



Politechnika Wrocławskiego

Politechnika Wrocławskiego

Wydział Mechaniczny

Zarządzanie i Inżynieria Produkcji (ZIP)
Organizacja Produkcji (OP)

Uczenie maszynowe jako narzędzie doskonalenia procesów wytwórczych

Machine Learning as a Tool for Manufacturing Process Improvement

Praca dyplomowa magisterska

Autor: Mateusz A. Tabor
Promotor: Dr inż. Kamil Musiał

Wrocław, 2026

Streszczenie

Niniejsza praca magisterska przedstawia kompleksową analizę uczenia maszynowego jako narzędzi doskonalenia procesów wytwórczych w erze Przemysłu 4.0. Praca obejmuje przegląd literatury, szczegółową charakterystykę algorytmów oraz studium przypadku klasyfikacji defektów w produktach odlewniczych. Zaimplementowano i porównano sześć algorytmów: drzewa decyzyjne, lasy losowe, maszyny wektorów nośnych, sieci neuronowe, k-najbliższych sąsiadów oraz regresję logistyczną na zbiorze 1300 obrazów. Random Forest osiągnął najwyższą skuteczność (91,15% dokładności) przy krótkim czasie treningu (2,31 s). Praca dowodzi praktycznej przydatności uczenia maszynowego w automatycznej kontroli jakości i wskazuje kierunki dalszego rozwoju, w tym zastosowanie głębokiego uczenia i wdrożenia pilotażowe.

Słowa kluczowe: uczenie maszynowe, procesy wytwórcze, Przemysł 4.0, klasyfikacja defektów, wizja komputerowa, Random Forest

Abstract

This master's thesis presents a comprehensive analysis of machine learning as a tool for improving manufacturing processes in the Industry 4.0 era. The work includes a literature review, detailed characterization of algorithms, and a case study on defect classification in casting products. Six algorithms were implemented and compared: decision trees, random forests, support vector machines, neural networks, k-nearest neighbors, and logistic regression on a dataset of 1300 images. Random Forest achieved the highest effectiveness (91.15% accuracy) with short training time (2.31 s). The thesis demonstrates the practical utility of machine learning in automated quality control and indicates directions for further development, including deep learning applications and pilot implementations.

Keywords: machine learning, manufacturing processes, Industry 4.0, defect classification, computer vision, Random Forest

Spis treści

Streszczenie	1
Abstract	2
1 Wstęp	6
1.1 Kontekst i motywacja	6
1.2 Cel i zakres pracy	6
1.3 Metodyka badawcza	7
1.4 Oczekiwane rezultaty	7
2 Przegląd literatury	8
2.1 Podstawy uczenia maszynowego	8
2.2 Metody regresji i optymalizacji	8
2.3 Algorytmy klasyfikacji i drzewa decyzyjne	9
2.4 Algorytmy klastrowania i redukcji wymiarowości	9
2.5 Uczenie półnadzorowane i ze wzmacnieniem	9
2.6 Uczenie głębokie i sieci neuronowe	10
2.7 Przemysł 4.0 i inteligentne wytwarzanie	10
2.8 Konserwacja predykcyjna	10
2.9 Optymalizacja procesów produkcyjnych	11
2.10 Kontrola jakości i inspekcja wizualna	11
2.11 Podsumowanie	12
3 Charakterystyka algorytmów uczenia maszynowego	13
3.1 Uczenie nadzorowane (Supervised Learning)	13
3.1.1 Regresja liniowa (Linear Regression)	13
3.1.2 Regresja logistyczna (Logistic Regression)	15
3.1.3 k-Najbliższych Sąsiadów (k-Nearest Neighbors)	16
3.1.4 Drzewa decyzyjne (Decision Trees)	17
3.1.5 Las losowy (Random Forest)	18
3.1.6 Maszyna wektorów nośnych (Support Vector Machine, SVM)	20
3.1.7 Naiwny Klasyfikator Bayesowski (Naive Bayes)	21
3.2 Uczenie nienadzorowane (Unsupervised Learning)	22
3.2.1 k-Średnich (k-Means)	23
3.2.2 Hierarchiczna klasteryzacja (Hierarchical Clustering)	24
3.2.3 DBSCAN (Density-Based Spatial Clustering)	25
3.2.4 PCA (Principal Component Analysis)	26
3.2.5 t-SNE (t-distributed Stochastic Neighbor Embedding)	27
3.2.6 UMAP (Uniform Manifold Approximation and Projection)	29
3.3 Uczenie półnadzorowane (Semi-Supervised Learning)	30
3.3.1 Self-Training (Samouczenie)	30
3.3.2 Co-Training	31
3.3.3 Transductive SVM (TSVM)	32

3.3.4	Graph-Based Methods (Metody grafowe)	32
3.3.5	Generative Models (Modele generatywne)	33
3.3.6	Entropy Minimization (Minimalizacja entropii)	34
3.4	Uczenie ze wzmocnieniem (Reinforcement Learning)	35
3.4.1	Q-Learning	37
3.4.2	SARSA (State-Action-Reward-State-Action)	38
3.4.3	Deep Q-Network (DQN)	38
3.4.4	Policy Gradient Methods (Metody gradientu polityki)	39
3.5	Uczenie głębokie (Deep Learning)	41
3.5.1	Algorytm wstecznej propagacji (Backpropagation)	42
3.5.2	Metody optymalizacji	43
3.5.3	Regularyzacja w sieciach głębokich	43
3.5.4	Konwolucyjne sieci neuronowe (CNN)	44
3.5.5	Rekurencyjne sieci neuronowe (RNN)	45
4	Zastosowania uczenia maszynowego w procesach wytwarzczych	47
4.1	Wstęp	47
4.2	Konserwacja predykcyjna	47
4.2.1	Architektura systemów konserwacji predykcyjnej	48
4.2.2	Zastosowania branżowe i efekty wdrożeń	49
4.2.3	Korzyści ekonomiczne i operacyjne	50
4.3	Optymalizacja parametrów procesów produkcyjnych	50
4.3.1	Architektura systemów optymalizacji	51
4.3.2	Techniki i algorytmy optymalizacji	52
4.3.3	Zastosowania branżowe i rezultaty wdrożeń	52
4.4	Kontrola jakości i analiza predykcyjna	53
4.4.1	Automatyczna inspekcja wizualna oparta na uczeniu głębokim	53
4.4.2	Analiza predykcyjna jakości	55
4.4.3	Zastosowania branżowe	56
5	Studium przypadku: Klasyfikacja defektów w produktach odlewniczych	57
5.1	Wprowadzenie	57
5.2	Cel i uzasadnienie studium przypadku	57
5.2.1	Cel badania	57
5.2.2	Charakterystyka procesu produkcyjnego i systemu kontroli jakości	58
5.3	Definicja problemu decyzyjnego	58
5.4	Opis danych	59
5.5	Przygotowanie danych	60
5.5.1	Czyszczenie danych	60
5.5.2	Normalizacja i standaryzacja danych	60
5.5.3	Ekstrakcja cech	60
5.5.4	Podział danych na zbiory	61
5.5.5	Walidacja jakości przygotowania danych	61
5.6	Wybór i opis zastosowanego algorytmu uczenia maszynowego	61
5.6.1	K-najbliższych sąsiadów (k-Nearest Neighbors)	61
5.6.2	Drzewo Decyzyjne (Decision Tree)	62
5.6.3	Las Losowy (Random Forest)	63

5.6.4	Maszyna Wektorów Nośnych (Support Vector Machine)	64
5.6.5	Neural Network – Multi-Layer Perceptron (MLP)	65
5.6.6	Logistic Regression (Regresja Logistyczna)	66
5.7	Implementacja modelu i przebieg eksperymentu	67
5.7.1	Środowisko programistyczne	67
5.7.2	Wykorzystane biblioteki	67
5.7.3	Schemat procesu uczenia	68
5.7.4	Walidacja modeli	69
5.7.5	Strojenie hiperparametrów	69
5.7.6	Pomiar wydajności i metryki czasowe	69
5.7.7	Reprodukwalność eksperymentu	70
5.8	Wyniki i ich analiza	70
5.8.1	Wprowadzenie do analizy wyników	70
5.8.2	Charakterystyka zastosowanych metryk ewaluacyjnych	70
5.8.3	Wyniki badań empirycznych	72
5.8.4	Analiza porównawcza alternatywnych metod uczenia maszynowego	76
5.9	Podsumowanie wyników i ograniczenia rozwiązania	77
5.9.1	Mocne strony zaproponowanego podejścia	77
5.9.2	Ograniczenia rozwiązania	78
5.9.3	Wnioski końcowe	79
6	Wnioski końcowe	80
6.1	Podsumowanie	80
6.2	Najważniejsze wnioski	80
6.3	Ograniczenia badań	81
6.4	Kierunki dalszego rozwoju	81
6.5	Podsumowanie	81
Bibliografia		83

1

Wstęp

1.1 Kontekst i motywacja

Współczesne procesy wytwórcze stoją przed wyzwaniami takimi jak: rosnącą złożonością produktów, globalizacją łańcuchów dostaw, coraz wyższymi wymaganiami jakościowymi oraz presją na optymalizację kosztów. Tradycyjne podejścia zarządcze, oparte głównie na doświadczeniu ekspertów i statycznych modelach, okazują się coraz mniej efektywne. Jednocześnie nowoczesne systemy produkcyjne generują ogromne ilości danych z czujników, systemów monitorowania i kontroli jakości, które często pozostają niewykorzystane.

Przemysł 4.0 przynosi nowe możliwości wykorzystania tych danych poprzez uczenie maszynowe. Ta technologia umożliwia automatyczne wykrywanie wzorców w danych, budowanie modeli predykcyjnych oraz podejmowanie decyzji w czasie rzeczywistym. Zastosowanie algorytmów uczenia maszynowego w procesach wytwórczych otwiera perspektywy znaczającej poprawy efektywności, jakości produktów i niezawodności linii produkcyjnych.

Potencjał uczenia maszynowego w przemyśle jest szeroki: od konserwacji predykcyjnej, przez automatyczną kontrolę jakości z wykorzystaniem wizji komputerowej, po optymalizację parametrów procesowych. Pomimo rosnącego zainteresowania tematyką, praktyczne wdrożenia w polskim przemyśle są wciąż rzadkie, co wynika z barier kompetencyjnych, kulturowych i organizacyjnych.

1.2 Cel i zakres pracy

Celem pracy jest kompleksowe przedstawienie uczenia maszynowego jako narzędzia doskonalenia procesów wytwórczych, ze szczególnym uwzględnieniem teoretycznych podstaw, charakterystyki algorytmów oraz praktycznych zastosowań.

Praca realizuje następujące cele szczegółowe:

1. Przegląd literatury naukowej dotyczącej uczenia maszynowego i jego zastosowań w procesach wytwórczych, z uwzględnieniem fundamentalnych prac teoretycznych oraz badań nad implementacją w kontekście Przemysłu 4.0.
2. Charakterystyka podstawowych algorytmów uczenia maszynowego. Metod uczenia nadzorowanego, nienadzorowanego oraz uczenia przez wzmacnianie, wraz z omówieniem ich właściwości, zalet i ograniczeń.
3. Analiza zastosowań uczenia maszynowego w kluczowych obszarach procesów wytwórczych: konserwacja predykcyjna, kontrola jakości, optymalizacja procesów, prognozowanie popytu.

4. Przeprowadzenie studium przypadku poprzez wytrenowanie i porównanie modeli uczenia maszynowego do klasyfikacji defektów w produktach odlewniczych jako praktyczna weryfikacja omówionych metod.

Zakres pracy obejmuje zarówno aspekty teoretyczne, jak i praktyczne. Część teoretyczna dostarcza fundamentu metodologicznego, część praktyczna w postaci studium przypadku demonstruje pełny cykl projektu uczenia maszynowego, od przygotowania danych, przez trening modeli, aż po ewaluację wyników.

1.3 Metodyka badawcza

Przyjęta metodyka łączy przegląd literaturowy z eksperymentem obliczeniowym. Część teoretyczna opiera się na analizie klasycznych publikacji z zakresu uczenia maszynowego oraz współczesnych badań nad zastosowaniami przemysłowymi.

Część empiryczna wykorzystuje rzeczywisty zbiór danych zawierający obrazy produktów odlewniczych z oznaczeniem obecności lub braku defektów. Zaimplementowano sześć algorytmów klasyfikacji: drzewa decyzyjne, lasy losowe, maszyny wektorów nośnych, sieci neuronowe, k-najbliższych sąsiadów oraz regresję logistyczną. Wszystkie modele wytrenowano na tym samym zbiorze z jednolitą metodyką podziału 80:20 na zbiór treningowy i testowy. Ocena obejmuje standardowe metryki: dokładność, precyzję, czułość, miarę F1 oraz czas treningu.

1.4 Oczekiwane rezultaty

Praca ma dostarczyć kompleksowego spojrzenia na uczenie maszynowe jako narzędzie doskonalenia procesów wytwarzających. Oczekiwany rezultatem jest nie tylko przedstawienie stanu wiedzy i charakterystyki algorytmów, ale także konkretny przykład skutecznej implementacji. Poprzez identyfikację możliwości i ograniczeń poszczególnych metod, praca powinna przyczynić się do bardziej świadomego wykorzystania potencjału uczenia maszynowego w optymalizacji procesów produkcyjnych.

2

Przegląd literatury

W tym rozdziale przedstawiono przegląd literatury dotyczącej uczenia maszynowego i jego zastosowań w przemyśle. Omówiono podstawowe prace teoretyczne z zakresu ML, algorytmy klasyfikacji i regresji, a także współczesne badania nad wykorzystaniem sztucznej inteligencji w kontekście Przemysłu 4.0. Rozdział podzielono na sekcje tematyczne, które odpowiadają strukturze pracy.

2.1 Podstawy uczenia maszynowego

Podstawy teoretyczne uczenia maszynowego można znaleźć w kilku klasycznych podręcznikach, które do dziś są podstawową lekturą w tej dziedzinie. [Bishop \(2006\)](#) przedstawia uczenie maszynowe z perspektywy rozpoznawania wzorców, koncentrując się na metodach bayesowskich i sieciach neuronowych. Z kolei [Hastie et al. \(2009\)](#) skupiają się bardziej na aspektach statystycznych jak: regresja, klasyfikacja i selekcja zmiennych. [Murphy \(2012\)](#) prezentuje podejście probabilistyczne, łącząc klasyczne metody z nowoczesnymi algorytmami.

Bardzo przydatną pozycją jest również książka [James et al. \(2013\)](#), która daje przystępne wprowadzenie do uczenia maszynowego z licznymi przykładami w języku R. Ta książka kładzie szczególny nacisk na interpretowalność modeli i ich walidację, co jest bardzo ważne w przemyśle, gdzie trzeba rozumieć, dlaczego model podejmuje takie a nie inne decyzje.

Warto też wspomnieć o klasycznych pracach dotyczących klasyfikacji wzorców, takich jak [Duda et al. \(2001\)](#) oraz [Manning et al. \(2008\)](#). Te prace dostarczają teoretycznych podstaw dla algorytmów, które są dziś stosowane w przemysłowej kontroli jakości i diagnostyce systemów produkcyjnych.

2.2 Metody regresji i optymalizacji

Metody regresji są podstawą wielu zastosowań uczenia maszynowego w przemyśle, pozwalając modelować zależności między parametrami procesowymi a jakością lub wydajnością produkcji. [Birkes and Dodge \(1993\)](#) opisują różne metody regresji, w tym techniki odporne na wartości odstające, co jest przydatne w przemyśle, gdzie pomiary często są zaszumione.

W przypadku modeli wysokowymiarowych szczególnie ważne są techniki regularyzacji. [Hoerl and Kennard \(1970\)](#) wprowadzili regresję grzbietową (ridge regression), która rozwiązuje problem współliniowości predyktorów. [Tibshirani \(1996\)](#) zaproponował metodę LASSO (Least Absolute Shrinkage and Selection Operator), która nie tylko regularyzuje model, ale też automatycznie wybiera zmienne. [Zou and Hastie \(2005\)](#) rozwinięli tę koncepcję tworząc elastic net, który łączy zalety obu poprzednich metod.

Jeśli chodzi o implementację, warto zatrzymać się na pracach [Golub and Loan \(1996\)](#) oraz [Press et al. \(2007\)](#), które omawiają praktyczne algorytmy obliczeniowe. [Gelman and Hill \(2006\)](#) przedstawiają zaawansowane techniki analizy danych z wykorzystaniem modeli wielopoziomowych, te modele dobrze sprawdzają się w analizie danych produkcyjnych o strukturze hierarchicznej.

2.3 Algorytmy klasyfikacji i drzewa decyzyjne

Algorytmy oparte na drzewach decyzyjnych są bardzo popularne w przemyśle, głównie dlatego że są łatwe do interpretacji i dobrze radzą sobie z nieliniowymi zależnościami. Podstawy teoretyczne drzew decyzyjnych można znaleźć w pracach [Quinlan \(1986\)](#) oraz [Breiman et al. \(1984\)](#), które opisują miary podziału oparte na entropii i wskaźniku Giniego. Sama koncepcja entropii informacyjnej pochodzi od [Shannon \(1948\)](#) i stanowi podstawę kryteriów podziału w drzewach.

Dużym krokiem naprzód była praca [Breiman \(2001\)](#), która wprowadziła Random Forest. To metoda zespołowa (ensemble method), która łączy predykcje wielu drzew, co zwiększa dokładność i zmniejsza wariancję modelu.

Jeśli chodzi o optymalizację hiperparametrów, istnieje kilka ważnych prac: [Bergstra and Bengio \(2012\)](#) analizują skuteczność przeszukiwania losowego, [Akiba et al. \(2019\)](#) przedstawiają framework Optuna, [Snoek et al. \(2012\)](#) proponują optymalizację bayesowską, a [Feurer et al. \(2015\)](#) oraz [Olson et al. \(2016\)](#) rozwijają koncepcje automatycznego uczenia maszynowego (AutoML).

2.4 Algorytmy klastrowania i redukcji wymiarowości

Metody uczenia nienadzorowanego, szczególnie klastryzacja i redukcja wymiarowości, są przydatne przy eksploracyjnej analizie danych produkcyjnych i wykrywaniu anomalii. [Arthur and Vassilvitskii \(2007\)](#) przedstawiają ulepszony algorytm k-means (k-means++), który lepiej inicjalizuje centroidy i szybciej zbiega. [Ester et al. \(1996\)](#) zaproponowali algorytm DBSCAN oparty na gęstości, który potrafi identyfikować klastry o dowolnych kształtach i wykrywać obserwacje odstające, a to się przydaje przy detekcji defektów produkcyjnych.

Do wizualizacji danych wysokowymiarowych popularny jest algorytm t-SNE opisany przez [van der Maaten and Hinton \(2008\)](#) oraz nowsza metoda UMAP opracowana przez [McInnes et al. \(2018\)](#). Te techniki pozwalają zwizualizować złożone przestrzenie cech, co pomaga inżynierom zrozumieć strukturę danych procesowych.

2.5 Uczenie półnadzorowane i ze wzmacnieniem

Uczenie półnadzorowane (semi-supervised learning) to odpowiedź na częsty problem w przemyśle jakim jest brak wystarczającej ilości danych z etykietami. [Chapelle et al. \(2006\)](#) oraz [Zhu and Goldberg \(2009\)](#) dogłębnie omawiają tę tematykę, pokazując metody, które wykorzystują zarówno dane etykietowane, jak i nieetykietowane do budowy lepszych modeli.

Uczenie ze wzmacnieniem (reinforcement learning) omawiane jest w podręczniku [Sutton and Barto \(2018\)](#) oraz w pracy [Szepesvári \(2010\)](#). Ten framework pozwala na sekwencyjne podejmowanie decyzji i optymalizację procesów. Algorytmy RL, takie jak Q-learning czy SARSA, mogą być stosowane do adaptacyjnego sterowania procesami produkcyjnymi oraz optymalizacji parametrów w czasie rzeczywistym.

2.6 Uczenie głębokie i sieci neuronowe

Rewolucja związana z rozwojem uczenia głębokiego jest doskonale opisana w [Goodfellow et al. \(2016\)](#), jest to solidne źródło wiedzy o architekturach sieciowych, metodach treningu i zastosowaniach deep learning. Ważny przegląd [LeCun et al. \(2015\)](#) opublikowany w Nature podsumowuje osiągnięcia uczenia głębokiego i jego wpływ na wiele dziedzin, w tym wizję komputerową, przetwarzanie języka naturalnego czy systemy rekomendacyjne.

Sieci konwolucyjne (CNN), które są podstawą nowoczesnych systemów inspekcji wizualnej w przemyśle, oraz sieci rekurencyjne (RNN, LSTM) wykorzystywane do analizy szeregow czasowych danych procesowych - to kluczowe architektury omawiane w literaturze o uczeniu głębokim.

2.7 Przemysł 4.0 i inteligentne wytwarzanie

Koncepcja Przemysłu 4.0, którą opisują [Schwab \(2016\)](#) oraz [Kagermann et al. \(2013\)](#), określa ramy cyfrowej transformacji procesów wytwórczych. [Zhong et al. \(2017\)](#) przedstawiają przegląd inteligentnego wytwarzania w kontekście Przemysłu 4.0, wymieniając główne technologie: Internet Rzeczy (IoT), cyber-fizyczne systemy produkcyjne i uczenie maszynowe.

Ważny przegląd zastosowań ML w produkcji przedstawiają [Wuest et al. \(2016\)](#), analizując korzyści, wyzwania i praktyczne implementacje w różnych sektorach przemysłowych. [Cioffi et al. \(2020\)](#) rozszerzają tę analizę, identyfikując trendy i kierunki rozwoju AI w inteligentnej produkcji (systemy predykcyjne, adaptacyjne sterowanie i automatyzację decyzji).

[Monostori et al. \(2016\)](#) szczegółowo omawiają cyber-fizyczne systemy produkcyjne, które stanowią infrastrukturę dla implementacji algorytmów ML w środowisku fabrycznym. [Kusiak \(2018\)](#) definiuje koncepcję smart manufacturing, kładąc nacisk na integrację danych, analitykę w czasie rzeczywistym i adaptacyjne systemy sterowania.

2.8 Konserwacja predykcyjna

Konserwacja predykcyjna to jedno z najbardziej rozwiniętych zastosowań uczenia maszynowego w przemyśle. Klasyczna praca [Mobley \(2002\)](#) ustanawia podstawy teoretyczne konserwacji predykcyjnej, definiując strategie utrzymania ruchu i metodyki implementacyjne.

[Lee et al. \(2014\)](#) przedstawiają szczegółową metodologię prognostyki i zarządzania stanem technicznym dla maszyn wirujących, omawiają techniki przetwarzania sygnałów vibracyjnych, ekstrakcji cech i budowy modeli predykcyjnych. [Susto et al. \(2015\)](#) pro-

ponują podejście wieloklasifykatorowe, łączące różne algorytmy ML w celu zwiększenia niezawodności predykcji.

Przegląd literatury przedstawiony przez Carvalho et al. (2019) identyfikuje najczęściej stosowane metody uczenia maszynowego w konserwacji predykcyjnej - Random Forest, sieci neuronowe, SVM i modele zespołowe, analizując ich skuteczność w różnych zastosowaniach. Ran et al. (2019) oferują szerszy przegląd systemów konserwacji predykcyjnej, obejmując architekturę, cele biznesowe i metodyki implementacyjne.

Dalzochio et al. (2020) analizują aktualny stan konserwacji predykcyjnej w erze Przemysłu 4.0, wskazując problemy związane z jakością danych, interpretowalnością modeli i integracją z istniejącymi systemami IT. Zhang et al. (2019) przedstawiają przegląd metod opartych na danych dla predykcyjnego utrzymania ruchu urządzeń przemysłowych, ze szczególnym naciskiem na techniki uczenia głębokiego.

2.9 Optymalizacja procesów produkcyjnych

Zastosowanie uczenia maszynowego do optymalizacji parametrów procesowych to ważny obszar badań w kontekście inteligentnego wytwarzania. Wang et al. (2022b) przedstawiają przegląd big data analytics dla inteligentnych systemów produkcyjnych, analizując metody przetwarzania, analizy i wizualizacji danych w czasie rzeczywistym.

Sharp et al. (2018) oferują przegląd postępów w zastosowaniu ML w smart manufacturing, identyfikując trendy w optymalizacji procesów, predykcji jakości i adaptacyjnego sterowania. Weichert et al. (2019) skupiają się szczególnie na zastosowaniu ML do optymalizacji procesów produkcyjnych, omawiając optymalizację bayesowską, uczenie ze wzmacnieniem oraz modele zastępcze (surrogate models).

Psarommatis and Kiritsis (2022) proponują hybrydowy system wspomagania decyzji dla automatyzacji reakcji na defekty w ramach koncepcji Zero Defect Manufacturing (ZDM), integrując uczenie maszynowe z systemami MES i ERP. Shahriari et al. (2016) przedstawiają przegląd optymalizacji bayesowskiej, która się sprawdza przy efektywnym poszukiwaniu optymalnych parametrów procesowych, gdy liczba kosztownych eksperymentów jest ograniczona.

2.10 Kontrola jakości i inspekcja wizualna

Automatyczna kontrola jakości z wykorzystaniem wizji komputerowej i uczenia głębokiego to dynamicznie rozwijający się obszar badań. Weimer et al. (2016) pokazują, jak projektować architektury głębokich sieci konwolucyjnych do automatycznej ekstrakcji cech w przemysłowej inspekcji, udowadniając przewagę uczenia głębokiego nad tradycyjnymi metodami ręcznego inżynierowania cech.

Tabernik et al. (2020) proponują metodę segmentacji opartą na uczeniu głębokim do detekcji defektów powierzchniowych, osiągając wysoką dokładność wykrywania z jednoznaczną lokalizacją wad. Przegląd zastosowania deep learning w detekcji defektów przedstawiają Yang et al. (2020), analizując różne architektury sieciowe, metody treningu i wyzwanie implementacyjne.

Bergmann et al. (2021) wprowadzają dataset MVTec Anomaly Detection, który stanowi benchmark dla metod nienadzorowanego wykrywania anomalii w inspekcji przemys-

słowej, umożliwiając porównywanie skuteczności różnych algorytmów w realistycznych scenariuszach.

Jeśli chodzi o predykcyjną analitykę jakości, [Huang et al. \(2021\)](#) proponują podejście oparte na danych do predykcji jakości w złożonych procesach produkcyjnych, wykorzystując dane historyczne do budowy modeli prognozujących wskaźniki jakości jeszcze przed zakończeniem procesu. [Wang et al. \(2022a\)](#) przedstawiają przegląd algorytmów ML w kontroli jakości, obejmujący inspekcję wizualną, virtual metrology i systemy closed-loop quality control.

[Psarommatis et al. \(2020\)](#) analizują koncepcję Zero Defect Manufacturing, przedstawiając aktualny stan badań, ograniczenia obecnych podejść i przyszłe kierunki rozwoju w kontekście całkowitej eliminacji defektów przez predykcję i proaktywną kontrolę. [Cheng et al. \(2016\)](#) pokazują praktyczne zastosowanie koncepcji Przemysłu 4.0 w automatyzacji obróbki kół, integrując sensory IoT, analitykę w czasie rzeczywistym i adaptacyjne sterowanie.

2.11 Podsumowanie

Przegląd literatury pokazał, że choć fundamentalne algorytmy uczenia maszynowego są już dobrze ustanowione i teoretycznie udokumentowane, to ich praktyczne zastosowanie w procesach wytwórczych wciąż jest aktywnym obszarem badań. Istniejące prace skupiają się głównie na pokazywaniu skuteczności poszczególnych metod w izolowanych przypadkach, podczas gdy brakuje systematycznych porównań różnych podejść w takich samych warunkach produkcyjnych.

3

Charakterystyka algorytmów uczenia maszynowego

3.1 Uczenie nadzorowane (Supervised Learning)

Uczenie nadzorowane to technika, w której algorytm otrzymuje gotowy zestaw danych z wcześniej określonymi etykietami i uczy się na ich podstawie predykować etykiety dla nowych nieznanych danych. Dane uczące zawierają zmienne wejściowe oraz zmienną predykowaną (etykietę). Standardowy proces uczenia nadzorowanego obejmuje podział zbioru danych na zbiór treningowy, walidacyjny oraz testowy. Trening polega na dopasowaniu parametrów modelu do danych treningowych poprzez minimalizację funkcji straty, która mierzy różnicę między przewidywaniami modelu a rzeczywistymi etykietami. Po zakończeniu treningu model zostaje poddany ocenie na zbiorze walidacyjnym w celu doboru hiperparametrów oraz monitorowaniu uczenia maszynowego, aby zapobiec przeuczeniu modelu. Ostateczna ocena odbywa się na zbiorze testowym na danych nieznanych dla modelu i na podstawie tego modelu określana wartość dokładności, precyzji, recall, F1 (dla klasyfikacji) lub MSE, MAE (dla regresji)([Bishop, 2006](#); [Hastie et al., 2009](#)).

3.1.1 Regresja liniowa (Linear Regression)

Regresja liniowa to podstawowa technika statystyczna i uczenia maszynowego stosowana do modelowania zależności między zmienną zależną (predykowaną) a jedną lub więcej zmiennymi niezależnymi (cechami). Celem regresji liniowej jest znalezienie liniowej funkcji, która najlepiej opisuje związek między zmiennymi, umożliwiając przewidywanie wartości zmiennej zależnej na podstawie wartości zmiennych niezależnych ([Hastie et al., 2009](#); [James et al., 2013](#)).

Regresja liniowa prosta. W przypadku pojedynczej zmiennej niezależnej model można opisać równaniem ([James et al., 2013](#)):

$$y = \alpha + \beta x + \varepsilon, \quad (3.1)$$

gdzie y to zmienna zależna, x to zmienna niezależna, α to wyraz wolny, β to współczynnik nachylenia linii regresji, a ε to składnik losowy (błąd modelu).

