Statystyka wielowymiarowa
Laboratorium nr 4

Kamil Szkoła
Patryk Krukowski
Data Science, sem. 1

18 czerwca 2022

Spis treści

1 Wstęp

Naszym zadaniem jest przeanalizowanie zbiorów *Life_expectancy.csv* (zadanie regresji) oraz *Titanic.csv* (zadanie klasyfikacji) w kontekście zastosowania następujących modelów:

- Drzewa decyzyjne (regresyjne),
- Lasy losowe,
- Boosting.

2 Regresja

Przejdziemy teraz do analizy zbioru *Life_expectancy.csv*. Zmienną objaśnianą jest zmienna *Life.expectancy*. Załadujmy i przygotujmy dane.

```
> # Przygotowanie danych
> # Life expectancy
> lf <- read.csv('life_expectancy.csv')
> lf <- na.omit(lf)
> lf <- subset(lf, select = -c(Country, Year))
> lf$Status <- factor(lf$Status)
```

Przeanalizujemy teraz następujące modele:

2.1 Regresja

Drzewo regresyjne - dopasowujemy model.

```
> lf_tree <- tree(Life.expectancy ~ ., data = lf)
> summary(lf_tree)
```

```
Regression tree:
tree(formula = Life.expectancy ~ ., data = lf)
Variables actually used in tree construction:
[1] "Income.composition.of.resources" "HIV.AIDS"
[3] "Adult.Mortality"                 "thinness..1.19.years"
Number of terminal nodes:  10
Residual mean deviance:  9.704 = 15900 / 1639
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-13.61000  -1.87200   0.08767   0.00000   1.72100  14.97000
```

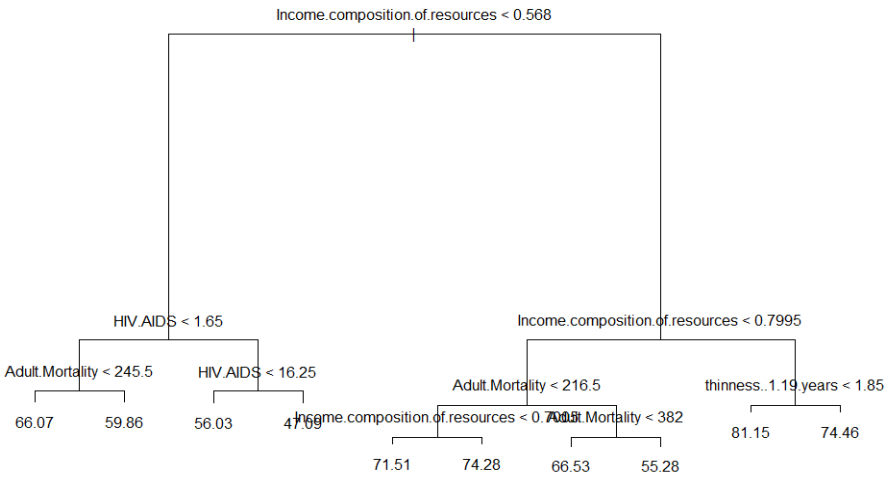
Istotne predyktory według dopasowanego modelu:

- Income.composition.of.resources,
- HIV.AIDS,
- Adult.Mortality,
- thinness..1.19.years.

Zwizualizujmy drzewo.

Ewaluowany model zinterpretujemy po przycięciu drzewa metodą sterowaną złożonością. Estymujemy błąd testowy, uprzednio dzieląc dostępny zbiór danych na część treningową i testową.

```
> set.seed(1)
> lf_train <- sample(nrow(lf), nrow(lf) * 0.75)
> lf_test <- -lf_train
> lf_train_tree <- tree(Life.expectancy ~ Income.composition.of.resources +
+                         HIV.AIDS +
+                         Adult.Mortality +
+                         thinness..1.19.years, data = lf, subset = lf_train)
> lf_pred <- predict(lf_train_tree, newdata = lf[lf_test,])
> mean((lf_pred - lf$Life.expectancy[lf_test])^2)
```



Rysunek 1: Drzewo regresyjne dla analizowanego zbioru danych

```
[1] 10.12635
```

Wyznaczamy przycięte (optymalne) drzewo, korzystając z metody walidacji krzyżowej.

