**Techniki eksploracji danych**

**Projekt zaliczeniowy**

Projekt polegał na przeglądzie, dopracowaniu i użyciu budowanych algorytmów uczenia maszynowego. Wykonanie projektu uwzględniało kroswalidację, wpływ parametrów modeli i porównanie wyników z bibliotekami R, takimi jak Caret, rpart czy nnet. Badane algorytmy to:

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

1. Klasyfikacja binarna - *Wholesale customers*

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

* Liczba wierszy: 440
* Liczba kolumn: 7
* Kolumna Y: „CHANNEL” (kanał sprzedażowy)

1. Klasyfikacja wieloklasowa - *Balance Scale*

<https://archive.ics.uci.edu/ml/datasets/Balance+Scale>

* Liczba obserwacji: 625
* Liczba atrybutów: 4
* Zmienna celu: 3 klasy (L, B, R)

1. Regresja - *Computer Hardware*

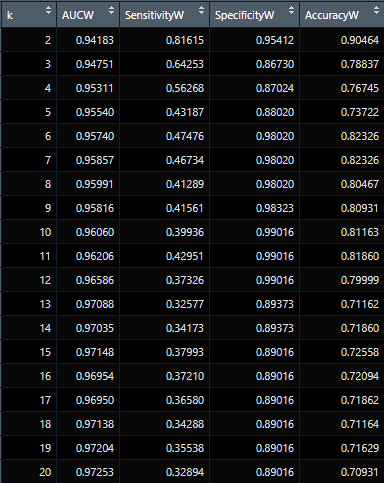
https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

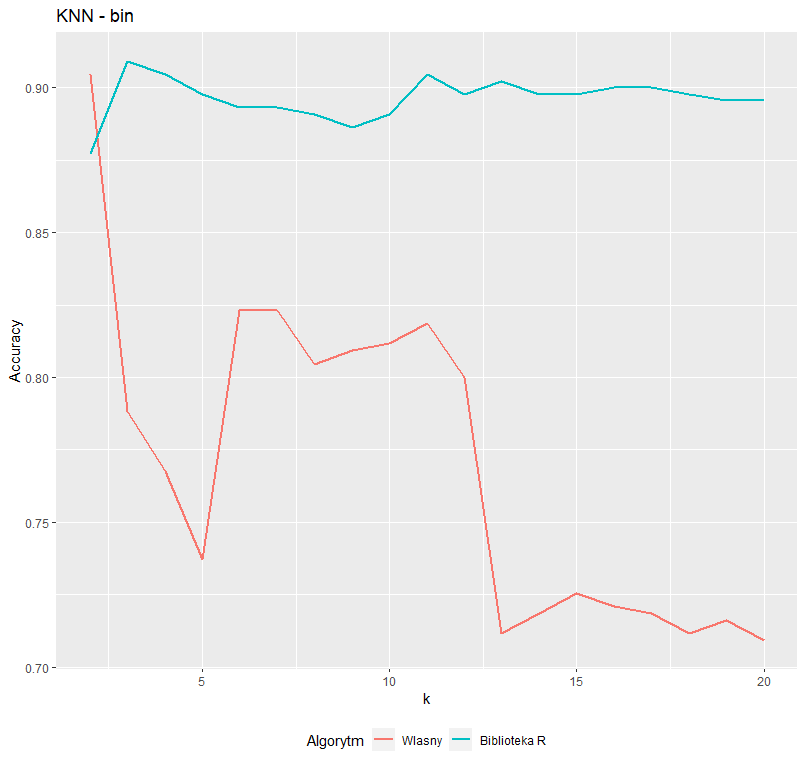
* Liczba wierszy: 209
* Liczba kolumn: 9
* Kolumna Y: zmienna numeryczna (wydajność sprzętu)
* Usunięto kolumnę 2 i 10

Najlepsze modele dla poszczególnych problemów  
klasyfikacji i regresji   
(wyniki dla zbioru walidacyjnego)

