Politechnika Wrocławska

Wydział Matematyki
Kierunek studiów: Matematyka stosowana
Specjalność: –

Praca dyplomowa – inżynierska

WYBRANE METODY UCZENIA CZĘŚCIOWO NADZOROWANEGO

Kamil Cyprian Iwach-Kowalski

słowa kluczowe:
uczenie częściowo nadzorowane, uczenie maszynowe, klasyfikacja, ocena skuteczności klasyfikacji, porównanie stabilności, LAMDA-SSL

krótkie streszczenie:
Praca skupia się na analizie porównawczej algorytmów uczenia częściowo nadzorowanego (SSL). W badaniach uwzględnione zostały algorytmy SSL, takie jak Tri-Training, Assemble, SemiBoost, LapSVM, TSVM i Label Propagation. Przeprowadzone eksperymenty oparte były na zbiorach danych CIFAR-10, IMDb i Breast Cancer. Zbadany został m.in. wpływ hiperparametrów i frakcji danych etykietowanych na wydajność oraz stabilność metod. Uzyskane wyniki wskazują na istniejący potencjał metod SSL, ale także na istotne znaczenie doboru parametrów.

Opiekun pracy dyplomowej	dr inż. Adam Zagdański	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław, rok 2025

Wrocław University
of Science and Technology

Faculty of Pure and Applied Mathematics
Field of study: Applied Mathematics
Specialty: –

Engineering Thesis

SELECTED SEMI-SUPERVISED LEARNING METHODS

Kamil Cyprian Iwach-Kowalski

keywords:
semi-supervised learning, machine learning, classification, performance evaluation, stability comparison, LAMDA-SSL

short summary:
The thesis focuses on the comparative analysis of semi-supervised learning (SSL) algorithms. SSL algorithms such as Tri-Training, Assemble, SemiBoost, LapSVM, TSVM and Label Propagation were tested on the CIFAR-10, IMDb and Breast Cancer datasets. In the conducted experiments, the effect of hyperparameters and labeled data fraction on the efficiency and stability of the methods were investigated. The results show the existing potential of SSL methods, but also the influence of the choice of parameters.

Supervisor	dr inż. Adam Zagdański	.....	.....
	Title/degree/name and surname	grade	signature

*For the purposes of archival thesis qualified to:**

a) category A (perpetual files)

b) category BE 50 (subject to expertise after 50 years)

** delete as appropriate*

stamp of the faculty

Wrocław, 2025

Spis treści

Wstęp

W dobie dynamicznie rozwijających się technologii, uczenie maszynowe (ang. *machine learning*), będące poddziedziną sztucznej inteligencji, stało się jednym z najbardziej istotnych i innowacyjnych obszarów badań w dziedzinie matematyki oraz informatyki. Uczenie maszynowe to proces, w którym systemy komputerowe zdobywają wiedzę lub umiejętności poprzez doświadczenie [8], co pozwala im na rozwiązywanie złożonych problemów i podejmowanie decyzji w sposób automatyczny.

W przypadku uczenia maszynowego kluczową rolę odgrywa matematyka, która stanowi fundament dla wszystkich algorytmów i modeli wykorzystywanych w tej dziedzinie. W szczególności statystyka, teoria prawdopodobieństwa, analiza numeryczna i algebra liniowa, zapewniają narzędzia niezbędne do analizy, projektowania i optymalizacji modeli uczenia maszynowego. Zrozumienie tych matematycznych koncepcji jest niezbędne do efektywnego modelowania złożonych systemów oraz interpretacji otrzymywanych wyników.

Istnieją różne rodzaje uczenia maszynowego, z których najbardziej rozpowszechnione to uczenie nadzorowane, nienadzorowane i uczenie ze wzmocnieniem. Uczenie nadzorowane (ang. *supervised learning*) polega na trenowaniu modelu na podstawie etykietowanych danych, gdzie każda próbka danych zawiera wejście oraz odpowiadające mu wyjście. W przeciwieństwie do tego, uczenie nienadzorowane (ang. *unsupervised learning*) nie wykorzystuje etykietowanych danych. Zamiast tego, algorytmy próbują odkryć ukryte wzorce występujące w nieetykietowanych danych. Z kolei uczenie ze wzmocnieniem (ang. *reinforcement learning*) to podejście, w przypadku którego model uczy się poprzez interakcje z otoczeniem, dążąc do maksymalizacji pewnej nagrody.

W ostatnich latach, uczenie maszynowe zyskało szczególną popularność dzięki rozwojowi dużych modeli językowych (ang. *large language models*, LLM) oraz modeli generujących obrazy. Przykładami takich modeli są Midjourney, który zrewolucjonizował podejście do generowania obrazów oraz GPT-4, zaawansowany model generatywny przetwarzania języka naturalnego.

Niniejsza praca dyplomowa koncentruje się na specyficznym rodzaju uczenia maszynowego, znanym jako uczenie częściowo nadzorowane (ang. *semi-supervised learning*, SSL). Metoda ta łączy elementy uczenia nadzorowanego i nienadzorowanego, wykorzystując zarówno etykietowane, jak i nieetykietowane dane w procesie trenowania modelu. Uczenie częściowo nadzorowane ma na celu rozwiązanie problemów związanych z kosztami i trudnościami w etykietowaniu dużych zestawów danych, oferując efektywne rozwiązania w sytuacjach, gdzie dostęp do etykiet jest ograniczony.

W ramach tej pracy przeprowadzono szczegółową analizę porównawczą, obejmującą sześć wybranych, najpopularniejszych algorytmów uczenia częściowo nadzorowanego. Każdy z nich został zbadany pod kątem skuteczności, efektywności obliczeniowej oraz stabilności, w kontekście różnych zagadnień praktycznych. W tym celu uwzględniono trzy zestawy danych, związane odpowiednio z klasyfikacją obrazów, tekstów oraz diagnostyką medyczną. Przeprowadzone eksperymenty pozwoliły na szczegółowe zbadanie,

jaki wpływ na wydajność porównywanych metod mają najważniejsze czynniki, takie jak wybór odpowiednich hiperparametrów czy wielkość frakcji obserwacji etykietowanych.

Wszystkie analizy zostały przeprowadzone z wykorzystaniem języka programowania Python, który jest obecnie standardem w dziedzinie uczenia maszynowego ze względu na jego elastyczność, obszerną bibliotekę narzędziową i szeroką społeczność użytkowników. Dodatkowe, bardziej techniczne informacje dotyczące implementacji przeprowadzonych eksperymentów, zostały umieszczone w dodatku znajdującym się na końcu niniejszej pracy. Kody źródłowe stworzonych programów są natomiast udostępnione w publicznym repozytorium GitHub pod adresem `https://github.com/github-kamilk/BSc_Thesis`.

Rozdział 1

Część teoretyczna

W niniejszym rozdziale zostaną przedstawione teoretyczne aspekty klasyfikacji i uczenia częściowo nadzorowanego. Na początku omówimy podstawowe metody nadzorowane (ang. *supervised learning*) i zdefiniujemy najpopularniejsze miary (kryteria) stosowane do oceny wydajności modeli klasyfikacyjnych. Następnie opiszemy najważniejsze aspekty uczenia częściowo nadzorowanego (ang. *semi-supervised learning*, SSL), przedstawiając kluczowe założenia oraz taksonomię tych metod. W ostatniej części zostaną opisane wybrane zagadnienia związane z przygotowaniem danych do klasyfikacji.

1.1 Wprowadzenie do klasyfikacji

Klasyfikacja to proces przypisywania etykiet do obiektów (obserwacji) na podstawie opisujących je cech, nazywanych także zmiennymi lub atrybutami. Jest to jedno z podstawowych zadań w dziedzinie uczenia maszynowego, które polega na uczeniu modelu rozpoznawania wzorców w danych i przyporządkowywaniu ich do odpowiednich kategorii. Przykładem klasyfikacji może być rozpoznawanie obrazów, gdzie algorytm uczy się identyfikować obiekty na zdjęciach i klasyfikować je jako „samochód", „drzewo", „zwierzę" itp.

Warto zauważyć, że klasyfikacja, podobnie jak regresja, jest przykładem uczenia nadzorowanego. Ogólnie rzecz biorąc, w uczeniu nadzorowanym zadaniem algorytmu jest nauczenie się rozpoznawania zależności między wejściem (X) i wyjściem (Y) na podstawie zbioru uczącego (treningowego). Znaleziona zależność wykorzystywana jest następnie do prognozowania odpowiedzi dla nowo zaobserwowanych wartości wejściowych. Wprowadźmy teraz formalną definicję reguły klasyfikacyjnej (klasyfikatora).

1.1.1 Reguła klasyfikacyjna

Definicja 1.1. (Reguła klasyfikacyjna) [6]
Reguła klasyfikacyjna, zwana krótko *klasyfikatorem*, jest funkcją

$$d : \mathcal{X} \to \mathcal{Y}, \tag{1.1}$$

gdzie $\mathcal{X}$ to przestrzeń próby (w szczególności $\mathcal{X} = \mathbb{R}^p$), $\mathcal{Y}$ to zbiór etykiet. W przypadku, gdy obserwujemy nowy wektor $\mathbf{x} \in \mathcal{X}$ (zawierający wartości p cech/charakterystyk opisujących dany obiekt), to prognozą etykiety Y jest $d(\mathbf{x})$.

Jeśli zbiór $\mathcal{Y}$ zawiera tylko dwie etykiety (np. $\mathcal{Y} = \{0, 1\}$ lub $\mathcal{Y} = \{\text{chory},\ \text{zdrowy}\}$), mówimy o klasyfikacji binarnej. W przeciwnym razie mamy do czynienia z klasyfikacją wieloklasową.

W praktyce, do konstrukcji klasyfikatora wykorzystujemy zbiór uczący (ang. *learning set*), oznaczany zwykle jako $\mathcal{L}_n$, zawierający n elementów (obserwacji) o znanej przynależności do klas (znanych etykietach klas), tzn. $\mathcal{L}_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, gdzie $\mathbf{x}_i \in \mathcal{X}$ oraz $y_i \in \mathcal{Y}$.

1.1.2 Standardowe metody klasyfikacji

Istnieje wiele metod klasyfikacji stosowanych w uczeniu maszynowym. W niniejszej pracy wykorzystano te najbardziej podstawowe i najpopularniejsze w zastosowaniach praktycznych. Są to: algorytm K-najbliższych sąsiadów (ang. *K-nearest neighbors*, KNN), naiwny klasyfikator bayesowski (*naive bayes*), maszyna wektorów nośnych (ang. *support vector machines*, SVM), drzewo decyzyjne (ang. *decision tree*) oraz las losowy (ang. *random forest*). Poniżej zamieszczono ich krótkie opisy na podstawie [6] i [15].

Algorytm K-najbliższych sąsiadów

Algorytm K-najbliższych sąsiadów (KNN, od ang. *K-nearest neighbors*) jest jedną z najbardziej podstawowych metod klasyfikacji stosowanych w uczeniu maszynowym i jest zaliczany do grupy metod nieparametrycznych. Ta metoda, pierwotnie zaproponowana przez Fixa i Hodgesa w 1951 roku, klasyfikuje obiekty poprzez głosowanie ich sąsiadów. Obiekt $\mathbf{x}$ zostaje przypisany do klasy j, jeśli większość z jego K najbliższych sąsiadów również przynależy do tej klasy.

Otrzymany na podstawie metody KNN estymator prawdopodobieństwa a posteriori można zapisać w następujący sposób

$$\hat{p}(j|\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{n} \mathbf{1}_{\{y_i = j\}} \cdot \mathbf{1}_{\left\{\rho(\mathbf{x}, \mathbf{x}_i) \leqslant \rho\left(\mathbf{x}, \mathbf{x}^{(K)}\right)\right\}}, \quad j = 1, \ldots, J, \tag{1.2}$$

gdzie $\mathbf{1}_{(A)}$ jest indykatorem zdarzenia A, y_i jest etykietą klasy i-tego obiektu, ρ jest miarą odległości, a $\mathbf{x}^{(K)}$ jest K-tym pod względem odległości od $\mathbf{x}$ punktem ze zbioru uczącego $\mathcal{L}_n$.

Dla tak sformułowanego $\hat{p}$, klasyfikator oparty na algorytmie KNN przyjmuje postać

$$\hat{d}_{KNN}(\mathbf{x}) = \arg\max_{j \in \mathcal{Y}} \hat{p}(j|\mathbf{x}). \tag{1.3}$$

W praktyce, jako miarę odległości ρ w algorytmie KNN wykorzystuje się najczęściej odległość euklidesową. Istotny wpływ na skuteczność klasyfikatora może mieć również wybór liczby sąsiadów K. W przypadku klasyfikacji binarnej, wykorzystujemy zwykle nieparzyste wartości K, co pozwala uniknąć sytuacji remisowych.

Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski (NB, od ang. *naive bayes*) to probabilistyczny algorytm klasyfikacyjny, który wykorzystuje twierdzenie Bayesa przy jednoczesnym (naiwnym) założeniu o wzajemnej niezależności cech.

Podstawą działania naiwnego klasyfikatora bayesowskiego jest reguła Bayesa, która w kontekście klasyfikacji może być wyrażona jako

$$P(j|\mathbf{x}) = \frac{P(\mathbf{x}|j) \cdot P(j)}{P(\mathbf{x})}, \tag{1.4}$$

gdzie

- $P(j|\mathbf{x})$ to prawdopodobieństwo, że obserwacja $\mathbf{x}$ należy do klasy j,
- $P(\mathbf{x}|j)$ to prawdopodobieństwo obserwacji $\mathbf{x}$ przy założeniu, że należy ona do klasy j, wyrażone poprzez funkcję gęstości $f_j(\mathbf{x})$,
- $P(j)$ to prawdopodobieństwo wystąpienia klasy j,
- $P(\mathbf{x})$ to całkowite prawdopodobieństwo wystąpienia obserwacji $\mathbf{x}$, które dla celów klasyfikacji jest często pomijane, ponieważ jest stałe dla wszystkich klas.

W przypadku naiwnego klasyfikatora bayesowskiego, $P(\mathbf{x}|j)$ jest modelowane jako iloczyn jednowymiarowych funkcji gęstości

$$\hat{f}_j(\mathbf{x}) = \hat{f}_j(x_1, x_2, \ldots, x_p) = \prod_{i=1}^{p} f_{ji}(x_i), \tag{1.5}$$

co odzwierciedla naiwne założenie o niezależności cech.

Z kolei estymator prawdopodobieństwa a priori wystąpienia j-tej klasy, tzn. $P(j)$, można zapisać jako

$$\hat{\pi}_j = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{Y_i=j\}}, \tag{1.6}$$

co ostatecznie pozwala na przedstawienie naiwnego klasyfikatora bayesowskiego w postaci

$$\hat{d}_{NB}(\mathbf{x}) = \arg\max_{j \in \mathcal{Y}} \hat{\pi}_j \hat{f}_j(\mathbf{x}). \tag{1.7}$$

Maszyna wektorów nośnych

Maszyna wektorów nośnych (SVM, od ang. *Support Vector Machine*) jest modelem uczenia nadzorowanego, który jest stosowany zarówno w zagadnieniach klasyfikacji jak i regresji. W przypadku klasyfikacji, SVM znajduje optymalną hiperpłaszczyznę dyskryminacyjną, która najlepiej oddziela obserwacje (wektory cech) należące do dwóch różnych klas. Hiperpłaszczyzna ta jest wybierana tak, aby maksymalizować margines, czyli odległość między dwiema równoległymi hierpłaszczyznami, zawierającymi najbliższe obserwacje należące do poszczególnych klas, które są nazywane wektorami nośnymi lub podpierającymi.

W przypadku klasyfikacji binarnej, model zakłada, że każda obserwacja $\mathbf{x_i}$ może być przypisana do jednej z dwóch klas $\mathcal{G}_1$ lub $\mathcal{G}_2$, oznaczanych odpowiednio etykietami $y_i = +1$ lub $y_i = -1$. Funkcja decyzyjna SVM jest definiowana jako

$$g(\mathbf{x}) = \mathbf{w}'\mathbf{x} + w_0, \tag{1.8}$$

gdzie $\mathbf{w}$ jest wektorem wag, a w_0 jest wyrazem wolnym. Klasyfikator liniowy przyjmuje wówczas postać

$$\hat{d}_{SVM}(\mathbf{x}) = \begin{cases} +1 & \text{jeśli } g(\mathbf{x}) > 0, \\ -1 & \text{jeśli } g(\mathbf{x}) \leqslant 0. \end{cases} \tag{1.9}$$

Każdy element próby zostaje poprawnie zaklasyfikowany, jeśli $y_i \cdot g(\mathbf{x_i}) > 0$ dla wszystkich i. Na rysunku 1.1 zwizualizowano działanie klasyfikatora SVM.

W celu znalezienia optymalnej hiperpłaszczyzny, SVM stosuje metody optymalizacyjne, takie jak metoda mnożników Lagrange'a. Problem optymalizacyjny polega na maksymalizacji marginesu poprzez minimalizację normy wektora wag $||\mathbf{w}||$, przy jednoczesnym zachowaniu prawidłowej klasyfikacji wszystkich punktów danych. W rzeczywistości, ze względu na potencjalną nieliniowość danych, może być konieczne zastosowanie tzw. „sztuczki jądrowej" (ang. *kernel trick*), która pozwala na wykorzystanie liniowego klasyfikatora w przestrzeni o wyższych wymiarach, gdzie dane mogą być liniowo separowalne. Istnieją różne rodzaje funkcji jądrowych, najpopularniejsze z nich zaprezentowano poniżej.

1. **Jądro liniowe**:
$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}', \tag{1.10}$$
przy wyborze jądra liniowego optymalna hiperpłaszczyzna jest wyznaczana w oryginalnej przestrzeni cech.

2. **Jądro wielomianowe**:
$$K(\mathbf{x}, \mathbf{x}') = \left(\gamma\mathbf{x}^T\mathbf{x}' + r\right)^d, \tag{1.11}$$
gdzie γ, r i d są parametrami.

3. **Jądro radialne (RBF)**:
$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma||\mathbf{x} - \mathbf{x}'||^2\right), \tag{1.12}$$
gdzie γ jest parametrem.

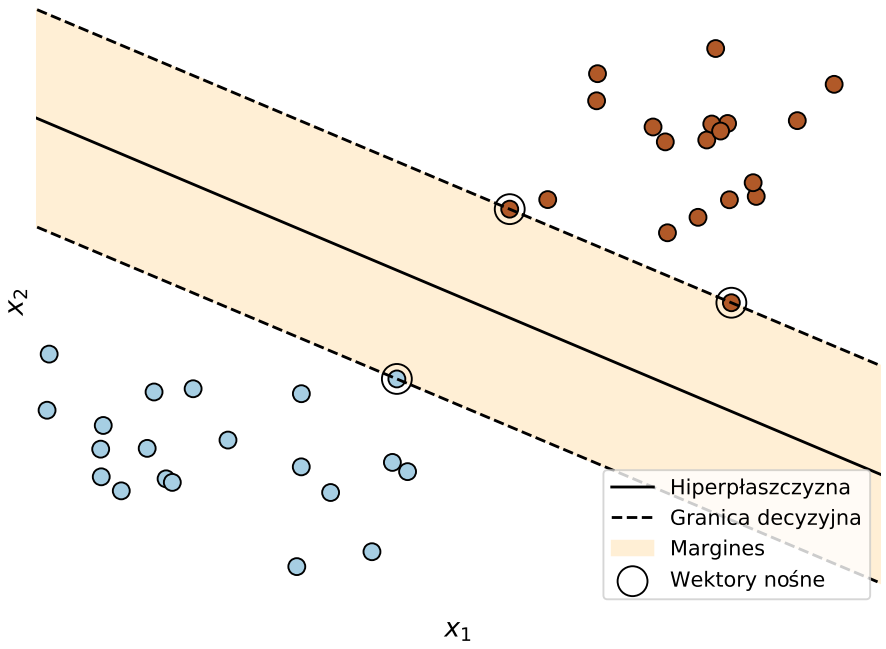
Wybór odpowiednich parametrów funkcji jądrowej (w szczególności parametrów d oraz γ) może mieć istotny wpływ na złożoność (elastyczność) reguły decyzyjnej, a w konsekwencji na skuteczność zbudowanego klasyfikatora. Zastosowanie funkcji jądrowych takich jak RBF lub wielomianowa umożliwia efektywne odwzorowanie danych do nowej przekształconej (nieliniowo) przestrzeni cech. Dzięki temu SVM zyskuje zdolność do tworzenia nieliniowych granic decyzyjnych.

Ze względu na to, że w rzeczywistości dane zazwyczaj nie są w pełni separowalne, wprowadza się zmienne ξ_i, które pozwalają na uwzględnienie pewnych odstępstw od idealnej rozdzielności klas. Problem optymalizacyjny jest wtedy formułowany jako minimalizacja

$$\frac{1}{2}\mathbf{w}'\mathbf{w} + C\sum_{i=1}^{n}\xi_i, \tag{1.13}$$

gdzie C jest parametrem kosztu, który wyraża kompromis między maksymalizacją marginesu, a dokładnością klasyfikacji.

Ostatecznie, rozważany problem optymalizacyjny jest dualny do pierwotnego problemu i jest rozwiązywany w odniesieniu do mnożników Lagrange'a α_i, które odpowiadają punktom danych. Rozwiązanie problemu dualnego pozwala znaleźć optymalne wartości $\mathbf{w}$ i w_0, które definiują hiperpłaszczyznę rozdzielającą klasy. Wektory nośne są tymi punktami danych, które mają niezerowe mnożniki Lagrange'a i leżą najbliżej hiperpłaszczyzny, wpływając bezpośrednio na jej położenie.



Rysunek 1.1: Wizualizacja binarnego klasyfikatora SVM z zaznaczoną hiperpłaszczyzną, granicami decyzyjnymi, obszarem marginesu i wektorami nośnymi

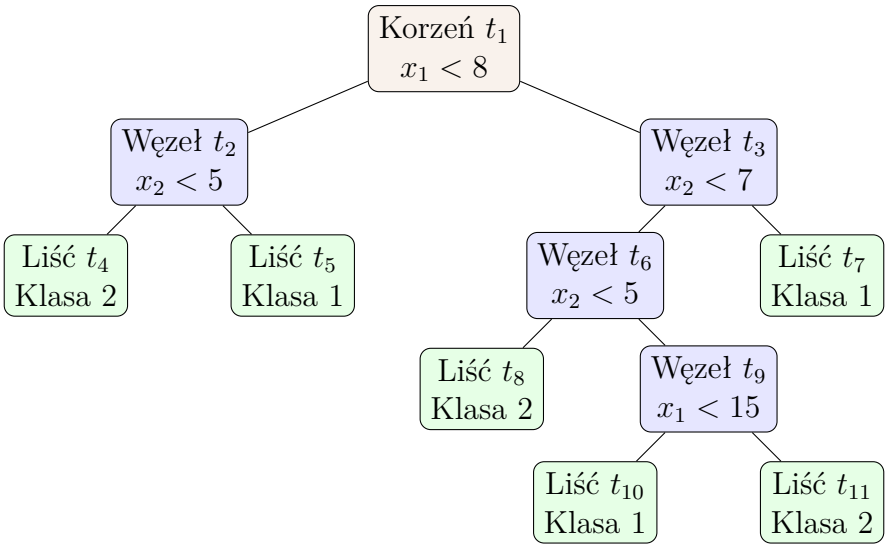
Źródło: opracowanie własne

Drzewo decyzyjne

Drzewo decyzyjne (DT, od ang. *decision tree*) jest popularnym modelem używanym w uczeniu maszynowym, który strukturalnie przypomina drzewo i pozwala na rozwiązywanie zagadnień związanych zarówno z klasyfikacją jak i regresją. Drzewo klasyfikacyjne jest tworzone poprzez rekurencyjny podział całej przestrzeni próby $\mathcal{X}$ na dwa rozłączne podzbiory, tak aby osiągnąć jak największą „jednorodność" w każdym z końcowych podzbiorów, czyli liściach drzewa.

Drzewo klasyfikacyjne, oznaczone jako T, składa się z węzłów t, z których korzeń t_1 jest punktem startowym. Liście, czyli węzły końcowe, które nie podlegają dalszemu podziałowi, tworzą zbiór $\hat{T}$. Funkcja indeksująca ind przypisuje każdemu liściowi dokładnie jedną etykietę klasy, a funkcja podziału s dla każdego węzła wewnętrznego określa, czy dana obserwacja zostanie przekierowana do lewego, czy prawego węzła potomnego. Na rysunku 1.2 przedstawiono przykładowe drzewo klasyfikacyjne. Klasyfikator zbudowany na podstawie drzewa decyzyjnego T można opisać jako

$$\hat{d}_{DT}(\mathbf{x}) = \sum_{t \in \hat{T}} ind(t) \cdot \mathbf{1}_{\{\mathbf{x} \in t\}}. \tag{1.14}$$



Rysunek 1.2: Przykładowe drzewo klasyfikacyjne o 11 węzłach. W węzłach wewnętrznych naniesiono kryteria podziału s, a liściom przyporządkowano etykiety klas

Źródło: opracowanie własne

Las Losowy

Las losowy (RF, od ang. *random forest*) jest algorytmem zespołowym, który wykorzystuje rodzinę modeli opartych na drzewach decyzyjnych, do osiągnięcia lepszej stabilności i dokładności w zadaniach klasyfikacyjnych lub regresyjnych. Metoda ta jest oparta na technice *bagging* (ang. *bootstrap aggregating*), która polega na tworzeniu wielu replikacji bootstrapowych z oryginalnego zbioru uczącego, a następnie trenowaniu niezależnych klasyfikatorów na każdej z tych prób.

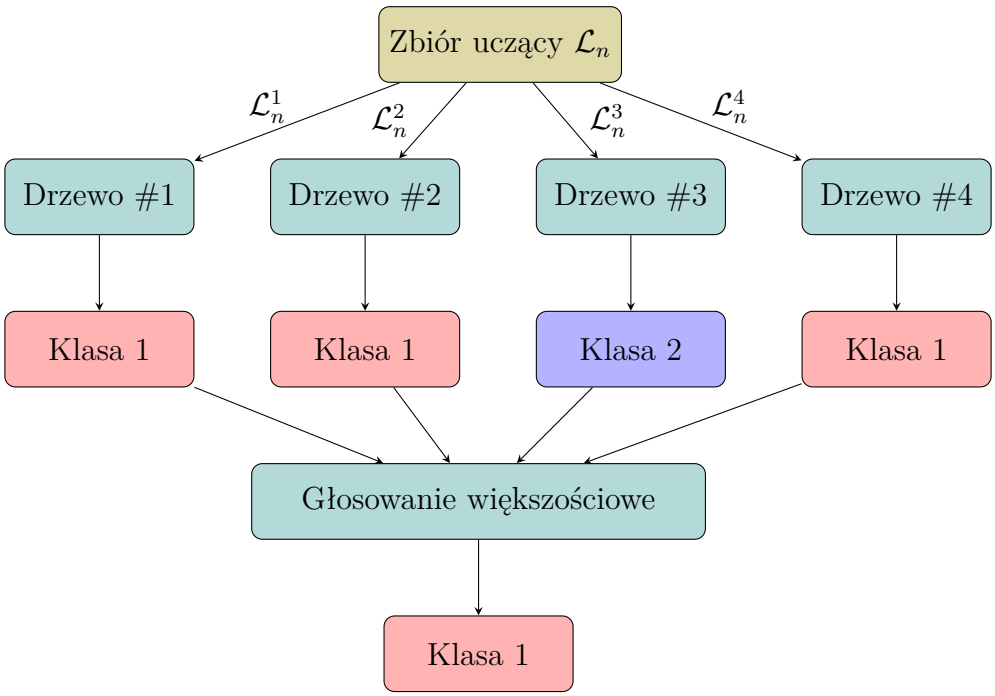
Istotnym aspektem RF, odróżniającym go od zwykłego algorytmu *bagging*, jest losowy wybór ustalonej liczby cech (zwykle mniejszej niż oryginalna liczba cech p; domyślnie przyjmuje się $m \approx \sqrt{p}$) do budowy poszczególnych drzew bazowych. Dzięki temu każde drzewo jest konstruowane z użyciem innej kombinacji cech, co przyczynia się do redukcji korelacji pomiędzy poszczególnymi drzewami klasyfikacyjnymi wchodzącymi w skład lasu. Dodatkowo, cechy te są wybierane niezależnie dla każdego węzła w drzewie, a na podstawie tych wybranych cech wyznaczany jest następnie optymalny podział tego węzła. Takie podejście w większym stopniu zmniejsza korelację między drzewami, co przekłada się na lepszą ogólną wydajność modelu.

Klasyfikacja na bazie lasu losowego odbywa się przez głosowanie większościowe, wykorzystujące prognozowane etykietki klas dla poszczególnych drzew. Na rysunku 1.3 przedstawiono ideę działania lasu losowego, zbudowanego z czterech drzew decyzyjnych. Podkreślmy, że w schemacie typu *bagging* każda próba bootstrapowa jest tworzona poprzez losowanie z oryginalnego zestawu danych z powtórzeniami. Oznacza to, że niektóre obserwacje mogą pojawić się więcej niż raz w jednej próbie, a niektóre wcale. Ostatecznie proces budowy lasu losowego wygląda następująco

1. Dla każdej iteracji $b = 1, 2, \ldots, B$, gdzie B to liczba klasyfikatorów (drzew) w lesie:
   (a) Pobierz próbę bootstrapową $\mathcal{L}_n^b$ z oryginalnej próby uczącej $\mathcal{L}_n$.
   (b) Skonstruuj klasyfikator $\hat{d}_b$ na podstawie pobranej próby bootstrapowej $\mathcal{L}_n^b$.

