Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa magisterska

na kierunku Informatyka i Systemy Informacyjne

w specjalności Metody sztucznej inteligencji

Analiza CV z zastosowaniem uczenia maszynowego w celu optymalizacji procesów rekrutacyjnych

Wojciech Klusek

Numer albumu 305943

promotor

dr inż. Marcin Luckner

WARSZAWA 2024

Streszczenie

Analiza CV z zastosowaniem uczenia maszynowego w celu optymalizacji procesów rekrutacyjnych

Celem niniejszej pracy magisterskiej było opracowanie i implementacja systemu umożliwiającego automatyczny przegląd kandydatów aplikujących na konkretne stanowisko. System wykorzystuje nowoczesne technologie przetwarzania języka naturalnego oraz uczenia maszynowego, wspierając procesy rekrutacyjne poprzez automatyzację analizy dokumentów aplikacyjnych.

W ramach pracy przeprowadzono ekstrakcję danych z dokumentów aplikacyjnych z wykorzystaniem pretrenowanego modelu NER, co umożliwiło wydobycie imion i nazwisk kandydatów. Zastosowano również pretrenowany model BART w celu generowania podsumowań treści CV, co pozwoliło na szybkie i czytelne przedstawienie najważniejszych informacji o kandydacie, takich jak doświadczenie zawodowe czy umiejętności. Ponadto, zaimplementowano moduł klasyfikacji CV oparty na modelu XGBoost, który został przetrenowany na odpowiednio przygotowanych danych, zawierających CV oraz kategorię zawodową, osiągając dokładność (ang. accuracy) klasyfikacji na poziomie 79% przy 24 klasach. Aby uszeregować kandydatów pod kątem dopasowania do wymagań ofert pracy, opracowano autorską sieć neuronową opartą na architekturze LSTM. Model został przetrenowany na zbiorze danych zawierającym CV, oferty pracy oraz przypisania do trzech klas: good fit, no fit i potential fit. Model osiągnął dokładność na poziomie 83%.

Uzyskane wyniki wskazują, że zaprojektowany system skutecznie wspiera procesy rekrutacyjne, przyspieszając analizę aplikacji i poprawiając trafność selekcji kandydatów, co czyni go wartościowym narzędziem w nowoczesnych procesach zarządzania zasobami ludzkimi.

Słowa kluczowe: NLP, LSTM, XGBoost, BERT, NER, analiza CV, uczenie maszynowe, automatyzacja rekrutacji.

Abstract

Resume analysis using machine learning to optimize recruitment process

The purpose of this thesis was to develop and implement a system that enables the automatic review of candidates applying for specific job openings. The system uses modern natural language processing and machine learning technologies to support recruitment processes by automating the analysis of application documents.

The work involved extracting data from application documents using a pretrained model with Named Entity Recognition (NER) techniques to identify candidates names. A pretrained BART model was also used to generate summaries of resume content, allowing for quick and clear representation of key candidate information, such as work experience or skills. In addition, a resume classification module based on the XGBoost model was implemented, which was pretrained on properly prepared data containing resumes and occupational category, achieving a classification accuracy of 79% with 24 classes. A custom neural network based on the LSTM architecture was developed to rank candidates in terms of matching job requirements. The model was trained on a dataset containing resumes, job offers and assignments to three classes: good Fit, no fit and potential fit. The model achieved an accuracy of 83%.

The results indicate that the designed system effectively supports recruitment processes, speeding up the analysis of applications and improving the accuracy of candidate selection, making it a valuable tool in modern human resource management processes.
Keywords: NLP, LSTM, XGBoost, BERT, NER, resume analysis, machine learning, recruitment automation.

Spis treści

1. Wstęp

Współczesny rynek pracy wymaga od działów zasobów ludzkich (HR), które odpowiadają za zarządzanie kapitałem ludzkim i wspieranie strategii rozwoju firmy, nie tylko znajomości najnowszych trendów, ale także skuteczności w szybkim i precyzyjnym dobieraniu najlepszych kandydatów do określonych stanowisk pracy. Automatyzacja procesów rekrutacyjnych, wykorzystująca zaawansowane technologie i algorytmy przetwarzania danych, staje się kluczowym elementem usprawniającym i przyspieszającym selekcję aplikantów na wstępnych etapach rekrutacji. Szczególnie istotne staje się to w sytuacji, gdy liczba kandydatów jest duża, a błędy ludzkie mogą skutkować pominięciem obiecujących aplikantów przez rekruterów.

Przetwarzanie języka naturalnego (NLP) i technologie uczenia maszynowego odgrywają coraz większą rolę w automatyzacji i optymalizacji procesów rekrutacyjnych. Analiza CV za pomocą tych technologii umożliwia nie tylko szybsze przetwarzanie dużych wolumenów dokumentów, które w ramach jednej organizacji mogą sięgać nawet dziesiątek tysięcy aplikacji rocznie, ale także bardziej precyzyjne dopasowanie kandydatów do specyficznych wymagań ofert pracy. W kontekście globalizacji rynku pracy i rosnącej konkurencji o najlepsze talenty, wykorzystanie zaawansowanych narzędzi analitycznych może stanowić kluczową przewagę strategiczną dla organizacji. Wdrożenie przetwarzania języka naturalnego i uczenia maszynowego do procesów rekrutacyjnych niesie ze sobą szereg korzyści wykraczających daleko poza oszczędność czasu. Te technologie umożliwiają zespołom HR:

- **Obniżenie kosztów rekrutacji:** Automatyzacja analizy CV znacząco redukuje koszty związane z ręcznym przeglądaniem aplikacji, pozwalając zespołom HR na bardziej efektywne wykorzystanie zasobów i skupienie się na strategicznych aspektach procesu rekrutacji.
- **Wczesne identyfikowanie najlepszych kandydatów:** Analiza CV pozwala algorytmom NLP i ML szybko identyfikować kandydatów posiadających najbardziej istotne umiejętności i doświadczenie, umożliwiając rekruterom skupienie się na najbardziej obiecujących aplikacjach.

- **Uzyskanie cennych informacji:** Analiza zagregowanych danych rekrutacyjnych pozwala zespołom HR zyskać cenne informacje o trendach wśród kandydatów, demografii siły roboczej i efektywności strategii rekrutacyjnych.

