Analiza Projektu

Wykonawca: Michał Krawczyk

Data wykonania: 13.06.2022

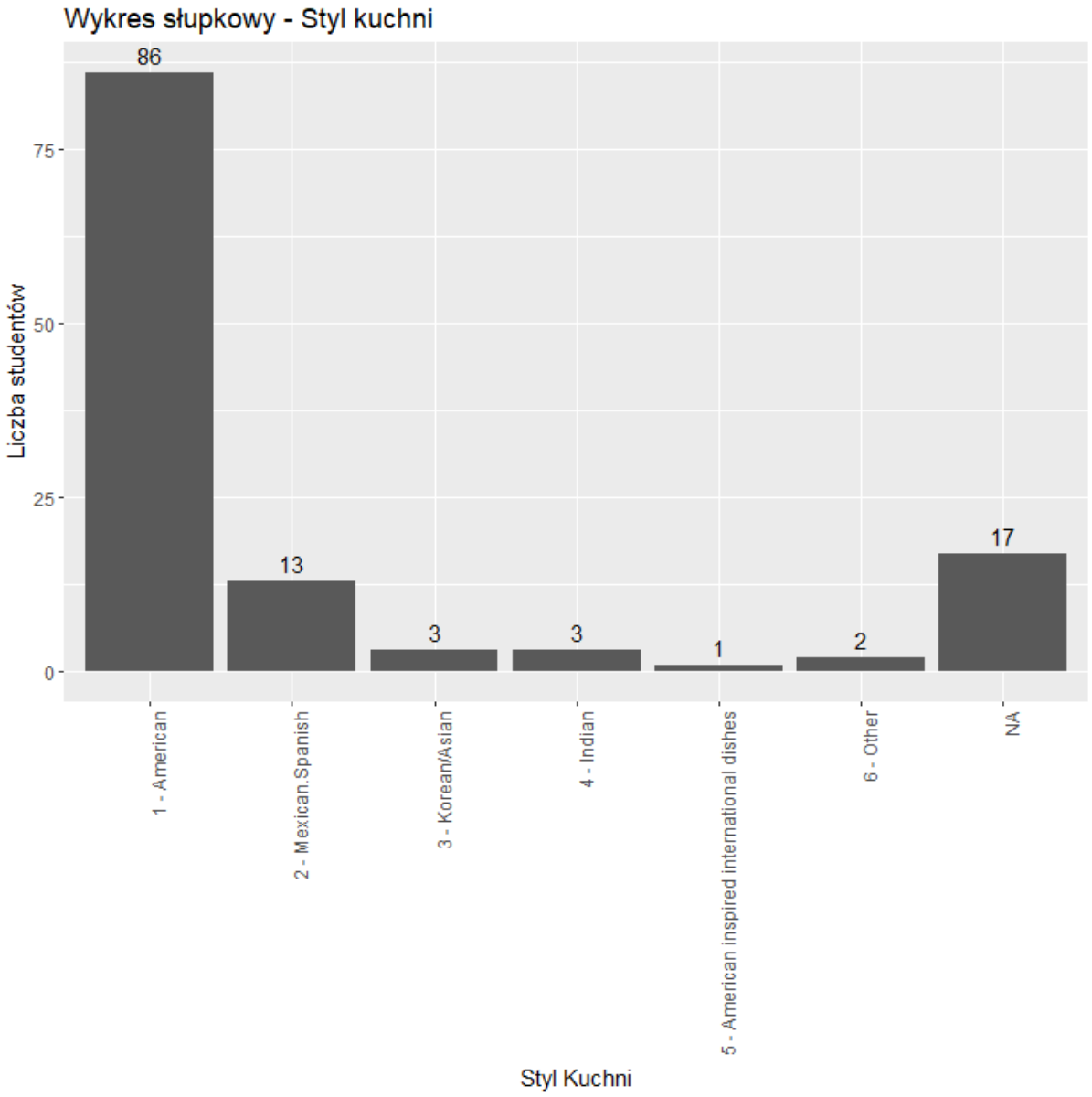
Spis treści

Opis zbioru danych

Do wykonania projektu użyłem zbioru danych „Food choices” z kaggle.com. Zbiór ten zawiera informacje dotyczące preferencji żywieniowych i nawyków żywieniowych 126 studentów Mercyhurst University uczestniczących w ankiecie. Zawiera różnorodne cechy opisujące ich preferencje dotyczące żywności oraz dane demograficzne. Opis używanych cech:

