Projekt - Wprowadzenie do ML

Anna Kaczmarek Piotr Mariusz Kozikowski Kacper Dulewicz

Spis treści

Celem projektu jest bazując na zbiorze danych dotyczącym Pokemonów budowa modeli regresyjnych przewidujących punkty życia Pokemona na podstawie pozostałych parametrów.

Zbiór `pokemons` zawiera spis pokemonów oraz ich dodatkowe paramatery i atrybuty takie jak:

- `Class` - typ/rodzaj ataku pokemona
- `HP` - punkty życia
- `Attack` - siła ataku
- `Defense` - siła obrony
- `Sp. Atk` - siła specjalnego ataku
- `Sp. Def` - siła specjalnej obrony
- `Speed` - prędkość

Poniżej wyświetlonych jest pięć pierwszych wierszy zbioru

X	class	hp	attack	defense	sp_attack	sp_defense	speed	num_class
0	Grass	45	49	49	65	65	45	1
1	Grass	60	62	63	80	80	60	1
2	Grass	80	82	83	100	100	80	1
3	Grass	80	100	123	122	120	80	1
4	Grass	80	82	83	100	100	80	1
5	Fire	39	52	43	60	50	65	2

```
library(tidyverse)
library(kableExtra)
library(dplyr)
library(car)
library(corrplot)
library(glmnet)
library(caret)
library(gridExtra)
library(ggplot2)
library(ggcorrplot)
```

```
pokemon_data <- read.csv("C:/Users/annak/OneDrive/Pulpit/MATEMATYKA STOSOWANA/II SEMESTR/WPROWADZENIE DO
kbl(head(pokemon_data)) |>
  kable_styling(bootstrap_options = c("striped", "hover"))
```

```
#str(pokemon_data)
#summary(pokemon_data)
#any(is.na(pokemon_data)) - brak braków danych
```

Podstawowe statystyki

Zobaczmy jak prezentują się podstawowe statystyki naszego zbioru, możemy zobaczyć zmienne takie jak min, max, średnią. Dzięki niej możemy lepiej zrozumieć nasze dane np.

```
df <- pokemon_data
df_numeric <- df |>
  select_if(is.numeric)
df_numeric <- df_numeric[, -c(1, ncol(df_numeric))]

st_op <- apply(df_numeric, 2, summary)
st_op <- rbind(st_op, St.dev = apply(df_numeric, 2, sd))
st_op <- as.data.frame(round(st_op, 2))

kbl(st_op, caption = "PODSTAWOWE STATYSTYKI OPISOWE>") |>
  kable_styling(bootstrap_options = c("striped", "hover"))
```

Analiza podstawowych statystyk opisowych ujawniła, że zmienna ‘hp’ (punkty życia) ma średnią wartość wynoszącą około 70.49, a jej rozkład jest nieco asymetryczny, z medianą wynoszącą 68.00, co wskazuje na lekką skośność rozkładu w prawo. Widzimy też, że wartość minimalna to 1, a maksymalna to 255, co wskazuje na sporą rozpiętość ‘hp’ w populacji. Odchylenie standardowe ‘hp’ wynosi około 26.87.

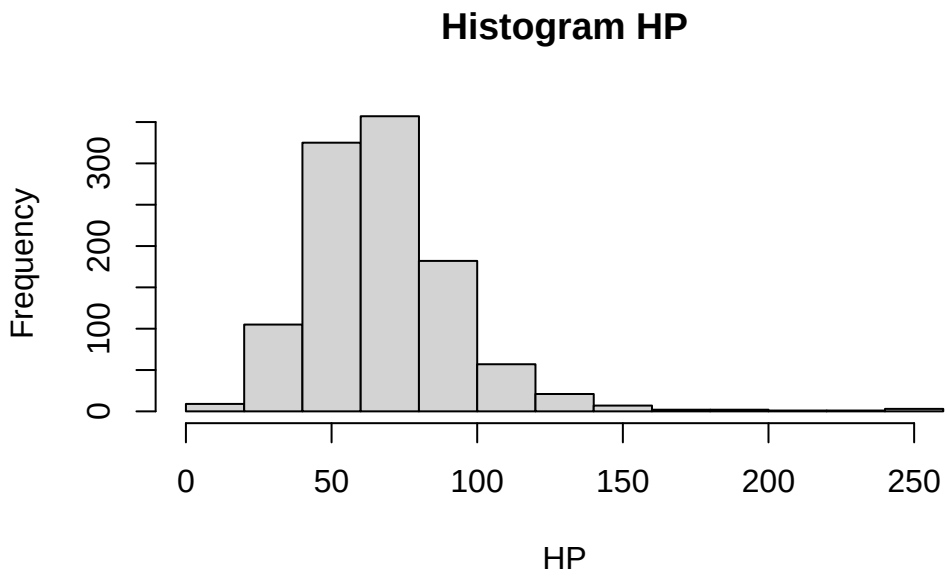
Zmienne ‘attack’, ‘defense’, ‘sp_attack’, ‘sp_defense’ i ‘speed’ mają zbliżone średnie, mieszczące się w przedziale od 68.79 do 80.94. Zmienne te również mają sporą rozpiętość, co sugeruje różnice między poszczególnymi pokemonami. Odchylenia standardowe w ich przypadku mieszczą się w przedziale od 27.93 do 32.64, co sugeruje porównywalne rozproszenie wokół średnich.

Tabela 1: PODSTAWOWE STATYSTYKI OPISOWE>

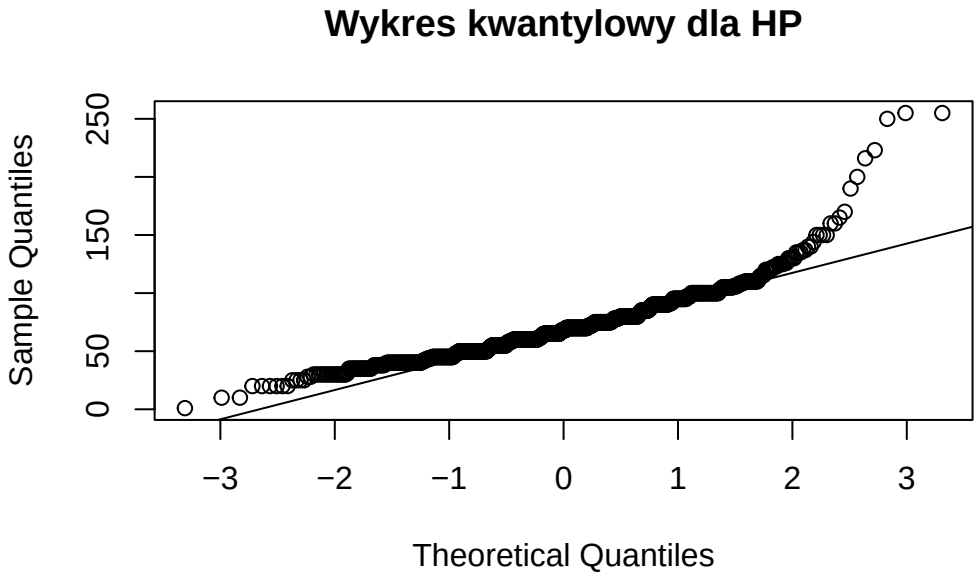
	hp	attack	defense	sp_attack	sp_defense	speed
Min.	1.00	5.00	5.00	10.00	20.00	5.00
1st Qu.	50.00	56.00	52.00	50.00	50.00	45.00
Median	68.00	80.00	70.00	65.00	70.00	65.00
Mean	70.49	80.94	74.97	73.27	72.48	68.79
3rd Qu.	84.00	100.00	90.00	95.00	90.00	90.00
Max.	255.00	190.00	250.00	194.00	250.00	200.00
St.dev	26.87	32.46	31.21	32.64	27.93	30.08