1.1. Cel i zakres pracy

Celem niniejszej pracy magisterskiej jest opracowanie i implementacja systemu, który umożliwi automatyczny przegląd kandydatów aplikujących na konkretne oferty pracy. System ten składa się z komponentów: ekstrakcji danych, podsumowywania, klasyfikacji oraz rankingu.

Praca rozpoczyna się od przeglądu literatury dotyczącej automatyzacji procesu rekrutacji oraz zastosowania technologii NLP i uczenia maszynowego w analizie dokumentów aplikacyjnych, co zostało omówione w rozdziale 2.1. Następnie szczegółowo przedstawiono wybrane techniki NLP (2.2) oraz uczenia maszynowego (2.3), które znalazły zastosowanie w tej pracy. W rozdziale 3.1 opisano funkcjonalności zaimplementowanego systemu, w tym mo6duł ekstrakcji danych, który umożliwia wydobywanie takich informacji jak imię, nazwisko, numer telefonu czy adres e-mail (3.2). Kolejnym istotnym elementem jest moduł podsumowania CV, generujący zwięzłe streszczenia kluczowych informacji zawartych w dokumentach aplikacyjnych. Jego celem jest ułatwienie rekruterom szybkiego dostępu do najważniejszych danych o kandydacie, takich jak umiejętności, doświadczenie zawodowe i osiągnięcia (3.3). Rozdział 3.4 poświęcono modułowi klasyfikacji CV, który automatycznie przypisuje dokumenty do predefiniowanych kategorii zawodowych na podstawie analizy treści za pomocą algorytmów uczenia maszynowego. Dodatkowo, w rozdziale 3.5 przedstawiono algorytm rankingu, oparty na technikach przetwarzania języka naturalnego, takich jak tokenizacja i wektoryzacja tekstu. Algorytm ten umożliwia uporządkowanie aplikacji kandydatów według kryteriów określonych w ofertach pracy, wspomagając ocenę ich dopasowania. Funkcjonalność systemu została zweryfikowana w testach przeprowadzonych na zestawach danych zawierających różnorodne CV oraz oferty pracy, a wyniki tych testów poddano szczegółowej analizie.

Praca demonstruje, w jaki sposób zaawansowane technologie mogą wspierać procesy rekrutacyjne, dostosowując je do współczesnych wymagań rynku pracy i podnosząc ich efektywność oraz precyzję.

2. Metody automatyzacji procesów rekrutacyjnych

2.1. Przegląd literatury

Ostatnie dekady przyniosły rewolucję w sposobie, w jaki firmy przeprowadzają procesy rekrutacyjne, głównie za sprawą rozwoju technologii internetowej. Ewolucja ta otworzyła drzwi do nowych możliwości, które znacząco zmieniły proces rekrutacji. W niniejszym rozdziale przedstawiony zostanie przegląd prac naukowych związanych z automatyzacją procesów rekrutacyjnych, które dokładniej analizują wykorzystanie nowoczesnych technologii w rekrutacji oraz ich wpływ na efektywność i skuteczność tego procesu.

2.1.1. Przetwarzanie języka naturalnego w rekrutacji

W pracy *Ranking résumés automatically using only résumés: A method free of job offers* [4] zaprezentowane zostały dwie metody oceny CV, które nie wymagają dostępu do ofert pracy ani innych zasobów semantycznych. Opierając się na koncepcji Inter-Résumé Proximity, metody te skupiają się na podobieństwie leksykalnym między CV wysłanymi przez kandydatów w odpowiedzi na tę samą ofertę pracy. Dodatkowo, proponowane jest wykorzystanie informacji zwrotnej dotyczącej istotności, zarówno na poziomie ogólnym, jak i leksykalnym, w celu poprawy oceny CV.

W opracowaniu *An automated resume screening system using natural language processing and similarity* [7] proponowany jest system automatycznej selekcji CV oparty na przetwarzaniu języka naturalnego w celu przezwyciężenia ograniczeń związanych z ręczną selekcją aplikacji. System ten automatycznie identyfikuje kandydatów najbardziej pasujących do oferowanego stanowiska, analizując informacje zawarte w ich CV, takie jak doświadczenie, wykształcenie i umiejętności. Dzięki wykorzystaniu NLP, system wyodrębnia kluczowe informacje z CV, tworząc ich syntetyczne reprezentacje. Następnie, wykorzystując techniki takie jak wektoryzacja i podobieństwo cosinusowe, system ocenia dopasowanie każdego kandydata do wymagań stanowiska opisanych w ofercie pracy. Podejście to pozwala na szybszą i bardziej efektywną selekcję kandydatów, a także na zmniejszenie ryzyka pominięcia wartościowych aplikacji.

W pracy *An efficient algorithm for ranking candidates in e-recruitment system* [21] przedstawiono algorytm rankingowy ukierunkowany na rozwiązanie problemu dużej czasochłonności i wysokich kosztów związanych z oceną kandydatów w rekrutacji elektronicznej. Proponowany algorytm rankingowy stanowi alternatywę dla standardowych metod opartych na wielokryteriowej analizie decyzyjnej (MCDM), które charakteryzują się wysoką złożonością obliczeniową. Algorytm ten integruje w sobie sortowanie macierzowe oraz techniki ograniczania przetwarzanych danych, co pozwala na zwiększenie jego skalowalności. Skuteczność rozwiązania została przetestowana na trzech różnych zestawach danych: rzeczywistych danych rekrutacyjnych, symulowanych danych (DBLP) oraz zestawach syntetycznych. Symulowane dane bazują na rzeczywistych danych z innych dziedzin, takich jak publikacje naukowe, podczas gdy dane syntetyczne są w pełni generowane sztucznie. Uzyskane obiecujące wyniki potwierdzają efektywność algorytmu w procesie rankingowania CV w realnym świecie.

2.1.2. Uczenie maszynowe w rekrutacji

W pracy *Resume Screening using NLP and LSTM* [3], autorzy przedstawiają system do automatycznego przeglądu CV, który wykorzystuje techniki NLP oraz rekurencyjne sieci neuronowe Long Short Term Memory (LSTM). Celem jest klasyfikacja CV na podstawie umiejętności kandydata, dopasowując je do odpowiednich ofert pracy. System analizuje dane z dokumentów aplikacyjnych i grupuje umiejętności w szersze kategorie, co poprawia trafność wyników.

