Aleksandra Olechowska

2025-05-18

Użyte biblioteki:

```
library(dplyr)
library(ggplot2)
library(stats4)
library(MASS)
library(glmnet)
library(randomForest)
library(gplots)
```

1. Eksploracja:

A)

Wczytanie plików do RStudio

```
X_train <- read.csv("C:/Users/Ola/Desktop/Semestr letni 2025/SAD/X_train.csv")
X_test <- read.csv("C:/Users/Ola/Desktop/Semestr letni 2025/SAD/X_test.csv")
y_train <- read.csv("C:/Users/Ola/Desktop/Semestr letni 2025/SAD/y_train.csv")

obs_var <- data.frame(
  Nazwa = c("X_train", "X_test", "y_train"),
  Liczba_obserwacji = c(nrow(X_train), nrow(X_test), nrow(y_train)),
  Liczba_zmiennych = c(ncol(X_train), ncol(X_test), ncol(y_train)),
  Liczba_pełnych_obserwacji = c(nrow(na.omit(X_train)), nrow(na.omit(X_test)), nrow(na.omit(y_train)))
)
knitr::kable(obs_var, caption = "Dane o liczbie obserwacji i zmiennych w zbiorach danych")
```

Table 1: Dane o liczbie obserwacji i zmiennych w zbiorach danych

Nazwa	Liczba_obserwacji	Liczba_zmiennych	Liczba_pełnych_obserwacji
X_train	6800	9000	6800
X_test	1200	9000	1200
y_train	6800	1	6800

Dane są pełne, cieszmy się.

B)

```
# Poznać nazwę zmiennej, którą chcęopisać
cat("Nazwa białka powierzchniowego", colnames(y_train))
```

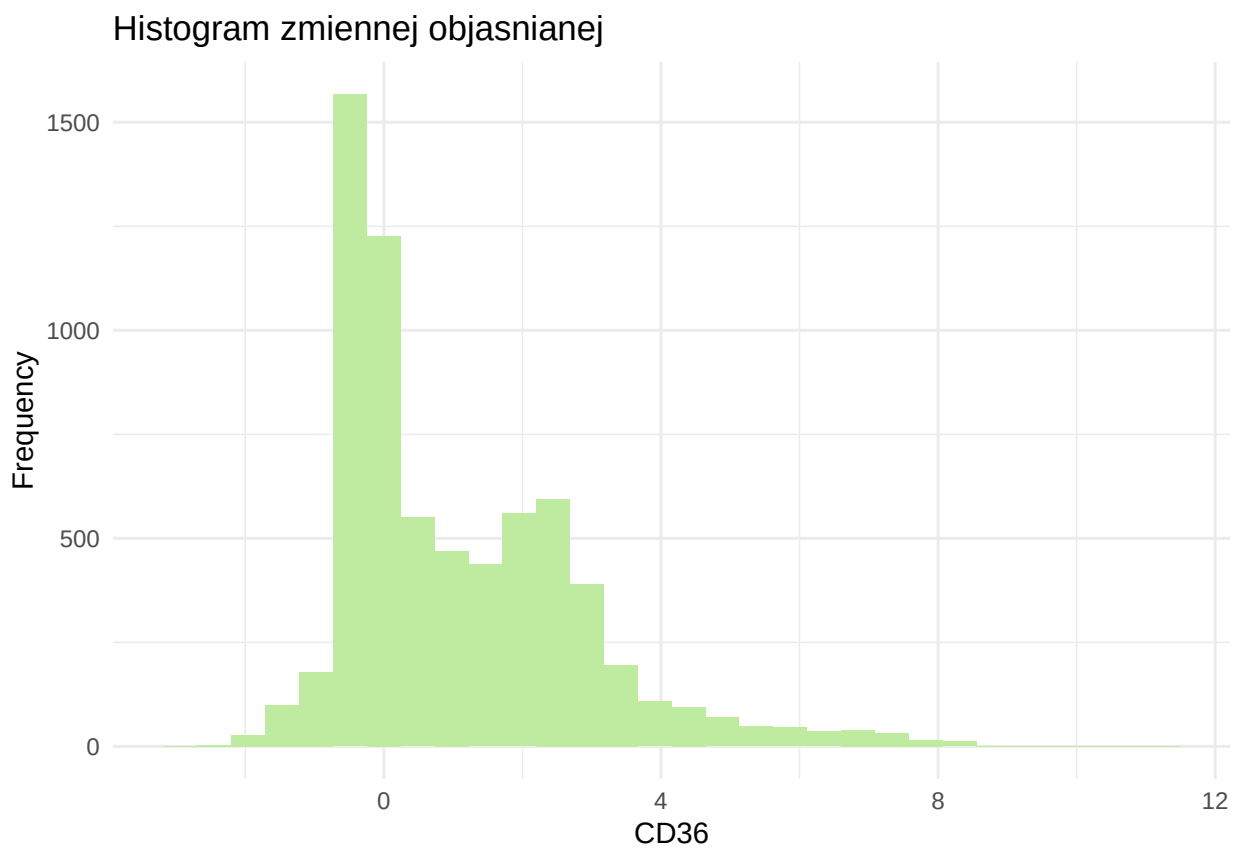
```
## Nazwa białka powierzchniowego CD36
```

```r
# zmienna z podstawowymi statystykami opisowymi
stats <- y_train %>% summarise(
  Średnia = mean(CD36),
  Mediana = median(CD36),
  Minimum = min(CD36),
  Maksimum = max(CD36),
  Odchylenie_standardowe = sd(CD36))

# Przedstawienie danych w postaci tabeli do Markdown
knitr::kable(stats, caption = "Statystyki opisowe iości białka CD36")
```

