CS_Natalia_Czop

Natalia Czop

2025-11-29

Przygotowanie Danych (Preprocessing)

Wczytanie Danych

```
# Wczytanie trzech zestawów danych
df1 <- read.csv("C:/IiAD/eksploracja_danych/CS/occupancy_data/datatraining.txt",
                header = TRUE, sep = ",")
df2 <- read.csv("C:/IiAD/eksploracja_danych/CS/occupancy_data/datatest.txt",
                header = TRUE, sep = ",")
df3 <- read.csv("C:/IiAD/eksploracja_danych/CS/occupancy_data/datatest2.txt",
                header = TRUE, sep = ",")

# Połączenie danych z wszystkich plików w jeden duży zbiór
# Podział na dane treningowe i testowe zostanie wykonany w dalszej części!
df <- bind_rows(df1, df2, df3)

# Wyświetlenie danych
head(df)
```

```
##                     date Temperature Humidity Light    CO2 HumidityRatio
## 1...1 2015-02-04 17:51:00       23.18  27.2720 426.0 721.25   0.004792988
## 2...2 2015-02-04 17:51:59       23.15  27.2675 429.5 714.00   0.004783441
## 3...3 2015-02-04 17:53:00       23.15  27.2450 426.0 713.50   0.004779464
## 4...4 2015-02-04 17:54:00       23.15  27.2000 426.0 708.25   0.004771509
## 5...5 2015-02-04 17:55:00       23.10  27.2000 426.0 704.50   0.004756993
## 6...6 2015-02-04 17:55:59       23.10  27.2000 419.0 701.00   0.004756993
##       Occupancy
## 1...1         1
## 2...2         1
## 3...3         1
## 4...4         1
## 5...5         1
## 6...6         1
```

Zrozumienie danych i określenie celu projektu

Definicja problemu

Typ Problemu: Projekt dotyczy problemu Klasyfikacji Binarnej (Binary Classification).

Cel Projektu: Celem jest stworzenie stabilnego modelu predykcyjnego, który na podstawie pomiarów środowiskowych (CO2, światło, temperatura, wilgotność) w danym momencie jest w stanie z wysoką trafnością przewidzieć stan pomieszczenia: Occupancy = 1 (Zajęte) lub Occupancy = 0 (Wolne).

Zbudowany model może posłużyć w systemach Smart Building (Inteligentnych Budynków), gdzie przewidywanie zajętości służy do:
-Oszczędzania energii (np. automatyczne wyłączanie wentylacji/ogrzewania/światła, gdy nikogo nie ma).
-Zarządzania przestrzenią (np. monitorowanie wykorzystania sal konferencyjnych/lekcyjnych).

Zmienna Zależna (Target): Occupancy (Zajętość).
Occupancy (ang. Zajętość) - dotyczy stwierdzenia, czy w danym pomieszczeniu (np. biurze, sali, pokoju) znajduje się człowiek, czy jest ono puste.
Typ Zmiennej: Binarna (klasyfikacja 0/1))
Occupancy = 1: Pomieszczenie jest zajęte (jest w nim osoba lub grupa ludzi).
Occupancy = 0: Pomieszczenie jest wolne (jest puste).

Zmienne Wyjaśniające (Predyktory):
Zmienne predykcyjne, które mam do dyspozycji, to parametry środowiska wewnętrznego w określonym czasie.

Temperature (Temperatura) - Określa temperaturę, która została zmierzona w pomieszczeniu. Ciało ludzkie emituje ciepło, więc można wstępnie przypuszczać, że obecność jednej lub więcej osób może prowadzić do wzrostu temperatury w zamkniętym pomieszczeniu.

Humidity (Wilgotność) - Określa wilgotność, która została zmierzona w pomieszczeniu. Ludzie wydzielają parę wodną przez oddychanie i pocenie się. Może to wpływać na wzrost wilgotności w pomieszczeniu.

Light (Światło) - Określa poziom natężenie światła w pomieszczeniu. Osoba wchodząca do pomieszczenia często włącza światło. Jeśli pomieszczenie jest zajęte, poziom natężenia światła jest zazwyczaj wyższy niż w pustym.

CO2 (Dwutlenek Węgla) - Określa poziom dwutlenku węgla w pomieszczeniu. Ludzie wydychają CO2. Kiedy pomieszczenie jest zajęte, poziom CO2 szybko rośnie. Kiedy jest puste, poziom CO2 spada lub utrzymuje się na stałym poziomie (zależnie od wentylacji).

HumidityRatio (Stosunek Wilgotności) - Jest to miara wilgotności, skorygowana o temperaturę i ciśnienie, ale jej wartość zależy pośrednio od wilgotności i temperatury.

date - moment wykonania pomiaru powyższych parametrów środowiska. Składa się z dokładnej daty (rok-miesiąc-dzień) i godziny. Zajętość zależy od pory dnia (np. dzień roboczy, godziny pracy, noc), co można wykorzystać do tworzenia nowych cech (np. godzina, dzień tygodnia).

Uzasadnienie Wyboru Modeli

Z uwagi na to, że wylosowałam możliwość wyboru dowolnego modelu, a problem jest klasyfikacją, wybrałam dwa modele klasyfikacyjne, które różnią się mechanizmem działania, co pozwala na dogłębne porównanie i interpretację wyników.

Model 1: Regresja Logistyczna (Logistic Regression)
Jest to podstawowy model liniowy dla problemów klasyfikacji binarnej. Jest szybki i wysoce interpretowalny (umożliwia oszacowanie prawdopodobieństwa zajętości) i dobrze sprawdzi się jako model bazowy do porównania z bardziej złożonym modelem.

Z uwagi na to, że Regresja Logistyczna (model pierwotnie brany pod uwagę) okazała się niestabilna w walidacji krzyżowej (generowała ostrzeżenia o całkowitej separacji), co świadczy o zbyt silnej relacji liniowej i problemach z estymacją, zastąpiono ją modelem Drzewa Decyzyjnego (C&RT).
Uzasadnienie wyboru:
- Zgodność z charakterem problemu (Problem do rozwiązania jest binarną klasyfikacją (Occupancy: 0/1). Model C&RT jest w pełni przystosowany do tego typu zadań.).

- Interpretowalność modelu (pokazuje hierarchię decyzji, umożliwia analizę progów zmiennych, pozwala łatwo zrozumieć, które zmienne mają największe znaczenie dla obecności osób w pomieszczeniu.).
- Dobrze radzi sobie przy zależnościach nieliniowych.
- Odporność na braki standaryzacji i skalowania.
- Możliwość wizualizacji.
Wada:
Proces analizuje jedną zmienną na raz, przez co nie ma możliwości uchwycenia interakcji między zmiennymi.

Model 2: Las Losowy (Random Forest)
Złożony, nieliniowy model, Wybrałam go jako drugi model, będący rozszerzeniem C&RT, ponieważ zapewnia większą stabilność, mniejsze ryzyko przeuczenia oraz lepszą jakość predykcji.
Uzasadnienie wyboru:
- Eliminuje wady pojedynczego drzewa (bagging - z danych treningowych losuje się wiele próbek z powtórzeniami, dla każdej próbki buduje osobne drzewo decyzyjne; losowy wybór cech do podziału w każdym węźle).
Random Forest dostarcza dodatkowo informacji o ważności zmiennych, co poszerza analizę interpretacyjną.
Podobnie jak C&RT, Random Forest nie wymaga standaryzacji, jest odporny na obecność zmiennych w różnych skalach.

Oba modele są zgodne z charakterem problemu (klasyfikacja binarna) i umożliwiają przeprowadzenie pełnej analizy według standardu CRISP-DM.

Sprawdzenie Struktury Danych

```
# Sprawdzenie wymiarów danych
dim(df)
```

```
## [1] 20560     7
```

```
# Sprawdzenie struktury danych
str(df)
```

```
## 'data.frame':    20560 obs. of  7 variables:
##  $ date         : chr  "2015-02-04 17:51:00" "2015-02-04 17:51:59" "2015-02-04 17:53:00" "2015-02-04
##  $ Temperature  : num  23.2 23.1 23.1 23.1 23.1 ...
##  $ Humidity     : num  27.3 27.3 27.2 27.2 27.2 ...
##  $ Light        : num  426 430 426 426 426 ...
##  $ CO2          : num  721 714 714 708 704 ...
##  $ HumidityRatio: num  0.00479 0.00478 0.00478 0.00477 0.00476 ...
##  $ Occupancy    : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
# Sprawdzenie braków danych
sum(is.na(df))
```

```
## [1] 0
```

```
sum(!complete.cases(df))
```

```
## [1] 0
```

```r
# Sprawdzenie podstawowych statystyk dla zmiennych
# Wykrywanie wartości błędnych, ekstremalnych i odstających
summary(df)
```

```
##      date              Temperature       Humidity         Light
##  Length:20560       Min.   :19.00   Min.   :16.75   Min.   :   0.0
##  Class :character   1st Qu.:20.20   1st Qu.:24.50   1st Qu.:   0.0
##  Mode  :character   Median :20.70   Median :27.29   Median :   0.0
##                     Mean   :20.91   Mean   :27.66   Mean   : 130.8
##                     3rd Qu.:21.52   3rd Qu.:31.29   3rd Qu.: 301.0
##                     Max.   :24.41   Max.   :39.50   Max.   :1697.2
##       CO2          HumidityRatio        Occupancy
##  Min.   : 412.8   Min.   :0.002674   Min.   :0.000
##  1st Qu.: 460.0   1st Qu.:0.003719   1st Qu.:0.000
##  Median : 565.4   Median :0.004292   Median :0.000
##  Mean   : 690.6   Mean   :0.004228   Mean   :0.231
##  3rd Qu.: 804.7   3rd Qu.:0.004832   3rd Qu.:0.000
##  Max.   :2076.5   Max.   :0.006476   Max.   :1.000
```

Interpretacja Wstępna

Zbór danych jest bardzo duży, składa się z ponad 20 tys. obserwacji.
Obserwacje dotyczą 7 cech:
- date (dokładna data i godzina)
- Temperature (temperatura)
- Humidity (wilgotność)
- Light
- CO2
- HumidityRatio
- Occupancy

Typy Zmiennych:
Zmienna date jest typu character. Wymaga konwersji na format czasu przed analizą. W dalszej części zostaną z niej wydzielone dodatkowe zmienne: godzina i dzień tygodnia, ponieważ to one mogą mieć znaczenie dla dalszej analizy. Zmienna Occupancy jest numeryczna (0/1), musi zostać przekształcona na typ factor (kategoryczny) do klasyfikacji.
Pozostałe zmienne są typu numerycznego.

Braki Danych:
Suma brakujących wartości (NA) wynosi 0, co oznacza, że nie ma potrzeby ich imputowania.

Analiza statystyk opisowych

Temperature:
Zakres: od 19.00°C do 24.41°C
Zakres wartości jest w pełni realistyczny dla pomieszczenia biurowego.
Brak ekstremalnych wartości odstających — wszystkie mieszczą się w normalnych wartościach dla klimatu wewnętrznego (ok. 19–25°C).

Humidity:
Zakres: 16.75% – 39.50%
Wilgotność od ok. 17% do 40% jest realistyczna w budynkach ogrzewanych zimą.
Brak wartości nienaturalnych (np. <0% lub >100%).
Rozkład wygląda sensownie — brak wskazań odstających.