Kolejna praca: *A machine learning-based human resources recruitment system for business process management: using LSA, BERT and SVM* [16] przedstawia system rekrutacyjny oparty na technikach uczenia maszynowego, takich jak analiza latentnej semantyki (LSA), BERT oraz maszyny wektorów nośnych (SVM). Autorzy badają, jak różne podejścia do wektoryzacji tekstu i próbkowania wpływają na efektywność przetwarzania danych z CV. Wyniki badania pokazują, że metody LSA i BERT skutecznie identyfikują kluczowe informacje z CV, a SVM optymalizuje wydajność modelu predykcyjnego.

W pracy *Resume screening and recommendation system using machine learning approaches* [25], autorzy proponują system rekomendacji kandydatów oparty na NLP i uczeniu maszynowym z wykorzystaniem regresji logistycznej. System analizuje CV pod kątem formatowania oraz treści, a następnie ocenia ich zgodność z wymaganiami stanowiska. Dodatkowo, system sugeruje alternatywne oferty pracy, które mogą pasować do kandydata na podstawie jego umiejętności i doświadczenia. Prace te pokazują, że uczenie maszynowe w rekrutacji ma szerokie zastosowanie - od analizy i oceny CV, przez prognozowanie dopasowania kandydata, aż po adaptacyjne systemy selekcji, które uczą się i dostosowują w czasie rzeczywistym. Dzięki takim rozwiązaniom

proces rekrutacyjny staje się bardziej efektywny, a ryzyko pomyłki w ocenie kandydatów jest zminimalizowane.

2.2. Wybrane techniki NLP

W tym rozdziale przedstawione zostaną wybrane techniki przetwarzania języka naturalnego, które zostały zastosowane w niniejszej pracy. Opisane metody stanowią kluczowe etapy przetwarzania tekstu, które umożliwiają analizę i automatyzację procesów związanych z rekrutacją. Skupimy się na technikach takich jak usuwanie słów nieinformatywnych, lematyzacja, wektoryzacja tekstu oraz ocena podobieństwa cosinusowego. Każda z tych metod odgrywa istotną rolę w poprawie efektywności i dokładności algorytmów stosowanych w analizie tekstów rekrutacyjnych.

2.2.1. Usuwanie słów nieinformatywnych

Usuwanie słów nieinformatywnych (ang. stop words) jest ważnym krokiem we wstępnym przetwarzaniu tekstu w ramach przetwarzania języka naturalnego. Słowa nieinformatywne to słowa, które często występują w języku, ale nie niosą istotnej wartości informacyjnej dla analizy tekstu. Są to najczęściej zaimki, spójniki, przyimki oraz inne powszechne słowa, które, choć niezbędne dla poprawnej składni językowej, nie przyczyniają się znacząco do zrozumienia kontekstu czy tematyki tekstu.

Słowa nieinformatywne są usuwane, aby zmniejszyć wielkość danych do analizy oraz skoncentrować się na słowach, które mają większe znaczenie semantyczne. Przykłady słów nieinformatywnych w języku angielskim obejmują takie słowa jak: "the", "is", "at", "which", "on". Usunięcie tych słów pozwala na zmniejszenie szumu w danych.

Poniżej przedstawiono przykładowe zdanie oraz wersję tego zdania po usunięciu słów nieinformatywnych:

Oryginalne zdanie:

"Experienced software engineer with a strong background in Python development and data analysis."

Zdanie po usunięciu słów nieinformatywnych:

"Experienced software engineer strong background Python development data analysis."

Jak widać, usunięcie słów nieinformatywnych nie zmienia znaczenia zdania, ale pozwala skupić się na najistotniejszych słowach.

Implementacja usuwania słów nieinformatywnych w analizie tekstu często polega na stworze-

niu listy tych słów, która jest następnie wykorzystywana do filtrowania tekstu. W popularnych bibliotekach NLP, takich jak NLTK, SpaCy czy scikit-learn, dostępne są wbudowane listy słów, które można łatwo zastosować.

W literaturze usuwanie słów nieinformatywnych jest szeroko uznawane jako efektywna metoda poprawy jakości przetwarzania tekstu. W książce *Introduction to Information Retrieval* [18] autorzy stwierdzają, że usuwanie tych słów pozwala na znaczące zmniejszenie wolumenu danych, które system musi przechowywać, co może prowadzić do zwiększenia efektywności przetwarzania zapytań. Autorzy podkreślają, że w większości nowoczesnych systemów przetwarzania informacji dodatkowy koszt związany z pominięciem słów nieinformatywnych jest stosunkowo niewielki.

2.2.2. Lematyzacja tekstu

Lematyzacja jest również ważnym etapem wstępnego przetwarzania tekstu w przetwarzaniu języka naturalnego. Proces ten polega na sprowadzaniu wyrazów do ich podstawowej formy, zwanej lematem. W przeciwieństwie do stemmingu (metody redukcji słów polegającej na obcinaniu końcówek wyrazów, co często prowadzi do powstania form niebędących rzeczywistymi słowami), lematyzacja wykorzystuje wiedzę o morfologii języka, aby przekształcić różne formy wyrazów w ich podstawową formę, co często pozwala zachować poprawność gramatyczną i znaczeniową.

Lematyzacja ma na celu zredukowanie złożoności językowej i ujednolicenie form wyrazów, co ułatwia analizę tekstu i poprawia wyniki algorytmów NLP. Jest szczególnie użyteczna w przypadkach, gdy różne formy tego samego wyrazu mogą być użyte w różnych kontekstach, ale mają to samo znaczenie semantyczne. Na przykład, słowa takie jak "running", "ran"i "runs"mogą być zredukowane do ich lematycznej formy "run".

Przykład lematyzacji:

- Oryginalne zdanie: "The candidates were applying for various engineering positions."
- Zdanie po lematyzacji: "The candidate be apply for various engineering position."

W powyższym przykładzie wyrazy "candidates", "were", "applying", "positions"zostały zredukowane odpowiednio do "candidate", "be", "apply", "position".