Rozkład zmiennej zależnej

```r
hist(pokemon_data$hp, main = "Histogram HP", xlab = "HP")
```



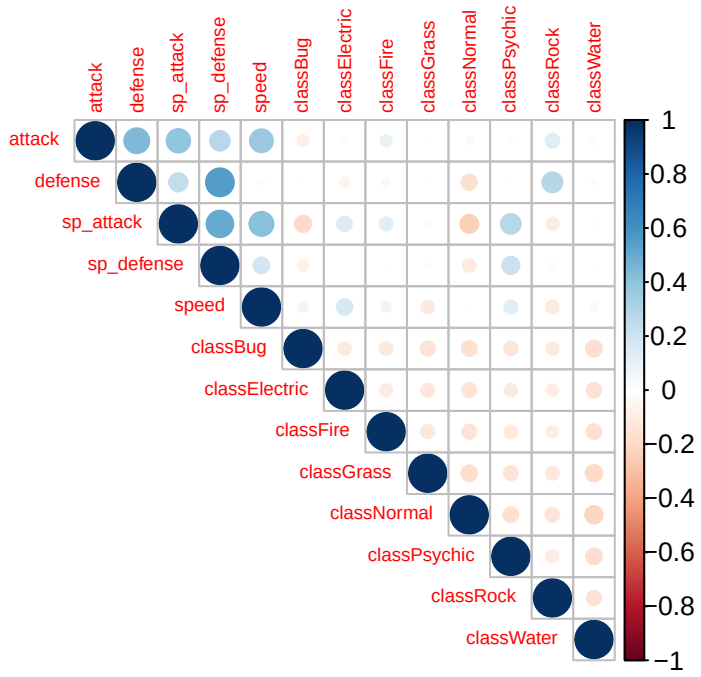

```r
qqnorm(pokemon_data$hp, main = "Wykres kwantylowy dla HP")
qqline(pokemon_data$hp)
```



Wykres kwantylowy, jak i test Shapiro-Wilka dla zmiennej ‘HP’ wskazują, że jej rozkład nie jest normalny, szczególnie w ogonach rozkładu. Lewy ogon sugeruje mniejszą rozpiętość małych wartości ‘HP’, a prawy - obecność wartości odstających i większą rozpiętość dużych wartości. Choć rozkład nie jest idealnie normalny, regresja liniowa jest na to odporna, więc na tym etapie nie będziemy transformować danych, ale będziemy monitorować wpływ wartości odstających.

Przygotowanie danych do modelowania

```r
class_counts <- table(pokemon_data$class)
threshold <- 50 # klasy z mniej niż 50 pokemonami zostaną pominięte i nie będzie dla nich utworzonej zmi
selected_classes <- names(class_counts[class_counts >= threshold])
pokemon_data_filtered <- pokemon_data |>
  filter(class %in% selected_classes)

# Utworzenie zmiennych "dummy" (zero-jedynkowych) z kolumny "class"
dummy_variables <- model.matrix(~ class - 1, data = pokemon_data_filtered)

dependent_variable <- pokemon_data_filtered$hp #Wyodrębnienie zmiennej zależnej

#  Wyodrębnienie zmiennych niezależnych
independent_variables_numeric <- pokemon_data_filtered |>
  select_if(is.numeric)
independent_variables_numeric <- independent_variables_numeric[, -c(1, ncol(independent_variables_numeri
independent_variables_numeric <- independent_variables_numeric[, -which(names(independent_variables_nume

independent_variables <- cbind(independent_variables_numeric, dummy_variables)

# Korelacja
cor_matrix <- cor(independent_variables)

# Macierz korelacji - heatmapa
corrplot(cor_matrix, method = "circle", type = "upper", tl.cex = 0.6)
```



Heatmapa korelacji przedstawia wizualizację zależności między zmiennymi niezależnymi, zarówno numerycznymi, jak i zmiennymi dummy. W zmiennych numerycznych na heatmapie można zaobserwować kilka istotnych trendów:

- Silne, dodatnie korelacje między zmiennymi attack, sp_attack, defense i sp_defense są wyraźnie widoczne. To sugeruje, że Pokemony z wyższymi wartościami jednej z tych statystyk, mają tendencję do posiadania wyższych wartości również pozostałych
- Zmienna speed jest słabo skorelowana z innymi zmiennymi
- Korelacje między defense i attack oraz sp_defense i sp_attack są trochę mniejsze od tych między atakiem i sp_attack oraz obroną i sp_defense, co może sugerować, że obrona fizyczna i specjalna są nieco bardziej niezależne.

Jeżeli chodzi o zmienne dummy widoczny jest brak silnych korelacji między nimi, co jest oczekiwane, ponieważ dany pokemon należy tylko do jednej klasy, a zmienne te tworzą zmienne wzajemnie wykluczające się.

Korelacje między zmiennymi numerycznymi a zmiennymi dummy są na ogół słabe i nie wykazują istotnych trendów.

Podział danych na zbiór treningowy i testowy

W celu oceny zdolności generalizacji modeli, dane zostały podzielone na dwa rozłączne zbiory: zbiór treningowy (80% danych) i zbiór testowy (20% danych). Podziału dokonano z użyciem funkcji createDataPartition z pakietu caret, która zapewnia losowy, ale powtarzalny podział danych.

```
set.seed(2137)  # setseed dla powtarzalności
train_index <- createDataPartition(dependent_variable, p = 0.8, list = FALSE)
```

```
train_data <- as.matrix(independent_variables[train_index, ])
train_labels <- dependent_variable[train_index]
test_data <- as.matrix(independent_variables[-train_index, ])
test_labels <- dependent_variable[-train_index]
```

Regresja Wieloraka

W celu uniknięcia problemu współliniowości, wynikającego z zależności liniowej między zmiennymi dummy i wyrazem wolnym w modelu regresji, zdecydowano się na usunięcie jednej z kolumn zmiennych dummy. Pomimo iż nie tworzono zmiennych dummy dla wszystkich klas, sama konstrukcja zmiennych zero-jedynkowych powoduje, że suma ich dla każdej obserwacji wynosi 1. Usuwając jedną kolumnę, zlikwidowano tę zależność, umożliwiając prawidłowe oszacowanie współczynników modelu i obliczenie wskaźnika VIF.

