Podstawy Sztucznej Inteligencji

Sprawozdanie do zadania domowego

Przykład zastosowania logiki rozmytej do zamawiania piwa w sklepach detalicznych

Stanisław Denkowski

Nr indeksu 305288

22 listopada 2020

Omówienie przykładu sterownika rozmytego

Celem zadania było zaimplementowanie sterownika bazującego na regułach logiki rozmytej dla 3-4 zmiennych wejściowych i 1 zmiennej wyjściowej.

Zdecydowałem się na opracowanie sterownika, który w swoim założeniu ma wspomagać planowanie wielkości zamówień uzupełniających poziom zapasu piwa w sklepie detalicznym.

Każdego dnia osoba odpowiedzialna za planowanie zapasu w sklepach musi podjąć decyzję, jaką ilość piwa należy zamówić, żeby z jednej strony go nie zabrakło kolejnego dnia (co powodowałoby tzw. sprzedaż utraconą i finansowe straty sklepu), zaś z drugiej strony, żeby dostawa nie przepełniła magazynu sklepowego (zbyt duży zapas oznacza również ryzyko jego przeterminowania i tym samym strat).

Wejścia i wyjście sterownika

Zmienną wyjściową sterownika jest - Buy - ilość piwa do zamówienia w celu uzupełnienia zapasu w sklepie. Na końcu obliczeń celowo pomijam kwestię zaokrąglenia do pełnych skrzynek, aby dokładniej analizować wyniki.

Dla opisania reguł przyjąłem następujące wartości **zmiennej wyjściowej Buy**:

- Low - mała ilość od 0 do 50 jm (jednostek miary - np. litrów)
- Medium - średnia ilość od 25 do 75 jm
- High - duża ilość od 50 do 100 jm

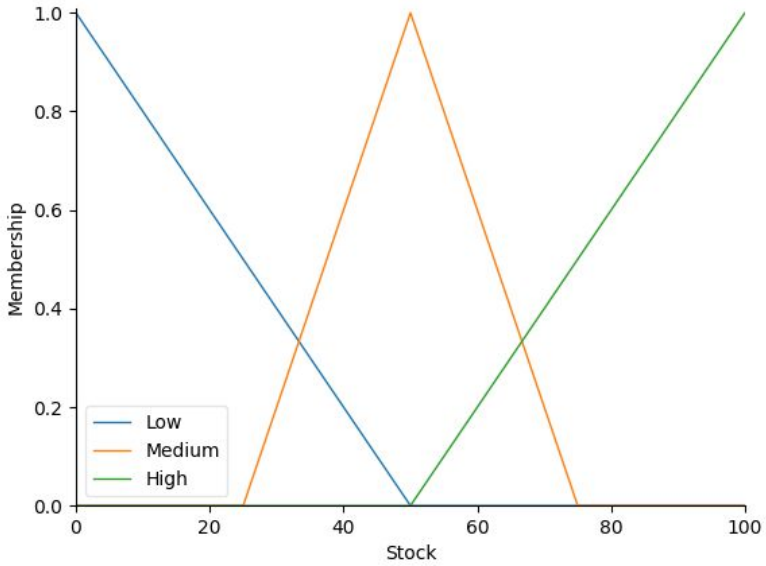
Aby odpowiedzieć na pytanie - jaką ilość należy zamówić - planista musi wziąć pod uwagę szereg różnych zmiennych wejściowych, spośród których wybrałem 3 najistotniejsze. Poniżej opisuję **3 wybrane zmienne wejściowe**, jak również wartości, które będą przyjmować:

- **Stock** - aktualny zapas piwa na stanie sklepu (mały zapas - duży zakup)
    - Low - niski zapas od 0 do 50 jm (litrów)
    - Medium - średni zapas od 25 do 75 jm
    - High - wysoki zapas od 50 do 100 jm

