Projekt Metody Sztucznej Inteligencji
Informatyka 2rok 4sem.

Prowadzący: Mirosław Kordos

Klasyfikacja Drzewem Decyzyjnym
Przemysław Dyrcz

Projekt nr 1: Klasyfikatory i modele regresyjne z selekcją danych

Część 1

Drzewo decyzyjne jest modelem uczenia maszynowego, który służy do podejmowania decyzji na podstawie zestawu reguł decyzyjnych wyodrębnionych z danych treningowych. Oto ogólny opis kroków, jak działa drzewo decyzyjne:

Krok 1: Podział danych treningowych:

Na początku, wszystkie dane treningowe są na tym samym poziomie drzewa.
Algorytm wybiera atrybut (cechę), który najlepiej dzieli dane na bardziej jednorodne grupy.

Krok 2: Wybór kryterium podziału:

Algorytm wybiera kryterium (np. entropia, czystość Gini), które najlepiej różnicuje klasy lub redukuje niepewność w każdym węźle drzewa.

Krok 3: Tworzenie węzłów:

Po wybraniu atrybutu i kryterium podziału, tworzone są węzły drzewa dla każdej możliwej wartości tego atrybutu.

Dane treningowe są podzielone na podzbiory według wartości wybranego atrybutu.

Krok 4: Powtarzanie procesu:

Proces podziału i tworzenia węzłów jest powtarzany dla każdego utworzonego węzła, dopóki nie zostaną spełnione pewne kryteria stopu (np. maksymalna głębokość drzewa, minimalna liczba obserwacji w węźle).

Krok 5: Budowa drzewa:

Proces tworzenia węzłów kontynuuje się, aż wszystkie dane treningowe zostaną dokładnie sklasyfikowane lub osiągnięte zostaną kryteria stopu.

Krok 6: Przycinanie drzewa (opcjonalne):

Po zbudowaniu drzewa można zastosować proces przycinania, który polega na usuwaniu niepotrzebnych gałęzi, aby zmniejszyć ryzyko przeuczenia modelu.

Krok 7: Klasyfikacja/regresja:

Gdy drzewo jest już zbudowane, można użyć go do klasyfikacji lub regresji nowych danych poprzez przekazanie ich przez strukturę drzewa i przypisanie do odpowiednich klas lub wartości.

Drzewa decyzyjne są stosunkowo łatwe do interpretacji i mogą być używane do zarówno klasyfikacji, jak i regresji w różnych dziedzinach, takich jak medycyna, biznes czy przetwarzanie sygnałów. Ich głęboka struktura pozwala na efektywne wyodrębnianie reguł decyzyjnych z danych.

W RapidMinerze, w kontekście algorytmu drzew decyzyjnych, terminologia dotycząca kryteriów podziału jest zbliżona do terminologii używanej w innych narzędziach i frameworkach do uczenia maszynowego. Oto krótkie wyjaśnienie każdego z tych kryteriów:

Accuracy: W kontekście drzew decyzyjnych, "accuracy" oznacza poprawność klasyfikacji, czyli odsetek poprawnie sklasyfikowanych przypadków w stosunku do ogólnej liczby przypadków. W praktyce, im wyższa wartość "accuracy", tym lepiej model radzi sobie z klasyfikacją.

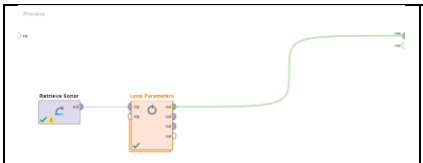
Gain Ratio: "Gain Ratio" (stosunek zysku) to miara używana do oceny jakości podziału w drzewie decyzyjnym. Jest ona oparta na stosunku między "information gain" (zyskiem informacyjnym) a "split information" (informacją o podziale). Zysk informacyjny mierzy

zmniejszenie niepewności klasyfikacji po podziale węzła drzewa, a split information mierzy stopień, do jakiego podział jest równomierny.

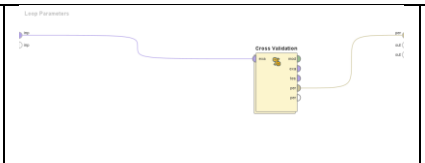
Gini Index: Indeks Giniego jest miarą zanieczyszczenia węzła drzewa decyzyjnego. Im niższy indeks Giniego, tym lepiej węzeł jest czysty, co oznacza, że jest lepiej oddzielony od innych klas. Indeks Giniego mierzy prawdopodobieństwo, że losowo wybrana próbka zostanie źle sklasyfikowana.

Information Gain: Zysk informacyjny to miara, która określa, jak bardzo podział węzła drzewa zmniejsza niepewność klasyfikacji. Jest obliczany jako różnica między niepewnością przed podziałem a niepewnością po podziale. Im większy zysk informacyjny, tym lepszy podział.

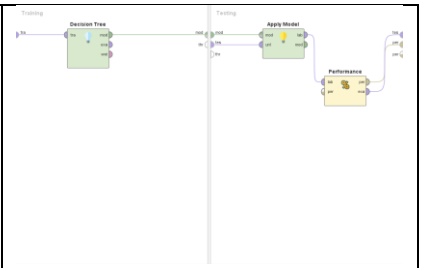
W przypadku RapidMinera, możesz wybierać spośród tych kryteriów, aby określić, które będą stosowane podczas tworzenia drzewa decyzyjnego w Twoim modelu. Każde z tych kryteriów ma swoje zalety i zastosowania w zależności od charakterystyki danych i celu modelu.



Rys.1 Process

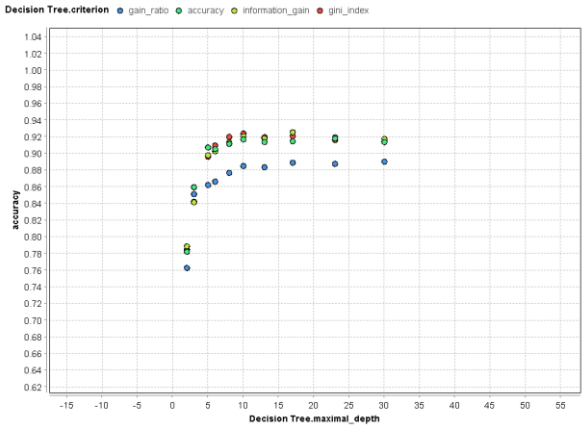


Rys.2 Loop Parameters



Rys. Cross Validation

Testowanie modelu na danych: “sonar”



Rys.1 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”sonar”

Loop Parameters (40 rows, 4 columns)

iteration	Decisio...	Decisio...	accu...
26	informatio...	13	0.783
18	informatio...	8	0.769
14	informatio...	6	0.764
19	gini_index	8	0.754
10	informatio...	5	0.751
24	accuracy	10	0.750
11	gini_index	5	0.750
30	informatio...	17	0.745
31	gini_index	17	0.740
6	informatio...	3	0.736
20	accuracy	8	0.736
32	accuracy	17	0.736
4	accuracy	2	0.735
22	informatio...	10	0.726
39	gini_index	30	0.726
40	accuracy	30	0.721

Rys.2 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”sonar”.

