Wstęp

Cel i zakres pracy

Przegląd literatury i analiza rozwiązań

Rozdział ten stanowi techniczne wprowadzenie do zagadnienia, oparte na analizie literatury i istniejących rozwiązań inżynierskich. Analiza literatury i istniejących rozwiązań stanowi istotny etap procesu projektowego, umożliwiający lepsze zrozumienie kontekstu danego problemu oraz identyfikację potencjalnych obszarów doskonalenia. Rozdział skupia się na przeglądzie literatury związanej z tematyką pracy inżynierskiej oraz analizie istniejących rozwiązań, mającej na celu dostarczenie solidnej podstawy teoretycznej i technologicznej dla dalszych etapów badawczych. W tym rozdziale zostaną dogłębnie poruszone teoretyczne kwestie związane z tematem pracy.

Strumienie danych

Strumieniem danych nazywamy uporządkowany zestaw danych, gdzie każda wartość jest przypisana do określonego momentu czasowego.

Strumień składa się z punktów danych, najczęściej zbieranych w regularnych odstępach czasowych, co pozwala na dokładniejszą analizę zmian w czasie. W ramach szeregów czasowych można identyfikować różne wzorce, trendy, sezonowe wahania oraz nieregularne zdarzenia.

W pracy A.Arsau, S.Babu, J.Widom(1) strumien danych jest nazywany nieograniczonym zbiorem elementów krotek należących do schematu strumienia i stempli czasowych tych elementów.

$$S = (s, t) \tag{1}$$

Z tego wynika, że charakterystyczną naturalną cechą strumieni jest szeregowość. Wartości nie są jedynym przedmiotem analizy, ale głównie ich kolejność i kontekst który zarysowują w czasie.

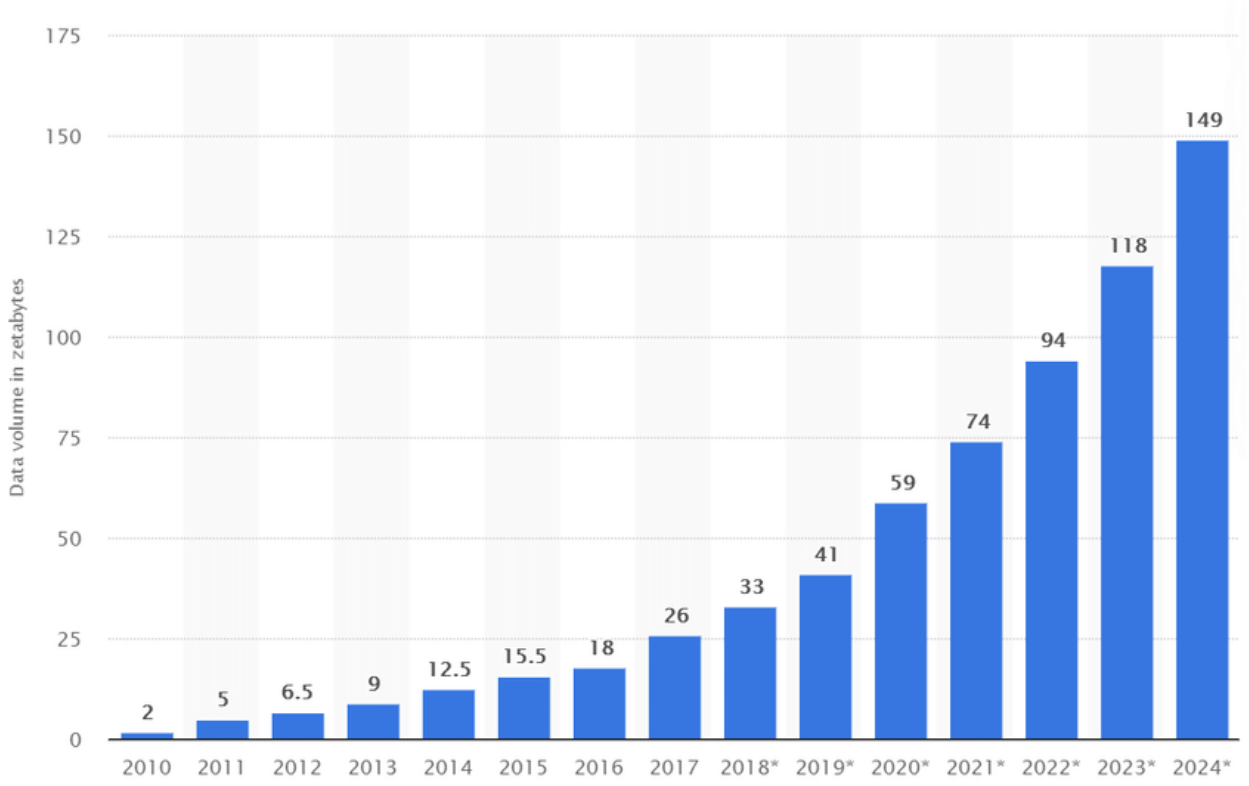
Szeregi czasowe są używane do monitorowania i prognozowania zmian,co pozwala wspierać procesy decyzyjne Mogą obejmować dane z różnych dziedzin, takich jak gospodarka, nauki przyrodnicze, zdrowie, finanse czy technologia.