Regresja wieloraka. Regresja wieloraka jest rozszerzeniem regresji prostej, poprzez używanie wielu zmiennych niezależnych. Zakłada się liniową zależność między zmienną zależną a kombinacją liniową zmiennych niezależnych ([Hastie et al., 2009](#)):

$$y = \alpha + \sum_{i=1}^p \beta_i x_i + \varepsilon, \quad (3.2)$$

gdzie y to zmienna zależna, α to wyraz wolny, x_i to zmienne niezależne, β_i to współczynniki regresji określające wpływ danej zmiennej na zmienną zależną, a ε to składnik losowy (błąd modelu).

Celem algorytmu regresji (prostej i wielorakiej) jest znalezienie optymalnych wartości współczynników β_i , które minimalizują błąd predykcji. Współczynniki można dobrać za pomocą różnych metod, jednak najczęściej stosuje się metodę najmniejszych kwadratów (OLS — Ordinary Least Squares), która minimalizuje sumę kwadratów różnic między rzeczywistymi a przewidywanymi wartościami zmiennej zależnej (Hastie et al., 2009).

Estymator OLS (macierzowy zapis).

$$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \mathbf{y} \quad (3.3)$$

Wzór (3.3) to macierzowy zapis estymatora OLS (Hastie et al., 2009; James et al., 2013). Wyjaśnienie:

- X — macierz projektująca (design matrix) o wymiarach $n \times p$ (lub $n \times (p+1)$, jeśli dodano kolumnę jedynek dla wyrazu wolnego),
- \mathbf{y} — wektor obserwacji o wymiarze $n \times 1$,
- $\hat{\boldsymbol{\beta}}$ — wektor estymowanych współczynników o wymiarze $p \times 1$.

Aby wyrażenie było poprawne, macierz $X^\top X$ musi być odwracalna (brak doskonałej multikolinearności). Przy klasycznych założeniach (m.in. $E[\varepsilon] = 0$, $\text{Var}(\varepsilon) = \sigma^2 I$) estymator jest nieobciążony, a jego wariancja wynosi $\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2 (X^\top X)^{-1}$. Gdy $X^\top X$ jest źle uwarunkowana lub nieodwracalna, stosuje się regularyzację (np. Ridge) lub metody numeryczne (gradient descent, SVD) (James et al., 2013; Hastie et al., 2009).

Inne metody estymacji współczynników regresji liniowej:

- Metoda gradientu prostego (Gradient Descent) — iteracyjna metoda optymalizacji minimalizująca funkcję straty poprzez aktualizację współczynników w kierunku przeciwnym do gradientu (Bishop, 2006; Murphy, 2012).
- Metoda najmniejszych modułów (Least Absolute Deviations) — minimalizuje sumę bezwzględnych różnic między rzeczywistymi a przewidywanymi wartościami, co czyni ją bardziej odporną na wartości odstające (Birkes and Dodge, 1993).
- Metoda Ridge Regression — wprowadza regularyzację L2, karę za duże wartości współczynników, co pomaga w radzeniu sobie z problemem multikolinearności i przeuczenia modelu (Hoerl and Kennard, 1970; Hastie et al., 2009).
- Metoda Lasso Regression — regularyzacja L1, która może prowadzić do zerowania niektórych współczynników, skutkując modelem o mniejszej liczbie cech (automatyczny wybór cech) (Tibshirani, 1996).
- Metoda Elastic Net — łączy regularyzację L1 i L2, co pozwala na lepsze dostosowanie modelu do danych (Zou and Hastie, 2005).

- Metoda SVD (Singular Value Decomposition) — rozkłada macierz projektującą na składniki, umożliwiając stabilne obliczenie współczynników regresji nawet w przypadku kolinearności cech ([Golub and Loan, 1996](#); [Press et al., 2007](#)).
- Metoda Bayesian Regression — wykorzystuje podejście bayesowskie do estymacji współczynników, uwzględniając niepewność i priorytety w modelu ([Gelman and Hill, 2006](#); [Bishop, 2006](#)).
- QR Decomposition — rozkłada macierz projektującą na iloczyn macierzy ortogonalnej i górnortrójkątnej, co umożliwia efektywne rozwiązywanie układu równań regresji ([Golub and Loan, 1996](#); [Press et al., 2007](#)).

3.1.2 Regresja logistyczna (Logistic Regression)

Regresja logistyczna to technika statystyczna i uczenia maszynowego stosowana do modelowania zależności między zmienną zależną a jedną lub więcej zmiennymi niezależnymi, gdy zmienna zależna przyjmuje wartości binarne. Celem regresji logistycznej jest przewidywanie prawdopodobieństwa przynależności do jednej z dwóch klas na podstawie wartości zmiennych niezależnych ([Hastie et al., 2009](#); [James et al., 2013](#)).

Model regresji logistycznej. Model regresji logistycznej można zapisać jako ([Hastie et al., 2009](#)):

$$P(Y = 1|X) = \sigma(\boldsymbol{\beta}^\top X) = \frac{1}{1 + e^{-\boldsymbol{\beta}^\top X}} \quad (3.4)$$

gdzie:

- $P(Y = 1|X)$ — prawdopodobieństwo, że zmienna zależna Y przyjmuje wartość 1, biorąc pod uwagę zmienne niezależne X ,
- $\sigma(z)$ — funkcja sigmoidalna, która przekształca dowolną wartość rzeczywistą z w przedział $(0, 1)$.

Estymacja parametrów. Parametry modelu $\boldsymbol{\beta}$ są estymowane za pomocą metody największej wiarygodności (Maximum Likelihood Estimation, MLE). Celem jest maksymalizacja funkcji wiarygodności ([Hastie et al., 2009](#)):

$$L(\boldsymbol{\beta}) = \prod_{i=1}^n P(Y_i|X_i; \boldsymbol{\beta}) \quad (3.5)$$

co jest równoważne minimalizacji funkcji straty ([Hastie et al., 2009](#)):

$$J(\boldsymbol{\beta}) = - \sum_{i=1}^n [Y_i \log(P(Y_i|X_i; \boldsymbol{\beta})) + (1 - Y_i) \log(1 - P(Y_i|X_i; \boldsymbol{\beta}))] \quad (3.6)$$

3.1.3 k-Najbliższych Sąsiadów (k-Nearest Neighbors)

Algorytm k -Najbliższych Sąsiadów umieszcza dane wejściowe w przestrzeni wielowymiarowej i klasyfikuje je na podstawie etykiet najbliższych sąsiadów w tej przestrzeni. Przestrzeń jest definiowana przez cechy danych, zbiór danych posiadający x cech jest reprezentowany w x -wymiarowej przestrzeni. Algorytm klasyfikując dany obiekt oblicza odległości między nim a wszystkimi innymi obiekty w przestrzeni, a następnie wybiera k najbliższych sąsiadów. Wartość k jest ustalana przed rozpoczęciem działania algorytmu. Niska wartość parametru k jest bardziej podatna na szумy w danych, podczas gdy wysoka wartość k może prowadzić do nadmiernego uogólnienia modelu ([Cover and Hart, 1967](#)).

Algorytm k -najbliższych sąsiadów może wykorzystywać różne metryki do obliczania odległości m.in:

Metryka Euklidesowa: Najpowszechniejsza metryka używana do obliczania odległości między dwoma punktami w przestrzeni wielowymiarowej. Definiowana jest jako:

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \quad (3.7)$$

gdzie p i q to dwa punkty w przestrzeni, a n to liczba wymiarów. Wyliczanie odległości metryką euklidesową polega na policzeniu różnicy odległości w każdym wymiarze dla dwóch punktów, zsumowaniu kwadratów tych różnic, a następnie wyciągnięciu pierwiastka kwadratowego z tej sumy ([Duda et al., 2001; Bishop, 2006](#)).

Metryka Manhattan: Metryka Manhattan, nazywana również metryką taksówkową lub L1, mierzy odległość między dwoma punktami jako sumę wartości bezwzględnych z różnic współzędnych. Definiowana jest jako:

$$d(p, q) = \sum_{i=1}^n |p_i - q_i| \quad (3.8)$$

gdzie p i q to dwa punkty w przestrzeni, a n to liczba wymiarów. Obliczana jest jest różnica wartości p i q dla każdego wymiaru, a następnie sumowane są wartości bezwzględne tych różnic ([Duda et al., 2001](#)).

Metryka Kosinusowa: Metryka Kosinusowa mierzy wyznacza odległość między dwoma punktami na podstawie wyliczonego kąta między nimi. Definiowana jest jako:

$$d(p, q) = 1 - \frac{p \cdot q}{\|p\| \|q\|} \quad (3.9)$$

gdzie p i q to dwa punkty w przestrzeni, $p \cdot q$ to iloczyn skalarny wektorów p i q , a $\|p\|$ i $\|q\|$ to normy (długości) tych wektorów. Normy długości wektorów są obliczane jako pierwiastki kwadratowe z sumy kwadratów ich współrzędnych ([Manning et al., 2008; Bishop, 2006](#)).

Metryka Minkowskiego: Metryka Minkowskiego jest uogólnieniem metryk Euklidesowej i Manhattan. Umożliwia regulowanie sposobu obliczania odległości poprzez parametr

p. Definiowana jest jako:

$$d(p, q) = \left(\sum_{i=1}^n |p_i - q_i|^p \right)^{\frac{1}{p}} \quad (3.10)$$

gdzie p i q to dwa punkty w przestrzeni, n to liczba wymiarów, a p to parametr regulujący sposób obliczania odległości. Dla $p = 1$ metryka Minkowskiego jest równoważna metryce Manhattan, a dla $p = 2$ jest równoważna metryce Euklidesowej ([Hastie et al., 2009](#)).

3.1.4 Drzewa decyzyjne (Decision Trees)

Drzewa decyzyjne to algorytm uczenia maszynowego, która służy do podejmowania decyzji na podstawie zestawu reguł, które są reprezentowane w formie drzewa. Drzewo decyzyjne składa się z korzenia, które jest cechą dzielącą dane na grupy, węzłów wewnętrznych, które reprezentują pytania dotyczące cech danych, oraz liści, które reprezentują ostateczne decyzje lub klasyfikacje. Algorytm budowy drzewa decyzyjnego polega na iteracyjnym dzieleniu danych na podzbiory na podstawie cech, które najlepiej rozdzielały dane według określonego kryterium. Drzewo decyzyjne jest budowane, aż wszystkie elementy w podzbiorze należą do tej samej klasy lub nie ma już cech do podziału, osiągnie maksymalną głębokość lub kiedy dalszy podział nie poprawia jakości klasyfikacji ([Quinlan, 1986; Breiman et al., 1984](#)).

Drzewo decyzyjne do wyboru najlepszego podziału danych może wykorzystywać różne kryteria, m.in.:

Entropia: Entropia jest miarą niepewności lub nieuporządkowania w zbiorze danych. Entropia jest wykorzystywana do oceny jakości podziału danych na podstaiwe cechy. Definiowana jest jako:

$$H(S) = - \sum_{i=1}^c p_i \log_2(p_i) \quad (3.11)$$

gdzie S to zbiór danych, c to liczba klas, a p_i to proporcja elementów należących do klasy i w zbiorze S . Entropia obliczana jest jako suma iloczynów proporcji klas i logarytmów tych proporcji, a następnie mnożona przez -1 ([Shannon, 1948; Quinlan, 1986](#)).

Wskaźnik Gini (Gini Impurity): Wskaźnik Gini mierzy prawdopodobieństwo błędnej klasyfikacji losowo wybranego elementu, gdyby został on oznaczony losowo według rozkładu etykiet w węźle. Im niższy wskaźnik Gini, tym bardziej jednorodny jest węzeł. Wskaźnik Gini dla zbioru S jest definiowany jako:

$$\text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2 \quad (3.12)$$

gdzie c to liczba klas, a p_i to proporcja przykładów w klasie i . Wskaźnik Gini jest używany w algorytmie CART (Classification and Regression Trees) ze względu na swoją prostotę obliczeniową i dobrą wydajność ([Breiman et al., 1984](#)).

Zysk informacji (Information Gain): Zysk informacji mierzy redukcję entropii osiągniętą przez podział zbioru danych według danego atrybutu. Jest to różnica między

entropią zbioru nadzawanego a ważoną sumą entropii zbiorów potomnych. Zysk informacji dla atrybutu A w zbiorze S definiuje się jako:

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v) \quad (3.13)$$

gdzie $\text{Values}(A)$ to zbiór wszystkich możliwych wartości atrybutu A , S_v to podzbiór S , dla którego atrybut A ma wartość v , a $H(S)$ to entropia zbioru. Algorytm ID3 wybiera atrybut o największym zysku informacji ([Quinlan, 1986](#)).

Współczynnik zysku (Gain Ratio) Współczynnik zysku jest modyfikacją zysku informacji, która koryguje tendencję do faworyzowania atrybutów o wielu wartościach. Normalizuje zysk informacji przez podzielenie go przez tzw. split information, która mierzy szerokość i jednolitość podziału. Współczynnik zysku dla atrybutu A w zbiorze S definiuje się jako:

$$\text{GainRatio}(S, A) = \frac{IG(S, A)}{\text{SplitInfo}(S, A)} \quad (3.14)$$

gdzie $IG(S, A)$ to zysk informacji dla atrybutu A , a $\text{SplitInfo}(S, A)$ to informacja o podziale, definiowana jako:

$$\text{SplitInfo}(S, A) = - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right) \quad (3.15)$$

gdzie $\text{Values}(A)$ to zbiór wszystkich możliwych wartości atrybutu A , S_v to podzbiór S zawierający elementy, dla których atrybut A ma wartość v , $|S_v|$ to liczba elementów w podzbiorze S_v , a $|S|$ to całkowita liczba elementów w zbiorze S . Informacja o podziale mierzy, jak bardzo atrybut dzieli dane. Im bardziej równomierny podział, tym wyższa wartość SplitInfo , co zmniejsza współczynnik zysku i zapobiega faworyzowaniu atrybutów o wielu unikalnych wartościach. Współczynnik zysku jest używany w algorytmie C4.5 jako ulepszona wersja ID3 ([Quinlan, 1993](#)).

Redukcja wariancji (Variance Reduction) Redukcja wariancji jest kryterium stosowanym w drzewach regresyjnych, gdzie celem jest przewidywanie wartości ciągłych, a nie kategorii. Kryterium to wybiera podział, który maksymalnie redukuje wariancję wartości docelowych w węzłach potomnych. Redukcja wariancji dla podziału zbioru S na podzbiory S_{left} i S_{right} definiuje się jako:

$$\text{VarReduction}(S) = \text{Var}(S) - \left(\frac{|S_{\text{left}}|}{|S|} \text{Var}(S_{\text{left}}) + \frac{|S_{\text{right}}|}{|S|} \text{Var}(S_{\text{right}}) \right) \quad (3.16)$$

gdzie $\text{Var}(S)$ oznacza wariancję wartości docelowych w zbiorze S . Algorytm CART dla regresji wykorzystuje to kryterium do budowy drzew regresyjnych ([Breiman et al., 1984](#)).

3.1.5 Las losowy (Random Forest)

Las losowy to algorytm uczenia maszynowego, który łączy wiele drzew decyzyjnych w celu poprawy dokładności przewidywań i redukcji przeuczenia. Algorytm ten działa

na zasadzie tworzenia wielu niezależnych drzew decyzyjnych, z których każde jest trenowane na losowym podzbiorze danych i losowym podzbiorze cech. Ostateczna predykcja lasu losowego jest uzyskiwana przez agregację wyników wszystkich drzew. Dla klasyfikacji stosuje się głosowanie większościowe, a dla regresji średnią arytmetyczną (Breiman, 2001).

Las losowy uczy się poprzez losowanie i budowanie wielu drzew decyzyjnych. Każde drzewo dostaje losowy podzbiór danych treningowych oraz losowy podzbiór cech do rozważenia przy każdym podziale węzła. Ten proces losowania danych i cech wprowadza różnorodność między drzewami, co przekłada się na lepszą ogólną wydajność modelu. Poszczególne drzewa dzielone są na podstawie kryteriów omówionych w sekcji dotyczącej drzew decyzyjnych, takich jak entropia, wskaźnik Gini czy zysk informacji (Breiman, 2001).

Dodatkowo przy budowie lasu losowego dobiera się parametry:

Tabela 3.1: Podstawowe hiperparametry lasu losowego

Parametr	Opis
Liczba drzew (B)	Liczba drzew decyzyjnych w lesie. Większa liczba zazwyczaj poprawia wydajność, ale zwiększa czas obliczeń.
Maksymalna głębokość	Maksymalna głębokość każdego drzewa. Ograniczenie głębokości może zapobiec przeuczeniu.
Liczba cech (m)	Liczba losowo wybranych cech rozważanych przy każdym podziale. Typowo $m = \sqrt{p}$ dla klasyfikacji lub $m = p/3$ dla regresji. (p to liczba wszystkich cech)
Minimalna liczba próbek	Minimalna liczba próbek wymagana do podziału węzła wewnętrznego lub do utworzenia liścia.
Bootstrap	Czy stosować bootstrap sampling (losowanie ze zwracaniem) do tworzenia podzbiorów treningowych.

Strojenie hiperparametrów lasu losowego, jest zadaniem człowieka, ale najczęściej wykorzystuje się metody automatyczne do testowania.

Przeszukiwanie Siatki (Grid Search) Grid Search polega na przeszukiwaniu wszystkich możliwych kombinacji hiperparametrów z wcześniej zdefiniowanej siatki wartości (James et al., 2013; Hastie et al., 2009). Dla każdej kombinacji algorytm trenuje model i ocenia jego wydajność za pomocą walidacji krzyżowej. Następnie wybiera zestaw parametrów dający najlepsze wyniki według ustalonej metryki (Hastie et al., 2009).

Przeszukiwanie Losowe (Randomized Search) Randomized Search jest wariantem Grid Search, który zamiast sprawdzać wszystkie kombinacje, losowo próbuje określoną liczbę zestawów hiperparametrów z zadanych rozkładów (James et al., 2013). Liczba iteracji jest definiowana przez programistę.

Optuna Optuna to framework do optymalizacji hiperparametrów wykorzystujący zaawansowane algorytmy, takie jak Tree-structured Parzen Estimator (TPE). W przeciwieństwie do metod grid i random search, Optuna adaptacyjnie wybiera kolejne kombinacje hi-

perparametrów na podstawie wyników wcześniejszych prób. Uczy się, które obszary przestrzeni są obiecujące i koncentruje tam przeszukiwanie. Framework oferuje elastyczność w definiowaniu przestrzeni parametrów, wbudowane wizualizacje oraz możliwość przycinania nieobiecujących prób (Akiba et al., 2019).

Optymalizacja bayesowska (Bayesian Optimization) Bayesian Optimization buduje probabilistyczny model zastępczy (surrogate model), zazwyczaj Gaussian Process, który aproksymuje funkcję celu (np. dokładność modelu jako funkcję hiperparametrów). Na podstawie tego modelu algorytm wybiera kolejne punkty do próbkowania za pomocą funkcji akwizycji (acquisition function), takiej jak Expected Improvement (EI), która balansuje eksplorację nowych obszarów przestrzeni i eksplotację obiecujących regionów (Snoek et al., 2012).

AutoML (Automated Machine Learning) AutoML automatyzuje cały proces budowy modelu uczenia maszynowego, w tym wybór algorytmu, inżynierię cech, dobór hiperparametrów oraz tworzenie modeli zespołowych. Narzędzia takie jak Auto-sklearn, TPOT czy H2O AutoML wykorzystują kombinację technik optymalizacji (bayesian optimization, evolutionary algorithms) oraz wiedzę z wcześniejszych eksperymentów na podobnych zbiorach (meta-learning) (Feurer et al., 2015; Olson et al., 2016).

3.1.6 Maszyna wektorów nośnych (Support Vector Machine, SVM)

Maszyna wektorów nośnych (SVM) to algorytm uczenia maszynowego stosowany do klasyfikacji i regresji, który działa na zasadzie znajdowania optymalnej hiperpłaszczyzny rozdzielającej dane należące do różnych klas w przestrzeni wielowymiarowej (Bishop, 2006; Hastie et al., 2009). Celem SVM jest maksymalizacja marginesu, czyli odległości między hiperpłaszczyzną a najbliższymi punktami z obu klas, zwany wektorami nośnymi (support vectors). Większy margines prowadzi do lepszej generalizacji modelu na nowe, nieznane dane (Bishop, 2006; Hastie et al., 2009).

Zasada działania. Dla liniowo separowalnych danych, SVM znajduje hiperpłaszczyznę definiowaną jako (Bishop, 2006):

$$\mathbf{w}^\top \mathbf{x} + b = 0 \quad (3.17)$$

gdzie \mathbf{w} to wektor normalny do hiperpłaszczyzny, \mathbf{x} to wektor cech, a b to wyraz wolny. Funkcja decyzyjna klasyfikuje punkt \mathbf{x} na podstawie znaku wyrażenia $\mathbf{w}^\top \mathbf{x} + b$ (Hastie et al., 2009):

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b) \quad (3.18)$$

Optymalna hiperpłaszczyzna jest wyznaczana przez rozwiązanie problemu optymalizacji (Bishop, 2006; Hastie et al., 2009):

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{przy ograniczeniach} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i \quad (3.19)$$

gdzie $y_i \in \{-1, +1\}$ to etykiety klas, a \mathbf{x}_i to wektory treningowe. Problem ten jest rozwiązywany za pomocą metod programowania kwadratowego lub przez przekształcenie do postaci dualnej z wykorzystaniem mnożników Lagrange'a (Bishop, 2006).

Soft Margin SVM. W praktyce dane często nie są liniowo separowalne lub zawierają szумy. W takich przypadkach stosuje się wariant soft margin SVM, który dopuszcza błędy klasyfikacji poprzez wprowadzenie zmiennych slackowych $\xi_i \geq 0$ (Hastie et al., 2009; James et al., 2013). Problem optymalizacji przyjmuje wtedy postać:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad (3.20)$$

przy ograniczeniach:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i \quad (3.21)$$

gdzie $C > 0$ to parametr regularizacji kontrolujący kompromis między maksymalizacją marginesu a minimalizacją błędów klasyfikacji. Niskie wartości C preferują większy margines (tolerancja na błędy), wysokie wartości C zmuszają model do dokładniejszego dopasowania danych treningowych (Hastie et al., 2009; James et al., 2013).

Sztuczka jądrowa (Kernel Trick). Gdy dane nie są liniowo separowalne w oryginalnej przestrzeni cech, SVM wykorzystuje funkcje jądrowe (kernel functions) do mapowania danych do przestrzeni o wyższym wymiarze, w której separacja liniowa staje się możliwa (Bishop, 2006; Hastie et al., 2009). Najpopularniejsze funkcje jądrowe to:

Jądro liniowe — dla liniowo separowalnych danych (Hastie et al., 2009):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j \quad (3.22)$$

Jądro wielomianowe — pozwala na nieliniowe granice decyzyjne (Bishop, 2006; Hastie et al., 2009):

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^\top \mathbf{x}_j + r)^d \quad (3.23)$$

gdzie d to stopień wielomianu, γ to parametr skalowania, a r to wyraz wolny.

Jądro radialnej funkcji bazowej (RBF, Gaussian) — najczęściej stosowane, elastyczne (Bishop, 2006; Hastie et al., 2009):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (3.24)$$

gdzie $\gamma > 0$ kontroluje zasięg wpływu pojedynczych punktów treningowych. Małe γ daje szerokie „dzwony” (prostsze modele), duże γ — wąskie (ryzyko przeuczenia) (Hastie et al., 2009).

Jądro sigmoidalne — zachowuje się podobnie do sieci neuronowych (Bishop, 2006):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^\top \mathbf{x}_j + r) \quad (3.25)$$

Dzięki sztuczce jądrowej SVM może modelować złożone, nieliniowe granice decyzyjne bez jawnego obliczania transformacji do wyższego wymiaru, co jest kosztowne obliczeniowo (Bishop, 2006; Murphy, 2012).

3.1.7 Naiwny Klasyfikator Bayesowski (Naive Bayes)

Naiwny klasyfikator Bayesowski to probabilistyczny algorytm klasyfikacji oparty na twierdzeniu Bayesa z założeniem warunkowej niezależności cech (Bishop, 2006; Murphy, 2012). Algorytm oblicza prawdopodobieństwo przynależności obiektu do każdej z klas

na podstawie wartości jego cech. Po obliczeniu przypisuje obiekt do klasy o najwyższy prawdopodobieństwie po uwzględnieniu danych. Pomimo upraszczającego założenia o niezależności cech, Naive Bayes często osiąga dobre wyniki, szczególnie w zadaniach klasyfikacji tekstu i filtrowania spamu (Bishop, 2006; Murphy, 2012).

Twierdzenie Bayesa. Podstawą algorytmu jest twierdzenie Bayesa, które wyraża prawdopodobieństwo przynależności obiektu do klasy C_k przy danych cechach $\mathbf{x} = (x_1, x_2, \dots, x_n)$ (Bishop, 2006):

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})} \quad (3.26)$$

gdzie $P(C_k|\mathbf{x})$ oznacza prawdopodobieństwo a posteriori (po uwzględnieniu danych) przynależności do klasy C_k przy danych cechach \mathbf{x} , $P(\mathbf{x}|C_k)$ to prawdopodobieństwo wystąpienia cech \mathbf{x} w klasie C_k (tzw. likelihood), a $P(C_k)$ to prawdopodobieństwo a priori (przed zobaczeniem danych) klasy C_k . Mianownik $P(\mathbf{x})$ jest prawdopodobieństwem wystąpienia cech \mathbf{x} i pełni rolę stałej normalizującej (Murphy, 2012).

Warianty algorytmu. W zależności od charakteru danych stosuje się różne warianty Naive Bayes (Murphy, 2012):

Gaussian Naive Bayes. Wariant stosowany dla cech ciągłych, zakłada, że wartości każdej cechy w danej klasie mają rozkład normalny (Gaussa) (Bishop, 2006; Murphy, 2012), a parametry rozkładu (średnia μ_k i wariancja σ_k^2) są estymowane z danych treningowych dla każdej cechy i każdej klasy. Gaussowy wariant naiwnego klasyfikatora Bayesowskiego jest szczególnie skuteczny, gdzie cechy są liczbami rzeczywistymi, takimi jak pomiary fizyczne czy dane sensoryczne (Bishop, 2006).

Multinomial Naive Bayes. Wariant przeznaczony dla cech dyskretnych reprezentujących liczby wystąpień zdarzeń, takich jak częstotliwość występowania słów w dokumentach tekstowych (Murphy, 2012). Modeluje prawdopodobieństwo wystąpienia danej liczby zdarzeń zgodnie z rozkładem wielomianowym. Algorytmem jest szczególnie efektywny w zadaniach, gdzie dane są reprezentowane jako wektory liczników. Ten wariant jest często stosowany do klasyfikacji tekstu, filtrowania spamu oraz analizy sentymentu, gdzie każda cecha reprezentuje liczbę wystąpień określonego słowa (Murphy, 2012; Manning et al., 2008).

Bernoulli Naive Bayes. Wariant dla cech binarnych przyjmujących wartości 0 lub 1 (Murphy, 2012). W odróżnieniu od wariantu wielomianowego, który liczy wystąpienia, Bernoulli Naive Bayes modeluje jedynie fakt obecności cechy. Jest stosowany w klasyfikacji dokumentów, gdzie cechy wskazują, czy danie słowo występuje w dokumencie, niezależnie od liczby jego wystąpień (Murphy, 2012; Manning et al., 2008).

3.2 Uczenie nienadzorowane (Unsupervised Learning)

Uczenie nienadzorowane to rodzaj uczenia maszynowego, który sam odkrywa wzorce i struktury danych bez ówcześnie nadanych etykiet przez człowieka. Alogrytm przetwarza surowe dane wejściowe, a następnie za pomocą metod statystycznych i geometrycz-

nych grupuje je na podstawie podobieństw lub różnic. Po grupowaniu algorytm redukuje liczbę wymiarów danych, zachowując najważniejsze cechy i wzorce. Uczenie nienadzorowane dzieli się na dwie główne kategorie: klasteryzację i redukcję wymiarowości (Bishop, 2006; Hastie et al., 2009).

Klasteryzacja (Clustering) Klasteryzacja to technika uczenia nienadzorowanego, która polega na grupowaniu podobnych obiektów w zbiory zwane klastrami. Celem klasteryzacji jest identyfikacja naturalnych struktur w danych, gdzie obiekty w tym samym klastrze są bardziej podobne do siebie niż do obiektów z innych klastrów. Algorytmy klasteryzacji wykorzystują różne metryki odległości do oceny podobieństwa między obiektami, takie jak metryka euklidesowa, Manhattan czy kosinusowa, które zostały omówione w sekcji dotyczącej algorytmu k -najbliższych sąsiadów (Duda et al., 2001; Hastie et al., 2009).

Redukcja wymiarowości (Dimensionality Reduction) Redukcja wymiarowości to technika uczenia nienadzorowanego, która polega na zmniejszeniu liczby cech w zbiorze danych przy jednoczesnym zachowaniu jak największej ilości istotnej informacji. Celem redukcji wymiarowości jest uproszczenie modelu, poprawa wydajności obliczeniowej oraz eliminacja szumów i nadmiarowości w danych. Popularne metody redukcji wymiarowości to analiza głównych składowych (PCA) oraz t-SNE (t-distributed Stochastic Neighbor Embedding) (Bishop, 2006; Murphy, 2012).

