[UMA] – Dokumentacja końcowa projektu

Uczenie aktywne (temat nr 1)

Michał Szwejk Damian D'Souza

1 Streszczenie założeń dokumnetacji wstępnej

Głównym zadaniem projektu było przygotowanie programu realizującego schemat uczenia aktywnego. Projekt zakładał opracowanie dwóch modeli uczenia maszynowego – wybrano *Las Losowy* i *SVM* – a następnie porównanie ich działania w środowisku uczenia aktywnego.

W założeniach wstępnych wyszczególniono także proces przygotowania danych do treningu. Określono, że testowany zbiór danych zostanie tak spreparowany, aby stał się on mocno niezrównoważony – klasa mniejszościowa powinna występować kilknaście razy mniej niż większościowa. Pondto wyróżniono, że ze zbioru danych zostanie usunięte większość etykiet w celu odpowiedniego przygotowania do procesu uczenia aktywnego.

Jako główną miarę ewaluacji jakości modelu wykorzystano pole pod krzywą *precision-recall* (PR AUC), ponieważ skupia się ona na jakości predykcji klasy pozytywnej, a nie na licznych przykładach klasy większościowej.

2 Uczenie aktywne

2.1 Opis procesu

Uczenie aktywne to metoda, która pozwala algorytmom uczenia maszynowego osiągać lepsze wyniki w przypadkach gdy zbiór danych jest mocno niezbalansowany i zawiera wiele brakujących etykiet. Model sam, bez ingerencji użytkownika, wybiera przykłady uczące, które według niego są najbardziej wartościowe dla treningu. Następnie zewnętrzne źródło (zazwyczaje ekspert domenowy) manualnie etykietuje wyznaczone próbki. Takie podejście pozwala minimializować potrzebę posiadania dużego zbioru danych nie poświęcając przy tym skuteczności modelu.

Uczenie aktywne jest procesem iteracyjnym. Pseudokod metody może być przedstawiony następująco:

Algorithm 1 Uczenie aktywne

Wybierz początkowy zbiór przykładów trenujących T z całego zbioru danych P;
$P := P - T$;
Poetykietuj przykłady ze zbioru T;
repeat
    Przygotuj model h ucząc się na przykładach ze zbioru T;
    Wybierz nowe przykłady Q ze zbioru P bazując na wcześniej przygotowanym modelu h;
    Poetykietuj dane ze zbioru Q;
    $T := T \cup Q$;
    $P := P \setminus Q$;
until Poetykietowano 100% danych lub spełniono zadane kryterium stopu;
return h.

Głownym ograniczeniem uczenia aktywnego jest długotrwały proces etykietowania danych, który wymaga możliwie jak najepszej "wyroczni". Dodatkowo proces może być kosztowny, ponieważ w każdej iteracji trzeba od nowa dopasowywać model do danych.

2.2 Inicjalizacja zbioru etykietowanego

Na początku uczenia aktywnego wyznaczany jest początkowy zbiór etykietowany, który stanowi bazę dla pierwszej iteracji modelu. Wariant *random* realizuje losowy dobór próbek, natomiast inicjalizacja *cluster* ukierunkowuje wybór na różne regiony przestrzeni cech, tak aby zapewnić lepszą reprezentatywność zbioru startowego. Taki dobór zwiększa prawdopodobieństwo uwzględnienia różnych klas i przyspiesza stabilizację wczesnych estymacji modelu.

2.3 Sposoby wybierania próbek

W uczeniu aktywnym należy zdefiniować kryterium, według którego model będzie wybierał próbki przeznaczone do anotacji. Przygotowane zostały trzy warianty:

- **uncertainty sampling** – największa niepewność predykcji:

$$\max_{x \in P} \left(1 - \max_{\hat{y} \in C} P(\hat{y}|x)\right)$$

- **diversity sampling** – największa różnorodność:

$$\max_{x \in P} \left(\min_{x' \in T} \delta(x, x')\right)$$

  gdzie $\delta(x, x')$ – miara podobieństwa między dwoma próbkami (np. euklidesowa).

- **random** – losowo, zgodnie z rozkładem jednostajnym.

