Kramek A. i Szwajka K. (Redaktorzy)

# Predykcja w układach mechanicznych i automatycznych 2021 – metody statystyczne i sztuczna inteligencja

Wydano za zgodą Rektora

O p i n i o d a w c a
prof. dr hab. inż. Jarosław Sęp
dr hab Paweł Przybyłowicz, prof. AGH

R e d a k t o r  n a c z e l n y
Wydawnictw Politechniki Rzeszowskiej
dr hab. inż. Lesław GNIEWEK, prof. PRz

R e d a k t o r
Agnieszka Kramek i Krzysztof Szwajka

P r z y g o t o w a n i e  m a t r y c
Patrycja Kuziora

P r o j e k t  o k ł a d k i
Joanna Mikuła

A u t o r z y
Maryna Bulakh
Andrzej Chmielowiec
Leszek Klich
Tomasz Kycia

*sztuczna inteligencja, przetwarzanie sygnałów, statystyka, testy statystyczne, rozkłady prawdopodobieństwa, ryzyko, szacowanie ryzyka*



ISBN 978-83-7934-532-8

Oficyna Wydawnicza Politechniki Rzeszowskiej
al. Powstańców Warszawy 12, 35-959 Rzeszów

Nakład 100 + 40 egz. Ark. wyd. 7,62. Ark. druk. 9,25.

Oddano do druku w grudniu 2021 r. Wydrukowano w grudniu 2021 r.

Drukarnia Oficyny Wydawniczej, al. Powstańców Warszawy 12, 35-959 Rzeszów

Zam. nr 69/21

# Przedmowa redaktora


Monografia pt. *Predykcja w układach mechanicznych i automatycznych 2021 - metody statystyczne i sztuczna inteligencja* opublikowana została w serii Materiały, Technologie, Konstrukcje, Eksploatacja. Autorzy podjęli problematykę bardzo aktualnego i interesującego zagadnienia związanego z usprawnieniem i organizacją pracy układów mechanicznych. Predykcja jest strategią, dzięki której możliwe jest optymalne wykorzystanie maszyn i urządzeń poprzez przewidywanie ich stanu na podstawie pomiarów prowadzonych w czasie rzeczywistym oraz analizie już istniejących danych w szerokim zakresie czasowym.

W pierwszej części monografii zatytułowanej *Analiza statystyczna dużych zbiorów danych w kontroli jakości i utrzymaniu ruchu* autorzy zaprezentowali wykorzystywanie metod statystycznych w kontroli jakości i zarządzaniu przedsiębiorstwem. Rosnąca liczba czujników na liniach produkcyjnych wiąże się z koniecznością stosowania szybszych i skuteczniejszych metod zarówno kontroli jakości, jak i wyszukiwania powiązań pomiędzy zmiennymi charakteryzującymi procesy. Dlatego też niniejszy rozdział został poświęcony wykorzystaniu narzędzi statystycznych i informatycznych do efektywnego rozwiązywania niektórych problemów związanych z analizą dużych zbiorów danych pomiarowych.

W kolejnej części opracowania pod tytułem *Metody sztucznej inteligencji w predykcyjnym utrzymaniu ruchu* autor zwrócił uwagę na aspekt wykorzystania algorytmów sztucznej inteligencji w wielu dziedzinach życia oraz w przemyśle. Zastosowanie odpowiednio przygotowanych środowisk programistyczno-sprzętowych pozwala na precyzyjne określenie żywotności pracy maszyny w fabryce czy dokładne planowanie serwisu i utrzyma-

nie maszyn. Artykuł jest wstępem do bardziej szczegółowych badań nad wdrażaniem uczenia maszynowego. Autor wykorzystał sztuczną inteligencję w systemach bazujących na przetwarzaniu dźwięku. Zastosowane w pracy narzędzia programistyczne są ogólnodostępne i darmowe. Pozwalają analizować w czasie rzeczywistym zachodzące procesy w maszynie i na bieżąco informować o stanie urządzeń.

Ostatni rozdział *Prognozowanie ryzyka wypadków przy pracy w przedsiębiorstwie produkcyjnym* przedstawia model prognozowania wartości ryzyka bezpieczeństwa w ciągu okresów przeszłych, bieżących oraz przyszłych. Obejmuje on podejście prognozowania według trzech scenariuszy: optymistycznego, przybliżono-realnego oraz pesymistycznego. W celu uwzględnienia jak największej ilości wskaźników przeprowadzono prognozowanie metodą liniową oraz probabilistyczną. Analiza uzyskanych wyników pokazuje różne ścieżki możliwych sposobów zapewnienia bezpieczeństwa.

Autorzy niniejszej monografii – pracownicy naukowo-dydaktyczni podejmują zarysowane tu problemy. Opracowanie to jest doskonałym przykładem współpracy naukowców reprezentujących różne specjalności, ale skupiających swoje zainteresowania w obszarze predykcji w układach mechanicznych i automatycznych.

dr Agnieszka Kramek
Politechnika Rzeszowska

dr inż. Krzysztof Szwajka
Politechnika Rzeszowska

# Przedmowa Prezesa WSM

East Automotive Alliance

Każdy dałby fortunę aby móc przewidzieć przyszłość i skutecznie na nią oddziaływać. Oczywiście pod warunkiem, że wydana fortuna zwróciłaby się z nawiązką. Ludzkość podejmowała działania w kierunku przewidywania przyszłości już od jej zarania. Wszystkim znane są np. przypadki wykorzystywania w starożytności wiedzy astronomicznej do sterowania decyzjami politycznymi. Jeszcze bliższym przykładem są prognozy pogodowe. Wprawdzie z dużą dokładnością można je przewidzieć i przygotować się do anomalii ale jeszcze na razie trudno na nie wpływać.

Metody predykcji są dzisiaj szeroko wykorzystywane w przemyśle. Jednym z obszarów gdzie ich znaczenie jest fundamentalne, są wielkoseryjne procesy produkcyjne. Każdy z zarządzających takimi procesami zawsze chce obserwować niezakłócony takt wytwarzania wyrobów na swojej linii produkcyjnej. Ale niestety nawet najlepiej zorganizowane procesy nie są stu procentowo skuteczne. Uważa się, że wskaźnik OEE (Overall Equipment Efficency) na poziomie ponad 80% jest już efektywny. Jednym ze składników wpływających na rezultat OEE jest UT (Up Time), czyli dostępny czas pracy urządzeń produkcyjnych. Jedną z jego składowych jest z kolei TDT (Technical Down Time), czyli czas przestojów urządzeń z powodu awarii technicznych. Oczywiście zadaniem zarządzających procesami produkcji jest jego minimalizowanie. Można to robić w sposób reaktywny, czyli podejmować działania po wystąpieniu awarii, lub planowy, wyprzedzający pojawienie się awarii. Ten planowy sposób zawiera w sobie dwie metody: prewencyjną i predykcyjną. Zastosowanie każdej z tych trzech metod w określonych przypadkach ma swoje techniczne, technologiczne, a przede wszystkim ekonomiczne uzasadnienie. Tematyka tej monografii skupia się na wykorzystywaniu metod predykcyjnych w utrzymaniu ruchu

jako tych najbardziej skutecznych ale niekoniecznie w każdym przypadku najtańszych i mających powszechne zastosowanie. O finalnym ich użyciu zawsze decyduje analiza opłacalności.

Program predykcyjnego utrzymania ruchu zainicjowany został przez Wschodni Sojusz Motoryzacyjny w ramach europejskiego programu DRIVES (Development and Research on Innovative Vocational and Eductional Skills), którego celem jest przygotowanie programów kształcenia pod kątem przyszłych potrzeb branży motoryzacyjnej w Europie. Predykcyjne utrzymanie ruchu ma charakter uniwersalny i ma zastosowanie we wszelkiego rodzaju firmach produkcyjnych. Wschodni Sojusz Motoryzacyjny i Wydział Mechaniczno-Technologiczny Politechniki Rzeszowskiej w Stalowej Woli podjęły się wprowadzenia tej tematyki do programu nauczania na poziomie studiów inżynierskich i magisterskich. Dzisiejsze wymagania dla skuteczności procesów produkcyjnych wymuszają coraz szersze stosowanie metod predykcyjnych. Skoncentrowanie na Podkarpaciu takich branż produkcyjnych jak motoryzacja, lotnictwo czy przetwórstwo metalowe i spożywcze, gdzie występuje duże umaszynowienie, automatyzacja i robotyzacja, skutkuje zapotrzebowaniem na wysoko wykwalifikowanych specjalistów w tej dziedzinie. Do dalszego rozwoju metod predykcyjnych przyczyni się na pewno wprowadzanie narzędzi przemysłu 4.0. Gromadzenie dużej ilości danych bezpośrednio z urządzeń i ich zautomatyzowana analiza to podstawa poprawnego wnioskowania i skutecznego podejmowania decyzji.

Jestem przekonany, że monografia *Predykcja w układach mechanicznych i automatycznych*, którą oddajemy do Waszej dyspozycji, przyczyni się do wzrostu poziomu wiedzy wykładowców, studentów i pracowników firm produkcyjnych, a w efekcie do podniesienia poziomu skuteczności procesów wytwórczych.

Ryszard Jania
Prezes Wschodniego Sojuszu Motoryzacyjnego

Badania oraz ich publikacja zostały zrealizowane dzięki środkom na utrzymanie potencjału badawczego Politechniki Rzeszowskiej w ramach umowy PB28.KI.21.001.

Badania zostały zrealizowane dzięki wykorzystaniu aparatury z projektu: *Utworzenie naukowo-badawczego Laboratorium Międzyuczelnianego w Stalowej Woli* w ramach Programu Operacyjnego Rozwój Polski Wschodniej 2007-2013, Oś Priorytetowa I – Nowoczesna Gospodarka, Działanie I.3 – Wspieranie Innowacji, numer projektu: POPW.01.03.00-18-016/12-00.

# Spis treści

# 1. Analiza statystyczna dużych zbiorów danych w kontroli jakości i utrzymaniu ruchu

Andrzej Chmielowiec[1]
Politechnika Rzeszowska, achmie@prz.edu.pl

Leszek Klich[2]
Politechnika Rzeszowska, l.klich@prz.edu.pl

**Streszczenie** Wykorzystywanie metod statystycznych w kontroli jakości i zarządzaniu przedsiębiorstwem sięga swoimi korzeniami początków XX wieku. Na początku XXI wieku ta dziedzina nauki stanęła przed wyzwaniem przetwarzania dużych zbiorów danych pomiarowych. Rosnąca liczba czujników na liniach produkcyjnych wiąże się z koniecznością stosowania szybszych i skuteczniejszych metod zarówno kontroli jakości, jak i wyszukiwania powiązań pomiędzy zmiennymi charakteryzującymi procesy. Dlatego też niniejszy rozdział został poświęcony wykorzystaniu narzędzi statystycznych i informatycznych do efektywnego rozwiązywania niektórych problemów związanych z analizą dużych zbiorów danych pomiarowych.

[1]ORCID: 0000-0001-6629-0029, Wydział Mechaniczno-Technologiczny Politechniki Rzeszowskiej, Kwiatkowskiego 4, 37-450 Stalowa Wola
[2]ORCID: 0000-0001-6099-2417, Wydział Mechaniczno-Technologiczny Politechniki Rzeszowskiej, Kwiatkowskiego 4, 37-450 Stalowa Wola

## 1.1. Wprowadzenie

Produkcja seryjna sięgająca dziesiątek, a nawet setek milionów sztuk rocznie nie jest w dzisiejszych czasach rzeczą nadzwyczajną. Nowoczesne fabryki pełne robotów i automatyki w sposób ciągły wytwarzają ogromne ilości dóbr. Poziom automatyzacji wymusza jednak konieczność należytej kontroli procesu produkcyjnego. Wiąże się to na ogół z koniecznością zainstalowania dużej liczby czujników, które rejestrują stan maszyn oraz jakość wytwarzanych produktów. Liczba czujników sięga $10^3$ w przypadku średniej wielkości przedsiębiorstwa, a dochodzi nawet do $10^5$ w przypadku bardzo dużych fabryk. Dlatego też stan produkcji w chwili $t$ możemy opisywać jako ciąg danych pomiarowych $S_t = (s_{1,t}, s_{2,t}, \ldots, s_{m,t})$, gdzie $s_{i,t}$ są wynikami pomiarów z poszczególnych czujników. Zapisując stan przedsiębiorstwa w kolejnych chwilach czasu $t_1 < t_2 < \cdots < t_r$ otrzymujemy macierz, która odzwierciedla zmiany procesu produkcyjnego:

$$(s_{i,t_j}) = \begin{pmatrix} s_{1,t_1} & s_{2,t_1} & \cdots & s_{m,t_1} \\ s_{1,t_2} & s_{2,t_2} & \cdots & s_{m,t_2} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1,t_r} & s_{2,t_r} & \cdots & s_{m,t_r} \end{pmatrix}. \tag{1.1}$$

Dane z macierzy pomiarów mogą być analizowane w celu wykrycia nieprawidłowości lub określonych zależności pomiędzy zdarzeniami. Nawet z pozoru niewielka macierz staje się bardzo wymagająca obliczeniowo jeżeli konieczne jest niezależne analizowanie podmacierzy w niej zawartych. Do wyznaczania wzajemnych powiązań pomiędzy zdarzeniami oraz wykrywania nieprawidłowości bardzo często wykorzystywane są narzędzia statystyczne. Pozwalają one w sposób automatyczny przetwarzać dane zawarte w takiej macierzy i wskazywać ciekawe powiązania pomiędzy zmiennymi składającymi się na stan procesu produkcyjnego. W niniejszym rozdziale zaprezentowane zostaną podstawowe narzędzia analizy statystycznej, które mogą znaleźć zastosowanie w wykrywaniu zależności i wzajemnych powiązań między wartościami przedstawionej macierzy.

Badanie statystyczne szeregów czasowych pochodzących z procesu produkcyjnego jest dzisiaj bardzo powszechną metodą oceny jakości tego procesu. Zaawansowane metody analizy danych pozwalają na kontrolę jakości procesu, szacowanie niezawodności, czy też badanie odporności konstrukcji/projektu. Zapoczątkowana przez Shewarta [82] statystyczna kontrola procesu [63, 96, 62] stanowi dzisiaj bardzo mocno rozwiniętą metodę zarządzania procesem produkcyjnym. Obejmuje ona swoim zasięgiem zarówno

analizę funkcji jednej zmiennej [63, 96], jak i analizę funkcji wielu zmiennych [56, 59]. Szczególnie dużo z metod statystycznych czerpie teoria niezawodności, o czym świadczą między innymi monografie Barlowa i Proschana [10], Ansella i Phillipsa [5], Johnsona i pozostałych [41], Biroliniego [11], Woo [95], Grynchenko i Alfyorova [30]. Jej obszarem badań jest zmienność funkcji jakości w czasie, która nad wyraz dobrze wyraża się w terminach rachunku prawdopodobieństwa. Rozkłady modelujące cykl życia maszyn i urządzeń pozwalają w efektywny sposób zarządzać liniami produkcyjnymi – ich niezawodnością działania i jakością wytwarzanych elementów. Szczególnie ważnym pojęciem jest dla tej dziedziny rozkład Weibulla [92, 93], którego prekursorami byli Fréchet [29] oraz Fisher i Tippett [27]. Rozkład ten odgrywa szczególną rolę w teorii niezawodności, o czym świadczą chociażby publikacje Johnsona [40] i Lai [49]. Dużo szczegółowych informacji na jego temat można znaleźć między innymi w monografiach Murthyego [65], Lai [50] i McPhersona [61]. Wprowadzone przez Taguchiego [84] metody badania odporności konstrukcji (*Robust Design*) również w sposób znaczący korzystają z różnorodnych narzędzi statystycznych. Zastosowanie ich w przedsiębiorstwach produkcyjnych skutkowało 2-krotnym [43], a w niektórych przypadkach nawet 4-krotnym zmniejszeniem zmienności procesu produkcyjnego [71]. Do przełomowych osiągnięć w tej dziedzinie możemy zaliczyć wyniki opublikowane przez: Kackera [42], Leona i pozostałych [52], Boxa [13], Naira [66] i Tsui [88]. Metody zapoczątkowane przez Taguchiego zostały również rozszerzone na projektowanie odporności w oparciu o wiele charakterystyk. Problemy te poruszają między innymi: Logothetis i Haigh [55], Pignatiello [72], Elsayed i Chen [23] oraz Tsui [89]. Cechą wspólną opisanych powyżej zagadnień jest intensywne korzystanie z narzędzi statystycznych na zbiorach danych pochodzących z procesu produkcyjnego. Należy podkreślić, że wraz ze wzrostem liczby czujników i rozmiaru baz danych, coraz większy nacisk kładziony jest na efektywność przetwarzania. Dlatego też głównym zadaniem niniejszej publikacji jest pokazanie, w jaki sposób współczesne narzędzia informatyczne i metody numeryczne pomagają radzić sobie z niektórymi problemami analizy danych.

Statystyczna kontrola procesu wpisuje się w znacznie szerszy problem analizy statystycznej, jakim jest wyszukiwanie różnego rodzaju anomalii w szeregu czasowym. Problem ten jest przez ostatnie 20 lat bardzo intensywnie badany. Zaproponowano wiele algorytmów i technik wyszukiwania anomalii pod warunkiem, że znany jest przedział jej poszukiwania. Za przykład mogą tu posłużyć wyniki uzyskane przez Keogha i pozostałych [45, 46]

oraz Senina i pozostałych [78]. Niemniej jednak cały czas ogromnym wyzwaniem jest przeszukanie całego zbioru dostępnych danych. Za przykład obrazujący poziom złożoności problemu może nam tutaj posłużyć jednowymiarowy ciąg $10^5$ obserwacji pojedynczej wielkości – może to być na przykład jeden z wymiarów produkowanego elementu. Podkreślmy, że seria produkcyjna tej wielkości nie jest niczym nadzwyczajnym i z łatwością jest osiągana w warunkach produkcji seryjnej. Dla takiej serii istnieje ponad $1.6 \cdot 10^{14}$ pod-serii, które mogą zawierać różnego rodzaju anomalie. Przykład ten pokazuje, jak bardzo od strony obliczeniowej komplikuje się sytuacja w momencie, gdy nie ma informacji co do potencjalnej lokalizacji anomalii. Pewne propozycje dotyczące próby rozwiązania tego problemu od strony algorytmicznej daje publikacja [17], ale można powiedzieć, że jest to zaledwie odkrycie wierzchołka góry lodowej. Należy podkreślić, że poziom złożoności tego rodzaju zagadnień znacznie wzrasta, gdy zamiast prostych ciągów/wektorów pojawiają się macierze danych. Dobrym przykładem złożoności tego zagadnienia jest przeglądowy artykuł Ebnera i Henzego [22], który opisuje metody i problemy testowania normalności w przestrzeniach wielowymiarowych.

Współczesne metody znajdowania anomalii w szeregach czasowych można podzielić na trzy zasadnicze grupy w zależności od generowanych przez nie rezultatów. Rozróżniamy algorytmy znajdowania anomalii w punkcie, anomalii strukturalnej i anomalii serii (w przypadku wielu serii). Przez anomalię w punkcie rozumiemy odstępstwo wartości pojedynczego pomiaru od wartości znajdujących się w szeregu [32, 25]. Z kolei anomaliami strukturalnymi nazywamy takie podciągi danej serii, których statystyki odbiegają od własności wyznaczonych dla całego szeregu [44, 98, 78]. W pewien sposób powiązane są z tym zagadnieniem anomalie serii, które polegają na znajdowaniu odstępstw pomiędzy całymi ciągami pomiarów [38, 51]. Na popularności zyskują w ostatnim czasie algorytmy wykorzystujące metody sztucznej inteligencji. Jest to niewątpliwie przyszłościowy kierunek badań, o czym świadczy chociażby zaangażowanie w ten obszar firmy Intel. Doświadczenia zebrane przez firmę w tej dziedzinie zostały opublikowane przez Wanga i pozostałych [91]. Również publikacja Chalapathy i Chawla [15] zawiera przegląd metod wykorzystujących głębokie uczenie maszynowe na potrzeby wykrywania anomalii w szeregach czasowych.

W artykule [21] wprowadzony został podział algorytmów wykrywania anomalii w szeregach czasowych ze względu na rodzaj użytej metody. Ding

i pozostali rozróżnili między innymi metody: klasyfikacji, najbliższego sąsiedztwa, klasteryzacji i statystyczne. Należy podkreślić, że wszystkie trzy ostatnie metody w mniejszym lub większym stopniu bazują na wnioskowaniu statystycznym i rachunku prawdopodobieństwa. Dlatego też szybkie wyznaczanie wartości statystyk jest bardzo istotnym zagadnieniem w kontekście wydajności systemów wykrywania anomalii.

Niniejszy rozdział podzielony jest na dwie części. W pierwszej prezentowane są wybrane metody analizy statystycznej od strony teoretycznej, a druga część przedstawia ujecie praktyczne – implementację opisanych wcześniej metod w języku Python. Oczywiście opisane w kolejnych częściach podejście do analizy dużych zbiorów danych nie wyczerpuje katalogu metod, które mogą być w takim kontekście użyte. Niemniej jednak opisane trzy zagadnienia stanowią niejako podstawę, która może być wykorzystana do bardziej zaawansowanych obliczeń i uczenia maszynowego.

## 1.2. Wybrane metody analizy statystycznej dużych zbiorów danych

W tej części rozważane będą metody analizy wariancji oraz dwa rozkłady prawdopodobieństwa, które odgrywają kluczową rolę z punktu widzenia zarządzania jakością i teorii niezawodności. Rozkładami tymi są rozkład normalny (Gaussa) i rozkład Weibulla. Opis poszczególnych metod pozwalających na badanie określonych własności tych rozkładów zostanie przedstawiony ze szczególnym uwzględnieniem aspektu implementacyjnego. Automatyzacja oceny uzyskanych wyników i szybkość realizacji obliczeń jest bowiem niezwykle ważna w przypadku przetwarzania i przeszukiwania dużych zbiorów danych pomiarowych.

### 1.2.1. Analiza wariancji

Wariancja jest miarą koncentracji danych wokół wartości średniej. Im mniejsza wariancja, tym wyniki są bardziej skoncentrowane. Z kolei duża wartość wariancji wskazuje na rozrzut statystyczny i znaczące odległości pomiędzy punktami analizowanego zbioru danych. W warunkach produkcyjnych duża zmienność procesu może wskazywać na problemy jakościowe. Dlatego też warto analizę danych rozpocząć właśnie od analizy wariancji. W tym miejscu należy zwrócić uwagę, że analiza procesu nie może zakładać badania wariancji tylko w obrębie ciągów pomiarowych o określonej długości. Ustalenie

rozmiaru okna czasowego może dać bowiem niekompletny obraz procesu. Dlatego też w dalszych rozważaniach pod uwagę będzie brany uporządkowany chronologicznie ciąg pomiarów $x_1, \ldots, x_n$ reprezentujący wartości jednej ze zmiennych losowych definiowanych przez proces produkcyjny. Załóżmy, że dla każdego elementu $x_i$ wyznaczony zostanie ciąg wariancji, które będą wyliczone dla otoczeń tego elementu. To znaczy, że ustalony został pewien ciąg promieni $\delta_1, \ldots, \delta_m$, dla którego wyznaczamy wariancje

$$\nu_{i,j}(x_{i-\delta_j}, \ldots, x_{i+\delta_j}) = \frac{1}{D_j} \sum_{k=i-\delta_j}^{i+\delta_j} x_k^2 - \left( \frac{1}{D_j} \sum_{k=i-\delta_j}^{i+\delta_j} x_k \right)^2, \tag{1.2}$$

gdzie $D_j = 2\delta_j + 1$. Zaznaczmy, że w powyższym wzorze celowo użyto obciążonego estymatora wariancji. Wynika to z faktu, że jest on efektywniejszy w implementacji, a przejście do estymatora nieobciążonego wymaga jedynie pomnożenia otrzymanej wielkości przez $\frac{2\delta_j+1}{2\delta_j}$. Dodatkowo w rozdziale tym będziemy zakładali, że kolejne promienie $\delta_j$ zwiększają się o pewną z góry ustaloną wielkość $w$. Oznacza to, że $\delta_j = j \cdot w$. Dla tak przyjętych założeń można wykazać, że niezależne wyznaczenie wszystkich wariancji $\nu_{i,j}$ dla $i \in \{1, \ldots, n\}$ oraz $j \in \{1, \ldots, m\}$ wymaga około $\frac{1}{6}mn^2$ operacji. Jeżeli przyjąć, że $m = \frac{n}{w}$ dla pewnego ustalonego $w$, to złożoność obliczeniowa algorytmu wyznaczającego zbiór wariancji $\nu_{i,j}$ jest rzędu $O(n^3/w)$. W praktyce oznacza to tyle, że dla niewielkiej liczby $10^4$ pomiarów i wartości $w = 100$ konieczne jest wykonanie $10^{10}$ operacji. Dlatego też do wyznaczania całego zbioru wariancji wykorzystamy podejście przedstawione w Lemacie 1 [17]. Pozwoli ono zredukować złożoność obliczeniową rozpatrywanego przez nas problemu do poziomu $O(n^2/w)$, co znacząco poprawi wydajność obliczeń. Na przykład dla wspomnianych wcześniej $10^4$ pomiarów czas obliczeń skróci się 10 000 razy.

Do efektywnej implementacji algorytmu wyznaczania zbioru wariancji $\nu_{i,j}$ wykorzystana zostanie formuła Welforda [94, 14] w postaci podanej przez Knutha [48]. Zakłada ona, że jeśli $\mu(s), \nu(s)$ są odpowiednio średnią arytmetyczną i wariancją ciągu pomiarów $s$, to dla $s = (x_1, \ldots, x_k)$ i $s^+ = (x_1, \ldots, x_k, x_{k+1})$ zachodzą następujące zależności

$$\mu(s^+) = \mu(s) + \frac{1}{k+1}(x_{k+1} - \mu(s)), \tag{1.3}$$

$$\nu(s^+) = \nu(s) + (x_{k+1} - \mu(s))(x_{k+1} - \mu(s^+)). \tag{1.4}$$

Formuła ta będzie wykorzystywana w początkowej fazie obliczeń – do momentu, gdy ciąg $s$ nie osiągnie rozmiaru $2\delta + 1$. W kolejnym etapie będzie

z kolei wykorzystywana metoda okienkowa przedstawiona w [17]. Wykorzystuje ona fakt, że dla $s = (x_1, \ldots, x_k)$ i $s' = (x_2, \ldots, x_{k+1})$ spełnione są następujące równości

$$\mu(s') = \mu(s) + \frac{x_{k+1} - x_1}{k}, \tag{1.5}$$

$$\nu(s') = \nu(s) + \frac{x_{k+1} - x_1}{k}(x_1 + x_{k+1} - \mu(s) - \mu(s')). \tag{1.6}$$

Przytoczone powyżej zależności pozwalają na zdefiniowanie Algorytmu 1.1 przeznaczonego do wyznaczenia wariancji dla wszystkich pomiarów $x_i$ dla zadanego promienia $\delta$. Wywołanie tej procedury dla wszystkich promieni $\delta_j = j \cdot w$ daje możliwość efektywnego wyznaczenia całego zbioru wariancji $\nu_{i,j}$, co zostało przedstawione w Algorytmie 1.2.

**Algorytm 1.1:** Procedura wyznaczania wariancji metodą okienkową o stałej szerokości okna

**Wejście:** Ciąg pomiarów pewnej zmiennej losowej $(x_1, \ldots, x_n)$, promień badanych indeksów $\delta$.
**Wyjście:** Ciąg wariancji $\nu_i$ odpowiadający promieniowi $\delta$.

// Inicjacja przetwarzania
$\mu_1 \leftarrow \frac{1}{1+\delta} \sum_{k=1}^{\delta+1} x_k$;
$\nu_1 \leftarrow \frac{1}{1+\delta} \sum_{k=1}^{\delta+1} (x_k - \mu_1)^2$;
$\mu_n \leftarrow \frac{1}{1+\delta} \sum_{k=n-\delta}^{n} x_k$;
$\nu_n \leftarrow \frac{1}{1+\delta} \sum_{k=n-\delta}^{n} (x_k - \mu_n)^2$;
// Przetwarzanie lewego krańca ciągu $i \in \{2, \ldots, \delta + 1\}$
**for** $i = 2, \ldots, \delta + 1$ **do**
    $\mu_i \leftarrow \mu_{i-1} + \frac{1}{\delta+i}(x_{i+\delta} - \mu_{i-1})$;
    $\nu_i \leftarrow \nu_{i-1} + (x_{i+\delta} - \mu_{i-1})(x_{i+\delta} - \mu_i)$;
**end**
// Przetwarzanie prawego krańca ciągu $i \in \{n - \delta, \ldots, n - 1\}$
**for** $i = n - 1, \ldots, n - \delta$ **do**
    $\mu_i \leftarrow \mu_{i+1} + \frac{1}{\delta+n-i+1}(x_{i-\delta} - \mu_{i+1})$;
    $\nu_i \leftarrow \nu_{i+1} + (x_{i-\delta} - \mu_{i+1})(x_{i-\delta} - \mu_i)$;
**end**
// Przetwarzanie środkowej części ciągu $i \in \{\delta + 2, \ldots, n - \delta - 1\}$
**for** $i = \delta + 2, \ldots, n - \delta - 1$ **do**
    $\mu_i \leftarrow \mu_{i-1} + \frac{1}{2\delta+1}(x_{i+\delta} - x_{i-\delta-1})$;
    $\nu_i \leftarrow \nu_{i-1} + \frac{1}{2\delta+1}(x_{i+\delta} - x_{i-\delta-1})(x_{i-\delta-1} + x_{i+\delta} - \mu_{i-1} - \mu_i)$;
**end**

**Algorytm 1.2:** Procedura wyznaczania zbioru wariancji dla wielu rozmiarów okna

**Wejście:** Ciąg pomiarów pewnej zmiennej losowej $(x_1, \ldots, x_n)$, krok zwiększania promienia $w$.
**Wyjście:** Ciąg wariancji $\nu_{i,j}$ odpowiadających wartościom $x_i$ i promieniom $\delta_j = w \cdot j$.

1 **for** $j = 1, 2, \ldots, m$ **do**
2 | $\nu_{*,j} \leftarrow$ Wyznacz wariancje dla promienia $\delta = w \cdot j$ za pomocą Algorytmu 1.1;
3 **end**

### 1.2.2. Rozkład normalny

Rozkład normalny, jako rozkład graniczny dla prawa wielkich liczb pojawia się nadzwyczaj często w praktyce zarządzania jakością. Wiele modeli zakłada bowiem, że cechy procesu, czy produktu mogą być modelowane za pomocą zmiennej losowej $X = \mu + Y$, gdzie $\mu$ jest pewną ustaloną wartością, a zmienna losowa $Y$ ma rozkład normalny. Nie zawsze jednak warunek ten jest spełniony. Pojawiają się momenty, w których zmienna losowa przestaje mieć rozkład normalny. Wykrycie takiej sytuacji jest kluczowe z punktu widzenia kontroli jakości, gdyż może świadczyć o zaburzeniu procesu produkcyjnego. W związku z tym metody służące do weryfikacji normalności rozkładu stanowią bardzo istotny element kontroli procesu produkcyjnego.

Testowanie normalności rozkładu jest już klasycznym problemem teorii prawdopodobieństwa i statystyki. Pierwsze analizy tego zagadnienia prowadzone były już przez Fishera [26] i Pearsona [70] w okresie międzywojennym. Istnieje wiele przykładów testowania jednowymiarowego rozkładu normalnego. Jedne z najczęściej używanych, to testy Andersona-Darlinga [4], Shapiro-Wilka [80], Shapiro-Francia [79] i Kołmogorowa-Smirnova [60, 54]. Z kolei analiza skośności i kurtozy (trzeciego i czwartego momentu) została wykorzystana do konstrukcji pierwszych testów normalności dla rozkładów wielowymiarowych [9, 57, 58]. Pod koniec XX wieku Bowman [12] zaproponował natomiast test normalności rozkładów wielowymiarowych bazujący na gładkości gęstości. Vasicek z kolei opracował test wykorzystujący entropię rozkładu normalnego [90]. W literaturze można też znaleźć kilka propozycji testów bazujących na funkcjach charakterystycznych, które obejmują [24, 18, 37], test BHEP [8, 36] oraz testy energetyczne [83]. Testowanie normalności rozkładu zwłaszcza w przypadku wielowymiarowym jest obecnie bardzo intensywnie badanym zagadnieniem. Świadczą

o tym chociażby publikacje z ostatnich lat takich autorów jak Mori i inni [64], Henze i Visagie [35], Tenreiro [85], Thas i Ottoy [86] oraz Zhu i inni [99]. Obszerne przeglądy metod testowania normalności rozkładu można znaleźć w pracach Henze [34] oraz Das i Imon [19].

Przypomnijmy, że rozkład normalny jest rozkładem dwuparametrycznym oznaczanym jako $\mathcal{N}(\mu, \sigma^2)$, gdzie $\mu$ jest wartością oczekiwaną, a $\sigma$ odchyleniem standardowym. Na Rysunku 1.1 przedstawiony jest wykres gęstości prawdopodobieństwa dla parametrów $\mu = 0$ i $\sigma = 1$ wraz z zaznaczonymi przedziałami $[-1, 1]$, $[-2, 2]$ i $[-3, 3]$, z których całka stanowi odpowiednio 0.683, 0.954 i 0.997. Całka ta określa prawdopodobieństwo z jakim zmienna losowa przyjmuje wartości z określonego przedziału. Funkcja gęstości dla rozkładu normalnego dana jest wzorem

$$f_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{1.7}$$

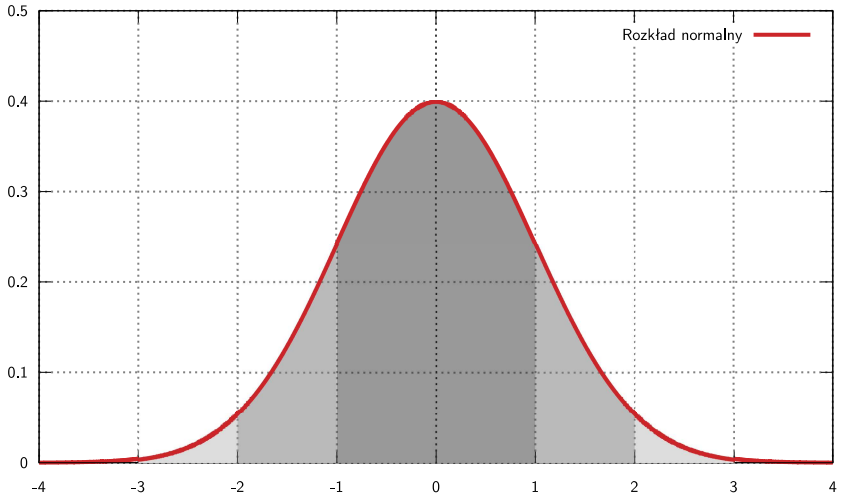
Natomiast jego dystrybuanta jest wyrażana za pomocą wzoru

$$F_{\mu,\sigma}(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right), \tag{1.8}$$

gdzie $\operatorname{erf}(x)$ jest tak zwaną funkcją błędu zdefiniowaną jako

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, \mathrm{d}t. \tag{1.9}$$

Prezentowanie systematycznego opisu wszystkich znanych testów weryfikujących normalność rozkładu nie jest celem niniejszego rozdziału. Podana na początku tej części obszerna literatura daje czytelnikowi możliwość zapoznania się z konkretnymi metodami. Przykłady użycia części z tych metod zostaną również zaprezentowane w dalszej części rozdziału jako element wykorzystania konkretnych bibliotek do analizy statystycznej. Aby jednak zarysować złożoność problematyki statystycznego testowania normalności rozkładu przedstawiony zostanie opis testu Shapiro-Wilka. Test ten jest po dzień dzisiejszy uznawany za jeden z najsilniejszych testów normalności. Jego główną wadą są jednak ograniczenia numeryczne, które uniemożliwiają testowanie dużych próbek. Aktualnie użytkowane biblioteki do analizy statystycznej pozwalają przeprowadzać test na wektorach złożonych z około 5000 próbek. Niemniej jednak zalecane jest, aby każdorazowo weryfikować w dokumentacji pakietu numerycznego maksymalną liczbę danych




Rysunek 1.1: Funkcja gęstości gęstości rozkładu normalnego dla parametrów $\mu = 0$ i $\sigma = 1$.

wejściowych, które mogą być poprawnie zbadane przez daną implementację.

Istotą testu Shapiro-Wilka jest porównanie wariancji próby z wariancją, którą powinien mieć rozkład normalny w przypadku gdyby dane faktycznie pochodziły z takiego rozkładu. Test ten odpowiada więc na pytanie, w jakim stopniu próba ma szansę reprezentować rozkład normalny. Statystyka testowa Shapiro-Wilka ma postać

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_i\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}, \tag{1.10}$$

gdzie ciąg $x = (x_1, \ldots, x_n)$ jest ciągiem statystyk pozycyjnych (wyrazy ciągu posortowane są rosnąco), a $\bar{x}$ jest wartością średnią z podanego ciągu statystyk. Generalnie wartości statystyki $W$ są liczbami z przedziału $[0, 1]$. W przypadku, gdy testowana jest próba pochodząca z rozkładu normalnego, to statystyka testowa $W$ dąży do jedności wraz ze wzrostem liczebności próby. Wykorzystany w teście wektor współczynników $a = (a_1, \ldots, a_n)$ jest dobrany w taki sposób, aby wielkość $\frac{1}{\sqrt{n-1}} \sum_{i=1}^{n} a_i x_i$ była najlepszym liniowym nieobciążonym estymatorem odchylenia standardowego przy założeniu normalności testowanego rozkładu. Największym problemem w przypadku tego testu jest wyznaczenie wektora współczynników $a$, który powi-

nien spełniać równanie

$$a = \frac{V^{-1}m}{(m^T V^{-1} V^{-1} m)^{\frac{1}{2}}}, \tag{1.11}$$

gdzie $m$ jest wektorem wartości oczekiwanych dla próby statystyk pozycyjnych rozmiaru $n$, a macierz $V$ jest macierzą kowariancji statystyk pozycyjnych z próby i wektora $m$. Problemy obliczeniowe związane z wyznaczaniem współczynników wektora $a$ są tematem dość aktualnym. Zwracali na nie uwagę już Shapiro i Wilk [80], duży wkład w to zagadnienie miał Royston [76, 75, 77], a ostatnio Günner i pozostali [31] przedstawili metodę wykorzystującą test Shapiro-Wilka w szybkim przetwarzaniu sygnałów. Należy zauważyć, że dość częstym podejściem jest również operowanie na przybliżeniach wektora $a$. Jedną z najpowszechniej stosowanych zależności jest

$$a \approx \frac{\hat{m}}{(\hat{m}^T \hat{m})^{\frac{1}{2}}}, \tag{1.12}$$

gdzie $\hat{m}_i = \Phi^{-1}((i - 0.375)/(n + 0.25))$.

Przedstawiony powyżej skrócony opis testu Shapiro-Wilka jest zaledwie szkicem problematyki związanej z tym testem i ma jedynie zwrócić uwagę czytelnika na fakt, że problematyka testowania normalności rozkładu jest tematem bardzo rozległym. Dlatego też przed użyciem konkretnych funkcji testujących normalność warto poświęcić czas na dokładniejsze zapoznanie się z własnościami testu, który ma zostać użyty. Szczególną uwagę należy przy tym zwrócić na ograniczenia jakie dana metoda posiada.

### 1.2.3. Rozkład Weibulla

Jak już zostało wspomniane we wstępie, rozkład Weibulla jest przykładem rozkładu modelującego czas życia produktu. Ostatnie lata pokazują jego intensywne wykorzystywanie w zagadnieniach związanych z modelowaniem: wytrzymałości szkła [47], postępującej korozji wżerowej [81], zużycia adhezyjnego metali [73], awaryjności powłok [3], awaryjności materiałów kruchych [28], awaryjności materiałów kompozytowych [67], zużycia elementów betonowych [53], trwałości zmęczeniowej stopów aluminium [33], trwałości zmęczeniowej odlewów Al-Si [1], wytrzymałości włókien z tereftalanu polietylenu [97] lub awaryjności złączy pod wpływem ścinania [7]. To bardzo szerokie spektrum zastosowań nie wiąże się jednak z koniecznością automatycznego przetwarzania dużych zbiorów danych. Pokazuje jednak jak

uniwersalny jest rozkład Weibulla jeśli chodzi o badanie awaryjności i starzenia produktów.