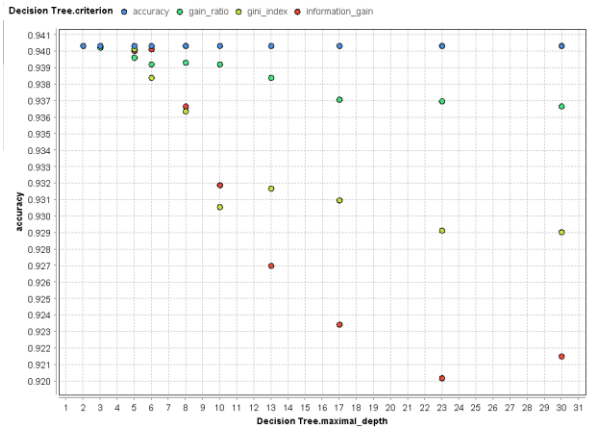
Wnioski:

-Dla kryterium information_gain oraz maksymalnej głębokości 13, wystąpiła najwyższa wartość trafności modelu wynosząca 0.783,

-im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości

kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 10, nie poprawiała wyników,
-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”,

Testowanie modelu dla danych “coil2000”:



Rys.3 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”coil2000”

Loop Parameters (40 rows, 4 columns)

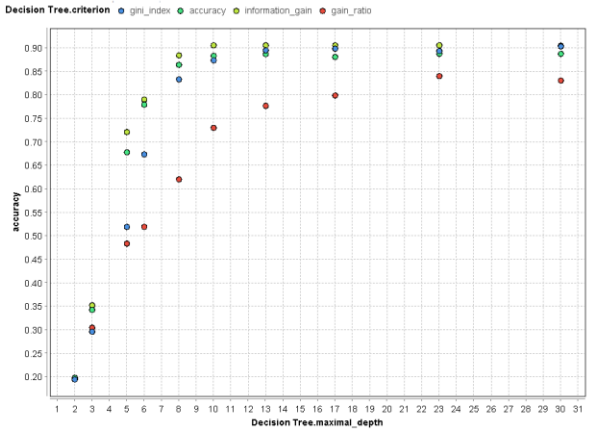
.	Decisio...	Decisio...	accu...
.	10	accuracy	0.940
.	2	accuracy	0.940
.	2	gini_index	0.940
.	2	gain_ratio	0.940
.	2	informatio...	0.940
.	13	accuracy	0.940
.	3	accuracy	0.940
.	5	accuracy	0.940
.	3	gini_index	0.940
.	3	informatio...	0.940
.	17	accuracy	0.940
.	6	accuracy	0.940
.	8	accuracy	0.940
.	23	accuracy	0.940
.	30	accuracy	0.940
.	3	gain_ratio	0.940

Rys.4 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”coil2000”

Wnioski:
-Dla kryterium “accuracy” oraz maksymalnej głębokości 10, wystąpiła najwyższa wartość trafności modelu wynosząca 0.940,
-dla kryterium “accuracy” wartosc maksymalnej długości liścia nie ma wpływu natomiast widać, że dla pozostałych kryteriów im wiekśza wartość maksymalnej długości liścia tym

gorsza trafność modelu,

Testowanie modelu na danych: “optdigits”



Rys.5 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”optdigits”

Loop Parameters (40 rows, 4 columns)

iteration	Decisio...	Decisio...	accu...
19	23	informatio...	0.906
16	10	informatio...	0.906
17	13	informatio...	0.906
18	17	informatio...	0.906
20	30	informatio...	0.905
30	30	gini_index	0.903
28	17	gini_index	0.899
27	13	gini_index	0.895
29	23	gini_index	0.894
39	23	accuracy	0.888
40	30	accuracy	0.888
37	13	accuracy	0.887
15	8	informatio...	0.884
36	10	accuracy	0.883
38	17	accuracy	0.881
26	10	gini_index	0.874

Rys.6 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”optdigits”.

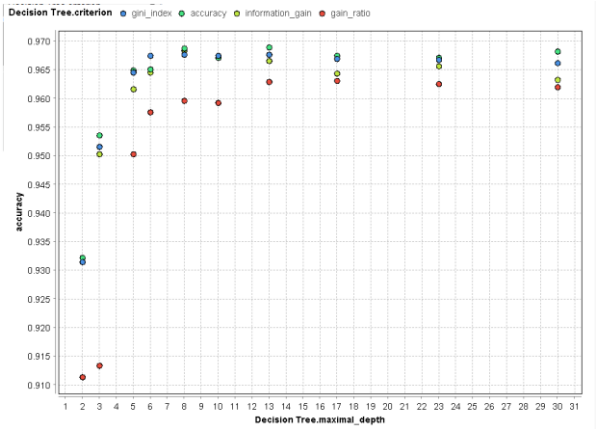
Wnioski:

-dla kryterium information_gain oraz maksymalnej głębokości 23, wystąpiła najwyższa wartość trafności modelu wynosząca 0.906,

-im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 10, nie poprawiała

wyników
-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”,

Testowanie modelu na danych: “page-blocks”



Rys.7 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”page-blocks”

Loop Parameters (40 rows, 4 columns)

..	Decisio...	Decisio...	accu...
.	13	accuracy	0.969
.	8	accuracy	0.969
.	8	informatio...	0.968
.	30	accuracy	0.968
.	13	gini_index	0.968
.	8	gini_index	0.968
.	6	gini_index	0.967
.	10	gini_index	0.967
.	17	accuracy	0.967
.	10	informatio...	0.967
.	23	accuracy	0.967
.	10	accuracy	0.967
.	17	gini_index	0.967
.	23	gini_index	0.967
.	13	informatio...	0.967
.	30	gini_index	0.966

Rys.8 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”page-blocks”

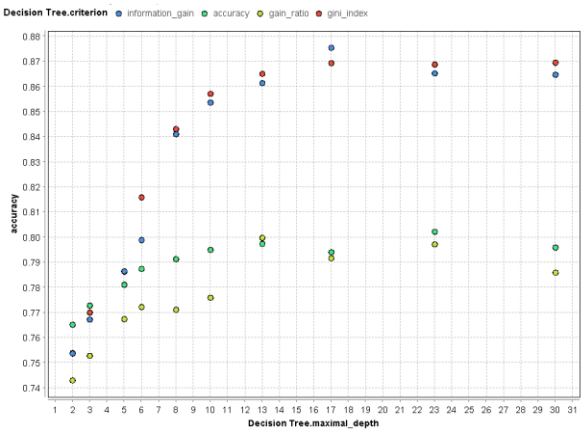
Wnioski:

-dla kryterium "accuracy” oraz maksymalnej głębokości 13, wystąpiła najwyższa wartość trafności modelu wynosząca 0.969,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 6, nie poprawiała wyników

-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”,

Testowanie modelu na danych: “phoneme”



Rys.9 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”phoneme”

Loop Parameters (40 rows, 4 columns)

..	Decisio...	Decisio...	accu...
.	17	informatio...	0.875
.	30	gini_index	0.870
.	17	gini_index	0.869
.	23	gini_index	0.869
.	23	informatio...	0.865
.	13	gini_index	0.865
.	30	informatio...	0.865
.	13	informatio...	0.861
.	10	gini_index	0.857
.	10	informatio...	0.854
.	8	gini_index	0.843
.	8	informatio...	0.841
.	6	gini_index	0.816
.	23	accuracy	0.802
.	13	gain_ratio	0.800
.	6	informatio...	0.799

Rys.10 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”phoneme”

