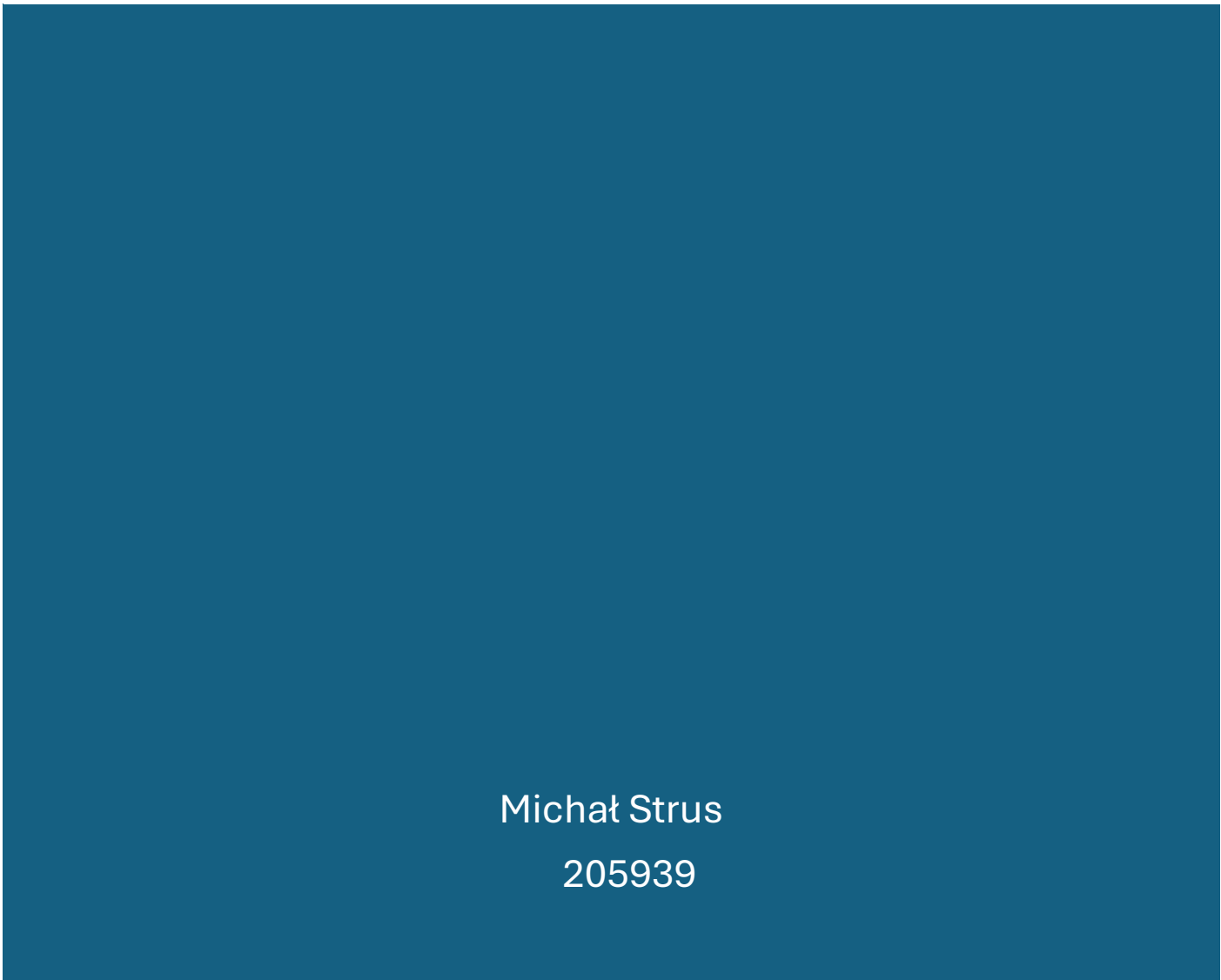


PORÓWNANIE ALGORYTMÓW UCZENIA MASZYNOWEGO: DRZEWA DECYZYJNE, SIECI NEURONOWE ORAZ KNN



Michał Strus

205939

Wstęp

Współczesne metody uczenia maszynowego oferują wiele podejść do rozwiązywania problemów klasyfikacji i regresji. W niniejszym raporcie porównano trzy popularne algorytmy: drzewa decyzyjne, algorytm k-najbliższych sąsiadów (KNN) oraz sieci neuronowe. Celem analizy jest porównanie wydajności własnych implementacji tych algorytmów z ich odpowiednikami dostępnymi w pakietach R, takich jak caret, rpart oraz neuralnet. Dodatkowo przeanalizowano wpływ hiperparametrów na jakość modeli oraz przedstawiono propozycje wizualizacji wyników. W kodzie pojawia się jedynie niewielka część przeprowadzonych crosswalidacji aby zaoszczędzić czas. Pełne wyniki dostępne są w pliku Excel wyniki.xlsx. Crosswalidacja została przeprowadzona z parametrem kfold=5 dla różnych hiperparametrów w zależności od algorytmu. Rozważane były 3 zbiory danych bez braków danych. Po crosswalidacji zostały wybrane najlepsze hiperparametry modeli oraz została sporządzona predykcja i ewaluacja wyników na różnych metrykach na zbiorach testowych, które stanowiły 30% danych. Wyjątkiem był algorytm KNN, w którym zbiór zawsze był wyłącznie 200 elementowy przed podziałem. W zbiorze do regresji wszystkie zmienne były liczbowe, natomiast w zbiorze do klasyfikacji binarnej duża część zmiennych była factorami i w przypadku sieci neuronowej zastosowane zostało kodowanie zero-jedynkowe (one-hot encoding). Natomiast zbiór do klasyfikacji wieloklasowej miał prawie 4500 obserwacji i aż 37 kolumn przez co klasyfikatory potrzebowały dużo czasu, aby dobrze dopasować się do danych, a i po czasie ten problem nie został bardzo dobrze rozwiązany.

Drzewa decyzyjne

Drzewa decyzyjne to jeden z podstawowych algorytmów uczenia maszynowego stosowanych w klasyfikacji i regresji. Model ten dzieli przestrzeń decyzyjną na podzbiory, korzystając z iteracyjnych podziałów na podstawie kryteriów takich jak entropia, współczynnik Gini'ego czy suma kwadratów błędów (SS). Każdy węzeł w drzewie reprezentuje pytanie dotyczące cechy (atributu) danych, a każda gałąź odpowiada możliwej wartości tej cechy. Liście drzewa wskazują końcową decyzję – przewidywaną klasę w przypadku klasyfikacji lub wartość w regresji.

Zalety i wady

Zalety

- ✅ **Łatwość interpretacji** – drzewa decyzyjne są intuicyjne i mogą być przedstawione w postaci graficznej.
- ✅ **Brak konieczności normalizacji danych** – algorytm nie wymaga przekształcania zmiennych numerycznych do tego samego zakresu wartości.
- ✅ **Obsługa danych numerycznych i kategoriowych** – drzewa decyzyjne mogą pracować

zarówno na wartościach liczbowych, jak i jakościowych.

✓ **Dobra wydajność na małych i średnich zbiorach danych** – mogą szybko dopasować się do struktury danych.

✓ **Selekcja cech** – drzewa automatycznie wybierają istotne atrybuty, eliminując te mniej istotne.

✓ **Możliwość modelowania interakcji między zmiennymi** – drzewa mogą wychwycić nieliniowe zależności między cechami.

Wady

✗ **Przeuczenie (overfitting)** – głębokie drzewa mogą dopasować się zbyt dokładnie do danych treningowych, co prowadzi do słabej generalizacji. Można temu zaradzić poprzez przycinanie drzewa (pruning).

✗ **Wrażliwość na zmiany w danych** – niewielka zmiana w danych treningowych może prowadzić do znacznie innej struktury drzewa.

✗ **Mniejsza skuteczność w porównaniu z bardziej zaawansowanymi metodami** – takie algorytmy jak lasy losowe (Random Forest oparty na wielu drzewach decyzyjnych) czy gradient boosting (np. XGBoost) często osiągają lepsze wyniki.

✗ **Problemy z dużą liczbą klas** – przy wielu klasach w klasyfikacji drzewa mogą stać się zbyt skomplikowane.

✗ **Podejmowanie decyzji na podstawie lokalnych podziałów** – drzewa mogą ignorować globalne zależności w danych.

Analiza hiperparametrów

Najważniejsze hiperparametry wpływające na wyniki modelu to: - **maksymalna głębokość drzewa** – kontroluje poziom rozgałęzienia modelu. - **minimalna liczba próbek w liściu** – wpływa na ogólną zdolność modelu do generalizacji. **Metoda przycinania drzewa (prune)** – pomaga w redukcji nadmiernego dopasowania. Parametr **cf** (confidence factor) lub **cp** (complexity parameter) w drzewach decyzyjnych służy do regulacji procesu przycinania drzewa (pruning), co pomaga zapobiegać przeuczeniu.