Implementacja lematyzacji wymaga użycia słowników morfologicznych i zasad gramatycznych języka. W popularnych bibliotekach NLP, takich jak NLTK, SpaCy, czy StanfordNLP, dostępne są wbudowane lematyzatory, które można łatwo zastosować.

Lematyzacja tekstu została szeroko omówiona w literaturze jako efektywna metoda poprawy wyników analiz NLP. W pracy *Stemming and Lemmatization: A Comparison of Retrieval Performances* [2] zaproponowano porównanie dokładności odzyskiwania dokumentów opartych na

stemmingu i lematyzacji. Stemming to procedura redukcji wszystkich słów z tym samym rdzeniem do wspólnej formy, podczas gdy lematyzacja usuwa końcówki fleksyjne i zwraca podstawową lub słownikową formę słowa. Porównania przeprowadzono również między tymi dwoma technikami a algorytmem bazowym (tj. bez przetwarzania językowego). Opracowano wyszukiwarkę i testowano algorytmy na zbiorze testowym. Wyniki wskazują, że stemming i lematyzacja przewyższają algorytm bazowy. Jeśli chodzi o techniki modelowania języka, lematyzacja uzyskała lepszą precyzję w porównaniu do stemmingu, jednak różnice są nieistotne. Ogólnie wyniki sugerują, że techniki modelowania języka poprawiają odzyskiwanie dokumentów, przy czym technika lematyzacji przynosi najlepsze rezultaty.

2.2.3. Wektoryzacja tekstu

Wektoryzacja tekstu to proces przekształcania danych tekstowych w formę numeryczną, która może być używana przez algorytmy uczenia maszynowego. Celem wektoryzacji jest reprezentowanie tekstu w taki sposób, aby algorytmy mogły analizować i przetwarzać go w sposób efektywny. Wektoryzacja jest niezbędnym krokiem w wielu aplikacjach przetwarzania języka naturalnego, takich jak klasyfikacja tekstu, analiza sentymentu czy rozpoznawanie nazw własnych.

Jedną z powszechnie stosowanych metod wektoryzacji tekstu jest tokenizacja, która polega na konwersji tekstu na sekwencje numeryczne. W przeciwieństwie do metod takich jak Word2Vec czy GloVe, które generują gęste wektory osadzeń słów (reprezentujące semantyczne znaczenie słów w przestrzeni wektorowej), tokenizacja koncentruje się na przekształceniu słów na unikalne identyfikatory numeryczne. Proces ten zazwyczaj obejmuje następujące kroki:

1. Na podstawie dostarczonego korpusu tekstowego, tworzy się słownik, który mapuje każde unikalne słowo na unikalny identyfikator numeryczny.

2. Tekst jest przekształcany w sekwencje liczb całkowitych, gdzie każda liczba odpowiada słowu w słowniku. Na przykład, zdanie "The quick brown fox"może zostać przekonwertowane na sekwencję liczb odpowiadających słowom w słowniku, np. [1, 2, 3, 4].

3. Proces tokenizacji może być dostosowywany poprzez różne parametry, takie jak limit liczby najczęściej występujących słów, minimalna liczba wystąpień słowa, czy traktowanie tekstu bez uwzględnienia wielkości liter.

Metoda ta jest często stosowana w połączeniu z innymi technikami wektoryzacji, takimi jak osadzenia słów (ang. embeddings), aby uzyskać bardziej złożone i semantycznie bogate reprezentacje tekstu. Tokenizacja jest prostą, ale skuteczną metodą wstępnej obróbki tekstu, która

pozwala na przygotowanie danych tekstowych do dalszego przetwarzania i analizy w modelach uczenia maszynowego.

W literaturze, wektoryzacja tekstu jest szeroko opisana, między innymi w pracach takich jak *Speech and Language Processing* autorstwa Daniela Jurafsky'ego i Jamesa H. Martina [15] oraz w artykule *Efficient Estimation of Word Representations in Vector Space* [20] autorstwa Tomáša Mikolova i innych. W książce Jurafsky'ego i Martina szczegółowo omawiane są różne techniki przekształcania tekstu na reprezentacje numeryczne, takie jak tokenizacja oraz osadzenia słów, natomiast praca Mikolova wprowadza model Word2Vec, który stał się fundamentem
dla wielu zaawansowanych metod wektoryzacji tekstu. Obie prace podkreślają znaczenie wektoryzacji w kontekście przetwarzania języka naturalnego, zwłaszcza w takich zadaniach jak klasyfikacja tekstu czy analiza sentymentu.

2.2.4. Podobieństwo cosinusowe

Podobieństwo cosinusowe to miara stosowana w analizie tekstu i informatyce do określenia stopnia podobieństwa między dwoma wektorami. Jest często używane w przetwarzaniu języka naturalnego do oceny podobieństwa między dokumentami lub fragmentami tekstu.
Metoda ta mierzy kąt między dwoma wektorami w przestrzeni wektorowej i jest definiowana jako cosinus kąta między nimi.

Podobieństwo cosinusowe między dwoma wektorami $\mathbf{A}$ i $\mathbf{B}$ można obliczyć za pomocą następującego wzoru:

$$\text{podobieństwo cosinusowe}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{2.1}$$

gdzie $\mathbf{A} \cdot \mathbf{B}$ to iloczyn skalarny wektorów $\mathbf{A}$ i $\mathbf{B}$, a $\|\mathbf{A}\|$ i $\|\mathbf{B}\|$ to normy tych wektorów.

Podobieństwo cosinusowe przyjmuje wartości w przedziale od -1 do 1. Wartość 1 oznacza, że wektory są identyczne pod względem kierunku, wartość 0 oznacza brak podobieństwa, a wartość -1 wskazuje, że wektory są przeciwnie skierowane.

Podobieństwo cosinusowe jest szeroko stosowane w różnych dziedzinach, takich jak wyszukiwanie informacji, analiza dokumentów czy rekomendacje produktów, ponieważ efektywnie mierzy podobieństwo w wysokowymiarowych przestrzeniach wektorowych. Jest to również kluczowe narzędzie w systemach automatyzacji przetwarzania CV, szczególnie w kontekście dopasowywania kandydatów do ofert pracy.

