Case Study: Predykcja cen taksówek

Kamil Pyzik

2026-01-10

Contents

1 Cel i zakres projektu

Głównym celem projektu jest zbudowanie modelu regresyjnego przewidującego cenę przejazdu taksówką. Użyty zbiór danych to *taxi_price*.

Projekt porównuje dwie strategie modelowania - nieliniową i liniową:

- Model 1 (**Random Forest**): wylosowany model. Algorytm lasów losowych bardzo dobrze wychwytuje złożone, nieliniowe zależności. Ponieważ nie obsługuje braków danych, zastosowano zaawansowaną imputację algorytmem **missForest**. Wybrano ją również w celu zachowania spójności z wylosowanym modelem.
- Model 2 (**Elastic Net**): model liniowy z regularyzacją łączącą kary L1 (Lasso) i L2 (Ridge). Wymaga on kompletnych danych numerycznych (One-Hot Encoding). Jego główną zaletą jest "przezroczystość" – podaje konkretne współczynniki (np. cena za kilometr), co pozwala zweryfikować, czy zjawisko ma charakter prosty (liniowy), czy złożony oraz jakie w głównej mierze wpływają na wynik.

Hipoteza badawcza: Weryfikuję, czy złożoność obliczeniowa lasu losowego jest uzasadniona. Jeśli prostszy model liniowy (Elastic Net) osiągnie zbliżone wyniki, będzie on preferowany w podobnych analizach ze względu na łatwość interpretacji.

2 Przygotowanie środowiska i danych

2.1 Biblioteki

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(caret)
library(randomForest)
library(missForest)
library(glmnet)
library(lubridate)
library(corrplot)
library(patchwork)
library(lmtest)
library(nortest)
library(knitr)
library(corrplot)

# Ustawienie ziarna dla powtarzalności wyników
set.seed(2026)
```

2.2 Wczytanie i wstępne przygotowanie danych do analizy

```
# Wczytanie danych
dane <- read.csv("taxi_trip_pricing.csv", na.strings = c("", "NA"))

# Zmiana nazw niektórych kolumn
dane <- dane %>%
  rename(
    Distance = Trip_Distance_km,
    Time = Time_of_Day,
    Day = Day_of_Week,
    Passengers = Passenger_Count,
    Traffic = Traffic_Conditions,
    Duration = Trip_Duration_Minutes,
    Price = Trip_Price
  )

# Konwersja zmiennych kategorycznych na faktory
dane$Traffic <- as.factor(dane$Traffic)
dane$Weather <- as.factor(dane$Weather)
dane$Day <- as.factor(dane$Day)
dane$Time <- as.factor(dane$Time)

str(dane)
```

```
## 'data.frame':    1000 obs. of  11 variables:
##  $ Distance       : num  19.4 47.6 36.9 30.3 NA ...
```

```
##  $ Time           : Factor w/ 4 levels "Afternoon","Evening",..: 3 1 2 2 2 1 1 2 3 1 ...
##  $ Day            : Factor w/ 2 levels "Weekday","Weekend": 1 1 2 1 1 2 1 2 1 1 ...
##  $ Passengers     : num  3 1 1 4 3 2 4 3 3 2 ...
##  $ Traffic        : Factor w/ 3 levels "High","Low","Medium": 2 1 1 2 1 3 1 NA 1 2 ...
##  $ Weather        : Factor w/ 3 levels "Clear","Rain",..: 1 1 1 NA 1 1 2 1 1 2 ...
##  $ Base_Fare      : num  3.56 NA 2.7 3.48 2.93 2.55 3.51 2.97 2.77 3.39 ...
##  $ Per_Km_Rate    : num  0.8 0.62 1.21 0.51 0.63 1.71 1.66 1.87 1.78 1.52 ...
##  $ Per_Minute_Rate: num  0.32 0.43 0.15 0.15 0.32 0.48 NA 0.23 0.34 0.47 ...
##  $ Duration       : num  53.8 40.6 37.3 116.8 22.6 ...
##  $ Price          : num  36.3 NA 52.9 36.5 15.6 ...
```

2.3 Opis zmiennych

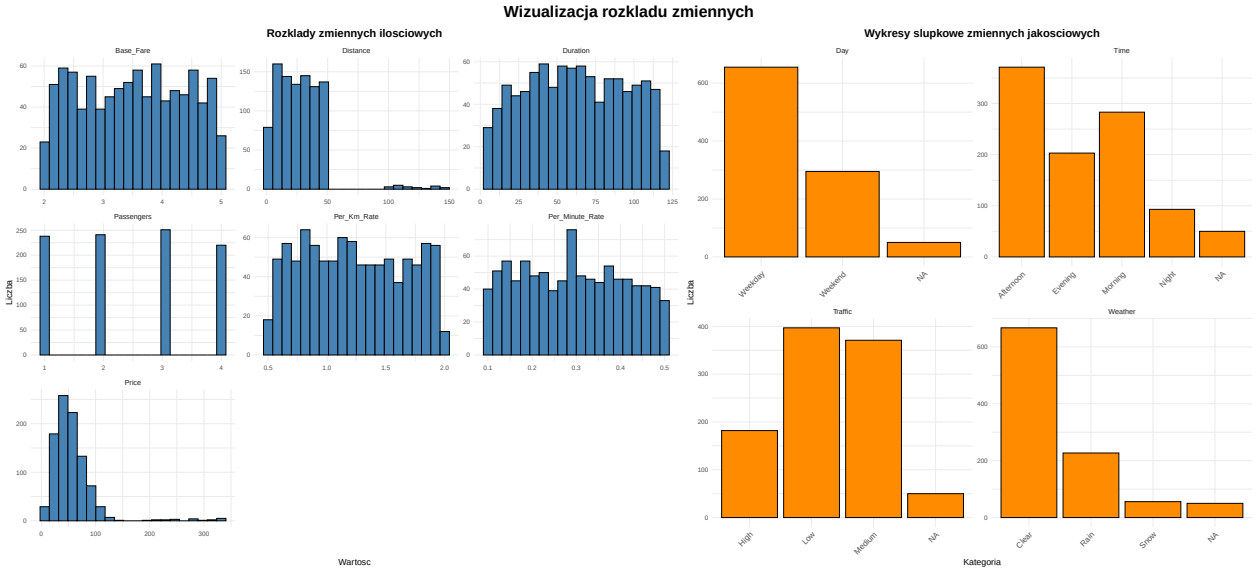
- **Distance** - długość trasy przejazdu wyrażona w kilometrach.
- **Time** - pora dnia, w której rozpoczął się przejazd (Morning, Afternoon, Evening, Night).
- **Day** - dzień tygodnia, w którym odbył się przejazd (Weekday, Weekend).
- **Passengers** - liczba pasażerów podróżujących taksówką w trakcie przejazdu.
- **Traffic** - zmienna kategoryczna opisująca natężenie ruchu drogowego podczas przejazdu (low, medium, high).
- **Weather** - zmienna kategoryczna opisująca warunki pogodowe w trakcie przejazdu (clear, rain, snow).
- **Base_Fare** - opłata początkowa naliczana na początku przejazdu, niezależna od dystansu i czasu trwania.
- **Per_Km_Rate** - stawka naliczana za każdy przejechany kilometr.
- **Per_Minute_Rate** - stawka naliczana za każdą minutę trwania przejazdu.
- **Duration** - całkowity czas trwania przejazdu wyrażony w minutach.
- **Price** - całkowita cena przejazdu taksówką wyrażona w dolarach amerykańskich (USD). Jest to zmienna objaśniana wykorzystywana w modelach regresyjnych.

3 Eksploracyjna analiza danych i braki danych

3.1 Wizualizacja zmiennych ilościowych i jakościowych