Table 2: Statystyki opisowe iości białka CD36

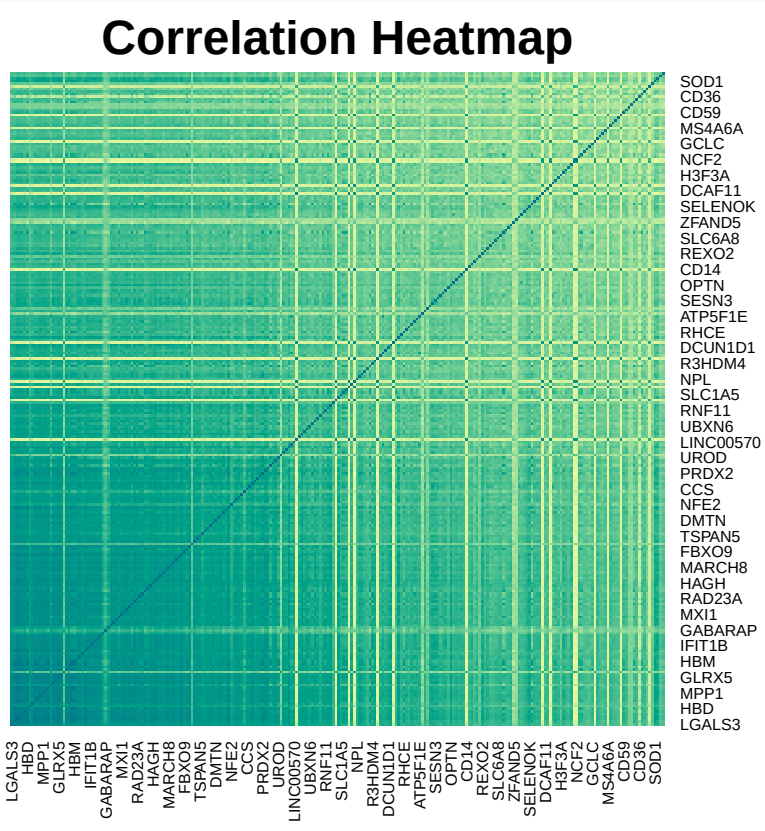
Średnia	Mediana	Minimum	Maksimum	Odchylenie_standardowe
1.06348	0.4714118	-2.823421	11.35048	1.758311

```r
# Histogram zmiennej CD36 z pliku train_y.csv
ggplot(y_train, aes(CD36, fill = "oganre")) +
  geom_histogram(bins = 30, fill = "#BEEB9F") + ggtitle("Histogram zmiennej objaśnianej") +
  ylab("Frequency") +
  theme_minimal()
```



C)

```r
# Tworzymy wektor z korelacjami między X_train i y_train i nawazmi kolumn
cor_vals <- sapply(X_train, function(col) cor(col, y_train))

# Wybieramy pierwsze 250 nazw kolumn z X_train
top_250_names <- names(sort(cor_vals, decreasing = TRUE))[1:250]

# Wybieramy kolumny z nazwami z top_250_names
top_250_x_train <- X_train[, top_250_names]

# Tworzymy macierz korelacji
cor_matrix <- cor(top_250_x_train)

# Tworzymy heatmapę
heatmap(cor_matrix,
        Colv = NA,         # Wyłącza grupowanie w kolumnach
        Rowv = NA,         # Wyłącza gurpowanie w wierszach
        col = colorRampPalette(c("#FFFF9D", "#BEEB9F", "#00A388", "#03738C"))(200),
        scale = "none",
        main = "Correlation Heatmap")
```

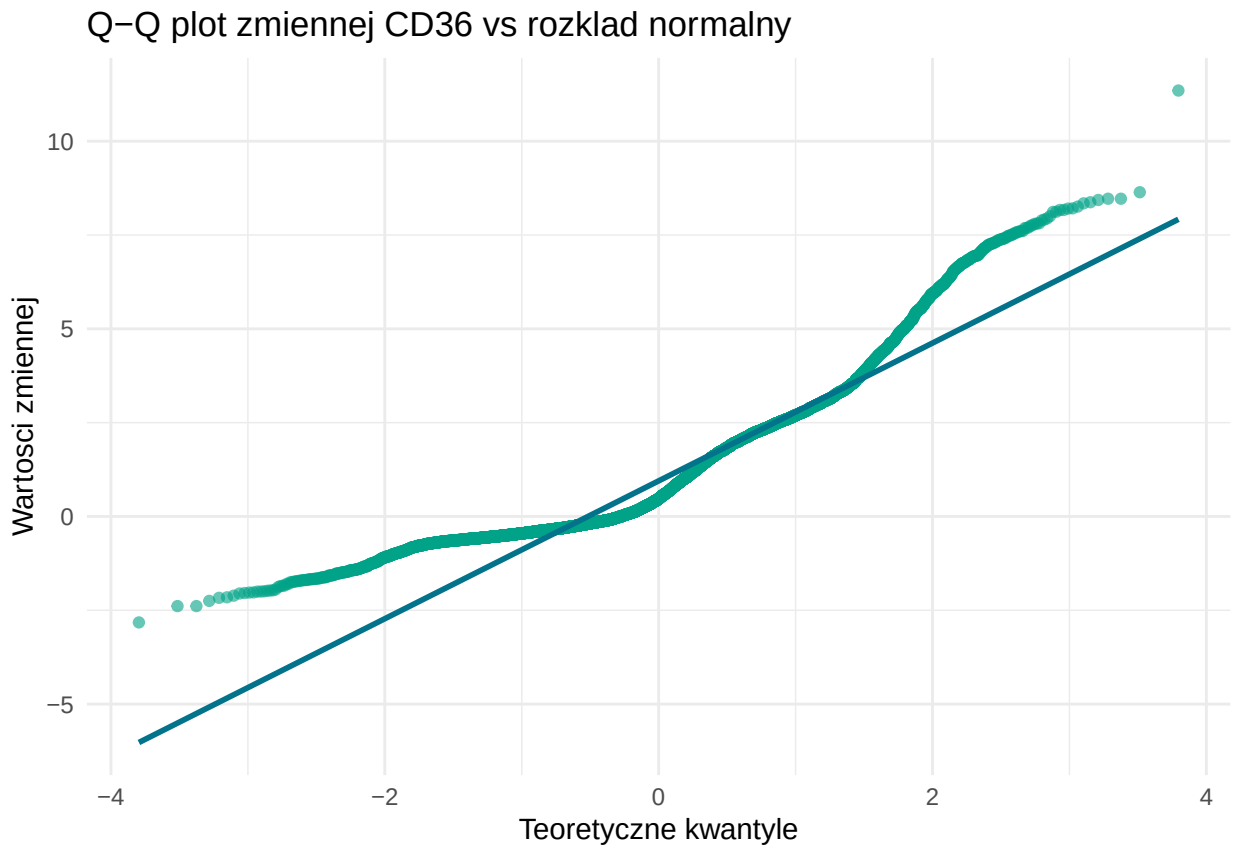


Jak widać kolor ciemniejszy oznacza większą korelację, a bardziej żółty mniejszą wartości korelacji. Wykres nie bierze wartości bez wartości bezwzględnej.

2. Testy statystyczne:

A)

```
ggplot(y_train, aes(sample = CD36)) +
  stat_qq(color = "#00A388", alpha = 0.6) +
  stat_qq_line(color = "#03738C", linewidth = 1) +
  labs(title = "Q-Q plot zmiennej CD36 vs rozkład normalny",
       x = "Teoretyczne kwantyle",
       y = "Wartości zmiennej") +
  theme_minimal()
```



Jak standaryzuje się wyniki obserwacji, to mamy $z = (x - \mu)/\sigma$. Na wykresie przedstawiona jest prosta, z której pochodziłyby wyniki, gdyby byłyby z rozkładu normalnego. Zatem prosta ma wzór $y = x * \sigma + \mu$. Czyli dla wartości x=0 funkcja przyjmuje wartość średniej zmiennej CD36. Nachylenie prostej natomiast to oschylenie standardowe CD36. To już trudniej wyczytać z wykresu. Zatem informacje takie jak odchylenie standardowe i średnia są zawarte na wykresie, da się je odczytać, ale nie z dużą dokładnością.