Przykłady zastosowań szeregów czasowych w informatyce to prognozowanie ruchu w sieciach komputerowych, monitorowanie wydajności systemów, analiza logów serwerów, predykcja awarii sprzętu czy prognozowanie trendów w danych ekonomicznych.

Wykrywanie wyjątków w strumieniach danych

Procesy gromadzenia danych, mimo postępu technologicznego, zawsze niosą ze sobą pewne ryzyko i nie są idealne. Istnieje wiele czynników, zarówno technicznych, jak i ludzkich, które mogą wprowadzić błędy do zebranych danych. Może on wynikać z wadliwego sprzętu pomiarowego, błędu ludzkiego lub przypadkowego zbiegu okoliczności. Dane przesyłane do analizy mogą zawierać szum, błędy pomiarowe, wartości niemożliwe lub w skrajnych przypadkach nie mieć wartości.

Proces czyszczenia danych stał się integralnym i fundamentalnym krokiem w procesie analizy danych, w szczególności w dzisiejszych czasach kiedy ilość przesyłanych danych z roku na rok jest coraz większa.



Rysunek 1: Ilość danych stworzonych / pobranych / skopiowanych w latach 2010 - 2021 z prognozami do roku 2024 (2)

Dzięki czyszczeniu analiza staję się dokładniejsza a modele lepiej spełniają swoją role w prognozowaniu kolejnych wartości. Podstawowym krokiem czyszczenia danych jest wykrywanie i usuwanie wyjątków z serii danych.

Wyjątkiem nazywamy obserwację, której wartość znacząco różni się od innych wartości w losowej próbie z populacji. Określenie "znacząco różni" nie jest precyzyjnym określeniem.

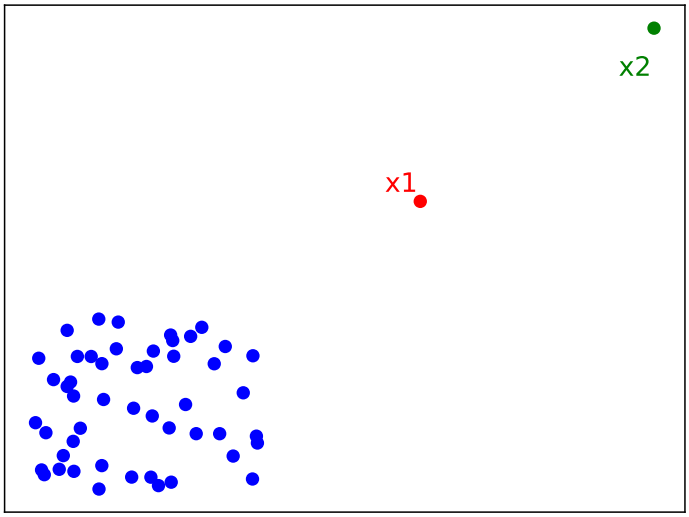
Kontekst każdej analizy jest wyjątkowy. W rozumieniu tej definicji, każda analiza ma za zadanie zdefiniować czym i jaka będzie znacząca różnica.