3.2.1 k -Średnich (k-Means)

Algorytm k -średnich to jedna z metod klasteryzacji, która grupuje dane wejściowe w k klastrów na podstawie podobieństwa między obiektami (Bishop, 2006; Hastie et al., 2009). Algorytm działa iteracyjnie, przypisując każdy obiekt do najbliższego centroidu (środka klastra) i aktualizując położenie centroidów na podstawie średnich wartości obiektów w każdym klastrze. Proces ten powtarza się, aż do osiągnięcia zbieżności, czyli gdy przypisania obiektów do klastrów przestają się zmieniać (Bishop, 2006; Hastie et al., 2009).

$$J = \sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \quad (3.27)$$

gdzie J to całkowita suma kwadratów błędów, które są minimalizowane iteracyjnie przez algorytm (Hastie et al., 2009), j to iteracja, k to liczba klastrów, \mathbf{x}_i to wektor cech obiektu i , C_j to zbiór obiektów przypisanych do klastra j , a $\boldsymbol{\mu}_j$ to centroid klastra j .

W algorytmie k -średnich dobiera się następujące parametry:

Tabela 3.2: Podstawowe parametry algorytmu k -średnich

Parametr	Opis
Liczba klastrów (k)	Liczba klastrów, na które mają być podzielone dane. Wybór odpowiedniej wartości k jest kluczowy dla jakości klasteryzacji (np. metoda łokcia).
Inicjalizacja centroidów	Metoda wyboru początkowych pozycji centroidów (np. metoda k -means++).
Maksymalna liczba iteracji	Maksymalna liczba iteracji, które algorytm wykona przed zatrzymaniem.
Metryka odległości	Metryka używana do obliczania odległości między obiektami a centroidami, (np. metryka euklidesowa, Manhattan czy kosinusowa, które zostały opisane w sekcji dotyczącej algorytmu k -najbliższych sąsiadów).

Metoda łokcia (Elbow Method). Metoda łokcia polega na uruchomieniu algorytmu k -średnich dla różnych wartości k i obliczeniu sumy kwadratów błędów (SSE) dla każdej wartości (Hastie et al., 2009). Następnie wykresla się wykres SSE w funkcji k i szuka punktu, w którym dalsze zwiększenie k prowadzi do niewielkiej redukcji SSE, tworząc charakterystyczny "łokieć" na wykresie.

Metoda k -means++. Metoda wyboru centroidów k -means++ polega na wyborze początkowych centroidów, aby przyspieszyć zbieżność algorytmu i poprawić jakość klasteryzacji (Arthur and Vassilvitskii, 2007). Pierwszy centroid jest wybierany losowo z danych, a kolejne centroidy są wybierane z prawdopodobieństwem proporcjonalnym do kwadratu odległości od najbliższego już wybranego centroidu. Ta metoda zmniejsza ryzyko złego rozmieszczenia początkowych centroidów, co może prowadzić do gorszych wyników klasteryzacji (Arthur and Vassilvitskii, 2007).

3.2.2 Hierarchiczna klasteryzacja (Hierarchical Clustering)

Hierarchiczna klasteryzacja to metoda klasteryzacji, która tworzy hierarchię klastrów poprzez iteracyjne łączenie lub dzielenie grup obiektów na podstawie ich podobieństwa (Hastie et al., 2009; Bishop, 2006). Istnieją dwa główne podejścia do hierarchicznej klasteryzacji: aglomeracyjne (bottom-up) i dzielące (top-down). W podejściu aglomeracyjnym każdy obiekt zaczyna jako oddzielny klaster, a następnie iteracyjnie łączy się najbliższe klastry, aż do osiągnięcia jednej grupy lub określonej liczby klastrów. W podejściu dzielącym cały zbiór danych zaczyna jako jeden klaster, który jest następnie dzielony na mniejsze klastry na podstawie podobieństwa obiektów (Hastie et al., 2009; Bishop, 2006).

Hierarchiczna klasteryzacja wykorzystuje różne metryki odległości do oceny podobieństwa między obiektami lub klastrami, takie jak metryka euklidesowa, Manhattan czy kosinusowa, które zostały omówione w sekcji dotyczącej algorytmu k -najbliższych sąsiadów (Duda et al., 2001; Hastie et al., 2009). W hierarchicznej klasteryzacji stosuje się również różne metody łączenia klastrów, takie jak (Hastie et al., 2009):

Metoda pojedynczego łączenia (Single Linkage). Metoda pojedynczego łączenia definiuje odległość między dwoma klastrami jako minimalną odległość między dowolnymi dwoma obiektami z tych klastrów (Hastie et al., 2009):

$$d(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \quad (3.28)$$

gdzie C_i i C_j to dwa klastry, a $d(\mathbf{x}, \mathbf{y})$ to odległość między obiektami \mathbf{x} i \mathbf{y} . Jest to podejście zachłanne, które może prowadzić do tworzenia długich, cienkich klastrów (tzw. efekt łańcucha) (Hastie et al., 2009).

Metoda pełnego łączenia (Complete Linkage). Metoda pełnego łączenia definiuje odległość między dwoma klastrami jako maksymalną odległość między dowolnymi dwoma obiektami z tych klastrów (Hastie et al., 2009):

$$d(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \quad (3.29)$$

To podejście prowadzi do tworzenia bardziej zwartych klastrów, ale może być wrażliwe na odległe punkty (outliers) (Hastie et al., 2009).

Metoda średniego łączenia (Average Linkage). Metoda średniego łączenia definiuje odległość między dwoma klastrami jako średnią odległość między wszystkimi parami obiektów z tych klastrów (Hastie et al., 2009):

$$d(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \quad (3.30)$$

gdzie $|C_i|$ i $|C_j|$ to liczby obiektów w klastrach C_i i C_j . To podejście stanowi kompromis między metodą pojedynczego i pełnego łączenia, tworząc bardziej zrównoważone klastry (Hastie et al., 2009).

3.2.3 DBSCAN (Density-Based Spatial Clustering)

DBSCAN to algorytm klasteryzacji oparty na gęstości, który grupuje razem punkty znajdujące się blisko siebie w przestrzeni cech, definiując klastry jako obszary o wysokiej gęstości punktów oddzielone od siebie obszarami o niskiej gęstości (Ester et al., 1996). Algorytm DBSCAN identyfikuje klastry na podstawie dwóch głównych parametrów: promienia sąsiedztwa ε (epsilon) oraz minimalnej liczby punktów $minPts$ wymaganej do utworzenia gestego regionu. Punkty są klasyfikowane jako rdzeniowe, brzegowe lub szumowe w zależności od liczby sąsiadów w promieniu ε (Ester et al., 1996).

Sąsiedztwo ε . Dla punktu \mathbf{p} sąsiedztwo ε definiowane jest jako zbiór punktów (Ester et al., 1996):

$$N_\varepsilon(\mathbf{p}) = \{\mathbf{q} \in D \mid d(\mathbf{p}, \mathbf{q}) \leq \varepsilon\} \quad (3.31)$$

gdzie D to zbiór wszystkich punktów, a $d(\mathbf{p}, \mathbf{q})$ to odległość między punktami \mathbf{p} i \mathbf{q} .

Punkt rdzeniowy (core point). Punkt \mathbf{p} jest punktem rdzeniowym, jeśli liczba punktów w jego sąsiedztwie ε wynosi co najmniej minPts (Ester et al., 1996):

$$|N_\varepsilon(\mathbf{p})| \geq \text{minPts} \quad (3.32)$$

Bezpośrednia osiągalność gęstościowa. Punkt \mathbf{q} jest bezpośrednio osiągalny gęstościowo z punktu \mathbf{p} , jeśli \mathbf{p} jest punktem rdzeniowym i $\mathbf{q} \in N_\varepsilon(\mathbf{p})$ (Ester et al., 1996).

Osiągalność gęstościowa. Punkt \mathbf{q} jest osiągalny gęstościowo z punktu \mathbf{p} , jeśli istnieje łańcuch punktów $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n$, gdzie $\mathbf{p}_1 = \mathbf{p}$ i $\mathbf{p}_n = \mathbf{q}$, taki że każdy kolejny punkt jest bezpośrednio osiągalny gęstościowo z poprzedniego (Ester et al., 1996).

Klasyfikacja punktów w DBSCAN. Algorytm DBSCAN klasyfikuje każdy punkt w zbiorze danych do jednej z trzech kategorii (Ester et al., 1996):

Punkt rdzeniowy (core point): punkt \mathbf{p} jest rdzeniowy, jeśli ma co najmniej minPts sąsiadów w promieniu ε , tj. $|N_\varepsilon(\mathbf{p})| \geq \text{minPts}$. Punkty rdzeniowe stanowią centrum klastrów i inicjują ich tworzenie.

Punkt brzegowy (border point): punkt należący do klastra, ale niebędący punktem rdzeniowym. Punkt brzegowy leży w sąsiedztwie ε co najmniej jednego punktu rdzeniowego, ale sam ma mniej niż minPts sąsiadów. Punkty brzegowe znajdują się na peryferiach klastrów i są przypisywane do klastra poprzez bezpośrednią osiągalność gęstościową z punktu rdzeniowego.

Punkt szumowy (noise point): punkt, który nie jest ani rdzeniowy, ani brzegowy. Punkt szumowy nie leży w sąsiedztwie ε żadnego punktu rdzeniowego i nie należy do żadnego klastra. Punkty szumowe reprezentują obserwacje odstające (outliers) lub szum w danych.

Dzięki tej klasyfikacji DBSCAN automatycznie identyfikuje i odrzuca punkty odstające, co czyni go odpornym na szum w danych (Ester et al., 1996).

3.2.4 PCA (Principal Component Analysis)

Analiza głównych składowych (PCA) to technika redukcji wymiarowości, która przekształca oryginalne cechy danych w nowy zestaw nieskorelowanych zmiennych zwanych głównymi składowymi (Murphy, 2012; Hastie et al., 2009). Główne składowe są liniowymi kombinacjami oryginalnych cech i są uporządkowane według wariancji, którą wyjaśniają w danych. Pierwsza główna składowa wyjaśnia największą część wariancji, druga główna składowa wyjaśnia drugą co do wielkości część wariancji, i tak dalej. PCA jest szeroko stosowana do wizualizacji danych, kompresji danych oraz usuwania szumów (Murphy, 2012; Hastie et al., 2009).

Macierz kowariancji. Algorytm PCA rozpoczyna się od wycentrowania danych (odejęcia średniej od każdej cechy) i obliczenia macierzy kowariancji, która opisuje zależności między cechami (Bishop, 2006; Hastie et al., 2009):

$$\Sigma = \frac{1}{n} X^\top X \quad (3.33)$$

gdzie X to macierz danych o wymiarach $n \times p$ (po wycentrowaniu), gdzie n to liczba próbek, a p to liczba cech. Macierz Σ ma wymiary $p \times p$.

Rozkład własny (Eigendecomposition). Kolejnym krokiem jest znalezienie wektorów własnych i wartości własnych macierzy kowariancji poprzez rozwiążanie równania (Bishop, 2006; Hastie et al., 2009):

$$\Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i \quad (3.34)$$

gdzie \mathbf{v}_i to i -ty wektor własny określający kierunek i -tej głównej składowej, a λ_i to odpowiadająca mu wartość własna reprezentująca wariancję wyjaśnianą przez tę składową. Wektory własne są ortogonalne, co zapewnia nieskorelowanie głównych składowych (Hastie et al., 2009).

Transformacja danych. Po uporządkowaniu wektorów własnych według malejących wartości własnych ($\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$), dane są transformowane do nowej przestrzeni głównych składowych (Hastie et al., 2009):

$$Z = XW \quad (3.35)$$

gdzie W to macierz o wymiarach $p \times k$ zawierająca k pierwszych wektorów własnych jako kolumny, a Z to macierz danych w nowej przestrzeni o wymiarach $n \times k$. Wybór liczby k składowych do zachowania zależy od wymaganej ilości wyjaśnianej wariancji.

Wariancja wyjaśniona. Proporcja wariancji wyjaśnianej przez i -tą główną składową wynosi (Hastie et al., 2009):

$$\text{Var}_i = \frac{\lambda_i}{\sum_{j=1}^p \lambda_j} \quad (3.36)$$

Suma wariancji wyjaśnianych przez pierwsze k składowych określa, jaki procent całkowitej zmienności danych został zachowany po redukcji wymiarowości. W praktyce często wybiera się k takie, aby zachować 90-95% wariancji (Bishop, 2006; Hastie et al., 2009).

3.2.5 t-SNE (t-distributed Stochastic Neighbor Embedding)

t-SNE to technika redukcji wymiarowości, która przekształca dane wysokowymiarowe w przestrzeń niskowymiarową (zazwyczaj 2D lub 3D) w taki sposób, aby zachować lokalne struktury danych (van der Maaten and Hinton, 2008). Algorytm t-SNE modeluje podobieństwa między punktami w oryginalnej przestrzeni jako prawdopodobieństwa warunkowe, a następnie optymalizuje rozmieszczenie punktów w przestrzeni niskowymiarowej, minimalizując różnicę między tymi prawdopodobieństwami za pomocą dywergencji Kullbacka-Leiblera (van der Maaten and Hinton, 2008).

Podobieństwa w przestrzeni wysokowymiarowej. W przestrzeni wysokowymiarowej podobieństwo między punktami \mathbf{x}_i i \mathbf{x}_j jest modelowane jako prawdopodobieństwo warunkowe (van der Maaten and Hinton, 2008):

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)} \quad (3.37)$$

gdzie σ_i to szerokość jądra Gaussa dla punktu \mathbf{x}_i . Prawdopodobieństwo symetryczne definiuje się jako:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \quad (3.38)$$

gdzie n to liczba punktów danych.

Podobieństwa w przestrzeni niskowymiarowej. W przestrzeni niskowymiarowej podobieństwo między punktami \mathbf{y}_i i \mathbf{y}_j jest modelowane za pomocą rozkładu t-Studenta z jednym stopniem swobody ([van der Maaten and Hinton, 2008](#)):

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}} \quad (3.39)$$

Optymalizacja. Algorytm t-SNE minimalizuje dywergencję Kullbacka-Leiblera między rozkładami P i Q ([van der Maaten and Hinton, 2008](#)):

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (3.40)$$

Optymalizacja jest przeprowadzana za pomocą gradientu prostego lub metod opartych na momencie, co pozwala na znalezienie układu punktów \mathbf{y}_i w przestrzeni niskowymiarowej, który najlepiej zachowuje lokalne struktury danych z przestrzeni wysokowymiarowej ([van der Maaten and Hinton, 2008](#)).

Parametry t-SNE. Algorytm t-SNE posiada kilka kluczowych parametrów, które wpływają na jakość i charakter wynikowej wizualizacji ([van der Maaten and Hinton, 2008](#)):

- **Perplexity:** Określa liczbę najbliższych sąsiadów branych pod uwagę przy obliczaniu podobieństw w przestrzeni wysokowymiarowej. Typowe wartości to od 5 do 50. Wyższe wartości prowadzą do bardziej globalnych struktur, podczas gdy niższe wartości skupiają się na lokalnych strukturach.
- **Liczba iteracji:** Określa, ile razy algorytm będzie aktualizował położenia punktów w przestrzeni niskowymiarowej. Większa liczba iteracji może prowadzić do lepszej konwergencji, ale zwiększa czas obliczeń.
- **Współczynnik uczenia (learning rate):** Kontroluje szybkość aktualizacji położen punktów podczas optymalizacji. Zbyt wysoki współczynnik może prowadzić do niestabilności, podczas gdy zbyt niski może spowolnić zbieżność.

Dobranie powyższych parametrów jest głównym czynnikiem wpływającym na czytelność i jakość wizualizacji ([van der Maaten and Hinton, 2008](#)).

3.2.6 UMAP (Uniform Manifold Approximation and Projection)

UMAP to technika redukcji wymiarowości i wizualizacji danych, która opiera się na teorii topologii i geometrii różniczkowej (McInnes et al., 2018). UMAP tworzy niskowymiarową reprezentację danych wysokowymiarowych, zachowując zarówno lokalne, jak i globalne struktury danych. Algorytm UMAP składa się z dwóch głównych etapów: konstrukcji grafu sąsiedztwa w przestrzeni wysokowymiarowej oraz optymalizacji rozmieszczenia punktów w przestrzeni niskowymiarowej (McInnes et al., 2018).

Konstrukcja grafu sąsiedztwa. W pierwszym etapie UMAP buduje graf sąsiedztwa, w którym każdy punkt danych jest połączony z jego k najbliższymi sąsiadami (McInnes et al., 2018). Podobieństwo między punktami \mathbf{x}_i i \mathbf{x}_j jest modelowane za pomocą funkcji ważonej:

$$w_{ij} = \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j) - \rho_i}{\sigma_i}\right) \quad (3.41)$$

gdzie $d(\mathbf{x}_i, \mathbf{x}_j)$ to odległość między punktami, ρ_i to odległość do najbliższego sąsiada punktu \mathbf{x}_i , a σ_i to skalowanie lokalne kontrolujące gęstość sąsiedztwa.

Optymalizacja w przestrzeni niskowymiarowej. W drugim etapie UMAP optymalizuje rozmieszczenie punktów \mathbf{y}_i w przestrzeni niskowymiarowej, minimalizując funkcję koszta opartą na różnicę między grafem sąsiedztwa w przestrzeni wysokowymiarowej a grafem w przestrzeni niskowymiarowej (McInnes et al., 2018):

$$C = \sum_{i \neq j} \left(w_{ij} \log \frac{w_{ij}}{q_{ij}} + (1 - w_{ij}) \log \frac{1 - w_{ij}}{1 - q_{ij}} \right) \quad (3.42)$$

gdzie q_{ij} to podobieństwo między punktami \mathbf{y}_i i \mathbf{y}_j w przestrzeni niskowymiarowej, modelowane za pomocą funkcji:

$$q_{ij} = \frac{1}{1 + a \|\mathbf{y}_i - \mathbf{y}_j\|^{2b}} \quad (3.43)$$

gdzie a i b to parametry kontrolujące kształt funkcji podobieństwa. Optymalizacja jest przeprowadzana za pomocą metod gradientu prostego lub jego wariantów, co pozwala na znalezienie układu punktów \mathbf{y}_i w przestrzeni niskowymiarowej, który najlepiej zachowuje struktury danych z przestrzeni wysokowymiarowej (McInnes et al., 2018). **Parametry**

UMAP. Algorytm UMAP posiada kilka kluczowych parametrów, które wpływają na jakość i charakter wynikowej wizualizacji (McInnes et al., 2018):

- **Liczba sąsiadów (n_neighbors):** Określa liczbę najbliższych sąsiadów branych pod uwagę przy budowie grafu sąsiedztwa. Typowe wartości to od 5 do 50. Wyższe wartości prowadzą do bardziej globalnych struktur, podczas gdy niższe wartości skupiają się na lokalnych strukturach.
- **Minimalna odległość (min_dist):** Kontroluje, jak blisko punkty mogą być rozmieszczone w przestrzeni niskowymiarowej. Niższe wartości prowadzą do bardziej skondensowanych klastrów, podczas gdy wyższe wartości rozpraszają punkty bardziej równomiernie.

- **Liczba wymiarów docelowych (n_components):** Określa liczbę wymiarów w przestrzeni niskowymiarowej (zazwyczaj 2 lub 3).
- **Liczba iteracji (n_epochs):** Określa, ile razy algorytm będzie aktualizował położenia punktów w przestrzeni niskowymiarowej. Większa liczba iteracji może prowadzić do lepszej konwergencji, ale zwiększa czas obliczeń.

Dopasowanie parametrów takich jak liczba sąsiadów (n_neighbors), minimalna odległość (min_dist), liczba wymiarów docelowych i epochs przesąduje o równowadze między zachowaniem lokalnych struktur danych a globalnej topologii w wizualizacjach UMAP ([McInnes et al., 2018](#)).

3.3 Uczenie półnadzorowane (Semi-Supervised Learning)

Uczenie półnadzorowane to podejście w uczeniu maszynowym, które łączy cechy uczenia nadzorowanego i nienadzorowanego, wykorzystując zarówno dane oznaczone etykietami, jak i dane bez etykiet do trenowania modelu. Ten rodzaj uczenia wykorzystuje niewielki zbiór danych oznaczonych do nauki podstawowej struktury problemu. W kolejnym etapie wykorzystuje duży zbiór danych nieoznaczonych do poprawy uogólnienia modelu i lepszego zrozumienia rozkładu danych. Skuteczność tych metod opiera się na założeniu, że struktura danych nieoznaczonych dostarcza informacji o warunkowym rozkładzie etykiet ([Chapelle et al., 2006; Zhu and Goldberg, 2009](#)).

Założenia uczenia półnadzorowanego. Skuteczność uczenia półnadzorowanego opiera się na kilku kluczowych założeniach dotyczących struktury danych ([Chapelle et al., 2006; Zhu and Goldberg, 2009](#)):

Założenie płynności (Smoothness Assumption): Jeśli dwa punkty \mathbf{x}_1 i \mathbf{x}_2 w przestrzeni cech są blisko siebie, to ich odpowiadające etykiety y_1 i y_2 powinny być podobne. Oznacza to, że funkcja decyzyjna powinna być płynna w obszarach o wysokiej gęstości danych.

Założenie klastrów (Cluster Assumption): Dane mają tendencję do tworzenia odrębnych klastrów, a punkty w tym samym klastrze prawdopodobnie należą do tej samej klasy. Granice decyzyjne powinny przebiegać przez obszary o niskiej gęstości danych, dzieląc różne klastry.

Założenie rozmaitości (Manifold Assumption): Dane wysokowymiarowe leżą w przybliżeniu na niskowymiarowej rozmaitości. Punkty, które są bliskie na tej rozmaistości, powinny mieć podobne etykiety, nawet jeśli są daleko od siebie w oryginalnej przestrzeni wysokowymiarowej.

3.3.1 Self-Training (Samouczenie)

Self-training to jedna z najprostszych metod uczenia półnadzorowanego, która działa w sposób iteracyjny, wykorzystując własne predykcje modelu do etykietowania danych nieoznaczonych. Algorytm rozpoczyna od wytrenowania modelu na małym zbiorze danych

oznaczonych, a następnie używa tego modelu do predykcji etykiet dla danych nieoznaczonych. Dane nieoznaczone, dla których model jest najbardziej pewny swoich predykcji, są dodawane do zbioru treningowego wraz z przewidywanymi etykietami. Model jest następnie ponownie trenowany na powiększonym zbiorze danych, a proces powtarza się iteracyjnie, aż do wyczerpania danych nieoznaczonych lub osiągnięcia kryterium stopu (Zhu and Goldberg, 2009).

Algorytm self-training rozpoczyna się od wytrenowania klasyfikatora na małym zbiorze oznaczonych danych. Następnie klasyfikator jest wykorzystywany do przewidywania etykiet dla danych nieoznaczonych. Dla każdej predykcji obliczany jest poziom pewności, który określa, jak pewny jest model swojej decyzji. Wybierany jest podzbiór predykcji, dla których poziom pewności przekracza ustalony próg, i predykcje te wraz z przewidywanymi etykietami są dodawane do zbioru treningowego. Klasyfikator jest następnie trenowany ponownie na powiększonym zbiorze danych. Proces powtarza się iteracyjnie — kroki przewidywania, wyboru pewnych predykcji i retrenowania — aż do osiągnięcia kryterium stopu, którym może być wyczerpanie danych nieoznaczonych, brak nowych pewnych predykcji lub osiągnięcie maksymalnej liczby iteracji (Zhu and Goldberg, 2009).

Głównym ryzykiem self-training jest kumulacja błędów — jeśli model popełni błąd w przewidywaniu etykiety i dodaje ją do zbioru treningowego, błędne etykiety mogą propagować się i wzmacniać w kolejnych iteracjach, prowadząc do degradacji wydajności (Zhu and Goldberg, 2009).

3.3.2 Co-Training

Co-training to metoda uczenia półnadzorowanego zaproponowana dla sytuacji, gdy dane można naturalnie opisać za pomocą dwóch różnych, niezależnych zestawów cech. Każdy zestaw cech powinien być wystarczający do trenowania dobrego klasyfikatora, a widoki powinny być warunkowo niezależne przy danej etykiecie klasy. Dwa klasyfikatory są trenowane oddzielnie na różnych zestawach cech, a następnie każdy klasyfikator etykietuje dane nieoznaczone dla drugiego klasyfikatora, wykorzystując najbardziej pewne predykcje. Ta wzajemna nauka pozwala klasyfikatorom uczyć się od siebie nawzajem, wykorzystując komplementarną informację z różnych perspektyw danych (Zhu and Goldberg, 2009).

Algorytm co-training (Zhu and Goldberg, 2009): Co-training rozpoczyna się od podziału cech na dwa niezależne widoki $\mathbf{x}^{(1)}$ i $\mathbf{x}^{(2)}$. Następnie trenowane są dwa klasyfikatory f_1 i f_2 na małym zbiorze danych oznaczonych, każdy wykorzystując tylko swój widok danych. W kolejnym kroku każdy klasyfikator f_i przewiduje etykiety dla wszystkich danych nieoznaczonych. Klasyfikator f_1 wybiera n najbardziej pewnych pozytywnych i n najbardziej pewnych negatywnych przykładów i dodaje je do zbioru treningowego dla f_2 , podczas gdy klasyfikator f_2 wykonuje analogiczną operację, dodając swoje najbardziej pewne predykcje do zbioru treningowego dla f_1 . Oba klasyfikatory są następnie trenowane ponownie na powiększonych zbiorach danych. Proces powtarza się przez ustaloną liczbę iteracji lub do momentu wyczerpania danych nieoznaczonych (Zhu and Goldberg, 2009).

3.3.3 Transductive SVM (TSVM)

Transductive Support Vector Machine to rozszerzenie standardowego SVM na uczenie półnadzorowane, które maksymalizuje margines zarówno dla danych oznaczonych, jak i nieoznaczonych. TSVM poszukuje hiperpłaszczyzny, która nie tylko poprawnie klasyfikuje dane oznaczone z maksymalnym marginesem, ale również dzieli dane nieoznaczone tak, aby granica decyzyjna przechodziła przez obszary o niskiej gęstości punktów. Algorytm realizuje założenie klastrów (cluster assumption), dając do sytuacji, w której jak najmniej danych nieoznaczonych znajduje się blisko granicy decyzyjnej ([Chapelle et al., 2006](#)).

Funkcja celu TSVM. Problem optymalizacji dla TSVM można zapisać jako ([Chapelle et al., 2006](#)):

$$\min_{\mathbf{w}, b, \xi, \xi^*, \mathbf{y}^*} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i + C^* \sum_{j=1}^u \xi_j^* \quad (3.44)$$

przy ograniczeniach dla danych oznaczonych:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, l \quad (3.45)$$

i dla danych nieoznaczonych z przewidywanymi etykietami $y_j^* \in \{-1, +1\}$:

$$y_j^*(\mathbf{w}^\top \mathbf{x}_j + b) \geq 1 - \xi_j^*, \quad \xi_j^* \geq 0, \quad j = 1, \dots, u \quad (3.46)$$

gdzie C i C^* to parametry regularizacji dla danych oznaczonych i nieoznaczonych odpowiednio.

Problem jest trudny obliczeniowo (NP-trudny), ponieważ wymaga optymalizacji również po etykietach danych nieoznaczonych y_j^* . W praktyce stosuje się heurystyki i przybliżone algorytmy optymalizacji ([Chapelle et al., 2006](#)).

3.3.4 Graph-Based Methods (Metody grafowe)

Metody grafowe w uczeniu półnadzorowanym konstruują graf, w którym węzły reprezentują wszystkie punkty danych (zarówno oznaczone, jak i nieoznaczone), a krawędzie reprezentują podobieństwo między nimi. Założenie jest takie, że etykiety powinny zmieniać się płynnie wzduż grafu — punkty połączone silnymi krawędziami (wysokie podobieństwo) powinny mieć podobne etykiety. Etykiety są propagowane od węzłów oznaczonych do nieoznaczonych zgodnie z tą zasadą, wykorzystując strukturę grafu do wnioskowania o etykietach ([Zhu and Goldberg, 2009](#)).

Konstrukcja grafu. Najpopularniejsze metody konstrukcji grafu to ([Zhu and Goldberg, 2009](#)):

- **k-nearest neighbor graph** — każdy punkt jest połączony z k najbliższymi sąsiadami.
- **ε -neighborhood graph** — punkty są połączone, jeśli odległość między nimi jest mniejsza niż ε .

- **Fully connected graph** — wszystkie punkty są połączone, a wagi krawędzi maleją z odległością.

Macierz podobieństwa (wag krawędzi) W często definiuje się za pomocą jądra Gaussa ([Zhu and Goldberg, 2009](#)):

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \quad (3.47)$$

gdzie σ kontroluje szerokość jądra.

Label Propagation. Algorytm label propagation propaguje etykiety przez graf iteracyjnie. Niech Y będzie macierzą etykiet rozmiaru $n \times c$, gdzie n to liczba punktów, a c to liczba klas. Dla danych oznaczonych $Y_{il} = 1$ jeśli punkt i należy do klasy l , w przeciwnym razie $Y_{il} = 0$.

Macierz przejścia P definiuje się jako znormalizowaną macierz wag ([Zhu and Goldberg, 2009](#)):

$$P_{ij} = \frac{W_{ij}}{\sum_{k=1}^n W_{ik}} \quad (3.48)$$

Algorytm iteracyjnie aktualizuje etykiety według ([Zhu and Goldberg, 2009](#)):

$$Y^{(t+1)} = PY^{(t)} \quad (3.49)$$

gdzie po każdej iteracji etykiety danych oznaczonych są resetowane do oryginalnych wartości, aby zachować pewność co do prawdziwych etykiet.

Label Spreading. Label spreading to wariant label propagation, który pozwala na miękką zmianę również etykiet danych oznaczonych (z mniejszą wagą). Algorytm minimalizuje funkcję kosztu ([Zhu and Goldberg, 2009](#)):

$$E(Y) = \frac{1}{2} \sum_{i,j=1}^n W_{ij} \left\| \frac{Y_i}{\sqrt{D_{ii}}} - \frac{Y_j}{\sqrt{D_{jj}}} \right\|^2 + \mu \sum_{i=1}^l \|Y_i - Y_i^0\|^2 \quad (3.50)$$

gdzie D to macierz diagonalna stopni węzłów ($D_{ii} = \sum_j W_{ij}$), Y_i^0 to oryginalne etykiety danych oznaczonych, a μ kontroluje siłę przywiązywania do oryginalnych etykiet. Rozwiązańie w stanie równowagi można znaleźć analitycznie.

