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)

2) Klasyfikacja wieloklasowa:
   https://archive.ics.uci.edu/ml/datasets/seeds
   - Liczba obserwacji: 210
   - Liczba atrybutów: 7
   - Zmienna celu: 3 klasy (3 odmiany pszenicy)

3) 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

k	AUCT	CzuloscT	SpecyficznoscT	JakoscT	AUCW	CzuloscW	SpecyficznoscW	JakoscW
2	0.99264	0.84973	1.00000	0.95113	0.92800	0.74469	0.95692	0.89534
3	0.98921	0.73349	1.00000	0.91336	0.93490	0.68147	0.97039	0.88604
4	0.98516	0.64808	1.00000	0.88564	0.94273	0.58696	0.97352	0.86047
5	0.98235	0.55622	1.00000	0.85567	0.94327	0.59207	0.96070	0.84653
6	0.97959	0.51564	0.99774	0.84106	0.94364	0.53509	0.96392	0.83023
7	0.97811	0.66808	0.98654	0.88312	0.94256	0.55865	0.96392	0.83954
8	0.97626	0.66868	0.98432	0.88162	0.94254	0.54721	0.96392	0.83953
9	0.97481	0.62372	0.98506	0.86751	0.94393	0.51012	0.97335	0.83489
10	0.97413	0.59273	0.98692	0.85869	0.94569	0.52755	0.97315	0.83954
11	0.97338	0.55093	0.98840	0.84608	0.94539	0.54207	0.97315	0.84420
12	0.97247	0.55247	0.98954	0.84708	0.94805	0.53493	0.97638	0.84420
13	0.97181	0.63294	0.98729	0.87204	0.95286	0.53035	0.97960	0.84420
14	0.97113	0.60277	0.98729	0.86222	0.95372	0.50016	0.97960	0.83489
15	0.97019	0.57490	0.98729	0.85315	0.95580	0.47427	0.97960	0.82790

KNN – Wyniki dla własnej implementacji

Klasyfikacja Wieloklasowa

k	ACCT	ACCW
2	0.9701657	0.9388889
3	0.9690608	0.9666667
4	0.9707182	0.9722222
5	0.9574586	0.9555556
6	0.9662983	0.9500000
7	0.9552486	0.9555556
8	0.9535912	0.9555556
9	0.9403315	0.9611111
10	0.9414365	0.9555556
11	0.9348066	0.9500000
12	0.9397790	0.9555556
13	0.9325967	0.9500000
14	0.9375691	0.9500000
15	0.9370166	0.9500000

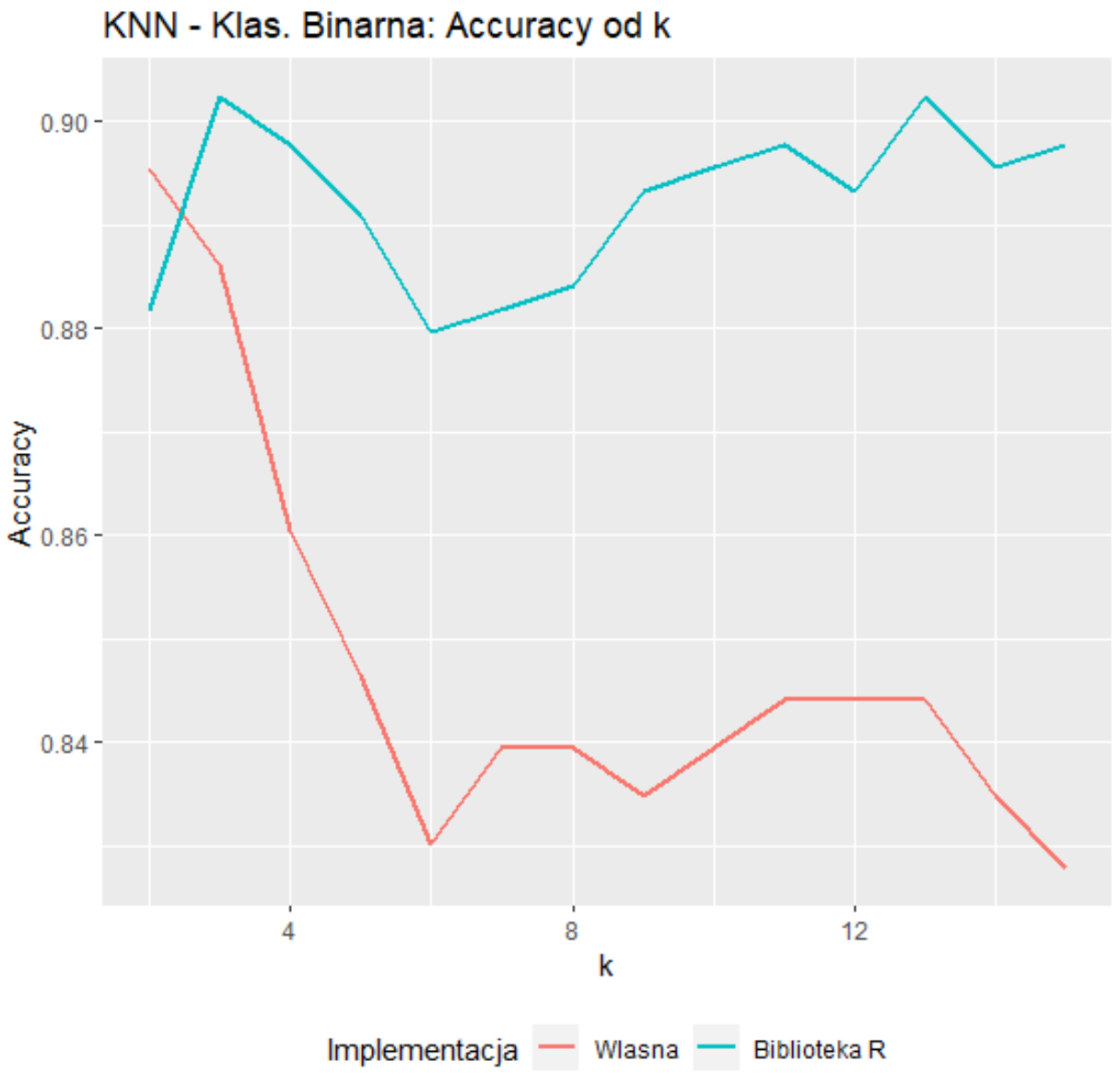
KNN – Wyniki dla własnej implementacji

Regresja

k	MAET	MSET	MAPET	MAEW	MSEW	MAPEW
2	16.87500	1165.929	0.1970260	38.79737	12052.33	0.3615248
3	21.90421	2575.403	0.2437053	39.25263	12859.78	0.3673015
4	25.84500	3549.011	0.2887305	39.87895	13363.14	0.3696979
5	27.96032	4293.957	0.3059486	40.41158	13204.65	0.3758878
6	29.75088	4937.173	0.3233505	42.41842	14097.60	0.3988828
7	32.00346	5714.079	0.3443611	42.95263	14578.13	0.3978475
8	33.04408	6075.597	0.3541601	43.43158	15238.49	0.3989798
9	33.91380	6499.445	0.3599101	43.66316	16010.17	0.4032374
10	34.42516	6851.680	0.3660996	44.15053	16981.64	0.4020563
11	35.14799	7355.128	0.3735588	44.40096	17732.91	0.4065181
12	35.67952	7724.387	0.3800890	45.22807	18485.48	0.4130954
13	36.53563	8238.422	0.3869427	46.08866	19153.11	0.4255978
14	37.54053	8651.320	0.3965016	45.66353	18954.51	0.4293609
15	38.18232	8974.916	0.4015892	45.64175	19104.13	0.4263452

