Uczenie maszynowe

Marcin Jarczewski
Mikolaj Szawerda

27 stycznia 2024

Spis treści

Streszczenie

Projekt polega na zaimplementowaniu inkrementacyjnej indukcji reguł. Zbiór reguł ma ulegać przebudowie na podstawie sekwencyjnie nadchodzących porcji danych lub pojedynczych przykładów.

1 Opis projektu

1.1 Indukcja reguł

Zadanie indukcji reguł polega na wyznaczeniu zbioru reguł. Pojedyczna reguła składa się z części warunkowej - kompleksu determinującego pokrywanie danego przykładu i części decyzyjnej - przyporządkującej klasę na podstawie rezultatu pokrywania. Reguł, które w części decyzyjnej mają taką samą klasę są połączone alternatywą. Cechą wyznaczonych reguł powinna być maksymalna ogólność - w pokrywanych przykładach przez kompleks powinna dominować jedna klasa.

1.2 Przebudowa zbioru reguł

Zadaniem przebudowy reguł jest dynamiczne generowanie i usuwanie już wygenerowanych reguł w zależności od sekwencyjnie pojawiających się danych. Celem przebudowy jest wyindukowanie nowej wiedzy, utwierdzenie już stworzonej wiedzy oraz adekwatna zmiana aktualnej wiedzy. Badane algorytmy powinny więc pamiętać niepokryte/źle sklasifikowane przykłady i w przypadku pojawienia się statystcznie znaczącej próbki dokonać odpowiednich akcji na zbiorze reguł.

1.3 Zasady przebudowy

Dodatkowo przebudowywanie ustalonego zbioru reguł, będzie odbywać się w określony sposób:

1. Nowy przykład jest sklasyfikowany poprawnie, wtedy nic nie robimy
2. Nowy przykład nie jest pokryty, wtedy tworzymy nową regułę, bazując na wszystkich niepokrytych przykładach.
3. Nowy przykład jest błędnie sklasyfikowany, wówczas mamy dwie możliwości:
   - jeśli są to przypadki pojedyncze, tj ich liczba nie przekracza $X\%$ wszystkich przypadków, to ignorujemy je
   - w przeciwnym przypadku, usuwamy wszystkie reguły, które błędnie zaklasyfikowały dane przypadki i przebudowujemy zbiór reguł dla nowo niepokrytych danych.

2 Opis algorytmów

Wykorzystamy dwa algorytmy implementujące podejście sekwencyjnego pokrywania, które były podane na wykładzie.

2.1 CN2

Algorytm dla zadanego zbioru trenującego generuje reguły, na zasadzie specjalizacji kompleksów. Reguła zostaje utworzona poprzez iteracyjne generowanie kompleksów z aktualnie utworzonych(poczynając od najbardziej ogólnej), konkretyzując po każdych możliwych wartościach selektorów. Po każdej iteracji wybierane jest N najlepszych kompleksów. Szeregowanie reguł odbywa się na podstawie wartości entropii $-\sum_i p_i log_2(p_i)$ i faktu czy reguła jest statystycznie znacząca - na podstawie testu χ^2

```
%D - zbior trenujacy
%S - mozliwe wartosci selektorow
cn2(D):
    RULES = []
    while BEST_CPX is null or D is empty:
        BEST_CPX = find_best_complex(D)
        if BEST_CPX is not null:
            D' = examples from D covered by BEST_CPX
            D = D \ D'
            CLASS = most common class in D'
            RULES += BEST_CPX
```

```
    ret RULES
find_best_complex(D):
    STAR = ?
    BEST_CPX = null
    while STAR is not empty:
        NEW_STAR = {(x and y) where x in STAR, y in S} - all possible specializations of
        NEW_STAR = NEW_STAR \ STAR
        for COMPLEX in NEW_STAR:
            if COMPLEX statistically significant AND COMPLEX > BEST_CPX:
                BEST_CPX = COMPLEX
        while len(NEW_STAR) > MAX_CPX_LEN:
            NEW_STAR = NEW_STAR \ wors_from(NEW_STAR)
        STAR = NEW_STAR
    return BEST_CPX
```

2.2 AQ

Algorytm opiera się na generowaniu kompleksów, które pokrywają losowo wybrany przykład pozytywny i nie pokrywają przykładów negatywnych. Reguła utworzona zostaje z kompleksu, który pokrywa najmniejszą ilość przykładów negatywnych. Ponieważ bazowanie na tylko "dodatnim ziarnie"prowadziło bardzo często do całkowitego zignorowania przykładów przeciwnych - powstawały tylko reguły mówiące o klasie 0, w implementacji ziarno jest losowane z całego dotąd nie pokrytego zbioru i w zależności od rodzaju ziarna, gwiazda powstaje pokrywając wszystkie pozytywne/negatywne i nie pokrywając ziarna. W celach pamięciowych regułu przechowują odwrócony warunek - reguła jest słownikiem, który dla każdej kolumny przechowuje zbiór dozwolonych wartości dla przykładu negatywnego i jednocześnie jest zbiorem wartości niedozwolonych na przykładu pozytywnego.

```
%P - przykłady
%N[SEED] - przykłady negatywne względem seed

aq(P, N)
    COVER = ?
    while COVER != P  || max_it% dopoki cover nie pokrywa wszystkich przykładów, lub mak
        SEED = x where x in P and x not in COVER and class(x) least common
        STAR = star(SEED, N[SEED]) % zbior kompleksow, ktore pokrywaja SEED, ale nie N
        BEST = max(STAR) % ustalana na podstawie ilości pokrytych przykładów
        COVER += BEST
    ret COVER
star(SEED, NEG)
    star = null
    while NEG*star != null
        neg = x where x in NEG and x in star
        star -> modify complex such that SEED in star and neg not in star
        star -> leave only most general complexes
        while len(star) < MAX_CPX
            star = star \ worst_from(star)
    return star
```

3 Zbiory danych

W ramach projektu, będziemy korzystać z dwóch zbiorów:

3.1 Bank Marketing

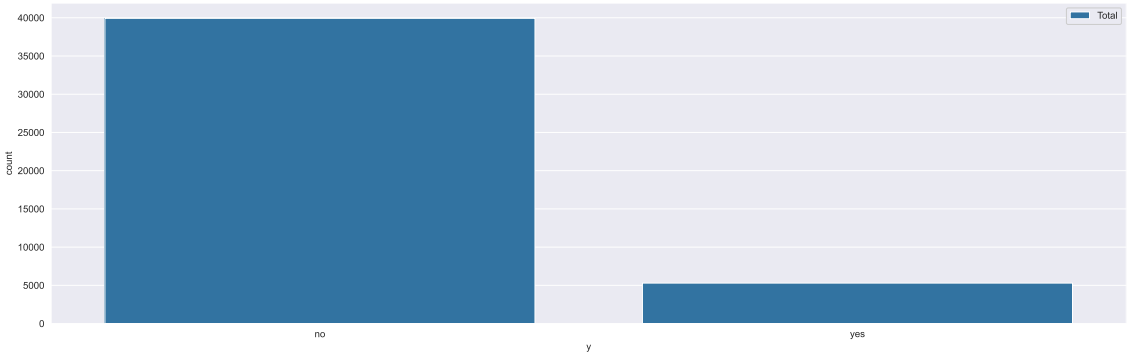
3.1.1 Opis atrybutów

Opisujący dane klientów portugalskiego banku w trakcie prowadzenia telefonicznej kampanii reklamowej, której celem było zachęcenie klientów do skorzystania z lokaty. Zadaniem klasyfikacji w zbiorze jest przewidzenie na podstawie cech, czy klient weźmie lokatę. Zbiór składa się z cech:

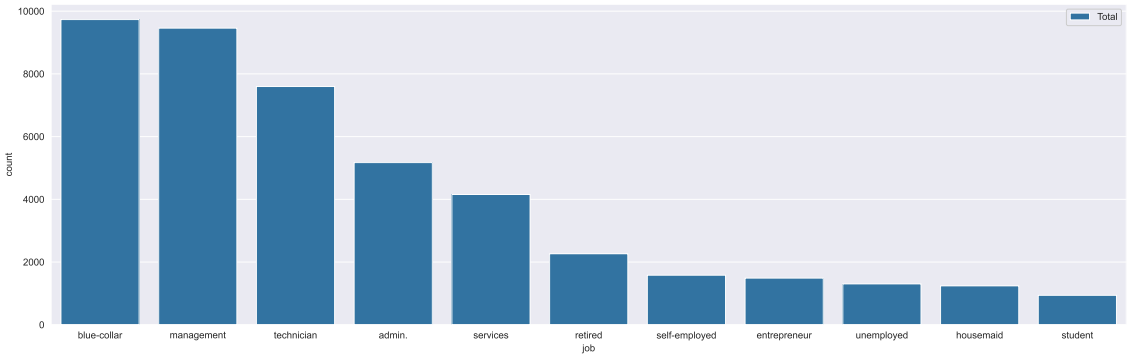
- age (liczba)
- job : rodzaj zatrudnienia: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student" "blue-collar","self-employed","retired","technician","services")
- marital : stan cywilny (kategorie: "married","divorced","single"; note: "divorced"means divorced or widowed)
- education (kategorie: "unknown","secondary","primary","tertiary")
- default: czy posiada kredyt? (binanry: "yes","no")
- balance: średni roczny stan konta, w euro (liczba)
- housing: czy posiada nieruchomość? (binarny: "yes","no")
- loan: czy ma własną lokatę? (binarny: "yes","no")
- contact: rodzaj kontaktu (kategorie: "unknown","telephone","cellular")
- day: dzień miesiąca, ostatniego kontaktu (liczba)
- month: miesiąc, ostatniego kontaktu (kategorie: "jan", "feb", "mar", ..., "nov", "dec")
- duration: długość trwania rozmowy, w sekundach (liczba)
- campaign: liczba kontaktów przeprowadzonych w ramach ostatniej kampani reklamowej (liczba, zawiera ostatni kontakt)
- pdays: liczba dni, które minęła od poprzedniej kampani (liczba, -1 oznacza brak wcześniejszego kontaktu)
- previous: liczba kontaktów przeprowadzonych przed tą kampanią dla danego klienta
- poutcome: wynik kampanii (kategorie: "unknown","other","failure","success")

3.1.2 Analiza danych

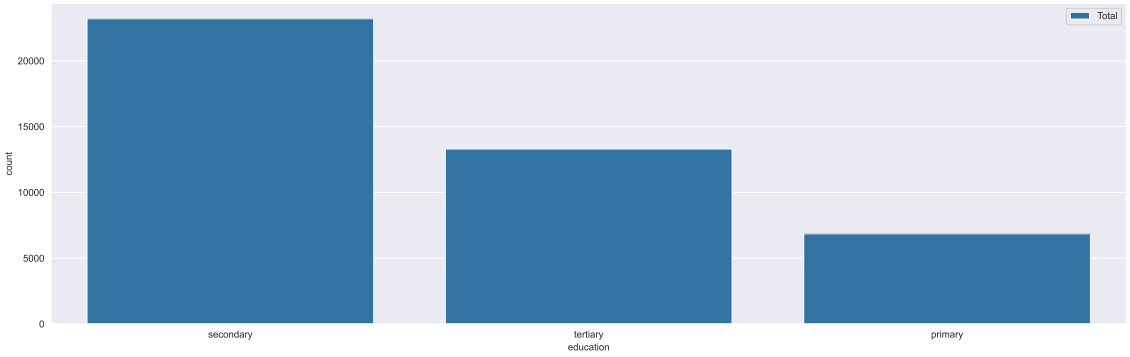
Atrybut	Liczba brakujących wartości
age	0
job	288
marital	0
education	1857
default	0
balance	0
housing	0
loan	0
contact	13020
day	0
month	0
duration	0
campaign	0
pdays	0
previous	0
poutcome	36959
y	0



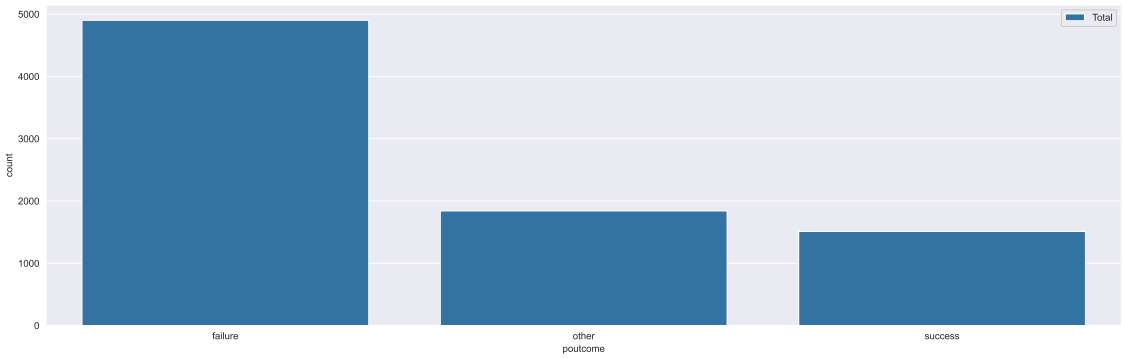
Rysunek 1: Rozkład klasy



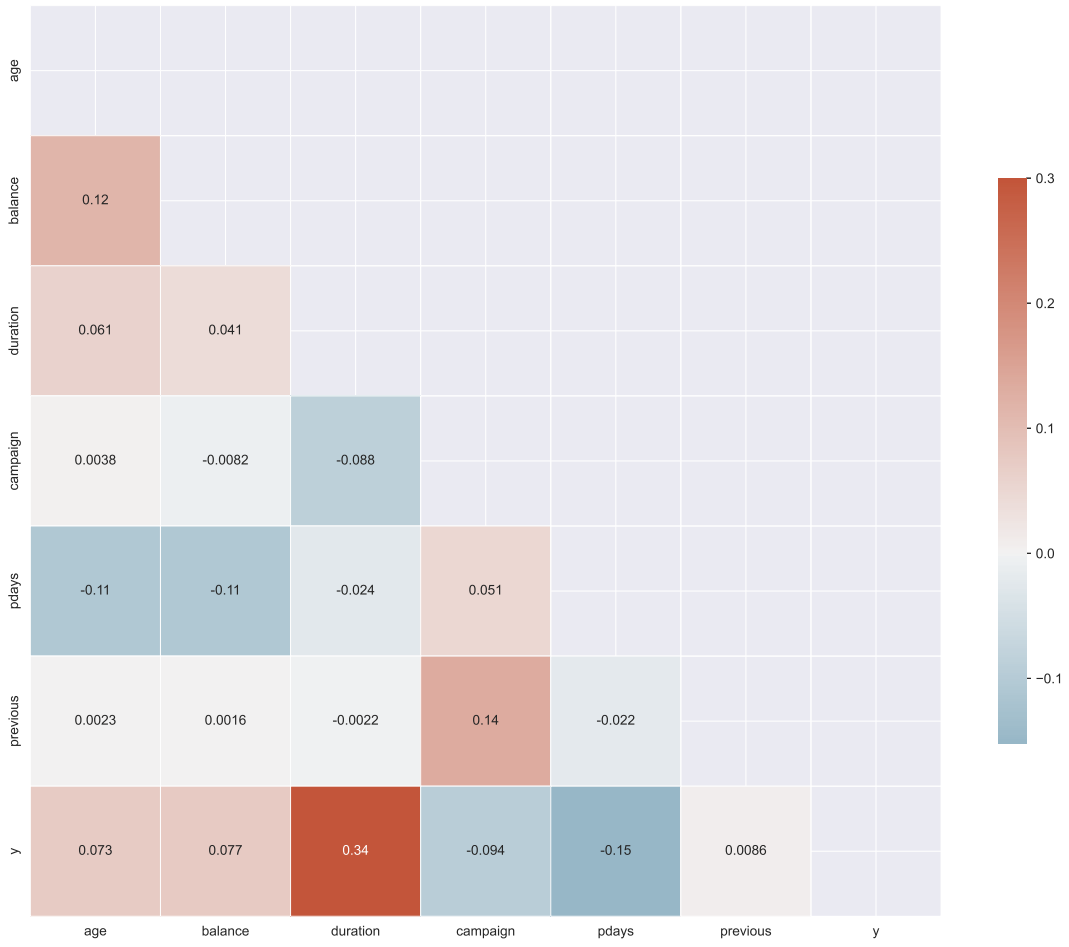
Rysunek 2: Rozkład wartości atrybutu *job*



Rysunek 3: Rozkład wartości atrybutu *education*



Rysunek 4: Rozkład wartości atrybutu *poutcome*



Rysunek 5: Korelacje między atrybutami ciągłymi

3.2 Adult

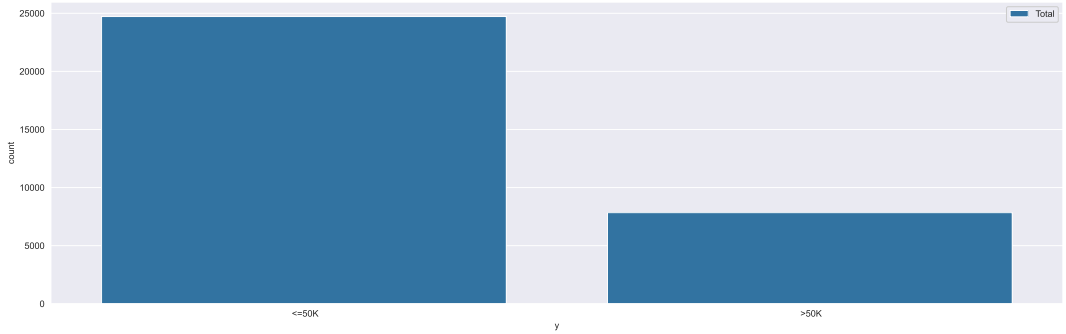
3.2.1 Opis atrybutów

Zbiór danych zawierający zestaw cech osób dorosłych w celu przewidzenia czy dana osoba zarabia więcej czy mniej niż 50 tys. dolarów rocznie. W skład cech wchodzą:

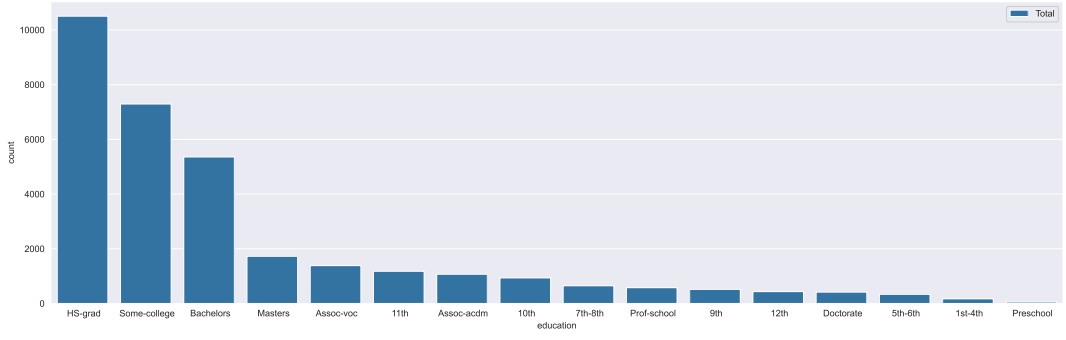
- age: atrybut ciągły.
- workclass: stan zatrudnienia - kategorie: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt: atrybut ciągły - wartość wyznaczona na podstawie charakterystyki demograficznej(osoby z podobną charakterystyką posiadają zbliżone wartości tej cechy)
- education: wykształcenie - kategorie: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- education-num: atrybut ciągły - liczba lat edukacji
- marital-status: stan cywilny - kategorie: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: zawód - kategorie: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: relacja - kategorie: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: rasa - kategorie: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: płeć - kategorie: Female, Male.
- capital-gain: atrybut ciągły.
- capital-loss: atrybut ciągły.
- hours-per-week: atrybut ciągły. - tygodniowa liczba godzin pracy
- native-country: kraj pochodzenia - kategorie: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, TrinadadTobago, Peru, Hong, Holand-Netherlands.

3.2.2 Analiza danych

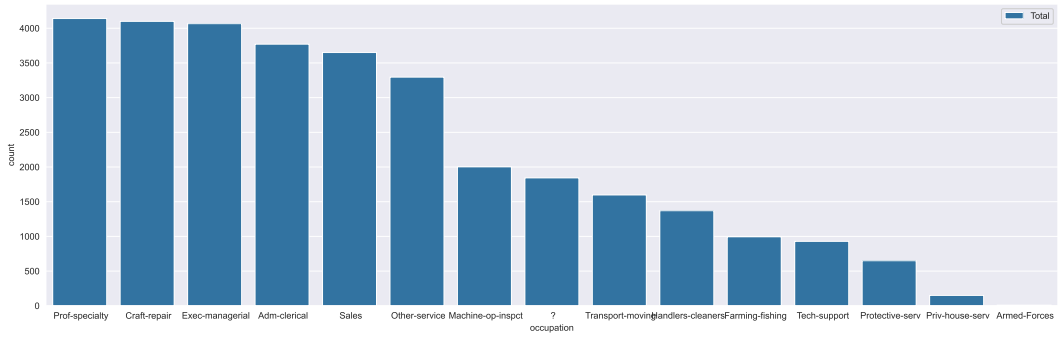
Atrybut	Liczba brakujących wartości
age	0
workclass	1836
fnlwgt	0
education	0
education-num	0
marital-status	0
occupation	1843
relationship	0
race	0
sex	0
capital-gain	0
capital-loss	0
hours-per-week	0
native-country	583
y	0



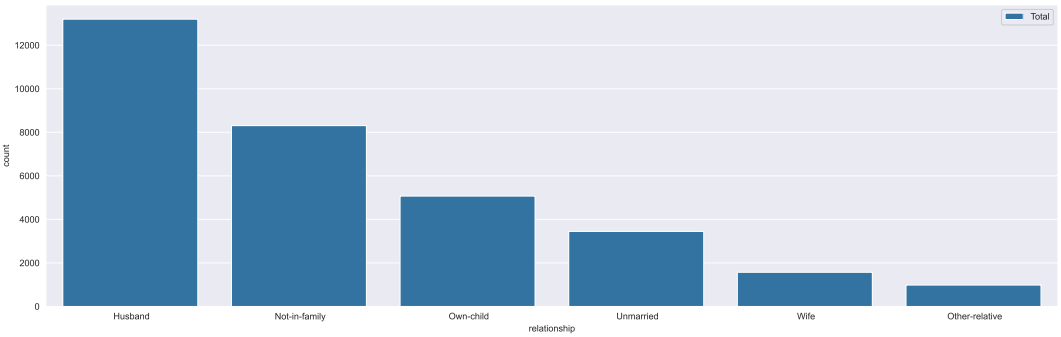
Rysunek 6: Rozkład klasy



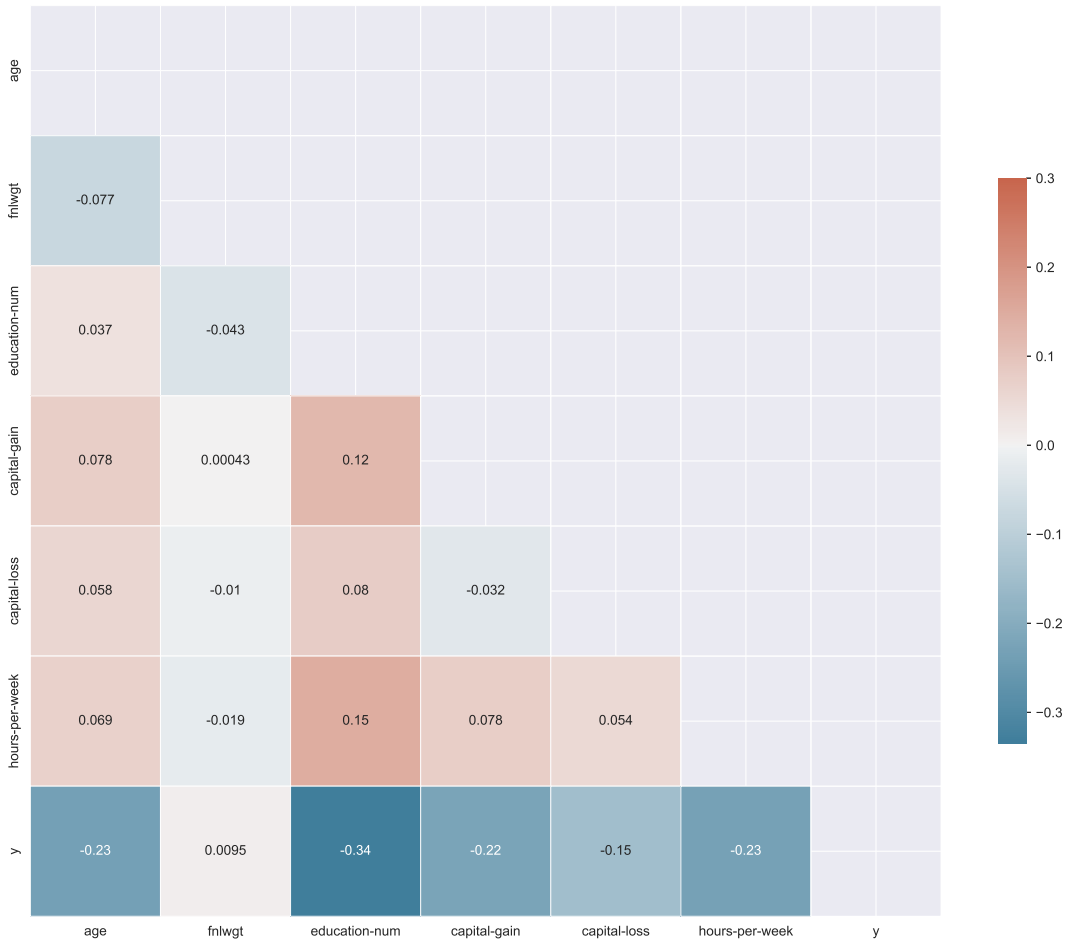
Rysunek 7: Rozkład wartości atrybutu *education*



Rysunek 8: Rozkład wartości atrybutu *occupation*



Rysunek 9: Rozkład wartości atrybutu *relationship*



Rysunek 10: Korelacje między atrybutami ciągłymi

4 Plan eksperymentów

Zgodnie z celem projektu, zadaniem eksperymentów będzie zbadanie poprawności implementacji dynamicznej przebudowy reguł, a także porównanie dwóch algorytmów w środowisku dwóch różnych zbiorów danych zadania klasyfikacji binarnej oraz wybranych sposobów rozstrzygania ostatecznej odpowiedzi modelu z ustalonego zbioru reguł.