3 Opis algorytmów

3.1 Random forest

Las losowy jest algorytmem, który generuje wiele różnych drzew w procesie uczenia i przypisuje klasy, będące dominantą wyników poszczególnych drzew. W odróżnieniu od pojedynczego drzewa tendencja modelu do nadmiernego dopasowania jest dużo mniejsza. Każde drzewo trenowane jest na tylu przykładach ile jest w zbiorze trenującym, jednakże losowane są one ze zwracaniem. Ponadto ograniczony jest zestaw atrybutów, wybierane jest ich wyłącznie $\sqrt{|A|}$ (bez zwracania), gdzie A – to zbiór atrybutów.

Każde drzewo wchodzące w skład lasu losowego jest drzewem typu *CART*. Reguły podziału w poszczególnych węzłach są konstruowane w taki sposób, aby po podziale minimalizować wartość **zanieczyszczenia Gini'ego** (w przypadku zadania klasyfikacji). Miara ta jest dana wzorem:

$$Gini(x) = \sum_{i=0}^{|C|} 1 - p_i^2$$

gdzie p_i – prawdopodobieństwo i-tej klasy w węźle, a C – zbiór klas.

3.2 SVM

Maszyna wektorów nośnych jest algorytmem klasyfikacji, który wyznacza optymalną hiperpłaszczyznę rozdzielającą klasy w przestrzeni cech. Dla problemu binarnego hiperpłaszczyzna jest definiowana przez wektor wag $\mathbf{w}$ i wyraz wolny b, a klasyfikacja nowej próbki $\mathbf{x}$ odbywa się na podstawie znaku funkcji decyzyjnej $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$.

Kluczową ideą SVM jest maksymalizacja marginesu, czyli odległości między hiperpłaszczyzną a najbliższymi punktami każdej z klas (tzw. wektorami nośnymi). W przypadku danych nieliniowo rozdzielnych wprowadza się parametr kary C, który kontroluje kompromis między maksymalizacją marginesu a minimalizacją błędu klasyfikacji. W implementacji zastosowano standaryzację cech na podstawie średniej i odchylenia standardowego wyznaczanego na zbiorze treningowym, a parametry te są następnie używane podczas predykcji. Uczenie wag realizowane jest metodą Pegasos (SGD) z losowym tasowaniem próbek oraz krokiem uczenia malejącym według $\eta_t = \frac{\eta}{1+\eta\lambda t}$, gdzie $\lambda = \frac{1}{C \cdot n}$, minimalizując funkcję celu zawierającą **funkcję straty zawiasowej**:

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

gdzie $y_i \in \{-1, 1\}$ – etykieta i-tej próbki, a C – parametr kary.

W sytuacjach wymagających probabilistycznego wyjścia modelu (co jest kluczowe np. w uczeniu aktywnym), możliwe jest zastosowanie metody **Platta**. Pozwala ona na dopasowanie

funkcji sigmoidalnej do wartości funkcji decyzyjnej SVM, co umożliwia przekształcenie ich w oszacowania prawdopodobieństwa:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(A \cdot f(\mathbf{x}) + B)}$$

gdzie parametry A i B wyznaczane są poprzez minimalizację ujemnej logarytmicznej funkcji wiarygodności na zbiorze treningowym.

4 Zbiory danych

4.1 Pomocniczy

W ramach prostych testów sprawdzających poprawność implementacji naszych algorytmów wykorzystano odniesienie do wartości zwracanej przez klasyfikatory opracowane w bilbliotece *SKLEARN*. Do testów wykorzystano zbiór danych dotyczący jakości win. Zadanie sprowadzono do klasyfikacji binarnej – jakość mniejsza niż 6 wskazuje na klasę 0, a większa na 1. Zbiór zawiera 1599 przykładów i 11 atrybutów, z czego wszystkie są ciągłe.