Naszym zdaniem optymalny rozmiar drzewa to 5. Przycinamy.

```
> lf_train_pruned <- prune.tree(lf_train_tree, best = 5)
```

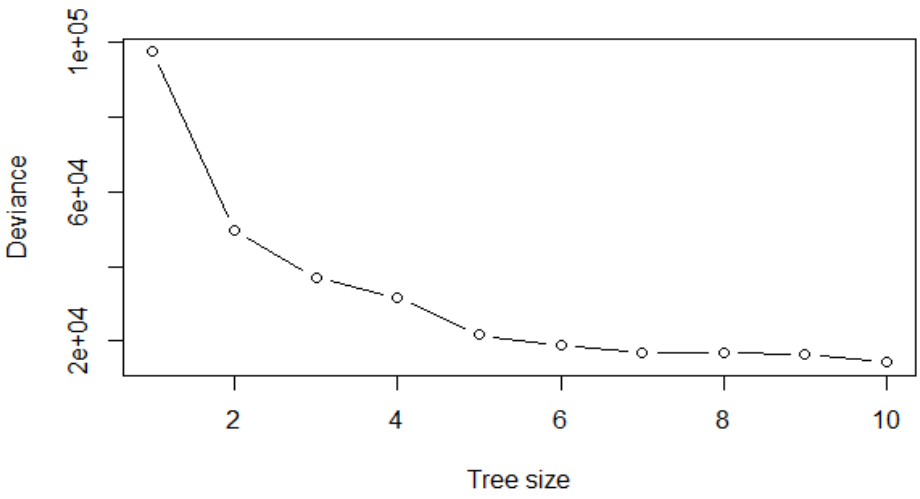
Wizualizujemy.

Estymujemy błąd testowy, licząc błąd średniokwadratowy.

```
> lf_pred_pruned <- predict(lf_train_pruned, newdata = lf[lf_test,])
> mean((lf_pred_pruned - lf$Life.expectancy[lf_test])^2)
```

```
[1] 16.22373
```

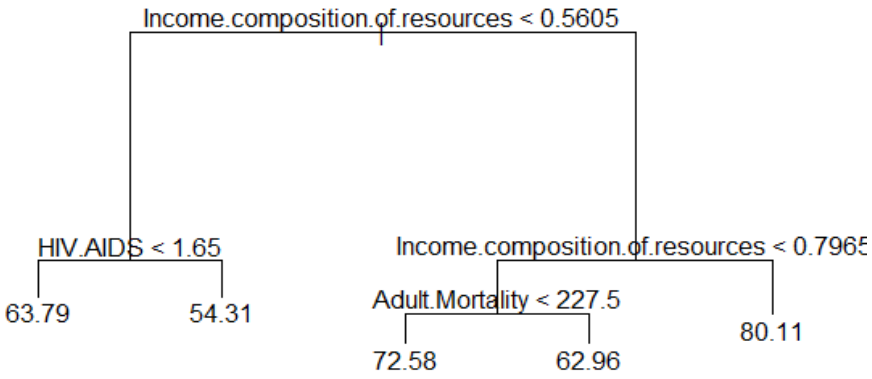
```
>
```



Rysunek 2: Wybór optymalnego rozmiaru drzewa regresyjnego

Najbardziej znaczącym predyktorem okazała się zmienna *Income.composition.of.resources* (rysunek nr 3). Jest to skład ogólnego dochodu danej grupy ludności. Widzimy, że największa oczekiwana długość życia wynosi 80.11 i jest to wartość wyestymowana dla grupy ludzi, których *Income.composition.of.resources* przekracza wartość 0.8. Oczywistym jest, że zmienna *HIV.AIDS* (śmierć / 1000 urodzeń) ma istotny wpływ na oczekiwaną długość życia. W zależności od wartości tego współczynnika, oczekiwana długosć życia waha się pomiędzy wartościami 54.31 a 63.79. Zmienna *Adult.Mortality* oznacza liczbę śmierci ludzi pomiędzy 15 a 60 rokiem życia.

Co prawda, błąd testowy jest większy niż w przypadku MSE poprzedniego drzewa, ale zyskujemy większą interpretowalność.