B)

H1: Rozkład zmiennej objasnianej różni się od rozkładu normalnego. H0: Rozkład zmiennej objaśnianej nie różni się od rozkładu normalnego. Ze względu na dużą liczeboność grupy wybrano test Kołmogorova-Smirnova.

```
# zmiana klasy, aby można było użyć funkcji ks.test
cd36 <- y_train[[1]]

#test Kołmogorowa-Smirnova
ks_test <- ks.test(cd36, "pnorm")
```

```r
# Ładna tabelka na przedstawienie wyników
ks_results1 <- data.frame(
  Test = "Kolmogorov-Smirnov",
  Statistic = ks_test$statistic,
  pvalue = format(ks_test$p.value, scientific = TRUE)
  # Formatowanie dla małych wartości p-value
)

knitr::kable(ks_results1, caption = "Wyniki testu Kołmogorowa-Smirnowa dla
             zmiennej objaśnianej")
```

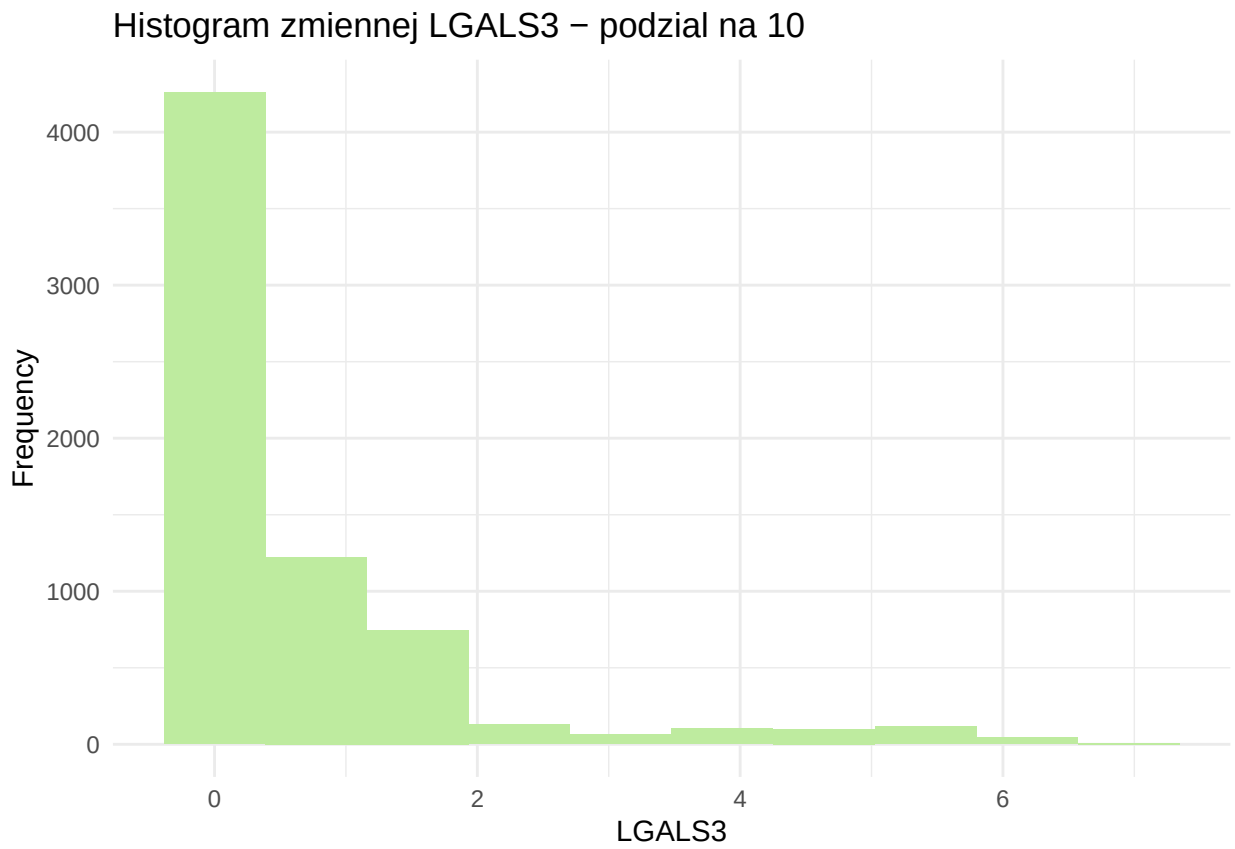
Table 3: Wyniki testu Kołmogorowa-Smirnowa dla zmiennej objaśnianej

	Test	Statistic	pvalue
D	Kolmogorov-Smirnov	0.2924667	0e+00

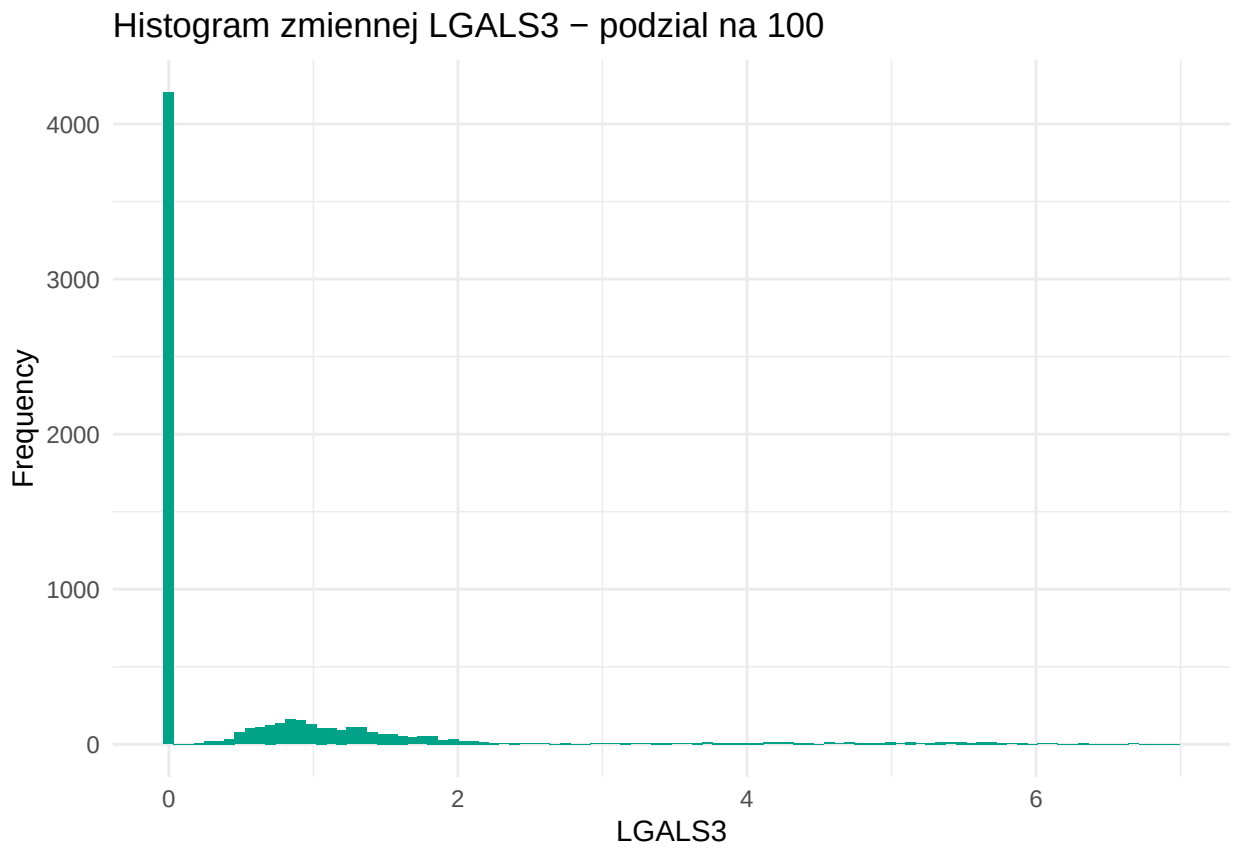
P-value jest bardzo małe. Zatem na poziomie istotności 0.05 są podstawy do odrzucenia H0. Przyjmujemy hipotezę alternatywną, że zmienna objaśniana nie ma rozkładu normalnego.