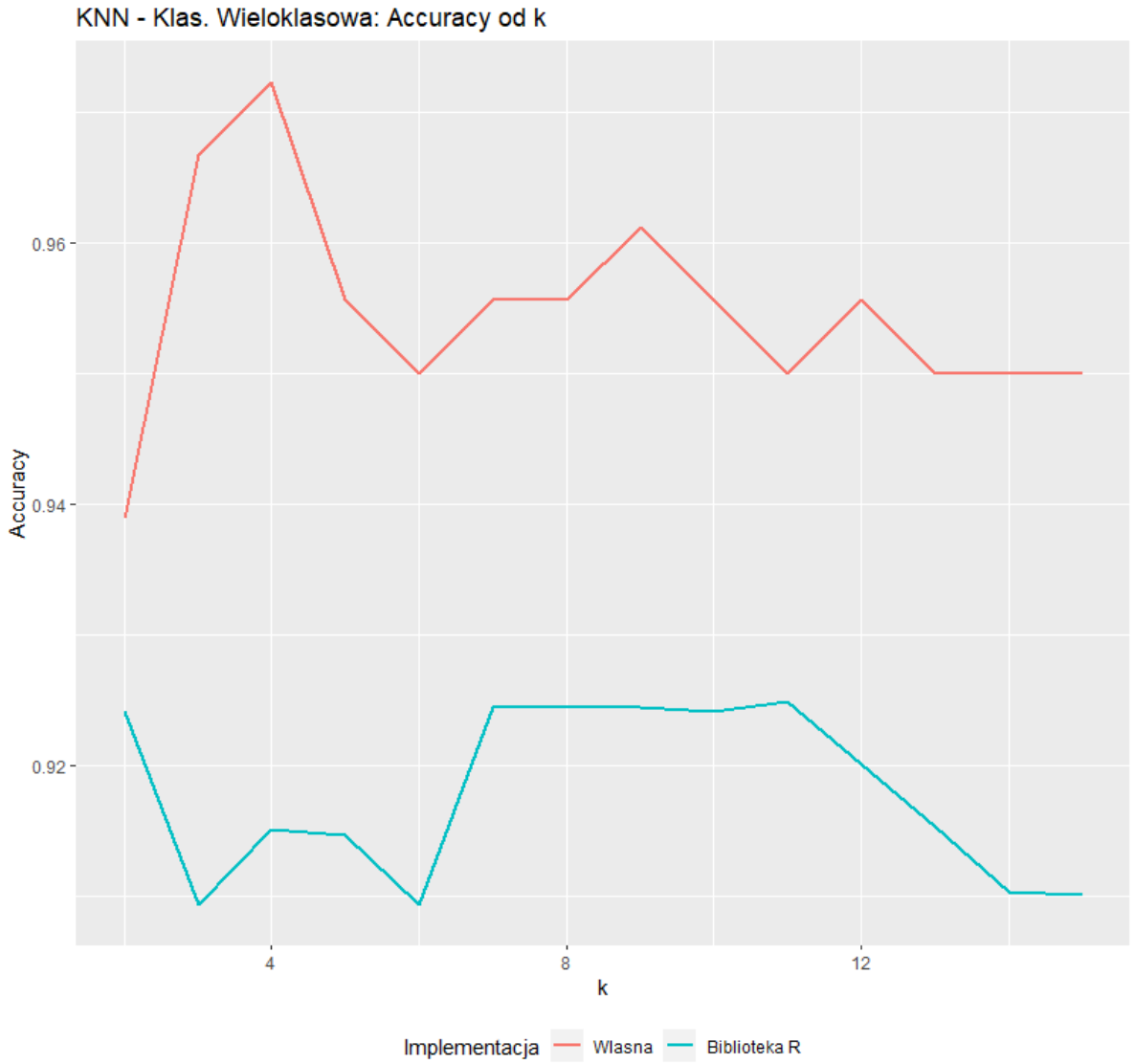
KNN – porównanie własnej implementacji z biblioteką R (knn)

Klasyfikacja Binarna



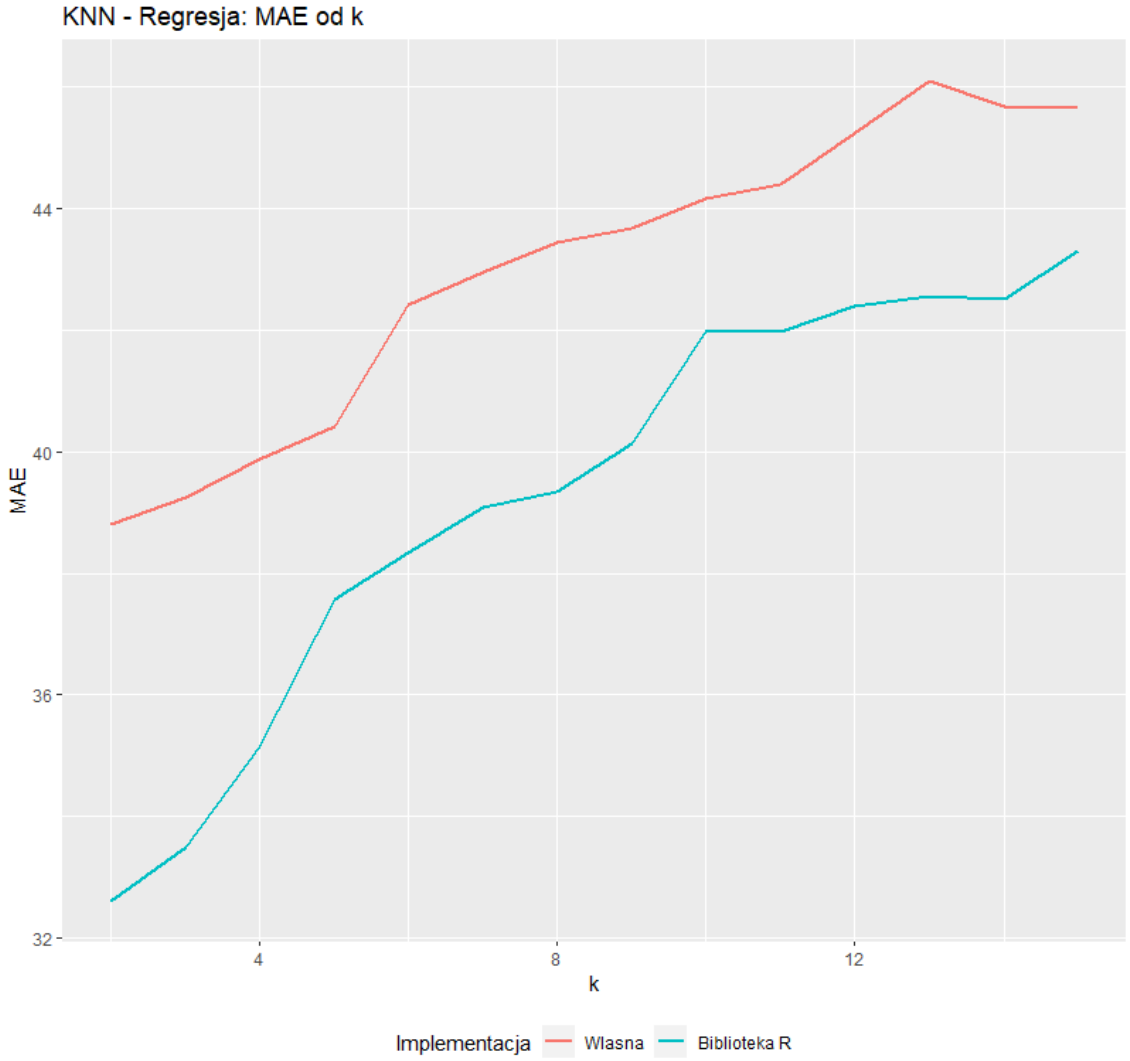
KNN – porównanie własnej implementacji z biblioteką R (knn)

Klasyfikacja Wieloklasowa



KNN – porównanie własnej implementacji z biblioteką R (knn)

Regresja



Algorytm KNN:

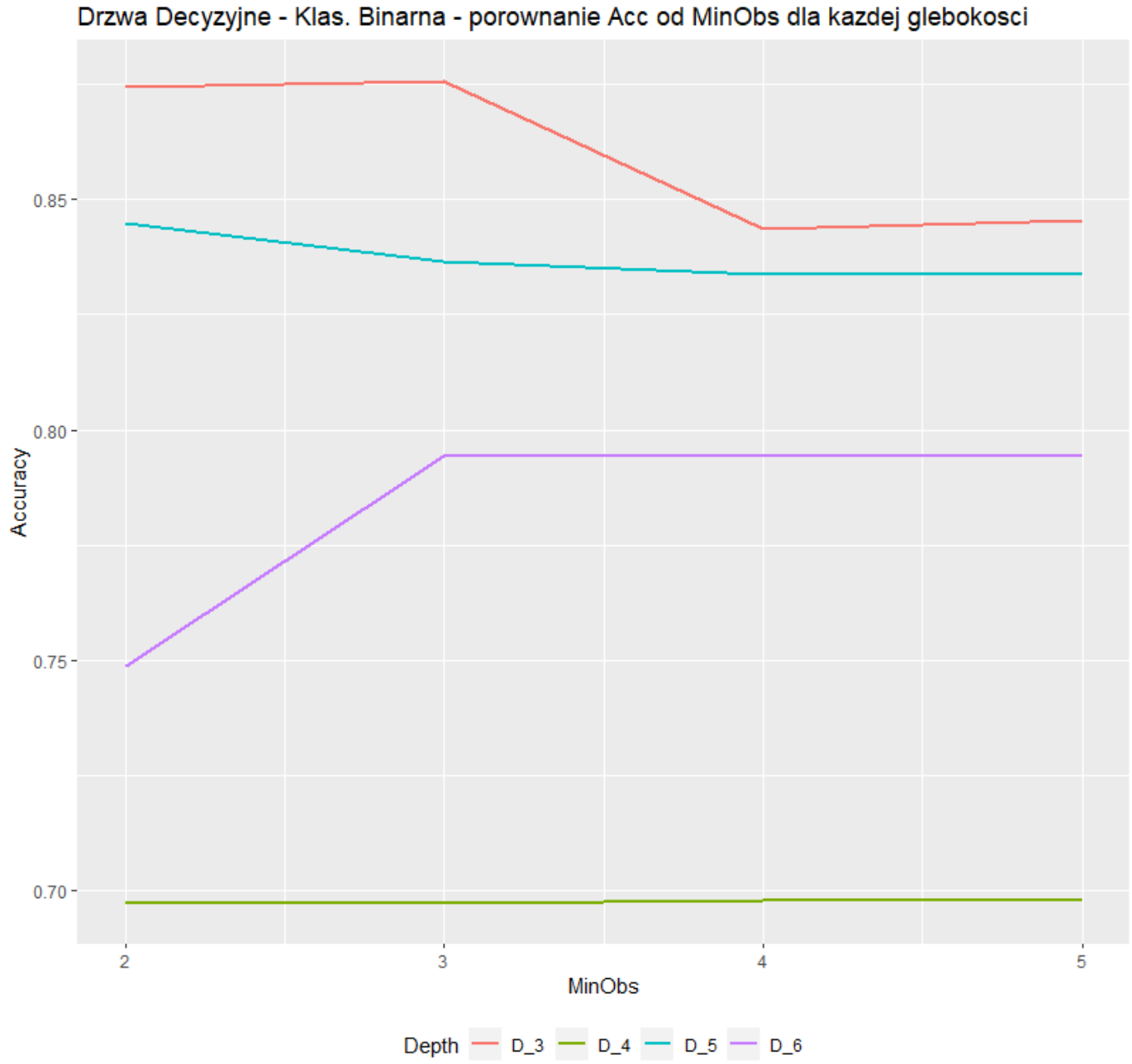
Porównanie algorytmów KNN, własnej implementacji i z biblioteki R pokazuje, że algorytmy nie różnią się aż tak drastycznie przy ocenie przynależności danej próbki do danej klasy. Widać oczywiście różnicę, czasem kilkuprocentową, w predykcji modeli, ale jak na własną ideową implementację takiego algorytmu 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.

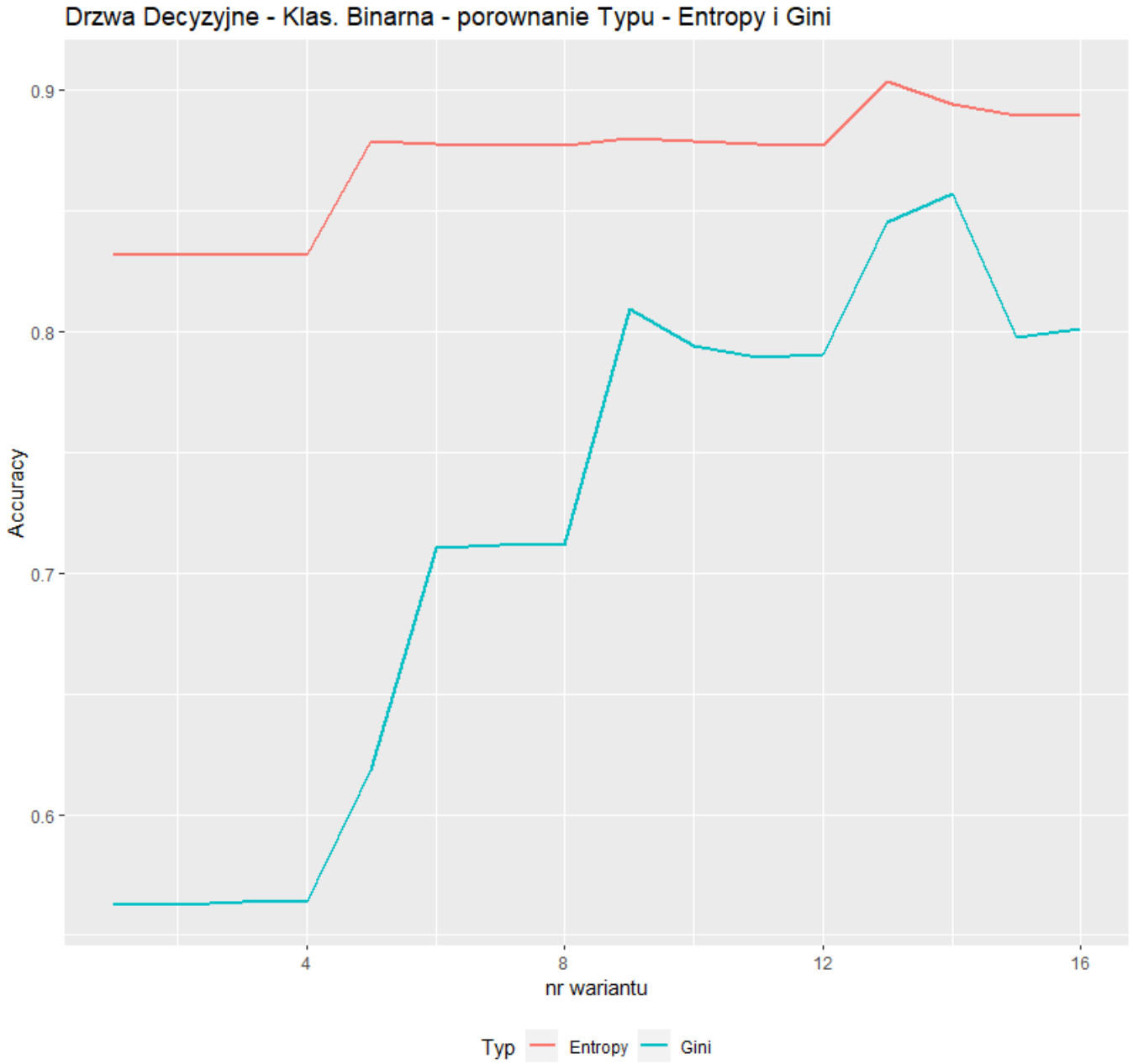
Drzewa Decyzyjne (własna implementacja)

