Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Informatyki

Projekt dyplomowy

Biblioteka Datasets dla Elixira

Datasets library for Elixir

Autorzy: Radosław Rolka, Weronika Wojtas
Kierunek studiów: Informatyka
Opiekun pracy: dr inż. Aleksander Smywiński-Pohl

Kraków, 2025

Spis treści

Rozdział 1

Cel prac i wizja produktu

Celem pracy jest stworzenie biblioteki w języku Elixir, która umożliwi łatwe pobieranie, przetwarzanie i zarządzanie zbiorami danych, które są powszechnie wykorzystane w uczeniu maszynowym. Biblioteka powinna oferować szeroki wybór gotowych zbiorów danych, a także możliwość dodawania własnych.

1.1. Opis dziedziny problemu

Praca z modelami uczenia maszynowego jest ściśle związana z wykorzystaniem danych, które stanowią fundament dla procesów trenowania i walidacji algorytmów. Dostęp do dobrze przygotowanych i różnorodnych zbiorów danych jest kluczowy dla efektywnego rozwoju i nauki modeli. Zbiory danych muszą być nie tylko obszerne i reprezentatywne, ale również odpowiednio przetworzone i znormalizowane, co często stanowi wyzwanie ze względu na duży nakład czasu i zasobów wymaganych w procesie przygotowania. Nieodłącznym elementem pracy z danymi jest ich czyszczenie, skalowanie, oraz odpowiednie formatowanie, które umożliwia integrację danych wejściowych z modelami. Ponadto, same dane często pochodzą z różnorodnych źródeł i są zapisane w różnych formatach, co dodatkowo komplikuje ich użyteczność bezpośrednio po pozyskaniu.

1.2. Motywacja

Podczas studiów zainteresowaliśmy się językami programowania funkcyjnego, szczególnie Elixirem. Choć na początku jego podejście może wydawać się nietypowe, szybko dostrzegliśmy, jak prosty i elegancki jest ten język. Podczas pracy z Elixirem zauważyliśmy, że brakuje w nim biblioteki do zarządzania zbiorami danych, która w innych językach jest szeroko stosowana, szczególnie w sztucznej inteligencji.

Postanowiliśmy, że to będzie temat naszej pracy inżynierskiej. Chcemy stworzyć bibliotekę w Elixirze, inspirowaną Hugging Face Datasets [4], która pozwoli programistom łatwiej pracować ze zbiorami danych i ułatwi dostęp do nich.

1.3. Rola produktu

Głównym celem biblioteki jest uproszczenie i automatyzacja zarządzania, przetwarzania oraz optymalizacja zbiorów danych. Umożliwienie łatwego dostępu do różnorodnych zbiorów danych pozwoli na szybsze rozpoczęcie pracy nad projektami, eliminując konieczność manualnego zbierania i konfigurowania danych. Integracja funkcji automatycznego czyszczenia, normalizacji, skalowania i augmentacji danych znacząco zredukuje czasochłonne procesy przygotowywania danych, co jest zazwyczaj barierą w szybkim prototypowaniu i testowaniu modeli uczenia maszynowego. Użytkownikami końcowymi projektowanej biblioteki są przede wszystkim analitycy, specjaliści od uczenia maszynowego oraz studenci zajmujący się analizą danych i sztuczną inteligencją, którym to narzędzie ma za zadanie zwiększyć produktywność poprzez automatyzację rutynowych zadań.

1.4. Obszary funkcjonalne

1. **Pobieranie i zarządzanie zbiorami danych**

    - Pobieranie gotowych zbiorów danych z różnych źródeł: Implementacja mechanizmów umożliwiających pobieranie danych z platform takich jak Hugging Face Hub [1], Kaggle [3] czy innych repozytoriów.
    - Dodawanie własnych zbiorów danych: Możliwość integracji i zarządzania własnymi zestawami danych w systemie.

2. **Przeglądanie zbiorów danych**

    - Łatwe przeglądanie dostępnych zbiorów danych: Interfejsy umożliwiające szybki podgląd i analizę dostępnych danych.
    - Filtrowanie zbiorów danych: Narzędzia do selekcji danych na podstawie określonych kryteriów.