W celu porównania algorytmów i możliwych do zmiany części algorytmu podzielimy nasz zbiór na

zbiory: uczący i testowy. W ramach zbioru testowego chcemy zaobserwować poprawność klasyfikacji przykładów na podstawie ogólnie znanych metryk. W celu zapewnienia zaobserwowania przebudowy reguł wykonamy osobny eksperyment, w którym modelelowi będą podawane sekwencyjnie dane loswej wielkości z góry ustalonym rozkładem - który będzie miał intuicyjną interpretację zmiennej przewidywanej. Następnie po wyczerpaniu przez model pierwszych porcji danych, podane zostaną kolejne porcje o przeciwstawnej interpretacji w ilości teoretycznie zmuszającej model do całkowitej przebudowy reguł.

5 Opis rozwiązania

Do przeprowadzenia algorytmów skorzystaliśmy z algorytmu CN2 - z biblioteki Orange Data Mining library. Dla wygody zaimplementowaliśmy klasę `CN2_Runner`, która stanowi abstrakcję nad użyciem algorytmu i ułatwia testowanie przebudowy reguł.

5.1 AQ

Algorytm AQ został zaimplementowany przez nas samodzielnie z użyciem bibliotek pandas, oraz numpy. Implementacja algorytmu przyjeła następujące założenia

1. Reguła jest słownikiem zbiorów, gdzie zbiór przechowuje informacje jakie elementy może mieć przykład, by zostać zaklasyfikowany - w trakcie określania gwiazdy, słownik służy do klasyfikacji przykładów negatywnych i odrzucenia ziarna(dzięki temu nie ma konieczności przechowywania różnicy wszystkich wartości z różnicą seeda z negatywem)
2. Predykcja odbywa się na zasadzie głosowania - powyżej 0.5 reguł musi twierdzić o 1 klasie, by przykład dostał tę klasę
3. Wybór gwiazd częściowych odbywa się na podstawie ilości pokrywanych przykładów negatywnych
4. Wybór reguły do pokrycia odbywa się na podstawie ilości prawidłowo przydzielanych klas przez reguły
5. Reguła ma predykat tylko równościowy
6. Ziarno jest całkowicie losowane - może być reprezentantem klasy 0 albo 1
7. Negatyw do tworzenia częściowych gwiazd również jest losowany z niepokrytego dotąd zbioru negatywów
8. algorytm posiada hiperparametry:
    - "star_it ilość iteracji jakie algorytm maksymalnie poświęci na znalezienie gwiazdy pokrywającej wszystkie negatywy
    - 'it' - ilość iteracji jakie algorytm maksymalnie poświęci na znalezienie zbioru reguł
    - 'max_cpx' - maksymalna ilość rozważanych częściowych gwiazd w trakcie poszukiwania gwiazdy
    - 'max_rules' - maksymalna ilość przechowywanych reguł
    - 'no_diff_it' - w przypadku poszukiwania gwiazdy/reguł algorytm mógłby potencjalnie wykonywać się w nieskończonej pętli, jest to parametr, który określa przez jaką ilość iteracji wynik może nie ulegać zmiania, by nie przerwać pętli(wynika to z losowania ziarna/przykładu negatywnego)

W celu sprawdzenia i porównania działania algorytmów użyliśmy `src/cn2_experiments.py` oraz `notebooks/07-AQ-experiments.ipynb`. Natomiast do przeprowadzenia eksperymentów związanych z przebudową reguł używamy kodu w pliku `src/compare.py`

6 Eksperymenty

6.1 Działanie algorytmów

Do testowania zostało użyte pierwsze 10000 wierszy w obu zbiorach i zostały one podzielone przy pomocy funkcji `train_test_split` z pakietu `sklearn`. Rozmiary zbiorów testowych stanowiły 20%.

Metric	**Adult**	**Bank**
Train Time	62.76158381	13.88964128
Test Time	0.04255009	0.02983761
Accuracy	0.8125	0.953
Recall	0.87732095	0.25974026
F1 Score	0.87586892	0.29850746

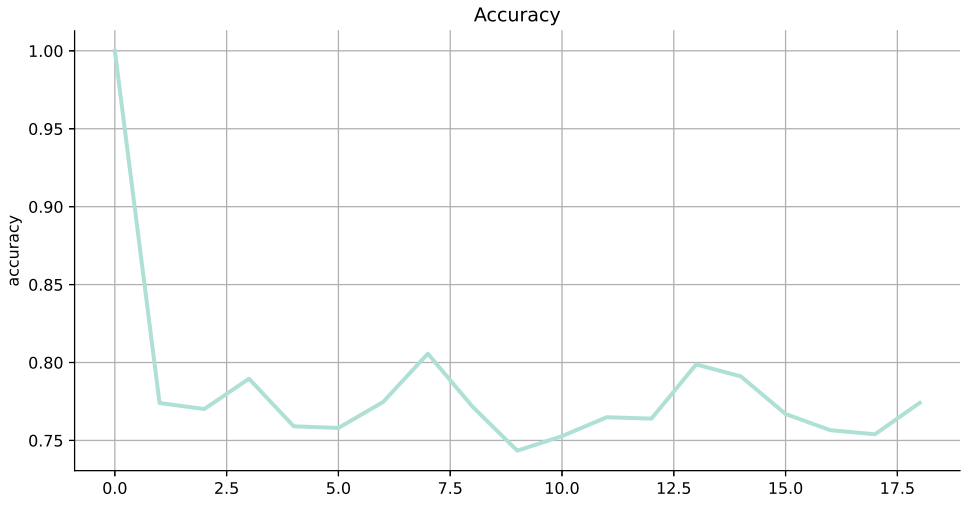
Tabela 1: Wyniki uruchomienia na zbiorach *Adult* i *Bank* algorytmu *CN2*

6.2 Przebudowa reguł

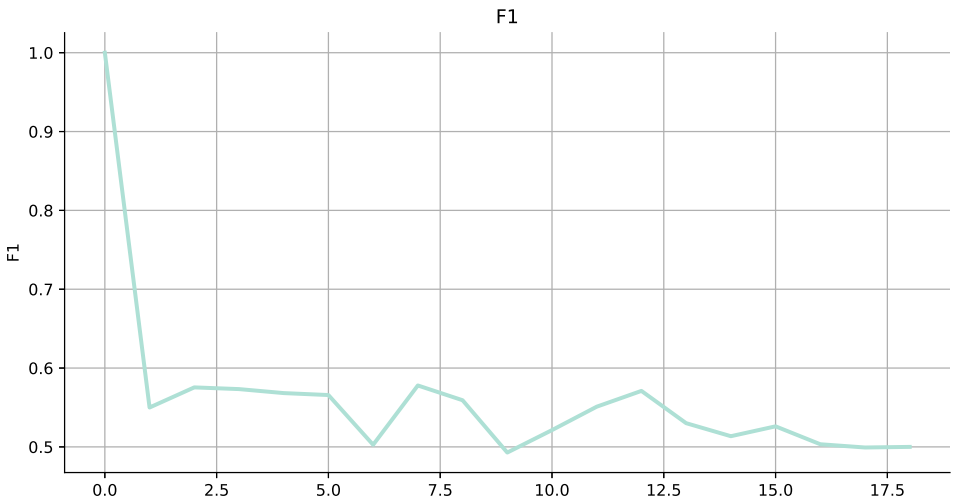
Aby zaobserwować przebudowę reguł zastosowaliśmy gotową implementację algorytmu `KMeans` z pakietu `sklearn`. W tym celu w notatniku `notebooks/05-groupping-data.ipynb` grupujemy dane w 20 grup a następnie sortujemy je po ilorazie klasy pozytywnej i wszystkich przykładów w danej grupie. Intuicja polega na tym, że kolejne grupy będą posiadały przykłady, których ciężej się nauczyć co świadczy o większym zbalansowaniu zbioru.

Na zbiorze *Adult*, widać dłuższe działanie algorytmu oraz niższą dokładność w porównaniu do zbioru *Bank*. Wpływ na to może mieć struktura danych, i trudniejsze dane do nauczenia. Porównanie zarobków a udzielenie pożyczki, które jest dużo prostsze do przewidzenia.

