SZKOŁA GŁÓWNA HANDLOWA W WARSZAWIE
WARSAW SCHOOL OF ECONOMICS

Studium licencjackie

Kierunek: Metody Ilościowe w Ekonomii i Systemy Informacyjne

Specjalność: Metody Analizy Decyzji

Forma studiów: stacjonarne

Imie i nazwisko: Michał Cisek

Nr albumu: 56283

Wyznaczenie reguł asocjacyjnych z finansowych szeregów czasowych na przykładzie indeksu giełdowego WIG20

Praca licencjacka napisana
w Kolegium Analiz Ekonomicznych
w Katedrze Matematyki i Ekonomii Matematycznej
pod kierunkiem naukowym
dr hab. Michała Ramszy

Warszawa 2015

Spis treści

1 Wprowdzenie

Analiza techniczna koncentruje się na badaniu zachowań rynku, głównie poprzez analizę wykresów oraz wolumenu zawartych transakcji. Propagatorzy tej teorii powołują się na 3 reguły (Murphy (1999)). Pierwsza z nich zakłada, że rynek dyskontuje wszystko. Oznacza to, że każda informacja, zarówno ekonomiczna, polityczna czy gospodarcza, mogąca wpłynąć na cenę danej akcji czy innego instrumentu finansowego, ma w nim pełne odzwierciedlenie. Druga teza stanowi, iż ceny podlegają trendom, tzn. wykazaują tendencję do poruszania się w jednym kierunku. W analizie technicznej wszystkie ruchy cen, w różnym zakresie czasowym klasyfikowane są na trendy spadkowe, wzrostowe i horyzontalne. Teza ta jest przeciwstawna do teorii stanowiącej o tym, że notowania rynków finansowych poruszają się zgodnie z procesem błądzenia losowego. Takie stwierdzenie zaprzecza istnieniu takiego pojęcia jak trend, ponieważ implikuje ono, iż poprzednie ruchy cen nie wpływają na jej przyszłe zachowania. Ostatnim założeniem analizy technicznej jest pogląd, że historia się powtarza. Stanowisko to wywodzi się z faktu, iż analiza bada zachowanie rynku, który tworzony jest przez wielu inwestorów. Badanie rynku sprowadza się więc do badania zachowań ludzi w różnych sytuacjach rynkowych, a te pozostają niezmienne i dlatego możemy wyciągać wnioski z przeszłych notowań rynku.

Zarzutów wobec analizy technicznej na przestrzeni lat pojawiało się wiele. Jako przykład można podać wyżej wspomniane dyskusje na temat faktu istnienia trendów na rynkach finansowych. Dodatkowo powstawało wiele opracowań sugerujących bezużyteczność tej metody do celów inwestycyjnych. Rozbieżność zdań w tej kwestii jest jednak spora. Niektóre badania sugerują bezwartościowość ponad 5 tysięcy reguł stosowanych w analizie technicznej, wskazując jednocześnie że lepsze zastosowania mogą mieć na rynkach wschodzących niż na rozwiniętych. Twórcy innych artykułów sugerują natomiast, że użycie nawet tak prostych narzędzi analizy technicznej jak np. średnia ruchoma potrafi przynosić lepsze rezultaty niż przy użyciu takich modeli szeregów czasowych jak model autoregresyjny (AR), GARCH bądź EGARCH.

Poza wskaźnikami i oscylatorami, jednym z podstawowych narzędzi analizy technicznej są formacje cenowe, czyli pewne powtarzające się układy cenowe występujące sekwencyjnie w czasie, używane do przewidywania przyszłych ruchów notowań. Formacje klasyfikowane są na te sygnalizujące odwrócenie panującego trendu oraz na te

zwiastujące jego kontynuację. Powodem popularności tego narzędzia jest przede wszystkim prostota, a także łatwość dostępu do opisu wszystkich formacji, które można znaleźć zarówno w internecie, jak i innych źródłach (Bulkowski i Wiley (2005)). Rodzi to jednak pewne problemy. Głównym z nich jest subiektywność i dwuznacznaczność formacji. John J. Murphy w książce Analiza techniczna rynków finansowych zacytował Richarda Teweles, który określił, że formacje widoczne na wykresach są tworem niemal całkowicie subiektywnym i że nikomu nie udało się dotychczas poddać ich matematycznej kwantyfikacji. Dlatego też czytając analizy przygotowywane przez specjalistów z różnych domów maklerskich często można natrafić na spore rozbieżności dotyczące prognoz cen tego samego instrumentu finansowego. Bierze się to z wcześniej wspomnianej niejasności w interpretacji wykresu cen, ponieważ dla jednego analityka dana sekwencja będzie odbierana przykładowo jako formacja głowy z ramionami (formacja odwrócenia trendu), natomiast drugi odczyta ją jako formacje klinu (formacja kontynuacji trendu).

Podsumowując rozważania na temat przydatności analizy technicznej, możemy stwierdzić że niewątpliwą zaletą jest prostota rozpoczęcia inwestycji za jej pomocą (do korzystania z wielu metod analizy technicznej, tj. Price Action potrzebny jest tylko dostęp do wykresu notowań), a także mnogość publikacji dotyczących opisu i oceny jej narzędzi. Trzeba jednak zauważyć także spore zarzuty, szczególnie jeśli chodzi o zastosowanie podstawowego narzędzia analizy technicznej, czyli formacji cenowych. Chcąc więc wydobyć pewne powtarzające się schematy, które niosłyby ze sobą pewną prognostyczną wartość, prawdopodobnie najlepszym rozwiązaniem byłoby użycie narzędzi data miningu. Na przestrzeni ostatnich lat powstało wiele artykułów (Akkaya i Uzar (2011), Zhang i Zhou (2004), Weigend (1996)) potwierdzających rosnącą rolę tej interdyscyplinarnej dziedziny nauki w zastosowaniach finansowych, głównie dzięki temu, że w sprawny sposób możliwe jest wyciągnięcie wartościowych informacji z dużych zbiorów danych. Ponadto możliwe jest uzyskanie reguł asocjacyjnych, które pomagają w odkryciu związków w pozornie nieskorelowanych danych. Celem tej pracy jest zastosowanie własnego algorytmu, korzystającego m.in. właśnie z mechanzmów data minigu i metod statystycznych do ekstrakcji takich formacji z finansowego szeregu czasowego, dzięki którym przy ponownym pojawieniu będziemy w stanie z wysokim prawdopodobieństwem przewidzieć przyszły ruch cen. Dokonana zostanie także ocena użytego algorytmu.

2 Metodologia

Finansowe szeregi czasowe głównie z powodu niestacjonarności, wrażliwości na małe zmiany oraz zawierania w sobie dużej ilości szumów są problematyczne w prognozowaniu. Aby dokonać wyciągnięcia pewnych zależności z nich, należy przejść przez pewne etapy, w których dochodzi do obróbki i przygotowania danych. W literaturze znajduje się wiele artykułów poświęconych tej tematyce. Przykładowo Dante et al. (2010) starają się odszukać korelacji w ruchach cen między spółkami wchodzącymi w skład indeksu giełdowego w Madrycie (Madrid Stock Exchange General Index). W tym celu użyty zostaje algorytm ECLAT. Zanim jednak zostaje on zastosowany, następuje wygładzanie szeregu czasowego metodą Gaussian Kernel Smoothing. Jest to wygładzanie podobne do użycia arytmetycznej średniej ruchomej. W tym przypadku jednak, starsze obserwacje są mniej istotne przy obliczaniu średniej. Dodatkowo wagi dla starszych obserwacji rozłożone są zgodnie z funkcją Gaussa (`http://bossafx.pl/fx/narzedzia/patterns-recognizer/obliczanie-formacji/`). W niektórych artykułach (Gandhmal (2011), Khan et al. (2011), Aragianni i Fetsos (2010)) dodana była faza klastrowania danych, najczęściej za pomocą metody k-średnich, bądź też Optymalizacji Stadnej Cząsteczek (Particle Swarm Optimization). W przypadku tych prac, do wygenerowania zbioru reguł użyto algorytmu CIR (Consistent Item set Rule algorithm), podobnego swym działaniem do algorytmu Apriori, a także algorytmu MFP (Most Frequent Pattern algorithm). Często też pojawiającą się koncepcją jest zastosowanie sieci neuronowych do prognozowania cen notowań (Kaastra i Boyd (1996), Virili i Freisleben (2001)).

Pomimo wielu podejść do tego tematu, pewne schematy postępowania dla różnych artykułów są podobne. Z tego względu w dalszej części pracy opisane są szczegółowo kolejne kroki zaproponowanego przeze mnie algorytmu, zawierające techniki podobne do tych użytych w wyżej wspomnianych pracach.

2.1 Czyszczenie danych

Pierwszym etapem algorytmu jest wyczyszczenie danych za pomocą odpowiedniego filtra, tak aby z jednej strony ułatwić łatwiejsze rozpoznawanie trendów, ale z drugiej strony,

żeby pierwotny szereg nie utracił swoich właściwości. Z pomocą przychodzą tutaj filtry pierwotnie stosowane jako układy elektroniczne do przetwarzania sygnału. Do ich grona można zaliczyć np.: filtr Hodricka-Prescotta czy Baxtera Kinga. Ich zadaniem jest oczyszczenie procesów ze wskazanych składników. Pierwszy z nich eliminuje trend długofalowy, podczas gdy drugi dokonuje eliminacji trendu wraz z cyklicznością (Kufel (2005)). Kolejnym przykładem może być rodzina filtrów dolnoprzepustowych, do których należy przykładowo filtr Butterwortha. Głównym ich zadaniem jest przepuszczanie tylko tych częstotliwości sygnału pierwotnego, które są poniżej ustalonej częstotliwości granicznej. Dzięki temu możemy pozbyć się z szeregu szumów o dużej częstotliwości. Warto tutaj nadmienić, że występują one powszechnie praktycznie w każdym odbiorniku radiowym.

Najczęstszym sposobem pozbywania się szumów z szeregu czasowego jest jednak zastosowanie jednej z metod wygładzania. Wyróżniamy tutaj m.in. metodę średniej, metodę ruchomej średniej, metodę ruchomej mediany oraz metodę wielomianu gładzącego (Huk (2001)). Do tej ostatniej grupy należy filtr Savitzky'ego-Golaya. Zasada działania jest następująca. Dla podanej liczby punktów (liczba ta musi być nieparzysta) dopasowywany jest wielomian stopnia przez nas z góry określonego. Wartość tego wielomianu w punkcie środkowym jest elementem wygładzonym. To postępowanie wykonywane jest dla kolejnych punktów szeregu czasowego. Wzór (1) przedstawia ogólne równanie na wartości wygładzone za pomocą tego filtra, gdzie a_i są współczynnikami dopasowania wielomianu, NP oznacza liczbę punktów wybranych do dopasowania wielomianu, a H jest normą. Tabela zawierająca współczynniki dopasowania wielomianu kwadratowego oraz odpowiednie normy znajduje się w Dodatku A

$$Y_t = \frac{1}{H} \sum_{i=-(NP-1)/2}^{(NP-1)/2} a_i x_{t+i}. \tag{1}$$

2.2 Segmentacja

Po dokonaniu odpowiedniego czyszczenia danych, należy przeprowadzić segmentację szeregu. Głównym motywem przyświecającym temu zadaniu jest znaczące przyspieszenie działania algorytmu wyznaczania reguł asocjacyjnych. Należy jednak dokonać tego za pomocą takiego narzędzia, które pozwoli na jak najlepsze zebranie informacji o charakterystyce danego szeregu.

Pierwszym proponowanym algorytmem jest algorytm segmentacji kawałkami liniowej (piecewise linear segmentation). Jednym ze sposobów zastosowania go jest użycie środowiska do obliczeń statystycznych R, a konkretnie funkcji $linearSegmentation$ zawartej w pakiecie ifultools (Constantine i Percival (2014)). Funkcja ta przyjmuje dwa argumenty: $n.fit$ oraz $angle.tolerance$. Pierwszy argument określa szerekość okien, na które dzielony jest szereg czasowy. Dla pierwszego okna dopasowywana jest prosta za pomocą metody najmniejszej kwadratów, a na jej podstawie wyliczany jest kąt nachylenia do osi odciętych. Następnie ten sam schemat powtarzany jest dla kolejnego okna i następuje porównanie kątów nachylenia. Jeśli różnica w nachyleniu prostych przekracza wielkość argumentu $angle.tolerance$, wtedy punkt pomiędzy tymi oknami jest uznany jako punkt zmiany. W przeciwnym razie powstaje nowa prosta, powstała z połączenia dwóch okien. Kolejno nowo powstała prosta porównywana jest z sąsiednią i takie postępowanie przebiega sekwencyjnie aż do ostatniego okna.

Drugim podejściem zastosowanym w tej pracy jest użycie narzędzia, które dokona segmentacji szeregu w tych miejscach, gdzie zmianie ulegają pewne statystyczne własności (np. średnia czy wariancja). Istnieje kilka algorytmów, które tego dokonują, a mianowicie: Binary Segmentation, Segment Neighbourhoods oraz Pruned Exact Linear Time (PELT). Zasada działania tego pierwszego jest następująca:

1. Przeprowadzany jest test statystyczny na istnienie pojedyńczego punktu zmiany (średniej lub wariancji). Jeśli punkt zmiany zostaje zidentyfikowany, dane dzielone są w tym miejscu na dwie części

2. Procedura poszukiwania pojedyńczego punktu zmiany jest powtarzana dla obu części. W przypadku znalezienia tych punktów, dane są dalej dzielone

3. Proces ten powtarzany jest do momentu, w którym w żadnej z części nie zostanie znaleziony punkt zmiany

Algorytm ten jest popularny ze względu na szybkość obliczeń, jednak odbywa się to czasami kosztem dokładności znalezionych punktów.

Segment Neighbourhoods prezentuje przeciwne własności— jest algorytmem dokładnym, ale o większej złożoności obliczeń.