W zupełnie inny sposób rozkład Weibulla został potraktowany w publikacji [16], która pokazuje jak rozkład ten może być wykorzystany do optymalizacji kosztów eksploatacji. Uogólnienie przedstawionego tam podejścia na przykład na wszystkie części zamienne wykorzystywane w dużym przedsiębiorstwie wiąże się już z analizą statystyczną i optymalizacyjną dużego zbioru danych (na ogół liczba wykorzystywanych części zamiennych sięga tysięcy). Innym potencjalnym zastosowaniem może być optymalizacja kosztów serwisowania gwarancyjnego samochodów sprzedawanych przez danego producenta. W takim kontekście mamy do czynienia z jeszcze większą bazą danych, gdyż na ogół model samochodu sprzedawany jest w setkach tysięcy, a niekiedy i milionach egzemplarzy.

Gęstość prawdopodobieństwa trójparametrycznego rozkładu Weibulla zadana jest równaniem

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t-\tau}{\alpha}\right)^{\beta-1} e^{-((t-\tau)/\alpha)^{\beta}}, \quad \alpha, \beta > 0, \tau \geq 0, t \geq \tau. \tag{1.13}$$

Dla tak przyjętych oznaczeń $\alpha$ nazywamy parametrem skali, $\beta$ nazywamy parametrem kształtu, a $\tau$ nazywamy parametrem położenia. Na Rysunku 1.2 przedstawiono przykładowe wykresy funkcji gęstości prawdopodobieństwa $f$ dla rozkładu Weibulla. Jak widać rozkład ten może przybierać bardzo różne formy zależnie od przyjętych parametrów. Funkcja gęstości $f$ pozwala na wyznaczenie dystrybuanty $F$, która w przypadku tego rozkładu przyjmuje następującą postać

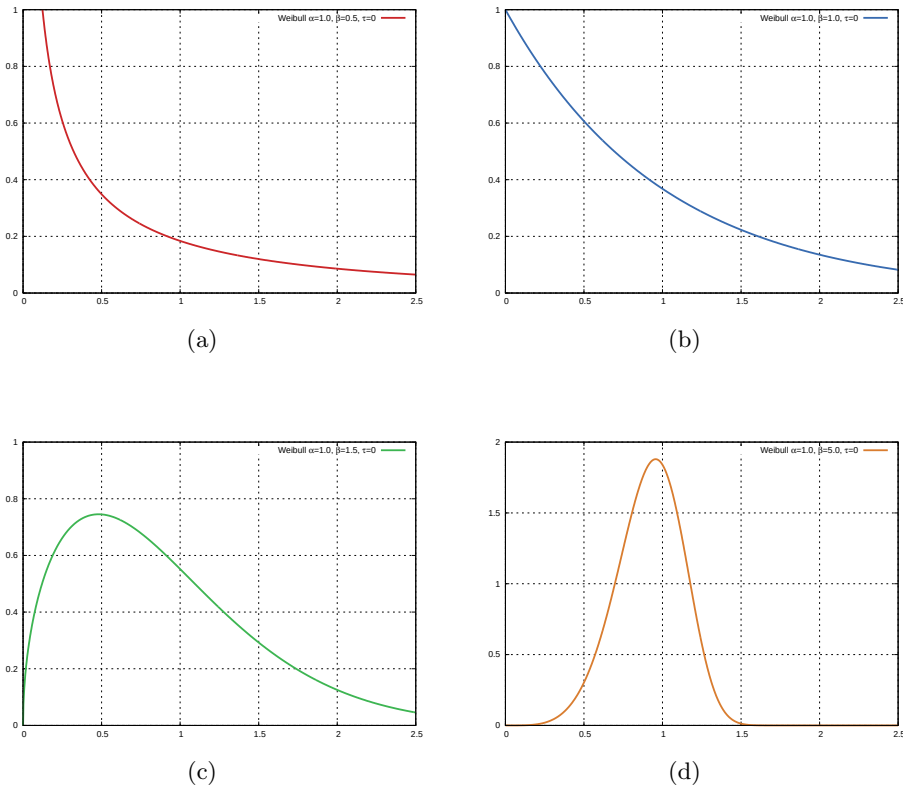
$$F(t) = 1 - e^{-((t-\tau)/\alpha)^{\beta}}. \tag{1.14}$$

Ponadto wartość oczekiwana i wariancja wyrażają się wzorami

$$\mu = \tau + \alpha\Gamma\left(1 + \frac{1}{\beta}\right) \tag{1.15}$$

$$\sigma^2 = \alpha^2\left[\Gamma\left(1 + \frac{2}{\beta}\right) - \Gamma^2\left(1 + \frac{1}{\beta}\right)\right]. \tag{1.16}$$

Ze względu na postać funkcji gęstości $f$, jak i samej dystrybuanty $F$ do wyznaczenia parametrów rozkładu stosowane są najczęściej metody minimalnego prawdopodobieństwa lub regresji liniowej (metoda najmniejszych kwadratów). Przegląd możliwych metod można znaleźć w publikacjach Rossa




(a) (b)

(c) (d)

Rysunek 1.2: Przykładowe wykresy gęstości dla rozkładu Weibulla o określonych parametrach: (a) $\alpha = 1, \beta = 0.5, \tau = 0$, (b) $\alpha = 1, \beta = 1.0, \tau = 0$, (c) $\alpha = 1, \beta = 1.5, \tau = 0$ i (d) $\alpha = 1, \beta = 5.0, \tau = 0$

[74] i Jacquelina [39]. Warto w tym miejscu podkreślić, że postać dystrybuanty rozkładu Weibulla (1.14) daje możliwość zastosowania podstawienia, które przekształca ją w zależność liniową. Dzięki temu w bardzo prosty sposób można powiązać metody oparte na wykresach prawdopodobieństwa [87, 6, 20, 69, 2] z metodą wyznaczania parametrów przy użyciu regresji linowej. Można zatem stwierdzić, że rozkład ten wyjątkowo dobrze wpisuje się w automatyczną analizę dużych zbiorów danych. Przykładowe użycie odpowiednich funkcji bibliotecznych do wyznaczania parametrów rozkładu Weibulla zostanie pokazane w kolejnej części tego rozdziału.

## 1.3. Implementacja metod automatycznej analizy

Część ta poświęcona jest praktycznemu podejściu do wybranych zagadnień analizy dużych zbiorów danych. Poruszone w niej zostały cztery zasadnicze tematy. Pierwszy z nich traktuje o procesie walidacji i przygotowania danych do analizy. Zaś kolejne trzy wykorzystują dodatkowe narzędzia, aby w sposób praktyczny przedstawić podejście do rozwiązania problemów opisanych do tej pory jedynie od strony teoretycznej. Problemami tymi są: analiza wariancji, badanie normalności rozkładu, wyznaczanie parametrów rozkładu Weibulla.

Badania przeprowadzono przy pomocy języka Python oraz pakietu SciPy, który zawiera szereg algorytmów numerycznych. Wybór akurat tego zestawu narzędzi podyktowany był wykorzystaniem języka Python w kolejnych opracowaniach dotyczących predykcji systemów produkcyjnych. W konsekwencji zrezygnowano z wykorzystywania specjalistycznych komercyjnych środowisk. Narzędzia wchodzące w skład pakietu SciPy oferują bardzo duże możliwości analizy, zaś w skład bibliotek SciPy (ang. Scientific Python) wchodzą:

- Numpy - udostępnia klasę ndarray, reprezentującą tablicę o dowolnych wymiarach wraz z szeroką gamą wydajnych funkcji służących do przekształcania macierzy oraz podsumowania zawartych w niej informacji;
- Pandas udostępniający m.in. obiekty DataFrames, w których można przechowywać dane różnego typu, gdzie wiersze odpowiadają obserwacjom, zaś kolumny analizowanym zmiennym. DataFrames umożliwia tworzenie podzbiorów, wczytywanie danych ze źródeł zewnętrznych, grupowania, łączenia czy obliczania podstawowych statystyk;
- Mathplotlib - ściśle powiązany z biblioteką Numpy, generuje wykresy na podstawie macierzy oraz wektorów.

Pakiet SciPy wykorzystuje bibliotekę NumPy do przetwarzania i operacji na wielowymiarowych strukturach danych. Jest podzielony na podmoduły odpowiadające tematycznie wybranej dziedzinie metod numerycznych. Z kolei Pandas został zbudowany na bazie biblioteki NumPy i służy między innymi do wczytywania danych oraz wykonywania dodatkowych operacji na tabelach. Każdy nowo utworzony obiekt DataFrame przechowuje dane w tabelach analogicznie jak dzieje się to w przypadku silników

bazodanowych. Struktury tworzą proste w zastosowaniu podstawy wspomagające odkrywanie wiedzy. Dostępne metody pozwalają na wykonywanie szeregu operacji: grupowania, modyfikacji, przekształceń kolumn, wykonywania zapytań oraz generowania wykresów. Oprócz tego, dostępny jest obiekt serii, który jest jednowymiarowym obiektem przypominającym tablicę składającą się z listy wartości oraz indeksu. Warto zaznaczyć, że obiekt DataFrame może zostać także użyty do reprezentacji zbioru o większej liczbie wymiarów, udostępniając zaawansowane funkcje ich obsługi.

### 1.3.1. Czyszczenie danych, przygotowanie do analizy

Skuteczna analiza musi być oparta na podstawie poprawnych i odpowiednio przygotowanych danych. W związku z tym dane wymagają wstępnego przetworzenia (Data Pre-processing). Celem tego etapu jest walidacja poprawności i kompletności zbioru, wybór przypadków, ich odpowiedni podział oraz wstępna obróbka.

Czyszczenie danych (ang. data cleaning data cleansing, data scrubbing) to proces wyszukiwania i usuwania błędów oraz niespójności w celu polepszenia ich jakości. Proces ten jest niezbędny, ponieważ analiza błędnych lub niespójnych danych może prowadzić do wyciągania niepełnych lub błędnych wniosków.

Do operacji czyszczenia danych zalicza się:

- Kontrolę danych pod kątem wartości niezgodnych z rzeczywistością. Niezgodności te mogą wynikać z błędów ich zapisu lub ekstrakcji, błędów związanych z łączenia wielu źródeł lub systemów pomiarowych. Wykrywanie tego typu błędów odbywa się na podstawie porównania wartości odstających od typowych wartości lub skontrolowanie ich pod względem logicznym oraz zbadanie ich rozkładów. Błędne dane zastępuje się poprawną wartością, wartością średnią z sąsiadujących przypadków lub wartością wynikającą z modelu opisującego tę zmienną.
- Wykrywanie pustych wartości, czyli niekompletności zbioru może wynikać z wielu przyczyn. Jedną z nich może być brak danych lub ich utrata podczas ekstrakcji. Tego typu przypadki można w analizie pominąć, uzupełnić je na podstawie innych źródeł, zastąpić średnią z sąsiadujących wartości lub wyznaczyć ich wartości na podstawie procedur opisujących daną zmienną.

- Przeszukiwanie szeregów pod kątem danych odstających i zaszumionych to proces polegający na walidacji pod kątem występowania błędnych wartości. Wartości odstające można wykrywać za pomocą testów, analizy reszt modelu lub przy pomocy wykresów. Niemniej jednak ich ewentualna modyfikacja musi być bardzo przemyślana. Może się bowiem okazać, że obserwowany zbiór nie jest wcale nieprawidłowością, a realnym pomiarem, który świadczy o różnego rodzaju problemach.
- Identyfikacja danych anachronicznych to poprawianie błędnych lub wypełnianie pustych wartości na podstawie domysłu.

Informacje w przedsiębiorstwie udostępniane są w różnych formatach. Niektóre z nich mają ściśle określoną strukturę, z której relatywnie łatwo można wyciągnąć interesujące nas wartości. Takimi źródłami są na przykład automatyczne logi zapisywane w standardzie JSON, CSV i XML. Jednakże dane mogą być także pozyskiwane w sposób mniej usystematyzowany – na przykład z plików PDF. Tego rodzaju baza została właśnie wykorzystana na potrzeby niniejszego opracowania. Pliki PDF są często wykorzystywane jako wydruki i wydają się być wygodne w użyciu. Niestety, o wiele trudniej analizować je pod kątem zawartości i przetworzyć istotne dane do postaci tabelarycznej niż w przypadku plików tekstowych. Warto także zauważyć, że pliki PDF przechowują informacje w formacie binarnym, co czyni je o wiele bardziej złożonymi w porównaniu do plików tekstowych. Pliki te, oprócz danych tekstowych przechowują także informacje o czcionce oraz układzie tekstu, stąd szybkość ich przetwarzania zależy między innymi od ich zawartości.

Wstępna ekstrakcja danych z kilkuset plików PDF do pliku csv wymagała częściowego zautomatyzowania procesu przetwarzania plików źródłowych. Niezbędne okazały się także działania polegające na analizie zbiorów przy pomocy skryptów i wyszukiwanie anomalii w seriach na podstawie wartości granicznych dla każdej z dostępnych zmiennych. Każda iteracja przetwarzania była zakończona wielokrotną walidacją pod kątem błędów oraz występowania wartości pustych i wyboru dalszego postępowania w tego typu przypadkach.

Dla języka Python istnieje kilka bibliotek, za pomocą których można zautomatyzować proces ekstrakcji danych z formatu PDF. Interesująca jest biblioteka PyPDF2, która sprawdziła się dla plików zawierających dane tabelaryczne, choć biblioteka okazała się być niewystarczająca. Napisany al-

gorytm ekstrakcji oparty o bibliotekę PyPDF2 zawierał dodatkowo funkcje porządkujące oraz walidujące. Proces ekstrakcji wykazywał wiele błędów w przypadkach, gdy przetwarzany dokument źródłowy zawierał inną strukturę tabel. W przypadku danych przetwarzanych na potrzeby analizy, czas niezbędny na wstępne automatyczne przetworzenie wszystkich dokumentów z formatu PDF do pliku CSV, wyniósł około jednej godziny.

Kolejnym etapem była ostateczna walidacja i ocena zbioru, dokładniejsze czyszczenie oraz anonimizacja danych. Jednocześnie w tym kroku rozwiązano problem pustych wartości, gdyż niektóre funkcje statyczne wykonywane w procesie analizy danych są wrażliwe na tego typu wystąpienia. W literaturze przedmiotu znaleźć można trzy główne metody wypełniania pustych wartości [68]:

- imputacja (wypełnianie danych na podstawie obserwacji zbioru);
- interpolacja (szacowanie wartości na podstawie punktów sąsiednich, będąca także jedną z metod imputacji);
- usuwanie wybrakowanych fragmentów.

Po wstępnej analizie zbioru danych przyjęto ostatnią z metod, ponieważ zbiór obserwacji był kompletny, zaś puste wartości znajdowały się w wierszach, które w istocie były nieprzydatne w dalszej analizie.

Wczytanie zbioru z pliku CSV do DataFrame odbywa się przy użyciu metody read_csv(), która jako podstawowy argument przyjmuje ścieżkę do pliku zawierającego rozdzielone znakami wartości (domyślnie są to przecinki). Po załadowaniu do pamięci plik jest zwracany jako dwuwymiarowa struktura z oznaczonymi osiami.

Metoda read_csv() może przyjmować więcej niż jeden argument. Dodatkowe argumenty umożliwiają zmianę zachowania algorytmu odczytu i formatowania. Na Rysunku 1.3 użyto parametru usecols, do którego przekazano listę kolumn do importu. Dzięki temu, import może dotyczyć wyłącznie wskazanych kolumn, co skraca czas i zmniejsza wykorzystanie pamięci w przypadku większych zbiorów danych.

```
import numpy as np
import pandas as pd
kolumny = ["Data", "Day", "Maszyna", "Operator", "P1", "P2", "P3", "P4", "P5", "
    P6", "P7", "P8", "P9", "P10", "P11", "P12", "P13", "P14", "P15", "P16", "P17
    ", "P18", "P19", "P20"]
df = pd.read_csv("new.csv", squeeze=True, decimal=".", delimiter=";",
    warn_bad_lines=True, usecols = kolumny)
```

Rysunek 1.3: Wczytanie zbioru danych do DataFrame

Pozostałe istotne argumenty dla metody read_csv(), to:

- squeeze - zwraca serię danych w przypadku plików, które zawierają wyłącznie jedną kolumnę;
- decimal - znak rozdzielania miejsc dziesiętnych w pliku;
- delimiter (alias: sep) - znak rozdzielający kolumny w pliku. Domyślnym znakiem jest przecinek, jednakże w przypadku zastosowania innego znaku rozdzielającego, algorytmy odczytu mogą niepoprawnie zinterpretować plik. Ponadto pliki źródłowe mogą także zawierać separatory o rozmiarze dłuższym niż 1 znak, co w konsekwencji może zostać błędnie zinterpretowane jako wyrażenia regularne;
- error_bad_lines - zgłoszenie wyjątku w przypadku napotkania przez interpreter wierszy, które zawierają zbyt wiele pól. Spowoduje to, że obiekt nie zostanie wypełniony danymi. Wartość False parametru oznacza, że błędne linie nie zostaną pominięte podczas odczytu;
- warn_bad_lines - jeśli wartość argumentu error_bad_lines została ustawiona na False, zaś warn_bad_lines na True, to zostanie wyświetlone ostrzeżenie dla każdego wiersza, który zawiera błąd;
- usecols - selektywne wskazanie kolumn, które mają zostać odczytane;
- header - sposób postępowania z nagłówkiem pliku (może on zawierać nazwy kolumn). Domyślnie metoda rozpoznaje nazwy kolumn na podstawie pierwszego wiersza, jednakże przekazanie wartości header=None spowoduje, że nagłówek zostanie pominięty podczas odczytu;
- skiprowslist - liczba początkowych linii do pominięcia od indeksu zaczynającego się od 0. Argument może zostać przekazany jako funkcja anonimowa, np. x: x in [0, 2];

- parse_dates - parametr domyślnie wyłączony. Włączenie pozwala na identyfikację i parsowanie dat z czytanego pliku poprzez analizę kolumn zawierających datę/godzinę. Opcja ta umożliwia także konkatenację daty lub daty i czasu z kilku kolumn źródłowych;
- dtype - wymuszenie określonych typów danych dla wskazanych kolumn, np. 'a': np.float64, 'b': np.int32, 'c': 'Int64';
- compression - obsługa odczytu plików skompresowanych. Domyślnie wykrywana jest kompresja na podstawie rozszerzenia pliku (.gz, .bz2, .zip). Wartość None oznacza brak kompresji;
- thousand - przypisanie separatora tysięcy;
- low_memory - opcja domyślnie włączona, oznaczająca przetwarzanie pliku porcjami, powodująca mniejsze zużycie pamięci podczas jego analizy. W przypadku typów mieszanych warto ustawić opcję na False i wymusić typowanie kolumn przy pomocy parametru dtype;
- nrows - argument, który umożliwia ograniczenie liczby wczytanych linii. Przydatny podczas pracy ze zbiorami charakteryzującymi się dużym rozmiarem.

Po wczytaniu zbioru do pamięci, obiekt DataFrame udostępnia badaczowi kilka przydatnych metod kontrolnych. Na przykład bezpośrednio po wczytaniu pliku można użyć metod head() lub tail(), które wyświetlą domyślnie pięć pierwszych lub pięć ostatnich wierszy w postaci tabelarycznej. Wyświetlanie próbki zbioru pozwala na szybki wgląd w jego strukturę oraz wstępną ocenę jakości próbki lub populacji. Przekazując opcjonalny parametr w postaci liczby całkowitej do wymienionych metod, można zwiększyć liczbę wyświetlanych wierszy w przypadku, gdy zajdzie potrzeba szerszego spojrzenia na analizowany zbiór.

```
1 Index Data  Day Maszyna Operator P1  P2 P3 ... P17   P18    P19      P20
2 0 2020-05-01 7  K6  OPER5 7.27 0.109 0.001 ... 0.015 0.0 92.217
3 1 2020-04-01 6  K1  OPER4 7.06 0.108 0.001 ... 0.0005  0.013  0.0  92.416
4 2 2020-01-19 7  K7  OPER4 7.25 0.111 0.001 ... 0.0005  0.016  0.0  92.218
5 3 2020-09-01 4  K10 OPER5 7.22 0.102 0.001 ... 0.0005  0.014  0.0  92.274
6 4 2020-12-01 7  K3  OPER4 7.21 0.091 0.001 ... 0.0005  0.016  0.0  92.280
```

Rysunek 1.4: Wyświetlenie początku zbioru danych przy pomocy metody head()

Metoda tail() wykonana na obiekcie DataFrame wyświetli domyślnie pięć ostatnich wierszy zbioru z DataFrame.

```
Index Data Day  Maszyna Operator P1   P2   P3   ...     P17   P18   P19 P20
38595 2020-12-14 1 K6  OPER4 6.99   0.089   0.001 ... 0.0005   0.016   0.0   92.505
38596 2020-12-15 2 K15 OPER4 7.10   0.093   0.003 ... 0.0005   0.019   0.0   92.360
38597 2020-03-12 4 K9 OPER5 7.27   0.097   0.001 ... 0.0005   0.013   0.0   92.224
38598 2020-12-22 2 K14 OPER15 7.22   0.108   0.001 ... 0.0005   0.013   0.0   92.275
38599 2020-12-14 1 K10 OPER6 7.19   0.090   0.001 ... 0.0005   0.014   0.0   92.312
```

Rysunek 1.5: Lista ostatnich wierszy wyświetlona przy użyciu metody tail()

Każdy utworzony obiekt DataFrame posiada unikatowy indeks, który umożliwia jednoznaczną identyfikację każdego wiersza. W przypadku, gdy w sposób jawny nie zdefiniowano kolumny indeksu, tworzona jest automatycznie dodatkowa kolumna o nazwie index (typ numeryczny), identyfikująca każdy wiersz. Zamiast domyślnego, można definiować własne indeksy, którymi może być kolumna zbioru zawierająca datę lub ciąg tekstowy. Tworzenie własnych indeksów wykorzystywane jest między innymi przez część wbudowanych funkcji analizy i wizualizacji.

Przy pomocy metody tail() można uzyskać informację o liczbie obserwacji za pomocą automatycznie utworzonej kolumny index, która w przypadku badanego zbioru wynosi 38599 (ostatni element zbioru). W przypadku zbiorów zawierającego większą liczbę kolumn, widok nie wyświetli ich wszystkich, lecz wyłącznie kolumny skrajne. Aby poznać dokładny kształt zbioru danych, można użyć polecenia df.shape, co zwróci w odpowiedzi informacje na temat kształtu w postaci liczby obserwacji (wierszy) oraz kolumn (38600, 24).

Po wstępnym zapoznaniu się ze strukturą i zawartością zbioru, należy zbadać wiersze w kolumnach pod kątem ewentualnych braków, czyli występujących wartości NaN (not a number). W tym celu można wykorzystać polecenie pd.isnull(df). Automatyzację procesu walidacji pod tym kątem realizuje w sposób kompleksowy program z Rysunku 1.6.

```
ile_puste = pd.DataFrame(df.isnull().any(), columns=['Nulls'])
ile_puste['Pustych'] = pd.DataFrame(df.isnull().sum())
ile_puste['Pustych %'] = round((df.isnull().mean()*100),2)
print(ile_puste)
```

Rysunek 1.6: Analiza zbioru pod kątem występowania pustych wartości

Wywołanie kodu z Rysunku 1.6 uruchomi analizę wartości wierszy w kolumnach obiektu DataFrame pod kątem występowania braków, wyświetlając w efekcie podsumowanie. Raport zawiera wyniki zawierające listę nazw kolumn, wartości logiczne informujące o występowaniu wartości NaN

w wierszach danej kolumny oraz liczbę bezwzględną i procentową wartości pustych dla każdej z kolumn.

| | Nulls | Pustych | Pustych % |
|---|---|---|---|
| Data | False | 0 | 0.0 |
| Day | False | 0 | 0.0 |
| Maszyna | False | 0 | 0.0 |
| Operator | False | 0 | 0.0 |
| P1 | False | 0 | 0.0 |
| P2 | False | 0 | 0.0 |
| P3 | False | 0 | 0.0 |
| P4 | False | 0 | 0.0 |
| P5 | False | 0 | 0.0 |
| P6 | False | 0 | 0.0 |
| P7 | False | 0 | 0.0 |
| P8 | False | 0 | 0.0 |
| P9 | False | 0 | 0.0 |
| P10 | False | 0 | 0.0 |
| P11 | False | 0 | 0.0 |
| P12 | False | 0 | 0.0 |
| P13 | False | 0 | 0.0 |
| P14 | False | 0 | 0.0 |
| P15 | False | 0 | 0.0 |
| P16 | False | 0 | 0.0 |
| P17 | False | 0 | 0.0 |
| P18 | False | 0 | 0.0 |
| P19 | False | 0 | 0.0 |
| P20 | False | 0 | 0.0 |

Rysunek 1.7: Informacja o wartościach pustych w obiekcie DataFrame

Analizując rezultat działania programu (Rysunek 1.7), zauważyć można pozytywny wynik walidacji. Tego typu efekt dobrze świadczy o wstępnym przygotowaniu wyników do badań. W przypadku negatywnej walidacji, ewentualne braki zostaną wyświetlone w kolumnach 3 i 4. Choć w niektórych przypadkach wiersze zawierające puste wartości można po prostu pominąć, często zachodzi potrzeba wykonania dodatkowych czynności polegających na wypełnieniu pustych wartości za pomocą odpowiednich algorytmów. Na koniec należy jeszcze raz wykonać walidację i upewnić się, że wartości puste zostały uzupełnione.

Procesy walidacyjne muszą obejmować także testy pod kątem typów, które mogły zostać narzucone automatycznie w sposób niepoprawny, bądź mało efektywny już podczas wczytywania pliku. Informację na temat przypisania typów można uzyskać przy pomocy polecenia dtypes, które wyświetli raport na temat automatycznie przypisanych typów zmiennych do poszczególnych kolumn. Wynikiem działania operacji df.dtypes jest widok 1.8, który zawiera wykaz kolumn wraz z informacją na temat przypisanych do nich typów.

```
Data          object
Day            int64
Maszyna       object
Operator      object
P1           float64
P2           float64
P3           float64
P4           float64
P5           float64
P6           float64
P7           float64
P8           float64
P9           float64
P10          float64
P11          float64
P12          float64
P13          float64
P14          float64
P15          float64
P16          float64
P17          float64
P18          float64
P19          float64
P20          float64
dtype: object
```

Rysunek 1.8: Lista wszystkich kolumn i przypisanych typów

Obserwacja wyników z raportu 1.8 potwierdza, że kolumny P1..P20 posiadają poprawnie przypisane typy numeryczne. Jednakże potencjalny problem znajduje się w kolumnie Data, która została zadeklarowana jako typ object. W przypadku, gdy zbiór podlegać będzie filtrowaniu na podstawie dat lub też praca ze zbiorem wymaga typu data/czas, konieczna jest konwersja przy pomocy dodatkowego przypisania: df['Data']= pd.to_datetime( df['Data']) - bezpośrednio pod wierszem zawierającym instrukcję importującą. Ponowne wywołanie polecenia dtypes powinno potwierdzić zmianę typu kolumny Data na właściwy typ: datetime64[ns].

W przypadku analizy dużych zbiorów, należy liczyć się z dużym zużyciem pamięci. W tego typu przypadkach dodatkową czynnością wstępnego przygotowania zbioru do analizy może być optymalizacja typów numerycznych. Informację o aktualnym wykorzystaniu pamięci przez zbiór można uzyskać przy pomocy polecenia df.info(memory_usage='deep'):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38600 entries, 0 to 38599
Data columns (total 24 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Data       38600 non-null  datetime64[ns]
 1   Day        38600 non-null  int64
 2   Maszyna    38600 non-null  object
 3   Operator   38600 non-null  object
 4   P1         38600 non-null  float64
 5   P2         38600 non-null  float64
 6   P3         38600 non-null  float64
 7   P4         38600 non-null  float64
 8   P5         38600 non-null  float64
 9   P6         38600 non-null  float64
 10  P7         38600 non-null  float64
 11  P8         38600 non-null  float64
 12  P9         38600 non-null  float64
 13  P10        38600 non-null  float64
 14  P11        38600 non-null  float64
 15  P12        38600 non-null  float64
 16  P13        38600 non-null  float64
 17  P14        38600 non-null  float64
 18  P15        38600 non-null  float64
 19  P16        38600 non-null  float64
 20  P17        38600 non-null  float64
 21  P18        38600 non-null  float64
 22  P19        38600 non-null  float64
 23  P20        38600 non-null  float64
dtypes: datetime64[ns](1), float64(20), int64(1), object(2)
memory usage: 10.9 MB
```

Rysunek 1.9: Podsumowanie zużycia pamięci RAM przez zbiór

Jak wynika z raportu 1.9, wczytany zbiór zajmuje stosunkowo niewiele pamięci RAM (około 11 MB). W tym wypadku dalsza optymalizacja typów nie jest konieczna. Jednakże przetwarzanie większych zbiorów wymaga przede wszystkim zapoznania się z dostępnymi w pakiecie typami w celu potencjalnych zmian typów kolumn. Część dostępnych typów zmiennych nie jest ściśle związana z pakietem Pandas. W rzeczywistości bowiem ich źródła pochodzą z pakietu NumPy. Dotyczy to typów numerycznych int i float, typów dotyczących czasu: timedelta64[ns] i datetime64[ns] oraz typu logicznego bool. Warto podkreślić, że pochodzące z pakietu Numpy typy int oraz float z NumPy nie są także równoważne klasom znanym z języka Python. Zostały one bowiem zaimplementowane bezpośrednio z języka C. Jeśli zaś mowa o typach tekstowych, są one przechowywane w pamięci jako obiekt. Ze względu na przeznaczenie pakietu związane z analizą zbiorów danych, większość przetwarzanych wartości zadeklarowana jest jako typy numeryczne. Typy te posiadają własne podtypy, charakteryzujące się innym zapotrzebowaniem na pamięć. W przypadku float, którego typem domyślnym jest float64, do dyspozycji pozostają podtypy float16 i float32. Liczba w nazwie odnosi się bezpośrednio do liczby bitów wykorzystanych do przechowywania zmiennej w pamięci. Z kolei typ całkowity int64 posiada następujące podtypy: int8, int16 oraz int32. Z kolei typ int8 może przecho-

wywać wartości całkowite z zakresu od -128 do 127. Tabela 1.1 zawiera zestawienie zbiorcze dostępnych typów oraz konieczny do ich przechowywania rozmiar pamięci.

Tablica 1.1: Tabela dostępnych typów zmiennych i ich rozmiar w pamięci

| Typ | Rozmiar (bajty) | Zakres | Użycie |
|---|---|---|---|
| bool | 1 | 0;1 | prawda; fałsz |
| int8 | 1 | -128; 127 | liczby całkowite |
| int16 | 2 | -32768; 32767 | liczby całkowite |
| int32 | 4 | -2147483648; 2147483647] | liczby całkowite |
| int64 | 8 | -9223372036854775808; 9223372036854775807 | liczby całkowite |
| uint8 | 1 | 0; 255 | liczby naturalne |
| uint16 | 2 | 0; 65535 | liczby naturalne |
| uint32 | 4 | 0; 4294967295 | liczby naturalne |
| uint64 | 8 | 0; 18446744073709551615 | liczby naturalne |
| float16 | 2 | - 6.55040e+04; 6.55040e+04 | liczby rzeczywiste |
| float32 | 4 | - 3.4028235e+38; 3.4028235e+38 | liczby rzeczywiste |
| float64 | 8 | -1.7976931348623157e+308; 1.7976931348623157e+308 | liczby rzeczywiste |
| datetime | 8 | – | data i czas |
| object | zmienny | – | tekst |
| category | zmienny | – | kategorie |

Przeprowadzanie badania zbiorów o dużej liczbie obserwacji, może doprowadzić do sytuacji, w której pamięć urządzenia okaże się być niewystarczająca. Jako jedną z metod radzenia sobie z tego typu problemem, wyróżnić można wykorzystanie mechanizmu wymuszania podtypów podczas importowania zbioru do pamięci RAM. Operacja ta może przynieść korzyści w postaci mniejszego zapotrzebowania na pamięć, a w konsekwencji - możliwość analizy zbiorów o większej liczbie obserwacji. Podczas dokonywania tego typu konwersji, istotne jest, by graniczne wartości zmiennych mieściły się w zakresie danego podtypu (Tabela 1.1). Konwersji na podtypy można dokonać także po zaimportowaniu. W tym wypadku konwersja kolumny polega na użyciu metody astype(). Jeszcze inną przyczyną konwersji są przypadki, w których z jakiegoś powodu kolumna została zadeklarowana jako typ object, podczas gdy oczekiwanym typem był typ numeryczny. Typ object jest powszechnie wykorzystywany przez pakiet do przechowywania ciągów znaków. Zatem w przypadku, gdy mamy do czynienia z konwersją z typu object do typu numerycznego, można tego dokonać konwertując go do typu numerycznego (Rysunek 1.10).

```
1 df['string_kol'] = df['string_kol'].astype('int')
2 lub
3 df['string_kol'] = pd.to_numeric(df['string_kol'], errors='coerce')
```

Rysunek 1.10: Konwersja typu znakowego do numerycznego (całkowitego)

Warto zwrócić uwagę na występujący w metodzie to_numeric() para-

metr errors='coerce', który modyfikuje algorytm konwersji w taki sposób, że nieprawidłowo rozpoznane wartości zostaną automatycznie wypełnione wartością NaN.

Konwersji typu znakowego do typu numerycznego (zmiennoprzecinkowego) można dokonać na kilka sposobów. Analogicznie jak w przypadku konwersji do typu całkowitego, do konwersji zmiennoprzecinkowej zmienia się jedynie parametr metody astype('float'). Podczas konwersji zostanie automatycznie przypisany 64-bitowy typ zmiennoprzecinkowy. Dzięki elastyczności algorytmów konwersji w metodzie astype(), można wpływać na precyzję docelowej zmiennej, dopisując jako parametr odpowiedni podtyp. W rezultacie wybierając typ 'float128', można wpływać na zwiększenie precyzji zmiennej, zaś wybierając 'float16' - wpływać na optymalizację pamięci RAM (Rysunek 1.11).

```
1 df['string_kol'] = df['string_kol'].astype('float')
2 df['string_kol'] = df['string_kol'].astype('float128')
3 df['string_kol'] = df['string_kol'].astype('float16')
```

Rysunek 1.11: Konwersja typu znakowego na typ numeryczny (zmiennoprzecinkowy) z uwzględnieniem podtypów

Konwersja typów kolumn, w których znajdują się typy mieszane, jest nieco problematyczna. Nie można bowiem zakładać, że mieszana zawartość zostanie poprawnie skonwertowana. Dla przykładu pole może zawierać wartość tekstową + numeryczną, co uniemożliwia bezpośrednią konwersję na typ numeryczny. Co więcej - próba zmiany typu może w takich przypadkach doprowadzić do sytuacji, w której wartości mieszane zostaną skonwertowane do wartości NaN. Z tego powodu wymagane jest dodatkowe działanie, polegające na usunięciu wartości tekstowej przed dokonaniem konwersji. Inną metodą radzenia sobie z kolumnami o zawartości mieszanej jest rozdzielenie wartości tekstowej na część numeryczną oraz tekstową i przypisanie ich do kilku kolumn. Pozwala to na zachowanie obu wartości w przypadku, gdy obie są używane podczas badań.

Analizowany zbiór zawiera przypadek zmiennej o zawartości mieszanej. Jest to kolumna Maszyna (wartości K1..K20). Kolumna zawiera informacje o maszynie produkcyjnej, do której przypisane zostały wykorzystane wartości parametrów w procesie produkcyjnym oraz operator. W tym przypadku wartości z kolumny służą wyłącznie grupowaniu wyników na wykresie, zatem dalsze operacje nie są konieczne. Mimo to, niektóre z algorytmów, jako

argumenty przyjmują wyłącznie zmienne numeryczne i w takich przypadkach kolumna może wymagać konwersji zawartości mieszanej do wartości numerycznej. Konwersję tego typu kolumn można przeprowadzić na kilka sposobów. Pierwszym z nich jest użycie metody astype() jednocześnie z metodą replace():

```
df['Maszyna'] = df['Maszyna'].str.replace('K', '').astype('int16')
print(df.dtypes)
```

Rysunek 1.12: Konwersja typu mieszanego do int16 przy użyciu metody astype()

Konwersja kolumn zawierających wartości mieszane do postaci numerycznej może zostać także przeprowadzona przy wykorzystaniu funkcji anonimowej (lambda), co stanowi zwięzłą alternatywę czyszczenia i konwersji problematycznych wartości (Rysunek 1.13).

```
df['Maszyna'] = df['Maszyna'].apply(lambda v: v.replace('K','').replace(',',''))
    .astype('int16')

lub bardziej precyzyjnie:

df['Maszyna'] = df['Maszyna'].apply(lambda x: x.replace('K', '').replace('k', ''
    )
                                if isinstance(x, str) else x).astype('int16')
```

Rysunek 1.13: Konwersja typu mieszanego do int16 za pomocą funkcji lambda

Bardziej wyrafinowaną metodą konwersji jest użycie wyrażeń regularnych. Pozwala to na usunąć zbędne znaki nienumerycznych z ciągu w przypadku bardziej skomplikowanych ciągów znaków. Metoda ta wykorzystuje również replace(), jednakże w połączeniu z wyrażeniem regularnym.

```
df['Maszyna'] = df['Maszyna'].replace({'\$': '', 'K': ''}, regex=True).astype('
    int16')
print(df.dtypes)
```

Rysunek 1.14: Konwersja wartości mieszanej do typu numerycznego przy użyciu wyrażeń regularnych i metody replace()

W przypadku pracy z plikami o dużym rozmiarze, aby zmniejszyć rozmiar pliku, można wykorzystać kompresję. Pandas zawiera obsługę automatycznej dekompresji plików. Można także włączyć konkretny rodzaj kompresji, poprzez dodanie parametru compression=rodzaj w metodzie read_csv().

Wracając do metody importu zbioru i zmniejszenia wykorzystania pamięci, na etapie odczytu można dokonać jawnego typowania kolumn, wykorzystując dodatkowe parametry metody read_csv(). Dzięki temu, podczas importu pliku do obiektu DataFrame, zostaną przypisane odpowiednio wybrane typy zmiennych, co zostało zrealizowane na Rysunku 1.15.

```
df = pd.read_csv('dane.gz', decimal='.', sep = ',', compression = 'gzip',
dtype = {'P1':np.float32, 'P2':np.float32, 'P3':np.float64 , 'P5':np.float32, '
    Day':np.uint8})
```

Rysunek 1.15: Konwersja typów na etapie importu

Wczytanie zbioru, konwersja typów i przeprowadzone czyszczenie zbioru pozwala na pobranie informacji statystycznych dotyczących przetwarzanych danych. W tym celu można posłużyć się metodą describe(), która wygeneruje podstawowe informacje na podstawie zmiennych numerycznych. W rezultacie dla każdej kolumny numerycznej wyświetlone zostaną następujące informacje: liczba obserwacji, średnia, odchylenie standardowe, wartości minimalne oraz maksymalne, mediana, drugi (dolny) kwartyl oraz trzeci (górny) kwartyl.