Usunięta kategoria ‘class’ staje się kategorią referencyjną dla pozostałych zmiennych dummy.

```
independent_variables_no_alias <- as.data.frame(independent_variables[, -which(colnames(independent_vari
```

Model

```
model <- lm(dependent_variable ~ ., data = independent_variables_no_alias)
summary(model)
```

```
Call:
lm(formula = dependent_variable ~ ., data = independent_variables_no_alias)

Residuals:
    Min      1Q  Median      3Q     Max 
-68.940 -11.657  -2.828   8.292 175.250 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)   27.13453    4.30449   6.304 5.19e-10 ***
attack         0.25879    0.03246   7.973 6.48e-15 ***
defense       -0.08702    0.03915  -2.223  0.02657 *  
sp_attack      0.13864    0.03436   4.035 6.08e-05 ***
sp_defense     0.29534    0.04061   7.272 9.65e-13 ***
speed         -0.09034    0.03174  -2.846  0.00455 ** 
classBug      -3.52457    3.71679  -0.948  0.34332    
classElectric -4.25029    4.16054  -1.022  0.30734    
classFire      0.01726    4.00881   0.004  0.99656    
classGrass     1.42086    3.66101   0.388  0.69806    
classNormal   15.41048    3.59323   4.289 2.05e-05 ***
```

```
classPsychic   0.12421    4.01773   0.031  0.97535
classWater     5.31657    3.45269   1.540  0.12406
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.26 on 688 degrees of freedom
Multiple R-squared:  0.3256,    Adjusted R-squared:  0.3138
F-statistic: 27.68 on 12 and 688 DF,  p-value: < 2.2e-16
```

Model regresji wielorakiej wskazuje, że statystyki pokemonów takie jak ‘attack’, ‘sp_attack’ i ‘sp_defense’ mają istotny, pozytywny wpływ na ‘HP’, natomiast ‘defense’ i ‘speed’ mają istotny, negatywny wpływ. Przynależność do klasy ‘Normal’ również okazała się istotnym predyktorem, wskazując na wyższe ‘HP’ w porównaniu z kategorią referencyjną. Pozostałe klasy Pokemonów, nie wykazały istotnego wpływu po uwzględnieniu statystyk podstawowych.

Model wyjaśnia około 31% wariancji ‘HP’, a standardowy błąd resztowy wynosi 21.26, co sugeruje, że model jest dobry, ale nie idealny. Analiza modelu potwierdziła statystyczną istotność zależności, jednak widać, że sam model nie wyjaśnia całej zmienności w wartości ‘HP’.

Diagnostyka modelu

VIF

```
# VIF
vif_values <- vif(model)
vif_df <- data.frame(
  Variable = names(vif_values),
  VIF = vif_values
)
kbl(vif_df, caption = "Wskaźniki VIF (Variance Inflation Factor)>", row.names = F)  |>
  column_spec(1, bold = TRUE) |>
    kable_styling(
        bootstrap_options = c("striped", "hover", "condensed", "responsive"),
        full_width = F,
        position = "center"
        )
```

W modelu regresji wielorakiej nie występuje problem silnej współliniowości. Wszystkie wartości VIF dla zmiennych niezależnych są relatywnie niskie, nie przekraczając wartości 3. Zmienne numeryczne z zakresu statystyk pokemonów uzyskały VIF między 1.4 i 2.1. Wskaźniki VIF dla zmiennych dummy oscylują między 2.1 a 2.9.

Wartości VIF poniżej 5, a tym bardziej 10, sugerują brak poważnych problemów współliniowości, co oznacza, że model regresji liniowej może być z powodzeniem użyty w tym przypadku bez konieczności wykluczania zmiennych.

Tabela 2: Wskaźniki VIF (Variance Inflation Factor)>

Variable	VIF
attack	1.706895
defense	2.058709
sp_attack	1.979113
sp_defense	1.940881
speed	1.465282
classBug	2.237188
classElectric	2.196648
classFire	2.155237
classGrass	2.436124
classNormal	2.785281
classPsychic	2.531888
classWater	2.892033

Tabela 3: Metryki modelu regresji wielorakiej>

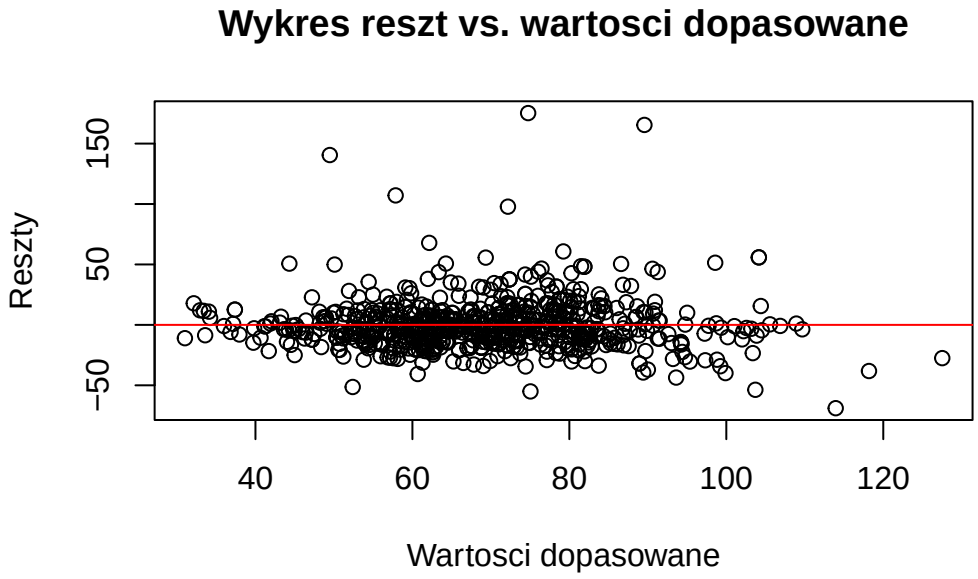
Metric	Value
MSE	394.1833977
RMSE	19.8540524
MAE	14.2971345
R-squared	0.3683939