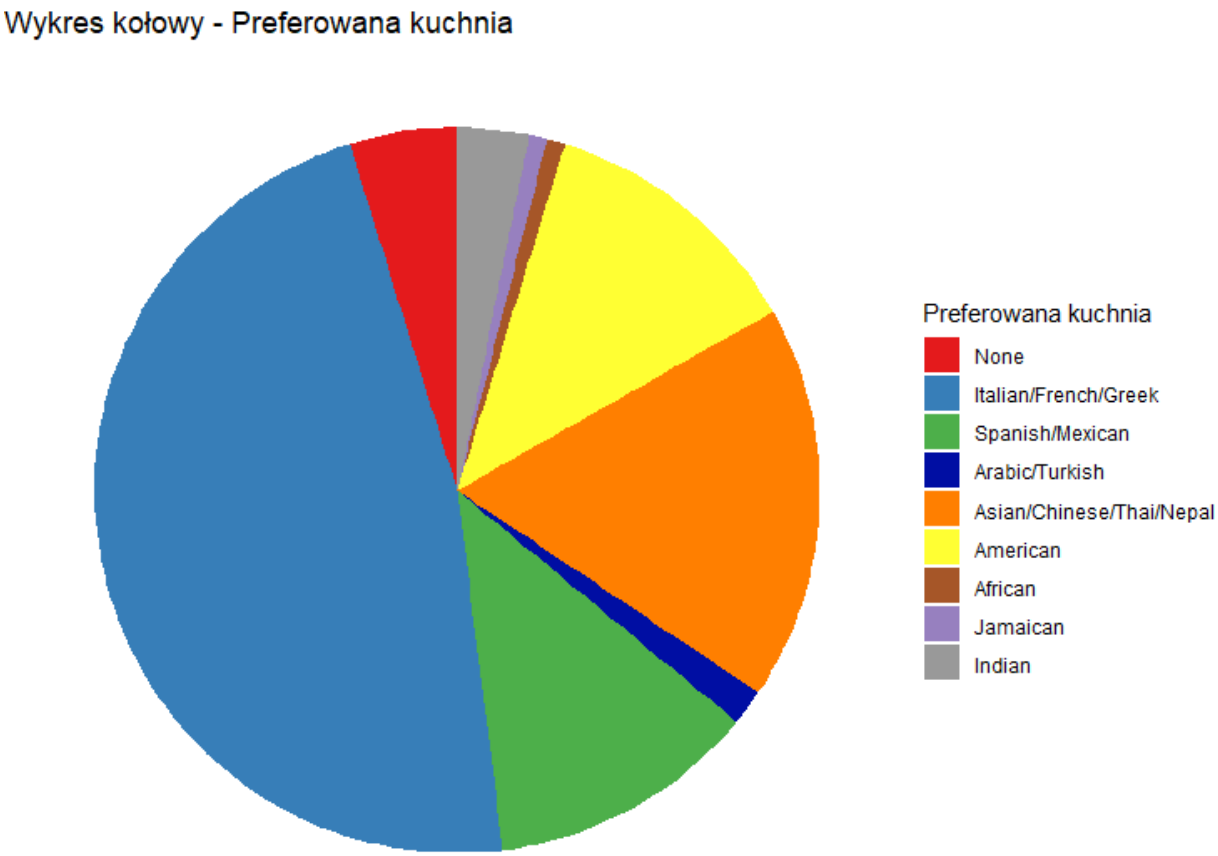
1. **GPA**: Średnia ocen uczestnika w szkole.
2. **Gender**: Płeć uczestnika (mężczyzna lub kobieta).
3. **breakfast**: Do wyboru: owsianka lub donut
4. **calories_day**: Czy spożywanie kalorii jest ważne?
5. **coffee**: Do wyboru: frapuccino lub espresso
6. **comfort_food**: Ulubione potrawy lub przekąski.
7. **comfort_food_coded**: To co w pkt. 6 tylko do wyboru z listy.
8. **cook**: Częstotliwość gotowania posiłków w domu.
9. **cuisine**: Styl kuchni jaki się jadło podczas dorastania (np. włoska, amerykańska, meksykańska itp.).
10. **eating_changes**: Zmiany w nawykach żywieniowych od pójścia do college’u.
11. **eating_changes_coded**: To co pkt. 10 tylko wybór z listy.
12. **eating_out**: Częstość jedzenia „na mieście”.
13. **employment**: Zatrudnienie.
14. **ethnic_food**: Jak chętnie odpowiadany je jedzenie regionalne?
15. **exercise**: Częstotliwość wykonywania ćwiczeń fizycznych w tygodniu.
16. **father_education**: Wykształcenie ojca.
17. **fav_cuisine**: Ulubiony styl kuchni.
18. **fav_cuisine_coded**: To co pkt. 17 tylko do wyboru.
19. **fav_food**: Ulubione potrawy uczestnika (ugotowane w domu / kupione w sklepie / oba).
20. **fries**: Frytki z McDonald’s lub domowe.
21. **fruit_day**: Jak chętnie jesz owoce.
22. **grade_level**: Na jakim poziomie studiów jest badany.
23. **greek_food**: Jak chętnie odpowiadany je jedzenie greckie?
24. **healthy_feel**: Jak zdrowo czuje się badany.
25. **ideal_diet_coded**: Wybór idealnej diety według badanego.
26. **income**: Dochód uczestnika.