```
 Day     P1     P2     P3     P4   ...    P18   P19    P20
count   38600   38600   38600   38600   ...   38600   38600   38600
mean   3.964896   7.239774   0.098265   0.001866   ...   0.015582   0.0   92.246311
std   1.990908   0.766107   0.008221   0.001870   ...   0.001874   0.0   0.761936
min 1.000000 5.480000   0.049000   0.001000   ...   0.007000   0.0   87.751000
25%   2.000000   6.980000   0.093000   0.001000   ...   0.014000   0.0   92.269000
50%   4.000000   7.090000   0.098000   0.001000   ...   0.015000   0.0   92.390000
75%   6.000000   7.210000   0.104000   0.002000   ...   0.017000   0.0   92.512250
max   7.000000   11.745000   0.180000   0.025000   ...   0.073000   0.0   94.891000
[8 rows x 21 columns]
```

Rysunek 1.16: Informacje statystyczne o zbiorze

Podsumowanie statystyczne wykonane za pomocą metody describe(), której wyniki zostały przedstawione na Rysunku 1.16, stanowi punkt wejściowy dla dalszej analizy.

Analiza procesów produkcyjnych wymagać może zbadania wartości granicznych, takich jak np. wartość minimalna i maksymalna serii. Badanie rozstępu można przeprowadzić przy użyciu metod min() oraz max(), zaś wyniki obliczeń można przypisać zarówno jako nowe kolumny, jak i do nowego obiektu. Dla przykładu, aby dodać nowe kolumny zawierające wartości minimalne oraz maksymalne do istniejącego obiektu, można zapisać:

```
df["minP1"] = df["P1"].min()
df["maxP1"] = df["P1"].max()
df.head()
```

Rysunek 1.17: Obliczanie wartości minimalnej oraz maksymalnej dla pojedynczej kolumny

Jeśli zajdzie konieczność wyznaczania wartości dla większej liczby kolumn, należy dodać je do obiektu DataFrame, a następnie przypisać im wartości obliczone z wybranej kolumny lub wiersza przy pomocy odpowiedniej metody:

```
df["minP1"] = df["P1"].min()
df["maxP1"] = df["P1"].max()
...
df["minP10"] = df["P20"].min()
df["maxP10"] = df["P20"].max()
df.head()
```

Rysunek 1.18: Obliczanie wartości minimalnej oraz maksymalnej dla wielu kolumn

Wynikiem działania programu z Rysunku 1.18 będzie utworzenie dodatkowych kolumn, w których zostaną umieszczone wartości minimalne oraz maksymalne. Analogicznie dodawać można wartości średnie (metoda mean()). Obliczone wartości można także przypisać do zmiennych, co pozwoli na wykorzystanie ich we własnych algorytmach. Wartości można również grupować, co pozwala na uzyskanie wyników z podziałem na grupy. Fragment programu z Rysunku 1.19 demonstruje jedną z metod grupowania, która umożliwia uzyskanie wartości parametrów P1 wraz z podziałem na maszyny, na których odbywał się proces produkcji. Dodatkowo zaimplementowano obliczenia statystyczne dla każdej z wykorzystanych maszyn w badanym procesie.

```
import numpy as np
import pandas as pd
kolumny = ["Maszyna","P1"]
df = pd.read_csv("new.csv", decimal=".", delimiter=";", usecols = kolumny)
df_machines = df.groupby("Maszyna").agg([np.min, np.max, np.mean, np.var, np.std
    ])
```

Rysunek 1.19: Wykaz parametrów z podziałem na maszyny produkcyjne

Program z Rysunku 1.19 oblicza wartości minimalne, maksymalne i średnie oraz wariancję i odchylenie standardowe, grupując wyniki ze względu na wykorzystane maszyny. Dokonując obliczeń, przypisuje wyniki do nowej

zmiennej df_machines, która jest w istocie kolejnym obiektem typu DataFrame.

| | P1 | | | | |
|---|---|---|---|---|---|
| | amin | amax | mean | var | std |
| Maszyna | | | | | |
| K1 | 5.48 | 10.375 | 7.066938 | 0.044323 | 0.210529 |
| K10 | 6.15 | 10.842 | 7.083978 | 0.050787 | 0.225359 |
| K11 | 6.50 | 10.956 | 7.084999 | 0.061072 | 0.247127 |
| K12 | 6.51 | 11.580 | 7.607481 | 1.644585 | 1.282414 |
| K13 | 6.50 | 10.531 | 7.087212 | 0.063801 | 0.252589 |
| K14 | 6.50 | 10.967 | 7.087358 | 0.091063 | 0.301767 |
| K15 | 6.52 | 7.500 | 7.068180 | 0.028980 | 0.170236 |
| K16 | 6.52 | 11.461 | 7.660511 | 1.756858 | 1.325465 |
| K17 | 7.00 | 7.140 | 7.063333 | 0.005033 | 0.070946 |
| K18 | 7.01 | 7.190 | 7.092500 | 0.009225 | 0.096047 |
| K19 | 6.94 | 7.060 | 6.993333 | 0.003733 | 0.061101 |
| K2 | 5.68 | 11.093 | 7.100629 | 0.140454 | 0.374771 |
| K20 | 6.91 | 7.070 | 6.996667 | 0.006533 | 0.080829 |
| K3 | 6.00 | 11.634 | 7.372176 | 1.004159 | 1.002077 |
| K4 | 6.05 | 11.338 | 7.306324 | 0.797516 | 0.893037 |
| K5 | 6.50 | 10.419 | 7.083502 | 0.040428 | 0.201067 |
| K6 | 6.22 | 11.681 | 7.414546 | 1.120706 | 1.058634 |
| K7 | 6.42 | 11.075 | 7.088020 | 0.067986 | 0.260740 |
| K8 | 6.22 | 11.369 | 7.307899 | 0.796399 | 0.892412 |
| K9 | 6.50 | 11.745 | 7.408051 | 1.049051 | 1.024232 |

Rysunek 1.20: Wynik wykonania polecenia z Rysunku 1.19

```
import numpy as np
import pandas as pd
df = pd.read_csv("new.csv", decimal=".", delimiter=";")
df['Data']= pd.to_datetime(df['Data'])

df = df[df["Data"].isin(pd.date_range("2020-05-01", "2020-05-30"))]

df.drop('Day', axis=1, inplace=True)

print('Zbiór zawiera {} obserwacji i {} zmienne.'.format(df.shape[0], df.shape
    [1]))
```

Rysunek 1.21: Filtrowanie danych na podstawie zakresu dat

Wykorzystany zbiór zawiera znaną liczbę obserwacji, zatem badania dotyczyć mogą zarówno populacji, jak i próbki. Warto jednak podkreślić, że analizę można przeprowadzić poprzez wyodrębnienie próby z całości. Tego typu podejście stosowane jest w przypadkach analizy bardzo dużych zbiorów lub też w wyniku braku pełnych informacji na temat populacji. Ekstrakcja próbki ze zbioru ma duże znaczenie także w analizie wizualnej. Zbyt duża próba umieszczona na wykresie może zniekształcać percepcję odbioru, a w konsekwencji zakłócać możliwość dostrzeżenia prawidłowości ze względu na zbyt dużą strukturę. Z tego powodu, podczas analizy procesów produkcyjnych, warto umieszczać na wykresie wyłącznie podzbiór zawierający wyłącznie niezbędny zakres analizy. Do przygotowania próbki

z populacji można posłużyć się filtrowaniem. W efekcie tego procesu wydobywa się wybrany zakres, np. na podstawie zakresu indeksu, daty, godziny lub przedziału dat.

Filtrowanie na podstawie dat wymaga zwrócenia uwagi na mnogość dostępnych formatów przechowujących datę/godzinę. W przypadku analizowanego zbioru, plik zawiera datę oraz czas w dwóch kolumnach. Wartości te stanowią cenne informacje na temat czasu zapisu próbki. W tym wypadku optymalnym rozwiązaniem jest połączenie obu kolumn do nowej kolumny o nazwie Timestamp i jej konwersja do typu datetime. Wartościami źródłowymi są kolumny data oraz czas. Realizacja tego zadania polega na użyciu dodatkowego parametru parse_dates w metodzie read_csv(), w którym przekazać można formatowanie "TimeStamp": ["Data","Godzina"]. Dzięki temu, podczas importu zostanie utworzona nowa kolumna, do której automatycznie zostaną wstawione wartości z obu kolumn. Dodatkowo nowa kolumna zostanie skonwertowana do typu datetime (Rysunek 1.22).

```
kolumny = ["Data","Day", "Godzina", "Maszyna", "Operator", "P1", "P2", "P3", "P4
    ", "P5", "P6", "P7", "P8", "P9", "P10", "P11", "P12", "P13", "P14", "P15", "
    P16", "P17", "P18", "P19", "P20"]
df = pd.read_csv("new.csv", squeeze=True, parse_dates= {"TimeStamp" : ["Data","
    Godzina"]}, low_memory=True, decimal=".", delimiter=";", warn_bad_lines=True
    , error_bad_lines=False, usecols = kolumny)
df['TimeStamp']= pd.to_datetime(df['TimeStamp'])

#Utworzenie indeksu wraz z pozostawieniem kolumny TimeStamp
#df.index = df.TimeStamp

#Utworzenie indeksu TimeStamp i usunięcie duplikatu
df.set_index('TimeStamp', inplace=True)

#Sortowanie
df.sort_values(by=['TimeStamp'], inplace=True, ascending=True)

#Opcjonalne usunięcie anomalii powstałych na skutek błędów importu oraz
    wpisywania danych
df = df[(df['P1'] > 6.7) & (df['P1'] < 7.6)]
df = df[(df['P2'] > 0) & (df['P1'] < 1)]

#Filtrowanie "na dzień"
df = df.loc['2020-05-01']

#Filtrowanie zakresu dat
#start_date = '2020-05-02'
#end_date = '2020-05-02'
#mask = (df.index >= start_date) & (df.index <= end_date)
#df = df.loc[mask]

print(df.head(30))
print('Zbiór zawiera {} obserwacji i {} zmienne.'.format(df.shape[0], df.shape
    [1]))
```

Rysunek 1.22: Przykład utworzenie dodatkowej kolumny zawierającej datę i czas na podstawie dwóch kolumn

W programie na Rysunku 1.22 znajduje się także wiersz df.set_index( 'TimeStamp', inplace=True), który definiuje indeks na nowo utworzonej

kolumnie Timestamp, jednocześnie eliminując zbędny duplikat kolumny (inplace=True). Utworzenie indeksu umożliwia filtrowanie zbioru na podstawie pojedynczej daty lub przy użyciu zakresu dat. Operacja filtrowania dokonywana jest przy użyciu metody loc(), której argumentem jest wartość lub zmienna oznaczająca datę. W przypadku, gdy koniecznością jest filtrowanie na podstawie zakresu dat, można użyć dodatkowej zmiennej pomocniczej (mask) i przypisać do niej wybrany zakres.

Podczas filtrowania przydatna jest także możliwość usuwania zbędnych kolumn, które mogą istnieć bezpośrednio po imporcie (nie zostały pominięte), jak i zostać utworzone podczas pracy nad zbiorem. Aby usunąć kolumnę, należy wykorzystać metodę drop() i jako argument przekazać nazwę usuwanej kolumny.

Przydatną operacją jest też sortowanie zbioru, co zostało zilustrowane na Rysunku 1.22 w linii df.sort_values(by=['TimeStamp'], inplace=True, ascending=True). Jest to przydatna funkcjonalność w analizie, gdzie wymagane jest zachowanie chronologii, lub też rozmieszczenie zbiorów w wierszach na podstawie ich wartości. Jako parametry filtrowania, oprócz kolumny, której dotyczy sortowanie, podać można także jego kierunek (ascending).

Możliwości algorytmów filtrujących pozwalają także na używanie operatorów logicznych w celu uzyskiwania dokładniejszych wyników. Dzięki temu, możliwe jest filtrowanie z większą precyzją. Funkcjonalność ta jest przydatna w wielu przypadkach, dla przykładu jeśli zajdzie konieczność pozbycia się anomalii powstałych w wyniku importu danych. Jeśli obserwacje te z jakiegoś powodu nie zostały usunięte podczas czyszczenia danych, to znając wartości graniczne, można wyeliminować próbki, które odstają od zakresu wartości produkcyjnych. Tak przefiltrowany zbiór nie będzie powodował występowania potencjalnie fałszywych anomalii podczas dalszej analizy.

```
1 df = df[(df['P1'] > 6.7) & (df['P1'] < 7.6) & (df["Data"].isin(pd.date_range(
      data_from, data_to)))]
2 print('Zbiór zawiera {} obserwacji i {} zmienne.'.format(df.shape[0], df.shape
      [1]))
```

Rysunek 1.23: Wyodrębnienie zakresu na podstawie warunków granicznych oraz zakresu dat

Podczas filtrowania istnieje również możliwość ograniczania wyników wyłącznie dla konkretnej kolumny. Dla przykładu można porównać wartości parametrów analizowanej maszyny z podziałem na obsługujących ją operatorów. Tego typu rozwiązanie pozwala na zbadanie różnic w doborze parametrów dla każdej z maszyn, z obserwacją obsługującego ją operatora.

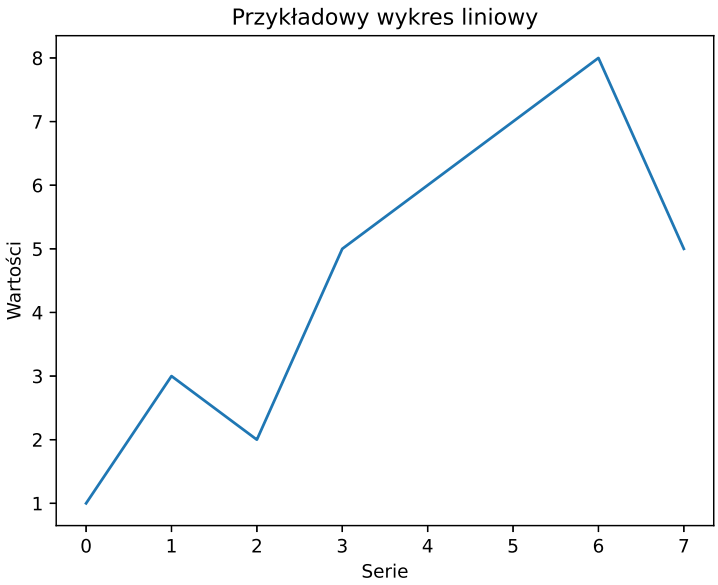
```
in_range_date_mach = df[df["Data"].isin(pd.date_range("2020-05-01", "2020-05-30"
    )) & (df["Maszyna"] == "K1")]
df_machines = in_range_date_mach.groupby("Operator").agg([np.min, np.max, np.
    mean, np.var, np.std])
df_machines.drop('Day', axis=1, inplace=True)
```

Rysunek 1.24: Filtrowanie zbioru przy użyciu zakresu dat i maszyny przy uwzględnieniu operatora

Jednym z istotnych celów statystyki jest prezentacja danych w przystępny i zrozumiały sposób. Choć prezentacja danych statystycznych w formie tabelarycznej jest z pewnością dokładniejsza, to w rzeczywistości dopiero zilustrowanie zjawiska w postaci wykresu graficznego pozwala ocenić najistotniejsze prawidłowości i tendencje zmian. Z tego powodu wykresy stanowią szeroko stosowany środek w prezentacji danych statystycznych. Ilustracja graficzna danego zjawiska, czy też przedstawienie struktury jego dynamiki, pozwala zaprezentować dane w czytelniejszej formie, a dzięki temu, wyciąganie wniosków staje się łatwiejsze.

```
import matplotlib.pyplot as plt
s = pd.Series([1, 3, 2, 5, 6, 7, 8, 5])
s.plot.line()
plt.title("Przykładowy wykres liniowy")
plt.ylabel("Wartości")
plt.xlabel("Serie")
plt.show()
```

Rysunek 1.25: Wykres liniowy wygenerowany przy użyciu zdefiniowanej serii danych




Rysunek 1.26: Wykres liniowy wygenerowany przez program z Rysunku 1.25

Graficzna prezentacja wyników badań możliwa jest dzięki kilku dostępnym bibliotekom. Jedną z najpopularniejszych jest Mathplotlib, która została wielokrotnie wykorzystana na potrzeby badań. Biblioteka integruje się z pakietem Pandas, a dzięki temu, serie danych można zarówno definiować samodzielnie, jak i wskazać ich źródło w obiekcie DataFrame. Najprostszym dostępnym wykresem jest wykres liniowy, który może być wykorzystany w celu zilustrowania wartości zebranych w danym okresie czasu. W tym przypadku wartości prezentowane są przy pomocy punktów, które łączone są linią od pierwszej do ostatniej wartości. Na osi x przyjąć można jednostkę czasu (indeks, datetime), zaś na osi y - wartości próbek z wybranej zmiennej. Forma prezentacji wykresu linowego pozwala na przedstawienie zmienności parametru w badanym okresie czasu. Wykresy można rozbudowywać o kolejne serie, wzbogacać je legendą, a także edytować ich style wizualne.

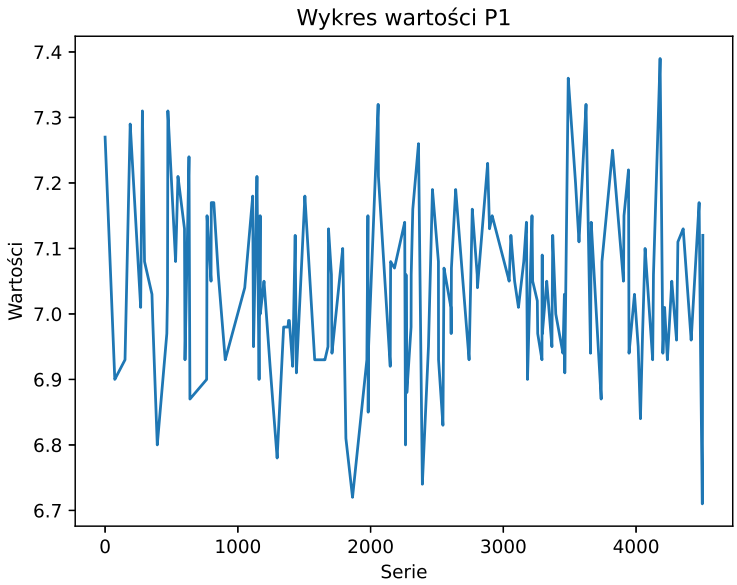
Zamiast generowania wartości parametrów lepiej wykorzystać obiekt DataFrame jako źródło serii (Rysunek 1.28). W tym wypadku na osi x umieszczono automatycznie wygenerowany indeks, zaś na osi y znajdują się wartości serii - w tym wypadku są to wartości parametru technologicznego

o nazwie P1.

```
import matplotlib.pyplot as plt
x = df.index
y = df['P1']
plt.plot(x,y)
plt.title("Wykres wartości P1")
plt.ylabel("Wartości")
plt.xlabel("Serie")
plt.show()
```

Rysunek 1.27: Tworzenie wykresu liniowego na podstawie serii P1

Wykres serii wygenerowany przez program z Rysunku 1.27 wyświetla przebieg wartości parametru P1 pobranego z obiektu DataFrame w czasie. Obserwacja tego typu wykresów umożliwia analizę dynamiki, jednak pomimo dość czytelnej formy, wizualizacja nie jest zbyt dokładna. Brak jest między innymi etykiet wartości, zatem tego typu wykres może posłużyć wyłącznie do zgrubnej oceny zjawiska lub do identyfikacji pojawiających się anomalii w procesie produkcyjnym.



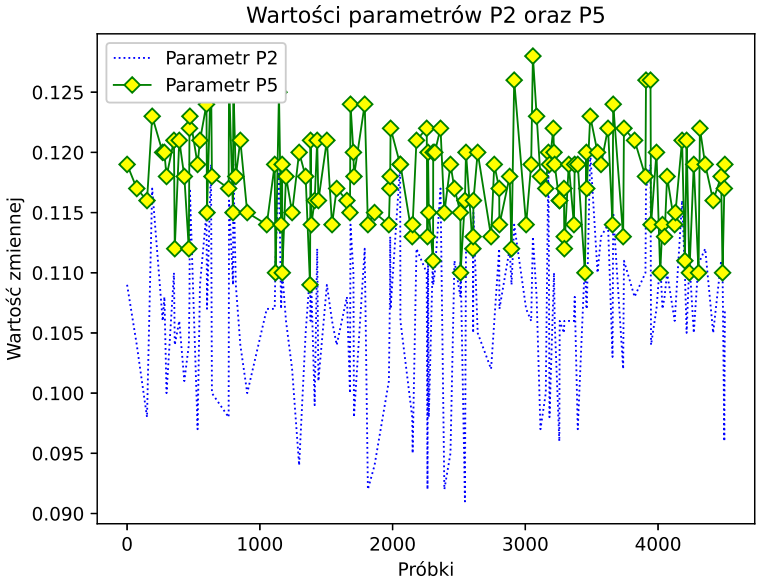

Rysunek 1.28: Wykres liniowy wartości parametru P1 na podstawie serii obiektu DataFrame

O wiele czytelniejszym, a jednocześnie dostarczającym więcej informa-

cji jest wykres zaprezentowany na Rysunku 1.30, na którym umieszczono dwie serie. Dzięki temu, że metoda plot() z biblioteki Matplotlib, umożliwia obserwację więcej niż jednej serii jednocześnie, możliwe jest projektowanie bardziej rozbudowanych wykresów. Należy mieć jednak na uwadze, że należy zestawiać ze sobą wyłącznie serie o podobnych wartościach granicznych oraz podobnej amplitudzie, by nie zaburzyć czytelności wykresu.

```
plt.title("Wartości parametrów P2 oraz P5")
plt.ylabel("Wartość zmiennej")
plt.xlabel("Próbki")
plt.plot(df.P2, "-b", linestyle='dotted', linewidth=1, label="Parametr P2")
plt.plot(df.P11, "-g", linewidth=1, marker='D', markerfacecolor='yellow', label=
    "Parametr P5")
plt.legend(loc="upper left")
plt.show()
```

Rysunek 1.29: Program generujący wykres liniowy zawierający obserwację wartości z więcej niż jednej serii




Rysunek 1.30: Wykres liniowy wartości serii P2 i P11 z dodatkowym formatowaniem

Program z Rysunku 1.29 generuje wykres serii głównej, dodatkowo wzbogacając go o markery, w zestawieniu z dodatkową serią wartości P2.

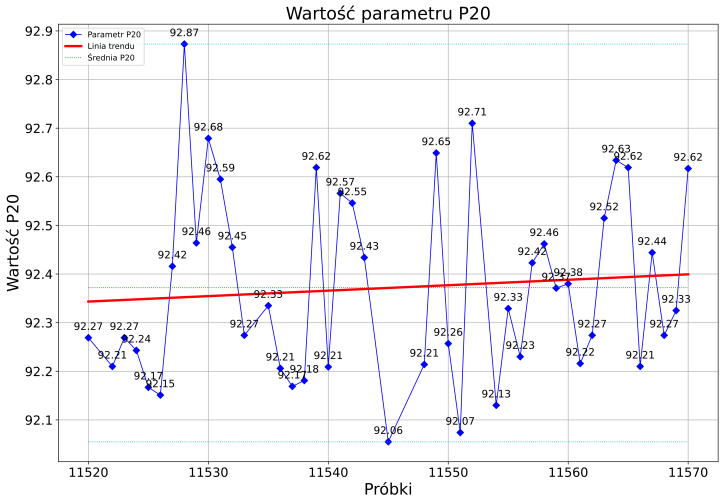
Wykresy tego typu mogą służyć analizie kilku parametrów produkcyjnych jednocześnie, wizualizując dodatkowe informacje np. na temat zależności pomiędzy wartościami serii.

Istnieją scenariusze, w których koniecznością staje się wizualizacja wartości parametrów produkcyjnych w przedziale czasowym, z dodatkowymi informacjami dotyczącymi wybranej serii. Przykładem tego typu wizualizacji jest program z Rysunku 1.31, który tworzy wykres wartości serii na podstawie kolumny P20. Dodatkowo dodaje do wykresu linię trendu, średnią oraz wartości minimalne oraz maksymalne. Wykres wzbogacono również o wyświetlanie etykiet wartości przy pomocy metody annotate(). Na prezentowanym wykresie, jako oś czasu x wykorzystano indeks obiektu DataFrame.

```
from scipy.stats import linregress
plt.title("Wartość parametru P20")
plt.ylabel("Wartość P20")
plt.xlabel("Próbki")

p20_mean = [np.mean(df.P20)]*len(df.index)
p20_min = [np.min(df.P20)]*len(df.index)
p20_max = [np.max(df.P20)]*len(df.index)

plt.plot(df.P20, "-b", linestyle='solid', marker='D', linewidth=1, label="
    Parametr P20")
slope, intercept, r_value, p_value, std_err = linregress(df.index, df.P20)
plt.plot(df.index, intercept + slope*df.index, 'r', linewidth=3, label='Linia
    trendu')

for x,y in zip(df.index, df.P20):
    label = "{:.2f}".format(y)
    plt.annotate(label, (x,y), textcoords="offset points", xytext=(0,10), ha='
    center')

plt.plot(df.index, p20_mean, "-g", linewidth=1, linestyle='dotted',
    markerfacecolor='green', label="Średnia P20")
plt.plot(df.index, p20_min, 'c', linewidth=1, linestyle='dotted')
plt.plot(df.index, p20_max, 'c', linewidth=1, linestyle='dotted')
plt.legend(loc="upper left")
plt.grid()

plt.show()
```

Rysunek 1.31: Program generujący wykres dla serii P20 z dodatkowymi informacjami

Program z Rysunku 1.31 generuje wykres na podstawie serii P20, zaś efektem jego działania jest wykres zaprezentowany na Rysunku 1.32. Można zauważyć, że charakteryzuje się on czytelnością, a jednocześnie dostarcza wielu innych informacji na temat wybranego parametru produkcyjnego. Warto zwrócić uwagę, że tego typu wizualizacja może okazać się mniej czytelna w przypadku, gdy wybrano okno czasowe o zbyt dużym rozmiarze. Wówczas konieczna jest dodatkowa korekta okna czasowego przy pomocy filtrowania.




Rysunek 1.32: Wykres liniowy serii P20 z linią trendu oraz liniami min i max

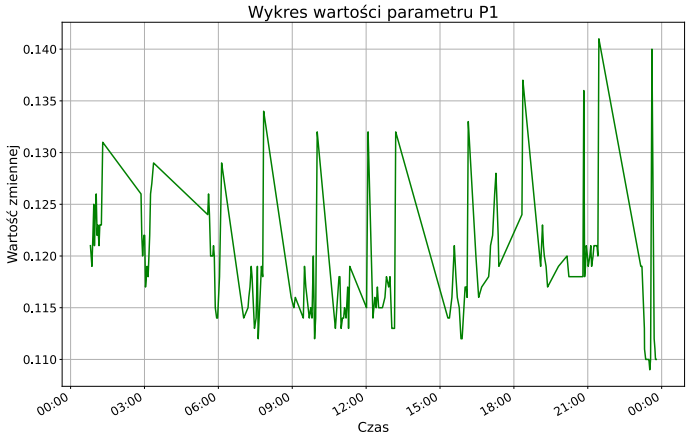
Zamiast indeksu umieszczanego na osi x oznaczającego czas, można wykorzystać inną jednostkę czasu. Umożliwi to kontrolę wartości parametru w wybranej jednostce. W zależności od potrzeby, oś x może wyświetlać podział w zakresie dat lub godzin, o ile wartości te zostały zarejestrowane w pliku źródłowym. Podczas prowadzenia badań, jako zmienną czasu zdefiniowano czas zapisu pojedynczej próbki. Aby wykorzystać kolumnę data lub czas na osi x wykresu, należy ustawić indeks na wybrane pole (w tym wypadku jest to utworzona kolumna TimeStamp) i dokonać odpowiedniego formatowania.

```
plt.figure(figsize=(15,10))
plt.title("Wartości parametrów P1")
plt.ylabel("Wartość zmiennej")
plt.xlabel("Czas")
plt.plot(df.P11, "-g", linewidth=1, label="Parametr P1")
plt.legend(loc="upper left")
plt.gcf().autofmt_xdate()
myFrm = mdates.DateFormatter('%H:%M')
plt.gca().xaxis.set_major_formatter(myFrm)
plt.show()
```

Rysunek 1.33: Wykres wartości P1 w funkcji czasu (godzinowy)

Wprowadzenie jednostki czasu na wykresie pozwala na weryfikację wartości serii i analizę zmian przy jednoczesnej obserwacji czasu, w którym wartość została pobrana. Może to stanowić źródło dodatkowej wiedzy na te-

mat zmian wartości produkcyjnych. W zestawieniu z innymi parametrami, informacje tego typu mogą służyć do szybkiej analizy potencjalnych błędów ze wskazaniem na dokładny czas ich wystąpienia.




Rysunek 1.34: Wykres liniowy serii z osią x na podstawie godziny pobrania próbki

### 1.3.2. Analiza wariancji

Pakiet SciPy dla języka Python dostarcza metody służące między innymi do obliczania wariancji. Jeżeli wariancja jest miarą odstępstwa wyrazów szeregu od średniej, to podczas analizy przydatne stają się także funkcje obliczania średniej czy odchylenia standardowego. Funkcje te dostępne są standardowo w obiekcie DataFrame i nazywają się odpowiednio mean() oraz std().

```
1 import numpy as np
2 import pandas as pd
3 kolumny = ["P1","P2","P3","P4","P5","P6","P7"]
4 df = pd.read_csv("new.csv", decimal=".", delimiter=";", usecols = kolumny)
5 df.agg(["mean", "std"])
```

Rysunek 1.35: Program wyświetlający średnią oraz odchylenie standardowe szeregów w zbiorze

Wynikiem działania programu z Rysunku 1.35 jest wyświetlenie wartości średnich oraz odchylenia standardowego dla wybranych kolumn w postaci tabelarycznej.

```
1             P1        P2        P3        P4        P5        P6        P7
2  mean   7.239774  0.098265  0.001866  0.005181  0.259077  0.005417  0.003169
3  std    0.766107  0.008221  0.001870  0.000894  0.014095  0.001218  0.000536
```

Rysunek 1.36: Efekt działania programu z Rysunku 1.35

Algorytm obliczania wariancji jest dostępny jako metoda var(). Zwraca on nieobciążoną wariancję względem żądanej osi (0 - wiersze, 1 - kolumny). Co istotne, algorytm ten przyjmuje opcjonalny argument o nazwie ddof, którego wartość domyślna wynosi 1. Pozwala on na wskazanie stopni swobody, które zostaną użyte podczas obliczeń. Wartość 0 pozwala obliczyć wariancję dla populacji, z kolei wartość 1 szacuje wariancję populacji z próby. Warto także dodać, że parametr ddof jest także dostępny w metodzie std().

```
1 import numpy as np
2 import pandas as pd
3 kolumny = ["P1","P2","P3","P4","P5","P6","P7"]
4 df = pd.read_csv("new.csv", squeeze=True, decimal=".", delimiter=";", usecols =
      kolumny)
5 df.var(axis=0, ddof=0)
```

Rysunek 1.37: Program obliczający wariancję dla szeregów P1-P10

Wykonanie programu z Rysunku 1.37 spowoduje wyświetlenie wyników obliczeń w postaci wykładniczej. Jest to spowodowane tym, że liczby są bliskie zeru (Rysunek 1.38). Jednakże w przypadku obliczania wariancji dla pojedynczej kolumny (df.var()['P1'])), wynik może zostać wyświetlony w postaci liczby rzeczywistej.

```
1  P1      5.869206e-01
2  P2      6.758497e-05
3  P3      3.496801e-06
4  P4      7.994444e-07
5  P5      1.986724e-04
6  P6      1.484323e-06
7  P7      2.870990e-07
8  P8      1.092365e-07
9  P9      5.347280e-08
10 P10     1.799473e-07
11 P11     3.386303e-05
12 dtype: float64
```

Rysunek 1.38: Wynik działania metody var() dla populacji

Zapis wykładniczy może utrudniać analizę wariancji, zatem warto dokonać zmiany wyników na wartości rzeczywiste. Można tego dokonać przy pomocy funkcji anonimowej, jak pokazano na przykładzie z Rysunku 1.39.

```
pd.set_option('display.float_format', lambda x: f'%.{len(str(x%1))-2}f' % x)
df.var(axis=0, numeric_only=1, ddof=1)
```

Rysunek 1.39: Program wyświetlający wariancję w postaci liczb rzeczywistych z parametrów P1-P10 dla populacji

W wyniku uruchomienia fragmentu kodu, liczby będą wyświetlane w czytelniejsze formie:

```
P1          0.5869206313934943
P2       0.000067584974600002
P3      0.0000034968005802185
P4      0.0000007994444471970
P5     0.00019867244303222985
P6     0.00000148432269181448
P7     0.00000028709898226175
P8     0.00000010923650377798
P9     0.00000005347279865904
P10     0.0000001799472992334
P11     0.0000338630271368678
dtype: float64
```

Rysunek 1.40: Wynik działania metody var() z funkcją formatującą lambda

Formatowanie wyników wpływa na precyzję wyświetlania liczb, zatem należy mieć na uwadze możliwe błędy wyświetlania w przypadku, gdy zostanie wybrana zbyt niska precyzja. Dla przykładu kod z Rysunku 1.41 ustawia precyzję wyświetlania wyników do 7 miejsc po przecinku.

```
pd.options.display.float_format = '{:.7f}'.format
df.var(axis=0, numeric_only=1, ddof=1)
```

Rysunek 1.41: Formatowanie precyzji liczba przy pomocy opcji float_format

W wyniku działania programu z Rysunku 1.41, wyświetlanie liczb rzeczywistych zostanie skrócone do określonej przez parametr {:.7f} precyzji. Należy zwrócić uwagę na potencjalne błędy spowodowane zbyt niską precyzją, co może zaburzyć wyświetlanie wyników (wyświetlanie wyłącznie zer). Precyzja powinna być każdorazowo dobrana na podstawie pełnych wyników, lub należy wstępnie ustawić ją na większy rozmiar.

```
P1     0.5869206
P2     0.0000676
P3     0.0000035
P4     0.0000008
P5     0.0001987
P6     0.0000015
P7     0.0000003
P8     0.0000001
P9     0.0000001
P10    0.0000002
P11    0.0000339
dtype: float64
```

Rysunek 1.42: Wynik działania opcji float_format z wybraną precyzją

Oprócz standardowej metody obliczania wariancji, dostępna jest także wariancja okienkowa i w tym wypadku jest ona dostępna w metodzie rolling().var(). Argumentem rolling() jest rozmiar okna badanego szeregu. W badaniu przyjęto przykładowe wartości parametry window=4 oraz window=8. Jeśli chodzi o badanie wariancji okienkowej szeregów, wybrano kolumny P1, P2, P5, P11 P18 oraz P20. Wyniki obliczeń przypisano do nowego obiektu o nazwie dfVar. Program realizujący obliczanie wariancji okienkowej znajduje się w Rysunku 1.43.

```
import numpy as np
import pandas as pd

kolumny = ["Data","Day", "Godzina", "Maszyna","Operator", "P1", "P3", "P5", "P20"]
df = pd.read_csv("new.csv", squeeze=True, low_memory=True, decimal=".",
     delimiter=";", warn_bad_lines=True, error_bad_lines=False, usecols = kolumny)

pd.options.display.float_format = '{:.8f}'.format
print('Zbiór DF zawiera {} obserwacji i {} zmiennych.'.format(df.shape[0], df.shape[1]))

dfVar = pd.DataFrame()
win = 4
dfVar['vP1'] = df.P1.rolling(win).var(ddof=0)
dfVar['vP2'] = df.P2.rolling(win).var(ddof=0)
dfVar['vP5'] = df.P5.rolling(win).var(ddof=0)
dfVar['vP11'] = df.P11.rolling(win).var(ddof=0)
dfVar['vP18'] = df.P18.rolling(win).var(ddof=0)
dfVar['vP20'] = df.P20.rolling(win).var(ddof=0)

dfVar.dropna(axis=0, how='any', inplace=True)
print(dfVar.head(10))
```

Rysunek 1.43: Program obliczający wariancję z populacji metodą okienkową dla wybranych parametrów

Wynik działania programu przedstawiono na Rysunku 1.44. Program podczas działania tworzy nowy obiekt DataFrame, w którym znajdują się obliczone wariancje serii wybranych parametrów.

```
1            vP1          vP2          vP5          vP11         vP18         vP20
2  3  0.00685000 0.00001125 0.00004869 0.00000369 0.00000125 0.00658469
3  4  0.00542500 0.00005850 0.00004525 0.00000650 0.00000169 0.00530500
4  5  0.00746875 0.00005069 0.00004550 0.00000869 0.00000069 0.00791069
5  7  0.00585000 0.00003519 0.00002150 0.00001025 0.00000050 0.00537050
6  8  0.00735000 0.00003719 0.00000969 0.00003219 0.00000050 0.00655625
7  9  0.01511875 0.00000425 0.00000969 0.00003725 0.00000069 0.01427769
8  10 0.01161875 0.00000225 0.00001225 0.00003925 0.00000125 0.01071519
9  11 0.01876875 0.00001850 0.00000919 0.00003269 0.00000169 0.01942319
10 12 0.01971875 0.00002319 0.00000875 0.00001800 0.00000150 0.02057475
11 13 0.00665000 0.00001400 0.00000669 0.00002869 0.00000069 0.00850500
```

Rysunek 1.44: Rezultat działania programu z Rysunku 1.43 dla okna 4

Po zmianie parametru definiującego rozmiar okna dla wariancji okienkowej, wykonanie programu z parametrem window=8 dało następujące wyniki:

```
1            vP1          vP2          vP5          vP11         vP18         vP20
2  8  0.00912500 0.00003398 0.00002925 0.00001894 0.00000094 0.00888648
3  9  0.01083594 0.00003294 0.00003011 0.00002244 0.00000125 0.01048698
4  10 0.00960000 0.00003211 0.00003194 0.00002473 0.00000111 0.00936550
5  11 0.01299844 0.00002948 0.00001661 0.00002336 0.00000111 0.01293161
6  12 0.01354844 0.00003075 0.00000936 0.00002523 0.00000100 0.01357450
7  13 0.01561094 0.00001419 0.00000975 0.00003311 0.00000094 0.01616961
8  14 0.01682500 0.00001619 0.00001686 0.00003194 0.00000150 0.01724211
9  15 0.01571094 0.00001469 0.00001569 0.00002936 0.00000161 0.01660325
10 16 0.01581094 0.00006025 0.00001619 0.00002086 0.00000144 0.01662748
11 17 0.00399844 0.00006248 0.00001619 0.00001898 0.00000075 0.00499498
```

Rysunek 1.45: Wynik działania programu z Rysunku 1.43 dla okna 8

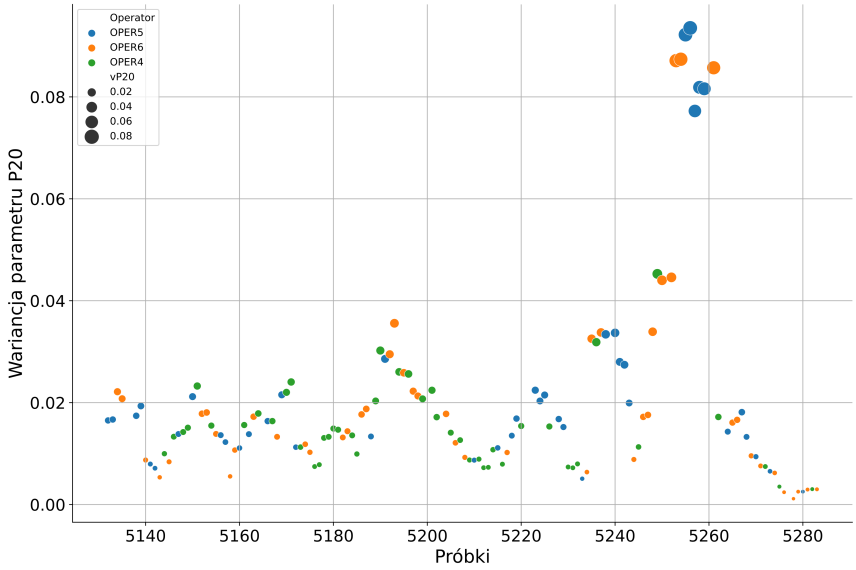
Do wygenerowania wykresu wariancji dla wybranej kolumny użyto biblioteki Seaborn, która jest nakładką na Matplotlib, jednak została wzbogacona o dodatkowe typy wykresów. Umozliwia ona szybkie tworzenie atrakcyjnie wyglądających wykresów.

Ze względu na dużą liczbę próbek w zbiorze, analiza wizualna wariancji została przeprowadzona na przefiltrowanym zbiorze w celu zmniejszenia zakresu zakresu. Zbiór wejściowy wykorzystano przy użyciu programu z Rysunku 1.43.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

wyk = sns.relplot(x=dfVar.index, y=dfVar.vP20,
                  size=dfVar.vP20, sizes=(15, 200),
                  hue=dfVar.Maszyna,
                  data=dfVar);
wyk.fig.set_figwidth(15)
wyk.fig.set_figheight(8)
wyk.set_ylabels("Wariancja parametru P20")
wyk.set_xlabels("Próbki")
wyk.ax.legend(loc=2)
plt.show()
```

Rysunek 1.46: Program generujący wykres wariancji dla parametru P20

Program z Rysunku 1.46 generuje przykład wykresu wariancji, uwzględniając grupowanie maszyn produkcyjnych oraz zmierzone wariancje dla parametru P20. Wynik działania programu widoczny jest na Rysunku 1.47.




