**Projekt zaliczeniowy**

**Techniki eksploracji danych**

**Zakres projektu**

Projekt miał na celu zastosowanie algorytmów uczenia maszynowego do poniższych problemów:

1. klasyfikacji binarnej
2. klasyfikacji wieloklasowej
3. regresji.

Problem te należało rozwiązać przy pomocy własnych implementacji oraz funkcji z bibliotek R:

* k-najbliższych sąsiadów
* drzewa decyzyjne
* sieci neuronowe.

Wyniki dla poszczególnych algorytmów należało porównać z funkcjami z bibliotek oraz przeanalizować wpływ hiper-parametrów na dokładność predykcji zaimplementowanych ręcznie funkcji .

Zbiory danych

1. Klasyfikacja binarna:

https://archive.ics.uci.edu/ml/datasets/Wholesale+customers

* Liczba obserwacji: 440
* Liczba atrybutów: 7
* Zmienna celu: zmienna binarna = „CHANNEL” (kanał sprzedażowy)

1. Klasyfikacja wieloklasowa:

https://archive.ics.uci.edu/ml/datasets/seeds

* Liczba obserwacji: 210
* Liczba atrybutów: 7
* Zmienna celu: 3 klasy (3 odmiany pszenicy)

1. Regresja:

https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

* Liczba obserwacji: 209
* Liczba atrybutów: 9
* Zmienna celu: zmienna numeryczna (relatywna wydajność modeli)

Przekształcenia danych:

* Dane do klasyfikacji binarnej:
  + Usunięta została kolumna z klasami (w celu uproszczenia obliczeń)
  + Zamieniono kolumnę „CHANNEL” na typ „factor” i ustalono ją jako zmienną celu
* Dane do klasyfikacji wieloklasowej:
  + Usunięto wiersze w wartościami ‘NA’
  + Kolumnę nr 8 zamieniono na klasy i ustalono zmienną celu
* Dane do problemu regresji:
  + Usunięto 2 i 10 kolumnę ze względu, iż 2 kolumna zawierała dużo indywidualnych oznaczeń modeli sprzętu, co mogło by wpływać na uczone modele. Kolumna 10 została odrzucona ze względu na dane predykcyjne zespołu który opracowywał te dane.

Kroswalidacja – We wszystkich przypadkach użyto parametru *kFold = 10*, w celu zebrania odpowiedniej statystyki modeli.

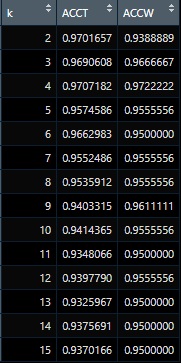
Najlepsze modele dla poszczególnych algorytmów   
(wyniki dla zbioru walidacyjnego)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Algorytm** | **Implementacja** | **Parametry** | **Ocena jakości dla zbioru Testowego**  **(Trafność)** |
| **Klasyfikacja Binarna** | KNN | Własna | k = 2 | 0.8953 |
|  | Drzewa decyzyjne | Własna | depth = 6, minobs = 2, Entropy, overfit = none | 0.9069 |
|  | Sieci NN | Własna | h = (5,5), lr = 0.001,  *iter =* 200000 | 0.9046 |
|  | KNN | Biblioteka R | k = 3 | 0.9090 |
|  | Drzewa decyzyjne | Biblioteka R | max depth = 4 | 0.9204 |
|  | Sieci NN | Biblioteka R | h = 5 | 0.9157 |
| **Klasyfikacja Wieloklasowa** | KNN | Własna | k = 4 | 0.9722 |
|  | Drzewa decyzyjne | Własna | depth = 6, minobs = 2, Gini, overfit = prune, cf = 0.1 | 0.9222 |
|  | Sieci NN | Własna | h = (5,5), lr = 0.001,  *iter =* 200000 | 0.9055 |
|  | KNN | Biblioteka R | k = 11 | 0.9248 |
|  | Drzewa decyzyjne | Biblioteka R | max depth = 2 | 0.8938 |
|  | Sieci NN | Biblioteka R | h = 3 | 0.9552 |
| **Regresja** | KNN | Własna | k = 2 | 38.79 |
| **( MAE )** | Drzewa decyzyjne | Własna | depth = 5, minobs = 2 | 31.36 |
|  | Sieci NN | Własna | h = (4,4), lr = 0.001,  *iter =* 100000 | 31.18 |
|  | KNN | Biblioteka R | k = 2 | 32.61 |
|  | Drzewa decyzyjne | Biblioteka R | max depth = 3 | 47.38 |
|  | Sieci NN | Biblioteka R | h = 10 | 0.0013 |