Porównanie wyników

Wyniki modeli drzew decyzyjnych porównano dla klasyfikacji binarnej, wieloklasowej oraz regresji. Zarówno własny algorytm, jak i rpart, uzyskały podobne wyniki, jednak wbudowany algorytm wykazał lepszą optymalizację dla głębokich drzew.

Regresja

Fragment tabeli z crosswalidacji dla drzew decyzyjnych w problemie regresji.

Kolumna1	max_depth	min_samples	pruning_method	criterion	cf	MAE_t	MSE_t	MAPE_t	MAE_w	MSE_w	MAPE_w
1	1	1	prune	SS	0,1	4,68	38,70	25,92	4,80	41,94	26,66
2	5	1	prune	SS	0,1	3,23	22,21	17,65	3,54	27,61	19,16
3	10	1	prune	SS	0,1	2,29	15,03	12,43	2,89	22,12	15,39
4	15	1	prune	SS	0,1	1,72	11,28	9,37	2,51	19,31	13,24
5	1	5	prune	SS	0,1	2,32	16,76	12,68	2,97	23,84	15,93
6	5	5	prune	SS	0,1	2,27	15,50	12,31	2,88	22,32	15,30

7	10	5 prune	SS	0,1	2,16	14,16	11,57	2,77	20,91	14,60
8	15	5 prune	SS	0,1	2,06	13,15	10,99	2,69	19,85	14,07
9	1	10 prune	SS	0,1	2,35	15,98	12,65	2,92	22,30	15,47
10	5	10 prune	SS	0,1	2,33	15,48	12,47	2,88	21,60	15,16
11	10	10 prune	SS	0,1	2,30	14,95	12,21	2,83	20,94	14,86
12	15	10 prune	SS	0,1	2,27	14,51	12,00	2,79	20,39	14,62
13	1	15 prune	SS	0,1	2,45	16,37	13,07	2,95	22,04	15,54
14	5	15 prune	SS	0,1	2,44	16,03	12,94	2,92	21,56	15,34
15	10	15 prune	SS	0,1	2,42	15,69	12,79	2,89	21,11	15,14
16	15	15 prune	SS	0,1	2,40	15,40	12,66	2,86	20,72	14,97
17	1	1 prune	SS	0,2	2,54	16,77	13,44	2,98	21,97	15,66
18	5	1 prune	SS	0,2	2,49	16,15	13,21	2,94	21,49	15,44
19	10	1 prune	SS	0,2	2,38	15,34	12,62	2,87	20,94	15,04
20	15	1 prune	SS	0,2	2,27	14,57	12,00	2,79	20,44	14,63

Najlepszą wartość drzewo osiągało z parametrami max_depth=15, min_samples=1, pruning_method=prune oraz dla parametru cf=0,1 kryterium natomiast było jedno-suma kwadratów reszt. Drzewo charakteryzowała podobna skuteczność zarówno na zbiorze uczącym jak i walidacyjnym, co jest dobrym prognostykiem do wykorzystania tego modelu. Błędy wynosiły odpowiednio 3,41 dla MAE, 32,77 dla MSE oraz 15,37 % dla MAPE, co świadczy o dobrym dopasowaniu modelu do danych. (Metryki na zbiorze testowym).

MAE	MSE	MAPE
3.409333	32.768229	15.376065

Z kolei dla algorytmu wbudowanego za pomocą crosswalidacji szukałem optymalnej wartości parametru cp, który nie do końca pokrywał się z powyższą wartością. Natomiast pozostałe parametry pozostały domyślne. Wyniki dla algorytmu wbudowanego są bardzo zbliżone do tych z opracowanego drzewa decyzyjnego.

MAE	MSE	MAPE
3.147348	25.304130	15.331913

Klasyfikacja binarna

W przypadku klasyfikacji binarnej testowane były te same parametry. Najlepszą kombinacją okazało się być max_depth=10, min_samples=2, pruning_method=prune, kryterium Entropii oraz cf=0,1.

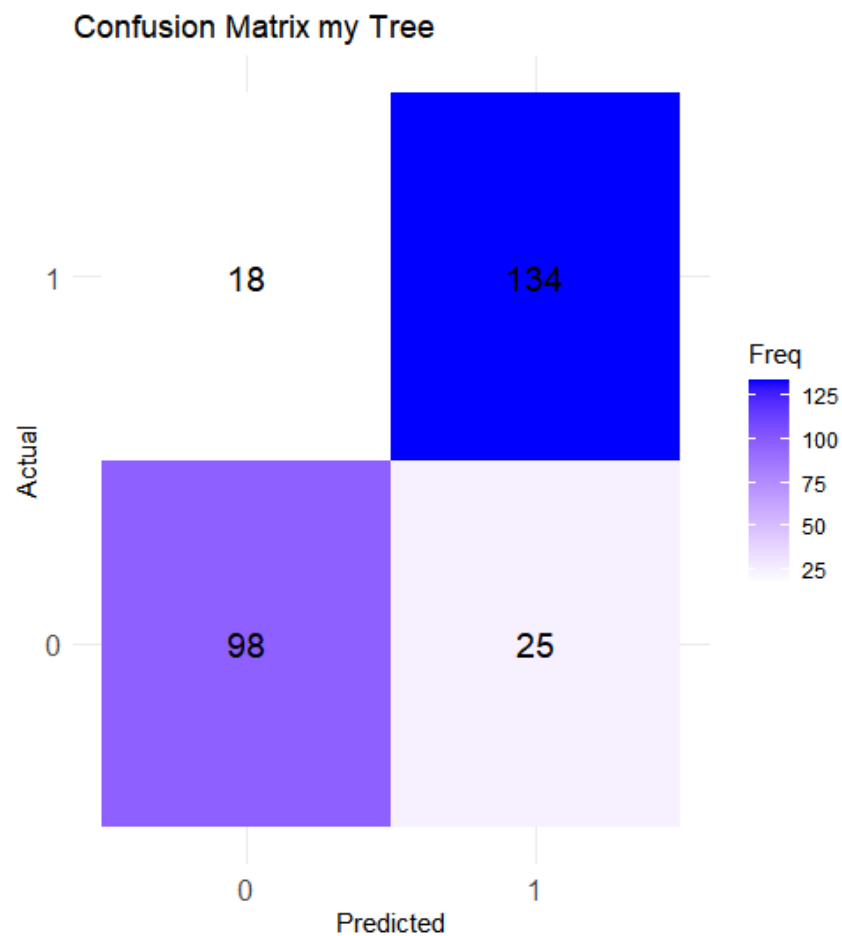
Kolumna1	max_depth	min_samples	pruning_method	criterion	cf	AUC_t	Sensitivity_t	Specificity_t	Accuracy_t	AUC_w	Sensitivity_w	Specificity_w	Accuracy_w
1	3	2	prune	Entropy		0,1	0,919	0,850	0,849	0,849	0,904	0,845	0,846
2	5	2	prune	Entropy		0,1	0,941	0,885	0,860	0,871	0,908	0,861	0,849
3	7	2	prune	Entropy		0,1	0,958	0,911	0,881	0,895	0,911	0,874	0,855
4	10	2	prune	Entropy		0,1	0,968	0,931	0,901	0,914	0,912	0,885	0,864
5	3	5	prune	Entropy		0,1	0,958	0,914	0,890	0,901	0,911	0,877	0,861
6	5	5	prune	Entropy		0,1	0,959	0,916	0,885	0,899	0,911	0,879	0,859
7	7	5	prune	Entropy		0,1	0,962	0,919	0,889	0,903	0,911	0,879	0,859
8	10	5	prune	Entropy		0,1	0,966	0,923	0,894	0,907	0,913	0,877	0,861
9	3	10	prune	Entropy		0,1	0,960	0,915	0,888	0,900	0,911	0,873	0,859
10	5	10	prune	Entropy		0,1	0,960	0,911	0,888	0,898	0,912	0,873	0,857
11	7	10	prune	Entropy		0,1	0,960	0,911	0,887	0,897	0,913	0,870	0,856
12	10	10	prune	Entropy		0,1	0,961	0,911	0,885	0,897	0,914	0,868	0,855
13	3	15	prune	Entropy		0,1	0,958	0,907	0,881	0,893	0,913	0,866	0,854
14	5	15	prune	Entropy		0,1	0,957	0,905	0,879	0,891	0,913	0,865	0,853
15	7	15	prune	Entropy		0,1	0,957	0,903	0,878	0,889	0,914	0,864	0,852
16	10	15	prune	Entropy		0,1	0,957	0,902	0,876	0,888	0,914	0,865	0,851
17	3	2	none	Entropy		0,1	0,954	0,899	0,875	0,886	0,913	0,864	0,851
18	5	2	none	Entropy		0,1	0,955	0,900	0,875	0,886	0,913	0,865	0,851
19	7	2	none	Entropy		0,1	0,957	0,903	0,877	0,889	0,914	0,867	0,852
20	10	2	none	Entropy		0,1	0,959	0,908	0,881	0,893	0,914	0,869	0,854