4.2 Główny

Jako główny zbiór danych wykorzystywany w pętli uczenia aktywnego wybrano zestaw zawierający zdjęcia kwiatów. Pierwotnie podjęto taką decyzję, ponieważ założono, że przykłady będą etykietowane manualnie co jest łatwiejsze do zrealizowania patrząc na zdjęcie, a nie wektor cech. Jak się później okazało, takie podejście jest niemożliwe do zrealizowania ze względów ograniczeń czasowych jeżeli chcemy przeprowadzić dokładne eksperymenty bazujące na n-krotnych powtórzeniach walidacji krzyżowej. Przed przystąpieniem do przeprowadzenia eksperymentów został zbiór został przetworzony w następujący sposób:

- Pozostawienie wyłącznie próbek zklasyfikowanych jako mniszki lekarskie lub słoneczniki – ograniczenie się do klasyfikacji binarnej;
- Usunięcie prawie wszystkich próbek słoneczników (ostateczny stosunek klas to 1:20) – stworzenie niezbalansowanego zbioru danych;
- Zamaskowanie etykiet 75% przykładów (stają się one całkowicie niewidoczne dla modelu) – symulacja sytuacji braku etykiet;
- Przetworzenie zdjęcia do postaci reprezentacji wektorowej z wykorzystaniem wcześniej wytrenowanej sieci neuronowej *ResNet50*.

Ostatecznie zbiór danych składa się z 680 przykładów – 30 słoneczników (klasa 1) i 650 mniszków lekarskich (klasa 0). Każdy przykład ma dokładnie 2048 atrybutów (wszystkie ciągłe).

5 Eksperymenty

5.1 Porównanie z *SKLEARN*

Tak jak wspomniano w sekcji dotyczącej opisu zbiorów danych przeprowadzono testy porównawcze z biblioteką *SKLEARN* w celu odniesienia uzyskanych wyników do wartości referencyjnych. Przeprowadzonych zostało 10 niezależnych uruchomień, każde z innym ziarnem startowym wykorzystywanym do dzielenia zbiorów danych na trenujący i testujący (80% – 20%) oraz jako parametr algorytmu uczącego. Następnie zagregowano wyniki. Parametry modeli zostały ustawione jednakowo, zgodnie z domyślnymi w *SKLEARN*. Do porównania wykorzystano standardowe miary oceny jakości klasyfikatora.

5.1.1 Las losowy

algorytm	średnia	std	max	min
forest	0.777	0.009	0.790	0.761
SKLEARN	0.800	0.004	0.807	0.790

Tabela 1: Porównanie miary **accuracy**.

algorytm	średnia	std	max	min
forest	0.840	0.006	0.848	0.828
SKLEARN	0.847	0.003	0.852	0.837

Tabela 2: Porównanie miary **F1-Score**.

algorytm	średnia	std	max	min
forest	0.925	0.009	0.935	0.905
SKLEARN	0.874	0.009	0.884	0.855

Tabela 3: Porównanie miary **recall**.

algorytm	średnia	std	max	min
forest	0.770	0.007	0.783	0.757
SKLEARN	0.822	0.005	0.834	0.814

Tabela 4: Porównanie miary **precision**.

Wyniki są porównywalne, a obserwowane różnice wynikają najpewniej z odmiennych sposobów implementacji algorytmów. Otrzymane wartości są na tyle zbliżone, że można uznać nasze implementacje za poprawne.

5.1.2 SVM

algorytm	średnia	std	max	min
svm	0.713	0.013	0.731	0.690
SKLEARN	0.740	0.008	0.754	0.728

Tabela 5: Porównanie miary **accuracy**.

algorytm	średnia	std	max	min
svm	0.775	0.012	0.788	0.752
SKLEARN	0.805	0.006	0.815	0.794

Tabela 6: Porównanie miary **F1-Score**.

algorytm	średnia	std	max	min
svm	0.783	0.035	0.839	0.710
SKLEARN	0.846	0.009	0.856	0.826

Tabela 7: Porównanie miary **recall**.

algorytm	średnia	std	max	min
svm	0.769	0.022	0.799	0.719
SKLEARN	0.767	0.006	0.780	0.759

Tabela 8: Porównanie miary **precision**.

W porównaniu SVM widoczny jest spadek *accuracy*, *F1* i *recall* względem implementacji referencyjnej, przy zachowaniu podobnej *precision*. Różnice te wynikają z odmiennego sposobu uczenia (Pegasos) jednak kierunek wyników pozostaje spójny z wersją referencyjną.