3. **Przetwarzanie i transformacja danych**

    - Czyszczenie danych: Funkcje do usuwania błędów i niekompletnych rekordów.
    - Normalizacja danych: Metody standaryzacji wartości w zbiorach danych.
    - Tokenizacja: Proces dzielenia tekstu na mniejsze jednostki, takie jak słowa czy zdania.
    - Tworzenie podzbiorów danych: Możliwość dzielenia większych zbiorów na mniejsze, bardziej zarządzalne części.

4. **Przykłady rozwiązań**

    - Przykłady użycia: Praktyczne scenariusze i case studies demonstrujące zastosowanie poszczególnych funkcji.

1.5. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne odgrywają kluczową rolę w zapewnieniu, że stworzona biblioteka nie tylko spełni swoje zadania funkcjonalne, ale również będzie przyjazna dla użytkownika. Poniżej przedstawiono główne wymagania niefunkcjonalne dla projektu:

- Wydajność - Biblioteka powinna efektywnie zarządzać i przetwarzać duże zbiory danych z minimalnym opóźnieniem.
- Kompatybilność - Interfejs powinien być kompatybilny z różnymi systemami operacyjnymi i integrować się z istniejącymi popularnymi narzędziami i bibliotekami w ekosystemie Elixira.
- Dokumentacja - Kompletna i zrozumiała dokumentacja techniczna jest niezbędna, by użytkownicy mogli efektywnie wykorzystywać wszystkie funkcje biblioteki.

1.6. Przegląd dostępnych rozwiązań

Jednym z głównych narzędzi w tej dziedzinie rozwiązań jest biblioteka datasets od Hugging Face [4], która jest szeroko stosowana w społeczności uczenia maszynowego. Biblioteka datasets oferuje łatwy dostęp do szerokiej gamy zbiorów danych w różnych językach programowania. Oferuje ona również różnorodne narzędzia do przetwarzania i transformacji danych.

W kontekście Elixira, który nadal jest dynamicznie rozwijającym się językiem, nie istnieje jeszcze takie narzędzie, które w pełni odpowiadałoby potrzebom użytkowników w zakresie zarządzania i przetwarzania danych dla uczenia maszynowego, co tworzy przestrzeń na rynku dla nowego rozwiązania, które może lepiej odpowiadać na unikalne potrzeby społeczności Elixira, zwiększając efektywność ich pracy dzięki specjalizowanym narzędziom dostosowanym do ich środowiska i metod pracy.

1.7. Analiza technologiczna

Stos technologiczny został zaprojektowany z myślą o maksymalnym wykorzystaniu możliwości języka Elixir oraz jego ekosystemu. Do obliczeń numerycznych oraz operacji na tensorach wykorzystamy bibliotekę NX [5]. Biblioteka zapewnia wydajność w przeprowadzaniu operacji matematycznych, szczególnie w kontekście obliczeń związanych z dużymi zbiorami danych i sztuczną inteligencją. NX oferuje wsparcie dla operacji na tensorach, które są kluczowe w procesach uczenia maszynowego oraz analizy danych.

Zintegrowany z NX jest także Explorer [2], który będziemy wykorzystywać w naszej pracy do efektywnego zarządzania i analizy danych. Explorer to biblioteka, która umożliwia pracę z dwoma głównymi typami struktur danych: seriami, oraz dataframe'ami. Te struktury pozwalają na wygodne i szybkie eksplorowanie danych, co jest szczególnie istotne podczas analizy informacji. Explorer, jako backend, korzysta z Polars, biblioteki napisanej w języku Rust co przekłada się na znaczną poprawę wydajności w obliczeniach z dużymi zbiorami.

Do współpracy z modelami głębokiego uczenia maszynowego w naszym projekcie zastosujemy bibliotekę Bumblebee [8], która pozwala na łatwą integrację z pretrenowanymi modelami sieci neuronowych. Bumblebee umożliwia dostęp do popularnych modeli, które zostały udostępnione przez platformy sztucznej inteligencji, takie jak Hugging Face Transformers [9]. Ta

biblioteka umożliwi łatwą implementację i wykorzystanie zaawansowanych modeli AI w naszej pracy, co pozwoli na efektywne wdrożenie algorytmów uczenia maszynowego i głębokiego uczenia w środowisku Elixira.

