**Projekt**

Projekt obejmuje wybór i wstępną analizę danych do problemów klasyfikacji binarnej, wieloklasowej i regresji. Kolejnym etapem jest użycie zbudowanych modeli KNN, drzew decyzyjnych i sieci neuronowych do predykcji. W tym celu wykonana została kroswalidacja modeli, czyli wielokrotne wylosowanie danych uczących i walidacyjnych, co pozwoli na uśrednienie statystyk takich jak dokładność, AUC, MAE czy MAPE. Wykonane zostały również badania wpływu hiper-parametrów na wyniki danych modeli.

Dane:

* Klasyfikacja binarna - Breast Cancer https://archive.ics.uci.edu/ml/datasets/Breast+Cancer
* Klasyfikacja wieloklasowa - Abalone

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

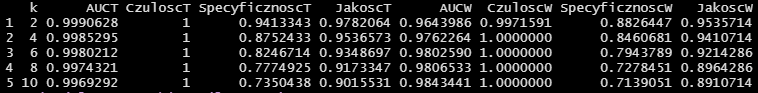
* Regresja - Automobile  
  <https://archive.ics.uci.edu/ml/datasets/Automobile>

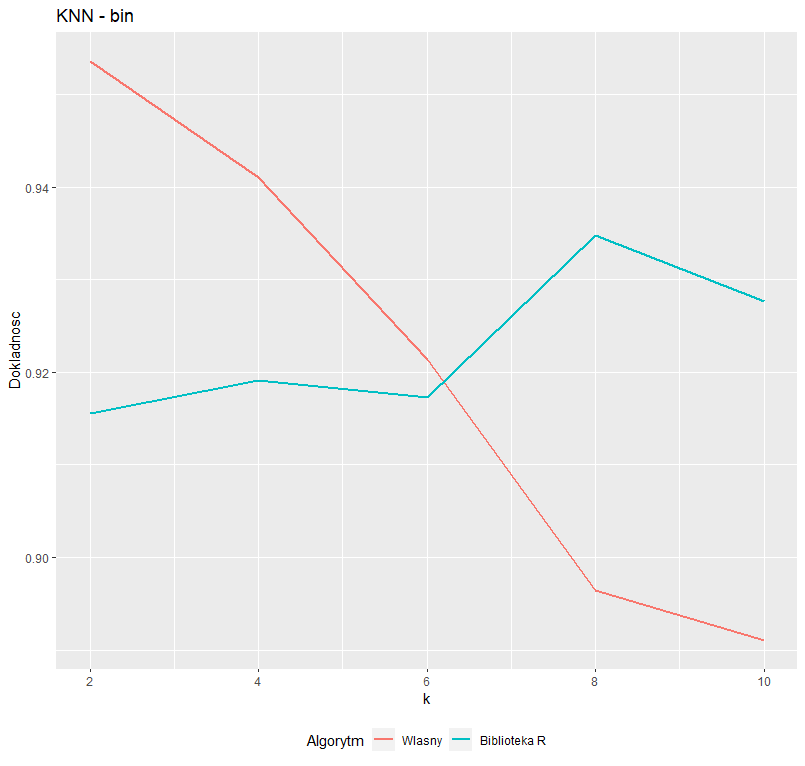
Dane do klasyfikacji binarnej to 286 wierszy i 9 kolumn z wartościami uczącymi. Kolumna nr 1 zawiera klasę “no-recurrence-events” i “recurrence-events”. Rozkład klas to 201:85. W danych pojawiały się dane ze znakami zapytania – wiersze te zostały usunięte przez co ostateczny rozkład klas zmienił się – 196:81. Kolumny z informacjami uczącymi zawierały różne wartości numeryczne i faktor. Kolumny z faktorem zostały zmienione na odpowiednie wartości numeryczne odpowiadające poziomom.

