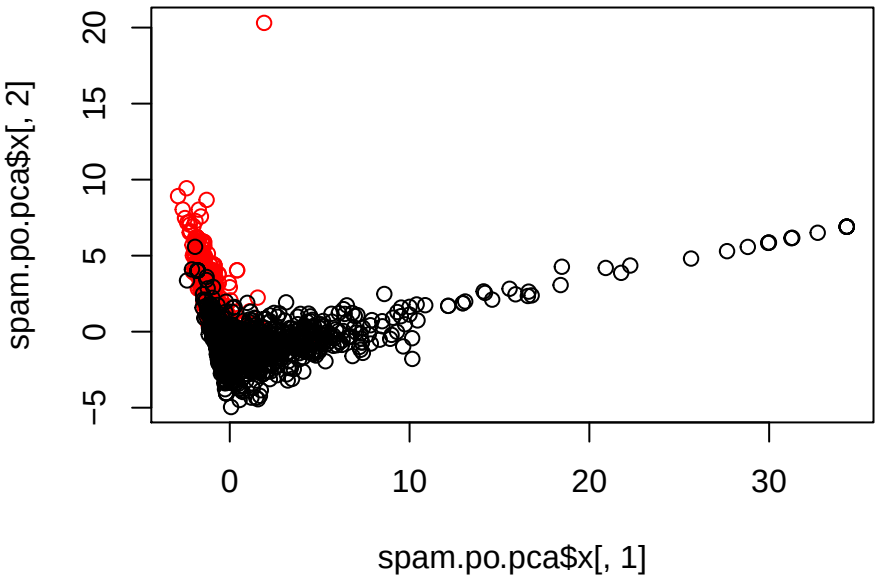



Rysunek 11: Analiza zmienności wyjaśnianej przez kolejne składowe główne

Przy tak dużej ilości zmiennych, stawianie sobie progu np. 90% wyjaśnianej zmienności, zaprowadziło by nas do dalszego rozważanie niemalże wszystkich zmiennych. Jednak nie taki był nasz cel, musimy zatem w inny sposób ustalić, ile składowych głównych brać pod uwagę. W tym celu wyznaczyliśmy wartość średnią wariancji i zaobserwowaliśmy, że każda z 20 pierwszych składowych głównych, wyjaśnia więcej niż średnia wartość zmienności. Jednak weryfikując ten wynik, wykorzystując ocenę wizualną, łatwo zauważyć, że w przypadku składowych 12-20 jest to niewielka różnica, zatem rozsądnym wyborem wydaje się być 11 składowych głównych.

Pierwsze dwie składowe główne wyjaśniają zaledwie nieco ponad 17% zmienności. Poniżej

prezentujemy wykres rozrzutu, w przestrzeni tych dwóch składowych.




Rysunek 12: Wykres rozrzutu w przestrzeni dwóch pierwszych składowych głównych

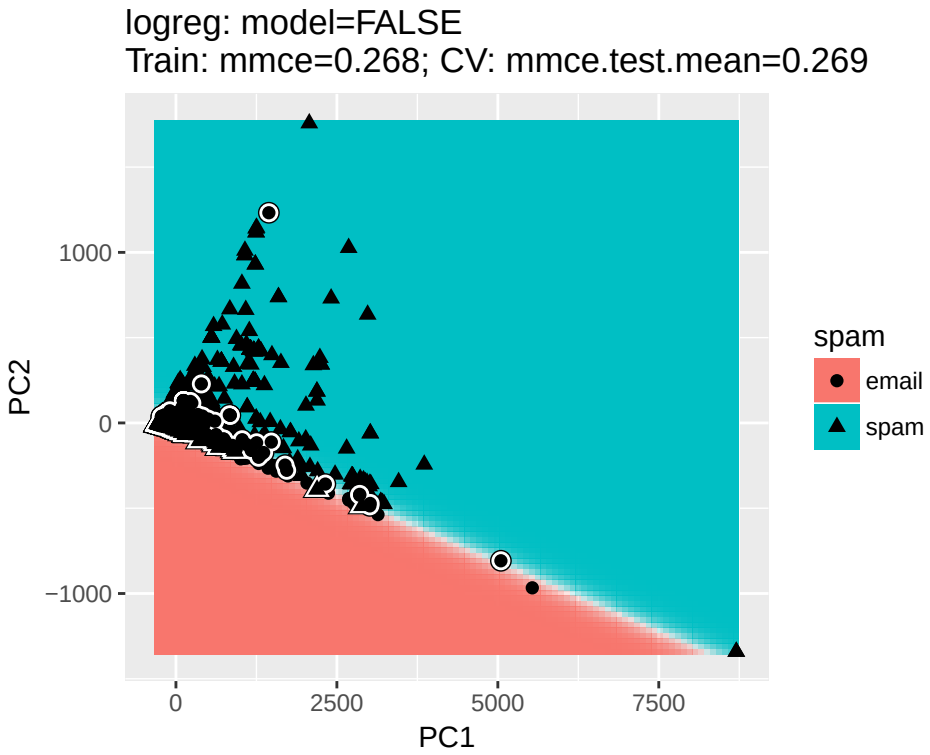
# 5 Klasyfikacja z redukcją wymiaru

Spróbujmy teraz przeprowadzić podobną analizę z uprzednim zastosowaniem algorytmu PCA w celu redukcji wymiaru. Do dalszych analiz wybieramy 11 składowych głównych. Porównamy wyniki wskaźników takich jak w poprzednim roździale, to zanczy skuteczności, AUC, czułości i specyficzności. Po dobraniu parametrów do niektórych metod, sprawdzona zostanie skuteczność na zbiorze testowym w celu porównania do zbioru treningowego. Dla każdej metody zostaną przedstawione graficznie granice podziału dla pierwszych dwóch składowych głównych.

```
smp<- sample(dim(spam.PCA)[1],dim(spam.PCA)[1]*0.2)
test <-spam.PCA[smp,]
train<-spam.PCA[-smp,]
task<-makeClassifTask(data=train, target = "spam")
```