W pracy *A Machine Learning approach for automation of Resume Recommendation system* [24] opisano wykorzystanie podobieństwa cosinusowego do oceny zgodności treści CV z wyma-

ganiami stanowisk, co umożliwia efektywne sortowanie kandydatów. Autorzy podkreślają, że po wstępnym sklasyfikowaniu CV za pomocą różnych klasyfikatorów, podobieństwo cosinusowe, w połączeniu z metodą k-NN, jest stosowane do identyfikacji CV najbardziej zbliżonych do opisu stanowiska, co znacząco przyspiesza proces selekcji kandydatów.

2.3. Wybrane techniki uczenia maszynowego

W tym rozdziale przedstawione zostaną techniki uczenia maszynowego, które zostały zastosowane w niniejszej pracy do automatyzacji procesów rekrutacyjnych. Skupimy się na czterech kluczowych metodach: XGBoost (ang. Extreme Gradient Boosting), który wykorzystano do klasyfikacji CV kandydatów do odpowiednich kategorii, NER (ang. Named Entity Recognition) używanym do ekstrakcji imion i nazwisk z CV, LSTM (ang. Long Short Term Memory), zastosowanym do rankingu kandydatów od najlepszego do najgorszego, oraz BART (ang. Bidirectional and Auto-Regressive Transformers), który posłużył do generowania podsumowań CV.
Każda z tych metod odgrywa istotną rolę w poprawie precyzji i efektywności procesów rekrutacyjnych, wspomagając trafniejsze podejmowanie decyzji na podstawie analizowanych danych.

2.3.1. XGBoost

XGBoost (*XGBoost: A Scalable Tree Boosting System* [5]) to technika uczenia maszynowego, która zdobyła popularność dzięki swojej wydajności i skuteczności w zadaniach związanych z predykcją, takich jak klasyfikacja, regresja, czy budowa rankingu. XGBoost jest oparty na algorytmie gradient boosting, który tworzy silny model predykcyjny poprzez łączenie wielu słabych modeli bazowych, najczęściej drzew decyzyjnych. Algorytm ten został zaprojektowany z myślą o optymalizacji wydajności oraz radzeniu sobie z problemami wysokowymiarowych danych. Do głównych problemów wysokowymiarowych danych należą: przekleństwo wymiarowości, gdzie XGBoost radzi sobie z nadmierną liczbą cech poprzez zastosowanie mechanizmu selekcji i regularyzacji cech, automatycznie identyfikując najbardziej istotne cechy, redukując wpływ mniej znaczących zmiennych i zapobiegając nadmiernemu dopasowaniu modelu oraz rozcieńczenie danych (ang. data sparsity), gdzie w zbiorach o wysokiej wymiarowości często występuje duża liczba pustych lub rzadkich wartości. XGBoost posiada wbudowane mechanizmy obsługi takich danych, w tym specjalne strategie przypisywania wartości domyślnych i efektywne metody przetwarzania rzadkich macierzy.

Kluczowym aspektem XGBoost jest jego zdolność do iteracyjnego poprawiania błędów wcześniejszych modeli. Proces ten polega na dodawaniu kolejnych drzew decyzyjnych, które koncen-

trują się na poprawie błędów przewidywań popełnionych przez poprzednie drzewa. Każde nowe drzewo jest dopasowywane do reszt z poprzedniego kroku, co pozwala na stopniowe minimalizowanie błędów modelu.

XGBoost wprowadza szereg optymalizacji, które przyczyniają się do jego wysokiej wydajności:

- Wprowadza regularyzację L1 i L2, co pomaga w zapobieganiu nadmiernemu dopasowaniu (ang. overfitting) modelu. Regularizacja kontroluje złożoność modelu, co pozwala na bardziej ogólne przewidywania.

- Algorytm dobrze radzi sobie z brakującymi wartościami w danych, automatycznie przypisując je do najbardziej optymalnych gałęzi w drzewie decyzyjnym, bez potrzeby wcześniejszego uzupełniania braków. XGBoost ocenia, która z gałęzi drzewa przyniesie największą korzyść (minimalizację straty lub maksymalizację zysku) w przypadku obserwacji z brakującymi wartościami, co pozwala na lepsze dopasowanie modelu i bardziej efektywne wykorzystanie danych.

- XGBoost został zaprojektowany z myślą o optymalizacji wykorzystania pamięci oraz wydajności obliczeniowej. Używa on rzadkich macierzy danych, co redukuje ilość potrzebnej pamięci.

- XGBoost wspiera równoległe przetwarzanie, co znacząco przyspiesza budowę modelu na dużych zbiorach danych. Algorytm pozwala na równoczesne tworzenie drzew oraz równoległą optymalizację przycinania drzew (ang. tree pruning).

XGBoost znalazł szerokie zastosowanie w wielu dziedzinach, takich jak analiza danych finansowych, detekcja oszustw, prognozowanie sprzedaży, czy przewidywanie chorób. Jego zdolność do przetwarzania dużych ilości danych, skuteczność w radzeniu sobie z danymi o wysokiej wymiarowości oraz odporność na nadmierne dopasowanie sprawiają, że jest to jedno z najczęściej używanych narzędzi w praktycznych zastosowaniach uczenia maszynowego.

W pracy *A Comparative Study of Heart Disease Prediction Using Machine Learning Techniques* [1] XGBoost został porównany z innymi algorytmami klasyfikacyjnymi, takimi jak regresja logistyczna, maszyny wektorów nośnych, Naive Bayes, k-NN, drzewo decyzyjne, Adaboost oraz wielowarstwowy perceptron. Badanie to wykazało, że XGBoost osiągał najwyższą dokładność w przewidywaniu chorób serca, uzyskując 89% dokładności (ang. accuracy) w danych bez wartości odstających oraz 84,6% w danych z wartościami odstającymi. Autorzy podkreślają, że zastosowanie techniki SMOTE-Tomek Link (ang. Synthetic Minority Oversampling Technique Tomek Link) do równoważenia danych miało kluczowe znaczenie dla uzyskania stabilnych

wyników, a wykorzystanie metody k-NN, choć szybsze, nie dorównywało efektywności XGBoost w analizie złożonych wzorców w danych.