Rysunek 3: Optymalne drzewo regresyjne

2.2 Lasy losowe

Dopasowujemy model lasów losowych do części treningowej oraz wyznaczamy istotne zmienne.

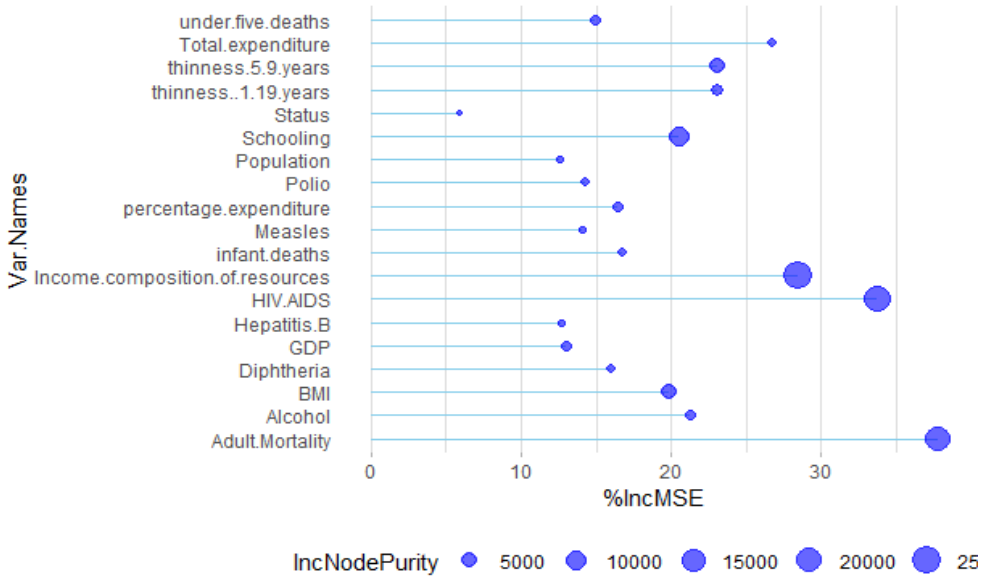
```
> set.seed(2)
> lf_rf_train <- randomForest(Life.expectancy ~ ., data = lf, subset = lf_train,
+                             importance = TRUE)
> lf_imp_data <- as.data.frame(importance(lf_rf_train))
> lf_imp_data$Var.Names <- row.names(lf_imp_data)
```

Wykres nr 4 odczytujemy następująco - wybieramy te zmienne, dla których *%IncMSE* jest najwyższe, gdyż oznacza to, że błąd MSE rośnie o daną wartość, gdy dana zmienna zostanie usunięta. Rozmiar węzłów jest podyktowany przez *czystość węzłów (IncNodePurity)*.

Istotne zmienne:

- Income.composition.of.resources,
- HIV.AIDS,
- Adult.Mortality,
- Total.expenditure.

Dopasowujemy nowy model (dla 500 drzew) oraz liczymy estymatę błędu testowego.



Rysunek 4: Zmienne istotne dla modelu lasu losowego

```
> lf_rf_train <- randomForest(Life.expectancy ~ Income.composition.of.resources +
+                               HIV.AIDS +
+                               Adult.Mortality +
+                               Total.expenditure,
+                             data = lf, subset = lf_train,
+                             importance = TRUE,
+                             ntree = 500)
> lf_rf_pred <- predict(lf_rf_train, newdata = lf[lf_test,])
> mean((lf_rf_pred - lf$Life.expectancy[lf_test])^2)
```

```
[1] 4.93307
```

Otrzymaliśmy znacznie mniejszy błąd średniokwadratowy niż w przypadku drzew regresyjnych.