## 5.1 Regresja Logistyczna

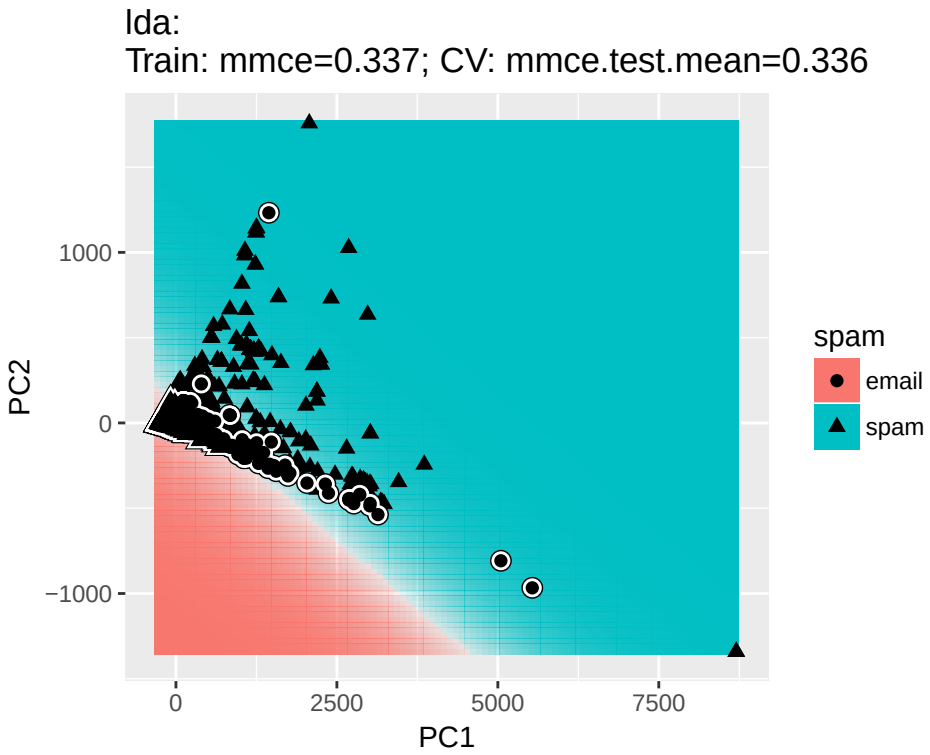
```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.8810106     0.9416694     0.9148017     0.8283682
```




Rysunek 13: Klasyfikacja metodą RL po PCA

## 5.2 LDA

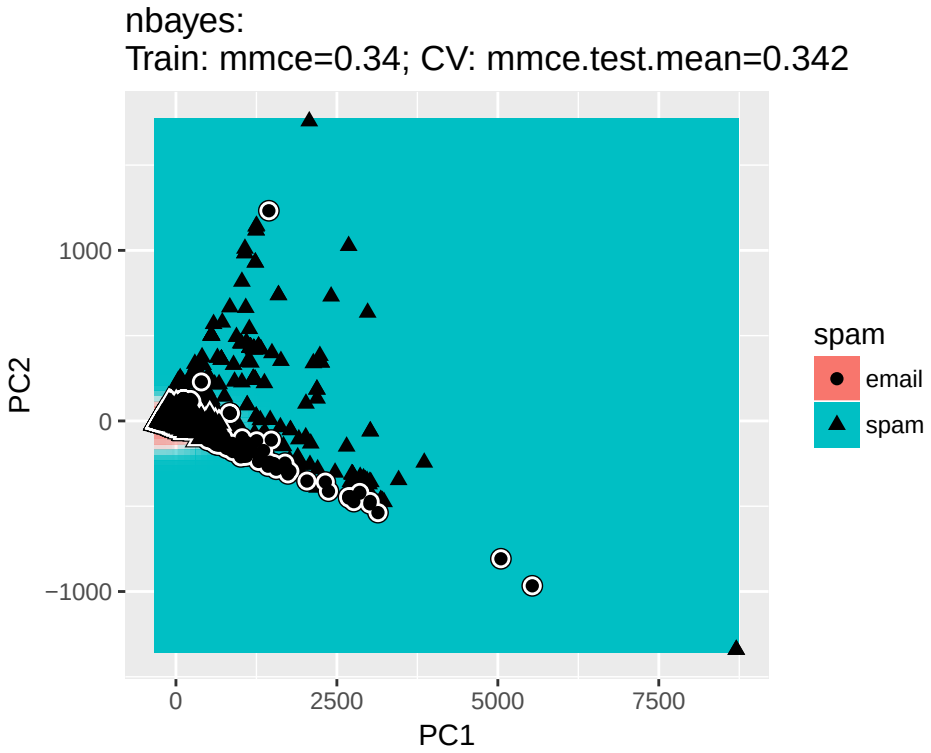
```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.8207048     0.8909489     0.9018692     0.6942315
```




Rysunek 14: Klasyfikacja metodą LDA po PCA

## 5.3 Klasyfikator Naiwny Bayesa

```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.6625892     0.8332681     0.9335468     0.2404859
```

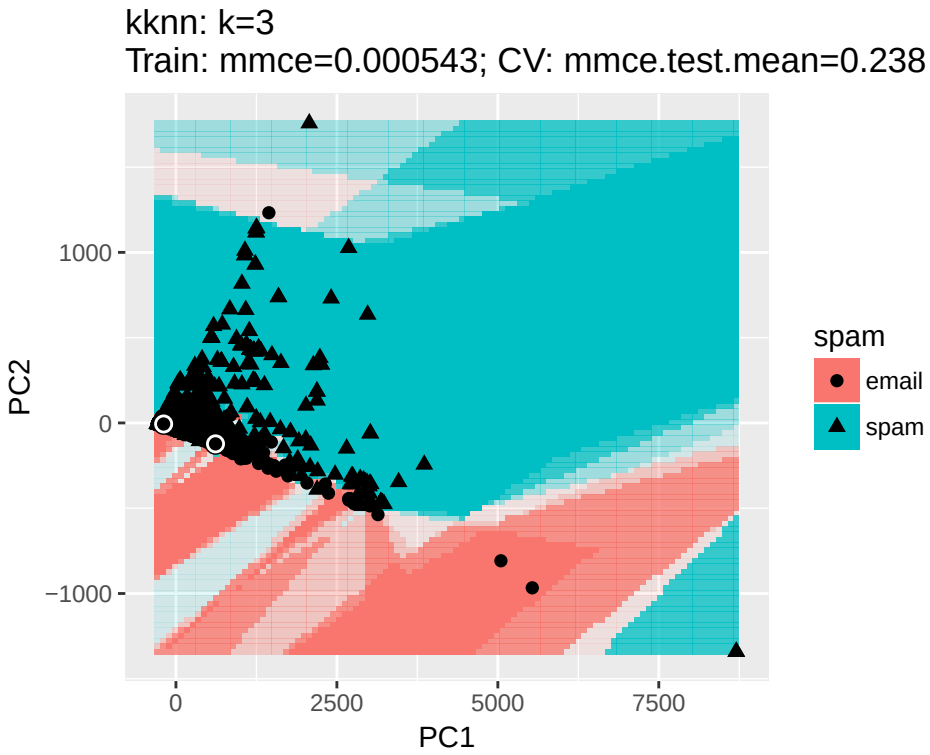



Rysunek 15: Klasyfikator Naiwny Bayes'a po PCA

## 5.4 Metoda k-Najbliższych Sąsiadów

```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.8755769     0.9302321     0.8822542     0.8651737
```

```
mod <- train(knn.learner, task= task)
test.pred<-predict(mod,newdata = test)
knn.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
knn.test.acc

## [1] 0.8608696
```

