# Warsztaty Badawcze. Projekt 1

*Karol Prusinowski*

*14 października 2017*

## Wstępna analiza

Dane składają się z 50 zmiennych obiaśniających i jednej kolumny, która opisuje klasę. Próbek jest 50000. Każda próbka może należeć do `klasa +` lub `klasa -` i klasy te są zrównoważone. Pośród zmiennych są zmienne nominalne (posiadające 2 lub 4 różne wartości) oraz zmienne ilościowe przyjmujące wartości między 0 i 100, wszystkie o rozkładzie jednorodnym.

## Przygotowanie danych treningowych i testowych

Zmienne nominalne zostały przekształcone przy pomocy funkcji `model.matrix`, a dane zostały podzielone na zbiór testowy oraz treningowy w proporcjach 1:4.

```
data = data.frame(y = data$y, model.matrix(y ~., data)[,-1]);
indexes <- sample.int(nrow(data), 0.2*nrow(data));
train <- data[-indexes,];
test <- data[indexes,];
```

Modele będą testowane poprzez sprawdzenie jak wiele spośród 20% najwyżej ocenionych próbek należy do `klasa +`

```
test_model <- function(pred, t) {
  mean(t[order(pred, decreasing = TRUE)[1:(0.2*nrow(t))],'y'] =='klasa +')
}
```

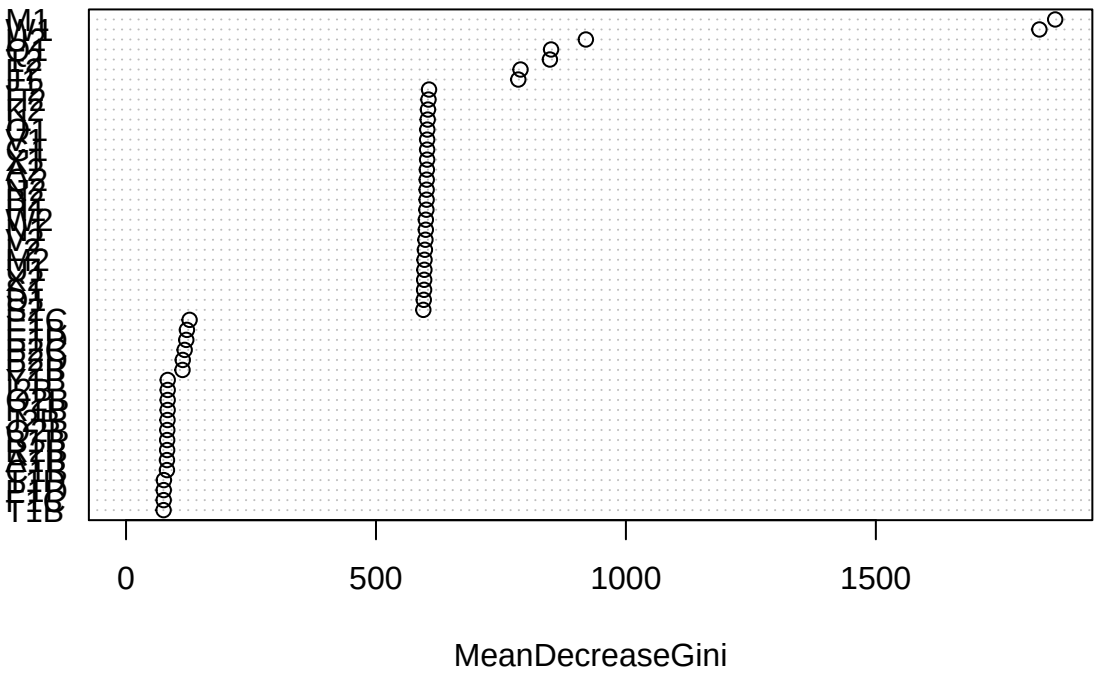
## Selekcja zmiennych

Pierwszą próbą wyboru zmiennych była funkcja `step` dla `glm` metodą `forward`, dla wszystkich danych.

```
modelGlm <- step(glm(y~0, data=data, family="binomial"), direction = "forward",
                 scope = formula(glm(y~.+0, data, family="binomial")), trace=FALSE)
stepCols <- names(modelGlm$coefficients)
```

Drugą sposobem było wykonanie klasyfikacji metodą randomForest i znalezienie istotnych zmiennych metodą `importance` oraz `varImpPlot`.

```
modelRF <- randomForest(y~., data=data)
imp <- importance(modelRF);
varImpPlot(modelRF, n.var = 50, main="Istotność zmiennych dla randomForest")
```




Na wykresie można zaobserwować duży skok ważności zmiennych około wartości 500, więc wybrane zostaną wszystkie zmienne dla których ta wartość jest większa niż 500.

```
rF1Cols = c(colnames(data[,-1])[imp>500]);
```

Można też zaobserwować wiele zmiennych blisko wartości 500, więc w kolejnej próbie wybrane zostało 7 zmiennych z wartością powyżej 700.

```
rF2Cols = c(colnames(data[,-1])[imp>700]);
```

Poniższy wykres przedstawia wybrane zmienne dla każdego sposobu.