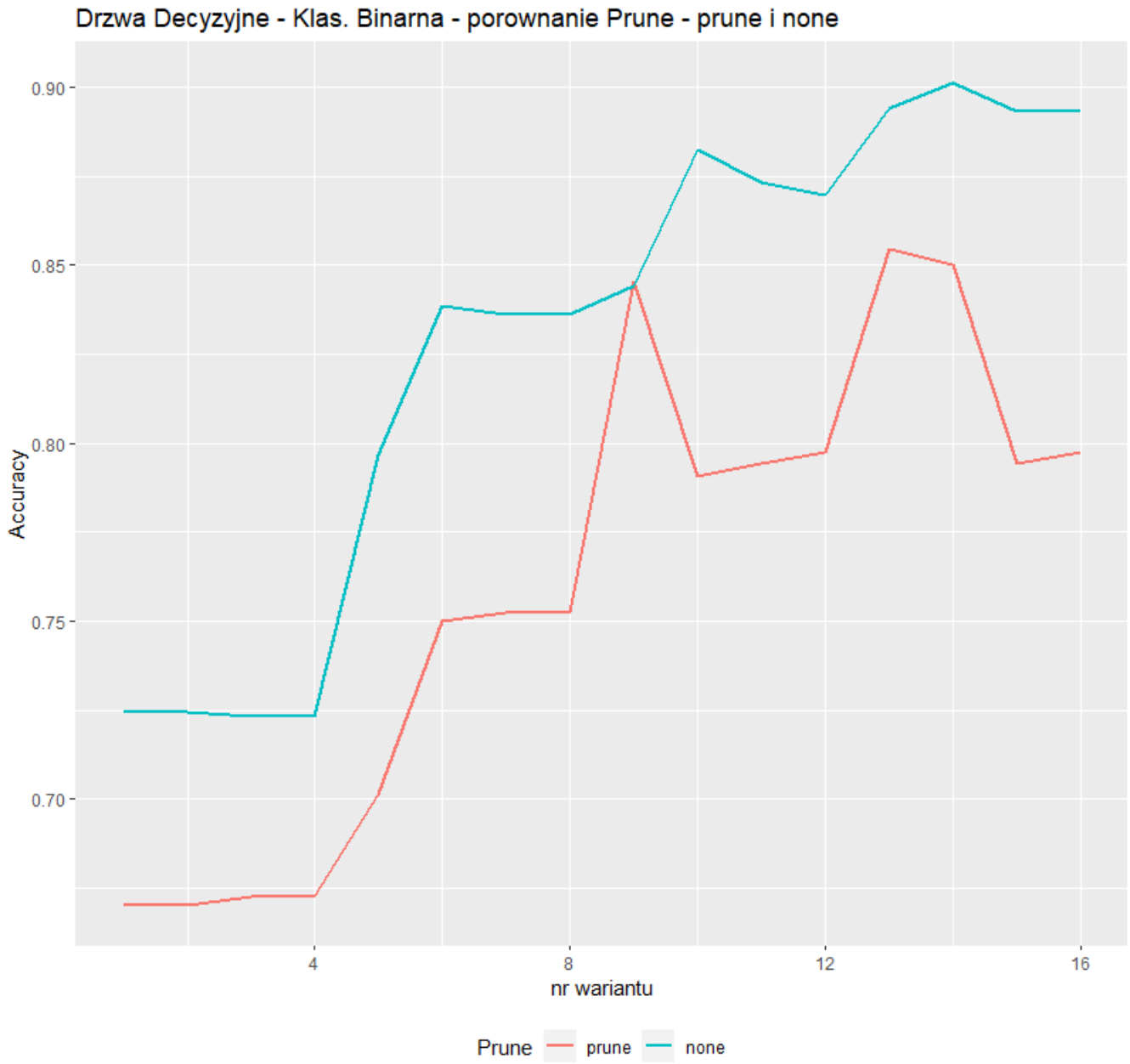
Klasyfikacja Binarna

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

depth	minobs	type	overfit	cf	AUCT	CzuloscT	SpecyficznoscT	JakoscT	AUCW	CzuloscW	SpecyficznoscW	JakoscW
3	2	Entropy	none	0.10	0.97412	0.56153	0.98914	0.85013	0.95267	0.55634	0.96249	0.84418
3	2	Entropy	none	0.25	0.97412	0.56153	0.98914	0.85013	0.95267	0.55634	0.96249	0.84418
3	2	Entropy	prune	0.10	0.94596	0.55442	0.88397	0.77633	0.93215	0.54667	0.86249	0.77210
3	2	Entropy	prune	0.25	0.94885	0.63332	0.98136	0.86777	0.93510	0.63953	0.95904	0.86745
3	2	Gini	none	0.10	0.97123	0.71965	0.97757	0.89371	0.95471	0.48367	0.65913	0.60465
3	2	Gini	none	0.25	0.97123	0.71965	0.97757	0.89371	0.95471	0.48367	0.65913	0.60465
3	2	Gini	prune	0.10	0.94502	0.68788	0.76147	0.73779	0.92859	0.45542	0.55291	0.52093
3	2	Gini	prune	0.25	0.94502	0.68788	0.76147	0.73779	0.92859	0.45542	0.55291	0.52093
3	3	Entropy	none	0.10	0.97403	0.55996	0.98914	0.84963	0.95267	0.55634	0.96249	0.84418
3	3	Entropy	none	0.25	0.97403	0.55996	0.98914	0.84963	0.95267	0.55634	0.96249	0.84418
3	3	Entropy	prune	0.10	0.94596	0.55442	0.88397	0.77633	0.93215	0.54667	0.86249	0.77210
3	3	Entropy	prune	0.25	0.94846	0.63489	0.98062	0.86777	0.93510	0.63953	0.95904	0.86745
3	3	Gini	none	0.10	0.97222	0.71965	0.97757	0.89371	0.95402	0.48367	0.65913	0.60465
3	3	Gini	none	0.25	0.97222	0.71965	0.97757	0.89371	0.95402	0.48367	0.65913	0.60465
3	3	Gini	prune	0.10	0.94502	0.68788	0.76147	0.73779	0.92859	0.45542	0.55291	0.52093
3	3	Gini	prune	0.25	0.94502	0.68788	0.76147	0.73779	0.92859	0.45542	0.55291	0.52093
3	4	Entropy	none	0.10	0.97391	0.55839	0.98914	0.84912	0.95267	0.55634	0.96249	0.84418
3	4	Entropy	none	0.25	0.97391	0.55839	0.98914	0.84912	0.95267	0.55634	0.96249	0.84418
3	4	Entropy	prune	0.10	0.94596	0.55442	0.88397	0.77633	0.93215	0.54667	0.86249	0.77210
3	4	Entropy	prune	0.25	0.94846	0.63489	0.98062	0.86777	0.93510	0.63953	0.95904	0.86745
3	4	Gini	none	0.10	0.97290	0.71493	0.97832	0.89271	0.95429	0.47778	0.65913	0.60232
3	4	Gini	none	0.25	0.97290	0.71493	0.97832	0.89271	0.95429	0.47778	0.65913	0.60232
3	4	Gini	prune	0.10	0.94413	0.68946	0.76000	0.73729	0.92959	0.46899	0.55291	0.52558
3	4	Gini	prune	0.25	0.94413	0.68946	0.76000	0.73729	0.92959	0.46899	0.55291	0.52558
3	5	Entropy	none	0.10	0.97382	0.55682	0.98914	0.84862	0.95267	0.55634	0.96249	0.84418


Drzwa Decyzyjne - Klas. Binarna - porownanie Acc od MinObs dla kazdej glebokosci
0.85
0.80
0.75
0.70
Accuracy
2
3
4
5
MinObs
Depth
D_3
D_4
D_5
D_6


Drzwa Decyzyjne - Klas. Binarna - porownanie Typu - Entropy i Gini
0.9
0.8
0.7
0.6
Accuracy
4
8
12
16
nr wariantu
Typ
Entropy
Gini