2. Klasyfikuj obserwację $\mathbf{x}$ poprzez agregację predykcji wszystkich klasyfikatorów $\hat{d}_b(x)$ według reguły większościowej, która przypisuje obserwacji tę klasę, która otrzymała najwięcej głosów

$$\hat{d}_{\mathrm{RF}}(\mathbf{x}) = \arg\max_{j \in \mathcal{Y}} \sum_{b=1}^{B} \mathbf{1}_{\left\{\hat{d}_b(\mathbf{x})=j\right\}}. \tag{1.15}$$



Rysunek 1.3: Schemat lasu losowego (ang. *Random Forest*) z czterema klasyfikatorami

Źródło: opracowanie własne

1.1.3 Miary wykorzystywane do oceny skuteczności klasyfikacji

Ocena i porównanie skuteczności algorytmów klasyfikacyjnych wymaga określenia adekwatnych miar (kryteriów), które będą wykorzystane do oceny trafności przewidywań. Większość popularnych miar opiera się na macierzy (lub tabeli) pomyłek (ang. *confusion matrix*). Aby wprowadzić formalne definicje tych kryteriów, załóżmy, że rozważamy zagadnienie klasyfikacji binarnej, ustalając jedną z klas (zwykle tę ważniejszą) jaką tzw. klasę pozytywną (ang. *positive*), a drugą jako klasę negatywną (ang. *negative*). Dzięki temu jesteśmy w stanie formalnie zdefiniować pojęcie macierzy pomyłek.

Definicja 1.2. (Macierz pomyłek) [2]
W przypadku klasyfikacji binarnej, macierz pomyłek (ang. *confusion matrix*), przedstawiona w tab. 1.1, jest tabelą 2×2, której kolumny odpowiadają rzeczywistym klasom rozważanych obiektów, a wiersze klasom przewidzianym przez klasyfikator. Macierz ta składa się z następujących elementów.

- Przypadki prawdziwie pozytywne (ang. *True Positives*, TP) — klasa rzeczywista instancji jest pozytywna i klasyfikator poprawnie przewiduje klasę pozytywną.
- Przypadki fałszywie pozytywne (ang. *False Positives*, FP) — klasa rzeczywista instancji jest negatywna, ale klasyfikator błędnie przewiduje klasę pozytywną.

- Przypadki prawdziwie negatywne (ang. *True Negatives*, TN) — klasa rzeczywista instancji jest negatywna i klasyfikator poprawnie przewiduje klasę negatywną.
- Przypadki fałszywie negatywne (ang. *False Negatives*, FN) — klasa rzeczywista instancji jest pozytywna, ale klasyfikator błędnie przewiduje klasę negatywną.

		Klasa rzeczywista	
		pozytywna (+)	negatywna (−)
Klasa przewidywana	pozytywna (+)	TP	FP
	negatywna (−)	FN	TN

Tabela 1.1: Macierz pomyłek

Wykorzystując definicję 1.2 możemy teraz określić popularne miary trafności klasyfikacji, takie jak: dokładność, precyzja, czułość i miara F1.

Definicja 1.3. Dokładność [2]
Dokładność (ang. *accuracy*) to stosunek liczby poprawnie zaklasyfikowanych instancji do całkowitej liczby instancji w zbiorze danych.

$$\text{dokładność} = \frac{TP + TN}{TP + FP + FN + TN}, \tag{1.16}$$

gdzie TP, FP, FN, TN są określone jak w Definicji 1.2.

Zakres wartości tej miary to [0, 1], gdzie 0 oznacza całkowitą niepoprawność klasyfikacji, a 1 idealną klasyfikację. Dokładność dostarcza ogólną informację o skuteczności modelu, lecz może być myląca w przypadku niezrównoważonych klas.

Definicja 1.4. Precyzja [2]
Precyzja (ang. *precision*) to stosunek liczby prawdziwie pozytywnych instancji do sumy liczby prawdziwie pozytywnych i fałszywie pozytywnych instancji.

$$\text{precyzja} = \frac{TP}{TP + FP}, \tag{1.17}$$

gdzie TP i FP to odpowiednio liczba prawdziwie pozytywnych i fałszywie pozytywnych instancji, zgodnie z Definicją 1.2.

Zakres wartości tej miary to [0, 1], gdzie wartość 1 oznacza, że wszystkie klasyfikacje pozytywne są poprawne, a 0, że żadna z klasyfikacji pozytywnych nie jest poprawna. W medycynie, wysoka precyzja oznacza, że większość przypadków zidentyfikowanych jako pozytywne odpowiada osobom faktycznie chorym.

Definicja 1.5. Czułość [2]
Czułość (ang. *sensitivity* lub *recall*) to stosunek liczby prawdziwie pozytywnych instancji do sumy prawdziwie pozytywnych i fałszywie negatywnych instancji.

$$\text{czułość} = \frac{TP}{TP + FN}, \tag{1.18}$$

gdzie TP to liczba prawdziwie pozytywnych, a FN to liczba fałszywie negatywnych instancji, odnosząc się do Definicji 1.2.

Zakres wartości tej miary to [0, 1], gdzie wysoka czułość wskazuje na dobrą zdolność modelu do wykrywania pozytywnych przypadków. Jest szczególnie ważna w kontekście zagadnień związanych z diagnostyką medyczną, gdzie niewykrycie choroby choroby (FN) może mieć poważne konsekwencje dla zdrowia pacjenta.

Definicja 1.6. Miara F1 [2]
Miara F1 (ang. *F1 score*) jest harmoniczną średnią precyzji i czułości.

$$\text{miara F1} = \frac{2}{\frac{1}{\text{Precyzja}} + \frac{1}{\text{Czułość}}} = \frac{2TP}{2TP + FP + FN}, \tag{1.19}$$

gdzie precyzja i czułość są określone jak w Definicjach 1.4 i 1.5.

Zakres wartości tej miary to [0, 1], gdzie wartość 1 oznacza idealną równowagę między precyzją a czułością. Miara F1 jest szczególnie użyteczna w sytuacjach, gdzie równie ważne jest unikanie fałszywie negatywnych, jak i fałszywie pozytywnych wyników. Z tego względu jest to również bardziej odpowiednie kryterium skuteczności w przypadku danych niezbalansowanych niż dokładność, zdefiniowana w (1.16).

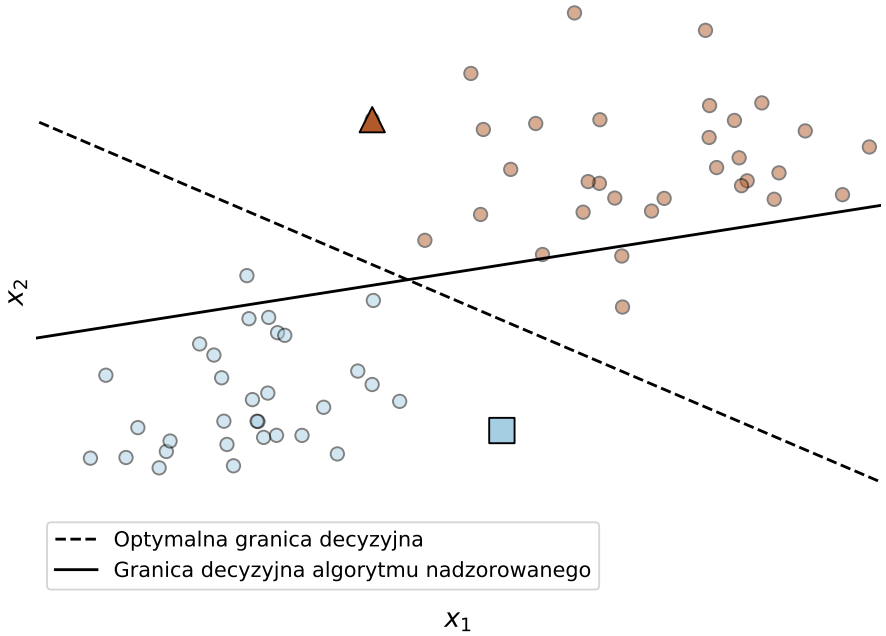
1.2 Uczenie częściowo nadzorowane

1.2.1 Wprowadzenie

Zdobycie danych etykietowanych (ang. *labeled data*) może być problematyczne, kosztowne lub czasochłonne w przypadku konieczności ręcznego oznaczania przez specjalistów. Tymczasem dane nieetykietowane (ang. *unlabeled data*) są zazwyczaj łatwiejsze do pozyskania. Uczenie częściowo nadzorowane (SSL, od ang. *semi-supervised learning*) rozwiązuje ten problem, łącząc w sobie elementy uczenia nadzorowanego i nienadzorowanego. Zamiast opierać się tylko na zbiorze $\mathcal{L}_n = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$, gdzie każdy przypadek składa się z wektora cech $\mathbf{x_i}$ i odpowiadającej mu etykiety y_i, bądź tylko na zbiorze obserwacji nieetykietowanych $\mathcal{U}_m = \{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$, uczenie częściowo nadzorowane wykorzystuje zarówno dane $\mathcal{L}_n$ jak i $\mathcal{U}_m$. Należy podkreślić, że liczba nieetykietowanych obserwacji m jest zazwyczaj znacznie większa niż liczba przypadków nieetykietowanych n, przy czym $m \gg n$. W ogólności, naszym celem jest skonstruowanie bardziej skutecznej reguły decyzyjnej (klasyfikatora) na bazie $\mathcal{L}_n$ i $\mathcal{U}_m$ niż wykorzystując jedynie obserwacje etykietowane $\mathcal{L}_n$.

Istnieje wiele spotykanych w praktyce zagadnień, w przypadku których mamy zwykle do czynienia z dużą liczbą obserwacji nieetykietowanych i ich wykorzystanie może pomóc w budowaniu efektywnych modeli klasyfikacyjnych. Przykładem mogą być strony WWW, obrazy, dokumenty oraz dane biomedyczne. W pierwszym przypadku, etykietowanie wymaga czytania lub oglądania i namysłu, a czasami wspomagane jest dodatkowymi źródłami, podczas gdy nieetykietowane dane są często dostępne w większej ilości i mogą być automatycznie pozyskiwane przy minimalnych kosztach. W kontekście danych biomedycznych, związanych np. z predykcją struktur genetycznych, kategoryzacja poszczególnych próbek wymaga głębokiej wiedzy eksperckiej i jest często procesem długotrwałym, podczas gdy nowoczesne urządzenia diagnostyczne, takie jak sekwenatory DNA, są w stanie generować ogromne ilości danych eksperymentalnych, które mogą być wykorzystane zgodnie ze schematem uczenia częściowo nadzorowanego, ułatwiając i przyspieszając tym samym proces badawczy.

Analogią dla działania uczenia częściowo nadzorowanego może być ludzki mechanizmach poznawczy. Człowiek od najmłodszych lat angażuje się w proces klasyfikacji i kategoryzacji obiektów, wykorzystując często ograniczone informacje i wskazówki. Przykładem może być sposób, w jaki dzieci uczą się rozpoznawać gatunki zwierząt. W kontekście zmiennej $\mathbf{x}$ reprezentującej cechy (np. zwierząt) oraz zmiennej y określającej kategorię (np. gatunek — pies), proces nauki odbywa się poprzez interakcje z nauczycielem-rodzicem, który wskazuje i definiuje obiekt: „*To jest pies*". Następnie, dziecko samo, bez dalszych wyjaśnień, obserwuje inne zwierzęta i dokonuje ich klasyfikacji na podstawie wcześniej nabytej wiedzy oraz własnych obserwacji. Analogiczny proces ma miejsce w SSL, gdzie model uczenia maszynowego wykorzystuje ograniczoną liczbę etykietowanych przykładów (tj. danych, dla których znamy odpowiednią kategorię), aby nauczyć się klasyfikowania nowych, nieetykietowanych danych. Podobnie jak dziecko wykorzystuje obserwacje nieobjaśnionych przypadków do rozwijania własnych umiejętności klasyfikacji, tak modele SSL wykorzystują nieetykietowane dane do ulepszania swoich predykcji.



Rysunek 1.4: Porównanie granicy decyzyjnej uzyskanej przez algorytm nadzorowany (ang. *supervised*) dla danych etykietowanych, z optymalną granicę decyzyjną. Przezroczyste, okrągłe punkty to dane uznane za nieetykietowane, które zostały oznaczone zgodnie ze swoją prawdziwą przynależnością do klas

Źródło: opracowanie własne

Aby lepiej zrozumieć potencjalne korzyści wynikające z wykorzystania dodatkowych danych nieetykietowanych, przedstawimy teraz poglądowy przykład. Na wykresie 1.4 dokonano porównania granicy decyzyjnej wyznaczonej na bazie algorytmu uczenia nadzorowanego (ang. *supervised learning*) oraz optymalnej granicy decyzyjnej. Przezroczyste, okrągłe punkty oznaczają dane, które zostały uznane za nieetykietowane. W rzeczywistości

mają one przypisane etykietki klas, jednak zakładamy, że w procesie uczenia algorytm nie wykorzystuje tej informacji. Linia ciągła przedstawia granicę decyzyjną utworzoną przez metodę nadzorowaną, która oddziela dwie klasy jedynie na podstawie dostępnych dwóch etykietowanych punktów. Przerywana linia wskazuje na optymalną granicę decyzyjną, która najlepiej oddzielałaby klasy. Wykres ten podkreśla potencjał i zarazem wskazuje na istotność informacji, jakie niosą ze sobą dane nieetykietowane.

Uczenie częściowo nadzorowane wymaga mniej wysiłku analityka (rezygnujemy z czasochłonnego i często kosztownego etykietowania danych) i może prowadzić do wyższej dokładności, co sprawia, że jest to obszar o dużym znaczeniu zarówno teoretycznym, jak i praktycznym. Chociaż nieetykietowane dane nie zawsze muszą być pomocne i ich wpływ może być złożony, to odpowiednie zastosowanie metod SSL może pozwolić na skuteczniejsze wykorzystanie dostępnych danych i budowanie bardziej efektywnych modeli klasyfikacyjnych. [11] [12] [14]

1.2.2 Założenia przyjmowane w przypadku uczenia częściowo nadzorowanego

W uczeniu częściowo nadzorowanym przyjmuje się trzy kluczowe założenia, które pomagają w lepszym wykorzystaniu zarówno etykietowanych, jak i nieetykietowanych danych w procesie konstruowania modeli klasyfikacyjnych [12]. Należy podkreślić, że założenia te stanowią podstawę większości algorytmów uczenia częściowo nadzorowanego (SSL), których skuteczność zależy od ich spełnienia.

Założenie bliskości punktów (ang. *smoothness assumption*)

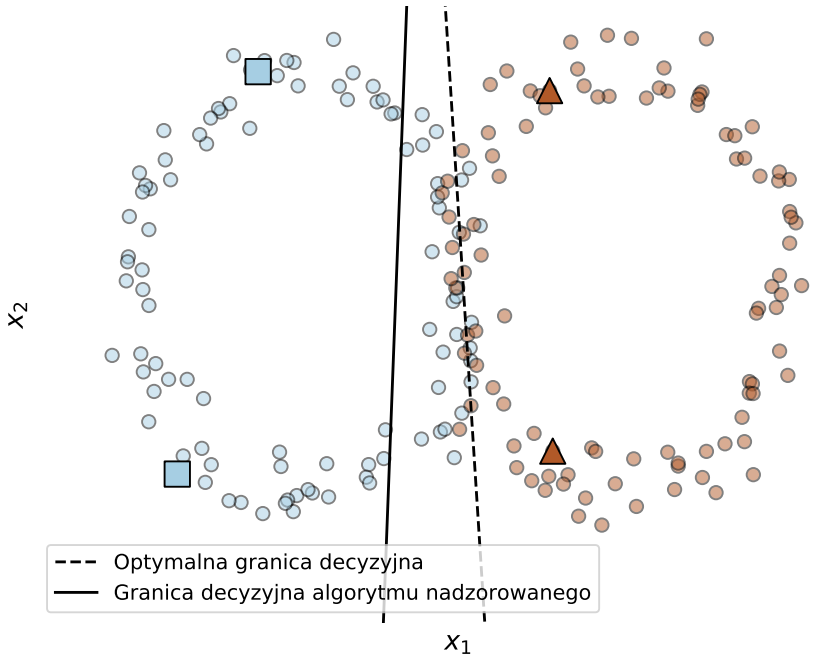
Założenie to opiera się na przekonaniu, że punkty leżące blisko siebie w przestrzeni cech z dużym prawdopodobieństwem będą miały przypisaną tę samą etykietę klasy. Przykładowo, jeżeli punkt danych $\mathbf{x}_1$ należy do zbioru etykietowanego $\mathcal{L}$ i dwa punkty $\mathbf{x}_2$ oraz $\mathbf{x}_3$ są nieetykietowane, $\mathbf{x}_2, \mathbf{x}_3 \in \mathcal{U}$, to jeśli $\mathbf{x}_1$ i $\mathbf{x}_2$ są bliskie w sensie jakiejś zdefiniowanej metryki w przestrzeni cech, a $\mathbf{x}_2$ i $\mathbf{x}_3$ także są bliskie, ale para $\mathbf{x}_1$ i $\mathbf{x}_3$ już nie, wówczas można zakładać, że $\mathbf{x}_3$ powinien posiadać tę samą etykietę co $\mathbf{x}_1$, nawet jeśli bezpośrednie podobieństwo między $\mathbf{x}_1$ i $\mathbf{x}_3$ nie jest silne. Jest to związane z sukcesywną propagacją etykiet, gdzie informacja o przynależności do klasy przebiega przez łańcuch bliskich sobie punktów w przestrzeni cech.

Założenie niskiej gęstości (ang. *low-density assumption*)

W tym przypadku zakładamy, że granica decyzyjna klasyfikatora powinna przebiegać przez regiony przestrzeni charakteryzujące się niską gęstością, tzn. takie, dla których obserwujemy niewielką liczbę punktów danych. Oznacza to, że jeśli $p(\mathbf{x})$ reprezentuje rozkład danych, to granica decyzyjna będzie przebiegała przez obszar, dla którego wartość $p(\mathbf{x})$ będzie niska. To założenie jest powiązane z założeniem o bliskości punktów, ponieważ granica decyzyjna umieszczona w obszarze o wysokiej gęstości danych naruszałaby je, przypisując różne etykiety podobnym obserwacjom. W regionach o wysokiej gęstości, gdzie możemy oczekiwać wielu obserwacji, zakłada się, że punkty będą miały przypisaną tę samą etykietę klasy. Poglądowa sytuacja przedstawiona na rys. 1.4 ilustruje to założenie.

Założenie dotyczące rozmaitości (ang. *manifold assumption*)

W tym przypadku bierzemy pod uwagę fakt, że wymiarowość wielu danych może być sztucznie zawyżona. Obserwacje zazwyczaj reprezentowane są jako punkty w wysoko wymiarowej przestrzeni cech $\mathbb{R}^p$, chociaż często skoncentrowane są one wzdłuż pewnych rozmaitości (ang. *manifold*) o niższym wymiarze, czyli przestrzeni topologicznych, które są lokalnie euklidesowe. Dla przykładu, gdy rozważamy 3-wymiarową przestrzeń cech, w której wszystkie punkty leżą na powierzchni kuli, można powiedzieć, że dane leżą na 2-wymiarowej rozmaitości. W uczeniu częściowo nadzorowanym będziemy zakładali, że punkty danych leżące na tej samej rozmaitości mają przypisaną tę samą etykietę. Dzięki temu nieznane etykietki klas dla obserwacji należących do zbioru $\mathcal{U}$ możemy przypisać na podstawie etykietowanych przypadków należących do tej samej rozmaitości. Rys. 1.5 poglądowo ilustruje to założenie. W tym przypadku wydaje się naturalne, że właściwa granica decyzyjna przebiega pomiędzy dwoma podzbiorami, przypominającymi okręgi.



Rysunek 1.5: Ilustracja założenia dotyczącego rozmaitości (ang. *manifold assumption*). Porównanie granicy decyzyjnej uzyskanej przez algorytm nadzorowany dla danych etykietowanych, z optymalną granicę decyzyjną. Przezroczyste, okrągłe punkty to dane uznane za nieetykietowane, które zostały oznaczone zgodnie ze swoją prawdziwą przynależnością do klas

Źródło: opracowanie własne

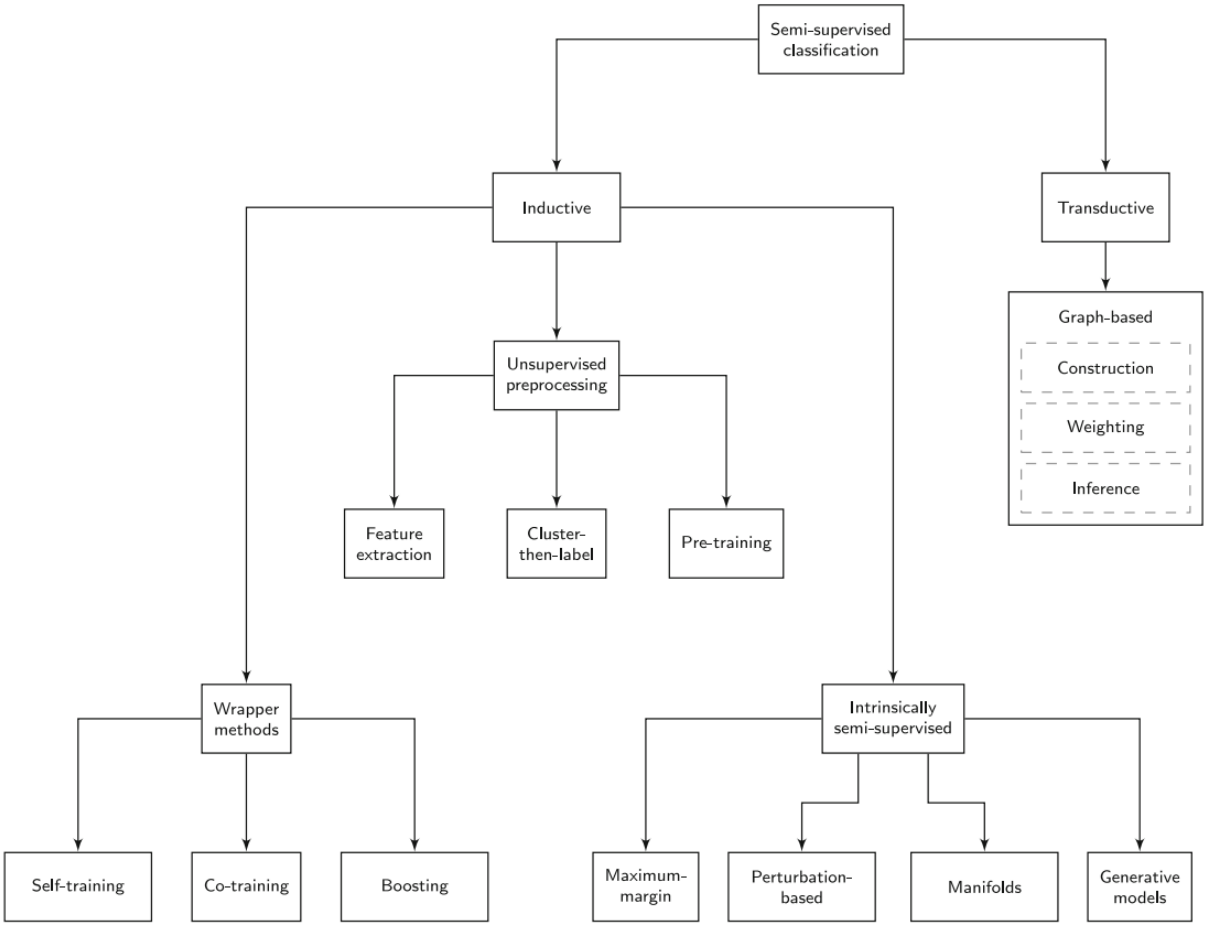
1.2.3 Taksonomia metod uczenia częściowo nadzorowanego

Istnieje wiele algorytmów klasyfikacji częściowo nadzorowanej, które różnią się między sobą przyjętymi założeniami, sposobem wykorzystania nieetykietowanych danych oraz ich powiązaniem z metodami uczenia nadzorowanego. W niniejszej pracy przyjęto taksonomię zaproponowaną w [12], przedstawioną na rysunku 1.6. Kluczowe w niej jest rozróżnienie między metodami indukcyjnymi (ang. *inductive*), a transdukcyjnymi (ang. *transductive*). Metody indukcyjne starają się znaleźć uniwersalny model klasyfikacyjny, który może być stosowany do przewidywania etykiet nieobserwowanych wcześniej danych (tzn. dowolnego wektora $\mathbf{x}$ należącego do przestrzeni cech $\mathcal{X}$), podczas gdy metody transdukcyjne koncentrują się wyłącznie na przewidywaniu etykiet dla konkretnych nieetykietowany punktów.

W ramach metod indukcyjnych wyróżniamy podejścia, które wykorzystują nieetykietowane dane na etapie wstępnego przetwarzania (ang. *unsupervised preprocessing*), co może obejmować ekstrakcję cech (ang. *feature extraction*), grupowanie a następnie etykietowanie (ang. *cluster-then–label*) oraz wstępne trenowanie (ang. *pre-training*). W obrębie metod indukcyjnych znajdują się metody typu *wrapper*, które bezpośrednio wykorzystują nieetykietowane dane w procesie uczenia. Wśród nich wyróżniamy techniki takie jak samouczenie (ang. *self-training*), współucznie (ang. *co-training*) oraz boosting, które różnią się sposobem uwzględnienia danych nieetykietowanych przez algorytmy nadzorowane. Dodatkowo, w przypadku metod indukcyjnych wyróżniamy algorytmy wewnętrznie częściowo nadzorowane (ang. *intrinsically semi-supervised*), które obejmują metody maksymalizacji marginesu (ang. *maximum-margin*), oparte na zaburzeniu danych (ang. *perturbation-based*), rozmaitościach (ang. *manifolds*) oraz modele generatywne (ang. *generative models*).

Z kolei metody transdukcyjne, zazwyczaj oparte na grafach (ang. *graph-based*), są grupowane w oparciu o etapy konstrukcji (ang. *construction*), ważenia (ang. *weighting*) oraz wnioskowania (ang. *inference*) w grafie, które związane są z wykorzystaniem nieetykietowanych danych.

Każda z wymienionych kategorii algorytmów obejmuje różne strategie i techniki wykorzystania danych nieetykietowanych do poprawy skuteczności klasyfikacji. Metody typu *wrapper* rozszerzają istniejące klasyfikatory nadzorowane poprzez dodawanie pseudo-etykiet do danych nieoznaczonych, które następnie są wykorzystywane w procesie ponownego uczenia, umożliwiając klasyfikatorom ulepszanie swoich modeli. Z kolei, podejście *unsupervised preprocessing* skupia się na wstępnym przetwarzaniu nieetykietowanych danych w celu ekstrakcji cech lub wstępnego klastrowania, które później pomagają w treningu klasyfikatorów nadzorowanych, lecz bez bezpośredniego wpływu na proces uczenia klasyfikatora. Metody *intrinsically semi-supervised* natomiast włączają nieetykietowane dane bezpośrednio do funkcji celu lub procedury optymalizacji, co pozwala na bardziej holistyczne wykorzystanie dostępnych danych. W przeciwieństwie do nich, metody *graph-based* działają bezpośrednio na danych uczących, używając grafów do reprezentowania i propagowania informacji (etykiet) między punktami danych. [12] Pomimo różnic w podejściach, wszystkie te metody mają na celu wykorzystanie dostępnych danych nieetykietowanych w celu poprawy wyników klasyfikacji. Przedstawiona taksonomia umożliwia zrozumienie różnorodności istniejących metod.



Rysunek 1.6: Taksonomia metod uczenia częściowo nadzorowanego

Źródło: [12]

1.2.4 Opis zastosowanych algorytmów uczenia częściowo nadzorowanego

Istnieje wiele algorytmów SSL, które zgodnie z przyjętą w podrozdziale 1.2.3 taksonomią można przyporządkować do odpowiednich kategorii, w zależności od zastosowanego podejścia i wykorzystywanej strategii uczenia. W badaniach wykorzystano wybrane algorytmy zaimplementowane w bibliotece LAMDA-SSL, dostępnej w języku Python. Zawiera ona zarówno metody statystyczne, jak i oparte na uczeniu głębokim, co pozwala na efektywne wykorzystanie danych. [3] [4]

W ramach badań przeprowadzonych na potrzeby niniejszej pracy, porównano ze sobą następujące algorytmy uczenia częściowo nadzorowanego: Tri-Training, Assemble, Semi-Boost, LapSVM, TSVM i Label Propagation. Wybrano je m.in. ze względu na łatwość implementacji i obsługi w języku Python. Umożliwiło to również skupienie się na większej liczbie różnorodnych podejść do SSL, które są reprezentatywne dla tej dziedziny, lecz nie wymagają złożoności obliczeniowej charakterystycznej dla metod uczenia głębokiego, co jest istotne ze względu na ograniczony czas przeznaczony na symulacje. Poniżej zamieszczono krótkie opisy wybranych metod, opierające się na [12], [14] oraz dokumentacji opracowanej dla biblioteki LAMDA-SSL [3].

Tri-Training

Algorytm Tri-Training, należący do grupy *co-training* (porównaj rys. 1.6), jest metodą uczenia częściowo nadzorowanego, którego podstawą jest wykorzystanie trzech klasyfikatorów bazowych. W przypadku tego podejścia są one trenowane naprzemiennie, a proces uczenia opiera się na idei wzajemnej walidacji i wykorzystywania przewidywań dwóch klasyfikatorów do trenowania trzeciego.

Każdy z trzech klasyfikatorów jest trenowany na niezależnym, losowo wybranym podzbiorze danych treningowych, co zwiększa różnorodność i poprawia zdolności uogólniające modeli. Jeśli dwa z trzech klasyfikatorów osiągną konsensus co do przewidywanej etykiety dla danego punktu danych, etykieta ta jest przekazywana do trzeciego klasyfikatora. Takie postępowanie pozwala na iteracyjne wzbogacanie zestawu danych treningowych o nowe, nieoznakowane wcześniej próbki.

Podstawową zaletą algorytmu Tri-Training jest to, że nie opiera się on na probabilistycznych przewidywaniach pojedynczych klasyfikatorów (tzn. wykorzystuje jedynie prognozowane etykietki klas), co pozwala na zastosowanie tej metody dla szerokiej gamy algorytmów uczenia nadzorowanego. Warto zwrócić uwagę, że efektywność metody Tri-Training może być zwiększona poprzez zastosowanie heterogenicznych klasyfikatorów, zamiast homogenicznych modeli tego samego rodzaju. W ten sposób wprowadza się dodatkową różnorodność i zwiększa odporność procedury na błędy.

W przeprowadzonej analizie porównawczej metod SSL (patrz rozdział 2), będziemy analizowali m.in. to, jak wybór różnych konfiguracji klasyfikatorów bazowych wpływa na skuteczność tego algorytmu. W szczególności, uwzględnione zostaną (opisane w podrozdziale 1.1.2) metody: KNN, Naive Bayes, Decision Tree, Random Forest, SVM.