5.2 Uczenie aktywne

5.2.1 Założenia

Podczas przeprowadzania eksperymentów dotczyczących uczenia aktywnego poczynione zostały założenia:

- Próbki nie są ręcznie etykietowane. Informacja o obecności etykiety jest kodowana za pomocą flagi binarnej (0 – brak etykiety, 1 – etykieta obecna). Po każdym wywołaniu selektora, który zwraca próbki do poetykietowania, flaga binarna dla danego przykładu jest zmieniana z 0 na 1. Na podstwie ustalonych bitów model decyduje, który przykład może zostać wykorzystany w zbiorze treningowym;
- Do ewaluacji wyników wykorzystano n-krotną waldiację krzyżową (k-fold), z parametrami $n = 5$ i $k = 5$. Pondato upewniono się, że w każdym zbiorze testowym znajduje się co najmniej jeden przykład z klasy mniejszościowej;
- Jako miarę oceny jakości wykorzystano pole pod krzywą *precsion-recall* (PR-AUC), ponieważ skupia się ona na jakości predykcji klasy pozytywnej, a nie licznych przykładach klasy większościowej;
- Każdy selektor, który wskazuje dane do poetykietowania, zwraca je w grupach liczących dokładnie 10 próbek;
- Wszystkie elementy losowe (np. generowanie drzew w lesie losowym) są przeprowadzane z odpowiednio ustawionymi ziarnami dzięki czemu wyniki są reprodukowalne.

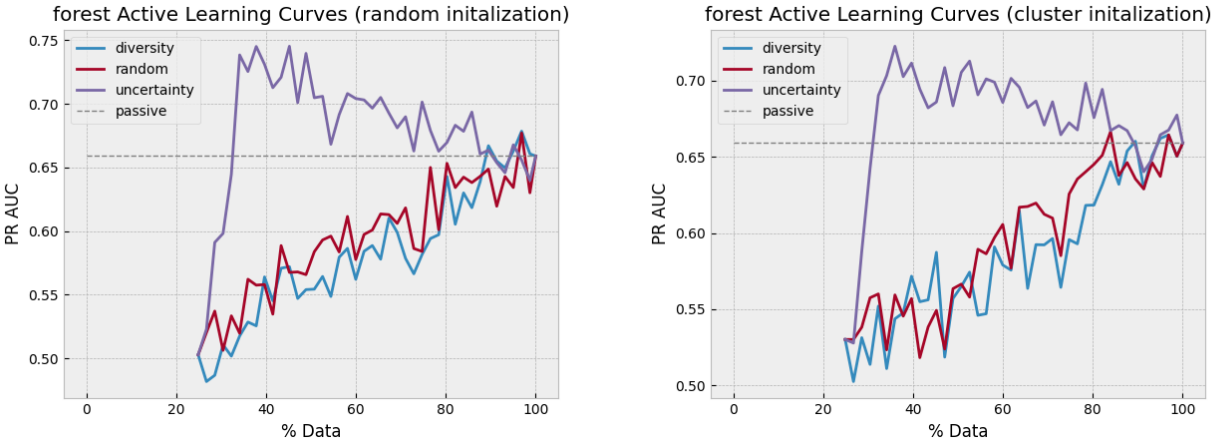
5.2.2 Zagregowane wyniki

wariant	25%	30%	50%	75%	100%
RF-clus-div	0.530	0.514	0.565	0.596	0.659
RF-clus-rand	0.530	0.557	0.566	0.626	0.659
RF-clus-unc	0.530	0.642	0.705	0.672	0.659
RF-rand-div	0.503	0.511	0.554	0.582	0.659
RF-rand-rand	0.503	0.506	0.584	0.584	0.659
RF-rand-unc	0.503	0.598	0.705	0.702	0.659
SVM-clus-div	0.602	0.682	0.709	0.815	0.823
SVM-clus-rand	0.602	0.690	0.725	0.807	0.823
SVM-clus-unc	0.602	0.813	0.825	0.843	0.823
SVM-rand-div	0.651	0.709	0.796	0.813	0.823
SVM-rand-rand	0.651	0.732	0.764	0.801	0.823
SVM-rand-unc	0.651	0.796	0.846	0.841	0.823