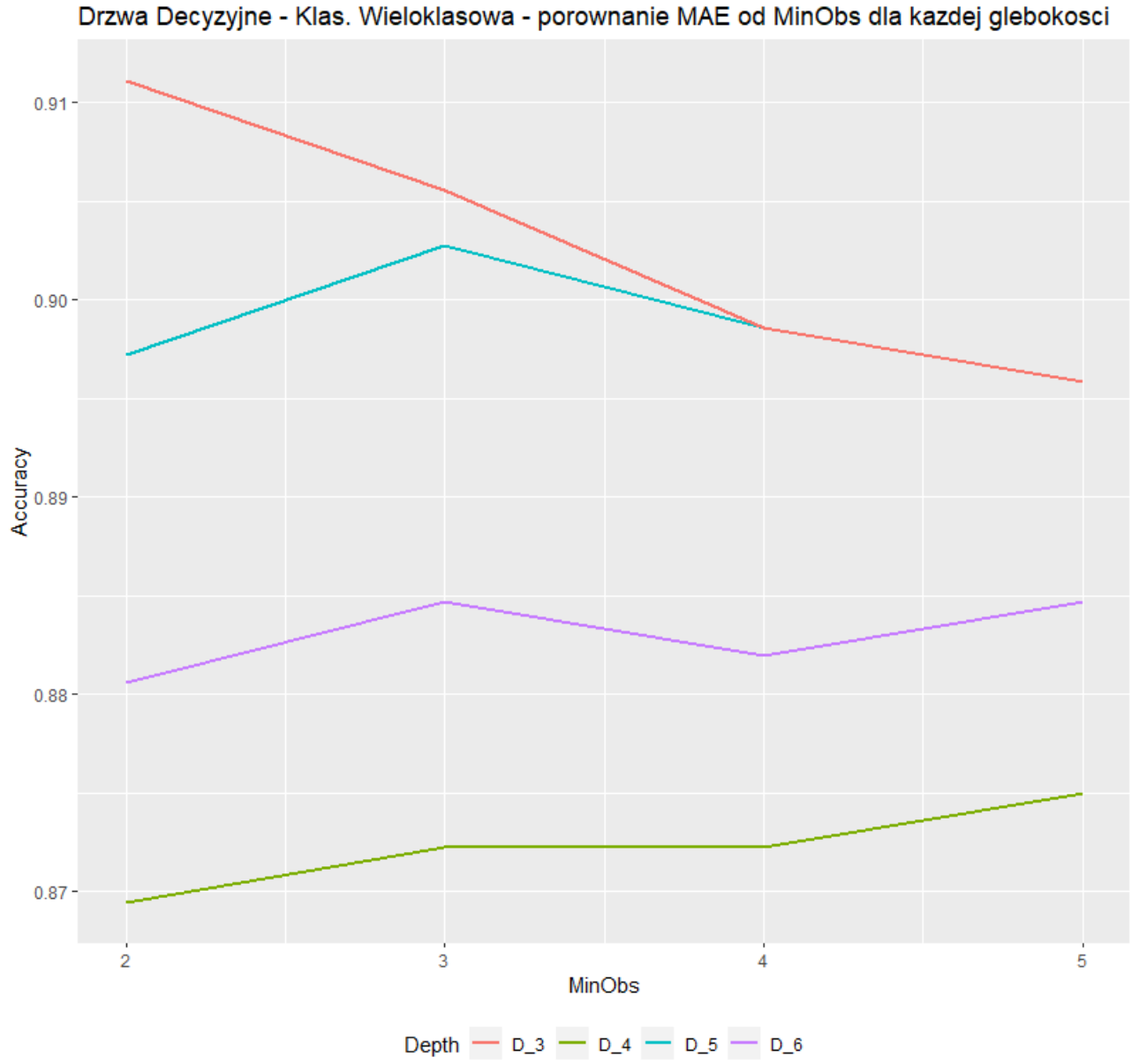

Drzwa Decyzyjne - Klas. Binarna - porownanie Prune - prune i none
0.90
0.85
0.80
0.75
0.70
Accuracy
4
8
12
16
nr wariantu
Prune
prune
none

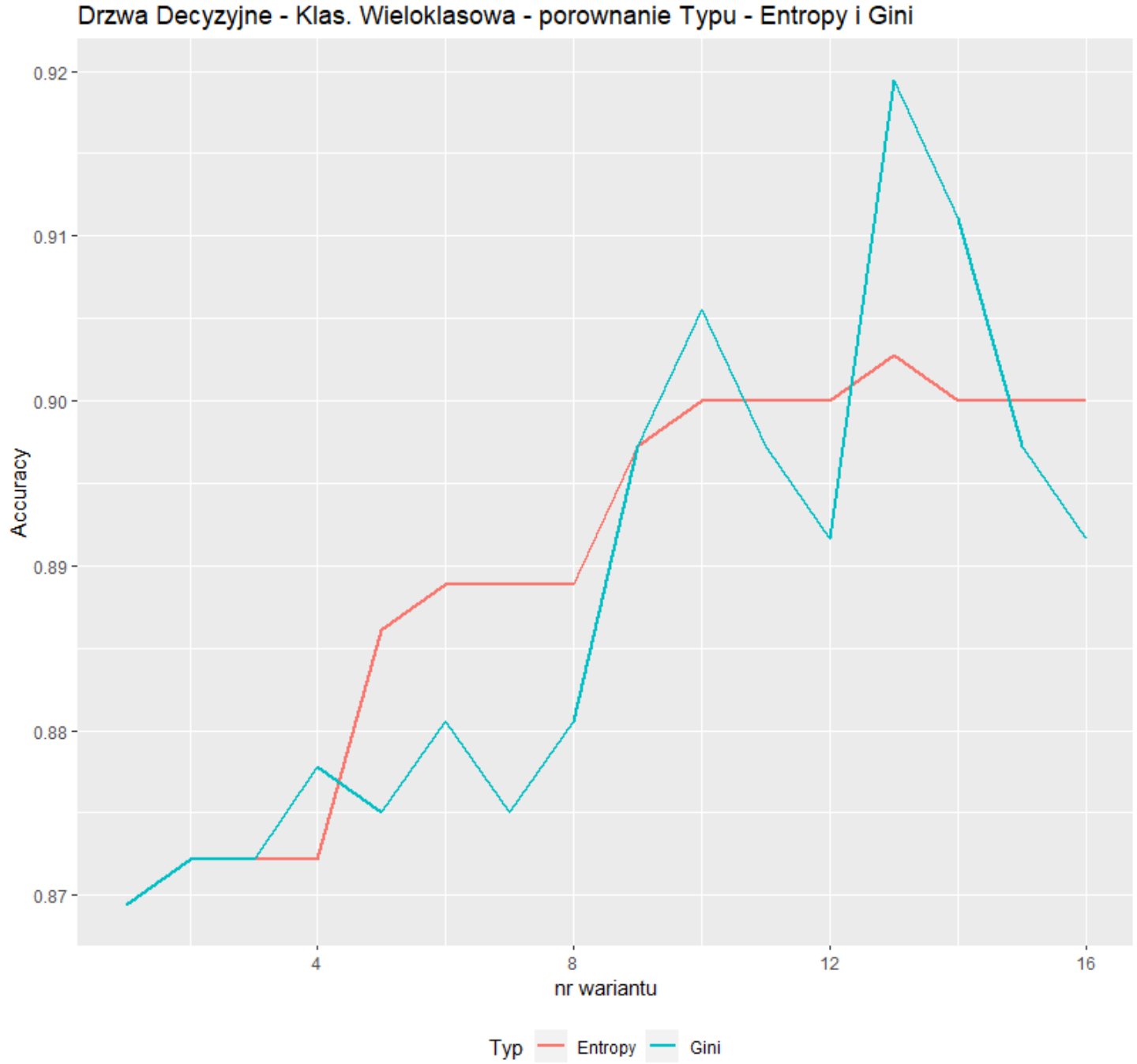
Drzewa Decyzyjne (własna implementacja)

