Politechnika Wrocławska

Politechnika Wrocławska
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

Spis treści

1

Charakterystyka algorytmów uczenia maszynowego

1.1 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 predyktować 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 danych 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).

1.1.1 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.

Regresja liniowa prosta. W przypadku pojedynczej zmiennej niezależnej model można opisać równaniem:

$$y = \alpha + \beta x + \varepsilon, \tag{1.1}$$

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:

$$y = \alpha + \sum_{i=1}^{p} \beta_i x_i + \varepsilon, \tag{1.2}$$

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.

Estymator OLS (macierzowy zapis).

$$\widehat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \mathbf{y} \tag{1.3}$$

Wzór (1.3) to macierzowy zapis estymatora OLS. 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 $n \times 1$,
- $\widehat{\boldsymbol{\beta}}$ — 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. $E[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2 I$) estymator jest nieobciążony, a jego wariancja wynosi $\mathrm{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2 (X^\top X)^{-1}$. Gdy $X^\top X$ jest źle uwarunkowana lub nieodwracalna, stosuje się regularyzację (np. Ridge) lub metody numeryczne (gradient descent, SVD)(James et al., 2013).

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 górnotrójkątnej, co umożliwia efektywne rozwiązanie układu równań regresji (Golub and Loan, 1996; Press et al., 2007).

1.1.2 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 przyjmuję wartości binarne (np. 0 lub 1, tak lub nie). Celem regresji logistycznej jest przewidywanie prawdopodobieństwa przynależności do jednej z dwóch klas na podstawie wartości zmiennych niezależnych.

Model regresji logistycznej. Model regresji logistycznej można zapisać jako:

$$P(Y = 1|X) = \sigma(\boldsymbol{\beta}^\top X) = \frac{1}{1 + e^{-\boldsymbol{\beta}^\top X}} \tag{1.4}$$

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:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(Y_i|X_i; \boldsymbol{\beta}) \tag{1.5}$$

co jest równoważne minimalizacji funkcji straty:

$$J(\boldsymbol{\beta}) = -\sum_{i=1}^{n} [Y_i \log(P(Y_i|X_i; \boldsymbol{\beta})) + (1 - Y_i) \log(1 - P(Y_i|X_i; \boldsymbol{\beta}))] \tag{1.6}$$

Właściwości modelu. Model regresji logistycznej ma kilka istotnych właściwości:

- Wynikiem modelu jest prawdopodobieństwo, co czyni go odpowiednim narzędziem do klasyfikacji binarnej.
- Model jest odporny na wartości odstające, ponieważ wykorzystuje funkcję sigmoidalną.
- Można go łatwo rozszerzyć na problemy wieloklasowe (np. regresja wielomianowa).

1.1.3 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 obiektami w przestrzeni, a następnie wybiera k najblizszych sąsiadów. Wartość k jest ustalana ustalana przed rozpoczęciem działania algorytmu. Niska wartość parametru k jest bardziej podatna na szumy 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:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{1.7}$$

gdzie p i q to dwa punkty w przestrzeni, a n to liczba wymiarów. Wyliczanie odległości metryką euklidesową polega na policzeniu różnicy ogległości 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 miedzy dwoma punktami jako sumę watości bezwzględnych z różnic współżędnych. Definiowana jest jako:

$$d(p, q) = \sum_{i=1}^{n} |p_i - q_i| \tag{1.8}$$

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:

$$d(p, q) = 1 - \frac{p \cdot q}{\|p\| \|q\|} \tag{1.9}$$

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. Umożliwa regulowanie sposobu obliczania odległości poprzez parametr p. Definiowana jest jako:

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{\frac{1}{p}} \tag{1.10}$$

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).

1.1.4 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 rozdzielają 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:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{1.11}$$

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:

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2 \tag{1.12}$$

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

$$\mathrm{IG}(S, A) = H(S) - \sum_{v \in \mathrm{Values}(A)} \frac{|S_v|}{|S|} H(S_v) \tag{1.13}$$

gdzie 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:

$$\mathrm{GainRatio}(S, A) = \frac{\mathrm{IG}(S, A)}{\mathrm{SplitInfo}(S, A)} \tag{1.14}$$

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

$$\mathrm{SplitInfo}(S, A) = -\sum_{v \in \mathrm{Values}(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right) \tag{1.15}$$

gdzie 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:

$$\mathrm{VarReduction}(S) = \mathrm{Var}(S) - \left(\frac{|S_{\text{left}}|}{|S|} \mathrm{Var}(S_{\text{left}}) + \frac{|S_{\text{right}}|}{|S|} \mathrm{Var}(S_{\text{right}}) \right) \tag{1.16}$$

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

1.1.5 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 budownie lasu losowego dobiera się parametry:

Tabela 1.1: 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. Dla każdej kombinacji algorytm trenuje model i ocenia jego wydajność za pomocą walidacji krzyżowej. Następnie wybiera zestaw parametrów dający najlepsze wyniki według ustalonej metryki.

przeszukiwanie losowe (Randomized Search) Randomized Search jest wariantem Grid Search, który zamiast sprawdzać wszystkie kombinacje, losowo próbkuje określoną liczbę zestawów hiperparametrów z zadanych rozkładów. 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 eksploatację 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).

