**Techniki eksploracji danych**

**Projekt końcowy**

**Zakres projektu**

Podczas przygotowania projektu należało sprawdzić poznane i zbadane algorytmy uczenia maszynowego:

1. K-najbliższych sąsiadów (KNN)
2. Drzewa decyzyjne
3. Sieci neuronowe (NN)

Docelowymi problemami były:

1. klasyfikacja binarna
2. klasyfikacja wieloklasowa
3. regresja

Podczas laboratoriów i prac domowych tworzono i zgłębiano działanie wcześniej wspomnianych algorytmów. Stworzone zostały funkcje, które pozwoliły sprawdzić działanie algorytmów i wykonanie kroswalidacji w celu uśredniania wyników. Po wykonaniu obliczeń i predykcji dla każdego algorytmu należało również porównać wyniki z funkcjami z bibliotek. Aby wybrać najlepszy model stworzono siatkę hiper-parametrów.

Dane wykorzystane do projektu:

1. Klasyfikacja binarna - (wrzuciłbym nazwę zbioru)

Liczba wierszy: 470

Kolumn z danymi objaśniającymi: 16

1. Klasyfikacja wieloklasowa - Iris

Liczba wierszy: 150

Kolumn z danymi objaśniającymi: 4

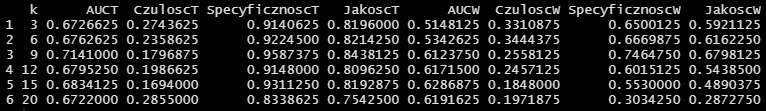
1. Regresja -- (wrzuciłbym nazwę zbioru)

Liczba wierszy: 414

Kolumn z danymi objaśniającymi: 6

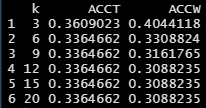
KNN – klasyfikacja binarna

* Wyniki uśrednione dla siatki parametrów: k = 3, 6, 9, 12, 15, 20 oraz kFold = 8



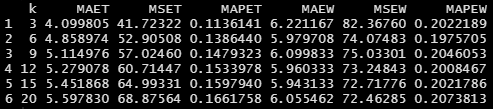
KNN – klasyfikacja wieloklasowa

* Wyniki uśrednione dla siatki parametrów: k = 3, 6, 9, 12, 15, 20 oraz kFold = 8