Podczas gdy oba powyższe algorytmu stanowią pewien rodzaj kompromisu pomiędzy szybkością, a dokładnością, Pruned Exact Linear Time łączy ich mocne strony. Dzięki temu wyszukane punkty zmiany są wyznaczone precyzyjnie, a dzięki zastosowaniu dynamicznego programowania, algorytm jest efektywny obliczeniowo.

Do zastosowania powyższych algorytmów zastosowano funkcję $cpt.mean$ z pakietu changepoint (Killick et al. (2014)), która w zależności od wybranej metody przyjmuje odpowiedni argument $method$ ("$PELT$" dla Pruned Exact Linear Time, "$BinSeg$" dla Binary Segmentation, "$SegNeigh$" dla Segment Neighbourhoods). Ponadto funkcja ta pozwala na specyfikacje odpowiedniej funkcji, która będzie zapobiegać zbytniemu dopasowaniu (sytuacji, w której punktów zmiany byłoby zbyt dużo). Dostępne funkcje kary to przede wszystkim kryteria informacyjne (Schwarza, Hannana-Quinna, Akaike'a), ale można ją także zdefiniować samemu. Za wybór odpowiedniego testu statystycznego odpowiada argument $test.stat$, przyjmujący wartość "$Normal$", w przypadku gdy nasze dane mają rozkład normalny, lub "$CUSUM$" w przeciwnym wypadku.

Po dokonaniu segmentacji, zarówno za pomocą funkcji $linearSegmentation$, jak i $cpt.mean$ dokonane zostanie połaczenie tych segmentów, w których trend (wzór (2)) jest podobny. Taka procedura jest zastosowana, w celu ograniczenia tych fragmentów szeregu, które mają porównywalną charakterystyke

$$Trend_i = \frac{y_t - y_{t-1}}{y_{t-1}} * 100. \tag{2}$$

W tym celu zaimplementowałem własną funkcję $MergeSimilarTrends$, której kod znajduje się ponizej. Przyjmuje ona 3 argumenty:

- szereg - dane szeregu czasowego
- changepoints - wektor zawierający miejsca punktów zmiany
- threshold - próg, poniżej którego sąsiednie segmenty są łączone

```
1 MergeSimilarTrends<-function(szereg,changepoints,threshold){
2
3   changepoints[length(changepoints)+1]<-length(szereg)
4   changepoints1<-c(rep(NA,length(changepoints)+1))
5   changepoints1[2:length(changepoints1)]<-changepoints
```

```
  changepoints1[1]<-1
  changepoints<-changepoints1

  df<-data.frame(changepoint=changepoints,close=szereg[changepoints])

  trend<-rep(NA,length(changepoints))
  trend[1]=((df[1,2]-szereg[1])/szereg[1])*100
  for (i in 2:length(changepoints)){
    trend[i]=(((df[i,2]-df[i-1,2])/df[i-1,2])*100)
  }
  df<-data.frame(df,trend)

  similar<-rep(NA,length(changepoints))
  similar[1]='F'
  for (i in 2:length(changepoints)){
    ifelse(abs(df[i,3]-df[i-1,3])<=threshold,similar[i]<-'T',similar[i]<-'F')
  }
  df<-data.frame(df,similar)
  df$similar<-as.character(similar)

  rows_to_delete<-which(grepl('T',df$similar))-1
  df1<-df[-c(rows_to_delete),-c(3,4)]

  trend1<-rep(NA,length(df1[,1]))
  trend1[1]=((df1[1,2]-szereg[1])/szereg[1])*100
  for (i in 2:length(df1[,1])){
    trend1[i]=(((df1[i,2]-df1[i-1,2])/df1[i-1,2])*100)
  }
  df1<-data.frame(df1,Trend=trend1)

  DF<<-df1
  segmenty<<-df1$changepoint
}
```

2.3 Symboliczna reprezentacja segmentów

Posiadając podzielony szereg czasowy na segmenty o odpowiedniej charakterystyce trendu w nim panującego, oraz długości, należy dokonać budowy słownika, tak aby wyniki otrzymane z algorytmu użytego do wyznaczenia reguł asocjacyjnych były czytelne i łatwe w interpretacji.

Zakładając z góry, iż wiemy ile klas znajduje się w szeregu, moglibyśmy dokonać odpowiedniego podziału na podstawie histogramu czasu trwania fragmentów, oraz trendu, dobierając te parametry według własnej opinii. Takie postępowanie byłoby jednak subiektywne i mogłoby prowadzić końcowo do błędnych wniosków.

Biorąc to pod uwagę, racjonalnym pomysłem byłoby użycie jednego z narzędzi analizy skupień. Biecek, Przemysław Trajkowski (2011) określił ją jako zbiór metod pozwalajacych na wyróznienie zbiorów obserwacji (nazywanych skupieniami lub klastrami) podobnych do siebie. Procedura podziału na zbiory nazywana jest grupowaniem bądź klasteryzacją.

Jak wspominałem wcześniej we wprowadzeniu do tego rozdziału, często stosowanym w podobnych pracach narzędziem jest metoda k-średnich. Dokonuje ona podziału zbioru na zadaną ilość grup, a dokładniej rzecz ujmując, wyznacza współrzędne punktów, które będą środkami klastrów. Dany segment będzie należał do tego zbioru, którego środek znajduje się w najmniejszej odległości od niego.

2.4 Rule extraction

Po skończonej fazie przygotowania danych, przychodzi kolej na ich eksplorację, czyli uzyskanie użytecznych informacji. Jedną z najpopularniejszych metod eksploracji jest metoda asocjacyjna (metoda odkrywania asocjacji), która polega na przeszukiwaniu rekordów bazy danych w celu znalezienia w niej powtarzających się zależności. Stosuje się ją tam, gdzie występuja potrzeba scharakteryzowania związków przyczynowo-skutkowych pomiędzy danymi.

Istnieją dwie miary określające reguły asocjacyjne: wsparcie (support), oraz pewność (confidence). Wsparcie jest parametrem, który określa jaki procent wszystkich reguł asocjacyjnych stanowi ta wybrana reguła. Pewność reguły stanowi jaki procent transakcji zawartych w lewej stronie reguły prowadził do prawej strony. Wzory (3) oraz (4) przed-

stawiają formalny zapis powyższych parametrów

$$wsparcie = P(A \cap B) = \frac{liczba\ transakcji\ zawierajacych\ A\ i\ B}{calkowita\ liczba\ transakcji}, \tag{3}$$

$$pewnosc = P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{liczba\ transakcji\ zawierajacych\ A\ i\ B}{liczba\ transakcji\ zawierajacych\ A}. \tag{4}$$

W opracowaniach znajduje się wiele algorytmów, służących generowaniu reguł asocjacyjnych, które powinny dawać podobne rezultaty, jednak różnią się efektywnością obliczeniową. Na potrzeby tej pracy, użyty zostanie najpopularniejszy algorytm Apriori.

Swoje działanie uzasadnia na podstawie własności funkcji wsparcia, która mówi, że powiększenie zbioru nieczęstego (zbioru zdarzeń, który występuje rzadziej niż zadane przez użytkownika minimalne wsparcie) o dodatkowe składowe nie spowoduje utworzenia zbioru częstego. Innymi słowy, w każdej iteracji tego algorytmu, odrzucane są te zbiory, dla których wsparcie jest niewystarczające. Powiększane są natomiast zbiory częste, do momentu kiedy jest to możliwe.

2.5 Ocena jakości algorytmu

Ostatnim elementem jest ocena tego, jak znalezione przez algorytm asocjacje sprawdzają się do prognozowania. W tym celu zbiór danych podzielony zostanie na dwie części: zbiór treningowy i zbiór testowy. Na pierwszym z nich dokonywana jest faza uczenia, tzn. generowane są na nim reguły asocjacyjne. W drugiej fazie, zwanej fazą testowania, następuje weryfikacja dokładności wyznaczonych asocjacji. Dokładniej rzecz ujmując, na zbiorze testowym wyszukiwane są lewe strony reguł, a następnie sprawdzane jest czy następny segment pokrywa się z tym zawartym w asocjacji. Na tej podstawie wyliczany jest współczynnik dokładności (wzór (5)), stanowiący procent wszystkich poprawnie zakwalifikowanych przypadków.

Ocena trafności prognostycznej wyznaczonych reguł zwana jest często sprawdzianem krzyżowym. Pozwala uniknąć ona tzw. błędu trzeciego rodzaju. Bez jego użycia, niemożliwe jest określenie czy wyznaczone reguły będą działały równie dobrze dla danych, nie użytych to ich konstrukcji. Jeśli dane asocjacje będą sprawdzały się równie dobrze na zbiorze treningowym i testowym, wtedy określa się, że zbiór reguł pozytywnie przeszedł sprawdzian krzyżowy

$$wspolczynnik\ dokladnosci = \frac{poprawnie\ zaklasyfikowane\ reguly}{wszystkie\ reguly\ ukazane\ w\ zbiorze\ testowym}. \quad (5)$$

3 Zastosowanie algorytmu dla indeksu giełdowego WIG20

Rozdział ten zawiera zastosowanie algorytmu opisanego we wcześniejszej części pracy, dla Warszawskiego Indeksu Giełdowego dwudziestu największych spółek notowanych na Giełdzie Papierów Wartościowych w Warszawie.

WIG20 jest indeksem giełdowym reprezentującym 20 największych i najpłynniejszych spółek notowanych na Giełdzie Papierów Wartościowych w Warszawie. Wyliczany jest od 16 kwietnia 1994 r., przyjmując poziom bazowy w wysokości 1000 punktów. Publikowany będzie do końca grudnia 2015 r., ponieważ zostanie docelowo zastąpiony przez indeks WIG30. Wzór (6) przedstawia formułę pozwalającą wyliczyć indeks na daną chwilę. WIG20 jest indeksem cenowym, ponieważ bierzę pod uwagę tylko ceny zawartych transakcji (bez dywidend) przy kalkulacji. Skład tego indeksu ulega zmianie, kiedy dana spółka nie spełnia określonych wymogów (m.in. wymogów dotyczących liczby akcji w wolnym obrocie). Zmiany te dokonywane są podczas rewizji kwartalnych przeprowadzanych w trzeci piątek czerwca, września i grudnia, oraz podczas rewizji rocznej, dokonywanej w trzeci piątek marca

$$WIG20(t) = \frac{M(t)}{M(0) * K(t)} * 1000. \tag{6}$$

Powodów, dla których warto użyć akurat tego indeksu, zamiast wielu innych, czy też innych instrumentów finansowych jest kilka:

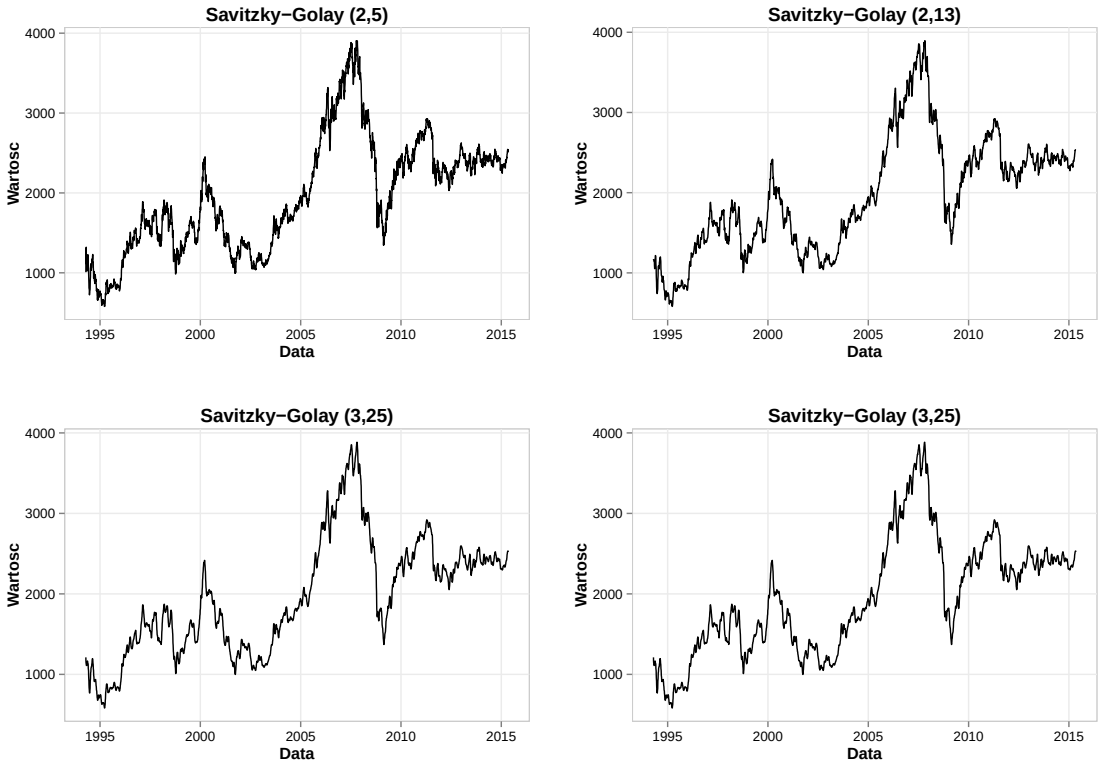
- pozwala śledzić koniunkturę w sektorze największych przedsiębiorstw,
- uznawany jest jako barometr sytuacji gospodarczej w kraju,
- stanowi podstawę do oceny wyników inwestycyjnych, zarówno dla inwestorów indywidualnych, jak i instytucjonalnych,
- jest instrumentem bazowym dla kontraktów terminowych i opcji notowanych na GPW.

Dane użyte w dalszej części pracy zostały pobrane z serwisu internetowego stooq.pl i zawierają wartości WIG20 pomiędzy 18 kwietnia 1994 r. a 8 maja 2015 r. (5235 obserwacji). Notowania indeksu w tym okresie przedstawia rysunek 1.