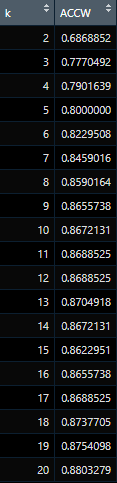
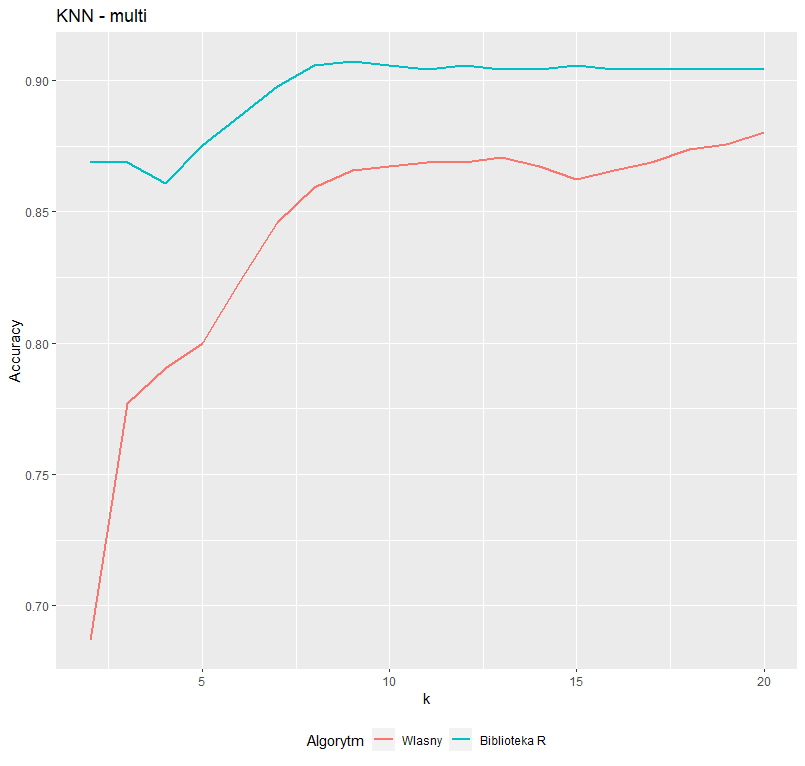
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Model** | **Implementacja** | **Parametry** | **Dokładność predykcji** |
| **Klasyfikacja Binarna** | KNN | Własna | k = 2 | 0.91 |
|  | Drzewa decyzyjne | Własna | depth = 7, minobs = 4, Gini, overfit = prune | 0.92 |
|  | Sieci NN | Własna | h = (4,6), lr = 0.01,  *iter =* 20000 | 0.91 |
|  | KNN | R - caret | k = 3 | 0.91 |
|  | Drzewa decyzyjne | R - Rpart | maxdepth = 3 | 0.92 |
|  | Sieci NN | R - nnet | h = 4 | 0.91 |
| **Klasyfikacja Wieloklasowa** | KNN | Własna | k = 20 | 0.88 |
|  | Drzewa decyzyjne | Własna | depth = 7, minobs = 2, Gini, overfit = prune,  cf = 0.2 | 0.75 |
|  | Sieci NN | Własna | h = (5,10) / (4,6,8,10),  lr = 0.01,  *iter =* 100000 | 0.92 |
|  | KNN | R - caret | k = 9 | 0.91 |
|  | Drzewa decyzyjne | R - Rpart | maxdepth = 8 | 0.78 |
|  | Sieci NN | R - nnet | h = 7 | 0.96 |
| **Regresja** | KNN | Własna | k = 2 | 38.74 |
| **(MAE)** | Drzewa decyzyjne | Własna | depth = 5, minobs = 2, SS | 41.22 |
|  | Sieci NN | Własna | h = (8),  lr = 0.01,  *iter =* 100000 | 34.29 |
|  | KNN | R - caret | k = 3 | 31.54 |
|  | Drzewa decyzyjne | R - Rpart | maxdepth = 3 | 49.96 |
|  | Sieci NN | R - nnet | h = 8 | 0.0016 |