C) i)

```r
# Korzystam z stworzonej wcześniej zmiennej cor_vals, która zawiera korelacje
# między zmiennymi objaśniającymi i zmienną objaśnianą
# Wybieramy pierwszą nazwę kolumny z cor_vals
top_name <- names(sort(cor_vals, decreasing = TRUE))[[1]]

# cat("Najbardziej skorelowana zmienna nazywa się:",  top_name)

# Wybieramy kolumnę z nazwą z top_name
top_x_train <- X_train[, top_name]

# Histogramy dla ziennej, aby zobaczyć, co w ogóle może pasować
ggplot(data.frame(top_x_train), aes(top_x_train)) +
  geom_histogram(bins = 10, fill = "#BEEB9F") +
  ggtitle("Histogram zmiennej LGALS3 - podział na 10") +
  ylab("Frequency") + xlab("LGALS3") +
  theme_minimal()
```




```
ggplot(data.frame(top_x_train), aes(top_x_train)) +
  geom_histogram(bins = 100, fill = "#00A388") +
  ggtitle("Histogram zmiennej LGALS3 - podział na 100") +
  ylab("Frequency") + xlab("LGALS3") +
  theme_minimal()
```



Po tym jak wyglądają histogramy, to ewidentnie chcemy dpasować rozkład, który jest tylko liczb dodatnich i jest duże prawdopodobieństwo otrzymania małej liczy i wraz ze wzrostem wartości prawdopodobieństwo jej otrzymania maleje. Takie wymagania spełnia np. rozkład wykładniczy. Aby sprawdzić jego dopasowanie, najpierw obliczymy MLE, dla jego parametru. Potem sprawdzimy dopasowanie rozkładu.

```
# Używamy zmiennej cor_vals z wcześniejszego zadania
# Bierzemy nazwę najbardziej skorelowanej zmiennej
top_var_name <- names(which.max(cor_vals))
cat("Najbardziej skorelowana zmienna objaśniająca z objaśnianą nazywa się",
    top_var_name)
```

```
## Najbardziej skorelowana zmienna objaśniająca z objaśnianą nazywa się LGALS3
```

```
# Obcinamy macierz X_train do kolumny nabardziej skorelowanej z zmienną objaśnianą
top_var <- X_train[[top_var_name]]

# Obcinamy wektor tylko do wartości nieujemnych
top_var <- top_var[top_var >= 0]

# Zakładając, że zmienne pochodzą z rozkładu wykładniczego,
# Dopasowuję mle dla parametru tego rozkładu
fit_exp <- fitdistr(top_var, densfun = "exponential")
cat("Lambda_kat wynosi ", fit_exp[[1]])
```

```
## Lambda_kat wynosi  1.533841
```

H1: Rozkład zmiennej różni się od rozkładu wykłądniczego z parametrem Lambda = 1.533841. H0: Rozkład zmiennej objaśnianej nie różni się od rozkładu wykładniczego z parametrem Lambda = 1.533841.

```r
# Sprawdzenie za pomocą testu Kołmogorova-Smirnova jak dobrze dane są dopasowane
# do rozkładu wykładniczego z paramterem MLE
lambda_hat <- fit_exp$estimate[[1]]
ks_exp <- ks.test(top_var, "pexp", rate = lambda_hat)

# Przedstawienie wyników w tabeli
ks_results_exp <- data.frame(
  Test = "K-S Test",
  Statistic = ks_exp$statistic,
  `p-value` = format(ks_exp$p.value, scientific = TRUE)) # Formatowanie
  #bardzo małych wartości p-value

knitr::kable(ks_results_exp, caption = "Wyniki testu Kołmogorowa-Smirnowa
             dla dopasowania rozkładu Wykładniczego")
```

Table 4: Wyniki testu Kołmogorowa-Smirnowa dla dopasowania rozkładu Wykładniczego

	Test	Statistic	p.value
D	K-S Test	0.6182353	0e+00

P-value jest bardzo małe, więc mamy podstawy do odrzucenia hipotezy zerowej o braku różnicy między rozkładem wykładniczym i rozkładem, z którego pochodzą dane. Estymator wyszedł zaskakująco mały, ale wydaje mi się, że nie ma błędu w obliczeniach.