27. **indian_food**: Jak chętnie odpowiadany je jedzenie indyjskie?
28. **italian_food**: Jak chętnie odpowiadany je jedzenie włoskie?
29. **life_rewarding**: Czy odpowiadający czuje, że życie jest nagradzające?
30. **marital_status**: Stan cywilny odpowiadającego.
31. **meals_dinner_friend**: Częstotliwość spożywania obiadu z przyjaciółmi.
32. **mother_profession**: Zawód matki uczestnika.
33. **nutritional_check**: Czy odpowiadający spożywa wartości odżywcze?
34. **on_off_campus**: Miejsce zamieszkania odpowiadającego.
35. **parents_cook**: Jak często w tygodniu gotują rodzice odpowiadającego?
36. **pay_meal_out**: Jak dużo odpowiadający zapłaciłby za jedzenie „na mieście”?
37. **persian_food**: Jak chętnie odpowiadany je jedzenie perskie?
38. **self_perception_weight**: Samoocena wagi.
39. **sports**: Uprawianie sportu (tak / nie).
40. **thai_food**: Jak chętnie odpowiadany je jedzenie tajskie?
41. **type_sports**: Preferowane rodzaje aktywności fizycznej.
42. **vitamins**: Czy bierzesz jakieś witaminy?
43. **weight**: Waga.