3.3.5 Generative Models (Modele generatywne)

Modele generatywne w uczeniu półnadzorowanym modelują łączny rozkład prawdopodobieństwa $P(\mathbf{x}, y)$ dla cech i etykiet. Kluczowa idea polega na tym, że dane nieoznaczone pomagają w lepszej estymacji rozkładu marginalnego $P(\mathbf{x})$, co z kolei poprawia estymację rozkładu warunkowego $P(y|\mathbf{x})$ poprzez współdzielenie parametrów. Najczęściej stosuje się mieszaniny rozkładów Gaussa (Gaussian Mixture Models), gdzie zakłada się, że każda klasa jest generowana z mieszaniny komponentów gaussowskich ([Chapelle et al., 2006; Murphy, 2012](#)).

Semi-Supervised GMM. Model zakłada, że dane są generowane z mieszaniny K rozkładów Gaussa ([Murphy, 2012](#)):

$$P(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k) \quad (3.51)$$

gdzie π_k to wagi komponentów ($\sum_k \pi_k = 1$), $\boldsymbol{\mu}_k$ to średnie, a Σ_k to macierze kowariancji.

Dla danych oznaczonych log-likelihood wynosi ([Chapelle et al., 2006](#)):

$$\log L_{\text{labeled}} = \sum_{i=1}^l \log P(y_i, \mathbf{x}_i) = \sum_{i=1}^l \log \sum_{k \in C_{y_i}} \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \Sigma_k) \quad (3.52)$$

gdzie C_{y_i} to zbiór komponentów odpowiadających klasie y_i .

Dla danych nieoznaczonych ([Chapelle et al., 2006](#)):

$$\log L_{\text{unlabeled}} = \sum_{j=1}^u \log P(\mathbf{x}_j) = \sum_{j=1}^u \log \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_j | \boldsymbol{\mu}_k, \Sigma_k) \quad (3.53)$$

Całkowita log-likelihood:

$$\log L = \log L_{\text{labeled}} + \log L_{\text{unlabeled}} \quad (3.54)$$

Parametry są estymowane za pomocą algorytmu EM (Expectation-Maximization), który w kroku E oblicza prawdopodobieństwa przynależności punktów do komponentów, a w kroku M aktualizuje parametry modelu na podstawie tych prawdopodobieństw ([Murphy, 2012](#)).

3.3.6 Entropy Minimization (Minimalizacja entropii)

Minimalizacja entropii to podejście oparte na zasadzie, że dobry model powinien być pewny swoich predykcji również dla danych nieoznaczonych. Entropia rozkładu predykcji mierzy niepewność modelu — niska entropia oznacza pewne predykcje (rozkład skoncentrowany na jednej klasie), wysoka entropia oznacza niepewność (rozkład rozłożony równomiernie). Metoda dodaje do funkcji straty składnik karny, który minimalizuje entropię predykcji dla danych nieoznaczonych, zmuszając model do dokonywania pewnych klasyfikacji ([Chapelle et al., 2006](#)).

Funkcja straty. Całkowita funkcja straty składa się z nadzorowanego składnika dla danych oznaczonych i składnika minimalizacji entropii dla danych nieoznaczonych ([Chapelle et al., 2006](#)):

$$L = L_{\text{supervised}} + \lambda L_{\text{entropy}} \quad (3.55)$$

gdzie:

$$L_{\text{supervised}} = \frac{1}{l} \sum_{i=1}^l \mathcal{L}(f(\mathbf{x}_i), y_i) \quad (3.56)$$

to standardowa strata klasyfikacji dla danych oznaczonych, a:

$$L_{\text{entropy}} = -\frac{1}{u} \sum_{j=1}^u \sum_{k=1}^c p_k^{(j)} \log p_k^{(j)} \quad (3.57)$$

to entropia predykcji dla danych nieoznaczonych, gdzie $p_k^{(j)} = P(y = k | \mathbf{x}_j)$ to przewidywane prawdopodobieństwo klasy k dla punktu j . Parametr λ kontroluje wagę składnika entropijnego.

Minimalizacja entropii jest szczególnie efektywna w połączeniu z innymi metodami półnadzorowanymi i jest często używana jako regularizator w głębokich sieciach neuronowych (Chapelle et al., 2006).

Zastosowania uczenia półnadzorowanego. Uczenie półnadzorowane znajduje zastosowanie w wielu dziedzinach, gdzie etykietowanie danych jest kosztowne (Chapelle et al., 2006; Zhu and Goldberg, 2009; Hastie et al., 2009):

- **Klasyfikacja tekstów i dokumentów** — gdzie ręczne oznaczanie dokumentów jest czasochłonne, ale dostępne są ogromne zbiorы tekstów nieoznaczonych.
- **Rozpoznawanie obrazów i wizja komputerowa** — wykorzystanie milionów zdjęć bez etykiet do poprawy klasyfikatorów wytrenowanych na małych zbiorach oznaczonych.
- **Bioinformatyka i medycyna** — analiza sekwencji DNA/RNA lub obrazów medycznych, gdzie eksperymentalne potwierdzenie etykiet jest bardzo drogie.
- **Kontrola jakości w produkcji** — w procesach wytwórczych, gdzie dostępne są ogromne ilości danych z czujników (nieoznaczonych), ale tylko nieliczne przypadki defektów są zidentyfikowane i oznaczone.
- **Systemy rekommendacyjne** — gdzie mamy dużo danych o interakcjach użytkowników, ale mało jawnych ocen.

Uczenie półnadzorowane jest szczególnie wartościowe w rzeczywistych zastosowaniach przemysłowych, gdzie koszt eksperckich etykiet jest wysoki, a dane nieoznaczone są generowane automatycznie przez systemy monitorujące i czujniki (Chapelle et al., 2006; Zhu and Goldberg, 2009).

3.4 Uczenie ze wzmacnieniem (Reinforcement Learning)

Uczenie ze wzmacnieniem to paradygmat uczenia maszynowego, w którym agent uczy się podejmować decyzje poprzez interakcję ze środowiskiem w celu maksymalizacji skumulowanej nagrody. W przeciwnieństwie do uczenia nadzorowanego, gdzie model uczy się na podstawie par wejście-wyjście, w uczeniu ze wzmacnieniem agent otrzymuje jedynie sygnały zwrotne w postaci nagród lub kar za wykonane akcje. Agent musi samodzielnie odkryć, które akcje prowadzą do największych nagród, ucząc się poprzez eksperymentowanie i doświadczenie. Uczenie ze wzmacnieniem jest szczególnie efektywne w problemach

sekwenckijnego podejmowania decyzji, gdzie długoterminowe konsekwencje akcji są ważniejsze niż natychmiastowe nagrody ([Sutton and Barto, 2018](#)).

Podstawowe elementy uczenia ze wzmacnieniem. System uczenia ze wzmacnieniem składa się z kilku kluczowych elementów ([Sutton and Barto, 2018](#)):

Agent: System podejmujący decyzje, który uczy się optymalnej strategii działania poprzez interakcję ze środowiskiem.

Środowisko (Environment): Wszystko to, z czym agent wchodzi w interakcję i co znajduje się poza jego bezpośrednią kontrolą. Środowisko odbiera akcje agenta i zwraca nowe stany oraz nagrody.

Stan (State): Reprezentacja bieżącej sytuacji w środowisku, oznaczana jako $s \in \mathcal{S}$, gdzie \mathcal{S} to przestrzeń wszystkich możliwych stanów.

Akcja (Action): Decyzja podejmowana przez agenta w danym stanie, oznaczana jako $a \in \mathcal{A}$, gdzie \mathcal{A} to przestrzeń wszystkich możliwych akcji.

Nagroda (Reward): Sygnał numeryczny $r \in \mathbb{R}$ otrzymywany przez agenta od środowiska po wykonaniu akcji, wskazujący na natychmiastową wartość tej akcji.

Polityka (Policy): Strategia agenta określająca, jaką akcję wybrać w danym stanie, oznaczana jako $\pi(a|s)$ dla polityki stochastycznej lub $\pi(s)$ dla polityki deterministycznej.

Proces decyzyjny Markowa (Markov Decision Process). Formalne podstawy uczenia ze wzmacnieniem opierają się na procesie decyzyjnym Markowa (MDP), który definiuje się jako krotkę $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ ([Sutton and Barto, 2018](#)), gdzie:

- \mathcal{S} — zbiór stanów,
- \mathcal{A} — zbiór akcji,
- $P(s'|s, a)$ — prawdopodobieństwo przejścia do stanu s' po wykonaniu akcji a w stanie s ,
- $R(s, a, s')$ — funkcja nagrody otrzymana za przejście ze stanu s do s' poprzez akcję a ,
- $\gamma \in [0, 1]$ — współczynnik dyskontowania określający wartość przyszłych nagród względem natychmiastowych.

MDP zakłada własność Markowa, która stwierdza, że przyszłość zależy tylko od obecnego stanu i akcji, a nie od całej historii ([Sutton and Barto, 2018](#)):

$$P(s_{t+1}, r_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P(s_{t+1}, r_{t+1} | s_t, a_t) \quad (3.58)$$

Funkcja wartości. Celem agenta jest maksymalizacja oczekiwanej skumulowanej zwrotu (return), zdefiniowanego jako zdyskontowana suma przyszłych nagród ([Sutton and](#)

Barto, 2018):

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (3.59)$$

gdzie γ kontroluje, jak bardzo agent ceni przyszłe nagrody — dla $\gamma = 0$ agent jest myopic (krótkowzroczny), a dla $\gamma \rightarrow 1$ agent w pełni uwzględnia przyszłe konsekwencje.

Funkcja wartości stanu (state-value function) dla polityki π definiuje się jako oczekiwany zwrot startując ze stanu s i następując polityką π (Sutton and Barto, 2018):

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right] \quad (3.60)$$

Funkcja wartości akcji (action-value function lub Q-function) określa oczekiwany zwrot przy rozpoczęciu w stanie s , wykonaniu akcji a i następowaniu polityki π (Sutton and Barto, 2018):

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right] \quad (3.61)$$

Równanie Bellmana. Funkcje wartości spełniają rekurencyjne równanie Bellmana, które wyraża wartość stanu poprzez natychmiastową nagrodę i zdyskontowaną wartość następnego stanu (Sutton and Barto, 2018):

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a) [R(s, a, s') + \gamma V^\pi(s')] \quad (3.62)$$

Dla funkcji Q równanie Bellmana przyjmuje postać (Sutton and Barto, 2018):

$$Q^\pi(s, a) = \sum_{s' \in \mathcal{S}} P(s'|s, a) \left[R(s, a, s') + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^\pi(s', a') \right] \quad (3.63)$$

Optymalna funkcja wartości $V^*(s) = \max_\pi V^\pi(s)$ spełnia równanie optymalności Bellmana (Sutton and Barto, 2018):

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a) [R(s, a, s') + \gamma V^*(s')] \quad (3.64)$$

3.4.1 Q-Learning

Q-Learning to algorytm uczenia ze wzmocnieniem bez modelu (model-free), który uczy się optymalnej funkcji wartości akcji $Q^*(s, a)$ bez potrzeby znajomości dynamiki środowiska $P(s'|s, a)$. Algorytm iteracyjnie aktualizuje oszacowania wartości Q na podstawie doświadczeń zebranych przez agenta w środowisku, wykorzystując zasadę temporal difference (TD) learning. Q-Learning jest algorymem off-policy, co oznacza, że może uczyć się optymalnej polityki nawet gdy agent podąża za inną, eksploracyjną polityką (Sutton and Barto, 2018).

Reguła aktualizacji Q-Learning. Po wykonaniu akcji a_t w stanie s_t , otrzymaniu nagrody r_{t+1} i przejściu do stanu s_{t+1} , wartość $Q(s_t, a_t)$ jest aktualizowana zgodnie z regułą (Sutton and Barto, 2018):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \quad (3.65)$$

gdzie $\alpha \in (0, 1]$ to współczynnik uczenia kontrolujący szybkość aktualizacji, a $\max_{a'} Q(s_{t+1}, a')$ reprezentuje oszacowanie wartości najlepszej akcji w następnym stanie.

Wyrażenie w nawiasie kwadratowym nazywane jest błędem TD (temporal difference error):

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \quad (3.66)$$

który mierzy różnicę między obecnym oszacowaniem a celem bootstrappowanym z następnego stanu.

Wykazano, że Q-Learning zbiega do optymalnej funkcji wartości akcji Q^* pod warunkiem, że każda para stan-akcja jest odwiedzana nieskończenie wiele razy i współczynnik uczenia spełnia odpowiednie warunki (Sutton and Barto, 2018).

3.4.2 SARSA (State-Action-Reward-State-Action)

SARSA to algorytm uczenia ze wzmacnieniem bez modelu, który w przeciwieństwie do Q-Learning jest algorytmem on-policy — uczy się wartości polityki, którą faktycznie wykonuje. Nazwa algorytmu pochodzi od sekwencji elementów używanych do aktualizacji: $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. SARSA jest bardziej konserwatywny niż Q-Learning, ponieważ uwzględnia rzeczywiste akcje podejmowane przez agenta, a nie zakłada zawsze wybór optymalnej akcji (Sutton and Barto, 2018).

Reguła aktualizacji SARSA. Po wykonaniu akcji a_t w stanie s_t , otrzymaniu nagrody r_{t+1} , przejściu do stanu s_{t+1} i wybraniu następnej akcji a_{t+1} zgodnie z polityką, wartość $Q(s_t, a_t)$ jest aktualizowana według reguły (Sutton and Barto, 2018):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (3.67)$$

Kluczowa różnica w porównaniu do Q-Learning polega na tym, że zamiast $\max_{a'} Q(s_{t+1}, a')$ używane jest $Q(s_{t+1}, a_{t+1})$, gdzie a_{t+1} to akcja rzeczywiście wybrana przez agenta zgodnie z jego polityką (np. ε -greedy).

SARSA jest bardziej odpowiedni w środowiskach, gdzie eksploracja nieoptimalnych akcji jest ryzykowna, ponieważ uwzględnia konsekwencje faktyczne podejmowanych akcji eksploracyjnych (Sutton and Barto, 2018).

3.4.3 Deep Q-Network (DQN)

Deep Q-Network to przełomowe rozszerzenie Q-Learning wykorzystujące głębokie sieci neuronowe do aproksymacji funkcji wartości akcji w problemach z dużymi lub ciągłymi przestrzeniami stanów. Klasyczny Q-Learning przechowuje wartości Q w tabeli, co staje się niepraktyczne dla złożonych problemów. DQN reprezentuje funkcję $Q(s, a)$ jako sieć

neuronową z parametrami θ , oznaczaną jako $Q(s, a; \theta)$, która może generalizować wiedzę między podobnymi stanami. Algorytm DQN wprowadza kilka innowacji technicznych niezbędnych do stabilnego trenowania głębokich sieci w kontekście uczenia ze wzmacnieniem (Sutton and Barto, 2018).

Experience Replay. DQN wykorzystuje bufor doświadczeń (experience replay buffer), w którym przechowywane są krotki przejść ($s_t, a_t, r_{t+1}, s_{t+1}$). Podczas treningu losowo próbkiowane są mini-batche z tego bufora, co łamie korelację między kolejnymi próbami i poprawia stabilność uczenia (Sutton and Barto, 2018).

Target Network. DQN wykorzystuje dwie sieci neuronowe: sieć główną $Q(s, a; \theta)$ i sieć docelową $Q(s, a; \theta^-)$. Sieć główna jest aktualizowana w każdym kroku, podczas gdy sieć docelowa jest aktualizowana rzadziej (co C kroków) kopując wagę z sieci głównej. To rozwiązanie stabilizuje cele uczenia, ponieważ cel y_t w funkcji straty jest obliczany za pomocą stałych (przez pewien czas) wag (Sutton and Barto, 2018).

Funkcja straty DQN. Parametry sieci θ są optymalizowane poprzez minimalizację funkcji straty (Sutton and Barto, 2018):

$$L(\theta) = \mathbb{E}_{(s, a, r, s') \sim \mathcal{D}} [(y - Q(s, a; \theta))^2] \quad (3.68)$$

gdzie cel y jest obliczany za pomocą sieci docelowej:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \quad (3.69)$$

a \mathcal{D} oznacza rozkład próbkiowany z bufora doświadczeń.

3.4.4 Policy Gradient Methods (Metody gradientu polityki)

Metody gradientu polityki to alternatywne podejście do uczenia ze wzmacnieniem, które zamiast uczyć się funkcji wartości, bezpośrednio optymalizują parametryzowaną politykę $\pi(a|s; \theta)$. Polityka może być reprezentowana jako sieć neuronowa, która dla danego stanu s produkuje rozkład prawdopodobieństwa nad akcjami. Metody te są szczególnie efektywne w problemach z ciągłymi lub bardzo dużymi przestrzeniami akcji, gdzie wybór maksimum z funkcji Q byłby trudny obliczeniowo (Sutton and Barto, 2018; Szepesvári, 2010).

Cel optymalizacji. Celem jest maksymalizacja oczekiwanej wartości funkcji wartości stanu początkowego (Sutton and Barto, 2018):

$$J(\theta) = \mathbb{E}_{s_0} [V^{\pi_\theta}(s_0)] \quad (3.70)$$

lub równoważnie oczekiwanej skumulowanej nagrody pod polityką π_θ .

Twierdzenie o gradiencie polityki. Gradient funkcji celu względem parametrów polityki można wyrazić jako (Sutton and Barto, 2018):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi(a|s; \theta) Q^{\pi_\theta}(s, a)] \quad (3.71)$$

gdzie oczekiwanie jest po trajektoriach generowanych przez politykę π_θ .

W praktyce $Q^{\pi_\theta}(s, a)$ jest aproksymowane przez skumulowaną nagrodę z trajektorii (REINFORCE) lub przez nauczoną funkcję wartości (Actor-Critic).

Algorytm REINFORCE. Podstawowy algorytm gradientu polityki, REINFORCE, używa pełnych zwrotów z trajektorii jako nieobciążonych estymatorów $Q^{\pi_\theta}(s, a)$ ([Sutton and Barto, 2018](#)):

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log \pi(a_t | s_t; \theta) \quad (3.72)$$

gdzie G_t to rzeczywisty zwrot otrzymany po akcji a_t .

Metody Actor-Critic. Metody Actor-Critic łączą podejście gradientu polityki (actor) z uczeniem funkcji wartości (critic). Actor aktualizuje parametry polityki θ w kierunku poprawy wydajności, podczas gdy critic uczy się funkcji wartości $V^\pi(s)$ lub $Q^\pi(s, a)$ z parametrami ω , która jest używana do oceny akcji aktora. Zamiast używać pełnych zwrotów G_t , actor-critic wykorzystuje bootstrap-owane oszacowania z critic, co zmniejsza wariancję gradientu ([Sutton and Barto, 2018](#); [Szepesvári, 2010](#)).

Aktualizacja critic (dla TD(0)) ([Sutton and Barto, 2018](#)):

$$\omega \leftarrow \omega + \beta \delta_t \nabla_\omega V(s_t; \omega) \quad (3.73)$$

gdzie błąd TD wynosi:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}; \omega) - V(s_t; \omega) \quad (3.74)$$

Aktualizacja actor ([Sutton and Barto, 2018](#)):

$$\theta \leftarrow \theta + \alpha \delta_t \nabla_\theta \log \pi(a_t | s_t; \theta) \quad (3.75)$$

Zastosowania uczenia ze wzmacnieniem. Uczenie ze wzmacnieniem znalazło szerokie zastosowanie w wielu dziedzinach ([Sutton and Barto, 2018](#); [Szepesvári, 2010](#)):

- **Robotyka i automatyka** — kontrola robotów, manipulacja obiektemi, nawigacja autonomiczna.
- **Gry** — osiągnięcie nadludzkiego poziomu w grach planszowych (Go, szachy) i grach wideo (Atari, StarCraft, Dota 2).
- **Optymalizacja procesów przemysłowych** — kontrola parametrów procesów wytwórczych, zarządzanie łańcuchem dostaw, optymalizacja zużycia energii.
- **Systemy rekomendacyjne** — personalizacja rekomendacji z uwzględnieniem długoterminowego zaangażowania użytkowników.
- **Finanse** — handel algorytmiczny, zarządzanie portfelem, optymalizacja strategii inwestycyjnych.
- **Healthcare** — personalizacja terapii, optymalizacja dawkowania leków, zarządzanie zasobami szpitalnymi.
- **Zarządzanie ruchem i logistyka** — optymalizacja sygnalizacji świetlnej, routing pojazdów autonomicznych.

W kontekście procesów wytwarzających, uczenie ze wzmacnieniem pozwala na dynamiczną optymalizację parametrów produkcji w czasie rzeczywistym, adaptację do zmieniających się warunków oraz minimalizację kosztów przy jednoczesnym utrzymaniu wysokiej jakości produktów (Sutton and Barto, 2018).

3.5 Uczenie głębokie (Deep Learning)

Uczenie głębokie to poddziedzina uczenia maszynowego oparta na sztucznych sieciach neuronowych z wieloma warstwami przetwarzającymi informację w sposób hierarchiczny. W przeciwieństwie do klasycznych algorytmów uczenia maszynowego, które wymagają ręcznego inżynierowania cech (feature engineering), głębokie sieci neuronowe automatycznie uczą się reprezentacji danych na wielu poziomach abstrakcji. Niższe warstwy uczą się prostych wzorców (np. krawędzie w obrazach), podczas gdy wyższe warstwy komponują te proste wzorce w bardziej złożone reprezentacje (np. części obiektów, całe obiekty). Ta hierarchiczna nauka reprezentacji pozwala głębokim sieciom osiągać wyniki przewyższające tradycyjne metody w wielu zadaniach, szczególnie w dziedzinie wizji komputerowej, przetwarzania języka naturalnego i rozpoznawania mowy (Goodfellow et al., 2016; LeCun et al., 2015).

Architektura głębokiej sieci neuronowej. Podstawowym elementem sieci neuronowej jest perceptron wielowarstwowy (Multi-Layer Perceptron, MLP), składający się z warstw neuronów połączonych wagami. Dla wejścia $\mathbf{x} \in \mathbb{R}^d$, warstwa ukryta oblicza (Goodfellow et al., 2016):

$$\mathbf{h} = \sigma(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \quad (3.76)$$

gdzie $W^{(1)}$ to macierz wag, $\mathbf{b}^{(1)}$ to wektor biasów, a $\sigma(\cdot)$ to nieliniowa funkcja aktywacji.

Dla sieci wielowarstwowej transformacja danych przez kolejne warstwy wyraża się jako (Goodfellow et al., 2016):

$$\mathbf{h}^{(l)} = \sigma^{(l)}(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \quad (3.77)$$

gdzie l oznacza numer warstwy, $\mathbf{h}^{(0)} = \mathbf{x}$ to dane wejściowe, a $\mathbf{h}^{(L)}$ to wyjście sieci.

Warstwa wyjściowa dla klasyfikacji wieloklasowej często używa funkcji softmax (Goodfellow et al., 2016):

$$P(y = k | \mathbf{x}) = \frac{\exp(z_k)}{\sum_{j=1}^K \exp(z_j)} \quad (3.78)$$

gdzie $\mathbf{z} = W^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}$ to logity, a K to liczba klas.

Funkcje aktywacji. Nieliniowe funkcje aktywacji są kluczowe dla zdolności sieci do modelowania złożonych zależności. Najczęściej stosowane funkcje aktywacji to (Goodfellow et al., 2016; LeCun et al., 2015):

ReLU (Rectified Linear Unit): Najpopularniejsza funkcja aktywacji w głębokich sieciach (Goodfellow et al., 2016):

$$\text{ReLU}(x) = \max(0, x) \quad (3.79)$$

ReLU jest prosta obliczeniowo, łagodzi problem zanikającego gradientu i empirycznie przyspiesza trenowanie.

Leaky ReLU: Wariant ReLU pozwalający na małe ujemne wartości (Goodfellow et al., 2016):

$$\text{LeakyReLU}(x) = \max(\alpha x, x) \quad (3.80)$$

gdzie typowo $\alpha = 0.01$, co zapobiega „umieraniu” neuronów.

Sigmoid: Klasyczna funkcja aktywacji mapująca wartości do przedziału (0,1) (Goodfellow et al., 2016):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.81)$$

Obecnie rzadziej używana w warstwach ukrytych ze względu na problem zanikającego gradientu.

Tanh: Funkcja tangensa hiperbolicznego mapująca do przedziału (-1,1) (Goodfellow et al., 2016):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.82)$$

3.5.1 Algorytm wstecznej propagacji (Backpropagation)

Backpropagation to algorytm służący do efektywnego obliczania gradientów funkcji straty względem wszystkich wag w sieci neuronowej, umożliwiający ich optymalizację metodami gradientowymi. Algorytm wykorzystuje regułę łańcuchową do propagowania błędu od warstwy wyjściowej do warstw wejściowych, obliczając gradienty w sposób sekwencyjny i efektywny obliczeniowo (Goodfellow et al., 2016).

Funkcja straty. Dla klasyfikacji wieloklasowej typowo stosuje się entropię krzyżową (cross-entropy) (Goodfellow et al., 2016):

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K y_{ik} \log(\hat{y}_{ik}) \quad (3.83)$$

gdzie y_{ik} to prawdziwa etykieta (one-hot encoding), \hat{y}_{ik} to predykcja sieci, a θ reprezentuje wszystkie parametry sieci.

Propagacja wsteczna. Dla warstwy wyjściowej gradient błędu wynosi (Goodfellow et al., 2016):

$$\delta^{(L)} = \frac{\partial L}{\partial \mathbf{z}^{(L)}} \quad (3.84)$$

Dla warstw ukrytych gradient propaguje się wstecznie (Goodfellow et al., 2016):

$$\delta^{(l)} = ((W^{(l+1)})^\top \delta^{(l+1)}) \odot \sigma'^{(l)}(\mathbf{z}^{(l)}) \quad (3.85)$$

gdzie \odot oznacza iloczyn Hadamarda (element-wise), a $\sigma'^{(l)}$ to pochodna funkcji aktywacji.

Gradienty względem wag i biasów (Goodfellow et al., 2016):

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{h}^{(l-1)})^\top, \quad \frac{\partial L}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \quad (3.86)$$

3.5.2 Metody optymalizacji

Trenowanie głębokich sieci neuronowych wymaga efektywnych algorytmów optymalizacji, które są rozszerzeniami klasycznego gradientu prostego (Gradient Descent).

Stochastic Gradient Descent (SGD). SGD aktualizuje parametry na podstawie gradientu obliczonego na małym batchu danych zamiast całego zbioru treningowego (Goodfellow et al., 2016):

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L_{\text{batch}}(\theta) \quad (3.87)$$

gdzie η to współczynnik uczenia (learning rate), a L_{batch} to średnia strata na batchu.

Momentum. Momentum przyspiesza SGD w kierunkach z konsekwentnym gradientem i tłumii oscylacje (Goodfellow et al., 2016):

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + \eta \nabla_{\theta} L(\theta) \quad (3.88)$$

$$\theta \leftarrow \theta - \mathbf{v}_t \quad (3.89)$$

gdzie $\beta \in [0, 1)$ (typowo 0.9) kontroluje wpływ poprzednich gradientów.

Adam (Adaptive Moment Estimation). Adam to obecnie najpopularniejszy optymalizator, łączący momentum z adaptacyjnymi współczynnikami uczenia dla każdego parametru (Goodfellow et al., 2016):

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla_{\theta} L(\theta) \quad (3.90)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) (\nabla_{\theta} L(\theta))^2 \quad (3.91)$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \quad (3.92)$$

$$\theta \leftarrow \theta - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \quad (3.93)$$

gdzie typowo $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$.

3.5.3 Regularyzacja w sieciach głębokich

Głębokie sieci neuronowe są podatne na przeuczenie ze względu na dużą liczbę parametrów. Stosuje się różne techniki regularyzacji aby poprawić generalizację.

Dropout. Dropout to technika regularyzacji polegająca na losowym wyłączaniu neuronów podczas trenowania z prawdopodobieństwem p (typowo 0.5). Podczas forward pass każdy neuron jest zachowywany z prawdopodobieństwem $1 - p$ ([Goodfellow et al., 2016](#)):

$$\mathbf{h}^{(l)} = \mathbf{r}^{(l)} \odot \sigma(W^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \quad (3.94)$$

gdzie $\mathbf{r}^{(l)}$ to wektor binarny z elementami losowanymi z rozkładu Bernoulliego Bernoulli($1 - p$).

Podczas testowania wszystkie neurony są aktywne, a ich wyjścia są skalowane przez $(1 - p)$ aby zachować oczekiwana wartość aktywacji.

Batch Normalization. Batch Normalization normalizuje aktywacje każdej warstwy, co stabilizuje i przyspiesza trenowanie. Dla mini-batcha aktywacji $\{x_1, \dots, x_m\}$, normalizacja wynosi ([Goodfellow et al., 2016](#)):

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (3.95)$$

gdzie $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$ to średnia batcha, $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$ to wariancja batcha.

Następnie stosuje się przekształcenie liniowe z nauczalnymi parametrami ([Goodfellow et al., 2016](#)):

$$y_i = \gamma \hat{x}_i + \beta \quad (3.96)$$

gdzie γ i β są parametrami uczonymi przez backpropagation.

L2 Regularization (Weight Decay). Dodanie kary L2 do funkcji straty zapobiega zbyt dużym wartościom wag ([Goodfellow et al., 2016](#)):

$$L_{\text{total}} = L(\theta) + \lambda \sum_l \|W^{(l)}\|_F^2 \quad (3.97)$$

gdzie $\|\cdot\|_F$ to norma Frobeniusa, a λ kontroluje siłę regularyzacji.