KNN – Wyniki dla własnej implementacji  
**Klasyfikacja Binarna**



KNN – Wyniki dla własnej implementacji  
**Klasyfikacja Wieloklasowa**

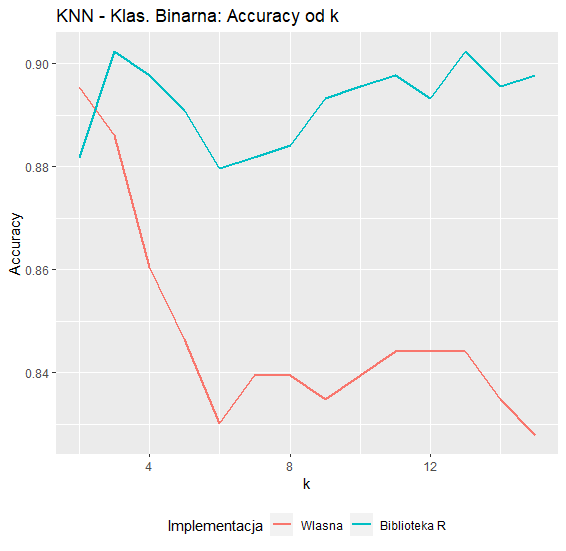


KNN – Wyniki dla własnej implementacji  
**Regresja**

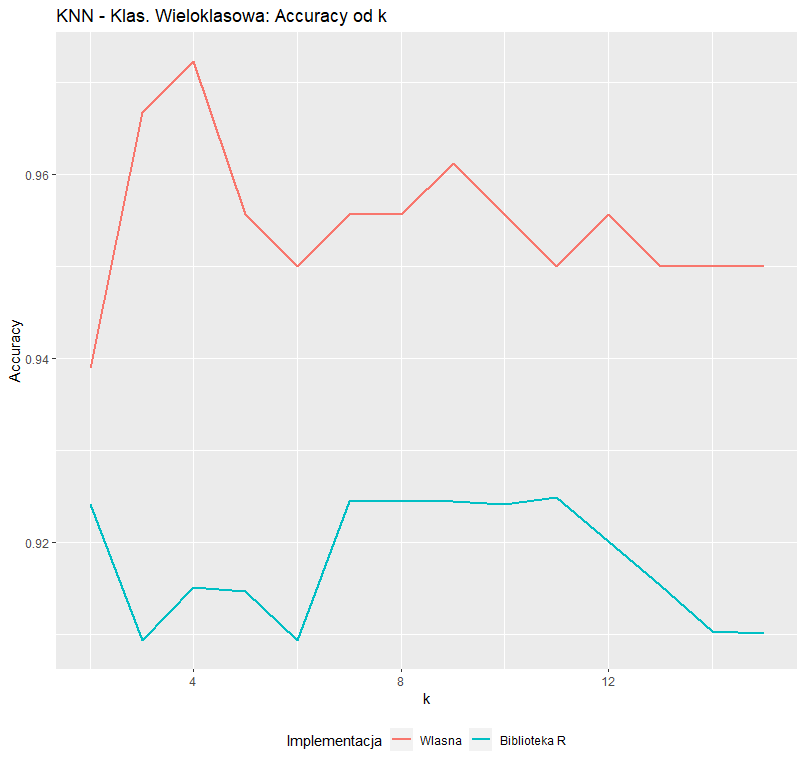
Obraz zawierający tekst

Opis wygenerowany automatycznie

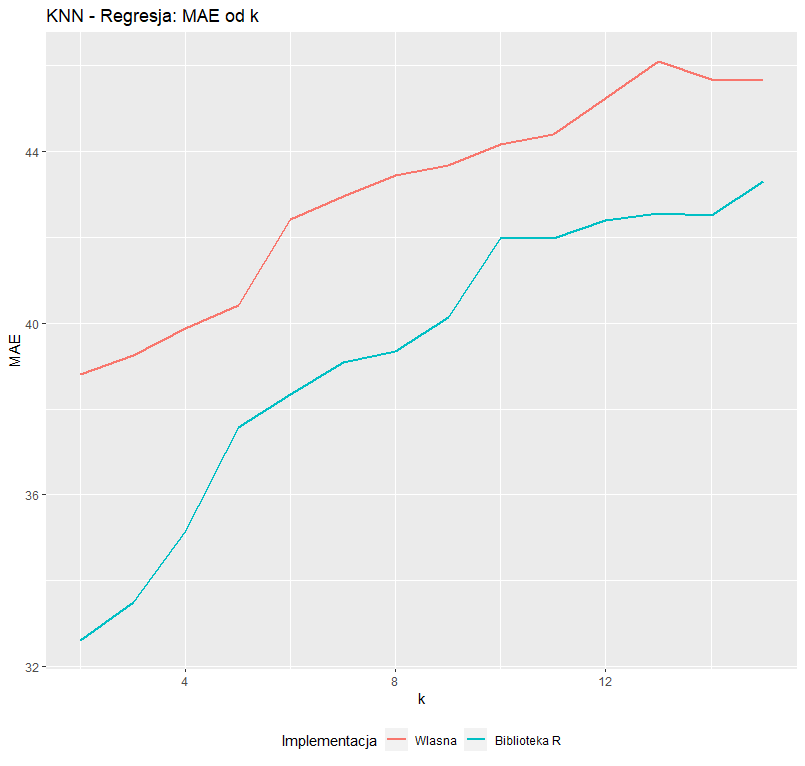