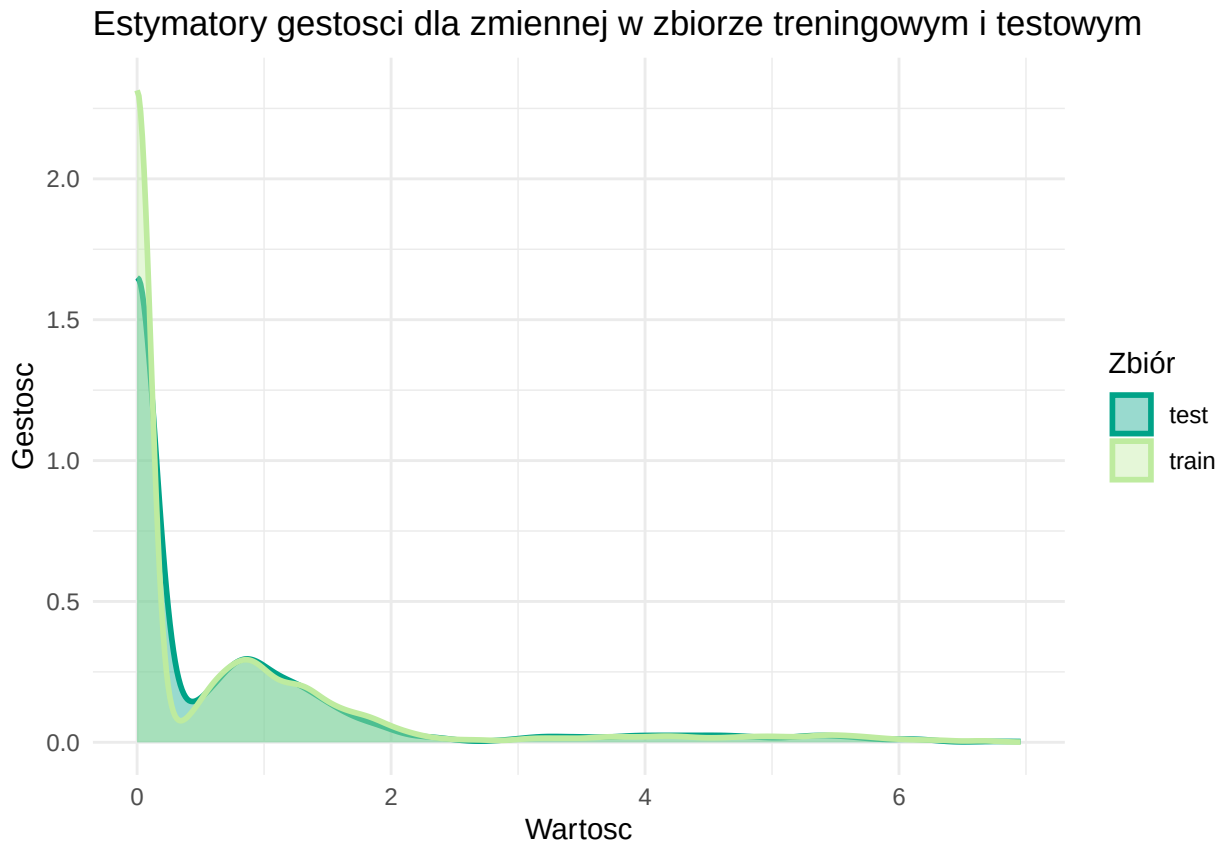
C) ii)

H0: Zmienna LGALS3 ma podobny rozkład w zbiorze testowym i treningowym. To znaczy, że wartości median (lub średnich) nie różnią się. H1: Lokalizację median (lub średnich) zmiennej w zbiorze testowym i treningowym różni się

```r
# Wektor z wartościami zmiennej LGALS3 z zbioru testowego
top_x_test <- X_test[, top_name]

# Tworzymy data.frame złożoną z kolumny LGALS3 z x_train i x_test
top_x_matrix <- data.frame(
  value = c(top_x_train, top_x_test),
  set = c(rep("train", length(top_x_train)), rep("test", length(top_x_test)))
)
```

Zróbmy wykres estymatora gęstości dla obu zmiennych, aby mieć pogląd na to, czy w ogóle możliwym jest aby były podobne.

```r
ggplot(top_x_matrix, aes(x = value, fill = set, color = set)) +
  geom_density(alpha = 0.4, size = 1) +
  scale_fill_manual(values = c("train" = "#BEEB9F", "test" = "#00A388")) +
  scale_color_manual(values = c("train" = "#BEEB9F", "test" = "#00A388")) +
  labs(
    title = "Estymatory gęstości dla zmiennej w zbiorze treningowym i testowym",
    x = "Wartość", y = "Gęstość", fill = "Zbiór", color = "Zbiór"
  ) +
  theme_minimal()
```



Wyglądają całkiem podobnie, ale ewidentnie nie mają rozkładu normalnego, nawet nie będę tego sprawdzać. Zatem trzeba użyć nie test t-studenta, a test Wilcoxona do sprawdzenia wartości median.

```
wil_test <- wilcox.test(top_x_train, top_x_test)

# Przedstawienie wyników w tabeli
wil_test_results <- data.frame(
Test = "Test Wilcoxona",
Statistic = wil_test$statistic,
`p-value` = format(wil_test$p.value, scientific = TRUE) # Formatowanie
#bardzo małych wartości p-value
)
knitr::kable(wil_test_results, caption = "Wyniki testu Wilcoxona
do sprawdzenia dopasowania rozkładów zmiennej LGALS3 w zbiorze treningowym i testowym")
```

Table 5: Wyniki testu Wilcoxona do sprawdzenia dopasowania rozkładów zmiennej LGALS3 w zbiorze treningowym i testowym

	Test	Statistic	p.value
W	Test Wilcoxona	4078264	9.785303e-01

P-value jest bardzo duże, więc nie mamy podstaw do odrzucenia hipotezy zerowej na poziomie. Zatem przyjmujemy, że zmienna jest taka sama w obu zbiorach treningowyhm i testowym.

3. ElasticNet:

A)

Model ElasticNet powstał w celu zrównoważenia metod LASSO i regresji grzbietowej. W tym modelu szacowane są współczynniki regresji w taki sposób, aby minimalizować sumę błędu kwadratowego (klasyczne LZNK) oraz wartości zależnych tylko od wartości współczynników z regresji. Te współczynniki są włożone w normy L1 (z LASSO) i L2 (z regresji grzbietowej). Ich względny udział określa hiperparametr alfa, który przyjmuje wartości od 0 do 1. Współczynnik lambda reguluje ogólną siłę regularyzacji, czyli to, jak mocno istotne dla modelu są wartości współczynników w normach. Wygląda na to, że im większa jest wartość hiperparametru lambda, tym silniejszy jest nacisk na karanie dużych wartości współczynników regresji. Oznacza to, że model będzie bardziej "preferował" współczynniki bliższe zeru.

Gdy alfa równa się 0, model redukuje się do regresji grzbietowej. Gdy alfa wynosi 1, ElasticNet staje się modelem LASSO. Dla wartości alfa między 0 a 1 model korzysta z obu rodzajów regresji jednocześnie.