```r
# Cechy ilościowe
num_vars <- c("Distance", "Passengers", "Base_Fare", "Per_Km_Rate",
              "Per_Minute_Rate", "Duration", "Price")

# Przekształcenie cech ilościowych do postaci długiej
dane_num_long <- dane %>%
  select(all_of(num_vars)) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value")

# Wykres rozkładu cech ilościowych
p1 <- ggplot(dane_num_long, aes(x = Value)) +
  geom_histogram(bins = 20, fill = "steelblue", color = "black") +
  facet_wrap(~ Variable, scales = "free", ncol = 3) +
  labs(
    title = "Rozkłady zmiennych ilościowych",
    x = "Wartość",
```

```
    y = "Liczba"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
    axis.text = element_text(size = 8)
  )

# Cechy jakościowe
cat_vars <- c("Time", "Day", "Traffic", "Weather")

# Przekształcenie cech jakościowych do postaci długiej
dane_cat_long <- dane %>%
  select(all_of(cat_vars)) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Category")

# Wykres rozkładu cech jakościowych
p2 <- ggplot(dane_cat_long, aes(x = Category)) +
  geom_bar(fill = "darkorange", color = "black") +
  facet_wrap(~ Variable, scales = "free", ncol = 2) +
  labs(
    title = "Wykresy słupkowe zmiennych jakościowych",
    x = "Kategoria",
    y = "Liczba"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
    axis.text.y = element_text(size = 8)
  )

# Wykres zmiennych ilościowych i jakościowych
combined_plot <- p1 + p2 +
  plot_layout(widths = c(1.2, 1)) +
  plot_annotation(
    title = "Wizualizacja rozkładu zmiennych",
    theme = theme(plot.title = element_text(size = 20, hjust = 0.5, face = "bold"))
  )

print(combined_plot)
```



Analiza rozkładu zmiennych ilościowych: Base_Fare, Duration, Per_Km_Rate, Per_Minute_Rate charakteryzują się płaskimi rozkładami. W związku z tym wykonano test Kołmogorowa-Smirnowa, który odrzucił hipotezę o ciągłym rozkładzie jednostajnym. Zmienna Passengers również charakteryzuje się wizualnie płaskim wykresem. Wykonano dla niej test Chi-kwadrat, który wykazał brak podstaw do odrzucenia hipotezy o jednostajnym charakterze tych danych (p-value=0.55). Distance i Price mają rozkłady prawoskośne, dla modelu liniowego należy przetestować zastosowanie transformacji logarytmicznej. Cena większości kursów wynosi maksymalnie około 100 USD. Skrajne wartości wynoszą ponad 300 USD. Dystans większości kursów nie przekracza 50 km. Skrajne długości kursów wynoszą do 150 km. Dla tej cechy obserwowany jest wyraźny brak danych w przedziale 50 km - 100 km. W ramach preprocessingu przetestowano eliminację wartości odstających przy użyciu metody 3-sigma ($\bar{x} \pm 3\sigma$). Eksperyment ten skutkował usunięciem kursów dalekobieżnych, co automatycznie wyeliminowało najwyższe kwoty transakcji (potwierdzając silną korelację Dystans-Cena). Ostatecznie zdecydowano się nie usuwać tych obserwacji, interpretując je jako poprawne logistycznie kursy międzymiastowe (np. transfery lotniskowe), które stanowią istotny element analizowanego zjawiska, mimo że mogą utrudniać dopasowanie modeli liniowych.

Analiza rozkładu zmiennych jakościowych: Dla każdej ze zmiennych występują braki danych. Większość kursów odbywa się w dni robocze. Dominującą porą jest popołudnie. Pogoda w trakcie trwania większości kursów jest dobra, natężenie ruchu niskie lub średnie.

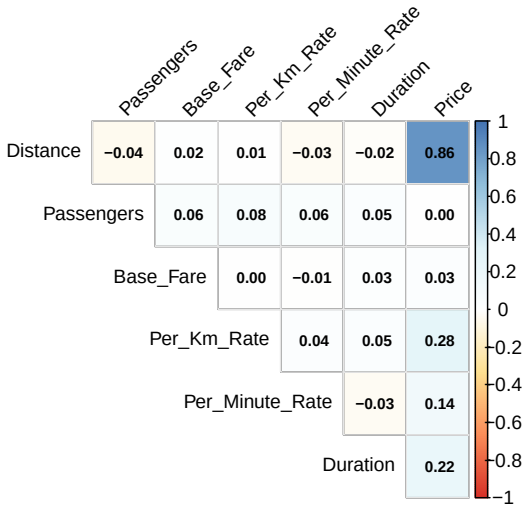
3.2 Analiza macierzy wartości współczynników korelacji między zmiennymi ilościowymi

```
# Wybór tylko zmiennych numerycznych
dane_num_viz <- dane %>%
  select_if(is.numeric) %>%
  na.omit()

# Obliczenie współczynników korelacji
cor_pearson <- cor(dane_num_viz, method = "pearson")
cor_spearman <- cor(dane_num_viz, method = "spearman")

col_palette <- colorRampPalette(c("#D73027", "#FC8D59", "#FEE090",
                                  "#FFFFFF",
                                  "#E0F3F8", "#91BFDB", "#4575B4"))(200)

# Wizualizacja
```

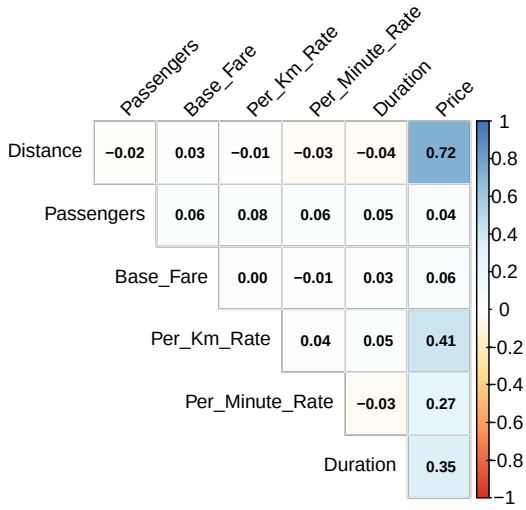
```r
par(mfrow = c(1, 2), oma = c(0, 0, 4, 0))

corrplot(cor_pearson,
         method = "color",
         col = col_palette,
         type = "upper",
         diag = FALSE,

         addCoef.col = "black",
         number.cex = 0.7,

         tl.col = "black",
         tl.srt = 45,
         tl.cex = 0.8,

         addgrid.col = "gray90",
         outline = TRUE,

         title = "Korelacja liniowa Pearsona",
         mar = c(0, 0, 2, 0)
)

corrplot(cor_spearman,
         method = "color",
         col = col_palette,
         type = "upper",
         diag = FALSE,
         addCoef.col = "black",
         number.cex = 0.7,
         tl.col = "black",
         tl.srt = 45,
         tl.cex = 0.8,
         addgrid.col = "gray90",
         outline = TRUE,
         title = "Korelacja Spearmana (monotoniczna)",
         mar = c(0, 0, 2, 0)
)


mtext("Macierze wartości współczynników korelacji między zmiennymi ilościowymi", outer = TRUE, cex = 1.5
```

Macierze wartosci współczynników korelacji miedzy zmiennymi ilosciowymi

Korelacja liniowa Pearsona



Korelacja Spearmana (monotoniczna)