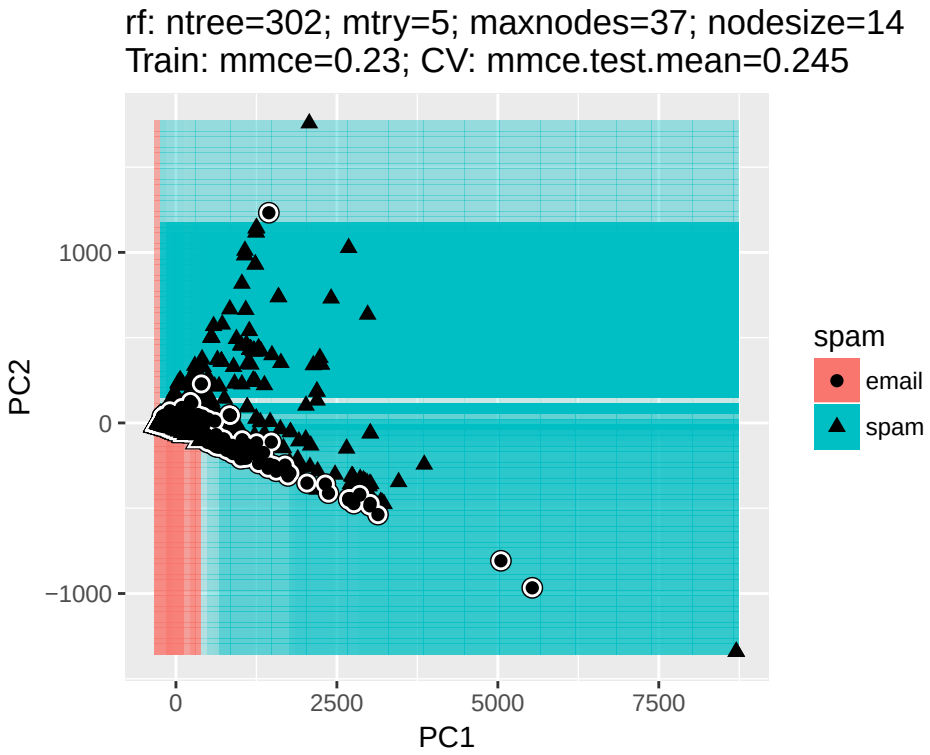



Rysunek 16: Klasyfikacja metodą kNN po PCA

## 5.5 Lasy losowe

```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.8777498     0.9472163     0.9107908     0.8262848
```

```
mod <- train(randForest.learner, task= task)
test.pred<-predict(mod,newdata = test)
randForest.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
randForest.test.acc

## [1] 0.8728261
```

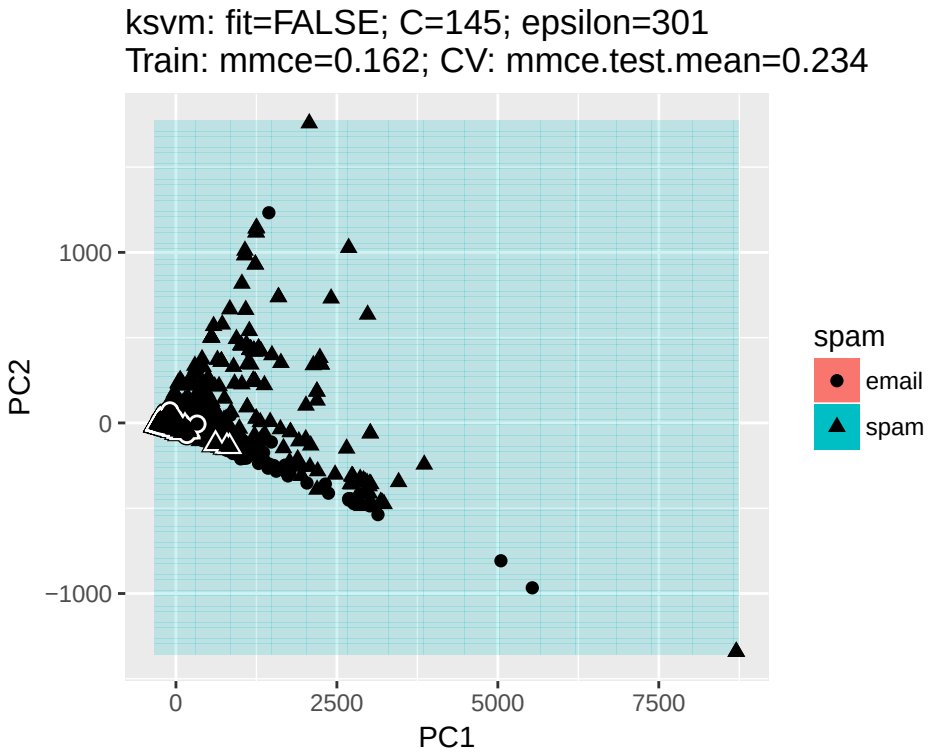



Rysunek 17: Las losowy po PCA

## 5.6 Maszyna wektorów nośnych

```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9016569     0.9499120     0.9259664     0.8637897
```

```
## [1] 0.8923913
```

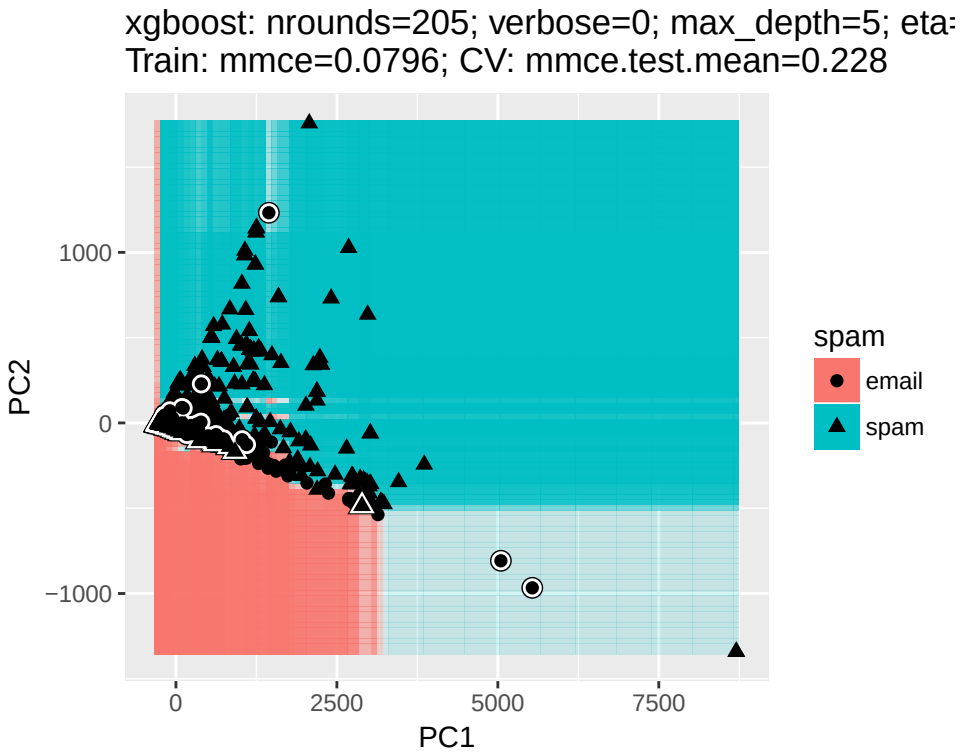



Rysunek 18: Klasyfikacja metodą SVM po PCA

## 5.7 XGBoost

```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9103574     0.9683098     0.9250736     0.8874129
```

```
## [1] 0.901087
```