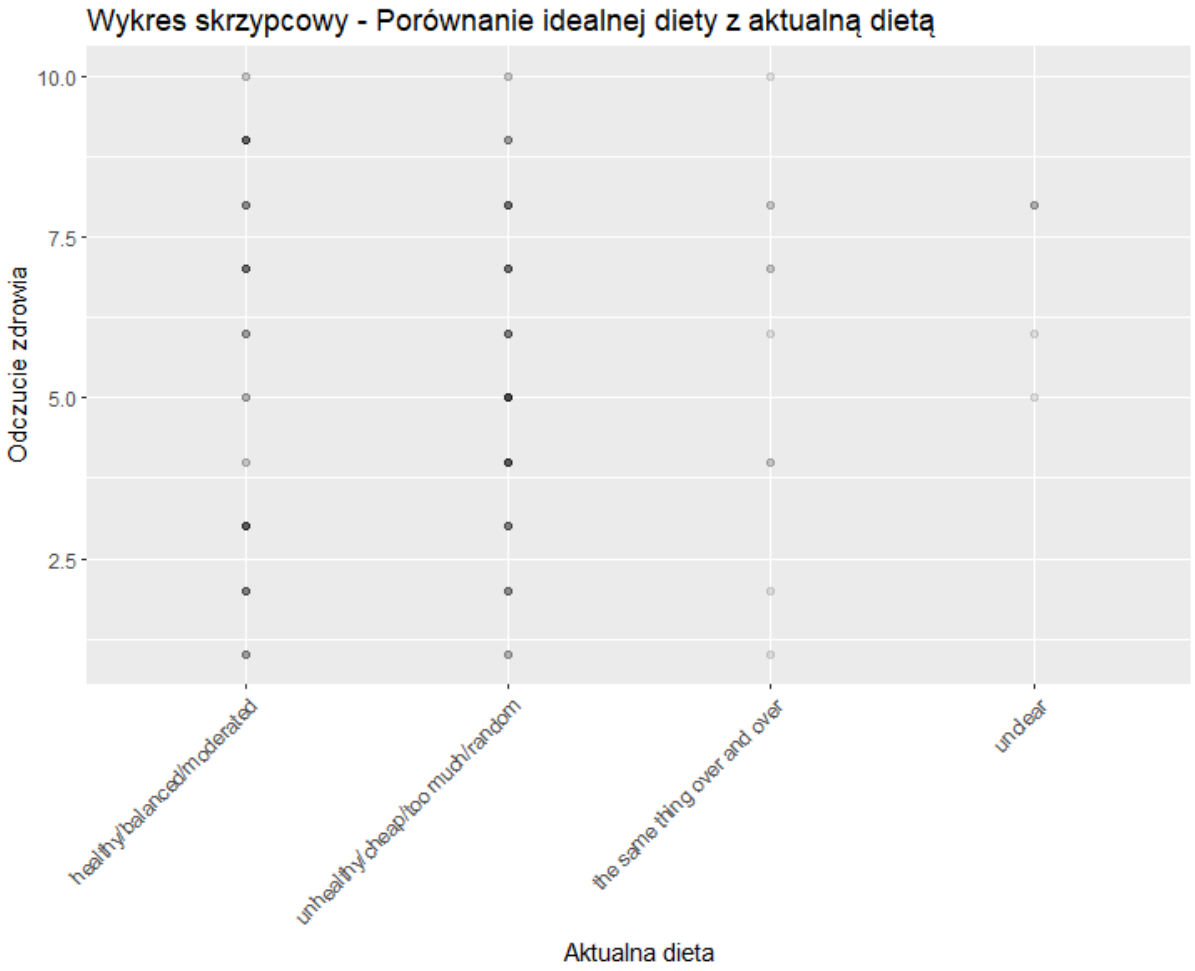
Rysunek 1. Styl kuchni

Ten wykres przedstawia liczbę studentów dorastających jedząc potrawy z danego stylu kuchni. Widać tutaj, że najwięcej studentów dorastało w kuchni Amerykańskiej co jest spowodowane tym, że badanie zostało przeprowadzone w USA. Wielu odpowiadających nie udzieliło odpowiedzi. Drugim najbardziej popularnym stylem kuchni był Meksykańsko/Hiszpański. Może to być spowodowane tym, że USA sąsiaduje z Meksykiem.



Rysunek 2. Preferowana kuchnia

Ten wykres przedstawia preferowany styl kuchni studentów. Warto zauważyć, że najbardziej preferowanym stylem jest Włoski/Francuski/Grecki a taka odpowiedź nie występowała w pytaniu o styl kuchni z jakim się dorastało. Bardzo popularny wśród studentów jest również styl Azjatycki.



Rysunek 3. Dieta

Na tym wykresie widać, że najwięcej studentów je tanio i słabo ale równie dużo je zdrowo. Ci, którzy jedzą zdrowo mają skrajne odczucie zdrowia, ponieważ zaznaczali głównie wysokie oraz niskie wartości. Jedzący niezdrowo są raczej zgodni, że czują się „średnio”, zaznaczali wartości w środku.

Opis kodu

rpart

Postanowiłem wykorzystać powyżej opisane dane do przewidywania preferowanego stylu jedzenia studentów. W tym celu użyłem biblioteki „rpart”. Biblioteka „rpart” to biblioteka uczenia maszynowego, która służy do budowania drzew klasyfikacji i regresji.

Najpierw załadowałem potrzebne biblioteki:

```
library(rpart)
library(dplyr)
```

Następnie załadowałem dane do zmiennej w celu ich dalszej edycji.

```
# Załadowanie danych do zmiennej
data_corrected <- data

# Usunięcie niepotrzebnych kolumn, usunąłem wszystkie kolumny zawierające takie dane,
# na które ankietowani mogli odpowiedzieć w dowolny sposób (pytania otwarte)
data_corrected <- select(data_corrected, -comfort_food, -comfort_food_reasons,
                         -diet_current, -eating_changes, -father_profession,
                         -fav_cuisine, -healthy_meal, -meals_dinner_friend,
                         -mother_profession, -type_sports, -food_childhood, -ideal_diet)

# Usunięcie niepotrzebnych kolumn, usunąłem wszystkie kolumny zawierające takie dane,
# które były odpowiedziami na pytania o to jaką ilość kalorii ocenia odpowiadający
# w danym produkcie
data_corrected <- select(data_corrected, -calories_chicken, -calories_scone,
                         -tortilla_calories, -turkey_calories, -waffle_calories)

# Zmieniam typ zmiennych na numeric
data_corrected$weight <- as.numeric(data_corrected$weight)
data_corrected$GPA <- as.numeric(data_corrected$GPA)
```