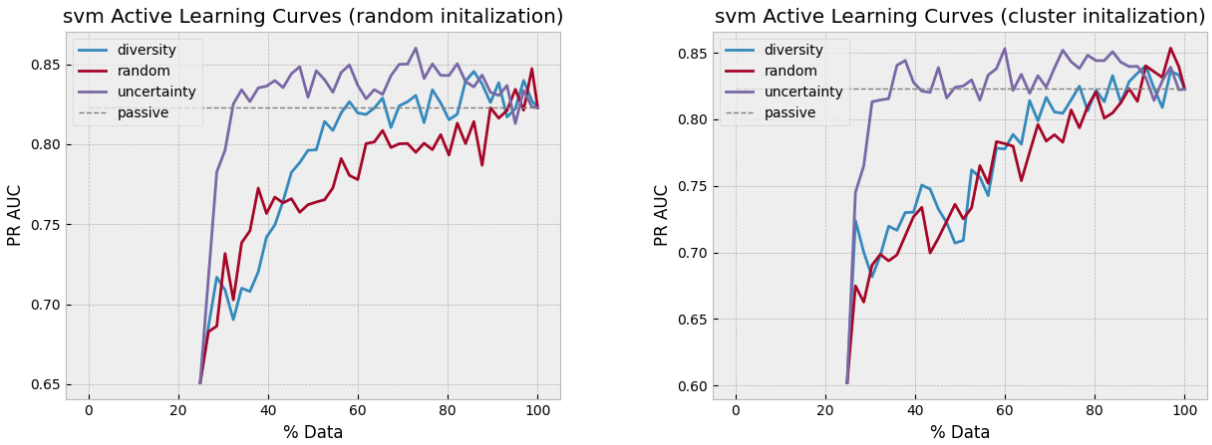
Tabela 9: Uśrednione wartości PR-AUC na różnych etapach uczenia.

Wyniki pokazują, że warianty SVM uzyskują wyższe wartości PR-AUC już przy 25% poetykietowanych danych, a ich przewaga utrzymuje się również przy 50% i 75% etykiet. Najszybszy przyrost jakości w obu modelach zapewnia selekcja *uncertainty*, co widać szczególnie dla inicjalizacji klastrowej, gdzie SVM przekracza 0.8 PR-AUC już przy 30%. Dla lasu losowego poprawa jest bardziej umiarkowana i zależna od strategii: losowe próbkowanie rośnie najwolniej, a strategie *uncertainty* i *diversity* wyraźnie przyspieszają naukę w środkowej fazie (50%–75%).

5.2.3 Wykresy zbieżności



Rysunek 1: Krzywe uczenia w zależności od % poetykietowanych danych dla lasu losowego.

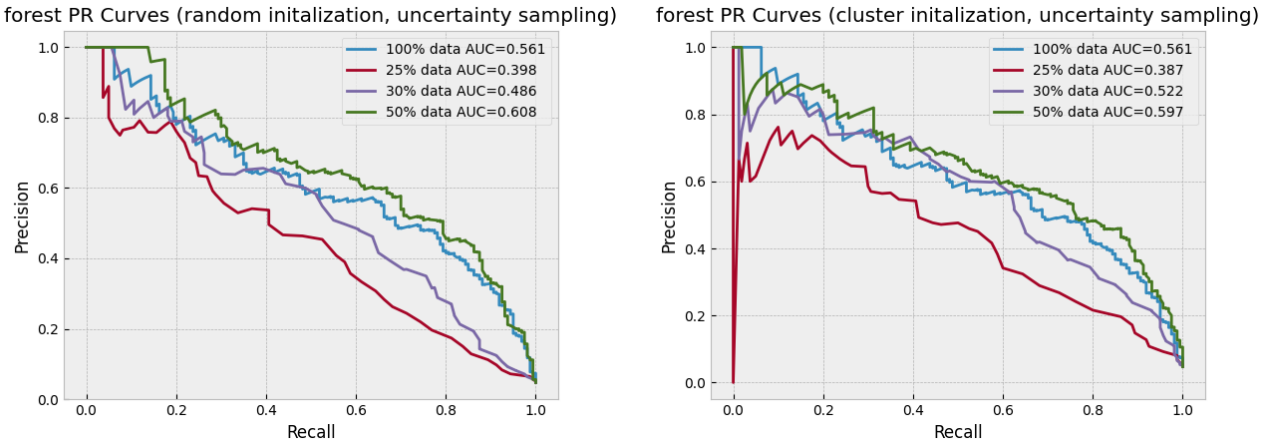


Rysunek 2: Krzywe uczenia w zależności od % poetykietowanych danych dla SVM.