Zbiór dla klasyfikacji wieloklasowej został wybrany przez prowadzącego. Jest to zbiór 4177 wierszy i 8 kolumnami, z dodatkową kolumną klas. W zbiorze nie występowały żadne dane typu NA lub uniemożliwiające uczenie modeli. W tym przypadku tylko pierwsza kolumna zawierająca 3 poziomy została zamieniona na wartości numeryczne. Docelowe klasy wstępnie były opisane przy pomocy wartości numerycznych, ale ze względu na problem klasyfikacji zamieniono tę kolumnę na factor.

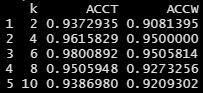
Algorytmy z własną implementacją oraz wbudowane (z bibliotek R) zostały poddane kroswalidacji 8 razy - parametr *kFold = 8*.

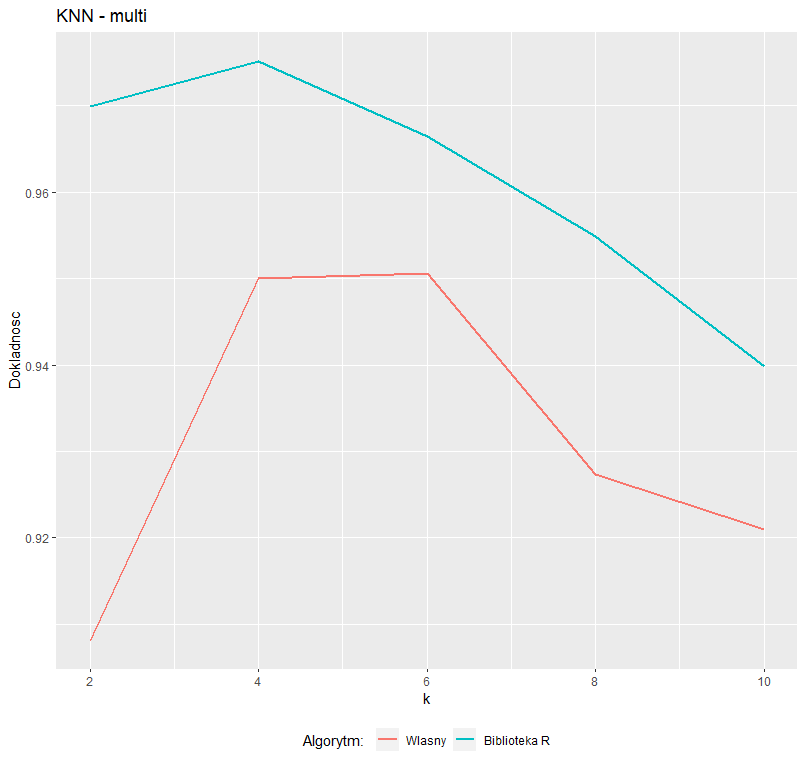
KNN – Wyniki dla własnej implementacji i porównanie z wbudowaną  
**Klasyfikacja Binarna**



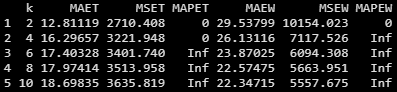


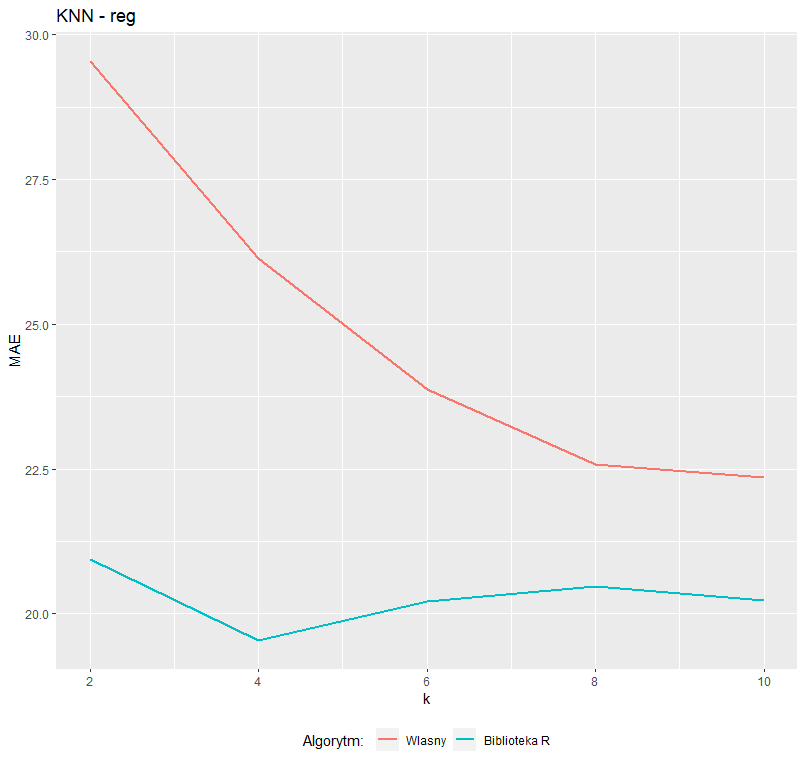
KNN – Wyniki dla własnej implementacji i porównanie z wbudowaną  
**Klasyfikacja Wieloklasowa**