```
# Reset ustawień graficznych
par(mfrow = c(1, 1), oma = c(0,0,0,0))
```

W celu zachowania przejrzystości raportu, zdecydowano się zaprezentować macierz wartości współczynników korelacji ograniczoną wyłącznie do zmiennych ilościowych. Wstępna analiza pełnej macierzy (obejmującej zmienne kategoryczne po transformacji One-Hot Encoding) wykazała, że wartości współczynników korelacji między cechami jakościowymi a zmienną celu były marginalne i wprowadzały szum informacyjny, utrudniając interpretację kluczowych zależności.

Wnioski z analizy macierzy wartości współczynników korelacje między zmiennymi ilościowymi:

1. **Dominacja dystansu:** Najsilniejszym predyktorem ceny jest dystans, wykazujący bardzo silną korelację dodatnią zarówno w ujęciu liniowym Pearsona ($r = 0.86$), jak i monotonicznym Spearmana ($r = 0.72$). Potwierdza to intuicyjny fakt, że długość trasy jest głównym składnikiem taryfy.
2. **Nieliniowość stawek:** Zmienna *Per_Km_Rate* wykazuje umiarkowaną korelację Pearsona (0.28), która wyraźnie wzrasta w korelacji Spearmana (0.41). Sugeruje to, że wpływ stawki za kilometr na cenę końcową ma charakter monotoniczny, ale nie ściśle liniowy – co stanowi silny argument za zastosowaniem modeli nieliniowych, takich jak Random Forest.
3. **Brak wpływu liczby pasażerów:** Zmienna *Passengers* wykazuje zerową lub śladową korelację z ceną (0.00 - 0.04). Oznacza to, że liczba pasażerów nie wpływa na koszt przejazdu, co jest zgodne z typowymi systemami taryfowymi (cena za auto, a nie za osobę).

3.3 3.3 Analiza mechanizmu braku danych

```
# Obliczenie liczby i procentu braków dla każdej kolumny
missing_summary <- data.frame(
  Zmienna = names(dane),
  Liczba_Brakow = sapply(dane, function(x) sum(is.na(x))),
  Procent_Brakow = sapply(dane, function(x) round(mean(is.na(x)) * 100, 2))
)

# Sortowanie malejąco
missing_summary <- missing_summary[order(-missing_summary$Procent_Brakow), ]
```

```r
# Wyświetlenie tabeli
kable(missing_summary,
      row.names = FALSE,
      caption = "Liczba i udział procentowy braków danych w poszczególnych zmiennych.")
```

Table 1: Liczba i udział procentowy braków danych w poszczególnych zmiennych.

Zmienna	Liczba_Brakow	Procent_Brakow
Distance	50	5.0
Time	50	5.0
Day	50	5.0
Passengers	50	5.0
Traffic	50	5.0
Weather	50	5.0
Base_Fare	50	5.0
Per_Km_Rate	50	5.0
Per_Minute_Rate	50	5.0
Duration	50	5.0
Price	49	4.9

```r
cat("Liczba wierszy z co najmniej jednym brakiem:", sum(!complete.cases(dane)),
    "(",round(mean(!complete.cases(dane)) * 100, 2),"%)\n")
```

```
## Liczba wierszy z co najmniej jednym brakiem: 438 ( 43.8 %)
```

Zaobserwowano równomierny rozkład braków – niemal każda zmienna objaśniająca zawiera dokładnie **5.0% (50)** brakujących wartości. Braki te nie są skorelowane (nie występują w tych samych wierszach), a w rezultacie, aż **43.8% (438 z 1000)** wszystkich obserwacji jest niekompletnych.

W świetle powyższych wyników odrzucono strategię usuwania wierszy z brakami, gdyż wiązałaby się z utratą niemal połowy zbioru danych, co znacząco obniżyłoby moc statystyczną modeli.

W celu weryfikacji mechanizmu powstawania braków danych w zmiennej celu (*Price*), przeprowadzono serię testów statystycznych badających zależność między faktem braku ceny a wartościami pozostałych atrybutów. Dobór testów podyktowany był charakterystyką zmiennych.

1. **Test Wilcoxona (Manna-Whitneya):** Zastosowano dla zmiennych ilościowych (np. *Distance*, *Duration*) ze względu na ich niesymetryczne rozkłady, dla których założenia parametrycznego testu t-Studenta nie byłyby spełnione.
2. **Test Fishera / Chi-Kwadrat:** Zastosowano dla zmiennych jakościowych. Algorytm automatycznie dobiera test Fishera w przypadkach małych liczebności w tabelach kontyngencji, aby zapewnić rzetelność wyniku.

```r
if(!"missing_price" %in% names(dane)) {
  dane$missing_price <- is.na(dane$Price)
}


results_mcar <- data.frame(
  Cecha = character(),
  Typ_Testu = character(),
  P_Value = numeric(),
  Werdykt = character(),
```

```r
  stringsAsFactors = FALSE
)

# Lista kolumn do sprawdzenia
features_to_check <- setdiff(names(dane), c("Price", "missing_price"))

for (feature in features_to_check) {

  x <- dane[[feature]]
  y <- dane$missing_price

  # Pomijamy, jeśli zmienna ma same NA lub jedną wartość
  if (length(unique(x)) < 2) next

  if (is.numeric(x)) {
    test <- wilcox.test(x ~ y) # Test Wilcoxona
    test_name <- "Wilcoxon"
    p_val <- test$p.value

  } else {
    tbl <- table(x, y)
    # Używamy Fishera (dla małych próbek) lub Chi-Kwadrat
    # tryCatch obsłuży ewentualne błędy przy bardzo dziwnych danych
    try({
      if (any(tbl < 5)) {
        test <- fisher.test(tbl, simulate.p.value = TRUE)
        test_name <- "Fisher"
      } else {
        test <- chisq.test(tbl)
        test_name <- "Chi-Squared"
      }
      p_val <- test$p.value
    }, silent = TRUE)
  }

  # Interpretacja wyniku
  werdykt <- ifelse(p_val > 0.05, "Brak Zależności", "Zależnośc")

  # Zapisanie wyniku
  results_mcar <- rbind(results_mcar, data.frame(
    Cecha = feature,
    Typ_Testu = test_name,
    P_Value = round(p_val, 2),
    Werdykt = werdykt
  ))
}