Metryki

```
predictions_lm <- predict(model, newdata = as.data.frame(test_data))

mse_lm <- mean((predictions_lm - test_labels)^2)
rmse_lm <- sqrt(mse_lm)
mae_lm <- mean(abs(predictions_lm - test_labels))
r2_lm <- 1 - (sum((predictions_lm - test_labels)^2) / sum((test_labels - mean(test_labels))^2))

metrics_lm <- data.frame(
    Metric = c("MSE", "RMSE", "MAE", "R-squared"),
    Value = c(mse_lm, rmse_lm, mae_lm, r2_lm)
)
kbl(metrics_lm, caption = "Metryki modelu regresji wielorakiej>") |>
      column_spec(1, bold = TRUE) |>
      kable_styling(
        bootstrap_options = c("striped", "hover", "condensed", "responsive"),
        full_width = F,
        position = "center"
        )
```

Analiza reszt (wykresy reszt vs. wartości dopasowane)

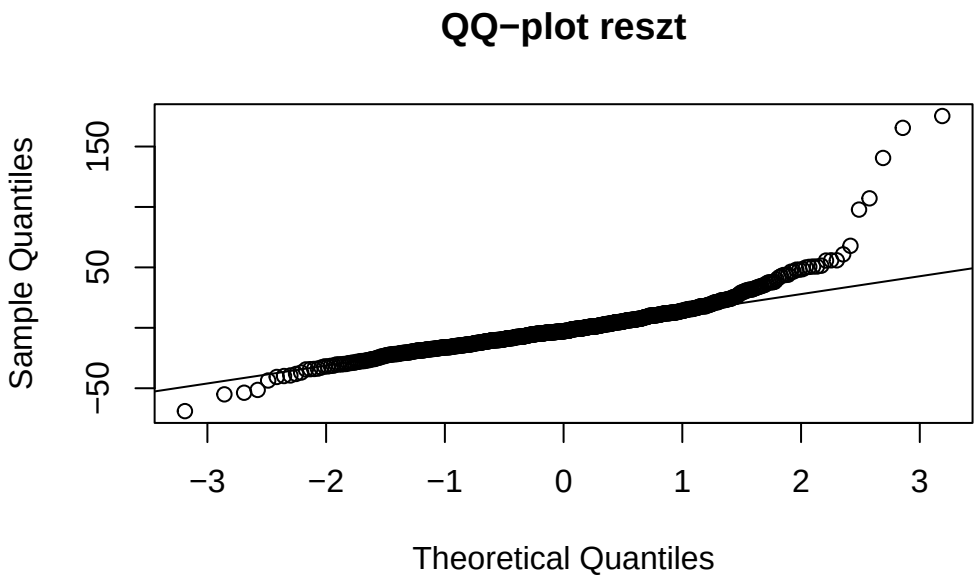
```
# Analiza reszt (wykresy reszt vs. wartości dopasowane)
plot(model$fitted.values, model$residuals,
     xlab = "Wartości dopasowane", ylab = "Reszty",
     main = "Wykres reszt vs. wartości dopasowane")
abline(h = 0, col = "red")
```



Analiza wykresu reszt vs. wartości dopasowane pozwala ocenić, czy model spełnia założenie o homoskedastyczności (stałej wariancji reszt). Na wykresie widoczny jest losowy układ punktów wokół poziomej linii (reszta = 0), co jest generalnie pożądane. Nie obserwuje się wyraźnego wzorca, takiego jak lejek, czy zakrzywienie, który sugerowałby poważne naruszenie założeń modelu. Jednakże, można zauważyć, że rozrzut reszt nie jest idealnie jednorodny w całym zakresie wartości dopasowanych. Dla wyższych wartości dopasowanych rozrzut reszt wydaje się nieco większy, co może wskazywać na występowanie niewielkiej heteroskedastyczności, choć nie jest ona bardzo wyraźna.

QQ-plot

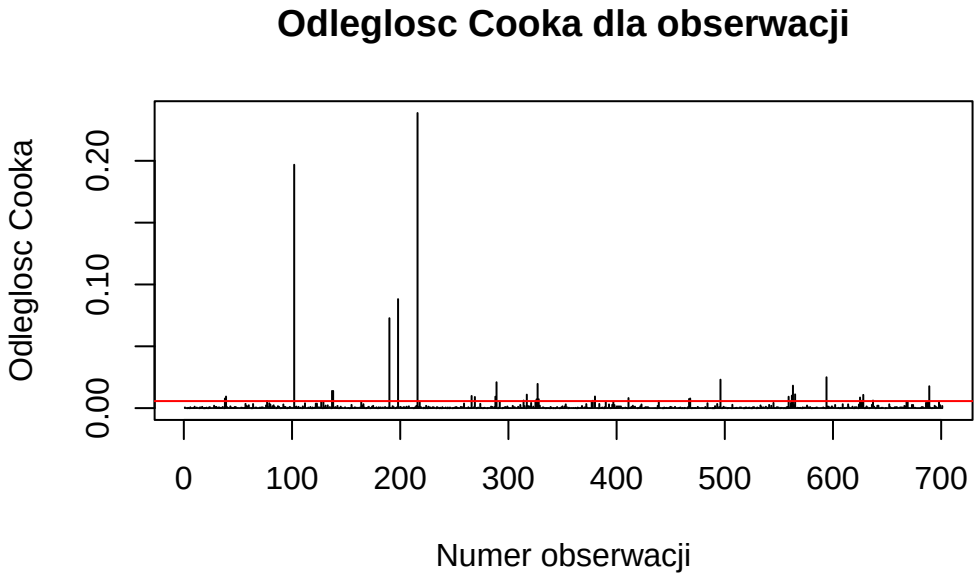
```
# QQ-plot
qqnorm(model$residuals, main = "QQ-plot reszt")
qqline(model$residuals)
```



Wykres kwantylowy reszt wskazuje na pewne odchylenia od normalności, szczególnie na końcach rozkładu, co jest zgodne z wcześniej zaobserwowanym rozkładem zmiennej zależnej ‘HP’, jednak odchylenia nie są duże i nie dyskwalifikują modelu.

Obserwacje odstające - odległość Cooka

```
cooks_dist <- cooks.distance(model)
plot(cooks_dist, type = "h",
     xlab = "Numer obserwacji", ylab = "Odległość Cooka",
     main = "Odległość Cooka dla obserwacji")
abline(h = 4/length(cooks_dist), col = "red") # Wartość progowa
```




```
# Zidentyfikowanie obserwacji odstających
threshold <- 4/length(cooks_dist) # domyślny próg
high_outlier_threshold <- threshold * 2 # wyższy próg
outliers <- which(cooks_dist > high_outlier_threshold)

if(length(outliers)>0){
outliers_df <- data.frame(Outlier_Number = outliers)
kbl(outliers_df, caption = paste("Numery obserwacji odstających (odległość Cooka > ", round(high_outlier
  column_spec(1, bold = TRUE) |>
      kable_styling(
        bootstrap_options = c("striped", "hover", "condensed", "responsive"),
        full_width = F,
        position = "center"
        )
  } else {
    print(paste("Nie znaleziono obserwacji odstających powyżej progu: ", round(high_outlier_threshold, 3
  }
```

Wykres odległości Cooka wskazuje, że kilka obserwacji wykazuje znacznie większe wartości odległości Cooka niż pozostałe. Obserwacje z odległością Cooka przekraczającą **dwukrotność progu (który został wyliczony jako 4 podzielone przez liczbę obserwacji)**, uznano za wartości odstające i zostały one przedstawione w poniższej tabeli. Zazwyczaj obserwacje o odległości Cooka większej niż 4/n (gdzie n to liczba obserwacji) są traktowane jako odstające i to też było naszym progiem, ale tutaj pokazujemy te, które mocno od niego odbiegają. Ich obecność należy wziąć pod uwagę przy interpretacji wyników regresji i ewentualnie zastanowić się nad ich usunięciem lub modyfikacją.