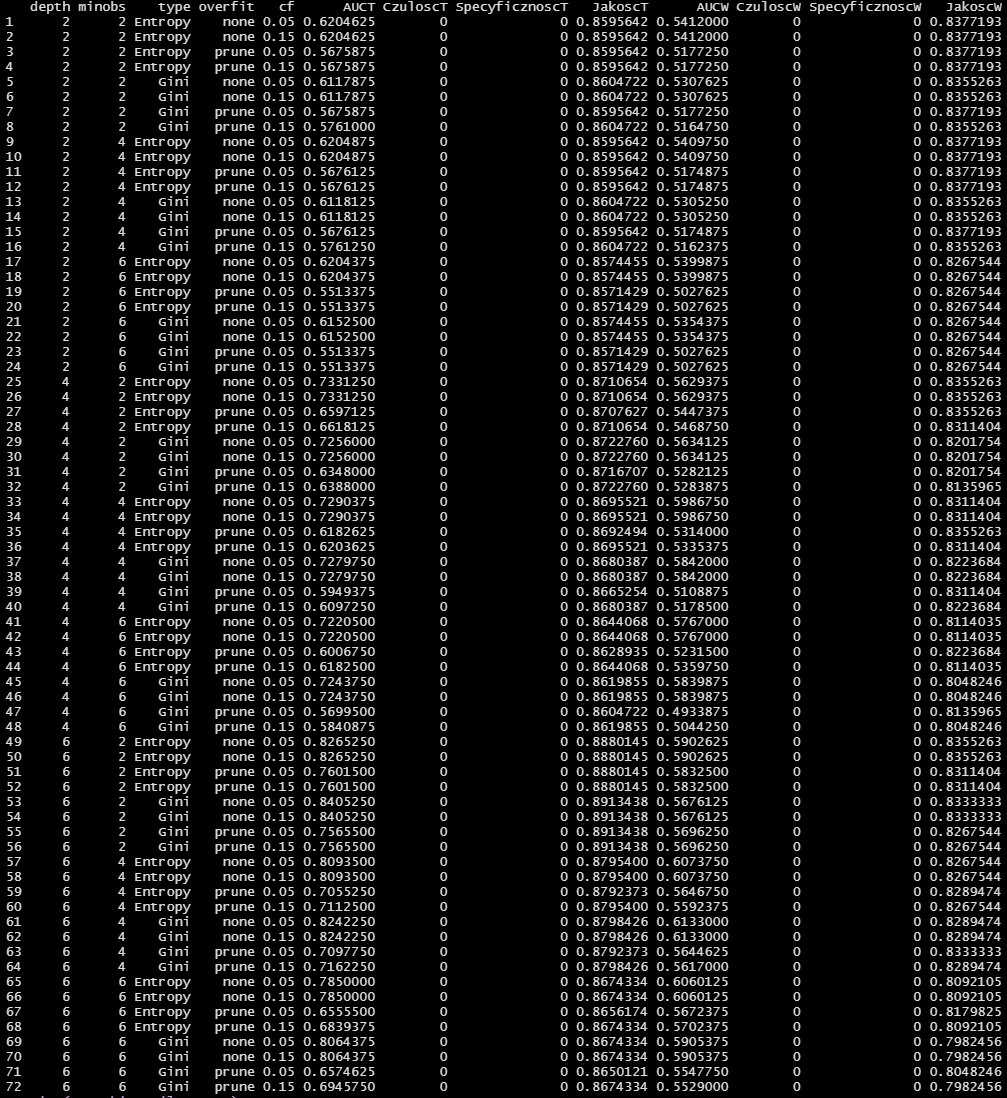
KNN – klasyfikacja binarna

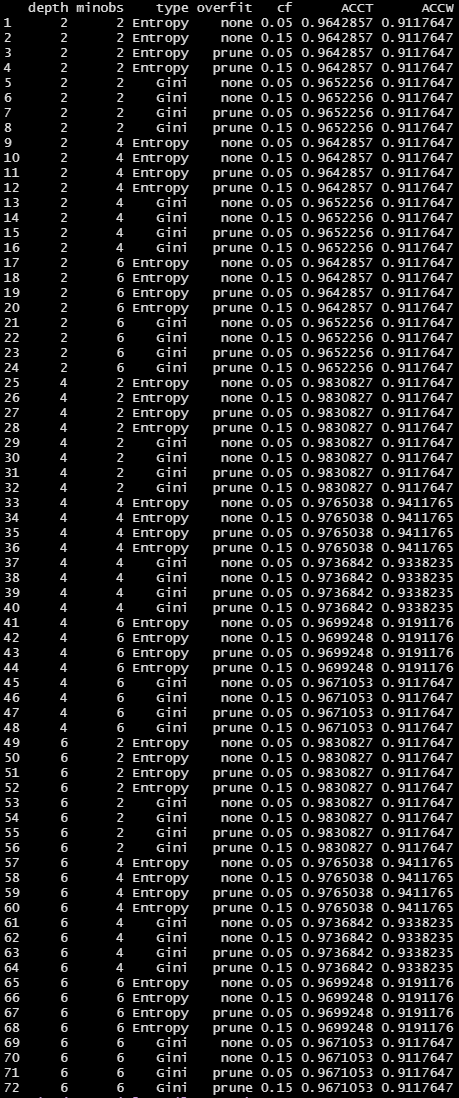
* Wyniki uśrednione dla siatki parametrów: k = 3, 6, 9, 12, 15, 20 oraz kFold = 8



Drzewa decyzyjne – klasyfikacja binarna

* Wyniki uśrednione dla siatki parametrów:   
  głębokość = 2, 4, 6 / minimum obserwacji = 2, 4, 6 / typ = Entropia i Gini /   
  przycinanie = tak (cf = 0.05, 0.15) i nie / kFold = 8