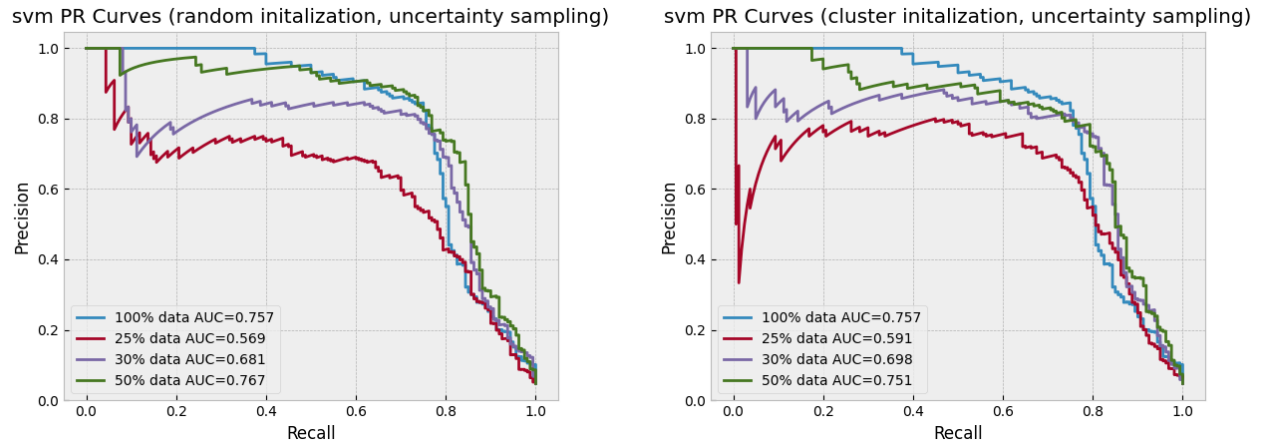
Wnioski:

- Wybieranie próbek, co do których model jest najbardziej niepewny daje najlepsze wyniki, ponieważ taka selekcja w pełni wykorzystuje informacje zawarte w zbiorze danych niepoetykietowanych. Zarówno dla lasu losowego jak i SVM model już po paru iteracjach procesu uczenia aktywnego osiąga wyniki powyżej progu wyznaczonego przez model nauczony na 100% danych;
- Strategie *diversity* i *random* nie osiągają zadowalających wyników w ramach uczenia aktywnego – prawie najwyższą wartość miary jakości uzyskują dopiero przy 100% poetykietowanych danych. Oznacza to, że proces etykietowania nie może zostać przerwany na wcześniejszym etapie bez istotnej utraty jakości modelu;
- Strategia *diversity* radzi sobie słabo, ponieważ w przypadku tak dużej liczby atrybutów (2048) musimy mierzyć się z problemem wymiarowości. W tak dużej przestrzeni zanikają różnice niepodobieństwa przez co próbki wybierane są prawie że losowo;
- Selektor losowy (*random*) w żaden sposób nie wykorzystuje informacji o modelu lub zbiorze niepoetykietowanych danych przez co większość nowych przykładów nie wnosi istotnej informacji i w żaden sposób nie poprawia jakości predykcji.

5.2.4 Przykładowe krzywe *precision-recall*



Rysunek 3: Krzywe *precision-recall* dla lasu losowego.



Rysunek 4: Krzywe *precision-recall* dla SVM.

Wnioski:

- Klastrowa inicjalizacja powoduje niestabilność na początkowych etapach uczenia, które objawiają się jako szpilki na wykresie.
- Nie zawsze przy 100% poetykietowanych danych model zwraca największe wartości miary jakości (PR AUC). W przypadku strategii *uncertainty* model jest już "nasycony" przy 50% przetworzonych daynch, dalsze trenowanie nie zwiększa jakości.
- Wszystkie wykresy są stosunkowo gładkie – nia ma żadnego wyraźnego załamania, po którym precyzja malałaby drastycznie. Oznacza to, że proces uczenia przebiega stabilnie.