Poniżej przedstawiono szczegółowy opis algorytmu, na podstawie [10].

1. **Inicjalizacja klasyfikatorów.** Algorytm inicjuje trzy klasyfikatory m_i oraz trzy podzbiory S_i, pochodzące ze zbioru danych etykietowanych L.

2. **Trening początkowy.** Każdy klasyfikator m_i jest trenowany na swoim podzbiorze danych S_i.

3. **Iteracyjne przewidywanie i trenowanie.** Dla każdego klasyfikatora m_i, w każdej iteracji wykonuje się następujące kroki:

   (a) Reset zbioru L_i. Tworzony jest pusty zbiór L_i, który będzie przechowywał nowe etykiety dla danych nieoznaczonych.

   (b) Etykietowanie danych nieoznaczonych. Dla każdego elementu $\mathbf{x}$ z nieoznaczonego zbioru U, sprawdza się, czy dwa pozostałe klasyfikatory (m_j, m_k) zgadzają się co do etykiety $p(\mathbf{x})$. Jeżeli tak, czyli $p_j(\mathbf{x}) = p_k(\mathbf{x})$, to przykład $\mathbf{x}$ wraz z etykietą $p_j(\mathbf{x})$ jest dodawany do zbioru L_i

   $$L_i \cup \{(\mathbf{x}, p_j(\mathbf{x}))\}.$$

   (c) Aktualizacja klasyfikatora. Klasyfikator m_i jest trenowany ponownie na połączonym zbiorze etykietowanych danych L oraz nowo etykietowanych danych L_i

   $$L \cup L_i.$$

4. **Warunek stopu.** Iteracje są kontynuowane do momentu, aż żaden z klasyfikatorów m_i nie zmienia swoich przewidywań na zbiorze U.

5. **Głosowanie większościowe.** Po zakończeniu iteracji, na każdym przykładzie ze zbioru U stosowane jest głosowanie większościowe obejmujące trzy klasyfikatory, aby ustalić ostateczną etykietę.

Assemble

Algorytm Assemble, skrót od ang. *Adaptive Semi-Supervised Ensemble*, to metoda uczenia częściowo nadzorowanego (SSL), należąca do grupy *boosting* (patrz rys. 1.6), która wykorzystuje pseudo-etykietowanie (ang. *pseudo-labeling*) nieoznakowanych punktów danych po każdej iteracji.

Metoda polega na adaptacyjnym tworzeniu zespołu klasyfikatorów, poprzez iteracyjne przypisywanie i aktualizowanie wag nieoznakowanych próbek danych w celu poprawy dokładności klasyfikacji. Proces rozpoczyna się od przypisania początkowych wag i tymczasowych etykiet dla nieoznakowanych próbek, a następnie w kolejnych iteracjach trenowany jest nowy klasyfikator na bazie danych z aktualnie przypisanymi wagami. Jego wynikom (przewidywaniom) przypisywane są odpowiednie wagi i są one dodawane do agregowanej funkcji klasyfikacyjnej. Jeśli wynik klasyfikatora wskazuje na niską skuteczność predykcji dla danej próbki, proces jest przerywany i zwracany jest aktualny zespół klasyfikatorów. W przeciwnym razie, proces jest kontynuowany z uwzględnieniem nowego klasyfikatora, który właśnie został wytrenowany. Ostatecznie, etykiety nieoznakowanych próbek są aktualizowane na podstawie wyników agregowanej funkcji klasyfikacyjnej, zmieniając przy tym rozkład wag. [1]

Podstawowe elementy konfiguracji metody Assemble, decydujące o jej efektywności, to wybór klasyfikatora bazowego oraz ustalenie liczby iteracji, oznaczonej jako T. Wyselekcjonowanie adekwatnego klasyfikatora bazowego jest kluczowe, gdyż to on determinuje zdolność modelu do radzenia sobie ze złożonością danych — ich różnorodnością albo jednorodnością. Z tego względu, jego dobór ma bezpośredni wpływ na efektywność całego zespołu klasyfikatorów. Spektrum potencjalnych modeli bazowych jest szerokie i otwarte, umożliwiając zastosowanie praktycznie każdego algorytmu uczenia maszynowego, od prostych metod, takich jak drzewa decyzyjne czy regresja logistyczna, aż po skomplikowane techniki, jak na przykład sieci neuronowe. Z kolei parametr T, definiujący liczbę generowanych klasyfikatorów bazowych, ma bezpośredni wpływ na złożoność końcowego modelu oraz jego zdolności uogólniające.

SemiBoost

Algorytm SemiBoost (grupa *boosting*) charakteryzuje się połączeniem elementów uczenia grafowego z boostingiem. W przeciwieństwie do algorytmu Assemble, który wykorzystuje jedynie różnice między prognozowanymi, a rzeczywistymi etykietami lub pseudo-etykietami do przypisywania wag poszczególnym przypadkom, SemiBoost dodatkowo uwzględnia podobieństwo między danymi. W tym celu wykorzystywany jest graf sąsiedztwa (ang. *predefined neighbourhood graph*). Każdy punkt danych (zarówno oznaczony, jak i nieoznaczony) jest reprezentowany jako węzeł w grafie, a krawędzie grafu reprezentują podobieństwo między tymi punktami. Algorytm przyznaje większą wagę próbkom leżącym blisko w przestrzeni cech oraz tym o dużej rozbieżności pod względem przewidywanych etykiet, w stosunku do innych obserwacji.

Algorytm SemiBoost, z jednej strony, używa standardowego modelu klasyfikacji typu *boosting*, wyrażając ostateczną predykcję etykiety jako liniową kombinację predykcji poszczególnych klasyfikatorów. Z drugiej strony, algorytm podczas każdej iteracji buduje

nowy słaby klasyfikator (ang. *weak learner*), którego zadaniem jest minimalizacja funkcji kosztu składającej się z dwóch wyrazów. Pierwszy składnik kary dotyczy rozbieżności między pseudo-etykietami próbek nieetykietowanych, a rzeczywistymi etykietami próbek etykietowanych, wyznaczonej na podstawie bliskości odpowiednich obserwacji w przestrzeni cech. Drugi składnik wykorzystuje tylko próbki nieetykietowane i za pomocą wag podobieństwa nakłada odpowiednią karę za błędne predykcje. Ostatecznie, celem jest „nagradzanie" klasyfikatorów, które konsekwentnie przypisują etykietki punktom danych leżących na tej samej rozmaitości (porównaj *manifold assumption* w podrozdziale 1.2.2). Omawiana funkcja kosztu formalnie określona jest w następujący sposób

$$\arg\min_{F_T} \; L_L\left(\hat{\mathbf{y}}, A, F_T\right) + \lambda \cdot L_U\left(\hat{\mathbf{y}}, A, F_T\right), \tag{1.20}$$

gdzie L_L i L_U to funkcje kosztu wyrażające niezgodność, odpowiednio, wśród połączonych etykietowanych i nieetykietowanych danych oraz wyłącznie nieetykietowanych danych. Parametr $\lambda \in \mathbb{R}$ to stała, wpływająca na wagę składnika L_U. Macierz A jest symetryczną macierzą $n \times n$ sąsiedztwa. Wreszcie, F_T oznacza wspólną funkcję predykcji zespołu klasyfikatorów w chwili T.

Kluczowe parametry[1] konfiguracyjne metody SemiBoost, które są istotne dla jej skuteczności, obejmują wybór klasyfikatora bazowego, typ jądra podobieństwa oraz liczne parametry kontrolujące konstrukcję i ewolucję modelu. Wybór klasyfikatora bazowego jest fundamentalny, gdyż od jego charakterystyk zależy zdolność modelu do odpowiedniej reakcji na specyfikę danych, ich złożoność oraz strukturę. Jądro podobieństwa (ang. *similarity kernel*) może być funkcją jądra RBF lub opartą na najbliższych sąsiadach. Określa ono sposób, w jaki obliczane jest podobieństwo między punktami danych, co bezpośrednio wpływa na konstrukcję grafu sąsiedztwa. Dodatkowo, parametr T określa maksymalną liczbę modeli w zespole oraz liczbę iteracji algorytmu. Ten parametr ma bezpośredni wpływ na zdolność modelu do generalizacji i złożoność końcowego klasyfikatora zespołowego. Inne parametry, takie jak liczba sąsiadów dla jądra KNN, parametr gamma dla jądra RBF, umożliwiają dokładne dostosowanie działania metody do danych i oczekiwanych rezultatów, zwiększając tym samym jej elastyczność i efektywność.

LapSVM

Tradycyjne warianty metody SVM (patrz podrozdział 1.1.2) skupiają się na przestrzeni cech, nie uwzględniając głębszych zależności strukturalnych między próbkami. LapSVM (ang. *Laplacian Support Vector Machine*), należący do grupy *manifolds* (patrz rys. 1.6), rozszerza klasyczny algorytm SVM, wprowadzając dodatkowy składnik regularyzacyjny Laplace'a do funkcji celu, co pozwala uwzględnić ważne charakterystyki danych leżących na rozmaitościach.

W tym przypadku składnik regularyzacyjny oparty na rozmaitości (ang. *manifold regularization*) jest określony jako

$$\|f\|_I^2 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} \cdot (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2, \tag{1.21}$$

gdzie W to symetryczna, ważona macierz sąsiedztwa, której elementami są podobieństwa między punktami danych (w szczególności $W_{ij} = 0$ jeżeli punkty nie są połączone), a $f(\mathbf{x}_i)$ to predykcja modelu dla punktu danych $\mathbf{x}_i$. Składnik regularyzacyjny Laplace'a możemy

[1] Dostępne w implementacji algorytmu SemiBoost w bibliotece LAMDA-SSL.

przedstawić jako $\mathbf{f}^T \cdot L \cdot \mathbf{f}$, dla którego L jest macierzą Laplace'a $L = D - W$, z diagonalną macierzą D będącą macierzą liczby sąsiadów (ang. *degree matrix*) $D_{ii} = \sum_{j=1}^{n} W_{ij}$, a $\mathbf{f} \in \mathbb{R}^n$ jest wektorem predykcji f dla każego $\mathbf{x}_i$. Ostatecznie, funkcja celu dla algorytmu LapSVM przyjmuje postać

$$\arg\min_f \ \frac{1}{l} \sum_{i=1}^{l} \ell\left(f(\mathbf{x}_i), y_i\right) + \gamma \cdot \|f\|_K^2 + \gamma_U \cdot \|f\|_I^2, \tag{1.22}$$

gdzie ℓ to funkcja straty mierząca błąd predykcji, a γ i γ_U to parametry określające wpływ składników regularyzacyjnych.

Podsumowując, algorytm buduje model grafowy na podstawie wszystkich dostępnych obserwacji, wykorzystując macierz wag określającą podobieństwo między próbkami. Regularyzacja Laplace'a wpływa na wyniki klasyfikacji w taki sposób, aby były one jak najbardziej spójne z etykietkami klas dla sąsiadujących obserwacji w grafie. W ten sposób maksymalizowany jest margines między hiperpłaszczyzną rozdzielającą, a wektorami nośnymi oraz zapewniona spójność przewidywań dla punktów leżących na danej rozmaitości. Podczas implementacji algorytmu LapSVM, istotnymi parametrami, na które należy zwrócić szczególną uwagę, są funkcja odległości oraz funkcja jądra. Funkcja odległości jest kluczowa dla konstrukcji grafu sąsiedztwa, ponieważ jej porawny wybór pozwala odzwierciedlić złożoność przestrzeni cech. Z kolei funkcja jądra odgrywa zasadniczą rolę w procesie transformacji danych do przestrzeni o wyższej wymiarowości, co jest charakterystyczne dla klasycznej metody SVM.

TSVM

Algorytm TSVM (ang. *Transductive Support Vector Machine*) wykorzystuje podejście transdukcyjne, które polega na prognozowaniu etykiet dla nieoznakowanych próbek danych i znalezieniu hiperpłaszczyzny rozdzielającej, która maksymalizuje odległość od wektorów nośnych. Początkowo, TSVM wykorzystuje wszystkie dostępne oznakowane dane do trenowania wyjściowego klasyfikatora SVM. Następnie rozpoczyna się proces iteracyjny, w którym poszukiwane są pary nieoznakowanych próbek, które mogły zostać błędnie zaklasyfikowane. Następuje modyfikacja ich etykiety, a następnie ponownie trenowany jest klasyfikator w celu skorygowania tych przewidywań. Dostosowywanie i ponowne trenowanie trwa aż do osiągnięcia zbieżności. Kluczową cechą TSVM jest zastosowanie różnych wartości kar C dla oznakowanych i nieoznakowanych próbek, co uwzględnia różną wagę przypisywaną tym dwóm rodzajom danych w procesie optymalizacji. W ten sposób, algorytm stopniowo zwiększa wpływ nieoznakowanych próbek na klasyfikator, poprzez zmianę wartości C_L i C_U w celu uzyskania optymalnego rozwiązania.

Warto podkreślić, że TSVM nie wykorzystuje żadnych założeń o strukturze danych i koncentruje się na bezpośredniej propagacji etykiet między oznakowanymi, a nieoznakowanymi próbkami. Jest to inne podejście niż w przypadku algorytmu LapSVM, który uwzględnia wiedzę o geometrii danych poprzez regularyzację Laplace'a, kładąc tym samym nacisk na zachowanie spójności strukturalnej w danych. W przypadku implementacji tego algorytmu, jego najważniejszym parametrem, podobnie jak w przypadku klasycznego SVM, jest wybór funkcji jądrowej.

Label Propagation

Propagacja etykiet (ang. *Label Propagation*) to iteracyjny algorytm, który rozpowszechnia (propaguje) prognozowane etykiety klas między węzłami grafu, opierając się na wagach

krawędzi. Metoda należy do grupy *Inference in graphs* (porównaj rys. 1.6). Celem algorytmu jest propagacja etykiet z danych etykietowanych na dane nieetykietowane poprzez macierz sąsiedztwa A i minimalizację średniego ważonego błędu kwadratowego wyznaczonego dla par sąsiadujących węzłów. Macierz A jest znormalizowaną macierzą wag, gdzie każdy wiersz reprezentuje rozkład prawdopodobieństwa przejść do innych węzłów z danego węzła

$$A_{ij} = \frac{W_{ij}}{\sum_{k \in N(v_i)} W_{ik}}, \tag{1.23}$$

gdzie W_{ij} to waga krawędzi łączącej węzły i i j w grafie, $\sum_{k \in N(v_i)} W_{ik}$ to suma wag wszystkich krawędzi wychodzących z węzła v_i, służąca jako stała normalizacyjna, dzięki której sumy elementów w każdym wierszu macierzy A są równe 1. Co ważne, metoda ma gwarancję zbieżności, co jest dużą zaletą algorytmu Label Propagation w porównaniu do innych metod uczenia częściowo nadzorowanego ze względu na szybkość działania. Kluczowym parametrem jest wybór jądra, np. RBF lub KNN, ze względu na jego bezpośredni wpływ na strukturę macierzy wag W. Krótki opis najważniejszych etapów algorytmu Label Propagation przedstawiono poniżej.

1. Inicjalizacja macierzy przejścia A, na podstawie wag krawędzi W.
2. Przypisanie etykiet $\hat{\mathbf{y}}$ dla danych. W przypadku danych nieetykietowanych, inicjalizacja $\hat{\mathbf{y}}$ jako wartości losowe.
3. Powtarzanie aż do osiągnięcia zbieżności.
   (a) Propaguj etykiety: $\hat{\mathbf{y}} = A^T \cdot \hat{\mathbf{y}}$.
   (b) Zastąp wartości $\hat{\mathbf{y}}$ dla danych etykietowanych ich prawdziwymi etykietami.

1.3 Wybrane metody przygotowania danych

Porównanie skuteczności metod uczenia częściowo nadzorowanego (SSL) wymaga wyboru i odpowiedniego przygotowania danych. Omówimy teraz pokrótce wybrane, najważniejsze zagadnienia z tym związane. Dodatkowe, choć bardziej techniczne, informacje związane z przetwarzaniem wstępnym rozważanych zbiorów danych zostały również zamieszczone w podrozdziale 2.1.2.

1.3.1 Podział danych na zbiór uczący i testowy

Podczas przeprowadzania eksperymentów istniała potrzeba podziału zbioru danych nie tylko na część treningową i testową, ale również etykietowaną i nieetykietowaną.

Procedura podziału danych na dwa podzbiory przebiegała następująco. Dla zbioru

$$\mathcal{L}_n = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\},$$

gdzie $\mathbf{x_i}$ jest wektorem cech, a y_i etykietą klasy ($i = 1, 2, \ldots, n$), funkcja podziału $f : \mathcal{L}_n \to \{0, 1\}$ losowo przyporządkowuje każdy element $\mathcal{L}_n$ do jednej z dwóch grup. W ten sposób uzyskiwane są dwa rozłączne podzbiory, które można wykorzystać jako zbiory uczące i testowe lub odpowiednio – etykietowane i nieetykietowane. W drugim przypadku należy dodatkowo usunąć etykiety y_i chcąc uzyskać zbiór nieetykietowany. W ogólności otrzymujemy podzbiory $\mathcal{L}_1$ i $\mathcal{L}_2$ takie, że

$$\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset \quad \wedge \quad |\mathcal{L}_1| + |\mathcal{L}_2| = |\mathcal{L}_n|.$$

1.3.2 TF-IDF

Metoda TF-IDF (ang. *Term Frequency-Inverse Document Frequency*) to technika przetwarzania tekstów, stosowana przy przygotowywaniu danych tekstowych do analizy oraz ich konwersji na format wymagany przez algorytmy uczenia maszynowego.

Term Frequency (TF) wyraża częstość występowania danego termu i (np. słowa lub frazy) w pojedynczym dokumencie j w stosunku do częstości występowania najczęstszego słowa w tym dokumencie. Wartość TF dla termu i w dokumencie j jest definiowana jako

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}, \tag{1.24}$$

gdzie f_{ij} oznacza liczbę wystąpień termu i w dokumencie j, a $\max_k f_{kj}$ to maksymalna liczba wystąpień najczęściej występującego termu w tym samym dokumencie.

Inverse Document Frequency (IDF) z kolei pomaga zredukować wagę często występujących słów w całej kolekcji dokumentów, podkreślając znaczenie rzadszych termów. Wartość IDF dla termu i jest obliczana według wzoru[2]

$$IDF_i = \log_2 \left(\frac{N}{n_i} \right), \tag{1.25}$$

gdzie N reprezentuje całkowitą liczbę dokumentów w zbiorze, a n_i wskazuje na liczbę dokumentów zawierających term i.

Ostateczny wynik TF-IDF, będący iloczynem wartości TF i IDF, określa wagę termu i w dokumencie j i jest wyznaczony jako

$$TFIDF_{ij} = TF_{ij} \cdot IDF_i. \tag{1.26}$$

Wysoki wynik TF-IDF wskazuje na słowa (frazy), które są charakterystyczne dla danego dokumentu, lecz nie są powszechne w całym zbiorze, co czyni je potencjalnie wartościowymi cechami w późniejszym etapie modelowania predykcyjnego. [9]

1.3.3 Standaryzacja danych

Głównym celem standaryzacji danych jest redukcja wpływu różnych skal pomiarowych dla poszczególnych zmiennych, co pozwala na efektywniejsze uczenie się i identyfikowanie wzorców przez algorytmy. Jest to szczególnie ważne w przypadku metod klasyfikacji, które opierają się na odległościach między punktami danych, takich jak sieci neuronowe, KNN czy SVM. W pracy zastosowano klasyczną metodę standaryzacji danych, znaną jako standaryzacja z-score. Polega ona na przekształceniu każdej zmiennej w taki sposób, aby miała ona średnią równą zero i odchylenie standardowe równe jeden. Osiąga się to poprzez odjęcie od każdej wartości zmiennej jej średniej próbkowej i podzielenie wyniku przez próbkowe odchylenie standardowe. Proces ten można zapisać za pomocą wzoru

$$\tilde{X_j} = \frac{X_j - m_j}{s_j}, \tag{1.27}$$

gdzie X_j oznacza oryginalne wartości j-tej zmiennej, m_j to średnia próbkowa dla tej zmiennej, a s_j to próbkowe odchylenie standardowe.

[2]W zależności od implementacji stosowane są różne podstawy logarytmu. Przykładowo w bibliotece Scikit-learn stosuje się logarytm naturalny.

Rozdział 2

Część praktyczna

W niniejszym rozdziale zostanie przeprowadzona analiza skuteczności algorytmów uczenia częściowo nadzorowanego. Na początek opiszemy wykorzystane zbiory danych i przedstawimy ich najważniejsze własności, które mogą wpływać na skuteczność rozważanych metod. Kolejno, zostaną sformułowane pytania badawcze i zaproponowane adekwatne eksperymenty oraz przedstawiona będzie ogólna procedura wykorzystywana do oceny efektywności algorytmów. W dalszej części zaprezentujemy i szczegółowo zinterpretujemy wyniki przeprowadzonych eksperymentów.

2.1 Dane wykorzystane w analizie

2.1.1 Wstęp

W tej części przedstawiono szczegółowe informacje na temat trzech różnych zbiorów danych wykorzystywanych w przeprowadzonej analizie porównawczej metod uczenia częściowo nadzorowanego. Te zbiory to: CIFAR-10 (z ograniczeniem do klas psów i kotów), IMDb Reviews oraz Breast Cancer Wisconsin. Zbiory zostały wybrane z myślą o zbadaniu i porównaniu skuteczności algorytmów, z uwzględnieniem zróżnicowanej natury i złożoności problemu. Każdy z powyższych zbiorów danych wymagał zastosowania specyficznych metod przetwarzania wstępnego (ang. *preprocessig*), co zostało opisane poniżej.

2.1.2 Opis zbiorów danych

CIFAR-10 (zdjęcia psów i kotów) [5]

Zbiór CIFAR-10 to popularny zbiór danych w dziedzinie wizji komputerowej, składający się z 60 000 kolorowych obrazów o wymiarach 32×32 piksele, rozdzielonych równomiernie na 10 klas. Co ważne, zbiór jest domyślnie podzielony na zbiór treningowy (o wielkości 50 000) i testowy (10 000).

W kontekście tej pracy, w celu ograniczenia się do klasyfikacji binarnej, skupiono się wyłącznie na dwóch klasach: psach (klasa 5) i kotach (klasa 3). Te dwie klasy zostały wybrane ze względu na ich wzajemne podobieństwa oraz wyzwania jakie stawiają w procesie klasyfikacji. Klasa psów została uznana za klasę pozytywną, a kotów — negatywną. Na rysunku 2.1 przedstawiono przykładowe zdjęcia znajdujące się w zbiorze.

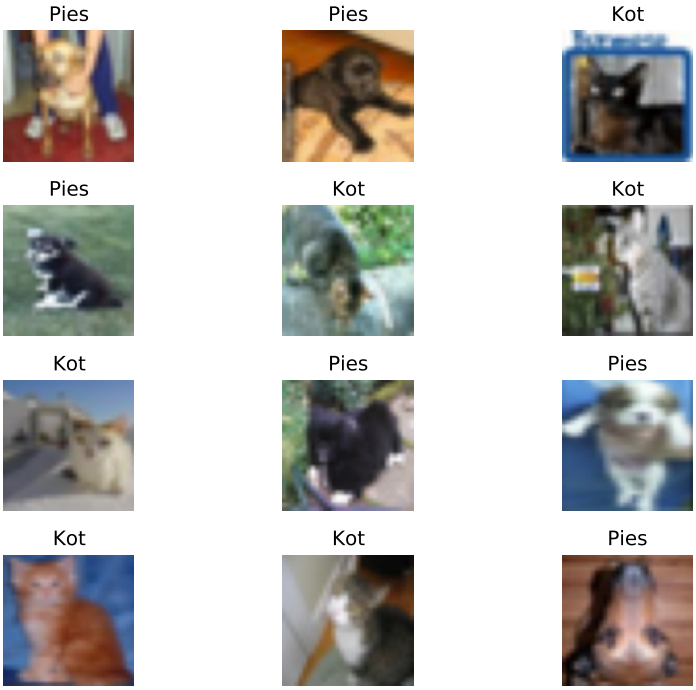
Obrazy, które pierwotnie były kolorowe, przekształcono na skalę szarości w celu zmniejszenia wymiarowości danych. Zredukowano w ten sposób liczbę kanałów każdego obrazu

z trzech (RGB) do jednego. Dzięki temu możliwe było zredukowanie wymagań dotyczących zasobów obliczeniowych.

Następnie wykonano standaryzację. W tym celu, podzielono wartość każdego piksela przez 255, co przekształciło wartości z zakresu 0–255 na zakres 0–1, po czym odjęto średnią i podzielono przez odchylenie standardowe obliczone na zbiorze treningowym.

Poniżej przedstawiono najważniejsze charakterystyki dotyczące ostatecznej wielkości analizowanego zbioru.

- Liczba obserwacji w zbiorze treningowym – 10 000, po równo dla obu klas.
- Liczba obserwacji w zbiorze testowym – 2000, po równo dla obu klas.
- Liczba cech dla każdego przypadku – 1024.



Rysunek 2.1: Przykładowe elementy zbioru CIFAR-10

Źródło: opracowanie własne

IMDb [7]

Zbiór danych IMDb Reviews składa się z 50 000 recenzji filmowych, z których każda jest oznaczona jako pozytywna lub negatywna. Teksty recenzji są zróżnicowane pod względem długości i złożoności, co stanowi wyzwanie w kontekście przetwarzania języka naturalnego. Zbiór jest domyślnie podzielony na treningowy i testowy, oba liczące 25 000 przypadków. Recenzje są domyślnie zmodyfikowane, poprzez usunięcie interpunkcji oraz ujednolicenie wielkości liter. Przykładowe fragmenty zamieszczono w tabeli 2.1.

W celu przekształcenia tekstów na format użyteczny w uczeniu maszynowym, zastosowano technikę TF-IDF (ang. *Term Frequency-Inverse Document Frequency*), opisaną w podrozdziale 1.3.2. Dokonano w ten sposób konwersji dokumentów tekstowych na macierz

cech TF-IDF, gdzie każdy wiersz reprezentuje dokument, a każda kolumna określoną cechę (słowo). Metoda ta pozwala uwzględnić ważność słów w kontekście całego zbioru danych. Dodatkowo, ograniczono się do 1000 najważniejszych cech, co pozwoliło na zredukowanie wymiarowości danych oraz skupienie się na najbardziej istotnych elementach tekstów.

Poniżej przedstawiono najważniejsze charakterystyki dotyczące ostatecznej wielkości rozważanego zbioru.

- Liczba obserwacji w zbiorze treningowym – 25 000, po równo dla obu klas.
- Liczba obserwacji w zbiorze testowym – 25 000, po równo dla obu klas.
- Liczba cech dla każdego przypadku – 1000.

Tabela 2.1: Przykładowe fragmenty dwóch recenzji z zestawu danych IMDb

Recenzja pozytywna	**Recenzja negatywna**
this film was just brilliant casting location scenery story direction everyone's really suited the part they played and you could just imagine being there robert redford's is an amazing actor and now the same being director norman's father came from the same scottish island as myself so i loved the fact there was a real connection with this film the witty remarks throughout the film were great it was just brilliant so much that i bought the film as soon as it was released for retail and would recommend it to everyone to watch and the fly fishing was amazing really cried at the end it was so sad and you know what they say if you cry at a film it must have been good and this definitely was	this has to be one of the worst films of the 1990s when my friends i were watching this film being the target audience it was aimed at we just sat watched the first half an hour with our jaws touching the floor at how bad it really was the rest of the time everyone else in the theatre just started talking to each other leaving or generally crying into their popcorn that they actually paid money they had earnt working to watch this feeble excuse for a film it must have looked like a great idea on paper but on film it looks like no one in the film has a clue what is going on crap acting crap costumes i can't get across how embarrasing this is to watch save yourself an hour a bit of your life

Źródło: opracowanie własne

Breast Cancer [13]

Zbiór danych Breast Cancer, dostępny na stronie UCI Machine Learning Repository[1], zawiera cechy diagnostyczne, które zostały wyznaczone na podstawie obrazów biopsji aspiracyjnej cienkoigłowej (FNA) raka piersi. Zbiór ten zawiera 569 próbek z 30 cechami dla każdej próbki. Każda z nich oznaczona jest jako 0 (nowotwór złośliwy) lub 1 (nowotwór łagodny). Jest to domyślne kodowanie dostępne w bibliotece scikit-learn, która była wykorzystana do zaimportowania danych.

[1] `https://archive.ics.uci.edu/dataset/14/breast+cancer`

Dane podzielono na zbiór treningowy i testowy w proporcji 80/20. Wykonano również standaryzację poprzez odjęcie średniej i podzielenie przez odchylenie standardowe na podstawie statystyk wyznaczonych dla zbioru treningowego. Przypomnijmy, że przeprowadzenie standaryzacji jest kluczowe zwłaszcza w przypadku algorytmów, które są wrażliwe na skalę danych, takich jak np. SVM.

Poniżej przedstawiono najważniejsze charakterystyki dotyczące ostatecznej wielkości analizowanego zbioru.

- Liczba obserwacji w zbiorze treningowym – 455, z czego 286 przypadków z klasy pozytywnej i 169 z klasy negatywnej.
- Liczba obserwacji w zbiorze testowym – 114, z czego 71 przypadków z klasy pozytywnej i 43 z klasy negatywnej.
- Liczba cech dla każdego przypadku – 30.

2.2 Pytania badawcze i proponowane eksperymenty

Pytania badawcze

W celu zbadania i porównania skuteczności metod uczenia częściowo nadzorowanego (SSL) dla różnych scenariuszy, postawiono następujące pytania badawcze.

1. Co możemy powiedzieć na temat skuteczności algorytmów SSL dla domyślnych hiperparametrów, w kontekście różnych zbiorów danych?
2. W jaki sposób zmiana hiperparametrów wpływa na skuteczność modeli SSL i czy można wskazać optymalne konfiguracje dla poszczególnych zbiorów danych?
3. Czy i w jaki stopniu frakcja danych etykietowanych, tzn. stosunek liczby obserwacji etykietowanych do nieetykietowanych wpływa na skuteczność oraz stabilność algorytmów SSL?