Light:
Zakres: 0 – 1697.2 lux
Bardzo wiele obserwacji ma 0 lux → prawdopodobnie pomiary z nocy, lub czujnik w ciemnym pomieszczeniu.
Wartości maksymalne ~1700 lux są możliwe przy: mocnym oświetleniu sztucznym, świetle dziennym przez okno.
Maksimum 1697 lux nie jest odstające — światło słoneczne może dawać >10 000 lux, więc to raczej nie jest anomalia.

CO2:
Zakres: 412.8 – 2076.5 ppm
400 ppm to poziom zbliżony do powietrza zewnętrznego → minimum realistyczne.
Wartości powyżej 1000 ppm oznaczają słabą wentylację, a maksimum ~2076 ppm jest możliwe w zamkniętych pomieszczeniach przy dużym zagęszczeniu osób.
2076 ppm to wysoka wartość, ale nadal fizycznie możliwa — nie odstaje nienaturalnie.

HumidityRatio:
Zakres: 0.002674 – 0.006476
Ta zmienna to stosunek pary wodnej do suchego powietrza (kg/kg).
Zakres w pełni realistyczny.
Wartości odpowiadają 17–40% wilgotności względnej (w zależności od temperatury).

Occupancy:
Zakres: 0 lub 1
Średnia: 0.231 → oznacza, że ok. 23% obserwacji ma Occupancy = 1.
Dane są silnie niezbalansowane (większość = 0).
Może to świadczyć o tym, że większość czasu pomieszczenie jest puste.
Konieczność użycia metryk odpornych na niezbalansowanie (AUC-ROC, Czułość, Kappa).

Wnioski:
Analiza nie wykazała braków danych (0 wartości NA).
Brak anomalii i wartości odstających: Zakresy wszystkich zmiennych są realistyczne i odpowiadają rzeczywistym pomiarom środowiskowym.
Jedyną cechą wymagającą interpretacji jest duża liczba wartości Light = 0, co prawdopodobnie wynika z pomiarów w nocy lub w ciemnym pomieszczeniu.
Maksimum CO2 (~2076 ppm) oraz Light (~1700 lux) jest wysokie, ale fizycznie możliwe i nie stanowi anomalii.
Zmienna Occupancy jest silnie niezbalansowana (23% klasy 1), co może wymagać odpowiedniego podejścia w modelowaniu klasyfikacyjnym.

Inżynieria Cech (Feature Engineering)

```r
# Przekształcenie typów zmiennych, utworzenie nowych zmiennych
df <- df %>%
  mutate(date = ymd_hms(date)) %>%  # Przekształcenie zmiennej 'date' na format daty
  # Ujednolicenie i zaokrąglanie pomiarów (korygowanie sekundy 59 do następnej minuty)
  mutate(date = if_else(second(date) == 59, ceiling_date(date, unit = "minute"), date)) %>%
  mutate(
    Hour = hour(date),
    Minute = minute(date),
    Time = paste0(str_pad(Hour, 2, pad = "0"), ":",
                            # Czas wykonania pomiaru godzina:minuta
                             str_pad(Minute, 2, pad = "0")) %>% as.factor(),
    DayOfWeek = wday(date, label = TRUE, abbr = TRUE) # Dzień Tygodnia
  )
```

```r
# Przekształcenie zmiennej docelowej i zmiennych Hour i Time na Factor (Klasyfikacja)
df <- df %>%
  mutate(
    Occupancy = factor(Occupancy, levels = c(0, 1), labels = c("Wolne", "Zajete")),
    Hour = as.factor(Hour),
    Time = as.factor(Time))

# Ostateczne selekcje kolumn
df <- df %>%
  select(-date, -Minute, -Time)


# Struktura danych po Inżynierii Cech
str(df)
```

```
## 'data.frame':    20560 obs. of  8 variables:
##  $ Temperature  : num  23.2 23.1 23.1 23.1 23.1 ...
##  $ Humidity     : num  27.3 27.3 27.2 27.2 27.2 ...
##  $ Light        : num  426 430 426 426 426 ...
##  $ CO2          : num  721 714 714 708 704 ...
##  $ HumidityRatio: num  0.00479 0.00478 0.00478 0.00477 0.00476 ...
##  $ Occupancy    : Factor w/ 2 levels "Wolne","Zajete": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Hour         : Factor w/ 24 levels "0","1","2","3",..: 18 18 18 18 18 18 18 18 18 19 ...
##  $ DayOfWeek    : Ord.factor w/ 7 levels "niedz\\."<"pon\\."<..: 4 4 4 4 4 4 4 4 4 4 ...
```

```r
head(df)
```

```
##       Temperature Humidity Light    CO2 HumidityRatio Occupancy Hour DayOfWeek
## 1...1       23.18  27.2720 426.0 721.25   0.004792988    Zajete   17     śr\\.
## 2...2       23.15  27.2675 429.5 714.00   0.004783441    Zajete   17     śr\\.
## 3...3       23.15  27.2450 426.0 713.50   0.004779464    Zajete   17     śr\\.
## 4...4       23.15  27.2000 426.0 708.25   0.004771509    Zajete   17     śr\\.
## 5...5       23.10  27.2000 426.0 704.50   0.004756993    Zajete   17     śr\\.
## 6...6       23.10  27.2000 419.0 701.00   0.004756993    Zajete   17     śr\\.
```

Przekształcono zmienną 'date' na format daty. Następnie ujednolicono dane, ponieważ niektóre pomiary były wykonane o godzinie hh:mm:59 co dawało dwa pomiary wykonane w tej samej minucie. Po przeprowadzeniu analizy stwierdzono, że były to pomiary, które powinny być przypisane do następnej minuty, ponieważ odczyty były wykonywane co minutę.
W kolejnym kroku wyodrębniono kolumny Hour - z godziną i Minute - z minutą, a następnie połączono je w zmienną Time, która określa dokładny czas wykonania pomiaru. Dodano tę kolumny łączącą godzinę i minutę licząc na poprawę jakości predykcji.
Utworzono również nową zmienną DayOfWeek, która określa dzień tygodnia pomiaru.
Utworzone zmienne kategoryczne Hour i DayOfWeek, są kluczowe dla uchwycenia cykliczności zjawiska.

Zaplanowano użycia predyktora Hour (mniejsza liczba kategorii) w modelu Regresji Logistycznej, a predyktora Time (dużej liczby kategorii) w modelu Random Forest. Złożony model nieliniowy potrafi wykorzystać bardziej szczegółową informację czasową (godzina + minuty), podczas gdy model liniowy musi opierać się na ogólniejszej informacji (tylko godzina).

Przekształcono zmienną docelową - Occupacy i zmienne Hour i Time na typ factor.

Dokonano ostatecznej selekcji kolumn. Usunięto ze zbioru zmienną date (zbędne informacje dotyczące roku, miesiąca i dnia) oraz Minute (informacje będzie zawarta w zmiennej Time).

Po analizie, przeprowadzonej w dalszej części projektu, stwierdzono, że zmienna Time także jest zbędna, z powodu zbyt dużej liczby kategorii i ją również usunięto ze zbioru danych.

Eksploracyjna Analiza Danych (EDA)

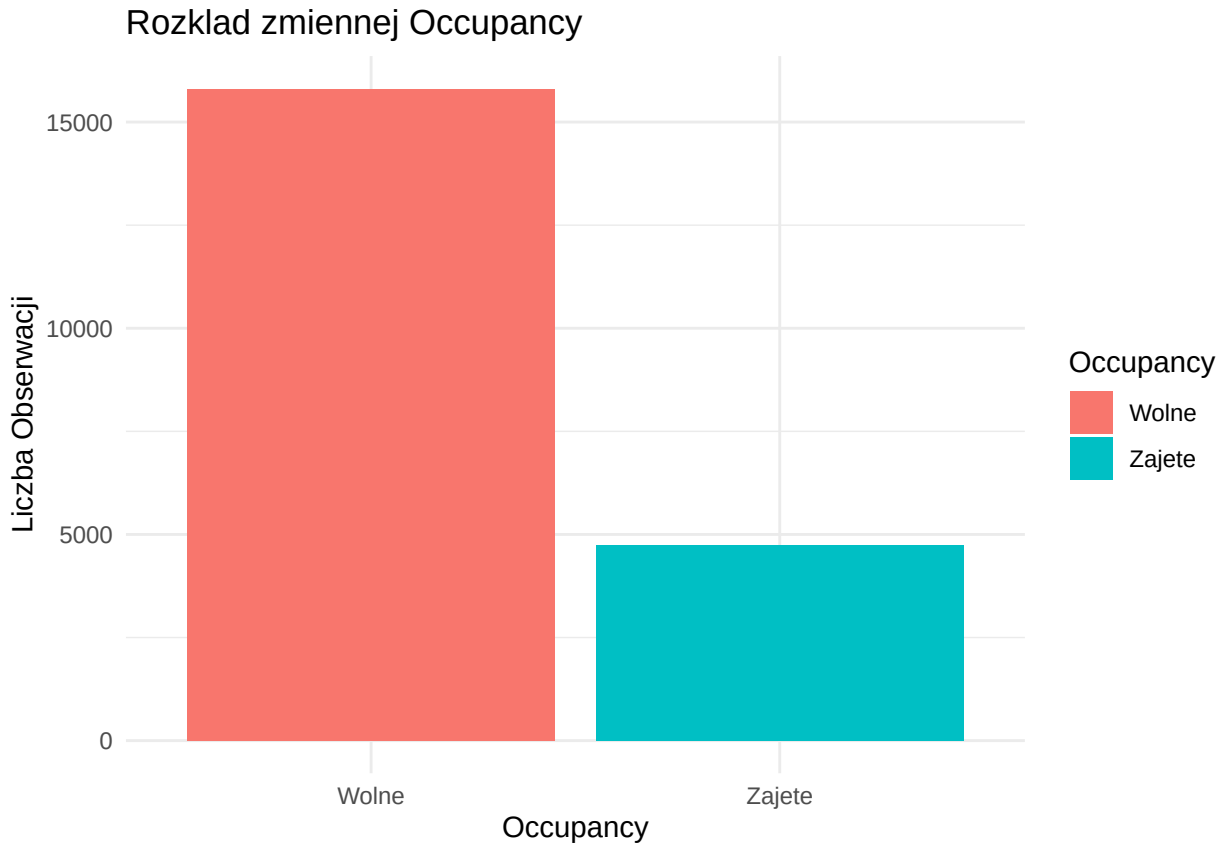
```
# Sprawdzenie rozkładu klas w zmiennej docelowej - Occupancy
table(df$Occupancy)
```

```
##
##  Wolne Zajete
##  15810   4750
```

```
prop.table(table(df$Occupancy))
```

```
##
##     Wolne    Zajete
## 0.7689689 0.2310311
```

```
# Wizualizacja rozkładu klas
hist_occupancy <- ggplot(df, aes(x = Occupancy, fill = Occupancy)) +
  geom_bar() +
  labs(title = "Rozkład zmiennej Occupancy", y = "Liczba Obserwacji") +
  theme_minimal()
print(hist_occupancy)
```