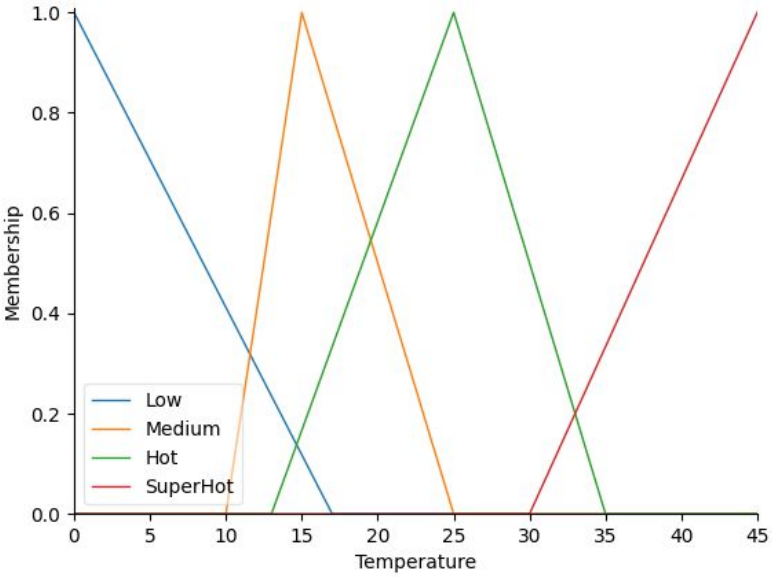
- **Temp** - prognozowana temperatura zewnętrzna (będzie cieplej - większy popyt)
    - Low - niska temperatura od 0 (lub mniej) do 17 stopni Celsjusza
    - Medium - średnia temperatura od 10 do 25 stopni Celsjusza
    - Hot - bardzo ciepło od 13 do 35 stopni Celsjusza
    - SuperHot - upalnie od 30 do 45 stopni Celsjusza lub więcej

- **Weekend** - dzień tygodnia mówi, czy zbliża się weekend (wzrost popytu na piwo)
    - Yes - oznacza, że zaczynamy robić zakupy weekendowe (dni 3-5)
    - No - oznacza, że do weekendu daleko i nie ma on wpływu na zakupy (dni 0-3-4)
    - Przyjąłem, że tydzień roboczy ma 6 dni - w niedzielę sklepy są zamknięte. Dni tygodnia numeruję od 0(poniedziałek).

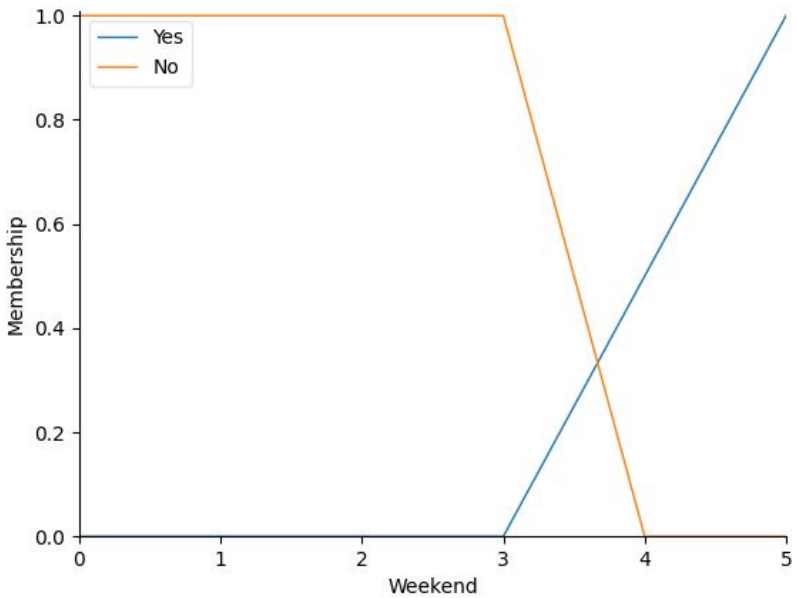
Opisane wyżej 3 zmienne wejściowe i 1 zmienną wyjściową opisałem zobrazowanymi poniżej rozkładami przynależności do poszczególnych zbiorów.