Efekt Maskowania

Efekt maskowania *(ang. masking effect)* jest obecnym problemem w wykrywaniu wyjątków, wpływa on negatywnie na dokonywane analizy. Dlatego metoda wykrywania wyjątków powinna być odporna na działanie efektu i wykryć zamaskowaną anomalię.

Maskowaniem wyjątku nazywamy zjawisko nie wykrycia wyjątku, z powodu wpływu większej anomalii na statystykę testową, która determinuję wyjątek.

Efekt maskowania może wystąpić w sytuacji, gdy analiza, z góry narzuca wykrycie i usunięcie ustalonej liczby wyjątków. Maskowanie wystąpi w przypadku nieoszacowania liczby wyjątków. Ciekawym przypadkiem jest sytuacja odwrotna, gdy założenie liczby wyjątków przeszacowuję faktyczną liczbę wyjątków. Dochodzi do przeciwnego efektu zwanego **swamping**, kiedy element bliskiego skupiska zostaje rozpoznany jako wyjątek



Rysunek 2: Efekt Maskowania: Wyjątek *x2* jest bardziej odstający, x2 może zamaskować wykrycie wyjątku *x1*

Algorytm Chen-Liu

Praca Chung Chen i Lon-Mu Liu *"Joint Estimation of Model Parameters and Outlier Effects in Time Series"* dokumentuję algorytm analizy strumienia danych. Podstawowym celem

badań było przedstawienia procedury wykrywania wyjątków, która uwzględnia możliwość istnienia fałszywych i zamaskowanych wyjątków. Dodatkowo była w stanie obliczyć wpływ wyjątków na model, oraz oszacować nowe parametry modelu.

Dzięki precyzyjnemu zdefiniowaniu czterech różnych typów wyjątków, które pojawiały się w poprzednich badaniach, staje się możliwe pełniejsze zrozumienie ich wpływu na dane badawcze. Określenie obliczonego wpływu staje się kluczową podstawą do przeprowadzenia korekty parametrów modelu oraz umożliwia dalszą analizę.

Przyjmijmy, że modelem będzie ARMA postaci:

$$Y_t = \frac{\theta(B)}{\alpha(B)\phi(B)} \tag{2}$$

Procedura "Chen-Liu" przedstawia szereg czasowy w następujący sposób:

$$Y_t^* = Y_t + \omega\xi(B)I_t(t_1) \tag{3}$$

Gdzie:

- Funkcja I_t przyjmuję wartość 1 kiedy występuję wyjątek w każdym innym wypadku jest równa 0.
- ω jest początkową wartością odchylenia
- $\xi(B)$ określa jak będzie kształtował się wpływ wyjątku w czasie.

Algorytm przyjmuję rozróżnia następujące wyjątki na następujące typy:

- *Additive Outlier* (AO): Efekt charakteryzuję się pojedynczą, nagłą anomalią.

$$AO : \xi(B) = 1 \tag{4}$$

- *Level Shift* (LS): Trwały, ciągła zmiana wartości.

$$LS : \xi(B) = \frac{1}{1-B} \tag{5}$$

- *Temporary change* (TC): Efekt słabnie w czasie. Dodatkowym parametrem jest δ która określa krzywiznę.

$$TC : \xi(B) = \frac{1}{1-\delta B} \qquad 0 < \delta < 1 \tag{6}$$

- *Innovational Outlier* (IO): Krzywa w czasie jest odzwierciedleniem modelu. W przypadku modelu ARMA wygląda następująco:

$$IO : \xi(B) = \frac{\theta(B)}{\alpha(B)\phi(B)} \tag{7}$$

W późniejszych pracach i implantacjach (**alsdkf?**) można napotkać na 5 typ wyjątków SLS. Ma za zadanie lepiej odwzorcowywać sezonowość szeregu czasowego niż typ IO, który nie musi dziedziczyć cech sezonowości z przyjętego modelu.

$$SLS : \xi(B) = 1/S \qquad S = 1 + B + ... + B^{s-1} \tag{8}$$

Algorytm postępowania jest iteracyjny i jest podzielony na 3 oddzielne etapy. Przedstawione poniżej kroki algorytmu są uproszczone. Dokładny opis procedury można znaleźć w oryginalnej pracy(**dupa?**):