2.3 Boosting

Dopasowujemy model i wykonujemy na nim funkcję *summary*.

```
> lf_boost_train <- gbm(Life.expectancy ~ ., data = lf[lf_train,],
+                       distribution = "gaussian",
+                       n.trees = 5000,
+                       interaction.depth = 4)
> summary(lf_boost_train)
```

```
                                                                 var     rel.inf
Income.composition.of.resources Income.composition.of.resources 44.68410043
HIV.AIDS                                                 HIV.AIDS 22.49506226
Adult.Mortality                                   Adult.Mortality 17.27788134
thinness.5.9.years                           thinness.5.9.years  2.08853065
Total.expenditure                             Total.expenditure  1.81339219
Schooling                                             Schooling  1.79632225
Alcohol                                                 Alcohol  1.44712660
percentage.expenditure                   percentage.expenditure  1.26088524
Population                                           Population  1.09603817
GDP                                                         GDP  0.92535591
BMI                                                         BMI  0.79119574
thinness..1.19.years                       thinness..1.19.years  0.76508276
Hepatitis.B                                         Hepatitis.B  0.65630871
under.five.deaths                             under.five.deaths  0.61947572
Polio                                                     Polio  0.61090564
Measles                                                 Measles  0.55607142
Diphtheria                                           Diphtheria  0.54020111
infant.deaths                                     infant.deaths  0.52180069
Status                                                   Status  0.05426318
```

Dopasowujemy teraz model Boosting jedynie do zmiennych, które są istotne. Liczymy estymatę błędu testowego. Zwracamy uwagę na to, że distribution = "gaussian", gdyż mamy do czynienia z problemem regresji.

```
> lf_new_boost_train <- gbm(Life.expectancy ~ Income.composition.of.resources +
+                           HIV.AIDS +
+                           Adult.Mortality,
+                         data = lf[lf_train,],
+                         distribution = "gaussian",
+                         n.trees = 5000,
+                         interaction.depth = 4)
> lf_new_pred_boost <- predict(lf_new_boost_train, newdata = lf[lf_test,], n.trees = 5000)
> # MSE
> mean((lf_new_pred_boost - lf$Life.expectancy[lf_test])^2)

[1] 5.516222
```

Błąd jest podobnej wielkości, co dla lasu losowego. Zwróćmy uwagę, że wykorzystywanych drzew jest 5000. Możemy sobie na to pozwolić, gdyż modelu tego (jak i lasu losowego) nie można przeuczyć.

3 Klasyfikacja

Tym razem analizować będziemy zbiór *Titanic.csv* w kontekście problemu klasyfikacji. Zmienną objaśnianą jest zmienna *Survived*. Przejdźmy do dopasowania wspomnianych wcześniej modeli.

```
> # Przygotowanie danych
> titanic <- read.csv('titanic.csv')
> # Usuwamy wartosci zakodowane jako NA
> titanic <- na.omit(titanic)
> # Poprawiamy kodowanie zmiennej Age
> titanic <- titanic[titanic$Age <= 3,]
> # Usuwamy niepotrzebna zmienna
> titanic$X <- NULL
> titanic$Survived <- as.factor(titanic$Survived)
```

3.1 Drzewo klasyfikacyjne

Dopasowujemy model.

```
> ti_tree <- tree(Survived ~ ., data = titanic)
> summary(ti_tree)
```

```
Classification tree:
tree(formula = Survived ~ ., data = titanic)
Variables actually used in tree construction:
[1] "Title"      "Pclass"     "Has_Cabin"  "FamilySize"
Number of terminal nodes:  7
Residual mean deviance:  0.7952 = 694.2 / 873
Misclassification error rate: 0.1636 = 144 / 880
```

Identyfikujemy istotne zmienne:

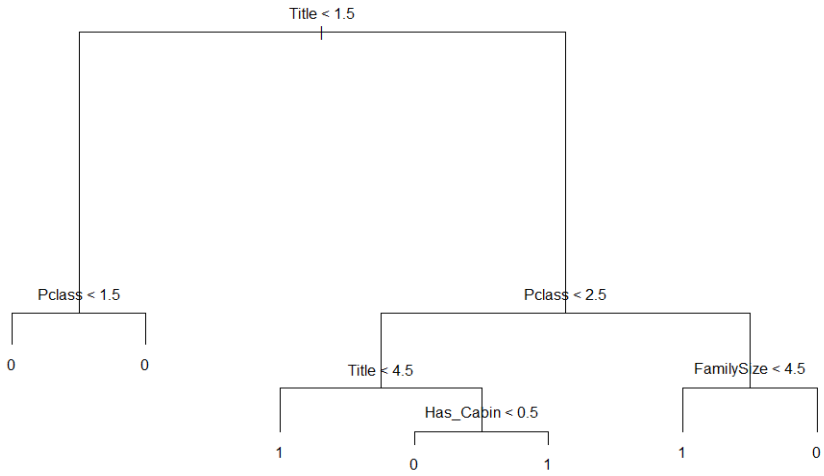
- Title,
- Pclass,
- Has_Cabin,
- FamilySize.