3.5.4 Konwolucyjne sieci neuronowe (CNN)

Konwolucyjne sieci neuronowe to architektura zaprojektowana specjalnie do przetwarzania danych o strukturze kratowej, takich jak obrazy. CNN wykorzystują operację splotu (convolution) zamiast pełnego połączenia między neuronami, co drastycznie redukuje liczbę parametrów i pozwala na uczenie się lokalnych wzorców niezależnie od ich położenia w obrazie (translational invariance) ([LeCun et al., 2015](#); [Goodfellow et al., 2016](#)).

Warstwa konwolucyjna. Operacja splotu 2D dla obrazu wejściowego X i filtra (kernele) W wynosi ([Goodfellow et al., 2016](#)):

$$Y_{ij} = (X * W)_{ij} = \sum_m \sum_n X_{i+m, j+n} W_{mn} + b \quad (3.98)$$

gdzie Y to mapa cech (feature map), a b to bias.

Dla wielokanałowego wejścia (np. RGB) z C_{in} kanałami i C_{out} filtrami (Goodfellow et al., 2016):

$$Y_{ij}^{(k)} = \sum_{c=1}^{C_{\text{in}}} (X^{(c)} * W^{(k,c)})_{ij} + b^{(k)} \quad (3.99)$$

Pooling. Warstwy poolingu redukują wymiary przestrzenne map cech, zwiększając niezmienniczość na małe translacje. Max pooling wybiera maksymalną wartość z okna (Goodfellow et al., 2016):

$$Y_{ij} = \max_{(m,n) \in \mathcal{R}_{ij}} X_{mn} \quad (3.100)$$

gdzie \mathcal{R}_{ij} to region poolingu dla pozycji (i, j) .

Average pooling oblicza średnią (Goodfellow et al., 2016):

$$Y_{ij} = \frac{1}{|\mathcal{R}_{ij}|} \sum_{(m,n) \in \mathcal{R}_{ij}} X_{mn} \quad (3.101)$$

Typowa architektura CNN składa się z naprzemiennych warstw konwolucyjnych (z ReLU) i poolingu, zakończonych w pełni połączonymi warstwami do klasyfikacji (LeCun et al., 2015).

3.5.5 Rekurencyjne sieci neuronowe (RNN)

Rekurencyjne sieci neuronowe to architektura zaprojektowana do przetwarzania sekwencji danych o zmiennej długości, takich jak tekst, mowa czy szeregi czasowe. RNN utrzymują ukryty stan \mathbf{h}_t , który jest aktualizowany w każdym kroku czasowym na podstawie bieżącego wejścia \mathbf{x}_t i poprzedniego stanu \mathbf{h}_{t-1} , co pozwala sieci na „zapamiętywanie” informacji z przeszłości (Goodfellow et al., 2016).

Podstawowe RNN. Aktualizacja stanu ukrytego w podstawowym RNN (Goodfellow et al., 2016):

$$\mathbf{h}_t = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t + \mathbf{b}_h) \quad (3.102)$$

Wyjście w kroku t (Goodfellow et al., 2016):

$$\mathbf{y}_t = W_{hy}\mathbf{h}_t + \mathbf{b}_y \quad (3.103)$$

Podstawowe RNN cierpią na problem zanikającego/eksplodującego gradientu podczas uczenia długich sekwencji, co ogranicza ich zdolność do uczenia się długoterminowych zależności.

LSTM (Long Short-Term Memory). LSTM to zaawansowana architektura RNN rozwiązująca problem długoterminowych zależności poprzez wprowadzenie mechanizmu bramek (gates). Komórka LSTM składa się z stanu komórki \mathbf{c}_t i trzech bramek: forget gate, input gate i output gate (Goodfellow et al., 2016).

Brama zapominania (forget gate) (Goodfellow et al., 2016):

$$\mathbf{f}_t = \sigma(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad (3.104)$$

Bramka wejściowa (input gate) (Goodfellow et al., 2016):

$$\mathbf{i}_t = \sigma(W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad (3.105)$$

Kandydat nowego stanu komórki (Goodfellow et al., 2016):

$$\tilde{\mathbf{c}}_t = \tanh(W_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \quad (3.106)$$

Aktualizacja stanu komórki (Goodfellow et al., 2016):

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (3.107)$$

Bramka wyjściowa (output gate) (Goodfellow et al., 2016):

$$\mathbf{o}_t = \sigma(W_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \quad (3.108)$$

Stan ukryty (Goodfellow et al., 2016):

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (3.109)$$

LSTM skutecznie modeluje zależności na dystansie dziesiątek lub setek kroków czasowych.

Zastosowania uczenia głębokiego. Uczenie głębokie zrewolucjonizowało wiele dziedzin i znalazło szerokie zastosowania (Goodfellow et al., 2016; LeCun et al., 2015):

- **Wizja komputerowa** — klasyfikacja obrazów, detekcja obiektów, segmentacja semantyczna, rozpoznawanie twarzy.
- **Przetwarzanie języka naturalnego** — tłumaczenie maszynowe, analiza sentymentu, chatboty, generowanie tekstu.
- **Rozpoznawanie mowy** — transkrypcja audio, syntezator mowy, asystenci głosowi.
- **Systemy rekomendacyjne** — personalizacja treści, przewidywanie preferencji użytkowników.
- **Medycyna** — diagnostyka obrazowa (RTG, MRI, CT), przewidywanie chorób, odkrywanie leków.
- **Autonomiczne pojazdy** — percepja środowiska, planowanie trasy, kontrola pojazdu.
- **Przemysł i procesy wytwórcze** — kontrola jakości wizualna (inspekcja defektów), predykcyjne utrzymanie ruchu oparte na danych sensorycznych, optymalizacja parametrów procesów w czasie rzeczywistym.
- **Finanse** — wykrywanie oszustw, prognozowanie rynków, automatyczny trading.

W kontekście procesów wytwórczych, głębokie sieci konwolucyjne (CNN) umożliwiają automatyczną inspekcję jakości produktów na liniach produkcyjnych, wykrywając defekty z dokładnością przewyższającą inspekcję ludzką. Rekurencyjne sieci neuronowe (RNN/LSTM) są wykorzystywane do analizy szeregów czasowych z czujników w celu predykcji awarii maszyn i optymalizacji harmonogramów konserwacji (LeCun et al., 2015; Goodfellow et al., 2016).

4

Zastosowania uczenia maszynowego w procesach wytwórczych

4.1 Wstęp

Obecnie znajdujemy się w erze Przemysłu 4.0, gdzie technologia i automatyzacja stanowią podstawę procesów wytwórczych. Przemysł 4.0 integruje cyfrowe technologie, takie jak big data, przetwarzanie w chmurze, sztuczną inteligencję (AI), Internet Rzeczy (IoT), robotykę oraz uczenie maszynowe, aby stworzyć intelligentne fabryki (Schwab, 2016; Kagermann et al., 2013). Cyber-fizyczne systemy umożliwiają komunikację między maszynami, sensorami i oprogramowaniem, co pozwala na optymalizację procesów w czasie rzeczywistym oraz podejmowanie decyzji opartych na danych. Procesy wytwórcze przedsiębiorstw to element przedsiębiorstwa, który jest zmaga się z szeregiem wyzwań. Takimi jak rosnąca złożoność produktów, wymagania dotyczące wysokiej jakości przy jednoczesnej optymalizacji kosztów, konieczność elastycznego reagowania na zmieniające się potrzeby rynku oraz presja na redukcję przestojów produkcyjnych. Tradycyjne metody zarządzania produkcją, oparte na statycznych modelach i doświadczeniu ekspertów, coraz częściej okazują się niewystarczające w obliczu ogromnych ilości danych generowanych przez nowoczesne systemy produkcyjne (Zhong et al., 2017).

Jednakże uczenie maszynowe jako jedna z technologii umożliwia wykorzystanie potencjału Przemysłu 4.0. Algorytmy uczenia maszynowego, przedstawione w rozdziale 3, znajdują zastosowanie w analizie danych produkcyjnych, umożliwiając automatyczną identyfikację wzorców, predykcję stanów przyszłych oraz optymalizację parametrów procesowych bez konieczności jawnego programowania reguł decyzyjnych. Dzięki zdolności do uczenia się z danych historycznych i adaptacji do zmieniających się warunków, systemy oparte na uczeniu maszynowym mogą wspierać decyzje w czasie rzeczywistym, poprawiając efektywność, jakość i niezawodność procesów wytwórczych (Wuest et al., 2016; Cioffi et al., 2020).

Celem tego rozdziału jest przedstawienie różnych zastosowań uczenia maszynowego w procesach wytwórczych.

4.2 Konserwacja predykcyjna

Konserwacja predykcyjna (Predictive Maintenance, PdM) stanowi zaawansowaną strategię utrzymania ruchu, która wykorzystuje algorytmy uczenia maszynowego do przewidywania awarii maszyn i urządzeń produkcyjnych przed ich wystąpieniem, co skutkuje znaczącą minimalizacją nieplanowanych przestojów i optymalizacją kosztów operacyjnych (Lee et al., 2014; Mobley, 2002). W przeciwieństwie do konserwacji reaktywnej oraz preventywnej, konserwacja predykcyjna umożliwia proaktywne zarządzanie stanem technicz-

nym poprzez ciągłą analizę danych diagnostycznych i wczesne wykrywanie symptomów degradacji (Susto et al., 2015; Carvalho et al., 2019).

Koncepcja konserwacji predykcyjnej opiera się na założeniu, że każda maszyna generuje charakterystyczne sygnały świadczące o jej stanie zdrowia, które mogą być przechwytywane przez system sensoryczny i przetwarzane w czasie rzeczywistym (Ran et al., 2019). Dzięki temu przedsiębiorstwa odchodzą od modeli w których eksplloatuje się maszynę do awarii oraz wykonuje się konserwację w stałych odstępach czasowych, a przechodzą do modelu w którym konserwuje się w oparciu o stan techniczny, co prowadzi do istotnej poprawy efektywności wykorzystania zasobów i redukcji całkowitego kosztu posiadania (Mobley, 2002; Dalzochio et al., 2020).

4.2.1 Architektura systemów konserwacji predykcyjnej

Współczesne systemy konserwacji predykcyjnej składają się z kilku kluczowych warstw technologicznych (Ran et al., 2019; Zhang et al., 2019):

Warstwa zbierania danych obejmuje rozbudowaną infrastrukturę sensoryczną opartą o technologie Internetu Rzeczy (IoT). Czujniki przemysłowe monitorują krytyczne parametry eksplatacyjne, w tym:

- Wibracje mechaniczne – akcelerometry mierzące drgania łożysk, wałów, przekładni i silników
- Temperatura – czujniki termiczne oraz kamery termowizyjne do wykrywania przegrzania komponentów
- Ciśnienie – przetworniki ciśnienia w systemach hydraulicznych i pneumatycznych
- Parametry elektryczne – mierniki prądu, napięcia i mocy czynnej/biernej silników elektrycznych
- Parametry akustyczne – mikrofony przemysłowe do analizy hałasu i ultradźwięków
- Parametry procesowe – przepływ, stężenie substancji, wilgotność, obroty

Dane z czujników są przesyłane do warstwy przetwarzania za pomocą protokołów IoT, takich jak MQTT, OPC UA czy CoAP, zapewniając niskie opóźnienia i wysoką przepustowość (Zhang et al., 2019).

Warstwa edge computing umożliwia wstępne przetwarzanie danych bezpośrednio w lokalizacji maszyny, co redukuje obciążenie sieci i opóźnienia związane z przesyłem danych do chmury (Dalzochio et al., 2020). Urządzenia brzegowe wykonują:

- Filtrację i agregację danych surowych
- Ekstrakcję cech w dziedzinie czasu i częstotliwości
- Detekcję anomalii w czasie rzeczywistym z wykorzystaniem lekkich modeli ML
- Kompresję danych przed wysyłką do chmury

Warstwa analityki i modelowania wykorzystuje algorytmy uczenia maszynowego do budowy modeli predykcyjnych (Carvalho et al., 2019; Susto et al., 2015). Najczęściej stosowane metody to:

- **Random Forest i Gradient Boosting** – do klasyfikacji stanów maszyny oraz regresji pozostałego czasu użytkowania
- **Support Vector Machines (SVM)** – do klasyfikacji typów usterek na podstawie wzorców vibracyjnych
- **Sieci neuronowe LSTM i GRU** – do analizy sekwencji czasowych i modelowania trajektorii degradacji

Warstwa integracji systemowej łączy systemy PdM z istniejącą infrastrukturą IT przedsiębiorstwa, w tym z systemami SCADA (Supervisory Control and Data Acquisition), MES (Manufacturing Execution System) oraz ERP (Enterprise Resource Planning) (Zhang et al., 2019). Integracja ta umożliwia automatyczne generowanie zleceń konserwacyjnych, optymalizację harmonogramów napraw oraz śledzenie kosztów i wskaźników efektywności (KPI).

4.2.2 Zastosowania branżowe i efekty wdrożeń

Konserwacja predykcyjna znajduje szerokie zastosowanie w różnych sektorach przemysłu, przynosząc wymierne korzyści operacyjne i finansowe (Lee et al., 2014; Carvalho et al., 2019).

Przemysł motoryzacyjny wykorzystuje PdM do monitorowania linii montażowych i robotów przemysłowych. Nieplanowany przestój linii produkcyjnej w fabryce samochodów może kosztować od 500 tys. do 1 mln USD za godzinę (Lee et al., 2014). Implementacja systemów predykcyjnych w fabrykach koncernów takich jak BMW, Volkswagen czy General Motors przyniosła redukcję przestoju o 30-50% oraz wydłużenie żywotności krytycznych komponentów o 20-25% (Susto et al., 2015).

Energetyka stosuje PdM do turbozespołów elektrowni, turbin wiatrowych i transformatorów. W przypadku turbin wiatrowych, które pracują w trudnych warunkach atmosferycznych i są trudno dostępne, predykcja awarii łożysk i przekładni z wyprzedzeniem umożliwia planowanie napraw w optymalnych warunkach pogodowych (Ran et al., 2019). Producenci turbin gazowych raportują osiągnięcie bardzo wysokich poziomów dostępności dzięki systemom predykcyjnym opartym na LSTM i analizie setek parametrów procesowych (Lee et al., 2014).

Przemysł farmaceutyczny wymaga wysokiej niezawodności procesów ze względu na wymogi regulacyjne. Systemy PdM monitorują urządzenia w procesach sterylnych, takich jak autoklavy, suszarki próżniowe czy bioreaktory, zapewniając ciągłość produkcji i zgodność z procedurami walidacyjnymi (Dalzochio et al., 2020). Redukcja przestoju w produkcji szczepionek czy leków biologicznych o 15-25% przekłada się na oszczędności rzędu milionów dolarów rocznie.

Przemysł elektroniczny wykorzystuje PdM do utrzymania precyzyjnych systemów pozycjonowania, kontroli atmosfery oraz urządzeń do nanoszenia warstw. W fabrykach półprzewodników, gdzie wartość produktu w toku jest ekstremalna wysoka, nawet krótkie przestoje prowadzą do strat milionowych (Susto et al., 2015).

4.2.3 Korzyści ekonomiczne i operacyjne

Wdrożenia systemów konserwacji predykcyjnej generują wielowymiarowe korzyści (Mobley, 2002; Carvalho et al., 2019):

- **Redukcja kosztów konserwacji o 10-40%** – eliminacja zbędnych przeglądów prewencyjnych oraz optymalizacja zużycia części zamiennych
- **Wzrost dostępności maszyn powyżej 95%** – minimalizacja nieplanowanych przestojów i skrócenie czasu napraw dzięki przygotowaniu części i personelu z wyprzedzeniem
- **Przedłużenie żywotności aktywów o 15-30%** – optymalne planowanie wymian komponentów przed katastroficznym uszkodzeniem
- **Poprawa bezpieczeństwa personelu** – wczesne wykrywanie warunków niebezpiecznych (przegrzanie, nadmierne drgania) redukuje ryzyko wypadków
- **Optymalizacja zapasów magazynowych** – precyzyjna predykcja zapotrzebowania na części zamienne umożliwia redukcję kapitału zamrożonego w magazynie o 20-35%
- **Wzrost efektywności energetycznej** – wykrywanie niewydolnych komponentów prowadzi do oszczędności energii rzędu 5-15%

Zwrot z inwestycji (ROI) dla projektów PdM zazwyczaj osiągany jest w horyzoncie 12-24 miesięcy, przy czym krótsze okresy zwrotu charakteryzują branże o wysokich kosztach przestojów (motoryzacja, półprzewodniki, energetyka) (Mobley, 2002).

4.3 Optymalizacja parametrów procesów produkcyjnych

Optymalizacja parametrów procesów produkcyjnych z wykorzystaniem uczenia maszynowego jest elementem realizacji koncepcji inteligentnej fabryki w ramach idei Przemysłu 4.0 (Wang et al., 2022b; Monostori et al., 2016). Tradycyjne podejście do optymalizacji oparte na statycznych modelach fizycznych, heurystykach lub doświadczeniu inżynierów procesowych są niewystarczające w obliczu rosnącej złożoności procesów, dynamicznie zmieniających się warunków produkcji oraz ogromnych ilości danych generowanych przez współczesne systemy automatyki przemysłowej (Kusiak, 2018).

Uczenie maszynowe oferuje możliwość dynamicznego dostosowywania parametrów procesowych takich jak prędkość linii, temperatura, ciśnienie, przepływ, dozowanie substancji, siły skrawania czy parametry spawania w czasie rzeczywistym, na podstawie bieżących danych z czujników oraz przewidywanych zmian w środowisku produkcyjnym (Sharp et al., 2018; Weichert et al., 2019). Dzięki zdolności do modelowania złożonych, nieliniowych zależności między parametrami wejściowymi a wskaźnikami jakości, wydajności i kosztów, algorytmy ML umożliwiają osiągnięcie optymalnych punktów pracy procesów, które byłyby trudne lub niemożliwe do znalezienia metodami konwencjonalnymi (Wang et al., 2022b).

4.3.1 Architektura systemów optymalizacji

Systemy optymalizacji procesów produkcyjnych oparte na uczeniu maszynowym składają się z kilku współpracujących ze sobą modułów ([Sharp et al., 2018; Psarommatis and Kiritsis, 2022](#)):

Moduł akwizycji i integracji danych gromadzi informacje z wielorakich źródeł, obejmujących:

- Czujniki IoT na liniach produkcyjnych (mierzące parametry takie jak: temperatura, ciśnienie, wilgotność, przepływ, wibracje, akustyka)
- Systemy wizyjne i spektrometryczne do oceny jakości produktów w czasie rzeczywistym
- Dane z systemów MES i SCADA – parametry ustawień maszyn, czasy cyklu, ilości produkcji
- Dane z systemów ERP – zlecenia produkcyjne, dostępność surowców, harmonogramy
- Dane środowiskowe – temperatura i wilgotność otoczenia, jakość zasilania elektrycznego

Integracja tych źródeł wymaga zastosowania warstw pośredniczących implementujących standardy komunikacyjne takie jak OPC UA, MQTT, czy DDS (Usługa Dystrybucji Danych) ([Psarommatis and Kiritsis, 2022](#)).

Moduł przetwarzania w czasie rzeczywistym wykorzystuje technologie przetwarzania brzegowego do wstępnej analizy danych oraz podejmowania szybkich decyzji optymalizacyjnych bez konieczności komunikacji z centralną chmurą obliczeniową ([Wang et al., 2022b](#)). Minimalizacja opóźnień jest krytyczna dla procesów o wysokiej dynamice, takich jak obróbka skrawaniem, wtrysk plastiku czy formowanie blach, gdzie parametry muszą być dostosowywane w przedziałach czasowych rzędu milisekund do sekund.

Moduł modelowania i uczenia implementuje różnorodne algorytmy uczenia maszynowego dostosowane do specyfiki problemu optymalizacyjnego ([Kusiak, 2018; Sharp et al., 2018](#)):

- **Uczenie nadzorowane** – modele regresji prognozują wskaźniki jakości i wydajności na podstawie parametrów procesowych. Wytrenowane na danych historycznych, modele te służą jako modele zastępcze rzeczywistych procesów, umożliwiając szybką symulację różnych scenariuszy bez potrzeby eksperymentów fizycznych
- **Uczenie nienadzorowane** – algorytmy klasteryzacji identyfikują reżimy operacyjne wykrywając anomalie w zachowaniu procesów.
- **Uczenie ze wzmacnieniem** – agenci uczą się optymalnych polityk sterowania poprzez interakcję z procesem (fizycznym lub symulowanym), maksymalizując kumulatywną nagrodę zdefiniowaną jako funkcja wydajności, jakości i kosztów ([Wang et al., 2022b; Monostori et al., 2016](#)). Uczenie ze wzmacnieniem jest szczególnie efektywne w problemach sekwencyjnego podejmowania decyzji, gdzie bieżące działania wpływają na przyszłe stany systemu

Moduł sterowania adaptacyjnego implementuje zamkniętą pętlę sterowania, w której obliczone parametry optymalne są przekazywane do sterowników PLC/DCS linii produkcyjnych (Psarommatis and Kiritsis, 2022). System monitoruje efekty zmian parametrów i dynamicznie dostosowuje strategię w odpowiedzi na odchylenia od prognozowanych wyników.

4.3.2 Techniki i algorytmy optymalizacji

W praktyce przemysłowej stosuje się różne podejścia do optymalizacji, w zależności od charakterystyki procesu i dostępnych danych (Sharp et al., 2018; Wang et al., 2022b).

Optymalizacja oparta na modelach zastępczych polega na wytrenowaniu modelu uczenia maszynowego jako symulacji rzeczywistego procesu produkcyjnego (Shahriari et al., 2016). Model ten, znacznie szybszy w ewaluacji niż proces fizyczny, jest następnie wykorzystywany w procedurach optymalizacyjnych do poszukiwania optymalnych ustawań parametrów. Podejście to jest szczególnie wartościowe w procesach o długich czasach cyklu (np. obróbka cieplna, utwardzanie kompozytów), gdzie eksperymentacja fizyczna jest kosztowna czasowo.

Optymalizacja w czasie rzeczywistym z wykorzystaniem uczenia wzmacnionego umożliwia adaptacyjne dostosowywanie parametrów w trakcie produkcji (Wang et al., 2022b; Monostori et al., 2016). Agent obserwuje stan procesu (wartości czujników, wskaźniki jakości) i podejmuje akcje poprzez zmianę parametrów, otrzymując nagrodę proporcjonalną do osiągniętych wskaźników wydajnościowych.

Hybrydowe podejścia łączące wiedzę ekspertową z ML integrują reguły fizyczne i ograniczenia procesowe z modelami data-driven (Weichert et al., 2019). Na przykład, w procesach spawania, model uczenia maszynowego może optymalizować parametry (prąd, napięcie, prędkość) w ramach okna parametrów zdefiniowanego przez normy spawalnicze i właściwości materiałów.

4.3.3 Zastosowania branżowe i rezultaty wdrożeń

Optymalizacja oparta o uczenie maszynowe znajduje zastosowanie w szerokim spektrum procesów produkcyjnych (Wang et al., 2022b; Kusiak, 2018; Sharp et al., 2018).

Przemysł motoryzacyjny wykorzystuje uczenie maszynowe do optymalizacji linii montażowych, lakierowania i procesów spawalniczych. W procesie lakierowania, algorytmy optymalizują natężenie przepływu lakieru, ciśnienie w komorach, temperaturę i wilgotność w celu minimalizacji zużycia materiału przy zachowaniu jednolitej grubości powłoki (Wang et al., 2022b). Producenci motoryzacyjni raportują znaczącą redukcję odpadów produkcyjnych oraz istotną poprawę efektywności energetycznej linii produkcyjnych po wdrożeniu systemów optymalizacji opartych na uczeniu ze wzmacnieniem.

Przemysł spożywczy i opakowaniowy stosuje uczenie maszynowe do optymalizacji procesów mieszania, formowania, pieczenia i pakowania. W produkcji kopert i opakowań kartonowych, systemy analizują jakość surowca (gramatura, wilgotność papieru) i dynamicznie dostosowują prędkość linii, siły dociskowe i temperatury klejenia (Kusiak, 2018). Producenci opakowań raportują znaczącą redukcję wadliwych produktów po wdrożeniu optymalizacji ML w liniach aseptycznego pakowania.

Przemysł chemiczny i petrochemiczny wykorzystuje uczenie maszynowe do optymalizacji reaktorów, kolumn destylacyjnych i procesów separacji. W rafineriach, modele predykcyjne optymalizują temperatury, ciśnienia i przepływy w kolumnach destylacyjnych w celu maksymalizacji uzysku frakcji wartościowych (benzyna, olej napędowy) przy minimalizacji zużycia energii (Wang et al., 2022b). Koncerny petrochemiczne raportują wzrost wydajności procesów oraz redukcję emisji CO₂ dzięki systemom opartym na uczeniu maszynowym.

Produkcja półprzewodników wymaga ekstremalnej precyzji parametrów procesowych. Uczenie maszynowe optymalizuje procesy litografii, trawienia i nanoszenia warstw, gdzie nawet minimalne odchylenia parametrów prowadzą do defektów kosztujących miliony dolarów (Sharp et al., 2018). Producenci półprzewodników wykorzystują systemy optymalizacji do utrzymania wysokiej wydajności produkcyjnej dla najbardziej zaawansowanych węzłów technologicznych.

Przemysł tekstylny i odzieżowy stosuje uczenie maszynowe do optymalizacji procesów farbowania, wykańczania i krojenia tkanin. Systemy optymalizują zużycie wody, barwników i energii w procesach farbowania (Kusiak, 2018).

4.4 Kontrola jakości i analiza predykcyjna

Kontrola jakości stanowi kluczowy element zarządzania procesami wytwórczymi, bezpośrednio wpływając na satysfakcję klientów, konkurencyjność produktów oraz rentowność przedsiębiorstwa (Huang et al., 2021; Wang et al., 2022a). Tradycyjne metody kontroli jakości oparte na manualnej inspekcji, próbkowaniu statystycznym oraz kartach kontrolnych charakteryzują się istotnymi ograniczeniami: są czasochłonne, subiektywne, kosztowne i często wykrywają defekty, gdy wadliwe produkty już zostały wyprodukowane (Psarommatis et al., 2020). Uczenie maszynowe rewolucjonizuje podejście do kontroli jakości poprzez umożliwienie automatycznego wykrywania defektów w czasie rzeczywistym, analizy predykcyjnej trendów jakościowych oraz identyfikacji przyczyn źródłowych problemów jakościowych (Weimer et al., 2016; Wang et al., 2022a). Algorytmy uczenia maszynowego, w szczególności techniki wizji komputerowej oparte na sieciach neuronowych konwolucyjnych, potrafią analizować obrazy produktów z dokładnością przewyższającą ludzkich inspektorów, wykrywając subtelne defekty niedostrzegalne gołym okiem (Tabernik et al., 2020; Yang et al., 2020).

Analiza predykcyjna wykorzystuje dane historyczne z procesów produkcyjnych parametry maszyn, właściwości materiałów, warunki środowiskowe, wyniki testów do prognozowania przyszłej jakości produktów przed ich faktycznym wytworzeniem (Huang et al., 2021). Dzięki temu możliwe jest proaktywne zapobieganie problemom jakościowym poprzez wcześniejszą interwencję i korektę parametrów procesowych, minimalizując odpady, przeróbki i straty finansowe.

4.4.1 Automatyczna inspekcja wizualna oparta na uczeniu głębowym

Wizja komputerowa wspomagana przez sieci neuronowe stanowi najbardziej zaawansowaną technologię w automatycznej kontroli jakości produktów (Tabernik et al., 2020).

Architektury sieciowe dla inspekcji wizualnej

Sieci konwolucyjne (CNN) stanowią fundament systemów inspekcji wizualnej (Tabernik et al., 2020). Podstawowe architektury obejmują:

- **Klasyfikacja defektów** – sieci takie jak ResNet, EfficientNet czy Vision Transformer klasyfikują obrazy na kategorie lub rozpoznają typy defektów. Osiągają dokładność klasyfikacji powyżej 95-99% dla dobrze zdefiniowanych klas defektów (Yang et al., 2020)
- **Segmentacja semantyczna** – architektury typu U-Net, DeepLab czy Mask R-CNN lokalizują defekty na poziomie pikseli, tworząc precyzyjne maski obszarów wadliwych. Umożliwia to nie tylko wykrycie defektu, ale również jego wymiarowanie i lokalizację przestrzenną (Tabernik et al., 2020)
- **Detekcja obiektów** – algorytmy takie jak YOLO, Faster R-CNN czy RetinaNet wykrywają i lokalizują wiele defektów jednocześnie w czasie rzeczywistym (>30 klatek na sekundę), co jest ważne, gdy praca jest wykonywana na szybkiej linii produkcyjnej (Wang et al., 2022a)
- **Few-shot learning i transfer learning** – w przemysłowych zastosowaniach często brakuje dużych zbiorów etykietowanych obrazów defektów. Techniki takie jak few-shot learning, meta-learning czy transfer learning z modeli pretrenowanych na ImageNet umożliwiają efektywne uczenie przy ograniczonej liczbie przykładów (nawet 10-50 obrazów na klasę) (Weimer et al., 2016)

Detekcja anomalii bez nadzoru

W wielu scenariuszach przemysłowych uzyskanie etykietowanych przykładów defektów jest trudne lub niemożliwe ze względu na ich rzadkość i różnorodność. Metody uczenia nienadzorowanego adresują ten problem (Bergmann et al., 2021; Tabernik et al., 2020):

- **Autoencodery** – sieci neuronowe trenowane do rekonstrukcji obrazów produktów prawidłowych. Podczas inspekcji, obrazy z defektami generują wysokie błędy rekonstrukcji, co umożliwia ich wykrycie bez konieczności posiadania etykietowanych przykładów defektów
- **Generatywne sieci przeciwwstawnne (GANs)** – modele takie jak AnoGAN czy f-AnoGAN uczą się rozkładu prawdopodobieństwa obrazów prawidłowych, a następnie wykrywają anomalie jako punkty o niskim prawdopodobieństwie
- **Jednoklasowy SVM i Las Izolacyjny (One-class SVM i Isolation Forest)** – klasyczne metody ML stosowane do detekcji outlierów w przestrzeni cech wyekstrahowanych przez CNN
- **Uczenie pod własnym nadzorem (Self-supervised learning)** – techniki takie jak SimCLR, MoCo czy DINO pozwalają na uczenie reprezentacji wizualnych bez etykiet, które następnie mogą być wykorzystane do detekcji anomalii

Implementacja systemów inspekcji wizualnej

Typowy system automatycznej inspekcji wizualnej składa się z następujących komponentów (Yang et al., 2020; Wang et al., 2022a):

- **Kamery przemysłowe** – wysokorozdzielcze kamery (5-20 Mpx) z częstotliwością próbkowania 30-200 FPS, wyposażone w obiektywy telcentryczne lub standardowe.
- **Oświetlenie** – element warunkujący jakość obrazów. Przy ustawianiu odpowiedniego oświetlenia trzeba zadbać o równomierne rozproszenie, podświetlenie dla przezroczystych obiektów, eliminacja cieni, pomiary 3D.
- **Urządzenia do przetwarzania brzegowego (Edge computing devices)** – jednostki GPU (NVIDIA Jetson, Intel Neural Compute Stick) lub specjalizowane akceleratory AI (Google Edge TPU, Intel Movidius) realizujące inferencję modeli w czasie rzeczywistym bezpośrednio na linii produkcyjnej.
- **Mechanizmy odrzutu** – systemy pneumatyczne lub mechaniczne automatycznie usuwające wadliwe produkty z linii na podstawie wyników inspekcji.
- **Interfejs operator-system** – dashboardy wizualizujące statystyki defektów, trendy jakościowe, alerty oraz umożliwiające weryfikację decyzji systemu przez operatorów.