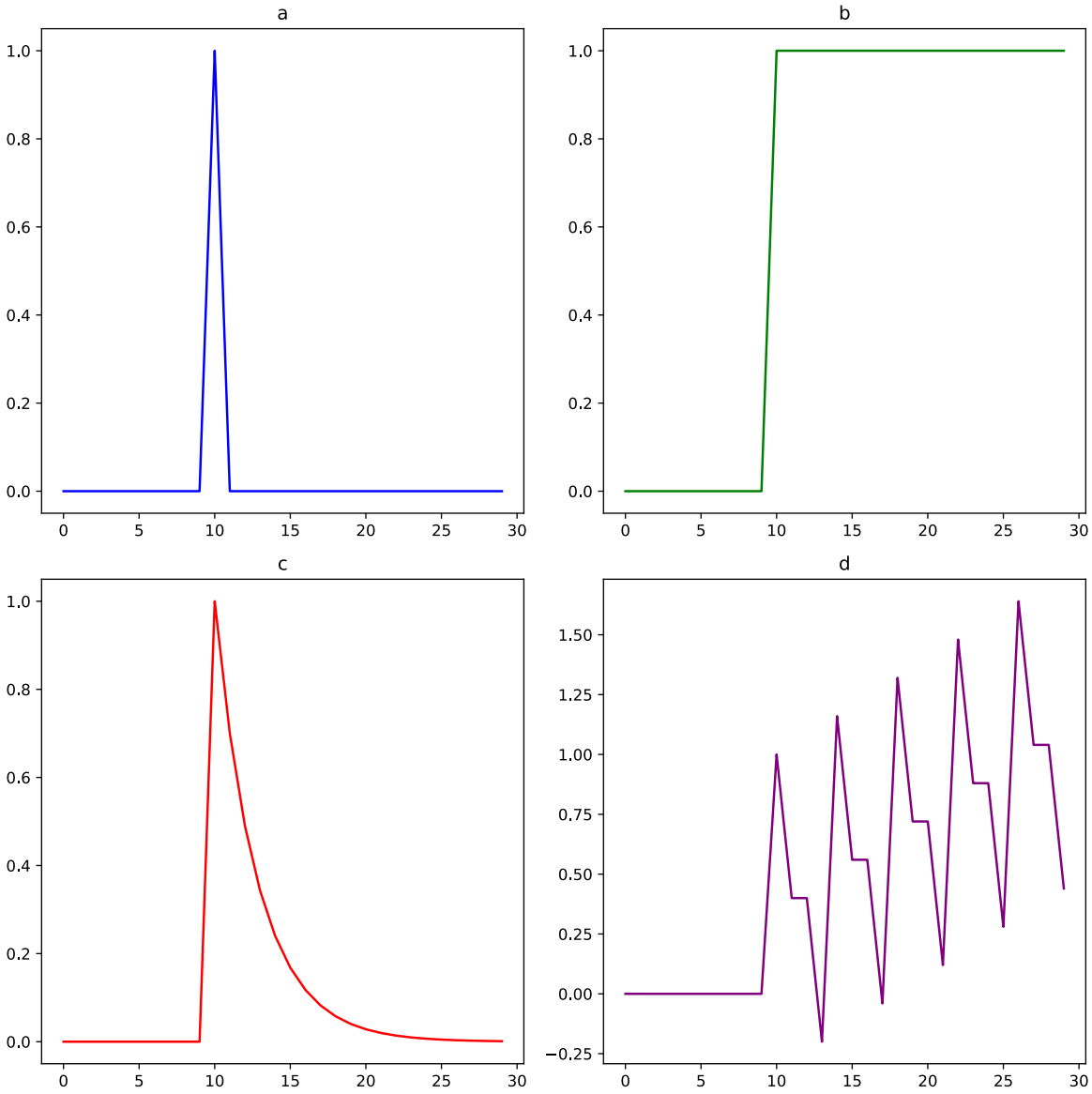
1. Obejmuję wykrycie potencjalnych wyjątków. W tym celu dokonuję się dopasowania przyjętego modelu do serii danych i obliczenia odchyleń dla każdego punktu. W następnym kroku, dla każdego punktu i szukanego typu obliczane są statystyki τ i ω. Jeśli statystyka $|\tau|$ w czasie t jest większa niż przyjęta wartość krytyczna C oznacza, że w tym punkcie wystąpił wyjątek. Jeżeli 2 lub więcej typów przekroczyła wartość krytyczną wybierany jest typ z największym współczynnikiem τ. Następuję obliczenie efektów wykrytych wyjątków i usunięcie z serii danych. Poprawiona seria danych zostaję ponownie analizowana zgodnie z poprzednimi krokami, dopóki w iteracji nie zostanie wykryty żaden wyjątek, lub zostanie przekroczona ustalona liczba iteracji.