Rysunek 1.47: Wykres wariancji parametru P20 dla parametru okna=8

Analizując wykres z Rysunku 1.47 widać wyraźne odstępstwa wartości serii na osi czasu x. Można to interpretować jako możliwe zakłócenia procesu produkcyjnego. Należy jednak mieć na uwadze, że informacje te są jedynie wskazówką dla dalszych badań pod kątem miejsca i czynników determinujących potencjalne błędy. W efekcie pozwoli to w dalszej analizie na odkrywanie zależności i w rezultacie wskazanie źródła zakłóceń.

W kolejnym kroku, można zmienić argument hue, co pozwoli na ponowne wygenerowanie wykresu z podziałem obejmującym inną kolumnę.

Dalsze badania nad eksploracją zbioru mogą polegać na analizie próbki pod kątem poszukiwania korelacji pomiędzy np. wykorzystanymi maszynami produkcyjnymi i operatorami, analizie wystąpienia anomalii ze względu na porę dnia czy dni tygodnia. W ten sposób można będzie potwierdzić lub odrzucić hipotezę o wpływie czynnika ludzkiego na zakłócenia procesu produkcyjnego.

### 1.3.3. Testowanie normalności rozkładu

Pierwszym krokiem analizy jest zrozumienie rozkładu cech. Pakiet SciPy zawiera wiele tego typu testów, zaś ich metody testujące zwracają prawdopodobieństwo P-value (krytyczny poziom istotności). Jest to najmniejszy poziom istotności, przy którym dla zaobserwowanej wartości statystyki testowej można odrzucić hipotezę zerową. Hipotezę zerową można z kolei odrzucić, gdy obliczone prawdopodobieństwo testowe w zmiennej $p$ okaże się nie większe od przyjętego poziomu istotności, które zwykle wynosi 0.05. Jeśli wartość $p \leq 0.05$, hipotezę można odrzucić, zaś w przypadku, gdy wartość $p > 0.05$, to prawdopodobnie zmienna ma wartości pochodzące z rozkładu normalnego.

Badanie szeregów pod kątem oceny normalności rozkładu można dokonać na dwa sposoby:

- statystyczny - wykonując jeden z testów statystycznych i otrzymać wartość $p$;
- graficzny - wizualizując rozkład zmiennej przy pomocy jednego z dostępnych wykresów.

Do testów normalności rozkładu wybrano próbkę danych z populacji (zakres dat) i eliminując anomalia sklasyfikowane jako błędy ekstrakcji lub błędy produkcyjne. Prawidłowe wartości parametru P1 muszą mieścić się w zakresie: P1 $>$ 6.7 oraz P1 $<$ 7.6).

```
data_from = "2020-05-01"
data_to = "2020-05-30"
df = df[(df['P1'] > 6.7) & (df['P1'] < 7.6) & (df["Data"].isin(pd.date_range(
    data_from, data_to)))]
print('Zbiór zawiera {} obserwacji i {} zmienne.'.format(df.shape[0], df.shape
    [1]))
```

Rysunek 1.48: Filtrowanie zbioru do testów rozkładu

Zaimportowany plik zawierający początkowo 38600 obserwacji, po zastosowaniu filtrów z Rysunku 1.48, zawiera dane z 361 obserwacji oraz 24 zmienne.

Pierwszym testem, który wykonano dla próbki jest test Shapiro-Wilka. Jest to test przeznaczony do testowania normalności rozkładów w zbiorach do 5 tysięcy próbek. Jest on dostępny w pakiecie shapiro, zaś implementacja testu wygląda następująco:

```
from scipy.stats import shapiro
stats, p = shapiro(df.P18)
print(stats,p)
if p > 0.05:
    print ("Rozkład normalny")
```

Rysunek 1.49: Test Shapiro-Wilka

Warto nadmienić, że test Shapiro-Wilka może zwracać błędne wyniki dla większych próbek, choć jest to preferowany test normalności rozkładu prawdopodobieństwa, ze względu na jego siłę.

Kolejną metodą testowania jest test D'Agostino–Pearsona. Jest to prosty test dostępny standardowo w bibliotece SciPy. Zakłada on, że próbka nie powinna mieć liczebności mniejszej niż 20 obserwacji. Jego implementacja na identycznym szeregu danych wygląda następująco:

```
stats, p = normaltest(df.P18)
print(stats, p)
if p > 0.05:
    print ("Rozkład normalny")
```

Rysunek 1.50: Test D'Agostino–Pearsona

Następnym przeprowadzonym testem z biblioteki SciPy jest test Kołmogorowa-Smirnowa. Jest to nieparametryczny test oceny zgodności rozkładu analizowanych zmiennych z rozkładem normalnym. Test ten jest wykorzystywany w przypadkach, gdy nie jest znana średnia lub odchylenie standar-

dowe populacji. Metodę tę należy wybierać dla próbek o $n > 100$. Podobnie jak inne testy statystyczne, również testuje hipotezę zerową wskazującą na rozkład zbliżony do rozkładu normalnego. Implementację testu ilustruje Rysunek 1.51.

```
from scipy.stats import kstest, norm
stats, p = kstest(df.P18, 'norm')
print(stats, p)
if p > 0.05:
    print ("Rozkład normalny")
```

Rysunek 1.51: Test Kołmogorowa-Smirnowa

Kolejny test oparty został na metodzie Andersona-Darlinga, która opiera się na statystycznych testach zgodności rozkładu z zadanym rozkładem wzorcowym. Testuje on hipotezę zerową próbki z populacji o określonym rozkładzie. Wartości krytyczne zależą od testowanej dystrybucji. Metoda działa dla dystrybucji normalnej, wykładniczej, logistycznej lub Gumbela. Domyślnie testowanym rozkładem jest rozkład normalny, lecz metoda przyjmuje również parametry do testowania innych rozkładów (expon, logistic, gumbel, gumbel_l, gumbel_r, extreme1).

```
from scipy.stats import anderson
result = anderson(df.P1)
print('Statystyka: %.3f' % result.statistic)
for i in range(len(result.critical_values)):
    sig_lev, crit_val = result.significance_level[i], result.critical_values[i]
    if result.statistic < crit_val:
        print(f'Rozkład wygląda na normalny: wartość krytyczna {crit_val},
    poziom istotności {sig_lev}')
    else:
        print(f'Hipoteza może zostać odrzucona: wartość krytyczna {crit_val},
    poziom istotności {sig_lev}')
```

Rysunek 1.52: Test normalności Anderson-Darling

Kolejny z przeprowadzonych testów to test zgodności chi-kwadrat. Jako często stosowany, służy on do weryfikowania hipotezy, czy obserwowana cecha w zbiorowości ma określony typ rozkładu. Badana próba powinna być wybrana w taki sposób, by uzyskać wystarczającą liczbę obserwacji, lecz na ogół wymóg ten wynosi od około pięciu do dziesięciu obserwacji.

```
from scipy.stats import chisquare
stat, p = chisquare(df.P1)
print('Statystyka: %.3f, p = %.3f' % (stat, p))
if p > 0.05:
    print ("Rozkład wygląda na normalny")
else:
    print ("Hipoteza odrzucona")
```

Rysunek 1.53: Test normalności typu Chi kwadrat

Następnym wykonanym testem jest Test Lillieforsa, oparty na teście Kołmogornowa-Smirnowa. Test ten można zastosować w przypadku próbek, w których nie jest znana wartość średnia oraz odchylenie standardowe dla populacji, z której próba pochodzi. Implementacja testu została przedstawiona na Rysunku 1.54.

```
from statsmodels.stats.diagnostic import lilliefors
stat, p = lilliefors(df.P1)
print('Statystyka: %.3f, p = %.3f' % (stat, p))
if p > 0.05:
    print ("Rozkład wygląda na normalny")
else:
    print ("Hipoteza odrzucona")
```

Rysunek 1.54: Test normalności rozkładu typu Lilliefors

Ostatnim z przedstawionych testów jest test normalności rozkładu Jarque–Bera. Jest to test występujący często w ekonometrii ze względu na swoją prostotę oraz znaną nieskomplikowaną postać rozkładu asymptotycznego. Konstrukcja statystyki testowej bazuje na wartościach momentów rozkładu zmiennej losowej obliczonych na podstawie próby empirycznej i porównaniu ich z momentami teoretycznymi rozkładu normalnego. Test weryfikuje hipotezę o jednowymiarowej normalności zmiennej losowej przeciwko innemu dowolnemu rozkładowi. Jego implementację przedstawia Rysunek 1.55.

```
from scipy.stats import jarque_bera
stat, p = jarque_bera(df.P1)
print('Statystyka: %.3f, p = %.3f' % (stat, p))
if p > 0.05:
    print ("Rozkład wygląda na normalny")
else:
    print ("Hipoteza odrzucona")
```

Rysunek 1.55: Test normalności rozkładu Jarque–Bera

Jeśli chodzi o metody graficzne do oceny normalności rozkładu, wykorzystano histogram dla parametru numerycznego, który jest podstawowym

i powszechnie używanym wykresem do szybkiej analizy danych pod kątem normalności rozkładu. W tego typu wykresie dane są przedstawione przy pomocy określonej liczby prostokątów, w których znajdują się posortowane dane zawierające liczbę obserwacji. Histogram można wygenerować za pomocą funkcji displot() z pakietu Seaborn. Dla celów wizualizacji wykorzystano pakiet i dostępną w bibliotece metodę displot(), która zapewnia możliwość szybkiego wygenerowania wykresu rozkładu oraz jego głównych tendencji. Histogramy dzielą wartości zmiennych ciągłych na dyskretne podziały i pokazują ilość wartości w każdym z tych przedziałów. Dodatkowo w wykresie wykorzystano opcję kde=True, w celu wygenerowania wykresu estymacji gęstości jądra, dla oszacowania funkcji gęstości prawdopodobieństwa zmiennej losowej.

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.displot(df, x=df.P1, kind="hist", kde=True, aspect=1.2)
5 plt.show()
```

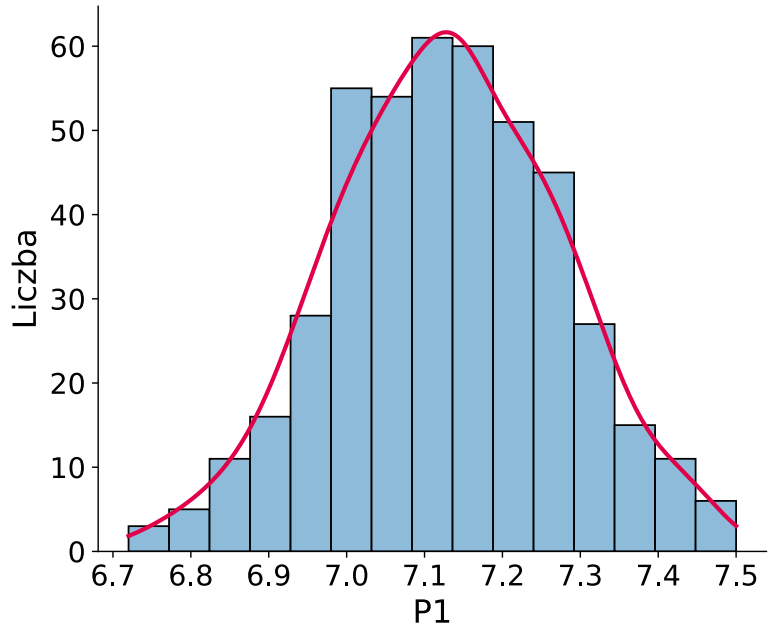
Rysunek 1.56: Program generujący histogram dla szeregu P1

Domyślnie liczba przedziałów jest automatycznie szacowana na podstawie próbki danych, lecz można ją zwiększyć dodając dodatkowy parametr bins=x, gdzie x oznacza liczbę przedziałów. Liczba przedziałów uzależniona jest od obszaru zmienności badanej cechy, liczebności zbiorowości oraz celu badania.

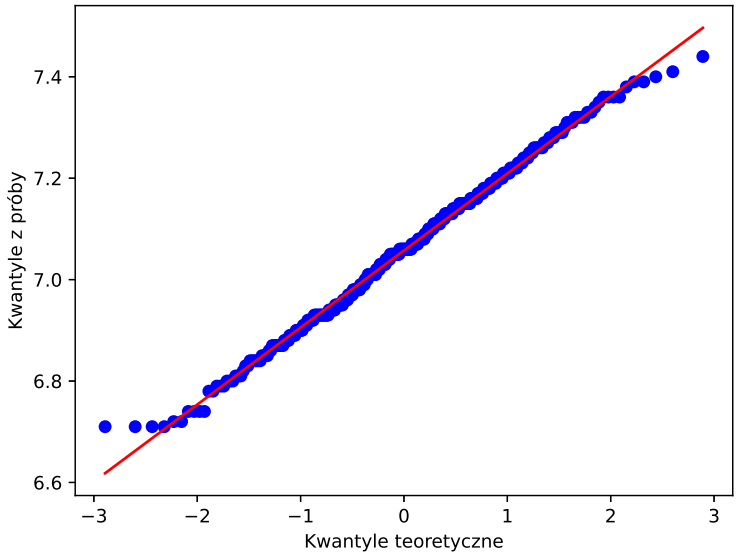
```
1 import scipy.stats as stats
2 import matplotlib.pyplot as plt
3 stats.probplot(df['P1'], dist="norm", plot=plt)
4 plt.show()
```

Rysunek 1.57: Kod generujący wykres typu QQ Plot dla szeregu P1

Kolejnym popularnym typem wykresu jest Q-Q Plot (wykres kwantyl kwantyl), który ułatwia ocenę rozkładu badanej zmiennej. Do utworzenia wykresu użyto funkcji stats.probplot() z pakietu SciPy, który generuje wykres kwantyli dla zmiennej P1 na osi y, w porównaniu do kwantyli teoretycznych rozkładu normalnego.




Rysunek 1.58: Histogram dla próbki z populacji szeregu P1




Rysunek 1.59: Wykres QQ plot dla próbki P1

Interpretacja wykresu 1.59 polega na obserwacji skupienia punktów wokół prostej. Widać wyraźną korelację dodatnią, w której wartości y mają tendencję do zwiększania się wraz ze wzrostem wartości x. Ponieważ zmienna ma rozkład normalny, jej wartości są zgodne z teoretycznymi kwantylami. Istotną informacją jest równomierne rozmieszczenie punktów blisko prostej (naprzemiennie). Oznacza to, że dane pochodzą z rozkładu normalnego. Ponieważ można zaobserwować niewielkie odstępstwa w postaci kilku punktów powyżej i poniżej prostej, jednak odstępstwa te są na tyle niewielkie, że testy statystyczne nie potwierdziły odstępstw od rozkładu normalnego.

### 1.3.4. Wyznaczanie parametrów rozkładu Weibulla

Kolejnym etapem badań nad zbiorem jest inżynieria niezawodności i analiza przetrwania. W tym celu posłużono się biblioteką Reliability, która posiada bogaty zestaw funkcji do tego typu analizy. Znacząco rozszerza też funkcjonalność scipy.stats, dostarczając wiele specjalistycznych narzędzi.

Możliwości biblioteki reliability:

- dopasowywanie rozkładów prawdopodobieństwa do danych, w tym cenzurowanych prawostronnie;
- obliczanie prawdopodobieństwa uszkodzenia dla interferencji naprężenie-wytrzymałość pomiędzy dowolną kombinacją obsługiwanych rozkładów;
- obsługa rozkładów prawdopodobieństwa wykładniczego, Weibulla, Gamma, Gumbela, normalnego, lognormalnego, logistycznego oraz beta;
- średni czas życia, kwantyle, podsumowania statystyk opisowych, losowe próbkowanie z rozkładów;
- wykresy funkcji gęstości prawdopodobieństwa (PDF), dystrybuanty rozkładu (CDF), funkcji przeżycia (SF), funkcji ryzyka (HF) i funkcji ryzyka skumulowanego (CHF);
- nieparametryczna estymacja funkcji przeżycia za pomocą metod Kaplana-Meiera, Nelsona-Aalena i korekty rang;
- testy zgodności (AICc, BIC, AD, Log-Likelihood);
- wykresy prawdopodobieństwa dla wszystkich obsługiwanych rozkładów;

- wykresy kwantyl-kwantyl i prawdopodobieństwo-prawdopodobieństwo;
- wzrost niezawodności, optymalny czas wymiany, sekwencyjne wykresy próbkowania, podobne rozkłady, planowanie testów niezawodności;
- interaktywne funkcje matplotlib, w tym eksplorator dystrybucji;
- fizyka awarii (wykres SN, naprężenie-odkształcenie, mechanika pękania);
- przyspieszone modele testowania żywotności obejmujące 4 rozkłady (Weibull, Exponential, Normal, Lognormal) oraz 6 modeli stresu życiowego (Exponential, Eyring, Power, Dual-Exponential, Dual-Power, Power-Exponential);
- średnia funkcja skumulowana i ROCOF dla systemów naprawialnych.

Biblioteka umożliwia dopasowanie rozkładów zarówno do pełnych, jak i niekompletnych danych. Dostępne moduły dopasowania (Fitters) zostały nazwane liczbą ich parametrów, dla przykładu Fit_Weibull_2P używa $\alpha$, $\beta$, zaś Fit_Weibull_3P używa $\alpha$, $\beta$, $\gamma$. Rozkłady dopasowywane są przy użyciu żądanej funkcji wraz z przekazanymi do niej błędami (można przekazać także dane ocenzurowane). W celach analizy zaleca się wstawienie co najmniej 4 próbek, gdyż dokładność dopasowania zależy od liczby przekazanych wartości. Do celów analizy została wykorzystana metoda Fit_Weibull_2P.

Przygotowanie zbioru do analizy niezawodności polega na przefiltrowaniu go i zbadaniu rozkładu Weibulla. W tym przykładzie użyto parametru P1.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from reliability.Fitters import Fit_Weibull_2P

kolumny = ["Data","P1"]
df = pd.read_csv("new.csv", squeeze=True, decimal=".", delimiter=";", usecols =
    kolumny)
df['Data']= pd.to_datetime(df['Data'])
data_from = "2020-05-01"
data_to = "2020-05-05"
df = df[(df['Data']> data_from) & (df['Data'] < data_to)]

pd.options.display.float_format = '{:.8f}'.format
print('Zbiór DF zawiera {} obserwacji i {} zmiennych.'.format(df.shape[0], df.
    shape[1]))

df['error'] = np.where((df['P1'] < 6.8) | (df['P1'] > 7.6), True, False)
print(df.error.value_counts())
df = df[(df['error'] == True)]

fail_data = df.index.tolist()

wb = Fit_Weibull_2P(failures=fail_data)
plt.legend()
plt.xlabel("Czas")
plt.show()
```

Rysunek 1.60: Program generujący wykres Weibulla z próbki

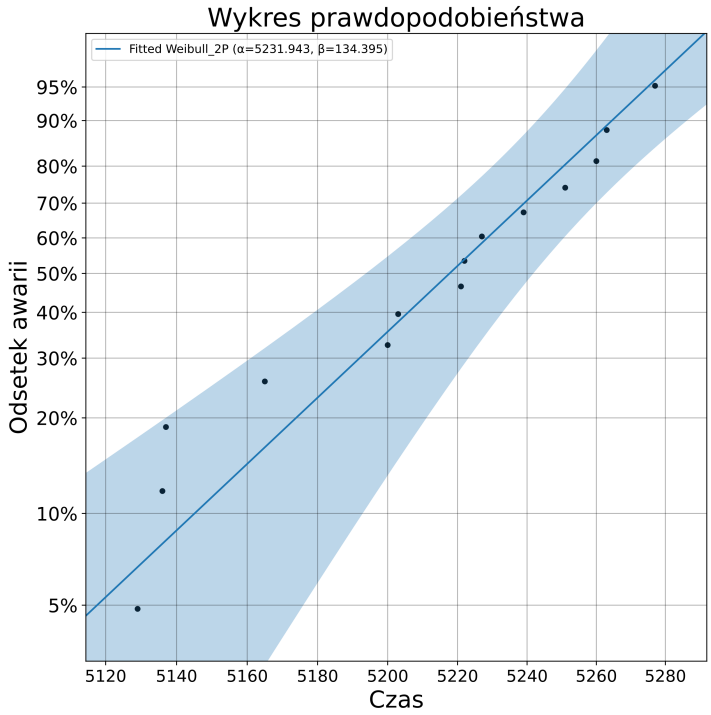
Program z Rysunku 1.60 importuje zbiór, filtruje go i wywołuje na obiekcie DataFrame iterację warunku wyznaczającego błędy w postaci dodatkowej kolumny logicznej. W tym wypadku błędy wyznaczane są na podstawie przekroczenia wartości produkcyjnych dla serii. W kolejnym kroku wywołana zostaje funkcja dopasowania Weibulla z pakietu reliability.

```
Zbiór DF zawiera 160 obserwacji i 2 zmiennych.
False    146
True      14
Name: error, dtype: int64
[5129, 5136, 5137, 5165, 5200, 5203, 5221, 5222, 5227, 5239, 5251, 5260, 5263,
    5277]
Results from Fit_Weibull_2P (95% CI):
Analysis method: Maximum Likelihood Estimation (MLE)
Failures / Right censored: 14/0 (0% right censored)

Parameter  Point Estimate  Standard Error       Lower CI       Upper CI
    Alpha   5231.94329283     10.96223802 5210.50175734 5253.47306156
     Beta    134.39549578     29.00363004   88.04172994  205.15441142

Goodness of fit        Value
 Log-likelihood -73.44960973
           AICc 151.99012855
            BIC 152.17733412
             AD   1.15198232
```

Rysunek 1.61: Wynik działania programu z Rysunku 1.60

Wynik działania programu z Rysunku 1.61 informuje, że próbka zawiera 146 wartości poprawnych oraz 14 błędów. Po odfiltrowaniu poprawnych wartości zawiera serię wyłącznie błędów w postaci poszczególnych numerów

indeksów obiektu fail_data = [5129, 5136, 5137, 5165, 5200, 5203, 5221, 5222, 5227, 5239, 5251, 5260, 5263, 5277]. Dane wyjściowe zawierają także przedziały ufności oraz błąd standardowy oszacowań parametrów. Wykres prawdopodobieństwa generowany jest automatycznie przy pomocy metody plt.show() (Rysunek 1.62).

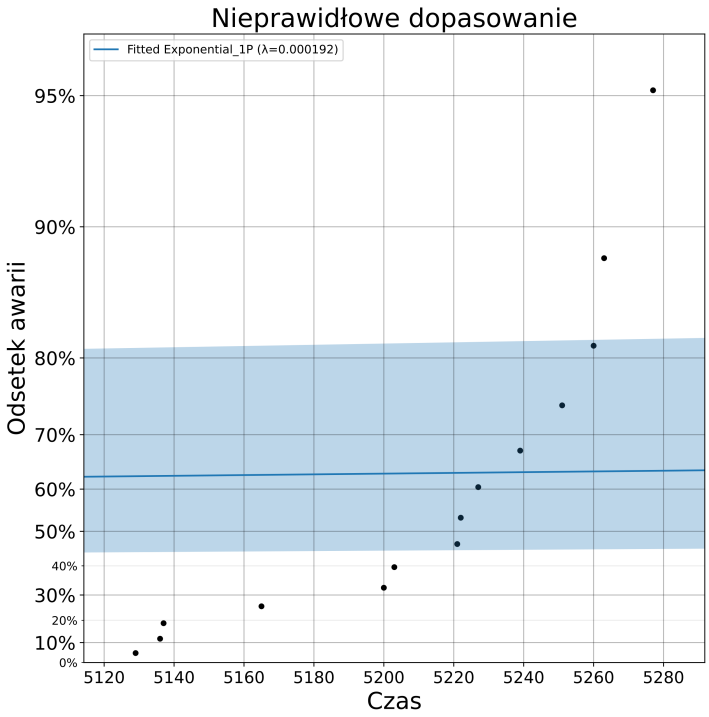



Rysunek 1.62: Wykres Weibulla na podstawie programu z Rysunku 1.60

Na podstawie wykresu (Rysunek 1.62) można zauważyć, jak modelowane są dane. Interpretacja wykresu polega na analizie położenia punktów, które powinny być umieszczone na linii prostej, jednakże w większości przypadków tak nie jest. Nieprawidłowe dopasowanie występuje, gdy linia lub krzywa utworzona przez punkty znacznie odbiega od linii prostej. Niewielkie odchylenia są tolerowane na końcach rozkładu, lecz większość punktów powinna podążać za prostą.

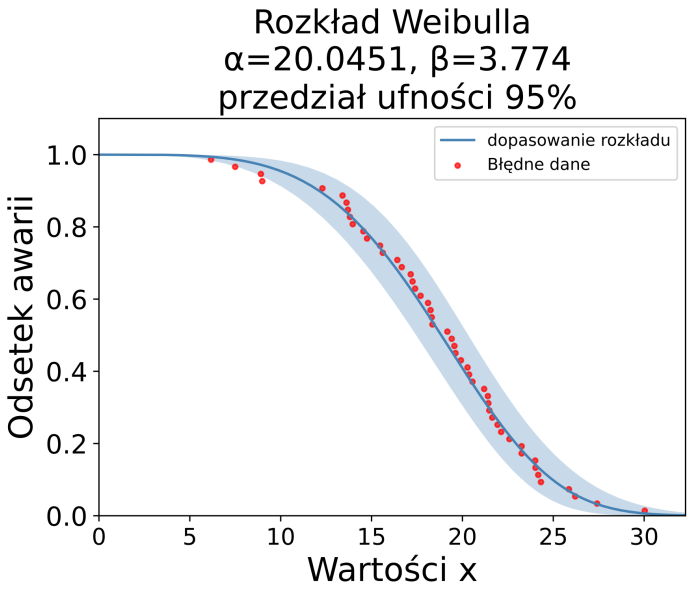
Skalując osie można dokonać oceny, czy empiryczny współczynnik danych awarii (punkty na wykresie) jest zgodny ze współczynnikiem dopa-

sowanego rozkładu. Złe dopasowanie jest widoczne, gdy linia lub krzywa utworzona przez punkty znacznie odbiega od linii prostej (Rysunek 1.63).




Rysunek 1.63: Przykład nieprawidłowego dopasowania

Aby wyświetlić punkty awarii obok punktów PDF, CDF, SF, HF lub CHF bez użycia skalowania osi, można wygenerować wykres przy pomocy funkcji plot_points, która kreśli punkty awarii na podstawie pozycji. Funkcja generuje punkty podobnie, jak w poprzednim przypadku, jednak nie skaluje osi lub dopasowanego rozkładu.




Rysunek 1.64: Przykład wykorzystania funkcji plot_points

Funkcja plot_points przyjmuje następujące argumenty:

- failures - tablica lub lista danych awarii;
- right_censored - opcjonalna tablica lub lista prawocenzurowanych danych;
- func - funkcja kreśląca wykres (PDF, CDF, SF, HF, CHF). Domyślnie CDF;
- $a$ - stała heurystyczna do wykreślania pozycji postaci $(k-a)/(n+1-2a)$. Wartość domyślna to $a = 0.3$;
- keywords - słowa kluczowe dla wykresu punktowego.

Praktyczne wykorzystanie funkcji plot_points() zaprezentowano na Rysunku 1.65.

```
1 weibull_fit = Fit_Weibull_2P(failures=data, show_probability_plot=False,
      print_results=False)
2 weibull_fit.distribution.SF(label='dopasowanie rozkładu', color='steelblue')
3 plot_points(failures=data, func='SF', label='błędne dane', color='red', alpha=0.7)
4 plt.legend()
5 plt.show()
```

Rysunek 1.65: Przykład użycia funkcji plot_points() dla wykresu SF

Wynikiem działania programu jest utworzenie wykresu rozkładu Weibulla, który znajduje się na Rysunku 1.64.

## 1.4. Podsumowanie

W przemyśle coraz większe znaczenie odgrywają systemy przeznaczone do analizy dużych zbiorów danych, a trend ten dotyczy wszystkich sektorów. Silnym impulsem do rozwoju tej dziedziny wiedzy jest stały wzrost znaczenia informacji oraz konieczność konkurowania na globalnych rynkach. Eksploracja zbiorów pod kątem pozyskania konkretnej wiedzy jest sposobem na zautomatyzowanie dostarczania odpowiedzi na uprzednio zadane pytania. Przeanalizowane i przygotowane w zrozumiały dla człowieka sposób informacje pochodzące z systemów produkcyjnych, posłużyć mogą do szybszej predykcji awarii i zużycia podzespołów czy monitorowania procesów.

Warto wykorzystać do tych celów powszechnie dostępne narzędzia, takie jak język Python oraz szereg dostępnych bibliotek. Choć podczas badań wykorzystano wyłącznie środowisko języka Python, istnieje szereg wygodnych środowisk dedykowanych, jak na przykład Jupyter czy Spyder. Narzędzia te wyposażone są w wygodne interfejsy oraz wbudowany system pomocy, co znacznie ułatwia i przyspiesza pracę nad zbiorem.

W niniejszym rozdziale przedstawione zostały wybrane zagadnienia automatycznej analizy danych pochodzących z procesów produkcyjnych. Autorzy chcieli w ten sposób pokazać, jak obszerna jest to tematyka. Drugim celem było zademonstrowanie jak nowoczesne narzędzia programistyczne mogą służyć we wspieraniu analizy danych w przedsiębiorstwach. Chęć pokazania konkretnych możliwości wynika z faktu, że wiele współcześnie stosowanych systemów informatycznych w bardzo niewielkim stopniu wykorzystuje te nowoczesne technologie.

## Bibliografia

[1] R. Aigner, M. Leitner, M. Stoschka. Fatigue strength characterization of Al-Si cast material incorporating statistical size effect. G. Henaff, redaktor, *12th International Fatigue Congress (FATIGUE 2018)*, wolumen 165 serii *MATEC Web of Conferences*, 2018.
https://doi.org/10.1051/matecconf/201816514002.

[2] A. Almeida, A. Loy, H. Hofmann. ggplot2 Compatible Quantile-Quantile Plots in R. *The R Journal*, 10(2):248–261, 2018.
https://doi.org/10.32614/RJ-2018-051.

[3] J.B. Almeida. Application of weilbull statistics to the failure of coatings. *Journal of Materials Processing Technology*, 92-93:257–263, 1999.
https://doi.org/10.1016/S0924-0136(99)00177-6.

[4] T.W. Anderson, D.A. Darling. Asymptotic Theory of Certain 'Goodness of Fit' Criteria Based on Stochastic Processes. *Annals of Mathematical Statistics*, 23(2):193–212, 1952.
https://doi.org/10.1214/aoms/1177729437.

[5] J.I. Ansell, M.J. Phillips. *Practical Methods for Reliability Data Analysis*. Oxford University Press, Oxford, 1994.

[6] N.H. Augustin, E.-A. Sauleau, S.N. Wood. On quantile quantile plots for generalized linear models. *Computational Statistics & Data Analysis*, 56(8):2404–2409, 2012.
https://doi.org/10.1016/j.csda.2012.01.026.

[7] Y.-L. Bai, Z.-W. Yan, T. Ozbakkaloglu, Q. Han, J.-G. Dai, D.-J. Zhu. Quasi-static and dynamic tensile properties of large-rupture-strain (LRS) polyethylene terephthalate fiber bundle. *Construction and Building Materials*, 232, 2020.
https://doi.org/10.1016/j.conbuildmat.2019.117241.

[8] N. Baringhaus, N. Henze. A consistent test for multivariate normality based on the empirical characteristic function. *Metrika*, 35:339–348, 1988.
https://doi.org/10.1007/BF02613322.

[9] N. Baringhaus, N. Henze. Limit distributions for mardia's measure of multivariate skewness. *Annals of Statistics*, 20(4):1889–1902, 1992.
https://doi.org/10.1214/aos/1176348894.

[10] R.E. Barlow, F. Proschan. *Statisitcal Theory of Reliability and Life Testing*. Holt, Rinehart, Austin, 1975.

[11] A. Birolini. *Reliability Engineering*. Springer, Berlin, Heidelberg, 2017.
https://doi.org/10.1007/978-3-662-54209-5.

[12] A.W. Bowman, P.J. Foster. Adaptive Smoothing and Density-Based Tests of Multivariate Normality. *Journal of the American Statistical Association*, 88(422):529–537, 1993.
https://doi.org/10.1080/01621459.1993.10476304.

[13] G. Box. Signal-to-Noise Ratios, Performance Criteria, and Transformations. *Technometrics*, 30(1):1–17, 1988.
https://doi.org/10.2307/1270311.

[14] G.E.P. Box, J.S. Hunter. Condensed calculations for evolutionary operation programs. *Technometrics*, 1(1):77–95, 1959.
https://doi.org/10.1080/00401706.1959.10489850.

[15] R. Chalapathy, S. Chawla. Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407, 2019.
http://arxiv.org/abs/1901.03407.

[16] A. Chmielowiec. *Rozkład Weibulla i jego zastosowanie w procesie optymalizacji kosztów eksploatacji elementów nienaprawialnych*, wolumen 1. Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów, 2020.
https://depot.ceon.pl/handle/123456789/19465.

[17] A. Chmielowiec. Algorithm for error-free determination of the variance of all contiguous subsequences and fixed-length contiguous subsequences for a sequence of industrial measurement data. *Computational Statistics*, 2021. https://doi.org/10.1007/s00180-021-01096-1.

[18] S. Csorgo. Testing for Normality in Arbitrary Dimension. *Annals of Statistics*, 14(2):708–723, 1986. https://doi.org/10.1214/aos/1176349948.

[19] K.R. Das, A.H.M.R. Imon. A brief review of tests for normality. *American Journal of Theoretical and Applied Statistics*, 5(1):5–12, 2016. https://doi.org/10.11648/j.ajtas.20160501.12.

[20] S.S. Dhar, B. Chakraborty, P. Chaudhuri. Comparison of multivariate distributions using quantile–quantile plots and related tests. *Bernoulli*, 20(3):1484–1506, 2014. https://doi.org/10.3150/13-BEJ530.

[21] J. Ding, Y. Liu, L. Zhang, J. Wang, Y. Liu. An anomaly detection approach for multiple monitoring data series based on latent correlation probabilistic model. *Applied Intelligence*, 44:340–361, 2016. https://doi.org/10.1007/s10489-015-0713-7.

[22] B. Ebner, N. Henze. Tests for multivariate normality – a critical review with emphasis on weighted $L^2$-statistics. *TEST*, 29:845–892, 2020. https://doi.org/10.1007/s11749-020-00740-0.

[23] E.A. Elsayed, A. Chen. Optimal levels of process parameters for products with multiple characteristics. *International Journal of Production Research*, 31(5):1117–1132, 1993. https://doi.org/10.1080/00207549308956778.

[24] T.W. Epps, L.B. Pulley. A test for normality based on the empirical characteristic function. *Biometrika*, 70(2):723–726, 1983. https://doi.org/10.2307/2335564.

[25] K. Evans, T. Love, S.W. Thurston. Outlier identification in model-based cluster analysis. *Journal of Classification*, 32:63–84, 2015. https://doi.org/10.1007/s00357-015-9171-5.

[26] R.A. Fisher. The moments of the distribution for normal samples of measures of departure from normality. *Proceedings of the Royal Society*, 130(812):16–28, 1930. https://doi.org/10.1098/rspa.1930.0185.

[27] R.A. Fisher, L.M.C. Tippett. Limiting forms of frequency distribution of the largest or smallest member of a sample. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24:180–190, 1928. https://doi.org/10.1017/S0305004100015681.

[28] S.L. Fok, B.C. Mitchell, J. Smart, B.J. Marsden. A numerical study on the application of the weibull theory to brittle materials. *Engineering Fracture Mechanics*, 68(10):1171–1179, 2001. https://doi.org/10.1016/S0013-7944(01)00022-4.

[29] M. Fréchet. Sur la loi de probabilité de l'écart maximum. *Annales de la Société Polonaise de Mathématique*, 6:93–116, 1927.

[30] O. Grynchenko, O. Alfyorov. *Mechanical Reliability*. Springer, Cham, 2020. https://doi.org/10.1007/978-3-030-41564-8.

[31] B. Guner, M.T. Frankford, J.T. Johnson. A study of the Shapiro-Wilk test for the detection of pulsed sinusoidal radio frequency interference. *IEEE transactions on Geoscience and Remote sensing*, 47(6):1745–1751, 2009. https://doi.org/10.1109/TGRS.2008.2006906.

[32] D.M. Hawkins. *Identification of outliers*. Springer, Dordrecht, 1980.

[33] M.A. Hemphill, T. Yuan, G.Y. Wang, J.W. Yeh, C.W. Tsai, A. Chuang, P.K. Liaw. Fatigue behavior of Al0.5CoCrCuFeNi high entropy alloys. *Acta Materialia*, 60(16):5723–5734, 2012. https://doi.org/10.1016/j.actamat.2012.06.046.

[34] N. Henze. Invariant tests for multivariate normality: a critical review. *Statistical Papers*, 43(4):467–506, 2002. https://doi.org/10.1007/s00362-002-0119-6.

[35] N. Henze, J. Visagie. Testing for normality in any dimension based on a partial differential equation involving the moment generating function. *Annals of the Institute of Statistical Mathematics*, 5:1–28, 2019. https://doi.org/10.1007/s10463-019-00720-8.

[36] N. Henze, T. Wagner. A New Approach to the BHEP Tests for Multivariate Normality. *Journal of Multivariate Analysis*, 62(1):1–23, 1997. https://doi.org/10.1006/jmva.1997.1684.

[37] N. Henze, B. Zirkler. A class of invariant and consistent tests for multivariate normality. *Communications in Statistics - Theory and Methods*, 19(10):3595–3617, 1990. https://doi.org/10.1080/03610929008830400.

[38] R.J. Hyndman, E. Wang, N. Laptev. Large-scale unusual time series detection. *2015 IEEE interna-tional conference on data mining workshop (ICDMW)*, strony 1616–1619, 2015. https://doi.org/10.1109/ICDMW.2015.104.

[39] J. Jacquelin. Inference of sampling on Weibull parameter estimation. *IEEE transactions on dielectrics and electrical insulation*, 3(6):809–816, 1996. https://doi.org/10.1109/94.556564.

[40] N.L. Johnson, S. Kotz, N. Balakrishnan. *Continuous Univariate Distributions*, wolumen 1. Wiley, New York, 1994.

[41] V.E. Johnson, M. Hamada, H. Martz, S. Reese, A. Wilson. *Modern Reliability Analysis: A Bayesian Perspective*. Springer, Berlin, Heidelberg, New York, 2005.

[42] R.N. Kackar. Off-Line Quality Control, Parameter Design, and the Taguchi Method. *Journal of Quality Technology*, 17(4):176–209, 1985. https://doi.org/10.1080/00224065.1985.11978964.

[43] R.N. Kackar, A.C. Shoemaker. *Robust design: A cost effective method for improving manufacturing process*, strony 159–174. Springer, Boston, 1989. https://doi.org/10.1007/978-1-4684-1472-1˙8.

[44] E. Keogh, J. Lin. Clustering of time-series subsequences is meaningless: implications for previousand future research. *Knowledge and Information Systems*, 8(2):154–177, 2005. https://doi.org/10.1007/s10115-004-0172-7.