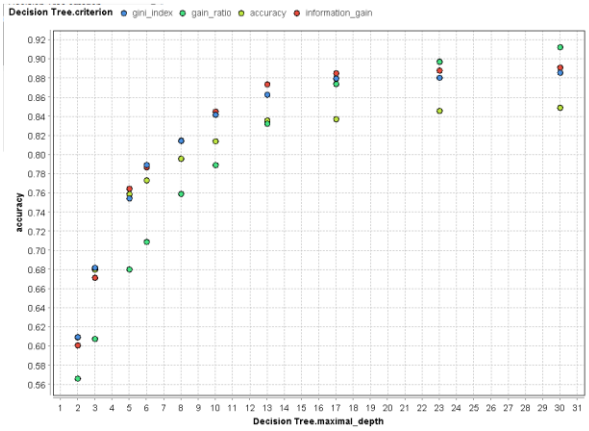
Wnioski:

-Dla kryterium information_gain oraz maksymalnej głębokości 17, wystąpiła najwyższa wartość trafności modelu wynosząca 0.875,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 13, nie poprawiała wyników

-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”, dla najgorszego kryterium oraz “accracy” dokładność zatrzymałą się w granicach 0.7-0.8 na głębokości liścia 6-10. Powyżej maksymalnej głębokości liścia wynoszącej 10 trafność nie zmieniała się znacząco,

Testowanie modelu na danych: “ring”



Rys.11 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”ring”.

Loop Parameters (40 rows, 4 columns)

..	Decisio...	Decisio...	accu...
.	30	gain_ratio	0.912
.	ta in a table	gain_ratio	0.897
.	30	informatio...	0.891
.	23	informatio...	0.888
.	30	gini_index	0.886
.	17	informatio...	0.885
.	23	gini_index	0.881
.	17	gini_index	0.880
.	17	gain_ratio	0.874
.	13	informatio...	0.874
.	13	gini_index	0.863
.	30	accuracy	0.849
.	23	accuracy	0.846
.	10	informatio...	0.845
.	10	gini_index	0.842
.	17	accuracy	0.837

Rys.12 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”ring”.

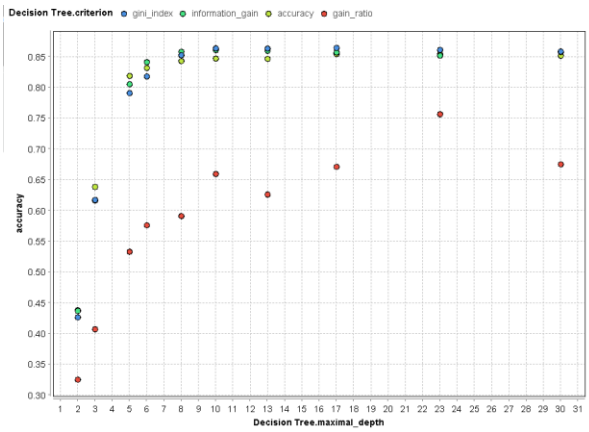
Wnioski:

-Dla kryterium gain_ratio oraz maksymalnej głębokości 30, wystąpiła najwyższa wartość trafności modelu wynosząca 0.912,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 17, nie poprawiała wyników

-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”, natomiast dla głębokości liścia wynoszącej 10 w górę, najgorszym kryterium okazuje się “accurcy”,

Testowanie modelu na danych: “satimage”:



Rys.13 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”satimage”.

Loop Parameters (40 rows, 4 columns)

.	Decisio...	Decisio...	accu...
.	17	gini_index	0.864
.	10	gini_index	0.864
.	13	gini_index	0.863
.	23	gini_index	0.861
.	10	informatio...	0.861
.	13	informatio...	0.860
.	30	gini_index	0.859
.	8	informatio...	0.858
.	30	informatio...	0.858
.	17	informatio...	0.857
.	23	accuracy	0.855
.	17	accuracy	0.854
.	8	gini_index	0.853
.	23	informatio...	0.852
.	30	accuracy	0.852
.	10	accuracy	0.847

Rys.14 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”satimage”.

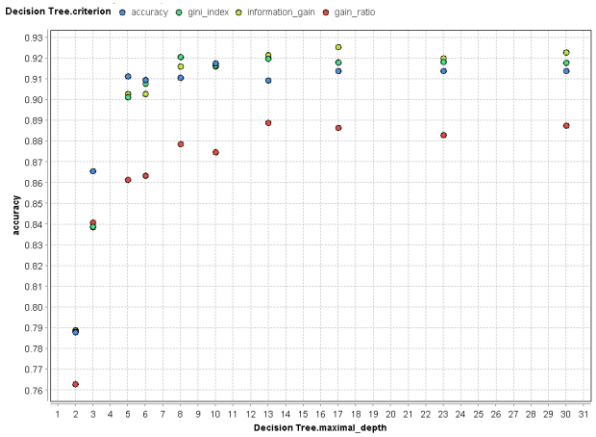
Wnioski:

-Dla kryterium gain_index oraz maksymalnej głębokości 17, wystąpiła najwyższa wartość trafności modelu wynosząca 0.864,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej między 7 a 8, nie poprawiała wyników,

-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”,

Testowanie modelu na danych: “spambase”:



Rys.15 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”spambase”.

Loop Parameters (40 rows, 4 columns)

..	Decisio...	Decisio...	accu...
.	17	informatio...	0.925
.	30	informatio...	0.923
.	13	informatio...	0.921
.	8	gini_index	0.921
.	23	informatio...	0.920
.	13	gini_index	0.920
.	23	gini_index	0.918
.	17	gini_index	0.918
.	30	gini_index	0.918
.	10	accuracy	0.918
.	10	gini_index	0.916
.	8	informatio...	0.916
.	10	informatio...	0.916
.	23	accuracy	0.914
.	30	accuracy	0.914
.	17	accuracy	0.914

Rys.16 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”spambase”.

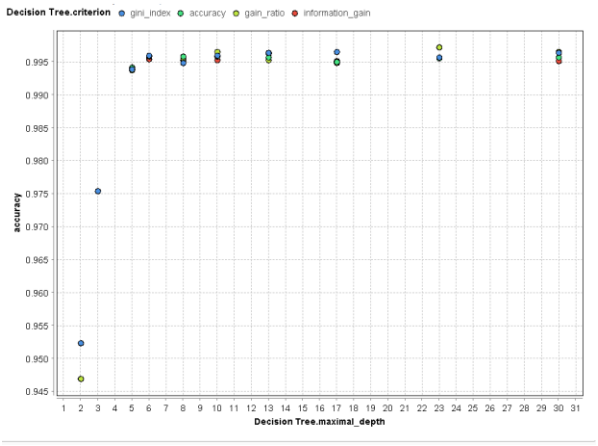
Wnioski:

-Dla kryterium information_gain oraz maksymalnej głębokości 17, wystąpiła najwyższa wartość trafności modelu wynosząca 0.925,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 5, nie poprawiała wyników,

-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”,

Testowanie modelu na danych: “thyroid”



Rys.17 Wykres zależności głębokości drzewa od trafności klasyfikacji

Loop Parameters (40 rows, 4 columns)

..	Decisio...	Decisio...	accu...
.	23	gain_ratio	0.997
.	10	gain_ratio	0.997
.	17	gini_index	0.997
.	30	gain_ratio	0.997
.	13	gini_index	0.996
.	30	gini_index	0.996
.	13	informatio...	0.996
.	10	gini_index	0.996
.	6	gain_ratio	0.996
.	6	gini_index	0.996
.	8	accuracy	0.996
.	10	accuracy	0.996
.	6	accuracy	0.996
.	23	gini_index	0.996
.	13	accuracy	0.996
.	30	accuracy	0.996

Rys.18 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”thyroid”.