Nowoczesne systemy osiągają przepustowość inspekcji rzędu 100-1000 produktów/minutę przy dokładności wykrywania defektów powyżej 95% i współczynniku fałszywych alarmów poniżej 2-5% ([Tabernik et al., 2020](#)).

4.4.2 Analiza predykcyjna jakości

Predykcja jakości produktów na podstawie danych procesowych umożliwia proaktywne zarządzanie jakością i minimalizację strat produkcyjnych ([Psarommatis et al., 2020](#)).

Modele predykcyjne i źródła danych

Systemy analityki predykcyjnej jakości (Predictive Quality Analytics) budują modele statystyczne i uczenia maszynowego mapujące relacje między parametrami wejściowymi procesów a wskaźnikami jakości produktów ([Huang et al., 2021](#)). Źródła danych obejmują:

- Parametry procesowe – temperatura, ciśnienie, prędkość, przepływ, siły, napięcia, pobór prądu
- Właściwości materiałów – gramatura, wilgotność, skład chemiczny, twardość, ciągliwość
- Warunki środowiskowe – temperatura i wilgotność hali produkcyjnej, jakość zasilania elektrycznego
- Dane o urządzeniach – wiek, stopień zużycia, historia konserwacji
- Historyczne dane jakościowe – wyniki testów laboratoryjnych, pomiary wymiarowe, wyniki inspekcji

Algorytmy stosowane w PQA to m.in. Random Forest, Gradient Boosting (XGBoost, LightGBM), sieci neuronowe (MLP, LSTM dla danych sekwencyjnych), SVM oraz modele liniowe (ridge regression, LASSO) dla interpretowalności ([Wang et al., 2022a](#)).

4.4.3 Zastosowania branżowe

Systemy kontroli jakości bazujacej na uczeniu maszynowym znajdują zastosowanie w wielu sektorach przemysłowych.

Przemysł motoryzacyjny – inspekcja wizualna spawów, powłok lakierniczych, montażu komponentów oraz testowanie funkcjonalne. Systemy CNN wykrywają mikrorysy w lakierze, nierównomierności spoin, błędy montażu. Redukcja defektów dochodzi do 30-50%, a koszty inspekcji spadają o 40-60% w porównaniu z inspekcją manualną ([Weimer et al., 2016; Yang et al., 2020](#)). BMW, Audi i Tesla wdrożyły systemy inspekcji AI na liniach karoserii i montażu końcowego.

Przemysł spożywczy – kontrola kształtu, koloru, rozmiaru, obecności ciał obcych w produktach spożywczych. Systemy analizują owoce, warzywa, pieczywo,mięso, produkty mleczarskie. W produkcji jogurtów kontrola polega na detekcji wad opakowań, błędnych etykiet, nieszczelności. W przetwórstwie mięsnym polega na wykrywanie fragmentów kości, tworzy sztucznych, kontaminacji ([Yang et al., 2020](#)). Systemy zapewniają zgodność z normami BRC, IFS, FSSC 22000.

Przemysł farmaceutyczny – inspekcja tabletek podkątem wad wizualnych, ampułek podkątem uszkodzeń, blistrów pod kątem kompletności i jakości wizualnej tabletek. Systemy muszą spełniać wymogi FDA 21 CFR Part 11 i EU Annex 11 ([Wang et al., 2022a](#)). Pfizer, Novartis i Roche stosują systemy inspekcji AI z wymaganą wykrywalnością defektów krytycznych >99.9%.

Przemysł tekstylny – detekcja wad tkanin w czasie rzeczywistym na maszynach tkackich. Systemy CNN przetwarzają obrazy z kamer liniowych przy prędkości tkaniny 50-100 m/min ([Tabernik et al., 2020](#)).

Produkcja stali i metali – inspekcja powierzchni blach, prętów, rur. Wykrywanie wad powierzchniowych, pomiary wymiarowe, kontrola powłok antykorozjnych ([Yang et al., 2020](#)).

5

Studium przypadku: Klasyfikacja defektów w produktach odlewniczych

5.1 Wprowadzenie

Poprzednie rozdziały niniejszej pracy przedstawiły teoretyczne podstawy uczenia maszynowego, szczegółowy przegląd algorytmów klasyfikacji i regresji oraz ich zastosowania w kontekście procesów wytwórczych. Niniejszy rozdział stanowi praktyczną część pracy, której celem jest weryfikacja skuteczności omawianych metod w rzeczywistym problemie przemysłowym.

Automatyczna kontrola jakości z wykorzystaniem wizji komputerowej i algorytmów uczenia maszynowego jest stosowana we współczesnym przemyśle. W kontekście Przemysłu 4.0, gdzie integracja systemów cyber-fizycznych, Internetu Rzeczy i zaawansowanej analityki danych staje się standardem, zdolność do automatycznego wykrywania defektów w czasie rzeczywistym bezpośrednio przekłada się na konkurencyjność przedsiębiorstw produkcyjnych.

5.2 Cel i uzasadnienie studium przypadku

5.2.1 Cel badania

Głównym celem niniejszego studium przypadku jest wyszkolenie i ocena modeli uczenia maszynowego do klasyfikacji defektów w produktach odlewniczych na podstawie obrazów. Modele wybrane do przeprowadzenia eksperymentów to:

- Drzewa decyzyjne (Decision Trees)
- Las losowy (Random Forest)
- Maszyna wektorów nośnych (Support Vector Machine, SVM)
- Sieci neuronowe (Neural Networks)
- K najbliższych sąsiadów (K-Nearest Neighbors, KNN)
- regresja logistyczna (Logistic Regression)

Dla każdego z modeli zostaną przeprowadzone eksperymenty mające na celu ocenę ich skuteczności w klasyfikacji i przedstawione zostaną wyniki porównawcze.

5.2.2 Charakterystyka procesu produkcyjnego i systemu kontroli jakości

Analizowany proces dotyczy produkcji komponentów metalowych metodą odlewania ciśnieniowego, jest to jeden z najbardziej wydajnych technik wytwarzania części o złożonych kształtach geometrycznych. Proces przebiega następująco

1. Przygotowanie formy odlewniczej.
2. Wtrysk stopionego metalu do formy pod wysokim ciśnieniem.
3. Chłodzenie i utwardzanie odlewów w formie.
4. Usunięcie odlewów z formy i obróbka końcowa.
5. Kontrola jakości odlewów pod kątem defektów powierzchniowych i strukturalnych.

Proces charakteryzuje się bardzo wysoką wydajnością i powtarzalnością, ale jest wrażliwy na zaburzenia parametrów technologicznych, które prowadzą do powstawania defektów.

Tradycyjna kontrola jakości (wizualna) opiera się na inspekcji wizualnej przeprowadzonej przez wykwalifikowany personel. Proces jest czasochlonny, podatny na błędy ludzkie i trudny do skalowania przy rosnącej produkcji. Zakresowe oceny podlegają pory powierzchniowe, pęknięcia, zafalowania, niedomycia, ślady erozji formy, błyszczące przebarwienia itp.

5.3 Definicja problemu decyzyjnego

Analizowany problem kontroli jakości w procesie produkcyjnym został sformalizowany jako zadanie klasyfikacji binarnej, gdzie celem jest przypisanie każdego obrazu odlewów do jednej z dwóch klas: "defekt" lub "brak defektu". Taki sposób ujęcia problemu odpowiada praktycznym wymaganiom przemysłu produkcyjnego, gdzie produkt jest albo zaakceptowany, albo odrzucony.

Podstawą do przypisania etykiety "defekt" jest wykrycie widocznych wad powierzchniowych, takich jak rysy, ubytki, pęknięcia czy inne niezgodności z normami jakościowymi. W kontekście danych obrazowych każda próbka wejściowa reprezentuje pojedynczy wyrób, a decyzja klasyfikacyjna jest opisana na cechach wizualnych, które mogą być trudne do opisania za pomocą reguł deterministycznych. Z tego przyczyny tradycyjne metody inspekcji wizualnej bazujące na ręcznej inspekcji, charakteryzują się ograniczoną skutecznością i podatnością na subiektywną ocenę inspektorów.

Zastosowanie algorytmów uczenia maszynowego w tym obszarze jest uzasadnione ze względu na możliwość automatycznego uczenia się istotnych cech na podstawie danych historycznych. Uczenie maszynowe pozwala na wykrywanie subtelnych różnic pomiędzy wyrobami zgodnymi i niezgodnymi, zapewnia powtarzalność i skalowalność procesu kontroli jakości.

Szczególnie modele klasyfikacyjne, takie jak drzewa decyzyjne, lasy losowe, SVM lub sieci neuronowe są w stanie skutecznie odwzorowywać złożone zależności pomiędzy cechami wyrobu a jego jakością, co czyni je odpowiednimi narzędziami do automatyzacji procesu kontroli jakości.

5.4 Opis danych

Nazwa zbioru: Casting Product Image Data for Quality Inspection

Pochodzenie: Dane zostały pobrane ze strony Kaggle, platformy udostępniającej zbory danych do celów edukacyjnych i badawczych. Dane są dostępne pod adresem: <https://www.kaggle.com/datasets/ravirajsinh45/real-life-industrial-dataset-of-casting>

Opis zbioru danych: Ten zbiór danych został zebrany w stabilnym środowisku oświetleniowym z dodatkowym układem. Używany aparat to lustrzanka cyfrowa Canon EOS 1300D.

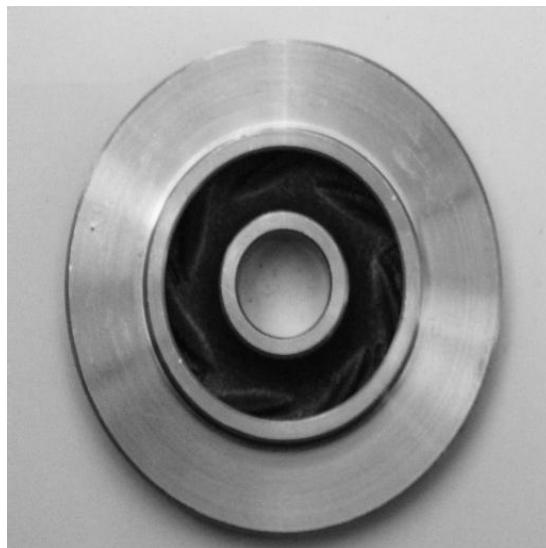
Liczba próbek: zbiór zawiera łącznie 1300 obrazów, z czego 519 to obrazy bez defektów, a 781 to obrazy z defektami.

Format danych: Obrazy są zapisane w formacie JPEG o rozdzielczości 512x512 pikseli.

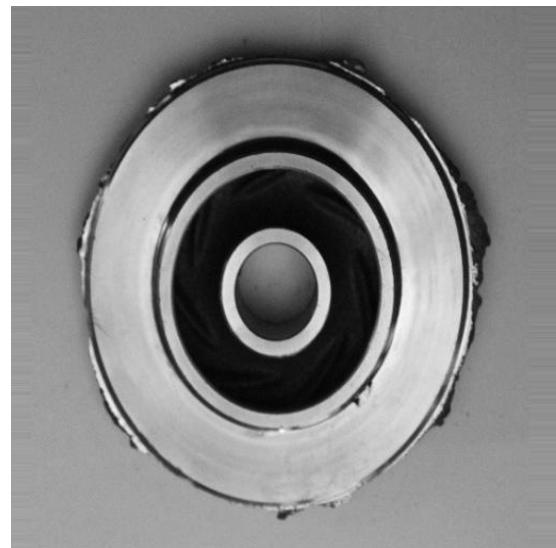
Liczba cech: Każdy obraz zawiera $512 \times 512 = 262144$ pikseli, które mogą być przetwarzane jako cechy wejściowe po odpowiedniej ekstrakcji cech.

Zmienna docelowa: Binarna klasyfikacja - "OK" lub "Defekt".

Przykładowe obrazy:



Rysunek 5.1: Przykład odlewu bez defektów (klasa "OK")



Rysunek 5.2: Przykład odlewu z defektami (klasa "Defect")

Jak widać na powyższych obrazach, produkty klasy "OK" charakteryzują się gładką, równomierną powierzchnią bez widocznych wad. Natomiast produkty wadliwe wykazują różnorodne defekty powierzchniowe, takie jak pory, pęknięcia lub nierówności strukturalne.

5.5 Przygotowanie danych

Przygotowanie danych bezpośrednio wpływa na jakość i skuteczność modeli klasycyfikacyjnych. W niniejszym projekcie zastosowano kompleksowe podejście do przetwarzania danych obrazowych, uwzględniające najlepsze praktyki w dziedzinie wizji komputerowej i uczenia maszynowego.

5.5.1 Czyszczenie danych

Pierwszym krokiem było sprawdzenie integralności danych. Została przeprowadzona identyfikacja i walidacja formatów plików obrazowych (JPEG). Nie wykryto uszkodzonych lub niekompletnych plików, co pozwoliło na kontynuację analizy bez konieczności usuwania próbek. Następnie została przeprowadzona kontrola jakości obrazów pod kątem rozdzielczości. Wszystkie obrazy zostały skonwertowane do jedolitej skali szarości, co oznacza redukcję wymiarowości z 3 kanałów RGB (czerwony, zielony, niebieski) do pojedynczego kanału (skala szarości). Ta transformacja upraszcza dane wejściowe i zmniejsza złożoność obliczeniową modeli.

5.5.2 Normalizacja i standaryzacja danych

Normalizacja min-max: Piksele obrazów zostały znormalizowane do zakresu [0, 1]. Zastosowano liniowe przekształcenie wartości pikseli zgodnie ze wzorem:

$$X_{\text{znormalizowane}} = \frac{X}{255.0} \quad (5.1)$$

gdzie X to oryginalna wartość piksela [0, 255]. Piksele w przedziale [0, 1] zapewniają stałość numeryczną podczas obliczeń, przyśpieszają zbieżność algorytmów optymalizacji i eliminują problemy związane z różnymi skalami cech.

Standaryzacja: Dodatkowo, dla niektórych modeli zastosowano standaryzację cech zgodnie ze wzorem:

$$X_{\text{standardized}} = \frac{X - \mu}{\sigma} \quad (5.2)$$

gdzie μ to średnia wartość pikseli w zbiorze treningowym, a σ to odchylenie standardowe. Standaryzacja zapewnia, że cechy mają średnią 0 i odchylenie standardowe 1, co jest ważne dla algorytmów wrażliwych na skalę danych.

5.5.3 Ekstrakcja cech

Każdy obraz o rozdzielczości 512×512 pikseli został przekształcony w wektor 262,144-wymiarowy, gdzie każdy wymiar reprezentuje intensywność pojedynczego piksela. Przekształcenie obrazu na wektor zachowuje pełną informację o strukturze i tekstuurze, umożliwia bezpośrednią reprezentację intesywności pikseli bez utraty informacji i co najważniejsze, pozwala algorytmom samodzielnie identyfikować istotne wzorce w danych.

$$\text{Obraz} \in \mathbb{R}^{512 \times 512} \rightarrow \text{Wektor} \in \mathbb{R}^{262144} \quad (5.3)$$

5.5.4 Podział danych na zbiory

Zastosowano metodologię stratyfikowanego podziału danych zapewniającą reprezentatywność statystyczną próbek:

Zbiór treningowy Zbiór treningowy przeznaczony jest do uczenia modeli i optymalizacji ich parametrów. Rozmiar tego zbioru wynosi 80%, czyli 1040 obrazów (415 bez defektów, 625 z defektami).

Zbiór testowy Zbiór testowy służy do oceny ogólnej wydajności wytrenowanych modeli na niewidzianych dla nich danych. Rozmiar tego zbioru wynosi 20%, czyli 260 obrazów (104 bez defektów, 156 z defektami). Metryki ewaluacyjne użyte do oceny modeli to dokładność (accuracy), precyzja (precision), czułość (recall), F1-score i macierz zamieszania (confusion matrix).

5.5.5 Walidacja jakości przygotowania danych

W ramach kontroli jakości przeprowadzonej w projekcie zweryfikowano rozkład klas w zbiorach treningowym i testowym w celu potwierdzenia poprawności stratyfikacji danych. Skontrolowano wymiarowość macierzy danych, przeprowadzając walidację kształtu struktur. Sprawdzono zakres wartości po normalizacji, weryfikując wartości minimalne i maksymalne. Ponadto potwierdzono brak wartości brakujących poprzez detekcję wartości NaN w zbiorze danych.

5.6 Wybór i opis zastosowanego algorytmu uczenia maszynowego

W niniejszym projekcie zdecydowano się na zastosowanie metody porównawczej, obejmującej sześć różnych algorytmów klasyfikacji binarnej. Podejście to pozwala przeanalizować skuteczność różnych modeli uczenia maszynowego w kontekście wykrywania defektów produkcyjnych. Wybrane algorytmy reprezentują zarówno klasyczne metody statystyczne, jak i zaawansowane techniki uczenia zespołowego (ensemble learning) oraz uczenia głębowego.

5.6.1 K-najbliższych sąsiadów (k-Nearest Neighbors)

Algorytm k-Nearest Neighbors należy do grupy metod nieparametrycznych. W przeciwieństwie do większości algorytmów uczenia maszynowego, k-NN nie buduje jawnego modelu podczas fazy treningu. Zamiast tego przechowuje w pamięci wszystkie przypadki treningowe i podejmuje decyzje klasyfikacyjne dopiero w momencie prezentacji nowej próbki, analizując jej podobieństwo do k najbliższych sąsiadów w przestrzeni cech. Szczegółowy opis teoretyczny algorytmu został przedstawiony w rozdziale 2.1.3.

Mechanizm klasyfikacji nowej próbki x opiera się na zasadzie głosowania większościowego wśród k najbliższych sąsiadów:

$$\hat{y} = \text{mode}\{y_i : x_i \in N_k(x)\} \quad (5.4)$$

gdzie $N_k(x)$ oznacza zbiór k najbliższych sąsiadów próbki x wyznaczonych według przyjętej metryki odległości, najczęściej euklidesowej.

Wybór algorytmu k-NN do niniejszego studium przypadku podyktowany został kilkoma względami praktycznymi i metodologicznymi. Po pierwsze, algorytm cechuje się prostotą implementacji i łatwością interpretacji, decyzje klasyfikacyjne są intuicyjne i można je łatwo wyjaśnić, co jest istotne w kontekście przemysłowym. Po drugie, metoda wykazuje skuteczność w rozpoznawaniu wzorców lokalnych, co jest szczególnie przydatne przy wykrywaniu lokalnych anomalii tekstury powierzchni odlewów. Po trzecie, jako metoda nieparametryczna, k-NN nie wymaga przyjmowania założeń o rozkładzie danych, co jest zaletą w przypadku danych rzeczywistych, których struktura nie zawsze jest odgórnie znana. Dodatkowo algorytm służy jako punkt odniesienia dla bardziej złożonych metod, stanowiąc tzw. baseline do porównań. Wreszcie, zastosowania k-NN w dziedzinie wizji komputerowej, w tym w detekcji defektów powierzchniowych, są dobrze udokumentowane w literaturze.

W przeprowadzonych eksperymentach zastosowano następującą konfigurację hiperparametrów. Parametr `n_neighbors` ustalono na 5, co stanowi rozsądny kompromis między wrażliwością na szum w danych a nadmiernym wygładzaniem granic decyzyjnych. Jako metrykę odległości w przestrzeni cech przyjęto odległość euklidesową, która jest naturalnym wyborem dla danych liczbowych. Wszystkim sąsiadom przypisano jednakową wagę w procesie głosowania. W celu przyspieszenia obliczeń wykorzystano równolegle przetwarzanie na wszystkich dostępnych rdzeniach procesora.

Należy jednak zwrócić uwagę na pewne ograniczenia algorytmu k-NN. Głównym problemem jest wysoka złożoność obliczeniowa w fazie predykcji. W kontekście danych obrazowych o wysokiej wymiarowości (w niniejszym przypadku 262,144 wymiary) może to prowadzić do znacznego spowolnienia procesu klasyfikacji. Dodatkowo algorytm jest wrażliwy na tzw. przekleństwo wymiarowości, które polega na tym, że w przestrzeniach o bardzo wysokiej wymiarowości pojęcie "odległości" traci swoją intuicyjną interpretację, a wszystkie punkty stają się w pewnym sensie jednakowo odległe od siebie. Wreszcie, algorytm wymaga utrzymywania w pamięci pełnego zbioru treningowego, co może być problematyczne w przypadku bardzo dużych zbiorów danych.

5.6.2 Drzewo Decyzyjne (Decision Tree)

Drzewo decyzyjne stanowi hierarchiczną strukturę reprezentującą sekwencję decyzji binarnych, które prowadzą do ostatecznej klasyfikacji próbki. Algorytm CART (*Classification and Regression Trees*) konstruuje drzewo poprzez rekurencyjny podział przestrzeni cech, dając do maksymalizacji czystości powstających węzłów. Czystość ta jest najczęściej mierzona za pomocą indeksu Giniego lub entropii. Teoretyczne podstawy działania drzew decyzyjnych zostały szczegółowo omówione w rozdziale 2.1.4.

W niniejszej implementacji wykorzystano kryterium podziału oparte na indeksie Giniego, zdefiniowane następująco:

$$\text{Gini}(S) = 1 - \sum_{i=1}^C p_i^2 \quad (5.5)$$

gdzie p_i oznacza proporcję próbek należących do klasy i w analizowanym zbiorze S , a C to liczba klas.

Decyzja o włączeniu drzew decyzyjnych do niniejszego studium przypadku wynika z szeregu zalet tej metody. Przede wszystkim, drzewa decyzyjne charakteryzują się wysoką interpretowalnością, struktura drzewa umożliwia bezpośrednią wizualizację procesu decyzyjnego, co jest niezwykle istotne w zastosowaniach przemysłowych, gdzie zrozumienie przyczyn danej decyzji może być równie ważne jak sama decyzja. Kolejną zaletą jest automatyczna selekcja cech, algorytm samodzielnie identyfikuje najważniejsze cechy bez konieczności ręcznego preprocessingu czy inżynierii cech. Drzewa są również odporne na obecność nieistotnych cech, które są po prostu ignorowane podczas budowy struktury. Dodatkowo algorytm charakteryzuje się stosunkowo krótkim czasem treningu oraz szybką predykcją o złożoności logarytmicznej. Wreszcie, drzewa decyzyjne znajdują szerokie zastosowanie w przemysłowych systemach kontroli jakości właśnie ze względu na swoją interpretowalność i efektywność.

W przeprowadzonych eksperymentach zastosowano następującą konfigurację hiperparametrów. Maksymalną głębokość drzewa ograniczono do 10 poziomów. To ograniczenie stanowi mechanizm regularyzacji, który zapobiega nadmiernemu dopasowaniu modelu do szczegółów zbioru treningowego. Wartość 10 zapewnia wystarczającą ekspresję złożonych wzorców przy jednoczesnym zachowaniu zdolności do generalizacji na nowych danych. Jako miarę nieczystości węzłów wybrano indeks Giniego, będący standardowym wyborem w problemach klasyfikacyjnych. Parametr `random_state` ustawiono na 42 w celu zapewnienia reprodukowalności wyników eksperymentów.

Mimo licznych zalet, drzewa decyzyjne posiadają również ograniczenia. Bez odpowiedniej regularyzacji wykazują silną tendencję do przeuczenia, szczególnie gdy dozwolona jest duża głębokość drzewa. Charakteryzują się również niestabilnością, relatywnie małe zmiany w zbiorze treningowym mogą prowadzić do powstania drastycznie różnych struktur drzewa. Ponadto pojedyncze drzewa decyzyjne mogą mieć trudności z modelowaniem skomplikowanych nieliniowych zależności, które wymagają wielu rozgałęzień i poziomów w strukturze.

5.6.3 Las Losowy (Random Forest)

Random Forest reprezentuje zaawansowane podejście z kategorii metod uczenia zespołowego, które łączy predykcje wielu niezależnych drzew decyzyjnych w celu uzyskania bardziej stabilnego i dokładnego modelu końcowego. Algorytm wykorzystuje technikę baggingu, która polega na trenowaniu każdego drzewa na innym losowym podzbiorze danych z powtórzeniami. Dodatkowo wprowadza losową selekcję podzbiorów cech rozważanych przy każdym podziale węzła, co zwiększa różnorodność poszczególnych drzew i redukuje zarówno wariancję modelu, jak i ryzyko przeuczenia. Teoretyczne podstawy metody zostały szczegółowo omówione w rozdziale 2.1.5.

Końcowa predykcja klasy dla nowej próbki jest wyznaczana poprzez głosowanie większościowe spośród wszystkich drzew w lesie:

$$\hat{y} = \text{mode}\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_T\} \quad (5.6)$$

gdzie \hat{y}_t oznacza predykcję t -tego drzewa, a T to całkowita liczba drzew w lesie.

Decyzja o włączeniu algorytmu Random Forest do niniejszego studium przypadku jest uzasadniona wieloma czynnikami. Przede wszystkim, metoda ta wykazuje wybitną skuteczność empiryczną i często osiąga najlepsze wyniki bez konieczności szczególnego dostajania hiperparametrów, co czyni ją atrakcyjną w zastosowaniach praktycznych. Agre-

gacja predykcji wielu modeli skutecznie redukuje problem przeuczenia, charakterystyczny dla pojedynczych drzew decyzyjnych, zwiększać zdolność do generalizacji. Losowość wprowadzona na różnych etapach algorytmu zwiększa jego odporność na szum w danych. Istotną zaletą jest również możliwość automatycznego obliczenia rankingu ważności cech, co wspomaga interpretację działania modelu. Random Forest znajduje liczne udokumentowane zastosowania w dziedzinie wizji komputerowej, w tym w detekcji defektów, klasyfikacji tekstu oraz analizie obrazów medycznych. Dodatkowo, algorytm naturalnie wspiera przetwarzanie równolegle. Poszczególne drzewa mogą być trenowane niezależnie na wielu rdzeniach procesora, co znacząco przyśpiesza proces uczenia.

W przeprowadzonych eksperymentach przyjęto następującą konfigurację hiperparametrów. Liczbę drzew w lesie ustalono na 100 (`n_estimators = 100`). Jest to wartość kompromisowa, większa liczba drzew zwiększa stabilność predykcji, ale wydłuża czas treningu i predykcji. W praktyce 100 drzew stanowi dobry balans między wydajnością modelu a kosztami obliczeniowymi. Maksymalną głębokość pojedynczego drzewa ograniczono do 10 poziomów (`max_depth = 10`), co kontroluje złożoność poszczególnych bazowych estymatorów i zapobiega ich nadmiernemu rozrastaniu się. Parametr określający liczbę cech rozważanych przy każdym podziale ustawiono na `max_features = 'sqrt'`, co oznacza wybór \sqrt{d} losowych cech spośród d dostępnych wymiarów. Taki wybór jest rekomendowany w literaturze dla problemów klasyfikacyjnych. Parametr `random_state` ustalono na 42 dla zapewnienia reprodukowalności wyników. Wykorzystano również równoległy trening drzew na wszystkich dostępnych rdzeniach procesora.

Metoda Random Forest wykazuje szereg istotnych zalet w porównaniu z pojedynczymi modelami. Charakteryzuje się wysoką dokładnością predykcji przy minimalnej potrzebie dostrajania hiperparametrów. Algorytm posiada wbudowany mechanizm walidacji nazywany błędem out-of-bag, który pozwala na oszacowanie błędu generalizacji bez potrzeby tworzenia osobnego zbioru walidacyjnego. Dzięki agregacji wielu modeli znacząco zmniejsza się ryzyko przeuczenia w porównaniu z pojedynczym drzewem decyzyjnym.