[45] E. Keogh, J. Lin, A. Fu. Hot sax: efficiently finding the most unusual time series subsequence. *Fifth IEEE international conference on data mining (ICDM'05)*, strona 8, 2005.
https://doi.org/10.1109/ICDM.2005.79.

[46] E. Keogh, J. Lin, S.H. Lee, H.V. Herle. Finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*, 11(1):1–27, 2006.
https://doi.org/10.1007/s10115-006-0034-6.

[47] K. Keshevan, G. Sargent, H. Conrad. Statistical analysis of the hertzian fracture of pyrex glass using the weibull distribution function. *Journal of Materials Science*, 15:839–844, 1980.
https://doi.org/10.1007/BF00552092.

[48] D.E. Knuth. *The Art of Computer Programming*, wolumen II: Seminumerical Algorithms. Addison-Wesley, wydanie 2, 1981.

[49] C. Lai, D.N. Murthy, M. Xie. *Weibull Distributions and Their Applications*, strony 63–78. Springer London, London, 2006.
https://doi.org/10.1007/978-1-84628-288-1˙3.

[50] C.-D. Lai. *Generalized Weibull Distributions*. Springer, Heidelberg, 2014.
https://doi.org/10.1007/978-3-642-39106-4.

[51] N. Laptev, S. Amizadeh, I. Flint. Generic and scalable framework for automated time-series anomaly detection. *KDD'15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, strony 1939–1947, 2015.
https://doi.org/10.1145/2783258.2788611.

[52] R.V. Leon, A.C. Shoemaker, R.N. Kackar. Performance Measure Independent of Adjustment: An Explanation and Extension of Taguchi's Signal-to-Noise Ratio. *Technometrics*, 29(3):253–265, 1987.
https://doi.org/10.2307/1269331.

[53] Q.S. Li, J.Q. Fang, D.K. Liu, J. Tang. Failure probability prediction of concrete components. *Cement and Concrete Research*, 33(10):1631–1636, 2003.
https://doi.org/10.1016/S0008-8846(03)00111-X.

[54] H.W. Lilliefors. On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967.
https://doi.org/10.1080/01621459.1967.10482916.

[55] N. Logothetis, A. Haigh. Characterizing and optimizing multi-response processes by the Taguchi method. *Quality and Reliability Engineering International*, 4(2):159–169, 1988.
https://doi.org/10.1002/qre.4680040211.

[56] C.A. Lowry, D.C. Montgomery. A review of multivariate control charts. *IIE Transactions*, 27(6):800–810, 1995.
https://doi.org/10.1080/07408179508936797.

[57] J.F. Malkovich, A.A. Afifi. On Tests for Multivariate Normality. *Journal of the American Statistical Association*, 68(341):176–179, 1973.
https://doi.org/10.2307/2284163.

[58] K.V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530, 1970.
https://doi.org/10.2307/2334770.

[59] R.L. Mason, C.W. Champ, N.D. Tracy, S.J. Wierda, J.C. Young. Assessment of multivariate process control techniques. *Journal of Quality Technology*, 29(2):140–143, 1997.
https://doi.org/10.1080/00224065.1997.11979743.

[60] F.J. Massey. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.
https://doi.org/10.1080/01621459.1951.10500769.

[61] J.W. McPherson. *Reliability Physics and Engineering: Time-To-Failure Modeling*. Springer, Cham, 2019.
https://doi.org/10.1007/978-3-319-93683-3.

[62] D.C. Montgomery. *Introduction to statistical quality control*. Wiley, New York, 2019.

[63] D.C Montgomery, W.H. Woodall. A Discussion on Statistically-Based Process Monitoring and Control. *Journal of Quality Technology*, 29(2):121–121, 1997.
https://doi.org/10.1080/00224065.1997.11979738.

[64] T.F. Mori, G.J. Szekely, M.L. Rizzo. On energy tests of normality. *Journal of Statistical Planning and Inference*, 213:1–15, 2021.
https://doi.org/10.1016/j.jspi.2020.11.001.

[65] D.N.P. Murthy, M. Xie, R. Jiang. *Weibull Models*. Wiley, New York, 2003.
https://doi.org/10.1002/047147326X.

[66] V.N. Nair. Taguchi's Parameter Design: A Panel Discussion. *Technometrics*, 34(2):127–161, 1992.
https://doi.org/10.2307/1269231.

[67] J.A. Newell, T. Kurzeja, M. Spence, M. Lynch. Analysis of recoil compressive failure in high performance polymers using two-, four-parameter weibull models. *High Performance Polymers*, 14:425–434, 2002.
https://doi.org/10.1177/095400830201400408.

[68] A. Nielsen. *Szeregi czasowe. Praktyczna analiza i predykcja z wykorzystaniem statystyki i uczenia maszynowego*. O. Reilly, New York, 2020.

[69] W.R. Oldford. Self-calibrating quantile–quantile plots. *The American Statistician*, 70(1):74–90, 2016.
https://doi.org/10.1080/00031305.2015.1090338.

[70] E.S. Pearson. A further development of tests for normality. *Biometrika*, 22(1-2):239–249, 1930.
https://doi.org/10.2307/2332073.

[71] M.S. Phadke, R.N. Kackar, D.V. Speeney, M.J. Grieco. Off-line quality control integrated circuit fabrication using experimental design. *Bell System Technical Journal*, 62(5):1273–1309, 1983.

[72] J.J. Pignatiello. Strategies for robust multiresponse quality engineering. *IIE Transactions*, 25(3):5–15, 1993.
https://doi.org/10.1080/07408179308964286.

[73] F.S. Queeshi, A.K. Sheikh. A probabilistic characterization of adhesive wear in metals. *IEEE Transactions on Reliability*, 46(1):38–44, 1997. https://doi.org/10.1109/24.589924.

[74] R. Ross. Bias and standard deviation due to Weibull parameter estimation for small data sets. *IEEE Transactions on Dielectrics and Electrical insulation*, 3(1):28–42, 1996. https://doi.org/10.1109/94.485512.

[75] P. Royston. Algorithm AS 181: the W test for normality. *Applied Statistics*, 31(2):176–180, 1982. https://doi.org/10.2307/2347986.

[76] P. Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 31(2):115–124, 1982. https://doi.org/10.2307/2347973.

[77] P. Royston. Approximating the Shapiro-Wilk W-test for non-normality. *Statistics and computing*, 2(3):117–119, 1992. https://doi.org/10.1007/BF01891203.

[78] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A.P. Boedihardjo, C. Chen, S. Frankenstein. Time seriesanomaly discovery with grammar-based compression. *Proceedings of the 18th international conference on extending database technology, EDBT 2015*, strony 481–492, 2015. https://doi.org/10.5441/002/edbt.2015.42.

[79] S.S. Shapiro, R.S. Francia. An Approximate Analysis of Variance Test for Normality. *Journal of the American Statistical Association*, 67(337):215–216, 1972. https://doi.org/10.1080/01621459.1972.10481232.

[80] S.S. Shapiro, M.B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, (3-4):591–611, 1965. https://doi.org/10.2307/2333709.

[81] A.K. Sheikh, J.K. Boah, D.A. Hansen. Statistical modelling of pitting corrosion and pipeline reliability. *Corrosion*, 46(3):190–197, 1990. https://doi.org/10.5006/1.3585090.

[82] W.A. Shewart. *Economic Control of Quality Manufactured Product*. D. Van Nostrand, New York, 1931.

[83] G.J. Szekely, M.L. Rizzo. A new test for multivariate normality. *Journal of Multivariate Analysis*, 93(1):58–80, 2005. https://doi.org/10.1016/j.jmva.2003.12.002.

[84] G. Taguchi. *Introduction to Quality Engineering: Designing Quality into Products and Processes*. Asian Productivity Organization, Tokyo, 1986.

[85] C. Tenreiro. A new test for multivariate normality by combining extreme and nonextreme BHEP tests. *Communications in Statistics - Simulation and Computation*, 46(3):1746–1759, 2017. https://doi.org/10.1080/03610918.2015.1011334.

[86] O. Thas, J.P. Ottoy. Some generalizations of the Anderson–Darling statistic. *Statistics & Probability Letters*, 64(3):255–261, 2003. https://doi.org/10.1016/S0167-7152(03)00169-X.

[87] M. Tiryakioğlu, J. Campbell. Weibull analysis of mechanical data for castings: A guide to the interpretation of probability plots. *Metallurgical and Materials Transactions A*, 41(12):3121–3129, 2010.
https://doi.org/10.1007/s11661-010-0364-6.

[88] K.-L. Tsui. A critical look at Taguchi's modelling approach. *Journal of Applied Statistics*, 23(1):81–95, 1996.
https://doi.org/10.1080/02664769624378.

[89] K.-L. Tsui. Robust design optimization for multiple characteristic problems. *International Journal of Production Research*, 37(2):433–445, 1999.
https://doi.org/10.1080/002075499191850.

[90] O. Vasicek. A Test for Normality Based on Sample Entropy. *Journal of the Royal Statistical Society. Series B*, 38(1):54–59, 1976.
https://www.jstor.org/stable/2984828.

[91] X. Wang, J. Lin, N. Patel, M. Braun. Exact variable-length anomaly detection algorithm for univariate and multivariate time series. *Data Minind and Knowledge Discovery*, 32:1806–1844, 2018.
https://doi.org/10.1007/s10618-018-0569-7.

[92] W. Weibull. A statistical theory of the strength of material. *Ingeniors Vetenskaps Akademiens Handligar*, 151:5–45, 1939.

[93] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18:293–296, 1951.

[94] B.P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.
https://doi.org/10.2307/1266577.

[95] S. Woo. *Reliability Design of Mechanical Systems*. Springer, Singapore, 2020.
https://doi.org/10.1007/978-3-319-50829-0.

[96] W.H. Woodall, K.-L. Tsui, G. R. Tucker. A Review of Statistical and Fuzzy Quality Control Charts Based on Categorical Data. H.-J. Lenz, P.-T. Wilrich, redaktorzy, *Frontiers in Statistical Quality Control*, strony 83–89, Heidelberg, 1997. Physica-Verlag HD.
https://doi.org/10.1007/978-3-642-59239-3˙7.

[97] S. Xie, H. Lin, Y. Wang, Y. Chen, W. Xiong, Y. Zhao, S. Du. A statistical damage constitutive model considering whole joint shear deformation. *International Journal of Damage Mechanics*, 29(6):988–1008, 2020.
https://doi.org/10.1177/1056789519900778.

[98] Y. Zhang, N. Meratnia, P. Havinga. Outlier detection techniques for wireless sensor networks: a survey. *IEEE Communications Surveys and Tutorials*, 12(2):159–170, 2010.
https://doi.org/10.1109/SURV.2010.021510.00088.

[99] L.-X. Zhu, H.L. Wong, K.-T. Fan. A test for multivariate normality based on sample entropy and projection pursuit. *Journal of Statistical Planning and Inference*, 45(3):373–385, 1995.
https://doi.org/10.1016/0378-3758(94)00058-4.

**Statistical analysis of big data sets in quality control and maintenance**

**Abstract:** The use of statistical methods in quality control and business management dates back to the beginning of the 20th century. At the beginning of the 21st century, this field of science faced the challenge of processing large sets of measurement data. The growing number of sensors on production lines is associated with the need to use faster and more effective methods of both quality control and searching for relationships between variables characterizing processes. Therefore, this chapter is devoted to the use of statistical and IT tools to effectively solve some problems related to the analysis of large sets of measurement data.

# 2. Metody sztucznej inteligencji w predykcyjnym utrzymaniu ruchu

Tomasz Kycia[1]
Politechnika Rzeszowska, t.kycia@prz.edu.pl

**Streszczenie**

Algorytmy sztucznej inteligencji mają ogromne zastosowanie w wielu dziedzinach życia, a jedną z nich jest niewątpliwie przemysł. Zastosowanie odpowiednio przygotowanych środowisk programistyczno - sprzętowych pozwala na precyzyjne określenie trwałości eksploatacyjnej maszyny w fabryce czy dokładne planowanie serwisu i utrzymanie maszyn. Ten artykuł jest wstępem do bardziej szczegółowych badań nad wdrażaniem uczenia maszynowego. Zastosowane w pracy narzędzie programistyczne są ogólnodostępne i darmowe, dzięki czemu każdy może próbować wykorzystać opisaną w artykule technologie i próbować tworzyć bardziej rozbudowane systemy. Pozwolą one analizować w czasie rzeczywistym zachodzące procesy w maszynie i na bieżąco informować o stanie urządzeń.

## 2.1. Wprowadzenie

Czy czwarta rewolucja przemysłowa jest przełomem? Czy nowe technologie, informatyzacja, wszechogarniająca sieć komputerowa jest drogą do nowego postrzegania przemysłu i świata? Czy korzystając z nowoczesnych

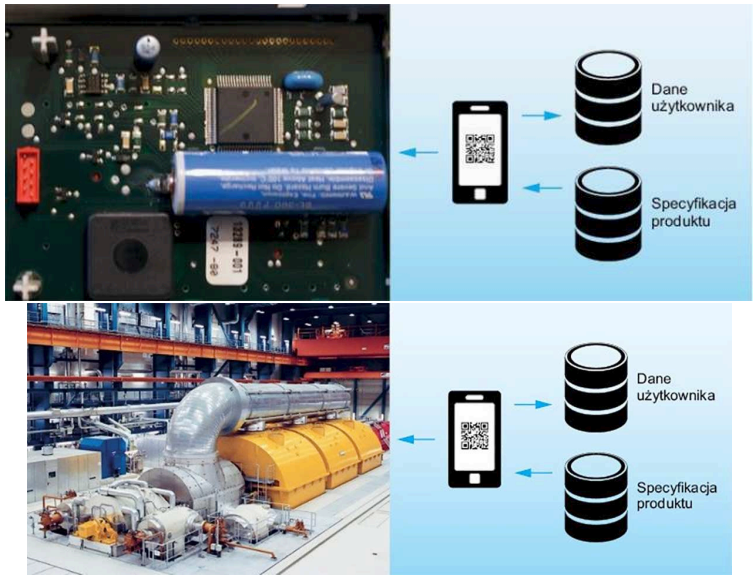
[1]ORCID: 0000-0002-7085-8665, Wydział Mechaniczno-Technologiczny Politechniki Rzeszowskiej, Kwiatkowskiego 4, 37-450 Stalowa Wola

technik obliczeniowych, sztucznej inteligencji potrafimy przewidzieć nadchodzące wydarzenia? To tylko nieliczne pytania jakie nasuwają się ludziom skutecznie żyjącym we współczesnych czasach, często w naszym widzeniu świata - przełomowych. Człowiek i maszyna... Gdzie jest wspólny mianownik pomiędzy nami? Na przestrzeni lat zawsze staraliśmy się wcielić w role „stwórcy", aby zbudować - stworzyć „coś" co będzie „żyło" samoistnie. Budujemy maszyny, roboty, sprzęt technicznie bardzo wysoko zaawansowany, który ma służyć ogólnie pojętemu dobru człowieka. W historii przemysłu możemy wyróżnić kilka przełomowych etapów, które umownie nazywamy rewolucjami przemysłowymi. Pierwsza rewolucja przemysłowa rozpoczęła się około roku 1760 i trwała mniej więcej do 1840. Uruchomiła ją budowa kolei i wynalezienie silnika parowego, a jej efektem było wprowadzenie produkcji mechanicznej. Druga rewolucja przemysłowa, która przypadła na przełom XIX i XX wieku, umożliwiła produkcję masową, co rozpoczęło się od zastosowania elektryczności i wprowadzenia linii produkcyjnej. Początek trzeciej rewolucji przemysłowej to lata sześćdziesiąte XX wieku. Zazwyczaj nazywa się ją rewolucją komputerową lub cyfrową, gdyż katalizatorami dla niej były pojawiające się kolejno: półprzewodniki i duże systemy komputerowe (lata sześćdziesiąte), komputery osobiste (lata siedemdziesiąte i osiemdziesiąte) oraz Internet" [14]. Czwarta rewolucja przemysłowa to zmiana, która radykalnie przekształciła przemysł jaki dotychczas był znany. Przemysłu 4.0 nie próbuje udowodnić, że robotyzacja czy komputeryzacja zrewolucjonizuje fabryki i przedsiębiorstwa – to dzieje się od lat 80-tych i sprawcą tego jest trzecia rewolucja przemysłowa. Nie chodzi również o nowe projekty IT czy korzystanie z aplikacji typu ERP (ang. *Enterprise Resource Planning*) lub programów wykorzystujących przetwarzanie brzegowe - Edge Computing, bo jest to naturalną konsekwencją digitalizacji przemysłu i jego cyfryzacji. Przemysł 4.0 wprowadza transformację cyfrową, co jest zupełnie innowacyjnym podejściem w ujęciu współczesnego przemysłu, ale jednak naturalną konsekwencją zdobyczy technologicznych poprzednich lat i pokoleń. Zbieranie danych z całego cyklu procesu produkcyjnego, pozwala wykorzystywać je do różnych zadań i celów takich jak optymalizacja procesów produkcji czy predykcja utrzymania ruchu linii produkcyjnych. Transformacja cyfrowa konsekwentnie prowadzi także do nowego podejścia w ograniczaniu i planowaniu kosztów funkcjonowania przedsiębiorstwa.

## 2.2. Predykcyjne utrzymanie ruchu

### 2.2.1. Przemysł 4.0

Przemysł 4.0 jest często traktowany jako pojęcie teoretyczne, abstrakcyjne, dlatego warto przedstawić jego podstawy na przykładach praktycznych. Weźmy jako przykład trzy obiekty przemysłowe [21]: sterownik, wentylator powietrza i turbogenerator, które udostępniają swoją reprezentację wirtualną w systemie IT.




Rysunek 2.1: Sterownik kontrolujący proces grzania i jego otoczenie [21]

Pokazany na rysunku 2.1 sterownik jest wykorzystywany w procesie grzania, wyprodukowany w pełni monitorowanym procesie produkcyjnym. Wiążąc go z numerem seryjnym i tworząc dla niego kod QR, możemy go zidentyfikować cyfrowo. Zeskanowanie kodu pozwoli uzyskać informację o miejscu, czasie i sposobie produkcji sterownika oraz o drodze, jaką przebył do urządzenia końcowego. Sterownik będzie można wykorzystać np. w całkowicie cyfrowej linii produkcyjnej.

Wspomniany sterownik możemy również wykorzystać „cyfrowo" gdzie jego kod będzie skanowany po zainstalowaniu w istniejącej strukturze elektroniczno - informatycznej. Informacje o jego czasie pracy, parametrach, producencie i dostawcy będą dostępne w systemie. System diagnostyczno - kontrolny będzie mógł rejestrować jego działanie i oceniać stan techniczny,


Rysunek 2.2: Przykładowy zespół maszynowy – wentylator powietrza i jego otoczenie [21]

tak aby przewidzieć pozostały czas eksploatacji sterownika i zapewnić bezpieczeństwo oraz ciągłość funkcjonowania procesu technologicznego, w którym bierze udział. Gdy prognoza wskaże zwiększone ryzyko awarii, system będzie mógł automatycznie zamówić urządzenie zastępcze. Nowy sterownik będzie można zainstalować w trakcie najbliższej konserwacji urządzenia grzewczego, unikając zbędnego przestoju i strat w produkcji w przypadku awarii [21].

Opisany przykład to jeden z podstawowych elementów idei Przemysłu 4.0, a także przykład zastosowania predykcji utrzymania ruchu. Korzyści płynące z tego typu rozwiązania bez wątpienia są oczywiste. Określenie stanu sprzętu i przewidywanie, kiedy należy przeprowadzić konserwację, są niezwykle istotne. Wdrożenie tego typu rozwiązań może prowadzić do znacznych redukcji kosztów, większej przewidywalności i większej dostępności systemów. Przemysł 4.0 to nowa rzeczywistość wymagająca nowego podejścia i nowych rozwiązań. Predykcyjne utrzymanie ruchu może stać się skuteczną odpowiedzią na nowe wyzwania a w nieodległej przyszłości stanie się standardem w podejściu do utrzymania ruchu.

### 2.2.2. Internet rzeczy - IoT

Współczesny Internet jest siecią maszyn, ale również siecią ludzi. Połączone ze sobą urządzenia służą różnym celom konkretnym użytkownikom. Natomiast Internet rzeczy jest czymś zupełnie innym. To sieć, która łączy ze sobą nie ludzi, a rzeczy - mnóstwo rzeczy [6]. Internet rzeczy (IoT, ang. *Internet of Things*) nie tylko łączy typowe urządzenia komputerowe, ale również mnóstwo innych obiektów. Główna idea Internetu rzeczy to połączenie ze sobą wszystkiego, co może mieć dostęp do sieci. Ta kombinacja

pozwoli wszystkim urządzeniom komunikować się ze sobą, spełniając wiele użytecznych funkcji. Oznacza to, że połączyć można dowolne sprzęty – nie tylko komputery, ale także różne rodzaje czujników i monitorów [6].

Wzajemne połączenie wspomnianych urządzeń może się odbywać w ramach istniejącej infrastruktury internetowej lub innej. Niekiedy budowanie własnej sieci np. bezprzewodowej wewnątrz firmy jest dużo bardziej zasadne i bezpieczne, a ze światem zewnętrznym połączona jest jedynie wydzielona do tego celu komórka. Internet rzeczy koncentruje się przede wszystkim na rzeczach. Zasadniczo „rzeczą" w ramach IoT może być wszystko, co jest wystarczająco duże, aby pomieścić nadajnik bezprzewodowy (korzystający z technologii Wi-Fi, Bluetooth lub dowolnego protokołu bezprzewodowego), i czemu można przydzielić indywidualny adres IP. Może to być coś tak małego jak spinacz lub tak dużego jak dom [6]. W ramach IoT wszystkie połączone ze sobą urządzenia to coś więcej niż tylko grupa pojedynczych części. Całość komunikuje się ze sobą nawzajem w inteligentny i zautomatyzowany sposób. Dowolne urządzenie łączy się z innymi stosownymi urządzeniami w swoim otoczeniu i udostępnia im zgromadzone przez siebie dane. Powstaje wówczas tzw. „inteligentne otoczenie" w sytuacji kiedy kilka urządzeń współdziała ze sobą, wykonując swoje zadania z wykorzystaniem informacji oraz danych dostępnych w ramach sieci. Te działania odbywają się automatycznie, realizując pewne oczekiwane przez użytkowników funkcjonalności jednak nie angażując ich w ten proces.

Internet rzeczy (IoT) odgrywa kluczową rolę w procesie tworzenia skutecznej strategii predykcyjnej. Strumienie danych pochodzące z różnego rodzaju czujników gromadzone są w chmurze, a ich analiza za pomocą modułu z zaimplementowanymi algorytmami sztucznej inteligencji umożliwia przewidywanie z dużą dokładnością ryzyko wystąpienia awarii. Dodatkowo na podstawie danych historycznych i bieżącej analizy występujących awarii można nieustannie usprawniać modele i przewidywać awarie z większą skutecznością oraz z uwzględnieniem zmiennych warunków występujących na hali produkcyjnej.

### 2.2.3. Jak działa predykcja utrzymania ruchu?

W procesach produkcyjnych jednym z ważniejszych zagadnień jest oszacowanie czasu maksymalnej wydajności maszyn – ich czasu bezawaryjnego działania oraz dobór takiej częstotliwości prac konserwacyjnych, aby uniknąć nieoczekiwanych awarii sprzętu. To trudne i złożone zagadnienie, na które można uzyskać miarodajną odpowiedź tylko za pomocą predykcji utrzymania ruchu. Możliwość przewidywania pozostałego okresu użytko-

wania części lub zasobu na podstawie danych w czasie rzeczywistym daje organizacjom bezprecedensowy sposób zarządzania i optymalizacji zasobów serwisowych. Predykcja utrzymania ruchu to strategia, która wykorzystuje narzędzia informatyczne - zasoby programowo-sprzętowe do monitorowania stanu urządzeń w celu wykrywania i rozpoznania nieoczekiwanych anomalii i problemów z wydajnością sprzętu. Na podstawie pomiarów system może uruchomić wstępnie zbudowane algorytmy predykcyjne w celu oszacowania, kiedy element wyposażenia może ulec awarii tak, aby prace konserwacyjne można było wykonać tuż przed tym, jak to nastąpi. Wiedząc, kiedy dana część ulegnie awarii, możemy zaplanować prace konserwacyjne tylko wtedy, gdy jest to rzeczywiście potrzebne, jednocześnie unikając nadmiernej konserwacji i zapobiegając nieoczekiwanej awarii sprzętu.

Istnieją trzy główne komponenty, które umożliwiają śledzenie stanu zasobów i ostrzeganie techników o nadchodzących awariach sprzętu:

1. Zainstalowane czujniki monitorujące wysyłają w czasie rzeczywistym dane dotyczące wydajności i stanu maszyny. Istnieje wiele różnych czujników, które można zainstalować. Mogą one mierzyć prądy elektryczne, wibracje, temperaturę, ciśnienie, poziom oleju, hałas, poziom korozji oraz wiele innych parametrów. Dodatkową zaletą stosowania czujników monitorujących stan jest możliwość dokładnego odwzorowania tego, co dzieje się wewnątrz danego elementu lub samej maszyny, bez jakichkolwiek zakłóceń procesu produkcyjnego. Innymi słowy, nie musimy zatrzymywać maszyny i demontować, aby przeprowadzić fizyczne inspekcje.

2. Internet rzeczy umożliwia komunikację między maszynami, pomaga w gromadzeniu i analizowaniu ogromnych ilości danych. Ważnym elementem jest połączenie komponentów i zasobów z centralnym systemem, który przechowuje napływające informacje. Tak opracowany system komunikuje się w oparciu o szybkie sieci WLAN, LAN lub WiFi, a dane trafiają do chmury. W ten sposób połączone elementy mogą się komunikować, współpracować, analizować dane, zalecać działania naprawcze lub podejmować działania bezpośrednio, w zależności od konfiguracji systemu.

3. Najważniejszą częścią predykcyjnego utrzymania ruchu i prawdopodobnie najtrudniejszą jest budowanie algorytmów predykcyjnych – prognostycznych. Zasadniczo musimy zbudować model, który będzie uwzględniał wiele różnych zmiennych oraz ich wzajemne powiązania i wpływ, co ostatecznie pozwoli przewidzieć awarię maszyn. Im wię-

cej zmiennych możemy użyć, tym dokładniejsze będą modele. Dlatego budowanie modeli predykcyjnych jest procesem iteracyjnym. Zdarza się, że w początkowej fazie pracy systemu może zaistnieć potrzeba zainstalowania czujników monitorujących stan i uruchomienie, aby zebrać dane bazowe i zakończyć wstępną budowę modele predykcyjnego. Z biegiem czasu zainstalowane czujniki będą generować coraz więcej danych, które można wykorzystać do ulepszenia początkowych modeli i tworzenia niemal doskonałych prognoz awarii.
W dużym uproszczeniu opracowane modele:

- przestrzegają zestawu z góry określonych reguł, które porównują obecne zachowanie zasobu z jego oczekiwanym zachowaniem
- informują o każdej anomalii, która wskazuje na stopniowe zużywanie się, elementu co w konsekwencji może doprowadzić do jego uszkodzenia
- na podstawie nieprawidłowości, bieżących warunków pracy, danych dotyczących wcześniejszych awarii i wszystkich innych zmiennych wbudowanych w model danych, algorytmy próbują przewidzieć punkty awarii.

Efektem końcowym jest autonomiczny system, który monitoruje warunki pracy za pomocą zainstalowanych czujników, rozumie i przewiduje wzorce tworzone przez anomalie danych, tworzy ostrzeżenia w przypadku odchylenia od ustalonych progów poprawnej pracy.

Mając na uwadze wszystkie wymienione elementy i zależności niezbędne do opracowania skutecznej predykcji utrzymania ruchu w pracy skupiono się na budowie prostego badawczego systemu mini przekładni, której działanie w różnych środowiskach pracy zostało poddane analizie sygnału akustycznego. Dzięki temu opracowany system - program oparty na algorytmach sztucznej inteligencji - jest w stanie rozpoznać - analizując próbki dźwięku zarejestrowanej pracy przekładni – w jakich warunkach pracuje układ. Budując i analizując tak prosty model, możemy poznać niezbędne narzędzia informatyczne, które skutecznie pozwolą opracować dobrze działający system.

## 2.3. Zastosowanie sztucznej inteligencji w systemach bazujących na przetwarzaniu dźwięku

### 2.3.1. Wirtualny asystent głosowy

Wykorzystywanie algorytmów sztucznej inteligencji, uczenia maszynowego jest szeroko stosowane w systemach rozpoznawania mowy, czy wirtualnych asystentach takich jak Alexa, Siri oraz Google Home. To w dużej mierze produkty zbudowane w oparciu o technologie NLP (ang. *natural language processing*), która pozwala wydobywać kontekst informacji słownej z sygnałów audio, gdzie dodatkowo wykorzystuje się uczenie maszynowe i sztuczną inteligencję. Połączenie tych technologii pozwala poprawnie rozpoznawać i interpretować polecenia głosowe oraz odpowiadać słownie na zadane przez użytkownika pytania czy zadania. Jak to się dzieje? W ten obszar wkracza uczenie maszynowe. Asystentom głosowym aplikowane jest codziennie miliony podobnych zapytań głosowych wraz z kontekstem oraz reakcją, której od nich oczekuje użytkownik. Dzięki temu za każdym kolejnym wywołaniem, asystent statystycznie powinien wiedzieć jakiej reakcji od niego oczekuje użytkownik. Dzięki temu każde obcowanie z asystentem dostarcza mu nowych danych niezbędnych do uczenia się i analizy, które z każdym dniem automatycznie ulepszają działanie tych algorytmów. Przeanalizowano pokrótce sposób działania takich asystentów. Pierwszym i podstawowym etapem jest zbieranie informacji przez asystenta, które przekazujemy mu w trakcie korzystania z systemu. Następnie te informacje są selekcjonowane i grupowane. Dla asystentów głosowych największe znaczenie mają trzy grupy słów informacji:

- intencje – to słowa po, których asystent podejmuje decyzję o tym jaką akcję ma wykonać. To cały zestaw słów takich jak: podaj, zadzwoń, wyświetl itd.

- dialogi – to słowa lub układ słów towarzyszący głównym poleceniom wydawanym asystentom. Dzięki tym „dodatkom" dialogowym korzystanie z asystenta staje się bardziej „ludzkie"

Wszystkie zebrane informacje gromadzone są w „chmurze danych" asystentów. Dane zbierane są w trakcie korzystania z systemu, ale również sam system pozyskuje je „nasłuchując" otoczenie i analizując dźwięki docierające do urządzeń wyposażonych w system. Aktualnie algorytmy sztucznej inteligencji, które obsługują oprogramowanie rozpoznawania mowy stają

się coraz dokładniejsze w procesie rozpoznawania, przetwarzania i odpowiadania na żądania w języku naturalnym.

### 2.3.2. Algorytmy rozpoznawania treści audio Content ID

Content ID to zautomatyzowany, skalowalny system serwisu YouTube, który umożliwia kontrolę praw autorskich właścicielom. Oznacza to, że system rozpoznaje w zamieszczanym przez dowolnego użytkownika filmie muzykę, dźwięki, które już funkcjonują w YouTube i są objęte prawami autorskimi. YouTube przyznaje Content ID tylko właścicielom praw autorskich, którzy spełniają określone kryteria. Aby użytkownik mógł uzyskać taki identyfikator, musi posiadać wyłączne prawa do zamieszczanych części oryginalnych materiałów. YouTube określa również wyraźne wytyczne dotyczące korzystania z systemu Content ID. Na bieżąco monitoruje wykorzystanie systemu Content ID i kontroluje zamieszczane utwory. System działania jest stosunkowo prosty.

Użytkownik (właściciel, twórca muzyki lub filmu) przesyła do serwisu YouTube specjalnie przygotowany plik DDEX[2], który w przypadku muzyki składa się z [8]:

- nagrania audio,
- metadanych (czyli opis tego „co jest" w nagraniu – tytuł, wykonawcy, twórcy itp.),
- informacji o prawach do nagrania (kto reprezentuje nagranie na danym terytorium),
- „match politicy" – czyli co YouTube ma „zrobić" jeśli inny użytkownik umieści film z danym nagraniem.

Serwis YouTube przesłane dane zapisuje w swoich bazach, a następnie dla przesłanego nagrania generuje specjalny kod *„accoustic fingerprint"* czyli dźwiękowy odcisk palca, który jest kompaktową sygnaturą, opartą na zawartości. Technologia ta umożliwia również monitorowanie dźwięku niezależnie od jego formatu i bez konieczności osadzania metadanych lub znaków

[2]Digital Data Exchange (DDEX) to organizacja członkowska typu non-profit, która powstała w 2006 roku i koncentruje się na tworzeniu standardów cyfrowego udostępniania muzyki. DDEX została założona przez konsorcjum wiodących firm medialnych, organizacji licencyjnych muzykę, właścicieli praw autorskich, dostawców usług cyfrowych i pośredników technicznych. DDEX jest obecnie standardem formatowania i dostarczania metadanych związanych z cyfrową dystrybucją muzyki w sieci Internet, a jego założenia są wdrażane na całym świecie.

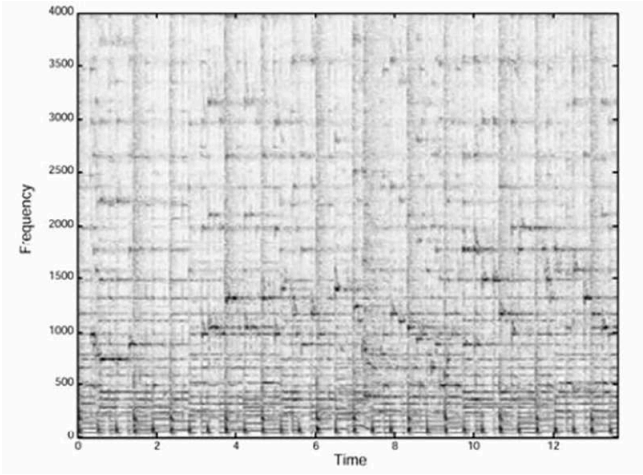
wodnych. Od tego momentu każdy nowo umieszczany plik dźwiękowy lub wideo otrzymuje taki „elektroniczny odcisk", który jest porównywany z danymi zapisanymi w bazie danych i sprawdzany czy przypadkiem YouTube nie posiada już referencji z takim nagraniem. Jeśli tak – serwis stosuje politykę, którą właściciel praw przekazał w zgłoszonym pliku.

Dźwiękowe „odciski palców" można wykonać na podstawie transformaty Fouriera (widma częstotliwości) danego utworu. Jeśli użyjemy główną ścieżkę dźwiękową jako punkt odniesienia, a następnie wygenerujemy mapę rozkładu częstotliwości w różnych punktach w czasie, możemy spróbować skorelować to z rozkładem częstotliwości testowanej próbki. W zależności od algorytmu możemy dokonać dopasowania i wybrać próbki, które są „wystarczająco blisko", aby zagwarantować dokładniejszą analizę. Zaletą tej metody jest to, że nie wymaga dokładnego dopasowania. Wszystko, czego szuka algorytm, to pewna korelacja między danymi z analizy próbki wzorcowej, a sygnałem próbki testowanej. Ta korelacja może wystąpić nawet w środku utworu - nie musisz dokonywać porównania w dokładnie tym czasie. Co więcej, system YouTube Content ID jest inteligentny i operuje również na określonych unikalnych pasmach częstotliwości, dzięki czemu uzyskuje pewien stopień eliminacji szumów, zakłóceń itd. Dzięki temu każda modyfikacja i zmiana utworu chronionego i przesyłana do serwisu jako nowy utwór (np. remix utworu) jest rozpoznawana jako utwór chroniony pomimo ewidentnych modyfikacji.

Część użytkowników serwisu YouTube próbowało oszukać system poprzez zmiany rozkładu częstotliwości, przesunięcia czasowe, przyśpieszanie lub zwalniania tempa utworu, czy maskowanie pasm, jednak algorytmy Content ID są algorytmami AI dzięki czemu po pewnym czasie te metody przestały być skuteczne. Jednak to rozwiązanie ma też swoje słabe strony. Algorytmy egzekwujące prawa autorskie, mogą generować tzw. „fałszywe alarmy" i sygnalizować naruszenie praw nawet w takich sytuacjach, gdzie jest to prawnie dozwolone i wówczas a powiązany z plikiem referencyjnym plik audio jest nieprawidłowo blokowany. Zjawisko fałszywych alarmów jest szczególnie problematyczne dla autorów piosenek, kompozytorów, artystów muzyki eksperymentalnej i innych, którzy tworzą muzykę łącząc własne wykonanie wokalne lub instrumentalne z utworami stworzonymi przez innych artystów, których nagrania zostały już zarejestrowane w systemie YouTube [4].

W procesie analizy i weryfikacji plików audio w systemie Content ID możemy wyróżnić kilka etapów, które są zależne od siebie i muszą następować w ustalonej kolejności.

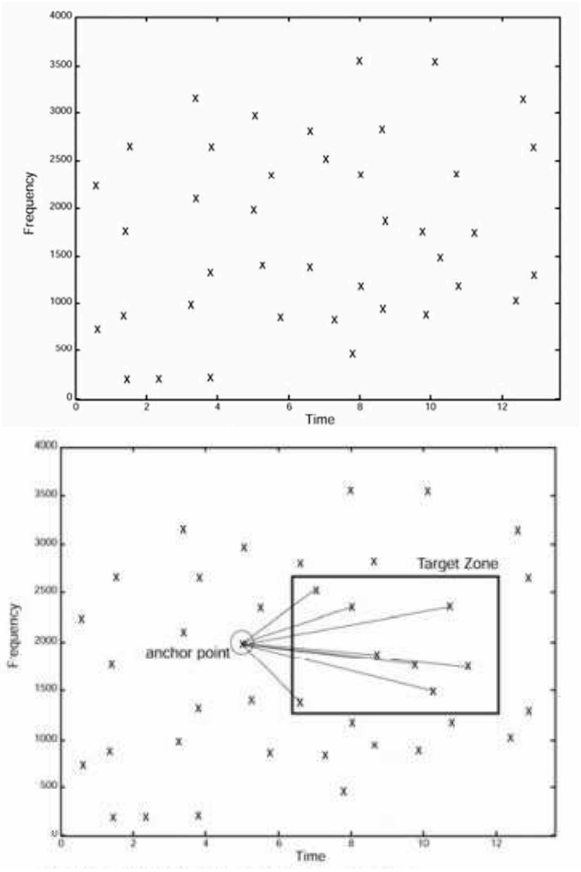
1. Wstępna analiza i przetwarzanie pliku audio - polega zazwyczaj na bardzo prostej, obróbce nagrania dotyczy takich działań jak konwersja formatu, konwersja dźwięku stereo na mono, zmiana częstotliwości próbkowania. Mogą być również zaimplementowane algorytmy normalizacji i kompresji dźwięku oraz wstępnego odszumiania. Wizualizacją tego procesu może być spektrogram wygenerowany z wykorzystaniem transformaty Fouriera.




Rysunek 2.3: Spektrogram pliku audio [5]

2. Wyszukiwanie maksimów lokalnych - polega na analizie położenia prążków spektralnych, a także punktów charakterystycznych o największej amplitudzie. W tym procesie wyodrębnianych jest kilkadziesiąt do kilkuset punktów charakterystycznych na sekundę i z reguły generuje się ich więcej, ale wybiera się jedynie te o największej intensywności.

3. Haszowanie to funkcje skrótu pozwalająca na ustalenie krótkich i łatwych do weryfikacji sygnatur dla dowolnie dużych zbiorów danych. Muzyczny odcisk palców jest tworzony z wygenerowanej mapy, w której pary punktów czasowo - częstotliwościowe są kombinatorycznie powiązane. Punkty kontrolne są wybierane, a każdy z nich jest przypisany odpowiedniej strefie docelowej. Każdy punkt kontrolny w strefie głównej jest sekwencyjnie parowany z punktami w swojej strefie docelowej, a każda para daje dwie składowe częstotliwości, oraz wyliczoną różnicę czasową między punktami [15].