Wizualizujemy drzewo decyzyjne.

Do drzewa figurującego na rysunku nr 5 nie zastosowaliśmy przycinania, ponieważ model ten jest, naszym zdaniem, wysoce interpretowalny.

Najważniejszym predyktorem okazała się zmienna *Title* oznaczająca tytuł danego pasażera (im wyższa wartość, tym ważniejszy tytuł). Jeżeli tytuł ten był niższy od pewnego progu, to ewaluowany model, niezależnie od *Pclass* (rodzaj biletu), przewidywał śmierć takiej osoby. Jeśli natomiast zmienna *Title* przekroczyła wspomniany próg i jeżeli rodzaj biletu przekroczył pewną wartość wraz z wielkością rodziny co najwyżej równą 4, to model przewiduje przeżycie danej osoby, a w przeciwnym przypadku śmierć.

Możemy wyciągnąć z tego następujący wniosek - z uwagi na ograniczoną liczbę szalup, mogła z nich skorzystać jedynie część osób, w związku z tym możemy zaobserować negatywną odpowiedź modelu w kontekście przeżycia danej osoby, jeżeli miała ona wieloczłonkową rodzinę.



Rysunek 5: Drzewo decyzyjne

Osoby z nisko klasowymi biletami i z tytułem niższym niż wartość 4.5, przeżywały. Stąd wniosek, że, być może, osobami o niższych tytułach były kobiety, jako że wraz z dziećmi miały one pierwszeństwo dostępu do szalup. Oczywiście odpowiednio wysoki tytuł znacząco mógł ułatwić (i ułatwiał) ewakuację.

Jeżeli dana osoba nie miała kabiny ($Has\_Cabin = 0$), to (mimo wysokiego tytułu!) nie przeżywała katastrofy. Natomiast jeżeli miała prywatną szalupę, to mogła się uratować, co wydaje się dość oczywistym wnioskiem.

Liczymy accuracy modelu oraz wyznaczamy macierz pomyłek.

```
> # Accuracy
> set.seed(1)
> ti_train <- sample(nrow(titanic), nrow(titanic) * 0.75)
> ti_test <- -ti_train
> ti_train_tree <- tree(Survived ~ Title +
+                         Pclass +
+                         Has_Cabin +
+                         FamilySize,
+                       data = titanic,
+                       subset = ti_train)
> ti_pred <- predict(ti_train_tree, newdata = titanic[ti_test,], type = 'class')
> summary(ti_pred)

  0   1
147  73
```

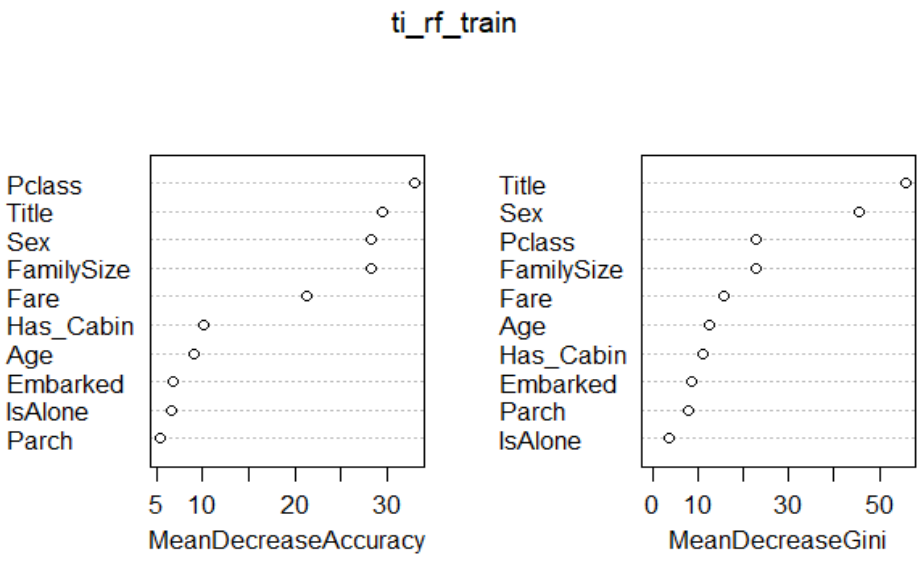
```
> table(ti_pred, titanic$Survived[ti_test])

ti_pred   0   1
      0 127  20
      1  17  56

> # Accuracy
> mean(ti_pred == titanic$Survived[ti_test])

[1] 0.8318182
```

Możemy zaobserwować wysoką dokładność modelu.