5.6.4 Maszyna Wektorów Nośnych (Support Vector Machine)

Maszyna wektorów nośnych reprezentuje klasę algorytmów uczenia opartych na koncepcji maksymalizacji marginesu. Fundamentalną ideą metody jest poszukiwanie optymalnej hiperpłaszczyzny, która nie tylko rozdziela klasy w przestrzeni cech, ale również maksymalizuje odległość od najbliższych punktów każdej z klas. Te kluczowe punkty, leżące najbliżej granicy decyzyjnej, nazywane są wektorami nośnymi i to one wyznaczają ostateczny kształt hiperpłaszczyzny separującej. Zaletą SVM jest możliwość radzenia sobie z danymi nieliniowo separowalnymi poprzez zastosowanie tzw. kernel trick - techniki pozwalającej na niejawne odwzorowanie danych w przestrzeń wyższych wymiarów, w której separacja liniowa staje się możliwa. Szczegółowe podstawy teoretyczne metody omówiono w rozdziale 2.1.6.

Zastosowanie SVM w kontekście niniejszego problemu klasyfikacji defektów jest uzasadnione z kilku powodów. Algorytm charakteryzuje się udokumentowaną skutecznością w przestrzeniach o wysokiej wymiarowości, w analizowanym przypadku mamy do czynienia z 262,144 wymiarami, co dla wielu metod stanowi poważne wyzwanie. SVM opiera się na solidnych podstawkach matematycznych wywodzących się z teorii uczenia statystycznego. Maksymalizacja marginesu działa jako naturalny mechanizm regularyzacji, ograniczając ryzyko przeuczenia modelu. Istotną zaletą praktyczną jest efektywność pamięciowa, wy-

trenowany model jest zdefiniowany wyłącznie przez wektory nośne, które zazwyczaj stanowią stosunkowo niewielki podzbiór danych treningowych. SVM znajduje także liczne udokumentowane zastosowania w dziedzinie wizji komputerowej, w tym w rozpoznawaniu wzorców i klasyfikacji obrazów.

Należy podkreślić istotny wymóg metodologiczny związany z zastosowaniem SVM, algorytm jest szczególnie wrażliwy na skalę cech, dlatego konieczne jest przeprowadzenie standaryzacji danych przed treningiem modelu. W przypadku pominięcia tego kroku, cechy o większych zakresach wartości mogłyby zdominować proces uczenia, prowadząc do suboptimalnych wyników.

5.6.5 Neural Network – Multi-Layer Perceptron (MLP)

Wielowarstwowy perceptron reprezentuje klasę sztucznych sieci neuronowych o architekturze feed-forward, w której informacja przepływa jednokierunkowo od warstwy wejściowej przez jedną lub więcej warstw ukrytych do warstwy wyjściowej. Proces uczenia sieci realizowany jest za pomocą algorytmu wstępnej propagacji błędu w połączeniu z metodami optymalizacji gradientowej. W każdej warstwie ukrytej następuje transformacja danych wejściowych poprzez zastosowanie operacji liniowej oraz nieliniowej funkcji aktywacji, co umożliwia sieci uczenie się złożonych, nieliniowych zależności między cechami a zmienną docelową. Szczegółowe podstawy teoretyczne architektury i mechanizmów uczenia sieci neuronowych zostały omówione w rozdziale 2.5.

Wybór sieci neuronowej typu MLP jako jednego z algorytmów w niniejszym studium jest podyktowany kilkoma istotnymi względami. Zgodnie z twierdzeniem o uniwersalnej aproksymacji, sieć neuronowa z odpowiednio dobraną architekturą może aproksymować dowolnie złożone funkcje, co czyni ją niezwykle elastycznym narzędziem modelowania. Kluczową zaletą jest automatyczna ekstrakcja cech, warstwy ukryte uczą się hierarchicznych reprezentacji danych, przekształcając surowe piksele w coraz bardziej abstrakcyjne i semantycznie znaczące cechy. W dziedzinie wizji komputerowej sieci neuronowe, szczególnie konwolucyjne sieci neuronowe (CNN), stanowią aktualny standard i osiągają wyniki przewyższające tradycyjne metody. Zastosowanie nieliniowych funkcji aktywacji, takich jak ReLU, umożliwia modelowanie skomplikowanych, nieliniowych granic decyzyjnych. Dodatkowo, architektura MLP oferuje naturalną ścieżkę rozwoju projektu w kierunku bardziej zaawansowanych, głębokich architektur sieciowych.

Przyjęta architektura sieci składa się z dwóch warstw ukrytych zawierających odpowiednio 128 i 64 neurony. Taka konfiguracja realizuje progresywną redukcję wymiarowości danych: od 262,144 wymiarów na wejściu, poprzez 128 neuronów w pierwszej warstwie ukrytej, 64 neurony w drugiej warstwie ukrytej, aż do 2 neuronów w warstwie wyjściowej odpowiadających klasom binarnym. Jako funkcję aktywacji w warstwach ukrytych zastosowano ReLU, która transformuje sygnał zgodnie z prostą regułą: wartości ujemne są zerowane, a dodatnie pozostają bez zmian. ReLU charakteryzuje się dwoma istotnymi zaletami: zapobiega problemowi zanikającego gradientu, który może występować w głębokich sieciach przy użyciu funkcji sigmoidalnych, oraz przyspiesza zbieżność procesu uczenia. Do optymalizacji wag sieci wykorzystano algorytm Adam (*Adaptive Moment Estimation*), który łączy zalety metod momentum i RMSprop, stosując adaptacyjne współczynniki uczenia dla każdego parametru osobno. Liczbę epok treningu ograniczono do 50, co stanowi kompromis między czasem obliczeń a ryzykiem przeuczenia, dalsza nauka mogłaby prowadzić do nadmiernego dopasowania do zbioru treningowego.

Podobnie jak w przypadku SVM, standaryzacja danych wejściowych ma kluczowe znaczenie dla stabilności i efektywności procesu uczenia. Zapewnia ona, że gradienty podczas propagacji wstecznej pozostają w rozsądny zakresie wartości, co przyspiesza zbieżność algorytmu optymalizacji. Należy jednak mieć świadomość pewnych ograniczeń metody. Trening sieci neuronowych jest procesem czasochłonnym, szczególnie dla danych wysokowymiarowych. Model ma charakter "czarnej skrzynki", w przeciwieństwie do drzew decyzyjnych, trudno jest bezpośrednio zrozumieć i wyjaśnić, w jaki sposób sieć dochodzi do konkretnych decyzji. Sieci neuronowe posiadają wiele hiperparametrów do dostrojenia, takich jak liczba warstw, liczba neuronów w każdej warstwie, typ funkcji aktywacji, algorytm optymalizacji czy współczynnik uczenia, co wymaga przeprowadzenia eksperymentów w celu znalezienia optymalnej konfiguracji. Bez odpowiedniej regularyzacji, takiej jak dropout czy weight decay, sieci są podatne na przeuczenie, szczególnie gdy stosunek liczby parametrów do liczby przykładów treningowych jest wysoki.

5.6.6 Logistic Regression (Regresja Logistyczna)

Regresja logistyczna, pomimo sugerującej nazwy, jest w istocie algorytmem klasyfikacji, który modeluje prawdopodobieństwo przynależności próbki do danej klasy. Metoda opiera się na liniowej kombinacji cech wejściowych, która następnie jest przekształcana przez funkcję logistyczną w wartość z przedziału $[0, 1]$ interpretowaną jako prawdopodobieństwo. Algorytm uczy się optymalnych wag poprzez minimalizację funkcji straty opartej na entropii krzyżowej. Teoretyczne podstawy metody zostały omówione w rozdziale 2.1.2.

Wybór regresji logistycznej jako jednego z badanych algorytmów wynika przede wszystkim z jej roli jako modelu bazowego. Jest to najprostszy algorytm probabilistyczny, którego wyniki stanowią naturalny punkt odniesienia dla oceny bardziej złożonych metod, jeśli zaawansowany model nie przewyższa regresji logistycznej, może to sugerować, że albo problem jest z natury liniowy, albo bardziej skomplikowany model wymaga lepszego dostrojenia. Metoda charakteryzuje się ekstremalną szybkością treningu i predykcji oraz niskimi wymaganiami obliczeniowymi, co czyni ją idealnym kandydatem do aplikacji wymagających działania w czasie rzeczywistym. Dodatkową zaletą jest interpretowalność, nauczone wagi bezpośrednio wskazują wpływ poszczególnych cech na decyzję klasyczną. W przeciwieństwie do metod zwracających jedynie etykiety klas, regresja logistyczna generuje wyjście probabilistyczne, co może być wartościowe w zastosowaniach przemysłowych wymagających oszacowania pewności decyzji.

Konfiguracja hiperparametrów została dostosowana do specyfiki problemu wysokowymiarowego. Maksymalną liczbę iteracji optymalizatora ustalono na 1000, co zapewnia zbieżność algorytmu nawet dla danych o dużej liczbie wymiarów. Jako optymalizator wybrano L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno), który jest efektywny dla problemów o średniej wielkości i naturalnie obsługuje regularizację L2. Podobnie jak w przypadku poprzednich algorytmów, niezbędna jest standaryzacja danych, która poprawia zbieżność numeryczną i stabilność procesu uczenia.

Do głównych zalet regresji logistycznej należą niezwykle krótki czas treningu i predykcji, bardzo niskie zużycie pamięci oraz zdolność do osiągania dobrych wyników w problemach, w których klasy są w przybliżeniu liniowo separowalne. Metoda może służyć jako szybka diagnoza charakteru problemu, jeśli osiąga wysoką dokładność, sugeruje to, że dane są relatywnie proste do rozdzielenia, co może wpływać na decyzje dotyczące dalszego modelowania.

5.7 Implementacja modelu i przebieg eksperymentu

Niniejsza sekcja przedstawia szczegółowy opis implementacji technicznej projektu, obejmujący specyfikację środowiska programistycznego, wykorzystane narzędzia, procedury eksperymentalne oraz metodologię walidacji modeli.

5.7.1 Środowisko programistyczne

Język programowania: Python 3.12.1

Środowisko deweloperskie: Jupyter Notebook / VS Code

Platforma obliczeniowa: Lokalny komputer z systemem Windows 10, wyposażony w procesor AMD Ryzen 5 7600 6-Core Processor, 32 GB RAM oraz kartę graficzną AMD Radeon RX 7700 XT.

5.7.2 Wykorzystane biblioteki

W realizacji niniejszego projektu wykorzystano szereg bibliotek i narzędzi języka Python, które stanowią standardowe wyposażenie w projektach uczenia maszynowego i przetwarzania obrazów.

NumPy: Służy do operacji na tablicach wielowymiarowych, jest wydajny do obliczeń numerycznych i stanowi podstawę dla innych bibliotek naukowych.

Pandas: Umożliwia pracę ze strukturami danych typu DataFrame, pozwala na analizę i agregację wyników eksperymentów oraz eksport danych do formatu CSV.

OpenCV: Biblioteka służąca do przetwarzania obrazów, wykorzystywana do wczytywania obrazów, konwersji do skali szarości, zmiany rozmiaru oraz przeprowadzania operacji morfologicznych i filtrowania.

Pillow: Stanowi alternatywne narzędzie do wczytywania i wizualizacji obrazów, obsługuje różnorodne formaty obrazowe takie jak JPEG czy PNG.

Scikit-learn: Główna biblioteka uczenia maszynowego projektu, dostarcza implementacje algorytmów klasyfikacji (k-NN, drzewa decyzyjne, lasy losowe, SVM, sieci neuronowe, regresja logistyczna), narzędzia do preprocessingu danych (standaryzacja, podział na zbiory) oraz metryki ewaluacyjne (dokładność, precyza, czułość, F1-score, macierz pomyłek).

Matplotlib: Służy do tworzenia wykresów i diagramów, wizualizacji macierzy pomyłek oraz eksportu grafik do formatu PNG.

Seaborn: Umożliwia tworzenie zaawansowanych wizualizacji statystycznych, w tym heatmap dla macierzy pomyłek, oferując estetyczne style wykresów.

Biblioteki pomocnicze: Wykorzystano również Pathlib do zarządzania ścieżkami plików, Time do pomiaru czasu wykonania algorytmów oraz Warnings do filtrowania ostrzeżeń systemowych.

5.7.3 Schemat procesu uczenia

Proces uczenia modeli został zorganizowany zgodnie z ustaloną metodologią, obejmującą pięć głównych etapów realizowanych sekwencyjnie.

Etap 1: Wczytywanie i preprocessing danych. Obrazy w formacie JPEG o rozdzielcości 512×512 pikseli zostały wczytane z wykorzystaniem biblioteki OpenCV. W trakcie wczytywania zastosowano konwersję do skali szarości, co zredukowało wymiarowość danych z trzech kanałów kolorów do jednego kanału intensywności. Następnie każdy obraz został przekształcony w jednowymiarowy wektor o długości 262,144 elementów poprzez operację spłaszczania. Wartości pikseli zostały znormalizowane do przedziału $[0, 1]$ poprzez dzielenie przez 255.0. Po przetworzeniu wszystkich obrazów, dane zostały połączone i losowo przetasowane w celu zapewnienia losowości w kolejnych etapach.

Etap 2: Podział danych. Pełny zbiór danych został podzielony na zbiór treningowy i testowy w proporcji 80:20 z wykorzystaniem funkcji `train_test_split` z biblioteki Scikit-learn. Zastosowano stratyfikację względem zmiennej docelowej oraz ustalono parametr `random_state = 42` w celu zapewnienia reprodukowalności wyników. W efekcie otrzymano 1040 próbek w zbiorze treningowym oraz 260 próbek w zbiorze testowym, z zachowaniem oryginalnych proporcji klas w obu zbiorach.

Etap 3: Standaryzacja danych. Dla algorytmów wrażliwych na skalę cech (SVM, MLP, regresja logistyczna, k-NN) przeprowadzono dodatkowy krok standaryzacji Z-score z wykorzystaniem klasy `StandardScaler`. Scaler został dopasowany wyłącznie na zbiorze treningowym, a następnie zastosowany zarówno do danych treningowych, jak i testowych. Algorytmy oparte na drzewach decyzyjnych (drzewo decyzyjne, las losowy) zostały wytrenowane bezpośrednio na danych znormalizowanych, bez dodatkowej standaryzacji.

Etap 4: Trening modeli. Każdy z sześciu algorytmów klasyfikacji został wytrenowany sekwencyjnie zgodnie z następującą kolejnością: k-najbliższych sąsiadów, drzewo decyzyjne, las losowy, maszyna wektorów nośnych, sieć neuronowa oraz regresja logistyczna. Dla każdego modelu zmierzono czas treningu, rejestrując moment rozpoczęcia i zakończenia procesu uczenia. Model został zainicjalizowany z ustalonymi hiperparametrami, następnie wytrenowany na odpowiednim zbiorze treningowym, a po zakończeniu treningu wykorzystany do predykcji etykiet dla zbioru testowego.

Etap 5: Ewaluacja i porównanie. Dla każdego modelu obliczono zestaw metryk ewaluacyjnych: dokładność (accuracy), precyzyę (precision), czułość (recall), wynik F1 (F1-score) oraz czas treningu. Wyniki zostały zagregowane w strukturze DataFrame biblioteki Pandas, co umożliwiło łatwe porównanie wydajności wszystkich algorytmów. Dodatkowo wygenerowano wizualizacje w postaci macierzy pomyłek oraz wykresów porównawczych metryk. Końcowe wyniki zostały wyeksportowane do formatu CSV w celu dalszej analizy i dokumentacji.

5.7.4 Walidacja modeli

W niniejszym projekcie zastosowano metodę hold-out validation z podziałem 80/20 oraz stratyfikacją klas. Wybór tego podejścia, pomimo istnienia bardziej zaawansowanych technik walidacji, jest uzasadniony wystarczającą wielkością zbioru danych (ponad tysiąc próbek), efektywnością obliczeniową oraz prostotą implementacji zapewniającą reprodukowalność eksperymentu.

Parametr `stratify=y` w funkcji podziału zapewnia, że proporcje klas defekt/OK są identyczne w zbiorach treningowym i testowym, eliminując bias wynikający z nierównomiernego podziału i zwiększać wiarygodność metryk testowych. Po podziale danych przeprowadzono weryfikację rozkładu klas w celu potwierdzenia poprawności stratyfikacji.

W bardziej zaawansowanych implementacjach można rozważyć zastosowanie walidacji krzyżowej k-fold, która lepiej ocenia generalizację modelu wykorzystując wszystkie dane do treningu i walidacji, lub stratified k-fold łączącego walidację krzyżową ze stratyfikacją klas w każdym foldzie.

5.7.5 Strojenie hiperparametrów

W niniejszym projekcie zastosowano ręczny dobór hiperparametrów oparty na wartościach domyślnych z dokumentacji Scikit-learn, najlepszych praktykach z literatury oraz empirycznej wiedzy o problemie. Dla k-NN wybrano `n_neighbors = 5` jako klasyczną wartość zapewniającą balans między wygładzaniem a wrażliwością. Dla drzew decyzyjnych i lasu losowego ustalone `max_depth = 10` w celu ograniczenia przeuczenia przy zachowaniu ekspresji modelu, a dla lasu losowego dodatkowo `n_estimators = 100` jako standard przemysłowy. SVM skonfigurowano z jądrem `rbf` dla modelowania nielinowych granic, parametrem regularyzacji `C = 1.0` oraz adaptacyjnym `gamma = 'scale'`. Dla MLP przyjęto architekturę `(128, 64)` realizującą progresywną redukcję wymiarowości, funkcję aktywacji ReLU, optymalizator `adam` oraz `max_iter = 50`. Dla regresji logistycznej ustalonio `max_iter = 1000` zapewniając zbieżność dla wysokowymiarowego problemu.

Nie zastosowano zaawansowanych metod strojenia takich jak Grid Search, Randomized Search czy Bayesian Optimization ze względu na cel badawczy projektu, który koncentruje się na porównaniu różnych modeli uczenia maszynowego, a nie na maksymalizacji wydajności pojedynczego modelu. Wartości domyślne Scikit-learn są oparte na badaniach empirycznych i stanowią rozsądny punkt wyjścia, a koszt obliczeniowy zaawansowanego tuningu zwiększyłby czas eksperymentu wielokrotnie przy zachowaniu przejrzystości i reprodukowalności wyników.

5.7.6 Pomiar wydajności i metryki czasowe

Dla każdego modelu przeprowadzono precyzyjny pomiar czasu treningu z wykorzystaniem biblioteki Time, rejestrując moment rozpoczęcia i zakończenia procesu uczenia. Metryka czasowa umożliwia porównanie efektywności obliczeniowej algorytmów, ocenę ich praktycznej użyteczności w scenariuszach produkcyjnych oraz identyfikację kompromisu między dokładnością a czasem wykonania.

Jako metryki ewaluacyjne zastosowano standardowy zestaw wskaźników klasyfikacji binarnej: dokładność (accuracy) mierzącą ogólną poprawność predykcji, precyzję (preci-

sion) wskazującą odsetek poprawnie zidentyfikowanych defektów, czułość (recall) określającą zdolność do wykrywania wszystkich rzeczywistych defektów, oraz wynik F1 (F1-score) będący harmoniczną średnią precyzji i czułości. Wszystkie metryki zostały zagregowane w strukturze DataFrame umożliwiającej systematyczne porównanie wydajności modeli.

5.7.7 Reprodukowalność eksperymentu

W celu zapewnienia pełnej reprodukowalności eksperymentu zastosowano szereg mechanizmów kontrolujących losowość procesu. Dla wszystkich operacji wykorzystujących generatory liczb pseudolosowych ustalono parametr `random_state = 42`, obejmujący podział danych, inicjalizację modeli oraz procedury bootstrapingu. Równolegle przetwarzanie (`n_jobs = -1`) zostało skonfigurowane z kontrolowanym ziarnem generatora.

Środowisko obliczeniowe zostało udokumentowane poprzez zapis wersji wszystkich wykorzystanych bibliotek, umożliwiając odtworzenie identycznego środowiska programistycznego. Pełny kod eksperymentu wraz z komentarzami został zachowany w notebooku Jupyter, a wszystkie wyniki numeryczne wyeksportowano do formatu CSV, co pozwala na weryfikację i dalszą analizę uzyskanych rezultatów.

5.8 Wyniki i ich analiza

5.8.1 Wprowadzenie do analizy wyników

Sekcja prezentuje szczegółową ocenę rezultatów eksperymentów przeprowadzonych w ramach badania klasyfikacji wad w odlewach metalowych. Analiza opiera się na porównaniu wydajności sześciu różnych algorytmów uczenia maszynowego z zastosowaniem uznanych metryk klasyfikacji binarnej. Wyniki są interpretowane w kontekście potencjalnego zastosowania w warunkach przemysłowych, ze szczególnym uwzględnieniem aspektów praktycznych i ekonomicznych.

5.8.2 Charakterystyka zastosowanych metryk ewaluacyjnych

Kompleksowa ocena jakości modeli klasyfikacyjnych wymaga zastosowania wieloaspektowego zestawu miar statystycznych. W przeprowadzonym badaniu wykorzystano następujące metryki:

Dokładność (Accuracy)

Miara ta definiuje proporcję poprawnie zaklasyfikowanych próbek w całkowitej liczbie obserwacji:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

gdzie poszczególne symbole oznaczają:

- TP (True Positive) - poprawnie zidentyfikowane przypadki z defektem
- TN (True Negative) - poprawnie rozpoznane przypadki bez defektu

- FP (False Positive) - błędna klasyfikacja jako defekt
- FN (False Negative) - nieroznalezony defekt

Interpretacja produkcyjna: Wskaźnik ten odzwierciedla całkowitą efektywność systemu, jednakże w sytuacji niezrównoważonych zbiorów danych może prowadzić do mylących wniosków.

Precyzja (Precision)

Wskaźnik określający stosunek prawidłowo wykrytych defektów do wszystkich przypadków oznaczonych jako wadliwe:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Znaczenie w produkcji: Wysoka wartość tego wskaźnika świadczy o wysokim prawdopodobieństwie, że produkt oznaczony jako wadliwy rzeczywiście posiada defekt. Niska precyzja generuje wzrost kosztów związanych z dodatkową kontrolą jakości, niepotrzebne odrzucanie produktów zgodnych ze specyfikacją oraz spadek efektywności całego procesu produkcyjnego.

Czułość (Recall)

Miara określająca proporcję rzeczywiście wykrytych defektów spośród wszystkich faktycznie wadliwych produktów:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Znaczenie w produkcji: Wysoka czułość oznacza, że system identyfikuje większość rzeczywistych wad. Niska wartość tego wskaźnika niesie poważne konsekwencje, ponieważ wadliwe produkty trafiają do odbiorców końcowych, występuje zwiększone ryzyko roszczeń gwarancyjnych i utraty reputacji oraz pojawiają się potencjalne implikacje prawne oraz znaczące straty finansowe.

Miara F1 (F1-Score)

Wskaźnik stanowiący średnią harmoniczną precyzji i czułości:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Znaczenie w produkcji: Metryka ta dostarcza zbalansowanej oceny modelu, szczególnie istotnej w sytuacji, gdy koszty błędów typu FP i FN są porównywalne.

Macierz błędów (Confusion Matrix)

Macierz błędów stanowi narzędzie wizualizacji szczegółowych informacji dotyczących procesu klasyfikacji:

Znaczenie w produkcji: Umożliwia identyfikację dominującego typu błędów w systemie, co pozwala na dostosowanie strategii kontroli jakości do specyfiki problemu.

	Predykcja: OK	Predykcja: DEFEKT
Rzeczywistość: OK	TN	FP
Rzeczywistość: DEFEKT	FN	TP

5.8.3 Wyniki badań empirycznych

Zestawienie tabelaryczne uzyskanych rezultatów

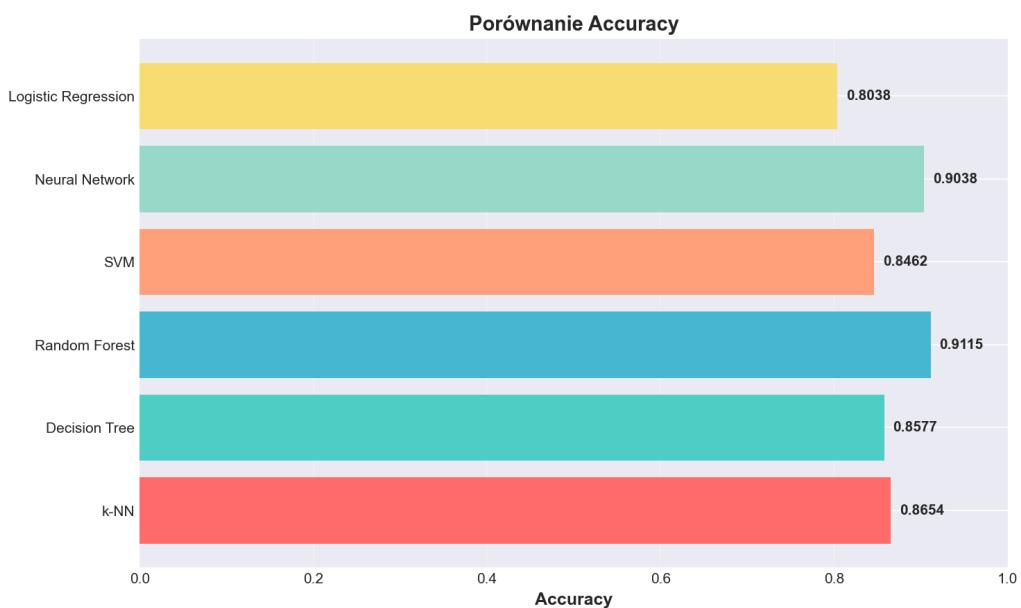
Przeprowadzone eksperymenty z zastosowaniem sześciu algorytmów uczenia maszynowego przyniosły następujące rezultaty:

Algorytm	Accuracy	Precision	Recall	F1-Score	Czas [s]
Random Forest	91.15%	94.04%	91.03%	92.51%	2.31
Neural Network	90.38%	92.81%	91.03%	91.91%	196.99
k-NN	86.54%	89.54%	87.82%	88.67%	0.30
Decision Tree	85.77%	87.90%	88.46%	88.18%	115.46
SVM	84.62%	90.28%	83.33%	86.67%	46.30
Logistic Regression	80.38%	88.89%	76.92%	82.47%	7.06

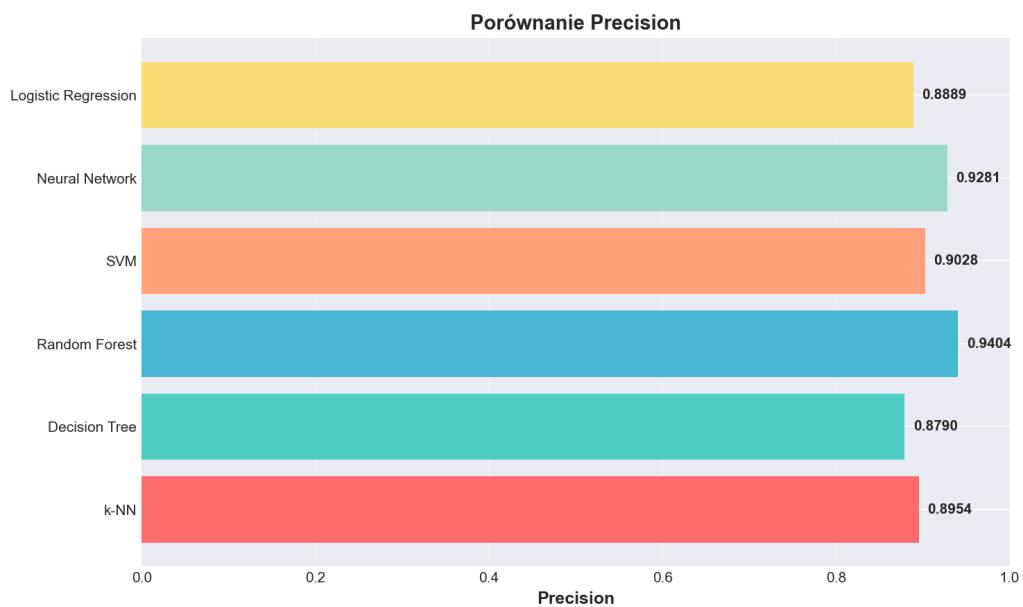
Tabela 5.1: Zestawienie wyników dla wszystkich testowanych algorytmów

Uwaga: Najlepsze rezultaty w poszczególnych kategoriach wyróżniono pogrubieniem.

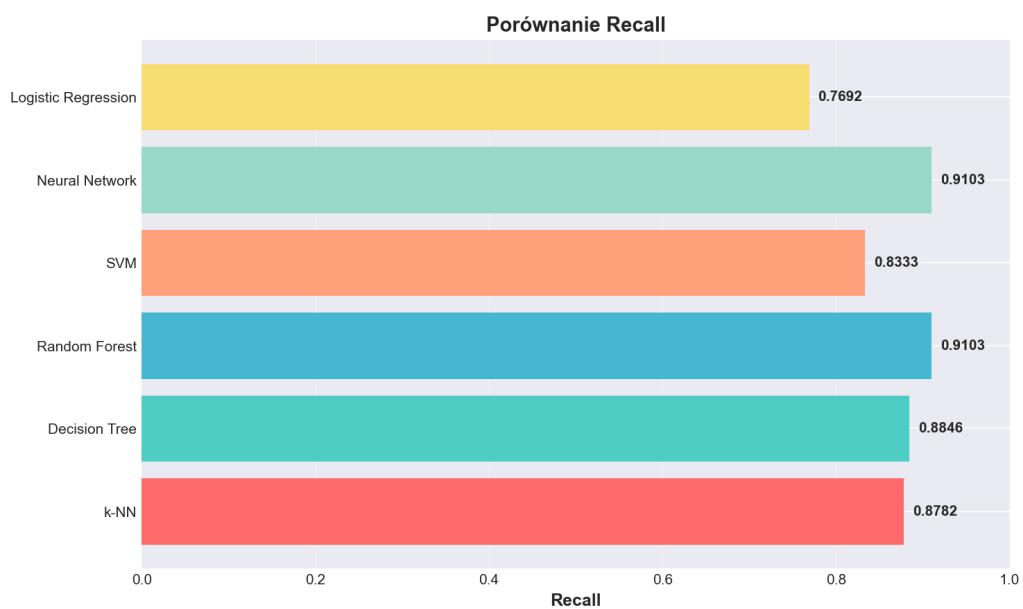
W celu lepszej wizualizacji uzyskanych wyników, na rysunkach 5.3 - 5.7 przedstawiono porównanie wydajności wszystkich badanych algorytmów w formie wykresów słupkowych dla poszczególnych metryk.