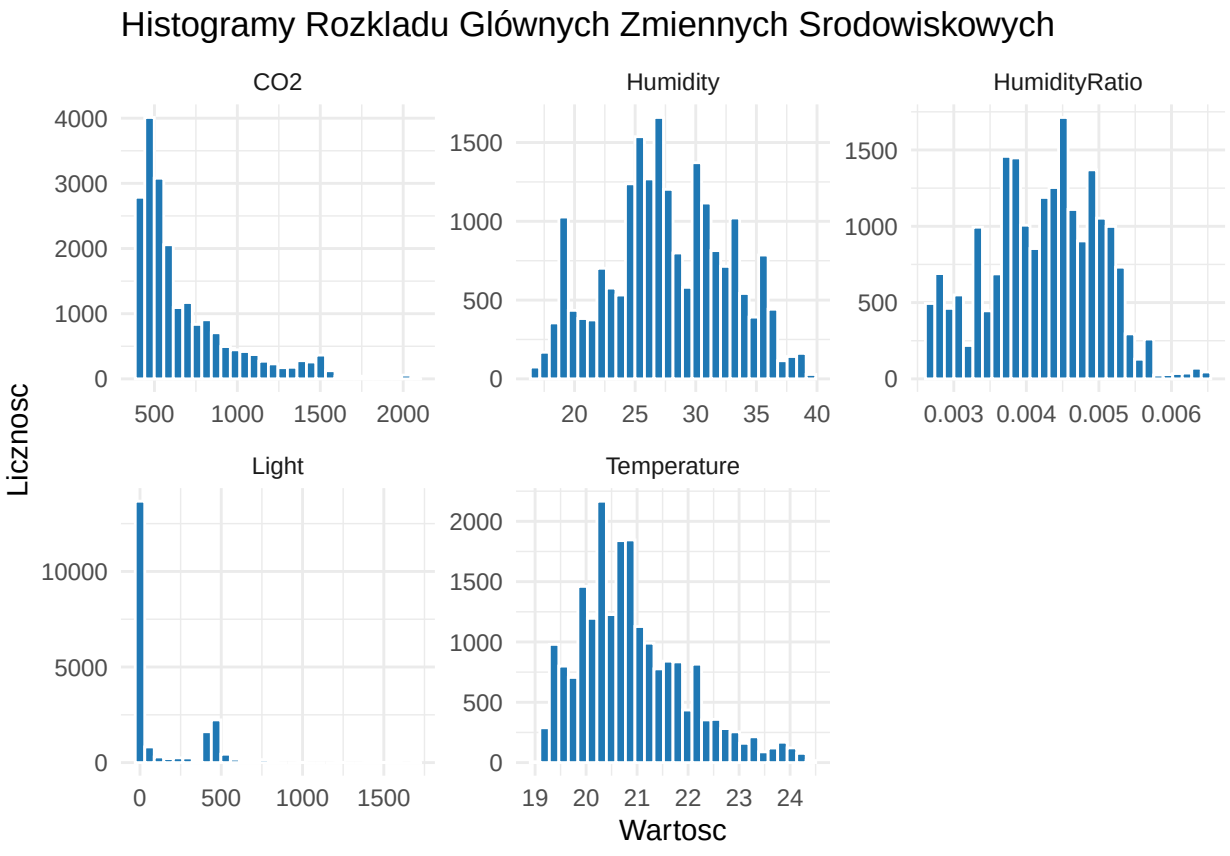
Analiza rozkładu klas

Przeprowadzona analiza potwierdziła, że klasy są niezbalansowane (77% obserwacji to klasa "Wolne", a 23% to klasa "Zajęte"). Wpłynie to na wybór miar oceny modelu. Konieczne będzie użycie metryk, takich jak AUC-ROC i Kappa.

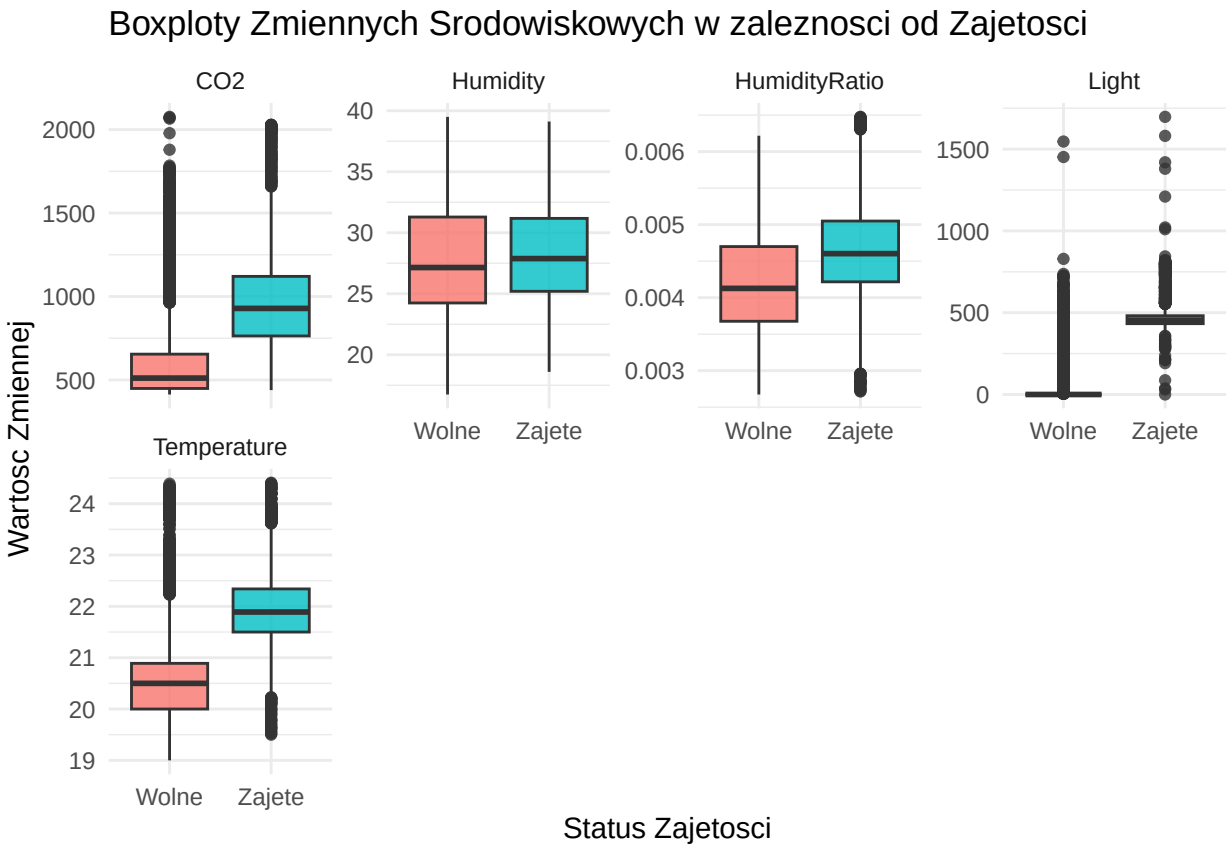
Analiza istotności zmiennych i ich użyteczności w modelu

Wizualizacje

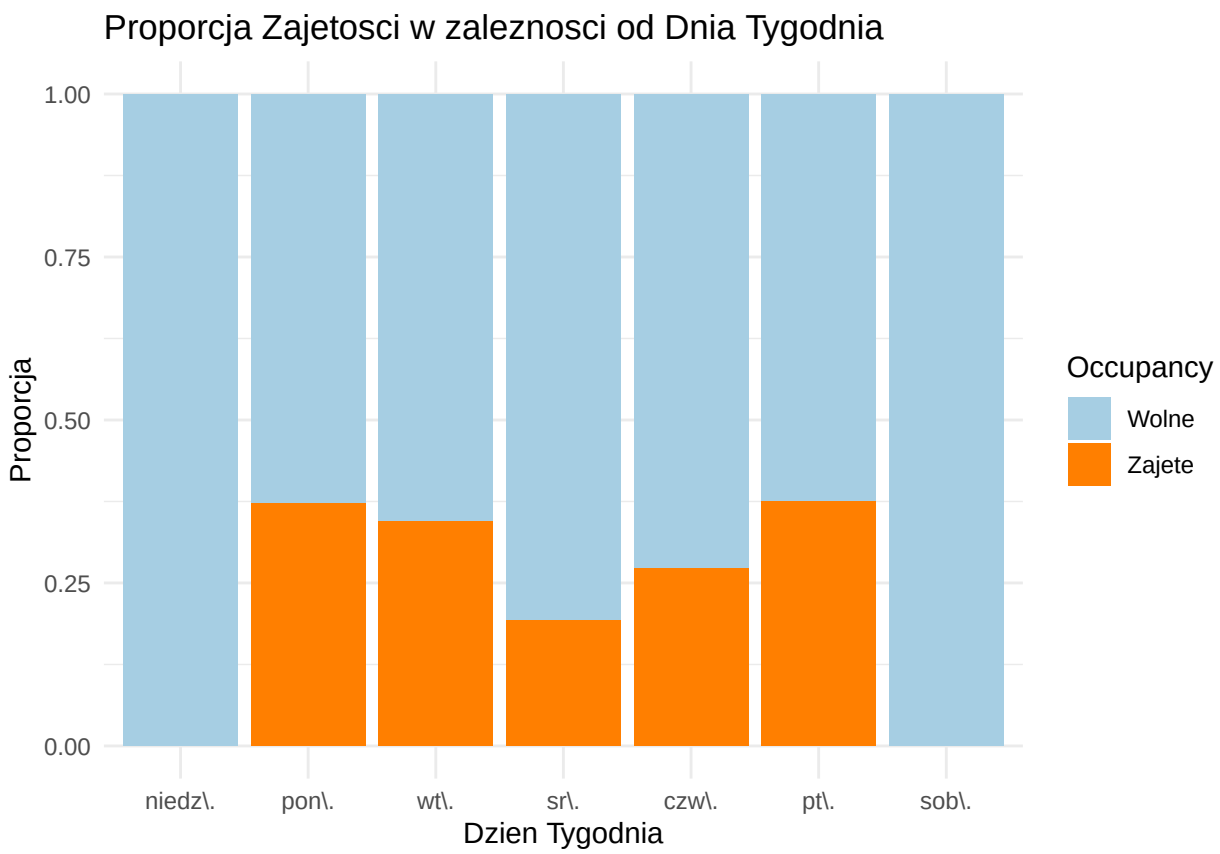
```
# Analiza zmiennych numerycznych: Temperature, Humidity, Light, CO2, HumidityRatio

# Histogramy
hist_all <- df %>%
  select(Temperature, Humidity, Light, CO2, HumidityRatio) %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value") %>%
  ggplot(aes(x = Value)) +
  geom_histogram(bins = 30, fill = "#1F78B4", color = "white") +
  facet_wrap(~ Variable, scales = "free") +
  labs(title = "Histogramy Rozkładu Głównych Zmiennych Środowiskowych",
       x = "Wartość", y = "Liczność") +
  theme_minimal()

print(hist_all)
```