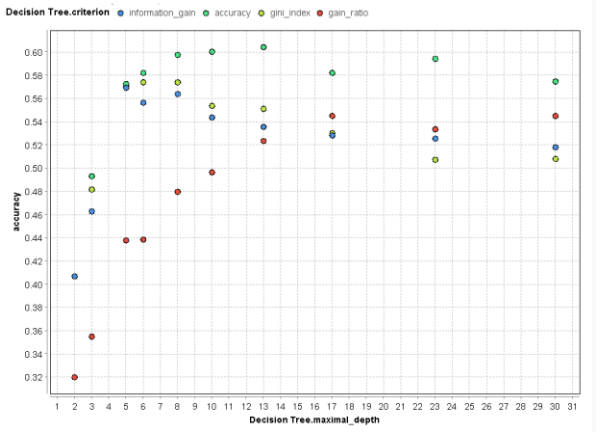
Wnioski:

-Dla kryterium gain_ratio oraz maksymalnej głębokości 23, wystąpiła najwyższa wartość trafności modelu wynosząca 0.997,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 5, nie poprawiała wyników

-dla tego modelu danych, wszystkie kryteria dobrze sobie poradziły,

Testowanie modelu na danych: “yeast”



Rys.19 Wykres zależności głębokości drzewa od trafności klasyfikacji dla zbioru ”yeast”.

Loop Parameters (40 rows, 4 columns)

..	Decisio...	Decisio...	accu...
.	13	accuracy	0.604
.	10	accuracy	0.600
.	8	accuracy	0.598
.	23	accuracy	0.594
.	17	accuracy	0.582
.	6	accuracy	0.582
.	30	accuracy	0.575
.	6	gini_index	0.574
.	8	gini_index	0.574
.	5	accuracy	0.573
.	5	gini_index	0.571
.	5	informatio...	0.569
.	8	informatio...	0.564
.	6	informatio...	0.557
.	10	gini_index	0.554
.	13	gini_index	0.551

Rys.20 Wykaz przedstawiajacy kryterium, głębokość drzewa, oraz trafność klasyfikacji dla zbioru ”yeast”.

Wnioski:

-Dla kryterium “accuracy” oraz maksymalnej głębokości 13, wystąpiła najwyższa wartość trafności modelu wynosząca 0.604,

- im większa maksymalna głębokość liścia tym większa dokładność modelu, dla większości kryteriów po osiągnięciu maksymalnej długości liścia wynoszącej 10, nie poprawiała wyników,

-najgorsze wyniki osiągneło w tym przypadku kryterium “gain_ratio”,

Wnioski ogólne z pierwszej częsci projektu:

- Wybór kryterium podziału: Wartości trafności modelu różniły się w zależności od użytego kryterium podziału. Nie ma jednoznacznie najlepszego kryterium dla wszystkich zbiorów danych. Dla niektórych zbiorów lepsze wyniki osiągane były przy użyciu "accuracy", dla innych "information_gain" lub "gain_ratio".
- Maksymalna głębokość drzewa: Zwiększanie maksymalnej głębokości drzewa często prowadziło do poprawy trafności klasyfikacji, ale tylko do pewnego punktu. Po osiągnięciu pewnej głębokości dalsze zwiększanie jej nie miało już istotnego wpływu lub nawet mogło prowadzić do pogorszenia wyników, co sugeruje możliwość przeuczenia modelu.
- Wpływ kryterium na różne zbiory danych: Dla niektórych zbiorów danych pewne kryteria lepiej się sprawdzały niż inne. Na przykład, dla zbioru "thyroid" wszystkie kryteria osiągnęły wysoką trafność, podczas gdy dla zbioru "yeast" kryterium "accuracy" dawało lepsze wyniki niż "gain_ratio".
- Potrzeba przycinania drzewa: W niektórych przypadkach, szczególnie gdy drzewo osiągało dużą głębokość, istniała potrzeba przycinania drzewa w celu uniknięcia przeuczenia modelu i poprawy jego ogólnej wydajności na danych testowych.
- Znaczenie odpowiedniego doboru parametrów: Wybór odpowiednich parametrów modelu, takich jak kryterium podziału czy maksymalna głębokość drzewa, ma istotny wpływ na efektywność i trafność klasyfikacji. Warto przeprowadzić badania empiryczne na różnych zbiorach danych, aby dobrać optymalne parametry dla konkretnego przypadku.

Podsumowanie:

Wnioski te pokazują, że skuteczność modeli drzew decyzyjnych zależy od wielu czynników, takich jak charakterystyka danych, wybór kryteriów podziału oraz odpowiedni dobór parametrów. Dlatego też ważne jest przeprowadzenie eksperymentów na różnych zbiorach danych i analiza wyników pod kątem optymalizacji modelu dla konkretnego przypadku zastosowania.

Część 2

Celem tej części projektu jest dokonanie analizy jednego wybranego algorytmu selekcji wektorów i jednego wybranego filtra cech na działanie klasyfikatora lub modelu regresyjnego przebadanego w pierwszej części. Należy przebadać trzy przypadki: a) bez selekcji danych b) z selekcją cech (z filtrem cech) c) z selekcją wektorów d) z selekcją zarówno cech, jak i wektorów, określić czego selekcja powinna być najpierw.

Algorytmy użyte w projekcie:

Select by CNN (Correlation-based Feature Selection - CFS)

Typ: Selekcja cech

Opis: Algorytm CFS wybiera cechy na podstawie analizy korelacji między cechami a klasami wyjściowymi oraz między samymi cechami. Dąży do znalezienia zestawu cech, które są wysoko skorelowane z klasą wyjściową, ale nisko skorelowane między sobą.

Kroki działania:

1. Oblicza korelacje między cechami a klasą wyjściową oraz między cechami nawzajem.
2. Tworzy różne podzbiory cech i ocenia je na podstawie średniej korelacji z klasą oraz średniej korelacji między cechami.

3. Wybiera podzbiór cech, który maksymalizuje wartość miary Merit, tj. wysoką korelację z klasą i niską redundancję między cechami.

Weight by Information Gain (IG)

Typ: Selekcja cech

Opis: Algorytm wagowania cech na podstawie Information Gain ocenia istotność cech poprzez mierzenie, jak dobrze każda cecha redukuje niepewność (entropię) w klasyfikacji. Cecha z wysokim Information Gain jest bardziej informatywna dla klasyfikacji.

Kroki działania:

4. Oblicza początkową entropię całego zbioru danych.
5. Dla każdej cechy oblicza entropię warunkową, dzieląc zbiór danych na podzbiory na podstawie wartości tej cechy.
6. Oblicza Information Gain dla każdej cechy jako różnicę między początkową entropią a średnią ważoną entropią warunkową.
7. Cecha o najwyższym Information Gain jest uznawana za najbardziej istotną.

Select by Random

Typ: Selekcja wektorów

Opis: Algorytm Select by Random wybiera losowo próbki (wiersze) z zestawu danych. Jest to prosta i szybka metoda, często stosowana jako punkt odniesienia do porównania z bardziej zaawansowanymi technikami selekcji.