Pomimo drobnych niedoskonałości, model regresji wielorakiej wydaje się być ogólnie odpowiedni dla analizowanych danych, a dalsza analiza będzie kontynuowana poprzez sprawdzenie, czy inne metody regresji dadzą lepsze rezultaty.

Tabela 4: Numery obserwacji odstających (odległość Cooka > 0.011)>

	Outlier_Number
102	102
137	137
138	138
190	190
198	198
216	216
289	289
327	327
496	496
563	563
594	594
689	689

Regresja Lasso

Siatka wartości parametru λ

```
lambda_seq <- 10^seq(2, -2, by = -.1)
```

Budowa modeli i wybór optymalnego λ za pomocą walidacji krzyżowej

```
cv_model <- cv.glmnet(train_data, train_labels, alpha = 1, lambda = lambda_seq, nfolds = 10)
best_lambda <- cv_model$lambda.min # minimalizujemy błąd walidacyjny
print(paste("Najlepsza wartość lambda:", best_lambda))
```

```
[1] "Najlepsza wartość lambda: 0.158489319246111"
```

```
# Model dla optymalnego lambda
lasso_model_best <- glmnet(train_data, train_labels, alpha = 1, lambda = best_lambda)
# Model dla 10*lambda optymalne
lambda_larger = best_lambda*10
lasso_model_larger <- glmnet(train_data, train_labels, alpha = 1, lambda = lambda_larger)
```

Analiza modeli

Funkcja do oceny modelu

```
evaluate_model <- function(model, lambda_value, test_data, test_labels, model_name) {
    predictions <- predict(model, s = lambda_value, newx = test_data) #predykcje dla podanego modelu na
    mse <- mean((predictions - test_labels)^2)
    rmse <- sqrt(mse)
    mae <- mean(abs(predictions - test_labels))
    r2 <- 1 - (sum((predictions - test_labels)^2) / sum((test_labels - mean(test_labels))^2))
```

```
    metrics <- data.frame(
        Metric = c("MSE", "RMSE", "MAE", "R-squared"),
        Value = c(mse, rmse, mae, r2)
    )

    coefs <- as.matrix(coef(model)) # współczynniki modelu
    coefs_df <- data.frame(
        Variable = rownames(coefs)[coefs != 0],
        Coefficient = as.numeric(coefs[coefs != 0])
    )

    output_list <- list()

    output_list$metrics_table <- kbl(metrics, caption = paste("Metryki modelu", model_name, ">")) |> # w
        column_spec(1, bold = TRUE) |>
        kable_styling(
          bootstrap_options = c("striped", "hover", "condensed", "responsive"),
          full_width = F,
          position = "center"
          )

    output_list$coefs_table <- kbl(coefs_df, caption = paste("Współczynniki modelu", model_name, ">")) |
        column_spec(1, bold = TRUE) |>
        kable_styling(
          bootstrap_options = c("striped", "hover", "condensed", "responsive"),
          full_width = F,
          position = "center"
          )

    # wykres pred vs real
    prediction_plot <- ggplot(data.frame(test_labels = test_labels, predictions = as.numeric(predictions
      geom_point() +
      geom_abline(intercept = 0, slope = 1, color="red")+
      labs(title=paste("Predykcja vs. rzeczywiste dla ", model_name, " na zbiorze testowym"), x="Rzeczyw
    output_list$prediction_plot <- prediction_plot

    return(output_list)
}
```

Ocena modeli i wyświetlenie wyników

```
output_best$metrics_table
```

```
output_best$coefs_table
```

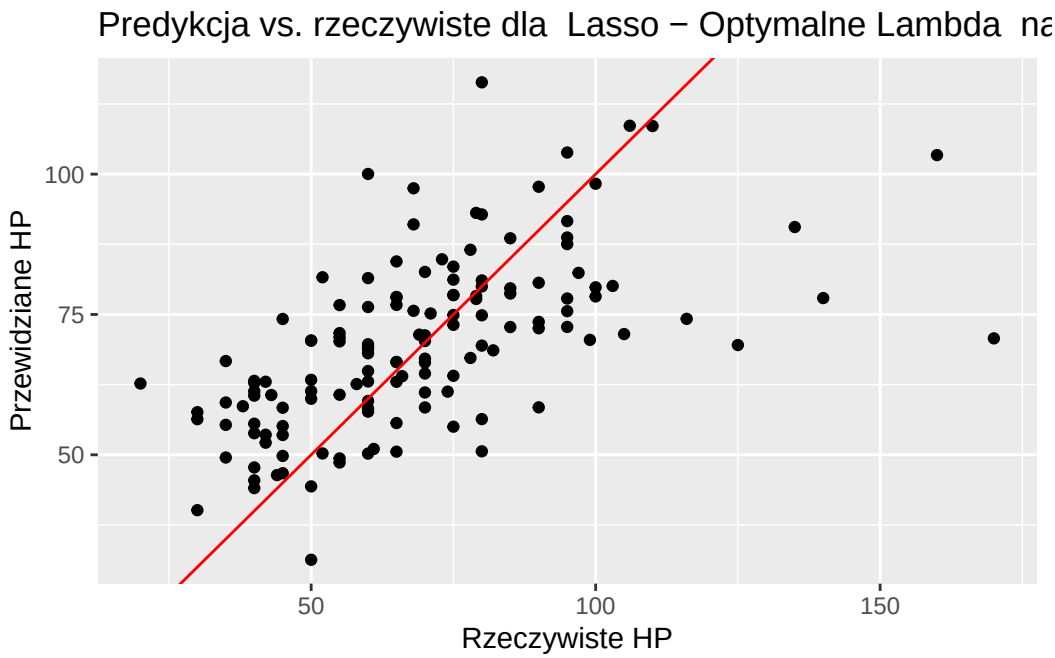
```
output_best$prediction_plot
```

Tabela 5: Metryki modelu Lasso - Optymalne Lambda >

Metric	Value
MSE	403.5353867
RMSE	20.0881902
MAE	14.5055681
R-squared	0.3534091

Tabela 6: Współczynniki modelu Lasso - Optymalne Lambda >

Variable	Coefficient
(Intercept)	28.5849667
attack	0.2415415
defense	-0.0870677
sp_attack	0.1116074
sp_defense	0.3130553
speed	-0.0665870
classBug	-5.9408478
classElectric	-4.1080563
classFire	-0.3421794
classGrass	0.8160057
classNormal	14.2027000
classRock	-0.2893222
classWater	4.1990463




```
output_larger$metrics_table
```