B)

```
# Przygotuj dane
X <- as.matrix(X_train) # musi być macierz, a nie ramka danych
y <- as.numeric(y_train[[1]])

# Zdefiniujmy siatkę hiperparametrów
alphas <- c(0, 0.5, 1)  # ridge, elasticnet, lasso
lambdas <- 10^seq(-4, 0, length.out = 3)

# Utwórz ramkę danych z kombinacjami
grid <- data.frame(
  alpha = alphas,
  lambda = lambdas
)

knitr::kable(grid, caption = "Ramka z hiperparametrami")
```

Table 6: Ramka z hiperparametrami

alpha	lambda
0.0	1e-04
0.5	1e-02
1.0	1e+00

```
grid_en <- expand.grid(alpha = alphas, lambda = lambdas)

# Tworzymy foldy na to i kolejne zadanie
nfolds <- 4
folds <- sample(rep(1:nfolds, length.out = nrow(X)))

# Pusta ramka na wyniki
results_en <- data.frame()

# Pętla po kombinacjach parametrów
for (i in 1:nrow(grid_en)) {
  a <- grid_en$alpha[i]
  lambda_val <- grid_en$lambda[i]
```

```r
  for (k in 1:nfolds) {
    # Podział danych na te z zbioru walidacyjnego i treningowego
    train_idx <- which(folds != k)
    val_idx <- which(folds == k)

    X_train_cv <- X[train_idx, ]
    y_train_cv <- y[train_idx]
    X_val_cv <- X[val_idx, ]
    y_val_cv <- y[val_idx]

    # Dopasowanie modelu do zbioru treningowego
    model <- glmnet(X_train_cv, y_train_cv, alpha = a, lambda = lambda_val, standardize = TRUE)

    # Predykcje
    preds_train <- predict(model, newx = X_train_cv)
    preds_val <- predict(model, newx = X_val_cv)

    # Obliczamy MSE
    mse_train <- mean((y_train_cv - preds_train)^2)
    mse_val <- mean((y_val_cv - preds_val)^2)

    # Zapis wyników
    results_en <- rbind(results_en, data.frame(
      fold = k,
      alpha = a,
      lambda = lambda_val,
      mse_train = mse_train,
      mse_val = mse_val,
      label = paste0("alpha =", a, "\n lambda=", signif(lambda_val, 2))
    ))
  }
}

#ładna tabela z wynikami
results_en_summary <- results_en %>%
  group_by(alpha, lambda) %>%
  summarise(
    Średni_błąd_treningowy = mean(mse_train),
    Średni_błąd_walidacyjny = mean(mse_val),
    .groups = "drop"
  )
knitr::kable(results_en_summary, caption = "Wyniki dla wszystkich hiperparametrów")
```

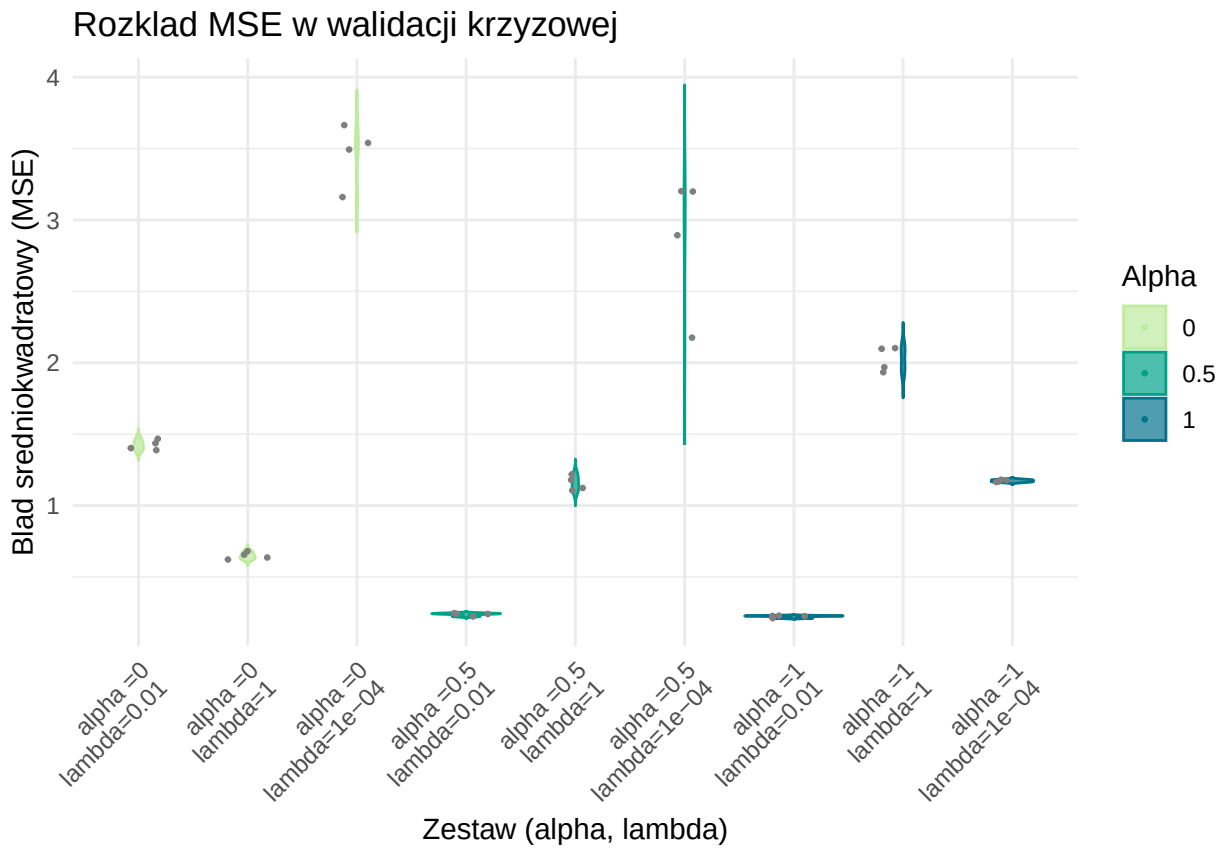
Table 7: Wyniki dla wszystkich hiperparametrów

alpha	lambda	Średni_błąd_treningowy	Średni_błąd_walidacyjny
0.0	1e-04	0.0003074	3.4640100
0.0	1e-02	0.0001457	1.4235133
0.0	1e+00	0.0962235	0.6491564
0.5	1e-04	0.0000621	2.8674892
0.5	1e-02	0.0871259	0.2386193
0.5	1e+00	1.1529065	1.1572933

alpha	lambda	Średni_błąd_treningowy	Średni_błąd_walidacyjny
1.0	1e-04	0.0002489	1.1733925
1.0	1e-02	0.1435133	0.2234708
1.0	1e+00	2.0241394	2.0257943

C)

```
# Wykres skrzypcowy
ggplot(results_en, aes(x = label, y = mse_val, fill = factor(alpha), color = factor(alpha))) +
  geom_violin(trim = FALSE, alpha = 0.7) +
  geom_jitter(width = 0.2, size = 0.5, aes(color = "black"), alpha = 1) +
  labs(
    title = "Rozkład MSE w walidacji krzyżowej",
    x = "Zestaw (alpha, lambda)",
    y = "Błąd średniokwadratowy (MSE)",
    fill = "Alpha",
    color = "Alpha"
  ) +
  # definiowanie kolorów
  scale_fill_manual(values = c("0" = "#BEEB9F", "0.5" = "#00A388", "1" = "#03738C")) +
  scale_color_manual(values = c("0" = "#BEEB9F", "0.5" = "#00A388", "1" = "#03738C"))+
  theme_minimal() +
  # nazwy są za długie, aby się zmieściły jedna obok drugiej i trzeba je ustawić pod kątem
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



D)