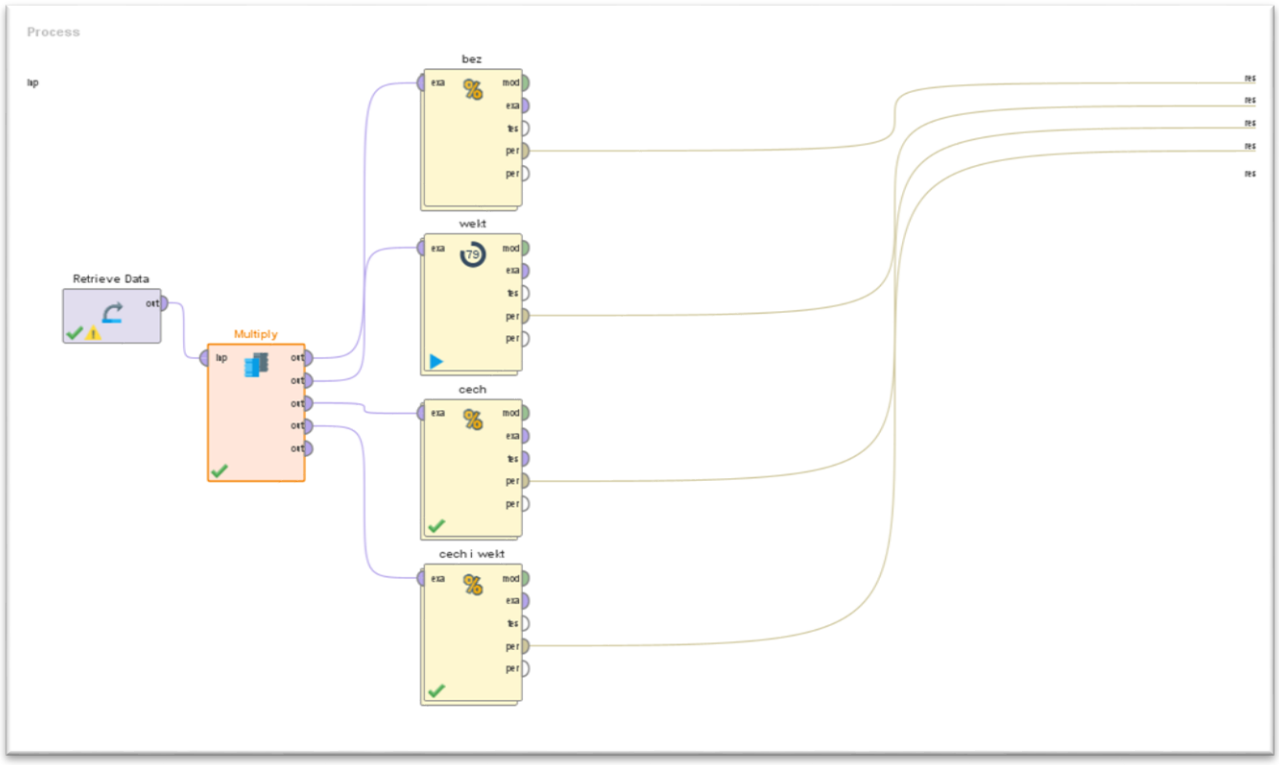
Kroki działania:

8. Losowo wybiera określoną liczbę próbek (wierszy) z zestawu danych.
9. Używa wybranych próbek do trenowania modelu i ocenia jego wydajność.
10. Proces może być powtórzony wielokrotnie, a średnia wydajność modeli używana jako punkt odniesienia.

Podsumowanie:

- **Select by CNN (CFS)**: Wybiera cechy na podstawie korelacji, dążąc do wysokiej korelacji z klasą i niskiej redundancji między cechami.

- **Weight by Information Gain (IG)**: Wybiera cechy na podstawie miary Information Gain, oceniając istotność cech w kontekście redukcji niepewności.
- **Select by Random**: Losowo wybiera próbki z zestawu danych, stosowane jako metoda bazowa lub do selekcji wektorów.



Rys.21 Podgląd ogólny procesu ułożonego w programie RapidMiner.

Podgląd na procesy dla kazdego przypadku.

 Bez selekcji danych	 Z selekcją cech	 Z selekcją wektorów	 Z selekcja cech i wektorów

Testowanie modelu dla danych “coil2000”:

Wyniki:

Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 87.69% +/- 2.21% (micro average: 87.69%)
ConfusionMatrix:
True:   1       0
1:      1330    84
0:      482     2701
```

Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 87.67% +/- 0.39% (micro average: 87.67%)
ConfusionMatrix:
True:   1       0
1:      1326    81
0:      486     2704
```

Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 83.42% +/- 0.34% (micro average: 83.42%)
ConfusionMatrix:
True:   1       0
1:      1263    213
0:      549     2572
```

Performance Vector (Cech i Wekt)
Result not stored in repository.

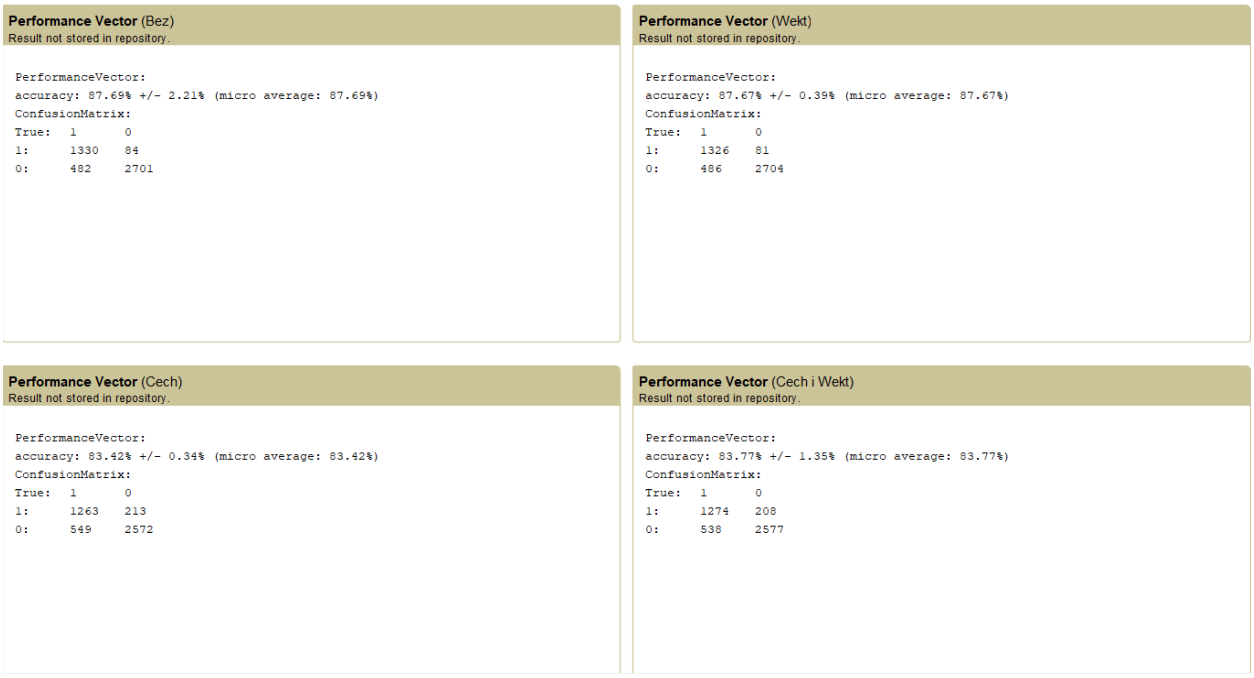
```
PerformanceVector:
accuracy: 83.77% +/- 1.35% (micro average: 83.77%)
ConfusionMatrix:
True:   1       0
1:      1274    208
0:      538     2577
```

Spostrzeżenia:

- Dla danych “coil200” najlepiej poradził sobie przyapadek bez żadnych dodatkowych algorytmów,
- Najgorzej poradził sobie algorytm cech (w kórym występują: -Select by CNN, Weight by information oraz Select by weights,
- Lepszą dokładność osiągnał połaczony algorytm cech i wektorów , niż sam algorytm cech.
- Zbiór danych dotyczący ubezpieczeń, zawiera dane klientów oraz informację, czy zgłosili roszczenie. Różne atrybuty dotyczące klientów (86 cech). Taka ilość informacji spowodowała korzystne warunki dla algorytmów selekcji cech.

Testowanie modelu na danych: “spambase”:

Wyniki:



Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 87.69% +/- 2.21% (micro average: 87.69%)
ConfusionMatrix:
True:   1       0
1:      1330    84
0:      482     2701
```

Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 87.67% +/- 0.39% (micro average: 87.67%)
ConfusionMatrix:
True:   1       0
1:      1326    81
0:      486     2704
```

Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 83.42% +/- 0.34% (micro average: 83.42%)
ConfusionMatrix:
True:   1       0
1:      1263    213
0:      549     2572
```

Performance Vector (Cech i Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 83.77% +/- 1.35% (micro average: 83.77%)
ConfusionMatrix:
True:   1       0
1:      1274    208
0:      538     2577
```

Spostrzeżenia:

- Dla zbioru danych “spambase” najlepszą dokładność uzyskał czsty model drzewadecyzyjnego oraz drzewo decyzyjne wsparte selekcją wektorów,
- Najgorzej poradził sobie algorytm z selekcją cech oraz połaczony algorytm cech i wektoró,
- Zbiór danych zawierający wiadomości e-mail sklasyfikowane jako spam lub nie-spam. Różne statystyki dotyczące treści e-maili (57 cech). Mniejsza ilość cech spowodowała gorszą dokładność dla algorytmu selekcji cech.

Testowanie modelu na danych: “yeast”

Wyniki:

Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 50.74% +/- 2.63% (micro average: 50.74%)
ConfusionMatrix:
```

True:	MIT	NUC	CYT	ME1	EXC	ME2	ME3	VAC	POX	ERL
MIT:	109	25	29	0	1	6	2	3	3	0
NUC:	2	38	22	1	0	2	3	0	0	1
CYT:	113	346	402	3	8	9	15	17	7	0
ME1:	2	0	0	27	15	8	0	1	1	0
EXC:	2	0	0	3	10	7	0	2	1	0
ME2:	3	2	2	8	0	14	2	1	0	0
ME3:	10	18	8	1	1	5	141	6	0	0
VAC:	0	0	0	1	0	0	0	0	0	0
POX:	3	0	0	0	0	0	0	0	8	0
ERL:	0	0	0	0	0	0	0	0	0	4

Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 50.20% +/- 1.33% (micro average: 50.20%)
ConfusionMatrix:
```

True:	MIT	NUC	CYT	ME1	EXC	ME2	ME3	VAC	POX	ERL
MIT:	114	27	36	2	2	6	4	3	4	0
NUC:	2	8	4	0	0	4	3	0	0	1
CYT:	106	375	413	2	7	12	22	15	7	0
ME1:	1	0	1	34	7	5	0	1	1	0
EXC:	1	0	1	3	16	5	0	3	0	0
ME2:	2	2	0	2	2	14	1	1	0	0
ME3:	14	17	7	1	1	5	133	6	0	0
VAC:	1	0	0	0	0	0	0	1	0	0
POX:	3	0	1	0	0	0	0	0	8	0
ERL:	0	0	0	0	0	0	0	0	0	4

Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 41.98% +/- 2.00% (micro average: 41.98%)
ConfusionMatrix:
```

True:	MIT	NUC	CYT	ME1	EXC	ME2	ME3	VAC	POX	ERL
MIT:	5	1	0	4	2	3	9	2	0	0
NUC:	1	8	8	2	1	2	6	1	1	1
CYT:	211	398	441	0	10	8	13	18	15	0
ME1:	6	4	5	22	8	21	0	2	1	2
EXC:	3	0	2	10	11	6	0	0	2	1
ME2:	0	0	1	5	3	6	3	0	0	1
ME3:	17	16	6	0	0	4	130	7	1	0
VAC:	0	2	0	1	0	1	2	0	0	0
POX:	1	0	0	0	0	0	0	0	0	0
ERL:	0	0	0	0	0	0	0	0	0	0

Performance Vector (Cech i Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 42.65% +/- 1.59% (micro average: 42.65%)
ConfusionMatrix:
```

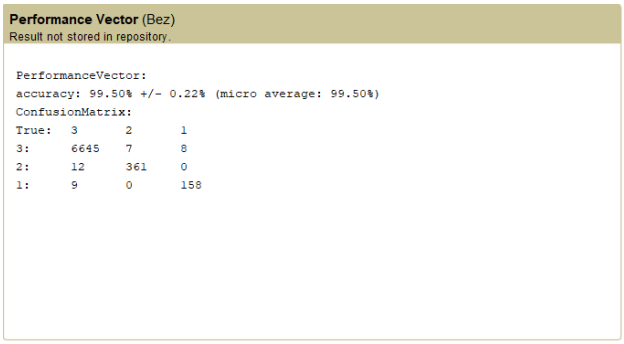
True:	MIT	NUC	CYT	ME1	EXC	ME2	ME3	VAC	POX	ERL
MIT:	6	5	0	5	2	7	9	3	1	0
NUC:	7	20	18	1	2	2	12	3	0	0
CYT:	211	384	433	1	10	8	13	15	16	0
ME1:	4	0	0	21	6	10	1	1	1	2
EXC:	4	3	8	3	13	6	0	4	1	3
ME2:	0	1	0	12	1	12	0	0	0	0
ME3:	12	16	4	1	0	6	128	4	1	0
VAC:	0	0	0	0	1	0	0	0	0	0
POX:	0	0	0	0	0	0	0	0	0	0
ERL:	0	0	0	0	0	0	0	0	0	0

Spostrzeżenia:

- Dla zbioru danych “yeast” żaden algorytm nie poradził sobie najlepiej, tak jak w poprzednich analizach najlepiej radzi sobie “czyste” drzewo decyzyjne oraz model z wykorzystaniem selekcji wektórów (Select by random),
- W dalszym ciągu wyniki dokładności dla modelu z selekcją cech oraz połączonego modelu cech i wektórów są niezadowalające.
- Zbiór danych dotyczący klasyfikacji białek drożdżowych do różnych lokalizacji komórkowych. Różne atrybuty białek (8 cech). Małą ilość cech spowodowała niską dokładność algorytmów selekcji cech oraz mieszanego algorytmu wektorów i cech.

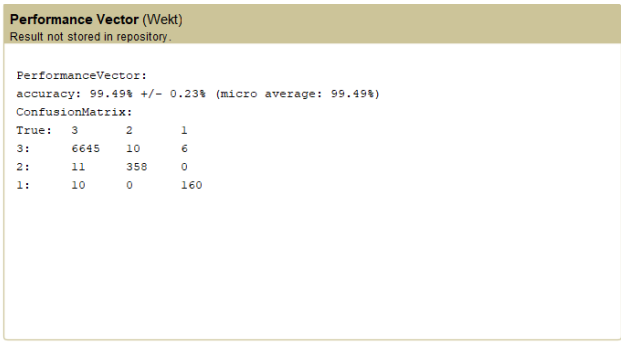
Testowanie modelu na danych: “thyroid”

Wyniki:



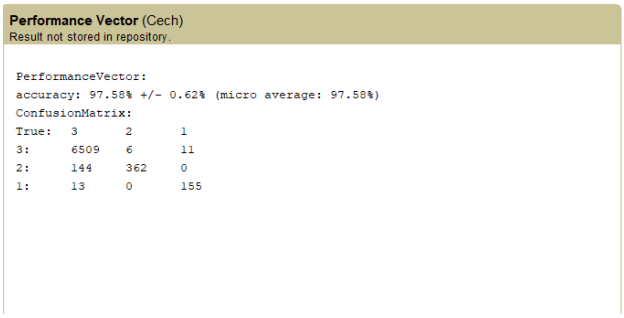
Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 99.50% +/- 0.22% (micro average: 99.50%)
ConfusionMatrix:
True:   3       2       1
3:      6645    7       8
2:      12      361     0
1:      9       0       158
```