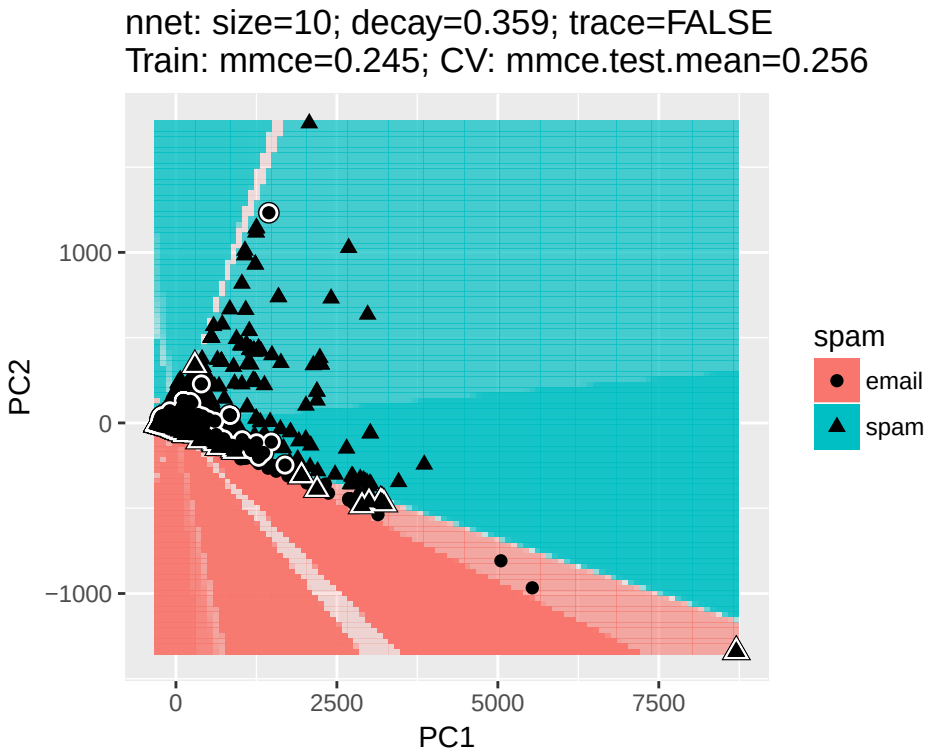



Rysunek 19: Klasyfikacja metodą SVM po PCA

## 5.8 Sieci neuronowe

```
## acc.test.mean auc.test.mean tpr.test.mean tnr.test.mean
##     0.9030160     0.9544155     0.9116757     0.8895156
```

```
mod <- train(nn_learner, task= task)
test.pred<-predict(mod,newdata = test)
nn.test.acc<-mean(test.pred$data[,1]==test.pred$data[,4])
nn.test.acc

## [1] 0.8869565
```




Rysunek 20: Klasyfikacja za pomocą sieci neuronowych po PCA

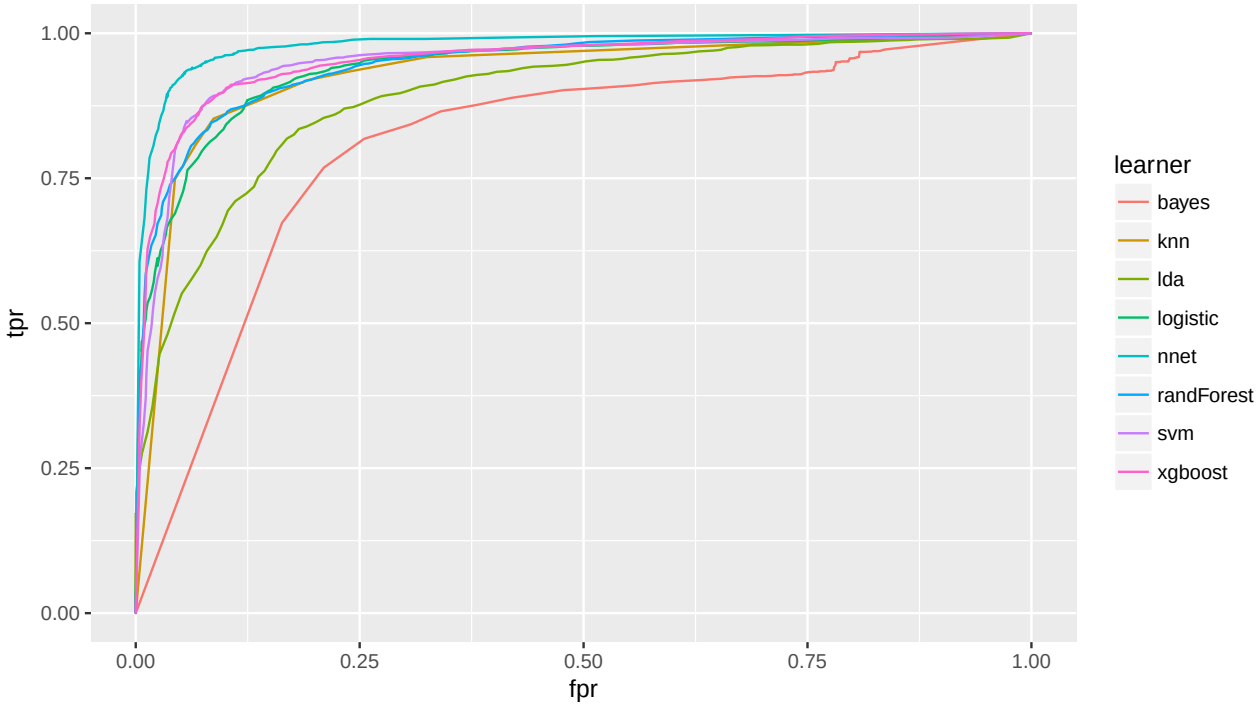
## 5.9 Porównanie

| | Skuteczność | AUC | Czułość | Specyficzność |
|---|---|---|---|---|
| Regresja Logistyczna | 0.88 | 0.94 | 0.91 | 0.83 |
| LDA | 0.82 | 0.89 | 0.90 | 0.69 |
| Naiwny Bayes | 0.66 | 0.83 | 0.93 | 0.24 |
| kNN | 0.88 | 0.93 | 0.88 | 0.87 |
| Lasy Losowe | 0.88 | 0.95 | 0.91 | 0.83 |
| Maszyna Wektorów Nośnych | 0.90 | 0.95 | 0.93 | 0.86 |
| XGBoost | 0.91 | 0.97 | 0.93 | 0.89 |
| Sieci neuronowe | 0.90 | 0.95 | 0.91 | 0.89 |

Tabela 2: Porównanie skuteczności metod klasyfikacji dla składowych głównych danych "spam"

Tym razem wyraźniej widać przewagę złożoności algorytmu XGBoost nad prostrzymi metodami np. Regresji Logistycznej. Skuteczność metody LDA wyraźnie spadła, natomiast pozostałe metody zaliczyły około 4-5% spadek skuteczności.