2.3.2. NER

Named Entity Recognition (*A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text* [6]) to technika przetwarzania języka naturalnego, która polega na identyfikowaniu i klasyfikowaniu nazwanych jednostek w tekście, takich jak imiona, nazwiska, nazwy firm, lokalizacje, daty i wiele innych. W kontekście tej pracy, NER jest używany do ekstrakcji imion i nazwisk z CV kandydatów, co stanowi ważny krok w dalszym przetwarzaniu i analizie dokumentów aplikacyjnych.

NER działa poprzez analizę sekwencji tekstu i przypisywanie odpowiednich etykiet do fragmentów, które rozpoznaje jako nazwane jednostki. Algorytm, korzystając z modelu językowego, rozpoznaje wzorce typowe dla nazw osób, takie jak struktura imienia i nazwiska, kontekst gramatyczny oraz miejsce w zdaniu. Na przykład, w zdaniu "Jan Kowalski aplikuje na stanowisko programisty", NER identyfikuje "Jan Kowalski"jako nazwę własną, a następnie przyporządkowuje jej etykietę osoba.

Proces ten może być wspierany przez wcześniej wytrenowane modele, które uczą się rozpoznawać nazwy osób na podstawie dużych zbiorów danych zawierających różne przykłady nazw własnych. Model NER analizuje cechy charakterystyczne sekwencji tekstowych, takie jak kapitalizacja, sąsiadujące słowa oraz pozycję w zdaniu, aby precyzyjnie określić, że "Jan Kowalski" to imię i nazwisko osoby, a nie inny typ jednostki nazwanej, np. lokalizacja czy organizacja.

W przypadku tej pracy zastosowano modele NER oparte na głębokich sieciach neuronowych, które potrafią precyzyjnie rozpoznawać jednostki nazwane w różnych kontekstach. Dzięki temu możliwa jest skuteczna ekstrakcja imion i nazwisk nawet z CV o zróżnicowanych formatach i stylach pisania.

NER ma szerokie spektrum zastosowań, a jedno z nich zostało szczegółowo opisane w pracy *Named Entity Recognition and Resolution in Legal Text* [8]. Autorzy skoncentrowali się na analizie dokumentów prawnych, wykorzystując NER do identyfikacji jednostek takich jak sędziowie, adwokaci oraz firmy. W badaniu przedstawiono trzy kluczowe metody NER: wyszukiwanie, zasady kontekstowe oraz modele statystyczne. Wyniki wykazały, że zastosowanie NER może znacząco poprawić dostęp do informacji o konkretnych jednostkach, co jest niezwykle istotne dla analizy i interpretacji dokumentów prawnych. Ponadto, w artykule omówiono system dedykowany rozpoznawaniu nazwanych encji w tekstach prawnych oraz techniki uczenia maszynowego stosowane do ich rozwiązania, co podkreśla wszechstronny potencjał NER w różnych dziedzinach.

2.3.3. BART

Bidirectional and Auto-Regressive Transformer (*BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* [17]) to zaawansowany model uczenia maszynowego, który może być wykorzystany do przetwarzania języka naturalnego. Został on wprowadzony w 2019 roku przez zespół Facebook AI. BART jest modelem typu seq2seq, który łączy cechy modelu auto-regresyjnego i auto-enkodera, co pozwala na efektywne radzenie sobie z zadaniami generowania tekstu, tłumaczenia oraz podsumowywania.
W trakcie pre-treningu BART wprowadza różne zakłócenia do tekstu (takie jak usunięcie słów, zmiana słów, itp.) i uczy się odbudować oryginalny tekst z zakłóconej wersji. Ten proces polega na nauczeniu modelu jak skutecznie rekonstruować tekst, co poprawia jego zdolności do generowania tekstu i zrozumienia kontekstualnego. BART jest również pre-trenowany jako model auto-regresyjny, który przewiduje następne słowo w sekwencji na podstawie poprzednich słów. To umożliwia modelowi generowanie spójnych i sensownych tekstów.

W niniejszej pracy BART został zastosowany do podsumowywania tekstu CV. Dzięki swojej zdolności do rozumienia kontekstowego i generowania koherentnych streszczeń, BART skutecznie przetwarza dokumenty, takie jak CV, i tworzy zwięzłe streszczenia. Model ten analizuje pełen kontekst dokumentu, co pozwala na skuteczne wydobycie kluczowych informacji i przedstawienie ich w skondensowanej formie.

BART został wprowadzony w pracy *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* [17] jako zaawansowany autoenkoder szumów przeznaczony do trenowania modeli sekwencyjnych. Znalazł on szerokie zastosowanie w różnych dziedzinach NLP, takich jak tłumaczenie maszynowe, generowanie tekstu, a także tworzenie streszczeń. Autorzy osiągnęli znaczące wyniki w zadaniach generowania tekstu, w tym w podsumowywaniu i tłumaczeniu. W przeprowadzonych eksperymentach model BART wykazał znaczną przewagę, osiągając poprawę wyników o nawet 6 punktów w metrykach ROUGE dla podsumowań na zbiorze XSum oraz o 1,1 punktu BLEU w zadaniu tłumaczenia, przewyższając systemy oparte na back-translation. Jego wszechstronność i efektywność czynią go wyjątkowo skutecznym narzędziem we współczesnym przetwarzaniu języka naturalnego.