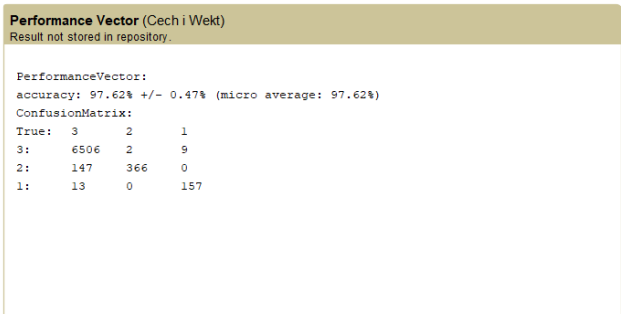
Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 99.49% +/- 0.23% (micro average: 99.49%)
ConfusionMatrix:
True:   3       2       1
3:      6645    10      6
2:      11      358     0
1:      10      0       160
```



Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 97.58% +/- 0.62% (micro average: 97.58%)
ConfusionMatrix:
True:   3       2       1
3:      6509    6       11
2:      144     362     0
1:      13      0       155
```



Performance Vector (Cech i Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 97.62% +/- 0.47% (micro average: 97.62%)
ConfusionMatrix:
True:   3       2       1
3:      6506    2       9
2:      147     366     0
1:      13      0       157
```

Spostrzeżenia:

- Dla zbioru danych “thyroid” wszystkie algorytmy osiągnęły dokładność powyżej poziomu 95%, różnica między “czystym” drzewem decyzyjnym oraz modelm z wykorzystaniem selekcji wektórów (Select by random) wyniosła 0,01 punkta procentowego,
- Zbiór danych Thyroid dotyczy diagnostyki chorób tarczycy na podstawie testów medycznych. Celem jest klasyfikacja pacjentów jako mających niedoczynność, nadczynność lub normalną czynność tarczycy, zawiera cechy takie jak wyniki różnych testów medycznych, wiek, płeć pacjenta itp. Co w tym przypadku pozytywnie wpłyneło na wyniki algorytmu selekcji cech

Testowanie modelu na danych: “satimage”:

Wyniki:

Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 59.66% +/- 1.60% (micro average: 59.66%)
ConfusionMatrix:
True:   3       4       5       7       2       1
3:      1219    148     4       42      0       24
4:      18      41      0       17      0       1
5:      0       0       415     11      11      2
7:      6       1       0       6       0       0
2:      0       3       8       0       656     4
1:      115     433     280     1432    36      1502
```

Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 63.14% +/- 8.63% (micro average: 63.14%)
ConfusionMatrix:
True:   3       4       5       7       2       1
3:      1224    159     0       46      0       28
4:      11      25      2       12      1       1
5:      0       1       410     9       12      2
7:      7       56      26      269     1       21
2:      0       2       12      0       657     3
1:      116     383     257     1172    32      1478
```

Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 76.94% +/- 10.61% (micro average: 76.94%)
ConfusionMatrix:
True:   3       4       5       7       2       1
3:      1266    137     4       36      1       28
4:      79      387     27      305     10      49
5:      0       0       367     34      60      7
7:      6       97      165     1113    40      191
2:      1       0       14      0       570     10
1:      6       5       130     20      22      1248
```

Performance Vector (Cech i Wekt)
Result not stored in repository.

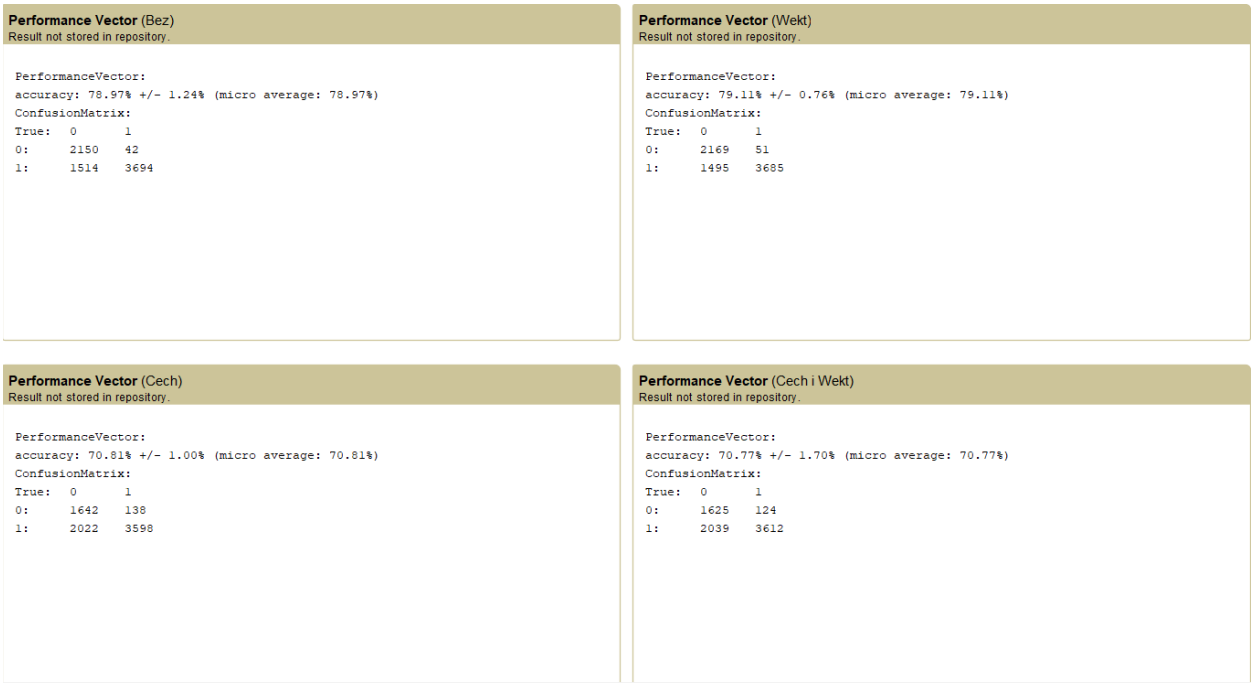
```
PerformanceVector:
accuracy: 76.22% +/- 10.81% (micro average: 76.22%)
ConfusionMatrix:
True:   3       4       5       7       2       1
3:      1263    143     4       34      1       27
4:      82      359     31      290     11      26
5:      0       0       359     34      65      6
7:      6       117     164     1119    44      220
2:      1       0       14      0       558     7
1:      6       7       135     31      24      1247
```

Spostrzeżenia:

- Zbiór danych “satimage” jest pierwszym dla którego najlepszą dokładność uzyskał algorytm z wykorzystaniem selekcji cech, nieco gorzej poradził sobie połączony,
- Zbiór danych “satimage” zawiera wiele cech (atrybutów) odnoszących się do różnych pomiarów pikseli, takich jak intensywności w różnych pasmach spektralnych, co wpływa na wyniki, dając przewage algorytmowi selekcji cech.