Wykres 1: WIG20 18 kwietnia 1994 r. - 8 maja 2015 r. *Źródło:* opracowanie własne

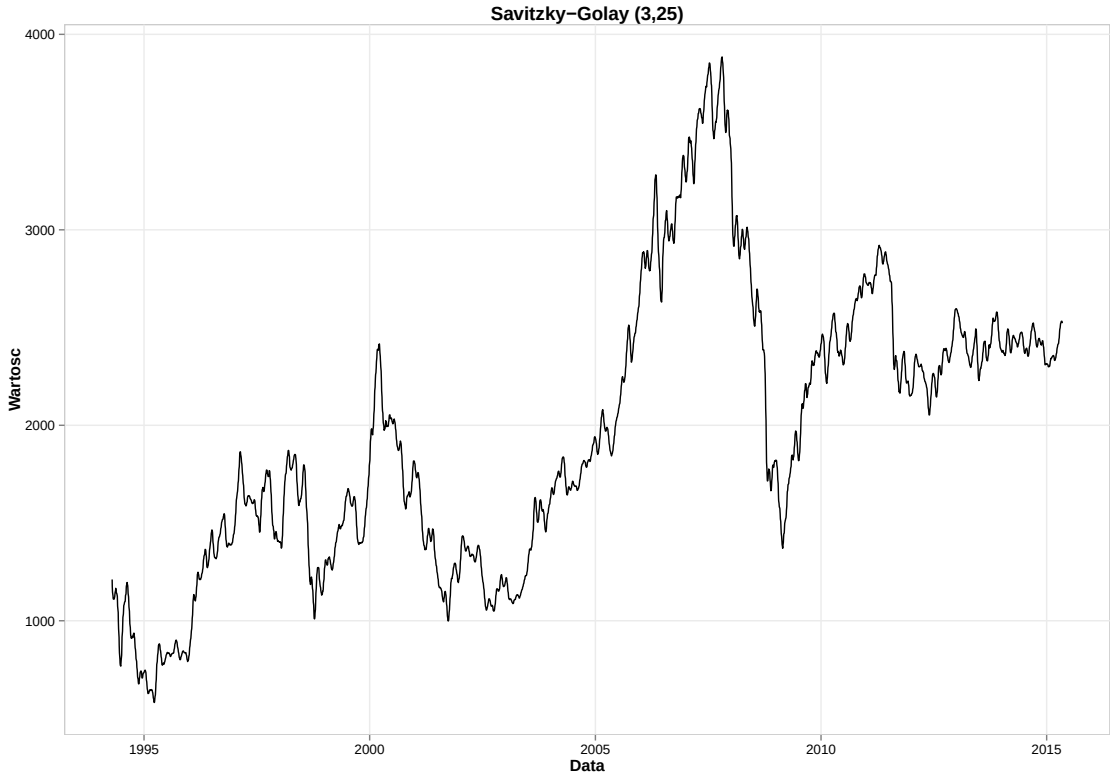
3.1 Czyszczenie danych filtrem Savitzky'ego-Golaya

Łatwo można zauważyć na podstawie rysunku 1, że notowania te zawierają wiele szumów, które uniemożliwiłyby poprawną klasyfikacje trendów w wybranych segmentach. Aby uniknąć tego problemu, zostanie użyty filtr Savitzky'ego-Golaya. Jak już wcześniej było wspomniane, określają go dwa parametry: stopień wielomianu użytego do filtracji, oraz liczba punktów użyta do oszacowania tego wielomianu. Należy tutaj jednak pamiętać, by dobrać te parametry tak, aby uzyskać kompromis pomiędzy wygładzaniem danych, a jak najwierniejszym oddaniem własności pierwotnego szeregu. Rysunek 2 przedstawia porównanie oryginalnego indeksu WIG20 wraz z 3 wykresami użycia filtru Savitzky'ego-Golaya dla różnych parametrów.



Wykres 2: Porównanie zastosowania filtru Savitzky'ego-Golaya dla różnych zestawów parametrów. *Źródło:* opracowanie własne

Jako, że przy tak długim szeregu czasowym zależy nam przede wszystkim na ograniczeniu krótkookresowych wahań, ostatni zestaw parametrów (szereg wygładzony za pomocą wielomianu trzeciego stopnia oszacowanego za pomocą 25 punktów) wydaje się być najbardziej odpowiedni. Można też dostrzec, że użycie takiej kombinacji nie powoduje zniekształcenia tendencji panującej na rynku w tym okresie (rysunek 3).



Wykres 3: Zastosowanie filtru Savitzky'ego-Golaya do indeksu WIG20. *Źródło:* opracowanie własne

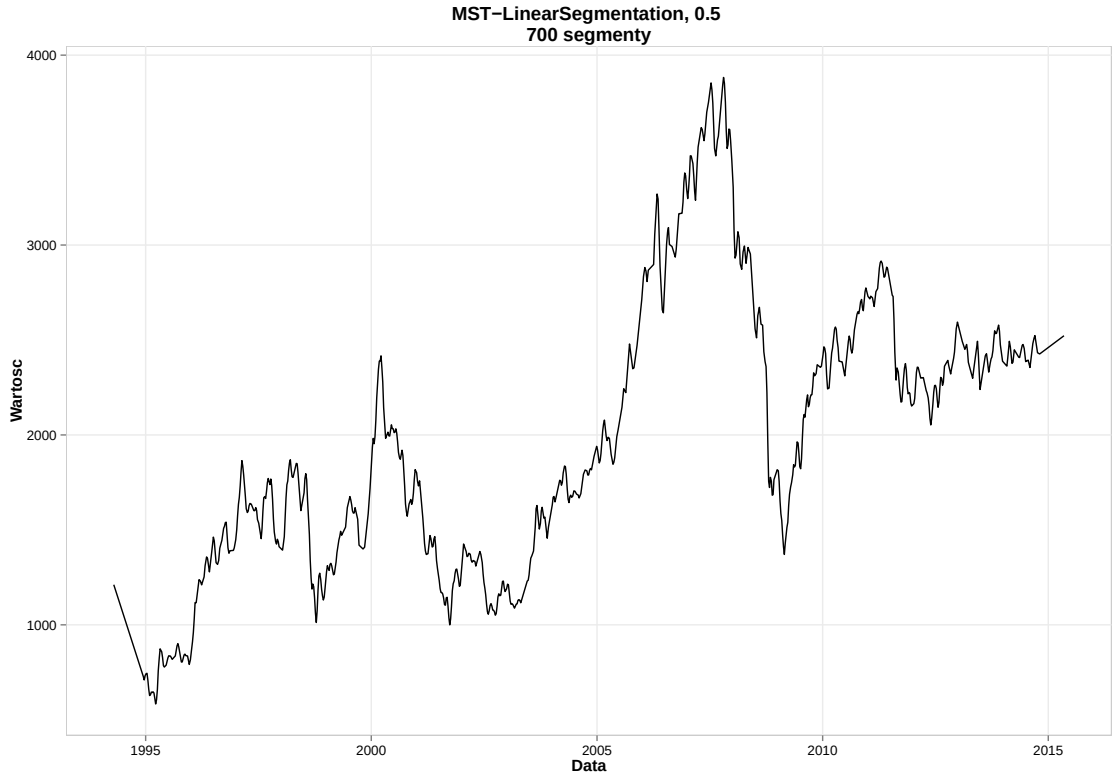
3.2 Segmentacja danych

Kolejny krok algorytmu stanowi segmentacja szeregu. Jako że proponowane wcześniej metody liniowej segmentacji i binarnej segmentacji różnią się znacząco w swoim działaniu, dlatego też ciężko stwierdzić użycie którego doprowadzi do lepszych wyników. Z tego względu dalsza część algorytmu przeprowadzana będzie oddzielnie dla obu metod.

Funkcja $linearSegmentation$ przyjmuje dwa argumenty określające początkową liczbę obserwacji zawartą w każdym segmencie, oraz kąt tolerancji, poniżej którego sąsiednie segmenty są łączone. W naszym przypadku wielkość okna ustawiamy na 5 dni, jako że wielkość ta odpowiada długości tygodnia roboczego, kiedy to występuje handel na rynku. Ponadto chcemy wyeliminować tendencje panujące na rynku o krótszym czasie trwania, ponieważ są one zazwyczaj o małym znaczeniu i prowadziłyby do nadmiaru informacji. Aby zapobiec powstaniu zbyt małej ilości segmentów spowodowanej dużą tolerancją przy łączeniu sąsiednich okien, drugi argument zostaje ustawiony na $0.1°$.

Oznacza to, że w przypadku gdy różnica w kącie nachylenia prostych między segmentami jest większa niż 0.1°, wtedy punkt zmiany zostaje odnotowany. W wyniku działania tej funkcji powstało 877 segmentów.

Następnie w celu połączenia tych okien, których trend jest zbliżony do siebie, stosujemy funkcję $MergeSimilarTrends$. Jako próg, poniżej którego dochodzi do zespojenia sąsiadujących elementów, przyjmujemy wartość 0.5%. Stosując taki schemat działania, ostatecznie wygenerowane zostaje 700 segmentów. Ich przebieg przedstawia rysunek 4.



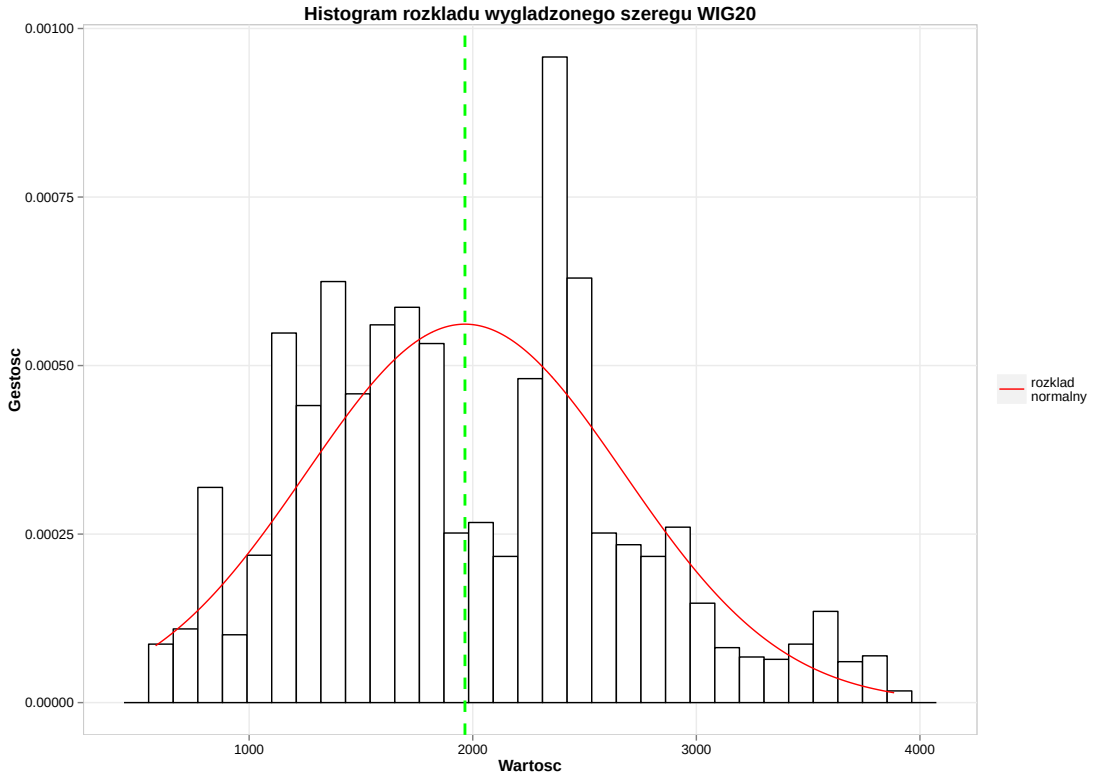
Wykres 4: Zastosowanie funkcji linearSegmentation, a następnie MergeSimilarTrends do odszumionego szeregu indeksu WIG20. *Źródło:* opracowanie własne

Drugim podejściem jest zastosowanie funkcji $cpt.mean$. Jeden z parametrów tej funkcji każe określić, czy dane użyte do segmentacji mają rozkład normalny w celu doboru odpowiedniego testu statystycznego.