Rysunek 2.4: Wyznaczanie maksimów lokalnych [17]

Współrzędne czasowe:

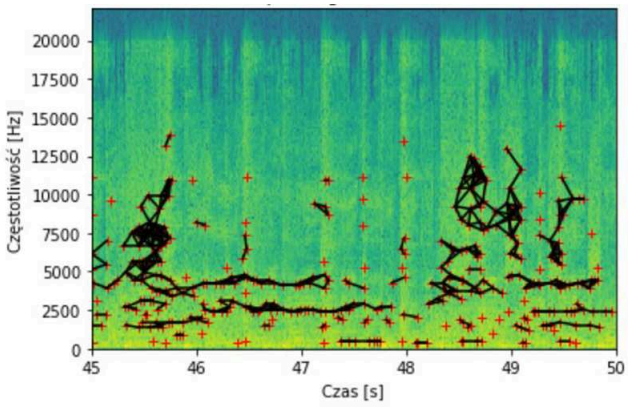
$$T_a < T_b < T_a + \Delta T$$

Współrzędne częstotliwościowe:

$$F_a - \Delta F \leq F_b \leq F_a + \Delta F$$

W wyniku tak powstałych punktów kształtuje się mapa haszy.

W wyniku procesu haszowania dla każdego nagrania można wygenerować listę rekordów i zapisać ją w bazie danych. Ta lista może składać się z następujących elementów [17]:




Rysunek 2.5: Mapa haszy [17]

- początek hasza,
- początek hasza z częstotliwości w Hz,
- długość hasza w sekundach,
- długość hasza w półtonach.

Porównywanie rekordów - jest ostatnim etapem w łańcuszku zadań algorytmu. Po tych wszystkich operacjach, wyszukiwanie podobnego nagrania sprowadza się do zapytania SQL[3] i znalezienia wspólnych haszów – w nagraniu badanym i w nagraniach referencyjnych. Tak znalezione rekordy sprawdza się pod kątem ich sekwencji w czasie, czy hasze, które występują po sobie w badanym nagraniu występują również w odpowiedniej kolejności w nagraniu referencyjnym, czy może ich wystąpienie to zwykły przypadek.

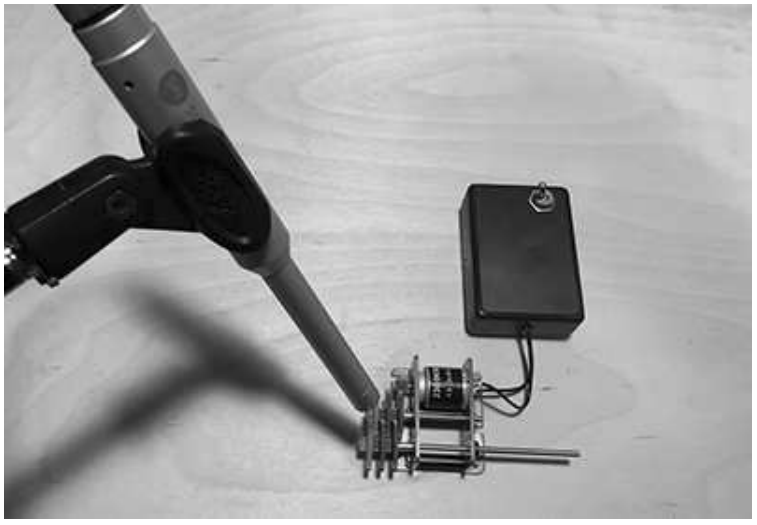
Podsumowując działający system Content ID jest przykładem tego w jaki sposób analizując sygnał dźwiękowy oraz poddając go odpowiednim przetworzeniom cyfrowym możemy rozpoznawać dość precyzyjnie nagrane dźwięki. Ta identyfikacja jest realizowana nie na zasadzie stałych wytycznych i cech charakterystycznych wygenerowanych w sposób niezmienny na bazie próbki sygnału, ale na zasadzie „dynamicznego" uczenia się zmian zachodzących w próbkach. Od pewnego czasu serwis YouTube korzysta równie z metod uczenia maszynowego oferowanego przez potentata jakim jest Google. Pozwala to na wyodrębnianie z próbek cech charakterystycznych

[3](ang. *Structured Query Language*) – strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych.

dla danych dźwięków, a następnie analizowanie i korelację – porównanie z próbkami umieszczonymi w bazie. Na tej podstawie algorytm „wyciąga wnioski" i podejmuje decyzję na ile to porównanie jest trafne i czy jednoznacznie może uznać przeszukiwanie za zakończone.

### 2.3.3. Struktura cyfrowych plików audio - sampling

Co określamy pojęciem dane audio? Bezpośrednio lub pośrednio, zawsze mamy kontakt z dźwiękiem. Nasz mózg nieustannie przetwarza i rozpoznaje dźwięki przekazując informację o środowisku i zachodzących w nim procesach. Prostym przykładem mogą być rozmowy z ludźmi jako sposób komunikowania się z innymi. Słowa są słyszane i rozpoznawane przez rozmówcę, a następnie interpretowane i rozumiane. Narząd słuchu potrafi bardzo precyzyjnie wyłapać z otoczenia wszelkie dźwięki, szmery, szumy i inne „zakłócenia", które są obecne w trakcie rozmowy czy komunikacji. Jednak mózg wie jak filtrować te dane i w procesie analizy interpretuje jedynie te, który mają w konkretnym momencie znaczenie i niosą faktyczną treść. Jednym z kluczowych elementów uczenia maszynowego jest odpowiednia preparacja i przygotowanie danych wejściowych. Danych, które pozwolą na efektywne nauczenie algorytmu wartości wzorcowych dla danych stanów przekładni.



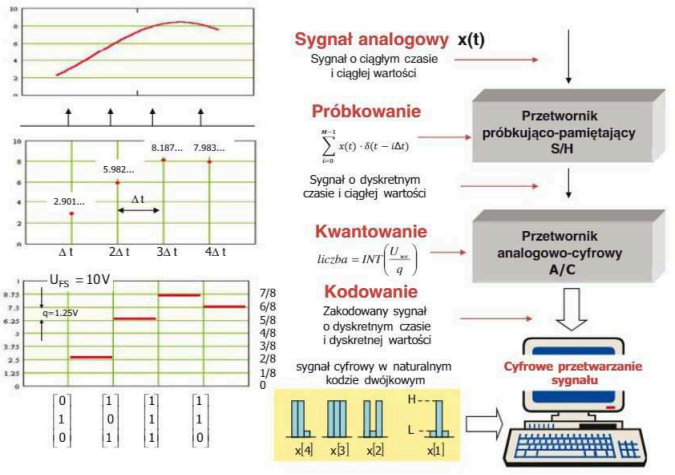
Rysunek 2.6: Mini przekładania z mikrofonem pomiarowym Sonarworks XREF-20

W jaki sposób zdobyć dane potrzebne do doświadczenia? Dźwięk, aby mógł być wykorzystany do przetwarza musi zostać odpowiednio zdigitali-

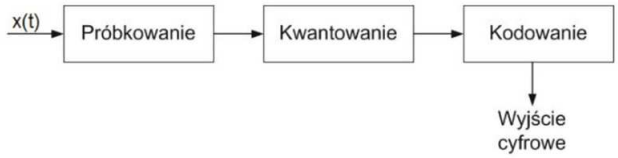
zowany z postaci analogowej do cyfrowej. Dzięki zastosowaniu układu rejestrującego dźwięk (mikrofon, interfejs audio, komputer z oprogramowaniem nagrywającym) możliwie było zarejestrowanie działania przekładni i wykonania odpowiednich próbek. Wszystko zaczyna się w naturalnym otoczeniu przekładni, które zostało odpowiednio przygotowane - wyizolowane i wolne od zakłóceń otoczenia. Znamy doskonale mechanizm rejestracji dźwiękowych sygnałów analogowych, gdzie zmiany ciśnienia wywoływane w otoczeniu mikrofonu powodują ruch membrany i są zamieniane na zmiany napięcia. Ten proces analizowany jest przez konwerter analogowo - cyfrowy, a w konsekwencji zamieniany na postać binarną, która opisuje, gdzie w danym ułamku sekundy znajdując się właśnie fala.

### 2.3.4. Proces digitalizacji próbki dźwiękowej działającej przekładni

Cechą charakterystyczną współczesnego przetwarzania sygnałów jest postępująca dominacja metod cyfrowych, które dokonują wszelkich operacji na sygnale jako operacji na ciągach binarnych reprezentujących ten sygnał [19]. Ważnym etapem tego procesu jest dyskretyzacja - przetwarzanie sygnału analogowego na sygnał cyfrowy czyli dyskretny. Sygnał analogowy, będący funkcją ciągłą określonego parametru, najczęściej względem czasu, podlega próbkowaniu.




Rysunek 2.7: Dyskretyzacja sygnału analogowego [3]



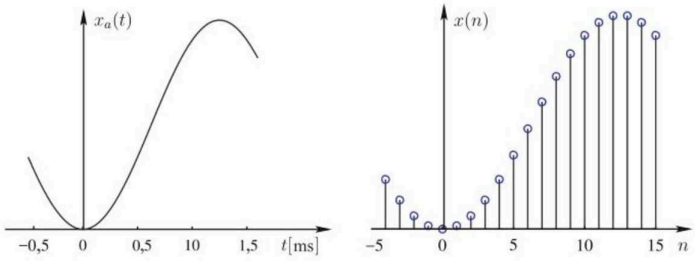

Rysunek 2.8: Przetwarzanie analogowo-cyfrowe [3]

Przetwarzanie analogowo cyfrowe dzielimy na trzy etapy: próbkowanie, kwantowanie, kodowanie.

**Próbkowanie** – polega na pobieraniu w równych odstępach czasu próbki sygnału analogowego i rejestrowaniu ich chwilowych wartości. Odbywa się w układzie próbkująco - pamiętającym, który może być elementem odrębnym lub zintegrowanym z przetwornikiem analogowo - cyfrowym. Próbkowanie możemy nazwać dyskretyzacją w czasie [3].

**Kwantowanie** – to proces odwzorowywania próbkowanych analogowych wartości napięcia na dyskretne poziomy napięcia, które są następnie reprezentowane przez liczby binarne (bity). To niezbędny proces, ponieważ analogowe wartości to liczby rzeczywiste występujące w kontinuum.

**Kodowanie** – polega na przypisaniu poziomom kwantowania zakodowanych liczb – słów kodowych, przy czym każdej próbce odpowiada określony poziom kwantowania i słowo kodowe.

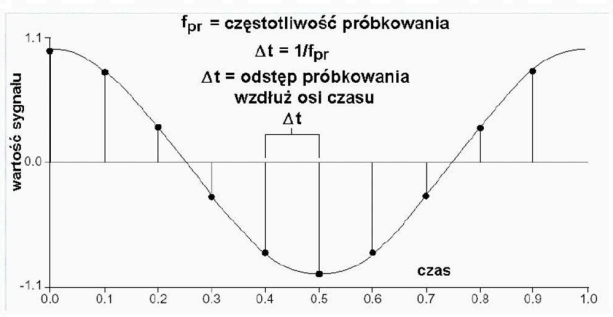



Rysunek 2.9: Sygnał analogowy $xa(t)$ i sygnał dyskretny $x(n)$ otrzymany przez próbkowanie $xa(t)$ z okresem $Ts$ dla $Ts = 1/10000s$ [3]

Inne bardzo ważne parametry dla procesu digitalizacji sygnału to częstotliwość próbkowania i głębia bitowa.

**Częstotliwość próbkowania** $f_s$ jest średnią liczbą próbek uzyskanych w czasie jednej sekundy (próbek na sekundę), a zatem $f_s = 1/T$.

W przypadku systemów cyfrowych górną granicę odpowiedzi określa




Rysunek 2.10: Częstotliwość próbkowania w czasie t [16]

częstotliwość próbkowania. Wybór częstotliwości próbkowania w systemie cyfrowym opiera się na twierdzeniu o próbkowaniu Nyquista – Shannona. Próbkowany sygnał może zostać odtwarzany dokładnie, jedynie w przypadku kiedy jest próbkowany z częstotliwością większą niż dwukrotność szerokości pasma sygnału, częstotliwości Nyquista.

$$[\mathrm{h}]\ f_p > 2f_s$$

Dlatego częstotliwość próbkowania 40 kHz jest matematycznie wystarczająca do uchwycenia wszystkich informacji zawartych w sygnale o składowych częstotliwości mniejszych lub równych 20 kHz. Szerokość pasma zapewniana przez częstotliwość próbkowania 44 100 Hz używaną przez standard dla płyt audio CD jest wystarczająco szeroka, aby pokryć cały zakres ludzkiego słuchu, który kształtuje się w przedziale od 20 Hz do 20 kHz.

Istnieje kilka standardów związanych z dźwiękiem cyfrowym. W przypadku komercyjnych płyt audio CD, producenci ustalili standard częstotliwości próbkowania 44,1 kHz z plikiem 16 bitowym, w oparciu o fakt, że najwyższa rejestrowana częstotliwość stanowi połowę częstotliwości próbkowania.

**Głębia bitowa** – w przypadku dźwięku cyfrowego z modulacją kodu impulsowego (PCM) to liczba bitów informacji w każdej próbce bezpośrednio odpowiadającej rozdzielczości każdej próbki. Próbki dźwięków – (sample) to amplituda fali w określonym przedziale czasu, gdzie głębia bitowa określa, jak szczegółowa będzie próbka, określana również jako zakres dynamiczny sygnału (w odniesieniu do danych audio jest to zwykle 16-bitowy, co oznacza, że próbka może mieć rozdzielczość 65 536 możliwych wartości). Głębia bitowa ma znaczenie tylko w odniesieniu do sygnału cyfrowego PCM.

Formaty inne niż PCM, takie jak formaty kompresji stratnej, nie mają skojarzonych głębi bitowych. Dlatego też w pracy próbki poddawane analizie i rozpoznaniu będą w formacie niekompresowanym wave, mono z częstotliwością próbkowania 22 kHz i rozdzielczością bitową 16 bitów. Głębia bitowa ogranicza również stosunek sygnału do szumu, rekonstruowanego sygnału do maksymalnego poziomu określonego przez błąd kwantyzacji. Ponadto głębia bitowa nie ma wpływu na pasmo przenoszenia, które jest ograniczone przez częstotliwość próbkowania.

Tablica 2.1: Przybliżony stosunek sygnału do szumu w relacji do rozdzielczości bitowej

| Rozdzielczość bitowa | Przybliżony stosunek sygnału do szumu |
|---|---|
| 4 | 24,08 dB |
| 8 | 48,16 dB |
| 12 | 66,22 dB |
| 16 | 96,33 dB |
| 20 | 120,41 dB |
| 24 | 144,49 dB |
| 32 | 192,66 dB |

Podczas digitalizacji sygnału analogowego, każda wartość jest przybliżana do najbliższej wartości dyskretnej. W związku z niedokładnością przypisania wartości powstaje błąd określany jako szum kwantyzacji. Szum ten może być zmniejszony poprzez zwiększenie liczby bitów opisujących każdą próbkę. Zwiększenie liczby bitów o jeden powoduje dwukrotne zwiększenie liczby poziomów kwantyzacji i w rezultacie zmniejszenie szumu kwantyzacji.

### 2.3.5. Sztuczna inteligencja – szansa czy zagrożenie?

Sztuczna inteligencja próbuje sprawić, by maszyny-komputery wykonywały te same czynności, które wykonują ludzie. Niektóre z tych czynności (np. rozumowanie) zazwyczaj są opisywane jako „inteligentne” [1], niektóre zaś takiej jak widzenie w opinii specjalistów inteligentne nie są. Ale wszystkie wymagają umiejętności psychologicznych, takich jak percepcja, kojarzenie, przewidywanie, planowanie, kontrola motoryczna, co umożliwia ludziom i zwierzętom realizowanie ich celów. Inteligencja stanowi bogato ustrukturowaną przestrzeń różnorodnych zdolności przetwarzania informacji. Stosownie do tego sztuczna inteligencja wykorzystuje wiele różnych technik,

realizując wiele różnych zadań. Praktyczne zastosowania sztucznej inteligencji można znaleźć w domu, samochodzie, biurze, szpitalu i w Internecie. Część z nich znajduje się poza naszą planetą, np. roboty wysłane na Księżyc i na Marsa czy satelity krążące w przestrzeni kosmicznej. Hollywoodzkie animacje, gry wideo i gry komputerowe, systemy nawigacji satelitarnej czy wyszukiwarka Google – wszystko to opiera się na technikach sztucznej inteligencji. Podobnie jak systemy wykorzystywane przez finansistów do przewidywania ruchów na giełdzie i przez rządy państw narodowych do wdrażania decyzji politycznych dotyczących zdrowia i transportu, podobnie jak aplikacje w telefonach.
Sztuczna inteligencja ma dwa główne cele[1]:

- **technologiczny** – sprawienie, by komputery i maszyny sterowane nimi wykonywały pewne działania mające zastosowanie w technice,
- **naukowy** – polega na wykorzystaniu pojęć i modeli sztucznej inteligencji w taki sposób, by pomogły one odpowiedzieć na pytania dotyczące ludzi i innych żywych istot.

## 2.4. Python – główne narzędzie uczenia maszynowego.

### 2.4.1. Uczenie maszynowe

Uczenie maszynowe (ang. *machine learning*) — zajmuje się teoretycznym i praktycznym zastosowaniem algorytmów analizujących dane, a mówiąc inaczej to jeden ze sposobów na odkrywanie tego co nieznane. Obecnie stanowi jedną z najciekawszych dziedzin informatyki. Aktualnie przetwarza się ogromne ilości danych, a za pomocą samo-uczących się algorytmów będących częścią uczenia maszynowego informacje te są przekształcane w rzeczywistą wiedzę. Dzięki licznym i potężnym bibliotekom o jawnym kodzie źródłowym, które powstały w ostatnich latach, prawdopodobnie teraz jest najlepszy czas, aby zainteresować się uczeniem maszynowym i nauczyć się wykorzystywać potężne algorytmy do wykrywania wzorców w przetwarzanych danych oraz prognozować przyszłe zdarzenia. Uczenie maszynowe wyewoluowało z badań nad sztuczną inteligencją, w których projektowano samo-uczące się algorytmy, zdolne do pozyskiwania wiedzy z informacji oraz tworzenia na ich podstawie prognoz [11].

Uczenie maszynowe czyli uczenie na przykładach okazało się skutecznym sposobem radzenia sobie z problemami, dla których znamy algoryt-

miczne rozwiązania, ale próby zaimplementowania takich algorytmów były bardzo kosztowne lub nieskuteczne. Takim przykładem może być system wykrywania możliwych oszustw popełnianych za pomocą kart kredytowych [20]. Uczenie maszynowe ma bardzo dużo potencjalnych zastosowań, ale nie zawsze może być skutecznie zrealizowane. W sytuacji, gdy nie dysponujemy odpowiednią liczbą wartościowych danych niestety nie uda nam się „wytrenować" modelu ucznia maszynowego [20]. Idea uczenia maszynowego polega na opracowaniu programu, który otrzyma pewne dane wejściowe. W oparciu o te dane program sam stworzy pewien model - równanie w oparciu, o które będzie próbował wyliczyć pewne wartości jakich można by się spodziewać w przyszłości. Z punktu widzenia programisty został napisany jeden program, ale ze względu na wprowadzone dane do przetwarzania zostaną wyznaczone różne modele i będą wykonywane różne predykcje. Tak właśnie w dużym uproszczenie działa uczenie maszynowe – program czyta dane, odkrywa w nich pewne zależności, buduje swój własny model, a następnie otrzymując nowe dane wejściowe próbuje wyznaczyć dane wyjściowe.

Zastosowanie zebranych danych i obliczeń jest ogromne. Coraz częściej tego typu algorytmy są szeroko stosowane w predykcji utrzymania, funkcjonowania i działania wielu urządzeń oraz maszyn w przedsiębiorstwach. Są to rozwiązania sprzętowo-programowe, w których gromadzone dane z czujników i elementów pomiarowych służą algorytmom do głębokich analiz, a w efekcie prognozowaniu awarii czy rozpoznaniu prac konserwacyjnych.

W uczeniu maszynowym jest pewnego rodzaju zależność ilości gromadzonych danych - wzorców w kontekście wydajności sprzętowej maszyn obliczeniowych. Uczenie maszynowe polega na wykrywaniu ukrytych w przykładach treningowych wzorców im większą ilością danych dysponujemy, tym lepsze osiągniemy rezultaty. Jednak wraz ze wzrostem ilości danych treningowych rosną również wymagania obliczeniowe. Problem wydajności obliczeń rozwiązały akceleratory graficzne ogólnego zastosowania (procesory GPU) [15]. Przewagą GPU nad CPU jest ich równoległa architektura pozwalająca na wykonywanie w ciągu sekundy tysięcy razy większej liczby prostych operacji matematycznych, takich jak mnożenie macierzy. Ważnym punktem na ścieżce rozwoju uczenia maszynowego było opracowanie nowych algorytmów pozwalających maszynom efektywnie uczyć się nie tylko na podstawie danych treningowych, lecz przede wszystkim na podstawie samodzielnie wyodrębnionych z tych danych abstrakcyjnych właściwości. Rozpoczęła się trwająca do dziś era głębokiego uczenia maszynowego (ang. *deep learning, DL*), w której sztuczna inteligencja uczy się na podstawie automatycznie wykrywanych abstrakcyjnych cech, z wykorzystaniem ta-

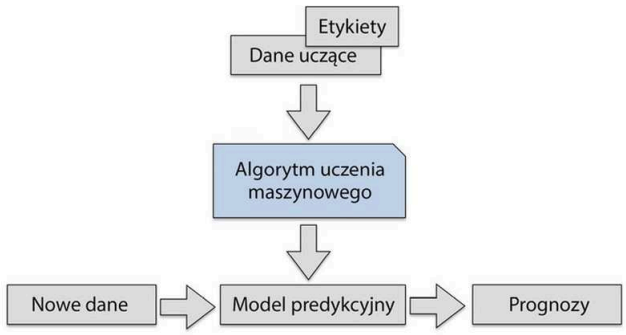
kich algorytmów uczenia maszynowego jak konwolucyjne sieci neuronowe i rekurencyjne sieci neuronowe [20].

Uczenie maszynowe ma również ogromne zastosowanie w życiu codziennym. Dzięki metodom opracowanym pod kątem uczenia maszynowego możemy stosować zaawansowane filtry antyspamowe w poczcie elektronicznej. Dodatkowo możemy wykorzystywać oprogramowanie do rozpoznawania mowy, tekstu, twarzy, rzetelnymi silnikami wyszukiwarek internetowych, wymagającymi programami szachowymi. Zanim zagłębimy się w analizę i zastosowanie algorytmów ucznia maszynowego, warto poznać metody i sposoby pracy tych algorytmów. W uczeniu maszynowym wyróżnić możemy trzy podstawowe jego odmiany:

- uczenie nadzorowane,
- uczenie nienadzorowane,
- uczenie przez wzmacnianie

### 2.4.2. Uczenie nadzorowane

Głównym zadaniem uczenia nadzorowanego jest uczenie modelu - programu za pomocą oznaczonych danych tzw. danych uczących, co pozwala przewidywać niewidoczne lub wygenerowane w przyszłości informacje. Tego typu uczenie odnosi się do zestawu próbek, w których pożądane dane wyjściowe są znane.




Rysunek 2.11: Ogólny schemat uczenia nadzorowanego [11]

Uczenie nadzorowane wykorzystuje zestaw szkoleniowych danych do uczenia modelu w celu uzyskania pożądanych wyników. Ten zestaw danych obejmuje dane wejściowe i prawidłowe wyniki, które umożliwiają modelowi

naukę w czasie. Algorytm mierzy swoją dokładność za pomocą funkcji strat, dostosowując się, aż błąd zostanie dostatecznie zminimalizowany. Uczenie nadzorowane można podzielić na dwa typy problemów podczas eksploracji danych - klasyfikację i regresję.

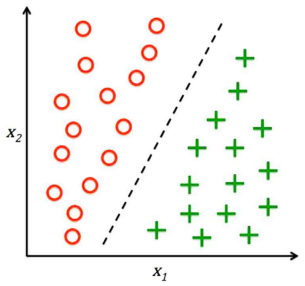
**Klasyfikacja** wykorzystuje algorytm, aby dokładnie przypisać dane testowe do określonych kategorii. Rozpoznaje określone jednostki w zbiorze danych i próbuje wyciągnąć wnioski na temat tego, jak te jednostki powinny być oznaczone lub zdefiniowane. Inaczej mówiąc klasyfikacja stanowi podkategorię uczenia nadzorowanego służącą do przewidywania etykiet klas w nowych wystąpieniach na podstawie dotychczasowych obserwacji. Etykiety klas to dyskretne, nieuporządkowane wartości, które określają przynależność poszczególnych instancji do wyznaczonych grup [11].

Dodatkowo klasyfikację możemy podzielić na klasyfikację binarną i wieloklasową.

**Klasyfikacja binarna** odnosi się do tych zadań klasyfikacyjnych, które mają dwie etykiety klas. Przykładami tej klasy mogą być:

- wykrywanie spamu w wiadomościach e-mail (spam lub nie),
- przewidywanie rezygnacji (tak czy nie),
- przewidywanie działania (np. kup lub nie).

Zazwyczaj zadania klasyfikacji binarnej obejmują jedną klasę, która jest stanem normalnym i inną klasę, która jest stanem negacji (tak/nie czyli 0 – 1). Klasie stanu normalnego przypisywana jest etykieta klasy 0 (+), a klasie zanegowanej jest przypisywana etykieta klasy 1(o).

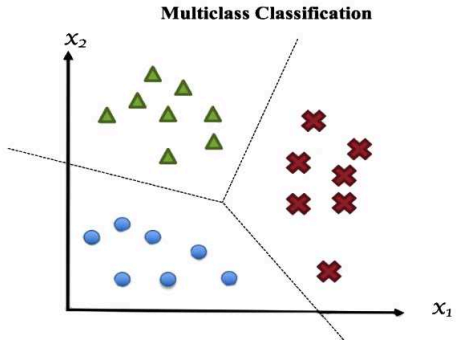



Rysunek 2.12: Przykład klasyfikacji binarnej [11]

**Klasyfikacja wieloklasowa** odnosi się do tych zadań klasyfikacyjnych, które mają więcej niż dwie etykiety klas. Przykłady obejmują:

- klasyfikacja twarzy.
- klasyfikacja gatunków roślin.
- optyczne rozpoznawanie znaków.

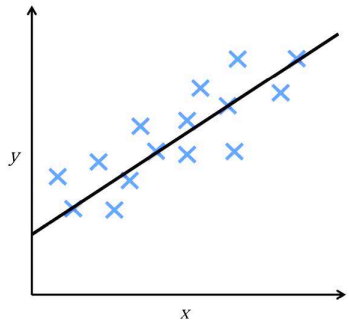
W przeciwieństwie do klasyfikacji binarnej, w klasyfikacji wieloklasowej nie występuje pojęcie wyników normalnych i nieprawidłowych. Zamiast tego przykłady są klasyfikowane jako należące do jednej z wielu znanych klas. W przypadku niektórych problemów liczba etykiet klas może być bardzo duża. Na przykład model może przewidywać, że zdjęcie należy do jednej z tysięcy lub dziesiątek tysięcy twarzy w systemie jej rozpoznawania. Problemy, które obejmują przewidywanie sekwencji słów, takie jak modele tłumaczenia tekstu, mogą być również uważane za szczególny rodzaj klasyfikacji wieloklasowej. Każde przewidywane słowo w sekwencji słów, obejmuje klasyfikację wieloklasową, w której rozmiar słownika określa liczbę możliwych klas, które można przewidzieć i które mogą mieć rozmiar dziesiątek lub setek tysięcy słów.




Rysunek 2.13: Przykład klasyfikacji wieloklasowej [11]

Regresja liniowa to nadzorowany algorytm uczenia maszynowego, w którym przewidywane dane wyjściowe mogą być aproksymowane przy pomocy pewnej liniowej zależności w określonym zbiorze funkcji. Służy do przewidywania wartości w ciągłym zakresie (np. sprzedaż, cena), a nie do klasyfikowania ich na kategorie (np. kot, pies). Istnieją dwa główne typy:

- prosta regresja liniowa wykorzystuje tradycyjną postać kierunkową, gdzie m i b są zmiennymi, których algorytm będzie się starał „nauczyć", aby uzyskać najdokładniejsze przewidywania. X reprezentuje dane wejściowe, y reprezentuje przewidywania.




Rysunek 2.14: Przykład regresji liniowej [11]

$$y = mx + b$$

- regresja wielowymiarowa jest bardziej złożonym równaniem liniowym z wieloma zmiennymi. Przykładem może być poniższe równanie, gdzie $w$ reprezentuje współczynniki lub wagi, których opracowany model spróbuje się nauczyć.

$$f(x, y, z) = w_1 x + w_2 y + w_3 z f(x, y, z) = w_1 x + w_2 y + w_3 z$$

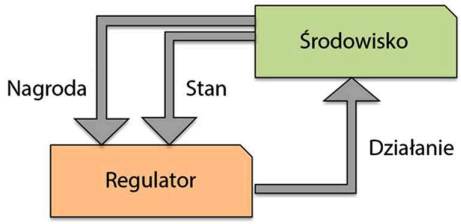
  Zmienne x, y, z reprezentują atrybuty lub odrębne informacje, które posiadamy o każdej obserwacji. W prognozach sprzedaży te atrybuty mogą obejmować wydatki na reklamę firmy w radiu, telewizji i gazetach.

$$\text{sprzedaż} = w_1 \cdot \text{Radio} + w_2 \cdot \text{TV} + w_3 \cdot \text{Gazety}$$

### 2.4.3. Uczenie przez wzmocnienie

Kolejnym rodzajem uczenia maszynowego jest uczenie przez wzmacnianie. W tym przypadku celem jest utworzenie systemu nazywanego agentem lub regulatorem, który poprawia własną skuteczność na podstawie interakcji ze środowiskiem. Informacje na temat bieżącego stanu środowiska zazwyczaj zawierają także tzw. sygnał nagrody. Poprzez oddziaływanie ze środowiskiem regulator może wykorzystywać uczenie przez wzmacnianie do trenowania szeregu działań dążących do maksymalizowania nagrody metodą prób i błędów [11]. Uczenie się ze wzmocnieniem ma zastosowanie w wielu złożonych problemach, których nie można rozwiązać za pomocą innych algorytmów uczenia maszynowego. Uczenie ze wzmocnieniem jest

bliżej sztucznej inteligencji, ponieważ posiada zdolność poszukiwania długoterminowego celu, jednocześnie samodzielnie eksplorując różne możliwości.




Rysunek 2.15: Przykład oddziaływania w modelu uczenia przez wzmacnianie [11]

Główne zalety uczenia ze wzmocnieniem [18]:

- Koncentruje się na problemie jako całości. Konwencjonalne algorytmy uczenia maszynowego są zaprojektowane tak, aby wyróżniać się w określonych podzadaniach, bez pojęcia pełnego obrazu. Uczenie ze wzmocnieniem nie dzieli problemu na podproblemy, działa bezpośrednio, aby zmaksymalizować długoterminową nagrodę. Ma oczywisty cel.
- Nie wymaga oddzielnego etapu zbierania danych. W uczeniu ze wzmocnieniem dane treningowe uzyskuje się poprzez bezpośrednią interakcję agenta ze środowiskiem. Dane szkoleniowe to doświadczenie agenta uczącego się, a nie oddzielny zbiór danych, które muszą zostać wprowadzone do algorytmu. Znacząco odciąża to osobę nadzorującą odpowiedzialną za proces szkolenia.
- Działa w dynamicznych, niepewnych środowiskach. Algorytmy uczenia ze wzmocnieniem są z natury adaptacyjne i zbudowane tak, aby reagować na zmiany w środowisku. W tego typu uczeniu maszynowym czas ma znaczenie.

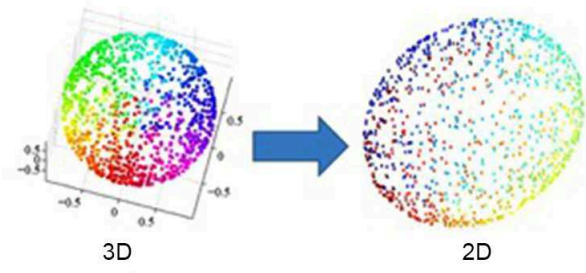
### 2.4.4. Uczenie nienadzorowane

Uczenie nienadzorowane to uczenie maszyny przy użyciu informacji, które nie są sklasyfikowane ani opatrzone etykietą oraz pracy algorytmu na tych informacjach bez wskazówek. Zadaniem maszyny jest grupowanie niesortowanych informacji według podobieństw, wzorców i różnic bez wcześniejszego uczenia danych. W przeciwieństwie do nadzorowanego uczenia się, tu

nie ma danych uczących, co oznacza, że maszyna nie zostanie przeszkolona. Maszyna jest ograniczona do samodzielnego znajdowania ukrytej struktury w nieoznaczonych danych [20]. Rozważmy przykład, w którym maszyna otrzymuje obraz przedstawiający zarówno wkręty, jak i śruby. Algorytm nie zna tych danych i nigdy wcześnie taki obraz nie był przez niego analizowany. W związku z tym maszyna nie ma pojęcia o cechach wkrętów i śrub, więc nie może sklasyfikować rozpoznanych obiektów jako „wkręty i śruby". Ale może podzielić je na kategorie według ich podobieństw, wzorców i różnic. Obraz zostanie podzielony na dwie części. Pierwsza może zawierać wszystkie „kształty" z wkrętami, a druga część może zawierać „kształty" ze śrubami. Uczenie nienadzorowane umożliwia modelowi samodzielną pracę w celu wykrycia wzorców i informacji, które wcześniej były niewykrywane. Tego typu uczenie zajmuje się głównie danymi bez etykiety. Ważnym etapem tego typu uczenia jest klastrowanie czyli organizowanie zestawów informacji w logiczne podzbiory – klastry, bez wcześniejszej wiedzy na temat przydziału grupowego poszczególnych danych [11]. Każdy klaster powstający w wyniku analizy definiuje zbiór obiektów wykazujących między sobą pewne podobieństwa i odróżniających się od elementów umieszczonych w pozostałych grupach.

Kolejną dziedziną uczenia nienadzorowanego jest redukcja wymiarowości (ang.*dimensionality reduction*) [11]. Redukcja wymiarowości odnosi się do technik zmniejszania liczby zmiennych wejściowych w danych uczących. Kiedy mamy do czynienia z danymi wysokowymiarowymi - każda obserwacja daje nam dużą liczbę wartości pomiarowych - często przydatne jest zmniejszenie wymiarowości poprzez rzutowanie danych do niższej wymiarowej podprzestrzeni, która oddaje „istotę" danych. Mniejsza liczba wymiarów wejściowych często oznacza odpowiednio mniej parametrów lub prostszą strukturę w modelu uczenia maszynowego, zwaną stopniami swobody. Model ze zbyt wieloma stopniami swobody prawdopodobnie przekroczy zestaw danych szkoleniowych i dlatego może nie działać dobrze na nowych danych. Dlatego ważne jest posiadanie prostych modeli, które dobrze uogólniają, a co za tym idzie, danych wejściowych z kilkoma zmiennymi wejściowymi. Redukcja wymiarowości to technika przygotowania danych wykonywana na danych przed modelowaniem [2].

Niekiedy redukowanie wymiarowości przydaje się również do wizualizacji danych np. zbiór cech wielowymiarowych można rzutować na jedno, dwu lub trójwymiarowe przestrzenie cech w celu wyświetlenia go w formie dwu lub trójwymiarowego wykresu punktowego albo histogramu [11].




Rysunek 2.16: Przykład redukcji wymiarowości [12]

### 2.4.5. Python - podstawowe narzędzie analizy dźwięku i budowy aplikacji

Python jest często nazywany językiem programowania ogólnego przeznaczenia, wysokiego poziomu. Filozofia języka Pythona kładzie nacisk na czytelność kodu oraz otwartość na dodatkowe rozszerzenia i biblioteki. Jego konstrukcje językowe, a także podejście obiektowe mają za zadanie ułatwić programistom w pisaniu przejrzystego, logicznego kodu dla wielu projektów zróżnicowanych przeznaczeniem oraz zaawansowaniem. Środowiska programistyczne dla Pythona są zazwyczaj w ramach licencji *open source*. Duża biblioteka standardowa Pythona, powszechnie wymieniana jako jedna z jego największych zalet, dostarcza narzędzi dostosowanych do wielu zadań. W przypadku aplikacji internetowych obsługiwanych jest wiele standardowych formatów i protokołów, takich jak MIME i HTTP. Zawiera moduły do tworzenia graficznych interfejsów użytkownika, gotowe rozwiązania do połączeń z bazami danych, generowania liczb pseudolosowych oraz wielu innych.
W tej pracy wykorzystane zostały biblioteki pozwalające tworzyć algorytmy uczenia maszynowego:

- **Scikit-learn** – darmowa biblioteka algorytmów z dziedziny uczenia maszynowego zbudowana na bazie modułu SciPy. Moduł scikit-learn udostępnia wiele algorytmów z dziedziny nadzorowanego i nienadzorowanego uczenia maszynowego w postaci interfejsu programistycznego.

- **Pandas** - to doskonała biblioteka do operacji na danych. Oprócz możliwości odczytu danych z praktycznie każdego źródła udostępnia zaawansowane narzędzia transformacji, filtrowania i sortowania danych, a także ich wizualizacji za pomocą biblioteki matplotlib.

- **Matplotlib** - jest podstawową biblioteką do generowania wszelkich wykresów w Python.
- **NumPy** – to biblioteka umożliwiająca wykonywanie operacji na macierzach w tym matematycznych i logicznych. Dodatkowo umożliwia operacje we/wy, transformatę Fouriera, podstawowe działania algebry liniowej, operacje statystyczne, symulacje losowe oraz inne. Na jej podstawie stworzono, między innymi bibliotekę Pandas.
- **Librosa** - to pakiet do analizy muzyki i dźwięku. Dostarcza elementów niezbędnych do tworzenia systemów wyszukiwania informacji muzycznych, analizy graficznej dźwięku, przetwarzania plików dźwiękowych. Może być wykorzystany do wyodrębnienia różnego rodzaju cech z segmentów audio, takich jak rytm, uderzenia i tempo.
- **Keras** - biblioteka, która zapewnia interfejs Pythona dla sztucznych sieci neuronowych. Keras działa jako interfejs dla biblioteki TensorFlow. Zawiera liczne implementacje powszechnie używanych bloków konstrukcyjnych sieci neuronowych, takich jak warstwy, cele, funkcje aktywacji, optymalizatory i szereg narzędzi ułatwiających pracę z obrazami i danymi tekstowymi, które upraszczają kodowanie niezbędne do pisania głębokiego kodu sieci neuronowej.

### 2.4.6. Pomiar pracy mini przekładni

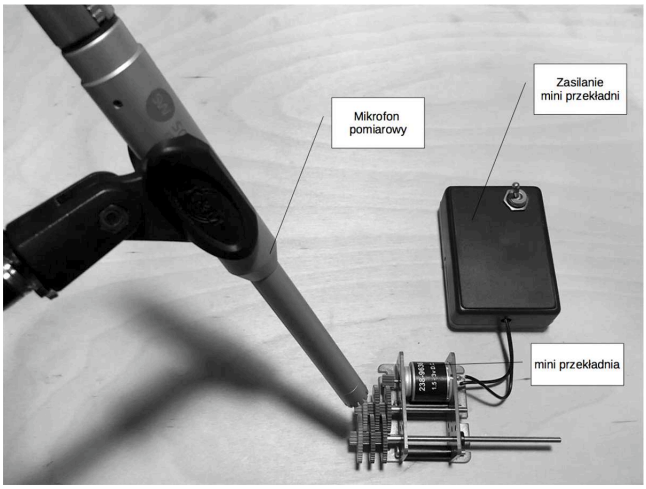
Jednym z głównych celów pracy była próba oszacowania etapu rozwoju technologii rozpoznawania sygnałów dźwiękowych i kwalifikacji ich w przestrzeni danych w różnym ujęciu - w systemach rozpoznawania mowy, w systemach identyfikacji plików muzycznych, systemach sterowanych dźwiękiem czy systemach pomiarowych i predykcyjnych. Opisane w pracy w bardzo ogólnym ujęciu zagadnienia wprowadzają jedynie w szalenie interesujący zakres tematyki analizy dźwiękowej, do której wpleciono coraz prężniej rozwijane uczenie maszynowe. Technologia uczenia maszynowego zyskuje niewiarygodną popularność i nabiera tempa rozwoju ze względu na bardzo szerokie zastosowanie, w którym rozpoznawanie i identyfikacja plików audio jest niewielkim wycinkiem całości.