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

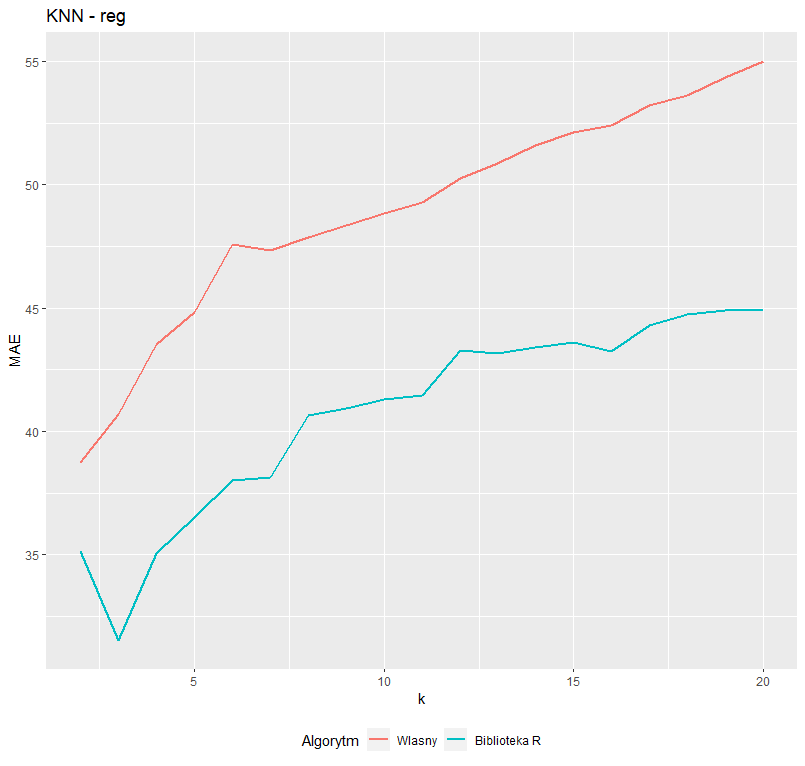




KNN – Wyniki dla własnej implementacji  
**Klasyfikacja Wieloklasowa**