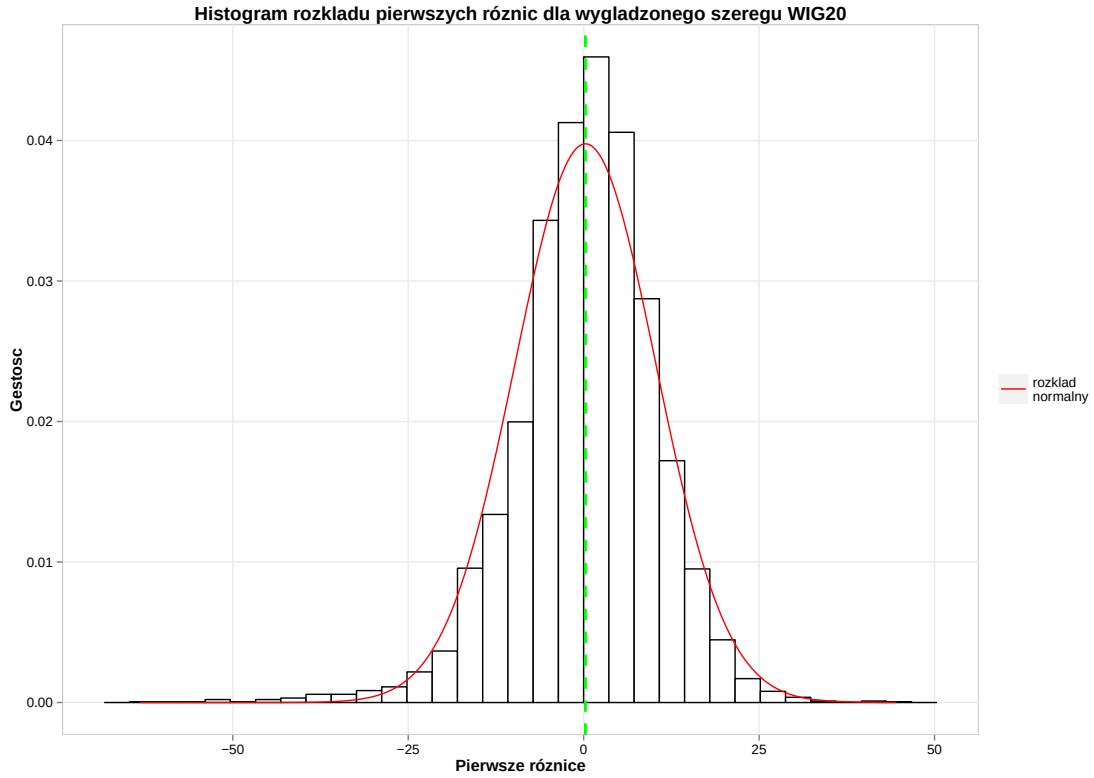
Aby tego dokonać, użyty zostanie test Kołmogorowa-Smirnova. Jest to test nieparametryczny, pozwalający na badanie podobieństwa rozkładu danej zmiennej z innym

rozkładem. W naszym przypadku hipoteza zerowa zakłada, że rozkład wygładzonych obserwacji indeksu WIG20 jest zbliżony do normalnego.

Histogramy odfiltrowanych obserwacji oraz ich pierwszych różnic przedstawia rysunek 5 oraz rysunek 6. Wartości p-value dla przeprowadzonych testów prezentuje tabela 1.



Wykres 5: Histogram rozkladu wygladzonego szeregu WIG20. *Źródło:* opracowanie własne



Wykres 6: Histogram rozkladu pierwszych różnic wygladzonego szeregu WIG20. *Źródło:* opracowanie własne

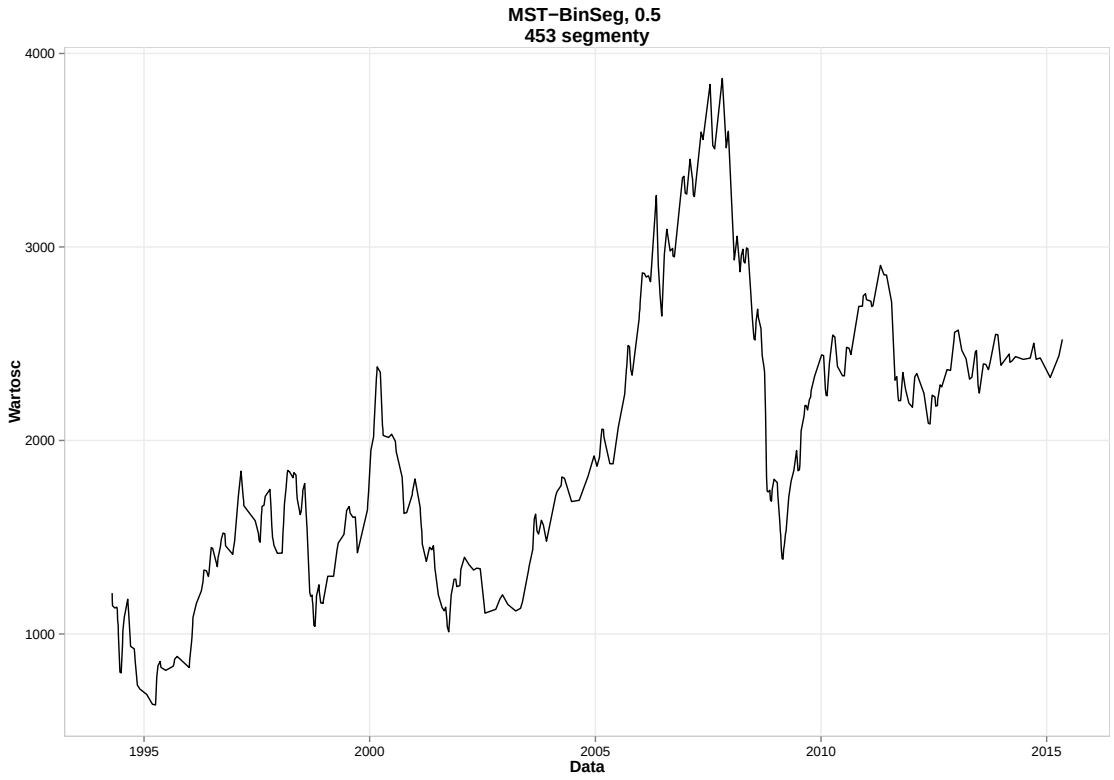
Tabela 1: Wartości p-value dla testu Kołmogorowa-Smirnova. *Źródło:* opracowanie własne

$data$	$wygladzone$	$diff(wygladzone)$
D	1	0.4221
p-value	<2.2e-16	<2.2e-16

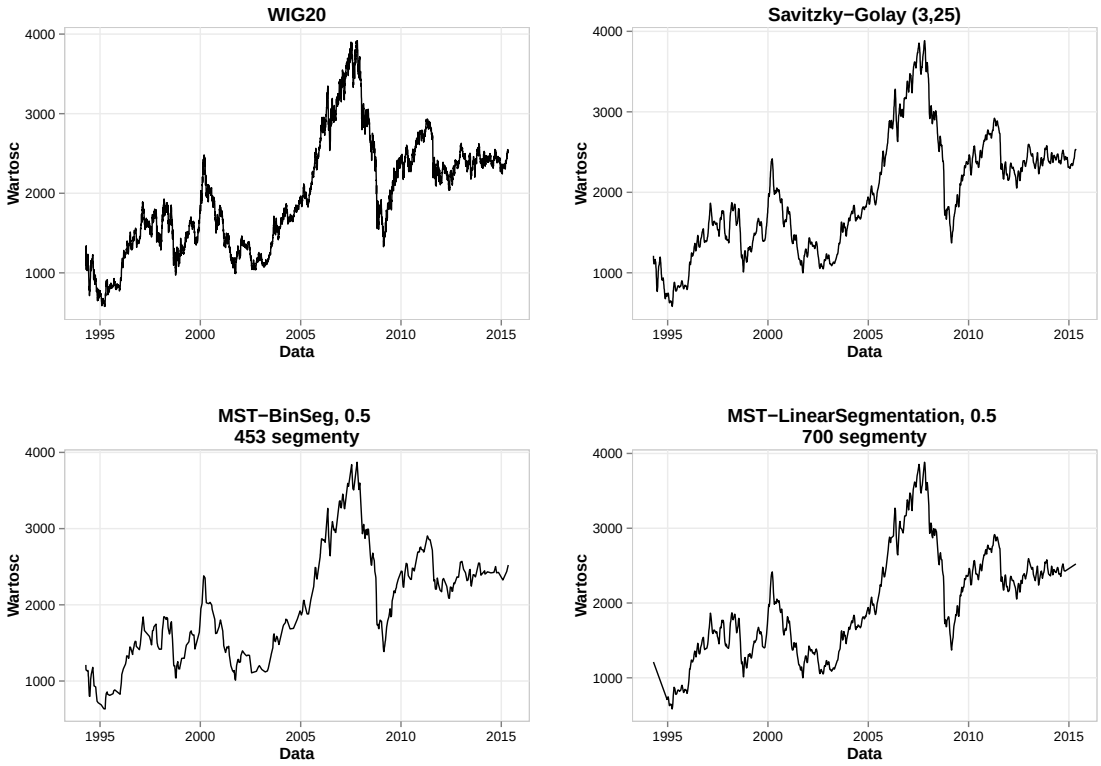
Na podstawie wartości p-value odrzucamy hipotezę zerową i stwierdzamy, iż nasza zmienna posiada rozkład różny od normalnego. Dlatego też argument $test.stat$ określamy jako $CUSUM$. Jako funkcję kary, chroniącą przed zbytnim dopasowaniem, przyjmujemy kryterium informacyjne Hannana-Quinna.

Po zaaplikowaniu tej funkcji otrzymujemy 1186 segmenty. Nastepnie, analogicznie jak w poprzedniej metodzie, stosujemy funkcję $MergeSimilarTrends$, przyjmując próg również w wysokości 0.5%. Ostatecznie szereg zostaje podzielony na 453 segmenty. Rysunek 7 przedstawia końcowy efekt użycia funkcji $cpt.mean$ oraz

$MergeSimilarTrends$, natomiast rysunek 8 pokazuje porównanie pierwotnego szeregu czasowego, szeregu po użyciu filtru oraż po użyciu liniowej i binarnej segmentacji.



Wykres 7: Zastosowanie funkcji cpt.mean, a następnie MergeSimilarTrends do odszumionego szeregu indeksu WIG20. *Źródło:* opracowanie własne



Wykres 8: Porównanie pierwotnego, odszumionego i posegmentowanego szeregu WIG20. *Źródło:* opracowanie własne

3.3 Budowa słownika

Przyszedł czas na konwersję otrzymanych segmentów na łatwe w interpretacji i wnioskowaniu klasy. Aby tego dokonać, przyjrzyjmy się bliżej podstawowym statystykom opisowym dotyczącym trendu oraz czasu trwania segmentów dla obu metod (tabela 2,3).

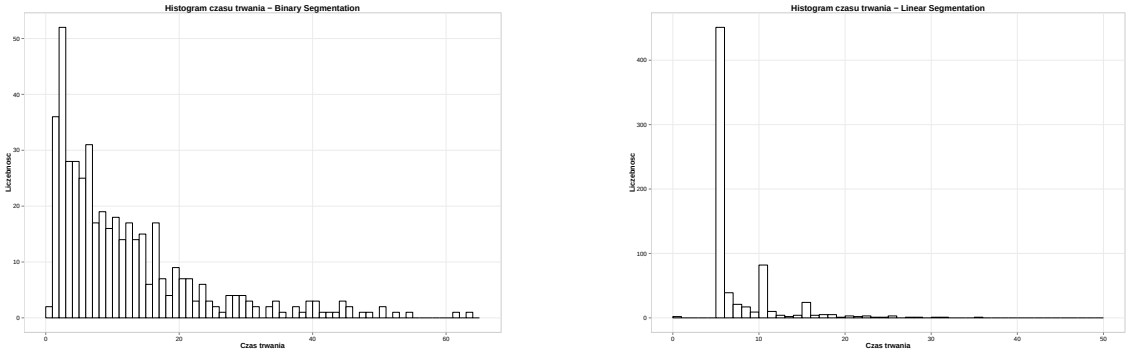
Tabela 2: Statystyki opisowe dotyczące trendu w segmentach. *Źródło:* opracowanie własne

Metoda	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
LinearSegmentation	-39.9000	-1.8170	0.1180	0.1874	2.1670	14.9300
BinarySegmentation	-19.92000	-2.03900	-0.09549	0.28410	2.49500	18.51000

Tabela 3: Statystyki opisowe dotyczące czasu trwania poszczególnych segmentów. *Źródło:* opracowanie własne

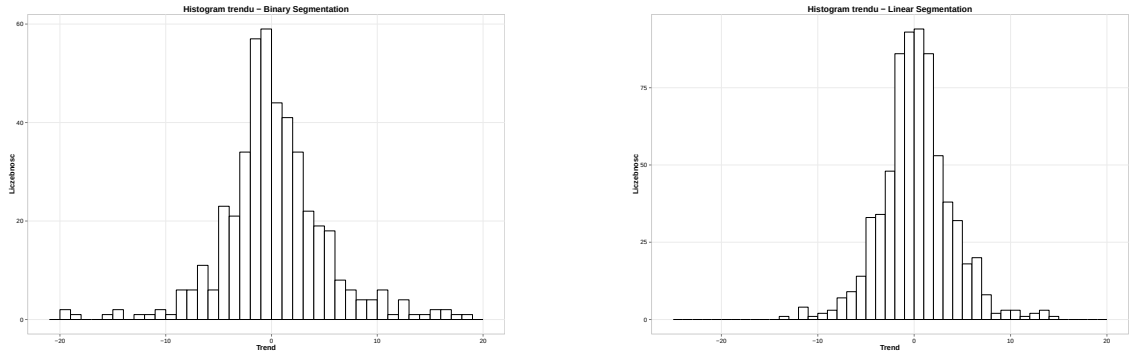
Metoda	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
LinearSegmentation	0.000	5.000	5.000	7.477	8.000	131.000
BinarySegmentation	0.00	3.00	8.00	11.55	16.00	66.00

Można zauważyć, iż segmenty utworzone za pomocą funkcji $linearSegmentation$ charakteryzują się bardziej ekstremalnymi wartościami jeśli chodzi o trend. Zastosowanie binarnej segmentacji prowadzi natomiast do powstania okien o węższym zakresie trendu, jednakże wartości pierwszego i trzeciego kwantyla sugerują, że więcej obserwacji znajduje się w ogonach rozkładu. Ujemna mediana wskazuje także na większą liczbe segmentów z trendem spadkowym. Jeśli chodzi o czas trwania poszczególnych okien, mediana i średnia w przypadku segmentacji liniowej wskazują na dużą liczbę fragmentów o długości 5 dni, czyli tych przyjętych w funkcji jako początkową szerokość każdego okna. Pomimo informacji uzyskanych za pomocą statystyk opisowych, chcąc dokonać podziału tych dwóch zmiennych na określoną ilość klas, potrzebowalibyśmy przeanalizować jaki rozkład mają te zmienne. W tym celu posłużymy się histogramami czasu trwania i trendu(wykres 9,10).