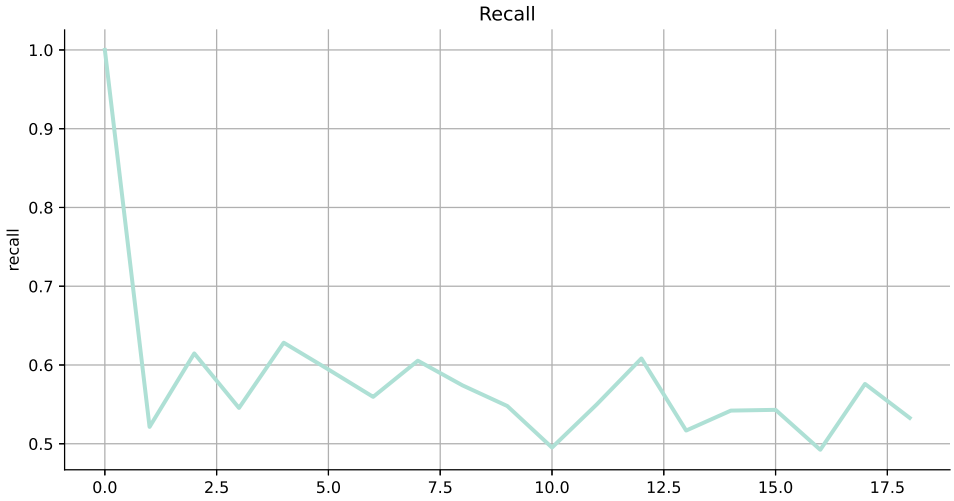
6.2.1 Przebudowa na zbiorze *Adult*



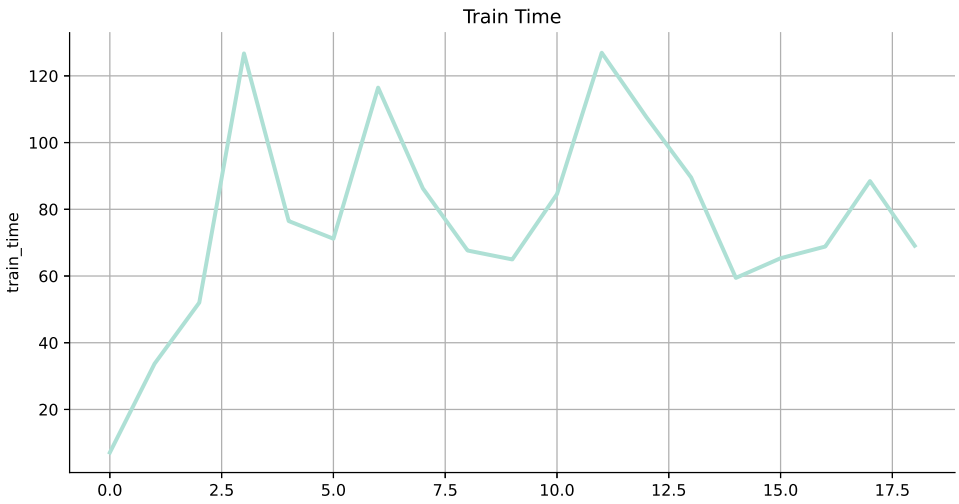
Rysunek 11: Dokładność modelu



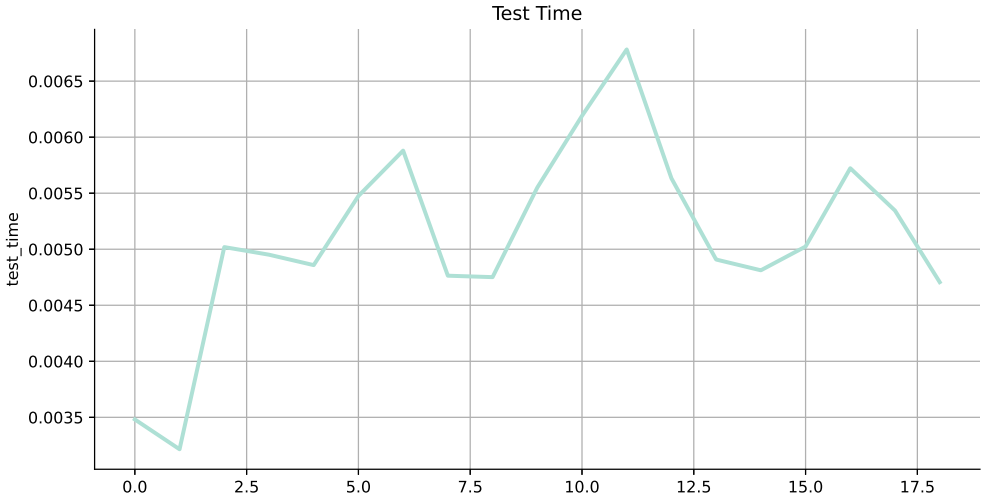
Rysunek 12: Wartość metryki *F1*



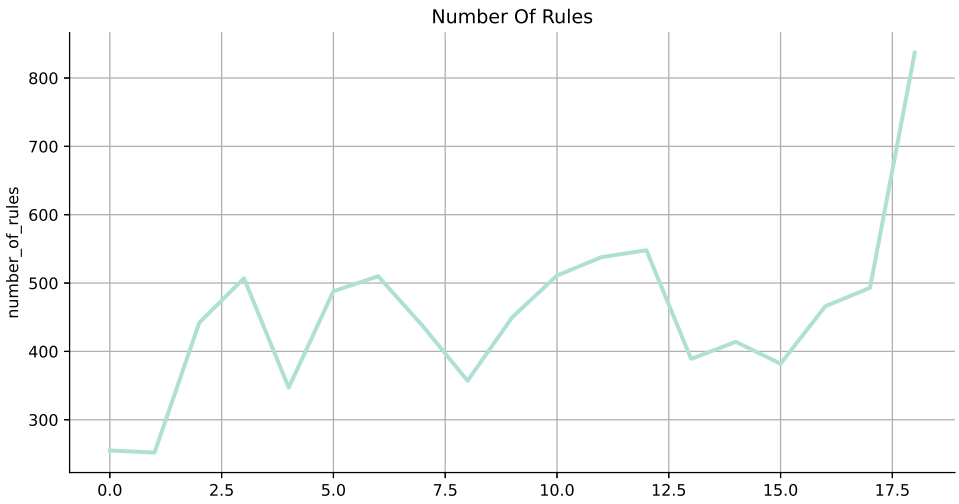
Rysunek 13: Wartość metryki *recall*



Rysunek 14: Czas uczenia [s]

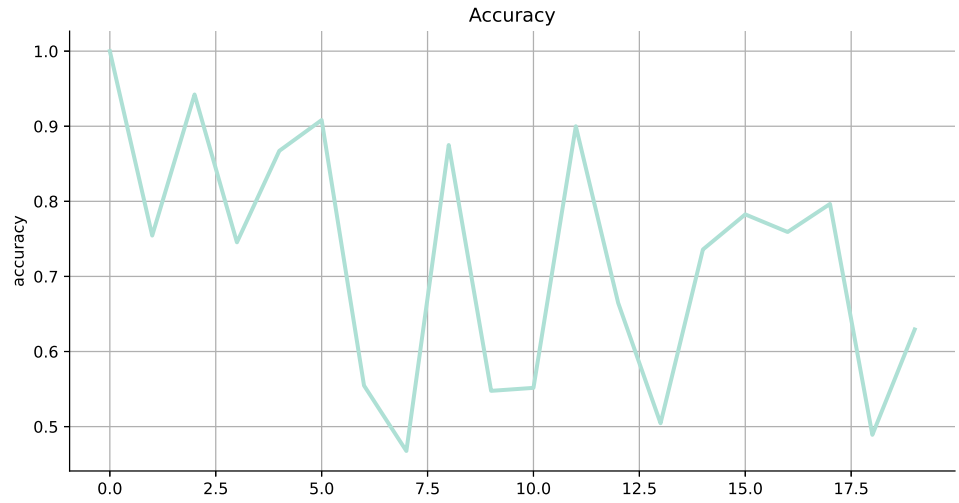


Rysunek 15: Czas testowania [s]