Macierz pomyłek wygląda następująco. Accuracy jest na poziomie 84,4% co wskazuje na dobrą jakość modelu, świadczy to o tym, że drzewo klasyfikuje 84,4% przypadków poprawnie. Widać tutaj, że zachowana jest równowaga klas i żadna z klas nie dominuje znacząco.

```
$confMat
      predicted
actual 0    1
      0 98  25
      1 18 134

$Youden
[1] 0.6783269 1.0000000

$Metrics
      AUC Sensitivity Specificity      Accuracy
0.8579375 0.7967480 0.8815789 0.8436364
```

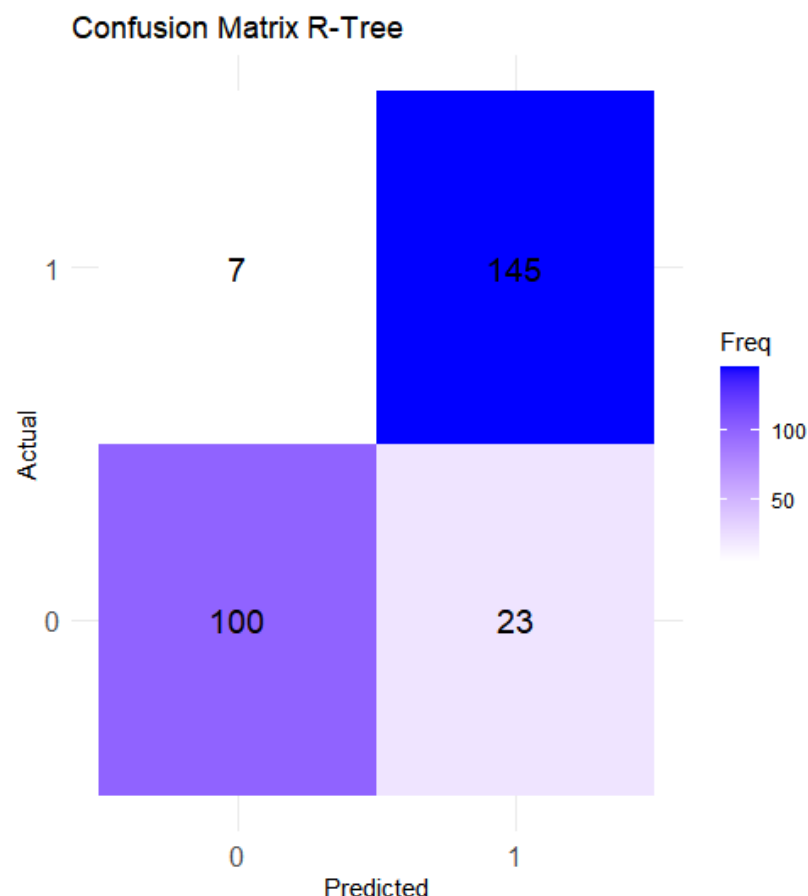


W przypadku wbudowanego algorytmu podobnie jak poprzednio testowany był ten sam parametr. Wyniki są lepsze od własnego algorytmu, ale nie dużo. Accuracy jest na poziomie 89% co świadczy o tym, że model jest bardzo dobry.

```
$ConfMat
      predicted
actual    0    1
    0 100   23
    1   7 145

$Youden
[1] 0.7669555 0.7000000

$Metrics
      AUC Sensitivity Specificity Accuracy
0.9118528 0.8130081 0.9539474 0.8909091
```



Klasyfikacja wieloklasowa

W przypadku klasyfikacji wieloklasowej parametry były testowane w analogiczny sposób, co w pozostałych przypadkach. Najlepszą kombinacją hiperparametrów okazało się być `max_depth=15`, `min_samples=5`, `pruning_method=prune`, `criterion=Entropy` oraz `cf=0,1`. Najwyższe accuracy nie było jednak w tym wypadku aż tak dobre bo wyniosło 77,4% na zbiorze walidacyjnym. Jednak ten problem decyzyjny był znacznie cięższym przypadkiem niż pozostałe, ponieważ zbiór danych zawierał 4500 wierszy i 37 kolumn.

Kolumna1	max_depth	min_samples	pruning_method	criterion	cf	AUC_t	Accuracy_t	AUC_w	Accuracy_w
1	15	5	prune	Entropy		0,1	0,918	0,835	0,774
2	15	10	prune	Entropy		0,1	0,911	0,821	0,765
3	3	10	prune	Entropy		0,1	0,900	0,818	0,768
4	10	10	prune	Entropy		0,1	0,905	0,817	0,766
5	15	5	prune	Gini		0,1	0,908	0,817	0,765
6	15	10	prune	Gini		0,1	0,906	0,815	0,764
7	15	5	prune	Entropy		0,05	0,906	0,815	0,765
8	15	10	prune	Entropy		0,05	0,905	0,814	0,764
9	10	10	prune	Gini		0,1	0,904	0,814	0,764
10	15	5	prune	Gini		0,05	0,905	0,814	0,764
11	15	5	prune	Entropy		0,2	0,905	0,813	0,764
12	15	10	prune	Gini		0,05	0,905	0,813	0,764
13	3	10	prune	Gini		0,1	0,903	0,813	0,764
14	3	15	prune	Entropy		0,1	0,902	0,813	0,763
15	15	5	prune	Gini		0,2	0,904	0,813	0,764
16	15	10	prune	Entropy		0,2	0,905	0,813	0,764
17	10	10	prune	Entropy		0,05	0,904	0,813	0,764
18	5	10	prune	Entropy		0,1	0,898	0,813	0,766
19	3	10	prune	Entropy		0,05	0,903	0,813	0,764
20	15	10	prune	Gini		0,2	0,904	0,813	0,764

Accuracy dla tego modelu wyniosło zaledwie 68% jednak był to ciężki problem decyzyjny oraz widoczna była nierównowaga klas.

```
$Matrix
      predicted
actual Dropout Enrolled Graduate
Dropout    127     27     23
Enrolled    28     36     42
Graduate    33     38    244

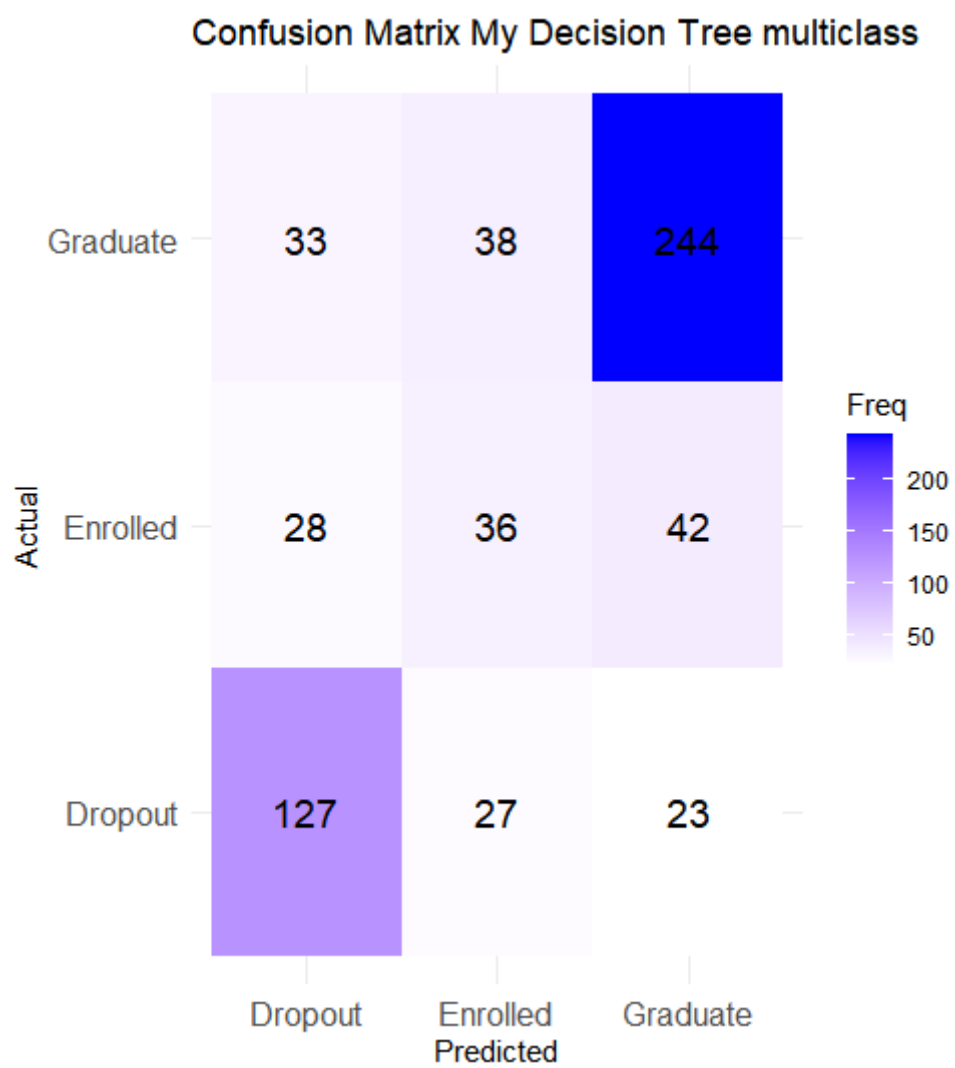
$Accuracy
[1] 0.680602

$Precision
      Dropout Enrolled Graduate
0.7175141 0.3396226 0.7746032

$Recall
      Dropout Enrolled Graduate
0.6755319 0.3564356 0.7896440

$F1_macro
[1] 0.6085893

$AUC_multi
[1] 0.7794451
```