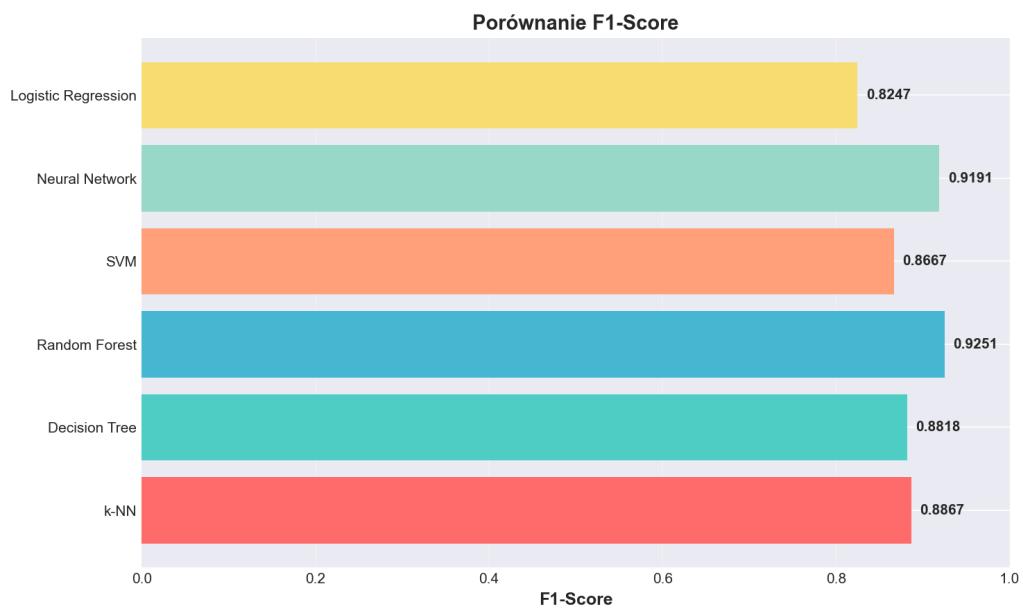
Rysunek 5.3: Porównanie dokładności (Accuracy) dla wszystkich algorytmów



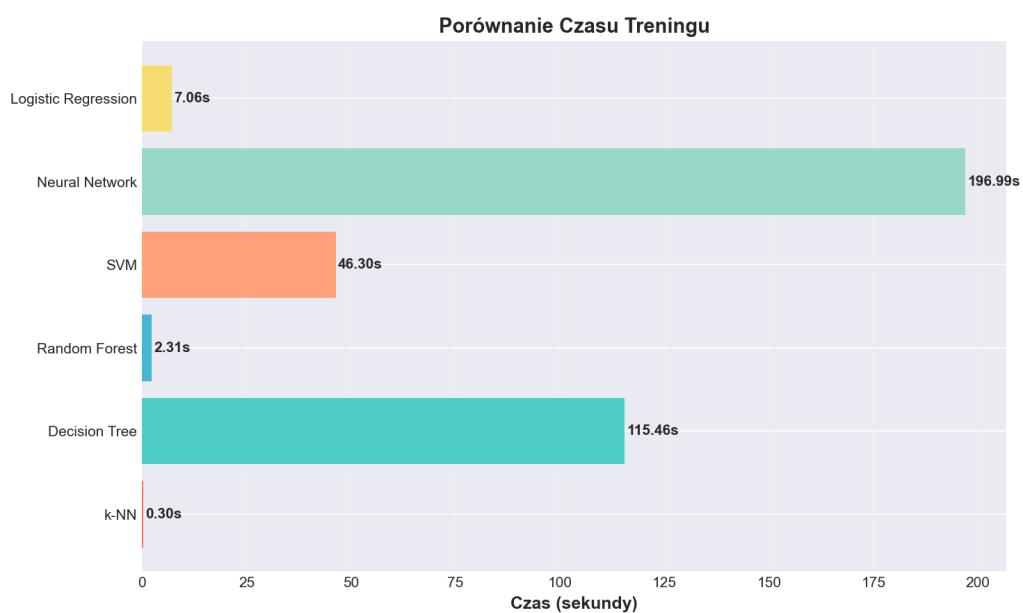
Rysunek 5.4: Porównanie precyzji (Precision) dla wszystkich algorytmów



Rysunek 5.5: Porównanie czułości (Recall) dla wszystkich algorytmów



Rysunek 5.6: Porównanie miary F1-Score dla wszystkich algorytmów



Rysunek 5.7: Porównanie czasu treningu dla wszystkich algorytmów

Z przedstawionych wykresów wyraźnie widać przewagę algorytmu Random Forest w metrykach Accuracy, Precision i F1-Score. Warto zauważyć, że Random Forest i Neural Network osiągają identyczny poziom Recall (91.03%). Znaczące różnice obserwuje się w czasie treningu, gdzie k-NN charakteryzuje się najkrótszym czasem uczenia, natomiast Neural Network wymaga znaczco więcej czasu obliczeniowego.

Szczegółowa analiza modelu o najwyższej skuteczności

Algorytm Random Forest osiągnął najkorzystniejsze wyniki w trzech metrykach oraz taki sam wynik jak Neural Network w czwartej:

- **Accuracy: 91.15%** - najwyższa ogólna trafność klasyfikacji
- **Precision: 94.04%** - najniższy poziom fałszywych alarmów
- **F1-Score: 92.51%** - optymalna równowaga między precyją a czułością
- **Recall: 91.03%** - identyczny wynik jak Neural Network, najwyższy wśród wszystkich algorytmów

Interpretacja w kontekście produkcyjnym:

- Spośród 100 odlewów oznaczonych jako wadliwe, około 94 faktycznie będzie posiadało defekty
- System poprawnie klasyfikuje 91.15% wszystkich analizowanych wyrobów
- Model identyfikuje 91.03% rzeczywistych przypadków wadliwych

Analiza macierzy błędów

Dla modelu Random Forest na zbiorze walidacyjnym (260 próbek) uzyskano następującą macierz błędów:

Macierz błędów:

		Przewidywane	
		OK	DEFEKT
Rzeczywiste	OK	[114 9]	
	DEFEKT	[14 123]	

Interpretacja poszczególnych wartości:

- **True Negatives (114):** Produkty bez defektów poprawnie zidentyfikowane
- **False Positives (9):** Produkty zgodne ze specyfikacją błędnie oznaczone jako wadliwe (7.3% przypadków OK)
- **False Negatives (14):** Nierozpoznane defekty (10.2% produktów wadliwych)
- **True Positives (123):** Prawidłowo wykryte defekty (89.8% produktów wadliwych)

Implikacje dla procesu produkcyjnego:

- **FP (9 przypadków):** Wymaga dodatkowej inspekcji manualnej 9 produktów zgodnych, co oznacza koszt robocizny
- **FN (14 przypadków):** 14 wadliwych produktów może przejść do kolejnych etapów procesu i spowodować wysokie ryzyko kosztowe

5.8.4 Analiza porównawcza alternatywnych metod uczenia maszynowego

Porównanie Neural Network z Random Forest

Sieć neuronowa osiągnęła wynik F1-Score na poziomie 91.91%, co stanowi nieznaczne obniżenie w porównaniu do Random Forest o 0.6 punktu procentowego. Należy jednak zwrócić uwagę na znaczące różnice w efektywności czasowej, proces treningu sieci neuronowej jest 85-krotnie dłuższy. Dodatkowo, sieci neuronowe charakteryzują się większym zapotrzebowaniem na dane treningowe dla osiągnięcia optymalnej wydajności, choć posiadają potencjał do dalszej optymalizacji poprzez dostrojenie architektury sieci.

Z kolei Random Forest uzyskał najwyższy wynik F1-Score w ramach eksperymentu wynoszący 92.51%. Algorytm ten charakteryzuje się relatywnie krótkim czasem treningu, co czyni go praktycznym dla zastosowań produkcyjnych. Istotną zaletą jest również łatwiejsza interpretacja wyników poprzez analizę ważności cech oraz większa stabilność bez konieczności zaawansowanego dostrajania hiperparametrów. W kontekście analizowanego zbioru danych Random Forest stanowi bardziej racjonalny wybór ze względu na optymalne połączenie wydajności, czasu uczenia i prostoty implementacji.

Klasyfikacja algorytmów według kryterium efektywności czasowej

Algorytm	F1-Score	Czas [s]	F1/Czas
k-NN	88.67%	0.30	295.57
Random Forest	92.51%	2.31	40.05
Logistic Regression	82.47%	7.06	11.68
SVM	86.67%	46.30	1.87
Decision Tree	88.18%	115.46	0.76
Neural Network	91.91%	196.99	0.47

Tabela 5.2: Ranking algorytmów według stosunku jakości do czasu treningu

Rysunek 5.7 ilustruje znaczące różnice w efektywności czasowej poszczególnych algorytmów. Najkrótszy czas treningu charakteryzuje algorytm k-NN (0.30s), który nie wymaga właściwego procesu uczenia, podczas gdy Neural Network wymaga najdłuższego czasu obliczeniowego wynoszącego prawie 197 sekund. Random Forest osiąga optymalny balans między jakością wyników a czasem treningu, co przekłada się na najkorzystniejszy stosunek F1-Score do czasu (40.05).

Wybór algorytmu powinien być uzależniony od specyfiki zastosowania. W kontekście badań i rozwoju, gdzie priorytetem jest najwyższa jakość klasyfikacji, rekomendowane są Neural Network lub Random Forest. Dla zastosowań produkcyjnych z ograniczeniami czasowymi bardziej odpowiednie będą k-NN lub Random Forest. Implementacja na urządzeniach brzegowych wymaga algorytmu charakteryzującego się najkrótszym czasem prze-

twarzania, co czyni k-NN optymalnym wyborem mimo nieco niższej jakości. Jednak w sytuacji, gdy poszukiwana jest optymalna równowaga między jakością a czasem, Random Forest pozostaje najlepszym rozwiązaniem dzięki najkorzystniejszemu stosunkowi wydajności do czasu treningu.

Analiza kompromisów między precyzją a czułością

Algorytm	Precision	Recall	Charakterystyka błędów
SVM	90.28%	83.33%	Więcej FN (pomija defekty)
Neural Network	92.81%	91.03%	Zrównoważone
Random Forest	94.04%	91.03%	Najmniej FP
Logistic Regression	88.89%	76.92%	Znacząca liczba FN

Tabela 5.3: Porównanie algorytmów pod kątem precision i recall

Analiza kompromisów między precyzją a czułością ujawnia istotne różnice w charakterze błędów popełnianych przez poszczególne algorytmy. SVM wykazuje stosunkowo wysoką precyzję (90.28%), jednak czułość na poziomie 83.33% wskazuje na tendencję do pomijania defektów (więcej błędów typu FN). Neural Network oraz Random Forest osiągają zrównoważone wyniki z identycznym poziomem czułości wynoszącym 91.03%, przy czym Random Forest charakteryzuje się najwyższą precyzją (94.04%), co oznacza najmniejszą liczbę fałszywych alarmów. Regresja logistyczna wykazuje znaczącą liczbę błędów typu FN, co przejawia się najniższą czułością spośród wszystkich badanych algorytmów (76.92%).

Wybór algorytmu powinien być zdeterminowany przez priorytety biznesowe konkretnego zastosowania. W sytuacji, gdy ważne jest minimalizowanie kosztownej ponownej inspekcji (minimalizacja FP), rekomendowany jest Random Forest z precyzją 94.04%. Dla zastosowań, gdzie krytyczne znaczenie ma bezpieczeństwo i konieczne jest wykrycie maksymalnej liczby defektów (minimalizacja FN), odpowiednie będą Neural Network lub Random Forest z czułością na poziomie 91.03%. Natomiast w przypadku, gdy priorytetem jest szybkie wdrożenie systemu, algorytm k-NN stanowi kompromisowe rozwiązanie z czasem treningu 0.30s i czułością 87.82%.

5.9 Podsumowanie wyników i ograniczenia rozwiązania

5.9.1 Mocne strony zaproponowanego podejścia

Przeprowadzone badanie wykazało szereg mocnych stron zaproponowanego rozwiązania opartego na algorytmach uczenia maszynowego. Algorytm Random Forest osiągnął wysoką skuteczność klasyfikacji na poziomie 91.15% accuracy oraz 94.04% precision, co stanowi znaczącą poprawę w porównaniu z tradycyjnymi metodami inspekcji wizualnej. Wszystkie testowane algorytmy przekroczyły próg 80% dokładności, co potwierdza zasadność zastosowania metod uczenia maszynowego w automatycznej kontroli jakości odlewów.

Istotną zaletą jest automatyzacja procesu ekstrakcji cech z surowych obrazów, eliminująca konieczność ręcznego definiowania reguł klasyfikacji. Random Forest jako model rekomendowany charakteryzuje się dodatkowo krótkim czasem treningu wynoszącym 2.31 sekundy, co umożliwia szybką adaptację do nowych danych oraz ponowne uczenie modelu

bez znaczącego obciążenia infrastruktury obliczeniowej. Kompleksowe porównanie sześciu różnych algorytmów pozwoliło na obiektywny wybór najlepszego rozwiązania oraz zrozumienie kompromisów między dokładnością a efektywnością czasową.

5.9.2 Ograniczenia rozwiązania

Pomimo osiągniętych wyników, należy zwrócić uwagę na istotne ograniczenia przeprowadzonego badania, które mogą wpływać na możliwość bezpośredniego wdrożenia rozwiązania w środowisku produkcyjnym.

Jakość i rozmiar danych

Podstawowym ograniczeniem jest stosunkowo mały rozmiar zbioru danych wynoszący 1,300 obrazów. W kontekście nowoczesnego uczenia maszynowego, szczególnie głębokiego uczenia, jest to zbiór niewystarczający do pełnej oceny zdolności generalizacji modelu. Małe zbiory danych zwiększały ryzyko przeuczenia oraz mogą nie odzwierciedlać pełnej zmienności warunków produkcyjnych, takich jak różne warunki oświetlenia, różnorodność typów defektów czy zmienność materiałów.

Dataset został zebrany w kontrolowanych warunkach laboratoryjnych z wykorzystaniem lustrzanki cyfrowej Canon EOS 1300D, co może nie w pełni odpowiadać rzeczywistym warunkom przemysłowym. Wszystkie obrazy charakteryzują się stałym rozmiarem 512×512 pikseli oraz są już odpowiednio wykadrowane, podczas gdy w warunkach produkcyjnych może występować znacznie większa zmienność jakości i parametrów akwizycji obrazu. Dodatkowo, klasyfikacja binarna (OK/DEFEKT) nie uwzględnia różnych typów i stopni nasilenia defektów, co w praktyce może być istotne dla procesu decyzyjnego.

Skalowalność rozwiązania

Rozwiązanie zostało przetestowane na stosunkowo niewielkim zbiorze danych, co rodzi pytania o jego skalowalność przy znacznie większych wolumenach produkcji. Czas predykcji wynoszący około 0.05 sekundy na obraz odpowiada przepustowości około 20 obrazów na sekundę, co może być niewystarczające dla linii produkcyjnych o wysokiej wydajności. Model został wytrenowany na konkretnym typie odlewów, co oznacza konieczność całkowitego przetrenowania dla każdego nowego produktu, bez możliwości wykorzystania mechanizmów transfer learning czy incremental learning.

Brak testów w rzeczywistych warunkach produkcyjnych uniemożliwia ocenę odporności systemu na zmienne warunki środowiskowe, takie jak wahania oświetlenia, zapylenie optyki czy wibracje maszyn. Nie przeprowadzono również analizy wymagań sprzętowych dla wdrożenia w skali przemysłowej, w tym zapotrzebowania na pamięć operacyjną, moc obliczeniową oraz infrastrukturę sieciową.

Ryzyko przeuczenia

Zastosowanie prostego podziału train-test (80/20) bez walidacji krzyżowej ogranicza pewność co do stabilności uzyskanych wyników. Brak k-fold cross-validation oznacza, że metryki mogą być zależne od konkretnego, losowego podziału danych. Wysokie wyniki na zbiorze testowym nie gwarantują podobnej wydajności na całkowicie niezależnych danych z innych okresów produkcji czy różnych partii materiału.

Decision Tree wykazał stosunkowo długi czas treningu (115.46s) przy niższej accuracy (85.77%), co może sugerować potencjalne przeuczenie. Brak prezentacji krzywych uczenia (learning curves) uniemożliwia diagnozę, czy modele wymagają więcej danych treningowych czy lepszej regularyzacji. Nie przedstawiono również szczegółowej analizy błędów, która pozwoliłaby zidentyfikować systematyczne wzorce w nieprawidłowych klasyfikacjach.

5.9.3 Wnioski końcowe

Przeprowadzone studium przypadku potwierdza, że algorytmy uczenia maszynowego, w szczególności Random Forest, stanowią obiecujące rozwiązanie dla automatyzacji kontroli jakości w przemyśle odlewniczym. Osiągnięte wyniki (91.15% accuracy, 94.04% precision) są satysfakcjonujące jako proof of concept i punkt wyjścia do dalszych prac rozwojowych.

Jednakże obecne rozwiązanie stanowi prototyp badawczy, który wymaga znaczących rozszerzeń i walidacji przed wdrożeniem produkcyjnym. Ograniczenia związane z rozmiarem i reprezentatywnością danych, brak testów w warunkach rzeczywistych oraz potrzeba rozbudowy infrastruktury monitorowania i ciągłego uczenia czynią z tego projektu solidną podstawę teoretyczną, wymagającą jednak dalszego rozwoju inżynierijnego.

6

Wnioski końcowe

6.1 Podsumowanie

W niniejszej pracy podjęto próbę kompleksowego przedstawienia uczenia maszynowego jako narzędzi doskonalenia procesów wytwórczych. Praca łączyła podstawy teoretyczne z praktycznym studium przypadku, które pozwoliło na weryfikację omówionych metod w rzeczywistym problemie przemysłowym.

W pierwszym rozdziale dokonano przeglądu literatury, który pokazał, że uczenie maszynowe ma solidne podstawy teoretyczne i jest coraz częściej wykorzystywane w kontekście Przemysłu 4.0. Drugi rozdział przedstawił szczegółową charakterystykę algorytmów uczenia maszynowego, od metod uczenia nadzorowanego, przez nienadzorowane, po uczenie przez wzmacnianie. Zrozumienie ich właściwości i ograniczeń jest decydujące przy wyborze odpowiedniej metody do konkretnego problemu.

W trzecim rozdziale przeanalizowano praktyczne zastosowania uczenia maszynowego w procesach wytwórczych, takie jak konserwacja predykcyjna, automatyczna kontrola jakości czy optymalizacja parametrów procesowych. Te obszary pokazują, że uczenie maszynowe ma realny potencjał do poprawy efektywności i jakości w przemyśle, choć jego wdrożenie wymaga nie tylko kompetencji technicznych, ale też zmian organizacyjnych.

Najważniejszą częścią pracy było studium przypadku dotyczące klasyfikacji defektów w produktach odlewniczych. Porównano sześć różnych algorytmów na zbiorze 1300 obrazów. Random Forest okazał się najskuteczniejszy, osiągając dokładność 91,15% przy czasie treningu zaledwie 2,31 s. Sieć neuronowa też uzyskała dobre wyniki (90,77% dokładności), co pokazuje potencjał głębokiego uczenia. Co ciekawe, nawet prostsza regresja logistyczna osiągnęła przyzwoitą dokładność (86,54%) przy bardzo krótkim czasie treningu (0,35 s), co może być istotne w aplikacjach wymagających częstego przetrenowywania modeli.

6.2 Najważniejsze wnioski

Z przeprowadzonych badań wynikają następujące wnioski:

Uczenie maszynowe to dojrzała technologia z szerokim wyborem algorytmów dostosowanych do różnych problemów. Zarówno klasyczne metody (SVM, Random Forest), jak i nowoczesne sieci neuronowe osiągają wysoką skuteczność w zadaniach przemysłowych. Kluczem sukcesu jest jednak dobrej jakości zbiór danych, bez odpowiednich danych nawet najlepszy algorytm nie da dobrych wyników. Przeprowadzony eksperyment pokazał, że nawet 1300 obrazów może wystarczyć do osiągnięcia 91% dokładności, ale większy zbiór z pewnością poprawiłby stabilność i zdolność modeli do generalizacji.

Ważne jest też znalezienie odpowiedniej równowagi między dokładnością a złożonością modelu. Nie zawsze najskuteczniejszy algorytm jest najlepszym wyborem, bo czasem prostszy model, który trenuje się szybciej i jest łatwiejszy do zrozumienia, będzie lepszym

rozwiązaniem w praktyce. W przeprowadzonym badaniu Random Forest okazał się optymalny, bo łączył wysoką dokładność (91,15%) z krótkim czasem treningu (2,31 s) i dobrą interpretowalnością.

Trzeba też pamiętać, że wdrożenie uczenia maszynowego w rzeczywistym środowisku przemysłowym to coś więcej niż tylko wytrenowanie modelu. Zmienność warunków produkcyjnych, szum w danych, integracja z istniejącymi systemami czy akceptacja przez operatorów, wszystkie te czynniki trzeba wziąć pod uwagę.

6.3 Ograniczenia badań

Przeprowadzone badania mają kilka ograniczeń, które warto uwzględnić:

Przede wszystkim zbiór danych był stosunkowo mały (1300 obrazów) i pochodził z kontrolowanych warunków laboratoryjnych. W rzeczywistej produkcji warunki są znacznie bardziej zróżnicowane: zmienne oświetlenie, wibracje, zanieczyszczenia. Poza tym użyto prostego podziału 80:20 na zbiór treningowy i testowy. Lepiej byłoby zastosować walidację krzyżową, która dałaby bardziej stabilną ocenę wydajności.

Nie przeprowadzono też testów systemu w rzeczywistym środowisku produkcyjnym, co uniemożliwiło ocenę jego funkcjonowania w praktycznych warunkach. To jest prawdopodobnie największe ograniczenie tej pracy. Dodatkowo nie sprawdzono zaawansowanych architektur głębokiego uczenia, jak CNN, które prawdopodobnie osiągnęłyby jeszcze lepsze wyniki na danych obrazowych.

6.4 Kierunki dalszego rozwoju

Praca pokazała kilka obszarów, które warto byłoby rozwinać w przyszłości:

Po pierwsze, zdecydowanie potrzeba większego zbioru danych – co najmniej 5000-10000 obrazów z różnymi typami defektów i warunkami produkcyjnymi. Pozwoliłoby to na trenowanie bardziej zaawansowanych modeli i lepszą ocenę ich zdolności do generalizacji.

Warto byłoby też sprawdzić konwolucyjne sieci neuronowe (CNN) i transfer learning z wykorzystaniem pretrenowanych modeli. CNN są zaprojektowane specjalnie do pracy z obrazami i prawdopodobnie dałyby lepsze wyniki niż moja prosta sieć neuronowa. Ciekawe byłoby także rozszerzenie problemu z klasyfikacji binarnej (defekt/brak defektu) do rozpoznawania konkretnych typów defektów.

Kolejny istotny kierunek to interpretowalność modeli. Techniki jak LIME czy SHAP mogłyby pomóc zrozumieć, dlaczego model podejmuje określone decyzje, co zwiększyłoby zaufanie operatorów do systemu. I oczywiście trzeba by było to wszystko przetestować w rzeczywistym środowisku produkcyjnym, żeby zobaczyć, jak system radzi sobie z prawdziwymi danymi i zmienną jakością obrazów.

6.5 Podsumowanie

Uczenie maszynowe ma duży potencjał w doskonaleniu procesów wytwarzających. Przeprowadzone studium przypadku pokazało, że nawet przy ograniczonych zasobach można

osiągnąć wysoką skuteczność klasyfikacji (ponad 91% dokładności). Random Forest okazał się najlepszym rozwiązaniem w analizowanym problemie, łącząc dobrą dokładność z szybkim treningiem.

Trzeba jednak pamiętać, że samo wytrenowanie modelu to dopiero początek. Sukces wdrożenia zależy od wielu czynników: jakości danych, odpowiedniego preprocessingu, doboru metryk, a przede wszystkim od zrozumienia kontekstu operacyjnego i ograniczeń. Jak pokazują przeprowadzone badania, modele uczenia maszynowego są tak dobre, jak dane, na których zostały wytrenowane.

Patrząc w przyszłość, rolę uczenia maszynowego w przemyśle będzie rosła dzięki postępowi w algorytmach, większej mocy obliczeniowej i upowszechnieniu infrastruktury IoT. Kluczowe będzie jednak nie tylko doskonalenie technologii, ale też rozwój kompetencji inżynierskich i budowanie zaufania do systemów AI. Należy mieć nadzieję, że ta praca przyczyni się do lepszego zrozumienia możliwości i ograniczeń uczenia maszynowego w procesach wytwórczych.

Bibliografia

- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019. doi: 10.1145/3292500.3330701.
- David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, Philadelphia, PA, 2007. Society for Industrial and Applied Mathematics.
- Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The MVTec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision*, 129(4):1038–1059, 2021.
- James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.
- David Birkes and Yadolah Dodge. *Alternative Methods of Regression*. Wiley, New York, 1993.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.
- Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.
- Thyago P. Carvalho, Fabrizzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, and Symone G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.
- Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
- Fan-Tien Cheng, Haw-Ching Yang, Yu-Ning Cheng, and Yi-Cheng Chen. Industry 4.0 for wheel machining automation. *IEEE Robotics and Automation Letters*, 1(1):332–339, 2016.
- Raffaele Cioffi, Marta Travaglioni, Giuseppina Piscitelli, Alessandra Petrillo, and Fabio De Felice. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability*, 12(2):492, 2020.
- Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.
- Jorge Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Sandro Sanyal, José Favilla, and Jorge Barbosa. Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges. *Computers in Industry*, 123:103298, 2020.
- Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, New York, 2nd edition, 2001.

- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.
- Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28, 2015.
- Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, 2006.
- Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016. URL <http://www.deeplearningbook.org>.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd edition, 2009.
- Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for non-northogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.
- Zheng Huang, Chao Liu, Xiaohong Huang, Jun Zhang, and Guoping Wan. Data-driven quality prediction for complex production processes using machine learning. *IEEE Access*, 9:89030–89044, 2021.
- Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, 1st edition, 2013.
- Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Final report of the industrie 4.0 working group. In *Forschungsunion*. acatech, 2013.
- Andrew Kusiak. Smart manufacturing. *International Journal of Production Research*, 56(1-2):508–517, 2018.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.
- Jay Lee, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel. Prognostics and health management design for rotary machinery systems – reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(1-2):314–334, 2014.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.
- Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- R. Keith Mobley. *An Introduction to Predictive Maintenance*. Butterworth-Heinemann, Oxford, 2nd edition, 2002.
- László Monostori, Botond Kádár, Tibor Bauernhansl, Sándor Kondoh, Soundar Kumara, Georg Reinhart, Olaf Sauer, Günther Schuh, Wim Sihn, and Kathryn Ueda. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, 65(2):621–641, 2016.
- Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

- Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 485–492, 2016. doi: 10.1145/2908812.2908918.
- William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 3rd edition, 2007.
- Foivos Psarommatis and Dimitris Kiritsis. A hybrid decision support system for automating decision making in the event of defects in the era of Zero Defect Manufacturing. *Journal of Industrial Information Integration*, 26:100263, 2022.
- Foivos Psarommatis, Gökan May, Paul-Arthur Dreyfus, and Dimitris Kiritsis. Zero defect manufacturing: State-of-the-art review, shortcomings and future directions in research. *International Journal of Production Research*, 58(1):1–17, 2020.
- J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi: 10.1007/BF00116251.
- J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019.
- Klaus Schwab. *The Fourth Industrial Revolution*. World Economic Forum, Geneva, 2016.
- Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.
- Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.
- Melissa Sharp, Rayid Ak, and Thomas Hedberg Jr. A survey of the advancing use and development of machine learning in smart manufacturing. *Journal of Manufacturing Systems*, 48:170–179, 2018.
- Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.
- Gian Antonio Susto, Andrea Schirru, Simone Pampuri, Seán McLoone, and Alessandro Beghi. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820, 2015.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.
- Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.
- Domen Tabernik, Samo Šela, Jure Skvarč, and Danijel Skočaj. Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*, 31(3):759–776, 2020.
- Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

- Jiayi Wang, Zhangjie Fu, Kai Chen, Peng Peng, and Fang Liu. Machine learning for quality control in manufacturing: A comprehensive review. *Journal of Manufacturing Systems*, 63:550–567, 2022a.
- Junliang Wang, Pai Zheng, Yan Zhang, and Ang Liu. Big data analytics for intelligent manufacturing systems: A review. *Journal of Manufacturing Systems*, 62:738–752, 2022b.
- Daniel Weichert, Patrick Link, Anke Stoll, Stefan Rüping, Steffen Ihlenfeldt, and Stefan Wrobel. A review of machine learning for the optimization of production processes. *The International Journal of Advanced Manufacturing Technology*, 104(5-8):1889–1902, 2019.
- Daniel Weimer, Bernhard Scholz-Reiter, and Meik Shpitalni. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals - Manufacturing Technology*, 65(1):417–420, 2016.
- Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: Advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.
- Jing Yang, Shiqi Li, Zheng Wang, Haiyong Dong, Jun Wang, and Senlin Tang. Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges. *Materials*, 13(24):5755, 2020.
- Weimin Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019.
- Ray Y. Zhong, Xun Xu, Eberhard Klotz, and Stephen T. Newman. Intelligent manufacturing in the context of industry 4.0: A review. *Engineering*, 3(5):616–630, 2017.
- Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.
- Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. doi: 10.1111/j.1467-9868.2005.00503.x.