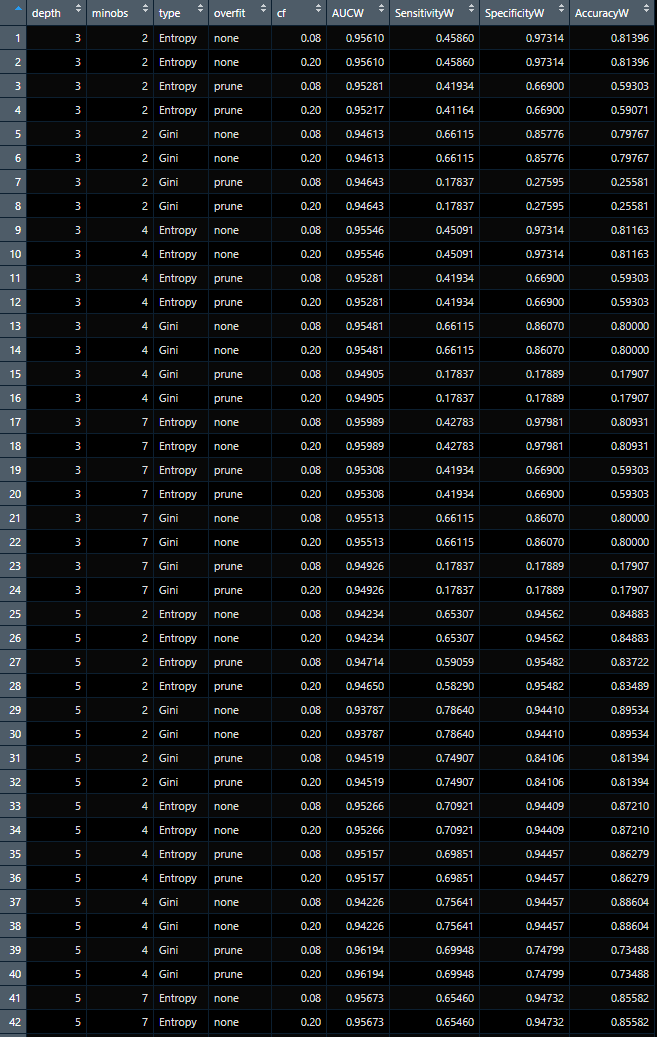
 

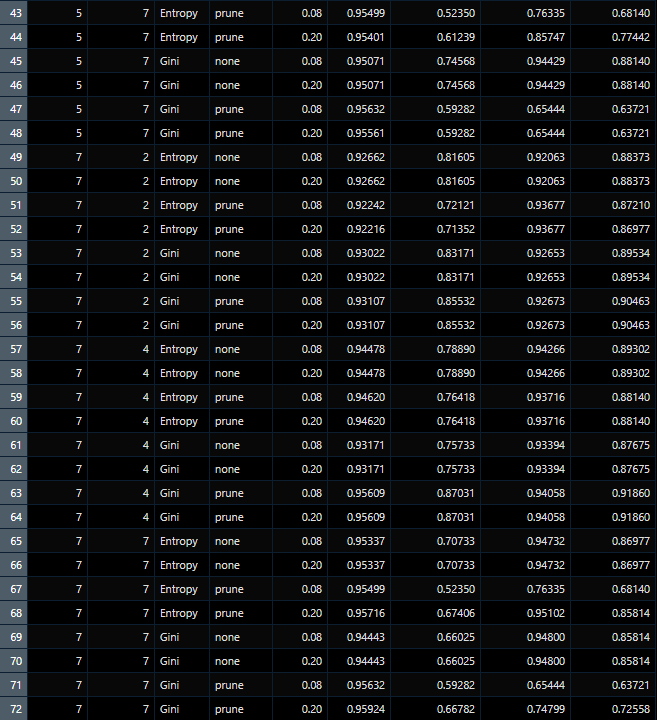
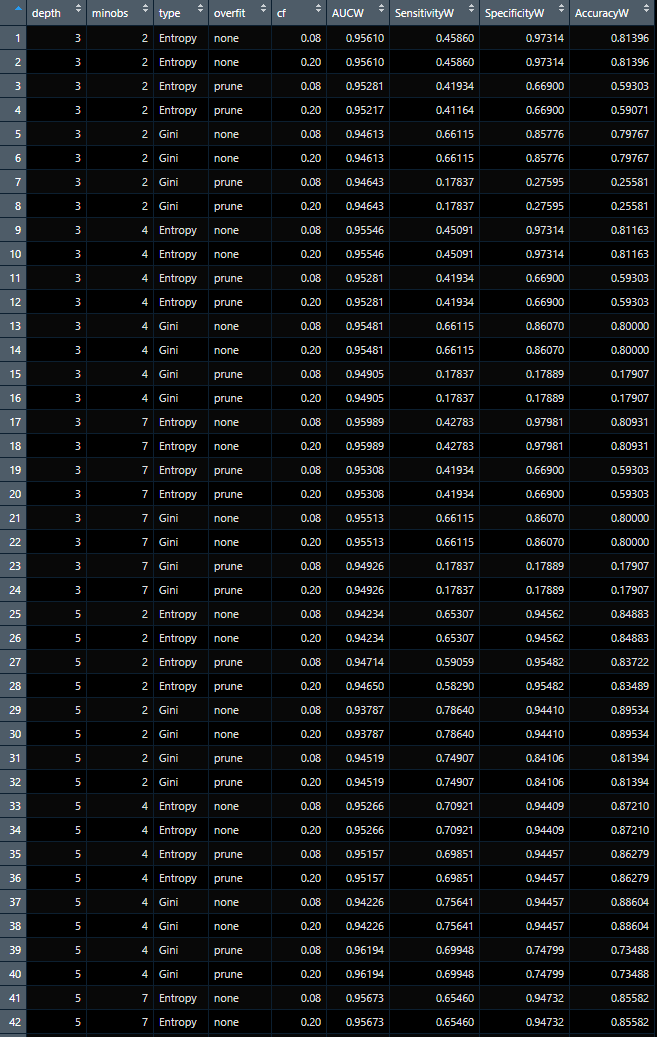
KNN – Wyniki dla własnej implementacji  
**Regresja**