KNN – porównanie własnej implementacji z biblioteką R (knn)  
**Klasyfikacja Binarna**



KNN – porównanie własnej implementacji z biblioteką R (knn)  
**Klasyfikacja Wieloklasowa**

****

KNN – porównanie własnej implementacji z biblioteką R (knn)  
**Regresja**



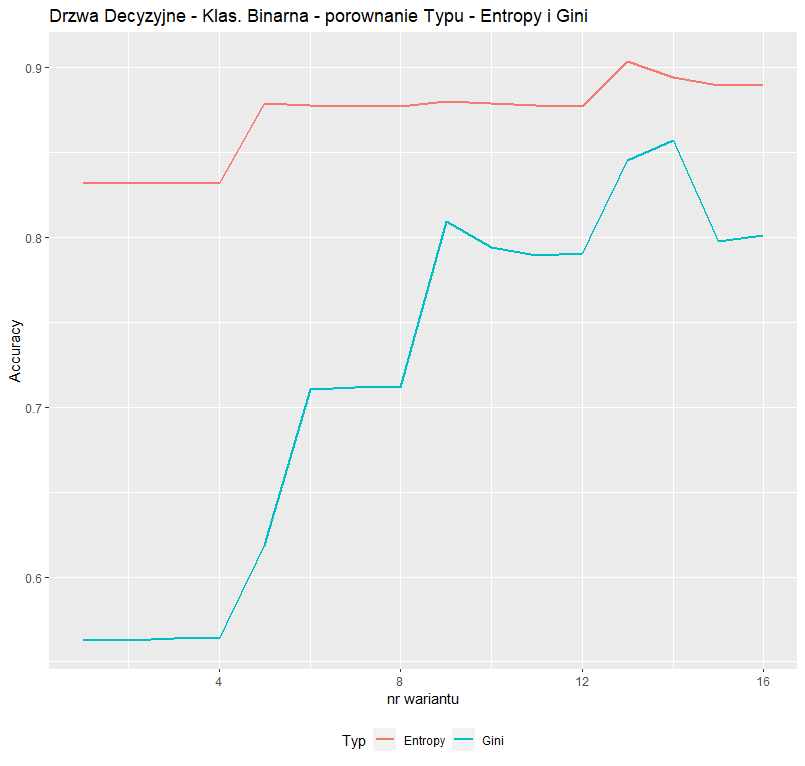
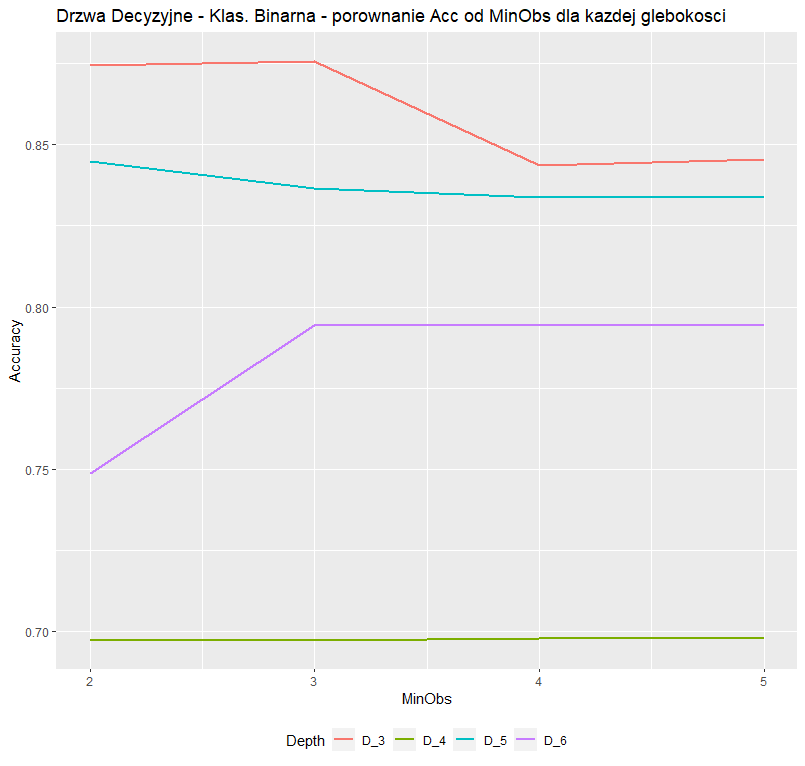
Algorytm KNN:

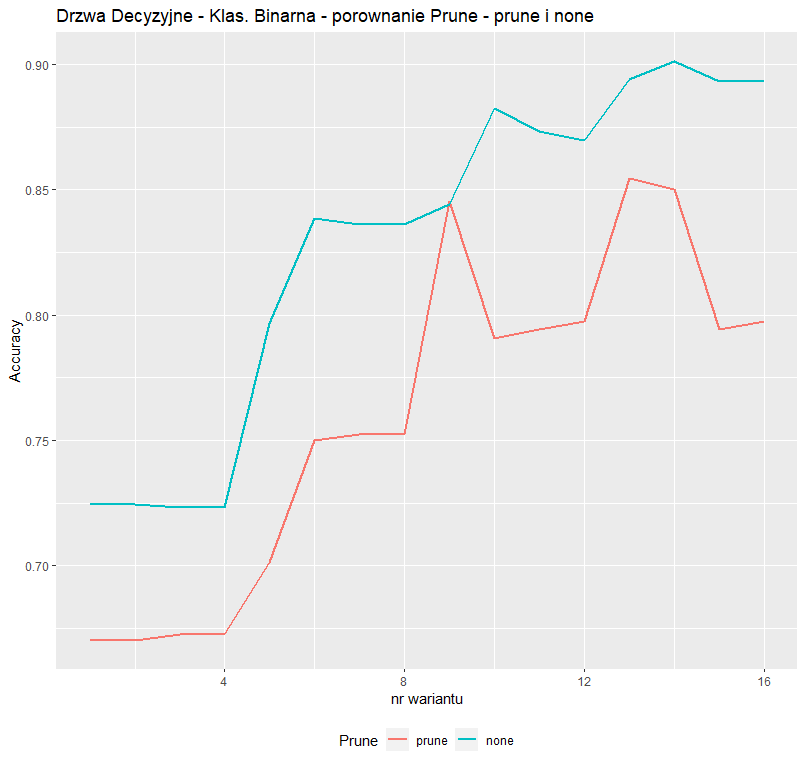
Porównanie algorytmów KNN, własnej implementacji i z biblioteki R pokazuje, że algorytmy nie różnią się aż tak drastycznie przy ocenie przynależności danej próbki do danej klasy. Widać oczywiście różnicę, czasem kilkuprocentową, w predykcji modeli, ale jak na własną ideową implementację takiego algorytmu wyszło to bardzo dobrze. Największym problemem jest czas wykonywania się takiego algorytmu, gdzie model z biblioteki R jest zdecydowanie górą w tej kwestii. Algorytm uczy się szybko, natomiast funkcja predykcyjna działa bardzo powoli. Stworzenie kilkudziesięciu modeli zajmuje kilka godzin, co nie pozwala na ogólne zastosowanie takiego algorytmu w praktyce. Sama predykcja, w porównaniu do modelu z biblioteki R, też mogła by być lepsza. Zostawia to pole do przyszłych usprawnień czy wykorzystania lepszych algorytmów-implementacji KNN.