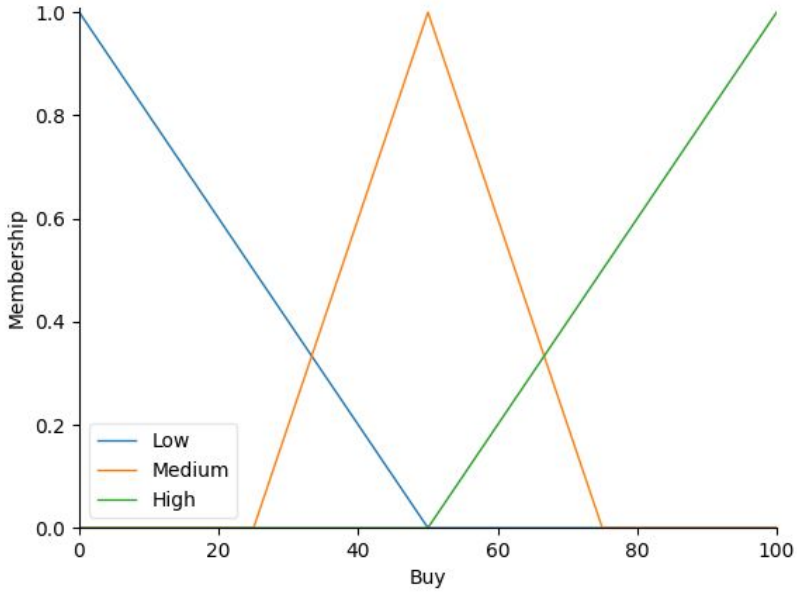
Rysunek 1 - bieżący zapas piwa na sklepie (na koniec dnia)



Rysunek 2 - temperatura (prognozowana na następny dzień)



Rysunek 3 - (następny) dzień tygodnia. Piątek uznaję już jako weekend, a niedzielę bez handlu.



Rysunek 4 - zmienna wyjściowa - ilość piwa do zakupu

Zaimplementowane reguły

Zgodnie z założeniami zadania opracowałem niektóre reguły tak, aby nie było potrzeby definicji reguł dla każdej kombinacji wartości zmiennych wejściowych (wówczas reguł byłoby 3*4*2=24). Cały zbiór wartości wejściowych jest jednak pokryty. Wprowadzone są następujące reguły dla zmiennej wyjściowej **Buy**:

Zmienna **Buy** dla zbliżającego się weekendu (**Weekend=Yes**)

Weekend shopping	High stock	Medium stock	Low stock
Low temp	Low order	Medium order	
Medium temp	Medium order	High order	
Hot temp	High order		
SuperHot temp	High order		

Zmienna **Buy** dla pozostałych dni tygodnia (**Weekend=No**)

Week shopping	High stock	Medium stock	Low stock
Low temp	Low order		Medium order
Medium temp	Low order	Medium order	
Hot temp	Medium order		
SuperHot temp	High order		

Uruchomienie programu

Kod mojego przykładu znajduje się w pliku Pythona (wykorzystuje python3, więc czasem może być potrzeba zastąpić wywołanie “**python …”** na “**python3 …**”) pod nazwą **stock-sci.py**. Program składa się z trzech głównych części: definicji zmiennych, definicji reguł i testów.

Program bazuje na bibliotece do logiki rozmytej **skfuzzy**:
https://github.com/scikit-fuzzy/scikit-fuzzy
Dodatkowo wykorzystywane są znane biblioteki Pythona:

- **numpy**
- **matplotlib**
- **pandas**

Wszystkie te biblioteki te można zainstalować standardową komendą:
> **pip install nazwa_biblioteki**

Po instalacji powyższych bibliotek uruchomienie mojego programu polega na podaniu komendy:
> python stock-sci.py const_val series1_val series2_val series3_val const_val_type
Np. dla ustalonej stałej Stock = 30 i serii temperatury 12, 15 i 21:
> python stock-sci.py 30 12 15 21 S
Uruchomienie programu bez parametrów uruchamia krótki opis wywołania programu.

NOWA WERSJA (Jedyne zmiany w stosunku do poprzedniej wersji, w tym pliku, to ten akapit):