Następnie podzieliłem zbiór na dane uczące i walidacyjne.

```
# Podział danych na zbiór treningowy i testowy (np. 90% treningowy, 10% testowy)
set.seed(123)  # Ustawienie ziarna losowości dla powtarzalności wyników
train_indices <- sample(1:nrow(data_corrected), 0.9*nrow(data_corrected))  # Indeksy próbek
treningowych
train_data <- data_corrected[train_indices, ]  # Zbiór treningowy
test_data <- data_corrected[-train_indices, ]  # Zbiór testowy
```

Potem stworzyłem model. Większość parametrów pozostawiłem na ustawienie domyślne, ponieważ ich zmiana nie poprawiała wyniku.

```
# Tworzenie modelu drzewa decyzyjnego
ctrl <- rpart.control(cp = 0.0001)
model <- rpart(fav_cuisine_coded ~ ., data = train_data, control = ctrl)
```

Następnie przewidziałem dane, sprawdziłem dokładność porównania oraz porównałem dane przewidziane z danymi walidacyjnymi.

```
# Dokonywanie predykcji na zbiorze testowym
predictions <- predict(model, newdata = test_data, type = "class")

# Ocenianie wyników modelu
accuracy <- sum(predictions == test_data$fav_cuisine_coded) / nrow(test_data)
cat("Dokładność modelu:", accuracy)

# Połączenie wektorów w ramkę danych
result <- data.frame(predictions, test_data$fav_cuisine_coded)

# Ustawienie indeksów
rownames(result) <- 1:nrow(result)

# Wydrukowanie ramki danych przedstawiającej porównanie danych przewidzianych
# i danych testowych
print(result)
```

Następnie przedstawiłem graficznie to porównanie.

```
# Dodanie indeksów do ramki danych result
result$Index <- rownames(result)
result$Index <- as.numeric(result$Index)

# Stworzenie wykresu z indeksami na osi X i danymi na osi Y
ggplot(result, aes(x = Index)) +
  geom_point(aes(y = predictions), color = "blue", size = 3, shape = 3) +
  geom_point(aes(y = test_data.fav_cuisine_coded), color = "red", size = 3, shape = 4) +
  labs(x = "Indeks", y = "Wartość") +
  ggtitle("Porównanie predykcji i realnych danych") +
  scale_x_continuous(breaks = result$Index)
```

Oraz pokazałem macierz pomyłek.