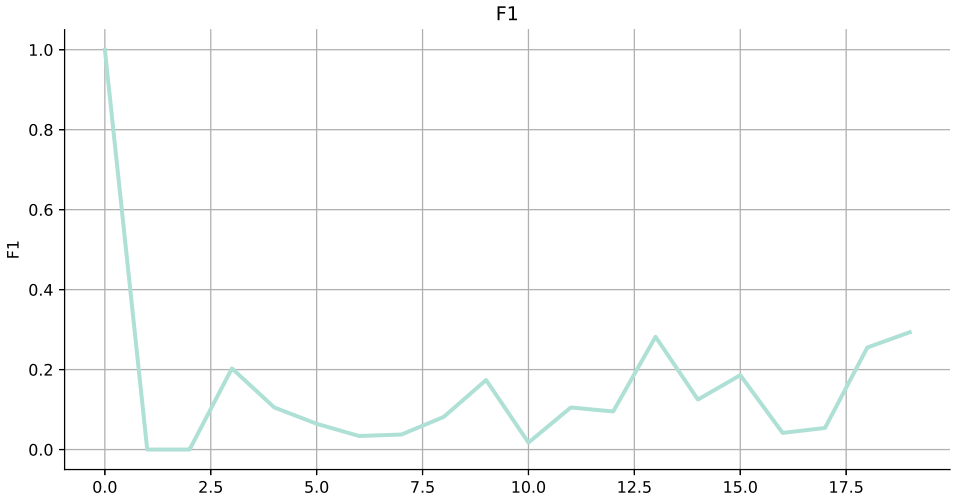


Rysunek 16: Liczba reguł

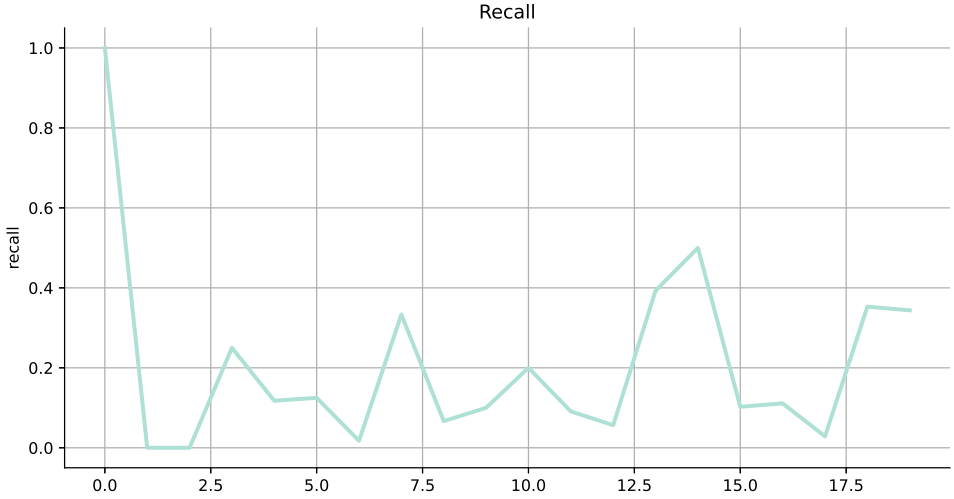
6.2.2 Przebudowa reguł na zbiorze *Bank*



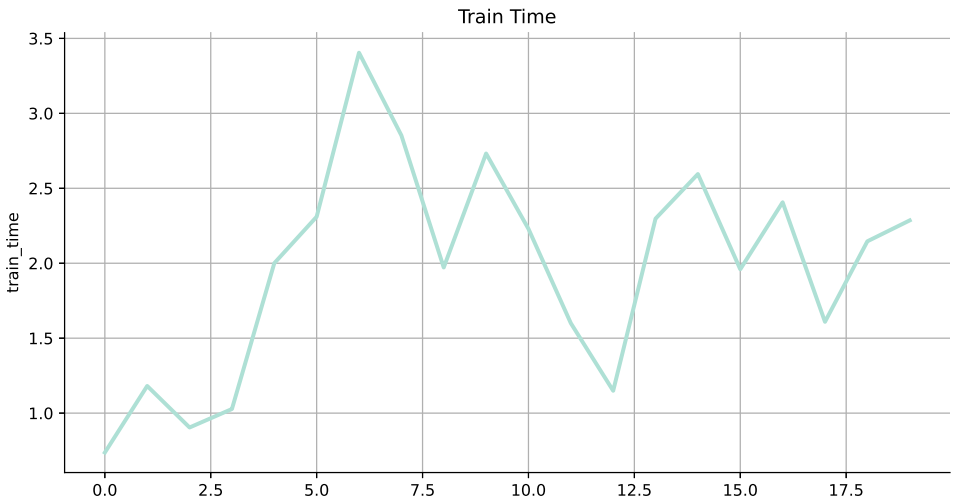
Rysunek 17: Dokładność modelu



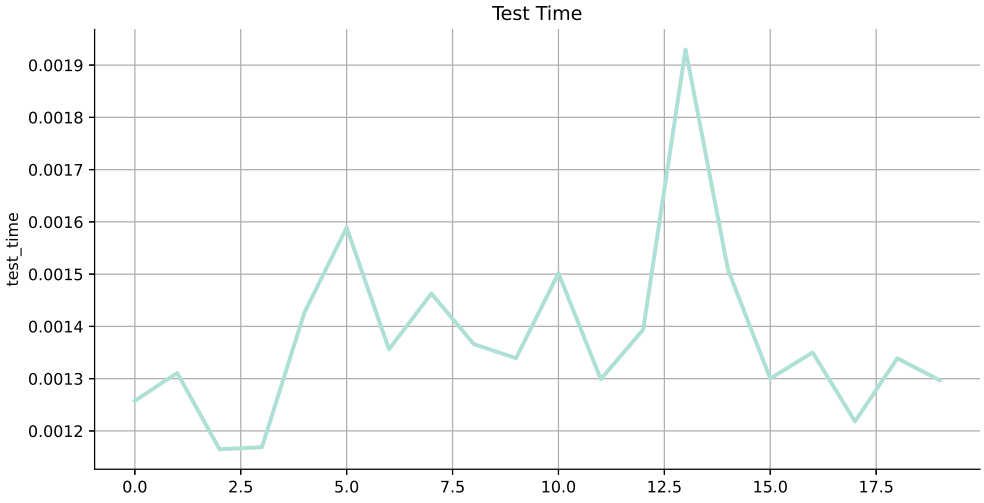
Rysunek 18: Wartość metryki *F1*



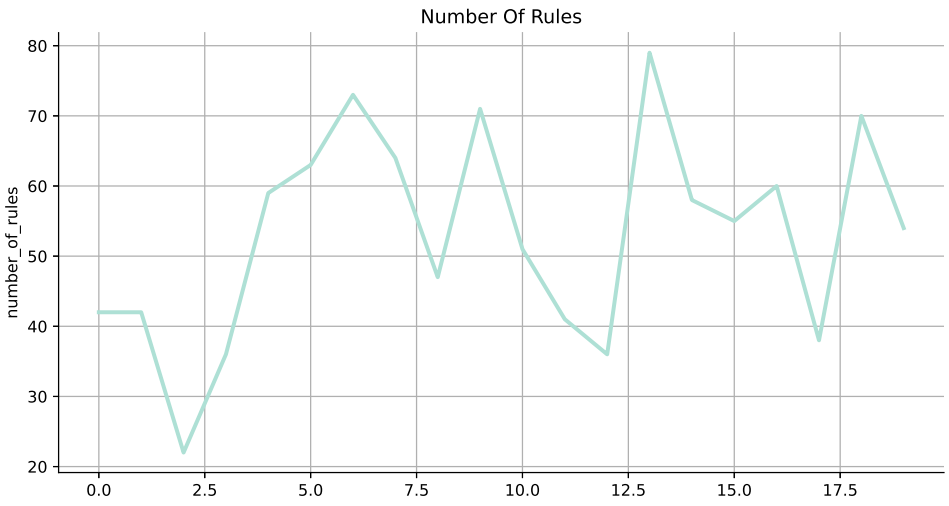
Rysunek 19: Wartość metryki *recall*



Rysunek 20: Czas uczenia [s]