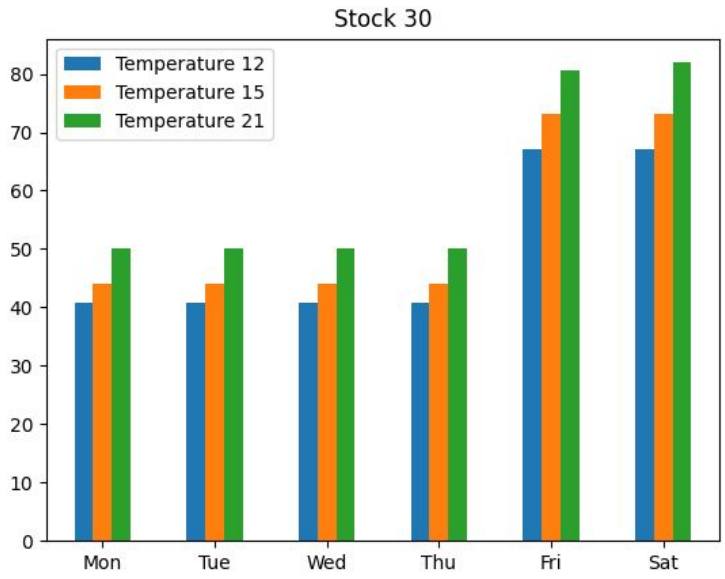
Zostawiłem na platformie upel poprzednią wersję programu, aby w razie nieporozumienia (jakbym ja źle zrozumiał termin i okazało się, że termin wysyłania zadania był wcześniejszy niż myślałem) było widać kiedy zostało oddane i jak się różni. Pokazuje, że nie zrobiłem tego na ostatnią chwilę i nie nie zdążyłem z oddaniem, tylko w ostatnim momencie uświadomiłem sobie możliwą drobną poprawkę.

Dodałem plik źródłowy stock.py, który jest modyfikacją stock-sci.py. Pierwotna wersja programu była wykorzystywana do tworzenia wykresów wykorzystanych w sprawozdaniu (w nowym archiwum pozostała niezmieniona, względem starej wersji), zmodyfikowana wersja ma za zadanie odpowiadać konkretną wartością na zapytania składające się z konkretnych zmiennych.

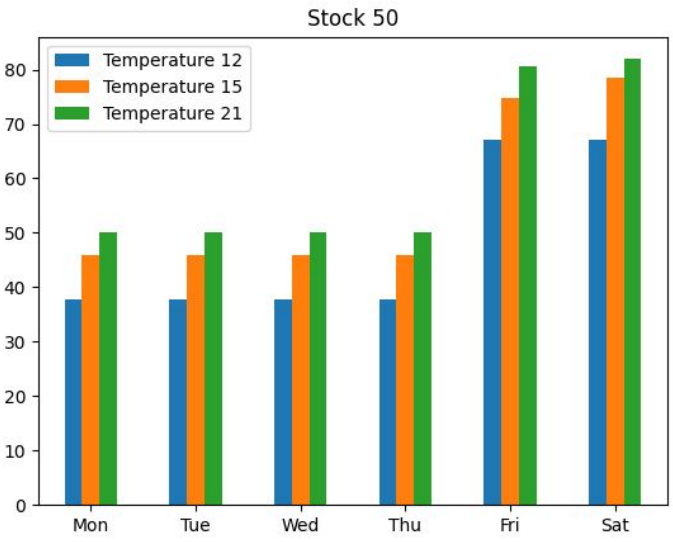
Zapytania powinny być postaci:
> python stock.py weekday current_stock temperature
Np.
> python stock.py SA 40 20
Uruchomienie programu bez parametrów lub ze złymi parametrami uruchamia krótki opis wywołania programu.
Weekday oczekuje MD dla Monday, TU dla Tuesday, WE dla Wednesday, TH dla Thursday, FR dla Friday lub SA dla Saturday.
Funkcja wyliczająca wynik była już wcześniej zaimplementowana, jednak nie przyszło mi do głowy by ją opakować i uznać za najważniejszą w całym programie, dlatego wcześniej tak skupiłem się na funkcjach tworzących wykresy.

Przykładowe symulacje

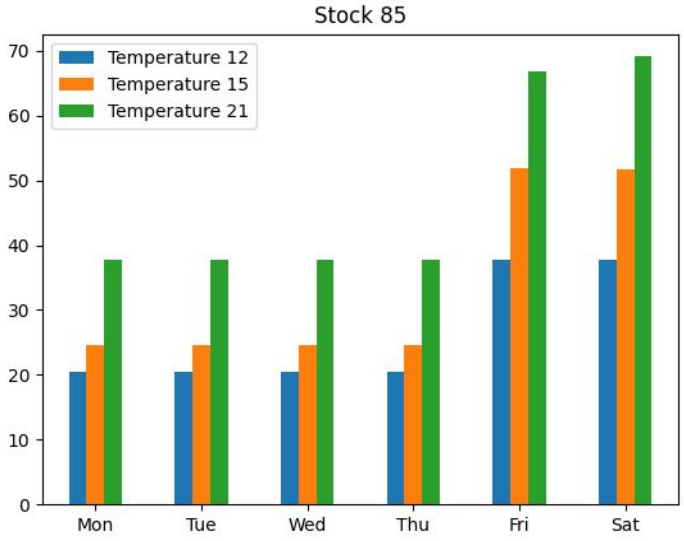
Dla wybranych zestawów parametrów przeprowadziłem symulacje, żeby ocenić działanie programu. Na osi Y widzimy wielkość zamówienia. Poniżej prezentuję graficznie wyniki.