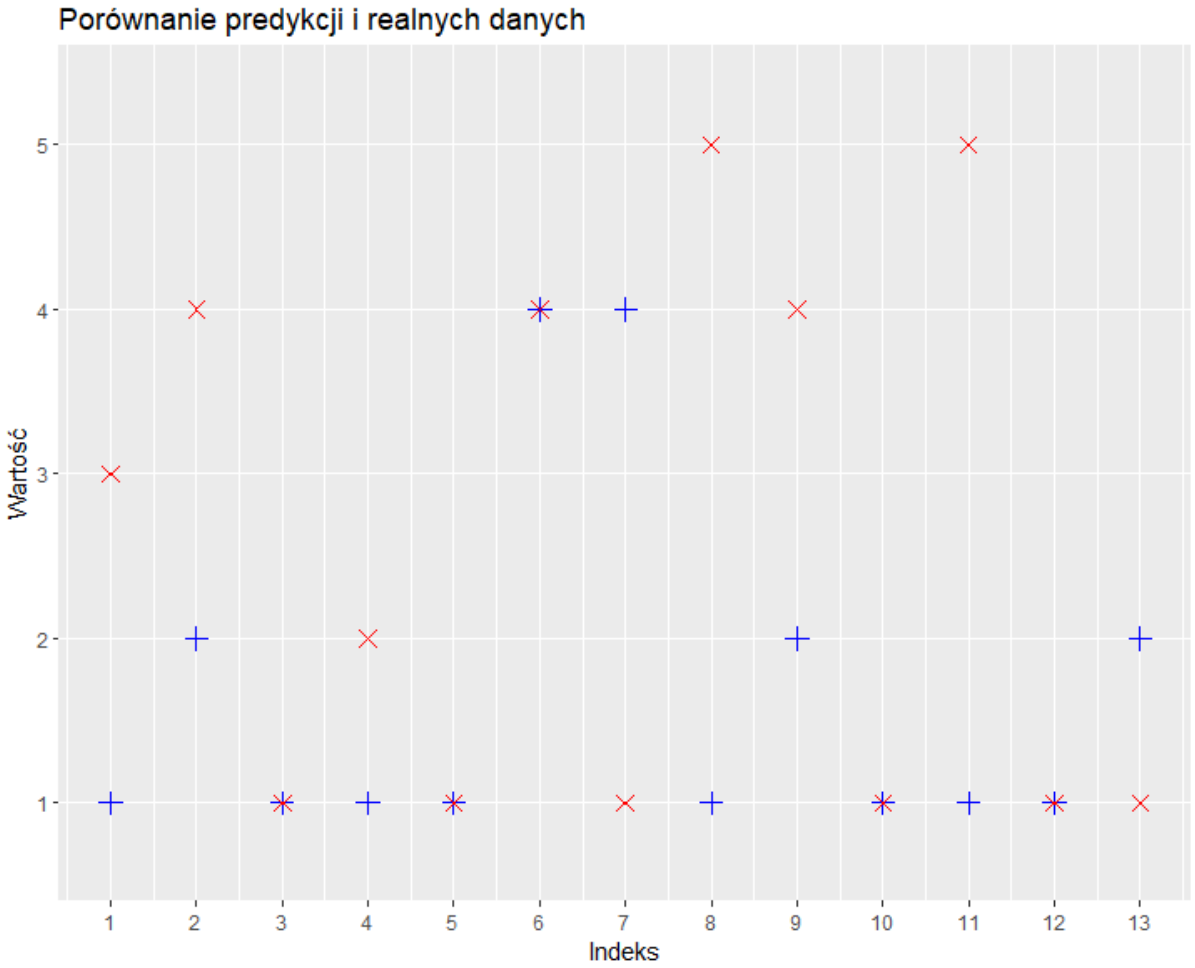
```
# Macierz pomyłek
library(caret)
confusionMatrix(result$predictions,result$test_data.fav_cuisine_coded)
```

Dokładność modelu wyszła niska, bo tylko 0,3846.

Porównanie danych przewidzianych oraz danych walidacyjnych:

```
   predictions test_data.fav_cuisine_coded
1            1                           3
2            2                           4
3            1                           1
4            1                           2
5            1                           1
6            4                           4
7            4                           1
8            1                           5
9            2                           4
10           1                           1
11           1                           5
12           1                           1
13           2                           1
```

Porównanie na wykresie:



Rysunek 4. Predykcja

Macierz pomyłek:

```
        predictions
targets 2 3 4 5 6
      2 4 1 1 0 2
      3 1 0 0 2 0
      5 1 0 0 1 0
```