Tabela 7: Metryki modelu Lasso - Większe Lambda >

Metric	Value
MSE	420.3970811
RMSE	20.5035870
MAE	14.3387092
R-squared	0.3263913

Tabela 8: Współczynniki modelu Lasso - Większe Lambda >

Variable	Coefficient
(Intercept)	33.0401350
attack	0.1697500
sp_attack	0.0744704
sp_defense	0.2302413
classBug	-4.2280470
classElectric	-0.7391962
classNormal	9.2753334
classWater	0.0972844

```
output_larger$coefs_table
```

```
output_larger$prediction_plot
```

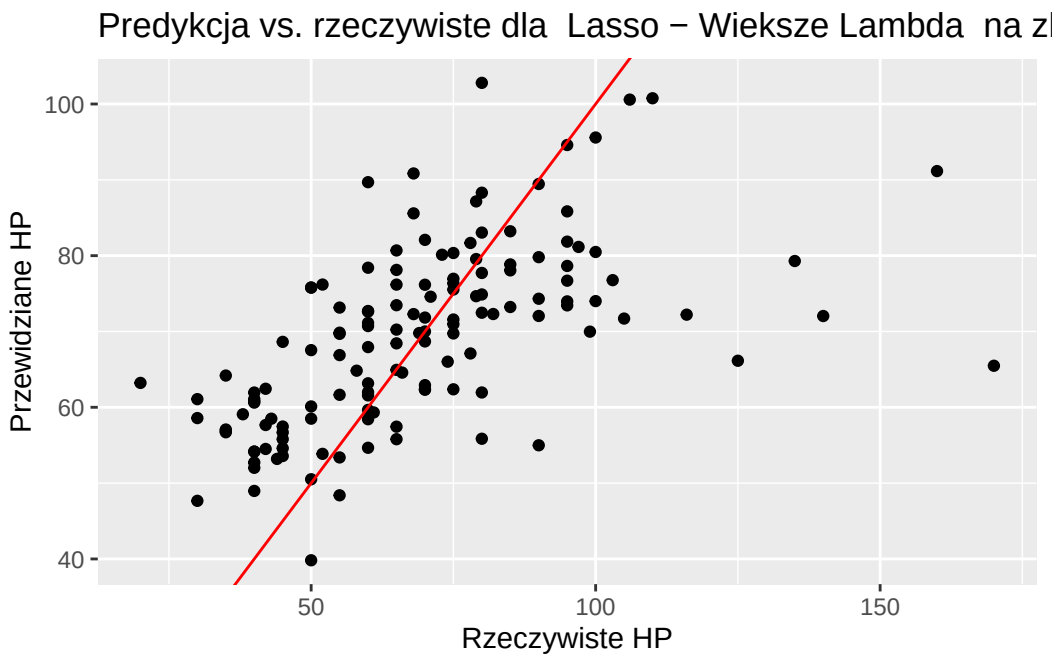


Tabela 9: Porównanie metryk modeli>

Model	MSE	RMSE	MAE	R_squared
Regresja Wieloraka	394.1834	19.85405	14.29713	0.3683939
Lasso - Optymalne Lambda	403.5354	20.08819	14.50557	0.3534091
Lasso - Większe Lambda	420.3971	20.50359	14.33871	0.3263913

Porównanie modeli

```
comparison_table <- data.frame(
    Model = c("Regresja Wieloraka", "Lasso - Optymalne Lambda", "Lasso - Większe Lambda"),
    MSE = c(mse_lm,
            mean((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2),
            mean((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2)),
    RMSE = c(rmse_lm,
            sqrt(mean((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2)),
            sqrt(mean((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2
    MAE = c(mae_lm,
            mean(abs(predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)),
            mean(abs(predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels))),
    R_squared = c(r2_lm,
                1 - (sum((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2)
             1 - (sum((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2
)

kbl(comparison_table, caption = "Porównanie metryk modeli>") |>
        kable_styling(bootstrap_options = c("striped", "hover"))
```

Tabela porównawcza prezentuje wyniki metryk (MSE, RMSE, MAE, R-kwadrat) dla trzech różnych modeli regresji, obliczonych na zbiorze testowym. Widać, że model regresji wielorakiej osiągnął najmniejszą wartość MSE (394.18), RMSE (19.85) i MAE (14.29), co oznacza, że generalnie lepiej pasuje do danych. Ma również najwyższą wartość R-kwadrat (0.368). Warto zauważyć, że model Lasso z optymalną wartością lambda uzyskał zbliżone metryki do modelu wielorakiego, z nieco gorszym wynikiem. Z kolei model Lasso z większym parametrem lambda ma najgorsze wyniki ze wszystkich 3 modeli, co wynika z tego, że ma tendencję do zaniżania parametrów.

Aby rzetelnie ocenić wydajność każdego modelu, wszystkie wyniki metryk (MSE, RMSE, MAE, R-kwadrat) zostały uzyskane na podstawie predykcji dokonanych na zbiorze testowym, który nie brał udziału w procesie trenowania modeli.

Regresja grzbietowa

Siatka wartości parametrów α i λ

```
alpha_seq <- seq(0.1, 0.9, by = 0.2) # kilka wartości alpha miedzy 0 i 1
lambda_seq <- 10^seq(2, -2, by = -.1)
```

Budowa modeli regresji grzbietowej

```
cv_model_ridge <- cv.glmnet(train_data, train_labels, alpha = 0, lambda = lambda_seq, nfolds = 10) # alp
best_lambda_ridge <- cv_model_ridge$lambda.min
print(paste("Najlepsza wartość lambda dla regresji grzbietowej:", best_lambda_ridge))
```

```
[1] "Najlepsza wartość lambda dla regresji grzbietowej: 3.16227766016838"
```

Ocena modeli dla dwóch różnych wartości parametru lambda

```
# Model dla optymalnego lambda
ridge_model_best <- glmnet(train_data, train_labels, alpha = 0, lambda = best_lambda_ridge)
# Model dla lambda większego od optymalnego
lambda_larger_ridge <- best_lambda_ridge * 10
ridge_model_larger <- glmnet(train_data, train_labels, alpha = 0, lambda = lambda_larger_ridge)

output_best_ridge <- evaluate_model(ridge_model_best, best_lambda_ridge, test_data, test_labels, "Regres
output_larger_ridge <- evaluate_model(ridge_model_larger, lambda_larger_ridge, test_data, test_labels, '

output_best_ridge$metrics_table
```

```
output_best_ridge$coefs_table
```

```
output_best_ridge$prediction_plot
```