Natomiast w przypadku algorytmu wbudowanego wyniki są następujące

```

$Matrix
      predicted
actual  Dropout Enrolled Graduate
Dropout   133     21     23
Enrolled   19     29     58
Graduate   20      9    286

$Accuracy
[1] 0.7491639

$Precision
      Dropout Enrolled Graduate
0.7514124 0.2735849 0.9079365

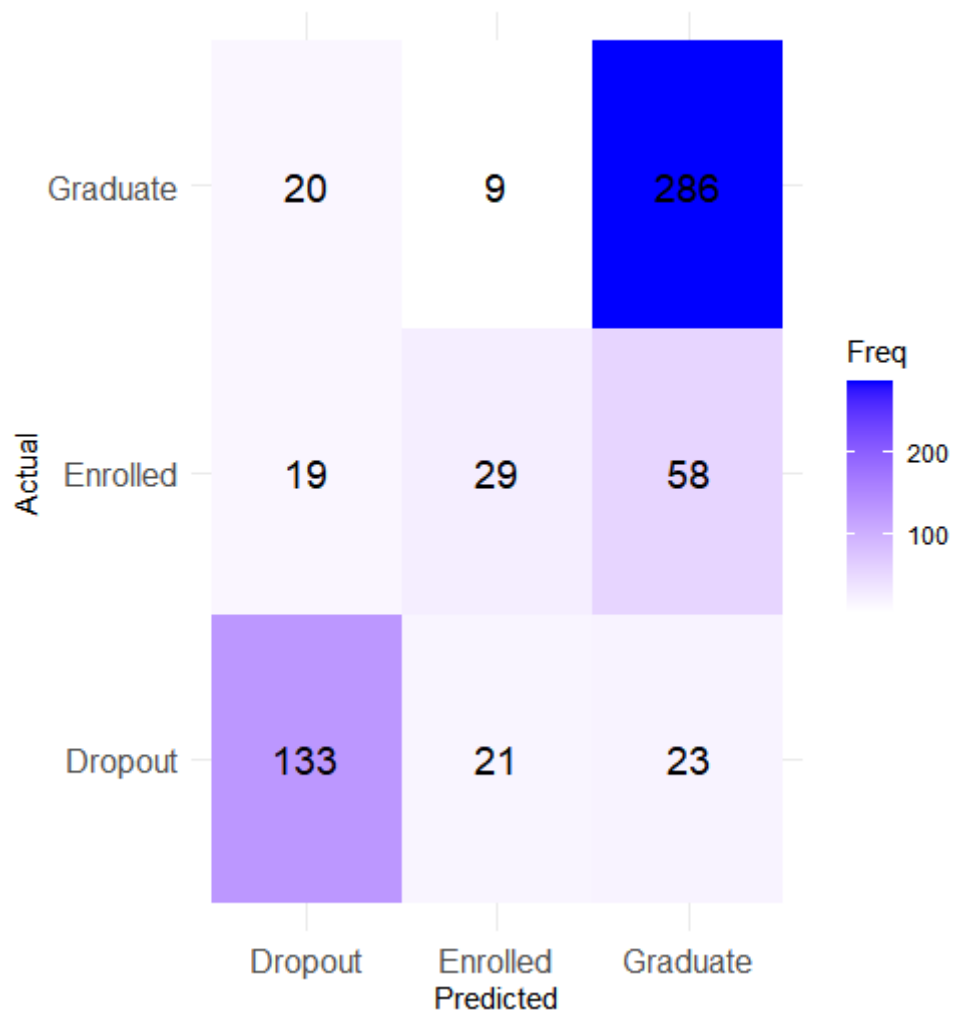
$Recall
      Dropout Enrolled Graduate
0.7732558 0.4915254 0.7792916

$F1_macro
[1] 0.6508008

$AUC_multi
[1] 0.7960764

```

Confusion Matrix R build Decision Tree multiclass



W tym przypadku model wbudowany radzi sobie lepiej niż w pozostałych, ponieważ jest lepszy pod względem accuracy o 6 punktów procentowych, jednak klasa Enrolled jest lepiej klasyfikowana przez własny algorytm.

K-najbliżsi sąsiedzi (KNN)

Algorytm klasyfikuje nowy punkt danych lub przewiduje wartość, biorąc pod uwagę k najbliższych sąsiadów w przestrzeni cech. Decyzja podejmowana jest na podstawie ich etykiet lub wartości.

Analiza hiperparametrów

Najważniejsze hiperparametry: **Liczba sąsiadów (k)** – mniejsza wartość k powoduje bardziej elastyczny model, podczas gdy wyższa wartość redukuje szumy. **Metody normalizacji** – wpływają na jakość klasyfikacji, np. standaryzacja lub min-max scaling. W crosswalidacji testowanym parametrem była liczba sąsiadów, natomiast normalizacja odbywała się poprzez min-max scaling.

Porównanie wyników

Porównanie wyników własnej implementacji KNN z algorytmem caret wykazało, że optymalizacja liczby sąsiadów ma kluczowe znaczenie dla poprawy wyników. Jednak opracowany algorytm KNN jest bardzo powolny w porównaniu z algorytmem wbudowanym, co w zasadzie wyklucza cel jego użycia.

Regresja

W przypadku opracowanego algorytmu najlepszymi parametrami okazała się być liczba sąsiadów $k=3$.

Kolumna1	k	MAE_t	MSE_t	MAPE_t	MAE_w	MSE_w	MAPE_w
1	3	1,32	4,89	5,30	2,31	13,37	9,51
2	5	1,65	7,49	6,58	2,41	15,51	9,85
3	7	1,84	9,72	7,35	2,50	17,18	10,13
4	10	2,02	12,01	8,05	2,63	19,02	10,58

Błędy oszacowań były dobre MAPE wynosiło około 12% co oznacza, że przeciętne oszacowania różniły się o 12% od rzeczywistej wartości zmiennej.

MAE	MSE	MAPE
2.762147	14.975009	11.975960

Natomiast w modelu wbudowanym podczas crosswalidacji najlepszym modelem okazał się ten z parametrem $k=2$. Jego wyniki są niewiele lepsze od opracowanego modelu.

MAE	MSE	MAPE
2.789831	17.579407	11.529518

Klasyfikacja binarna

W przypadku klasyfikacji binarnej najlepszym parametrem okazało się być ponownie $k=3$. Accuracy na zbiorze walidacyjnym sięgało 92,5% co jest bardzo dobrym wynikiem. Jednak należy pamiętać, że próba była bardzo mała aby zrekompensować czas.

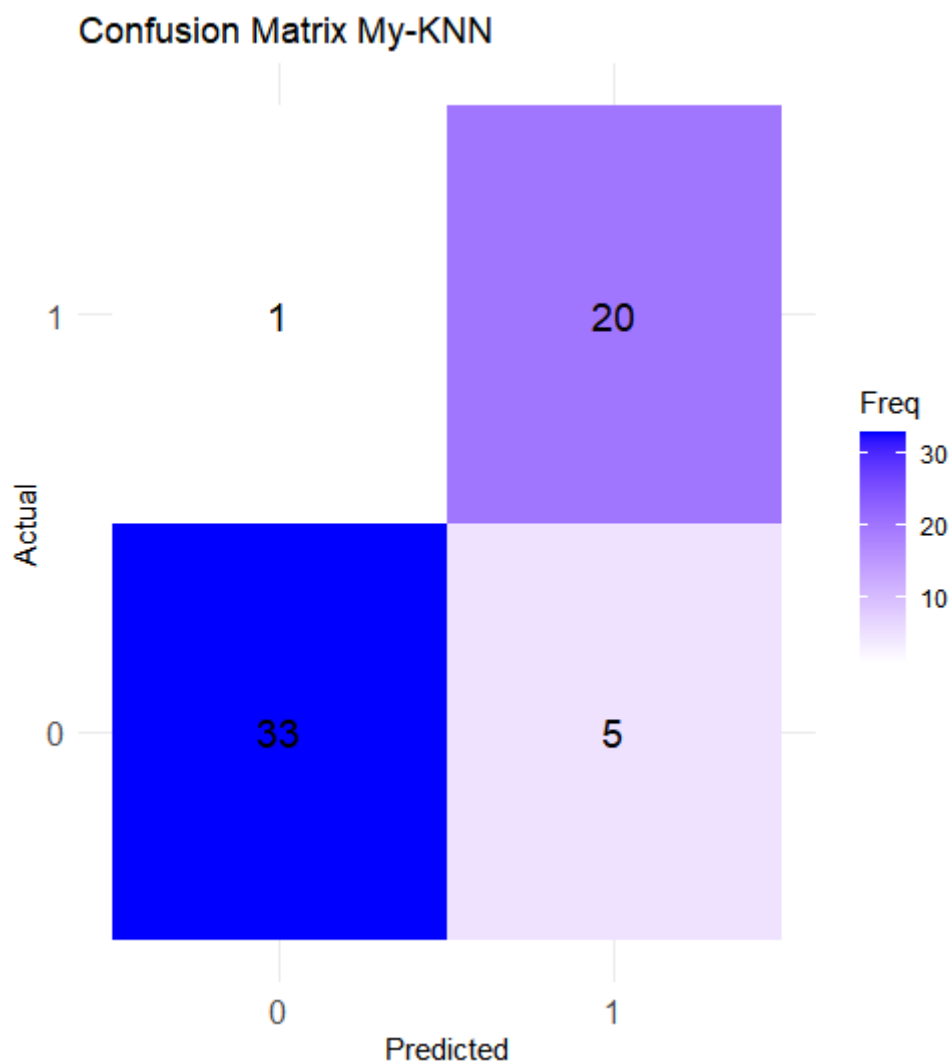
Kolumna1	k	AUC_t	Sensitivity_t	Specificity_t	Accuracy_t	AUC_w	Sensitivity_w	Specificity_w	Accuracy_w
1	3	0,991	0,966	0,885	0,944	0,959	0,921	0,940	0,925
2	5	0,986	0,942	0,921	0,937	0,959	0,903	0,970	0,923
3	7	0,983	0,929	0,937	0,932	0,960	0,900	0,974	0,922
4	10	0,979	0,919	0,947	0,928	0,960	0,893	0,980	0,919