Pomiary i zapis dźwięku wydobywającego się z mini przekładni został dokonany w warunkach izolacji akustycznej wolnej od hałasu i przypadkowych dźwięków, które mogłyby zakłócić rejestracje pracy przekładni.

Do analizy zostało wybranych kilka przypadków działania przekładni:

- poprawne działanie z dużą prędkością pracy (plik 1.wave),

- przekładnia nieznacznie uszkodzona (plik 2.wave),
- przekładnia zatrzymana w czasie pracy i ponownie uruchomiona z dużym uszkodzeniem (plik 3.wave),
- przekładnia z dużym stopniem uszkodzenia (plik 4.wave),
- poprawne działanie przekładni z prawidłową prędkością pracy (plik 5.wave),
- przekładnie nie uruchomiona (plik 6.wave),
- przekładnia w bardzo nierównomiernym trybie pracy (plik 7.wave).




Rysunek 2.17: Sposób rejestracji działania mini przekładni

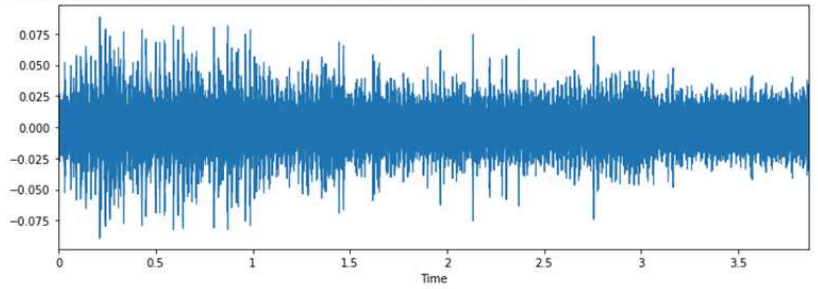
Próbki wave zostały zarejestrowane jako pliki mono z rozdzielczością 16 bitową i częstotliwością próbkowania 44,1 kHz. Wykorzystując bibliotekę Librosa możemy dokonać wizualizacji plików wave w formie wykresów.

Korzystając z narzędzi biblioteki Librosa, możemy również wyświetlić podgląd spektrogramu dowolnego pliku wave. Spektrogram to wizualny sposób przedstawienia siły sygnału lub „głośności" sygnału w czasie przy różnych częstotliwościach występujących w określonym przebiegu. Nie tylko można zobaczyć, czy jest więcej lub mniej energii w danym zakresie częstotliwości, ale można również analizować, jak poziomy energii zmieniają się w czasie [7]. Spektrogram jest zwykle przedstawiany jako tzw. mapa ciepła czyli obraz o intensywności pokazanej przez zmianę koloru lub jasności.

Rysunek 2.21 przedstawia spektrogram próbki dźwięku przekładni z dużym stopniem uszkodzenia.

```
data, sampling_rate = librosa.load('/content/drive/MyDrive/train/1.wav')
data
sampling_rate
plt.figure(figsize=(12,4))
librosa.display.waveplot(data, sr=sampling_rate)
```

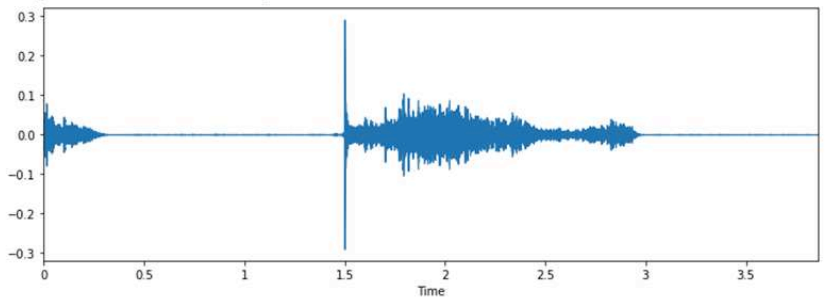
```
<matplotlib.collections.PolyCollection at 0x7fe3fc1bef50>
```




Rysunek 2.18: Wykres pliku 1.wave(poprawne działanie z dużą prędkością pracy)

```
data, sampling_rate = librosa.load('/content/drive/MyDrive/train/3.wav')
data
sampling_rate
plt.figure(figsize=(12,4))
librosa.display.waveplot(data, sr=sampling_rate)
```

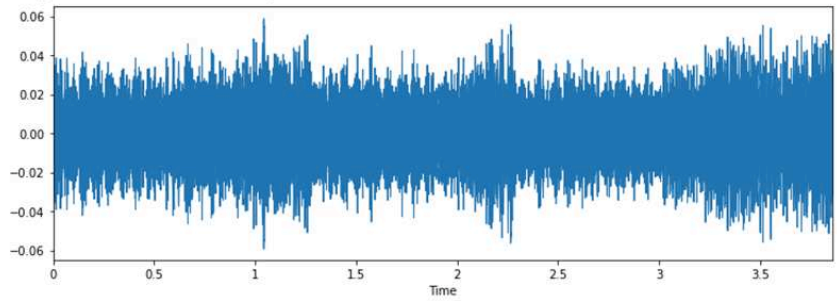
```
<matplotlib.collections.PolyCollection at 0x7fe3fc195d50>
```




Rysunek 2.19: Wykres pliku 3.wave(przekładnia zatrzymana w czasie pracy i ponownie uruchomiona z dużym uszkodzeniem)

```
data, sampling_rate = librosa.load('/content/drive/MyDrive/train/7.wav')
data
sampling_rate
plt.figure(figsize=(12,4))
librosa.display.waveplot(data, sr=sampling_rate)
```

```
<matplotlib.collections.PolyCollection at 0x7fe3fb2cab10>
```



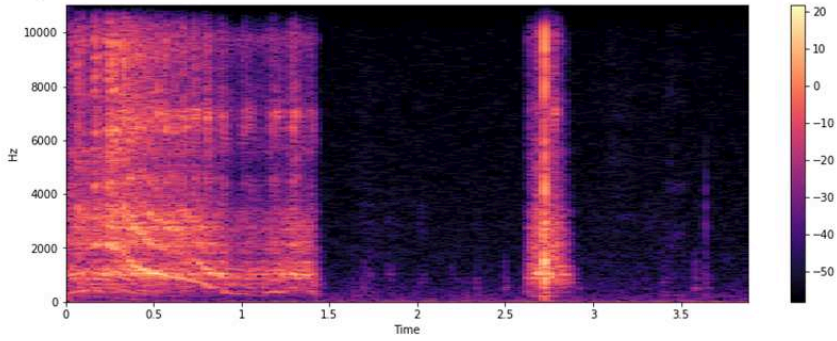

Rysunek 2.20: Wykres pliku 7.wave (przekładnia w bardzo nierównomiernym trybie pracy)

```
audio_data = '/content/drive/MyDrive/train/4.wav'
x , sr = lib.load('/content/drive/MyDrive/train/4.wav')
print(type(x), type(sr))
```

```
<class 'numpy.ndarray'> <class 'int'>
```

```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(14, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7fe3fae18950>
```




Rysunek 2.21: Spektrogram pliku 4.wave (przekładnia z dużym stopniem uszkodzenia)

Przedstawione przykłady pokazują, że korzystając z wbudowanych mechanizmów w biblioteki Python możemy dokonać bardzo precyzyjnej ana-

lizy próbek audio analizując wizualny aspekt dźwięku przez stosowanie odpowiednich wykresów. W pracy poruszono również aspekt kategoryzacji próbek dźwiękowych poprzez wyodrębnianie – ekstrakcję unikalnych i indywidualnych cech dla próbek, które w sposób jednoznaczny pozwalają je klasyfikować. Poniżej przedstawiono kilka podstawowych sposobów wyodrębniania tych cech.

**Ekstrakcja cech**

Każdy sygnał audio składa się z wielu funkcji. Należy jednak wyodrębnić cechy charakterystyczne dla problemu, który próbujemy rozwiązać. Przeanalizowano szczegółowo kilka funkcji.

**Przejście przez zero (ang. *zero crossing rate*)**

Crossing rate to szybkość zmian znaku wzdłuż sygnału, czyli szybkość, z jaką sygnał zmienia się z dodatniego na ujemny lub z powrotem. Ta funkcja jest często wykorzystywana zarówno do rozpoznawania mowy, jak i do wyszukiwania informacji o muzyce.

**Spectral centroid - widmowy środek ciężkości**

Spectral centroid jest stosowany w cyfrowym przetwarzaniu sygnałów w celu scharakteryzowania widma. Wskazuje, gdzie znajduje się środek masy widma. Percepcyjnie ma solidne połączenie z wrażeniem jasności dźwięku [13]. Ponieważ spectral centroid jest dobrym predyktorem „jasności" dźwięku, jest on szeroko stosowany w cyfrowym przetwarzaniu dźwięku i muzyki jako automatyczna miara barwy muzycznej [13].

**Spectral rolloff - częstotliwość opadania widma**

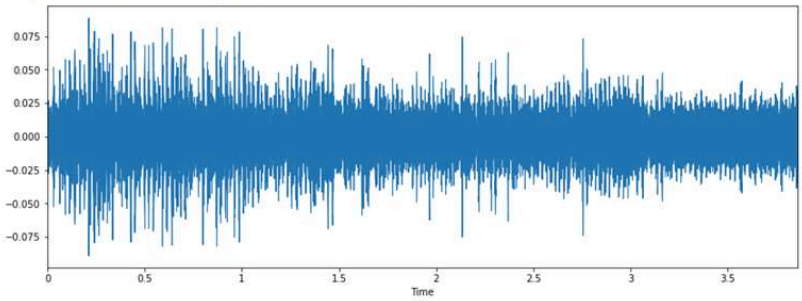
Częstotliwość opadania widma jest definiowana jako częstotliwość, poniżej której zawarty jest pewien procent (odcięcie) całkowitej energii widma. Częstotliwość rolloff może być użyta do rozróżnienia dźwięków harmonicznych (poniżej rolloff) i hałaśliwych (powyżej rolloff) [9].

**Mel-Frequency - współczynniki cepstralne**

Współczynniki cepstralne częstotliwości Mel (MFCC) sygnału to niewielki zestaw cech (zwykle około 10–20), które zwięźle opisują ogólny kształt obwiedni widmowej. Modeluje cechy ludzkiego głosu. Wartości MFCC nie są zbyt odporne na obecność dodatkowego hałasu, dlatego często normalizuje się ich wartości w systemach rozpoznawania mowy [22].

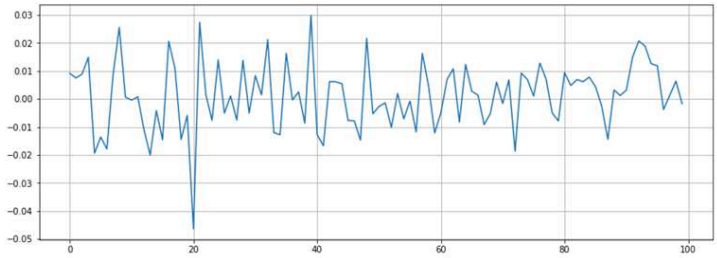
```
# ładowanie próbki dźwiękowej
x, sr = librosa.load('/content/drive/MyDrive/train/1.wav')
#wykres sygnału:
plt.figure(figsize=(14, 5))
librosa.display.waveplot(x, sr=sr)
```

```
<matplotlib.collections.PolyCollection at 0x7f5579b0e590>
```




Rysunek 2.22: Spektrogram pliku 1.wave (poprawne działanie przekładni z dużą prędkością pracy)

```
# powiększenie
n0 = 9000
n1 = 9100
plt.figure(figsize=(14, 5))
plt.plot(x[n0:n1])
plt.grid()
```




Rysunek 2.23: Powiększenie pliku 1.wave (poprawne działanie przekładni z dużą prędkością pracy)

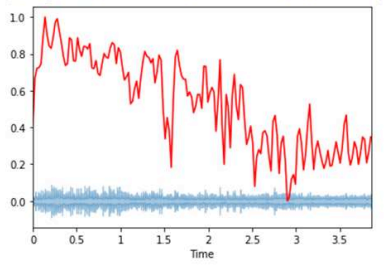
```
zero_crossings = librosa.zero_crossings(x[n0:n1], pad=False)
print(sum(zero_crossings))
```

```
47
```

Rysunek 2.24: Obliczenie współczynnika przejścia przez zero, które dla pliku 1.wave wynosi 47

```
import sklearn
spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
spectral_centroids.shape
#zmienna czasowa do wizualizacji
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)
# funkcja normalizowania środka ciężkości widma na potrzeby wizualizacji
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)
#Wykres Spectral Centroid wzdłuż przebiegu
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r')
```

```
[<matplotlib.lines.Line2D at 0x7f5579ac6dd0>]
```

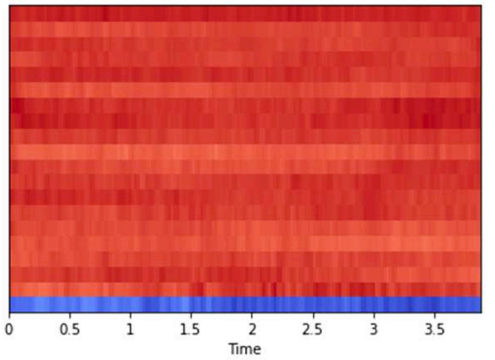



Rysunek 2.25: Spectral centroid dla pliku 1.wave

```
mfccs = librosa.feature.mfcc(x, sr=fs)
print (mfccs.shape)

(20, 167)
```

```
#wykres MFCCs:
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

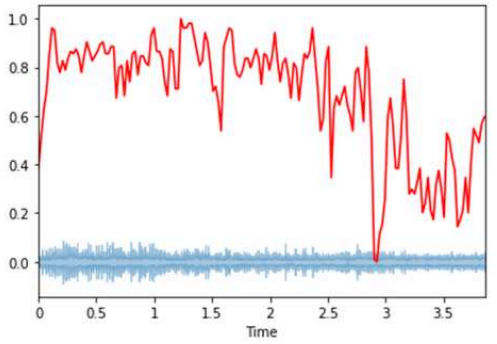
```
<matplotlib.collections.QuadMesh at 0x7f557b1e8e50>
```




Rysunek 2.26: MFCC dla pliku 1.wave to około 20 cech w 167 ramkach

```
spectral_rolloff = librosa.feature.spectral_rolloff(x+0.01, sr=sr)[0]
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_rolloff), color='r')
```