# Wyświetlenie raportu końcowego
kable(results_mcar,
      row.names = FALSE,
      caption = "Testy statystyczne zależności między brakiem ceny a pozostałymi cechami")
```

Table 2: Testy statystyczne zależności między brakiem ceny a pozostałymi cechami

Cecha	Typ_Testu	P_Value	Werdykt
Distance	Wilcoxon	0.71	Brak Zależności
Time	Fisher	0.37	Brak Zależności
Day	Chi-Squared	0.63	Brak Zależności
Passengers	Wilcoxon	0.75	Brak Zależności
Traffic	Chi-Squared	0.63	Brak Zależności
Weather	Fisher	0.72	Brak Zależności
Base_Fare	Wilcoxon	0.34	Brak Zależności
Per_Km_Rate	Wilcoxon	0.27	Brak Zależności
Per_Minute_Rate	Wilcoxon	0.24	Brak Zależności
Duration	Wilcoxon	0.74	Brak Zależności

Dla wszystkich analizowanych cech wartość p (*p-value*) znacząco przekracza poziom istotności $\alpha = 0.05$ (zakres od 0.24 do 0.75). Oznacza to brak podstaw do odrzucenia hipotezy zerowej o niezależności braków danych. Potwierdza to, że braki w zmiennej *Price* mają charakter całkowicie losowy (MCAR). W rezultacie zdecydowano się na usunięcie niekompletnych wierszy w zmiennej celu bez ryzyka obciążenia próby badawczej.

4 Podział danych oraz imputacja

```
#Podział danych

# Usunięcie wierszy z brakami danych dla zmiennej celu
dane_clean <- dane %>%
  filter(!is.na(Price)) %>%
  select(-missing_price)

complete_indices <- which(complete.cases(dane_clean))

final_4_indices <- sample(complete_indices, 4)

data_final_4 <- dane_clean[final_4_indices, ]
data_working <- dane_clean[-final_4_indices, ]

train_index <- createDataPartition(data_working$Price, p=0.8, list=FALSE)

data_train <- data_working[train_index, ]
data_test  <- data_working[-train_index, ]
```

Zgodnie z założeniami projektu do zbioru testowego wydzielono 4 obserwacje. Celem weryfikacji zdolności generalizacji modeli w typowych warunkach wybrano losowo rekordy kompletne. Resztę danych podzielono na zbiór treningowy i walidacyjny, stosując proporcję 80 do 20.

```
# Funkcja usuwająca kolumny logiczne
remove_logical_cols <- function(df) {
  logical_cols <- names(df)[sapply(df, is.logical)]

  if(length(logical_cols) > 0) {
    df <- df[, !names(df) %in% logical_cols]
```

```
  }
  return(df)
}

# Zastosowanie czyszczenia do obu zbiorów
data_train <- remove_logical_cols(data_train)
data_test <- remove_logical_cols(data_test)

# IMPUTACJA DANYCH TRENINGOWYCH
train_price <- data_train$Price
data_train_no_y = data_train %>% select(-Price)

mf_train <- missForest(data_train_no_y, verbose = FALSE)
train_rf_imputed <- mf_train$ximp

train_rf_imputed$Price <- train_price

# IMPUTACJA DANYCH TESTOWYCH
test_price <- data_test$Price
data_test_no_y <- data_test %>% select(-Price)

mf_test <- missForest(data_test_no_y, verbose = FALSE)
test_rf_imputed <- mf_test$ximp

test_rf_imputed$Price <- test_price
```

Braki danych uzupełniono wykorzystując algorytm missForest. Zapewnia on imputację zarówno danych ilościowych, jak i jakościowych. Ponadto missForest, wykorzystując lasy losowe, pozwala na imputację spójną z naturą modelu predykcyjnego oraz wylosowanym modelem.

Wzorową techniką imputacji powinno być dopasowanie algorytmu do danych treningowych, a następnie predykcja na zbiorze walidacyjnym. Ze względu na ograniczenia implementacyjne pakietu w R takie podejście nie jest możliwe. W związku z tym imputacja została wykonana osobno na zbiorze treningowym oraz walidacyjnym. Podczas imputacji usunięto z danych zmienną Price. Taki sposób postępowania pozwala na uniknięcie wycieku danych między zbiorami.

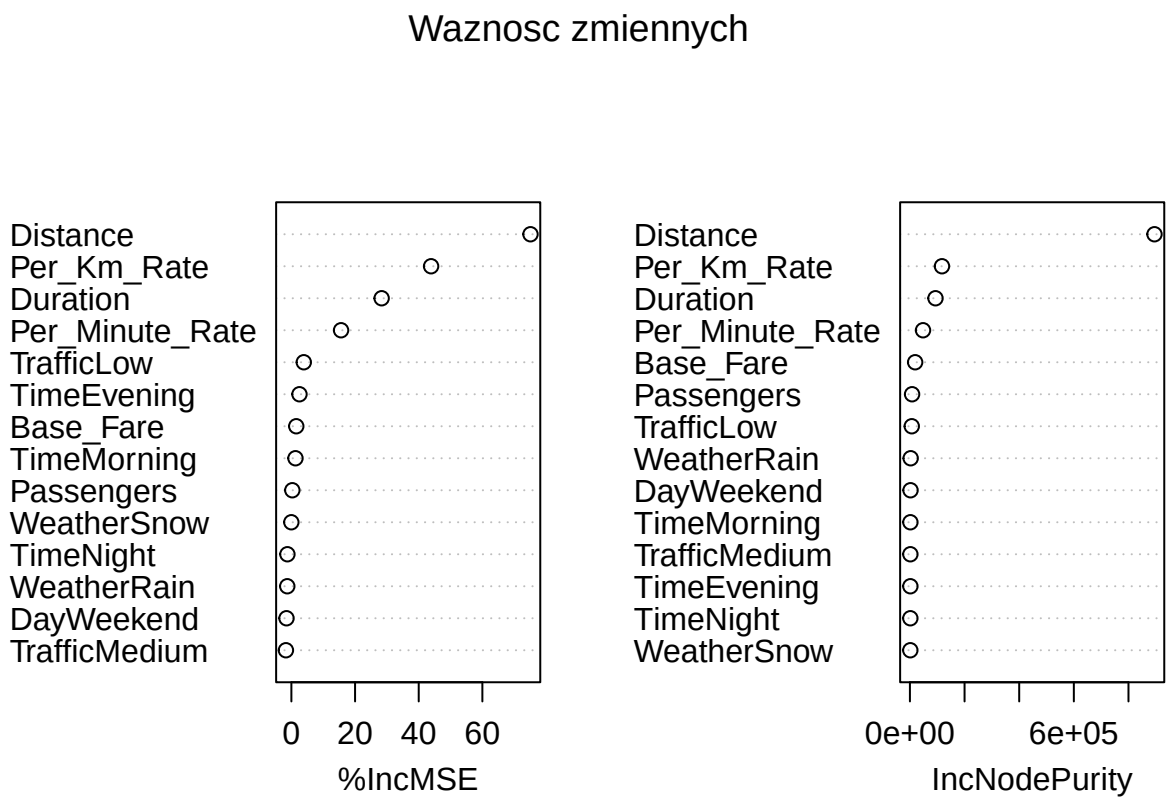
5 Model 1: Random Forest