1.8. Analiza ryzyka

W procesie projektowania i rozwijania nowej biblioteki istnieje wiele potencjalnych ryzyk, których zidentyfikowanie pozwala na lepsze przygotowanie, co z kolei zwiększa szanse na pomyślne zakończenie projektu. Są to między innymi:

- Adaptacja przez społeczność - Jako że Elixir jest stosunkowo mniej popularny niż inne języki wykorzystywane w dziedzinie uczenia maszynowego, takie jak Python, istnieje ryzyko, że biblioteka nie zyska szerokiego grona użytkowników. Promocja biblioteki i demonstrowanie jej wartości w rzeczywistych projektach będzie kluczowe.
- Integracja z istniejącymi narzędziami - Problemy z integracją nowej biblioteki z już istniejącymi ekosystemami i narzędziami, których niekompatybilność może powstrzymać potencjalnych użytkowników przed korzystaniem z biblioteki.
- Obsługą dużych zbiorów danych - Możliwe, że biblioteka nie będzie w stanie efektywnie procesować dużych zbiorów danych lub że wystąpią problemy z wydajnością.
- Niedostateczne testowanie - Niewystarczające testowanie w różnych środowiskach i scenariuszach użytkowania może prowadzić do niezauważonych błędów, które ujawnią się dopiero po wdrożeniu biblioteki.

1.9. Podsumowanie

Projekt ma na celu stworzenie biblioteki w języku Elixir, która będzie odpowiadała funkcjonalności biblioteki Hugging Face Datasets [4], umożliwiając łatwe pobieranie, przetwarzanie i zarządzanie zbiorami danych używanymi w uczeniu maszynowym. Biblioteka ta oferować będzie funkcje takie jak pobieranie gotowych zbiorów danych z różnych źródeł, możliwość dodawania własnych zbiorów danych, filtrowanie i przeglądanie dostępnych zbiorów, a także przetwarzanie danych (czyszczenie, normalizacja, tokenizacja). Użytkownicy będą mogli tworzyć podzbiory danych oraz integrować bibliotekę z innymi narzędziami w Elixirze, takimi jak Nx [5], Explorer [2] i Bumblebee [8]. Projekt zakłada również dostarczenie pełnej dokumentacji oraz przykładów użycia.

Rozdział 2

Zakres funkcjonalności

Celem niniejszego rozdziału jest przedstawienie specyfikacji funkcjonalnej projektowanej biblioteki. Specyfikacja została opracowana na podstawie analizy potrzeb użytkowników i rozmów konsultacyjnych z zainteresowanymi stronami.

2.1. Charakterystyka użytkownika

System zakłada istnienie jednego głównego rodzaju użytkownika – **programisty pracującego z językiem Elixir**, który zajmuje się tworzeniem, trenowaniem lub walidacją modeli uczenia maszynowego. Użytkownicy ci mogą być częścią większych zespołów badawczo-rozwojowych lub niezależnymi deweloperami.

Zakłada się, że użytkownik:

- posiada podstawową lub zaawansowaną znajomość języka Elixir,
- zna podstawy uczenia maszynowego oraz pracy z danymi,
- wymaga efektywnego i prostego dostępu do przygotowanych zbiorów danych,
- oczekuje narzędzi ułatwiających wstępne przetwarzanie danych, takich jak czyszczenie, normalizacja, tokenizacja.

2.2. Charakterystyka systemów współpracujących

Tworzona biblioteka do zarządzania zbiorami danych w języku Elixir zakłada ścisłą współpracę z wybranym zestawem zewnętrznych narzędzi i bibliotek, które pełnią kluczową rolę w zapewnieniu pełnej funkcjonalności systemu. Integracja tych komponentów pozwala na maksymalne wykorzystanie potencjału języka Elixir w kontekście przetwarzania danych na potrzeby uczenia maszynowego. Poniżej przedstawiono charakterystykę najważniejszych współpracujących systemów:

- **Źródła zbiorów danych** – Biblioteka umożliwia pobieranie popularnych zbiorów danych wykorzystywanych w uczeniu maszynowym z różnych publicznych repozytoriów, takich jak Hugging Face Datasets [4]. W związku z tym wymagana jest integracja z usługami HTTP i obsługą różnorodnych formatów danych.