Tabela 10: Metryki modelu Regresja Grzbietowa - Optymalne Lambda >

Metric	Value
MSE	403.1404782
RMSE	20.0783585
MAE	14.4297482
R-squared	0.3540418

Tabela 11: Współczynniki modelu Regresja Grzbietowa - Optymalne Lambda >

Variable	Coefficient
(Intercept)	32.1557600
attack	0.2050110
defense	-0.0441521
sp_attack	0.1173884
sp_defense	0.2615427
speed	-0.0457865
classBug	-7.8456407
classElectric	-6.2079589
classFire	-2.2965217
classGrass	-0.6564828
classNormal	11.4455453
classPsychic	-1.5339108
classRock	-2.7319101
classWater	2.1487698

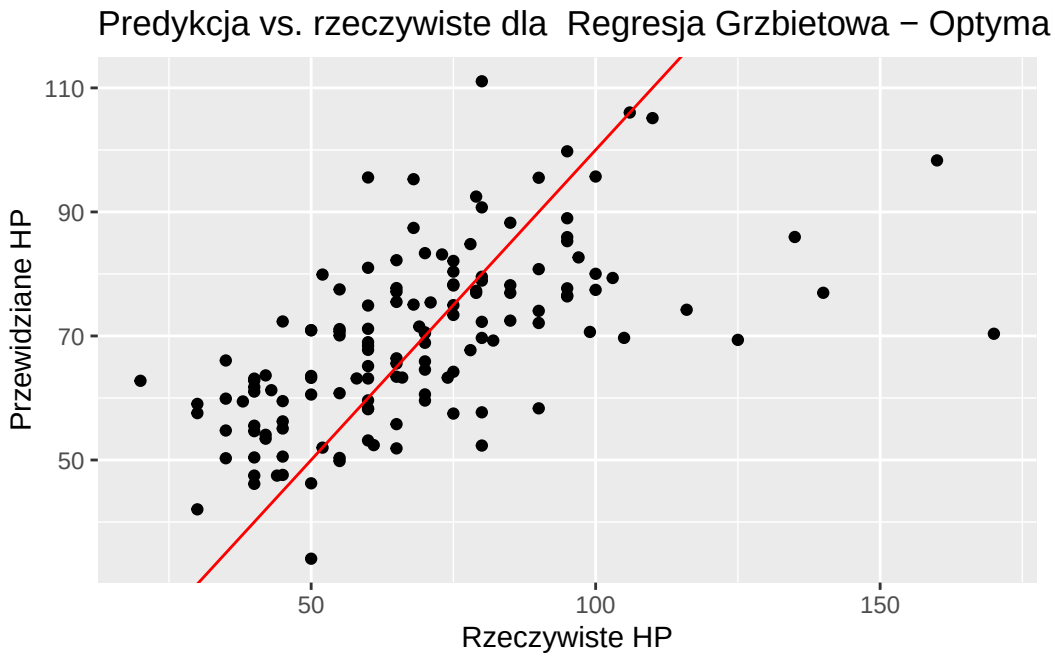


Tabela 12: Metryki modelu Regresja Grzbietowa - Większe Lambda >

Metric	Value
MSE	450.7848588
RMSE	21.2316947
MAE	15.0475150
R-squared	0.2777005

Tabela 13: Współczynniki modelu Regresja Grzbietowa - Większe Lambda >

Variable	Coefficient
(Intercept)	43.4202063
attack	0.0991670
defense	0.0322138
sp_attack	0.0781668
sp_defense	0.1246809
speed	0.0131263
classBug	-5.0670310
classElectric	-3.3089380
classFire	-0.5171267
classGrass	-0.2153849
classNormal	5.7175591
classPsychic	0.1075246
classRock	-1.4327904
classWater	1.0801202

```
output_larger_ridge$metrics_table
```

```
output_larger_ridge$coefs_table
```

```
output_larger_ridge$prediction_plot
```

Porównanie metryk modeli w tabeli

```
comparison_table_ridge <- data.frame(
    Model = c("Regresja Wieloraka", "Lasso - Optymalne Lambda", "Lasso - Większe Lambda", "Regresja Grzb
    MSE = c(mse_lm,
            mean((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2),
            mean((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2),
             mean((predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels)^2),
            mean((predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_labels)^
    RMSE = c(rmse_lm,
            sqrt(mean((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2)),
            sqrt(mean((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2
             sqrt(mean((predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels
            sqrt(mean((predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_lab
    MAE = c(mae_lm,
            mean(abs(predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)),
            mean(abs(predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)),
             mean(abs(predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels))
            mean(abs(predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_label
    R_squared = c(r2_lm,
                1 - (sum((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2
              1 - (sum((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2
              1 - (sum((predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels)
              1 - (sum((predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_lab
)
```

Tabela 14: Porównanie metryk modeli na zbiorze testowym>

Model	MSE	RMSE	MAE	R_squared
Regresja Wieloraka	394.1834	19.85405	14.29713	0.3683939
Lasso - Optymalne Lambda	403.5354	20.08819	14.50557	0.3534091
Lasso - Większe Lambda	420.3971	20.50359	14.33871	0.3263913
Regresja Grzbietowa - Optymalne Lambda	403.1405	20.07836	14.42975	0.3540418
Regresja Grzbietowa - Większe Lambda	450.7849	21.23169	15.04752	0.2777005

```
kbl(comparison_table_ridge, caption = "Porównanie metryk modeli na zbiorze testowym>") |>
        kable_styling(bootstrap_options = c("striped", "hover"))
```

Tabela porównawcza prezentuje metryki (MSE, RMSE, MAE, R-kwadrat) modeli regresji wielorakiej, Lasso (z optymalnym i większym lambda) oraz regresji grzbietowej (z optymalnym i większym lambda), obliczone na zbiorze testowym.

Model regresji wielorakiej nadal osiąga najlepsze wyniki w kategorii MSE, RMSE i MAE (MSE: 394.18, RMSE: 19.85, MAE: 14.29) i R-kwadrat (0.368). Model Lasso z optymalnym lambda jest nieco gorszy (MSE: 403.53, RMSE: 20.09, MAE: 14.51, R-squared 0.353) ale wciąż zbliżony do regresji wielorakiej.

Lasso z większym lambda, ma najgorsze wyniki (MSE: 420.4, RMSE: 20.5, MAE: 14.34 R-squared 0.326).