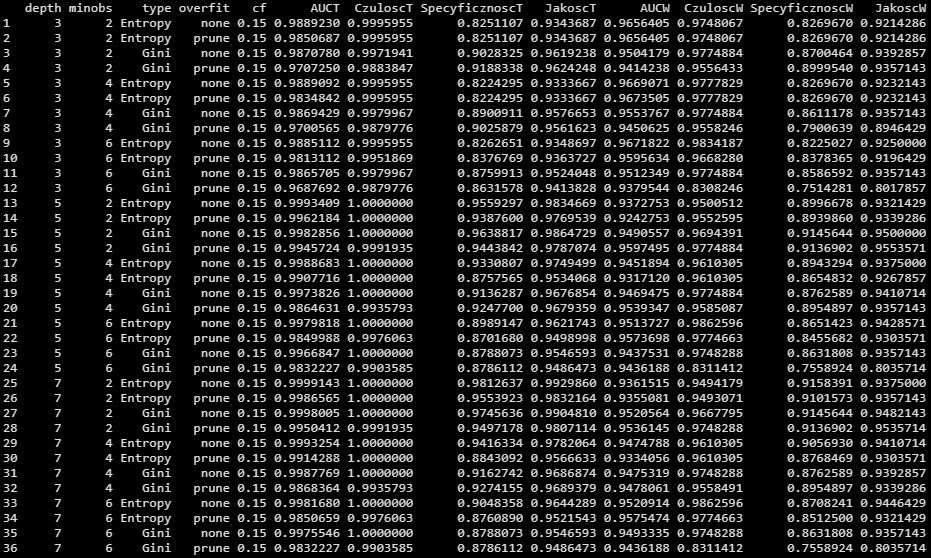
KNN – Wyniki dla własnej implementacji i porównanie z wbudowaną  
**Regresja**

