

# Projekt 1 - Systemy Decyzyjne

drużyna1, Uczestnicy: Bogna Pawlus

2022-12-08

## 1. Opis danych, wizualizacja i przekształcenia zmiennych

Zacznijmy od zamienienia 6 klas zmiennej GameType na liczby od 0 do 5:

```
set.seed(1)
levels(as.factor(sc_project_training_data$GameType))

## [1] "pvp" "pvt" "pvz" "tvt" "tvz" "zvz"

sc_project_training_data$GameType =
  as.numeric(unclass(as.factor(sc_project_training_data$GameType))) - 1
sc_project_training_data = sc_project_training_data[,-1]
sc_project_test_data$GameType = as.numeric(unclass(as.factor(sc_project_test_data$GameType))) - 1
```

Zbiór treningowy po przeróbce

```
dim(sc_project_training_data)
```

```
## [1] 34345   159
```

```
head(sc_project_training_data[, 1:6])
```

```
##   Winner GameType Minute Minerals.first Minerals.min Minerals.max
## 1      A       2       2          36           8         208
## 2      A       1      23          8           8           8
## 3      B       1      22         290          24        594
## 4      B       2      35        3360        1743       3402
## 5      A       2      13         501          24        501
## 6      A       4      21        1860         626       1860
```

Wektor wygranych dla zbioru treningowego

```
Y_train = sc_project_training_data[, 1]
length(Y_train)
```

```
## [1] 34345
```

```

head(Y_train)

## [1] "A" "A" "B" "B" "A" "A"

Zbiór testowy

dim(sc_project_test_data)

## [1] 10725   158

head(sc_project_test_data[, 1:6])

##   GameType Minute Minerals.first Minerals.min Minerals.max Minerals.last
## 1         1      14          537        207        569        322
## 2         4      10          243        31        487        487
## 3         4       6          88        52        296        138
## 4         2      13          183        93        753        292
## 5         5       3          210       10        338        112
## 6         1       4          113        9        243        109

```

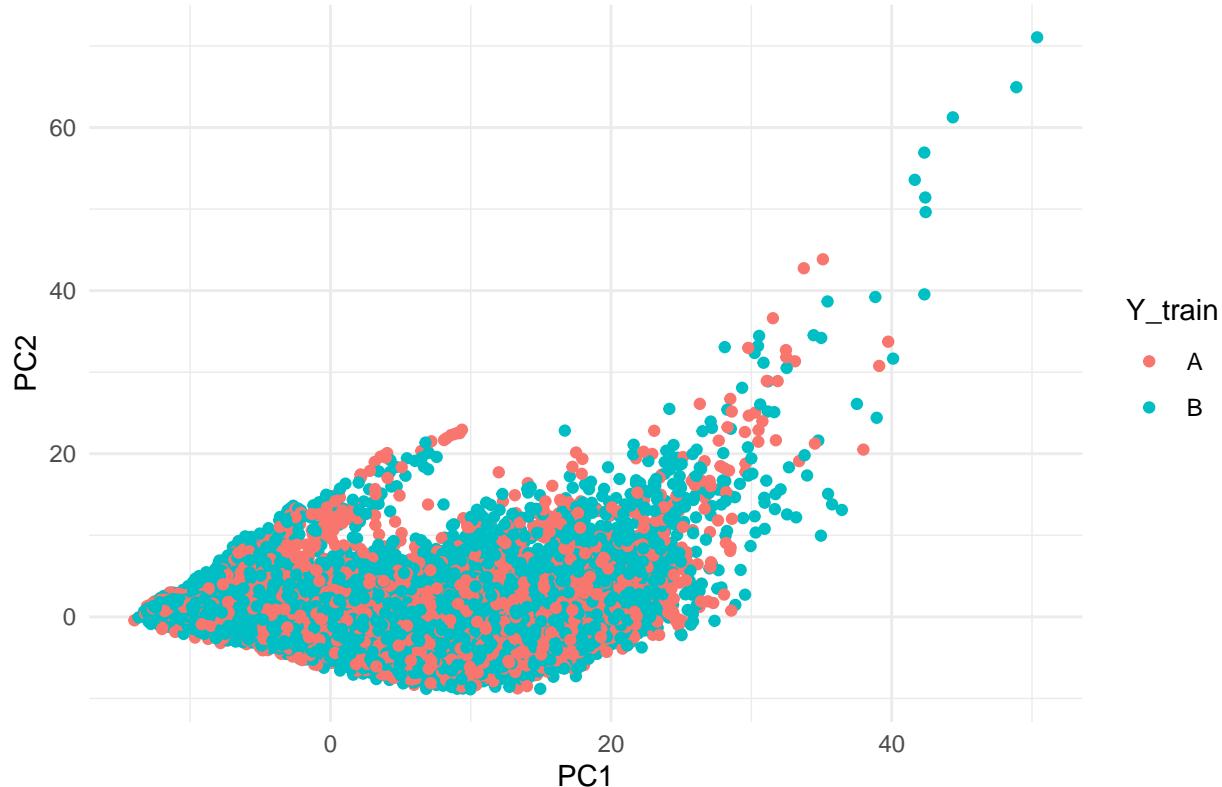
Zwizualizujemy dane treningowe za pomocą metody PCA:

```

pca <- prcomp(sc_project_training_data[, -1], center=TRUE, scale=TRUE)
#pca$x -> nowa przestrzeń
ggplot(data=data.frame(pca$x), aes(x=PC1, y=PC2, color=Y_train)) + geom_point() +
  ggtitle("Obserwacje treningowe, pokolorowane względem Winner") + theme_minimal()