Poniżej przedstawiono porównanie krzywych ROC dla wszystkich klasyfikatorów:




Rysunek 21: Porównanie krzywych ROC dla różnych klasyfikatorów (po PCA)

Podobnie tym razem, bazując na mierze dokładności i auc, możemy stwierdzić, że najlepiej poradziła sobie metoda lasów losowych.

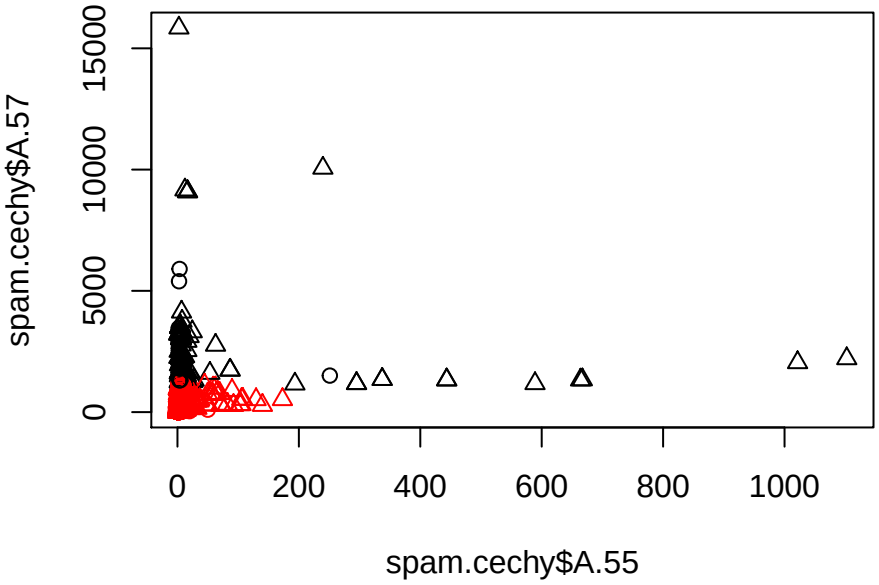
# 6 Analiza skupień

Załóżmy na chwilę, że nie wiemy jak sklasyfikowane są obserwacje, którymi dysponujemy, tzn. nie wiemy czy są spamem, czy nie. Usuwamy z danych kolumnę typu factor, wskazującą na ten podział. Wykorzystamy poznane metody klasteryzacji, mające na celu grupowanie danych w skupienia obiektów jak najbardziej do siebie podobnych, co wiąże się z rozpoznaniem struktury danych. Rozważymy metody grupujące, jak k-means i PAM oraz metody hierarchiczne, tj. AGNES i DIANA. W przypadku metody AGNES rozważymy 3 metody łączenia skupień: "average", "single" i "complete". Macierze niepodobieństw będą bazowały na odległości euklidesowej. Nie wykonujemy skalowania danych, ponieważ wariancja zmiennych jest jednorodna i taka operacja mogłaby pogorszyć wyniki. Na koniec ocenimy jakość grupowania.

## 6.1 Metody grupujące

### 6.1.1 K-means

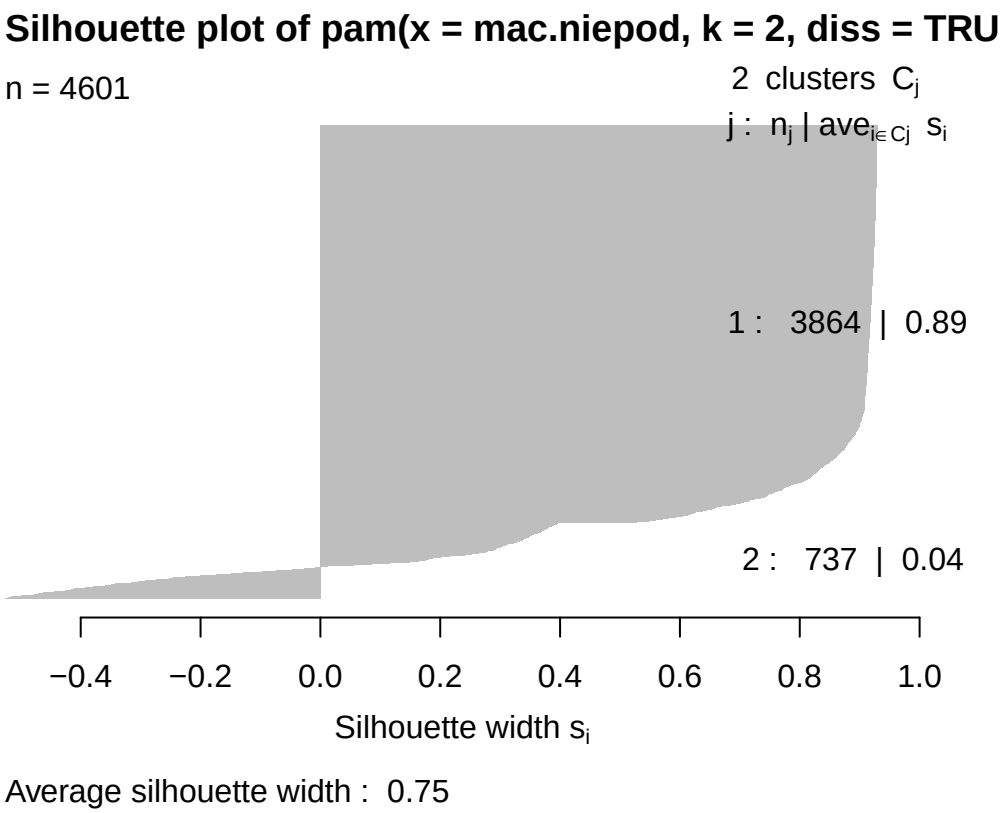
Ponieważ dysponujemy wyłącznie zmiennymi ilościowymi, możemy śmiało zastosować metodę k-means. Poniżesz przedstawiamy rezultat w przykładowej przestrzeni dwóch zmiennych.




Rysunek 22: Wizualizacja wyników analizy skupień metodą k-means (w przekładowej przestrzeni dwóch zmiennych)