Proponowane eksperymenty

Na podstawie postawionych pytań badawczych, zaproponowano trzy eksperymenty, które pozwolą zrozumieć wpływ różnych czynników na wydajność modeli SSL.

Eksperyment 1: Porównanie skuteczności metod dla domyślnych parametrów

W pierwszym eksperymencie skupiono się na ocenie wydajności algorytmów SSL dla domyślnych hiperparametrów. Celem jest zrozumienie, jak dobrze modele radzą sobie bez dodatkowych modyfikacji i dostosowań. W badaniach uwzględnione zostały algorytmy SSL, przedstawione w podrozdziale 1.2.4. Każdy z algorytmów przetestowano na trzech różnych zbiorach danych: CIFAR-10, IMDb oraz Breast Cancer, co pozwoliło na ocenę ich skuteczności i przydatności w przypadku danych o różnej złożoności. Zastosowano ustalony, jednokrotny podział danych dla wszystkich modeli, zarówno przy podziale na zbiór treningowy/testowy, jak i etykietowany/nieetykietowany.

Aby uzyskać pełniejszy obraz efektywności metod SSL, w analizie uwzględnione zostały również wybrane (standardowe) algorytmy uczenia nadzorowanego: Random Forest,

Decision Tree, Naive Bayes, K-Neighbors Classifier i SVM z jądrem liniowym (ang. *linear*) oraz radialnym (ang. *radial basis function*, RBF). Te metody były uczone wyłącznie na części danych z etykietami. Wyniki uzyskane dla metod nadzorowanych służą jako punkt odniesienia i nie są bezpośrednio porównywane z wynikami konkretnych algorytmów SSL. Celem tej części badania jest sprawdzenie, czy wykorzystanie danych nieetykietowanych przynosi praktyczne korzyści i czy przyczynia się do poprawy wyników w porównaniu z tradycyjnymi metodami nadzorowanymi, które nie są w stanie ich wykorzystać.

Eksperyment 2: Porównanie skuteczności metod dla różnych wartości hiperparametrów

Drugi eksperyment ma na celu sprawdzenie wpływu hiperparametrów na wydajność modeli SSL. Dla każdego algorytmu rozważono siatkę z możliwymi wartościami wybranych parametrów, aby znaleźć ich najbardziej optymalną konfigurację dla poszczególnych zbiorów danych. Podobnie jak w Eksperymencie 1, zastosowano ustalony, jednokrotny podział danych dla wszystkich modeli, zarówno przy podziale na zbiór treningowy/testowy, jak i etykietowany/nieetykietowany. Analiza ta pomoże w zbadaniu wrażliwości rozważanych algorytmów na poszczególne hiperparametry.

Eksperyment 3: Porównanie skuteczności i stabilności metod w zależności od frakcji danych etykietowanych

Trzeci eksperyment ma na celu zbadanie, czy i w jakim stopniu ilość dostępnych danych etykietowanych wpływa na efektywność oraz stabilność algorytmów SSL. Rozważając różne frakcje etykietowanych próbek w zbiorach danych, zbadano, jak algorytmy zachowują się w przypadku zmieniających się warunków i które z nich są najbardziej odporne na redukcję informacji dotyczącej przynależności obserwacji do poszczególnych klas. To badanie jest również szczególnie ważne dla oceny przydatności modeli SSL w zastosowaniach praktycznych, gdzie uzyskanie dodatkowych danych etykietowanych nie zawsze jest trywialną sprawą. Jest to jedyny eksperyment, w którym wielokrotnie stosowano podział na zbiory etykietowany/nieetykietowany i powtarzano symulacje. Podział na zbiory treningowy i testowy, podobnie jak we wcześniejszych eksperymentach, był ustalony.

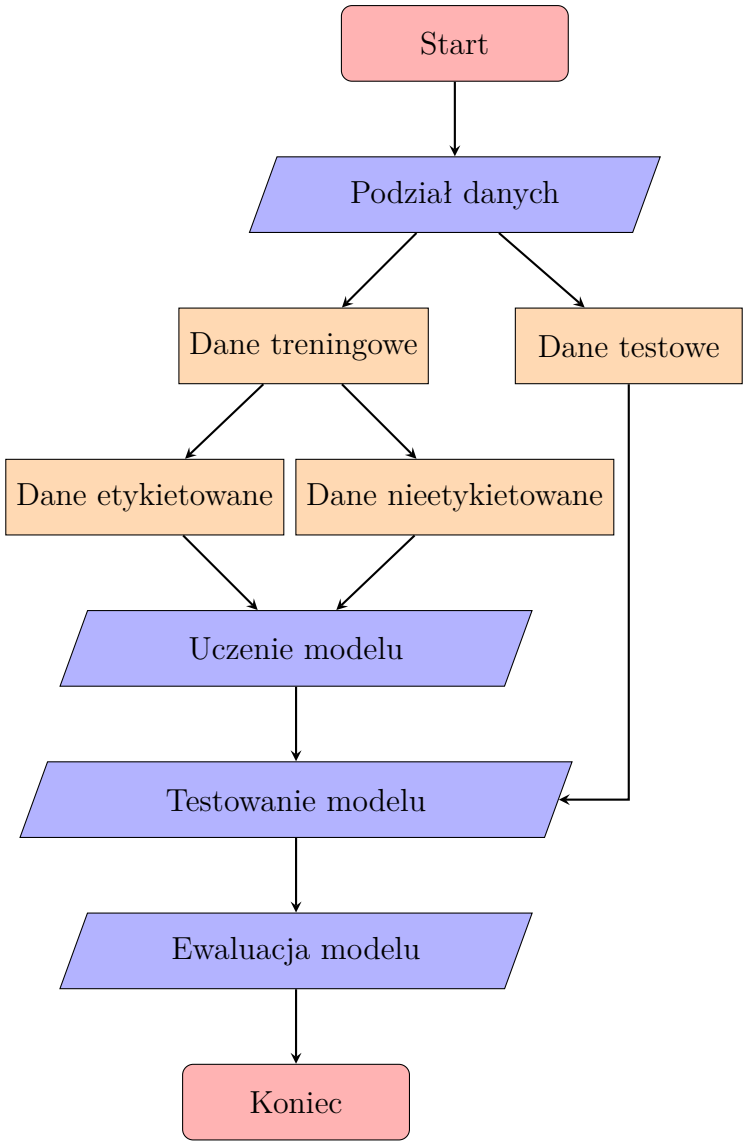
Dodatkowo, podobnie jak w pierwszym eksperymencie, wyniki dla algorytmów SSL zestawiono z rezultatami otrzymanymi dla wybranych (standardowych) algorytmów uczenia nadzorowanego. Pozwoli to uzyskać pełniejszy obraz i dokładniej przeanalizować użyteczność metod uczenia częściowo nadzorowanego.

2.3 Procedura wykorzystana do oceny skuteczności metod

Na potrzeby przeprowadzonych badań zaprojektowano uniwersalny schemat oceny mający na celu obiektywne porównanie wydajności rozważanych algorytmów uczenia częściowo nadzorowanego. Główne etapy zaproponowanej procedury przedstawiono na rysunku 2.2. W pierwszym kroku zastosowany jest jednokrotny podział zbioru danych na część treningową i testową. Następnie, z części treningowej usuwane są wybrane etykiety, aby utworzyć podzbiory etykietowane i nieetykietowane. Dzięki konsekwentnemu stosowaniu tego samego podziału na każdym etapie, zarówno w fazie przydzielania etykiet, jak i podczas dzielenia danych na uczące i testowe, zagwarantowana jest jednolitość warunków testowych.

Takie postępowanie ma na celu zapewnienie porównywalności wyników między różnymi algorytmami oraz zestawami danych. Po wstępnym przygotowaniu zbiorów, model jest trenowany, a następnie oceniany na danych, które nie uczestniczyły w procesie uczenia. Finałowym etapem jest ocena wydajności modelu, gdzie na podstawie zbioru testowego wyznaczane są odpowiednie kryteria (miary dokładności).

Do oceny skuteczności algorytmów wykorzystano popularne kryteria (patrz podrozdział 1.1.3), takie jak dokładność (ang. *accuracy*), precyzja (ang. *precision*), czułość (ang. *recall*) oraz miara F1. Przypomnijmy, że dokładność daje informację o ogólnej zdolności modelu do poprawnego klasyfikowania próbek. Precyzja skupia się na proporcji prawidłowo zidentyfikowanych pozytywnych przypadków wśród wszystkich, które model zaklasyfikował jako pozytywne. Czułość odzwierciedla zdolność modelu do wykrywania wszystkich rzeczywistych pozytywnych przypadków, natomiast miara F1 jest harmoniczną średnią precyzji i czułości. Oprócz kryteriów ściśle związanych z oceną dokładności klasyfikacji, będziemy badali również czas uczenia poszczególnych modeli. Pozwoli nam to na porównanie rozważanych algorytmów pod względem ich złożoności obliczeniowej, co również może znacząco wpływać na ich użyteczność w kontekście zastosowania do konkretnych zagadnień praktycznych.



Rysunek 2.2: Schemat procedury wykorzystanej do oceny skuteczności metod uczenia częściowo nadzorowanego

Źródło: opracowanie własne

2.4 Eksperyment 1: Porównanie skuteczności metod dla domyślnych parametrów

2.4.1 Wprowadzenie

W pierwszym przeprowadzonym eksperymencie sprawdzono wydajność algorytmów na trzech różnych zestawach danych: CIFAR-10, IMDb oraz Breast Cancer. Dla zbiorów CIFAR-10 i IMDb przyjęto domyślny podział na zbiór treningowy i testowy. Zbiór Breast Cancer podzielono natomiast w proporcji 80/20. We wszystkich zestawach usunięto 70% etykiet z danych treningowych, co oznacza, że wielkość frakcji danych etykietowanych wynosi 0,3. Dodatkowo, przeprowadzono analizę z wykorzystaniem standardowych metod uczenia nadzorowanego. Te algorytmy były trenowane wyłącznie na bazie ustalonej, ogra-

niczonej frakcji danych etykietowanych. Pozwoliło to uzyskać referencyjne wyniki, które posłużą jako punkt odniesienia do oceny efektywności algorytmów uczenia częściowo nadzorowanego.

2.4.2 Metodologia

W przeprowadzonej analizie zastosowano następujące algorytmy: Tri-Training, Assemble, SemiBoost, LapSVM, TSVM oraz Label Propagation. Dla modelu Tri-Training wybrano następujące trzy klasyfikatory bazowe:

- `base_estimator`: Random Forest,
- `base_estimator_2`: Linear SVM,
- `base_estimator_3`: Naive Bayes.

Dla modelu Assemble i SemiBoost zastosowano Random Forest jako klasyfikator bazowy (parametr `base_estimator`). Wszystkie pozostałe hiperparametry pozostały niezmienione. W tabeli 2.2 przedstawiono konfigurację parametrów stosowaną dla poszczególnych algorytmów. Zaznaczmy, że w tabeli nie wypisano wszystkich możliwych parametrów, a jedynie te, które wydają się najbardziej istotne i będę modyfikowane w kolejnych eksperymentach. Modele uczenia nadzorowanego były budowane dla domyślnych parametrów. Każdy z algorytmów oceniono pod kątem dokładności, precyzji, czułości, miary F1 oraz czasu uczenia.

Tabela 2.2: Hiperparametry algorytmów uczenia częściowo nadzorowanego w Eksperymencie 1

Algorytm	**Hiperparametry**
Tri-Training	• `base_estimator`: Random Forest • `base_estimator_2`: Linear SVM • `base_estimator_3`: Naive Bayes
Assemble	• `base_estimator`: Random Forest • `T`: 100
SemiBoost	• `base_estimator`: Random Forest • `T`: 300 • `similarity_kernel`: RBF
LapSVM	• `distance_function`: RBF • `kernel_function`: RBF • `gamma_d`: None • `gamma_k`: None
TSVM	• `max_iter`: -1 • `kernel`: RBF • `gamma`: scale
Label Propagation	• `max_iter`: 10000 • `kernel`: RBF • `gamma`: 1

Źródło: opracowanie własne

2.4.3 Wyniki

Uzyskane wyniki, dla poszczególnych zbiorów danych, przedstawiono w tabelach 2.3, 2.5 i 2.7. Wielkości referencyjne, dla modeli uczenia nadzorowanego, zostały zamieszczone w tabelach 2.4, 2.6 i 2.8. W każdej kolumnie pogrubiono najlepszy uzyskany wynik. Przy wynikach metod nadzorowanych nie zapisano czasu uczenia modelu K-najbliższych sąsiadów, ponieważ jest to algorytm leniwy (ang. *lazy learning*), który nie tworzy modelu predykcyjnego w fazie uczenia. Zamiast tego „uczenie" odbywa się podczas klasyfikacji, poprzez wyszukiwanie najbliższych sąsiadów w zbiorze danych dla każdej nowej próbki.

CIFAR-10

Tabela 2.3: Wyniki dla metod uczenia częściowo nadzorowanego dla zbioru CIFAR-10

Algorytm	**Dokładność**	**Precyzja**	**Czułość**	**F1**	**Czas uczenia** [s]
Tri-Training	0,608	0,609	0,601	0,605	294,089
Assemble	**0,625**	**0,646**	0,554	0,596	2268,513
SemiBoost	0,510	0,505	0,981	**0,667**	529,442
LapSVM	0,500	0,500	**1,000**	**0,667**	1170,307
TSVM	0,608	0,606	0,616	0,611	27762,706
Label Propagation	0,551	0,563	0,455	0,503	**2,837**

Źródło: opracowanie własne

Tabela 2.4: Referencyjne wyniki dla zbioru CIFAR-10 dla metod uczenia nadzorowanego

Algorytm	**Dokładność**	**Precyzja**	**Czułość**	**F1**	**Czas uczenia** [s]
Random Forest	**0,650**	**0,658**	0,623	**0,640**	6,797
Decision Tree	0,566	0,567	0,566	0,566	3,038
Linear SVM	0,534	0,533	0,546	0,540	62,247
Naive Bayes	0,598	0,590	**0,644**	0,616	**0,029**
KNN	0,583	0,589	0,548	0,568	-
RBF SVM	0,636	0,638	0,628	0,633	16,438

Źródło: opracowanie własne

Analiza wyników dla zbioru CIFAR-10, zaprezentowanych w tabeli 2.3, pozwala zauważyć istotne różnice w skuteczności poszczególnych algorytmów uczenia częściowo nadzorowanego.

Algorytm Tri-Training wykazał podobne rezultaty dla wszystkich kryteriów, osiągając dokładność na poziomie 60,8% z niewielkim odchyleniem w wartościach precyzji i czułości, co sugeruje porównywalną zdolność do prawidłowej identyfikacji obu klas (pozytywnej i negatywnej). Uzyskał on również drugi najkrótszy czas uczenia, wynoszący nieco ponad 294 sekundy.

Assemble uzyskał największą dokładność 62,5% i najwyższą precyzję 64,6%, co może świadczyć o jego nieco większej zdolności do poprawnej predykcji przypadków pozytywnych.

Jednakże czas uczenia jest znacznie dłuższy, przekraczający 2268 sekund, co może stanowić ograniczenie w praktycznych zastosowaniach.

SemiBoost wyróżnia wyjątkowa wysoka czułość na poziomie 98,1% oraz niska dokładność (51%) i precyzja (50,5%). Takie wyniki wskazują na model, który przypisuje wszystkie próbki do jednej klasy, co w praktyce jest niepożądane. Czas uczenia wynoszący około 529 sekund jest umiarkowany w porównaniu do innych metod.

LapSVM osiągnął maksymalną czułość przy dokładności i precyzji równych 50%. Są to wyniki jeszcze bardziej skrajne niż w przypadku SemiBoost. Dodatkowo, czas uczenia tego modelu jest stosunkowo długi i wynosi około 1170 sekund.

TSVM uzyskał wyniki podobne do algorytmu Tri-Training, jednakże, czas uczenia jest znacząco dłuższy, sięgając aż 27762 sekund, co czyni go najmniej efektywnym pod tym względem.

Label Propagation, z najkrótszym czasem uczenia wynoszącym tylko 2,84 sekundy, oferuje niższą dokładność (55,1%) i umiarkowaną precyzję (56,3%), co może wskazywać na ograniczenia tego modelu w zakresie generalizacji.

Podsumowując, wybór algorytmu dla zbioru CIFAR-10 powinien uwzględniać zarówno kryteria oceny dokładności, jak i ograniczenia czasowe. W kontekście skuteczności, Assemble wydaje się być najlepszym wyborem, chociaż jego czas uczenia może wymagać optymalizacji. Tri-Training i TSVM oferują zrównoważone podejście, należy jednak zwrócić uwagę na problematyczny czas uczenia obserwowany w przypadku TSVM. SemiBoost, LapSVM i Label Propagation bez dodatkowej optymalizacji hiperparametrów nie wydają się być dobrym wyborem dla tego zagadnienia klasyfikacji. Pierwsze dwa modele wykazały tendencję do nadmiernego przewidywania klasy pozytywnej, a Label Propagation miał problemy w zakresie generalizacji.

Porównując otrzymane rezultaty z wynikami referencyjnymi (dla metod uczenia nadzorowanego), można zauważyć, że żaden z algorytmów SSL nie przewyższa modelu Random Forest Classifier pod względem dokładności, co może wskazywać na wyższą skuteczność uczenia nadzorowanego. W tym przypadku, wydaje się, że potencjalna przewaga wynikająca z wykorzystania danych nieetykietowanych nie przełożyła się na znaczącą poprawę wyników. Należy jednak pamiętać, że różnice w wynikach mogą także być spowodowane brakiem doboru odpowiednich hiperparametrów lub wynikać ze specyfiki danych CIFAR-10. Warto zatem rozważyć dalsze eksperymenty, które pozwoliłyby lepiej wykorzystać potencjał danych nieetykietowanych.

IMDb

Tabela 2.5: Wyniki dla metod uczenia częściowo nadzorowanego dla zbioru IMDb

Algorytm	**Dokładność**	**Precyzja**	**Czułość**	**F1**	**Czas uczenia** [s]
Tri-Training	0,847	**0,848**	0,846	0,847	**18,149**
Assemble	0,789	0,782	0,801	0,791	8219,344
SemiBoost	0,501	0,501	**1,000**	0,667	16885,350
LapSVM	0,500	0,500	**1,000**	0,667	20792,758
TSVM	**0,852**	0,845	0,862	**0,854**	91529,197
Label Propagation	0,730	0,680	0,869	0,763	18,395

Źródło: opracowanie własne

Tabela 2.6: Referencyjne wyniki dla zbioru IMDb dla metod uczenia nadzorowanego

Algorytm	**Dokładność**	**Precyzja**	**Czułość**	**F1**	**Czas uczenia [s]**
Random Forest	0,821	0,825	0,815	0,820	6,633
Decision Tree	0,674	0,674	0,674	0,674	3,384
Linear SVM	**0,844**	**0,839**	**0,852**	**0,845**	0,120
Naive Bayes	0,806	0,809	0,801	0,805	**0,098**
KNN	0,681	0,676	0,693	0,685	-
RBF SVM	0,749	0,730	0,790	0,759	188,138

Źródło: opracowanie własne

Podobnie jak w przypadku zbioru CIFAR-10, wyniki uzyskane dla zadania klasyfikacji sentymentu, zaprezentowane w tabeli 2.5, również wykazały zróżnicowaną skuteczność algorytmów.

Algorytm Tri-Training osiągnął wysoką dokładność na poziomie 84,7%, z precyzją 84,8% oraz czułością 84,6%. Dodatkowo krótki czas uczenia (około 18 sekund) czyni go bardzo efektywną metodą dla klasyfikacji danych IMDb związanych z przetwarzaniem języka naturalnego.

Algorytm Assemble osiągnął o około 4–7 p.p. niższe wyniki niż Tri-Training. Znacznie dłuższy czas uczenia, który przekracza 8219 sekund, może stanowić przeszkodę w praktycznym stosowaniu modelu.

SemiBoost wykazał się niską dokładnością i precyzją (około 50%), lecz maksymalną czułością. Długi czas uczenia, wynoszący ponad 16888 sekund, również wskazuje na niską praktyczną przydatność tego podejścia. Uzyskane wyniki są podobne do statystyk dla zbioru CIFAR-10, co może sugerować, że model nie radzi sobie z danymi o złożonej strukturze.

LapSVM ponownie jak w przypadku zbioru CIFAR-10, prognozuje wszystkie przypadki jako pozytywne. Nie wydaje się więc być dobrym wyborem dla danych z dużą liczbą cech.

TSVM wyróżnia się najwyższą dokładnością wśród badanych modeli, wynoszącą 85,2%, z precyzją 84,5% i czułością 86,2%. Wysoka wartość miary F1 (85,4%) świadczy o dobrej równowadze między zdolnością do identyfikacji prawdziwie pozytywnych jak i prawdziwie negatywnych recenzji. Czas uczenia jest jednak znaczny i wynosi ponad 91529 sekund.

Label Propagation osiąga dokładność 73% oraz czułość 86,9%, ale niższą precyzję 68%, co może oznaczać większą liczbę fałszywie pozytywnych predykcji. Czas uczenia jest porównywalny do Tri-Training i wynosi około 18 sekund.

Podsumowując, wyniki dla zbioru IMDb wykazały, że TSVM i Tri-Training są najbardziej obiecującymi modelami pod względem wydajności, choć czas uczenia TSVM może być ograniczeniem. SemiBoost i LapSVM ponownie uzyskały przeciętne rezultaty, co może świadczyć o braku zdolności do prawidłowej predykcji w przypadku złożonych zbiorów danych, w szczególności związanych z analizą dokumentów tekstowych oraz obrazów. Label Propagation oferuje kompromis między dokładnością a czasem uczenia i może być użyteczny jeśli zależy nam przede wszystkim na czułości oraz jak najkrótszym czasie uczenia.

Warto zauważyć, że niektóre modele SSL są w stanie konkurować z metodami nadzorowanymi pod względem skuteczności. W szczególności, algorytmy Tri-Training i Assemble uzyskały lepsze wynik niż Linear SVM (najlepsza spośród rozważanych metod nadzorowanych). Potwierdza to, że specyfika danych może mieć znaczący wpływ na efektywność poszczególnych modeli klasyfikacyjnych.

Breast Cancer

Tabela 2.7: Wyniki dla metod uczenia częściowo nadzorowanego dla zbioru Breast Cancer

Algorytm	Dokładność	Precyzja	Czułość	F1	Czas uczenia [s]
Tri-Training	**0,965**	0,959	0,986	**0,972**	0,545
Assemble	**0,965**	0,959	0,986	**0,972**	90,810
SemiBoost	0,825	0,780	**1,000**	0,877	19,361
LapSVM	0,947	0,945	0,972	0,958	0,432
TSVM	0,956	**0,971**	0,958	0,965	0,386
Label Propagation	0,921	0,919	0,958	0,938	**0,313**

Źródło: opracowanie własne

Tabela 2.8: Referencyjne wyniki dla zbioru Breast Cancer dla metod uczenia nadzorowanego

Algorytm	Dokładność	Precyzja	Czułość	F1	Czas uczenia [s]
Random Forest	0,965	0,959	**0,986**	0,972	0,136
Decision Tree	0,947	0,945	0,972	0,958	0,002
Linear SVM	**0,982**	**0,986**	**0,986**	**0,986**	**0,001**
Naive Bayes	0,956	0,946	**0,986**	0,966	**0,001**
KNN	0,956	0,958	0,972	0,965	-
RBF SVM	0,965	0,972	0,972	0,972	0,004

Źródło: opracowanie własne

Analizując wyniki dla zbioru Breast Cancer, prezentowane w tabeli 2.7, można zaobserwować ogólnie wysoką skuteczność algorytmów uczenia częściowo nadzorowanego w zagadnieniu związanym z diagnostyką medyczną.

Algorytm Tri-Training oraz Assemble uzyskały identyczne wyniki, z najwyższą dokładnością na poziomie 96,5% oraz bardzo wysoką precyzją i czułością. Różnica czasu uczenia między tymi modelami jest jednak znacząca. Model oparty na algorytmie Tri-Training jest budowany niemal natychmiast i jest to około 90 sekund szybciej niż dla metody Assemble, której czas uczenia jest najdłuższy spośród wszystkich badanych metod i wynosi 90,81 sekund. Bardzo podobne, lecz gorsze o około 1–2 p.p. wyniki osiągnął LapSVM, którego czas uczenia był krótszy niż w Tri-Trainingu. Podobnie jest w przypadku TSVM, który osiągnął najwyższą precyzję (97,1%) w badanej grupie.

SemiBoost odbiega pod względem dokładności, osiągając wartość 82,5%, oraz precyzji 78%, lecz ponownie wykazuje maksymalną czułość 100%. Jego czas uczenia wydaje się krótki, ponieważ wynosi około 19 sekund, jednak wraz z Assemble są to jedyne algorytmy, których czas uczenia przekroczył sekundę.

Label Propagation ponownie okazał się algorytmem, który nie jest w stanie tak dobrze dostosować się do danych jak Tri-Training, Assemble, czy TSVM, jednak po raz kolejny był najszybciej uczącą się metodą.

Podsumowując, rezultaty uzyskane dla zbioru Breast Cancer wykazały, że większość testowanych algorytmów może być skutecznie stosowanych w diagnozowaniu raka piersi. Tri-Training i Assemble oferują najlepsze równowagi między dokładnością, precyzją i czułością, ale TSVM wyróżnia się najlepszą precyzją i bardzo krótkim czasem uczenia. LapSVM również prezentuje wysoką dokładność z krótkim czasem uczenia, udowadniając, że radzi sobie lepiej w przypadku danych o prostszej strukturze. SemiBoost ponownie ma tendencję do klasyfikowania większości przykładów jako pozytywne. Label Propagation oferuje natomiast dobry kompromis między dokładnością, a czasem uczenia.

2.4.4 Podsumowanie

Przeprowadzona w Eksperymencie 1 analiza, pokazała, że niewątpliwie złożoność danych ma bardzo duży wpływ na skuteczność rozważanych metod uczenia częściowo nadzorowanego. Dla każdego z algorytmów można było zauważyć, że im mniej złożony zbiór, tym lepsze wyniki. W szczególności SemiBoost i LapSVM uzyskiwały wartościowe wyniki dopiero dla zbioru Breast Cancer. Z kolei Tri-Training i TSVM niezależnie od danych, zawsze zwracały jedne z lepszych wyników. Warto również zwrócić uwagę na Label Propagation, który można nazwać najszybszym algorytmem, jednocześnie zawsze zwracającym wyniki gorsze o kilka punktów procentowych od wspomnianej powyżej pary metod.

Bardzo ważnym wnioskiem z całego eksperymentu jest to, że w ogólności algorytmy uczenia częściowo nadzorowanego nie dorównywały najlepszym metodom uczenia nadzorowanego. Mimo teoretycznych zalet płynących z wykorzystania dodatkowych informacji z danych nieetykietowanych, w praktyce nie zaobserwowano oczekiwanej poprawy efektywności. Jak wspomniano, może to wynikać z braku optymalizacji hiperparametrów (w eksperymencie wykorzystano wartości domyślne hiperparametrów dla poszczególnych algorytmów). Istotnym czynnikiem może być również wielkość frakcji obserwacji etykietowanych w powiązaniu ze złożonością samego zagadnienia klasyfikacyjnego (w eksperymencie ustalono dla wszystkich danych frakcję równą 0,3). Wpływ wymienionych czynników na skuteczność algorytmów SSL będzie m.in. szczegółowo badany w ramach kolejnych eksperymentów.

2.5 Eksperyment 2: Porównanie skuteczności metod dla różnych wartości hiperparametrów

2.5.1 Wprowadzenie

W tym eksperymencie będziemy analizowali wrażliwość wybranych metod uczenia częściowo nadzorowanego na zmiany hiperparametrów. Głównym celem przeprowadzonych symulacji było zbadanie, czy i w jakim stopniu wybór poszczególnych hiperparametrów dla algorytmów Tri-Traing, Assemble, SemiBoost, LapSVM, TSVM i Label Propagation, wpływa na uzyskane wyniki.

Poprzez ustalenie siatki wartości hiperparametrów dla każdego z analizowanych algorytmów, możliwe będzie uchwycenie zależności między zmianami tych parametrów a efektywnością modeli. Zastosowanie takiego podejścia pozwoli m.in. na zaobserwowanie, jak zmiany w hiperparametrach wpływają na zachowanie i wydajność modeli w przypadku danych o zróżnicowanej strukturze. Podkreślmy, że analiza koncentruje się na badaniu zmian o charakterze jakościowym w efektywności metod, celem nie było znalezie-

nie najlepszych możliwych kombinacji parametrów. Wszystkie symulacje były wykonane na identycznych danych, jak w Eksperymencie 1.

2.5.2 Metodologia

Każdy algorytm przetestowano na trzech zbiorach danych CIFAR-10, IMDb i Breast Cancer. Aby możliwe było porównanie wyników z tymi uzyskanymi w Eksperymencie 1, w analizie wykorzystano również identyczne podziały na zbiór uczący i testowy oraz zbiór danych estykietowany i nieetykietowany (dla frakcji obserwacji etykietowanych równej 0,3). Poniżej przedstawiono zestawienie wybranych (najważniejszych) hiperparametrów i ich wartości, które były brane pod uwagę w przypadku poszczególnych algorytmów SSL, tzn. Tri-Traing, Assemble, SemiBoost, LapSVM, TSVM i Label Propagation:

- Tri-Training
  - `base_estimator`: Random Forest;
  - `base_estimator_2`: Decision Tree, Linear SVM, Naive Bayes, KNN, RBF SVM;
  - `base_estimator_3`: Decision Tree, Linear SVM, Naive Bayes, KNN, RBF SVM;
- Assemble
  - `base_estimator`: Random Forest, Decision Tree, Naive Bayes;
  - `T`: 3, 10, 50, 100, 150, 300;
- SemiBoost
  - `base_estimator`: Random Forest, Decision Tree, Linear SVM, Naive Bayes, KNN, RBF SVM;
  - `T`: 3, 10, 50, 100, 150, 300;
  - `similarity_kernel`: rbf, knn;
- LapSVM
  - `distance_function`: rbf, linear, knn;
  - `kernel_function`: rbf, linear;
  - `gamma_k`: 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{3};
  - `gamma_d`: 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{3};
- TSVM
  - `max_iter`: 3, 10, 50, 100, 150, 300;
  - `kernel`: rbf, linear;
- Label Propagation
  - `kernel`: rbf, linear;
  - `max_iter`: 3, 10, 50, 100, 150, 300;
  - `gamma`: 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5};
  - `n_neighbors`: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19.