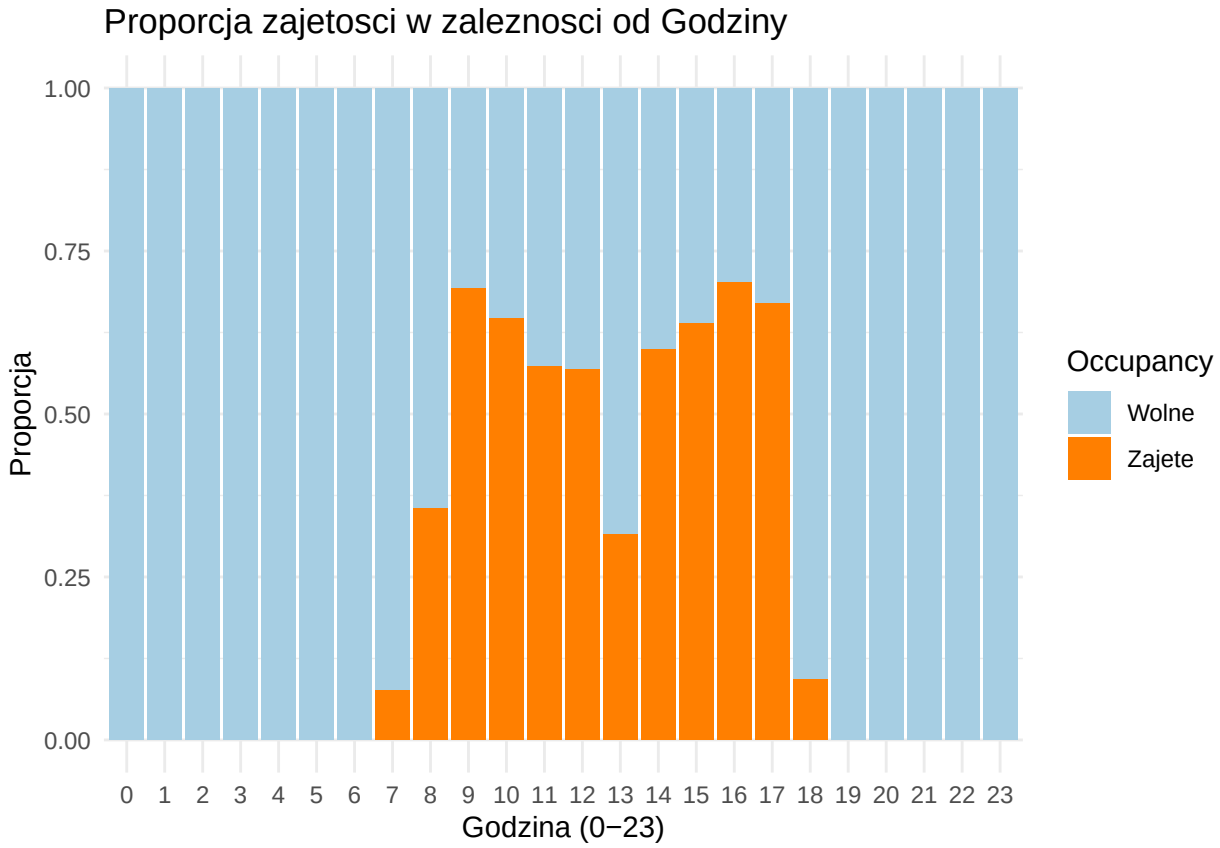
```r
# Boxploty
boxplot_occupancy <- df %>%
  select(Occupancy, Temperature, Humidity, Light, CO2, HumidityRatio) %>%
  pivot_longer(cols = c(Temperature, Humidity, Light, CO2, HumidityRatio),
               names_to = "Variable", values_to = "Value") %>%
  ggplot(aes(x = Occupancy, y = Value, fill = Occupancy)) +
  geom_boxplot(alpha = 0.8) +
  facet_wrap(~ Variable, scales = "free_y", ncol = 4) +
  labs(title = "Boxploty Zmiennych Środowiskowych w zależności od Zajętości",
       x = "Status Zajętości", y = "Wartość Zmiennej") +
  theme_minimal() +
  theme(legend.position = "none")

print(boxplot_occupancy)
```




```r
# Wykres Słupkowy - wizualizacja proporcji zajętości w zależności od dnia tygodnia
barplot_dayofweek <- ggplot(df, aes(x = DayOfWeek, fill = Occupancy)) +
  geom_bar(position = "fill") +
  labs(title = "Proporcja Zajętości w zależności od Dnia Tygodnia",
       y = "Proporcja", x = "Dzień Tygodnia") +
  scale_fill_manual(values = c("Wolne" = "#A6CEE3", "Zajete" = "#FF7F00")) +
  theme_minimal()

print(barplot_dayofweek)
```




```r
# Zależność od Godziny
viz_hour <- ggplot(df, aes(x = Hour, fill = Occupancy)) +
  geom_bar(position = "fill") +
  labs(title = "Proporcja zajętości w zależności od Godziny",
       y = "Proporcja", x = "Godzina (0-23)") +
  scale_fill_manual(values = c("Wolne" = "#A6CEE3", "Zajete" = "#FF7F00")) +
  theme_minimal()

print(viz_hour)
```



Histogramy pozwoliły na zwizualizowanie rozkładu, skośności i wartości odstających (outliers) poszczególnych zmiennych w zbiorze danych.

Boxploty pokazały, że zmienne CO2, Light, Temperature wykazują największą separację między klasami "Wolne" i "Zajęte". Średnie i rozkłady tych zmiennych są wyraźnie wyższe, gdy Occupancy = "Zajęte", co wskazuje na to, że są to kluczowe predyktory.

Dzień tygodnia: Zajętość występuje tylko w dni robocze (pon-pt). W weekendy pomieszczenia są "Wolne". Zmienna ta powinna zostać uwzględniona w modelu.

Godzina: Zajętość występuje głównie w typowych godzinach pracy/nauki (ok. 8:00–18:00), co jest oczekiwane i pozwala na silną predykcję.

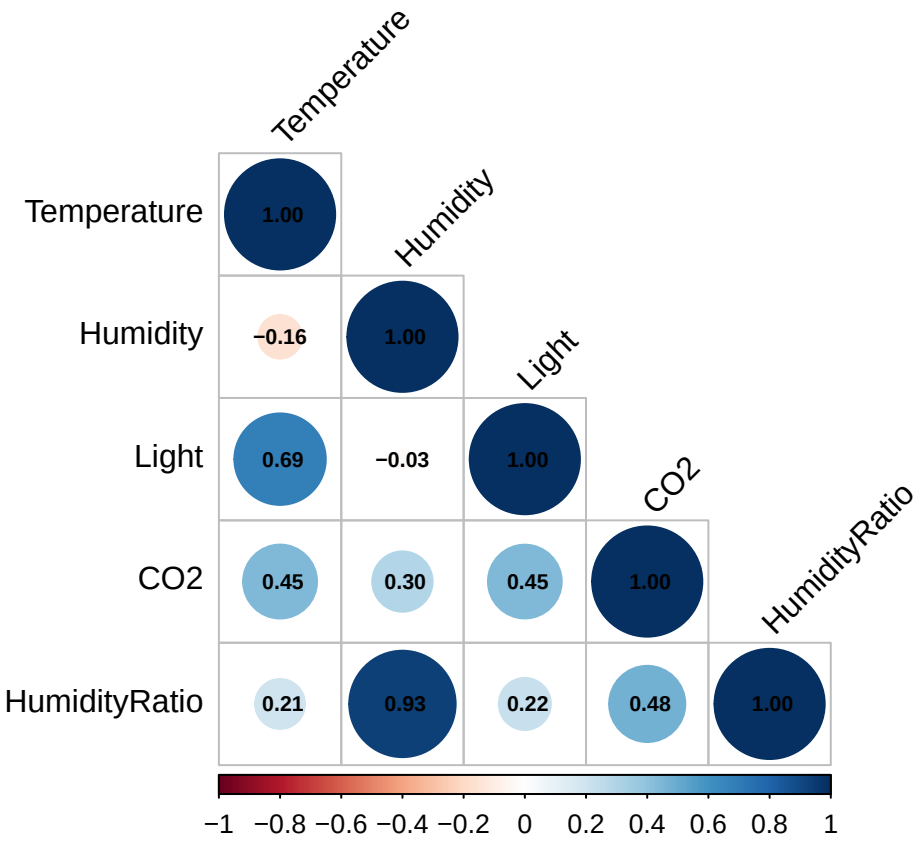
Korelacje

```
# Korelacje między zmiennymi numerycznymi
# Zmienne predykcyjne: Temperature, Humidity, Light, CO2
df_num <- df %>%
  select(Temperature, Humidity, Light, CO2, HumidityRatio)

# Macierz Korelacji
cor_matrix <- cor(df_num)
cor_matrix
```

```
##                Temperature    Humidity       Light       CO2 HumidityRatio
## Temperature      1.0000000 -0.15696350  0.68857106 0.4499887     0.2052796
```

```
## Humidity       -0.1569635  1.00000000 -0.02945884 0.2997463     0.9327237
## Light           0.6885711 -0.02945884  1.00000000 0.4481052     0.2233286
## CO2             0.4499887  0.29974628  0.44810517 1.0000000     0.4779647
## HumidityRatio   0.2052796  0.93272370  0.22332863 0.4779647     1.0000000
```

```r
# Wizualizacja Macierzy Korelacji
corrplot(cor_matrix, method = "circle", type = "lower", tl.col = "black", tl.srt = 45,
         addCoef.col = "black", number.cex = 0.7)
```




```r
# Usunięcie kolumny HumidityRatio - zbędne informacje
df <- df %>%
  select(-HumidityRatio)
```

Analiza Korelacji

Stwierdzono silną koliniowość między Humidity a HumidityRatio (0.93). W celu uproszczenia modeli i uniknięcia niestabilności (zwłaszcza w modelach liniowych), zmienna HumidityRatio została usunięta z dalszej analizy.

Umiarkowane Korelacje:
Temperature i Light (0.69) - Gdy rośnie natężenie światła (włączone), temperatura też nieznacznie wzrasta.
Temperature i CO2 (0.45) - Wzrost CO2 (obecność ludzi) prowadzi do wzrostu temperatury (emisja ciepła przez ciała).

Brak silnej korelacji pomiędzy większością zmiennych, jest korzystne dla modeli nieliniowych (CART, Random Forest).

Modelowanie

Podział zbioru danych na zbiory: treningowy, walidacyjny i testowy

```
# Ustawienie ziarna losowości dla powtarzalności wyników
set.seed(123)

# # Wyodrębnienie 4 obserwacji do ZBIORU TESTOWEGO
# test_index <- createDataPartition(df$Occupancy, p = 4 / nrow(df), list = FALSE)
# # Zbiór testowy - 4 obserwacje (do końcowej predykcji)
# df_test <- df[head(test_index, 4), ]
# # Zbiór pozostały (do podziału 70/30)
# remaining_data <- df[-test_index, ]

# # Wyodrębnienie 10 obserwacji do ZBIORU TESTOWEGO (50/50)
# df_test <- df %>%
#   group_by(Occupancy) %>%
#   # Losujemy 5 obserwacje z klasy "Wolne" i 5 z klasy "Zajete"
#   sample_n(5, replace = FALSE) %>%
#   ungroup()

# Wyodrębnienie 9 REPREZENTATYWNYCH obserwacji do ZBIORU TESTOWEGO (stosunek klas: 23/77)
# Losujemy 7 Wolne i 2 Zajęte
#(Prawidłowy podział reprezentatywny dla całego zbioru (23% Zajete / 77% Wolne))
df_test_wolne <- df %>% filter(Occupancy == "Wolne") %>% sample_n(7)
df_test_zajete <- df %>% filter(Occupancy == "Zajete") %>% sample_n(2)

df_test <- bind_rows(df_test_wolne, df_test_zajete) # Zbiór testowy - 9 obserwacje

# Zbiór pozostały (do podziału 70/30)
# Usuwamy z pierwotnego zbioru df wiersze, które trafiły do df_test
remaining_data <- anti_join(df, df_test)
```

```
## Joining with `by = join_by(Temperature, Humidity, Light, CO2, Occupancy, Hour,
## DayOfWeek)`
```

```
# Podział pozostałej części na ZIÓR TRENINGOWY (70%) i WALIDACYJNY (30%)
train_index <- createDataPartition(remaining_data$Occupancy, p = 0.70, list = FALSE)

# Zbiór treningowy (70% całości)
df_train <- remaining_data[train_index, ]

# Zbiór walidacyjny (30% całości)
df_val <- remaining_data[-train_index, ]

# Sprawdzenie ostatecznych rozmiarów podzbiorów
cat("Zbiór Uczący (df_train): ", nrow(df_train), "obserwacji,",
    round(nrow(df_train)/nrow(df)*100, 1), "%,\n")
```

```
## Zbiór Uczący (df_train):  14387 obserwacji, 70 %,
```

```
cat("Zbiór Walidacyjny (df_val): ", nrow(df_val), "obserwacji,",
    round(nrow(df_val)/nrow(df)*100,1), "%,\n")
```

```
## Zbiór Walidacyjny (df_val):  6164 obserwacji, 30 %,
```

```
cat("Zbiór Predykcyjny (df_test): ", nrow(df_test), "obserwacji.\n")
```

```
## Zbiór Predykcyjny (df_test):  9 obserwacji.
```

```
cat("\nReprezentatywność (Proporcje klasy 'Zajete'):\n")
```

```
##
## Reprezentatywność (Proporcje klasy 'Zajete'):
```

```
cat(paste("Cały Zbiór: ", round(prop.table(table(df$Occupancy))["Zajete"] * 100, 2), "%\n"))
```

```
## Cały Zbiór:  23.1 %
```

```
cat(paste("Treningowy: ", round(prop.table(table(df_train$Occupancy))["Zajete"] * 100, 2), "%\n"))
```

```
## Treningowy:  23.1 %
```

```
cat(paste("Walidacyjny: ", round(prop.table(table(df_val$Occupancy))["Zajete"] * 100, 2), "%\n"))
```

```
## Walidacyjny:  23.1 %
```

```
cat(paste("Testowy (9 obs.): ", round(prop.table(table(df_test$Occupancy))["Zajete"] * 100, 2), "%\n"))
```

```
## Testowy (9 obs.):  22.22 %
```