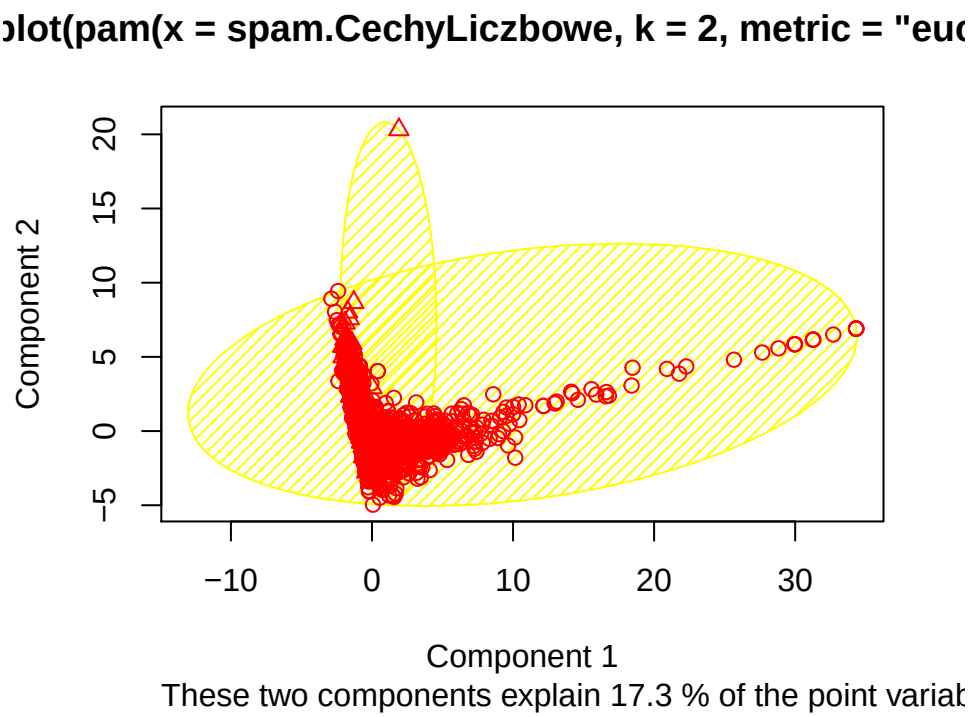
### 6.1.2 PAM

Wypróbujemy również nieco bardziej złożoną metodę PAM. Poniżej prezentujemy wykres podziału na klastry oraz średnią szerokość silhouette.




Rysunek 23: Wizualizacja wyników analizy skupień metodą PAM

Obserwując powyższy wykres możemy zaobserwować, że otrzymaliśmy skupienia, o wyraźnie różnych liczbach obserwacji. Przypomnijmy, że nasz zbiór składa się w ok. 60 % z maili i ok. 40% ze spamu, dlatego tak nierówne klastry, jakie otrzymaliśmy za pomocą metody PAM, na pierwszy rzut oka, wydają się nieodpowiednie, być może metoda PAM wykryła inną zależność w danych.

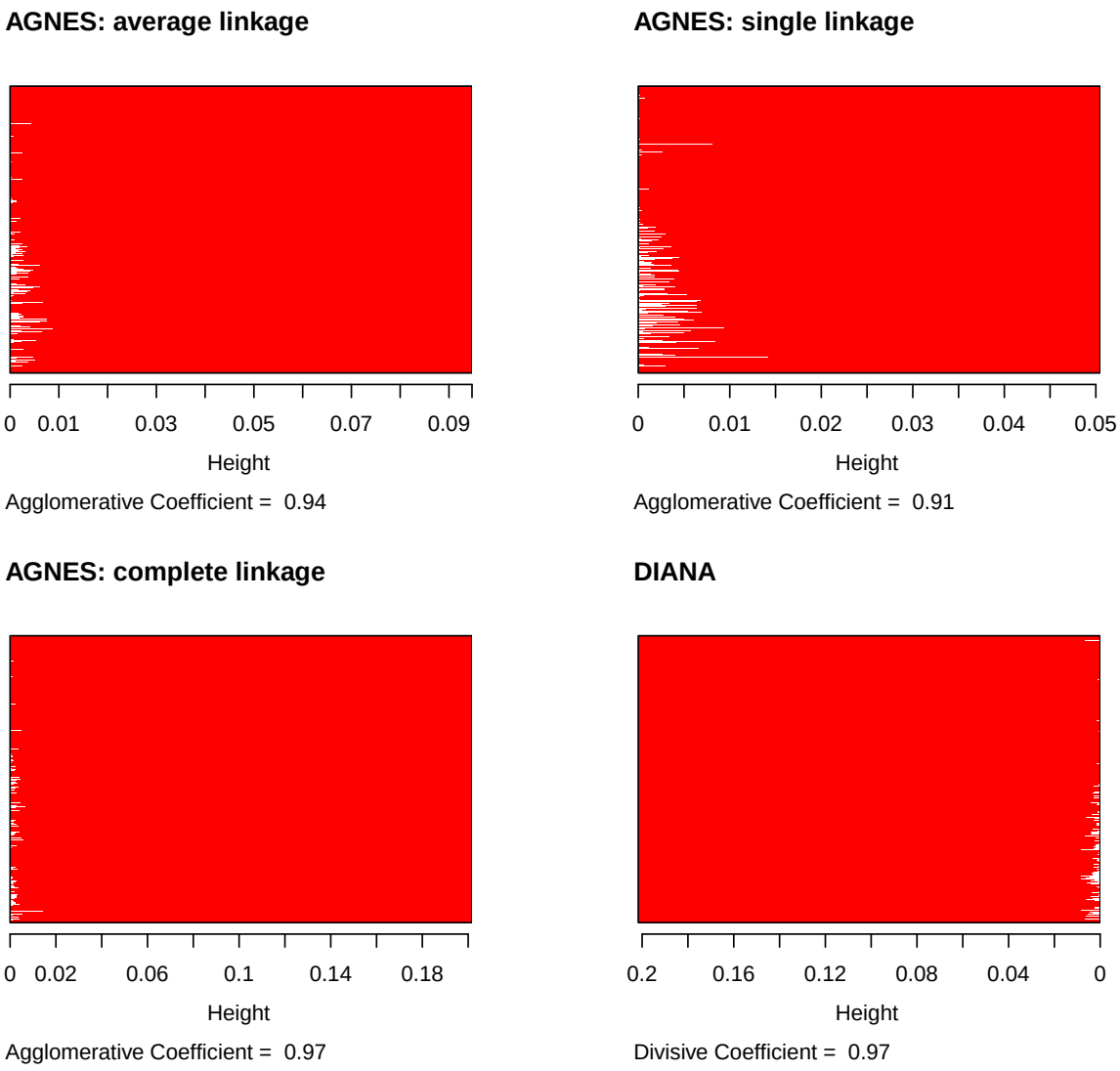



Rysunek 24: Wizualizacja wyników analizy skupień z wykorzystaniem PCA

Przejdziemy teraz do drugiej grupy metod analizy skupień.

## 6.2 Metody chierarchiczne

Podstawową wadą metod grupujących jest skuteczność wykrywanie skupień wypukłych, metody chierarchiczne nie posiadają tej wady. Rozważymy dwa warianty: metodę aglomeracyjną AGNES i metodę rozdzielania DIANA. Wyniki spróbujemy zwizualizować za pomocą wykresów typu 'bannerplot', dendrogramy w przypadku tak dużej ilości danych, są całkowicie nieczytelne.