3.2 Lasy losowe

Dopasowujemy model i identyfikujemy istotne zmienne.

```
> ti_rf_train <- randomForest(Survived ~ ., data = titanic, subset = ti_train,
+                             importance = TRUE)
```



Rysunek 6: Las losowy

MeanDecreaseAccuracy z wykresu nr 6 odczytujemy następująco - usunięcie danej zmiennej spowoduje spadek dokładności modelu o zadaną na wykresie (oś OY) wartość.

Istotne zmienne:

- Pclass,
- Title,
- Sex,
- FamilySize,
- Fare.

Dopasowujemy nowe drzewo ze wspomnianymi powyżej zmiennymi oraz liczymy dokładność odpowiedzi modelu.

```
> # Dopasowanie nowego drzewa
> ti_rf_new_train <- randomForest(Survived ~ Pclass + Title +
+                                 FamilySize +
+                                 Fare,
+                                 data = titanic, subset = ti_train,
+                                importance = TRUE,
+                                ntree = 500)
> ti_rf_new_pred <- predict(ti_rf_new_train, newdata = titanic[ti_test,])
> mean(ti_rf_new_pred == titanic$Survived[ti_test])

[1] 0.8227273
```

Accuracy jest zadowalająco wysokie.

3.3 Boosting

Przygotowujemy odpowiednio zmienną *Survived*, a następnie dopasowujemy model boostingu, w którym ustawiamy parametr *distribution* na *bernoulli*, gdyż mamy do czynienia z klasyfikacją binarną.

```
> # Boosting
> titanic$Survived <- as.vector(droplevels(titanic$Survived))
> ti_boost_train <- gbm(Survived ~ ., data = titanic[ti_train,],
+                       distribution = "bernoulli",
+                       n.trees = 5000,
+                       interaction.depth = 4)
> summary(ti_boost_train)

                  var   rel.inf
Fare             Fare 18.097447
Title           Title 17.487424
Age               Age 12.712337
Embarked     Embarked 12.460456
Pclass         Pclass 11.887450
FamilySize FamilySize  9.580708
Sex               Sex  5.509664
```

```
Parch           Parch  4.920148
Has_Cabin   Has_Cabin  4.466467
IsAlone       IsAlone  2.877898
```

Istotnymi zmiennymi okazały się:

- Fare,
- Title,
- Age,
- Embarked,
- Pclass.

Wyznaczamy część treningową i testową, realizujemy czytelniejsze kodowanie zmiennej *Survived* i dopasowujemy model do istotnych zmiennych, wymienionych powyżej.

```
> ti_train_data <- titanic[ti_train,]
> ti_test_data <- titanic[ti_test,]
> ti_train_data$Survived2 <- ifelse(ti_train_data$Survived==1,'yes','no')
> ti_test_data$Survived2 <- ifelse(ti_test_data$Survived == 1, 'yes', 'no')
> objControl <- trainControl(method='cv', number=10, returnResamp='none',
+                            summaryFunction = twoClassSummary,
+                            classProbs = TRUE)
> ti_new_boost_train <- train(ti_train_data[,c('Fare', 'Title', 'Embarked', 'Pclass')],
+                   ti_train_data[,'Survived2'],
+                   method='gbm',
+                   trControl = objControl,
+                   metric = "ROC",
+                   preProc = c("center", "scale"))
```

Liczymy dokładność odpowiedzi modelu.

```
> ti_new_pred_boost <- predict(ti_new_boost_train,
+                              newdata = titanic[ti_test,],
+                              n.trees = 5000,
+                              type = 'raw')
> mean(ti_new_pred_boost == ti_test_data$Survived2)

[1] 0.6863636
```

Acccuracy dla tego modelu nie jest zadowolające.