Ze zbioru danych wyodrębniono 3 podzbiory:
zbiór treningowy - 70% obserwacji (uczenie modelu),
zbiór walidacjny - 30% obserwacji (dobór hiperparametrów),
zbiór testowy do predykcji - 9 obserwacji (ostateczna ocena modelu).

Zadbano w tym kroku o:
reprezentatywność klas w każdym podzbiorze,
stabilność trenowania i walidacji,
możliwość końcowej, obiektywnej oceny modelu na małej, ale losowej próbie.

Na samym początku wyodrębniono zbiór testowy do predykcji. W założeniu projektu były to 4 obserwacje, jednak uznano, że jest to za mało, aby móc dokonać dokładnej oceny.
Następnie przetestowano pomysł wyodrębnienia 10 obserwacji (po 5 z każdej klasy), jednak po głębszym namyśle odrzucono ten pomysł z uwagi na zaburzenie stosunku klas w porównaniu do zbiorów treningowego i walidacyjnego.
Ostatecznie zdecydowano się na 9 losowych obserwacji, zachowując proporcje klas 23% Zajęte (2 obserwacje) i 77% Wolne (7 obserwacji), aby zachować reprezentatywność. To odzwierciedla rzeczywistą strukturę danych, dzięki czemu test końcowy nie będzie faworyzował żadnej z klas. Zbiór będzie niezależny od procesu uczenia i walidacji, aby finalna ocena modelu była wiarygodna. Aby zapobiec przeciekaniu informacji, wszystkie

rekordy użyte jako testowe zostały usunięte z reszty danych. To zapewnia, że: żadna obserwacja testowa nie pojawi się przypadkowo w zbiorze uczącym, wyniki na testowych danych są niezależne od procesu uczenia.

Pozostałe dane podzielono na na zbiór treningowy (70%) i walidacyjny (30%). Podział 70/30 daje dużo danych do trenowania modelu i wystarczająco duży zbiór walidacyjny do oceny jakości przed końcowym testem.
Użyto funkcji createDataPartition(), aby zachować taką samą proporcję klas w obu podzbiorach. Oba mają bardzo zbliżony procent klasy "Zajete" (ok. 23%), co zapewnia, że model trenuje i jest oceniany na realistycznych proporcjach.

Model 1: Regresja Logistyczna

```
#Model Regresji Logistycznej
model_log <- glm(Occupancy ~ .,
                 data = df_train,
                 family = binomial)
```

```
## Warning: glm.fit: dopasowane prawdopodobieństwa numerycznie okazały się być 0
## lub 1
```

```
print("Podsumowanie Modelu Regresji Logistycznej:")
```

```
## [1] "Podsumowanie Modelu Regresji Logistycznej:"
```

```
print(summary(model_log))
```

```
##
## Call:
## glm(formula = Occupancy ~ ., family = binomial, data = df_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.403e+01  2.151e+05   0.000   0.9999
## Temperature -1.313e+00  1.573e-01  -8.344   <2e-16 ***
## Humidity     6.427e-02  3.798e-02   1.692   0.0907 .
## Light        1.963e-02  9.851e-04  19.923   <2e-16 ***
## CO2          1.658e-03  5.415e-04   3.062   0.0022 **
## Hour1       -4.895e-02  2.504e+03   0.000   1.0000
## Hour2       -4.136e-02  2.514e+03   0.000   1.0000
## Hour3       -4.040e-02  2.498e+03   0.000   1.0000
## Hour4        5.125e-02  2.479e+03   0.000   1.0000
## Hour5        3.322e-02  2.515e+03   0.000   1.0000
## Hour6        3.113e-02  2.520e+03   0.000   1.0000
## Hour7        1.633e+01  1.781e+03   0.009   0.9927
## Hour8        1.715e+01  1.781e+03   0.010   0.9923
## Hour9        1.885e+01  1.781e+03   0.011   0.9916
## Hour10       1.809e+01  1.781e+03   0.010   0.9919
## Hour11       1.828e+01  1.781e+03   0.010   0.9918
## Hour12       1.684e+01  1.781e+03   0.009   0.9925
## Hour13       1.561e+01  1.781e+03   0.009   0.9930
```

```
## Hour14       1.869e+01  1.781e+03   0.010   0.9916
## Hour15       2.036e+01  1.781e+03   0.011   0.9909
## Hour16       2.294e+01  1.781e+03   0.013   0.9897
## Hour17       1.885e+01  1.781e+03   0.011   0.9916
## Hour18       1.877e+01  1.781e+03   0.011   0.9916
## Hour19       4.358e-01  2.488e+03   0.000   0.9999
## Hour20       2.006e-01  2.475e+03   0.000   0.9999
## Hour21       1.857e-01  2.486e+03   0.000   0.9999
## Hour22       8.994e-02  2.476e+03   0.000   1.0000
## Hour23       2.035e-01  2.533e+03   0.000   0.9999
## DayOfWeek.L -3.879e+01  8.538e+05   0.000   1.0000
## DayOfWeek.Q -5.966e+01  8.215e+05   0.000   0.9999
## DayOfWeek.C -2.732e+01  6.148e+05   0.000   1.0000
## DayOfWeek^4 -2.657e+01  3.640e+05   0.000   0.9999
## DayOfWeek^5 -7.996e+00  1.643e+05   0.000   1.0000
## DayOfWeek^6 -3.609e+00  4.954e+04   0.000   0.9999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15553.3  on 14386  degrees of freedom
## Residual deviance:  1018.3  on 14353  degrees of freedom
## AIC: 1086.3
##
## Number of Fisher Scoring iterations: 21
```

Regresja logistyczna binarna to rodzaj analizy regresji, w której zmienna zależna - Occupancy jest zmienną zero-jedynkowa: zakodowaną jako 0 (wolny) lub 1 (zajęty).

Zastosowanie modelu regresji logistycznej pozwala określić zarówno siłę, jak i kierunek zależności między czynnikiem jakościowym (typu klasowego) lub ilościowym (typu dyskretnego lub ciągłego) a dychotomiczną zmienną objaśnianą.

Interpretacja Wyników:
Analiza kolumny Pr(>|z|) pokazała, że tylko zmienne Temperature, Light i CO2 mają p-value < 0.05 więc są statystycznie istotne. Pozostałe zmienne nie są istotne statystycznie. Model przeucza się i jest niestabilny (za dużo kategorii (24 dla Hour, 7 dla dni tygodnia), model próbuje dopasować za dużo parametrów.
Zdecydowano się zbudować kolejny model logistyczny, tym razem bez uwzględnienia zmiennych Hour i DayofWeek w modelu.

```
# Model Regresji Logistycznej
model_log <- glm(Occupancy ~ Temperature + Humidity + Light + CO2,
                 data = df_train,
                 family = binomial)
```

```
## Warning: glm.fit: algorytm nie zbiegł się
```

```
## Warning: glm.fit: dopasowane prawdopodobieństwa numerycznie okazały się być 0
## lub 1
```

```
print("Podsumowanie Modelu Regresji Logistycznej (bez zmiennych Hour i DayOfWeek):")
```

```
## [1] "Podsumowanie Modelu Regresji Logistycznej (bez zmiennych Hour i DayOfWeek):"
```

```r
print(summary(model_log))
```

```
##
## Call:
## glm(formula = Occupancy ~ Temperature + Humidity + Light + CO2,
##     family = binomial, data = df_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.8952327  1.8264476   3.228  0.00125 **
## Temperature -0.9129674  0.0898928 -10.156  < 2e-16 ***
## Humidity     0.1009157  0.0182309   5.535  3.1e-08 ***
## Light        0.0241679  0.0006904  35.005  < 2e-16 ***
## CO2          0.0037990  0.0003110  12.216  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15553.3  on 14386  degrees of freedom
## Residual deviance:  1545.6  on 14382  degrees of freedom
## AIC: 1555.6
##
## Number of Fisher Scoring iterations: 25
```

Interpretacja Wyników:
Istotność Predyktorów: Wszystkie wybrane zmienne (Temperature, Humidity, Light, CO2) są statystycznie istotne w przewidywaniu Occupancy ($p < 0.05$).
Model bez godzin i dni działa bardzo dobrze, bo warunki fizyczne (światło i CO2) niemal jednoznacznie odróżniają pokój zajęty od wolnego.

Wartości Estimate są kluczowe, ponieważ wskazują, jak zmiana o 1 jednostkę danej zmiennej wpływa na logarytm szans (log-odds) na Occupancy:
Temperature, -0.913, kierunek wpływu: negatywny - wzrost temperatury o 1 stopień zmniejsza szansę na zajętość. (Wbrew intuicji, ale w praktyce ludzie często wychodzą, gdy robi się za gorąco, lub pomiary mogły być pobrane z pomieszczenia, gdzie jest włączona klimatyzacja, gdy nikogo nie ma).
Humidity, +0.101, kierunek wpływu: pozytywny - wzrost wilgotności o 1 jednostkę zwiększa szansę na zajętość. (Zgodne z intuicją: ludzie wydychają parę wodną).
Light, +0.024, kierunek wpływu: pozytywny - wzrost światła o 1 lux zwiększa szansę na zajętość. (Zgodne z intuicją: zapalone światło to obecność ludzi).
CO2, +0.004, kierunek wpływu: pozytywny - wzrost CO2 o 1 ppm zwiększa szansę na zajętość. (Zgodne z intuicją: ludzie wydychają CO2).

Ocena jakości modelu:
Null deviance (15553.3): Mierzy jakość modelu bazowego, który zawiera tylko intercept (stałą) i nie ma żadnych predyktorów.
Residual deviance (1545.6): Mierzy, jak dobrze dopasowany jest model w porównaniu do idealnego modelu (który miałby deviance = 0).
Wniosek: Ogromna różnica między Null deviance a Residual deviance oznacza, że model jest znacznie lepiej dopasowany niż model bazowy. Model bardzo skutecznie wyjaśnił większość zmienności w zajętości pomieszczenia. AIC (1555.6): Im niższa wartość AIC, tym lepszy model.