Drzewa decyzyjne – klasyfikacja wieloklasowa

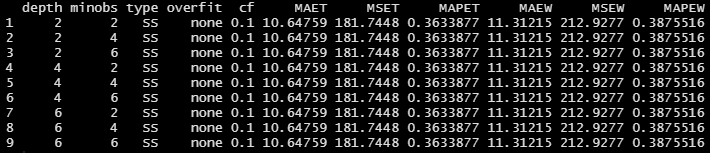
* Wyniki uśrednione dla siatki parametrów:

głębokość = 2, 4, 6

minimum obserwacji = 2, 4, 6   
typ = Entropia i Gini   
przycinanie = tak (cf = 0.05, 0.15) i nie kFold = 8

Drzewa decyzyjne – regresja

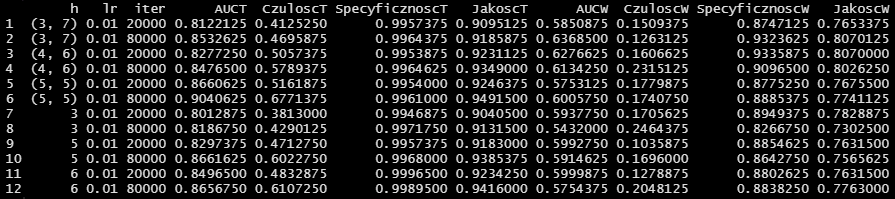
* Wyniki uśrednione dla siatki parametrów:   
  głębokość = 2, 4, 6 / minimum obserwacji = 2, 4, 6 / typ = SS /   
  przycinanie = nie / kFold = 8



Sieci neuronowe – klasyfikacja binarna

* Wyniki uśrednione dla siatki parametrów:   
  warstwy = c(3), c(5), c(6), c(3,7), c(4,6), c(5,5)  
  szybkość uczenia (lr) = 0.01

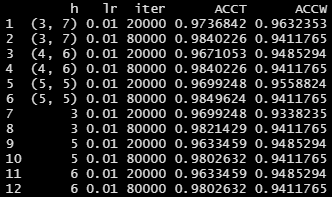
Iteracje (epoki) = 20000 i 80000  
kFold = 8



Sieci neuronowe – klasyfikacja wieloklasowa

* Wyniki uśrednione dla siatki parametrów:   
  warstwy = c(3), c(5), c(6), c(3,7), c(4,6), c(5,5)  
  szybkość uczenia (lr) = 0.01

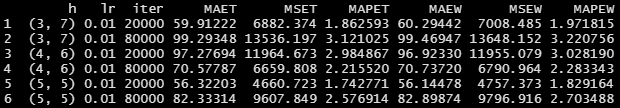
Iteracje (epoki) = 20000 i 80000  
kFold = 8



Sieci neuronowe – regresja

* Wyniki uśrednione dla siatki parametrów:   
  warstwy = c(3,7), c(4,6), c(5,5)  
  szybkość uczenia (lr) = 0.01