```
best_model_en <- results_en_summary %>%
  arrange(Średni_błąd_walidacyjny) %>%
  slice(1)

knitr::kable(best_model_en, caption = "Najlepszy model ElasticNet")
```

Table 8: Najlepszy model ElasticNet

alpha	lambda	Średni_błąd_treningowy	Średni_błąd_walidacyjny
1	0.01	0.1435133	0.2234708

Wybrano takie wartości lambda i alfa, aby średni błąd średniokwadratowy na zbiorach walidacyjnych był najmniejszy. Można zauważyć sporą różnicę miedzy błędem na zbiorach treningowych i walidacyjnych. To może wskazywać na pewne przetrenowanie modelu.

4. Lasy losowe:

A)

Wybrano następujące hiperparametry:

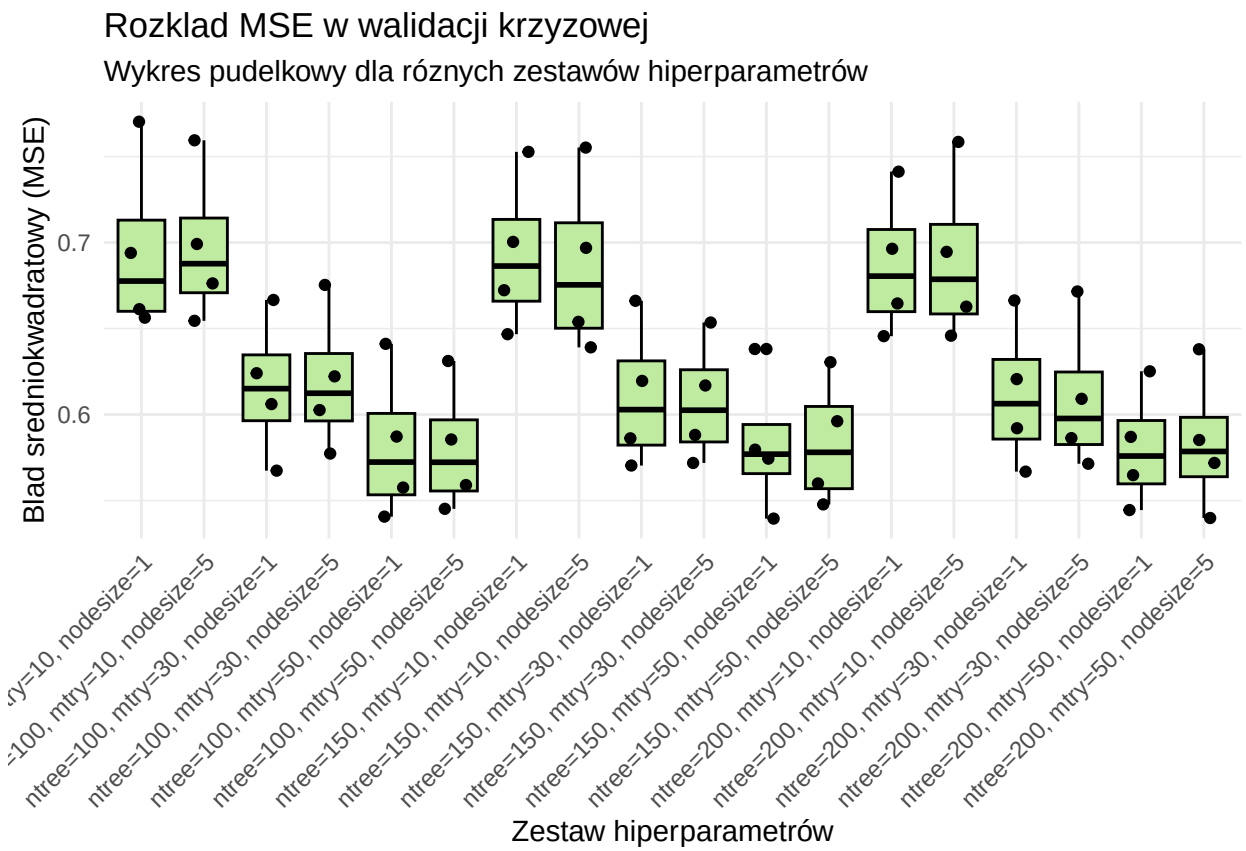
- `ntree` - liczba drzew decyzyjnych zostanie zbudowanych w lesie losowym, gdzie każde drzewo jest budowane niezależnie na losowo wybranej próbce danych
- `mtry` - liczba zmiennych wybieranych losowo przy każdym podziale
- `nodesize` - minimalna liczba obserwacji w liściu, jeśli liczba obserwacji w węźle spada poniżej tej wartości, drzewo przestaje się rozwidlać.

```
grid_rf <- expand.grid(
  ntree = c(100, 150, 200),
  mtry = c(10, 30, 50),
  nodesize = c(1, 5)
)

#Bierzemy te same foldy, co w wcześniejszym zadaniu, więc znowu używamy zmiennej fold

# tworzymy data.frame, do któego będzimy wpisywać wyniki
results_rf <- data.frame()

for (i in 1:nrow(grid_rf)) { # Dla każdego zesawu hiperparametrów (3x3x2 = 18)
  ntree_val <- grid_rf$ntree[i]
  mtry_val <- grid_rf$mtry[i]
  nodesize_val <- grid_rf$nodesize[i]

  for (k in 1:nfolds) { # dla każdego foldu
    # dzielimy dane na zbiór uczący i walidacyjny
    # do zbioru wchodzą indeksy obserwacji, nie z zbioru walidacyjnego dla danego foldu
    train_idx <- which(folds != k)
    # zbiór indeksów obserwacji do walidacji
    val_idx <- which(folds == k)

    # Bierzemy tylko wiersze zgodne z indeksami wyżej
    X_train_cv <- X_train[train_idx, ]
    y_train_cv <- y[train_idx]
    X_val_cv <- X_train[val_idx, ]
    y_val_cv <- y[val_idx]

    # ćwiczymy model lasu losowego na zdefiniowanym wyżej zbiorze treningowym
    model <- randomForest(
      x = X_train_cv,
      y = y_train_cv,
      ntree = ntree_val,
      mtry = mtry_val,
      nodesize = nodesize_val
    )

    # sprawdzamy co model przewidzi dla zbioru walidującego
    preds_val <- predict(model, newdata = X_val_cv)
    preds_train <- predict(model, newdata = X_train_cv)
    mse_val <- mean((preds_val - y_val_cv)^2)
    mse_train <- mean((preds_train - y_train_cv)^2)
```