```
# Trening modelu Random Forest

ctrl <- trainControl(method = "cv", number = 10)

grid_rf = expand.grid(mtry = seq(2, 7, by=1))
rf_model <- train(Price ~ .,
                  data = train_rf_imputed,
                  method = "rf",
                  trControl = ctrl,
                  importance = TRUE,
                  tuneGrid = grid_rf)

rf_predictions <- predict(rf_model, newdata = test_rf_imputed)

rf_metrics <- postResample(pred = rf_predictions, obs = test_rf_imputed$Price)
```

```r
print(rf_metrics)
```

```
##       RMSE   Rsquared        MAE
## 10.1240140  0.9482239  6.2859349
```

```r
varImpPlot(rf_model$finalModel, main = "Ważność zmiennych")
```



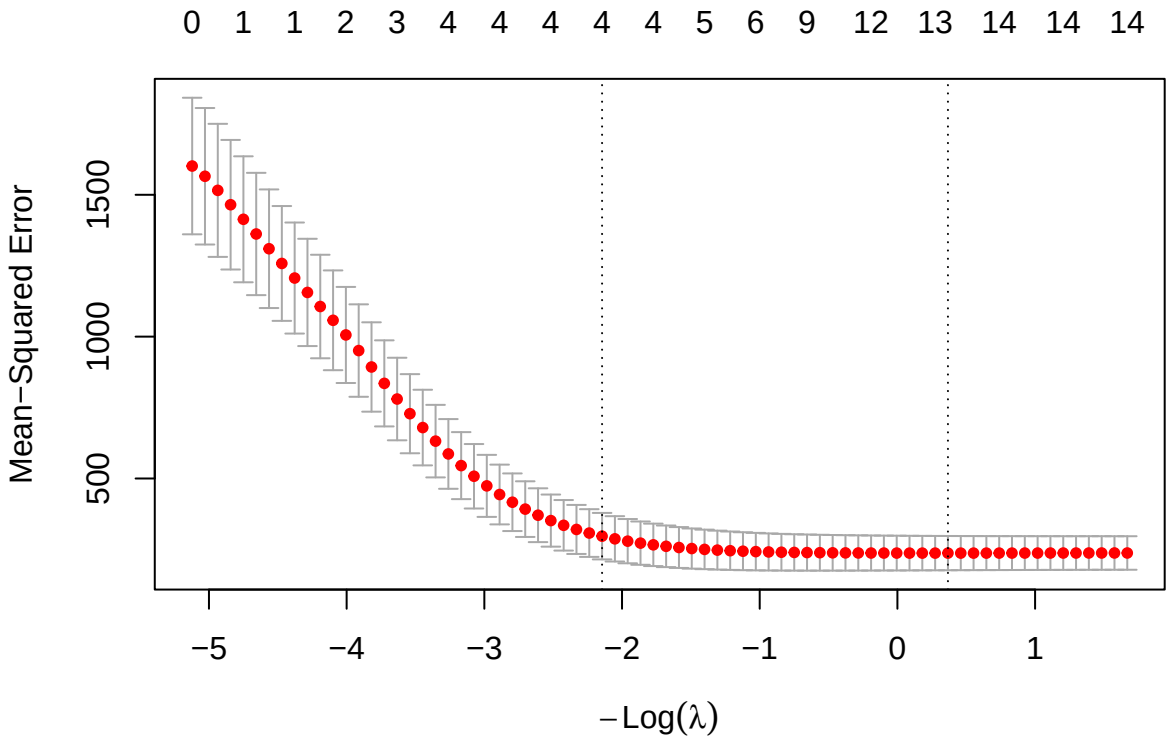
Uzyskane wyniki potwierdzają bardzo wysoką jakość dopasowania modelu nieliniowego do danych. Model wyjaśnia aż **95% zmienności** ceny przejazdu. Jest to wynik bardzo wysoki, świadczący o tym, że wybrane zmienne objaśniające (głównie dystans i stawki) zawierają niemal kompletną informację potrzebną do wyznaczenia ceny. Model skutecznie odwzorował logikę taryfową. Średni błąd bezwzględny informuje, że przeciętna pomyłka modelu wynosi ok. **6.29 USD**. W kontekście całego zbioru danych (gdzie ceny sięgają kilkuset dolarów), jest to błąd akceptowalny. Oznacza wysoką precyzję dla typowych, codziennych kursów. Wartość RMSE jest wyraźnie wyższa od MAE ($10.12 > 6.29$). Ponieważ RMSE silniej karze duże pomyłki (podnosząc błędy do kwadratu). Model, jest na ogół precyzyjny, ale w nielicznych przypadkach (ekstremalnie długie trasy lub nietypowe warunki) popełnia większe błędy, co zawyża tę metrykę.

Dla uzyskanego modelu wykonano również wykresy przedstawiające ranking zmiennych według ich wpływu na jakość modelu. Wykorzystują one dwie metryki: 1. **%IncMSE (Wzrost błędu średniokwadratowego):** Określa, jak bardzo pogorszyłaby się predykcja modelu, gdyby wartości danej zmiennej zostały losowo przemieszane (usunięcie informacji). 2. **IncNodePurity (Wzrost czystości węzłów):** Określa, jak dobrze dana zmienna dzieli dane na jednorodne grupy w strukturze drzewa.

Zmienna *Distance* jest bezapelacyjnie najważniejszym predyktorem w obu metrykach. Jest to zgodne z logiką taryfikatorów taksówek, gdzie odległość jest głównym składnikiem ceny. Zmienne *Per_Km_Rate* oraz *Duration* zajmują kolejne miejsca na podium. Potwierdza to, że model poprawnie zidentyfikował strukturę cenotwórczą: $Cena \approx f(Dystans, Stawka, Czas)$. Zmienna *TimeEvening* wykazuje wysoką pozycję w metryce %IncMSE, mimo niskiej pozycji w IncNodePurity. Oznacza to, że choć przejazdy wieczorem są zjawiskiem rzadszym (mały wpływ na strukturę drzewa), to ich wystąpienie ma krytyczny wpływ na cenę.

6 Model 2: Elastic Net Regression

```r
# Przygotowanie macierzy (One-Hot)
X_train_enet <- model.matrix(Price ~ . -1, data = train_rf_imputed)
y_train_enet <- train_rf_imputed$Price
X_test_enet <- model.matrix(Price ~ . -1, data = test_rf_imputed)
y_test_enet <- test_rf_imputed$Price


enet_cv <- cv.glmnet(
  x = X_train_enet,
  y = y_train_enet,
  alpha = 0.2,
  family = "gaussian",
  nfolds = 5,
  type.measure = "mse",
  standardize = TRUE
)

best_lambda <- enet_cv$lambda.min
plot(enet_cv)
```




```r
# Ocena
pred_enet <- predict(enet_cv, s = best_lambda, newx = X_test_enet)
enet_metrics <- postResample(pred = as.vector(pred_enet), obs = y_test_enet)
print(enet_metrics)
```

```
##       RMSE   Rsquared        MAE
## 16.2132403  0.8613402  9.5446239
```

```r
# Współczynniki
coef_enet <- coef(enet_cv, s = best_lambda)
print(as.matrix(coef_enet)[as.matrix(coef_enet) != 0, , drop=FALSE])
```