```

Obserwacje treningowe, pokolorowane względem Winner



Na początku spróbujemy przekształcić dane tak, żeby po zastosowaniu wizualizacji PCA układły się bardziej symetrycznie wokół 0. Analizując sparowane kolumny o nazwach x i Enemyx możemy zobaczyć, że mają bardzo podobną średnią i wariancję. Dlatego kolumny x i Enemyx połączymy w jedną kolumnę o wartościach x - Enemyx.

```
#łączymy zmienne objaśniające ze zbioru tesotowego i treningowego
wszystko = rbind(sc_project_training_data[,-1], sc_project_test_data)
#Y_train = sc_project_training_data[,1]
sklej0 = wszystko[,1:2]
sklej1 = wszystko[,3:74]
sklej2 = wszystko[,75:146]
sklej3 = wszystko[,147:152]
sklej4 = wszystko[,153:158]
sklejA = sklej1 - sklej2 #kolumny 3:74 odpowiadają kolumnom 75:146
sklejB = sklej3 - sklej4 #kolumny 147:152 odpowiadają kolumnom 153:158
wszystko_po_sklejeniu = cbind(sklej0, sklejA)
wszystko_po_sklejeniu = cbind(wszystko_po_sklejeniu, sklejB)
```

Sklejone dane treningowe i testowe po usunięciu połowy zmiennych:

```
wszystko_po_sklejeniu = data.frame(scale(wszystko_po_sklejeniu))
dim(wszystko_po_sklejeniu)
```

```
## [1] 45070     80
```