Drzewa Decyzyjne (własna implementacja)  
**Klasyfikacja Binarna**

**depth=c(3:6), minobs=c(2:5), type=c('Entropy', 'Gini'), overfit = c('none', 'prune'), cf=c(0.1, 0.25 )**

 …(fragment)

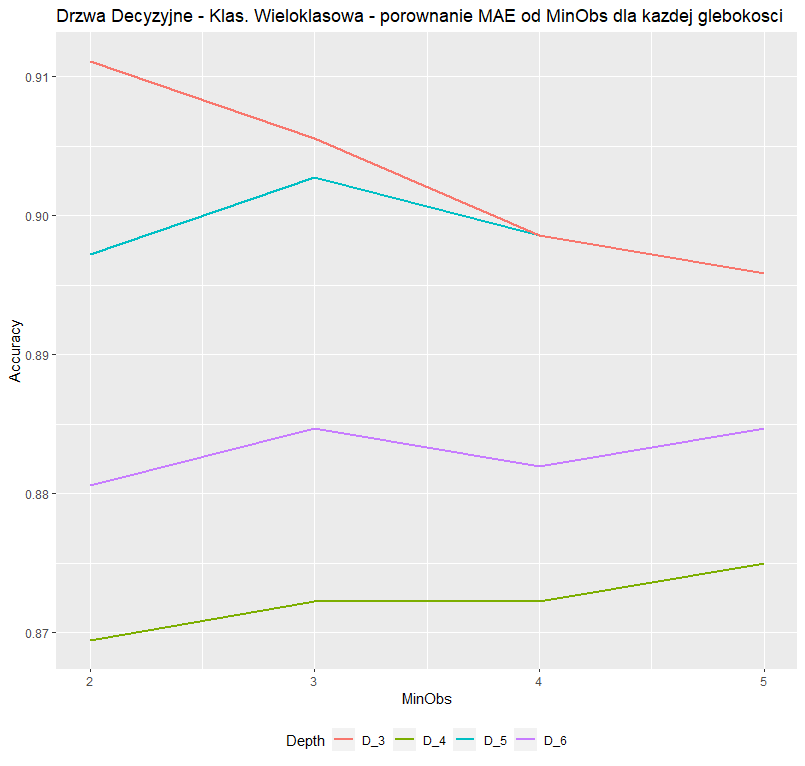


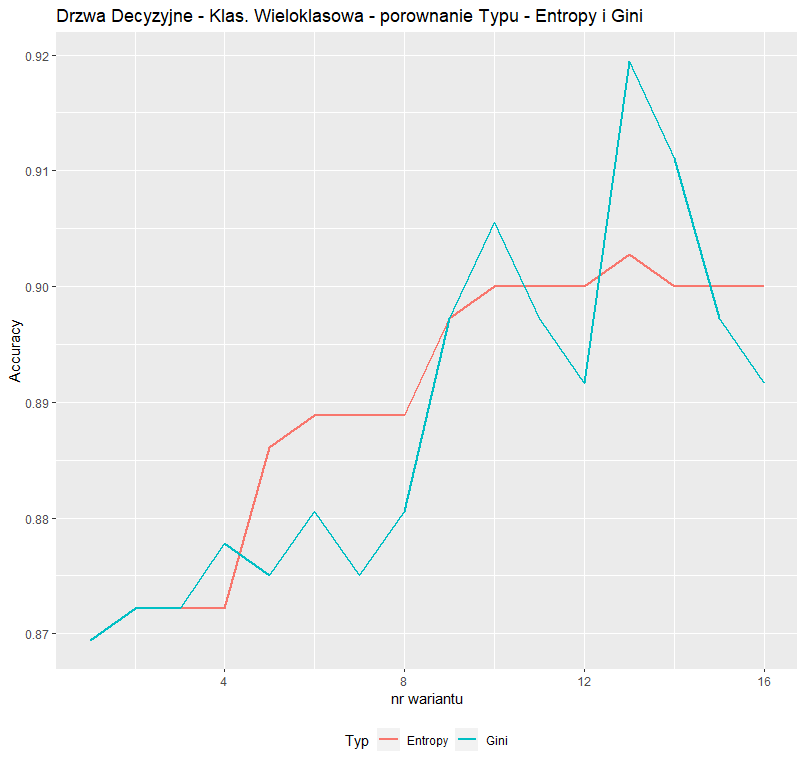


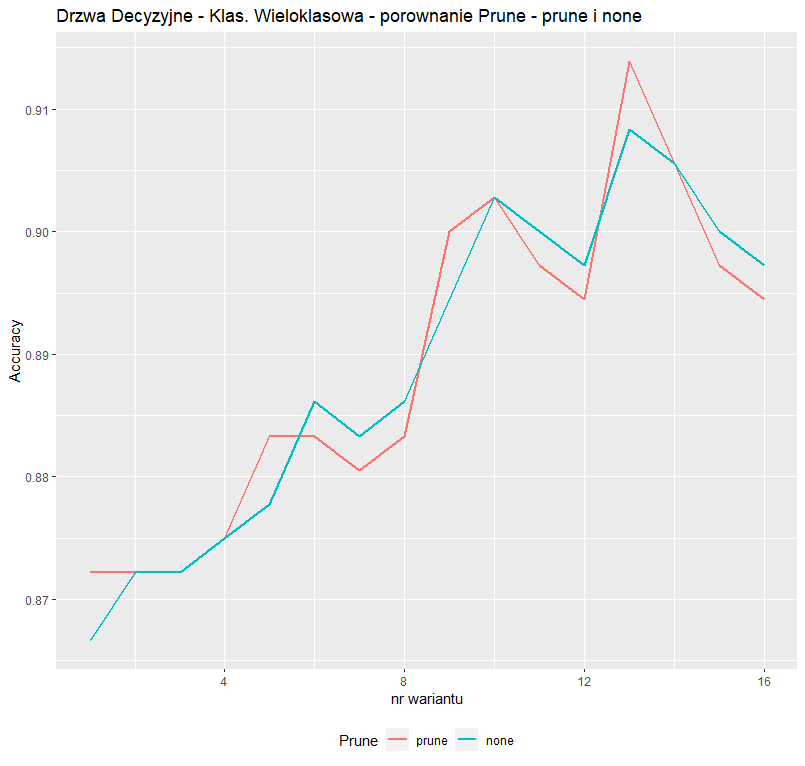
Drzewa Decyzyjne (własna implementacja)  