```
##                 s=0.6927122
## (Intercept)     -56.3362027
## Distance          1.6936357
## TimeAfternoon     0.3234873
## TimeEvening      -1.0843671
## TimeNight        -0.5279752
## DayWeekend       -0.5675089
## Passengers        0.1638899
## TrafficLow       -0.5040522
## TrafficMedium    -0.2798500
## WeatherSnow       1.5440944
## Base_Fare         0.6445116
## Per_Km_Rate      24.9613548
## Per_Minute_Rate  55.9718735
## Duration          0.2874898
```

Przeprowadzono diagnostykę transformacji metodą Box-Coxa, która dla zmiennej *Price* wskazała parametr $\lambda \approx 0$, sugerując zasadność zastosowania logarytmu naturalnego w celu normalizacji rozkładu.

Jednakże, eksperymenty empiryczne wykazały, że zastosowanie transformacji logarytmicznej drastycznie pogarsza jakość modelu Elastic Net (R^2 spada z poziomu 0.86 do 0.72).

Analiza wskazuje, że mechanizm kształtowania ceny w badanym zbiorze ma charakter liniowy ($Cena \propto Dystans$), a nie wykładniczy. Zastosowanie logarytmu na zmiennej celu ($\log Y \sim X$) wymusza na modelu poszukiwanie relacji wykładniczej ($Y \sim e^X$), co prowadzi do ogromnych błędów predykcji dla dłuższych tras.

Wynonano również transformacje logarytmiczną obu kolumn prawoskośnych - ceny i dystansu. Również nie podniosła ona jakości modelu.

Analiza współczynników modelu Elastic Net ujawnia fundamentalne ograniczenia podejścia liniowego w predykcji ceny kursu taksówki:

1. **Ujemny Intercept (-56.34 USD):** Model wyznaczył teoretyczną opłatę początkową na poziomie -56 dolarów. Jest to wartość niefizyczna i absurdalna biznesowo. Wynika ona z konieczności matematycznej kompensacji – model musiał drastycznie obniżyć "podłogę", aby zrównoważyć bardzo wysokie wagi przypisane stawkom.
2. **Przeszacowanie stawek (Rate Coefficients):** Zmienne *Per_Km_Rate* (24.96) oraz *Per_Minute_Rate* (55.97) otrzymały nienaturalnie wysokie wagi. Model próbuje symulować iloczyn ($Stawka \times Ilo$) poprzez dodawanie stałych, wysokich kwot za sam fakt obowiązywania wyższej taryfy. Prowadzi to do dużych błędów na trasach, gdzie stawka jest wysoka, ale dystans krótki.
3. **Wniosek końcowy:** Spadek R^2 do poziomu 0.86 (wobec 0.95 dla Random Forest) oraz wysoki błąd RMSE (16.21) potwierdzają, że relacja cenowa ma charakter **multiplikatywny**, a nie addytywny. Elastic Net, jako model liniowy, osiągnął tu swoje maksimum, nie będąc w stanie poprawnie odwzorować mechanizmu dynamicznego cennika bez ręcznego tworzenia zmiennych interakcji.

7 Analiza wykonanych modeli

7.1 Metryki modeli

```r
# MAPE
get_mape <- function(actual, predicted) {
```

```
  mean(abs((actual - predicted) / actual)) * 100
}

rf_mape <- get_mape(test_rf_imputed$Price, rf_predictions)
enet_mape <- get_mape(y_test_enet, as.vector(pred_enet))

model_comparison <- data.frame(
  Model = c("Random Forest", "Elastic Net"),
  RMSE = c(rf_metrics["RMSE"], enet_metrics["RMSE"]),
  Rsquared = c(rf_metrics["Rsquared"], enet_metrics["Rsquared"]),
  MAE = c(rf_metrics["MAE"], enet_metrics["MAE"]),
  MAPE_Percent = c(rf_mape, enet_mape)
)

kable(model_comparison,
      digits = 2,
      col.names = c("Model", "RMSE", "R²", "MAE", "MAPE"),
      caption = "Zestawienie metryk wykonanych modeli")
```

Table 3: Zestawienie metryk wykonanych modeli

Model	RMSE	R^2	MAE	MAPE
Random Forest	10.12	0.95	6.29	13.69
Elastic Net	16.21	0.86	9.54	19.94

Przeprowadzona analiza metryk (RMSE, R^2, MAE, MAPE) jednoznacznie wskazuje na **Random Forest** jako lepszy model do predykcji cen przejazdów.

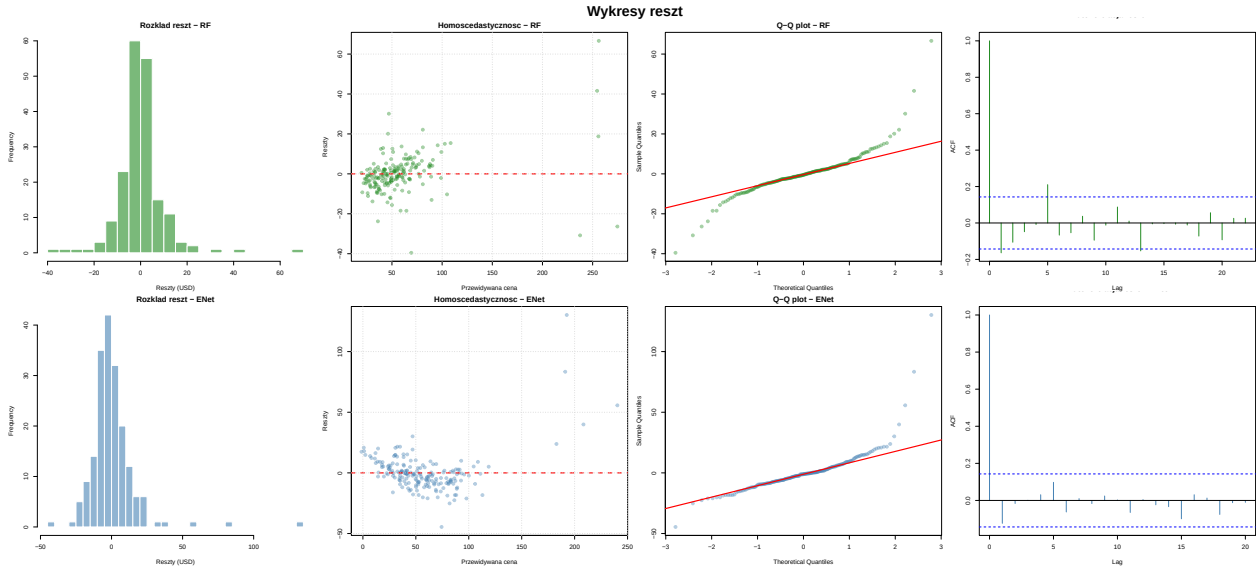
Model ten zdeklasował podejście liniowe (Elastic Net), osiągając znaczącą poprawę precyzji w każdym wymiarze:

- **Jakość dopasowania:** Model wyjaśnia aż **95% zmienności** ($R^2 = 0.95$), podczas gdy model liniowy napotkał barierę na poziomie 86%, nie będąc w stanie odwzorować nieliniowej struktury taryf.
- **Precyzja biznesowa:** Średni błąd procentowy (MAPE) jest dla Elastic Net jest o ponad 6 punktów procentowych wyższy od Random Forest, **19.94%** (Elastic Net) do **13.69%** (Random Forest).
- **Stabilność:** Istotnie niższa wartość RMSE (10.12 vs 16.21) dowodzi, że Random Forest jest znacznie bardziej odporny na wartości skrajne i lepiej radzi sobie z nietypowymi trasami.

7.2 Analiza reszt modeli