randomForest

Analogicznie do modelu rpart stworzyłem model używając randomForest. Najpierw oczyściłem dane aby nie zawierały pustych komórek.

```
# Usunięcie niepotrzebnych kolumn, usunąłem wszystkie kolumny zawierające takie dane,
# na które ankietowani mogli odpowiedzieć w dowolny sposób (pytania otwarte)
data_corrected <- select(data_corrected, -comfort_food, -comfort_food_reasons,
                         -diet_current, -eating_changes, -father_profession,
                         -fav_cuisine, -healthy_meal, -meals_dinner_friend,
                         -mother_profession, -type_sports, -food_childhood, -ideal_diet)

# Usunięcie niepotrzebnych kolumn, usunąłem wszystkie kolumny zawierające takie dane,
# które były odpowiedziami na pytania o to jaką ilość kalorii ocenia odpowiadający
# w danym produkcie
data_corrected <- select(data_corrected, -calories_chicken, -calories_scone,
                         -tortilla_calories, -turkey_calories, -waffle_calories)

# Zmieniam typ zmiennych na numeric
data_corrected$weight <- as.numeric(data_corrected$weight)
data_corrected$GPA <- as.numeric(data_corrected$GPA)

# Usunięcie kolumn zawierających dużo pustych danych w celu wyczyszczenia danych
# do modelu randomForest
# Obliczenie pustych danych
nan_counts <- colSums(is.na(data_corrected))

# Posortowanie danych
sorted_nan_counts <- sort(nan_counts, decreasing = TRUE)

# Wypisanie, które kolumny zawierają ile pustych danych
for (column in names(sorted_nan_counts)) {
  cat("Column:", column, "\tNaN Count:", sorted_nan_counts[column], "\n")
}

# Usunięcie kolumny "calories_day"
data_corrected <- data_corrected[, -which(names(data_corrected) == "calories_day")]
# Usunięcie kolumny "comfort_food_reasons_coded"
data_corrected <- data_corrected[, -which(names(data_corrected) == "comfort_food_reasons_coded")]
# Usunięcie kolumny "cuisine"
data_corrected <- data_corrected[, -which(names(data_corrected) == "cuisine")]
# Usunięcie kolumny "exercise"
data_corrected <- data_corrected[, -which(names(data_corrected) == "exercise")]
# Usunięcie kolumny "employment"
data_corrected <- data_corrected[, -which(names(data_corrected) == "employment")]
# Usunięcie kolumny "GPA"
data_corrected <- data_corrected[, -which(names(data_corrected) == "GPA")]
# Usunięcie kolumny "weight"
data_corrected <- data_corrected[, -which(names(data_corrected) == "weight")]
# Usunięcie kolumny "cook"
data_corrected <- data_corrected[, -which(names(data_corrected) == "cook")]
# Usunięcie kolumny "mother_education"
data_corrected <- data_corrected[, -which(names(data_corrected) == "mother_education")]
# Usunięcie kolumny "drink"
data_corrected <- data_corrected[, -which(names(data_corrected) == "drink")]
# Usunięcie kolumny "sports"
data_corrected <- data_corrected[, -which(names(data_corrected) == "sports")]

# Usunięcie wierszów z pustymi danymi
data_corrected <- na.omit(data_corrected)
```

Następnie stworzyłem model. Ustawiłem optymalne parametry dla, których dostałem najdokładniejszy wynik. Następnie przewidziałem dane, sprawdziłem dokładność porównania oraz porównałem dane przewidziane z danymi walidacyjnymi. Następnie przedstawiłem graficznie to porównanie.