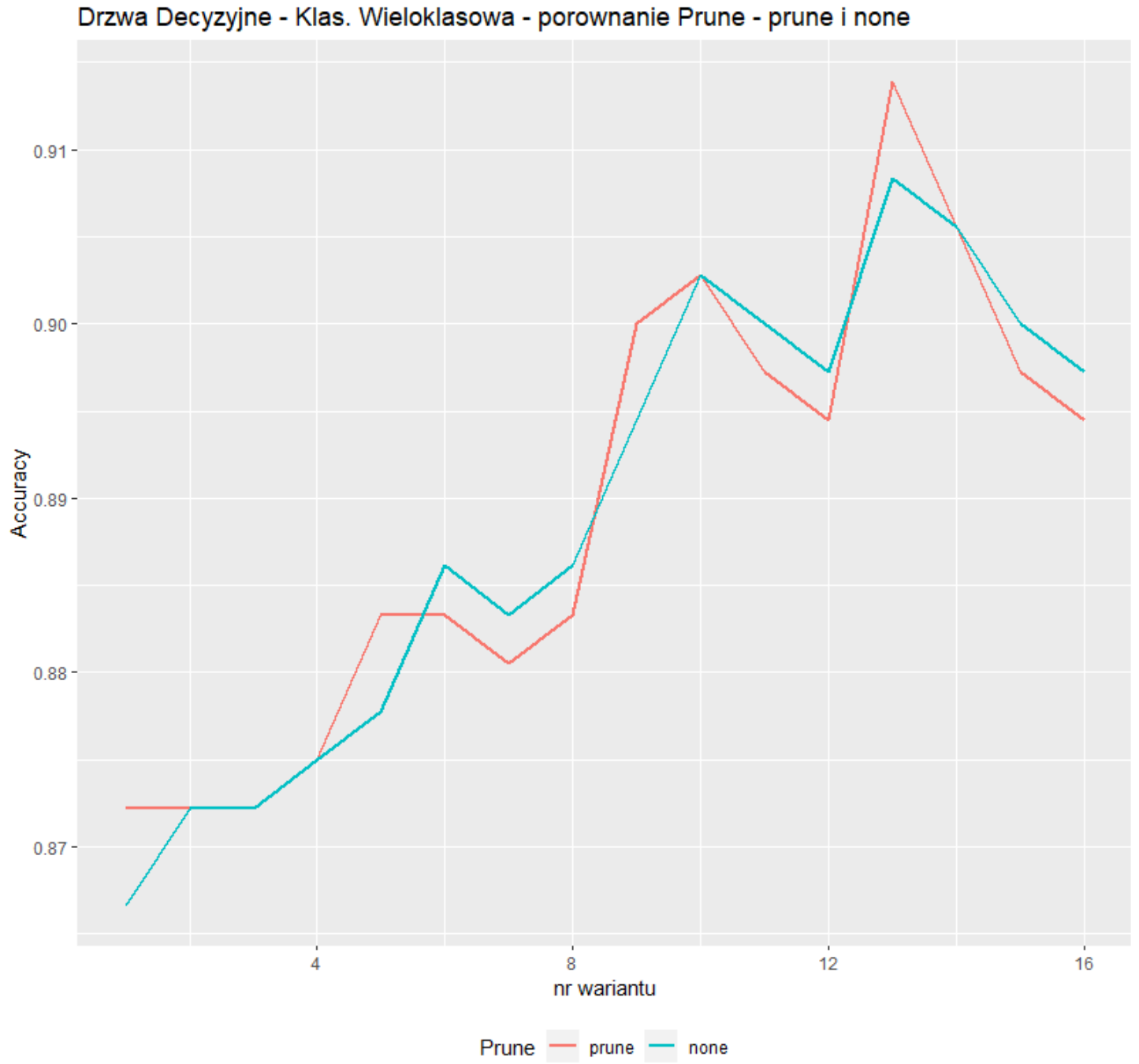
Klasyfikacja Wieloklasowa

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

depth	minobs	type	overfit	cf	ACCT	ACCW
3	2	Entropy	none	0.10	0.9375691	0.8666667
3	2	Entropy	none	0.25	0.9375691	0.8666667
3	2	Entropy	prune	0.10	0.9375691	0.8722222
3	2	Entropy	prune	0.25	0.9375691	0.8722222
3	2	Gini	none	0.10	0.9541436	0.8666667
3	2	Gini	none	0.25	0.9541436	0.8666667
3	2	Gini	prune	0.10	0.9541436	0.8722222
3	2	Gini	prune	0.25	0.9541436	0.8722222
3	3	Entropy	none	0.10	0.9375691	0.8722222
3	3	Entropy	none	0.25	0.9375691	0.8722222
3	3	Entropy	prune	0.10	0.9375691	0.8722222
3	3	Entropy	prune	0.25	0.9375691	0.8722222
3	3	Gini	none	0.10	0.9541436	0.8722222
3	3	Gini	none	0.25	0.9541436	0.8722222
3	3	Gini	prune	0.10	0.9541436	0.8722222
3	3	Gini	prune	0.25	0.9541436	0.8722222
3	4	Entropy	none	0.10	0.9375691	0.8722222
3	4	Entropy	none	0.25	0.9375691	0.8722222
3	4	Entropy	prune	0.10	0.9375691	0.8722222
3	4	Entropy	prune	0.25	0.9375691	0.8722222
3	4	Gini	none	0.10	0.9541436	0.8722222
3	4	Gini	none	0.25	0.9541436	0.8722222
3	4	Gini	prune	0.10	0.9541436	0.8722222
3	4	Gini	prune	0.25	0.9541436	0.8722222
3	5	Entropy	none	0.10	0.9375691	0.8722222


Drzwa Decyzyjne - Klas. Wieloklasowa - porownanie MAE od MinObs dla kazdej glebokosci
Accuracy
0.91
0.90
0.89
0.88
0.87
2
3
4
5
MinObs
Depth
D_3
D_4
D_5
D_6


Drzwa Decyzyjne - Klas. Wieloklasowa - porownanie Typu - Entropy i Gini
0.92
0.91
0.90
0.89
0.88
0.87
Accuracy
4
8
12
16
nr wariantu
Typ
Entropy
Gini

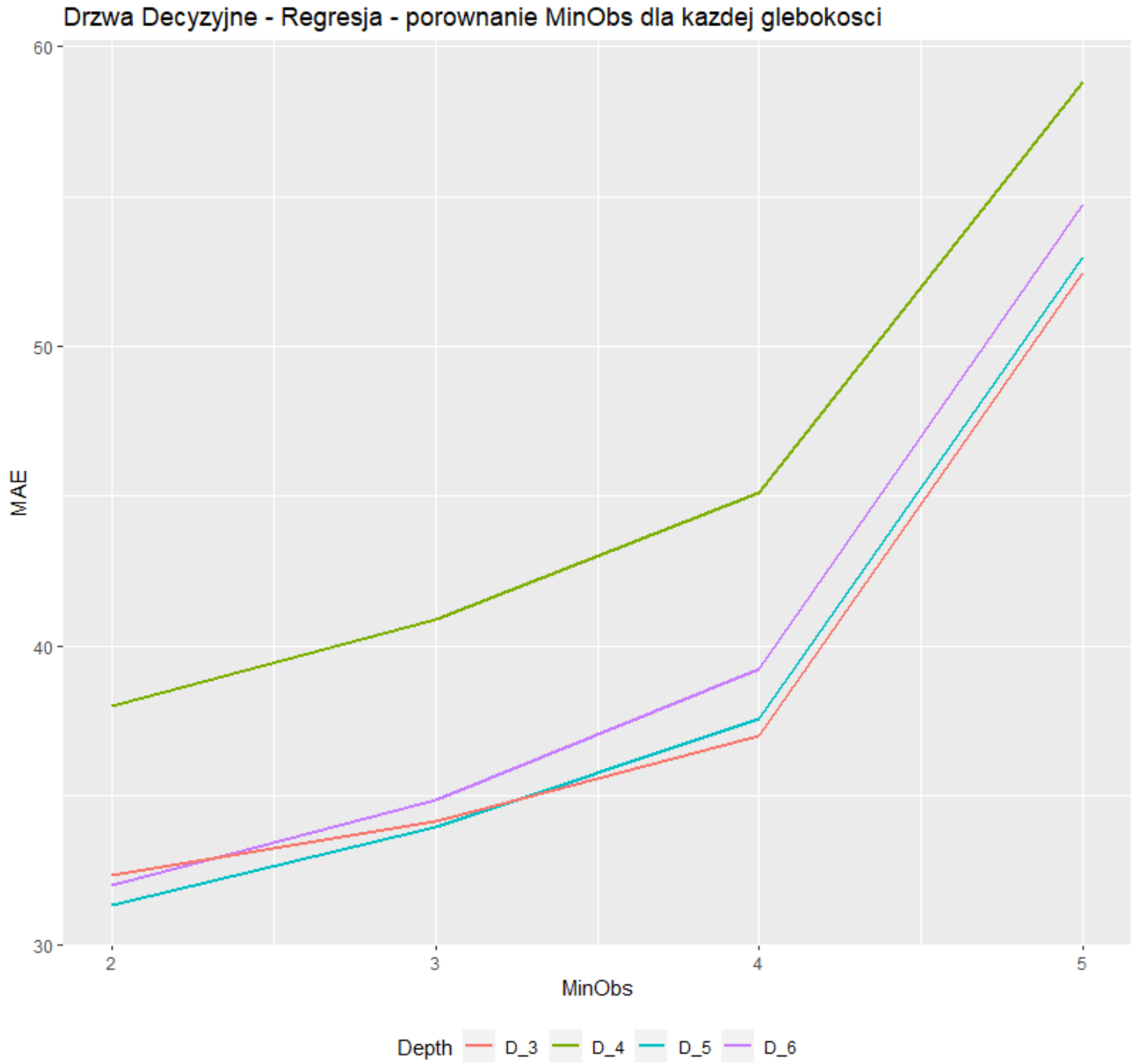

Drzwa Decyzyjne - Klas. Wieloklasowa - porownanie Prune - prune i none
Accuracy
0.91
0.90
0.89
0.88
0.87
4
8
12
16
nr wariantu
Prune
prune
none

Drzewa Decyzyjne (własna implementacja)