```
rf_reszty = test_rf_imputed$Price - rf_predictions
enet_preds_vec <- as.vector(pred_enet)
enet_reszty <- test_rf_imputed$Price - enet_preds_vec
par(mfrow = c(2, 4), mar = c(4, 4, 2, 1), oma = c(0, 0, 3, 0))

# RF
hist(rf_reszty,
     main = "Rozkład reszt - RF",
     col = alpha("forestgreen", 0.6),
     border = "white",
     breaks = 30,
     xlab = "Reszty (USD)")
```

```r
plot(rf_predictions, rf_reszty,
     main = "Homoscedastyczność - RF",
     pch = 19,
     col = alpha("forestgreen", 0.4),
     xlab = "Przewidywana cena",
     ylab = "Reszty")
abline(h = 0, col = "red", lwd = 2, lty = 2)
grid()

qqnorm(rf_reszty, main = "Q-Q plot - RF", pch = 19, col = alpha("forestgreen", 0.4))
qqline(rf_reszty, col = "red", lwd = 2)

acf(rf_reszty, main = "Autokorelacja reszt - RF", col = "forestgreen")


# ENet
hist(enet_reszty,
     main = "Rozkład reszt - ENet",
     col = alpha("steelblue", 0.6),
     border = "white",
     breaks = 30,
     xlab = "Reszty (USD)")

plot(enet_preds_vec, enet_reszty,
     main = "Homoscedastyczność - ENet",
     pch = 19,
     col = alpha("steelblue", 0.4),
     xlab = "Przewidywana cena",
     ylab = "Reszty")
abline(h = 0, col = "red", lwd = 2, lty = 2)
grid()

qqnorm(enet_reszty, main = "Q-Q plot - ENet", pch = 19, col = alpha("steelblue", 0.4))
qqline(enet_reszty, col = "red", lwd = 2)


acf(enet_reszty, main = "Autokorelacja reszt - ENet", lag.max = 20, na.action = na.pass, col = "steelblu

mtext("Wykresy reszt",
      outer = TRUE, cex = 1.5, font = 2)
```




```
par(mfrow = c(1, 1))
```

Wizualna ocena rozkładów reszt prowadzi do wniosku, że w obu przypadkach nie mamy do czynienia z idealnym rozkładem normalnym. Kluczowa różnica ujawnia się w analizie wartości skrajnych (ogonów rozkładu). Model Elastic Net generuje ekstremalne błędy sięgające aż 150 USD, podczas gdy maksymalna pomyłka modelu Random Forest wynosi około 60 USD. Potwierdza to znacznie wyższą stabilność i bezpieczeństwo modelu lasów losowych. Z wykresów rozrzutu reszt płynie wniosek, że ich wariancja nie jest stała. Oba modele mają problem z prawidłowych oszacowaniem wysokich cen kursów. Z wykresów Q-Q plot również nasuwa się konkluzja o odrzuceniu hiptezy o rozkładzie normalnym. Trzy wartości autokorelacji reszt modelu Random Forest nie znajdują się w przedziale ufności, z kolei dla modelu Elastic Net wszystkie słupki znajdują się w tym przedziale.

W celu formalnej weryfikacji powyższych obserwacji wizualnych przeprowadzono dedykowane testy statystyczne: - **Test Shapiro-Wilka** - weryfikacja normalności rozkładu (analiza kształtu histogramu), - **Test Andersona-Darlinga** - weryfikacja normalności ze szczególnym uwzględnieniem ogonów rozkładu (analiza Q-Q plot), - **Test Breuscha-Pagana** - weryfikacja założenia o homoskedastyczności (stałości wariancji reszt), - **Test Durbina-Watsona** weryfikacja występowania autokorelacji reszt pierwszego rzędu.

```
# Funkcja generująca jeden wiersz statystyk dla danego modelu
get_diagnostic_row <- function(residuals, predictions, model_name) {

  # 1. Shapiro-Wilk (p-value)
  if (length(residuals) > 5000) {
    shapiro_res <- shapiro.test(sample(residuals, 5000))
    shapiro_p <- shapiro_res$p.value
  } else {
    shapiro_res <- shapiro.test(residuals)
    shapiro_p <- shapiro_res$p.value
  }

  # 2. Anderson-Darling
  ad_res <- ad.test(residuals)
  ad_stat <- ad_res$statistic
  ad_crit <- 0.759

  # 3. Breusch-Pagan (p-value) - homoskedastyczność
  bp_model <- lm(residuals ~ predictions)
```

```
  bp_res <- bptest(bp_model)
  bp_p <- bp_res$p.value

  # 4. Durbin-Watson - autokorelacja
  dw_res <- dwtest(bp_model)
  dw_stat <- dw_res$statistic

  data.frame(
    Model = model_name,
    `Shapiro-Wilk p` = shapiro_p,
    `Breusch-Pagan p` = bp_p,
    `Anderson-Darling stat` = as.numeric(ad_stat),
    `AD crit (5%)` = ad_crit,
    `Durbin-Watson` = as.numeric(dw_stat),
    check.names = FALSE
  )
}

row_rf <- get_diagnostic_row(rf_reszty, rf_predictions, "RandomForest")
row_enet <- get_diagnostic_row(enet_reszty, as.vector(pred_enet), "Elastic Net")

diagnostics_table <- rbind(row_enet, row_rf)

diagnostics_formatted <- diagnostics_table
diagnostics_formatted$`Shapiro-Wilk p` <- round(diagnostics_formatted$`Shapiro-Wilk p`, 3)
diagnostics_formatted$`Breusch-Pagan p` <- round(diagnostics_formatted$`Breusch-Pagan p`, 3)

# Wyświetlenie tabeli za pomocą kable
kable(diagnostics_formatted,
      row.names =  FALSE,
      digits = 3,
      align = "c",
      caption = "Wyniki diagnostyki reszt (testy statystyczne)")
```

Table 4: Wyniki diagnostyki reszt (testy statystyczne)

Model	Shapiro-Wilk p	Breusch-Pagan p	Anderson-Darling stat	AD crit (5%)	Durbin-Watson
Elastic Net	0	0	8.161	0.759	2.208
RandomForest	0	0	6.269	0.759	2.311

Testy statystyczne potwierdziły one wnioski płynące z analizy wykresów. Dla obu modeli można odrzucić hipotezę o rozkładzie normalnym reszt, ich homoscedastyczności oraz normalności w kontekście wykresu Q-Q (statystyka testowa wyższa od wartości krytecznej). Wartość testu Durbina-Watsona (~2) dla modelu Elastic Net pozwala rozważyć hipotezę o autokorelacji reszt. W przypadku modelu lasów losowych skłania się do hipotezy o ujemnej autokorelacji reszt (~2.5).

8 Predykacja na zbiorze testowym