2.5.3 Wyniki

Dla poszczególnych algorytmów, w tabelach 2.9, 2.11, 2.12, 2.13, 2.14 i 2.15 przedstawiono uśrednione wyniki wraz z odchyleniem standardowym, obliczone na podstawie wszystkich możliwych kombinacji hiperparametrów. Dla porównanie, w tabeli 2.10 przedstawiono rezultaty uzyskane dla homogenicznego zestawu klasyfikatorów tzn. na bazie algorytmu KNN dla różnej liczby sąsiadów ($K = 3, 5, 7$). Pozostałe rezultaty zamieszczone są na mapach ciepła (rys. 2.3, 2.7, 2.8) i odpowiednich wykresach liniowych (rys. 2.4, 2.5, 2.6, 2.10 i 2.11).

Tri-Training

Wyniki przedstawione w tabeli 2.9 dla algorytmu Tri-Training pokazują, że odpowiedni dobór hiperparametrów może mieć wpływ na skuteczność tej metody. Dodatkowo obserwujemy duże zróżnicowanie wyników w zależności od zbioru danych. W szczególności dla zbioru IMDB odpowiednia konfiguracja parametrów może znacznie poprawić dokładność modelu, ponieważ w tym przypadku uzyskano najwyższe odchylenie standardowe. Dla zbioru Breast Cancer uzyskano odchylenie standardowe na poziomie $0,8\%$ co wskazuje na to, że w tym przypadku algorytm jest mniej wrażliwy na wybór hiperparametrów.

Mapy cieplne na rysunku 2.3 przedstawiające wyniki dla algorytmu Tri-Training z różnymi zestawami hiperparametrów dostarczają przekonującego dowodu na to, że odpowiedni dobór klasyfikatorów bazowych jest kluczowy dla tego algorytmu i wskazują na wrażliwość modelu na zmiany w konfiguracji.

W przypadku kryterium dokładności, dla zbioru CIFAR-10 różnica w uzyskanych wynikach wyniosła blisko 9 p.p. Najmniej skuteczne okazało się zastosowanie dwóch klasyfikatorów Linear SVM (54,8%), a najlepsze Decision Tree i KNN — 63,7%. Równie duże różnice można zaobserwować porównując wartości precyzji i czułości. W przypadku czasu uczenia, niewątpliwie zastosowanie Linear SVM wiąże się ze znacznym jego wydłużeniem. Przypomnijmy, że w pierwszym eksperymencie dokładność wyniosła 60,8%, zatem dobór hiperparametrów może znacznie wpłynąć na uzyskane wyniki.

W przypadku danych IMDb również obserwujemy istotną wrażliwość algorytmu Tri-Training na zmianę hiperparametrów. Najlepsze wyniki uzyskujemy dzięki zastosowaniu dwóch algorytmów Linear SVM, a w przypadku chęci zachowania heterogenicznego zestawu klasyfikatorów — Linear SVM i Naive Bayes. Rozrzut uzyskanych wyników jest jeszcze większy niż w przypadku zbioru CIFAR-10 i wynosi blisko 18 p.p. w przypadku dokładności. Czas uczenia znacznie się zwiększa po zastosowaniu choćby jednego klasyfikatora opartego na drzewach decyzyjnych (Decision Tree). W porównaniu z wynikami uzyskanymi w pierwszym eksperymencie, nie uzyskano dużo wyższej poprawy dokładności, jedynie o niecały 1 p.p.

Wyniki dla zbioru Breast Cancer charakteryzuje największa stabilność i mała wrażliwość na zmianę hiperparametrów. W przypadku kryterium dokładności rozrzut możliwych wyników jest mniejszy niż 4 p.p. Największe różnice można zaobserwować na diagonali macierzy, gdzie jedynie model z dwoma klasyfikatorami Linear SVM jest lepszy niż modele heterogeniczne. Podobnie jak dla zbioru IMDb, wyniki poprawiły się o maksymalnie 0,9 p.p. w przypadku dokładności i o 1,4 p.p. dla czułości, w porównaniu z pierwszym eksperymentem.

Podsumowując, algorytm Tri-Training jest bardzo wrażliwy na zmianę parametrów. Wybór bazowych klasyfikatorów, które są jego jedynymi hiperparametrami, może przynieść dużą zmianę w otrzymanych wynikach. Należy zwrócić uwagę, że prawdopodobnie algorytm

mógłby uzyskać jeszcze lepsze wyniki, gdyby dostrojono hiperparametry stosowanych klasyfikatorów bazowych. Dodatkowo, homogeniczny zestaw klasyfikatorów w większości przypadków spisał się gorzej niż zestawy heterogeniczne, co potwierdza, że istotne znaczenie dla skuteczności algorytmu Tri-Training może mieć różnorodność modeli bazowych.

Tabela 2.9: Uśrednione wyniki z odchyleniem standardowym dla algorytmu Tri-Training

	CIFAR-10	**IMDb**	**Breast Cancer**
Dokładność	0,614 (0,021)	0,808 (0,044)	0,957 (0,008)
Precyzja	0,620 (0,027)	0,803 (0,048)	0,952 (0,008)
Czułość	0,594 (0,038)	0,817 (0,035)	0,980 (0,010)
F1	0,606 (0,020)	0,810 (0,040)	0,966 (0,006)
Czas uczenia [s]	155,698 (148,608)	37,054 (19,907)	0,527 (0,143)

Źródło: opracowanie własne

Tabela 2.10: Wyniki dla algorytmu Tri-Training z klasyfikatorami bazowymi KNN dla liczby sąsiadów $K = 3, 5, 7$

	CIFAR-10	**IMDb**	**Breast Cancer**
Dokładność	0,588	0,685	0,965
Precyzja	0,594	0,68	0,972
Czułość	0,553	0,699	0,972
F1	0,573	0,689	0,972
Czas uczenia [s]	2,572	9,339	0,046

Źródło: opracowanie własne

Tabela 2.11: Uśrednione wyniki z odchyleniem standardowym dla algorytmu Assemble

	CIFAR-10	**IMDb**	**Breast Cancer**
Dokładność	0,594 (0,031)	0,760 (0,055)	0,957 (0,008)
Precyzja	0,598 (0,040)	0,757 (0,056)	0,954 (0,006)
Czułość	0,588 (0,037)	0,766 (0,056)	0,978 (0,012)
F1	0,592 (0,022)	0,761 (0,055)	0,966 (0,007)
Czas uczenia [s]	2282,223 (4131,565)	7275,171 (15757,419)	63,017 (177,202)

Źródło: opracowanie własne

Rysunek 2.3: Wyniki dla algorytmu Tri-Training, w zależności od dobranych hiperparametrów, przedstawione na mapach cieplnych

Źródło: opracowanie własne

Assemble

Algorytm Assemble przetestowano, zmieniając jego bazowy klasyfikator (`base_estimator`) oraz liczbę wykonywanych iteracji (`T`). Jako bazowych klasyfikatorów nie wybrano algorytmów Linear SVM i KNN, ponieważ nie mają one możliwości przypisania wag do obserwacji na etapie uczenia. Uśrednione wyniki przedstawione w tabeli 2.11 są niższe i towarzyszą im większe odchylenia standardowe niż dla metody Tri-Training. Potwierdza to, że Assemble jest kolejnym algorytmem, który wykazuje dużą wrażliwość na wybrane hiperparametry, co podkreśla konieczność ich starannego doboru dla zapewnienia optymalnego działania. Wykresy 2.4 pozwalają sformułować bardziej precyzyjne wnioski.

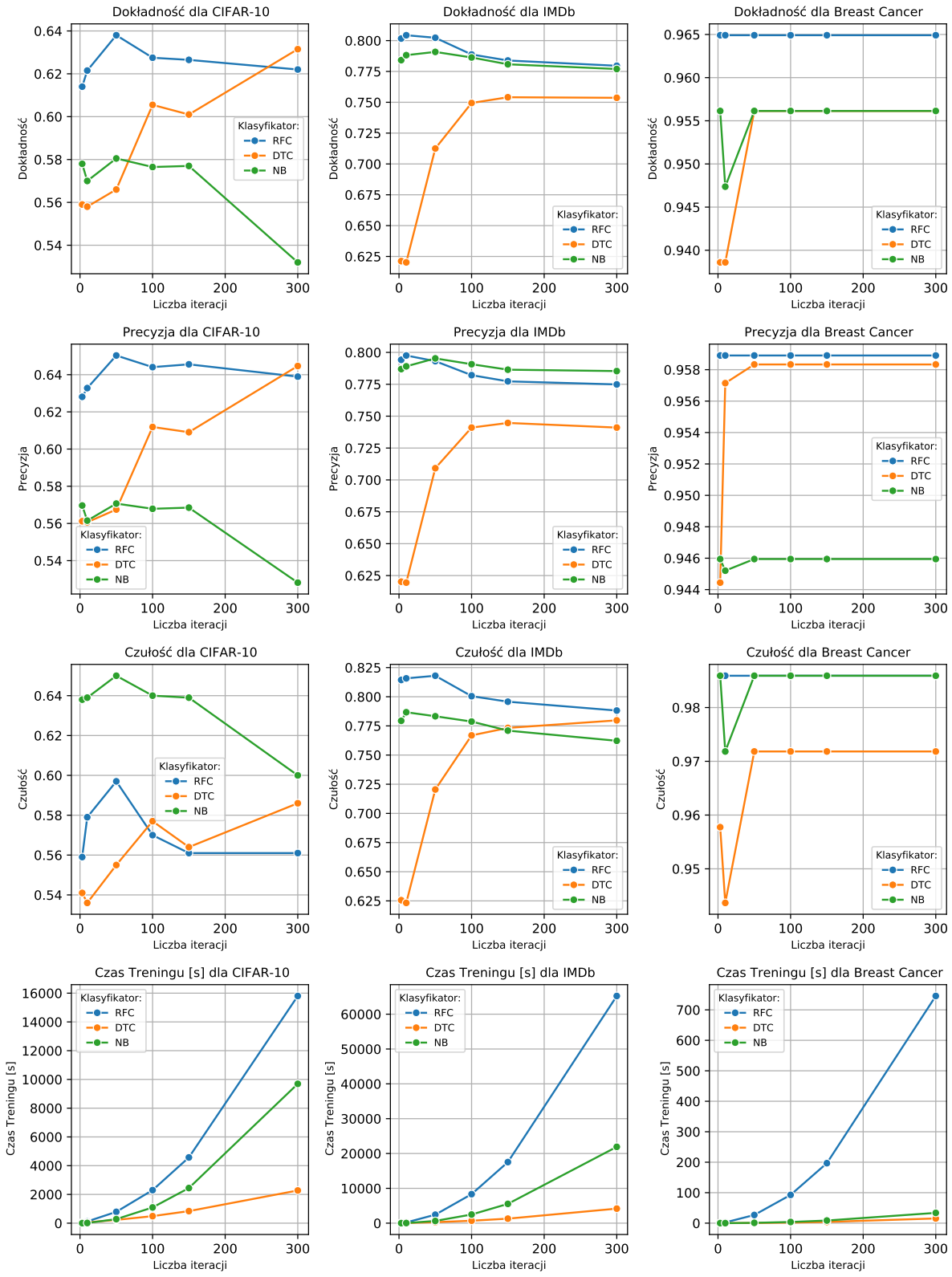
Dla zbioru CIFAR-10, najwyższą dokładność i precyzję osiąga klasyfikator Random Forrest, jednak wyniki dla tej metody oraz algorytmu Naive Bayes pogarszają się wraz ze wzrostem liczby iteracji. Taka tendencja może sugerować, że dla złożonych danych, takich jak CIFAR-10, zbyt duża liczba iteracji może prowadzić do przetrenowania modelu. Klasyfikator Decision Tree osiąga niższą dokładność niż Random Forest i jest jedynym, dla którego wraz z większą liczbą iteracji obserwujemy poprawę wszystkich kryteriów wykorzystywanych do oceny dokładności klasyfikacji. Algorytm Naive Bayes osiąga znacznie niższe wyniki dokładności, jednak jest najlepszy w wykrywaniu zdjęć psów (klasa *positive*). Warto zauważyć, że dla klasyfikatora Random Forest i liczby iteracji 50 otrzymujemy lepszy wynik niż w poprzednim eksperymencie (na bazie domyślnych wartości hiperparametrów).

W przypadku danych IMDb klasyfikatory Random Forest i Naive Bayes utrzymują relatywnie stabilne i wysokie wyniki. Decision Tree pokazuje znaczny wzrost dokładności przy początkowym zwiększeniu liczby iteracji, po czym stabilizuje się, co może sugerować, że optymalna liczba iteracji dla tego klasyfikatora jest w tym przypadku osiągana dość szybko. W przypadku tego zbioru nie udało się uzyskać znacząco lepszych wyników niż w pierwszym eksperymencie.

Na wykresie dla Breast Cancer, gdzie zadaniem jest klasyfikacja danych biomedycznych, Random Forest osiąga najwyższe wyniki, które są stabilne wraz ze wzrostem liczby iteracji. To sugeruje, że Random Forest dobrze radzi sobie z zadaniem i jest odporny na przeuczenie. Decision Tree i Naive Bayes również są stabilne po przekroczeniu małej liczby iteracji, jednak radzą sobie gorzej niż Random Forest. Decision Tree jest lepszy jeśli chodzi o precyzję, a Naive Bayes jeśli chodzi o czułość.

Można zaobserwować znaczący wzrost czasu uczenia wraz ze wzrostem liczby iteracji dla wszystkich klasyfikatorów, z najbardziej wyrazistym skokiem w przypadku Random Forest, w szczególności dla zbioru Breast Cancer. To sugeruje, że złożoność obliczeniowa tego algorytmu wzrasta najszybciej wraz z liczbą iteracji. Decision Tree i Naive Bayes również wykazują wzrost czasu treningu, ale jest on wolniejszy niż w przypadku Random Forest, co wskazuje na to, że te klasyfikatory są mniej złożone obliczeniowo.

Podsumowując, prezentowane wykresy wskazują na istotny wpływ wyboru klasyfikatora bazowego i liczby iteracji na wyniki algorytmu Assemble. W szczególności algorytm Random Forest wydaje się być najbardziej odpowiednim klasyfikatorem, który dobrze radzi sobie w przypadku różnych scenariuszy, będąc jednocześnie najbardziej kosztownym obliczeniowo. Podobnie jak w przypadku metody Tri-Training, możliwe, że uzyskane wyniki byłyby jeszcze lepsze, gdyby przeprowadzono optymalizację hiperparametrów stosowanych klasyfikatorów bazowych. Bez tej dodatkowej optymalizacji, jedynie w przypadku danych CIFAR-10 udało się uzyskać lepszy wynik niż w pierwszym eksperymencie.



Rysunek 2.4: Wyniki dla algorytmu Assemble, w zależności od dobranych hiperparametrów

Źródło: opracowanie własne

SemiBoost

W przypadku metody SemiBoost, podobnie jak dla Assemble, zmieniano bazowy klasyfikator i liczbę iteracji oraz dodatkowo jądro podobieństwa (`similarity_kernel`). Podobnie jak w pierwszym eksperymencie, SemiBoost średnio osiągał gorsze wyniki i charakteryzował się większymi odchyleniami standardowymi, niż poprzednie klasyfikatory, co potwierdzają wyniki w tabeli 2.12. Na podstawie wykresów 2.5 i 2.6 zauważono, że dobór parametrów dla tej metody jest kluczowy.

W przypadku jądra RBF, wybranie metody Naive Bayes jako bazowego klasyfikatora okazało się być kluczowe, aby dla danych CIFAR-10 i IMDb, algorytm SemiBoost przestał przewidywać tylko jedną klasę. Taki efekt przyniósł również klasyfikator bazowy Decision Tree, jednak z gorszym skutkiem. Dzięki zmianie parametrów udało się zwiększyć dokładność algorytmu o około 9 p.p dla zbioru CIFAR-10 i aż o około 30 p.p. dla IMDb. Na wykresach dla Breast Cancer, wszystkie klasyfikatory prezentują dużą zmienność pod względem dokładności i precyzji w zależności od liczby iteracji. Warto zauważyć, że w większości przypadków zwiększenie liczby iteracji nie poprawia skuteczności modelu, a jedynie wydłuża czas treningu.

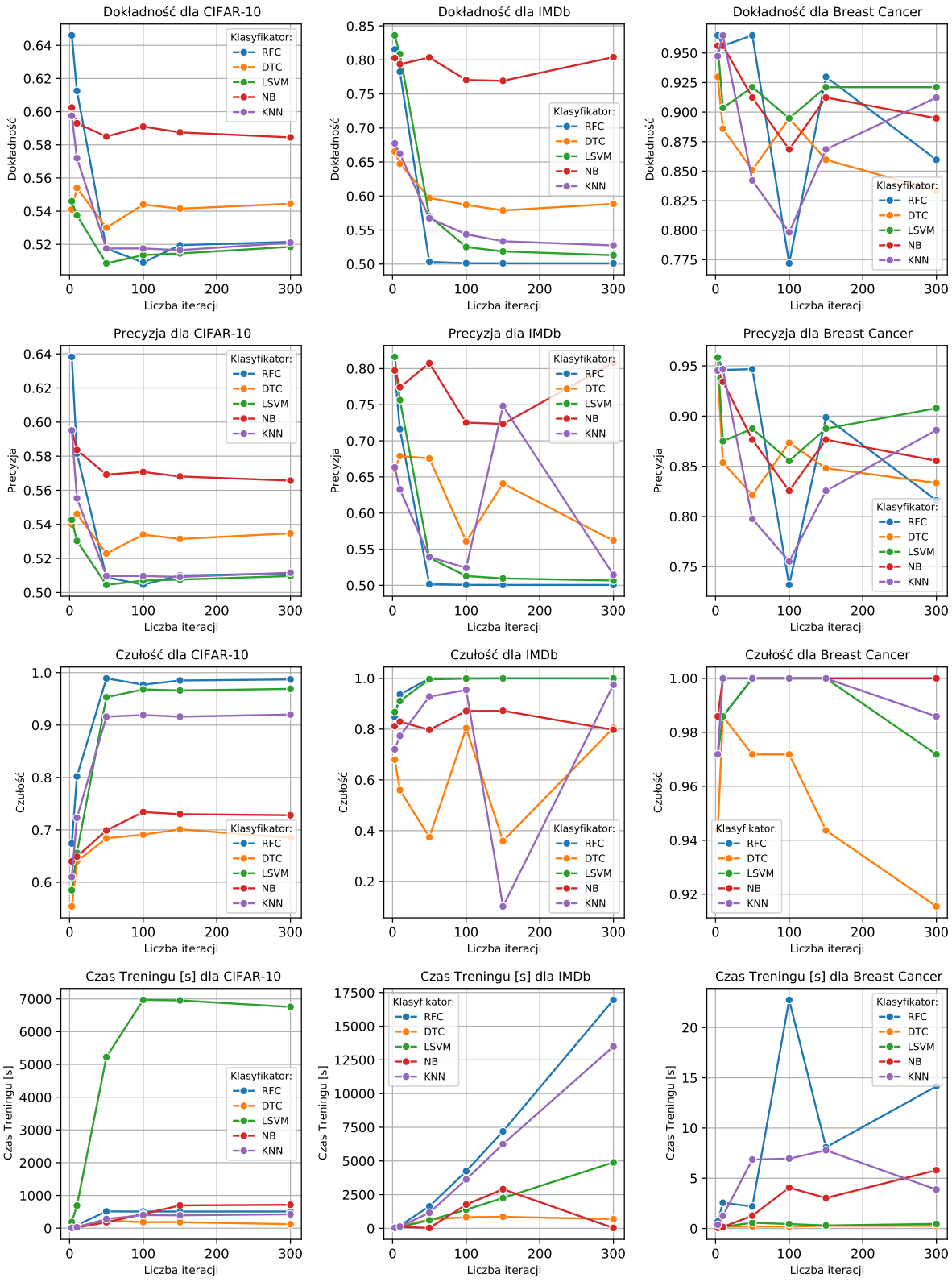
Zastosowanie jądra KNN pozytywnie wpływa na uzyskane wyniki i pozwala uzyskać dokładniejsze predykcje w przypadku większej liczby klasyfikatorów bazowych. Dla danych CIFAR-10, widoczne są znaczące różnice w dokładności pomiędzy różnymi klasyfikatorami bazowymi. Random Forest wykazuje najwyższą dokładność na początku, jednak zwiększanie liczby iteracji prowadzi do jej spadku, co może świadczyć o przeuczeniu. Na wykresie dla IMDb można zauważyć wyraźny podział na dwie grupy, Random Forest, Linear SVM i Naive Bayes są wyraźnie lepsze od KNN i Decision Tree. W przypadku danych Breast Cancer, podobnie jak dla jądra RBF, nie da się wskazać jednoznacznie najlepszego klasyfikatora, jedynie Decision Tree okazał się najsłabszy. Warto również zwrócić uwagę, że czas uczenia dla jądra KNN jest nieporównywalnie niższy niż dla RBF.

Podsumowując, eksperymenty przeprowadzone dla metody SemiBoost wyraźnie pokazały, że optymalizacja hiperparametrów, takich jak wybór bazowego klasyfikatora, liczba iteracji oraz typ jądra podobieństwa, odgrywa ważną rolę w skuteczności tej metody uczenia częściowo nadzorowanego. Zmiana parametrów może prowadzić do znacznej poprawy dokładności, co na przykład jest widoczne we wzroście wskaźników jakości dla danych CIFAR-10 i IMDb po zastosowaniu Naive Bayes z jądrem RBF. Algorytm SemiBoost jest wrażliwy na zmianę hiperparametrów, jednak ich odpowiednia optymalizacja może znacząco podnieść jego wydajność.

Tabela 2.12: Uśrednione wyniki z odchyleniem standardowym dla algorytmu SemiBoost

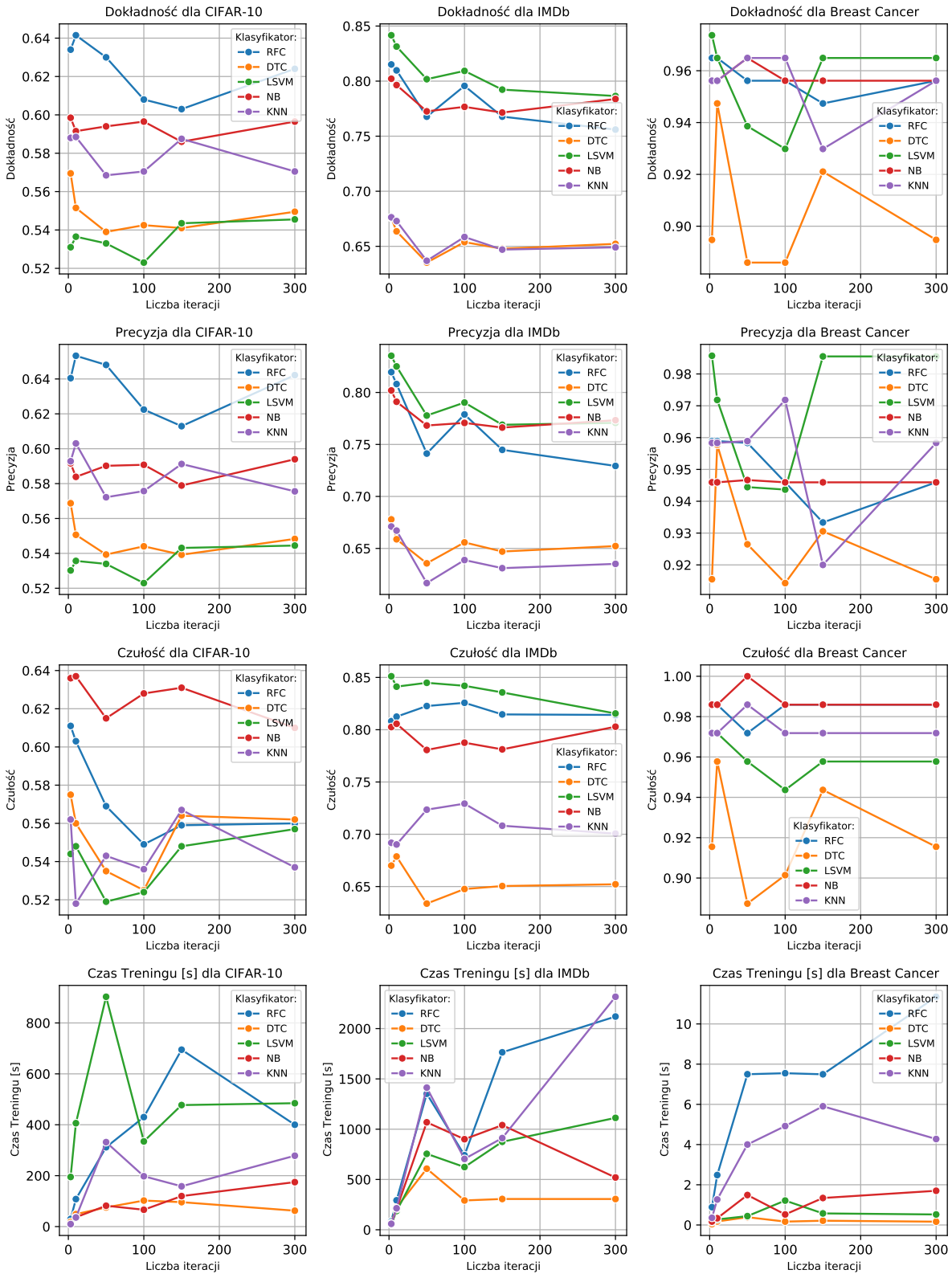
	CIFAR-10	**IMDb**	**Breast Cancer**
Dokładność	0,563 (0,038)	0,691 (0,110)	0,924 (0,045)
Precyzja	0,559 (0,041)	0,684 (0,105)	0,913 (0,057)
Czułość	0,678 (0,154)	0,791 (0,163)	0,976 (0,027)
F1	0,603 (0,045)	0,716 (0,109)	0,942 (0,032)
Czas uczenia [s]	666,644 (1592,475)	1558,983 (2971,203)	2,715 (4,113)

Źródło: opracowanie własne



Rysunek 2.5: Wyniki dla algorytmu SemiBoost, w zależności od dobranych hiperparametrów dla jądra RBF

Źródło: opracowanie własne



Rysunek 2.6: Wyniki dla algorytmu SemiBoost, w zależności od dobranych hiperparametrów dla jądra KNN

Źródło: opracowanie własne

LapSVM

W ramach eksperymentu badającego skuteczność algorytmu LapSVM, zmodyfikowano hiperparametry obejmujące zmianę funkcji odległości oraz funkcji jądrowych, a dodatkowo zmodyfikowano wartości parametru gamma dla funkcji RBF. Zmiany te nie miały jednak znaczącego wpływu na ogólną wydajność modelu.

W przypadku zbiorów CIFAR-10 i IMDb, analiza wyników wykazała niewielką zmienność w zakresie dokładności i precyzji, oscylującą w okolicach 50%, oraz maksymalną czułość, co implikuje, że klasyfikatory systematycznie przewidywały jedną klasę. Zmiany parametru gamma dla funkcji jądrowej RBF nie wpłynęły na poprawę tych miar wydajności, co potwierdzają wartości odchyleń standardowych w tabeli 2.13. Długotrwały czas uczenia stanowił również znaczące ograniczenie, utrudniające przeprowadzenie bardziej dogłębnych eksperymentów dotyczących hiperparametrów.

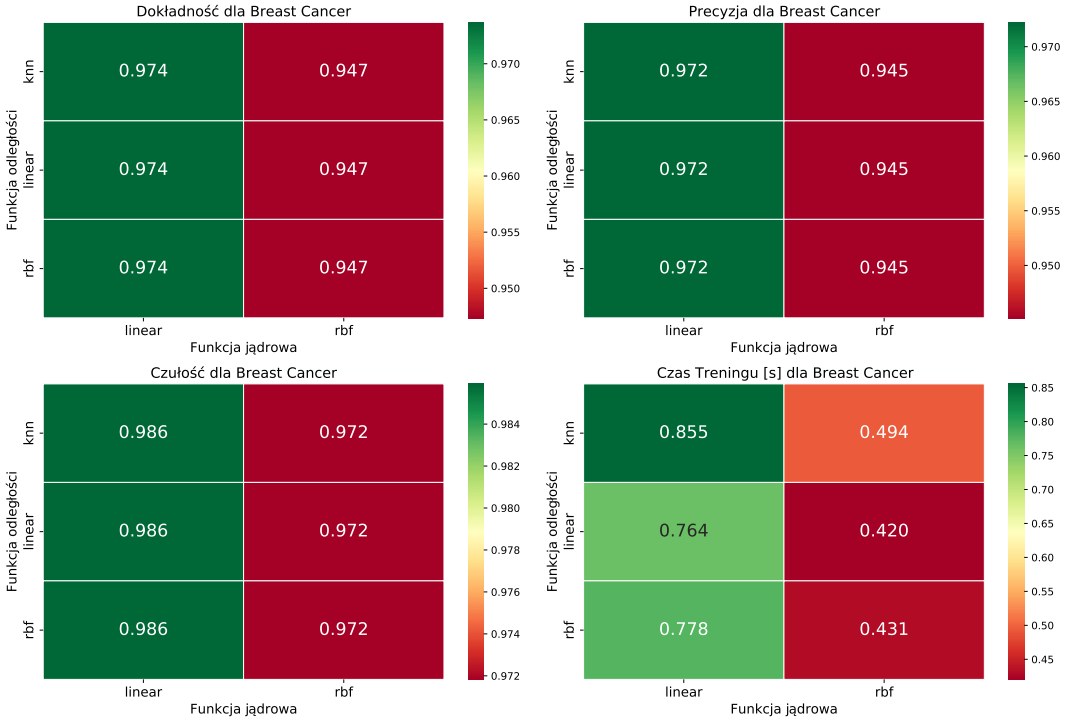
W kontekście analizy danych Breast Cancer, rezultaty przedstawiono w formie map cieplnych (rys. 2.7), co pozwoliło na wizualne zilustrowanie wpływu wybranych funkcji. Wybór rodzaju funkcji odległości nie ma wpływu na uzyskane wyniki. W przypadku funkcji jądrowej, jądro liniowe okazało się wydajniejsze w porównaniu z RBF. Następnie zbadano wpływ parametru gamma dla domyślnie przyjętych funkcji RBF, co zilustrowano na wykresach 2.8. Wartość gamma dla funkcji jądrowej odgrywa istotną rolę w wydajności modelu i przekłada się na znaczące różnice w wynikach.