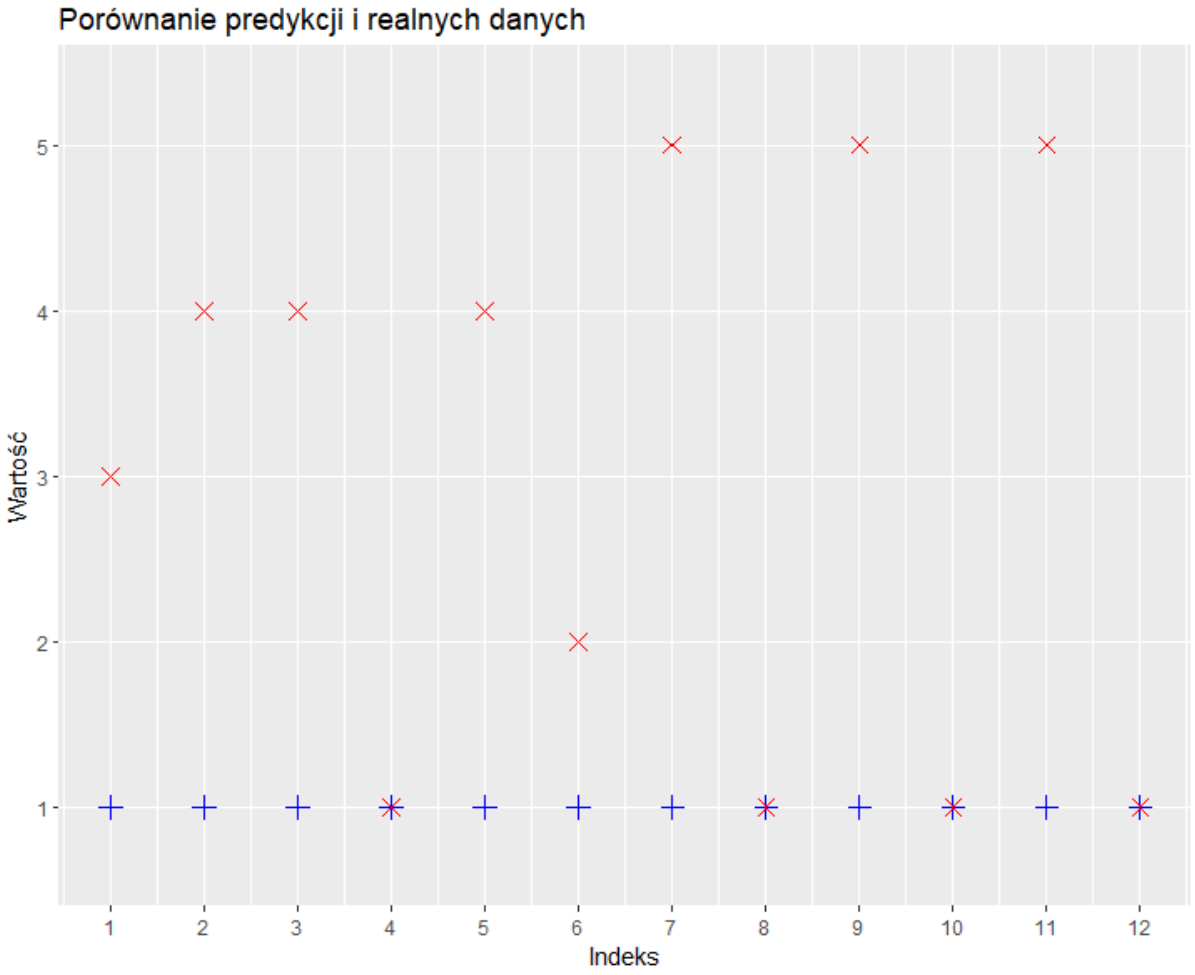
```
# Podział danych na zbiór treningowy i testowy
set.seed(123)
train_indices <- sample(1:nrow(data_corrected), 0.9 * nrow(data_corrected))
train_data <- data_corrected[train_indices, ]
test_data <- data_corrected[-train_indices, ]


# Tworzenie modelu lasu losowego
model <- randomForest(fav_cuisine_coded ~ ., data = train_data, ntree = 1000)

# Dokonywanie predykcji na zbiorze testowym
predictions <- predict(model, newdata = test_data)

# Ocenianie wyników modelu
accuracy <- sum(predictions == test_data$fav_cuisine_coded) / nrow(test_data)
cat("Dokładność modelu:", accuracy)

print(predictions)

# Połączenie wektorów w ramkę danych
result <- data.frame(predictions, test_data$fav_cuisine_coded)

# Ustawienie indeksów
rownames(result) <- 1:nrow(result)

# Wydrukowanie ramki danych przedstawiającej porównanie danych przewidzianych
# i danych testowych
print(result)

# Dodanie indeksów do ramki danych result
result$Index <- rownames(result)
result$Index <- as.numeric(result$Index)

# Stworzenie wykresu z indeksami na osi X i danymi na osi Y
ggplot(result, aes(x = Index)) +
  geom_point(aes(y = predictions), color = "blue", size = 3, shape = 3) +
  geom_point(aes(y = test_data.fav_cuisine_coded), color = "red", size = 3, shape = 4) +
  labs(x = "Indeks", y = "Wartość") +
  ggtitle("Porównanie predykcji i realnych danych") +
  scale_x_continuous(breaks = result$Index)
```

Dokładność modelu wyszła niska, bo tylko 0,3333.

Porównanie danych przewidzianych oraz danych walidacyjnych:

```
predictions test_data.fav_cuisine_coded
1           1                           3
2           1                           4
3           1                           4
4           1                           1
5           1                           4
6           1                           2
7           1                           5
8           1                           1
9           1                           5
10          1                           1
11          1                           5
12          1                           1
```

Porównanie na wykresie:



Rysunek 5. Predykcja

Macierz pomyłek:

```
       predictions
targets 2 3 4 5 6
      2 4 1 1 3 3
```

Podsumowanie

Zbiór danych „Food choices” zawiera wiele ciekawych danych, które można użyć do ich analizy oraz do przewidywania za pomocą modeli sztucznej inteligencji. Podjąłem się próby predykcji preferowanego stylu kuchni studentów lecz model rpart nie spełnił do końca moich oczekiwań i jest dość niedokładny. Model randomForest w ogóle nie poradził sobie z tym zbiorem, ponieważ przewidywał same jedynki. Gdy próbowałem go uczyć na innych parametrach np. na mniejszej liczbie ntree wtedy jego przewidywania były losowe.