- **Nx** [5] – Biblioteka opiera się na integracji z narzędziem Nx, które zapewnia funkcje numeryczne i przetwarzanie tensorów. Współpraca z Nx pozwala na bezproblemowe przygotowanie danych do trenowania modeli uczenia maszynowego w Elixirze.
- **Explorer** [2] – Do eksploracji, filtrowania i transformacji danych tabularycznych wykorzystywana jest biblioteka Explorer , która umożliwia przetwarzanie danych w sposób zbliżony do narzędzi takich jak Pandas [7] w Pythonie. Dzięki temu użytkownik może łatwo analizować i przygotowywać dane w ramach jednego ekosystemu.
- **Bumblebee** [8] – W celu dalszego wykorzystania przetworzonych danych, możliwa jest integracja z biblioteką Bumblebee, która dostarcza gotowe modele i narzędzia do pracy z NLP i uczeniem głębokim.
- **Lokalna pamięć masowa** – System wspiera lokalne przechowywanie danych oraz cache'owanie zbiorów, aby zminimalizować potrzebę wielokrotnego pobierania tych samych danych i zwiększyć wydajność pracy z dużymi zestawami.

2.3. Wymagania funkcjonalne

W poniższym rozdziale szczegółowo opisano wymagania funkcjonalne dla projektowanej biblioteki w języku Elixir, klasyfikując je zgodnie z metodologią MoSCoW [6] ukierunkowaną na najbardziej krytyczne aspekty funkcjonalności systemu.

Must Have

- Możliwość pobierania datasetów z zewnętrznych źródeł – biblioteka musi umożliwić użytkownikowi łatwy dostęp do zasobów danych oferowanych przez różne repozytoria.
- Integracja z narzędziami Elixir takimi jak Nx i Explorer – niezbędne jest zapewnienie kompatybilności i efektywnej współpracy narzędziowej.
- Pełna dokumentacja funkcji biblioteki – użytkownicy muszą mieć dostęp do jasnych i zrozumiałych instrukcji korzystania z biblioteki.

Should Have

- Możliwość dodawania własnych datasetów do biblioteki – funkcja ta pozwala użytkownikom na personalizację i rozszerzenie bazy danych.
- Narzędzia do czyszczenia i normalizacji danych – choć nie krytyczne, znacząco podnoszą wartość użytkową biblioteki.

Could Have

- Rozbudowane funkcje tokenizacji danych – ułatwiłyby przetwarzanie tekstu, zwiększając potencjalne obszary zastosowań biblioteki.
- Wtyczki wspierające nowsze frameworki i biblioteki w ekosystemie Elixir – mogą zwiększyć atrakcyjność biblioteki dla szerokiej grupy użytkowników.

Won't Have

- Automatyczne tłumaczenia dokumentacji na różne języki – choć przydatne w przyszłości, nie będą dostępne w pierwszej wersji produktu.
- Zaawansowane algorytmy sztucznej inteligencji do analizy danych – nie są planowane w aktualnym zakresie projektu.

2.4. Wymagania jakościowe

Poniżej przedstawione zostały wymagania niefunkcjonalne, które mają na celu zapewnienie wysokiej jakości tworzonej biblioteki oraz komfortu jej użytkowania. Odpowiednie spełnienie tych wymagań wpłynie pozytywnie na łatwość integracji, rozwój i utrzymanie projektu w dłuższym okresie.

- **Czytelny i spójny interfejs API** – Interfejs biblioteki powinien być intuicyjny i dobrze zaprojektowany, umożliwiając użytkownikowi szybkie rozpoczęcie pracy bez konieczności zapoznawania się z nadmiernie rozbudowaną dokumentacją. Nazewnictwo funkcji, struktur i modułów powinno być spójne i zgodne z konwencjami języka Elixir.
- **Wydajność** – Biblioteka powinna umożliwiać efektywne operacje na dużych zbiorach danych, takich jak filtrowanie, czyszczenie czy transformacja. Szczególny nacisk powinien zostać położony na optymalizację operacji na dużych zbiorach dancyh oraz minimalizację zużycia pamięci i czasu przetwarzania.
- **Skalowalność** – Projekt powinien być skalowalny zarówno pod względem wielkości obsługiwanych danych. Powinien umożliwiać bezproblemowe dodawanie nowych źródeł danych, funkcjonalności i formatów danych bez konieczności istotnej przebudowy istniejącej architektury.
- **Bezpieczeństwo danych** – W przypadku integracji z zewnętrznymi źródłami danych, komunikacja powinna być realizowana z użyciem bezpiecznych protokołów (np. HTTPS). System powinien być odporny na wstrzykiwanie niepoprawnych danych oraz błędne formaty plików.
- **Odporność na błędy** – Biblioteka powinna zawierać mechanizmy wykrywania i obsługi błędów, umożliwiające użytkownikowi uzyskanie jasnych komunikatów w przypadku problemów z danymi, połączeniem lub działaniem funkcji.
- **Kompatybilność z ekosystemem Elixira** – Projekt musi zapewniać pełną kompatybilność z innymi bibliotekami wykorzystywanymi w uczeniu maszynowym w języku Elixir, w szczególności Nx [5], Explorer [2] i Bumblebee [8]. Powinien też działać na różnych platformach systemowych wspierających środowisko Elixira.
- **Dokumentacja** – Biblioteka powinna być opatrzona pełną dokumentacją techniczną, zawierającą opisy funkcji, struktur danych, przykłady użycia oraz wskazówki dotyczące integracji z innymi narzędziami. Dokumentacja powinna być dostępna zarówno w kodzie (np. jako modułowe `@doc`), jak i w formie zewnętrznej (np. README, przewodniki).
- **Łatwość w utrzymaniu i rozwoju** – Kod źródłowy powinien być przejrzysty, modularny, zgodny z dobrymi praktykami programistycznymi i łatwy do testowania oraz rozszerzania. Projekt powinien uwzględniać przyszłą rozbudowę, np. o nowe formaty danych, dodatkowe transformacje lub integracje.