(a) Binary Segmentation (b) Linear Segmentation

Wykres 9: Histogram czasu trwania poszczególnych segmentów dla odpowiednich metod. *Źródło:* opracowanie własne



(a) Binary Segmentation (b) Linear Segmentation

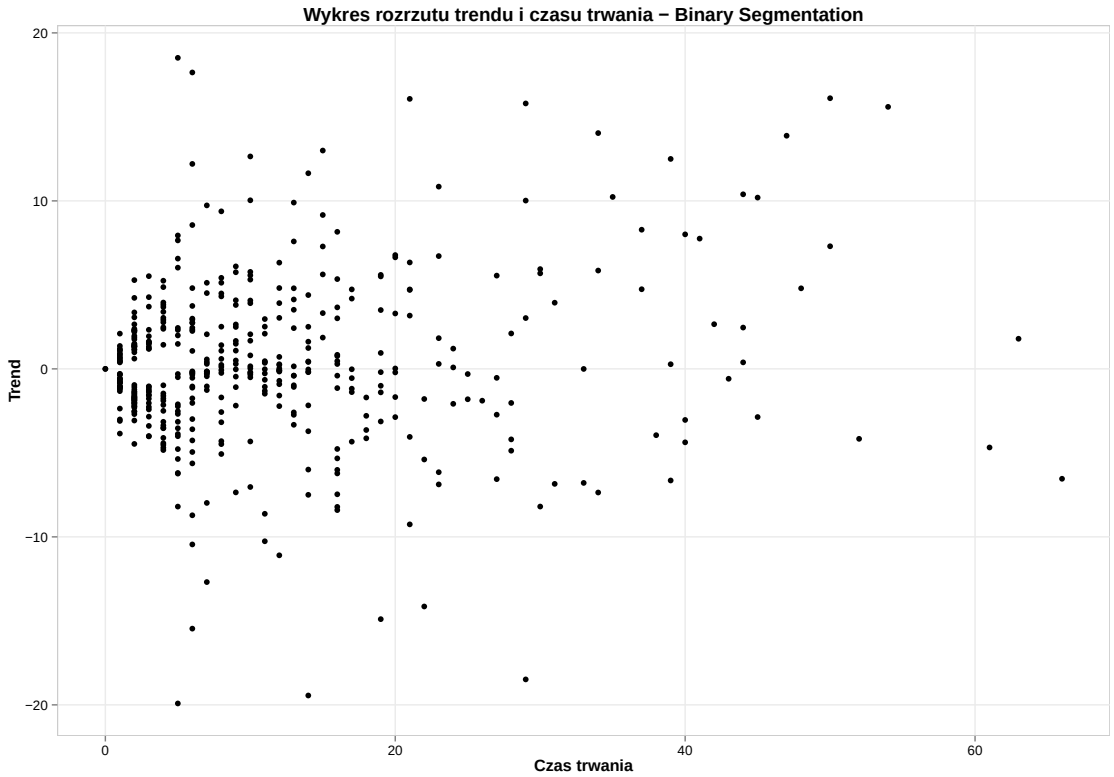
Wykres 10: Histogram trendów panujących w segmentach dla poszczególnych metod. *Źródło:* opracowanie własne

Jak widać, jeśli chodzi o czas trwania poszczególnych okien, segmentacja binarna zdaje się dawać bardziej rozłożone wyniki, podczas gdy użycie segmentacji liniowej, jak już wcześniej zostało zauważone, powoduje że znaczna większość fragmentów pozostaje długości podanej początkowo do funkcji. W przypadku trendu widać, że oba rozkłady są zbliżone do siebie, najczęstszą grupą segmentów są te o trendzie oscylującym koło zera. Widoczne jest jednak większe skupienie segmentów z tendencją spadkową dla $BinarySegmentation$.

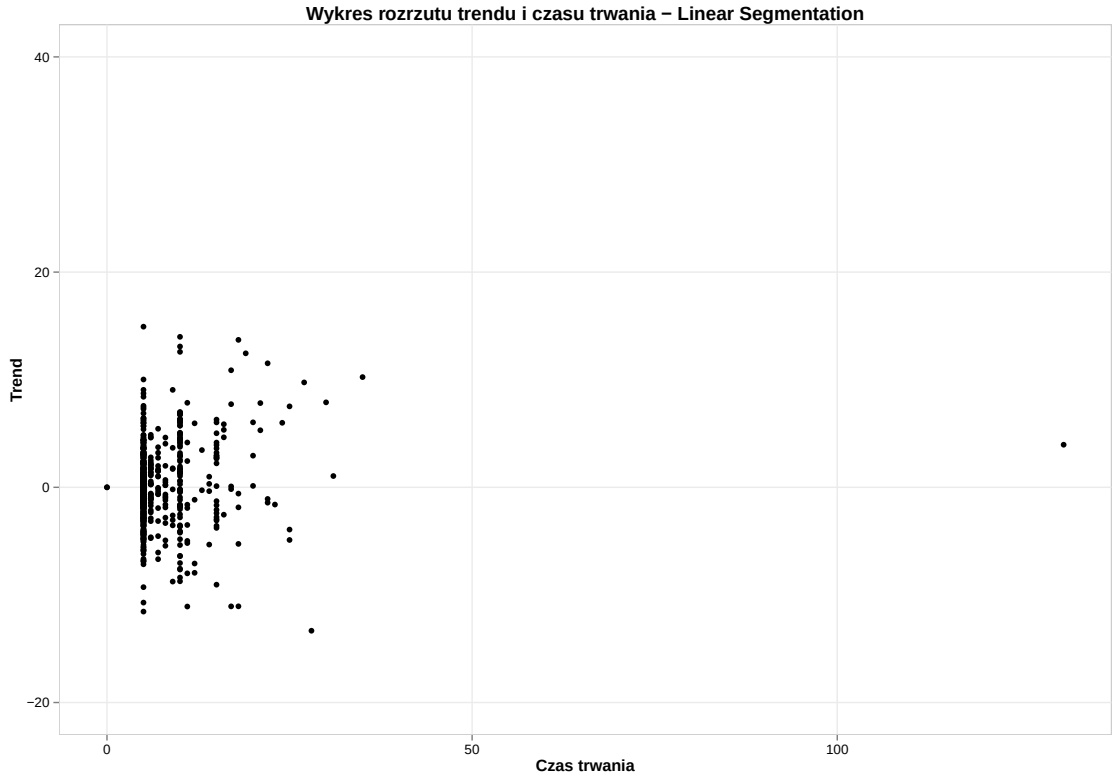
Posiadając takie dane możliwe jest wnioskowanie na temat stworzenia odpowiednich elementów do słownika. Przykładowo czas trwania okien utworzonych za pomocą funkcji $cpt.mean$ możnaby podzielić na 3 przedziały: $\langle 0, 5)$, $\langle 5, 15)$, $\langle 15, \infty)$ i sklasyfikować je za pomocą symboli: S (krótki czasowo segment), M (segment o średniej długości), L (segment długoterminowy). Powyższe uporządkowanie doprowadziłoby do następującej alokacji obserwacji w tych klasach: 37.75%, 36.87%, 25.38%. Analogicznie podziału możnaby dokonać dla trendu wydzielając 5 klas: VN (trendy o charakterystyce skrajnie spadkowej dla przedziału $(-\infty, -4\rangle$) , N (trendy spadkowe dla przedziału $(-4, -1\rangle$), C (trendy horyzontalne dla przedziału $(-1, 1\rangle$), P (trendy wzrostowe dla przedziału $(1, 5\rangle$) oraz VP (trendy o charakterystyce wyraźnie wzrostowej dla przedziału $(5, \infty\rangle$). Taki podział spowodowałby, że kolejno w każdej z tych klas znalazłoby się 13.9%, 24.72%, 22.74%, 25.6%, 13.04% obserwacji.

W wyniku powyższej reprezentacji symbolicznej utworzonych zostałoby 15 klas opisujących poszczególne okna (np.: klasa VN_L oznaczałaby segment o długim przedziale

czasowym, w którym występował trend spadkowy). Takie postępowanie sprawia jednak kilka problemów, z których głównym jest subiektywność. Dotyczy ona zarówno doboru liczby klas, jak i wartości granicznych. Tworząc bowiem wyżej wymienione klasy, starałem się dobrać parametry tak, aby alokacja obserwacji w każdej z grup była jak najbardziej zrównoważona. Jednakże takie postepowanie może okazać się błędne, ponieważ mogą istnieć w zbiorze pewne mniej liczne grupy o właściwościach odbiegających od pozostałych. Dlatego też właściwszym wydaje się zastosowanie metody k-średnich, która wyróżnia zbiory składające się z obserwacji podobnych do siebie, a jej jedynym parametrem jest liczba grup. Aby właściwie dobrać tą zmienną przyjrzyjmy się wykresom rozrzutu trendu i czasu trwania dla obu metod (wykres 11, 12).

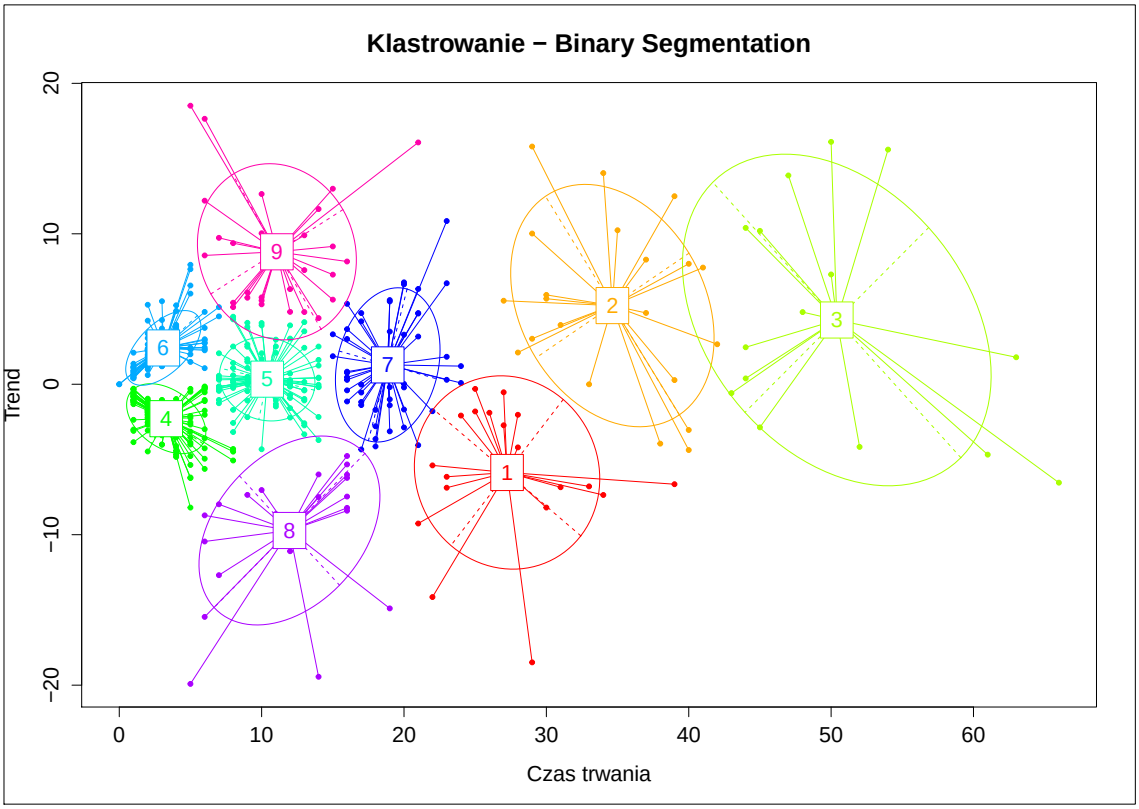


Wykres 11: Wykres rozrzutu trendu i czasu trwania dla Binary Segmentation.
Źródło: opracowanie własne



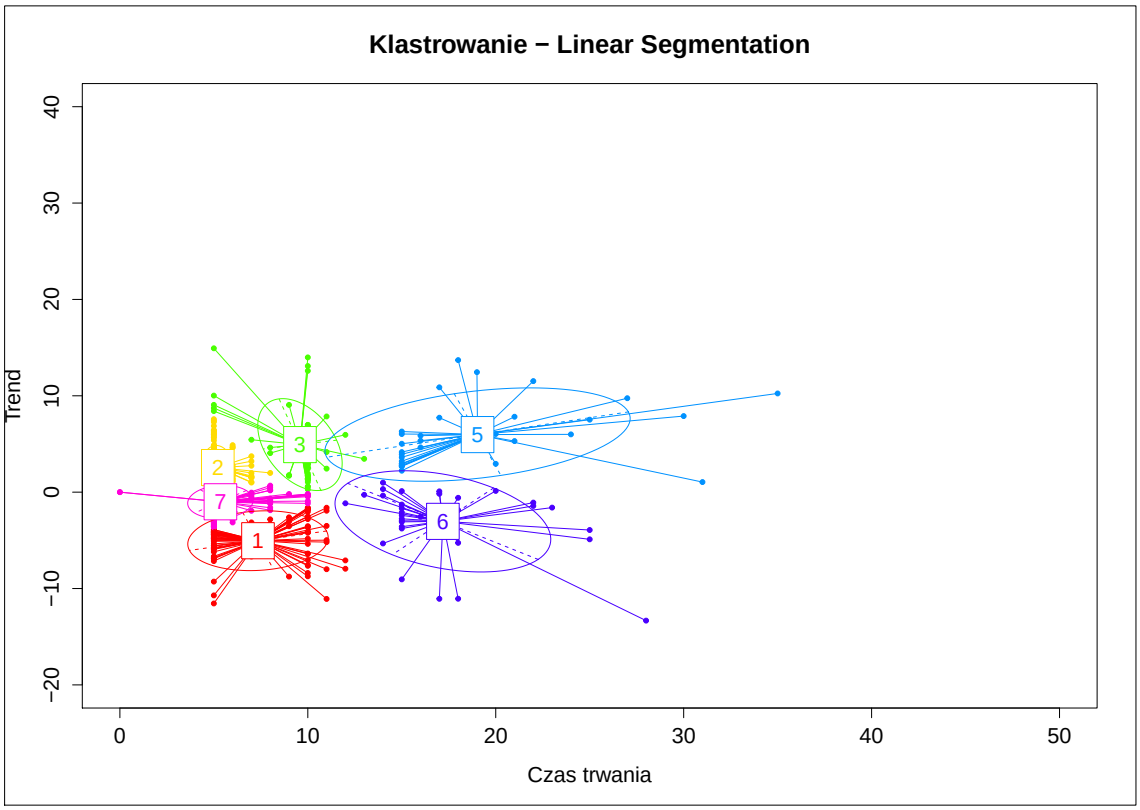
Wykres 12: Wykres rozrzutu trendu i czasu trwania dla Linear Segmentation.
Źródło: opracowanie własne

Jak można zauważyć, dla $BinarySegmentation$ segmenty są bardziej równomiernie rozłożone w stosunku do $linearSegmentation$, gdzie większość okienek skupiona jest dla czasu trwania pomiędzy 5-10 dni. To sugrowałoby większą ilość klas dla metody dzielącej szereg w miejscach istotnej zmiany średniej. Graficzną reprezentację klastrowania użytego do wyznaczenia odpowiednich klas przedstawia wykres 13 oraz 14.