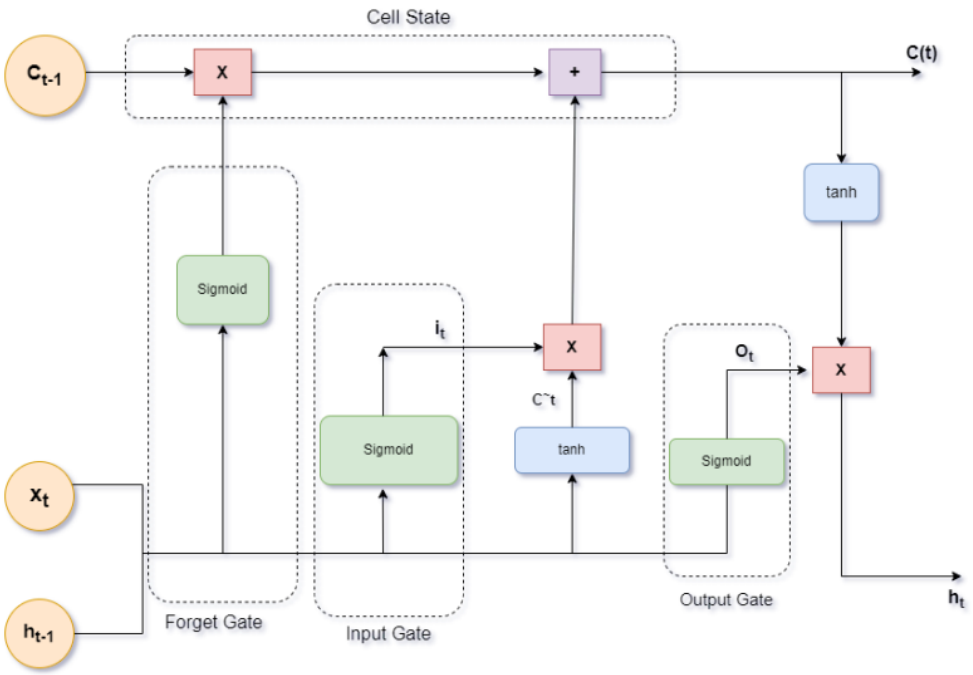
2.3.4. LSTM

Long Short-Term Memory (*Long Short-Term Memory*, LSTM [13]) to zaawansowany typ sztucznej sieci neuronowej zaprojektowany w celu skutecznego modelowania i przetwarzania sekwencji danych, szczególnie w przypadkach wymagających zachowania i analizy długoterminowych zależności między elementami w sekwencji. LSTM jest rozwinięciem klasycznych rekurencyjnych sieci neuronowych (RNN), które charakteryzują się zdolnością do przetwarzania danych sekwencyjnych, takich jak tekst, dźwięk czy serie czasowe. Jednak tradycyjne RNN napotykają poważne ograniczenia związane z problemem zanikania i eksplodowania gradientu podczas procesu propagacji wstecznej, co sprawia, że są one nieefektywne w modelowaniu długoterminowych zależności w danych. LSTM zostało wprowadzone w 1997 roku przez Seppa Hochreitera i Jürgena Schmidhubera jako odpowiedź na te ograniczenia. Architektura LSTM rozwiązuje wspomniane problemy dzięki wprowadzeniu specjalnej jednostki pamięci (ang. *memory cell*) oraz zestawu mechanizmów kontrolnych w postaci bramek, które umożliwiają selektywne zapamiętywanie, zapominanie oraz udostępnianie informacji. Kluczową cechą tego rozwiązania jest zdolność do dynamicznego zarządzania przepływem informacji w sieci, co czyni LSTM niezwykle skutecznym narzędziem do analizy danych o skomplikowanej strukturze sekwencyjnej. Każda komórka LSTM składa się z trzech głównych bramek kontrolnych:

- **Bramka zapomnienia (ang. Forget Gate):** Odpowiada za decydowanie, które informacje przechowywane w komórce pamięci powinny zostać usunięte. Na podstawie aktualnego wejścia oraz poprzedniego stanu ukrytego, bramka generuje wartości między 0 a 1, reprezentujące stopień, w jakim informacje z poprzedniego stanu są zachowywane lub odrzucane. Dzięki temu mechanizmowi model potrafi "zapomnieć"nieistotne informacje, redukując wpływ szumów w danych.

- **Bramka wejściowa (ang. Input Gate):** Steruje procesem aktualizacji stanu komórki o nowe informacje. Używa funkcji aktywacji sigmoidalnej, aby określić, które wartości mogą zostać zmodyfikowane, oraz funkcji tanh, aby wygenerować potencjalne nowe wartości do dodania. W ten sposób model selektywnie uczy się, które dane z bieżącego wejścia są istotne dla długoterminowej reprezentacji.

- **Bramka wyjściowa (ang. Output Gate):** Kontroluje, jakie części stanu komórki zostaną przekazane dalej jako wyjście. Wykorzystuje kombinację funkcji aktywacji sigmoidalnej i tanh, aby przetworzyć aktualny stan komórki i wygenerować wynikowy stan ukryty, który może służyć jako wejście do kolejnych kroków w sekwencji lub warstw w modelu.



Rysunek 2.1: LSTM [27]

LSTM znalazły szerokie zastosowanie w różnych dziedzinach, takich jak analiza sekwencji, prognozowanie szeregów czasowych, tłumaczenie maszynowe, generowanie tekstu oraz rozpoznawanie mowy. Dzięki zdolności do długoterminowego zapamiętywania informacji, LSTM są szczególnie skuteczne w modelowaniu danych sekwencyjnych, które mają długotrwałe zależności.

Praca *Sequence to Sequence Learning with Neural Networks* [26] stanowi kluczowy wkład w rozwój zastosowań LSTM do modelowania sekwencyjnego, szczególnie w kontekście tłumaczenia maszynowego. Autorzy pokazali, że model oparty na architekturze seq2seq, wykorzystujący dwie warstwy LSTM – enkoder i dekoder – potrafi efektywnie tłumaczyć tekst z jednego języka na inny, osiągając lepsze wyniki niż poprzednie podejścia. W eksperymentach model seq2seq z LSTM osiągnął zbliżoną jakość tłumaczeń do tradycyjnych systemów bazujących na frazach oraz poprawił się w automatycznych metrykach takich jak BLEU, co zapoczątkowało szerokie zastosowanie LSTM w przetwarzaniu języka naturalnego.