1.1.6 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. Celem SVM jest maksymalizacja marginesu, czyli odległości między hiperpłaszczyzną a najbliższymi punktami z obu klas, zwanymi 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:

$$\boldsymbol{w}^\top \boldsymbol{x} + b = 0 \tag{1.17}$$

gdzie $\boldsymbol{w}$ to wektor normalny do hiperpłaszczyzny, $\boldsymbol{x}$ to wektor cech, a b to wyraz wolny. Funkcja decyzyjna klasyfikuje punkt $\boldsymbol{x}$ na podstawie znaku wyrażenia $\boldsymbol{w}^\top \boldsymbol{x} + b$:

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{1.18}$$

Optymalna hiperpłaszczyzna jest wyznaczana przez rozwiązanie problemu optymalizacji:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{przy ograniczeniach} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1, \ \forall i \tag{1.19}$$

gdzie $y_i \in \{-1, +1\}$ to etykiety klas, a $\boldsymbol{x}_i$ 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 szumy. W takich przypadkach stosuje się wariant soft margin SVM, który dopuszcza błędy

klasyfikacji poprzez wprowadzenie zmiennych slackowych $\xi_i \geq 0$. Problem optymalizacji przyjmuje wtedy postać:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{1.20}$$

przy ograniczeniach:

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \ \forall i \tag{1.21}$$

gdzie $C > 0$ to parametr regularyzacji 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).

Kernel Trick (sztuczka jądrowa). 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. Najpopularniejsze funkcje jądrowe to:

Jądro liniowe — dla liniowo separowalnych danych:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j \tag{1.22}$$

Jądro wielomianowe — pozwala na nieliniowe granice decyzyjne:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i^\top \boldsymbol{x}_j + r)^d \tag{1.23}$$

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

Jądro radialnej funkcji bazowej (RBF, Gaussian) — najczęściej stosowane, elastyczne:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right) \tag{1.24}$$

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

Jądro sigmoidalne — zachowuje się podobnie do sieci neuronowych:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\gamma \boldsymbol{x}_i^\top \boldsymbol{x}_j + r) \tag{1.25}$$

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).

1.1.7 Naiwny Klasyfikator Bayesowski (Naive Bayes)

Naiwny klasyfikator Bayesowski to probabilistyczny algorytm klasyfikacji oparty na twierdzeniu Bayesa z założeniem warunkowej niezależności cech. Algorytm oblicza prawdopodobieństwo przynależności obiektu do każdej z klas na podstawie wartości jego cech. Po obliczeniu przypisuje obiekt do klasy o najwyższym 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 $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$:

$$P(C_k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_k)P(C_k)}{P(\boldsymbol{x})} \tag{1.26}$$

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

Warianty algorytmu. W zależności od charakteru danych stosuje się różne warianty Naive Bayes:

Gaussian Naive Bayes. Wariant stosowany dla cech ciągłych, zakłada, że wartości każdej cechy w danej klasie mają rozkład normalny (Gaussa), 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. 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 tekstów, filtrowania spamu oraz analizie sentymentu, gdzie każda cecha reprezentuje liczbę wystąpień określonego słowa(Murphy, 2012).

Bernoulli Naive Bayes. Wariant dla cech binarnych przyjmujących wartości 0 lub 1. 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 dane słowo występuje w dokumencie, niezależnie od liczby jego wystąpień (Murphy, 2012).

1.2 Uczenie nienadzorowane (Unsupervised Learning)

Uczenie nienadzorowane to rodzaj uczenia maszynowego, który sam odkrywa wzorce i struktury danych bez ówcześnie nadachynych etykiek przez człowieka. Alogrytm przetwarza surowe dane wejściowe, a następnie za pomocą metod statystycznych i geometrycznych 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).

1.2.1 k-Średnich (k-Means)

Algorytm k-średnich to jedna z metod klasteryzacji, która grupuje dane wejściowe w k klastrów na podstawie podobieństwa między obiektami. 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).

$$J = \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2 \tag{1.27}$$

gdzie J to całkowita suma kwadratów błędów, które są minimalizowane iteracyjnie przez algorytm, j to iteracja, k to liczba klastrów, $\boldsymbol{x}_i$ 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 sie następujące parametry:

Tabela 1.2: 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. Następnie wykreśla się wykres SSE w funkcji k i szuka punktu, w którym dalsze zwiększanie 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. 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).

1.2.2 Hierarchiczna klasteryzacja (Hierarchical Clustering)

Hierarchiczna klasteryzacja to metoda klasteryzacji, która tworzy hierarchię klastrów poprzez iteracyjne łączenie lub dzielenie grup obiektów na podstawie ich podobieństwa. 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:

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

$$d(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y}) \tag{1.28}$$

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

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

$$d(C_i, C_j) = \max_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y}) \tag{1.29}$$

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

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

$$d(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{y} \in C_j} d(\boldsymbol{x}, \boldsymbol{y}) \tag{1.30}$$

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.