2. W tym etapie zostaję sprawdzony wpływ potencjalnych wyjątków. Do tego celu zostaje użyty model regresji obliczyć wielkość wyjątku $\widehat{\omega}$. Obliczana jest ponownie τ_j korzystając ze wzoru: $\widehat{\tau}_j = \widehat{\omega}_j / std(\omega)$. Jeśli statystyka jest niższa niż wartość krytyczna C wyjątek jest usuwany z listy potencjalnych. Pętla zostaję przerwana w przypadku braku wykrycia błędu lub przekroczenia liczby iteracji. Następuję kolejne dopasowanie modelu skorygowanej serii.

3. Ostatnim etapem jest powtórzenie I i II fazy algorymu wykorzystując ponownie nowe parametry modelu z. W pierszej fazie nie koryguję parametrów. W 2 fazie $\widehat{\omega}$ jest końcową wartością.

Metody Wykrywania wyjątków

Na przestrzeni lat postwało wiele metod wykrywania wyjątków.



Rysunek 3: Porównanie efektów różnych wyjątków a) AO, b) LS, c) TC, d) IO ARIMA(0,1,1)(0,1,1)

Model ARIMA

W czasach przed opracowaniem modelu ARIMA prognozowanie wymagały posiadania wiedzy na temat matematycznego modelu procesu. W praktyce badawczej struktura szeregu czasowego bywa często niejednoznaczna, a wariancja składnika losowego jest znaczna. Pomimo tych trudności istnieje potrzeba nie tylko odkrywania ukrytych wzorców, ale również generowania prognoz. W tym celu została opracowana metodyka ARIMA, rozwinięta przez Boxa i Jenkinsa (1976), która zdobyła znaczną popularność w różnych dziedzinach.

Model ARIMA *AutoRegressive Integrated Moving Average* to model używany do analizy i prognozowania przyszłych wartości w oparciu o historyczne dane.

Nazwa ARIMA opisuje trzy główne składowe tego modelu: AutoRegressive (**AR**), Integrated (**I**) i Moving Average (**MA**).

- AutoRegressive (AR): Ta część modelu odnosi się do autoregresji, czyli zależności między bieżącą wartością szeregu a jego wcześniejszymi wartościami. Model AR opiera się na przekonaniu, że bieżąca wartość szeregu czasowego jest funkcją jej poprzednich wartości, co uwzględnia wpływ autokorelacji.

$$x(t) = \phi_1 \cdot x(t-1) + \phi_2 \cdot x(t-2) + ... + \phi_p \cdot x(t-p) + a(t) \tag{9}$$

- Integrated (I): Integracja dotyczy transformacji szeregu czasowego w celu uzyskania stacjonarności. Stacjonarność oznacza, że statystyki szeregu nie zmieniają się w czasie, co ułatwia analizę. Proces integracji polega na różniczkowaniu danych, czyli odejmowaniu od każdej wartości szeregu jej poprzedniej wartości.

- Moving Average (MA): Ta część modelu odnosi się do średniej ruchomej, czyli uwzględnienia pewnej liczby poprzednich składników losowych w modelu. Model MA zakłada, że bieżąca wartość szeregu czasowego jest sumą wcześniejszych błędów losowych.

$$x(t) = b_1 \cdot a(t-1) + b_2 \cdot a(t-2) + ... + b_q \cdot a(t-q) + \varepsilon(t) \tag{10}$$

Bibliografia

1. FOUNDATION, Python Software. Python Documentation [online]. 2023. [udostępniono 11.11.2023]. Pobrano: https://docs.python.org/.

2. BERISHA, Blend & MËZIU, Endrit. *Big Data Analytics in Cloud Computing: An overview*[1]. luty 2021. S.l.: s.n.