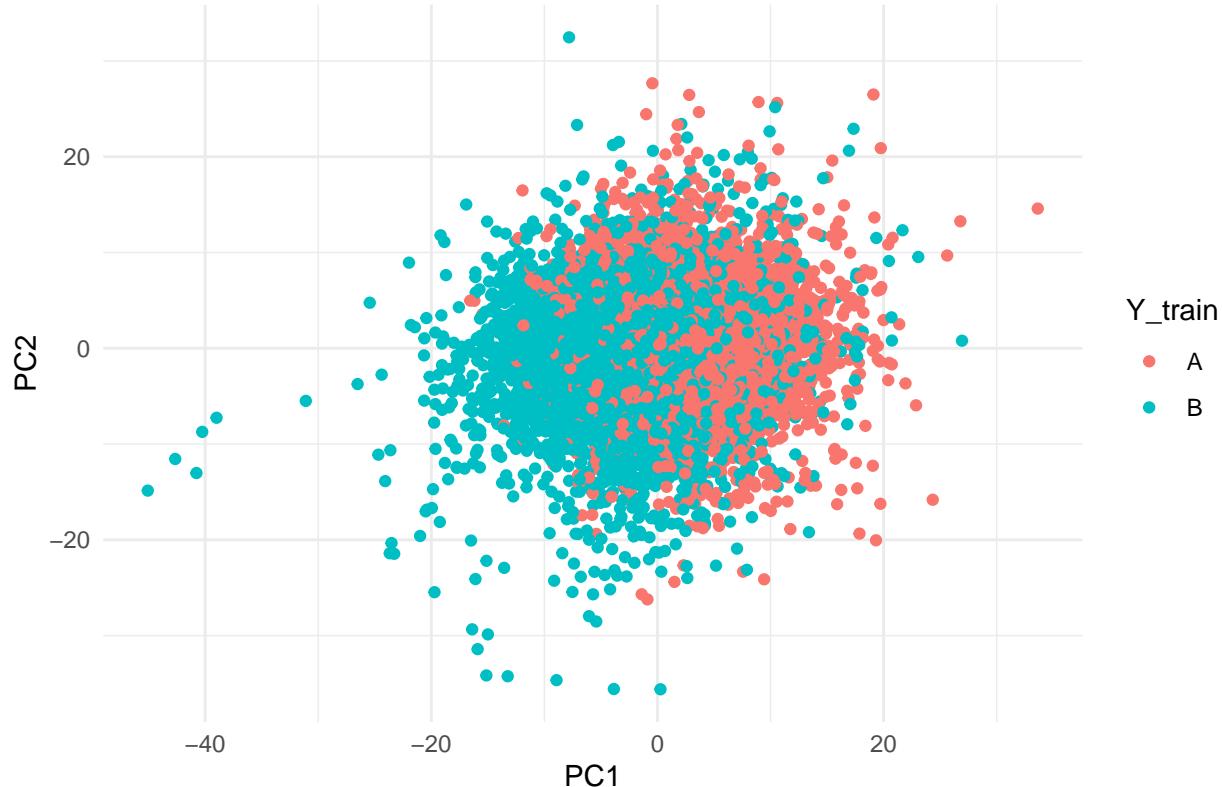
```
head(wszystko_po_sklejeniu[,1:6])
```

	GameType	Minute	Minerals.first	Minerals.min	Minerals.max	Minerals.last
## 1	0.03635722	-1.2397282	0.006068429	-0.01720527	-0.1390066	-0.0638855
## 2	-0.69030347	1.0360002	-1.838905553	-1.57713110	-1.9121425	-1.5344644
## 3	-0.69030347	0.9276321	-0.322379434	-0.01248966	-0.4112859	0.3990745
## 4	0.03635722	2.3364163	-0.417992289	-0.98579284	-0.4486413	-0.8902351
## 5	0.03635722	-0.0476800	0.065383812	-0.23695299	-0.3905329	-0.6562020
## 6	1.48967859	0.8192641	0.976361848	0.57507612	0.7641638	0.4637528

Przekształcone dane po wizualizacji PCA są bardziej symetryczne wokół 0 i łatwiej podzielić je na liniowo dwie grupy:

```
pca <- prcomp(wszystko_po_sklejeniu[1:nrow(sc_project_training_data), -1], center=TRUE, scale=TRUE)
#pca$x -> nowa przestrzeń
ggplot(data=data.frame(pca$x), aes(x=PC1, y=PC2, color=Y_train)) + geom_point() +
  ggtitle("Obserwacje treningowe, pokolorowane względem Winner") + theme_minimal()
```