- **Testowalność** – System powinien być w pełni testowalny. Moduły powinny być projektowane w sposób umożliwiający tworzenie testów jednostkowych oraz testów integracyjnych, co ułatwi utrzymanie wysokiej jakości kodu i szybką detekcję błędów w przyszłości.

2.5. Scenariusze użytkowania

Poniżej przedstawiono przykładowe scenariusze użytkowania systemu, które ilustrują typowe sytuacje, w jakich programista może korzystać z biblioteki. Scenariusze te odzwierciedlają główne funkcjonalności systemu i obrazują sposób interakcji użytkownika z interfejsem programistycznym (API) biblioteki.

- **Pobranie gotowego zbioru danych**

  Użytkownik chce szybko pobrać popularny zbiór danych w celu przetestowania modelu klasyfikacji tekstu. W tym celu korzysta z funkcji udostępnianych przez bibliotekę, które automatycznie pobierają dane z repozytorium, zapisują je lokalnie i przygotowują do dalszego przetwarzania.

- **Dodanie własnego zbioru danych**

  Użytkownik posiada własny zbiór danych zapisany w formacie CSV. Chce go załadować, wstępnie przefiltrować oraz znormalizować. Korzystając z biblioteki, użytkownik definiuje strukturę zbioru danych, wskazuje lokalizację pliku, a następnie stosuje dostępne funkcje do oczyszczenia danych i konwersji ich do odpowiedniego formatu.

- **Filtrowanie danych na podstawie kryteriów**

  Użytkownik analizuje zbiór danych zawierający opinie klientów i chce utworzyć podzbiór, który zawiera wyłącznie opinie pozytywne. Biblioteka umożliwia zastosowanie funkcji filtrowania na podstawie warunków logicznych, co pozwala szybko uzyskać interesujący użytkownika fragment danych.

- **Integracja danych z modelem uczenia maszynowego**

  Po przygotowaniu danych, użytkownik chce przesłać je jako tensory do modelu zaimplementowanego w bibliotece Bumblebee [8]. Biblioteka wspiera konwersję danych do struktury kompatybilnej z Nx[5] oraz umożliwia bezpośrednie przekazanie ich do dalszego przetwarzania przez model.

- **Przegląd dostępnych zbiorów danych**

  Użytkownik nie jest jeszcze zdecydowany, z jakim zbiorem chce pracować. Korzysta z funkcji przeglądania dostępnych zbiorów, które zawierają metadane, takie jak: źródło, liczba rekordów, typ danych (tekst, obraz, liczby), wymagania wstępne itp. Po zapoznaniu się z informacjami, wybiera odpowiedni zbiór i rozpoczyna pracę.

2.6. Podsumowanie

Ten rozdział dostarcza wszechstronnej analizy projektowanego systemu, obejmującej szczegółowy opis funkcji z uwzględnieniem priorytetów, opis procesów biznesowych oraz scenariuszy użytkowania, co pozwala lepiej zrozumieć i wizualizować funkcjonalności projektu. Ta

całościowa prezentacja ułatwia zarządzanie oczekiwaniami i planowanie dalszych etapów rozwoju systemu, uwzględniając ustalony zakres prac i zasoby, co jest kluczowe dla skutecznego planowania przyszłych etapów wdrożenia.