Natomiast na zbiorze testowym z najlepszym hiperparametrem model opracowany osiągał accuracy na poziomie 89% co jest bardzo dobrym wynikiem.

```
$ConfMat
      predicted
actual 0  1
      0 33  5
      1  1 20

$Youden
[1] 0.820802 1.000000

$Metrics
      AUC Sensitivity Specificity Accuracy
0.9323308 0.8684211 0.9523810 0.8983051
```

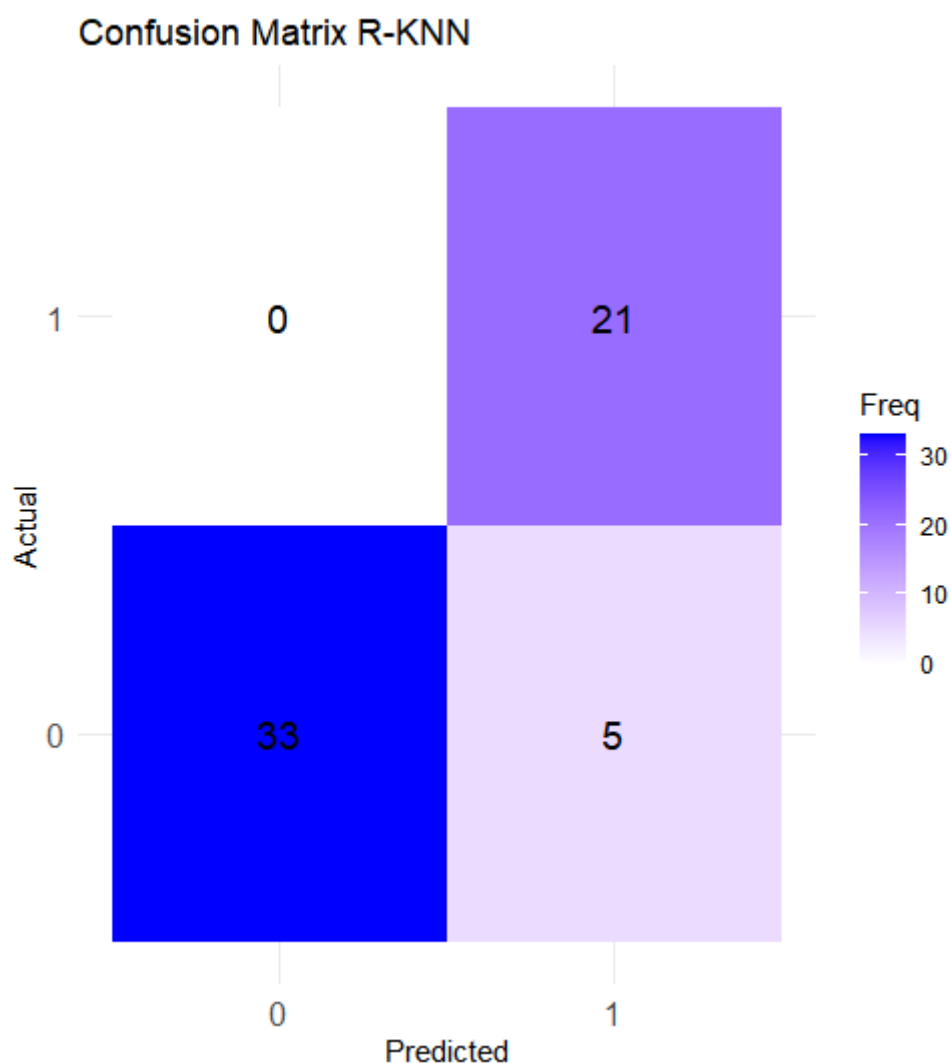


W przypadku algorytmu wbudowanego optymalną liczbą sąsiadów okazało się być $k=20$. Accuracy było tutaj wyższe o około 2 punkty procentowe, a więc wyniki są podobne.

```
$ConfMat
      predicted
actual 0 1
0 33 5
1 0 21

$Youden
[1] 0.8684211 0.6666667

$Metrics
      AUC Sensitivity Specificity Accuracy
0.9461153 0.8684211 1.0000000 0.9152542
```



Klasyfikacja wieloklasowa

W tym przypadku podobnie jak drzewo decyzyjne, algorytm miał znacznie większe trudności z poprawną klasyfikacją. Wynika to głównie ze specyfiki zbioru oraz doboru jego fragmentu. Wyniki z crosswalidacji były następujące

Kolumna1	K	AUC_t	Accuracy_t	AUC_w	Accuracy_w
1	3	0,871	0,744	0,775	0,655
2	5	0,853	0,716	0,769	0,645
3	7	0,837	0,699	0,759	0,647
4	10	0,821	0,685	0,749	0,636

Accuracy w tym wypadku wynosiło zaledwie 60%. Jednak widać jak rozkładały się klasy w zbiorze testowym.

```

$Matrix
      predicted
actual Dropout Enrolled Graduate
Dropout      10         0         8
Enrolled       4         1         6
Graduate       4         1        24

$Accuracy
[1] 0.6034483

$Precision
      Dropout  Enrolled  Graduate
0.55555556 0.09090909 0.82758621

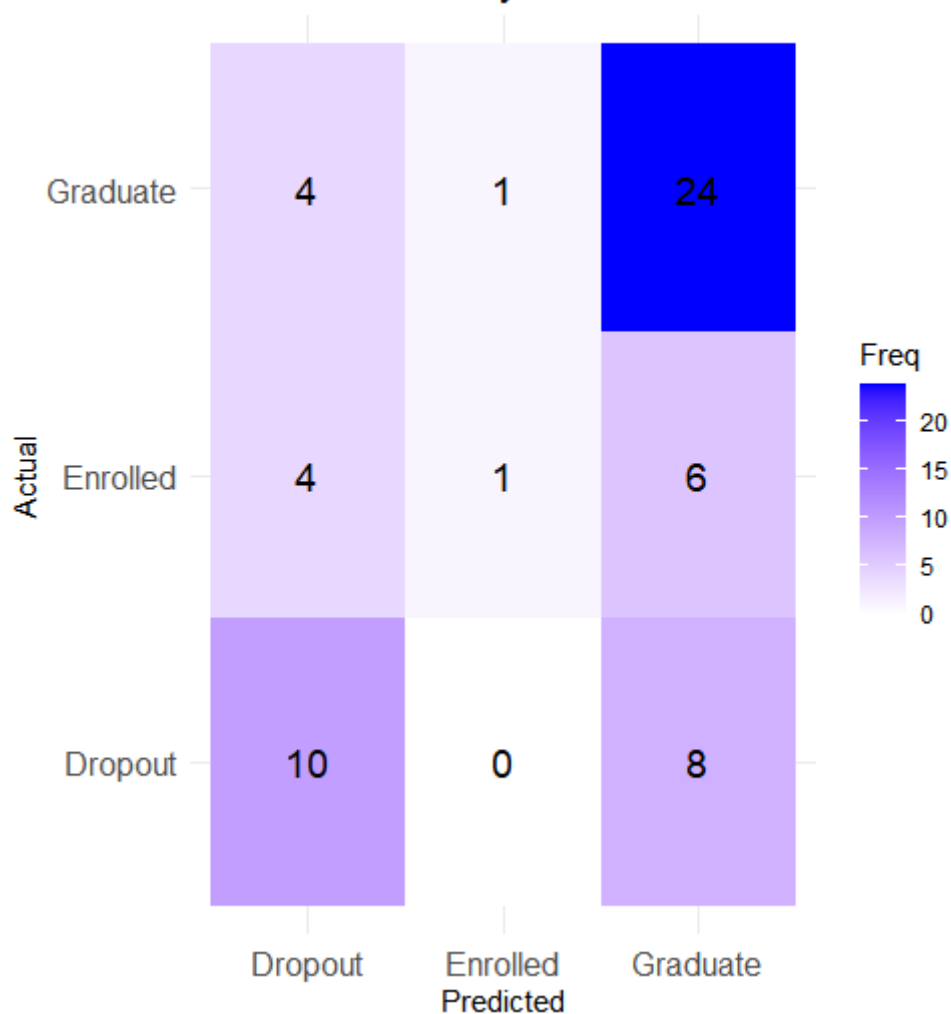
$Recall
      Dropout  Enrolled  Graduate
0.55555556 0.50000000 0.6315789

$F1_macro
[1] 0.4752732

$AUC_multi
[1] 0.6070475