1.2.3 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. Algorytm DBSCAN identyfikuje klastry na podstawie dwóch głównych parametrów: promienia sąsiedztwa ε (epsilon) oraz minimalnej liczby punktów $minPts$ wymaganej do utworzenia gęstego 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 $\boldsymbol{p}$ sąsiedztwo ε definiowane jest jako zbiór punktów:

$$N_\varepsilon(\boldsymbol{p}) = \{\boldsymbol{q} \in D \mid d(\boldsymbol{p}, \boldsymbol{q}) \leq \varepsilon\} \tag{1.31}$$

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

Punkt rdzeniowy (core point). Punkt $\boldsymbol{p}$ jest punktem rdzeniowym, jeśli liczba punktów w jego sąsiedztwie ε wynosi co najmniej $minPts$:

$$|N_\varepsilon(\boldsymbol{p})| \geq minPts \tag{1.32}$$

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

Osiągalność gęstościowa. Punkt $\boldsymbol{q}$ jest osiągalny gęstościowo z punktu $\boldsymbol{p}$, jeśli istnieje łańcuch punktów $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n$, gdzie $\boldsymbol{p}_1 = \boldsymbol{p}$ i $\boldsymbol{p}_n = \boldsymbol{q}$, taki że każdy kolejny punkt jest bezpośrednio osiągalny gęstościowo z poprzedniego.

Klasyfikacja punktów w DBSCAN. Algorytm DBSCAN klasyfikuje każdy punkt w zbiorze danych do jednej z trzech kategorii:

Punkt rdzeniowy (core point): punkt $\boldsymbol{p}$ jest rdzeniowy, jeśli ma co najmniej $minPts$ sąsiadów w promieniu ε, tj. $|N_\varepsilon(\boldsymbol{p})| \geq minPts$. 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).

1.2.4 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. 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 (odjęcia średniej od każdej cechy) i obliczenia macierzy kowariancji, która opisuje zależności między cechami:

$$\Sigma = \frac{1}{n} X^\top X \tag{1.33}$$

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 rozwiązanie równania:

$$\Sigma \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{1.34}$$

gdzie $\boldsymbol{v}_i$ 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.

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

$$Z = XW \tag{1.35}$$

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:

$$\text{Var}_i = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \tag{1.36}$$

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).

1.2.5 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. 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 $\boldsymbol{x}_i$ i $\boldsymbol{x}_j$ jest modelowane jako prawdopodobieństwo warunkowe:

$$p_{j|i} = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2 / 2\sigma_i^2)} \tag{1.37}$$

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

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{1.38}$$

gdzie n to liczba punktów danych.

Podobieństwa w przestrzeni niskowymiarowej. W przestrzeni niskowymiarowej podobieństwo między punktami $\boldsymbol{y}_i$ i $\boldsymbol{y}_j$ jest modelowane za pomocą rozkładu t-Studenta z jednym stopniem swobody:

$$q_{ij} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}} \tag{1.39}$$

Optymalizacja. Algorytm t-SNE minimalizuje dywergencję Kullbacka-Leiblera między rozkładami P i Q:

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{1.40}$$

Optymalizacja jest przeprowadzana za pomocą gradientu prostego lub metod opartych na momencie, co pozwala na znalezienie układu punktów $\boldsymbol{y}_i$ 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:

- **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 położeń punktów podczas optymalizacji. Zbyt wysoki współczynnik może prowadzić do niestabilności, podczas gdy zbyt niski może spowolnić zbieżność.

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

1.2.6 UMAP (Uniform Manifold Approximation and Projection)

UMAP to technika redukcji wymiarowości i wizualizacji danych, która opiera się na teorii topologii i geometrii różniczkowej. 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. Podobieństwo między punktami $\boldsymbol{x}_i$ i $\boldsymbol{x}_j$ jest modelowane za pomocą funkcji ważonej:

$$w_{ij} = \exp\left(-\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j) - \rho_i}{\sigma_i}\right) \tag{1.41}$$

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

Optymalizacja w przestrzeni niskowymiarowej. W drugim etapie UMAP optymalizuje rozmieszczenie punktów $\boldsymbol{y}_i$ w przestrzeni niskowymiarowej, minimalizując funkcję kosztu opartą na różnicy między grafem sąsiedztwa w przestrzeni wysokowymiarowej a grafem w przestrzeni niskowymiarowej:

$$C = \sum_{i \neq j} \left(w_{ij} \log \frac{w_{ij}}{q_{ij}} + (1 - w_{ij}) \log \frac{1 - w_{ij}}{1 - q_{ij}} \right) \tag{1.42}$$

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

$$q_{ij} = \frac{1}{1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^{2b}} \tag{1.43}$$

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 $\boldsymbol{y}_i$ 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:

- **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 przesądza o równowadze między zachowaniem lokalnych struktur danych a globalnej topologii w wizualizacjach UMAP (McInnes et al., 2018).

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.

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.

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.

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.

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 nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, 1st edition, 2013.

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.

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.

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.

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.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

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.

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.