Rozdział 3

Wybrane aspekty realizacji

Przyjęte założenia, struktura i zasada działania systemu, wykorzystane rozwiązania technologiczne wraz z uzasadnieniem ich wyboru, istotne mechanizmy i zastosowane algorytmy.

3.1. Wzory matematyczne

Przykład wzoru z odnośnikiem do literatury [**author2021title**]:

$$\Omega = \sum_{i=1}^{n} \gamma_i \tag{3.1}$$

Przykładowy odnośnik do wzoru (3.1).
Przykładowy wzór w tekście $\lambda = \sum_{i=1}^{n} \delta_i$, bez numeracji.

3.2. Algorytmy

Algorytm 1 przedstawia przykładowy algorytm zaprezentowany w [**fiorio2017algorithm2e**].

3.3. Fragmenty kodu źródłowego

Listing 3.1 przedstawia przykładowy fragment kodu źródłowego.

Algorytm 1: Przykładowy algorytm (źródło: [**fiorio2017algorithm2e**]).

input : A bitmap Im of size $w \times l$
output A partition of the bitmap
:

```
special treatment of the first line;
for i ← 2 to l do
    special treatment of the first element of line i;
    for j ← 2 to w do
        left ← FindCompress(Im[i, j − 1]);
        up ← FindCompress(Im[i − 1, ]);
        this ← FindCompress(Im[i, j]);
        if left compatible with this then // O(left,this)==1
            if left < this then Union(left,this);
            else Union(this,left);
        end
        if up compatible with this then                    // O(up,this)==1
            if up < this then Union(up,this);
            // this is put under up to keep tree as flat as possible
            else Union(this,up);
            // this linked to up
        end
    end
    foreach element e of the line i do FindCompress(p);
end
```

```python
# The maximum of two numbers

def maximum(x, y):

    if x >= y:
        return x
    else:
        return y

x = 2
y = 6
print(maximum(x, y),"is the largest of the numbers ", x, " and ", y)
```

Listing 3.1: Przykładowy fragment kodu (źródło: [**author2021title**])

Rozdział 4

Organizacja pracy

Struktura zespołu (role poszczególnych osób), krótki opis i uzasadnienie przyjętej metodyki i/lub kolejności prac, planowane i zrealizowane etapy prac ze wskazaniem udziału poszczególnych członków zespołu, wykorzystane praktyki i narzędzia w zarządzaniu projektem.

Rozdział 5

Wyniki projektu

Wskazanie wyników projektu (co konkretnie udało się uzyskać: oprogramowanie, dokumentacja, raporty z testów/wdrożenia, itd.), prezentacja wyników i ocena ich użyteczności (jak zostało to zweryfikowane — np. wnioski klienta/użytkownika, zrealizowane testy wydajnościowe, itd.), istniejące ograniczenia i propozycje dalszych prac.

Bibliografia

[1] H. Face. *Hugging Face Hub*. 2025. URL: https://huggingface.co/.

[2] C. Grainger i J. Valim. *Explorer: Fast and Elegant Data Exploration in Elixir*. 2025. URL: https://github.com/elixir-explorer/explorer.

[3] Kaggle. *Kaggle: Your Machine Learning and Data Science Community*. 2025. URL: https://www.kaggle.com.

[4] Q. Lhoest i in. „Datasets: A Community Library for Natural Language Processing". W: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online i Punta Cana, Dominican Republic: Association for Computational Linguistics, list. 2021, s. 175–184. arXiv: 2109.02846 [cs.CL]. URL: https://aclanthology.org/2021.emnlp-demo.21.

[5] S. Moriarity, J. Valim i P. O. L. Valente. *Nx - Numerical Elixir*. 2022. URL: https://github.com/elixir-nx/nx.

[6] P. Podgórni. *Metoda MoSCoW – co to jest?* 2024. URL: https://www.itvision.pl/experts/model-moscow-czym-jest-jak-korzystac/.

[7] T. pandas development team. *pandas-dev/pandas: Pandas*. Wer. latest. Lut. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[8] J. Valim i contributors. *Bumblebee: Pre-trained Neural Network Models in Elixir*. 2025. URL: https://github.com/elixir-nx/bumblebee.

[9] T. Wolf i in. „Transformers: State-of-the-Art Natural Language Processing". W: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, paź. 2020, s. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Spis rysunków

Spis tabel

Spis algorytmów

Spis listingów