```

Confusion Matrix My KNN multiclass



W przypadku algorytmu wbudowanego optymalną liczbą sąsiadów okazało się być $k=5$. Natomiast problem z rezultatami również występował, co potwierdza wcześniejsze przypuszczenia. Accuracy jest tutaj na poziomie 58% co jest wynikiem nawet mniejszym niż w opracowanym algorytmie.

```
$Matrix
      predicted
actual Dropout Enrolled Graduate
Dropout      7         2         9
Enrolled      2         2         7
Graduate      2         2        25

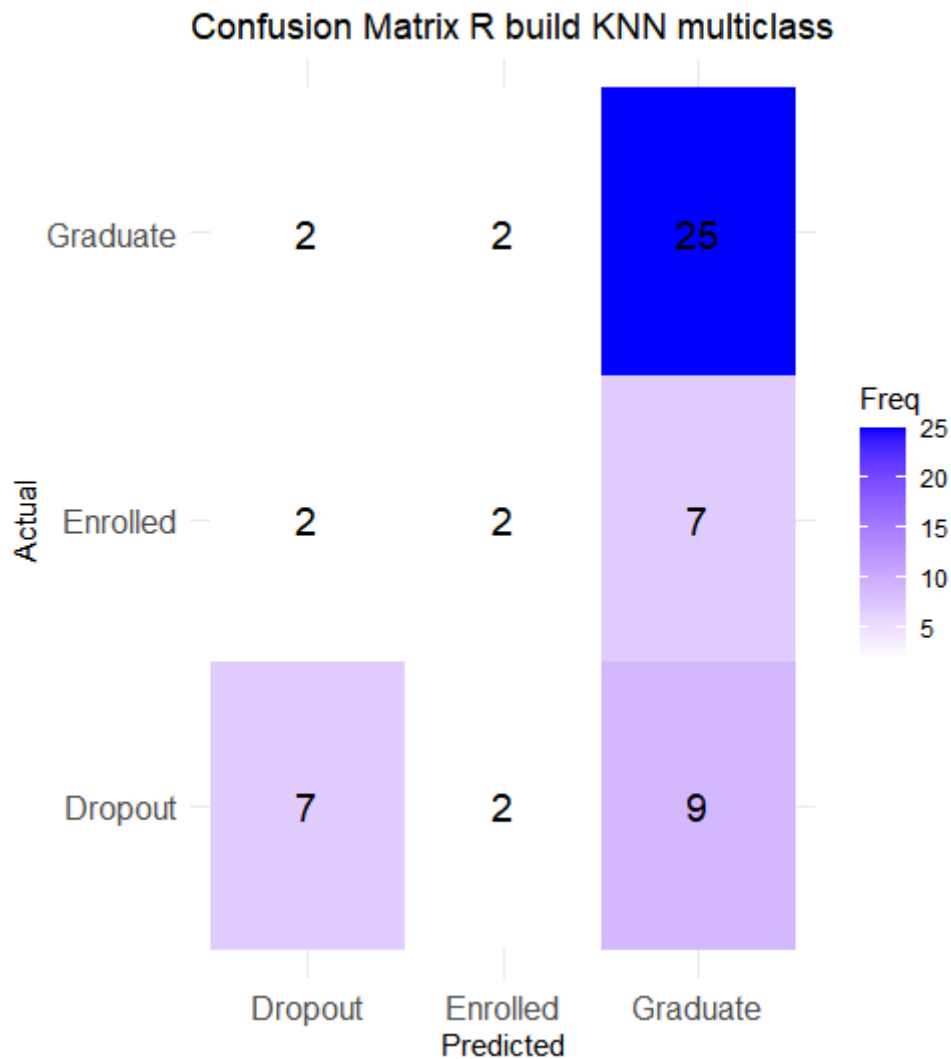
$Accuracy
[1] 0.5862069

$Precision
      Dropout Enrolled Graduate
0.3888889 0.1818182 0.8620690

$Recall
      Dropout Enrolled Graduate
0.6363636 0.3333333 0.6097561

$F1_macro
[1] 0.4774462

$AUC_multi
[1] 0.6693661
```

Sieci neuronowe

Sieci neuronowe są modelami inspirowanymi biologicznymi neuronami. Wykorzystują warstwy ukryte, funkcje aktywacji oraz algorytmy optymalizacyjne do nauki relacji w danych. Są aktualnie najpopularniejszym algorytmem, który jest wykorzystywany w wielu problemach między innymi w klasyfikacji oraz regresji.

Analiza hiperparametrów

Kluczowe hiperparametry: **Liczba neuronów w warstwach ukrytych** – większa liczba neuronów pozwala uchwycić bardziej złożone zależności, ale może prowadzić do przeuczenia. **Liczba epok (iteracji)** zbyt duża wartość może prowadzić do nadmiernego dopasowania. **Funkcja aktywacji** – sigmoid, tanh czy ReLU wpływają na szybkość i skuteczność uczenia.

Porównanie wyników

Własna implementacja sieci neuronowych uzyskała wyniki zbliżone do neuralnet, jednak wbudowany algorytm pozwala na lepszą optymalizację i stabilność wyników. W crosswalidacji w przypadku sieci neuronowej testowane były parametry takie jak iter, learning_rate, funkcja aktywacji w warstwach ukrytych oraz liczba neuronów w warstwach ukrytych.

Regresja

Wyniki były dobre w przypadku regresji- MAPE wynosiło około 16,33%.

Kolumna1	activation_hidden	lr	iter	seed	activation_output	hidden_units	MAE_t	MSE_t	MAPE_t	MAE_w	MSE_w	MAPE_w
1	sigmoid		0,01	10000	123 linear	64,32	3,17	19,51	16,10	3,42	22,28	17,18
2	sigmoid		0,01	10000	123 linear	128,64	3,07	18,78	15,29	3,35	22,03	16,55
3	sigmoid		0,001	10000	123 linear	64,32	3,15	20,03	15,82	3,40	22,86	16,94
4	sigmoid		0,001	10000	123 linear	128,64	3,17	20,69	15,84	3,41	23,49	16,98

```
MAE      MSE      MAPE
3.636175 30.220028 16.336224
```

Natomiast w przypadku algorytmu wbudowanego wartość MAPE wynosiła 10,95% co jest już wynikiem sporo lepszym.

```
MAE      MSE      MAPE
2.216012 11.219540 10.953995
```

Klasyfikacja binarna

W przypadku klasyfikacji binarnej problem został rozwiązany niemalże perfekcyjnie. Wyniki na zbiorze walidacyjnym i treningowym były idealne, co potwierdziło się później również na zbiorze testowym. Najlepszymi parametrami był learning rate=0.01, liczba neuronów odpowiednio w pierwszej i drugiej warstwie ukrytej 32 i 16 oraz funkcja aktywacji w warstwach ukrytych sigmoid.