## Klasyfikacja

Pierwszą metodą klasyfikacji jest funkcja glm dla każdego zbioru zmiennych.

```
glm1 <- glm(y~., data=train[,c(stepCols,"y")], family="binomial");
resGlm1 <- predict(glm1, test, type="response");
glm2 <- glm(y~., data=train[,c(rF1Cols,"y")], family="binomial");
resGlm2 <- predict(glm2, test, type="response");
glm3 <- glm(y~., data=train[,c(rF2Cols,"y")], family="binomial");
resGlm3 <- predict(glm3, test, type="response");

res <- data.frame(test_model(resGlm1, test),
                  test_model(resGlm2, test),
                  test_model(resGlm3, test))
colnames(res) <- c("glm dla stepCols",
                   "glm dla rF1Cols",
                   "glm dla rF2Cols")
kable(res)
```

| glm dla stepCols | glm dla rF1Cols | glm dla rF2Cols |
|---|---|---|
| 0.7165 | 0.6465 | 0.651 |

Można zauważyć, że największą dokładność osiągamy dla metody selekcji zmiennych `step`. jest to spodziewany wynik, gdyż te zmienne zostały wybrane tak, aby dostać jak najlepszy model funkcją regresji logistycznej.

Drugą metodą klasyfikacji jest funkcja randomForest.

```
rF1 <- randomForest(y~., data=train[,c(stepCols,"y")]);
resRF1 <- predict(rF1, test, type="prob")[,2];
rF2 <- randomForest(y~., data=train[,c(rF1Cols,"y")]);
resRF2 <- predict(rF2, test, type="prob")[,2];
rF3 <- randomForest(y~., data=train[,c(rF2Cols,"y")]);
resRF3 <- predict(rF3, test, type="prob")[,2];

res <- data.frame(test_model(resRF1, test),
                  test_model(resRF2, test),
                  test_model(resRF3, test))
colnames(res) <- c("randomForest dla stepCols",
                   "randomForest dla rF1Cols",
                   "randomForest dla rF2Cols")
kable(res)
```

| randomForest dla stepCols | randomForest dla rF1Cols | randomForest dla rF2Cols |
|---:|---:|---:|
| 0.8185 | 0.799 | 0.7905 |

Wyniki są lepsze niż dla prostego `glm`, a pierwsza metoda selekcji zmiennych wciąż daje najlepsze wyniki.

Ostatnią metodą jest XGBoost.

```
xg1 <- xgboost(data.matrix(train[,stepCols]), label=as.numeric(train[,"y"]) - 1,
               objective="binary:logistic", nrounds= 20, verbose=FALSE)
resXG1 <- predict(xg1, as.matrix(test[,stepCols]))
xg2 <- xgboost(data.matrix(train[,c(rF1Cols)]), label=as.numeric(train[,"y"]) - 1,
               objective="binary:logistic", nrounds= 20, verbose=FALSE)
resXG2 <- predict(xg2, as.matrix(test[,rF1Cols]))
xg3 <- xgboost(data.matrix(train[,rF2Cols]), label=as.numeric(train[,"y"]) - 1,
               objective="binary:logistic", nrounds= 20, verbose=FALSE)
resXG3 <- predict(xg3, as.matrix(test[,rF2Cols]))

res <- data.frame(test_model(resXG1, test),
                  test_model(resXG2, test),
                  test_model(resXG3, test))
colnames(res) <- c("xgboost dla stepCols",
                   "xgboost dla rF1Cols",
                   "xgboost dla rF2Cols")
kable(res)
```

| xgboost dla stepCols | xgboost dla rF1Cols | xgboost dla rF2Cols |
|---:|---:|---:|
| 0.825 | 0.8005 | 0.7995 |

Najwyższy wynik otrzymany został dla xgboost i wyboru zmiennych metodą step, wiec taki model zostanei wykorzystany do przydzielenia wyników zbiorowi testowemu.