**Klasyfikacja Wieloklasowa   
depth=c(3:6), minobs=c(2:5), type=c('Entropy', 'Gini'), overfit = c('none', 'prune'), cf=c(0.1, 0.25 )**

Obraz zawierający tekst, komputer

Opis wygenerowany automatycznie …(fragment)





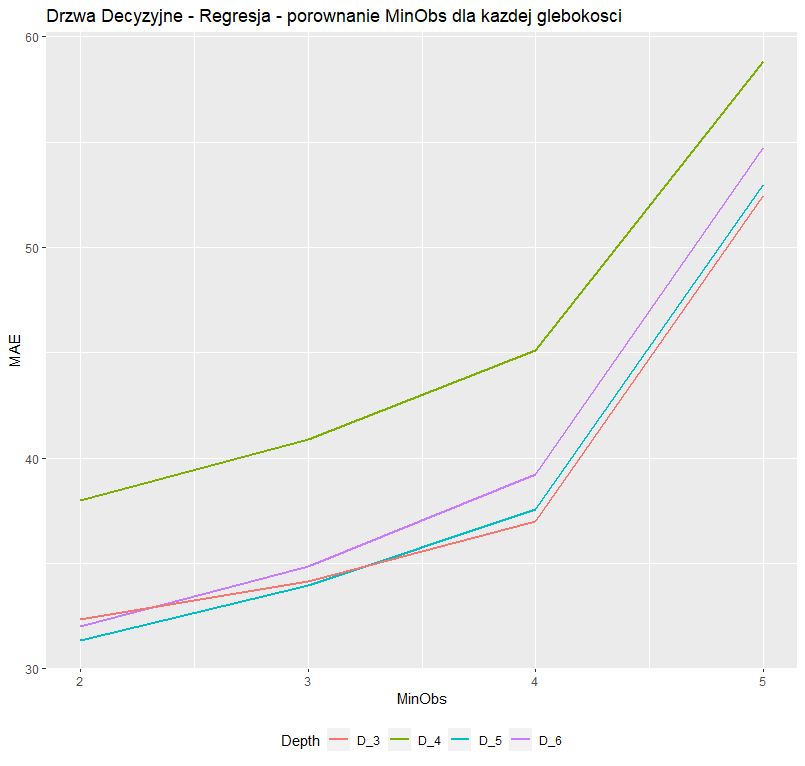


Drzewa Decyzyjne (własna implementacja)  
**Regresja**

**depth=c(3:6), minobs=c(2:5), type=c('SS'), overfit = c('none')**

Obraz zawierający tekst

Opis wygenerowany automatycznie …(fragment)