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

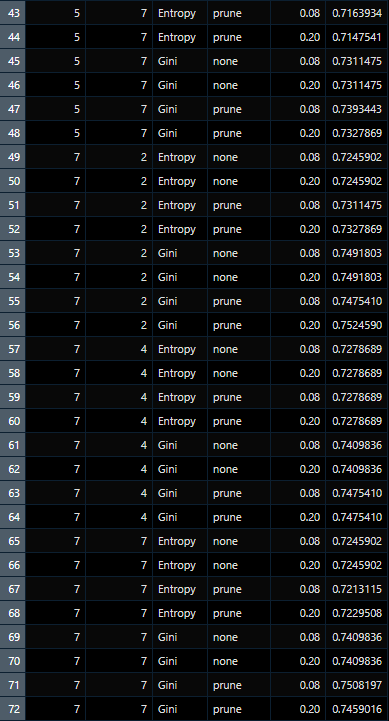
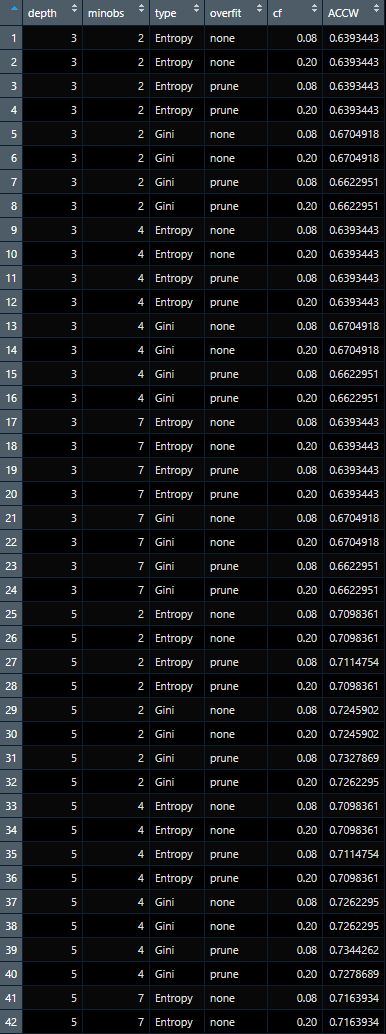
**Depth = c(3,5,7), minobs = c(2,4,7), type = c('Entropy', 'Gini'),   
overfit = c('none', 'prune'), cf = c(0.08, 0.2)**





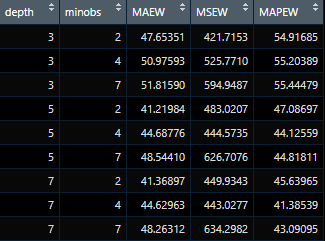
Drzewa Decyzyjne (własna implementacja)  
**Klasyfikacja Wieloklasowa   
Depth = c(3,5,7), minobs = c(2,4,7), type = c('Entropy', 'Gini'),   
overfit = c('none', 'prune'), cf = c(0.08, 0.2)**

Obraz zawierający budynek, wieża

Opis wygenerowany automatycznie

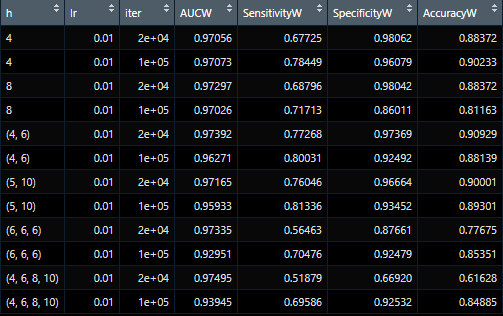
Drzewa Decyzyjne (własna implementacja)  
**Regresja**