6 Opis funkcjonalny

System składa się z dwóch głównych programów uruchamianych z linii poleceń: `src.main` realizującego pętlę uczenia aktywnego oraz `src.process_images` przygotowującego dane wejściowe na podstawie zdjęć. Uruchomienie odbywa się przez polecenie `python3 -m src.<moduł>`, gdzie nazwa modułu podawana jest bez rozszerzenia. Przykładowe polecenia:

- `python3 -m src.process_images`
- `python3 -m src.main`
- `python3 -m src.main -c configs/svm_cluster_random`

Dla `src.main` można wskazać własny plik konfiguracyjny przy pomocy flagi `-c` (lub `-config`). Domyślnie ładowany jest plik `config.toml`, a dla przetwarzania obrazów `image_processing_config.toml`. Jeżeli wskazany plik nie istnieje, system tworzy go automatycznie na podstawie zarejestrowanych klas konfiguracyjnych.

Konfiguracja przetwarzania obrazów

Konfiguracja przetwarzania danych steruje pobieraniem zbioru, ekstrakcją cech oraz przygotowaniem pliku wejściowego do uczenia aktywnego. W pliku znajdują się m.in.:

- `data_processing.output_dir` – katalog zapisu danych (np. `data/active_learning`);
- `data_processing.unlabeled_percentage` – odsetek przykładów z ukrytymi etykietami;
- `data_processing.majority_ratio` – docelowy stosunek klasy większościowej do mniejszościowej;
- `image_processing.model` – nazwa sieci CNN używanej do ekstrakcji cech (np. `resnet50`);
- `image_processing.data_dir` – katalog źródłowy ze zdjęciami;
- `image_processing.force_download` – wymuszenie ponownego pobrania danych.

Przykładowa konfiguracja:

```
[data_processing]
output_dir = "data/active_learning"
unlabeled_percentage = 75
majority_ratio = 20

[image_processing]
```

```
model = "resnet50"
data_dir = "data/flowers/images"
force_download = false
```

Konfiguracja uczenia aktywnego

Konfiguracja uczenia aktywnego definiuje model klasyfikatora, strategię doboru próbek oraz parametry walidacji. Dodatkowo dostępny jest moduł `python3 -m src.compare_with_sklearn <classifier>`, który uruchamia porównanie z implementacją *scikit-learn*. Moduł przyjmuje argument `classifier` (`svm` lub `forest`) oraz korzysta z ustawień z sekcji `aggregator` w pliku konfiguracyjnym. Najważniejsze sekcje:

- `cluster_initializer` – parametry inicjalizacji klastrami (np. liczba klastrów i proporcje);
- `svm` oraz `forest` – parametry modeli SVM i lasu losowego;
- `active_learner` – wybór klasyfikatora, selektora (`random`, `diversity`, `uncertainty`) i wielkości partii;
- `tester` – liczba podziałów, powtórzeń, progi i katalog zapisu wyników;

Przykładowa konfiguracja:

```
[cluster_initializer]
clusters = 2
center_ratio = 0.6
border_ratio = 0.4

[forest]
n_trees = 100
multiprocessing = false

[forest.tree_config]
max_depth = 10
min_samples_split = 2

[svm]
learning_rate = 0.1
penalty = 100
iter_count = 1000

[active_learner]
```

```
classifier = "forest"
selector = "uncertainty"
batch_size = 10
seed = 42
should_store_results = true

[tester]
save_dir = "results"
n_splits = 5
n_repeats = 3
labeled_ratio = 0.25
seed = 42
thresholds = [0.25, 0.3, 0.4, 0.5, 0.75, 1.0]
initializer = "cluster"

[aggregator]
runs = 10
metrics = ["accuracy", "precision", "recall", "f1"]
output = "results/compare"
```

7 Opis wykorzystanych narzędzi

W implementacji wykorzystano narzędzia i biblioteki wspierające zarówno obróbkę obrazów, jak i eksperymenty z uczeniem aktywnym. Projekt został napisany w języku Python 3.13, a środowisko uruchomieniowe może być skonfigurowane w ramach `venv` z instalacją zależności przez `pip`.

Najważniejsze użyte technologie i biblioteki:

- `numpy` i `pandas` – przetwarzanie danych i przygotowanie macierzy cech;
- `scikit-learn` – metryki jakości, podziały danych oraz implementacje referencyjne do porównań;
- `torch` i `torchvision` – ekstrakcja cech z obrazów przy użyciu sieci *ResNet50*;
- `kagglehub` – pobieranie zbioru zdjęć kwiatów;
- `ucimlrepo` – pobieranie zbioru jakości win do testów porównawczych;
- `matplotlib` – wsparcie dla wizualizacji wyników.