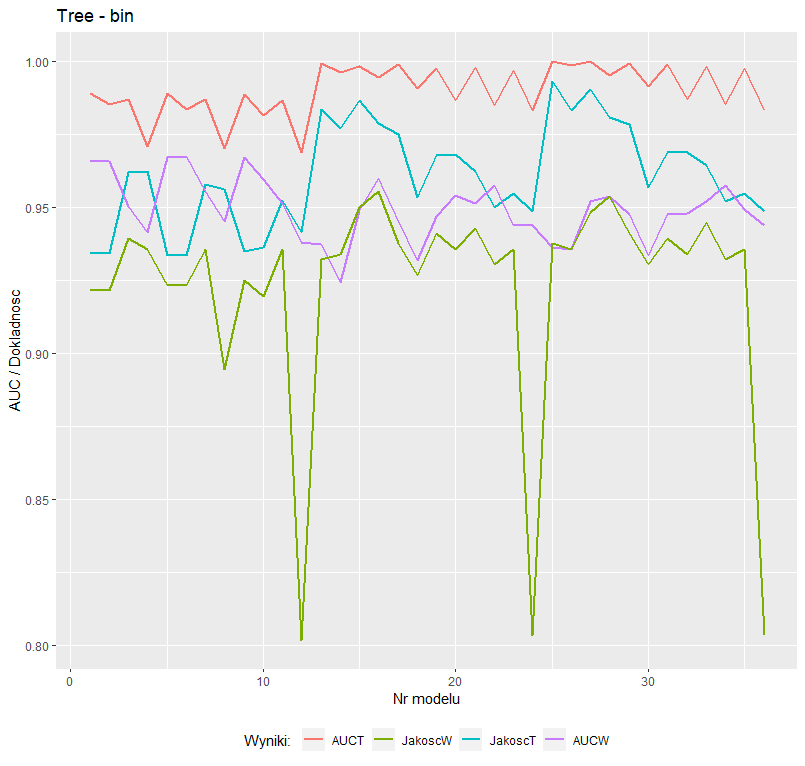




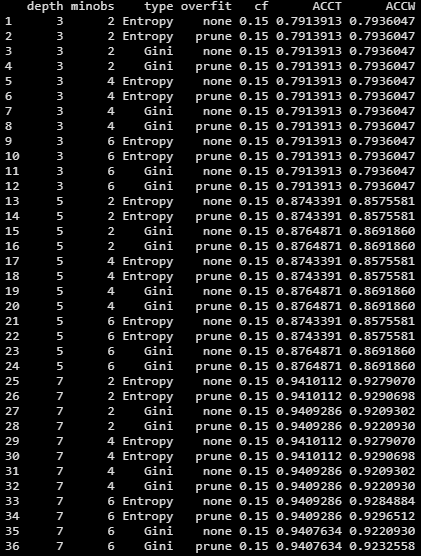
Drzewa Decyzyjne (własna implementacja)  
**Klasyfikacja Binarna**

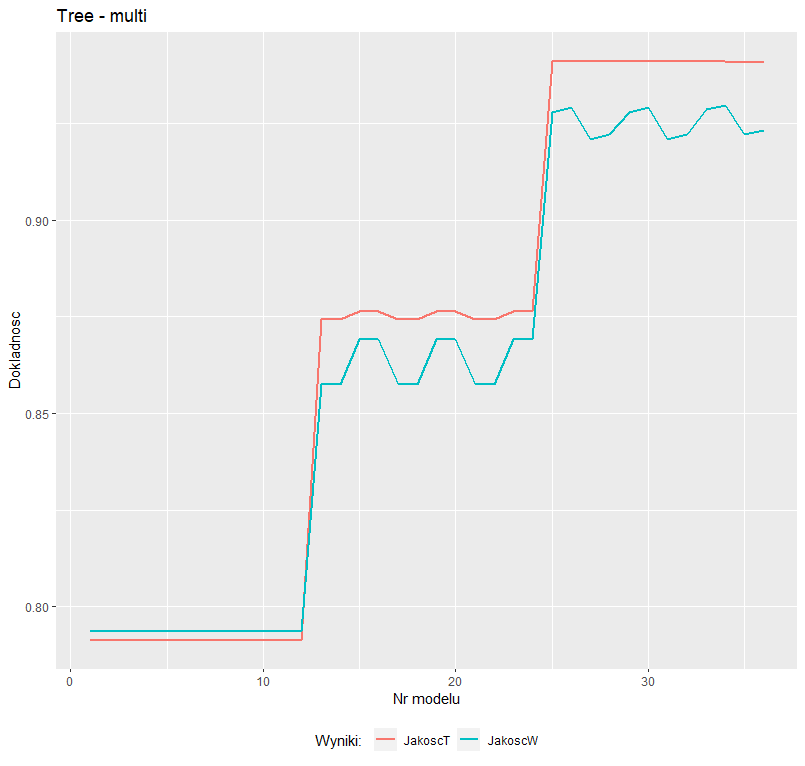
**Depth = c(3,5,7), minobs = c(2,4,6), type = c('Entropy', 'Gini'),   
overfit = c('none', 'prune'), cf = c(0.15)**