Podsumowując, LapSVM jest algorytmem, który prawdopodobnie nie radzi sobie z wielowymiarowymi danymi, co potwierdzają bardzo niskie wyniki dla zbiorów CIFAR-10 i IMDb. W przypadku mniej skomplikowanych zbiorów, takich jak Breast Cancer, na wyniki może wpływać rodzaj funkcji jądrowej oraz parametr gamma, jeśli stosujemy radialną funkcję jądrową.

Tabela 2.13: Uśrednione wyniki z odchyleniem standardowym dla algorytmu LapSVM

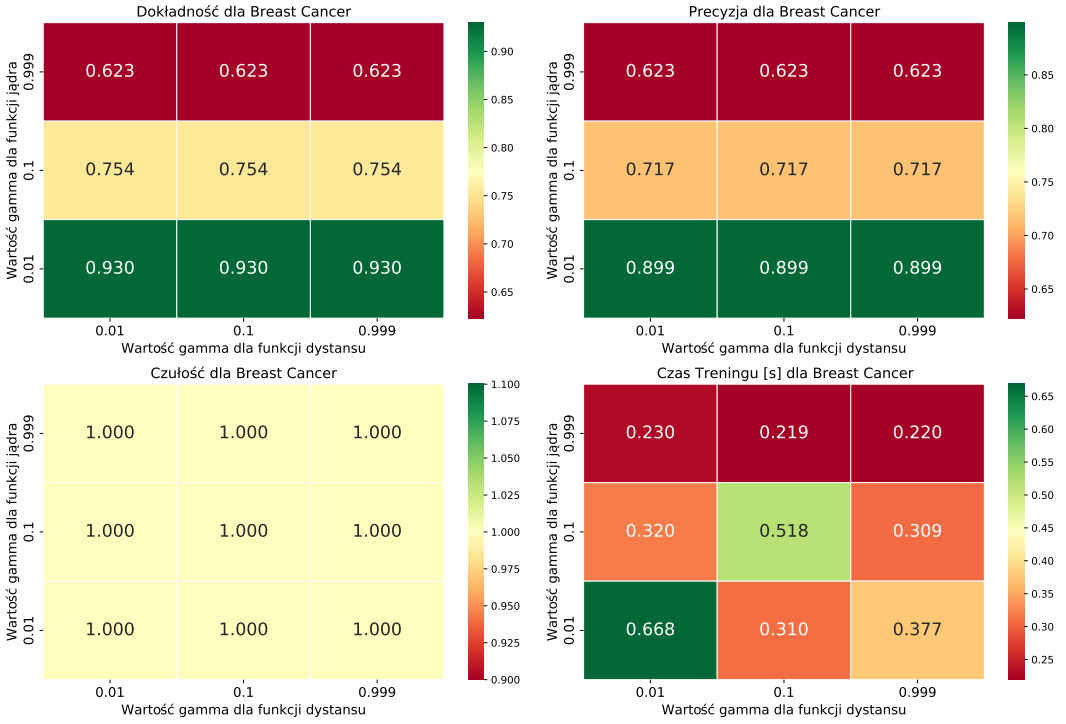
	CIFAR-10	**IMDb**	**Breast Cancer**
Dokładność	0,501 (0,001)	0,500 (0,000)	0,819 (0,151)
Precyzja	0,500 (0,001)	0,500 (0,000)	0,807 (0,150)
Czułość	0,997 (0,007)	1,000 (0,000)	0,993 (0,011)
F1	0,666 (0,001)	0,667 (0,000)	0,882 (0,090)
Czas uczenia [s]	3554,189 (5639,957)	21176,175 (287,370)	0,436 (0,214)

Źródło: opracowanie własne



Rysunek 2.7: Wyniki dla algorytmu LapSVM i zbioru Breast Cancer, w zależności od dobranych hiperparametrów

Źródło: opracowanie własne



Rysunek 2.8: Wyniki dla algorytmu LapSVM z funkcjami RBF, dla zbioru Breast Cancer, w zależności od wartości parametru gamma

Źródło: opracowanie własne

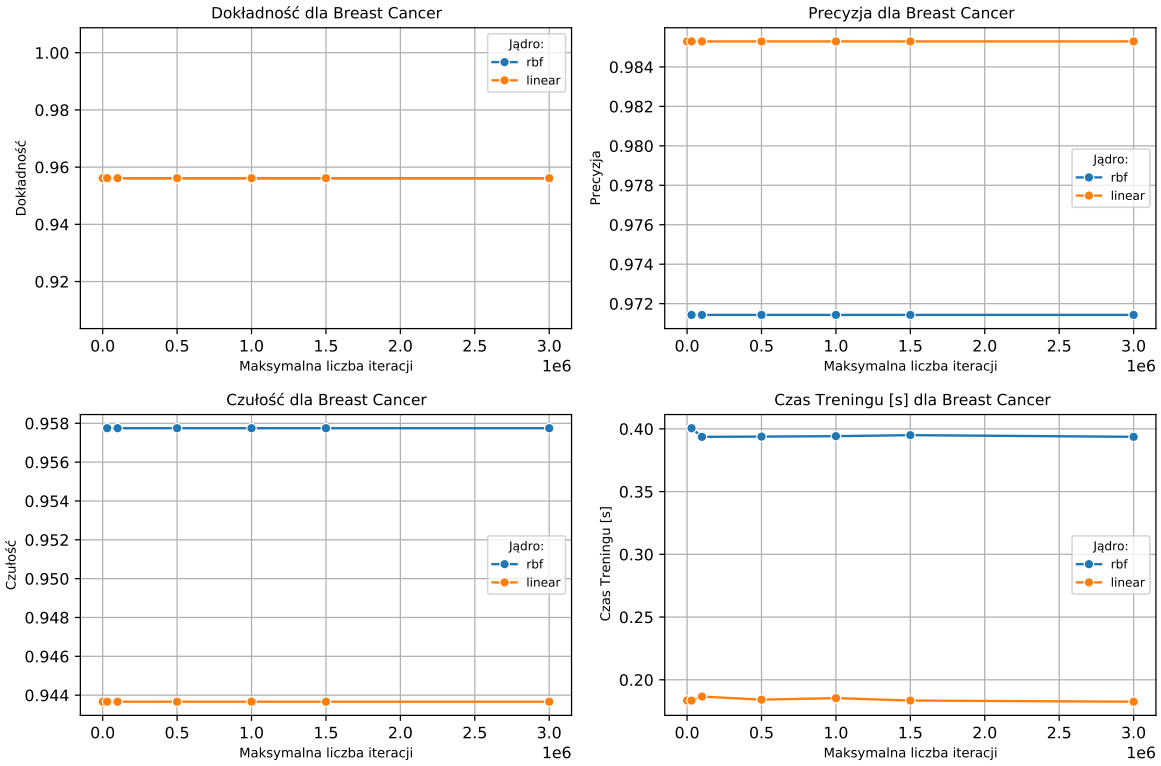
TSVM

Podczas przeprowadzania symulacji dla algorytmu TSVM, napotkano na problemy techniczne uniemożliwiające ukończenie analizy dla zbiorów CIFAR-10 i IMDb. Ograniczenie maksymalnej liczby iteracji sprawiało, że uczenie modelu nie kończyło się lub trwałoby dużo dłużej niż w pierwszym eksperymencie. W przypadku danych Breast Cancer również pojawił się ten problem, lecz znaczące zwiększenie liczby iteracji umożliwiło przeprowadzenie symulacji do końca. Otrzymane wyniki (patrz tabela 2.14 i rys. 2.9) nie ukazały wyraźnych różnic w skuteczności modelu. Liczba iteracji praktycznie nie ma wypływu na skuteczność algorytmu, natomiast zmiana jądra z radialnego na liniowe wpływa na precyzję, czułość i czas uczenia modelu, jednak również nie są to duże zmiany.

Tabela 2.14: Uśrednione wyniki z odchyleniem standardowym dla algorytmu TSVM dla zbioru Breast Cancer

Dokładność	0,956 (0,000)
Precyzja	0,979 (0,007)
Czułość	0,950 (0,007)
F1	0,964 (0,000)
Czas uczenia [s]	0,282 (0,109)

Źródło: opracowanie własne



Rysunek 2.9: Wyniki dla algorytmu TSVM, dla zbioru Breast Cancer, w zależności od dobranych hiperparametrów

Źródło: opracowanie własne

Label Propagation

W przypadku algorytmu Label Propagation zmieniano typ jądra, maksymalną liczbę iteracji oraz liczbę sąsiadów dla jądra KNN i wartość parametru gamma dla jądra RBF. Uśrednione wyniki przedstawione w tabeli 2.15 są niższe niż te uzyskane w pierwszym eksperymencie (dla domyślnych hiperparametrów), jednak stosunkowe duże wartości odchyleń standardowych wskazują, że hiperparametry mogą poprawić skuteczność tej metody, w szczególności w przypadku zbioru Breast Cancer.

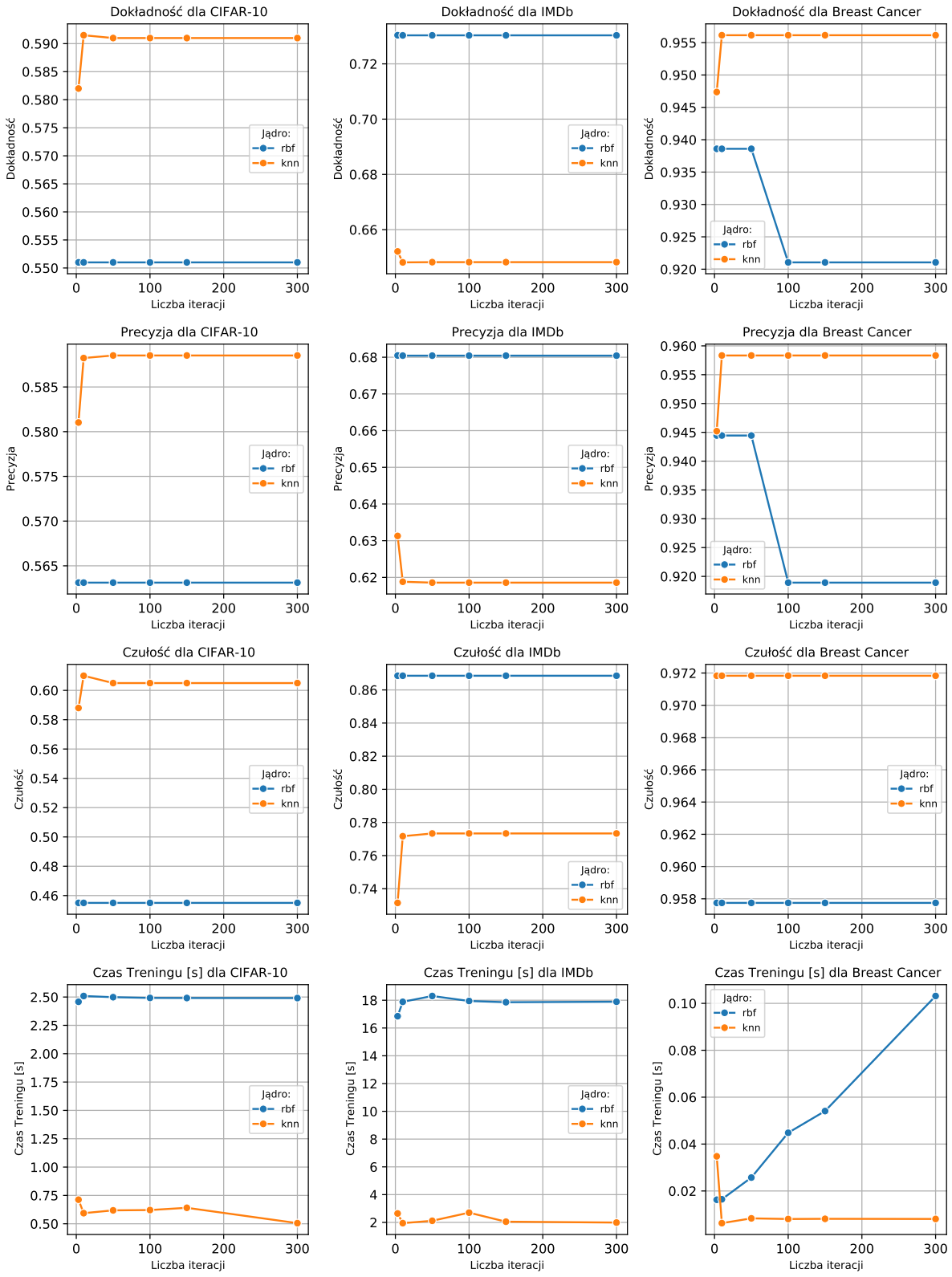
Na przedstawionych wykresach (rys. 2.10) możemy zaobserwować, że zwiększanie liczby iteracji nie wpływa na poprawę wyników. Może mieć to znaczenie jedynie dla bardzo małej liczby iteracji, po zwiększeniu której szybko następuje stabilizacja. Wyjątkiem jest zbiór Breast Cancer, gdzie dla jądra RBF po przekroczeniu 100 iteracji następuje spadek w dokładności i precyzji. Dodatkowo, jest to jedyny przypadek, gdzie wraz ze wzrostem powtórzeń wzrasta liniowo czas uczenia się modelu.

Wykresy przedstawione na rys. 2.11 ilustrują wpływ wyboru hiperparametru gamma w funkcji jądrowej RBF oraz liczby sąsiadów w jądrze KNN na skuteczność algorytmu. Dla danych CIFAR-10, widać, że dokładność dla jądra RBF osiąga maksimum dla wartości gamma równej 10^{-2}, co wskazuje na istnienie optymalnej wartości tego parametru, przy której model najlepiej radzi sobie z klasyfikacją obrazów. Zbyt niska lub zbyt wysoka wartość gamma prowadzi do znacznego spadku dokładności, co jest spowodowane niewystarczającym, bądź nadmiernym dopasowaniem modelu. Najlepiej obrazuje to wykres czułości. Warto również zwrócić uwagę, że uzyskana dokładność jest zdecydowanie lepsza niż w eksperymencie pierwszym o blisko 6 p.p. Natomiast dla jądra KNN, dokładność pozostaje niższa, lecz stabilna przy zmianie liczby sąsiadów.

Na wykresach dla zbioru IMDb, dokładność algorytmu z jądrem RBF osiągania maksimum przy niskich wartościach parametru gamma, a następnie spada. Jądro KNN ponownie przynosi niższą, choć bardziej stabilną, dokładność w porównaniu z wynikami dla zbioru CIFAR-10. W przypadku danych IMDb udało się uzyskać lepszą dokładność w granicach jednego p.p., w porównaniu z poprzednim eksperymentem.

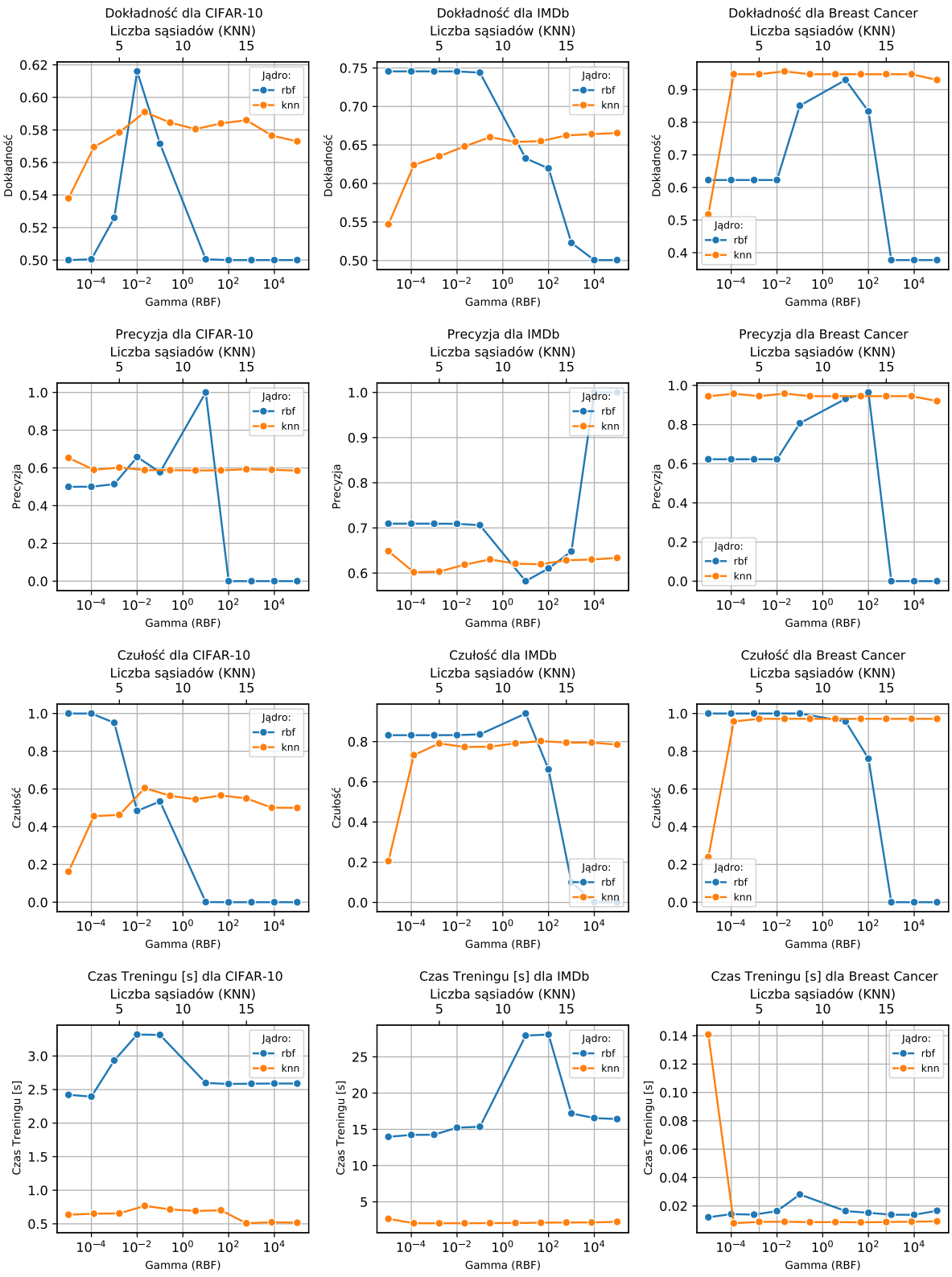
Wykresy z różną liczbą sąsiadów dla Breast Cancer potwierdzają, że jądro KNN jest lepszym wyborem niż jądro RBF, ponieważ dokładność jest stosunkowo wysoka i stabilna. Również w tym przypadku udało się poprawić wynik o dwa p.p. w porównaniu z dokładnością uzyskaną w ramach pierwszego eksperymentu.

Podsumowując, powyższe wyniki wskazują na to, że wybór jądra podobieństwa w algorytmie Label Propagation może mieć wpływ na wydajność modelu, chociaż ten wpływ wydaje się być ograniczony w kontekście liczby iteracji. Dodatkowo, metoda ta jest wrażliwa na wybór hiperparametru gamma w funkcji jądrowej RBF, przy czym optymalna wartość gamma różni się w zależności od specyfiki zbioru danych. Dla jądra KNN, liczba sąsiadów nie wpływa znacząco na otrzymane rezultaty.



Rysunek 2.10: Wyniki dla algorytmu Label Propagation, w zależności od dobranych hiperparametrów

Źródło: opracowanie własne



Rysunek 2.11: Wyniki dla algorytmu Label Propagation, w zależności od dobranych hiperparametrów, dla liczby iteracji `max_iter`=50

Źródło: opracowanie własne

Tabela 2.15: Uśrednione wyniki z odchyleniem standardowym dla algorytmu Label Propagation

	CIFAR-10	**IMDb**	**Breast Cancer**
Dokładność	0,557 (0,036)	0,662 (0,070)	0,831 (0,194)
Precyzja	0,519 (0,215)	0,670 (0,094)	0,812 (0,287)
Czułość	0,476 (0,264)	0,716 (0,253)	0,852 (0,309)
F1	0,468 (0,216)	0,644 (0,209)	0,819 (0,291)
Czas uczenia [s]	1,635 (1,044)	10,031 (8,490)	0,022 (0,029)

Źródło: opracowanie własne

2.5.4 Podsumowanie

W eksperymencie drugim, skoncentrowano się na analizie wrażliwości metod uczenia częściowo nadzorowanego na zmiany hiperparametrów. Symulacje przeprowadzone dla algorytmów Tri-Traing, Assemble, SemiBoost, LapSVM, TSVM i Label Propagation na zbiorach danych CIFAR-10, IMDb i Breast Cancer.

Zaobserwowane wyniki pokazały, że dostrojenie hiperparametrów często jest bardzo ważne dla wydajności modelu. W szczególności algorytm SemiBoost wykazywał duże zmiany skuteczności po dokonaniu zmiany jądra. Dla algorytmu Tri-Training zaobserwowano, że heterogeniczne konfiguracje klasyfikatorów bazowych mogą prowadzić do lepszych rezultatów niż homogeniczne. Z kolei w przypadku algorytmu Assemble, klasyfikator Random Forest wyróżniał się najwyższą dokładnością, a zwiększanie liczby iteracji nie wpływało pozytywnie na wyniki, co jest cennym wnioskiem natury praktycznej, ze względu na znaczny wzrost czasu uczenia, który towarzyszy zwiększeniu liczby powtórzeń.

Dla LapSVM, zmiany w funkcji odległości, jądra i wartościach parametru gamma nie przyniosły oczekiwanych rezultatów. Wykorzystywana implementacja algorytmu TSVM sprawiała trudności techniczne uniemożliwiające przeprowadzenie symulacji.

W przypadku Label Propagation, rezultaty wskazują na to, że wybór funkcji jądrowej RBF i KNN oraz odpowiednia optymalizacja hiperparametru gamma może znacząco wpływać na wyniki, szczególnie dla danych Breast Cancer, gdzie zastosowanie jądra KNN wydawało się być bardziej efektywne.

Podsumowując, przeprowadzone eksperymenty uwydatniły, jak kluczowe jest dostrojenie hiperparametrów dla każdego z algorytmów uczenia częściowo nadzorowanego i jak ich skuteczność różni się w zależności od specyfiki danych. Wyniki te podkreślają znaczenie doboru właściwych metod i ich parametrów w praktycznych zastosowaniach uczenia maszynowego, co może przyczynić się do znacznej poprawy skuteczności budowanych modeli klasyfikacyjnych.

2.6 Eksperyment 3: Porównanie skuteczności i stabilności metod w zależności od frakcji danych etykietowanych

2.6.1 Wprowadzenie

Uczenie częściowo nadzorowane, wykorzystujące zarówno etykietowane jak i nieetykietowane dane, oferuje możliwość poprawy skuteczności klasyfikacji w sytuacjach, gdzie dostęp do etykiet jest ograniczony. Ten eksperyment ma na celu zbadanie, jak zmieniająca się frakcja danych etykietowanych wpływa na wydajność i stabilność rozważanych algorytmów SSL. Przez frakcję danych etykietowanych, oznaczaną jako F_L, rozumiemy odsetek danych treningowych, który posiadał etykiety.

2.6.2 Metodologia

Ze względu na ograniczone zasoby obliczeniowe oraz biorąc pod uwagę wyniki otrzymane w drugim eksperymencie, w analizie uwzględnione zostaną algorytmy: Tri-Training, Assemble, SemiBoost i Label Propagation, w oparciu o zbiory danych CIFAR-10, IMDb i Breast Cancer. Wielkość frakcji danych etykietowanych F_L wykorzystywana w badaniach była następująca

$$F_L \in \{0{,}03;\ 0{,}05;\ 0{,}07;\ 0{,}1;\ 0{,}12;\ 0{,}15;\ 0{,}20;\ 0{,}25;\ 0{,}30\}.$$

Ze względu na chęć porównania stabilności algorytmów, w tym eksperymencie nie stosowano ustalonego podziału na dane etykietowane i nieetykietowane. Zamiast tego, dla każdej wartości F_L wykonano 30 powtórzeń, za każdym razem losowo dzieląc zbiór treningowy na część z etykietami i bez. W tabeli 2.16 przedstawiono wartości hiperparametrów wykorzystywane dla testowanych algorytmów.

2.6.3 Porównanie skuteczności algorytmów dla zmieniającej się frakcji obserwacji etykietowanych

Jak wspomniano, w tym eksperymencie skupiono się na analizie i porównaniu skuteczności algorytmów uczenia częściowo nadzorowanego w zależności od zmieniającej się frakcji danych etykietowanych. Głównym celem analizy jest zrozumienie, w jaki sposób proporcja danych etykietowanych i nieetykietowanych wpływa na wydajność rozważanych metod.

Uzyskane wyniki, dla poszczególnych zbiorów danych przedstawiono na wykresach 2.13, 2.14, 2.15 i 2.16. Zaznaczono na nich średnią, wraz z odchyleniem standardowym, wyznaczone na bazie otrzymanych 30 wyników dla każdej frakcji.

Na rysunku 2.12 przedstawiono krzywe uczenia, pokazujące wyniki osiągane przez wybrany algorytm uczenia nadzorowanego — Random Forest. Te referencyjne rezultaty dostarczają informacji o potencjalnym pułapie skuteczności, jaki może być osiągnięty, gdy wszystkie dane w zbiorze są etykietowane, podkreślając przy tym możliwy maksymalny potencjał ukryty w danych nieetykietowanych. Dodatkowo, aby ułatwić interpretację wyników, na wykresach zaznaczono maksymalną, rozważaną w przypadku algorytmów SSL, wielkość frakcji danych etykietowanych równą 0,3.

Tabela 2.16: Wartości hiperparametrów dla algorytmów i zestawów danych zastosowanych w Eksperymencie 3

Tri-Training

	CIFAR-10	IMDb	Breast Cancer
`base_estimator`	Random Forest	Random Forest	Random Forest
`base_estimator_2`	KNN	Linear SVM	Linear SVM
`base_estimator_3`	Decision Tree	Naive Bayes	Naive Bayes

Assemble

	CIFAR-10	IMDb	Breast Cancer
`base_estimator`	Random Forest	Random Forest	Random Forest
`T`	50	50	50

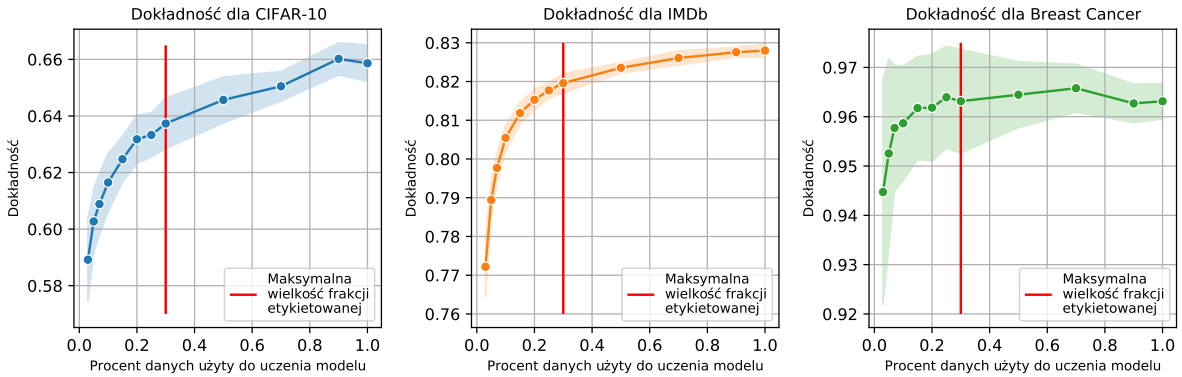
SemiBoost

	CIFAR-10	IMDb	Breast Cancer
`base_estimator`	Random Forest	Linear SVM	Linear SVM
`T`	10	10	10
`similarity_kernel`	KNN	KNN	KNN

Label Propagation

	CIFAR-10	IMDb	Breast Cancer
`kernel`	KNN	RBF	KNN
`max_iter`	100	100	100

Źródło: opracowanie własne



Rysunek 2.12: Uzyskane referencyjne wyniki dla algorytmu Random Forrest, w zależności od ilości użytych danych treningowych

Źródło: opracowanie własne

Tri-Training

Wykresy przedstawione na rys. 2.13 dla algorytmu Tri-Training ukazują tendencje w zmianie skuteczności modelu, w zależności od frakcji zbioru etykietowanego dla trzech różnych zbiorów danych: CIFAR-10, IMDb i Breast Cancer.

Dla zbioru danych CIFAR-10, zaobserwowano wyraźny trend wzrostowy w dokładności klasyfikacji wraz ze wzrostem frakcji zbioru etykietowanego. Przy najniższej rozważanej frakcji etykietowanych danych (3%), dokładność jest na średnim poziomie około 58%, a zwiększając frakcję do 30%, średnia dokładność wzrasta do około 62%. Krzywa uzyskana dla precyzji jest zbliżona kształtem i wartościami do krzywej dokładności. W przypadku czułości jej średnia wartość oscyluje wokół 55%. Rozpiętość odchylenia standardowego, widoczna jako zacieniony obszar, utrzymuje się na podobnym poziomie dla precyzji i dokładności, a dla czułości zmiejsza się wraz ze wzrostem rozmiaru frakcji, co może sugerować, że algorytm staje się stabilniejszy w miarę dostarczania większej liczby etykietowanych przykładów. Porównując otrzymane wyniki z wykresem 2.12, zauważono, że średnia dokładność jest niższa, niż w przypadku uczenia w pełni nadzorowanego.

Analizując wykresy otrzymane dla zbioru danych IMDb, trend wzrostowy jest jeszcze bardziej wyraźny niż w przypadku CIFAR-10. Dodatkowo ma to również przełożenie na czułość modelu. W tym przypadku, algorytm Tri-Training wydaje się zyskiwać znacznie więcej dzięki dodatkowym danym etykietowanym. Również uzyskane odchylenie standardowe jest znacznie mniejsze, co wskazuje na większą odporność modelu na zmieniające się dane. W tym przypadku algorytm Tri-Training uzyskuje zdecydowanie lepszą dokładność (84,5%) niż wybrany algorytm uczenia nadzorowanego (82%).