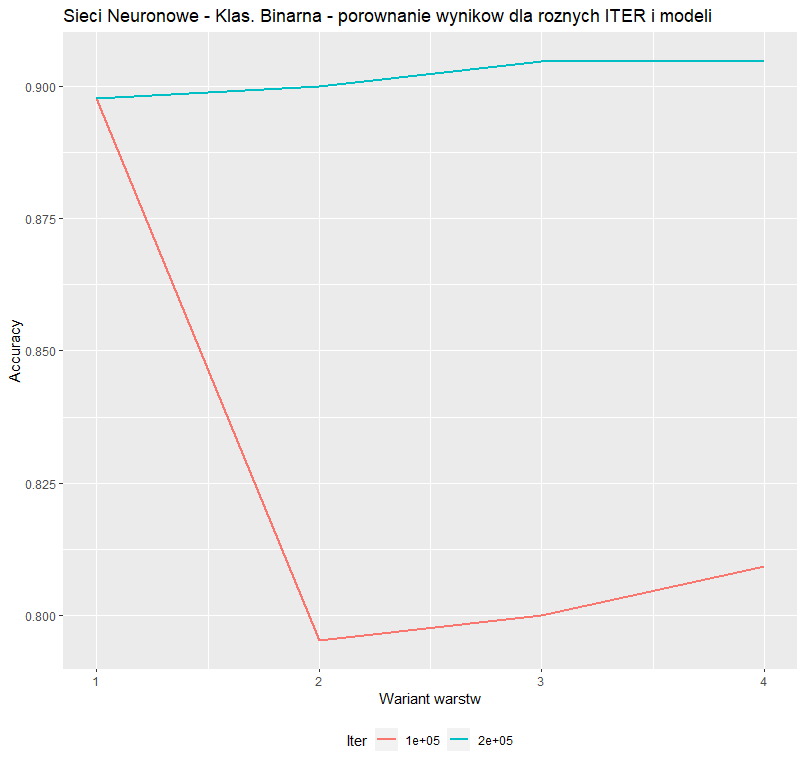


Sieci Neuronowe (własna implementacja – 2 warstwy neuronów)  
**Klasyfikacja Binarna**

h=list(c(3,4), c(4,4), c(5,5), c(6,6)), lr = c(0.001), iter = c(200000, 100000)

Obraz zawierający tekst, tablica wyników, monitor, kalkulator

Opis wygenerowany automatycznie

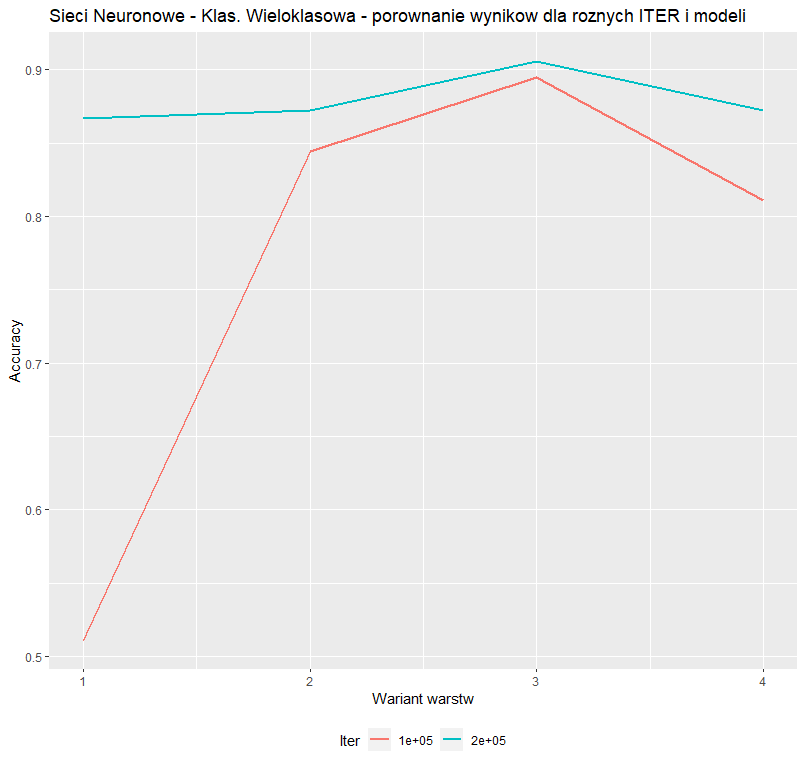


Sieci Neuronowe (własna implementacja – 2 warstwy neuronów)  
**Klasyfikacja Wieloklasowa**

h=list(c(3,4), c(4,4), c(5,5), c(6,6)), lr = c(0.001), iter = c(200000, 100000)