Drzewa Decyzyjne (własna implementacja)  
**Klasyfikacja Wieloklasowa   
Depth = c(3,5,7), minobs = c(2,4,6), type = c('Entropy', 'Gini'),   
overfit = c('none', 'prune'), cf = c(0.15)**

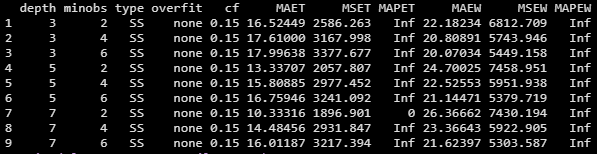


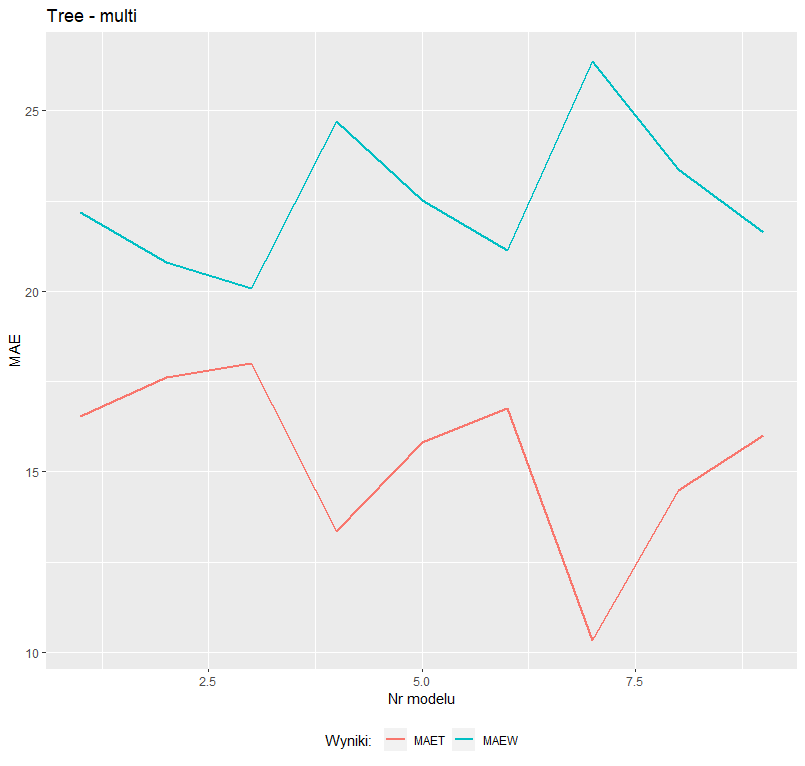


Drzewa Decyzyjne (własna implementacja)  
**Regresja**

**Depth = c(3,5,7), minobs = c(2,4,6),**

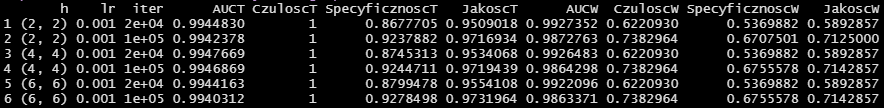
**type = c(‘SS’), overfit = c('none')**

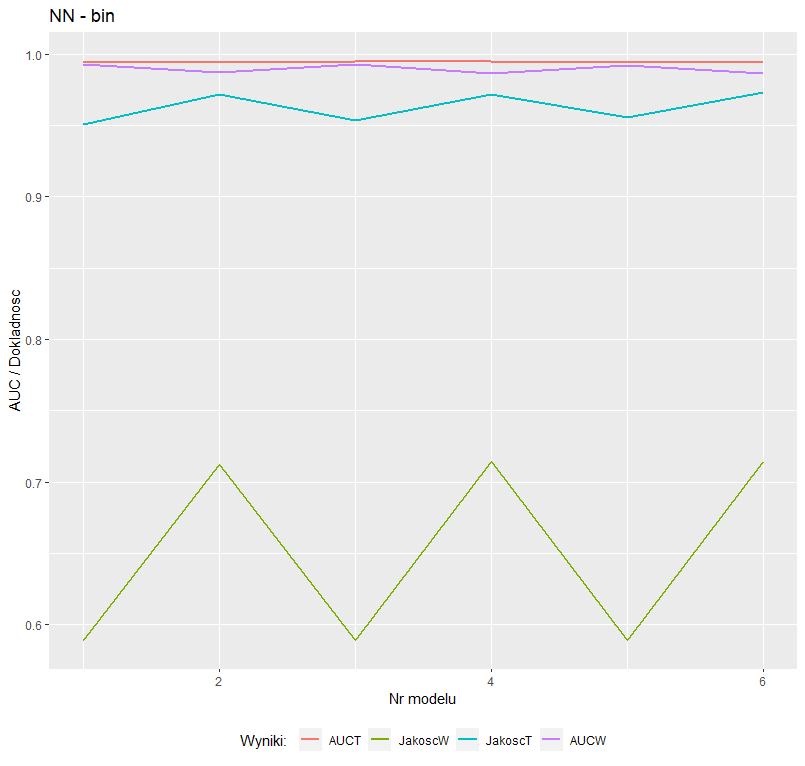




Sieci Neuronowe - własna implementacja  
**Klasyfikacja Binarna**

**h = c(2,2), c(4,4), c(6,6), lr = 0.001 , iteracje = c(20000, 100000)**



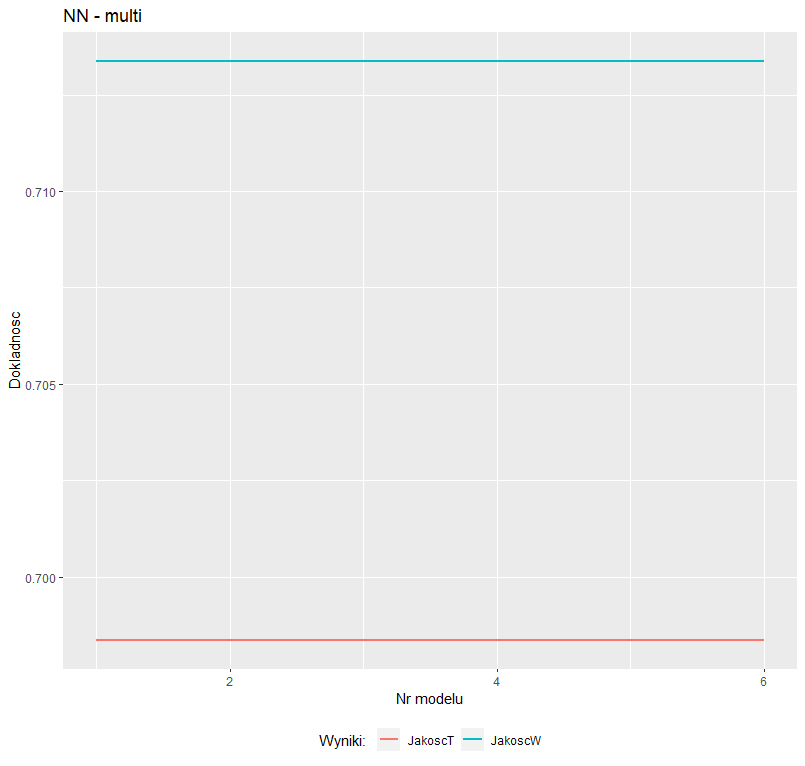


Sieci Neuronowe - własna implementacja  