Dla zbioru danych Breast Cancer dokładność i precyzja zaczynają się od wysokiego poziomu około 94% przy 3% danych etykietowanych, a wraz ze wzrostem frakcji etykietowanych danych, szybko stabilizują się na poziomie ponad 96% przy frakcji 30%. Czułość bardzo szybko stabilizuje się na poziomie około 98,5%, a wraz ze wzrostem frakcji danch etykietowanych znacznie maleje odchylenie standardowe. Otrzymane wyniki są bardzo zbliżone do tych, które otrzymujemy w przypadku klasyfikatora Random Forest.

Dla wszystkich zbiorów danych czas uczenia modelu wzrastał razem z rozmiarem frakcji etykietowanej oraz zaobserwowano niewielkie odchylenia standardowe. Podkreślmy, że widoczne na rys. 2.13 odchylenia standardowe w przypadku danych Breast Cancer wydają się większe niż dla innych danych (obserwujemy pasma o dużej szerokości) jedynie ze względu na mały rozrzut wyników, co wpływa na skalę wykresów.

Assemble

Rysunek 2.14 przedstawia wyniki (krzywe uczenia) uzyskane dla algorytmu Assemble. Dla wszystkich trzech zbiorów danych widać stopniowy i konsekwentny wzrost dokładności w miarę zwiększania frakcji etykietowanych danych. Niestety w żadnym przypadku nie udaje się osiągnąć progu ustanowionego przez metodę nadzorowaną, przy czym było to bliskie dla zbioru Breast Cancer.

Dla zbioru danych CIFAR-10 zarówno dokładność jak i precyzja zmieniają się od około 56% do około 62%. Wzrost jest niemal liniowy, co może sugerować, że algorytm Assemble w sposób efektywny wykorzystuje dodatkowe informacje pochodzące z etykietowanych danych, systematycznie poprawiając wydajność modelu wraz ze wzrostem ich ilości. Podobnie jak w przypadku poprzedniego algorytmu, czas treningu ponownie jest wprost proporcjonalny do wielkości frakcji.

Na wykresie dla danych IMDb również zaobserwowano znaczący wzrost dokładności, który jest bardziej wyraźny niż w przypadku zbioru CIFAR-10. Czułość poprawia się o około 5 p.p, podobnie jak dla zbioru obrazów, przy czym zauważalny jest spadek odchylenia standardowego. Czas uczenia ponownie rośnie wraz z rozmiarem frakcji.

W przypadku zbioru Breast Cancer, dokładność i precyzja, po początkowym wzroście, stabilizują się na poziomie około 95,5%. Czułość pozostaje na stałym, bardzo wysokim

poziomie około 98,5%. Inaczej niż w poprzednich symulacjach zachowywał się czas treningu, który utrzymywał się na stałym pozomie i wyniósł około 28 sekund, niezależnie od wielkości frakcji zbioru etykietowanego.

SemiBoost

Kolejnym badanym algorytmem jest SemiBoost, dla którego wyniki (otrzymane krzywe uczenia) zostały przedstawione na rysunku 2.15. Dla CIFAR-10 obserwujemy wzrost dokładności w miarę zwiększania frakcji danych etykietowanych. Dokładność zwiększa się z około 0,54 przy 3% danych etykietowanych do aż 0,63 przy 30% danych etykietowanych. Precyzja zachowuje podobną tendencję. Czułość oscyluje pomiędzy wartościami 0,5, a 0,6. Odchylenia standardowe maleją wraz ze wzrostem frakcji. W porównaniu z metodami nadzorowanymi, SemiBoost osiąga obiecujące wyniki, niewiele gorsze niż Random Forest.

W przypadku zbioru IMDb uzyskane krzywe są bardziej stabilne, wszystkie kryteria rosną i mają znacznie mniejsze odchylenia standardowe, w porównaniu z danymi CIFAR-10. Dla tych danych algorytmowi SemiBoost udaje się osiągnąć lepsze rezultaty niż metodzie nadzorowanej o około 1 p.p.

Wyniki dla zbioru Breast Cancer pokazują wzrost dokładności i precyzji z początkowych wartości około 0,90 przy 3% danych etykietowanych do około 0,96 przy 30%. Czułość zaczyna się od wysokiego poziomu 0,94 przy 3% danych etykietowanych i wzrasta o 2 p.p. Uzyskane wyniki są gorsze niż w przypadku algorytmu Random Forest, lecz należy zwrócić uwagę na duże odchylenia standardowe w uzyskanych wynikach.

Czas treningu tego algorytmu nie wykazuje jednoznacznego trendu. Dla danych CIFAR-10 czas rośnie liniowo, a wyniki nie odbiegają wyraźnie od średniej. W przypadku pozostałych zbiorów obserwujemy znacznie większe odchylenia standardowe, a średnie wartości oscylują w granicach 191 sekund dla zbioru IMDb oraz 0,2 s dla Breast Cancer.

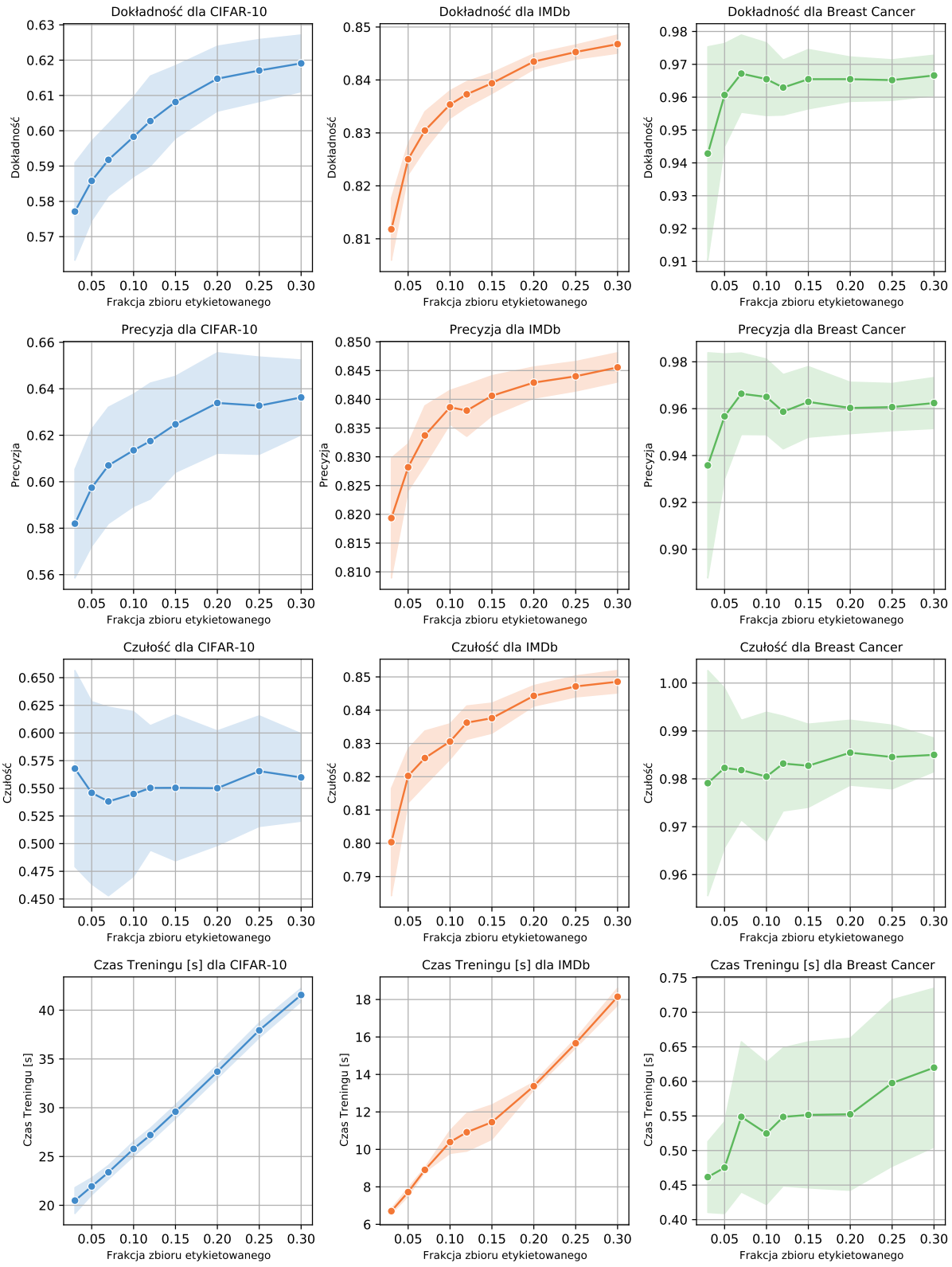
Label Propagation

Ostatnim rozważanym algorytmem jest Label Propagation. Analizując wykresy (rys. 2.16) uzyskane dla zbioru danych CIFAR-10 obserwujemy stały wzrost dokładności. Precyzja pokazuje podobny trend wzrostowy, zaczynając od około 52% i dochodząc do około 58%. Można zaobserwować zdecydowaną stabilizację wyników przy zwiększeniu frakcji obserwacji etykietowanych. Czułość oscyluje w granicach 50%.

Na wykresach dla zbioru IMDb zaobserwowano wzrost wszystkich trzech kryteriów, jednak w tym przypadku nie otrzymujemy wyników wyróżniających się na tle wcześniejszych algorytmów. Dokładność zwiększyła się o około 20 p.p. Precyzja i czułość również wzrosły. W przypadku tego zbioru, Label Propagation uzyskał największe odchylenia standardowe dla większości rozmiarów frakcji, w porównaniu do poprzednich algorytmów.

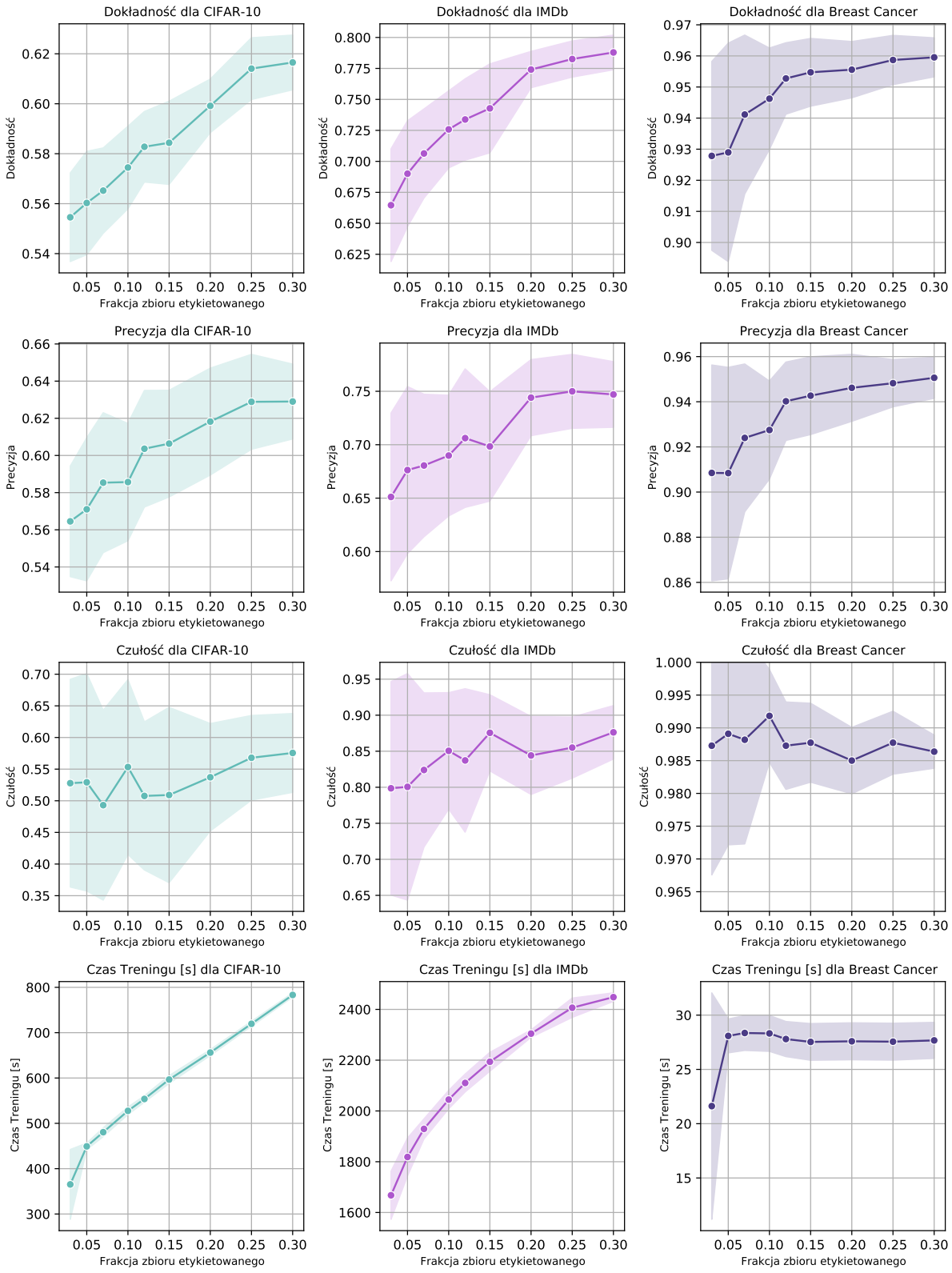
Dla zbioru Breast Cancer, wyniki pokazują bardzo wysokie wartości wszystkich miar skuteczności już przy niskich frakcjach danych etykietowanych. Precyzja i dokładność szybko osiągają stabilizację, w przeciwieństwie do czułości, która stale zmiejsza swoją wartość. Odchylenia standardowe są znacznie niższe dla tego zbioru danych, co może wskazywać na to, że algorytm Label Propagation jest szczególnie dobrze dostosowany do danych o mniejszej wymiarowości.

Biorąc pod uwagę czas uczenia, jest to jedyny algorytm, dla którego czas treningu maleje, wraz ze wzrostem frakcji zbioru etykietowanego.



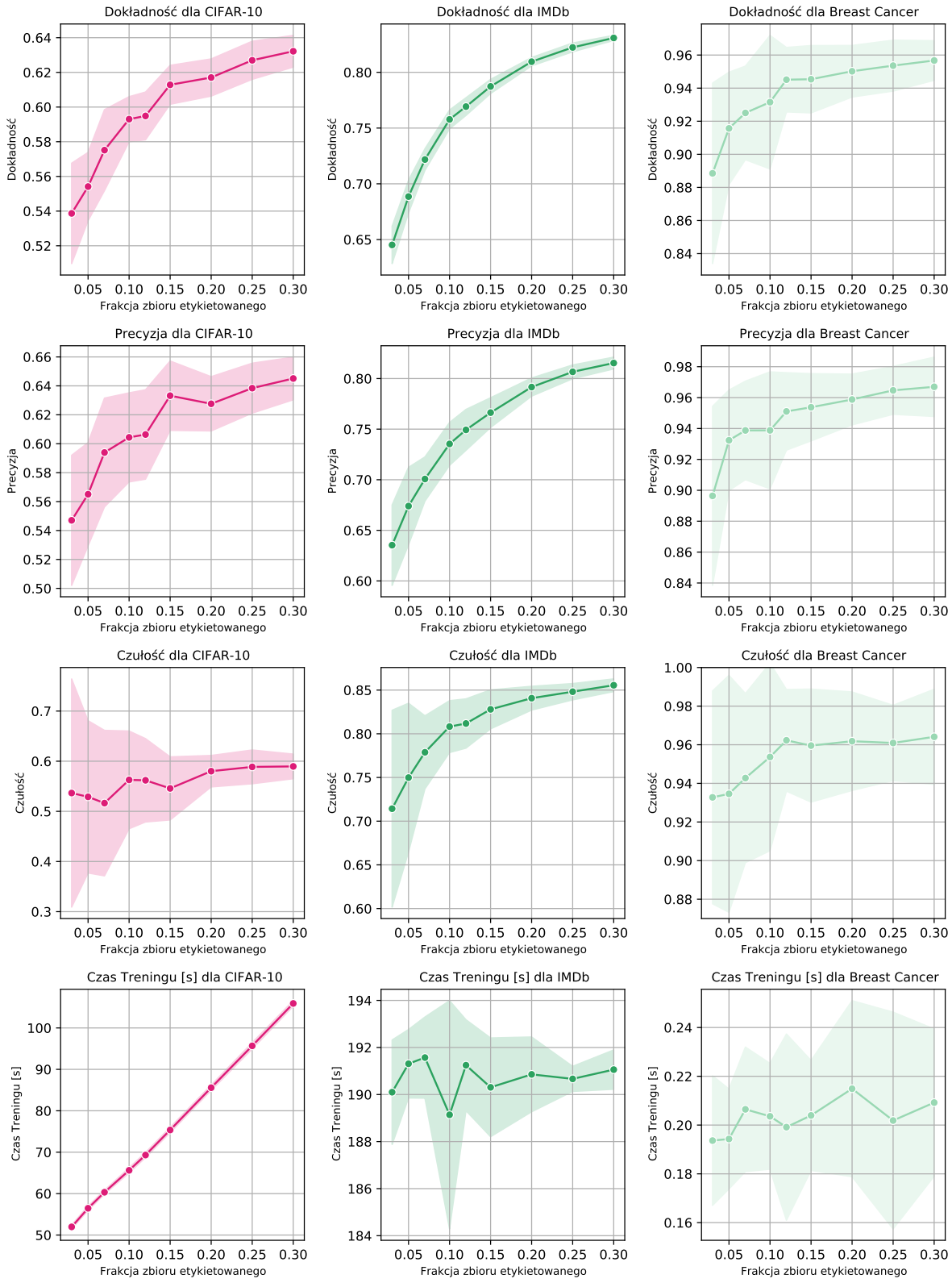
Rysunek 2.13: Wyniki dla algorytmu Tri-Training, w zależności od frakcji zbioru etykietowanego

Źródło: opracowanie własne



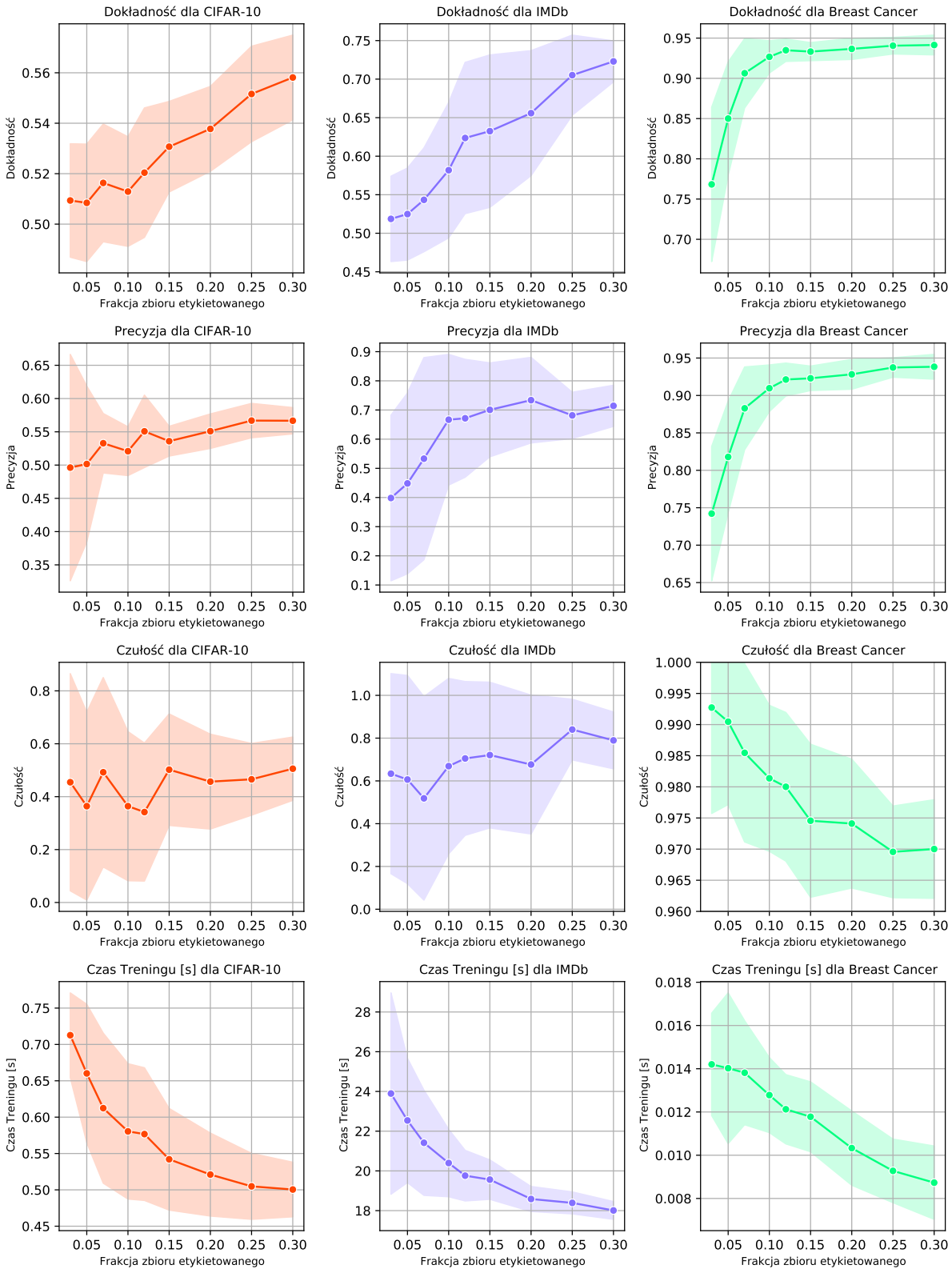
Rysunek 2.14: Wyniki dla algorytmu Assemble, w zależności od frakcji zbioru etykietowanego

Źródło: opracowanie własne



Rysunek 2.15: Wyniki dla algorytmu SemiBoost, w zależności od frakcji zbioru etykietowanego

Źródło: opracowanie własne



Rysunek 2.16: Wyniki dla algorytmu Label Propagation, w zależności od frakcji zbioru etykietowanego

Źródło: opracowanie własne

Podsumowanie

W przypadku metody Tri-Training zauważono, że wzrost frakcji danych etykietowanych w zbiorze CIFAR-10 przekłada się na umiarkowany wzrost dokładności z około 58% do 62%. Precyzja i czułość również wykazują wzrost, chociaż czułość rośnie w mniejszym stopniu. Obserwowane odchylenia standardowe pozostają na podobnym poziomie. Na zbiorze IMDb Tri-Training wykazuje znacznie lepszą dokładność w porównaniu do metody nadzorowanej, natomiast dla Breast Cancer osiąga wyniki zbliżone do Random Forest, z dokładnością przekraczającą 96% przy 30% frakcji danych etykietowanych.

Wyniki dla metody Assemble pokazują konsekwentny wzrost dokładności dla wszystkich trzech zbiorów danych, chociaż nie przekraczają one rezultatów metody nadzorowanej. W przypadku CIFAR-10, obserwuje się prawie liniowy wzrost dokładności i precyzji, a dla IMDb zauważalny jest również wzrost czułości. Dla zbioru Breast Cancer algorytm Assemble jest bliski osiągnięcia progów ustanowionych przez wybraną metodę uczenia nadzorowanego (Random Forest), z dokładnością stabilizującą się na poziomie około 95,5%.

SemiBoost wykazuje wzrost dokładności i precyzji dla danych CIFAR-10, chociaż nie obserwujemy równie wyraźnej tendencji dla czułości. Dla zbioru IMDb wyniki również rosną, z bardzo małym odchyleniem standardowym dla dokładności i większym dla czułości i precyzji. Dla Breast Cancer dokładność i precyzja znacząco wzrastają w porównaniu do początkowych wartości, a czułość utrzymuje się na wysokim poziomie.

Label Propagation prezentuje stały wzrost dokładności na zbiorze CIFAR-10, z precyzją i czułością utrzymującymi się na umiarkowanym poziomie. Na zbiorze IMDb wszystkie trzy miary skuteczności wykazują wzrost, ale z dużym odchyleniem standardowym. Dla Breast Cancer algorytm osiąga wysokie wartości kryteriów już przy niskich frakcjach danych etykietowanych, co sugeruje dobrą adaptację metody do zbioru o mniejszej wymiarowości.

Porównując czas uczenia dla wszystkich rozważanych w tym eksperymencie algorytmów SSL, Label Propagation wyróżnia się zmniejszającym czasem treningu wraz ze wzrostem frakcji etykietowanych danych, co odróżnia go od metod Tri-Training, Assemble i SemiBoost, gdzie czas uczenia wzrasta wraz z wielkością frakcji danych etykietowanych (dla SemiBoost dotyczy to jedynie zbioru CIFAR-10, w pozostałych przypadkach nie obserwujemy wyraźnej zmiany w czasie uczenia).

2.6.4 Porównanie stabilności algorytmów dla zmieniającej się frakcji obserwacji etykietowanych

W drugiej części analizy skupiono się na badaniu stabilności algorytmów uczenia częściowo nadzorowanego dla zmieniającej się frakcji danych etykietowanych. Stabilność, w tym kontekście, rozumiana jest jako uzyskanie powtarzalnych wyników dla ustalonej ilości danych etykietowanych, co możemy utożsamiać z małą wariancją (rozrzutem) wartości kryteriów dla wielokrotnie wyznaczanych (losowanych) podzbiorów danych etykietowanych. Innymi słowy, w przypadku metody stabilnej stosunkowo niewielkie zaburzenie danych nie powinno prowadzić do istotnie różniących się wyników.

Wyniki eksperymentów, ilustrujące stabilność czterech algorytmów uczenia częściowo nadzorowanego: Tri-Training, Assemble, SemiBoost i Label Propagation, z uwzględnieniem dziewięciu różnych frakcji danych etykietowanych (0,03; 0,05; 0,07; 0,10; 0,12; 0,15; 0,20; 0,25; 0,30), w zależności od zbioru danych, zostały zaprezentowane na wykresach pudełkowych (rysunki 2.17, 2.18 i 2.19).

CIFAR-10

Wykresy pudełkowe widoczne na rys. 2.17 pozwalają na porównanie stabilności algorytmów dla zbioru danych CIFAR-10. Analizując te wyniki, możemy zauważyć ogólną tendencję wzrostową dokładności w miarę zwiększania się frakcji danych etykietowanych. Każdy z algorytmów wykazał stopniową poprawę wydajności.

Metoda Tri-Training konsekwentnie poprawia swoją skuteczność wraz ze wzrostem frakcji danych etykietowanych, czemu towarzyszy wyraźnie zmniejszający się rozrzut wyników (wartości kryterium dokładności). Ta obserwacja wskazuje na stabilność algorytmu Tri-Training oraz pozytywny wpływ dodatkowych danych etykietowanych.

Algorytm Assemble, podobnie jak Tri-Training, poprawia swoją dokładność wraz ze wzrostem frakcji danych etykietowanych, ale wykazuje większy rozrzut wyników. Mimo że jego średnia dokładność wzrasta to wydajność jest bardziej zmienna i może być w dużo większym stopniu uzależniona od specyfiki danych.

Algorytm SemiBoost wykazuje znaczną poprawę dokładności przy większych frakcjach danych etykietowanych, osiągając przy tym podobny rozrzut wyników co algorytmy Tri-Training i Assemble. Wykresy bardzo dobrze obrazują jak duży wpływ na jego skuteczność ma frakcja danych etykietowanych. Wraz z jej wzrostem, średnia wartość kryterium rośnie szybciej niż w przypadku pozostałych algorytmów, finalnie uzyskując najwyższy wynik.

Metoda Label Propagation charakteryzuje się najniższą dokładnością dla wszystkich rozważanych frakcji danych etykietowanych. Ponadto rozrzut otrzymanych wyników jest największy dla każdej z frakcji, co czyni go najmniej stabilną metodą dla tego zbioru danych. Co więcej, w przypadku tego algorytmu zwiększenie frakcji danych etykietowanych nie przynosi poprawy stabilności.

IMDb

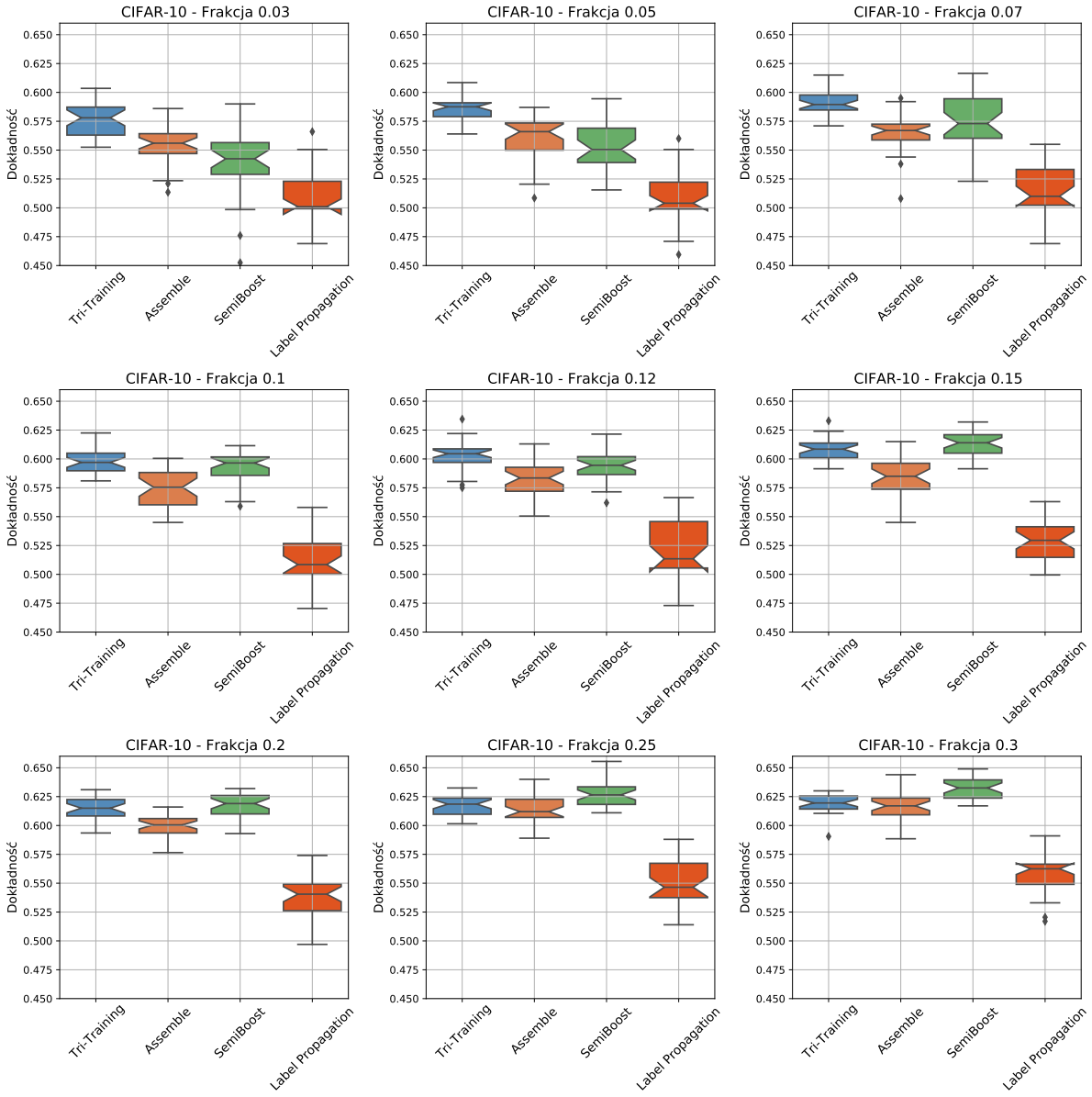
Wykresy pudełkowe widoczne na rys. 2.18 przedstawiają wyniki dokładności algorytmów zastosowanych dla zbioru danych IMDb przy różnych frakcjach danych etykietowanych.