Modele regresji grzbietowej (optymalny i większe lambda) wykazują gorsze dopasowanie niż model regresji wielorakiej, a regresja grzbietowa z większym lambda osiąga najgorszy wynik R-squared ze wszystkich modeli (0.278).

Zauważalne jest, że modele z większym parametrem lambda (zarówno dla Lasso jak i grzbietowej) osiągają gorsze rezultaty, ponieważ modele z większym parametrem lambda redukują współczynniki do zera, a to pogarsza ich zdolność do dopasowania się do danych.

Elastic Net

Budowa modeli Elastic Net

```
elastic_net_models <- list()

for (alpha in alpha_seq){
  cv_model_elastic <- cv.glmnet(train_data, train_labels, alpha = alpha, lambda = lambda_seq, nfolds = 1
  best_lambda_elastic <- cv_model_elastic$lambda.min
  elastic_net_model <- glmnet(train_data, train_labels, alpha = alpha, lambda = best_lambda_elastic)
  model_name <- paste0("Elastic Net - alpha=", round(alpha, 2))
   elastic_net_models[[model_name]] <- list(model= elastic_net_model, lambda=best_lambda_elastic, alpha
    print(paste("Najlepsza wartość lambda dla Elastic Net (alpha = ", round(alpha, 2), ") :", best_lambd
}
```

```
[1] "Najlepsza wartość lambda dla Elastic Net (alpha =  0.1 ) : 0.794328234724281"
[1] "Najlepsza wartość lambda dla Elastic Net (alpha =  0.3 ) : 0.316227766016838"
[1] "Najlepsza wartość lambda dla Elastic Net (alpha =  0.5 ) : 0.251188643150958"
[1] "Najlepsza wartość lambda dla Elastic Net (alpha =  0.7 ) : 0.251188643150958"
[1] "Najlepsza wartość lambda dla Elastic Net (alpha =  0.9 ) : 0.199526231496888"
```

Ocena modeli

```
output_elastic_net <- list()
for(model_name in names(elastic_net_models)){
   output_elastic_net[[model_name]] <- evaluate_model(elastic_net_models[[model_name]]$model, elastic_ne

output_elastic_net[[model_name]]$metrics_table
output_elastic_net[[model_name]]$coefs_table
output_elastic_net[[model_name]]$prediction_plot
}
```

Porównanie modeli

```
comparison_table_elastic <- data.frame(
  Model = c("Regresja Wieloraka", "Lasso - Optymalne Lambda", "Lasso - Większe Lambda",
            "Regresja Grzbietowa - Optymalne Lambda", "Regresja Grzbietowa - Większe Lambda"),
    MSE = c(mse_lm,
            mean((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2),
            mean((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2),
             mean((predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels)^2),
            mean((predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_labels)^
    RMSE = c(rmse_lm,
            sqrt(mean((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2)),
            sqrt(mean((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2
             sqrt(mean((predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels
            sqrt(mean((predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_lab
    MAE = c(mae_lm,
            mean(abs(predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)),
            mean(abs(predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)),
           mean(abs(predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels)),
            mean(abs(predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_label
   R_squared = c(r2_lm,
                1 - (sum((predict(lasso_model_best, s = best_lambda, newx = test_data) - test_labels)^2)
             1 - (sum((predict(lasso_model_larger, s = lambda_larger, newx = test_data) - test_labels)^2
             1 - (sum((predict(ridge_model_best, s = best_lambda_ridge, newx = test_data) - test_labels)
             1 - (sum((predict(ridge_model_larger, s = lambda_larger_ridge, newx = test_data) - test_lab
)
for(model_name in names(elastic_net_models)){
      predictions_elastic <- predict(elastic_net_models[[model_name]]$model, s =elastic_net_models[[mode
       mse_elastic <- mean((predictions_elastic - test_labels)^2)
       rmse_elastic <- sqrt(mse_elastic)
       mae_elastic <- mean(abs(predictions_elastic - test_labels))
      r2_elastic <- 1 - (sum((predictions_elastic - test_labels)^2) / sum((test_labels - mean(test_label
  comparison_table_elastic <- rbind(comparison_table_elastic,  data.frame(
        Model = model_name,
        MSE = mse_elastic,
       RMSE = rmse_elastic,
        MAE = mae_elastic,
      R_squared =  r2_elastic
    ))

}

kbl(comparison_table_elastic, caption = "Porównanie metryk modeli>") |>
        kable_styling(bootstrap_options = c("striped", "hover"))
```

Tabela porównawcza przedstawia metryki (MSE, RMSE, MAE, R-kwadrat) uzyskane dla wszystkich modeli (regresji wielorakiej, Lasso, grzbietowej i Elastic Net) na zbiorze testowym.

Model regresji wielorakiej nadal osiąga najlepsze wyniki w kategorii MSE, RMSE i MAE (MSE: 394.18, RMSE: 19.85, MAE: 14.29) i R-kwadrat (0.368).

Tabela 15: Porównanie metryk modeli>

Model	MSE	RMSE	MAE	R_squared
Regresja Wieloraka	394.1834	19.85405	14.29713	0.3683939
Lasso - Optymalne Lambda	403.5354	20.08819	14.50557	0.3534091
Lasso - Większe Lambda	420.3971	20.50359	14.33871	0.3263913
Regresja Grzbietowa - Optymalne Lambda	403.1405	20.07836	14.42975	0.3540418
Regresja Grzbietowa - Większe Lambda	450.7849	21.23169	15.04752	0.2777005
Elastic Net - alpha=0.1	403.1219	20.07789	14.50112	0.3540717
Elastic Net - alpha=0.3	403.6515	20.09108	14.52406	0.3532230
Elastic Net - alpha=0.5	403.5757	20.08919	14.51444	0.3533445
Elastic Net - alpha=0.7	403.3416	20.08337	14.49152	0.3537196
Elastic Net - alpha=0.9	403.3897	20.08456	14.49320	0.3536424

Modele Lasso i grzbietowe z optymalnymi parametrami mają nieco gorsze wyniki (metriki mieszczą się między 400 i 405 dla MSE, 20 i 21 dla RMSE, a 14 i 15 dla MAE, R-squared około 0.35). Widoczne jest, że wzrost parametru lambda (dla Lasso i grzbietowej) powoduje spadek jakości dopasowania. Zauważmy, że wszystkie metody regularyzacyjne (Lasso, Ridge, Elastic Net) nie pobiły wyników modelu regresji wielorakiej.

Modele Elastic Net o różnych wartościach parametru α, osiągają wyniki podobne do modeli Lasso i grzbietowej z optymalnymi wartościami parametrów lambda, ale nie poprawiają wyników regresji wielorakiej. Elastic Net z każdym z testowanych parametrów alpha mają bardzo podobne metryki do Lasso i Ridge, natomiast wszystkie mają wyraźnie gorsze wyniki od modelu regresji wielorakiej.