## Obserwacje treningowe, pokolorowane względem Winner



### Model 1 - regresja logistyczna

Dzielimy sc\_project\_training\_data na zbiory testowy i walidacyjny

```
edited_test = wszystko_po_sklejeniu[(nrow(sc_project_training_data)+1):nrow(wszystko_po_sklejeniu),]
edited_train = wszystko_po_sklejeniu[1:nrow(sc_project_training_data),]
#Y_train->dane kto wygrał

Y_train = unclass(as.factor(Y_train)) - 1 #zamieniamy A na 0 i B na 1
edited_train = cbind(Y_train, edited_train)
train_ind = sample.split(1:nrow(sc_project_training_data), 0.8)
train = edited_train[train_ind, ]
valid = edited_train[!train_ind, ]
```

Stworzymy model regresji logistycznej:

```
logistic_model <- glm(Y_train ~ ., data = train, family = "binomial")
#przewidywania dla zbioru walidacyjnego
predict <- predict(logistic_model, valid, type = "response")

predict = data.frame(predict)
colnames(predict)[1] = "pstwo_B"
predict$prawdziwe = valid$Y_train
#poprawiamy próg kwalifikacji do klas na 0.54, bo wtedy trafiamy więcej, niż przy 0.5
```

```

#to znaczy, cm[1,1] + cm[2,2] jest większe
predict$predicted.classes = ifelse(predict$pstwo_B > 0.54, 1, 0)
head(predict, n = 10L)

##      pstwo_B prawdziwe predicted.classes
## 4   0.8813141          1              1
## 6   0.2260064          0              0
## 7   0.7733249          1              1
## 18  0.5646083          1              1
## 21  0.8154330          1              1
## 29  0.6234367          0              1
## 35  0.6915894          1              1
## 41  0.4766418          0              0
## 52  0.7679255          0              1
## 61  0.6194562          1              1

mean(predict$prawdziwe == predict$predicted.classes) ##ile trafiono

## [1] 0.6587567

cm = confusionMatrix(factor(predict$prawdziwe), factor(predict$predicted.classes))$table
cm = cm/(sum(cm))
cm

##      Reference
## Prediction      0          1
##       0 0.15664580 0.24661523
##       1 0.09462804 0.50211093

cm[1,1] + cm[2,2]

## [1] 0.6587567

```

Predykcja dla edited\_test:

```

predict_test <- predict(logistic_model, edited_test, type = "response")
predict_test = data.frame(predict_test)
#predicted01 oznacza zamienione A na 1, B na 0
predict_test$predicted.01 <- ifelse(predict_test$predict > 0.54, 0, 1)
#predict liczy nam prawdopodobieństwo klasy B, dlatego zamieniamy na 1 - pstwo:
predict_test$a1minus = 1 - as.numeric(predict_test$predict_test)
predict_test$wynik = (predict_test$a1minus) ^ 0.892623
head(predict_test)

##      predict_test predicted.01     a1minus      wynik
## 34346    0.9790620          0 0.02093797 0.03171228
## 34347    0.1389528          1 0.86104720 0.87499096
## 34348    0.5702486          0 0.42975140 0.47054518
## 34349    0.6229138          0 0.37708621 0.41871757
## 34350    0.5615214          0 0.43847865 0.47906556
## 34351    0.6433692          0 0.35663085 0.39838254

```

```
#0.46^x = 0.5 => x=0.892623
#podnosimy do datej potęgi, bo wcześniej wzięliśmy predict_test$predict > 0.54
#predict_test$predicted.clases <- ifelse(predict_test$predicted.01 > 0.54, 'B', 'A')
#zgłoszenie:
write.table(predict_test$wynik, "zgloszenie1.txt", sep="\n", row.names=FALSE)
```

## Model 2 - sieć neuronowa

Będziemy dalej korzystać ze zmiennej z 1. rozdziału -> sklejonych i przeskalowanych danych treningowych z testowymi

```
dim(wszystko_po_sklejeniu)

## [1] 45070     80

head(wszystko_po_sklejeniu[,1:6])

##      GameType      Minute Minerals.first Minerals.min Minerals.max Minerals.last
## 1  0.03635722 -1.2397282   0.006068429  -0.01720527  -0.1390066   -0.0638855
## 2 -0.69030347  1.0360002  -1.838905553  -1.57713110  -1.9121425  -1.5344644
## 3 -0.69030347  0.9276321  -0.322379434  -0.01248966  -0.4112859   0.3990745
## 4  0.03635722  2.3364163  -0.417992289  -0.98579284  -0.4486413  -0.8902351
## 5  0.03635722 -0.0476800   0.065383812  -0.23695299  -0.3905329  -0.6562020
## 6  1.48967859  0.8192641   0.976361848   0.57507612   0.7641638   0.4637528
```