```
[<matplotlib.lines.Line2D at 0x7f557b4a1ed0>]
```




Rysunek 2.27: Spectral rolloff dla pliku 1.wave

## 2.5. Podsumowanie

W rozdziale przedstawiono i opisano główne zagadnienia związane wykorzystaniem algorytmów sztucznej inteligencji w darmowym środowisku programistycznym jakim jest Python. Opisane zostały również podstawy technologii rozpoznawania dźwięków i muzyki w popularnych rozwiązaniach stosowanych w aplikacjach mobilnych i webowych. Temat jest bardzo rozwojowy i pozwala na ogromne możliwości zastosowania uczenia maszynowego w rozwiązaniach przemysłowych i produkcyjnych. Wykorzystanie możliwości algorytmów sztucznej inteligencji w predykcji utrzymania ruchu pozwala nawet w otwartej - darmowej wersji na zbudowanie prawidłowo działającego systemu. Należy jednak pamiętać o odpowiednim przygotowaniu modelu, który pozwoli algorytmom poprawnie „uczyć się" i odpowiednio interpretować dane, co ma ogromne znaczenie na wynik końcowy.

# Bibliografia

[1] M.A. Boden. *Sztuczna Inteligencja. Jej Natura i Przyszłość.* Wydawnictwo Uniwersytetu Łódzkiego, Łódź, 2020.

[2] J. Brownlee. Introduction to dimensionality reduction for machine learning. *Machine Learning Mastery*, 5(1):14, 2020.

[3] J. Lal-Jadziak. *Przetwarzanie sygnałów. Wybrane zagadnienia.* Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, Toruń, 2020.

[4] T. Lester, D. Pachamanova. The dilemma of false positives: Making content id algorithms more conducive to fostering innovative fair use in music creation. *UCLA Entertainment Law Review*, 24(51):23, 2017.

[5] A. Li, C. Wang. An industrial-strength audio search algorithm. *DBLP*, 1(1):7, 2003.

[6] M. Miller. *The Internet of Things: How Smart TVs, Smart Cars, Smart Homes, and Smart Cities Are Changing the World*. Que publishing, New York, 2016.

[7] E. Ozimek. *Dźwięk i jego percepcja aspekty fizyczne i psychoakustyczne*. Wydawnictwo Naukowe PWN, Warszawa, 2018.

[8] C. Pacheco. You-tube-content-id-handbook, 2013.

[9] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. *CUIDADO I.S.T. Project Report*, 1(4):1–25, 2004.

[10] V. Petrock, J. Chung, R. Costa. *Voice-Enabled Technology StatPack:Current Forecasts for Digital Assistants and Speakers*. eMarketer Report, London, 2017.

[11] S. Raschka. *Python. Uczenie maszynowe*. Helion, Warszawa, 2017.

[12] P. Rosero, J.A.S. Castro, D. Peña, P. Diaz. Interactive data visualization using dimensionality reduction and similarity-based representations. *Lecture Notes in Computer Science*, 10(125):334–342, 2016.

[13] E. Schubert, J. Wolfe, A. Tarnopolsky. Spectral centroid and timbre in complex, multiple instrumental textures. *Proceedings of the 8th International Conference on Music Perception Cognition*, 8(9):654–657, 2004.

[14] K. Schwab. *The Fourth Industrial Revolution*. World Economic Forum, New Zealand, 2016.

[15] Oficjalna strona internetowa https://cloudfront.escholarship.org. Gpu computing, 2008.

[16] Oficjalna strona internetowa https://edu.pjwstk.edu.pl. Wykłady, 2015.

[17] Oficjalna strona internetowa https://popruntheworld.pl. Jak działa Content ID i Shazam?, 2018.

[18] Oficjalna strona internetowa https://www.synopsys.com. What-is-reinforcement-learning, 2020.

[19] J. Szabatin. *Podstawy teorii sygnałów*. Wydawnictwa Komunikacji i Łączności, Warszawa, 1982.

[20] M. Szeliga. *Praktyczne uczenie maszynowe*. Wydawnictwo Naukowe PWN, Warszawa, 2019.

[21] S. Szymaniec, M. Kacperak. *Utrzymanie ruchu w przemyśle. Informatyka i cyberbezpieczeństwo. Diagnostyka przemysłowa. Praktyka*. Wydawnictwo PWN, Warszawa, 2021.

[22] F. Zheng, G. Zhang, Z. Song. Comparison of different implementations of mfcc. *Journal of Computer Science and Technology*, 16(6):582–589, 2001.

**Artificial intelligence methods in predictive maintenance.**

**Abstract:** Artificial intelligence algorithms are of great use in many areas of life, and one of them is undoubtedly industry. The use of properly prepared programming and hardware environments allows you to precisely determine the lifetime of the machine in the factory or plan the service and maintenance of machines in detail. This article is an introduction to a more detailed study of machine learning implementation. The programming tool used at work is freely available and free, so everyone can try to use the technologies described in the article and try to create more complex systems. Systems that will allow you to analyze the processes taking place in the machine in real time and keep you informed about the state of the devices.

# 3. Prognozowanie ryzyka wypadków przy pracy w przedsiębiorstwie produkcyjnym

Maryna Bulakh[1]
Politechnika Rzeszowska, m.bulakh@prz.edu.pl

**Streszczenie** Niniejsza praca przedstawia model prognozowania wartości ryzyka wypadków przy pracy w przedsiębiorstwie produkcyjnym. W kolejnych rozdziałach ukazano zbiór wzorów, które pozwalają na obliczenie wartości ryzyk charakterystycznych dla przedsiębiorstwa produkcyjnego w ciągu okresów przeszłych, bieżących oraz przyszłych. Powstały w pracy model obejmuje podejścia prognozowania według trzech scenariuszy: optymistycznego, przybliżono-realnego oraz pesymistycznego. W celu uwzględnienia jak największej ilości wskaźników przeprowadzono prognozowanie metodą liniową oraz probabilistyczną. Pomimo ogólnych wskaźników bezpieczeństwa prognozowanie ryzyk apriorycznych dla przedsiębiorstwa produkcyjnego realizowano z uwzględnieniem zintegrowanych wskaźników. Na podstawie opracowanego modelu uzyskano wartości ryzyka, a następnie przedstawiono analizę ich zmiany oraz dynamikę. Wyniki obliczeń pozwalają na porównanie dużego obszaru wartości ewentualnych oraz uwzględnienie licznych sposobów zwiększenia poziomu bezpieczeństwa.

[1]ORCID: 0000-0003-4264-2303, Wydział Mechaniczno-Technologiczny Politechniki Rzeszowskiej, Kwiatkowskiego 4, 37-450 Stalowa Wola

## 3.1. Wprowadzenie

Współczesna literatura naukowa traktuje przedsiębiorstwo produkcyjne jako wyodrębniony organizacyjnie i ekonomicznie podmiot gospodarczy. Wśród głównych cech, którymi charakteryzuje się ta jednostka jest swoboda akceptacji decyzji ekonomicznych oraz odpowiedzialność za prawomocne zobowiązania. Biorąc pod uwagę dynamicznie zmieniające się warunki, praca nowoczesnego przedsiębiorstwa ciągle związana jest z rozwiązywaniem wzajemnie występujących problemów: szybkim przejściem do produkcji nowych wyrobów przy jednoczesnym wprowadzaniu najnowszych technologii i urządzeń; poprawą jakości produktu i obniżeniem kosztów produkcji, itp. Bardzo ważnym aspektem działalności jest również zapewnienie odpowiedniego poziomu bezpieczeństwa i utrzymanie go podczas wszystkich procesów, bez względu na zachodzące zmiany.

Większość zakładów produkcyjnych – niezależnie od rodzaju produkcji – cechuje obecnie wysoki stopień zautomatyzowania. Znacząco ułatwia to i przyspiesza proces produkcyjny, zwiększa jego wydajność, a także umożliwia większą standaryzację i ułatwia kontrolę jakości. Jednocześnie stwarza to jednak istotne zagrożenie dla osób pracujących w takim zakładzie. Awaria urządzenia, jego nienależyte zabezpieczenie lub obsługa, czy zwykła ludzka nieuwaga, mogą stać się przyczyną poważnego w skutkach wypadku. Praktyka pokazuje, że tego rodzaju zakłady mają na ogół dobrze opracowane procedury w zakresie bezpieczeństwa i higieny pracy, ale z ich przestrzeganiem w praktyce bywa różnie. Jednak nawet najlepiej opracowane i egzekwowane procedury mogą nie zapobiec zdarzeniom losowym czy błędom ludzkim, toteż wypadki na halach produkcyjnych zdarzały się, zdarzają i będą się zdarzać.

Dla każdego przedsiębiorstwa charakterystyczne jest zastosowanie odrębnego podejścia do oceny bezpieczeństwa. Jednak prawie na całym świecie panuje pogląd, że bezpieczeństwo można interpretować poprzez ryzyko, które może mieć charakter techniczny, zarządczy itd. [15].

Ryzyko określane jest najczęściej jako ciąg działań prowadzących do braku osiągnięcia oczekiwanych korzyści i powstania szkody lub straty [12]. Definicja ryzyka również została przedstawiona w pracach [11, 16, 17], których autorzy prezentują bardzo zbliżone poglądy. Według nich ryzyko definiowane jest jako niebezpieczeństwo czy też zagrożenie, które może uniemożliwiać osiągnięcie wyznaczonych celów przedsiębiorstwa.

Według A. Gaschi [8] ryzyko stanowi połączenie prawdopodobieństwa wystąpienia zagrożenia oraz jego negatywnego skutku oddziaływania. Analizując różne definicje, J. Stokłosa twierdzi, że z punktu widzenia niezawodności, ryzyko można określić jako prawdopodobieństwo wystąpienia strat materialnych. Ryzyko można przedstawić zatem jako kombinację prawdopodobieństwa wystąpienia niekorzystnego zdarzenia i jego następstw.

Jak stwierdza M. K. Gąsowska [9] ryzyko oznacza możliwość wystąpienia negatywnych lub pozytywnych konsekwencji przyszłych zdarzeń o znanym prawdopodobieństwie i sile wpływu.

W innych pracach naukowych ryzyko definiowane jest jako zdarzenie, które negatywnie wpływa na funkcjonowanie całego systemu i jego pożądane wskaźniki wydajności [7, 10].

Wielostronne poglądy na definicje ryzyka opisuje P. Sulewski. Autor [18] zwraca uwagę na to, że zarządzanie ryzykiem w logistyce jest procesem złożonym i wieloetapowym. Jego zdaniem wynika to m.in. ze specyfiki branży, przejawiającej się w możliwości kumulowania zagrożeń w kolejnych etapach łańcucha logistycznego i związanym z tym brakiem jednego właściciela ryzyka.

W pracy [5] został opisany proces zarządzania ryzykiem i jego podstawowe elementy, gdzie jednym z nich jest właśnie ocena ryzyka. Biorąc pod uwagę zasady i ogólne wytyczne dotyczące zarządzania ryzykiem opisane w normie ISO 31000:2018 (Zarządzanie ryzykiem) [2], proces ogólnej oceny ryzyka możemy przedstawić jako zbiór etapów:

– identyfikacji – proces wykrycia, zrozumienia i rejestracji ryzyka;

– analizy – proces określania konsekwencji, prawdopodobieństwa i poziomu zidentyfikowanych ryzyk;

– oceny – ustalenie wartości, poziomu i rodzaju ryzyka.

Listę metod oceny ryzyka oraz zalecenia dotyczące specyfiki ich doboru i zastosowania praktycznego zawiera norma ISO/IEC 31010 Risk management – Risk assessment techniques [1].

W odniesieniu do przedsiębiorstwa produkcyjnego najpopularniejszymi metodami, które pozwalają na uzyskiwanie wyjściowo wskaźników ilościowych, są:

– analizy drzewa awarii;

– analizy przyczyn i konsekwencji;

– analizy drzewa zdarzeń;

– analizy rodzajów i konsekwencji awarii;

– analizy poziomów ochrony (LOPA);

– ogólnej oceny niezawodności człowieka (HRA);

– analizy Markowa;

– modelowania metodą Monte Carlo;

– analizy bayesowskiej itp.

Analiza zalet i wad wyżej wymienionych metod została przedstawiona w Tablicy 3.1.

Tablica 3.1: Zalety i wady metod oceny ryzyka

| Lp | Metoda | Zaleta | Wada |
|---|---|---|---|
| 1. | Analiza rodzajów i konsekwencji awarii | - właściwe zrozumienie krytycznych procesów i zasobów;<br>- możliwość ponownej oceny procesów operacyjnych organizacji | - brak wiedzy osób uczestniczących w ankietach, wywiadach lub posiedzeniach roboczych;<br>- formułowanie uproszczonych lub nadmiernie optymistycznych oczekiwań |
| 2. | Analiza drzewa awarii | - możliwość analizy różnych czynników;<br>- skupienie się na konsekwencjach odmowy bezpośrednio związanych ze zdarzeniem niepożądanym;<br>- łatwe zrozumienie zachowania systemu poprzez analizę obrazu graficznego | - model statyczny, więc współzależności czasowe nie są brane pod uwagę;<br>- możliwość zastosowania tylko w odniesieniu do stanów binarnych (zawodność/niezawodność);<br>- brak możliwości oceny awarii spowodowanych błędem ludzkim |

| Lp | Metoda | Zaleta | Wada |
|---|---|---|---|
| 3. | Analiza przyczyn i skutków | - metoda ma podobne zalety, charakterystyczne dla metody „drzewa zdarzeń" i „drzewa awarii" | - zwiększona złożoność w porównaniu z metodami „drzewa zdarzeń" i „drzewa awarii" |
| 4. | Analiza drzewa zdarzeń | - możliwość jasnego, schematycznego przedstawienia analizowanych potencjalnych scenariuszy;<br>- umożliwia graficzne wyświetlanie sekwencji zdarzeń, których nie można wyświetlić za pomocą drzewa awarii | - potrzeba zastosowania innej metody w celu identyfikacji wszystkich potencjalnych zdarzeń;<br>- niektóre zależności mogą nie zostać dostrzeżone z powodu nieuwagi, co może prowadzić do optymistycznych ocen ryzyka |
| 5. | Analiza poziomów ochrony (LORA) | - wykorzystanie znacznie mniej czasu oraz zasobów niż analiza drzewa awarii;<br>- promowanie identyfikacji najbardziej krytycznych poziomów ochrony i koncentracji na nich zasobów;<br>- skupienie się na najpoważniejszych konsekwencjach | - skupienie się jednocześnie na jednej parze przyczynowo skutkowej i jednym scenariuszu;<br>- ryzyko ilościowe może nie uwzględniać ogólnych błędów;<br>- brak możliwości zastosowania do bardzo złożonych scenariuszy |
| 6. | Ogólna ocena niezawodności człowieka | - możliwość uwzględniania błędów ludzkich przy rozważaniu ryzyka;<br>- możliwość analizy rodzajów błędów ludzkich i ich wzorców | - złożoność osoby i różnorodność ludzi, co utrudnia określenie prostych typów awarii i ich prawdopodobieństw;<br>- brak możliwości jednoznacznego podziału (sukces/porażka) |
| 7. | Analiza Markowa | - możliwość obliczenia prawdopodobieństwa sprawnych systemów w kilku pogarszających się warunkach | - wymaga znajomości operacji na macierzach;<br>- komplikuje wymianę informacji o wynikach z personelem nietechnicznym |
| 8. | Modelowanie symulacyjne metodą Monte Carlo | - stosunkowo proste opracowanie modelu;<br>- możliwość rozbudowy modelu;<br>- wszystkich wpływów lub połączeń, które pojawiają się w rzeczywistości | - dokładność rozwiązań zależy od liczby symulacji;<br>- duże i złożone modele mogą przysporzyć wykonawcy trudności;<br>- metoda może niedostatecznie rozróżniać ważne konsekwencje od mało prawdopodobnych zdarzeń, a zatem nie pozwala na uwzględnienie w analizie skłonności organizacji do podejmowania ryzyka |
| 9. | Analiza bayesowska | - wyniki są łatwe do zrozumienia;<br>- metoda jest sposobem na zastosowanie subiektywnych wyobrażeń o problemie | - trudność określenia wszystkich interakcji w sieciach bayesowskich dla złożonych systemów;<br>- wymaga znajomości wielu prawdopodobieństw warunkowych, które zwykle określa się na podstawie ekspertyz |

Temat opracowania i wdrażania metod oceny ryzyka już od dłuższego czasu pozostaje aktualny i często spotykany w literaturze naukowej.

Według autora pracy [19] zarządzanie ryzykiem składa się z dwóch etapów: identyfikacji i analizy czynników wpływających na ryzyko oraz ilościowego pomiaru jego poziomu. W omawianej pracy w szczególności zostały opisane metody i modele oceny poziomu ryzyka: opisowa ocena ryzyka, analiza profilowa, systemy wczesnego ostrzegania, metody wyrównywania ryzyka, metoda szacowania ryzyka FMEA, katalogowanie czynników ryzyka, wstępna analiza zagrożeń PHA, ankiety eksperckie, spotkania planistyczne, przegląd dokumentacji, porównanie analogii, ocena planu, technika delficka, burza mózgów oraz analiza SWOT.

Przez autora pracy [14] zostały scharakteryzowane poszczególne rodzaje ryzyka z obszaru zarządzania zasobami ludzkimi, a także zostały przedstawione etapy analizy ryzyka tego obszaru.

Z kolei w pracy [20] został zaproponowany model zarządzania ryzykiem w przedsiębiorstwie produkcyjnym oparty na teorii ograniczeń, który pozwala na podjęcie decyzji o wyborze najważniejszych z punktu widzenia wyniku finansowego przedsiębiorstwa produktów w sytuacji, gdy istnieje ryzyko niewykonania w całości planu produkcyjnego.

Istotną rolę w procesie zarządzania ryzykiem odgrywa prognoza wartości ryzyka, ponieważ to pozwala przedsiębiorstwu na podjęcie decyzji umotywowanych.

## 3.2. Metody określania i prognozowania wartości ryzyka

Podstawą skutecznej decyzji zarządczej dla przedsiębiorstwa produkcyjnego mogą być wyniki analizy i uwzględnienie różnorodnych wartości ryzyk oraz ewentualna dynamika tych wskaźników. Niezbędne w tym procesie jest posiadanie odpowiednich wskaźników, które można opisać następującym wektorem:

$$R = (R_{\downarrow}(X), R_0(X), R_{\uparrow}(X)) \tag{3.1}$$

gdzie $R_{\downarrow}(X), R_0(X), R_{\uparrow}(X)$ to odpowiednio wartości ryzyka aposteriorycznego (przeszłego), bieżącego (teraźniejszego) i apriorycznego (przyszłego).

Wartości ryzyka aposteriorycznego $R_{\downarrow}$ można określić na podstawie danych statystycznych o liczbie naruszeń bezpieczeństwa w badanym przedsiębiorstwie, korzystając z teorii prawdopodobieństwa oraz istniejących metod oceny ryzyka [3, 6].

Celem sporządzenia prognozy opartej na proponowanym modelu zastosowana zostanie liczba naruszeń bezpieczeństwa, do której należą: ogólna ilość naruszeń, wypadki ze śmiertelnymi ofiarami, wypadki na skutek, których osoba została poszkodowana oraz groźne wypadki, które doprowadziły do poniesienia dużych strat, ale nie spowodowały powstania ofiar. Dane statystyczne pozyskano z jednego z polskich przedsiębiorstw produkcyjnych za okres 01.01.2016 r. – 31.03.2021 r.

Obliczanie wartości ryzyka dokonywane jest w interwałach czasowych wynoszących jeden rok. Ze względu na potrzebę porównywania oraz analizy wartości ryzyk obliczonych w tych interwałach musimy sprowadzić je do tych samych parametrów. Biorąc pod uwagę, że ostatnie dane uzyskane przez przedsiębiorstwo dotyczą okresu trzech miesięcy (01.01.2021 r. – 31.03.2021 r.), obliczone wskaźniki przyjmujemy jako wartość ryzyka na dzień przeprowadzenia oceny $R_{DO}$. W takim wypadku ryzyko bieżące $R_0$ będzie wartością częściowo prognozowaną. Ponieważ przy podejmowaniu skutecznych decyzji dobrze jest opierać się na kilku różnych możliwych wariantach, obliczenia wartości ryzyka bieżącego $R_0$ oraz ryzyk apriorycznych $R_{\uparrow}$ na przyszłe okresy będą realizowane w trzech scenariuszach.

Wykorzystując wartość ryzyka w dniu przeprowadzania oceny, wyznaczamy wartość ryzyka bieżącego $R_0$ według wzoru:

$$R_0 = \bar{R} + R_{DO} \tag{3.2}$$

gdzie $\bar{R}$ to prognozowana część ryzyka bieżącego, która jest obliczana dla każdego scenariusza osobno według wzorów 3.3, 3.4, 3.5.

Aby określić wartość ryzyka bieżącego w scenariuszu optymistycznym, obliczamy $\bar{R}_{opt}$:

$$\bar{R}_{opt} = \frac{\prod_{i=1}^{n} R_i}{\sum_{i=1}^{n} R_i} \tag{3.3}$$

Wartość tego wskaźnika dąży do minimum ponieważ chcemy uwzględnić najmniejsze ryzyko, które możemy przewidywać biorąc pod uwagę wartości ryzyk aposteriorycznych.

Otrzymaną wartość $\bar{R}_{opt}$ podstawiamy do wzoru 3.2, obliczając w ten sposób wartość ryzyka bieżącego według scenariusza optymistycznego ($R_0^{opt}$).

W celu uzyskania możliwości określenia ryzyka bieżącego w scenariuszu przybliżono-realnym obliczamy wartość $\bar{R}_{pr}$, a następnie $R_0$ według wzoru 3.2.

$$\bar{R}_{pr} = \frac{\sum_{i=1}^{n} \bar{R}_i}{n} \tag{3.4}$$

gdzie $\bar{R}_i$ to średnia wartość ryzyka.

W tym scenariuszu $\bar{R}_{pr}$ dąży do $\bar{R}$ obliczonego wobec ryzyk aposteriorycznych.

Prognozowanie wartości ryzyka bieżącego w scenariuszu pesymistycznym odbywa się z uwzględnieniem następującego wyrażenia:

$$\bar{R}_{ps} = \max R_{\downarrow n} \tag{3.5}$$

gdzie $\max R_{\downarrow n}$ to wartość ryzyka tego roku, w którym osiągnął największe znaczenie.

Stąd $R_0^{ps} \longrightarrow \max$, bo uwzględniamy największą wartość ryzyka, która może wystąpić w działalności przedsiębiorstwa.

W celu wyznaczenia wartości ryzyk apriorycznych $R_{\uparrow}$ możemy stosować wartości odchylenia charakterystycznego dla końcowego odcinka (roku) w porównaniu z początkowym odcinkiem badanego okresu, czyli roczny wzrost ryzyka:

$$\triangle_{kpi} = |R_{pi} - R_{ki}| \tag{3.6}$$

gdzie $R_{pi}$, $R_{ki}$ – wartości ryzyka końcowego i początkowego odcinków badanego okresu; k – początkowy odcinek badanego okresu; p – końcowy odcinek badanego okresu.

Korzystając z obliczonych wartości ryzyk bieżących, wykonujemy prognozowanie metodą liniową według scenariusza optymistycznego:

$$R_{\uparrow_{opt}} = R_0^{opt} + \triangle_R \tag{3.7}$$

Z kolei wskaźnik odchylenia obliczamy ze wzoru:

$$R_{opt} = \frac{\prod_{i=1}^{n} \triangle_i}{\sum_{i=1}^{n} \triangle_i} \longrightarrow \min \tag{3.8}$$

Dla uzyskania wartości ryzyka apriorycznego według scenariusza przybliżono-realnego stosujemy wzór:

$$R_{\uparrow_{pr}} = R_0^{pr} + \bar{\triangle} \tag{3.9}$$

gdzie:

$$\bar{\triangle} = \frac{\sum_{i=1}^{n} \triangle_i}{n} \tag{3.10}$$

gdzie $\triangle_i$ – wartość rocznego wzrostu ryzyka; $n$ – liczba dni.

W scenariuszu pesymistycznym wyrażenie do prognozy metodą liniową wygląda następująco:

$$R_{\uparrow_{ps}} = R_0^{ps} + \triangle_{\max_i} \tag{3.11}$$

gdzie $\triangle_{\max_i}$ – wzrost ryzyka tego roku, w którym osiągnął największą wartość.

W celu szerszego uwzględnienia możliwych opcji rozwoju zdarzeń i ich analizy wskazane jest, obok metody liniowej, prowadzenie prognozowania probabilistycznego według tych samych scenariuszy, przy użyciu formuł:

$$R_{pb}^{opt} = \frac{R_0^{opt} \triangle_R}{R_0^{opt} + \triangle_R} \tag{3.12}$$

$$R_{pb}^{pr} = \frac{R_0^{pr} \bar{\triangle}}{R_0^{pr} + \bar{\triangle}} \tag{3.13}$$

$$R_{pb}^{ps} = \frac{R_0^{ps} \triangle_{\max_i}/K}{R_0^{ps} + \triangle_{\max_i}/K} \tag{3.14}$$

gdzie $K$ – liczba dni w roku.

Następnie obliczamy podobne wskaźniki korzystając z danych uzyskanych po przeprowadzeniu zintegrowanej oceny stanu bezpieczeństwa przedsiębiorstwa, jak to zostało opisane w pracach [4, 13].

## 3.3. Analiza wartości ryzyk

Obliczenia wartości ryzyk aposteriorycznych i na dzień oceny wykonano na podstawie danych przedstawionych w Tablicy 3.2, zgodnie z metodą przedstawioną w artykule [6] według wzoru:

$$R_i = \frac{\sum W_S + W_P + W_{CS}}{\sum W_i} \tag{3.15}$$

gdzie $W_S$ - liczba wypadków z ofiarami śmiertelnymi; $W_P$ - liczba wypadków na skutek których osoba została poszkodowana; $W_{CS}$ - liczba wypadków z ciężkimi skutkami.

Tablica 3.2: Liczba wypadków naruszenia zasad bezpieczeństwa w przedsiębiorstwie produkcyjnym

| ***Rok*** | $W_S$ | $W_P$ | $W_{CS}$ | Liczba wypadków |
|---|---|---|---|---|
| 2016 | 0 | 3 | 0 | 602 |
| 2017 | 0 | 2 | 1 | 550 |
| 2018 | 1 | 9 | 4 | 541 |
| 2019 | 0 | 3 | 3 | 481 |
| 2020 | 0 | 2 | 3 | 477 |
| 2021 | 0 | 1 | 1 | 207 |

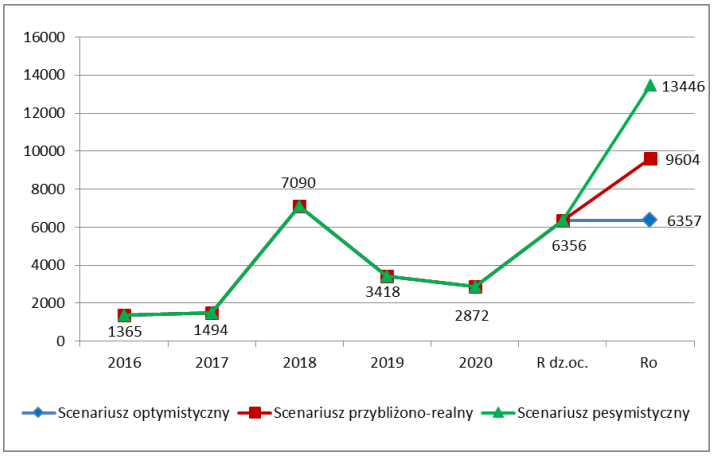
Wartości ryzyk aposteriorycznych, ryzyka na dzień oceny oraz ryzyka bieżącego przedstawiono w Tablicy 3.3, natomiast ich dynamikę zaprezentowano na Rysunku 3.1.

Tablica 3.3: Wyniki obliczeń wartości ryzyka bieżącego $\times 10^{-8}$

| **Rok** | $R_{ir}$ | $R_{id}$ | $\triangle_{kp}$ | $\triangle_R$ |
|---|---|---|---|---|
| 2016 | 498339 | 1365 | 0 | 1 |
| 2017 | 545455 | 1494 | 47116 | 1 |
| 2018 | 2587800 | 7090 | 2042346 | 1 |
| 2019 | 1247401 | 3418 | 1340399 | 1 |
| 2020 | 1048218 | 2872 | 199183 | 1 |
| **Rok** | $\bar{\triangle}$ | $R_0^{opt}$ | $R_0^{pr}$ | $R_0^{ps}$ |
| 2021 | 2485 | 6357 | 9604 | 13446 |

Poniższy wykres wskazuje na to, że wartość ryzyka bieżącego obliczona w scenariuszu optymistycznym, utrzymuje się na poziomie ryzyka w dniu oceny. Jednocześnie podobne ryzyko, liczone według przybliżono-realnego scenariusza, wzrasta o 51% w stosunku do ryzyka w dniu oceny oraz o 35% w stosunku do ryzyka w 2018 roku. Wartość ryzyka uzyskana w scenariuszu pesymistycznym jest o dwa razy większa od maksymalnej wartości aposteriorycznych ryzyk.

Jeśli chodzi o szerokość rozgałęzienia, które tworzą wartości ryzyk to




Rysunek 3.1: Wartość ryzyka bieżącego $\times 10^{-8}$

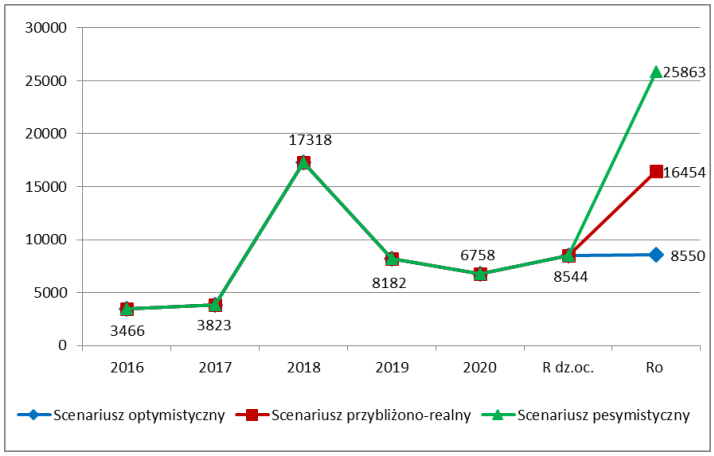
zależy ona od stabilności zachowywania ryzyk aposteriorycznych. Im łagodniejsza jest ich dynamika, tym węższe rozgałęzienie otrzymamy.

Wyniki obliczeń wartości ryzyka bieżącego z uwzględnieniem oceny zintegrowanej przedstawiono w Tablicy 3.4 oraz graficznie na Rysunku 3.2. Ten sam rysunek przedstawia kilka scenariuszy wizualizacji i porównania wyników.

Tablica 3.4: Wyniki obliczeń ryzyka bieżącego z uwzględnieniem oceny zintegrowanej $\times 10^{-8}$

| **Rok** | $R_{ir}$ | $R_{id}$ | $\triangle_{kp}$ | $\triangle_R$ |
|---|---|---|---|---|
| 2016 | 1265119 | 3466 | 0 | 13 |
| 2017 | 1395555 | 3823 | 130436 | 13 |
| 2018 | 6321249 | 17318 | 4925694 | 13 |
| 2019 | 2986469 | 8182 | 3334780 | 13 |
| 2020 | 12466597 | 6758 | 519872 | 13 |
| **Rok** | $\bar{\triangle}$ | $R_0^{opt}$ | $R_0^{pr}$ | $R_0^{ps}$ |
| 2021 | 6103 | 8550 | 16454 | 25863 |

Aby uzyskać bardziej wiarygodne informacje o wartości ryzyka apriorycznego, prognozowanie przeprowadza się dwiema metodami: liniową i pro-



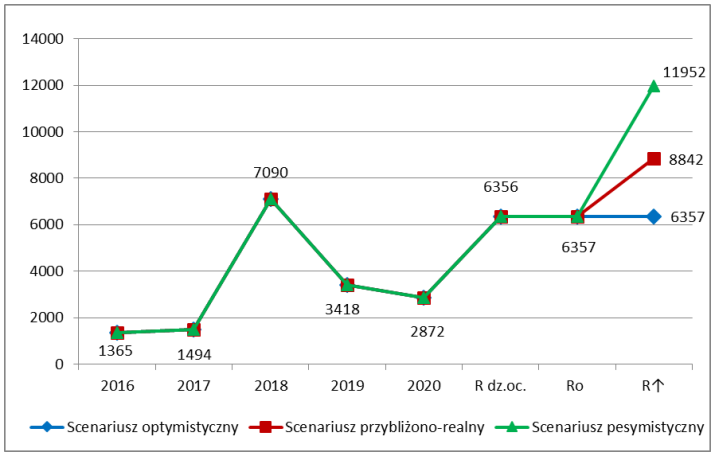

Rysunek 3.2: Wartość ryzyka bieżącego z uwzględnieniem danych oceny zintegrowanej $\times 10^{-8}$

babilistyczną. Jednocześnie wartości ryzyk bieżących przyjmowane są jako dane początkowe. Wyniki obliczeń podano w Tablicy 3.5.

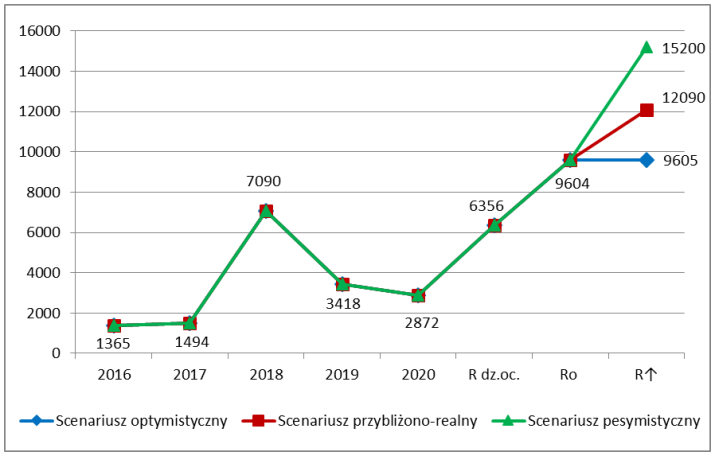
Tablica 3.5: Wyniki obliczeń ryzyka apriorycznego metodą liniową $\times 10^{-8}$

| Wartość ryzyka bieżącego | | Wartość ryzyka apriorycznego | |
|---|---|---|---|
| wg scenariusza | wartość | wg scenariusza | wartość |
| optymistycznego | 6357 | optymistycznego<br>przybliżono-realnego<br>pesymistycznego | 6357<br>8842<br>11952 |
| przybliżono-realnego | 9604 | optymistycznego<br>przybliżono-realnego<br>pesymistycznego | 9605<br>12090<br>15200 |
| pesymistycznego | 13446 | optymistycznego<br>przybliżono-realnego<br>pesymistycznego | 13447<br>15932<br>19042 |

Aby uzyskać możliwość porównania uzyskanych wyników konstruuje się wykresy (Rys. 3.3, Rys. 3.4, Rys. 3.5).




Rysunek 3.3: Wartość ryzyka apriorycznego obliczonego od wartości ryzyka bieżącego uzyskanej w scenariuszu optymistycznym $\times 10^{-8}$

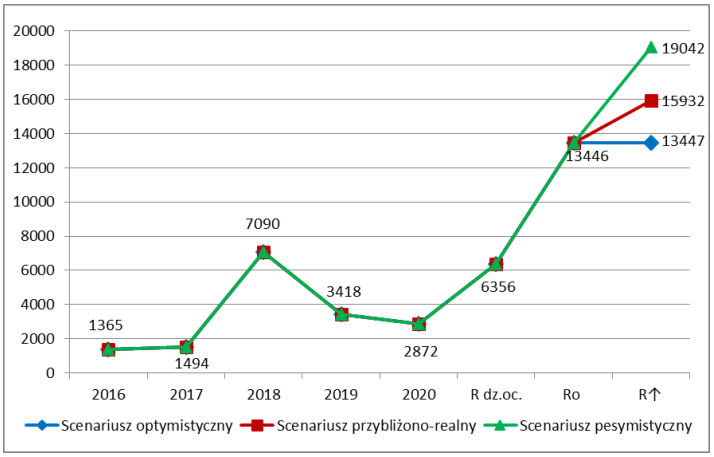



Rysunek 3.4: Wartość ryzyka apriorycznego obliczonego od wartości ryzyka bieżącego uzyskanej w scenariuszu przybliżono-realnym $\times 10^{-8}$

Na Rysunku 3.3 widzimy, że wartości ryzyka apriorycznego mają stałą tendencję wzrostową. Ryzyko uzyskane w scenariuszu optymistycznym utrzymuje się prawie na poziomie ryzyka w dniu oceny (wzrost poniżej 1%). Jednocześnie ryzyko aprioryczne uzyskane w przybliżono-realnym scenariuszu wzrasta o 39% w stosunku do wskaźnika w dniu oceny oraz o 24,7% w stosunku do wskaźnika w roku, w którym ryzyko osiągnęło maksymalną wartość. Ryzyko uzyskane w scenariuszu pesymistycznym jest prawie dwu-

krotnie wyższe niż w dniu oceny (wzrost o 88%) i rośnie o 68,6% w porównaniu ze wskaźnikiem ryzyka w 2018 r.

Ponieważ wartość ryzyka bieżącego obliczona w przybliżono-realnym scenariuszu rośnie w stosunku do wartości aposteriorycznych, ryzyko uzyskane w wyniku prognozowania liniowego ma podobny trend. Tym samym wartość ryzyka apriorycznego, uzyskana w scenariuszu optymistycznym, wzrasta o mniej niż 1% w stosunku do ryzyka bieżącego. Uzyskane w przybliżono-realnym scenariuszu ryzyka aprioryczne charakteryzują się wzrostem w stosunku do ryzyka bieżącego o 25,9%. Jednocześnie najbardziej rośnie ryzyko prognozowane w scenariuszu pesymistycznym, przekraczając o 58,3% ryzyko bieżące prawie 2,5-krotnie ryzyko w dniu oceny.




Rysunek 3.5: Wartość ryzyka apriorycznego obliczonego od wartości ryzyka bieżącego uzyskanej w scenariuszu pesymistycznym $\times 10^{-8}$

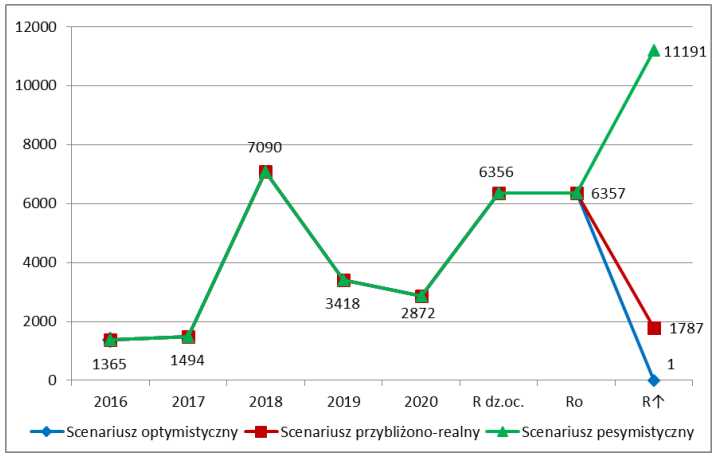
Na Rysunku 3.5 widzimy, że wartość prognozy ryzyka uzyskana w scenariuszu optymistycznym charakteryzuje się podobnym trendem do wcześniej rozważanych opcji – wzrost poniżej 1%. Jednocześnie ryzyko uzyskane w scenariuszu przybliżono-realnym wzrasta o 18,5% w porównaniu z ryzykiem bieżącym. Ryzyko aprioryczne liczone w scenariuszu pesymistycznym jest o 41,6% wyższe od ryzyka bieżącego i wzrasta prawie 3-krotnie w stosunku do ryzyka w dniu oceny.

Wyniki obliczeń ryzyka apriorycznego metodą probabilistyczną zostały przedstawione w Tablicy 3.6.

Tablica 3.6: Wyniki obliczeń ryzyka apriorycznego metodą probabilistyczną $\times 10^{-8}$

| Wartość ryzyka bieżącego | | Wartość ryzyka apriorycznego | |
|---|---|---|---|
| wg scenariusza | wartość | wg scenariusza | wartość |
| optymistycznego | 6357 | optymistycznego<br>przybliżono-realnego<br>pesymistycznego | 1<br>1787<br>11191 |
| przybliżono-realnego | 9604 | optymistycznego<br>przybliżono-realnego<br>pesymistycznego | 1<br>1975<br>11191 |
| pesymistycznego | 13446 | optymistycznego<br>przybliżono-realnego<br>pesymistycznego | 1<br>2098<br>11191 |

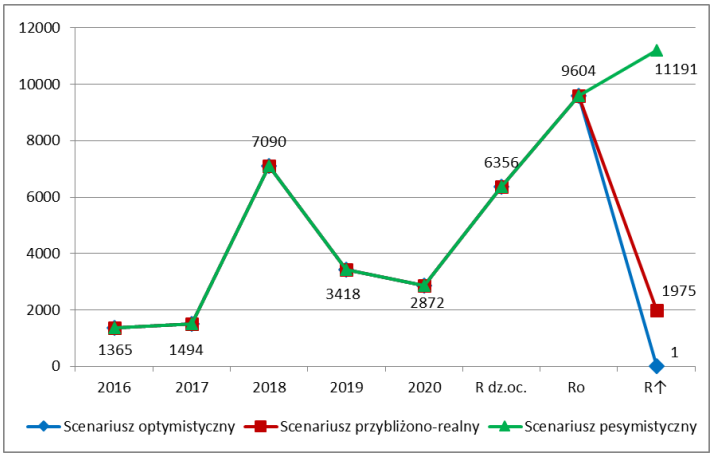
Aby uzyskać możliwość porównania uzyskanych wyników konstruuje się wykresy (Rys. 3.6, Rys. 3.7, Rys. 3.8).



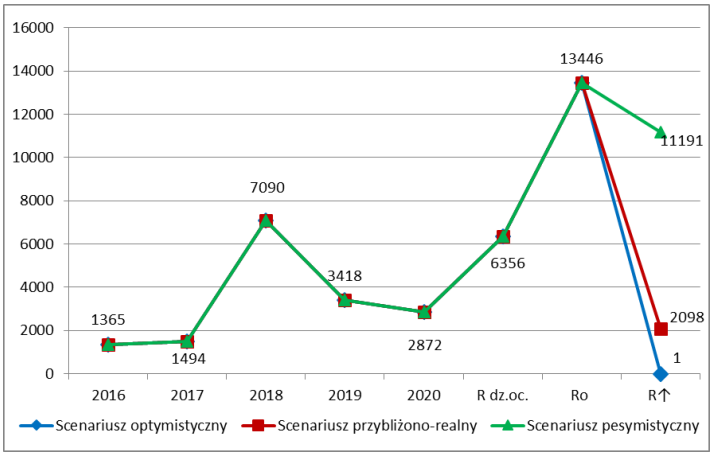

Rysunek 3.6: Wartość ryzyka apriorycznego obliczonego od wartości ryzyka bieżącego uzyskanej w scenariuszu optymistycznym metodą probabilistyczną $\times 10^{-8}$

Dane pokazane na Rysunku 3.6 charakteryzują się innym trendem niż uzyskane metodą liniową. Scenariusz optymistyczny redukuje ryzyko apriorycznе do najniższej możliwej wartości, przybliżono-realny - utrzymuje się

na poziomie 2016-2017, który charakteryzuje się minimalną wartością ryzyka, a dopiero scenariusz pesymistyczny wskazuje na szybki wzrost.




Rysunek 3.7: Wartość ryzyka apriorycznego obliczonego od wartości ryzyka bieżącego uzyskanej w scenariuszu przybliżono-realnym metodą probabilistyczną $\times 10^{-8}$




Rysunek 3.8: Wartość ryzyka apriorycznego obliczonego od wartości ryzyka bieżącego uzyskanej w scenariuszu pesymistycznym metodą probabilistyczną $\times 10^{-8}$

Wartości ryzyka obliczone przy użyciu prognozowania probabilistycznego nie mają jednakowego trendu. Podobnie jak w przypadku poprzedniej prognozy, wyniki obliczeń w scenariuszu optymistycznym wskazują na

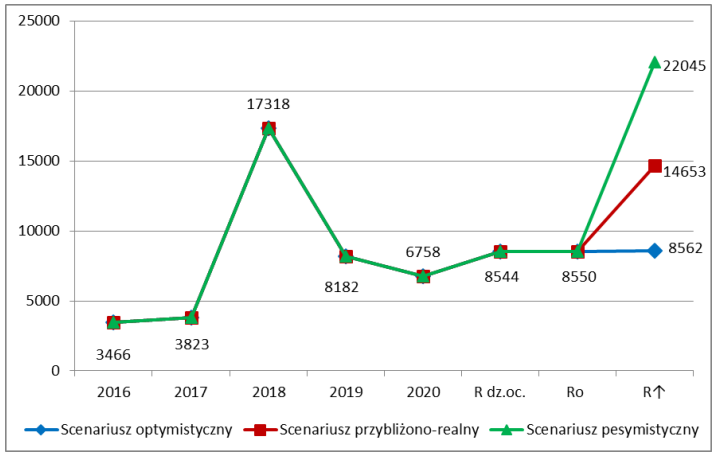
redukcję ryzyka do możliwie najniższej wartości. Ryzyko aprioryczne uzyskane w scenariuszu przybliżono-realnym jest zmniejszone prawie 5-krotnie w stosunku do ryzyka bieżącego, ale pozostaje w przedziale ryzyk aposteriorycznych. Jednocześnie prognozowane ryzyko liczone według scenariusza pesymistycznego nieznacznie rośnie i stanowi 15,6% wskaźnika ryzyka bieżącego.

Z danych przedstawionych na Rysunku 3.8, widzimy, że trend wartości ryzyka apriorycznego obliczony w scenariuszu pesymistycznym, różni się radykalnie od poprzednich wersji. Wszystkie prognozowane ryzyka charakteryzują się spadkiem w stosunku do ryzyka bieżącego: ryzyko obliczone w scenariuszu optymistycznym przyjmuje najniższą możliwą wartość, ryzyko obliczone w scenariuszu przybliżono-realnym spada o 84,4%, natomiast liczone wg scenariusza pesymistycznego – o 16,8%.

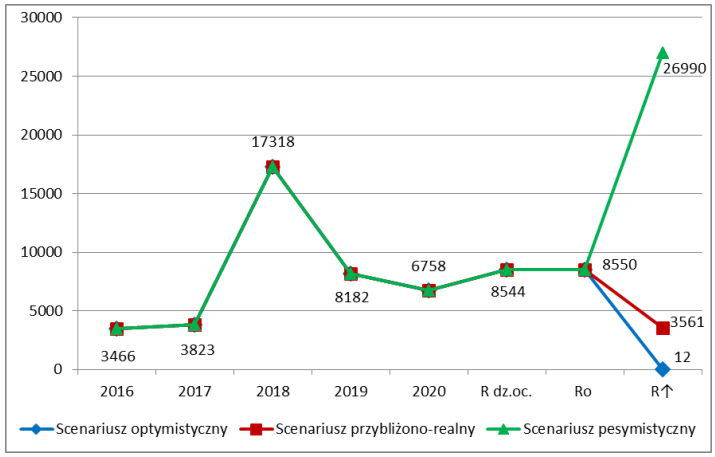
Obliczenia ryzyk apriorycznych z uwzględnieniem oceny zintegrowanej przeprowadzono metodą prognozowania liniowego (ML) i probabilistycznego (MP). Jednocześnie jako dane początkowe przyjmuje się wartości ryzyk bieżących uwzględniających ocenę zintegrowaną. Wyniki obliczeń przedstawiono w Tablicy 3.7, a ich prezentację graficzną na Rysunkach 3.9, 3.10.

Tablica 3.7: Wyniki obliczeń ryzyka apriorycznego z uwzględnieniem oceny zintegrowanej $\times 10^{-8}$

| Wartość ryzyka bieżącego | | Wartość ryzyka apriorycznego | | |
|---|---|---|---|---|
| wg scenariusza | wartość | wg scenariusza | wartość (ML) | wartość (MP) |
| optymistycznego | 8550 | optymistycznego | 8562 | 12 |
| | | przybliżono-realnego | 14653 | 3561 |
| | | pesymistycznego | 22045 | 26990 |
| przybliżono-realnego | 16454 | optymistycznego | 16466 | 12 |
| | | przybliżono-realnego | 22557 | 4452 |
| | | pesymistycznego | 29949 | 26990 |
| pesymistycznego | 25863 | optymistycznego | 25875 | 12 |
| | | przybliżono-realnego | 31966 | 4938 |
| | | pesymistycznego | 39358 | 26990 |




Rysunek 3.9: Ryzyko aprioryczne obliczone z uwzględnieniem oceny zintegrowanej metodą liniową od wartości ryzyka bieżącego uzyskanej w scenariuszu optymistycznym $\times 10^{-8}$




Rysunek 3.10: Ryzyko aprioryczne obliczone z uwzględnieniem oceny zintegrowanej metodą probabilistyczną od wartości ryzyka bieżącego uzyskanej w scenariuszu optymistycznym $\times 10^{-8}$

Dane pokazane na Rysunku 3.9, wskazują na wzrost ryzyka apriorycznego w stosunku do ryzyka bieżącego:

– o mniej niż 1% w scenariuszu optymistycznym,

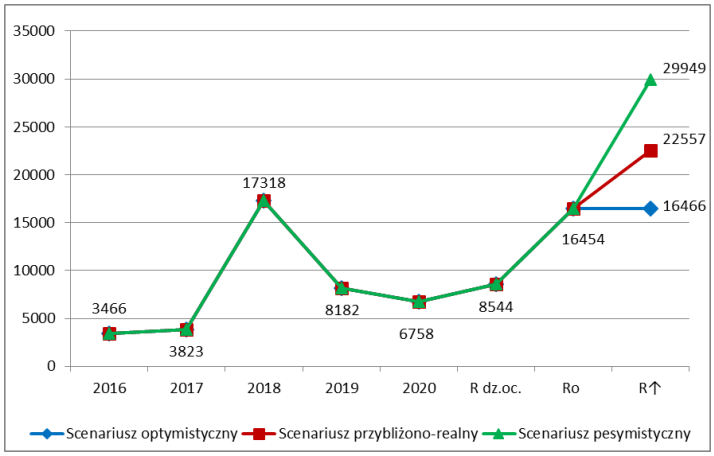
– o 71,5% w scenariuszu przybliżono-realnym,

– prawie 2,5-krotnie w scenariuszu pesymistycznym.

Wartości ryzyka obliczone przy użyciu prognozowania probabilistycznego nie wykazują tego samego trendu. Wyniki analizy w scenariuszu optymistycznym wskazują na redukcję ryzyka do najniższej możliwej wartości, w scenariuszu przybliżono-realnym pokazują spadek o 58,3%. Należy zauważyć, że pomimo znacznego spadku wskaźnik nie osiągnął minimalnej wartości typowej dla poprzedniego okresu.

Na tle tej tendencji spadkowej, typowej dla dwóch poprzednich prognoz, ryzyko obliczone w scenariuszu pesymistycznym znacznie wzrasta, ponad trzykrotnie przekraczając wskaźnik ryzyka bieżącego okresu.

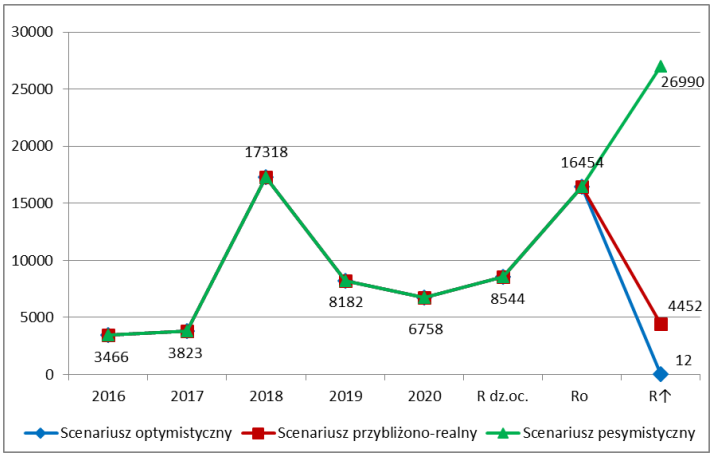
Wyniki obliczeń wartości ryzyka apriorycznego z uwzględnieniem oceny zintegrowanej, bazujących na wartości ryzyka bieżącego uzyskanego w scenariuszu przybliżono-realnym, przedstawiono na Rysunkach 3.11, 3.12.




Rysunek 3.11: Ryzyko aprioryczne obliczone z uwzględnieniem oceny zintegrowanej metodą liniową od wartości ryzyka bieżącego uzyskanej w scenariuszu przybliżono-realnym $\times 10^{-8}$

Na Rysunku 3.11 widzimy, że wartości prognozy ryzyka mają stałą tendencję wzrostową. Ryzyko uzyskane w scenariuszu optymistycznym pozostaje prawie na poziomie ryzyka bieżącego okresu (wzrost o niecały 1%). Jednocześnie ryzyko uzyskane w scenariuszu przybliżono-realnym wzrasta o 37% w stosunku do tego samego wskaźnika oraz o 30,3% w stosunku do roku, w którym ryzyko osiągnęło maksymalną wartość. Ryzyko uzyskane

w scenariuszu pesymistycznym jest prawie dwukrotnie większe niż w bieżącym okresie (wzrost o 82%).

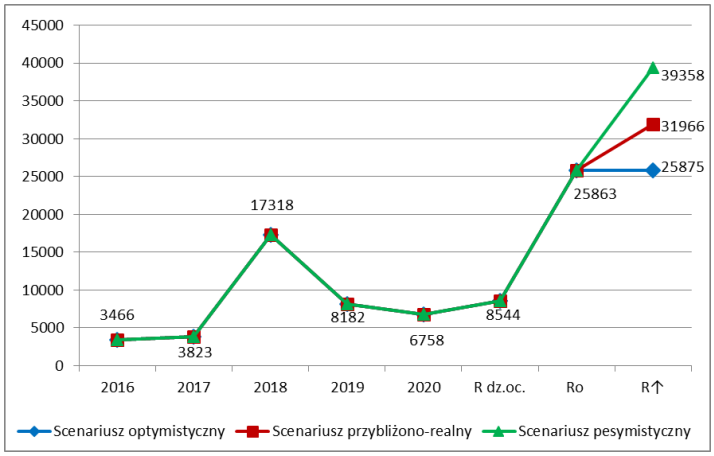



Rysunek 3.12: Ryzyko aprioryczne obliczone z uwzględnieniem oceny zintegrowanej metodą probabilistyczną od wartości ryzyka bieżącego uzyskanej w scenariuszu przybliżono-realnym $\times 10^{-8}$

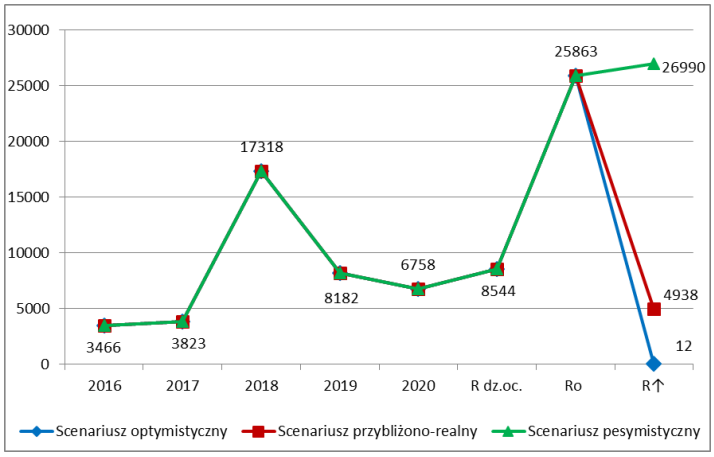
Dane pokazane na Rysunku 3.12 charakteryzują się innym trendem. Scenariusz optymistyczny ogranicza ryzyko do najniższej możliwej wartości, przybliżono-realny – utrzymuje się na poziomie 2016-2017, który charakteryzuje się minimalną wartością ryzyka, a dopiero scenariusz pesymistyczny wskazuje na szybki wzrost, czyli 64%.

Wyniki obliczeń wartości ryzyka przyszłego okresu z uwzględnieniem zintegrowanej oceny bazujących na wartości ryzyka bieżącego, uzyskanego w scenariuszu pesymistycznym, przedstawiono na Rysunkach 3.13, 3.14.

Ponieważ wartość ryzyka bieżącego obliczona w scenariuszu pesymistycznym rośnie w stosunku do wartości ryzyka aposteriorycznego, ryzyko uzyskane w wyniku prognozowania liniowego ma podobny trend. Tym samym ryzyko aprioryczne, uzyskane w scenariuszu optymistycznym, wzrasta o mniej niż 1% w stosunku do ryzyka bieżącego. Uzyskane w scenariuszu przybliżono-realnym ryzyko aprioryczne charakteryzuje się wzrostem w stosunku do ryzyka bieżącego o 23,6%. Jednocześnie najbardziej rośnie prognoza ryzyka w scenariuszu pesymistycznym, przekraczając wartość ryzyka bieżącego o 52,2%.




Rysunek 3.13: Ryzyko aprioryczne obliczone z uwzględnieniem oceny zintegrowanej metodą liniową od wartości ryzyka bieżącego uzyskanej w scenariuszu pesymistycznym $\times 10^{-8}$




Rysunek 3.14: Ryzyko aprioryczne obliczone z uwzględnieniem oceny zintegrowanej metodą probabilistyczną od wartości ryzyka bieżącego uzyskanej w scenariuszu pesymistycznym $\times 10^{-8}$

Wartości ryzyka obliczone przy użyciu prognozowania probabilistycznego nie mają tego samego trendu. Podobnie jak w przypadku poprzedniej prognozy, wyniki obliczeń w scenariuszu optymistycznym wskazują na redukcję ryzyka do możliwej najniższej wartości. Ryzyko aprioryczne uzy-

skane w scenariuszu przybliżono-realnym jest zmniejszone ponad 5-krotnie w stosunku do ryzyka bieżącego, ale pozostaje w przedziale ryzyk aposteriorycznych. Jednocześnie prognozowane ryzyko, liczone według scenariusza pesymistycznego, nieznacznie rośnie, co stanowi 4,4% wartości ryzyka bieżącego.

Dla opracowanego matematycznego modelu przeprowadzona ocena adekwatności według kryterium Fishera, która wykazała wystarczającą zbieżność wyników istniejących i przewidywanych, błąd nie przekracza 11,5%, co potwierdza możliwość jego zastosowania.

## 3.4. Podsumowanie

Przedstawiona praca jest kontynuacją dalszego rozwoju teoretycznych podstaw zarządzania ryzykiem w przedsiębiorstwie produkcyjnym. Mimo to, że ogólna ocena poziomu, wartości oraz skutków ryzyka jest tematem często podejmowanym w badaniach i pracach naukowców, podejmowanie prób tworzenia prognozy wymienionych wskaźników pozostaje poza obszarem ich zainteresowania. W treści niniejszej pracy zostały omówione artykuły, które zawierają opracowane metody i modele oceny ryzyka, w tym obliczenia jego wartości. Ta praca różni się tym, że przyjęty model pozwała na przewidywanie wartości ryzyk na przyszłe okresy działalności przedsiębiorstwa z uwzględnieniem wcześniejszy obliczonych wskaźników, co z kolei jest jej główna zaletą.

Model prognozowania ryzyka w przedsiębiorstwie produkcyjnym jest wpisany w specyfikę pracy. Stosowanie opracowanego modelu pozwala na tworzenie prognoz o trzech scenariuszach: optymistycznym, przybliżono-realnym i pesymistycznym. Prognozowanie wartości ryzyk w przedsiębiorstwie przeprowadzono metodą liniową oraz probabilistyczną. Oprócz ogólnych wskaźników sporządzanie prognoz ryzyk przeprowadzono z uwzględnieniem zintegrowanych wskaźników bezpieczeństwa. Na podstawie opracowanego modelu uzyskano określone wartości ryzyka, a następnie została przedstawiona analiza ich zmiany oraz dynamiki.

Takie możliwości pozwalają na rozszerzenie procesu zarządzania, nie ograniczając go *stricte* do działań bieżących, a zmusza do poszukiwania strategii stałego dbania o osiągnięcie oraz utrzymanie odpowiedniego poziomu ryzyka. Opracowany model nie jest bardzo skomplikowany i złożony,

więc jego stosowanie nie wymaga dużych zmian w strukturze organizacyjnej przedsiębiorstwa, poniesienia znaczących kosztów, doboru i rekrutacji wysoko kwalifikowanego personelu. Dodatkowo, w wyniku stosowania modelu w procesie zarządzania ryzykiem przedsiębiorstwo dostaje tak duży zbiór informacji o jego wartościach prognostycznych oraz tendencjach ich zmiany, że korzystając z niej może zapewnić odpowiedni poziom bezpieczeństwa w procesie podejmowania decyzji.

## Bibliografia


[1] Iso/iec 31010 risk management – risk assessment techniques (pn-en 31010:2010 zarządzanie ryzykiem–techniki oceny ryzyka).

[2] Pn – iso 31000:2018. risk management – guidelines.

[3] D. Baranovskyi, L. Muradian, M. Bulakh. The method of assessing traffic safety in railway transport. *IOP Conference Series: Earth and Environmental Science*, 666, 2021.

[4] M. Bulakh. Methodology for assessing the train traffic safety at the railway. *Science and transport progress*, 3 (87):138–146, 2020.

[5] M. Bulakh. Zarządzanie ryzykiem systemów logistycznych na podstawie analizy wielowymiarowej. *Monografia «Materiały, Technologie, Konstrukcje 1», Tom 1 «Predykcja w układach mechanicznych i automatycznych 2020 – modelowanie matematyczne i statystyczne» redakcja Janusz Weiss, Rozdział 3*, strony 73–97, 2020.

[6] M. Bulakh, A. Okorokov, D. Baranovskyi. Risk system and railway safety. *IOP Conference Series: Earth and Environmental Science*, 666, 2021.

[7] K. Czajkowska. Metody identyfikacji ryzyka w zarządzaniu ryzykiem w przedsiębiorstwie. *Journal of Modern Management Process*, 1(2):45–54, 2017.

[8] A. Gaschi. Wybrane metody identyfikacji ryzyka w procesach logistycznych. *Logistyka*, 4:122–129, 2013.

[9] M.K. Gąsowska. Wybrane problemy zarządzania ryzykiem operacyjnym w przedsiębiorstwie. *Zeszyty Naukowe. Tom III. Zarządzanie i Finanse*, 4:151–162, 2013.

[10] S. Gędek. Definiowanie ryzyka. *Prace naukowe uniwersytetu ekonomicznego we Wrocławiu*, 513:119–130, 2018.

[11] N. Iwaszczuk. *Ryzyko i bezpieczeństwo w działalności gospodarczej*. IGSMiE PAN, 2019.

[12] H. Lewandowska, O. Vikarczuk. Ryzyko w zarządzaniu przedsiębiorstwem. *Economics. Management. Innovations*, 05, 2020.

[13] A. Okorokov, M. Bulakh. Integral assessment of the state of railway train safety during technical audit. *Science and transport progress*, 5 (83):99–107, 2019.

[14] A. Polanowska. Analiza wybranych ryzyk z obszaru zarządzania zasobami ludzkimi. *Humanizacja pracy*, 2 (292):9–24, 2018.

[15] J. Stanik, R. Hoffmann, J. Napiórkowski. Zarządzanie ryzykiem w systemie zarządzania bezpieczeństwem organizacji. *Ekonomiczne Problemy Usług*, (123):321–336, 2016.

[16] A. K. Stasiuk-Piekarska. *Metodyka zarządzania ryzykiem organizacyjnym w systemach produkcyjnych. Rozprawa doktorska.* 2017.

[17] J. Stokłosa. Zarządzanie ryzykiem w łańcuchach transportowych. *Zeszyty Naukowe WSEI seria: Transport i Informatyka*, 1:87–98, 2011.

[18] P. Sulewski. Ryzyko w logistyce i sposoby jego minimalizacji. *Ekonomika i Organizacja Logistyki*, 2(4):71–83, 2017.

[19] M. Topczak, J. Patalas-Maliszewska. Model oceny poziomu ryzyka w przedsiębiorstwie produkcyjnym. *Zarządzanie Przedsiębiorstwem*, 22(4):14–21, 2019.

[20] J. Wrodarczyk. Zarządzanie ryzykiem w przedsiębiorstwie produkcyjnym zgodnie z założeniami teorii ograniczeń. *Modelowanie Preferencji a Ryzyko*, 96:215–225, 2011.

**Predicting safety risks in a manufacturing company**

**Abstract:** This article presents a model for predicting the magnitude of safety risk in a manufacturing company. The mathematical model allows you to calculate the magnitude of safety risks in a manufacturing company in the past, current and future periods. The model proposed in the work covers the approaches to forecasting according to three scenarios: optimistic, near-real and pessimistic. To obtain several results, forecasting was carried out using linear and probabilistic methods. Despite the general safety indicators, the prediction of priori risks of a manufacturing company was carried out taking into account integral indicators. On the basis of the developed mathematical model, the values of risks were obtained, and then an analysis of their dynamics was presented. The calculation results allow us to compare a wide range of possible values and take into account a huge number of possible safety methods.

# Spis rysunków

# Spis tablic