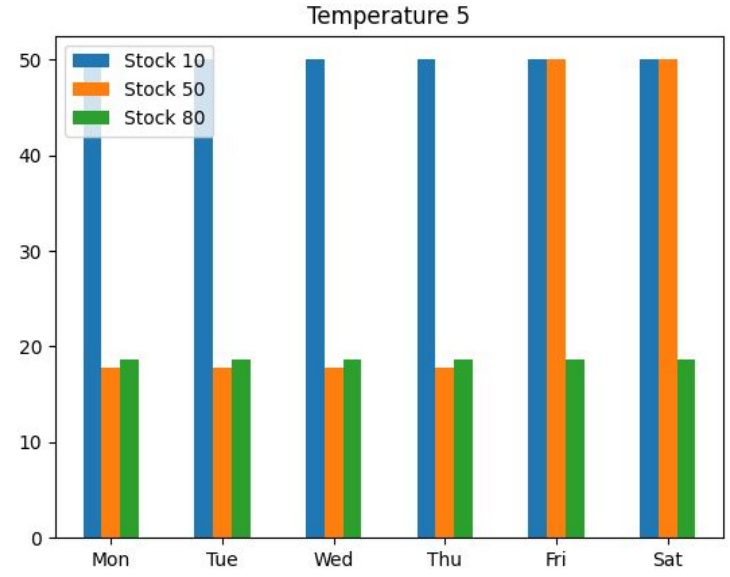
Rysunek 5 - Dla niskiego zapasu 30 obserwujemy wzrost zamówień w weekend. Zamówienia na piątek i sobotę są praktycznie takie same. Wzrost zamówień następuje również ze wzrostem temperatury.



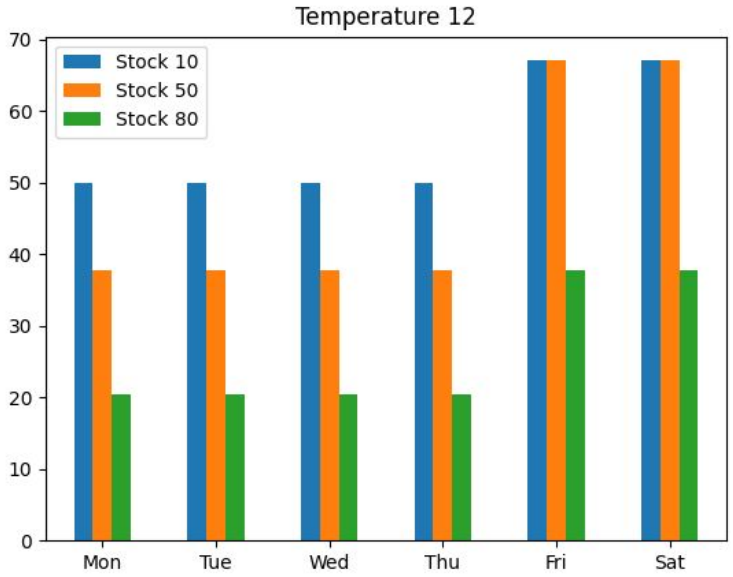
Rysunek 6 - Podobnie jak poprzednio, ale dla średniego zapasu 50. Widzimy lekki wzrost zamówień, a także lekkie zróżnicowanie zamówień piątek i sobota dla wyższych temperatur.