Iteracje (epoki) = 20000 i 80000  
kFold = 8

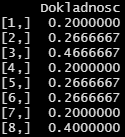


CARET - KNN – klasyfikacja binarna

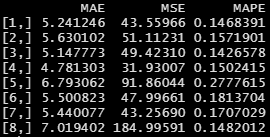
Obraz zawierający tekst

Opis wygenerowany automatycznie

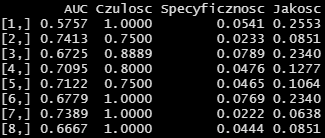
CARET - KNN – klasyfikacja wieloklasowa



CARET - KNN – regresja



RPART – Drzewa decyzyjne – klasyfikacja binarna

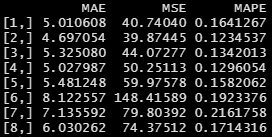


RPART – Drzewa decyzyjne – klasyfikacja wieloklasowa

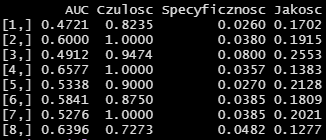
Obraz zawierający tekst

Opis wygenerowany automatycznie

RPART – Drzewa decyzyjne – regresja



NeuralNet – Sieci neuronowe (2 warstwy) – klas. binarna

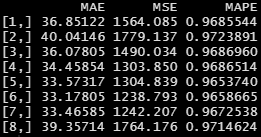


NeuralNet – Sieci neuronowe (2 warstwy) – klas. wieloklasowa

Obraz zawierający tekst

Opis wygenerowany automatycznie

NeuralNet – Sieci neuronowe (2 warstwy) – regresja



**Porównanie modeli własnych i wbudowanych**   
(wyniki dla walidacji)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Algorytm** | **Implementacja** | **Hiper-parametry** | **Ocena modelu**  **(Trafność)** |
| **Klasyfikacja Binarna** | KNN | Własna | k = 9 | 0.68 |
|  | Drzewa decyzyjne | Własna | depth = 2, minobs = 2, Entropy, overfit = none | 0.84 |
|  | Sieci NN | Własna | h = (3,7), lr = 0.01,  *iter =* 80000 | 0.81 |
|  | KNN | Biblioteka R - caret | k = 1 | 0.85 (max) |
|  | Drzewa decyzyjne | Biblioteka R - rpart | - | 0.26 (max) |
|  | Sieci NN | Biblioteka R - neuralnet | 2 warstwy | 0.26 (max) |
| **Klasyfikacja Wieloklasowa** | KNN | Własna | k = 3 | 0.40 |
|  | Drzewa decyzyjne | Własna | depth = 4, minobs = 4, Entropy, overfit = none | 0.94 |
|  | Sieci NN | Własna | h = (3,7), lr = 0.01,  *iter =* 20000 | 0.96 |
|  | KNN | Biblioteka R - caret | k = 1 | 0.47 (max) |
|  | Drzewa decyzyjne | Biblioteka R - rpart | - | 0.60 (max) |
|  | Sieci NN | Biblioteka R - neuralnet | 2 warstwy | 0.47 (max) |
| **Regresja** | KNN | Własna | k = 3 | 5.94 |
| **( MAE )** | Drzewa decyzyjne | Własna | depth = 2, minobs = 2, SS | 11.31 |
|  | Sieci NN | Własna | h = (5,5), lr = 0.01,  *iter =* 20000 | 56.14 |
|  | KNN | Biblioteka R - caret | k = 1 | 4.78 (max) |
|  | Drzewa decyzyjne | Biblioteka R - rpart | - | 4.69 (max) |
|  | Sieci NN | Biblioteka R - neuralnet | 2 warstwy | 33.17 (max) |

Wnioski

Wyniki, które wyszły dla poszczególnych algorytmów uczenia maszynowego różnią się od siebie. Trzeba wziąć pod uwagę fakt, że nie sprawdzono wszystkich możliwości hiper-parametrów – możliwe jest, że modele z implementacją własną mogłyby osiągać lepsze wyniki od wbudowanych w bibliotekach R. Jednakże ze względu na prostą implementację algorytmy jak KNN czy drzewa decyzyjne nie były zbyt szybkie w predykcji.

Porównując najlepsze wyniki własne implementacje nie są aż tak słabe w porównaniu do wbudowanych. W przypadku klasyfikacji binarnej najlepszym modelem okazał się KNN z biblioteki Caret (0.85), ale własne modele - drzewa decyzyjne (0.84) i sieć neuronowa (0.81), nie odbiegały aż tak mocno. Przy problemie klasyfikacji wieloklasowej to własna implementacja sieci neuronowej była najlepsza (0.96) a tuż za nią własne drzewa decyzyjne (0.94). Funkcje z bibliotek osiągały dość niskie wyniki. Regresja to zwycięstwo drzew decyzyjnych z biblioteki Rpart (MAE = 4.69) oraz KNN z Caret (MAE = 4.78). Najlepszą implementacją własną okazał się algorytm KNN z parametrem k=3 (MAE = 5.94).

Porównując te wyniki należałoby zbadać całą gamę hiper-parametrów, co w przypadku własnych implementacji zajęłoby dużo czasu. Należy też wziąć pod uwagę, że modele z bibliotek nie były w żaden sposób parametryzowane, co przy odpowiednich hiper-parametrach mogłoby zwiększyć ich precyzję.