Regresja

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

depth	minobs	MAET	MSET	MAPET	MAEW	MSEW	MAPEW
3	2	29.22273	2352.7711	0.4835632	38.00887	4265.022	0.5983872
3	3	30.61864	2811.7016	0.4847847	40.87349	5600.408	0.6032672
3	4	32.27375	3561.9933	0.4843673	45.11837	8492.440	0.6014128
3	5	36.58214	6046.6751	0.4976791	58.81987	17996.564	0.7235285
4	2	22.38707	1415.8500	0.3652313	32.02818	3290.424	0.5004616
4	3	24.11268	1953.4572	0.3678649	34.88590	4532.639	0.5048508
4	4	25.58308	2670.5328	0.3645920	39.20671	7449.640	0.5020594
4	5	31.24479	5349.6119	0.3854037	54.74393	17295.341	0.6388386
5	2	17.85441	925.7628	0.2916760	31.36320	3224.234	0.4276909
5	3	20.36195	1546.0081	0.2978319	33.94078	4422.077	0.4351318
5	4	23.01693	2425.2645	0.3078559	37.57832	7242.330	0.4378658
5	5	29.48949	5216.7994	0.3354030	53.00016	17176.488	0.5834490
6	2	15.13265	785.4453	0.2350143	32.33945	3353.183	0.4166639
6	3	18.75090	1469.3087	0.2519161	34.17418	4462.631	0.4127375
6	4	22.01302	2388.2307	0.2671966	37.00275	7225.073	0.4082736
6	5	28.69865	5190.8187	0.2988905	52.46102	17149.293	0.5526273

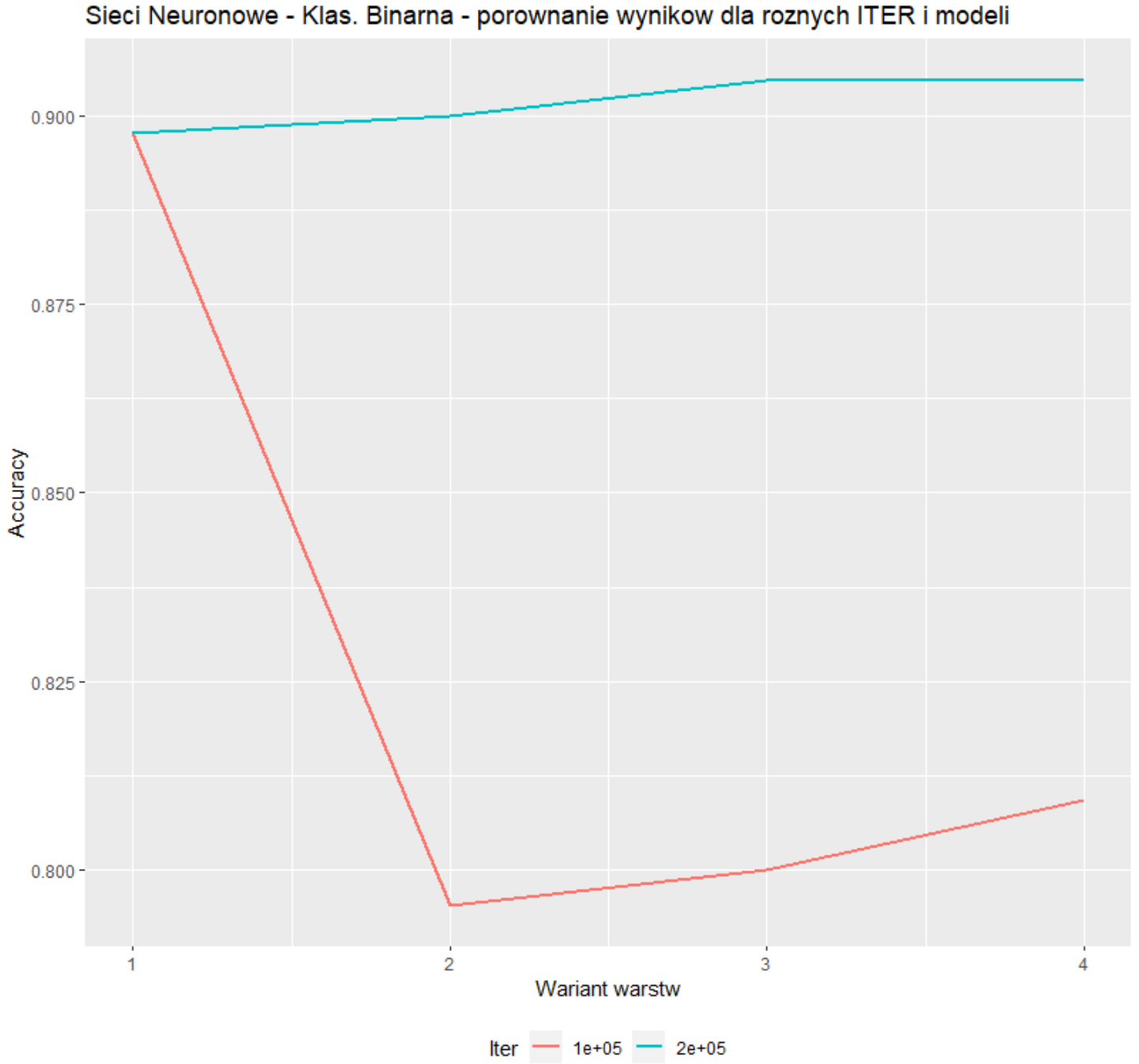

Drzwa Decyzyjne - Regresja - porownanie MinObs dla kazdej glebokosci
60
50
40
30
MAE
2
3
4
5
MinObs
Depth
D_3
D_4
D_5
D_6

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)

h	iter	AUCT	CzuloscT	SpecyficznoscT	JakoscT	AUCW	CzuloscW	SpecyficznoscW	JakoscW
(3, 4)	1e+05	0.96094	0.70039	0.98245	0.89068	0.95609	0.70676	0.97638	0.89767
(3, 4)	2e+05	0.95923	0.75072	0.97870	0.90453	0.95313	0.72863	0.96678	0.89769
(4, 4)	1e+05	0.95698	0.68802	0.98058	0.88539	0.95128	0.60728	0.87902	0.79536
(4, 4)	2e+05	0.95829	0.73707	0.98019	0.90101	0.95010	0.71751	0.97304	0.90000
(5, 5)	1e+05	0.96119	0.72425	0.98356	0.89923	0.95713	0.63266	0.87626	0.80001
(5, 5)	2e+05	0.95794	0.76082	0.97794	0.90730	0.94548	0.75679	0.96589	0.90465
(6, 6)	1e+05	0.96099	0.72503	0.98356	0.89949	0.95633	0.65923	0.87590	0.80931
(6, 6)	2e+05	0.95887	0.75306	0.98169	0.90731	0.94969	0.75590	0.96611	0.90465