Regresja logistyczna wykazała ostrzeżenia związane z całkowitą separacją klas (glm.fit: algorytm nie zbiegł się), co oznacza, że predyktory (zwłaszcza Light i CO2) niemal perfekcyjnie odróżniają pomieszczenia zajęte

od wolnych. Nie jest to błąd modelu, lecz efekt struktury danych. Nie jest to błąd modelu, lecz efekt struktury danych.
Model nadal ma bardzo dobrą skuteczność predykcyjną, jednak współczynniki mogą być niestabilne. Dlatego do właściwej predykcji zastosowano modele nieliniowe (CART, Random Forest), które nie mają problemu separacji i zapewniają większą stabilność.

```
# Model Regresji Logistycznej z walidacją krzyżową

# ctrl <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)
#
# set.seed(123)
#
# model_log2 <- train(Occupancy ~ Temperature + Humidity + Light + CO2,
#                     data = df_train,
#                     method = "glm",
#                     family = "binomial",
#                     trControl = ctrl,
#                     metric = "ROC")
#
# print("Podsumowanie Modelu Regresji Logistycznej z walidacją krzyżową:")
# print(summary(model_log2))
```

Model zbudowany z zastosowaniem walidacji krzyżowej, jednak ostatecznie nie wykorzystany w projekcie.

Model 2: Drzewo Decyzyjne C&RT

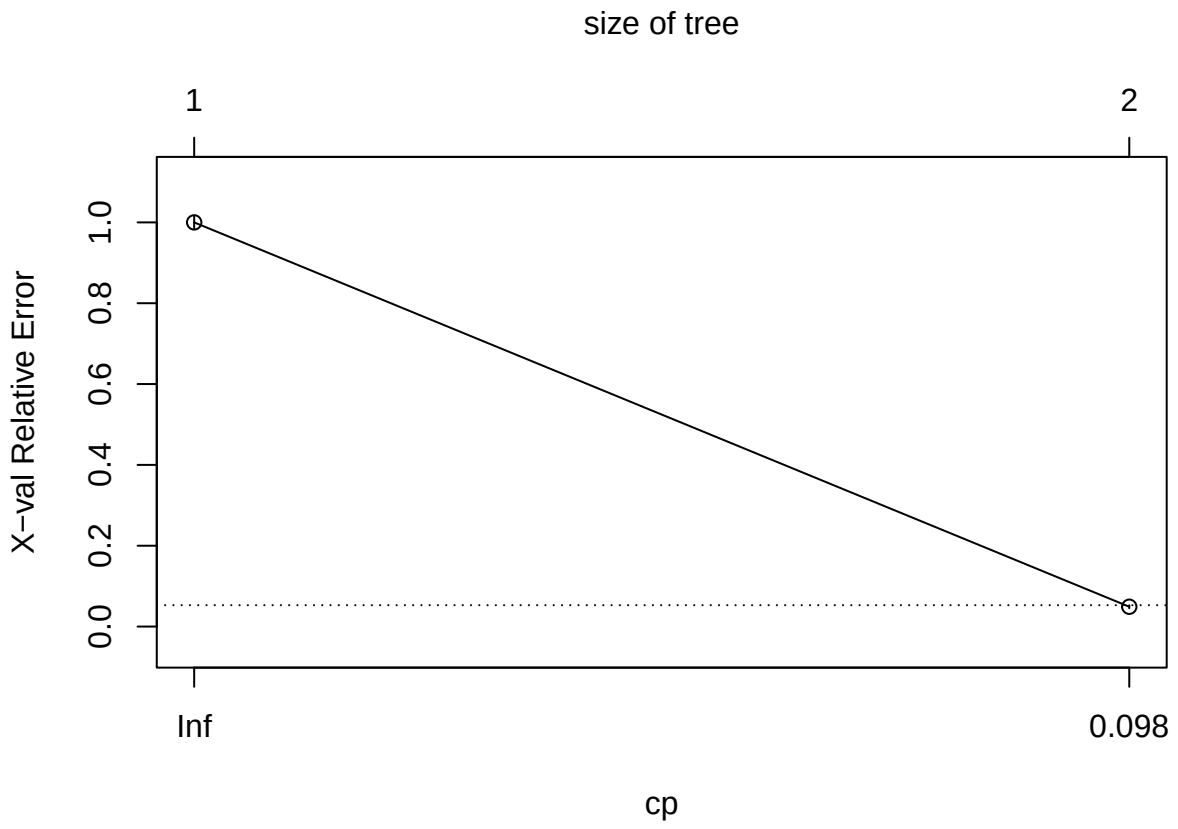
```
# Model Drzewo Decyzyjne C&RT

set.seed(123)

# Uczenie modelu na zbiorze treningowym
cart_model <- rpart(Occupancy ~ .,
                    data = df_train,
                    method = "class")

print("Podsumowanie Modelu Drzewo Decyzyjne C&RT:")
```

```
## [1] "Podsumowanie Modelu Drzewo Decyzyjne C&RT:"
```

```
print(cart_model)
```

```
## n= 14387
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 14387 3324 Wolne (0.768958087 0.231041913)
##   2) Light< 388.1667 10935   17 Wolne (0.998445359 0.001554641) *
##   3) Light>=388.1667 3452  145 Zajete (0.042004635 0.957995365) *
```

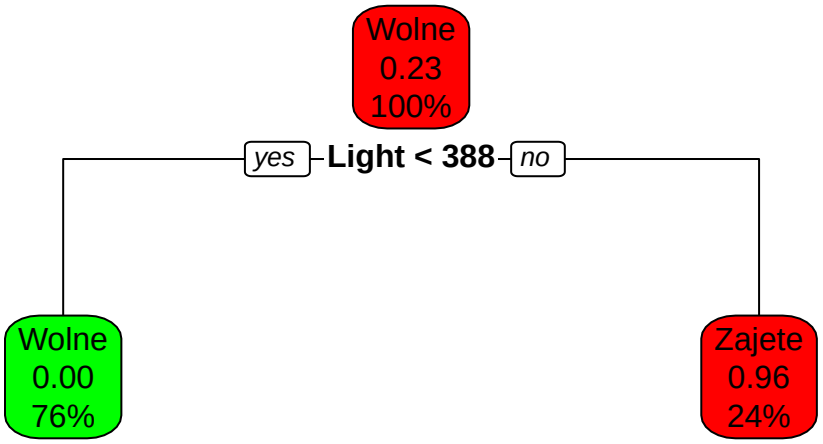
```r
# # Wizualizacja drzewa
# rpart.plot(cart_model, box.col=c("red", "green"))
# # Predykcja i ocena modelu na zbiorze testowym do walidacji
# cart_pred_final <- predict(cart_model, df_val, type = "class")
# # Macierz pomyłek
# confusionMatrix(cart_pred_final, df_val$Occupancy)

# Przycinanie drzewa, sprawdzenie czy nie zmieniły się wyniki klasyfikacji
plotcp(cart_model)
```




```r
printcp(cart_model)
```

```
##
## Classification tree:
## rpart(formula = Occupancy ~ ., data = df_train, method = "class")
##
## Variables actually used in tree construction:
## [1] Light
##
## Root node error: 3324/14387 = 0.23104
##
## n= 14387
##
##         CP nsplit rel error   xerror      xstd
## 1 0.95126      0  1.000000 1.000000 0.0152097
## 2 0.01000      1  0.048736 0.049037 0.0038191
```

```r
opt <- which.min(cart_model$cptable[,'xerror'])
cp <- cart_model$cptable[opt, 'CP']
pruned_cart <- prune(cart_model,cp)
rpart.plot(pruned_cart, box.col=c("red", "green"))
```




```r
# cart_pred_final2 <- predict(pruned_cart, df_val, type = "class")
# confusionMatrix(cart_pred_final2, df_val$Occupancy)

# # Predykcja i ocena modelu na zbiorze testowym do walidacji
# cart_pred_val <- predict(cart_model, df_val, type = "class")
# cat("\n--- Drzewo Decyzyjne C&RT ---\n")
# confusionMatrix(cart_pred_val, df_val$Occupancy)
#
# # Predykcja i ocena finalna na zbiorze testowym do predykcji
# cart_pred_final <- predict(cart_model, df_test, type = "class")
# cat("\n--- Zbiór Testowy (Finalny) ---\n")
# confusionMatrix(cart_pred_final, df_test$Occupancy)
# cat(sprintf("\nFINALNE ACCURACY: %.4f\n", mean(cart_pred_final == df_test$Occupancy)))
```

Interpretacja Modelu C&RT:
Uzyskane drzewo składa się wyłącznie z jednego podziału. Jedyną zmienną wykorzystaną do klasyfikacji była zmienna Light, reprezentująca poziom natężenia światła w pomieszczeniu. Model dokonuje klasyfikacji w następujący sposób:
Light $< 388.17 \rightarrow$ pomieszczenie klasyfikowane jako „Wolne”,
Light $>= 388.17 \rightarrow$ pomieszczenie klasyfikowane jako „Zajęte”.
Pozostałe predyktory (CO2, temperatura, wilgotność i wilgotność względna) okazały się niewykorzystywane w strukturze drzewa, co wskazuje na silną dominację zmiennej „Light” jako predyktora zajętości.

W celu poprawy generalizacji modelu i ograniczenia ryzyka nadmiernego dopasowania (overfittingu) zastosowano procedurę przycinania drzewa decyzyjnego. Wykorzystano wbudowaną tabelę kosztów złożoności (cptable) oraz wartość błędu krzyżowego (xerror), która wskazuje na jakość drzewa na kolejnych poziomach złożoności.
W analizowanym przypadku wykonanie pruningu nie zmieniło ani struktury drzewa, ani jakości klasyfikacji. Przeprowadzenie pruning potwierdziło, że drzewo C&RT ma wysoką stabilność, nie jest przeuczone mimo bardzo wysokiej dokładności, a jedynym kluczowym predyktorem okazała się zmienna Light.

```r
# # Model Drzewo Decyzyjne C&RT z walidacją krzyżową
# ctrl <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)
```

```
# set.seed(123)
#
# # Uczenie modelu na zbiorze treningowym
# cart_model <- train(Occupancy ~ .,
#                     data = df_train,
#                     method = "rpart",
#                     trControl = ctrl,
#                     metric = "ROC",
#                     tuneLength = 3)
#
# print("Podsumowanie Modelu Drzewo Decyzyjne C&RT z walidacją krzyżową:")
# print(cart_model)

# # Predykcja i ocena modelu na zbiorze testowym do walidacji
# cart_pred_val <- predict(cart_model, df_val, type = "class")
# cat("\n--- Drzewo Decyzyjne C&RT ---\n")
# confusionMatrix(cart_pred_val, df_val$Occupancy)
#
# # Predykcja i ocena finalna na zbiorze testowym do predykcji
# cart_pred_final <- predict(cart_model, df_test, type = "class")
# cat("\n--- Zbiór Testowy (Finalny) ---\n")
# confusionMatrix(cart_pred_final, df_test$Occupancy)
# cat(sprintf("\nFINALNE ACCURACY: %.4f\n", mean(cart_pred_final == df_test$Occupancy)))
```

Model zbudowany z zastosowaniem walidacji krzyżowej, jednak ostatecznie nie wykorzystany w projekcie.