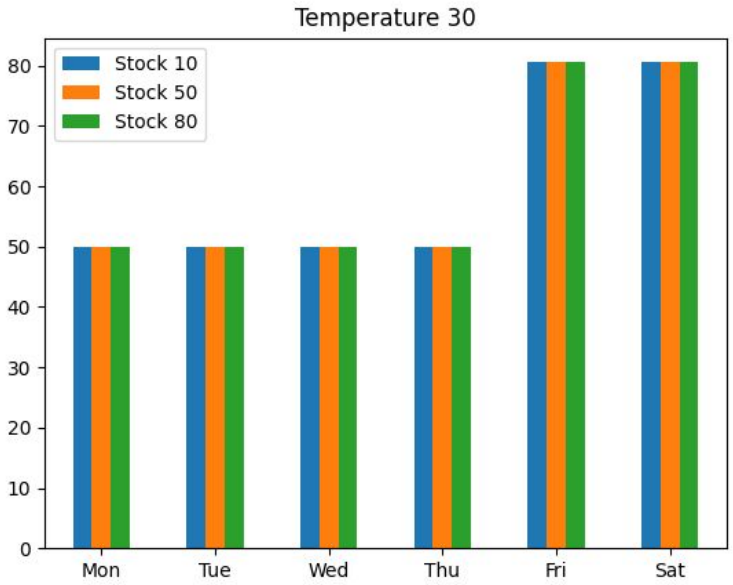
Rysunek 7 - Podobnie jak poprzednio, ale dla wysokiego zapasu 85. Widać rosnącą zależność od temperatury (rozrzut), żeby nie przepełnić zajętych magazynów (niska temperatura = małe zamówienia).



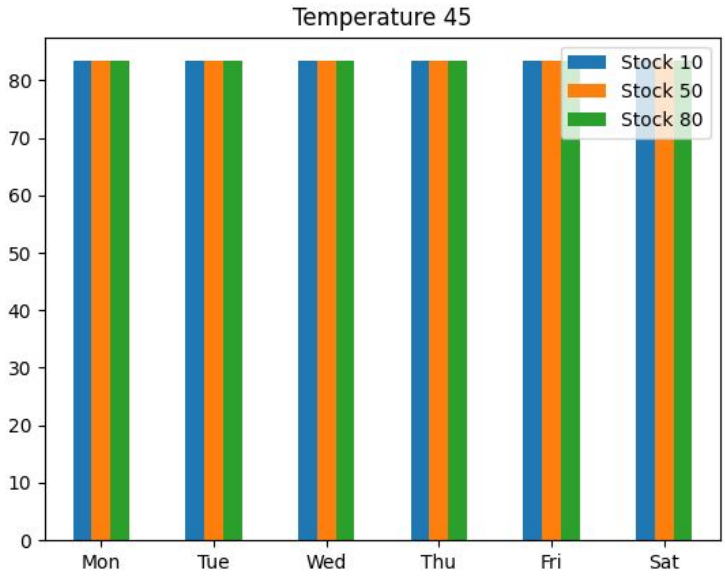
Rysunek 8 - Rysunek dla stałej, niskiej temperatury 5. Widzimy zależność od wielkości zapasu. Wysoki zapas, to stałe, minimalne zamówienia. Dla średniego małego zapasu widać zróżnicowanie wg dni tygodnia. Dla zapasu małego - stałe, nieco większe zamówienia.



Rysunek 9 - Dla stałej, nieco wyższej temperatury 12 widoczne są wzrosty weekendowe niezależnie od zapasu.



Rysunek 10 - DLa wysokiej temperatury zamówienia się stabilizują niezależnie od zapasu (planowana duża rotacja) i wyraźnie rosną w weekend.



Rysunek 11 - Dla prawdziwych upałów - zawsze zamawiane jest możliwie najwięcej. Inne parametry nie mają już znaczenia.

Wnioski końcowe

Głównym wnioskiem z przeprowadzonego ćwiczenia jest to, że opracowanie sterownika było dosyć łatwe, zaś słowne opisanie reguł eksperckich (gdy jest ciepło, kupuj więcej) znacząco zwiększa czytelność modelu i ułatwia mapowanie wiedzy na eksperckiej na reguły sterownika.

W przykładzie pominąłem liczne inne zmienne wejściowe, takie jak np. - powiązanie zamówień z możliwościami systemu transportowego, płynnością zasobów finansowych, upustami związanymi z większymi zamówieniami, zwiększoną sprzedażą piwa przed wydarzeniami sportowymi czy długimi weekendami, itp.

W praktyce podobne zadania można rozwiązywać szeregiem innych metod - chociażby regresją wieloraką czy modelami opartymi o sieci neuronowe. Ciekawe byłoby porównanie efektywności tych różnych metod planowania zapasu.