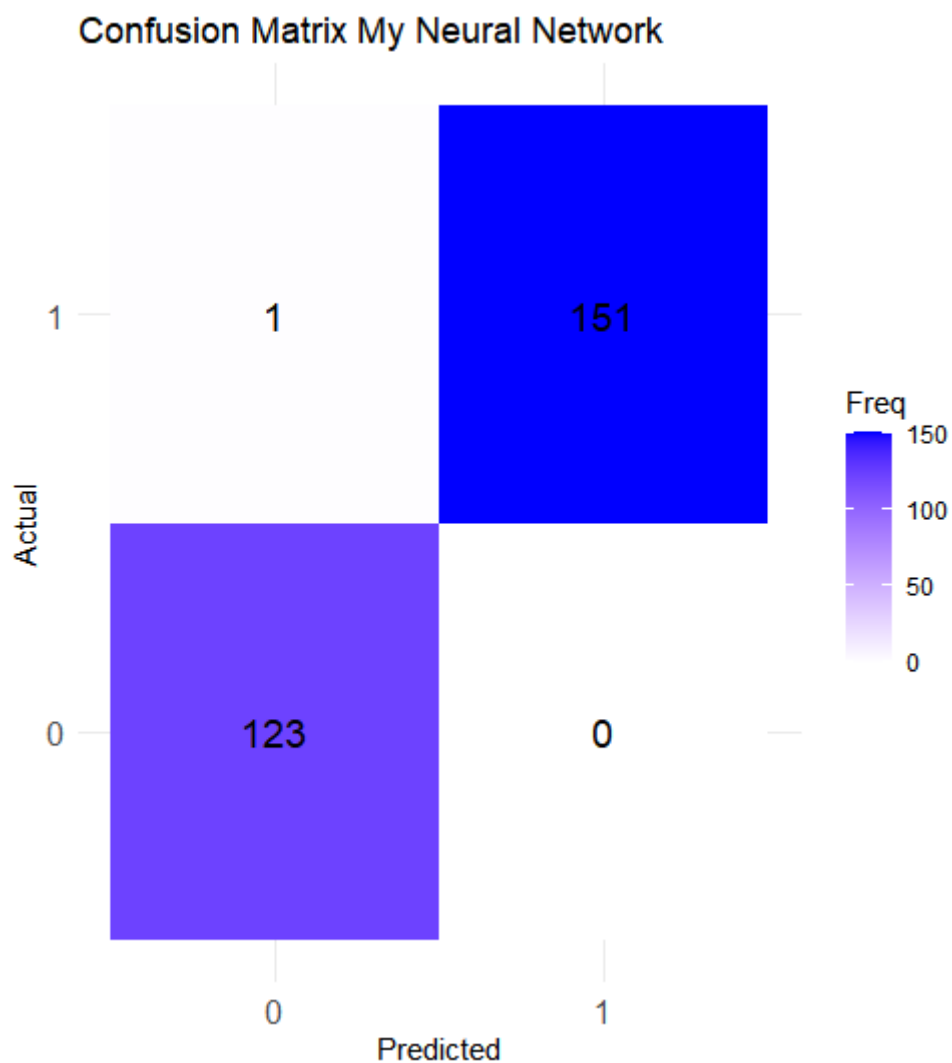
Kolumna1	activation_hidden	lr	iter	seed	activation_output	hidden_units	AUC_t	Sensitivity_t	Specificity_t	Accuracy_t	AUC_w	Sensitivity_w	Specificity_w	Accuracy_w
1	sigmoid		0,01	10000	123 sigmoid	32,16	1	1	1	1	1	1	1	1
2	tanh		0,01	10000	123 sigmoid	32,16	0,733596	0,6123913	0,9236602	0,7819672	0,73391	0,6067302	0,9492276	0,7907104

Jak widać tylko 1 obserwacja została źle zaklasyfikowana przez model.

```
$ConfMat
      predicted
actual  0    1
      0 123    0
      1   1 151

$Youden
[1] 0.9934211 0.4479868

$Metrics
      AUC Sensitivity Specificity Accuracy
0.9999465 1.0000000 0.9934211 0.9963636
```

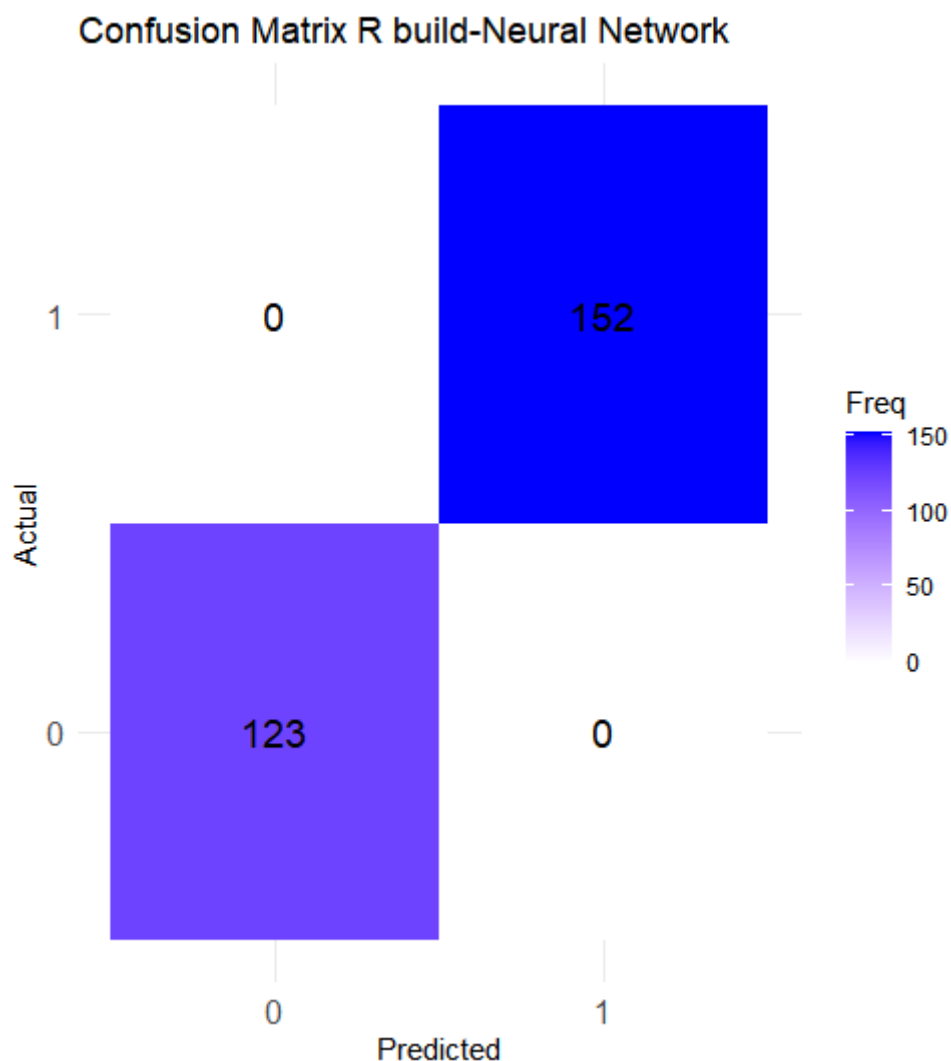


Natomiast w przypadku wbudowanej sieci neuronowej wyniki były niemal identyczne. A tutaj pojawił się idealny klasyfikator.

```
$ConfMat
      predicted
actual 0 1
0 123 0
1 0 152

$Youden
[1] 1.0000000 0.9936643

$Metrics
      AUC Sensitivity Specificity Accuracy
      1         1         1         1
```



Klasyfikacja wieloklasowa

W przypadku opracowanego algorytmu klasyfikator wieloklasowy nie był zbyt dobry, niestety złożoność obliczeniowa nie pozwoliła na użycie pochodnej funkcji softmax, przez co wyniki uległy prawdopodobnie pogorszeniu. Niemniej problem był również bardzo ciężki również dla innych klasyfikatorów co w połączeniu złożyło się na wynik ostateczny. Accuracy na zbiorze walidacyjnym największą wartość osiągnęło dla ukrytej funkcji aktywacji sigmoid, lr=0.01, iter=10000 oraz odpowiednio 32 i 16 neuronów w warstwach.

Kolumna1	activation_hidden	lr	iter	seed	activation_output	hidden_units	AUC_t	Accuracy_t	AUC_w	Accuracy_w
1	sigmoid		0,01	1000	123 softmax	10,5	0,811	0,729	0,807	0,727
2	sigmoid		0,01	1000	123 softmax	32,16	0,823	0,737	0,820	0,734
3	tanh		0,01	1000	123 softmax	10,5	0,781	0,674	0,780	0,676
4	tanh		0,01	1000	123 softmax	32,16	0,764	0,552	0,763	0,554
5	sigmoid		0,001	1000	123 softmax	10,5	0,754	0,575	0,753	0,577
6	sigmoid		0,001	1000	123 softmax	32,16	0,748	0,587	0,748	0,588
7	tanh		0,001	1000	123 softmax	10,5	0,737	0,579	0,737	0,580
8	tanh		0,001	1000	123 softmax	32,16	0,733	0,530	0,734	0,531
9	sigmoid		0,0001	1000	123 softmax	10,5	0,714	0,513	0,715	0,514
10	sigmoid		0,0001	1000	123 softmax	32,16	0,696	0,481	0,698	0,482
11	tanh		0,0001	1000	123 softmax	10,5	0,684	0,469	0,685	0,471
12	tanh		0,0001	1000	123 softmax	32,16	0,671	0,446	0,673	0,447