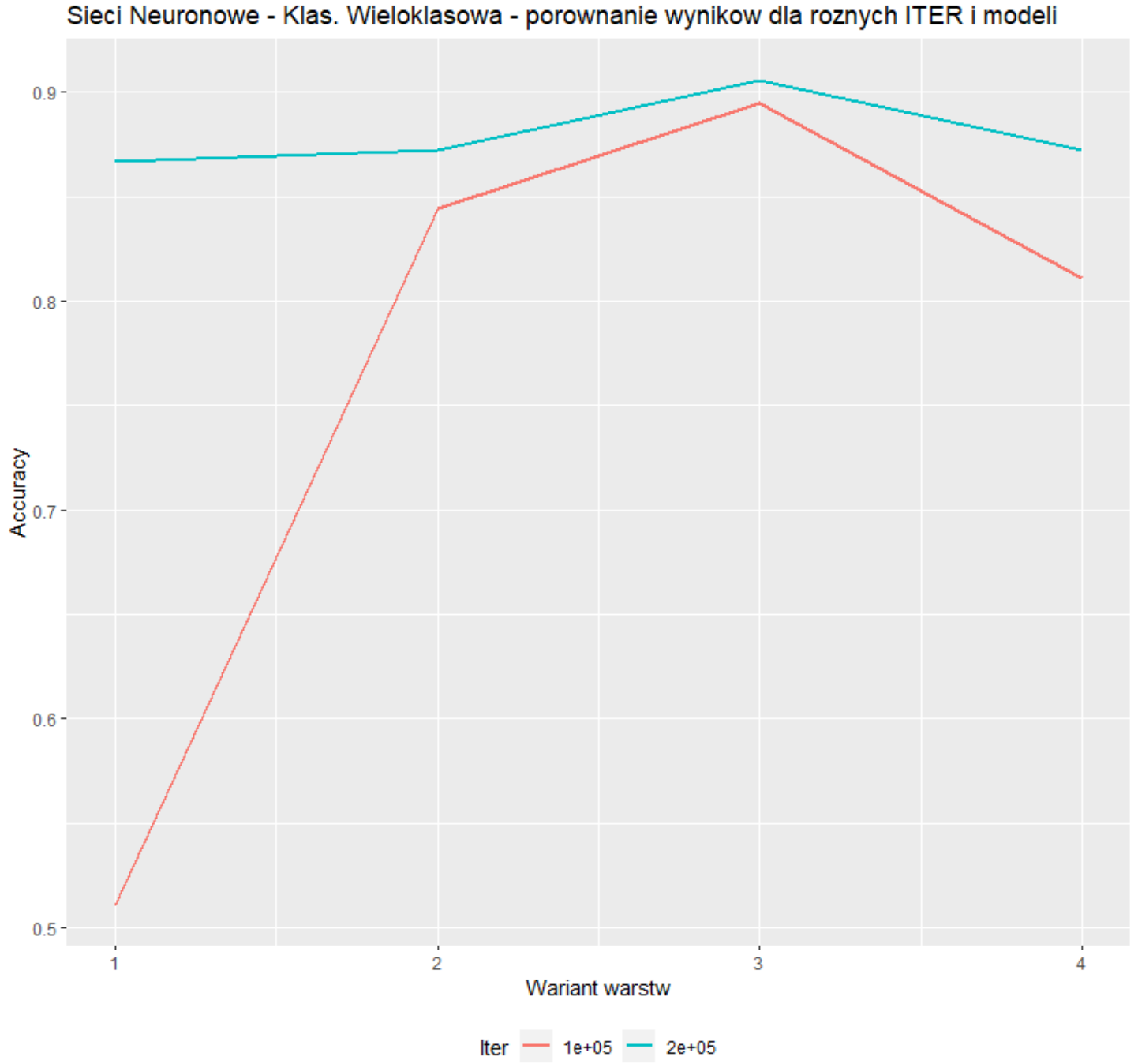

Sieci Neuronowe - Klas. Binarna - porownanie wynikow dla roznych ITER i modeli
0.900
0.875
0.850
0.825
0.800
Accuracy
1
2
3
4
Wariant warstw
Iter
1e+05
2e+05

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)

h	lr	iter	ACCT	ACCW
(3, 4)	0.001	1e+05	0.6049724	0.5111111
(3, 4)	0.001	2e+05	0.8635359	0.8666667
(4, 4)	0.001	1e+05	0.8414365	0.8444444
(4, 4)	0.001	2e+05	0.8889503	0.8722222
(5, 5)	0.001	1e+05	0.8839779	0.8944444
(5, 5)	0.001	2e+05	0.9143646	0.9055556
(6, 6)	0.001	1e+05	0.8430939	0.8111111
(6, 6)	0.001	2e+05	0.8867403	0.8722222

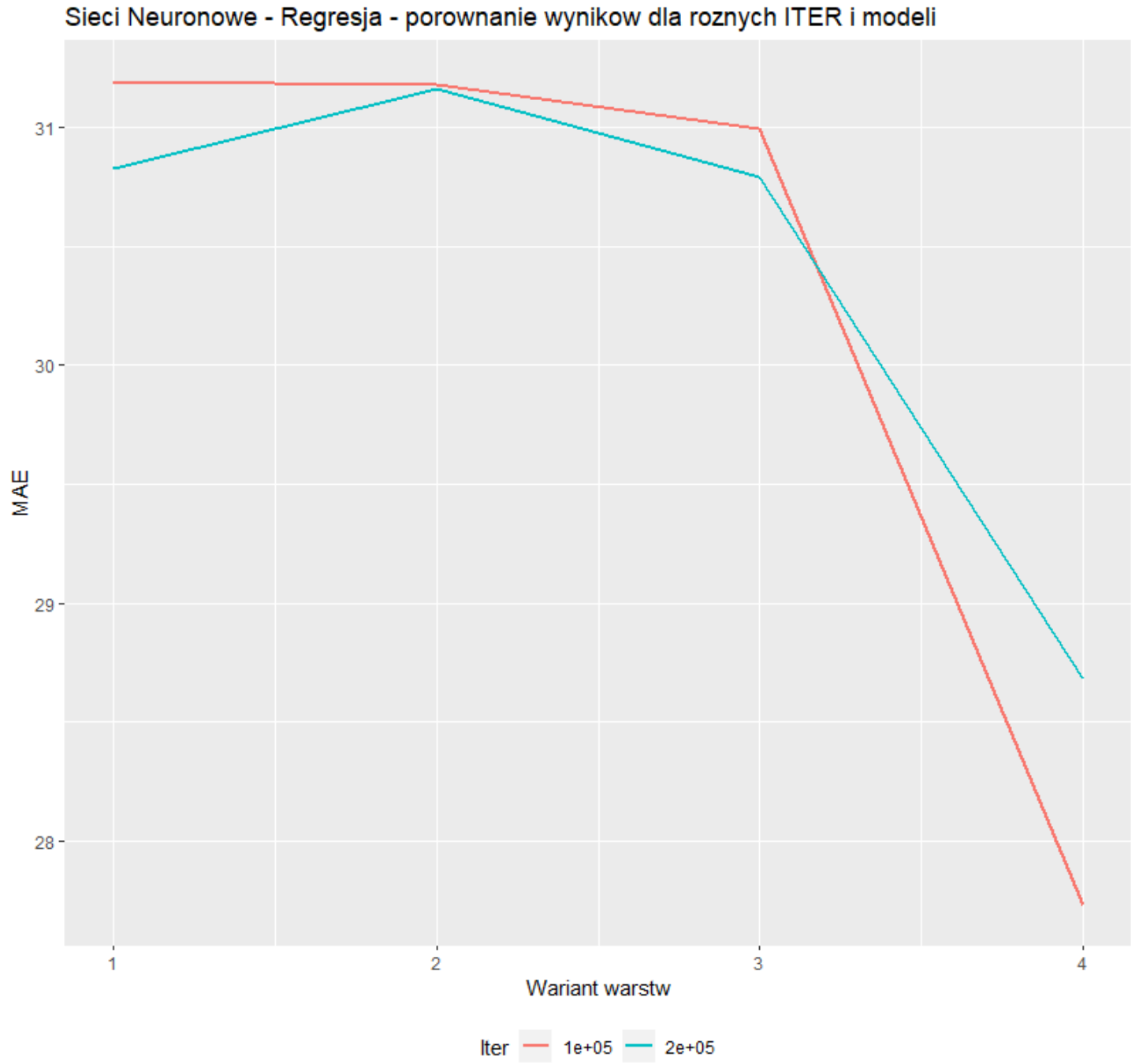

Sieci Neuronowe - Klas. Wieloklasowa - porownanie wynikow dla roznych ITER i modeli
0.9
0.8
0.7
0.6
0.5
Accuracy
1
2
3
4
Wariant warstw
Iter
1e+05
2e+05

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)

h	lr	iter	MAET	MSET	MAPET	MAEW	MSEW	MAPEW
(3, 4)	0.001	1e+05	24.16398	1613.416	0.3552689	31.19309	3585.234	0.4081838
(3, 4)	0.001	2e+05	23.11979	1332.094	0.3635762	30.82933	3462.394	0.4215786
(4, 4)	0.001	1e+05	24.26446	1571.302	0.3703998	31.18301	3414.982	0.4288213
(4, 4)	0.001	2e+05	22.97549	1297.611	0.3467906	31.16461	3638.375	0.3988075
(5, 5)	0.001	1e+05	23.19846	1398.962	0.3314272	30.99676	3666.906	0.3823512
(5, 5)	0.001	2e+05	22.61091	1218.453	0.3457922	30.79178	3694.317	0.3931596
(6, 6)	0.001	1e+05	22.68920	1185.368	0.3650767	27.73010	2376.196	0.3998898
(6, 6)	0.001	2e+05	22.41737	1130.176	0.3709302	28.67928	2744.125	0.4154235


Sieci Neuronowe - Regresja - porownanie wynikow dla roznych ITER i modeli
MAE
31
30
29
28
1
2
3
4
Wariant warstw
Iter
1e+05
2e+05

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.