**Klasyfikacja Wieloklasowa**

**h = c(2,2), c(4,4), c(6,6), lr = 0.001 , iteracje = c(20000, 100000)**

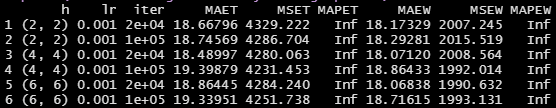
Obraz zawierający tekst

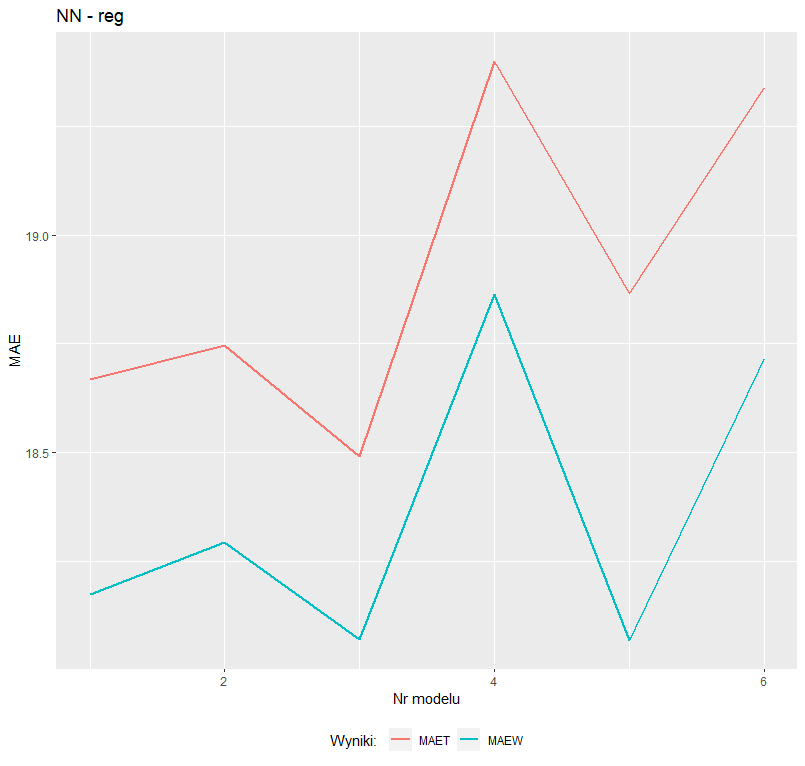
Opis wygenerowany automatycznie



Sieci Neuronowe - własna implementacja  
**Regresja**

**h = c(2,2), c(4,4), c(6,6), lr = 0.001 , iteracje = c(20000, 100000)**





**Tabela z wynikami i parametrami najlepszych modeli**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model | Implementacja | Parametry | Dokładność predykcji |
| Klasyfikacja Binarna | KNN | Własna | k = 2 | 0.95 |
|  | KNN | Biblioteka R | k = 8 | 0.93 |
|  | Drzewa decyzyjne | Własna | depth = 5, minobs = 2, Gini, overfit = prune, cf = 0.15 | 0.95 |
|  | Drzewa decyzyjne | Biblioteka R | maxdepth = 4 | 0.93 |
|  | Sieci NN | Własna | h = (6,6), lr = 0.001,  *iter =* 100000 | 0.71 |
|  | Sieci NN | Biblioteka R | h = 11 | 0.93 |
| Klasyfikacja Wieloklasowa | KNN | Własna | k = 6 | 0.95 |
|  | KNN | Biblioteka R | k = 4 | 0.97 |
|  | Drzewa decyzyjne | Własna | depth = 7, minobs = 6, Entropy, overfit = prune, cf = 0.15 | 0.93 |
|  | Drzewa decyzyjne | Biblioteka R | maxdepth = 16 | 0.94 |
|  | Sieci NN | Własna | h = (2,2), lr = 0.001,  *iter =* 20000 | 0.71 |
|  | Sieci NN | Biblioteka R | h = 14 | 0.99 |
| Regresja | KNN | Własna | k = 10 | 22.34 |
| (MAE) | KNN | Biblioteka R | k = 14 | 18.89 |
|  | Drzewa decyzyjne | Własna | depth = 3, minobs = 6, SS | 20.07 |
|  | Drzewa decyzyjne | Biblioteka R | maxdepth = 2 | 20.75 |
|  | Sieci NN | Własna | h = (4,4), lr = 0.001,  *iter =* 100000 | 18.86 |
|  | Sieci NN | Biblioteka R | h = 12 | 0.012 |