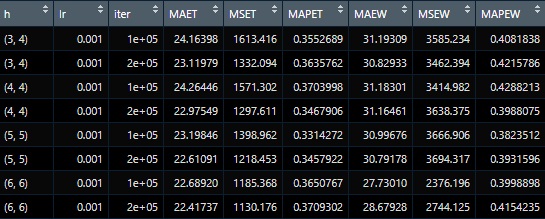
Obraz zawierający tekst, okno, tablica wyników

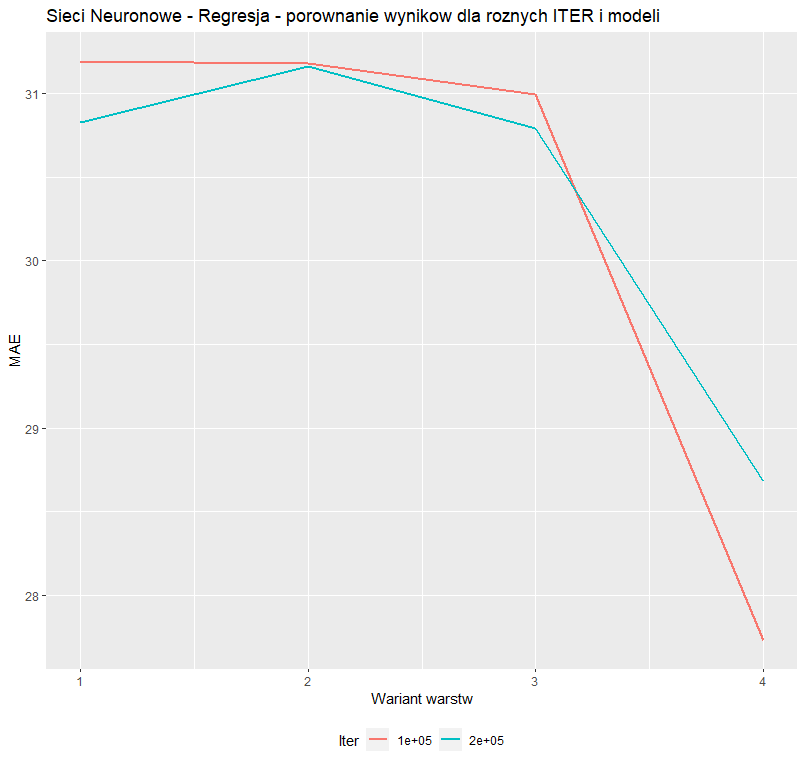
Opis wygenerowany automatycznie



Sieci Neuronowe (własna implementacja – 2 warstwy neuronów)  
**Regresja**

h=list(c(3,4), c(4,4), c(5,5), c(6,6)), lr = c(0.001), iter = c(200000, 100000)





**Podsumowanie**

Podsumowując poszczególne fragmenty związane z różnymi klasyfikatorami i ich działaniem warto spojrzeć na tabele z najlepszymi modelami dla poszczególnych algorytmów.   
Przy klasyfikacji binarnej najlepszym algorytmem okazał się *Drzewa decyzyjne* – z precyzją na poziomie 0.9204.  
Przy klasyfikacji wieloklasowej najlepszym predyktorem okazał się KNN własnoręcznie zaimplementowany. Pomimo wolnego działania i bardzo prostej idei, w tym przypadku, model ten osiągnął, przy k=4, trafienia na poziomie 0.9722.  
Przy problemie regresji to Sieci Neuronowe z biblioteki R (nnet) były najlepsze, MAE na poziomie 0.0013, osiągając wynik 3 rzędy lepszy niż inne modele. Najwidoczniej problem był tak złożony, że proste modele jak KNN i Drzewa Decyzyjne nie mogły sobie z nim dobrze poradzić.

Implementacja algorytmów pozwala poznać różnorodność i wiele możliwości kodu odpowiedzialnego ideowo za ten sam algorytm. Ideowe algorytmy można by przyśpieszyć lub użyć technik na poprawę ich precyzji predykcji.