Algorytm Tri-Training utrzymuje najwyższą dokładność w całym rozważanym zakresie frakcji etykietowanych, konsekwentnie zachowując przy tym bardzo niski rozrzut wyników. Niewątpliwie wypada najlepiej w porównaniu z pozostałymi konkurencyjnymi metodami.

Algorytm Assemble prezentuje znacznie niższą dokładność oraz zdecydowanie większą zmienność w wynikach, w porówaniu do metody Tri-Trianing. W tym przypadku rozstęp międzykwartylowy dla większości frakcji wynosił ponad 0,05, a różnica między maksimum i minimum nawet 0,15. Dopiero od frakcji 20% można zauważyć pewną stabilizację.

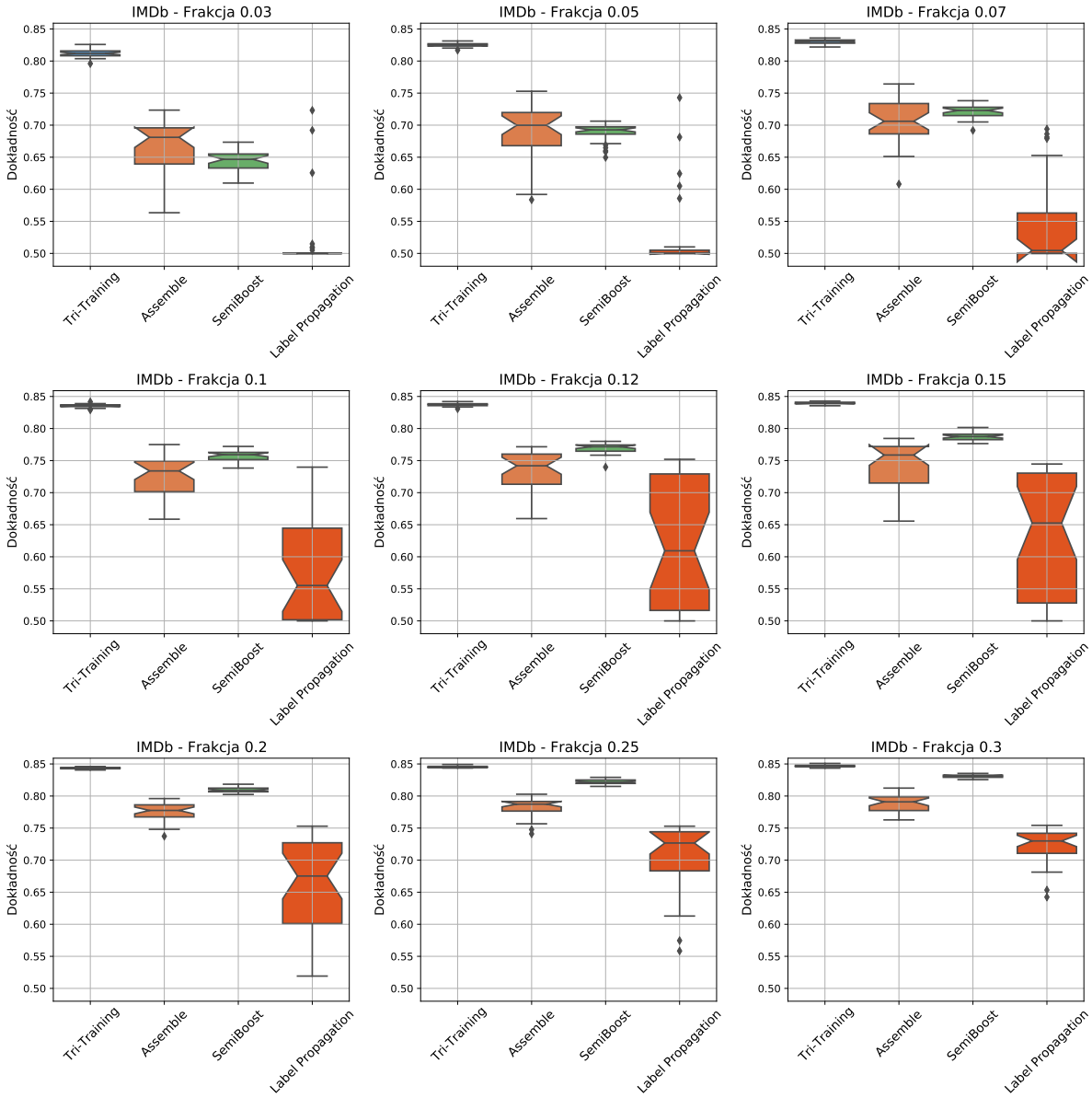
Wyniki uzyskane przez algorytm SemiBoost ponownie znacznie rosną wraz ze wzrostem frakcji, zachowując przy tym dużą stabilność — największa różnica między maksimum i minimum wynosi około 0,07 dla frakcji 3%. Wartości dokładności nie są tak wysokie jak dla metody Tri-Training, jednak dla większej frakcji danych etykietowanych może być on dobrą alternatywą.

Metoda Label Propagation wykazuje znaczną poprawę dokładności przy wyższych frakcjach danych etykietowanych, zaczynając od 0,5 dla frakcji 3% i kończąc na średnio 0,75. Może to oznaczać, że Label Propagation potrzebuje większej ilości danych etykietowanych, aby generować precyzyjne prognozy. Rozrzut wyników w przypadku metody Label Propagation jest największy, w szczególności przy najniższych frakcjach, co wskazuje na jego mniejszą stabilność w warunkach bardzo ograniczonej ilości danych etykietowanych.



Rysunek 2.17: Wykresy pudełkowe porównujące stabilności algorytmów dla zbioru CIFAR-10

Źródło: opracowanie własne



Rysunek 2.18: Wykresy pudełkowe porównujące stabilności algorytmów dla zbioru IMDb

Źródło: opracowanie własne

Breast Cancer

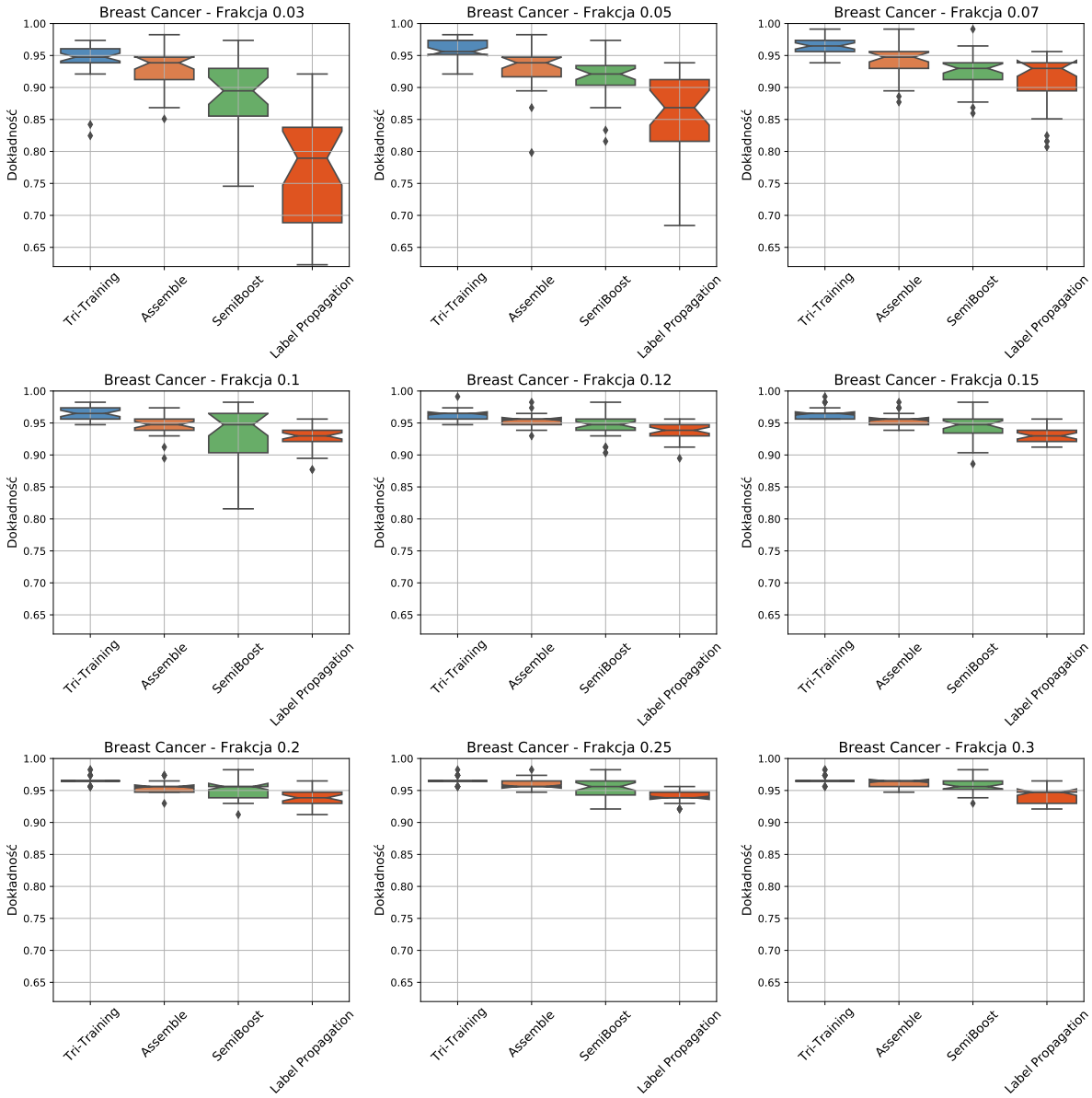
Dla zbioru Breast Cancer wyniki (w postaci wykresów pudełkowych) przedstawiono na rys. 2.19.

Algorytm Tri-Training ponownie wykazuje wysoką dokładność dla wszystkich rozważanych frakcji, z bardzo niewielkim rozrzutem wyników, co świadczy o jego stabilności i efektywności w wykorzystywaniu dostępnych danych etykietowanych. Możemy na tej podstawie wywnioskować, że algorytm Tri-Training jest niezawodny w przypadku różnych scenariuszy dotyczących dostępności etykiet, zachowując jednocześnie stosunkowe wąskie zakresy niepewności wyników.

Algorytm Assemble osiąga podobne, choć nieco gorsze wyniki, z większym rozrzutem, szczególnie przy mniejszych frakcjach etykietowanych danych. Z tego względu, można uznać tę metodę za mniej dokładną i mniej stabilną w porównaniu do algorytmu Tri-Training.

Metoda SemiBoost ma tendencję do niższej średniej dokładności, szczególnie przy mniejszych frakcjach danych etykietowanych. Wykazuje znaczną poprawę stabilności przy zwiększeniu liczby etykiet. Dla tego zbioru danych można ją uznać za mniej optymalny wybór, w porównaniu z poprzednimi algorytmami.

Algorytm Label Propagation dla mniejszych rozmiarów frakcji, wykazuje największy zakres wyników w porównaniu do innych algorytmów, co może wskazywać na jego małą stabilność. Dopiero od frakcji danych etykietowanych 0,1 algorytm zaczyna dokładniej przewidywać etykiety klas, zmniejszając jednocześnie swój rozrzut.



Rysunek 2.19: Wykresy pudełkowe porównujące stabilności algorytmów dla zbioru Breast Cancer

Źródło: opracowanie własne

Podsumowanie

Analiza stabilności algorytmów uczenia częściowo nadzorowanego, w zależności od zmieniającej się frakcji danych etykietowanych, ujawniła istotne różnice w dokładności i powtarzalności wyników między czterema badanymi metodami: Tri-Training, Assemble, SemiBoost i Label Propagation. Na podstawie wyników eksperymentów przeprowadzonych na zbiorach danych CIFAR-10, IMDb oraz Breast Cancer, można wyciągnąć następujące wnioski.

Algorytm Tri-Training wykazał się największą stabilnością i bardzo wysoką dokładnością wśród badanych metod, na wszystkich zbiorach danych. Charakteryzuje się on niewielkim rozrzutem wyników, co wskazuje na jego niezawodność i skuteczność w wykorzystywaniu dostępnych danych etykietowanych, niezależnie od ich frakcji.

W przypadku metody Assemble otrzymano zróżnicowane wyniki z większym rozrzutem, szczególnie przy mniejszych frakcjach danych etykietowanych, co wskazuje na większą zmienność w wydajności algorytmu. Chociaż średnia dokładność ogólnie rośnie wraz ze wzrostem frakcji danych etykietowanych, większy rozrzut może wskazywać na większą wrażliwość tej metody na specyfikę danych. W przypadku danych IMDb algorytm Assemble nie dorównał jednak metodzie Tri-Training.

Algorytm Semiboost wyróżniał się przede wszystkim znacznie szybszym wzrostem dokładności oraz zmniejszaniem rozrzutu, wraz ze rozbudową frakcji etykietowanej. Jego ogólna wydajność była bardzo dobra, w szczególności dla zbioru CIFAR-10 udało mu się uzyskać najlepsze rezultaty, porównywalne z metodą nadzorowaną Random Forest.

Label Propagation okazał się być algorytmem o najmniejszej stabilności i najniższej dokładności przy niższych frakcjach danych etykietowanych. Niemniej jednak, obserwowany wzrost dokładności przy wyższych frakcjach wskazuje na jego potencjał do efektywnej predykcji w przypadku, gdy dostępna jest większa ilość danych etykietowanych.

Podsumowując, wszystkie algorytmy poprawiają swoją skuteczność wraz ze wzrostem frakcji danych etykietowanych, jednak różnią się pod względem stabilności i zależności od ilości dostępnych danych etykietowanych. Tri-Training okazuje się być algorytmem o ogólnie największej stabilności i dokładności, SemiBoost i Assemble prezentują niższą stabilność, natomiast Label Propagation wymaga większej ilości danych etykietowanych, aby osiągnąć porównywalne wyniki.

Podsumowanie

W niniejszej pracy skoncentrowano się na porównaniu algorytmów uczenia częściowo nadzorowanego (SSL), wykorzystując odpowiednio zaprojektowane eksperymenty oceniające ich wydajności na różnorodnych zestawach danych.

W części teoretycznej przedstawiono zagadnienia dotyczące klasyfikacji oraz omówiono wybrane, najważniejsze algorytmy uczenia nadzorowanego. Zdefiniowane zostały również podstawowe miary wykorzystane do oceny skuteczności klasyfikacji binarnej. Następnie skupiono się na algorytmach uczenia częściowo nadzorowanego, które łączą w sobie elementy uczenia nadzorowanego i nienadzorowanego, wykorzystując dane etykietowane i nieetykietowane do poprawy dokładności predykcji. Przedstawiono kluczowe założenia SSL, takie jak założenie bliskości punktów, założenie niskiej gęstości oraz założenie dotyczące rozmaitości (ang. *manifolds*). Następnie zamieszczony został teoretyczny opis rozważanych algorytmów SSL, takich jak: Tri-Training, Assemble, SemiBoost, LapSVM, TSVM i Label Propagation.

W części praktycznej przeprowadzono szczegółową analizę porównawczą rozważanych algorytmów na bazie trzech różnych zbiorów danych: CIFAR-10, IMDb Reviews oraz Breast Cancer Wisconsin. Sformułowane zostały najważniejsze pytania badawcze, na które udało się odpowiedzieć, dzięki eksperymentom przeprowadzonym dla różnych konfiguracji hiperparametrów oraz różnych wielkości frakcji danych etykietowanych. Wyniki pokazują, że efektywność metod uczenia częściowo nadzorowanego zależy od złożoności danych i wybranego algorytmu. Algorytmy takie jak Tri-Training i TSVM wyróżniły się dobrymi wyniki dla wszystkich rozważanych danych, podczas gdy pozostałe metody, takie jak SemiBoost i LapSVM, osiągały lepsze rezultaty dopiero w przypadku mniej złożonych zbiorów danych, jak Breast Cancer Wisconsin.

W przypadku eksperymentu opartego na domyślnych wartościach hiperparametrów, podstawowym wnioskiem jest to, że choć teoretyczne zalety metod SSL związane z wykorzystaniem danych nieetykietowanych są istotne, praktyczna skuteczność algorytmów nie zawsze dorównuje oczekiwaniom. W szczególności, w porównaniu z metodami uczenia nadzorowanego, algorytmy SSL nie zawsze osiągały lepsze wyniki. Ponadto wybór najbardziej odpowiedniej metody SSL zależy od specyfiki danych. Przykładowo, algorytmy Tri-Training i TSVM wykazały się uniwersalnością i zadowalającą skutecznością w przypadku różnych zagadnień klasyfikacyjnych, podczas gdy inne metody, takie jak LapSVM i SemiBoost, były bardziej efektywne jedynie w określonych warunkach.

W ramach drugiego eksperymentu ustalono siatkę wartości hiperparametrów dla każdego algorytmu, by zbadać zależności między zmianami tych parametrów, a efektywnością modeli. Analiza wykazała, że Tri-Training wykazał dużą wrażliwość na zmianę parametrów, z lepszymi wynikami dla heterogenicznych konfiguracji klasyfikatorów bazowych. Assemble wykazał największą skuteczność z klasyfikatorem bazowym Random Forest, choć zwiększanie liczby iteracji nie wpłynęło pozytywnie na wyniki. W przypadku algorytmu SemiBoost zauważono natomiast, że optymalizacja hiperparametrów, takich jak wybór

bazowego klasyfikatora, liczba iteracji i typ jądra podobieństwa, jest kluczowa dla jego efektywności. LapSVM nie radził sobie dobrze z wielowymiarowymi danymi, choć dla mniej złożonych struktur, jak dane Breast Cancer, zmiany funkcji jądrowej i wartości parametru gamma miały wpływ na wyniki. W przypadku metody TSVM wystąpiły problemy techniczne, ograniczając możliwość dokładnej analizy. Badając skuteczność algorytmu Label Propagation, zauważono natomiast, że wybór jądra podobieństwa, ma wpływ na wydajność tej metody. Rozważane algorytmy SSL charakteryzowały się w ogólności dość dużą wrażliwością na zmiany hiperparametrów, co może znacząco wpływać na ich efektywność. W niektórych przypadkach, optymalizacja umożliwiała osiągnięcie wyników porównywalnych lub nieznacznie lepszych niż w przypadku metod uczenia nadzorowanego.

W eksperymencie dotyczącym wpływu wielkości frakcji danych etykietowanych oceniano, jak zmieniająca się proporcja danych etykietowanych i nieetykietowanych wpływa na wydajność oraz stabilność rozważanych algorytmów SSL. Na podstawie otrzymanych wyników możemy stwierdzić, że Tri-Training był najbardziej stabilny, wykazując niezawodność i skuteczność w wykorzystaniu dostępnych danych etykietowanych. Assemble prezentował zróżnicowane wyniki, z większym rozrzutem przy mniejszych frakcjach danych etykietowanych. SemiBoost charakteryzował się umiejętnym wykorzystaniem dodatkowych danych etykietowanych, osiągając bardzo dobre rezultaty dla największej rozważanej frakcji etykietowanej. Label Propagation był mniej stabilny i efektywny przy niższych frakcjach danych etykietowanych, ale uzyskiwał lepsze wyniki dla wyższych frakcji.

Podsumowując, przeprowadzone w ramach pracy badania pozwoliły na szczegółową analizę wydajności wybranych algorytmów uczenia częściowo nadzorowanego. Uzyskane wyniki wykazały, że skuteczność metod SSL pozostaje wysoce zależna od struktury danych i rozważanego zagadnienia klasyfikacyjnego i wymaga dalszych badań. Dostrojenie hiperparametrów ma kluczowe znaczenie dla wydajności każdego z uwzględnionych w analizie modeli SSL, w szczególności dla SemiBoost, który dopiero w Eksperymencie 2 i 3 zaprezentował swój pełny potencjał. Algorytmy różnią się pod względem stabilności i są zależne od ilości dostępnych danych etykietowanych. Tri-Training okazał się być najbardziej stabilnym i dokładnym algorytmem, SemiBoost i Assemble prezentowały dobrą stabilność, a Label Propagation wymagał większej ilości danych etykietowanych dla osiągnięcia porównywalnych wyników. Wszystkie te spostrzeżenia są istotne dla dalszych badań nad wykorzystaniem metod SSL w przypadku różnych, spotykanych w praktyce, scenariuszy.

Dodatek

Specyfikacja serwera

Symulacje zostały przeprowadzone na serwerze, którego konfiguracja sprzętowa przedstawiono poniżej. Należy zwrócić uwagę na dostępną wielkość pamięci RAM, która znacząco przewyższa ilości występujące w większości domowych stacji roboczych. Wpłynęło to znacznie na skrócenie czasu przeprowadzania eksperymentów.

- **Procesor** Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
  - Liczba rdzeni: 28 rdzeni (14 rdzeni na gniazdo)
  - Liczba wątków: 56 (2 wątki na rdzeń)
  - Częstotliwość taktowania: 3300 MHz (maksymalna), 1200 MHz (minimalna)
- **Rozmiar pamięci RAM** 256 GB

Wykorzystane biblioteki

Podczas przeprowadzania symulacji wykorzystano następujące wersje bibliotek dla środowiska Python:

- keras, wersja `2.11.0`;
- LAMDA-SSL, wersja `1.0.2`;
- matplotlib, wersja `3.5.3`;
- numpy, wersja `1.21.6`;
- pandas, wersja `1.3.5`;
- scikit-learn, wersja `1.0.2`;
- scipy, wersja `1.7.3`.

Skrypty umożliwiające przeprowadzenie symulacji

Specjalnie na potrzeby niniejszej pracy opracowano zestaw skryptów służących do automatyzacji procesu trenowania algorytmów uczenia częściowo nadzorowanego.

Poniżej przedstawiono pełny kod (patrz Listing 2.1) jednego ze skryptów, który ilustruje wykorzystanie różnych funkcji i bibliotek niezbędnych do przeprowadzenia symulacji.

Wykorzystuje on dynamiczną generację kombinacji hiperparametrów. Dzięki temu możliwa była łatwa konfiguracja odpowiednich parametrów oraz przeprowadzenie kompleksowych eksperymentów bez konieczności modyfikacji kodu. Skrypt automatycznie przetwarza dane, trenuje model z wykorzystaniem wybranych parametrów, a następnie ocenia wyniki. Wszystkie wyniki symulacji są zapisywane bezpośrednio do pliku, co ułatwia późniejszą analizę i porównanie różnych modeli.

```
from LAMDA_SSL.Algorithm.Classification.Assemble import Assemble
from LAMDA_SSL.Algorithm.Classification.LapSVM import LapSVM
from data_preprocessing import preprocess_cifar10, preprocess_imdb,
   preprocess_breast_cancer
from model_training import train_and_evaluate
from utils import split_labeled_unlabeled, save_results, svc_factory
import itertools

def generate_hyperparameter_combinations(hyperparams):
    """
    Generate all possible combinations of hyperparameters.
    :param hyperparams: A dictionary where the key is the hyperparameter
                        name and the value is a list of possible values.
    :return: A generator that yields dictionaries of
             hyperparameter combinations.
    """
    keys = hyperparams.keys()
    values = (hyperparams[key] if isinstance(hyperparams[key], list)
              else [hyperparams[key]] for key in keys)
    for combination in itertools.product(*values):
        yield dict(zip(keys, combination))

def train_model(model_name, model_function, dataset_name,
                preprocess_function, labeled_size,
                models_hyperparameters, folder_to_save_results):
    """
    Train a model with specified hyperparameters on a given dataset
    and save the results.
    :param model_name: Name of the model to train.
    :param model_function: The function that creates an instance
                           of the model.
    :param dataset_name: Name of the dataset to use.
    :param preprocess_function: Function used to preprocess the data.
    :param labeled_size: The size of the labeled data.
    :param models_hyperparameters: A dictionary of model names to their
                                   respective hyperparameter grids.
    :param folder_to_save_results: The directory to save the training
                                   results.
    """
    print(f"Alghoritm: {model_name}")
    print(f"Dataset: {dataset_name}")
    X_train, y_train, X_test, y_test = preprocess_function()
    X_labeled, y_labeled, X_unlabeled, y_unlabeled =
   split_labeled_unlabeled(X_train, y_train, labeled_size)
    for hyperparams in generate_hyperparameter_combinations(
                        models_hyperparameters[model_name]):
        print(f"Hyperparameters: {hyperparams}")
        try:
            model_params = {k: v() if callable(v)
                            else v for k, v in hyperparams.items()}
            model = model_function(**model_params)
```

```
        except Exception as e:
            print(f"Error during model initialization for {model_name}
                    with params {hyperparams}: {e}")
            continue
        try:
            train_results, transductive_results, test_results =  \
            train_and_evaluate(model_name, hyperparams,
                        dataset_name, model,
                        X_labeled, y_labeled, X_unlabeled, y_unlabeled,
                        X_test, y_test)
        except Exception as e:
            print(f"Error during training/evaluation for {model_name}
                    with params {hyperparams}: {e}")
            continue
        try:
            save_results(train_results, transductive_results,
                         test_results, model_name, dataset_name,
                         model_params, labeled_size,
                         folder_to_save_results)
        except Exception as e:
            print(f"Error during saving results for {model_name}
                    with params {hyperparams}: {e}")
            continue
        print(f"{dataset_name} Results:", test_results)

if __name__ == "__main__":
    datasets = {
                'breast_cancer': preprocess_breast_cancer,
                'cifar10' : preprocess_cifar10,
                'imdb' : preprocess_imdb
               }
    models = {
            'Assemble' : Assemble,
            'LapSVM' : LapSVM
            }
    models_hyperparameters = {
                'Assemble' : {
                    'base_estimator' : svc_factory,
                    'T' : [3, 10, 50, 100, 150, 300]
                }
                ,'LapSVM' : {
                    'distance_function' : ['rbf', 'linear', 'knn'],
                    'kernel_function' : ['rbf', 'linear']
                }
                }
    labeled_size = 0.3
    folder_to_save_results = \
    "experiments_results_all_parameters_Assemble_LapSVM"
    for model_name, model_function in models.items():
        for dataset_name, preprocess_function in datasets.items():
            train_model(model_name, model_function, dataset_name,
                        preprocess_function, labeled_size,
                        models_hyperparameters, folder_to_save_results)
```

Listing 2.1: Skrypt do automatycznego trenowania i ewaluacji modeli uczenia częściowo nadzorowanego z zastosowaniem różnych kombinacji hiperparametrów

Źródło: opracowanie własne

Bibliografia

[1] Bennett, K. P., Demiriz, A., Maclin, R. Exploiting unlabeled data in ensemble methods. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), KDD '02, Association for Computing Machinery, p. 289–296.

[2] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters 27*, 8 (2006), 861–874. ROC Analysis in Pattern Recognition.

[3] Jia, L.-H., Guo, L.-Z., Zhou, Z., Li, Y.-F. Lamda-ssl. a powerful and easy-to-use toolkit for semi-supervised learning. `https://ygzwqzd.github.io/LAMDA-SSL/`, 2022. [Online; dostęp 11 stycznia 2024].

[4] Jia, L.-H., Guo, L.-Z., Zhou, Z., Li, Y.-F. Lamda-ssl: Semi-supervised learning in python. *arXiv preprint arXiv:2208.04610* (2022).

[5] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images, 2009.

[6] Krzyśko, M., Wołyński, W., Górecki, T., Skorzybut, M. *Systemy uczące się. Rozpoznawanie wzorców, analiza skupień i redukcja wymiarowości*. Wydawnictwa Naukowo-Techniczne, Warszawa, 2008.

[7] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Portland, Oregon, USA, June 2011), Association for Computational Linguistics, pp. 142–150.

[8] Mitchell, T. M. *Machine learning*. McGraw-hill, 1997.

[9] Rajaraman, A., Ullman, J. D. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[10] Ruder, S., Plank, B. Strong baselines for neural semi-supervised learning under domain shift, 2018.

[11] Stefanowski, J. Uczenie się częściowo nadzorowane. `https://www.cs.put.poznan.pl/jstefanowski/pse/semi-supervised.pdf`. [Online; dostęp 10 stycznia 2024].

[12] van Engelen, J. E., Hoos, H. H. A survey on semi-supervised learning. *Machine Learning 109*, 2 (Feb. 2020), 373–440.

[13] Wolberg, W., Mangasarian, O., Street, N., Street, W. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

[14] Zhu, X. Semi-supervised learning literature survey. Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[15] Ćwik, J., Koronacki, J. *Statystyczne systemy uczące się. Wydanie drugie.* Akademicka Oficyna Wydawnicza EXIT Andrzej Lang, 2015.