```
# Random Forest
data_final_4_rf <- data_final_4 %>% mutate_if(is.character, as.factor)
```

```r
# Synchronizacja poziomów
for(col in names(data_final_4_rf)) {
  if(is.factor(data_final_4_rf[[col]]) && col %in% names(data_train)) {
    data_final_4_rf[[col]] <- factor(data_final_4_rf[[col]], levels = levels(data_train[[col]]))
  }
}

pred_final_rf = predict(rf_model, newdata = data_final_4_rf)

# Elastic Net
X_final_enet_raw <- model.matrix(Price ~ . -1, data = data_final_4_rf)
expected_cols <- colnames(X_train_enet)
X_final_enet <- matrix(0, nrow = nrow(X_final_enet_raw), ncol = length(expected_cols))
colnames(X_final_enet) <- expected_cols
common_cols <- intersect(colnames(X_final_enet_raw), expected_cols)
X_final_enet[, common_cols] <- X_final_enet_raw[, common_cols]

pred_final_enet <- predict(enet_cv, s = best_lambda, newx = X_final_enet)

# Tabela Wyników
final_showdown <- data.frame(
  Rzeczywista_Cena = round(data_final_4$Price, 2),
  RF_Pred = round(pred_final_rf, 2),
  ENet_Pred = round(as.vector(pred_final_enet), 2),
  RF_Error = round(abs(data_final_4$Price - pred_final_rf), 2),
  ENet_Error = round(abs(data_final_4$Price - as.vector(pred_final_enet)), 2)
)

kable(final_showdown,
      row.names =  FALSE,
      digits = 2,
      align = "c",
      caption = "Wyniki predykcji na zbiorze testowym")
```

Table 5: Wyniki predykcji na zbiorze testowym

Rzeczywista_Cena	RF_Pred	ENet_Pred	RF_Error	ENet_Error
75.93	74.47	79.34	1.46	3.42
19.97	27.82	32.95	7.85	12.98
45.17	41.25	43.56	3.93	1.62
30.94	38.13	34.72	7.19	3.78

```r
total_errors <- data.frame(
  Model = c("Random Forest", "Elastic Net"),
  Suma_Bledow_Absolutnych = c(sum(final_showdown$RF_Error),
                              sum(final_showdown$ENet_Error))
)

kable(total_errors,
      digits = 2,
      col.names = c("Model", "Łączna suma błędów"),
      caption = "Podsumowanie całkowitego odchylenia dla zbioru testowego",
```

```
      align = "c")
```

Table 6: Podsumowanie całkowitego odchylenia dla zbioru testowego

Model	Łączna suma błędów
Random Forest	20.43
Elastic Net	21.80

Weryfikacja modeli na zbiorze testowym ujawniła interesującą charakterystykę obu rozwiązań, wskazując na specyficzne obszary przewagi modelu liniowego, przy jednoczesnym zachowaniu ogólnej dominacji modelu Random Forest.

W analizowanej próbie różnica w jakości dopasowania była nieznaczna. Łączna suma błędów absolutnych wyniosła **20.43** dla Random Forest wobec **21.80** dla Elastic Net. Oznacza to, że w standardowych warunkach (brak wartości skrajnych) model liniowy potrafi dotrzymać kroku modelom złożonym.

Zauważono, że w 50% analizowanych przypadków (dla cen 45.17 USD oraz 30.94 USD) model Elastic Net uzyskał niższą wartość błędu jednostkowego niż Random Forest (np. błąd 1.62 USD vs 3.93 USD). Świadczy to o tym, że dla segmentów danych o silnej liniowej zależności, prosta regresja może być bardziej precyzyjna, unikając wariancji wprowadzanej przez strukturę drzewiastą.

Mimo lokalnych sukcesów, model Elastic Net przegrał rywalizację z powodu jednego istotnego odchylenia. Dla kursu o wartości **19.97 USD**, model liniowy przeszacował cenę aż o ~**13 USD (błąd 12.98)**, podczas gdy Random Forest pomylił się o 7.85 USD.

Wyniki te potwierdzają, że model Elastic Net jest modelem "sztywnym" – działa bardzo dobrze w typowych, liniowych scenariuszach, ale generuje duże błędy, gdy rzeczywistość odbiega od średniej. Random Forest, mimo że czasem mniej precyzyjny w prostych przypadkach, zapewnia lepszą **ochronę przed dużymi pomyłkami**, co ostatecznie czyni go bezpieczniejszym wyborem biznesowym.

9 Wnioski

9.0.1 Ograniczenia i ocena modelu liniowego (Elastic Net)

Mimo że model **Elastic Net** oferuje wysoką interpretowalność dzięki bezpośredniemu dostępowi do wag cech (współczynników regresji), w toku analizy ujawniły się jego **krytyczne ograniczenia strukturalne**:

- **Problematyczne predykcje ujemne:** Najpoważniejszym mankamentem modelu liniowego jest generowanie ujemnych wartości dla zmiennej celowej na zbiorze walidacyjnym. W kontekście analizowanego problemu (gdzie zmienna celowa przyjmuje z definicji wartości nieujemne), predykcje poniżej zera ($y < 0$) są błędem dyskwalifikującym model z zastosowań produkcyjnych bez wprowadzania sztucznych korekt.
- **Niedopasowanie (Underfitting):** Występowanie wartości ujemnych oraz analiza rozkładu reszt sugerują, że model liniowy nie jest w stanie wychwycić nieliniowych zależności oraz interakcji między cechami. Założenie o addytywności wpływu zmiennych okazało się zbyt restrykcyjne dla tego zbioru danych.

9.0.2 Przewaga Lasu Losowego

W przeciwieństwie do modelu liniowego, **Random Forest** poradził sobie ze strukturą danych znacznie lepiej:

- Jako model oparty na drzewach decyzyjnych, naturalnie radzi sobie z nieliniowością.
- Nie ekstrapoluje wyników w sposób niekontrolowany – predykcje mieszczą się w zakresie wartości obserwowanych w liściach drzewa treningowego, co eliminuje problem nielogicznych, ujemnych prognoz.
- Wyższa złożoność obliczeniowa przełożyła się na lepszą generalizację i stabilność wyników.

9.0.3 Weryfikacja hipotezy badawczej

Hipoteza: “Weryfikuję, czy złożoność obliczeniowa lasu losowego jest uzasadniona. Jeśli prostszy model liniowy (Elastic Net) osiągnie zbliżone wyniki, będzie on preferowany w podobnych analizach ze względu na łatwość interpretacji.”

Na podstawie przeprowadzonych eksperymentów hipotezę o preferencji modelu liniowego należy odrzucić, a wyższa złożoność obliczeniowa Lasu Losowego jest w pełni uzasadniona.

Mimo że Elastic Net jest modelem prostszym, nie spełnił on podstawowego warunku „osiągnięcia zbliżonych wyników”. Generowanie ujemnych predykcji jest błędem jakościowym, który sprawia, że model ten nie odzwierciedla poprawnie rzeczywistości fizycznej zjawiska. Zysk z prostoty interpretacji nie rekompensuje utraty poprawności logicznej modelu.