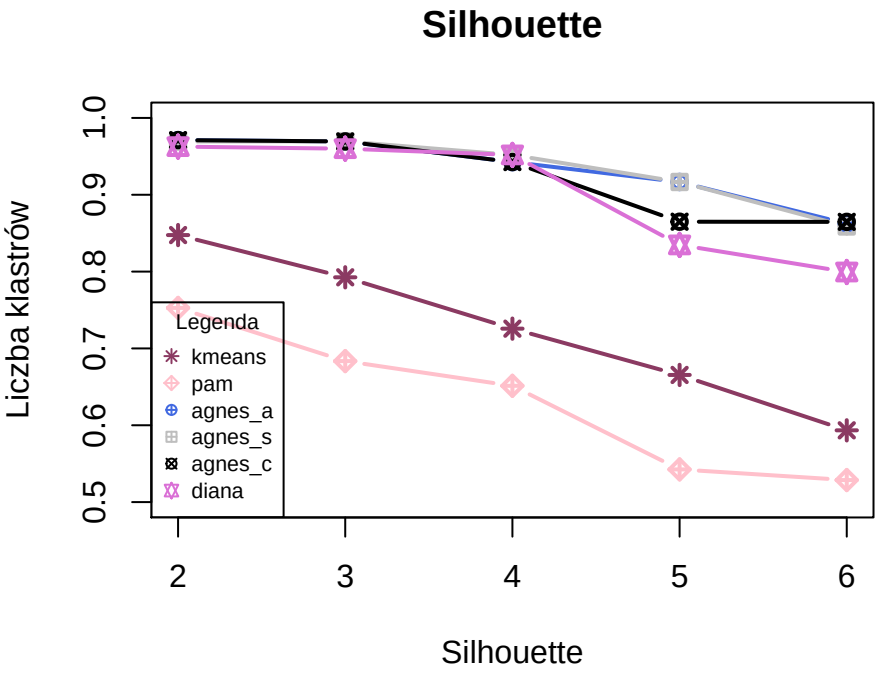



Rysunek 25: AGNES i DIANA: wykresy typu 'banner'

Widzimy, że metody "average" and "single" poradziły sobie lepiej od pozostałych.

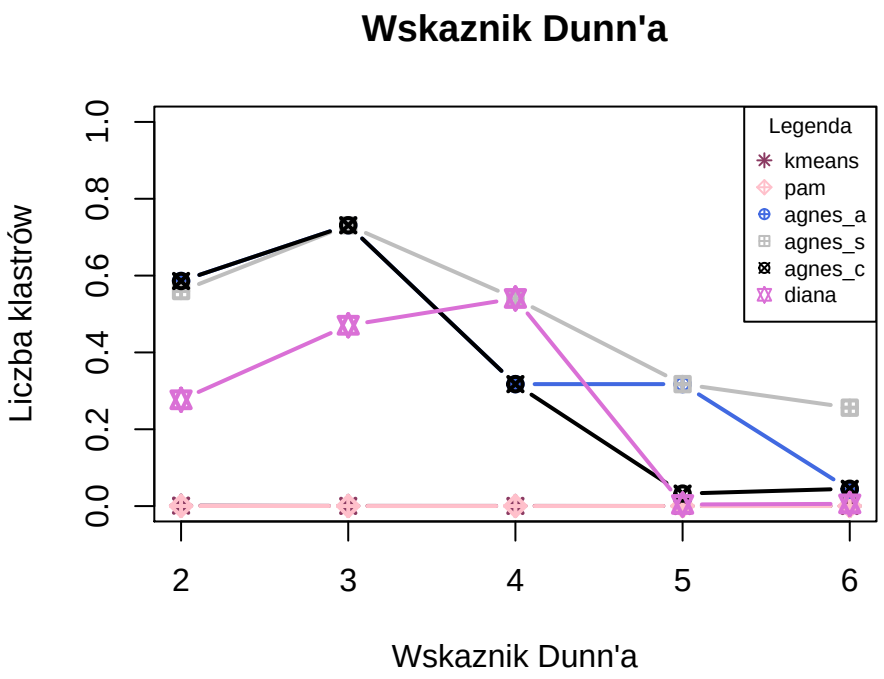
### 6.2.1 Wskaźniki wewnętrzne

W kolejnym kroku dokonamy analizy jakości grupowania, odwołując się wyłącznie do własności zawartych w danych. Weźmiemy pod uwagę m. in. takie własności jak: zwartość, spójność czy separacja przestrzenna. Posłużymy się wskaźnikami Silhouette i Dunn'a. Wskaźnik silhouette bierze pod uwagę zwartość klastrów i separację przestrzenną, poniżej prezentujemy wykres średniej wartości silhouette dla całej partycji w przypadku podziału na $k = 2, \ldots, 6$ klastrów.




Rysunek 26: Wskaźniki wewnętrzne: silhouette

Nietrudno zauważyć, że w przypadku każdej z metod miara silhouette osiąga największą wartość w przypadku podziału na 2 skupienia. Jaką kolejną miarę jakości grupowania rozważymy wskaźnik Dunn'a, przyjmujący wartości dodatnie, im większa jego wartość, tym lepsza jakość. Poniższy wykres przedstawia wartości wskaźnika Dunn'a dla poszczególnych metod w przypadku grupowania na od 2 do 6 klastrów.




Rysunek 27: Wskaźniki wewnętrzne: wsk. Dunn'a

Wartości wskaźników Dunn'a są bardzo odmienne. W przypadku metod chierarchicznych: AGNES i DIANY wskaźnik ten sugeruje odpowiednio podziały na 3 i 4 klastry. Wartości wskaźnika Dunn'a dla k-means i PAM są tak małe, że nie jest możliwy odczyt bezpośrednio z wykresu, dlatego zawarliśmy wartości w poniższej tabelce, gdzie agnesa oznacza AGNES z wykorzystaniem metody łączenia "average", agnesS z wykorzystaniem łączenia "single" a agnesc z wykorzystaniem łączenia "complete".

| | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| k-means | 0.00125 | 0.00053 | 0.00050 | 0.00023 | 0.00028 |
| pam | 0.00030 | 0.00016 | 0.00014 | 0.00019 | 0.00015 |
| $\text{agnes}_a$ | 0.58591 | 0.73105 | 0.31752 | 0.31752 | 0.04283 |
| $\text{agnes}_S$ | 0.55976 | 0.73105 | 0.54050 | 0.31752 | 0.25621 |
| $\text{agnes}_C$ | 0.58591 | 0.73105 | 0.31752 | 0.03251 | 0.04510 |
| diana | 0.27676 | 0.47069 | 0.54050 | 0.00423 | 0.00577 |

Tabela 3: Wskaźnik Dunn'a

Widzimy, że w przypadku metod k-means i pam, wskaźnik Dunn'a identyfikuje grupowanie najlepszej jakości dla 2 klastrów.