Wykres 13: Graficzna reprezentacja klastrowania dla Binary Segmentation.

Źródło: opracowanie własne



Wykres 14: Graficzna reprezentacja klastrowania dla Linear Segmentation.
Źródło: opracowanie własne

3.4 Wygenerowanie reguł asocjacyjnych

Ostatnim krokiem przed wyznaczeniem reguł asocjacyjnych jest podział szeregu na zbiór treningowy oraz zbiór testowy, w celu późniejszej oceny użytego algorytmu. Gdy mamy do czynienia z dużym zbiorem danych, około 50%-75% obserwacji przypisujemy do próby uczącej, a reszte do testowej. Najczęściej jednak zbiór testowy stanowi mniej niż 1/3 całej próby.

W naszym przypadku, segmenty wyznaczone metodą $BinarySegmentation$ stanowią o 36% mniej liczny zbiór w porównaniu do okienek wyznaczonych funkcją $linearSegmentation$ (453 w stosunku do 700). Dlatego też dla pierwszej metody dokonujemy podziału zbioru w taki sposób, że zbiór treningowy zawiera 80% obserwacji, natomiast testowy 20%. Na próbie treningowej stosujemy algorytm Apriori, dokonując selekcji tylko tych reguł, których pewność jest większa niż 0.5. Ze względu na stosun-

kowo małą liczebność zbioru (362 obserwacje), dana asocjacja musi pojawić się co najmniej 3 razy. Wyniki otrzymanych reguł przedstawia tabela 4.

Jak można zauważyć, ze względu na to, że klastry zawierające klasy 4 oraz 6 stanowiły najliczniejsze zbiory segmentów, dlatego też głównie one są składnikami wyznaczonych reguł.

Przykładowo, asocjacja nr. 2 stanowi, że segmenty o krótkim okresie trwania, z lekko negatywnym trendem, a następnie o długim okresie czasu (ok. 20-30 dni) z trendem wyraźnie spadkowym (do -20%), najczęściej zwiastują około 2 tygodniowy okres na rynku o trendzie horyzontalnym.

Asocjacja nr. 11 natomiast implikuje, że krótkoterminowe trendy spadkowe są powodowane przez sekwencje trzech segmentów: segmentu o czasie trwania pomiędzy ok. 7-20 dni z trendem wyraźnie wzrostowym (do 20%), krótkoterminowego trendu wzrostowego (0-5%), oraz segmentu o czasie trwania ok.10 dni z trendem horyzontalnym.

Tabela 4: Reguły wyznaczone za pomocą metody Binary Segmentation. *Źródło:* opracowanie własne

Lp.	Reguła	Wsparcie	Pewność
1	{lag-2=kl_2,lag-1=kl_6} => {lag0=kl_6}	0.01	0.50
2	{lag-2=kl_4,lag-1=kl_1} => {lag0=kl_7}	0.01	0.60
3	{lag-2=kl_4,lag-1=kl_8} => {lag0=kl_4}	0.02	0.60
4	{lag-2=kl_8,lag-1=kl_4} => {lag0=kl_4}	0.02	0.70
5	{lag-2=kl_9,lag-1=kl_6} => {lag0=kl_5}	0.01	0.50
6	{lag-2=kl_5,lag-1=kl_5} => {lag0=kl_4}	0.02	0.60
7	{lag-2=kl_5,lag-1=kl_4} => {lag0=kl_4}	0.04	0.61
8	{lag-2=kl_4,lag-1=kl_6} => {lag0=kl_6}	0.01	0.75
9	{lag-2=kl_4,lag-1=kl_4} => {lag0=kl_4}	0.08	0.51
10	{lag-3=kl_4,lag-2=kl_8,lag-1=kl_4} => {lag0=kl_4}	0.01	0.67
11	{lag-3=kl_9,lag-2=kl_6,lag-1=kl_5} => {lag0=kl_4}	0.01	0.60
12	{lag-3=kl_6,lag-2=kl_9,lag-1=kl_6} => {lag0=kl_6}	0.01	0.60
13	{lag-3=kl_5,lag-2=kl_5,lag-1=kl_4} => {lag0=kl_4}	0.01	0.67
14	{lag-3=kl_5,lag-2=kl_4,lag-1=kl_4} => {lag0=kl_4}	0.02	0.57
15	{lag-3=kl_6,lag-2=kl_5,lag-1=kl_5} => {lag0=kl_4}	0.01	0.60
16	{lag-3=kl_4,lag-2=kl_5,lag-1=kl_6} => {lag0=kl_6}	0.01	0.50
17	{lag-3=kl_4,lag-2=kl_5,lag-1=kl_4} => {lag0=kl_4}	0.01	1.00
18	{lag-3=kl_4,lag-2=kl_4,lag-1=kl_6} => {lag0=kl_6}	0.01	0.75
19	{lag-3=kl_4,lag-2=kl_4,lag-1=kl_4} => {lag0=kl_4}	0.04	0.53

Dla liniowej segmentacji, ze względu na większą ilość wyznaczonych segmentów, zbiór treningowy będzie zawierał 66% obserwacji (462 segmenty), natomiast zbiór testowy 33% (238 segmenty). Również w tym przypadku minimalną pewność reguł ustawiamy na 0.5, natomiast minimalne pojawienie się asocjacji zwiększamy do 5 razy. Wyniki przedstawia tabela 5.

Jak i poprzednio, najbardziej liczne klasy powstałe w wyniku klastrowania stanowią największą część składową utworzonych reguł asocjacyjnych.

Analizując strukturę wyznaczonych asocjacji, można zauważyć, że przewidywania dotyczą całkowicie tylko dwóch klas (segmentów do 10 dni o trendzie lekko wzrostowym bądź spadkowym). Przykładowo reguła nr. 7 określa, że 2 segmenty o lekkim trendzie spadkowym (do -2%) oraz jeden o silniejszej tendencji spadkowej (do -10%) poprzedzają segment ponownie o lekkim trendzie spadowym.

Tabela 5: Reguły wyznaczone za pomocą funkcji linearSegmentation. *Źródło:* opracowanie własne

Lp.	Reguła	Wsparcie	Pewność
1	{lag-2=kl_5,lag-1=kl_7} => {lag0=kl_7}	0.01	0.62
2	{lag-2=kl_2,lag-1=kl_3} => {lag0=kl_2}	0.01	0.55
3	{lag-2=kl_2,lag-1=kl_2} => {lag0=kl_2}	0.08	0.57
4	{lag-2=kl_2,lag-1=kl_7} => {lag0=kl_7}	0.06	0.59
5	{lag-3=kl_3,lag-2=kl_7,lag-1=kl_7} => {lag0=kl_7}	0.01	0.83
6	{lag-3=kl_1,lag-2=kl_1,lag-1=kl_7} => {lag0=kl_7}	0.01	0.56
7	{lag-3=kl_7,lag-2=kl_7,lag-1=kl_1} => {lag0=kl_7}	0.01	0.50
8	{lag-3=kl_2,lag-2=kl_2,lag-1=kl_2} => {lag0=kl_2}	0.04	0.53
9	{lag-3=kl_7,lag-2=kl_2,lag-1=kl_2} => {lag0=kl_2}	0.02	0.79
10	{lag-3=kl_2,lag-2=kl_2,lag-1=kl_7} => {lag0=kl_7}	0.02	0.53
11	{lag-3=kl_7,lag-2=kl_2,lag-1=kl_7} => {lag0=kl_7}	0.03	0.80
12	{lag-4=kl_2,lag-3=kl_2,lag-2=kl_2,lag-1=kl_2} => {lag0=kl_2}	0.02	0.53
13	{lag-4=kl_2,lag-3=kl_2,lag-2=kl_2,lag-1=kl_7} => {lag0=kl_7}	0.02	0.91
14	{lag-4=kl_2,lag-3=kl_2,lag-2=kl_7,lag-1=kl_7} => {lag0=kl_7}	0.01	0.50
15	{lag-4=kl_7,lag-3=kl_2,lag-2=kl_2,lag-1=kl_2} => {lag0=kl_2}	0.01	0.55
16	{lag-4=kl_7,lag-3=kl_7,lag-2=kl_2,lag-1=kl_2} => {lag0=kl_2}	0.01	0.75
17	{lag-4=kl_7,lag-3=kl_2,lag-2=kl_7,lag-1=kl_7} => {lag0=kl_7}	0.02	0.58
18	{lag-4=kl_7,lag-3=kl_7,lag-2=kl_2,lag-1=kl_7} => {lag0=kl_7}	0.02	0.89
19	{lag-4=kl_7,lag-3=kl_7,lag-2=kl_7,lag-1=kl_7} => {lag0=kl_7}	0.02	0.53

3.5 Ocena algorytmu

Oceny jakości algorytmu dokonujemy na próbie testowej, wyszukując lewych stron wcześniej wygenerowanych asocjacji, a następnie sprawdzając czy kolejny element pokrywa się z prawą stroną reguły. W kolejnym kroku wyliczany jest współczynnik dokładności, w celu weryfikacji skuteczności przewidywania. Rezultaty dla obu metod przedstawiają tabele 6 oraz 7.

Wyniki wskazują na dużo lepsze wyniki dla asocjacji otrzymanych funkcją $linearSegmentation$. Łącznie, dla wszystkich reguł współczynnik dokładności wynosi 0.5466667, co przy średniej pewności reguł równej 0.6363175 pozwala stwierdzić, że zbiór ten przeszedł pozytywnie sprawdzian krzyżowy.

Dla segmentacji binarnej na pierwszy rzut oka widać niską liczbę pojawienia się reguł na zbiorze testowym. Wynika to z jego małej liczebności (91 segmenty). Aż 8 na 19 reguł nie wystąpiło w próbie, dlatego też współczynnik dokładności dla wszystkich asocjacji wynosił 0.3, podczas gdy średnia pewność dla reguł wynosiła 0.6239613. Na tej podstawie możemy stwierdzić, że wygenerowane asocjacje nie przeszły sprawdzianu krzyżowego.

Tabela 6: Ocena jakości algorytmu dla segmentów wyznaczonych metodą Binary Segmentation. *Źródło:* opracowanie własne

Nr reguły	Prawidłowo zaklasyfikowane	Łączna liczba pojawień	Wspolczynnik dokładności
1	0.00	0.00	
2	0.00	1.00	0.00
3	0.00	0.00	
4	1.00	1.00	1.00
5	1.00	2.00	0.50
6	0.00	5.00	0.00
7	1.00	2.00	0.50
8	0.00	0.00	
9	1.00	3.00	0.33
10	0.00	0.00	
11	0.00	1.00	0.00
12	0.00	0.00	
13	0.00	0.00	
14	1.00	1.00	1.00
15	0.00	1.00	0.00
16	1.00	2.00	0.50
17	0.00	0.00	
18	0.00	0.00	
19	0.00	1.00	0.00

Tabela 7: Ocena jakości algorytmu dla segmentów wyznaczonych za pomocą funkcji linearSegmentation. *Źródło:* opracowanie własne

Nr reguły	Prawidłowo zaklasyfikowane	Łączna liczba pojawień	Wspolczynnik dokładności
1	4.00	5.00	0.80
2	4.00	6.00	0.67
3	13.00	25.00	0.52
4	13.00	20.00	0.65
5	2.00	3.00	0.67
6	0.00	4.00	0.00
7	2.00	10.00	0.20
8	7.00	13.00	0.54
9	4.00	8.00	0.50
10	8.00	11.00	0.73
11	2.00	3.00	0.67
12	3.00	7.00	0.43
13	5.00	6.00	0.83
14	4.00	8.00	0.50
15	2.00	4.00	0.50
16	3.00	5.00	0.60
17	1.00	2.00	0.50
18	2.00	2.00	1.00
19	3.00	8.00	0.38

4 Podsumowanie

Podsumowując rozważania na temat powyższego algorytmu można stwierdzić, że wydobywanie reguł asocjacyjnych z finansowych szeregów czasowych jest procesem wieloetapowym i złożonym. Wymaga ono odpowiedniego przygotowania i obróbki danych, zanim będzie możliwe jakiekolwiek wyciąganie wniosków na przyszłość. Jak jednak wskazują wyniki, zaproponowany powyżej algorytm, użyty do indeksu giełdowego WIG20, daje niezadowalające rezultaty. Mimo, że zbiór obserwacji wydaje się być duży (5235 obserwacji), niemożliwe jest wygenerowanie reguł zawierających większą liczbę segmentów, a przez to łatwiej interpretowalnych i podobnych do formacji znanych z analizy technicznej.