Testowanie modelu na danych: “ring”

Wyniki:



Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 78.97% +/- 1.24% (micro average: 78.97%)
ConfusionMatrix:
True:   0       1
0:      2150    42
1:      1514    3694
```

Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 79.11% +/- 0.76% (micro average: 79.11%)
ConfusionMatrix:
True:   0       1
0:      2169    51
1:      1495    3685
```

Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 70.81% +/- 1.00% (micro average: 70.81%)
ConfusionMatrix:
True:   0       1
0:      1642    138
1:      2022    3598
```

Performance Vector (Cech i Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 70.77% +/- 1.70% (micro average: 70.77%)
ConfusionMatrix:
True:   0       1
0:      1625    124
1:      2039    3612
```

Spostrzeżenia:

- Dla danych “coil200” najlepiej poradził sobie przyapadek bez żadnych dodatkowych algorytmów,
- Najgorzej poradził sobie algorytm cech (w kórym występują: -Select by CNN, Weight by information oraz Select by weights,
- Zbiór danych składający się z punktów w przestrzeni 2D rozmieszczonych w dwóch klasach (pierścieni). Zazwyczaj dwie cechy (współrzędne x i y).

W tym przypadku mała ilość cech nie spowodowała drastycznego pogorszenia wyników dla algorytmu selekcji cech.

Testowanie modelu na danych: “optdigits”

Wyniki:

Performance Vector (Bez)
Result not stored in repository.

```
PerformanceVector:
accuracy: 71.78% +/- 3.96% (micro average: 71.78%)
ConfusionMatrix:
True:   0     7     4     6     2     5     8     1     9     3
0:    521     1     7     2     0     4     5     1     1     0
7:      1   468    21     1     4     0     5     2    28     6
4:      6    11   440    12     2     5    10    40     8     2
6:      1     0    16   525     3     2     4     8     1     0
2:      2     2     1     9   147     4     8    16     2     3
5:      1     1    10     2    29   484     3    18     7    16
8:     17    63    30     4   339    14   460    94    27    57
1:      1     3    22     3    10     7    40   363    24     5
9:      0     3     2     0     2     6     0     2   146     3
3:      4    14    19     0    21    32    19    27   318   480
```

Performance Vector (Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 72.08% +/- 6.05% (micro average: 72.08%)
ConfusionMatrix:
True:   0     7     4     6     2     5     8     1     9     3
0:    522     2     7     2     5     5     7     0     2     3
7:      1   459    11     0     1     1     9     2    21     6
4:      9    16   421    13     3     6     3    30    10     0
6:      3     0    45   526    10     5     8    14     0     0
2:      0     3     0     6   211     4     2    43     2    14
5:      1     1     3     2     1   473     3     8    10     5
8:     13    32    21     4   231    11   364    56    22    44
1:      2     4    27     3     5     1    55   370    15     7
9:      0    20    10     0     4     8     2     9   215     3
3:      3    29    23     2    86    44   101    39   265   490
```

Performance Vector (Cech)
Result not stored in repository.

```
PerformanceVector:
accuracy: 42.79% +/- 4.41% (micro average: 42.79%)
ConfusionMatrix:
True:   0     7     4     6     2     5     8     1     9     3
0:    443    42    52    18    27     2   231    44     2     7
7:      6   201    49     0    89    21    75    73    42    37
4:     12    47   185    27    29     9    33    52     3    11
6:     31    11   108   460    60    10    55    34     0     0
2:      4    90    48    11   115    72    26    72    13    15
5:      0     5    13     1    62   250     3    58     8    47
8:     49    25    38    26    32     1    98    17     2     2
1:      8    37    12    15    35    13    17    61     4    11
9:      0    80    31     0    50   136     8   107   435   285
3:      1    28    32     0    58    44     8    53    53   157
```

Performance Vector (Cech i Wekt)
Result not stored in repository.

```
PerformanceVector:
accuracy: 44.36% +/- 3.57% (micro average: 44.36%)
ConfusionMatrix:
True:   0     7     4     6     2     5     8     1     9     3
0:    464    47    83    23    30     1   220    50     2     6
7:     12   214    52     0    72    15    60    74    46    36
4:     21    24   157    13    29     6    42    37     1     9
6:      9    15   100   453    53     7    52    36     0     0
2:      4    67    41     7   141    68    26    72    11    32
5:      0     5    11     0    54   261     3    50     0    46
8:     33    39    43    54    39     5   113    24     1     3
1:      7    22    19     8    23    11    24    41     9    12
9:      0    99    39     0    38    82     9    94   433   212
3:      4    34    23     0    78   102     5    93    59   216
```

Spostrzeżenia:

- Dla zbioru “optdigits” najlepiej poradził zobie model “czystego” drzewa decyzyjnego, natomiast najgorzej algorytm z wykorzystanie selekcji cech,
- Widać, że dla tego zbioru selekcja cech posiada mniej informacji przez co jego wydajnosc jest o wiele niższa niż przy selekcji wektorów.

Wnioski końcowe

11. **Efektywność algorytmów selekcji cech i wektorów:**
    - Algorytmy selekcji cech są bardziej skuteczne dla zbiorów danych z dużą ilością cech, jak w przypadku "coil2000" i "satimage".
    - W przypadku zbiorów danych z mniejszą ilością cech, jak "yeast" i "ring", selekcja cech nie przyniosła oczekiwanych rezultatów.
12. **Porównanie czystych modeli z modelami wspieranymi algorytmami:**
    - Czyste modele drzewa decyzyjnego często osiągały najwyższą dokładność, jak w przypadku danych "spambase", "yeast" i "optdigits".
    - Modele wspierane selekcją wektorów zazwyczaj uzyskiwały wyniki porównywalne z czystymi modelami, co sugeruje ich użyteczność w poprawie wydajności bez znacznego obniżenia dokładności.
13. **Wpływ charakterystyki zbiorów danych:**
    - Zbiory danych z dużą ilością różnorodnych cech (np. "thyroid", "coil2000", "satimage") korzystnie wpływają na działanie algorytmów selekcji cech.
    - Zbiory danych o ograniczonej ilości cech (np. "yeast", "ring") wymagają ostrożnego podejścia do selekcji cech, gdyż może to prowadzić do utraty istotnych informacji.
14. **Przyszłe kierunki badań:**
    - Warto zbadać możliwość łączenia różnych metod selekcji cech i wektorów, aby znaleźć optymalne rozwiązania dla konkretnych zbiorów danych.
    - Analiza wpływu preprocesingu danych na skuteczność algorytmów może również dostarczyć cennych informacji do dalszego doskonalenia modeli.

Rekomendacje

- Dla zbiorów danych z dużą ilością cech, warto rozważyć stosowanie algorytmów selekcji cech.
- Dla zbiorów danych o ograniczonej liczbie cech, czyste modele drzewa decyzyjnego lub modele wspierane selekcją wektorów mogą być bardziej skuteczne.
- Przy wyborze algorytmu należy uwzględniać specyfikę i charakterystykę zbioru danych, aby zoptymalizować dokładność modelu.