```
      # wpisywanie wiersza z wynikami kolejnego modelu
    results_rf <- rbind(results_rf, data.frame(
      ntree = ntree_val,
      mtry = mtry_val,
      nodesize = nodesize_val,
      fold = factor(k),
      mse_val = mse_val,
      mse_train = mse_train
    ))
  }
}
```

B)

```
# dodajemy kolumnę z nazwami parametrów
df_plot <- results_rf %>%
  mutate(param_set = paste0("ntree=", ntree, ", mtry=", mtry, ", nodesize=", nodesize))

ggplot(df_plot, aes(x = param_set, y = mse_val)) +
  geom_boxplot(color = "black", fill = "#BEEB9F") +
  geom_jitter(width = 0.2) +
  labs(
    title = "Rozkład MSE w walidacji krzyżowej",
    subtitle = "Wykres pudełkowy dla różnych zestawów hiperparametrów",
    x = "Zestaw hiperparametrów",
    y = "Błąd średniokwadratowy (MSE)"
  ) +
  theme_minimal() +
  # napisy po skosie
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



C)

```
# Liczymy średnie błędy walidacyjne i treningowe dla każdego zestawu hiperparametrów
results_summary <- results_rf %>%
  group_by(ntree, mtry, nodesize) %>%
  summarise(
    'Średnia błędu na zbiorze walidacyjnym' = mean(mse_val),
    'Średnia błędu na zbiorze treningowym' = mean(mse_train),
    .groups = "drop"
  )

knitr::kable(results_summary, caption = "Wyniki dla wybranych wartości hiperparaetrów")
```

Table 9: Wyniki dla wybranych wartości hiperparaetrów

ntree	mtry	nodesize	Średnia błędu na zbiorze walidacyjnym	Średnia błędu na zbiorze treningowym
100	10	1	0.6954509	0.1414748
100	10	5	0.6973460	0.1707024
100	30	1	0.6160408	0.0930100
100	30	5	0.6193796	0.1125495
100	50	1	0.5816300	0.0837320
100	50	5	0.5802154	0.0996536
150	10	1	0.6930325	0.1401708
150	10	5	0.6862792	0.1668532
150	30	1	0.6105596	0.0902745

ntree	mtry	nodesize	Średnia błędu na zbiorze walidacyjnym	Średnia błędu na zbiorze treningowym
150	30	5	0.6076072	0.1109889
150	50	1	0.5828988	0.0820061
150	50	5	0.5835706	0.0997999
200	10	1	0.6869451	0.1384023
200	10	5	0.6904183	0.1667586
200	30	1	0.6114541	0.0901348
200	30	5	0.6096008	0.1112046
200	50	1	0.5803550	0.0819493
200	50	5	0.5837198	0.0993874

5. Podsumowanie:

```
# Najlepszy model według średniego błędu walidacyjnego dla drzew losowych
best_rf_mean <- results_summary[which.min(results_summary$'Średnia błędu na zbiorze walidacyjnym'), ]
print(best_rf_mean)
```

```
## # A tibble: 1 x 5
##   ntree  mtry nodesize Średnia błędu na zbiorze walidac~1 Średnia błędu na zbi~2
##   <dbl> <dbl>    <dbl>                              <dbl>                  <dbl>
## 1   100    50        5                              0.580                 0.0997
## # i abbreviated names: 1: `Średnia błędu na zbiorze walidacyjnym`,
## #   2: `Średnia błędu na zbiorze treningowym`
```

```
best_rf <- results_rf %>%
  filter(ntree == best_rf_mean$ntree, mtry == best_rf_mean$mtry, nodesize == best_rf_mean$nodesize)

knitr::kable(best_rf, caption = "Wyniki dla 'najlepszych' wartości hiperparaetrów")
```

Table 10: Wyniki dla 'najlepszych' wartości hiperparaetrów

ntree	mtry	nodesize	fold	mse_val	mse_train
100	50	5	1	0.5590238	0.0991625
100	50	5	2	0.5451940	0.1017956
100	50	5	3	0.5855119	0.0995100
100	50	5	4	0.6311321	0.0981461

```
  # Inicjalizacja ramki wyników
  results <- data.frame(
    Model = character(),
    Fold = integer(),
    MSE = numeric(),
    stringsAsFactors = FALSE
  )


  # 1. ElasticNet
  best_en <- results_en %>%
    filter(alpha == best_model_en$alpha, lambda == best_model_en$lambda)


  results <- rbind(results, data.frame(
    Model = "ElasticNet",
    Fold = best_en$fold,
    MSE = best_en$mse_val
  ))


  # 2. Lasy losowe
  results <- rbind(results, data.frame(
    Model = "RandomForest",
    Fold = best_rf$fold,
    MSE = best_rf$mse_val
  ))
```

```r
  # 3. Model referencyjny - średnia z folda treningowego
  for (k in 1:nfolds) {
    idx_test <- which(folds != k)
    idx_train <- which(folds == k)

    y_pred <- rep(mean(y[idx_train]), length(idx_test))
    mse <- mean((y_pred - y[idx_test])^2)

    results <- rbind(results, data.frame(
      Model = "Model referencyjny",
      Fold = k,
      MSE = mse
    ))
  }

  knitr::kable(results, caption = "Wyniki modeli dla poszczególnych foldów")
```

Table 11: Wyniki modeli dla poszczególnych foldów

Model	Fold	MSE
ElasticNet	1	0.2117594
ElasticNet	2	0.2293779
ElasticNet	3	0.2255314
ElasticNet	4	0.2272144
RandomForest	1	0.5590238
RandomForest	2	0.5451940
RandomForest	3	0.5855119
RandomForest	4	0.6311321
Model referencyjny	1	3.1261960
Model referencyjny	2	3.1212129
Model referencyjny	3	3.0508058
Model referencyjny	4	3.0714681

```r
  summary_table <- results %>%
  group_by(Model) %>%
  summarise(
    srednie_MSE = mean(MSE),
    .groups = "drop"
  ) %>%
  arrange(srednie_MSE)

knitr::kable(summary_table, caption = "Porównanie modeli na podstawie średniego MSE")
```

Table 12: Porównanie modeli na podstawie średniego MSE

Model	srednie_MSE
ElasticNet	0.2234708
RandomForest	0.5802154
Model referencyjny	3.0924207

Podsumowując, nie da się nie zauważyć, że trzeci model obciążony jest największym błędem średniokwadra-

towym. Nic dziwnego, jest najmniej złożony. Najmniejszy błąd ma tutaj model grzbietowy (ElasticNet dla $\alpha = 1$), tylko trochę lepszy od modelu ElasticNet dla $\alpha = 0.5$. Bardzo możliwe, że gdyby dobrano lepsze parametry lub więcej foldów to drzewa losowe by sobie lepiej poradziły. Z drugiej strony złożoność obliczeniowa drzew losowych była ledwo do wytrzymania dla mojego komputera. Dlatego tym bardziej bym wybrała ElasticNet jako dostatecznie dobrą metodę dopasowania.