Dla segmentów wyznaczonych za pomocą funkcji dzielącej szereg w miejscach istotnej statystycznie zmiany średniej wyniki są niesatysfakcjonujące, ponieważ zbiór testowy jest zbyt mały, by było możliwe zweryfikowanie czy wyznaczone reguły decyzyjne potrafią poprawnie prognozować na danych nieużytych przy ich generowaniu. Problem może więc leżeć w tym, że funkcja binarnej segmentacji jest zbyt danochłonna i właściwe byłoby jej użycie dla dłuższych szeregów czasowych. Ograniczenie tej metody może wynikać także z niewłaściwego doboru odpowiedniej ilości klastrów, przez co niektóre obserwacje mogą być pomijane stosując algorytm Apriori, ponieważ liczebność danego klastra będzie zbyt mała. Niewątpliwym atutem tego podejścia jest fakt, że powstałe segmenty cechują się różnorodną charakterystyką, mniejszym skupieniem, a przez to prowadzą do powstania bardziej zróżnicowanych reguł decyzyjnych.

Jeśli chodzi o wyniki dla okienek wyznaczonych za pomocą metody liniowej segmentacji, są one poprawne pod względem sprawdzianu krzyżowego, jednak wyznaczone reguły niosą ze sobą małą wartość informacyjną, ze względu na to że ich składowe stanowią praktycznie wyłącznie dwie klasy. Problem stanowi więc tutaj charakterystyka segmentów wyznaczanych za pomocą tej metody. Tak jak wcześniej zostało stwierdzone, większość z nich jest długości równej tej podanej w argumencie funkcji. To powoduje, że algorytm klastrowania wyznacza 2 zbiory, które w przeciwieństwie do pozostałych charakteryzują się obecnością dużej liczby obserwacji, a to w konsekwencji prowadzi do uwzględniania tylko ich stosując algorytm Apriori. Plusem tej metody jest duża liczba

segmentów, pozwalająca na jednoznaczne określenie przydatności prognostycznej asocjacji.

Biorąc więc pod uwagę wszystkie powyższe wnioski, możliwe jest ulepszenie zaproponowanego algorytmu, tak aby otrzymać korzystniejsze rezultaty. Można tego dokonać na etapie segmentacji szeregu, stosując inną metodę, bądź na etapie odpowiedniego podziału segmentów, używając np.: kryterium Calinski-Harabasz do określenia właściwej liczby klastrów. Również zaaplikowanie innego algorytmu do wyznaczania reguł asocjacyjnych (np. algorytm Eclat) mogłoby przynieść pozytywny efekt.

A Dodatek: Tabela współczynników wielomianu kwadratowego dla filtra Savitzky'ego-Golaya

Jako że współczynniki dopasowania wielomianu są symetryczne ($a_i = a_{-i}$), tabela ta zawiera tylko współczynniki dla elementów nieujemnych.

NP	H	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
5	35	17	12	-3	0	0	0	0	0	0	0	0	0	0
7	21	7	6	3	-2	0	0	0	0	0	0	0	0	0
9	231	59	54	39	14	-21	0	0	0	0	0	0	0	0
11	429	89	84	69	44	9	-36	0	0	0	0	0	0	0
13	143	25	24	21	16	9	0	-11	0	0	0	0	0	0
15	1105	167	162	147	122	87	42	-13	-78	0	0	0	0	0
17	323	43	42	39	34	27	18	7	-6	-21	0	0	0	0
19	2261	269	264	249	224	189	144	89	24	-51	-136	0	0	0
21	3059	329	324	309	284	249	204	149	84	9	-76	-171	0	0
23	805	79	78	75	70	63	54	43	30	15	-2	-21	-42	0
25	5175	467	462	447	422	387	343	287	222	147	62	-33	-138	-253

B Dodatek: Kod R

```
# 1. WCZYTYWANIE DANYCH ------------------------------------------------

rm(list=ls())

wig20<-read.csv("wig20.csv",header=T)
attach(wig20)
install.packages("Rcpp")
library(Rcpp)
install.packages("changepoint")
install.packages("quantmod")
install.packages("signal")
install.packages("mFilter")
install.packages("arules")
install.packages("ggplot2")
install.packages("ade4")
install.packages("cluster")
install.packages("fpc")
library(quantmod)
library(changepoint)
library(signal)
library(ifultools)
library(arules)
library(zoo)
library(mFilter)
library(ggplot2)
library(grid)
library(ade4)
library(cluster)
library(fpc)

# 2. CZYSZCZENIE SZEREGU -----------------------------------------------

#FILTR Savitzky-Golay

plotSavitzkyGolay<-function(data,p,n){
  sgol<-sgolayfilt(data,p,n)
```

```
37    plot(sgol,type="l",col="black",xlab="czas",ylab="wartosc",
38         main=paste("Savitzky-Golay␣(", p ,",", n , ")",sep=""))
39  }
40
41  #FILTR Baxter-King
42  plotBaxterKing<-function(data,pl,pu){
43    baxterking<-bkfilter(data,pl,pu)
44    y<-baxterking$trend
45    plot(y,type="l",col="black",xlab="czas",ylab="wartosc",
46         main=paste("Baxter-King␣(", pl ,",", pu , ")",sep=""))
47  }
48
49  #Wykresy podsumowujace wygladzanie
50  par(mfrow=c(2,2))
51  x<-c(seq(1:5523))
52  wig<-as.ts(Zamkniecie)
53  plot(wig,type="l",col="black",main="WIG20",xlab="czas",ylab="wartosc")
54  plotSavitzkyGolay(Zamkniecie,3,25)
55  plotBaxterKing(Zamkniecie,2,100)
56
57  #Wybiera szereg uznany za najlepsze wygladzenie do dalszych operacji i zapisuje
58  #go jako 'wygladzone'
59  #1.Savitzky-Golay
60  #2.Baxter-King
61  wybor<-function(x){
62    ifelse(x==1,wygladzone<-as.numeric(sgol),wygladzone<-as.numeric(y))
63  }
64  wybor(1)
65  rm(sgol,y)
66
67  # 3.1. CHANGEPOINT PACKAGE ----------------------------------------------------
68
69  BinSeg<-cpt.mean(wygladzone,penalty="Hannan-Quinn",method = "BinSeg",Q=2000,
70                   test.stat="CUSUM")
71  SegNeigh<-cpt.mean(wygladzone,penalty="SIC",method = "SegNeigh",Q=600,
72                     test.stat="CUSUM")
73  PELT<-cpt.mean(wygladzone,method = "PELT", penalty = "Manual",
74                 pen.value = "n^(3/2)")
```

```
75 #changepoint'y
76 segmentyBINSEG<-cpts(BinSeg)
77
78 ks.test(wygladzone,y='pnorm',alternative='two.sided')
79
80 # 3.2. IFULTOOLS PACKAGE ------------------------------------------------
81
82 #funkcja linearSegmentation pozwala na segment szeregu czasowego na segmenty o
83 #wybranej szerokosci, nastepnie laczy te, gdzie kat pomiedzy segmentami jest
84 #mniejszy niz angle.tolerance
85
86 plotLinearSegmentation<-function(x,data,p,q,aspect=T){
87   segmenty<-as.numeric(linearSegmentation(x,data,n.fit=p,angle.tolerance=q,
88                                           aspect=T))
89   segmenty[length(segmenty)+1]=length(data)
90   segmenty1<-c(rep(NA,length(segmenty)+1))
91   segmenty1[2:length(segmenty1)]<-segmenty
92   segmenty1[1]=1
93   segmenty<-segmenty1
94   rm(segmenty1)
95   wartosci<-data[segmenty]
96   plot(segmenty,wartosci,type="l",xlab="czas",ylab="wartosc",
97        main=paste("linearSegmentation␣(", p ,",", q , ")␣\n",
98                   length(segmenty),"␣segmenty",sep=""))
99   segmentyLS<<-segmenty
100 }
101 x<-c(1:5235)
102 plotLinearSegmentation(x,wygladzone,5,0.1)
103
104 # 3.3. WLASNA FUNKCJA SEGMENTACJI ---------------------------------------
105
106
107 #laczy segmenty o podobnych trendach dla szeregu, ktory juz zostal wczesniej
108 #podzielony
109 MergeSimilarTrends<-function(szereg,changepoints,threshold){
110   changepoints[length(changepoints)+1]<-length(szereg)
111   changepoints1<-c(rep(NA,length(changepoints)+1))
112   changepoints1[2:length(changepoints1)]<-changepoints
```

```
113   changepoints1[1]<-1
114   changepoints<-changepoints1
115
116   df<-data.frame(changepoint=changepoints,close=szereg[changepoints])
117
118   trend<-rep(NA,length(changepoints))
119   trend[1]=((df[1,2]-szereg[1])/szereg[1])*100
120   for (i in 2:length(changepoints)){
121     trend[i]=(((df[i,2]-df[i-1,2])/df[i-1,2])*100)
122   }
123   df<-data.frame(df,trend)
124
125   similar<-rep(NA,length(changepoints))
126   similar[1]='F'
127   for (i in 2:length(changepoints)){
128     ifelse(abs(df[i,3]-df[i-1,3])<=threshold,similar[i]<-'T',similar[i]<-'F')
129   }
130   df<-data.frame(df,similar)
131   df$similar<-as.character(similar)
132
133   rows_to_delete<-which(grepl('T',df$similar))-1
134   df1<-df[-c(rows_to_delete),-c(3,4)]
135
136   trend1<-rep(NA,length(df1[,1]))
137   trend1[1]=((df1[1,2]-szereg[1])/szereg[1])*100
138   for (i in 2:length(df1[,1])){
139     trend1[i]=(((df1[i,2]-df1[i-1,2])/df1[i-1,2])*100)
140   }
141   df1<-data.frame(df1,Trend=trend1)
142
143   DF<<-df1
144   segmenty<<-df1$changepoint
145 }
146
147 #porownanie graficzne zastosowania Binary Segmentation dla roznych parametrow
148 #z pierwotnym wyrownanym szeregiem
149 par(mfrow=c(2,2))
150 #x<-c(1:5523)
```

```
151 plot(x,wygladzone,type="l")
152 MergeSimilarTrends(wygladzone,segmenty,0.5)
153 segmentyLSost<-segmenty
154 plot(DF[,1],DF[,2],type="l",xlab="czas",
155      ylab="wartosc",main=paste("MST-LinearSegmentation, 0.5 \n",
156                                length(segmenty)," segmenty",sep=""))
157 DF_LS<-DF
158
159 MergeSimilarTrends(wygladzone,segmenty,0.5)
160 segmentyBINSEGost<-segmenty
161 plot(DF[,1],DF[,2],type="l",xlab="czas",
162      ylab="wartosc",main=paste("MST-BinSeg, 0.5 \n",length(segmenty),
163                                " segmenty",sep=""))
164 DF_BINSEG<-DF
165
166 rm(DF,segmenty,segmentyBINSEG,segmentyLS,x)
167 #Z FUNKCJI MergeSimilarTrends DOSTAJEMY DATA FRAME- DF (zawierajaca szczegolowe
168 #dane, otrzymane po merge'u segmentow o podobnych trendach otrzymanych z funkcji
169 #cpt.Mean) oraz osobny wektor 'segmenty', zawierajacy nowe changepoint'y
170
171 # 4. SYMBOLICZNA REPREZENTACJA SEGMENTOW ---------------------------------
172
173 #DODANIE CZASU TRWANIA
174 DF<-DF_LS
175 DF<-DF_BINSEG
176
177 Duration<-c(rep(NA,length(DF[,1])))
178 Duration[1]=0
179 for (i in 2:length(DF[,1])){
180   Duration[i]<-DF[i,1]-DF[i-1,1]
181 }
182 DF<-data.frame(DF,Duration)
183 #Statystyki czasu trwania
184 summary(DF[,4])
185 hist(DF[,4],xlim=c(0,30),freq=T,breaks=500,density=50,col="black")
186
187 #Statystyki trendu
188 summary(DF[,3])
```

```
189 hist(DF[,3],xlim=c(-12,20),freq=T,breaks=100,density=50,col="black")
190
191 #funkcja BinsTrend do wyznaczania ile procentowo znajduje sie segmentow w
192 #zadanych przedzialach (x1-x4)
193 BinsTrend<-function(data,kolumna,x1,x2,x3,x4){
194   range1<-((nrow(data[which(data[,kolumna] <= x1),])) /
195              length(data[,kolumna]) * 100)
196   range2<-(nrow(data[which(data[,kolumna] > x1 & data[,kolumna] <= x2) ,])/
197              length(data[,kolumna]) * 100)
198   range3<-(nrow(data[which(data[,kolumna] > x2 & data[,kolumna] <= x3) ,])/
199              length(data[,kolumna]) * 100)
200   range4<-(nrow(data[which(data[,kolumna] > x3 & data[,kolumna] <= x4) ,])/
201              length(data[,kolumna]) * 100)
202   range5<-((nrow(data[which(data[,kolumna] > x4),])) /
203              length(data[,kolumna]) * 100)
204   w<-c(range1,range2,range3,range4,range5)
205   return(w)
206 }
207 #dwie propozycje dobrania przedzialow trendu
208 BinsTrend(DF_BINSEG,3,-4,-1,1,5)
209 BinsTrend(DF,3,-5,-2,2,5)
210
211 #funkcja BinsTime analogiczna to poprzedniej, tylko ze dla przedzialow czasowych
212 BinsTime<-function(data,kolumna,x1,x2){
213   range1<-((nrow(data[which(data[,kolumna] <= x1),])) /
214              length(data[,kolumna]) * 100)
215   range2<-(nrow(data[which(data[,kolumna] > x1 & data[,kolumna] <= x2) ,])/
216              length(data[,kolumna]) * 100)
217   range3<-((nrow(data[which(data[,kolumna] > x2),])) /
218              length(data[,kolumna]) * 100)
219   w<-c(range1,range2,range3)
220   return(w)
221 }
222 #dwie propozycje dobrania przedzialow czasu
223 BinsTime(DF_BINSEG,4,5,20)
224 BinsTime(DF,4,5,10)
225
226 DF_BINSEG<-DF
```

```
DF_LS<-DF
rm(DF)

# 4.1. Klastrowanie BINSEG ------------------------------------------------

df1<-data.frame(DF_BINSEG$Duration,DF_BINSEG$Trend)
kluster<-kmeans(df1,9)

plot(df1,pch=16,cex=0.5,main="Klastrowanie - Binary Segmentation",
     xlab="Czas trwania",ylab="Trend")
kmeansRes<-factor(kluster$cluster)
s.class(df1,fac=kmeansRes, add.plot=TRUE, col=rainbow(nlevels(kmeansRes)))

DF_BINSEG$cluster <- kluster$cluster
for(i in seq_along(DF_BINSEG[,1])){
  DF_BINSEG[i,5]<-paste("kl_",DF_BINSEG[i,5],sep="")
}
#data frame odpowiedni dla algorytmu apriori
DF_BINSEG[,5]<-as.factor(DF_BINSEG[,5])
nr_segmentu<-c(1:length(DF_BINSEG[,1]))
DF_BINSEG1<-data.frame(nr_segmentu,klasa=DF_BINSEG[,5])

#PODZIAl NA ZBIOR TRENINGOWY I TESTOWY

podzial<-function(dane,procent){
  zmiana<-round(procent*nrow(dane))
  DF_BINSEG_trening<<-dane[1:zmiana,]
  DF_BINSEG_test<<-dane[(zmiana+1):nrow(dane),]

}
podzial(DF_BINSEG1,0.8)

# 4.2. Klastrowanie LS ----------------------------------------------------

df1<-data.frame(DF_LS$Duration,DF_LS$Trend)
kluster<-kmeans(df1,7)

plot(df1,pch=16,cex=0.5,main="Klastrowanie - Linear Segmentation",
```

```
265       xlab="Czas trwania",ylab="Trend",
266       ylim=c(-20,40), xlim=c(0,50))
267 kmeansRes<-factor(kluster$cluster)
268 s.class(df1,fac=kmeansRes, add.plot=TRUE, col=rainbow(nlevels(kmeansRes)))
269
270 DF_LS$cluster <- kluster$cluster
271 for(i in seq_along(DF_LS[,1])){
272   DF_LS[i,5]<-paste("kl_",DF_LS[i,5],sep="")
273 }
274 #data frame odpowiedni dla algorytmu apriori
275 DF_LS[,5]<-as.factor(DF_LS[,5])
276 nr_segmentu<-c(1:length(DF_LS[,1]))
277 DF_LS1<-data.frame(nr_segmentu,klasa=DF_LS[,5])
278
279 #PODZIAl NA ZBIOR TRENINGOWY I TESTOWY
280
281 podzial<-function(dane,procent){
282   zmiana<-round(procent*nrow(dane))
283   DF_LS_trening<<-dane[1:zmiana,]
284   DF_LS_test<<-dane[(zmiana+1):nrow(dane),]
285
286 }
287 podzial(DF_LS1,0.66)
288
289 # 5. WYZNACZANIE REGUL ASOCJACYJNYCH -------------------------------------
290
291 z <- read.zoo(DF_LS_trening, header = TRUE, FUN = identity)
292 lags <- as.data.frame(lag(z[,2], -4:0))
293 lags[,1]<-as.factor(lags[,1])
294 lags[,2]<-as.factor(lags[,2])
295 lags[,3]<-as.factor(lags[,3])
296 lags[,4]<-as.factor(lags[,4])
297 lags[,5]<-as.factor(lags[,5])
298 lags[,6]<-as.factor(lags[,6])
299
300 pojawienie<-5
301 a <- apriori(lags,parameter = list( supp=pojawienie/nrow(lags),
302                                     conf=0.5,minlen=3))
```

```
303 c<-subset(a, subset = rhs %pin% "lag0=")
304 inspect(c)
305
306 reguly_BINSEG<-as(c, "data.frame")
307 reguly_LS<-as(c,"data.frame")
308
309 # 6. OCENA ALGORYTMU ----------------------------------------------------
310
311 #dla regul LS
312 ocena_regul<-function(dane,lhs,e1,e2,e3,e4,e5){
313   if (lhs==2){
314     z <- read.zoo(dane, header = TRUE, FUN = identity)
315     lags <- as.data.frame(lag(z[,2], -lhs:0))
316     lags[,1]<-as.factor(lags[,1])
317     lags[,2]<-as.factor(lags[,2])
318     lags[,3]<-as.factor(lags[,3])
319
320     lags_all<-lags[lags[,1]==paste("kl_",e3,sep="") &
321                      lags[,2]==paste("kl_",e4,sep=""),]
322     lags_all<-na.omit(lags_all)
323     zliczenie<-nrow(lags_all)
324     poprawne<-nrow(lags_all[lags_all[,3]==paste("kl_",e5,sep=""),])
325     wspolczynnik<-poprawne/zliczenie
326     wynik<-data.frame(poprawne,zliczenie,wspolczynnik)
327     return(wynik)
328
329   } else if (lhs==3) {
330
331     z <- read.zoo(dane, header = TRUE, FUN = identity)
332     lags <- as.data.frame(lag(z[,2], -lhs:0))
333     lags[,1]<-as.factor(lags[,1])
334     lags[,2]<-as.factor(lags[,2])
335     lags[,3]<-as.factor(lags[,3])
336     lags[,4]<-as.factor(lags[,4])
337
338     lags_all<-lags[lags[,1]==paste("kl_",e2,sep="") &
339                      lags[,2]==paste("kl_",e3,sep="") &
340                      lags[,3]==paste("kl_",e4,sep=""),]
```

```
341      lags_all<-na.omit(lags_all)
342      zliczenie<-nrow(lags_all)
343      poprawne<-nrow(lags_all[lags_all[,4]==paste("kl_",e5,sep=""),])
344      wspolczynnik<-poprawne/zliczenie
345      wynik<-data.frame(poprawne,zliczenie,wspolczynnik)
346      return(wynik)
347
348    } else if (lhs==4){
349
350      z <- read.zoo(dane, header = TRUE, FUN = identity)
351      lags <- as.data.frame(lag(z[,2], -lhs:0))
352      lags[,1]<-as.factor(lags[,1])
353      lags[,2]<-as.factor(lags[,2])
354      lags[,3]<-as.factor(lags[,3])
355      lags[,4]<-as.factor(lags[,4])
356      lags[,5]<-as.factor(lags[,5])
357
358      lags_all<-lags[lags[,1]==paste("kl_",e1,sep="") &
359                       lags[,2]==paste("kl_",e2,sep="") &
360                       lags[,3]==paste("kl_",e3,sep="") &
361                       lags[,4]==paste("kl_",e4,sep=""),]
362      lags_all<-na.omit(lags_all)
363      zliczenie<-nrow(lags_all)
364      poprawne<-nrow(lags_all[lags_all[,5]==paste("kl_",e5,sep=""),])
365      wspolczynnik<-poprawne/zliczenie
366      wynik<<-data.frame(poprawne,zliczenie,wspolczynnik)
367      return(wynik)
368
369    }
370  }
371
372  reg<-matrix(c(0,0,5,7,7,
373                0,0,2,3,2,
374                0,0,2,2,2,
375                0,0,2,7,7,
376                0,3,7,7,7,
377                0,1,1,7,7,
378                0,7,7,1,7,
```

```
                0,2,2,2,2,
                0,7,2,2,2,
                0,2,2,7,7,
                0,7,2,7,7,
                2,2,2,2,2,
                2,2,2,7,7,
                2,2,7,7,7,
                7,2,2,2,2,
                7,7,2,2,2,
                7,2,7,7,7,
                7,7,2,7,7,
                7,7,7,7,7),ncol=5,byrow=T)


Ocena_LS<-matrix(rep(NA,57),nrow=19)
for (i in 1:nrow(reg)){
  wynik<-ocena_regul(DF_LS_test,5-sum(reg[i,]==0)-1,reg[i,1],reg[i,2],
                     reg[i,3],reg[i,4],reg[i,5])
  Ocena_LS[i,1]<-wynik[1,1]
  Ocena_LS[i,2]<-wynik[1,2]
  Ocena_LS[i,3]<-wynik[1,3]
}
colnames(Ocena_LS)<-c("Prawidlowo␣zaklasyfikowane","laczna␣liczba␣pojawien",
                      "Wspolczynnik␣dokladnosci")

#dla regul BINSEG

reg<-matrix(c(0,0,2,6,6,
                0,0,4,1,7,
                0,0,4,8,4,
                0,0,8,4,4,
                0,0,9,6,5,
                0,0,5,5,4,
                0,0,5,4,4,
                0,0,4,6,6,
                0,0,4,4,4,
                0,4,8,4,4,
                0,9,6,5,4,
```

```
                0,6,9,6,6,
                0,5,5,4,4,
                0,5,4,4,4,
                0,6,5,5,4,
                0,4,5,6,6,
                0,4,5,4,4,
                0,4,4,6,6,
                0,4,4,4,4),ncol=5,byrow=T)


Ocena_BINSEG<-matrix(rep(NA,57),nrow=19)
for (i in 1:nrow(reg)){
  wynik<-ocena_regul(DF_BINSEG_test,5-sum(reg[i,]==0)-1,reg[i,1],reg[i,2],
                     reg[i,3],reg[i,4],reg[i,5])
  Ocena_BINSEG[i,1]<-wynik[1,1]
  Ocena_BINSEG[i,2]<-wynik[1,2]
  Ocena_BINSEG[i,3]<-wynik[1,3]
}
colnames(Ocena_BINSEG)<-c("Prawidlowo␣zaklasyfikowane","laczna␣liczba␣pojawien"
                          ,"Wspolczynnik␣dokladnosci")
```