**Wnioski**

Własna implementacja algorytmów takich jak KNN, Drzew Decyzyjnych czy Sieci Neuronowych opiera się na podstawowej idei tych modeli predykcyjnych. W projekcie zbadano własne implementacje w 3 różnych problemach – klasyfikacji binarnej, wieloklasowej oraz regresji. Zrobiono przegląd wyników w zależności od hiper-parametrów modeli oraz porównano dokładność czy MAE do wyników modeli zaimplementowanych w bibliotekach R. Aby wyniki były odpowiednio uśrednione wykonano kroswalidacje 8 razy – dane zostały losowane za każdym razem, aby tworzyć niepowtarzalne dane treningowe, a reszta danych stawała się danymi walidacyjnymi. W przypadku algorytmów z bibliotek R również taka kroswalidacja została wykonana.

Problem klasyfikacji binarnej opierał się na danych z wartościami numerycznymi. Najlepszym modelem z całego zestawienia okazały się KNN i drzewa decyzyjne – wynik dokładności 0.95, w obu przypadkach implementacja własna. Prawie wszystkie modele w zestawieniu osiągnęły wynik powyżej 0.90, jedynym odchyleniem tutaj był zaimplementowany własnoręcznie model sieci neuronowej, którego dokładność wyniosła 0.71 . Porównując modele KNN to zależność wyniku od parametru k jest odwrotna – własna implementacja dobrze sobie radzi przy małych k, a z biblioteki R przy coraz większych k. Przy drzewach decyzyjnych nie widać jakiegoś ogromnego wpływu hiper-parametrów na wynik. Jedynym odchyłkiem jest kombinacja, tylko w 3 przypadkach, parametru ‘Gini’ z ucinaniem drzewa i parametrem cf = 0.15 . Sieci neuronowe w tym przypadku ukazują jak ważna jest ilość iteracji. Modele z 20 tys. Iteracji radziły sobie dużo gorzej w porównaniu do 100 tys. Iteracji.

Klasyfikacja wieloklasowa to wygrana sieci neuronowej z biblioteki R – osiągnęła wynik dokładności 0.99 . Pozostałe modele uzyskiwały wyniki powyżej 0.93, z najlepszym modele ręcznie implementowanym KNN – 0.95 . W tym przypadku również własna sieć neuronowa była najgorsza = 0.71 . Wpływ hiper-parametrów - KNN – najlepsze k w obu przypadkach było równe 4, po tej wartości dokładność modelu, w obu przypadkach spadała. Przy drzewach decyzyjnych ważnym parametrem jest głębokość drzewa co bardzo dobrze widać na wykresie – każdy przeskok tego parametru skutkował zwiększeniem dokładności predykcji o kilka procent. Przy sieci neuronowej nie widać żadnego wpływu hiper-parametrów, co jest bardzo zaskakujące.

Regresja – ten problem był dość specyficzny ze względu na wartości równe 0.0 . Skutkowało to wynikami MAPE, których nie dało się opisać wartościami liczbowymi. Dlatego też w tym przypadku wynikami prównawczymi jest MAE. Tutaj sieci neuronowe osiągają najlepsze wyniki – implementacja z R osiąga minimalne MAE na poziomie 0.012, natomiast własna implementacja 18.86. Pozostałe modele miały wyniki na poziomie 18.89 – 22.34 .   
Dla własnej implentacji KNN parametr k im większy tym mniejsze MAE. W przypadku drzew decyzyjnych parametr ‘minobs’ był kluczowy i przy jego zmianie widać było spadki wyników MAE, natomiast zwiększający się parametr głębokości powodował zwiększanie MAE. Sieci neuronowe z drugiej strony osiągały niskie MAE, dużych zmian nie było, alczkolwiek im więcej iteracji modelu tym większe MAE. Możliwe, że sieć powoli się przeuczała i widać to na obu wynikach MAE – na zbiorze treningowym i walidacyjnym.