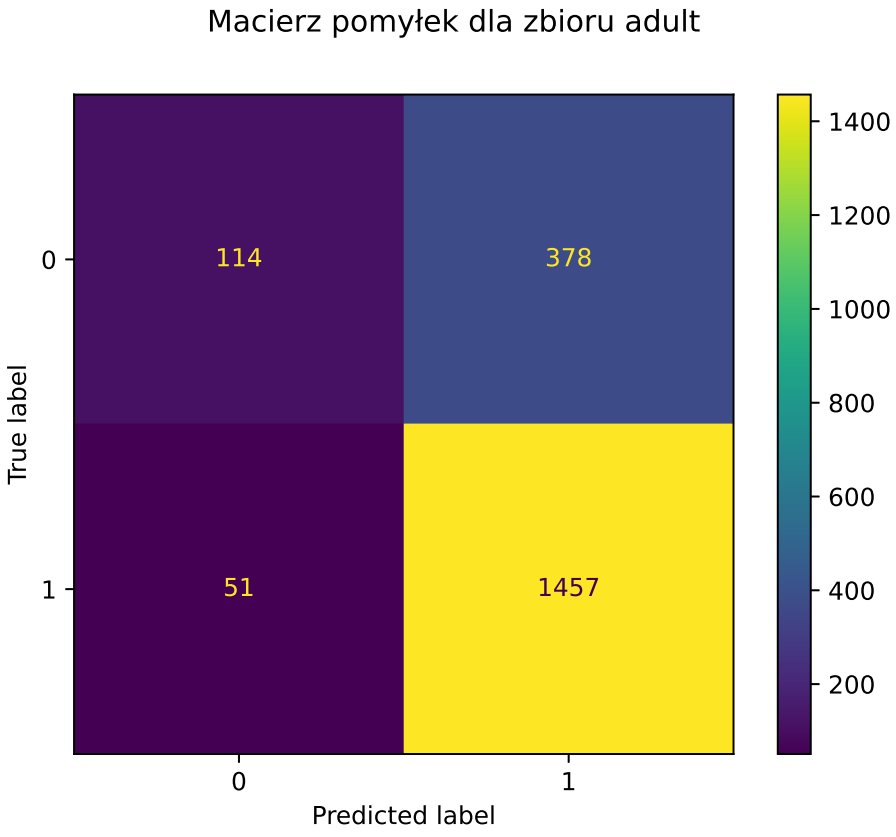
Rysunek 21: Czas testowania [s]



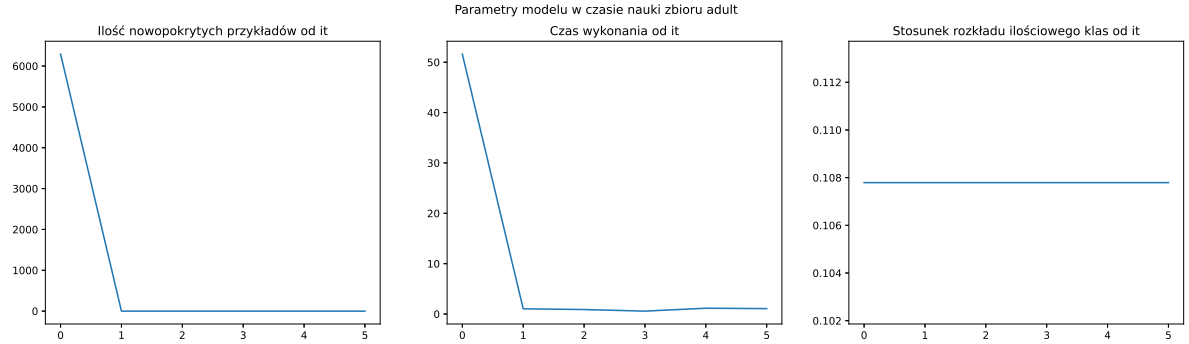
Rysunek 22: Liczba reguł

6.3 AQ

6.3.1 Zbiór adult



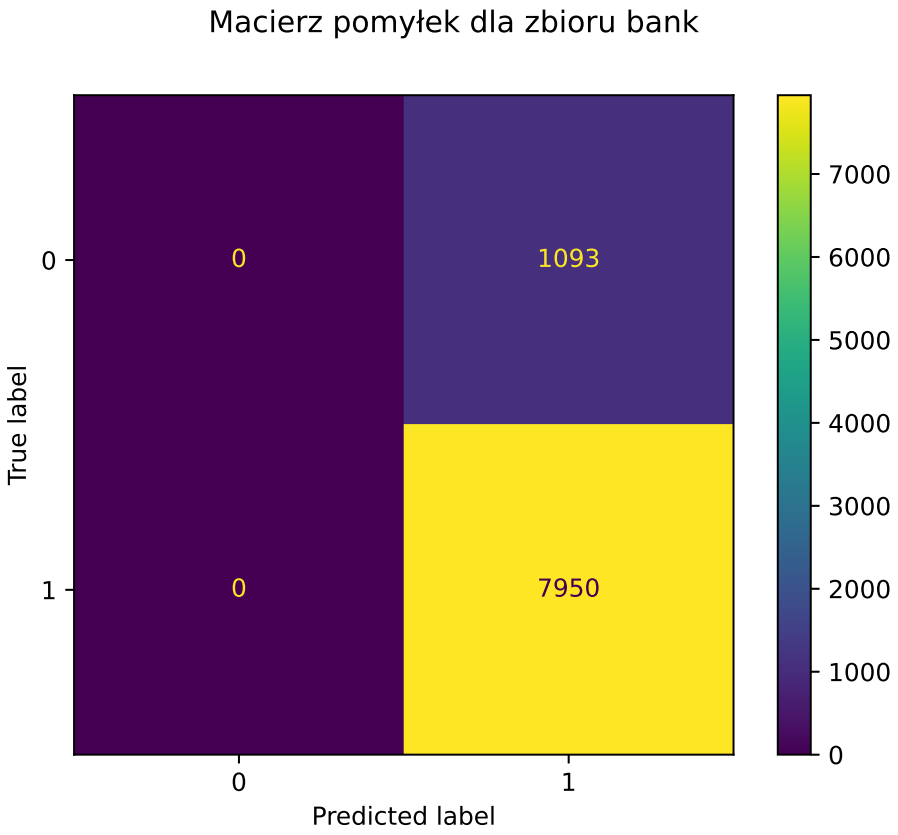
Rysunek 23: Adult macierz pomyłek



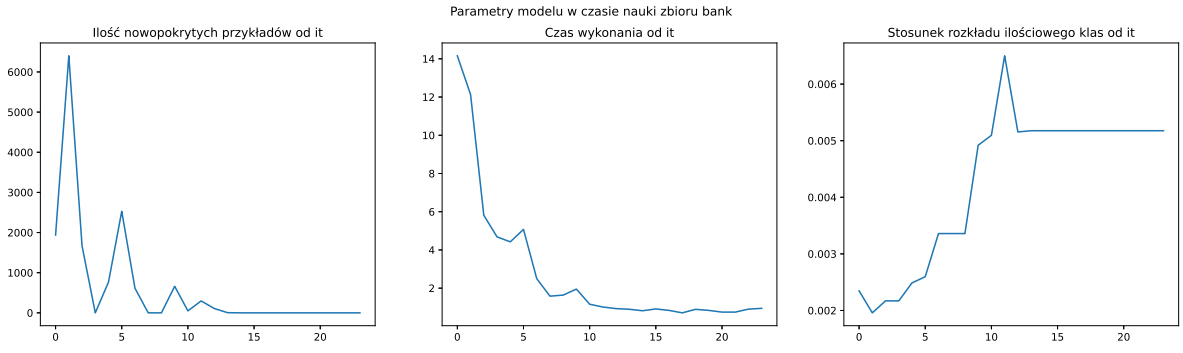
Rysunek 24: Adult wykresy w trakcie nauki

Własna implementacja algorytmu nie osiąga aż tak dobrych rezultatów jak gotowa implementacja CN2. Dla zbioru Adult algorytm średnio osiągał 0.8 skuteczności. Należy przedewszystkim zwrócić uwagę na ilość FN do TN - problem ten wynika głównie z niezbalalansowania danych uczących, algorytm ten przez swoją metodę działania jest bardzo podatny na różnego rodzaju szumy, nieprawidłowości w danych. Dość zaskakującym rezultatem działania algorytmu jest wykres nowo przybyłych pokrytych osobników. W trakcie eksperymentów przy wielokrotnych uruchomionach(z włączonym losowaniem ziarna ustalania gwiazd) algorytm osiągał najlepsze rezultaty, gdy na początku trafił na "reprezentatywne ziarno", które pozwoliło mu wyeliminować sporą część przykładów - jednakże potem algorytm nie był w stanie wystarczająco uszczegółowić reguł. Można było zauważyć zależność ilości rozważanych kompleksów do finalnej ilości reguł, oraz ilości iteracji potrzebnej na znalezienie jak najlepszej gwiazdy. Ilość rozważanych kompleksów dawała algorytmowi potencjał na znalezienie najlepszej kombinacji, ale jednocześnie znacząco wydłużało to proces uczenia i bardzo często otrzymane reguły były podobne do siebie. Ilość finalnych reguł pozwalała teorycznie algorytmowi utrzymać jak największą wiedzę, lecz alternatywnie zwiększało to szum i sprawiało, że algorytm podejmował decyzje bardziej jak maszyna losowa.