Literatura

Akkaya, G. C. i Uzar, C. (2011), 'Data Mining in Financial Application', *Journal of Modern Accounting and Auditing* **7**(12), 1362–1367.

Aragianni, S. T. K. i Fetsos, T. H. S. (2010), 'Extracting Formations From Long Financial Time Series Using Data Mining', **53**, 273–293.

Biecek, Przemysław Trajkowski, K. (2011), *Na przełaj przez Data Mining*.
URL: *http://www.biecek.pl/NaPrzelajPrzezDataMining*

Bulkowski, T. N. i Wiley, J. (2005), *Encyclopedia of Chart Patterns*.

Constantine, W. i Percival, D. (2014), *ifultools: Insightful Research Tools*. R package version 2.0-1.
URL: *http://CRAN.R-project.org/package=ifultools*

Dante, C., De Pison Francisco J, M. i Alpha, P. (2010), 'Finding temporal associative rules in financial time-series: A case of study in Madrid Stock Exchange (IGBM)', pp. 60–68.

Gandhmal, D. P. (2011), 'An Optimized Approach to Analyze Stock market using Data Mining Technique', *International Journal of Computer Applications* pp. 38–42.

Huk, M. (2001), Wygładzanie i filtrowanie danych z przeznaczeniem do interpretacji widm spektroskopowych.

Kaastra, I. i Boyd, M. (1996), 'Designing a neural network for forecasting financial and economic time series'.

Khan, A., Baharudin, B. i Khan, K. (2011), 'Mining Customer Data for Decision Making Using New Hybrid Classification Algorithm', *Journal of Theoretical and Applied Information Technology* .

Killick, R., Eckley, I. i Haynes, K. (2014), *changepoint: An R package for changepoint analysis*. R package version 1.1.5.
URL: *http://CRAN.R-project.org/package=changepoint*

Kufel, T. (2005), ‘Narzędzia ekonometrii dynamicznej w oprogramowaniu GRETL’, (2002), 6–8.

Murphy, J. J. (1999), *John J Murphy - Technical Analysis Of The Financial Markets.pdf*, Vol. 77.
URL: *http://www.ncbi.nlm.nih.gov/pubmed/20599625*

Virili, F. i Freisleben, B. (2001), ‘Neural Network Model Selection Financial Time Series Prediction Anders-Korn model selection strategies’, *Computing* (200 I).

Weigend, A. S. (1996), ‘Data Mining in Finance Report from the Post-NNCM-96 Workshop on Teaching Computer Intensive Methods for Financial Modeling and Data Analysis’, *Decision Technologies for Financial Engineering* pp. 399–412.
URL: *www.stern.nyu.edu/ aweigend*

Zhang, D. i Zhou, L. (2004), ‘Discovering golden nuggets: Data mining in financial application’, *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* **34**(4), 513–522.

Spis tablic

Spis rysunków

Streszczenie

Niniejsza praca przedstawia własny algorytm do wyznaczania reguł asocjacyjnych z finansowych szeregów czasowych, pozwalający na prognozowanie przyszłych ruchów cen. Algorytm został zastosowany dla indeksu giełdowego WIG20. Zawiera narzędzia statystyczne oraz używane w data mining'u i składa się z 5 kroków. Pierwszy krok stanowi czyszczenie danych z nadmiaru krótkoterminowych wahań i używa do tego celu filtr Savitzky'ego-Golaya. Drugim krokiem jest segmentacja szeregu na fragmenty o podobnej charakterystyce. Od tego momentu algorytm jednocześnie przeprowadzany jest dla dwóch metod (segmentacja liniowa oraz binarna), w celu porównania ich działania. Trzecim krokiem jest zbudowanie słownika, tzn. przypisanie segmentów o podobnej charakterystyce do odpowiednich klas. Czwarty etap to zastosowanie algorytmu Apriori do wyznaczenia reguł asocjacyjnych, natomiast ostatni to ocena ich własności prognostycznej, dzięki podziałowi zbioru na próbę treningową i testową. Wyniki powyższego schematu postępowania okazują się być niezadowalające, dlatego też sugerowana jest odpowiednia modyfikacja algorytmu.