Na zbiorze testowym accuracy sięgało 75% co wskazuje na niezły klasyfikator w kontekście tego zbioru.

```
$Matrix
      predicted
actual Dropout Enrolled Graduate
Dropout    139      16      22
Enrolled    27      24      55
Graduate    15      15     285

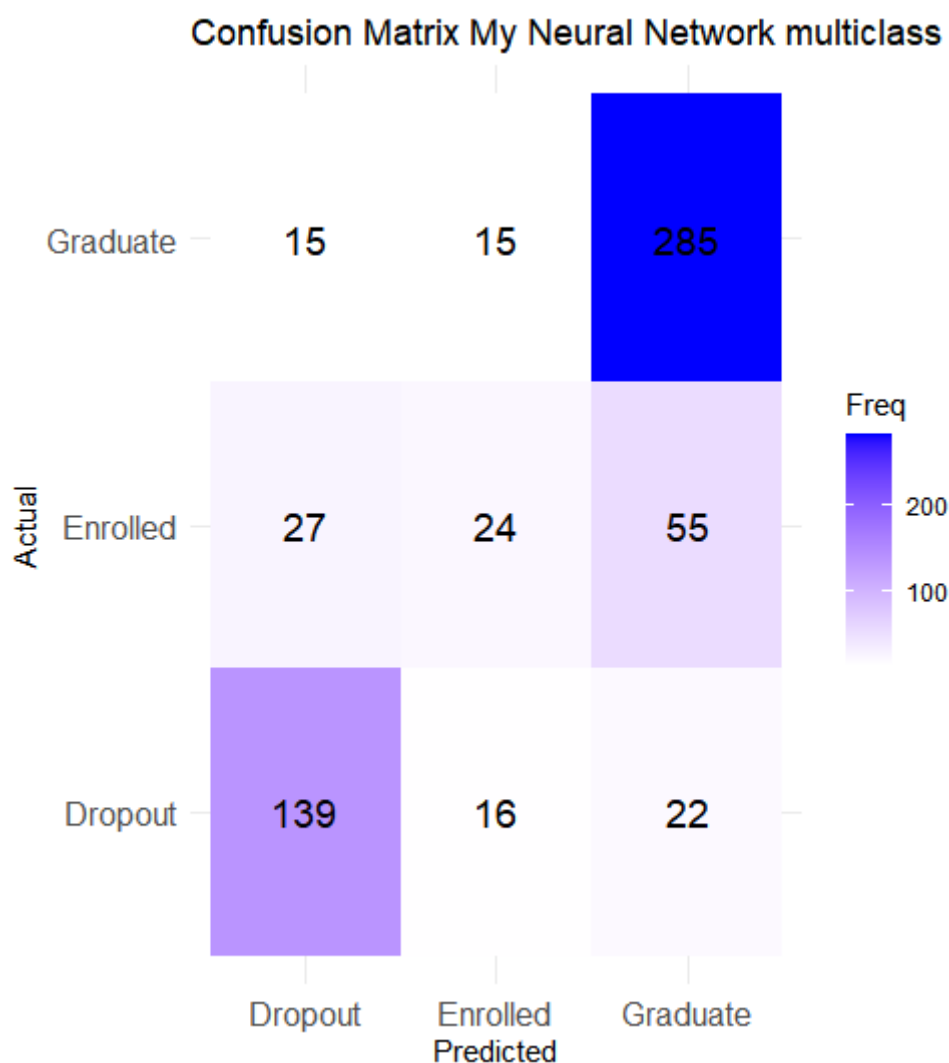
$Accuracy
[1] 0.7491639

$Precision
      Dropout Enrolled Graduate
0.7853107 0.2264151 0.9047619

$Recall
      Dropout Enrolled Graduate
0.7679558 0.4363636 0.7872928

$F1_macro
[1] 0.6388742

$AUC_multi
[1] 0.8267572
```



W przypadku wbudowanego algorytmu zostały wykorzystane odpowiednio 64 oraz 32 neurony. Pomimo braku pełnej poprawności opracowanego modelu, model wbudowany okazał się dawać gorsze wyniki w tym konkretnym problemie. Accuracy wynosiło około 67,5%.

```

$Matrix
      predicted
actual Dropout Enrolled Graduate
Dropout    122     31     24
Enrolled    22     34     50
Graduate    22     45    248

$Accuracy
[1] 0.6755853

$Precision
      Dropout Enrolled Graduate
0.6892655 0.3207547 0.7873016

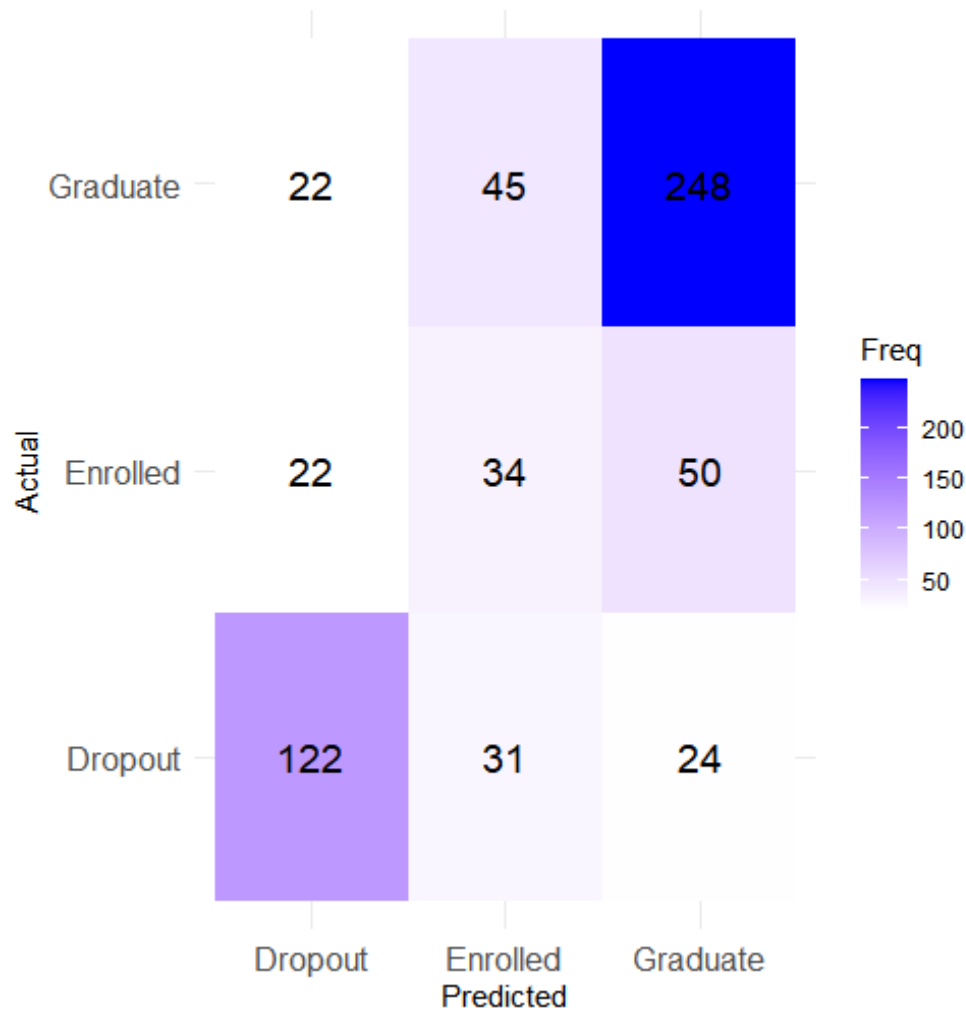
$Recall
      Dropout Enrolled Graduate
0.7349398 0.3090909 0.7701863

$F1_macro
[1] 0.6016117

$AUC_multi
[1] 0.7731677

```

Confusion Matrix R build Neural Network multiclass



Podsumowanie

Podsumowując algorytmy zbudowane od zera dawały naprawdę zbliżone wyniki do tych z pakietów dostępnych w języku R. Natomiast były sporo wolniejsze. Bardzo przydatnym narzędziem okazała się być crosswalidacja, która pozwoliła na dobranie hiperparametrów modelu bez zgadywania czy są one odpowiednie. Najwolniejszym algorytmem okazało się być KNN a następnie sieci neuronowe, które często potrzebowały dużo iteracji żeby osiągnąć dobry efekt. Zwiększanie liczby neuronów było dobre do pewnego momentu, gdy wyniki zaczęły się polepszać w dużo mniejszym stopniu niż złożoność czasowa. W drzewach natomiast ważnym było przycinanie oraz ustawianie maksymalnej głębokości drzewa jak i `min_samples` aby drzewo nie było zbyt mocno dopasowane do danych uczących. KNN natomiast sprawdzało się bardzo dobrze między innymi przez dodanie warunku z użyciem odległości Gowera. Do tego wszystkiego podczas klasyfikacji binarnej wyliczany był indeks Youdena, aby odnaleźć optymalne odcięcie progu decyzyjnego. Niestety nie było możliwym pozostawienie w kodzie wszystkich testowanych kombinacji hiperparametrów, gdy jego odpalenie zajęłoby kilka godzin. Sprawdzanie dokładnego czasu działania algorytmów zostało tu pominięte z racji, że gołym okiem widać, który chodź najdłużej, a który najkrócej oraz, że metody wbudowane są dużo szybsze.