**Depth = c(3,5,7), minobs = c(2,4,7), type = c(‘SS’), overfit = c('none')**



Sieci Neuronowe - własna implementacja  
**Klasyfikacja Binarna**

**h = c(4), c(8), c(4,6), c(5,10), c(6,6,6), c(4,6,8,10)  
lr = 0.01 / iteracje = c(20000, 100000)**



Sieci Neuronowe - własna implementacja  
**Klasyfikacja Wieloklasowa**

**h = c(4), c(8), c(4,6), c(5,10), c(6,6,6), c(4,6,8,10)  
lr = 0.01 / iteracje = c(20000, 100000)**

Obraz zawierający tekst, kalkulator

Opis wygenerowany automatycznie

Sieci Neuronowe - własna implementacja  
**Regresja**

**h = c(8), c(4,6), c(6,6,6), c(4,6,8,10)  
lr = 0.01 / iteracje = c(20000, 100000)**



**Podsumowanie i wnioski**

Zaimplementowane modele KNN, Drzew decyzyjnych i Sieci neuronowych bazują na podstawowej idei tych algorytmów. Nie zostały wprowadzone żadne dodatkowe algorytmy, które powodowałyby zwiększenie celności predykcji tych algorytmów. Sieci neuronowe zostały uogólnione dzięki czemu można wprowadzić dowolną ilość warstw o różnej ilości neuronów i sprawdzić ich działanie. Jeśli chodzi o kroswalidację, to w przypadku KNN zrównoleglono obliczenia, ponieważ pojedynczy model, w zależności od rozmiaru danych, model liczył się nawet kilka minut. Wykorzystano do tego funkcję „foreach” ustawiając odpowiednie parametry wejściowe, aby każda z instancji miała potrzebne dane do uzupełnienia tabeli wyjściowej. Dzięki temu siatka modeli z wybranymi parametrami, z uwzględnieniem kroswalidacji (kFold = 10 dla wszystkich obliczeń), policzyła się odpowiednio szybciej - w zależności od sprzętu.

W przypadku klasyfikacji binarnej wszystkie algorytmy zaimplementowane własnoręcznie osiągnęły bardzo podobne maksymalne wyniki w okolicy 0.90 - 0.92. Są to bardzo zadowalające wyniki w porównaniu do modeli z bibliotek R, które osiągnęły precyzję na tym samym poziomie. Dane do klasyfikacji wieloklasowej wykazały słabość algorytmu Drzew decyzyjnych. W przypadku obu implementacji drzewa osiągnęły najgorsze wyniki – 0.75 dla własnej implementacji oraz 0.78 dla Rpart. KNN osiągnął dokładność predykcji 0.88, a sieć neuronowa 0.92 . Tutaj funkcje z bibliotek okazały się nieco lepsze, z wynikami odpowiednio - 0.91 dla KNN oraz 0.96 dla sieci neuronowej. Ostatnim problemem była regresja. Modele zbudowane przy pomocy bibliotek R osiągnęły wyniki podobne do własnych implementacji. Najlepszym modelem dla regresji okazały się sieci neuronowe z biblioteki R, gdzie MAE wyniosło 0.0016 . Najlepszym własnym modelem jest w tym przypadku jest sieć neuronowa z MAE wynoszącym 34.29 . Pozostałe własne modele osiągnęły odpowiednio: KNN – 38.74 (caret = 31.54) oraz drzewa decyzyjne – 41.22 (Rpart = 49.96).

Wyniki, które zostały porównane dotyczą wybranych modeli i użytych parametrów. Może się okazać, że najlepsze parametry zostały pominięte w siatce hiper-parametrów. Podsumowując wyniki można śmiało stwierdzić, że własne implementacje nie odbiegają, aż tak bardzo od dopracowanych implementacji z bibliotek R. Można śmiało stwierdzić, że algorytmy z bibliotek są dużo lepiej zoptymalizowane pod kątem szybkości działania.