Model 3: Random Forest

```
# Model Random Forest

set.seed(123)

# Uczenie modelu na zbiorze treningowym
rf_model <- randomForest(Occupancy ~ .,
                         data = df_train,
                         mtry = floor(sqrt(ncol(df_train) - 1)), #pierwiastek z liczby predyktorów
                         ntree = 500,
                         importance = TRUE)

print("Podsumowanie Modelu Random Forest:")
```

```
## [1] "Podsumowanie Modelu Random Forest:"
```

```
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Occupancy ~ ., data = df_train, mtry = floor(sqrt(ncol(df_train) -      1)),
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.61%
## Confusion matrix:
##        Wolne Zajete class.error
## Wolne  11007     56 0.005061918
## Zajete    32   3292 0.009626955
```

```
# Ważność Cech
cat("\nWażność Cech w Modelu Random Forest:\n")
```

```
##
## Ważność Cech w Modelu Random Forest:
```

```
importance_rf <- importance(rf_model, type = 2)
print(importance_rf[order(importance_rf[,1], decreasing = TRUE), ])
```

```
##       Light        Hour Temperature         CO2   DayOfWeek    Humidity
##  2878.78413   896.79017   586.07062   513.55410   137.89244    86.51548
```

```
# # Predykcja i ocena modelu na zbiorze do walidacji
# rf_pred_val <- predict(rf_model, df_val, type = "class")
# cat("\n--- Random Forest ---\n")
# confusionMatrix(rf_pred_val, df_val$Occupancy)
#
# # Predykcja i ocena finalna na zbiorze testowym do predykcji
# rf_pred_final <- predict(rf_model, df_test, type = "class")
# cat("\n--- Zbiór Testowy (Finalny) ---\n")
# confusionMatrix(rf_pred_final, df_test$Occupancy)
# cat(sprintf("\nFINALNE ACCURACY: %.4f\n", mean(rf_pred_final == df_test$Occupancy)))
```

Uwaga: z racji za dużej ilości kategorii w zmiennej Time, została ona usunięta ze zbioru danych (zmiany dokonano na początku kodu).

Model Random Forest został wytrenowany na zbiorze treningowym z wykorzystaniem 500 drzew decyzyjnych, losowo wybierając 2 zmienne przy każdym podziale (mtry = pierwiastek z liczby predyktorów).

Trafność modelu — błąd OOB (Out-of-Bag):
Random Forest osiągnął bardzo wysoką skuteczność klasyfikacji, uzyskując błąd OOB na poziomie 0,61%, co odpowiada dokładności około 99,4%. OOB jest formą wewnętrznej walidacji na danych OOB, więc wynik jest wiarygodnym odzwierciedleniem jakości modelu. Tak niska wartość błędu świadczy o bardzo wysokiej jakości klasyfikacji, dużej stabilności lasu losowego, braku przeuczenia (overfittingu).

Macierz pomyłek — skuteczność dla każdej klasy:
Macierz pomyłek pokazuje, że model prawidłowo klasyfikuje ponad 99% przypadków w obu klasach, mimo ich nierównomiernego rozkładu.

Ważność zmiennych (Variable Importance):
Najważniejszą cechą okazała się zmienna Light, a kolejne to Hour, Temperature i CO2. Zmienne DayOfWeek i Humidity miały najmniejszy wpływ na predykcję.

Wyniki wskazują, że Random Forest jest modelem o najwyższej jakości i stabilności, dobrze dopasowanym do charakteru danych i bardzo skutecznym w przewidywaniu zajętości pomieszczeń.

```
# # Model Random Forest z walidacją krzyżową
# ctrl <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)
# set.seed(123)
#
# rf_model2 <- train(Occupancy ~ .,
#                    data = df_train,
#                    method = "rf",
#                    trControl = ctrl,
#                    metric = "ROC",
#                    tuneLength = 3)
#
# print("Podsumowanie Modelu Random Forest z walidacją krzyżową:")
# print(rf_model2)
#
# # Ważność Cech
# cat("\nWażność Cech w Modelu Random Forest:\n")
# importance_rf <- importance(rf_model2$finalModel, type = 2)
# print(importance_rf[order(importance_rf[,1], decreasing = TRUE), ])


# # Predykcja i ocena modelu na zbiorze do walidacji
# rf_pred_val <- predict(rf_model2, df_val, type = "class")
# cat("\n--- Random Forest z walidacją krzyżową---\n")
# confusionMatrix(rf_pred_val, df_val$Occupancy)
#
# # Predykcja i ocena finalna na zbiorze testowym do predykcji
# rf_pred_final <- predict(rf_model2, df_test, type = "class")
# cat("\n--- Zbiór Testowy (Finalny) ---\n")
# confusionMatrix(rf_pred_final, df_test$Occupancy)
# cat(sprintf("\nFINALNE ACCURACY: %.4f\n", mean(rf_pred_final == df_test$Occupancy)))
```

Model zbudowany z zastosowaniem walidacji krzyżowej, jednak ostatecznie nie wykorzystany w projekcie.

Ocena Modeli

```
#_____________________________________________
# Funkcja Oceny
#_____________________________________________

evaluate_model_full <- function(model, test_data) {
  # Predykcja KLAS (type = "class" jest natywne dla rpart i randomForest)
  preds_class <- predict(model, newdata = test_data, type = "class")
  # Predykcja PRAWDOPODOBIEŃSTW (type = "prob") dla AUC-ROC
  preds_prob <- predict(model, newdata = test_data, type = "prob")[, "Zajete"]

  cm <- confusionMatrix(preds_class, test_data$Occupancy, positive = "Zajete")
  auc <- as.numeric(roc(response = test_data$Occupancy, predictor = preds_prob)$auc)

  return(list(
    Accuracy = cm$overall['Accuracy'],
    Sensitivity = cm$byClass['Sensitivity'],
```

```
    Specificity = cm$byClass['Specificity'],
    AUC_ROC = auc,
    Kappa = cm$overall['Kappa'],
    CM_Object = cm #Zwraca obiekt Macierzy Konfuzji
    ))
}

#_____________________________________________
# Pełna Ocena na Zbiorze Walidacyjnym (df_val)
#_____________________________________________

# Ocena C&RT (cart_model)
eval_cart_val <- evaluate_model_full(cart_model, df_val)
```

```
## Setting levels: control = Wolne, case = Zajete
```

```
## Setting direction: controls < cases
```

```
# Ocena Random Forest (rf_model2)
eval_rf_val <- evaluate_model_full(rf_model, df_val)
```

```
## Setting levels: control = Wolne, case = Zajete
## Setting direction: controls < cases
```

```
# WYŚWIETLANIE MACIERZY KONFUZJI
cat("\n### Macierz Konfuzji: Drzewo Decyzyjne (C&RT) na zbiorze walidacyjnym\n")
```

```
##
## ### Macierz Konfuzji: Drzewo Decyzyjne (C&RT) na zbiorze walidacyjnym
```

```
print(eval_cart_val$CM_Object)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Wolne Zajete
##     Wolne   4692      8
##     Zajete    48   1416
##
##                Accuracy : 0.9909
##                  95% CI : (0.9882, 0.9931)
##     No Information Rate : 0.769
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9747
##
##  Mcnemar's Test P-Value : 1.872e-07
##
##             Sensitivity : 0.9944
##             Specificity : 0.9899
```

```
##          Pos Pred Value : 0.9672
##          Neg Pred Value : 0.9983
##              Prevalence : 0.2310
##          Detection Rate : 0.2297
##    Detection Prevalence : 0.2375
##       Balanced Accuracy : 0.9921
##
##        'Positive' Class : Zajete
##
```

```
cat("\n### Macierz Konfuzji: Las Losowy (Random Forest) na zbiorze walidacyjnym\n")
```

```
##
## ### Macierz Konfuzji: Las Losowy (Random Forest) na zbiorze walidacyjnym
```

```
print(eval_rf_val$CM_Object)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Wolne Zajete
##     Wolne   4722     19
##     Zajete    18   1405
##
##                Accuracy : 0.994
##                  95% CI : (0.9917, 0.9958)
##     No Information Rate : 0.769
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9831
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9867
##             Specificity : 0.9962
##          Pos Pred Value : 0.9874
##          Neg Pred Value : 0.9960
##              Prevalence : 0.2310
##          Detection Rate : 0.2279
##    Detection Prevalence : 0.2309
##       Balanced Accuracy : 0.9914
##
##        'Positive' Class : Zajete
##
```

```
# Tabela Porównawcza Walidacyjna
results_val_final <- data.frame(
  Model = c("Drzewo Decyzyjne (C&RT)", "Las Losowy (Random Forest)"),
  Accuracy = c(eval_cart_val$Accuracy, eval_rf_val$Accuracy),
  Sensitivity = c(eval_cart_val$Sensitivity, eval_rf_val$Sensitivity),
  Specificity = c(eval_cart_val$Specificity, eval_rf_val$Specificity),
  AUC_ROC = c(eval_cart_val$AUC_ROC, eval_rf_val$AUC_ROC),
```

```
  Kappa = c(eval_cart_val$Kappa, eval_rf_val$Kappa)
)

cat("\n== Tabela Porównania Modeli na Zbiorze Walidacyjnym (PEŁNA OCENA) ==\n")
```

```
##
## == Tabela Porównania Modeli na Zbiorze Walidacyjnym (PEŁNA OCENA) ==
```

```
print(results_val_final)
```

```
##                         Model  Accuracy Sensitivity Specificity   AUC_ROC
## 1    Drzewo Decyzyjne (C&RT) 0.9909150   0.9943820   0.9898734 0.9921277
## 2 Las Losowy (Random Forest) 0.9939974   0.9866573   0.9962025 0.9991828
##       Kappa
## 1 0.9746787
## 2 0.9831013
```

```
# #_____________________________________________________________
# # Ocena Końcowa na Zbiorze Testowym (df_test) - TYLKO ACCURACY
# #_____________________________________________________________
#
# # Predykcja C&RT
# cart_preds_class_test <- predict(cart_model, newdata = df_test, type = "class")
# accuracy_cart_test <- mean(cart_preds_class_test == df_test$Occupancy)
#
# # Predykcja Random Forest
# rf_preds_class_test <- predict(rf_model, newdata = df_test, type = "class")
# accuracy_rf_test <- mean(rf_preds_class_test == df_test$Occupancy)
#
# cat(sprintf("\nDrzewo Decyzyjne (C&RT) - FINALNA ACCURACY (df_test): %.4f\n", accuracy_cart_test))
# cat(sprintf("Las Losowy (Random Forest) - FINALNA ACCURACY (df_test): %.4f\n", accuracy_rf_test))
```

Interpretacja wyników - porównanie modeli C&RT vs Random Forest

Każdy z modeli oceniono pod kątem jego poprawności oraz jakości uzyskanych modeli i predykcji, a następnie wykonano porównanie między modelami. Pełna ocena modeli (Accuracy, AUC-ROC, Czułość, Specyficzność, Kappa) została przeprowadzona wyłącznie na reprezentatywnym zbiorze walidacyjnym (30%).

Ocena skuteczności modelu została przeprowadzona przy użyciu standardowych miar jakości klasyfikacji. Macierz pomyłek prezentuje zestawienie wyników predykcji względem rzeczywistych klas.
Zawiera cztery typy wyników:
TP (True Positive) – model poprawnie zaklasyfikował przypadek jako Zajęte.
TN (True Negative) – model poprawnie zaklasyfikował przypadek jako Wolne.
FP (False Positive) – model błędnie oznaczył Wolne jako Zajęte.
FN (False Negative) – model błędnie oznaczył Zajęte jako Wolne.
Macierz ta jest kluczowa, ponieważ pozwala ocenić zarówno trafność klasyfikacji, jak i typy popełnianych błędów.