Dodamy 1 wymiar do tych danych:

```
fun <- function(x) {
  return(1/sum(x^20))
}

wszystko_po_sklejeniu$control = apply(wszystko_po_sklejeniu, 1, fun)
wszystko_po_sklejeniu$control = scale(wszystko_po_sklejeniu$control)
```

Zaczniemy od znalezienia parametru hidden dla sieci neuronowej z 3 warstwami. Wykorzystamy do tego siatkę parametrów  $5 \times 5 \times 5$  i korzystając z walidacji krzyżowej, policzymy, dla jakiej konfiguracji parametrów przewidzimy największy procent wyników

```
set.seed(1)
podzial <- cut(1:nrow(edited_train), 5, labels = F)
podzial <- sample(podzial)

for(a in 1:5){
  for(b in 1:5){
    for(d in 1:5){
      cc = 0
      for(i in 1:5){
        train_ind = which(podzial != i)
        train = edited_train[train_ind, ]
        walid = edited_train[-train_ind, ]
```

```

myDF <- data.frame(cbind(train, class = Y_train[train_ind]))

myNNet <- neuralnet(class ~.,
                      data = myDF,
                      hidden = c(a, b, d), rep = 3,
                      stepmax = 10000,
                      linear.output = FALSE,
                      threshold = 10,
                      lifesign = "minimal",
                      likelihood = TRUE,
                      act.fct = "logistic",
                      err.fct = "sse"
)
predict = neuralnet::compute(myNNet, walid)
predict = predict$net.result
predict = data.frame(predict)

##X_1 -> pstwo A
##X_2 -> pstwo B
predict$prawdziwe = Y_train[-train_ind]
predict$pred_val = ifelse(predict$X1 > 0.5, "A", "B")
cc = cc + mean(predict$prawdziwe == predict$pred_val)
}
vect = c(a, b, d)
print(vect)
cc = cc / 5
print(cc)
print(".....")
}
}
}
}

```

```

## [1] "Najlepsze parametry dla 3-elementowego hidden"

## [1] "dla których zmienna cc jest największa:"

## [1] "[1] a b c"

## [1] "[1] 3 4 3"

## [1] "Uśredniony ze wszystkich prób cv procent dobrze trafionych wyników"

## [1] "[1] 0.6552919"

```

Aby obliczyć predykcje dla edited\_test będziemy więc przyjmować hidden = [3, 4, 3] i skorzystamy z walidacji krzyżowej.

```

x1 = rep(0, nrow(edited_test))
for(i in 1:5){
  train_ind = which(podzial != i)
  train = edited_train[train_ind, ]

```

```

walid = edited_train[-train_ind, ]

myDF <- data.frame(cbind(train, class = Y_train[train_ind]))

myNNet <- neuralnet(class ~.,
                      data = myDF,
                      hidden = c(3, 4, 3), rep = 3,
                      stepmax = 10000,
                      linear.output = FALSE,
                      threshold = 2,
                      lifesign = "minimal",
                      likelihood = TRUE,
                      act.fct = "logistic",
                      err.fct = "sse"
)
predict_test = neuralnet::compute(myNNet, edited_test)
predict_test = predict_test$net.result
predict_test = data.frame(predict_test)

x1 = x1 + predict_test$X1
print("zrobiono")

}

#bierzemy średnią predykcję dla zbioru testowego na podstawie wykonanej walidacji krzyżowej
x1 = x1 / 5
x1 = data.frame(x1)
#zgłoszenie 2:
write.table(x1$x1, "zgloszenie2.txt", sep="\n", row.names=FALSE)

```