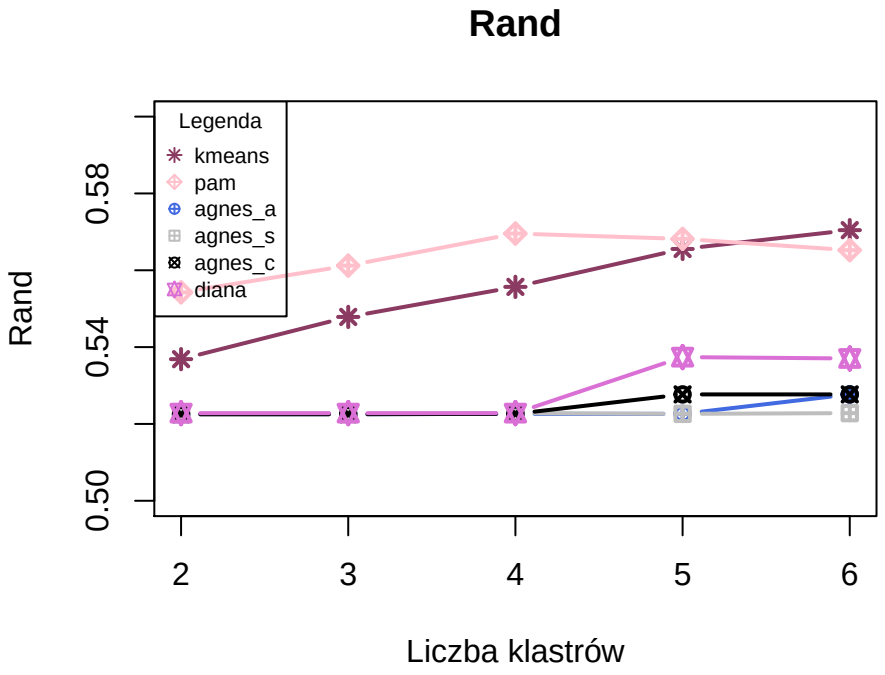
### 6.2.2 Wskaźniki zewnętrzne

W kolejnym kroku zajmiemy się analizą wskaźników zewnętrznych, czyli takich, które wykorzystują informację o rzeczywistej przynależności do skupień. Rozważymy 4 metody i przypadek 2 klastrów. Poniższa tabelka przedstawia ilość maili i spamu przyporządkowych do dwóch klastrów i % par, która właściwie znalazły się w osobnych skupieniach.

| | $e\text{-}mail_1$ | $e\text{-}mail_2$ | $spam_1$ | $spam_2$ | matched(proc) |
|---|---|---|---|---|---|
| k-means | 53 | 2735 | 191 | 1622 | 36 |
| pam | 2555 | 233 | 1309 | 504 | 66 |
| $agnes_a$ | 2788 | 0 | 1811 | 2 | 61 |
| $agnes_s$ | 2788 | 0 | 1812 | 1 | 61 |
| $agnes_c$ | 2788 | 0 | 1811 | 2 | 61 |
| diana | 2788 | 0 | 1808 | 5 | 61 |

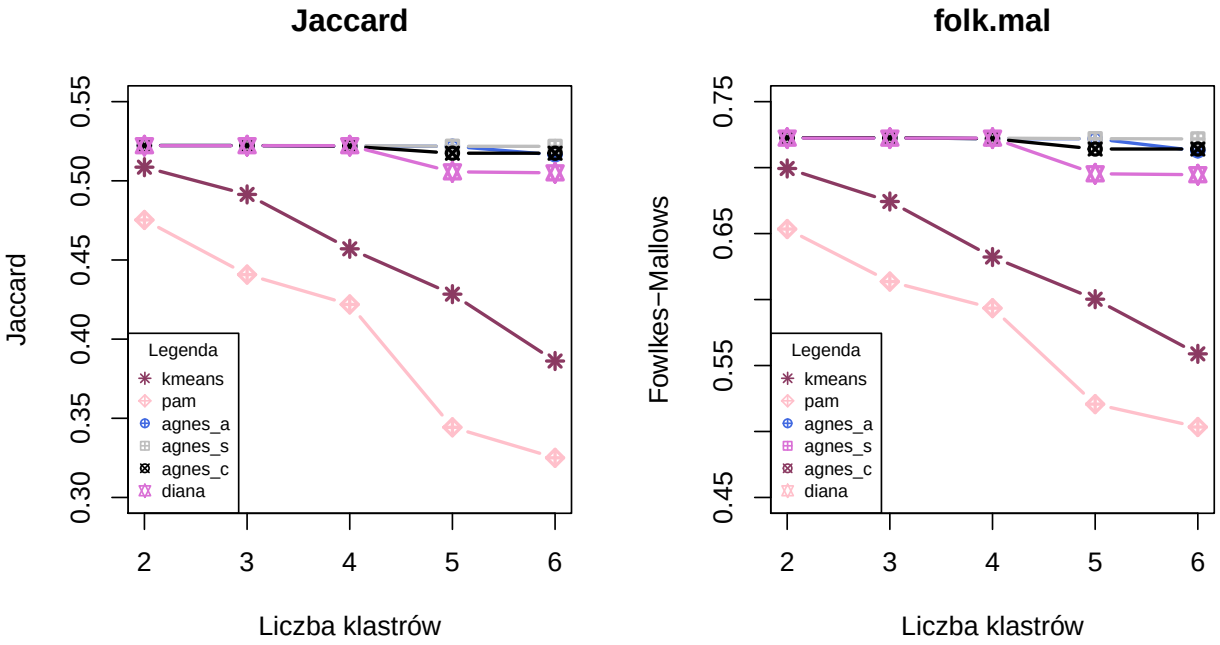
Tabela 4: Dopasowanie klas

W przypadku k-means i PAM otrzymujemy największy procent "pasujących" par (ostatnia kolumna). Analizując dalej wskaźniki zgodności partycji, przechodzimy do wyznaczenia wskaźników Randa, Jaccarda i Fowlkesa-Mallowsa. Wskaźnik Randa przyjmuje wartości od 0 do 1 i im większa jego wartość tym lepsza zgodność partycji. Poniżej prezentujemy wykres wartości wskaźnika Randa dla rozważanych metod zastosowanych do grupowania na od 2 do 6 skupień.




Rysunek 28: Wskaźniki zewnętrzne: wsk. Randa

Wskaźnik Randa osiąga większe wartości, zgodnie z wykresem, przy klasteryzacji na $3, 4$ lub $5$ skupień. Kierowanie się tylko jednym wskaźnikiem, mogłoby nas zaprowadzić do wyciągnięcia błędnych wniosków dotyczących wewnętrznej struktory danych, dlatego przeanalizujemy inne miary. Poniżej przedstawiamy wykres wskaźników Jaccarda i Fowlkesa-Mallowsa.




Rysunek 29: Wskaźniki zewnętrzne: Jaccarda i Fowlkesa-Mallowsa

Zarówno wskaźnik Jaccarda, jak i Fowlkesa-Mallowska przyjmują największe wartości przy klasteryzacji na 2 skupienia, w szczególności istotne różnice obserwujemy w przypadku k-means i PAM.

# 7 Podsumowanie

Podsumowując, spośród przeanalizowaniu licznych metod klasyfikacji, najlepszą okazała się być XGBoost. Jednak wyniki uzyskane za pomocą pozostałych rozważanych metod (poza Klasyfikatorem Naiwnym Bayesa), nie są istotnie gorsze. W szczególności bardzo dobrą skuteczność uzyskały metody Regresji Logistycznej i LDA, charakteryzujące się znacznie mniejszą złożonością obliczeniową i interpretowalnością. Jeśli chodzi o analizę skupień, rozważone metody nie niosą za sobą jednoznacznej informacji o strukturze wewnętrznej danych. Różnorodność wyników wywołuje wątpliwości. Podział wynikający z zastosowania powyższych metod może nie być tożsamy z podziałem na grupy "mail" i "spam" i może on reprezentować inne niezane partycje, za którymi stoją informacje dotyczące pewnej grupy maili, których odtworzenie może być niemożliwe ze względu na strukturę danych (nie mamy pełnych maili, tylko częstotliwość słów).