Oceniane parametry:
Accuracy (Dokładność): określa, jaka część obserwacji została sklasyfikowana poprawnie.
Sensitivity (Czułość): odsetek poprawnie wykrytych przypadków “Zajęte” (1). Wysoka czułość oznacza, że model bardzo rzadko błędnie oznacza zajęte pomieszczenia jako wolne.

Specificity (Specyficzność): Odsetek poprawnie wykrytych przypadków "Wolne" (0). Wysoka specyficzność oznacza, że model bardzo rzadko błędnie oznacza wolne pomieszczenia jako zajęte.
Kappa (Cohena): Wskaźnik zgodności modelu z rzeczywistością po uwzględnieniu zgodności przypadkowej.
AUC-ROC: Najlepsza miara do oceny zdolności modelu do rozróżniania klas. W przypadku niezbalansowanych danych jest bardziej miarodajny niż sama accuracy. Im bliżej 1, tym lepiej.

Drzewo Decyzyjne C&RT – interpretacja wyników walidacyjnych

Model C&RT osiągnął bardzo wysoką skuteczność na zbiorze walidacyjnym.
Accuracy = 0.9909 oznacza, że prawie 99,1% przypadków zostało sklasyfikowanych poprawnie.
Model bardzo dobrze rozpoznaje zarówno miejsca Wolne, jak i Zajęte:
Sensitivity (TPR) = 0.9944 – model poprawnie wykrywa 99,44% miejsc zajętych.
Specificity (TNR) = 0.9899 – model poprawnie identyfikuje 98,99% miejsc wolnych.
8 razy błędnie zaklasyfikował zajęte pomieszczenie jako wolne i 48 razy błędnie zaklasyfikował wolne pomieszczenie jako zajęto. Pokazuje to model ma wyższą tendencję to klasyfikowania pomieszczeń jako zajętych.
Współczynnik Kappa = 0.9747 potwierdza, że model przewyższa losowe klasyfikowanie i ma bardzo dobrą zgodność z rzeczywistością.

Wniosek:
Drzewo decyzyjne jest modelem bardzo dobrze dopasowanym. Mimo świetnych wyników wykazuje niewielką asymetrię błędów.

Las Losowy Random Forest – interpretacja wyników walidacyjnych

Model Random Forest osiągnął jeszcze lepsze wyniki niż drzewo pojedyncze.
Accuracy = 0.994 oznacza, że 99,4% obserwacji jest klasyfikowanych poprawnie.
Model jest wyjątkowo precyzyjny w identyfikacji obu klas:
Sensitivity = 0.9867 – poprawnie wykrywa 98,67% miejsc zajętych,
Specificity = 0.9962 – poprawnie wykrywa 99,62% miejsc wolnych.
Kappa = 0.9831 potwierdza bardzo wysoką zgodność modelu z rzeczywistymi danymi.
McNemar Test ma wartość p = 1, co oznacza, że błędy fałszywie dodatnie i fałszywie ujemne są symetryczne – model nie faworyzuje żadnej klasy.

Wniosek:
Random Forest działa bardzo stabilnie. Cechuje się minimalną asymetrią błędów, a wyniki są bardziej „zbalansowane" niż w przypadku drzewa.

Interpretacja porównawcza

Accuracy - Oba modele osiągają bardzo wysoką ogólną poprawność.
Sensitivity (Czułość) - Random Forest jest znacznie lepszy w poprawnym wykrywaniu przypadków "Zajęte" (klasa pozytywna), co jest krytyczne dla wdrożenia.
AUC-ROC - Random Forest osiąga wartość bliską 1.0, co świadczy o niemal perfekcyjnej zdolności do rozróżniania klas.
Kappa - Wartość Kappa dla Random Forest (> 0.9) wskazuje na niemal idealną zgodność klasyfikacji.
Dodatkowo Random Forest lepiej równoważy błędy pomiędzy klasami (McNemar p=1).
Model Random Forest jest zdecydowanie najlepszy pod kątem wszystkich miar jakości. Jest stabilny i skuteczny w identyfikacji klasy mniejszościowej ("Zajęte").

Predykcja dla 9 reprezentatywnych obserwacji

```
# #_____________________________________________________________
# # Ocena Końcowa na Zbiorze Testowym (df_test) - TYLKO ACCURACY
# #_____________________________________________________________

# Predykcja dla 9 Obserwacji
df_test_no_target <- df_test %>% select(-Occupancy)

# Predykcja C&RT
pred_cart_9_prob <- predict(cart_model, newdata = df_test_no_target, type = "prob")[, "Zajete"]
class_cart_9 <- predict(cart_model, newdata = df_test_no_target, type = "class")
accuracy_cart_test <- mean(class_cart_9 == df_test$Occupancy)

# Predykcja Random Forest
pred_rf_9_prob <- predict(rf_model, newdata = df_test_no_target, type = "prob")[, "Zajete"]
class_rf_9 <- predict(rf_model, newdata = df_test_no_target, type = "class")
accuracy_rf_test <- mean(class_rf_9 == df_test$Occupancy)

cat(sprintf("\nDrzewo Decyzyjne (C&RT) - FINALNA ACCURACY (df_test): %.4f\n", accuracy_cart_test))
```

```
##
## Drzewo Decyzyjne (C&RT) - FINALNA ACCURACY (df_test): 0.8889
```

```
cat(sprintf("Las Losowy (Random Forest) - FINALNA ACCURACY (df_test): %.4f\n", accuracy_rf_test))
```

```
## Las Losowy (Random Forest) - FINALNA ACCURACY (df_test): 0.8889
```

```
# Zestawienie wyników
results_9_obs <- data.frame(
  Obserwacja = 1:9,
  Prawdziwa_Klasa = df_test$Occupancy,
  CART_Predykcja = class_cart_9,
  CART_Prawdopodobienstwo = round(pred_cart_9_prob, 9),
  RF_Predykcja = class_rf_9,
  RF_Prawdopodobienstwo = round(pred_rf_9_prob, 9)
)

cat("Predykcje dla 9 obserwacji na zbiorze testowym:")
```

```
## Predykcje dla 9 obserwacji na zbiorze testowym:
```

```
print(results_9_obs)
```

```
##      Obserwacja Prawdziwa_Klasa CART_Predykcja CART_Prawdopodobienstwo
## 3604          1           Wolne          Wolne              0.001554641
## 3652          2           Wolne          Wolne              0.001554641
## 3264          3           Wolne          Wolne              0.001554641
## 825           4           Wolne          Wolne              0.001554641
## 5328          5           Wolne          Wolne              0.001554641
```

```
## 4127          6           Wolne          Wolne             0.001554641
## 2983          7           Wolne          Wolne             0.001554641
## 6655          8          Zajete          Wolne             0.001554641
## 2749          9          Zajete         Zajete             0.957995365
##      RF_Predykcja RF_Prawdopodobienstwo
## 3604        Wolne                 0.000
## 3652        Wolne                 0.000
## 3264        Wolne                 0.000
## 825         Wolne                 0.000
## 5328        Wolne                 0.000
## 4127        Wolne                 0.000
## 2983        Wolne                 0.000
## 6655        Wolne                 0.208
## 2749       Zajete                 1.000
```

Zdecydowano się zastosować większy zbiór testowy niż 4 obserwacje, w celu trafniejszej oceny utworzonych modeli. Zadbano, aby zbiór testowy był reprezentatywny (proporcje klas były takie same jak w zbiorach treningowym i walidacyjnym). Porównano: prawdziwą klasę, predykcję CART i prawdopodobieństwo nadane przez CART, predykcję Random Forest i prawdopodobieństwo RF.

Finall Accuracy wyniosło 0.89. Oznacza to, że 8 na 9 przypadków zostało poprawnie zaklasyfikowane.
W większości analizowanych obserwacji oba modele (CART i Random Forest) poprawnie klasyfikują miejsca jako „Wolne" lub „Zajęte", wykazując przy tym bardzo wysoką pewność predykcji. Zarówno CART, jak i Random Forest niezwykle konsekwentnie identyfikują klasę „Wolne" – CART przypisuje jej prawdopodobieństwa rzędu 0.0015 (co wskazuje na bardzo niskie prawdopodobieństwo klasy przeciwnej), a Random Forest nadaje wartości równe 0.000.

W przypadku klasy „Zajęte" modele również radzą sobie bardzo dobrze, co potwierdza obserwacja nr 9: CART przypisuje jej prawdopodobieństwo 0.958, a Random Forest 1.000, co wskazuje na wysoką pewność klasyfikacji.

Jedyny błąd w analizowanym fragmencie danych dotyczy obserwacji, która faktycznie należy do klasy „Zajęte", lecz oba modele sklasyfikowały ją jako „Wolne". Jest to przykład fałszywie ujemnej predykcji. CART popełnia ten błąd z dużą pewnością (prawdopodobieństwo klasy „Zajęte" to jedynie 0.00155), natomiast Random Forest wykazuje wyższą niepewność – prawdopodobieństwo klasy „Zajęte" wynosi 0.208, co może sugerować, że obserwacja zawiera cechy nietypowe lub graniczne. Random Forest jest mniej pewny błędnej decyzji, co świadczy o jego większej elastyczności.

Tabela predykcji pokazuje, że Las Losowy generuje wyższe prawdopodobieństwa dla poprawnych klas, co świadczy o jego większej pewności i stabilności w porównaniu do C&RT.

Wnioski

Podsumowanie Oceny i Wybór Modelu

Celem projektu było rozwiązaniu problemu klasyfikacji binarnej (wykrywanie zajętości pomieszczenia) przy użyciu dwóch modeli: Drzewa Decyzyjnego C&RT i Lasu Losowego (Random Forest). Pierwotnie rozważano jeszcze zastosowanie Regresji Logistycznej, jednak po dogłębnej analzie odrzucono ten pomysł.

Interpretowalność:
Analiza ważności cech wykazała, że zmienna Light jest kluczowym predyktorem.

Jakość:
Model Random Forest osiąga lepszą jakość niż pojedyncze drzewo (C&RT), z wynikiem AUC-ROC bliskim

1.0 i niemal idealną wartością Kappa. Cechuje się mniejszą asymetrią błędów. Jest stabilny i skuteczny w identyfikacji klasy mniejszościowej (“Zajęte”).

Rekomendacja:
Z uwagi na to, że w systemach Smart Building kluczowe jest minimalizowanie błędów fałszywie negatywnych (nieprzeoczenie obecności osoby), model o najwyższej Czułości (Sensitivity) i AUC-ROC jest modelem preferowanym. Do wdrożenia w systemie zarządzania budynkiem (np. do sterowania klimatyzacją i oświetleniem) rekomendowany jest Model Lasu Losowego (Random Forest). Zapewnia on najwyższą trafność predykcyjną i stabilność.

Uwagi

Oba modele dają bardzo podobne, a zarazem bardzo dobre wyniki predykcji. Skuteczność modeli sięga prawie 100% przez co ciężko je porównać i jednoznacznie ocenić. Zmienna Light okazała się ekstremalnie silnym predyktorem dlatego chcąc porównać modele na trudniejszych przypadkach, rozważyłabym usunięcie tej zmiennej (ponieważ to ona definiuje zajętość w głównej mierze). Dodatkowo rozważyłabym zbudowanie modelu kNN jako modelu bazowego do porównania wyników.