6.3.2 Zbiór bank



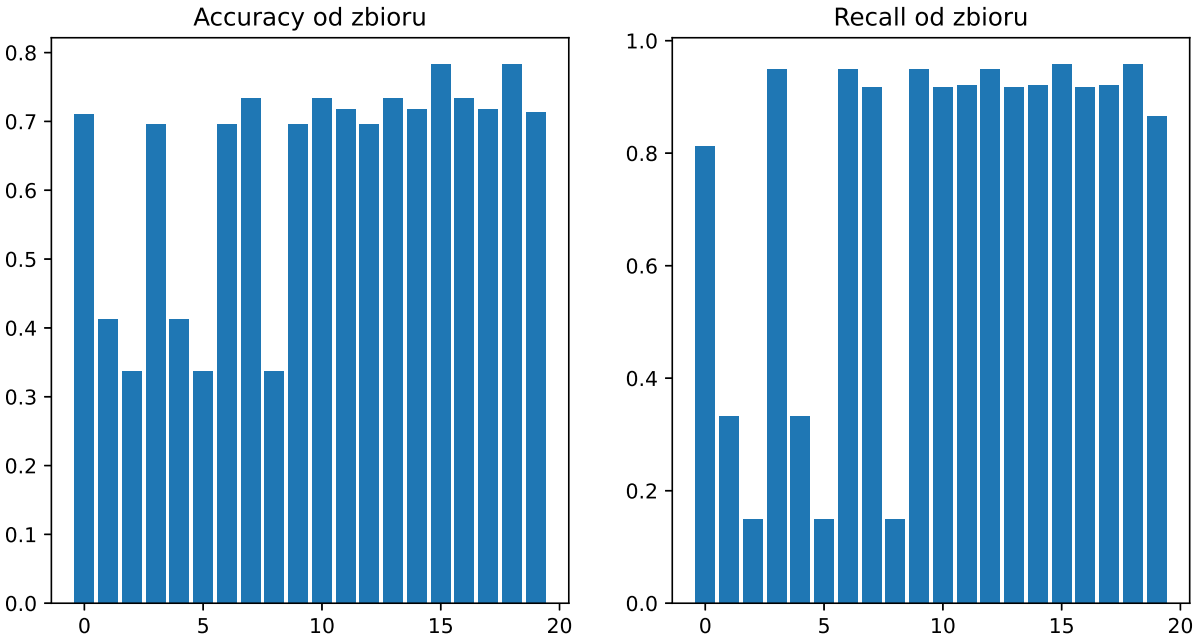
Rysunek 25:



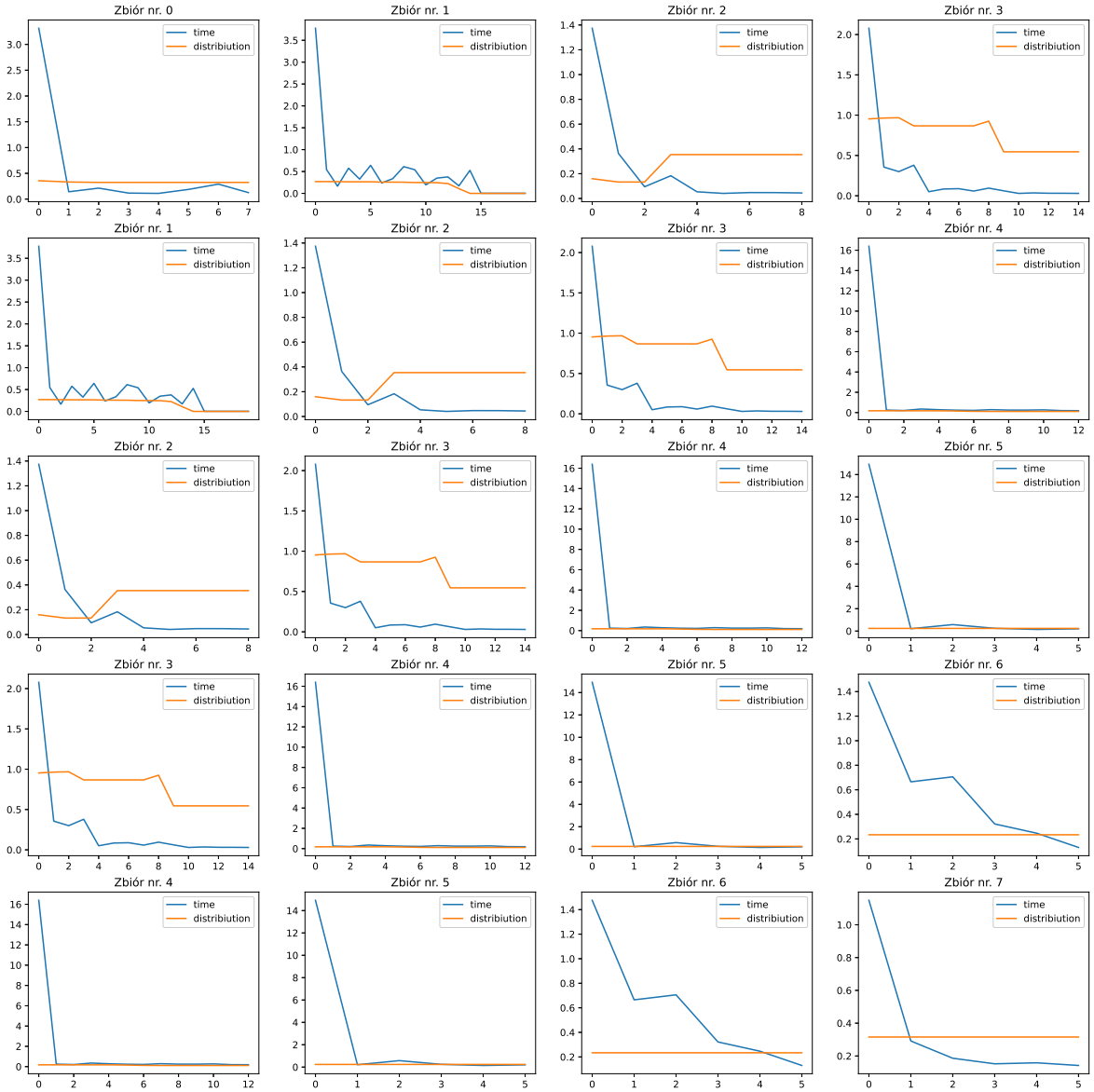
Rysunek 26: Liczba reguł

Algorytm totalnie nie był w stanie poradzić sobie z zbiorem bank. Najlepsze rezulaty jakie zostały osiągnięte to 0.58 skuteczności, co jest nieakceptowalną wartością. Zbiór ten posiadał sporo danych liczbowych, do których znacząco bardziej nadają się predykaty nierównościowe. Należy zwrócić uwagę, iż algorytm podobnie jak algorytm CN2 znacząco szybciej kończył wykonanie w przypadku zbioru danych Bank.

6.3.3 Przebudowa reguł



Rysunek 27: Wartości klasyfikacji



Rysunek 28: Wykresy nauki dla poszczególnych zbiorów

Algorytm AQ identycznie jak opisany wcześniej CN2 otrzymywał porcjami paczki danych, otrzymane z klasteryzacji przykładów przez algorytm K-mean. Zgodnie z oczekiwaniami algorytm przy początkowych znacząco różniących się od siebie paczkach osiągał gorsze rezultaty, by w raz z czasem wrócić do akceptowalnej skuteczności. Z wyżej przedstawionych wykresów można więc wysnuć wniosek, iż algorytm AQ potrzebuje dużej ilośći danych, aby skutecznie przebudować zbiór reguł. W zależności od zastosowań, mógłby to być hiperparametr algorytmu.

Z wykresów z poczególnych etapów należy zauważyć, iż gdy algorytm ma problem z znalezieniem

ogólnej wiedzy potrzebuje większej ilości iteracji, a ilości kolejno pokrywanych przykładów są mniejsze - algorytm buduje bardziej specjalizowane reguły.

7 Wnioski

Na większych zbiorach danych, algorytmy nie mają problemu aby uzyskać wysoką $> 80\%$ dokładność. Natomiast przy przebudowie reguł widać spore wachania dokładności na zbiorze *Bank*. Natomiast na zbiorze *Adult* te wachania są dużo mniejsze, algorytm jest w stanie lepiej przystosować i w tym celu stosuje ponad 10 razy więcej reguł niż na poprzednim zbiorze. Metody te nadają się do klasyfikacji binarnej jednak przy dynamicznej przebudowie reguł duży wpływ ma sama struktura danych.

Zaproponowana implementacja AQ w porównaniu z CN2 szybciej kończła proces nauki - poprzez proste reguły równościowe szybko pokrywała kolejne przykłady - co stawało się mankamentem w przypadku zbiorów silnie numerycznych, gdzie warunki różniościowe pozwoliłby osiągnąc zdecydowanie lepsze rezulaty. W czasie eksperymentów z metodą zwróciłem uwagę na bardzą silną zależność hiperparametrów. Znaczącą ograniczając ilość reguł metoda przy pierwszym przykładzie, który pozwalał jej usunać znaczną część przykładów kończyła pracę, natomiast gdy reguł było za dużo tworzył się szum i skuteczność algorytmu zbiegała do 50 procent.