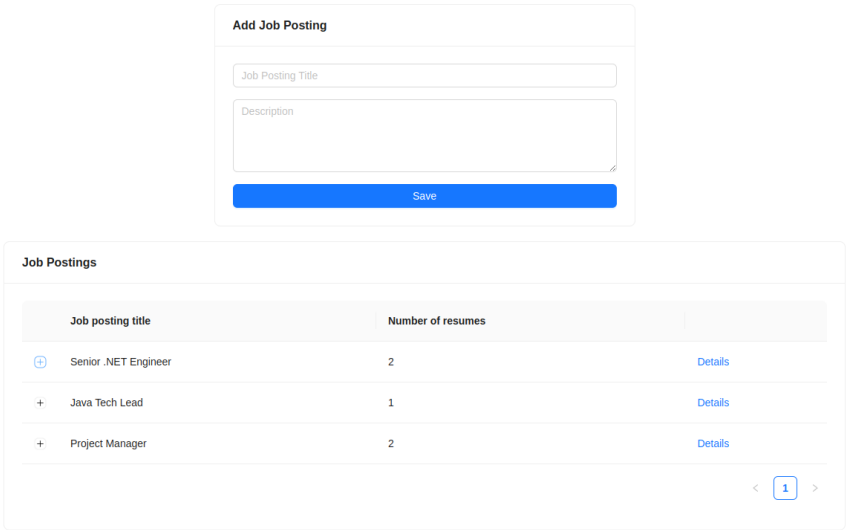
3. Architektura i implementacja systemu

3.1. Opis systemu zarządzania CV

W ramach pracy magisterskiej zaimplementowano system do automatyzacji przeglądu CV, umożliwiający zarządzanie ofertami pracy oraz aplikacjami kandydatów w sposób zautomatyzowany. Aplikacja składa się z webowego interfejsu użytkownika stworzonego w frameworku `React` [10] z wykorzystaniem biblioteki `Ant Design` [11] oraz języka `Typescript` [19]. Backend systemu, napisany w Pythonie [28] przy użyciu frameworka `Flask` [23], obsługuje logikę biznesową, przechowuje wagi modeli uczenia maszynowego i zarządza bazą danych `SQLite` [12]. W tym rozdziale omówiono główne funkcjonalności systemu, poparte odpowiednimi zrzutami ekranu.

3.1.1. Dodawanie ofert pracy

Pierwszy ekran systemu umożliwia użytkownikowi dodawanie nowych ofert pracy. Z tego poziomu możliwe jest również wyświetlenie listy wcześniej dodanych ofert pracy wraz z liczbą aplikantów, którzy przesłali swoje CV na daną ofertę. Rysunek 3.1 przedstawia widok tego ekranu.

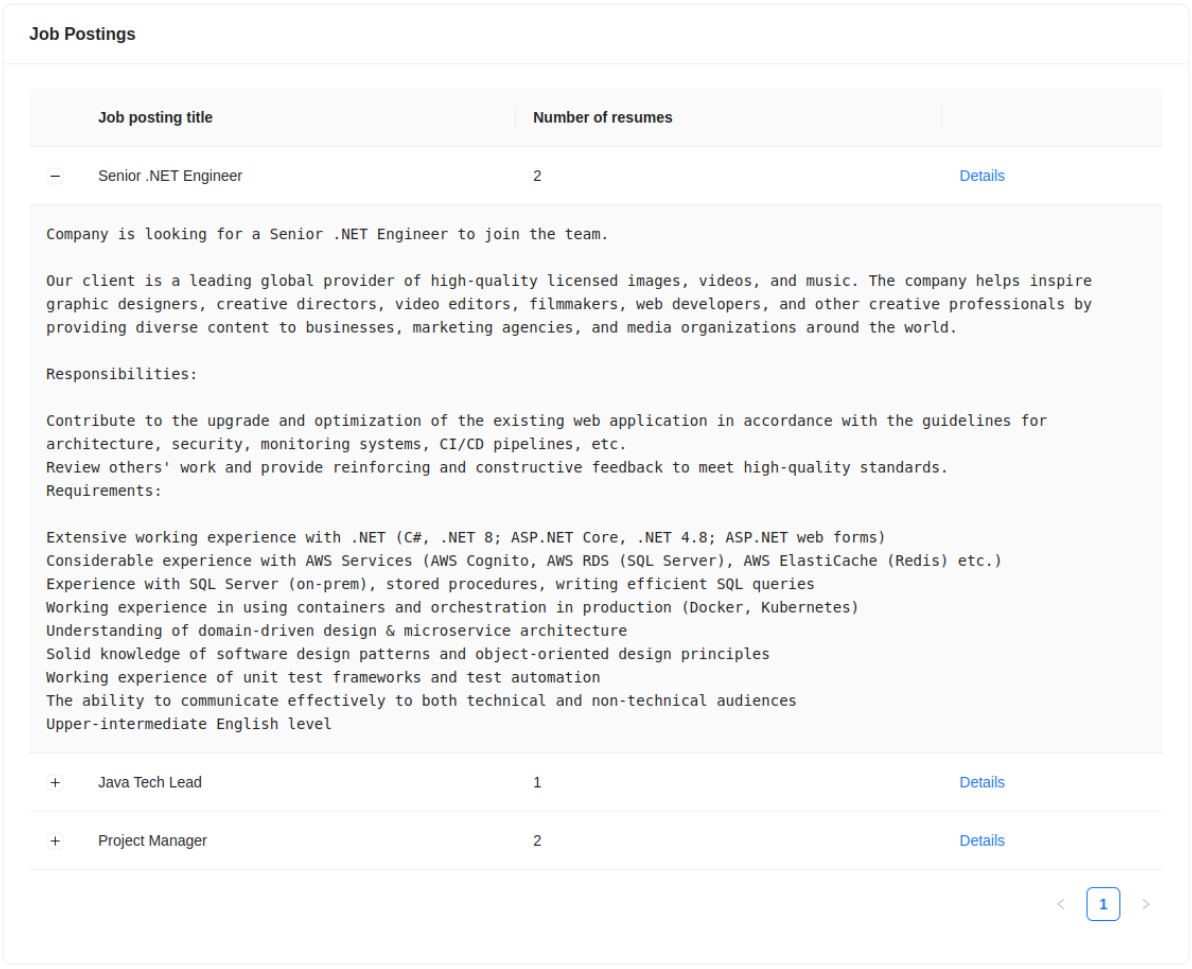


Rysunek 3.1: Ekran dodawania ofert pracy i lista dostępnych ofert.

Na rysunku 3.1 widzimy formularz do wprowadzania tytułu oferty pracy oraz jej opisu. Poniżej widoczna jest lista ofert pracy, która zawiera kolumny takie jak tytuł oferty oraz liczba aplikantów. System umożliwia szybkie dodawanie nowych ofert oraz monitorowanie liczby aplikacji.

3.1.2. Szczegóły oferty pracy

Po wybraniu konkretnej oferty z listy użytkownik może przejść do jej szczegółów. Rysunek 3.2 przedstawia ekran, na którym rozwinięty został rekord wybranej oferty pracy.

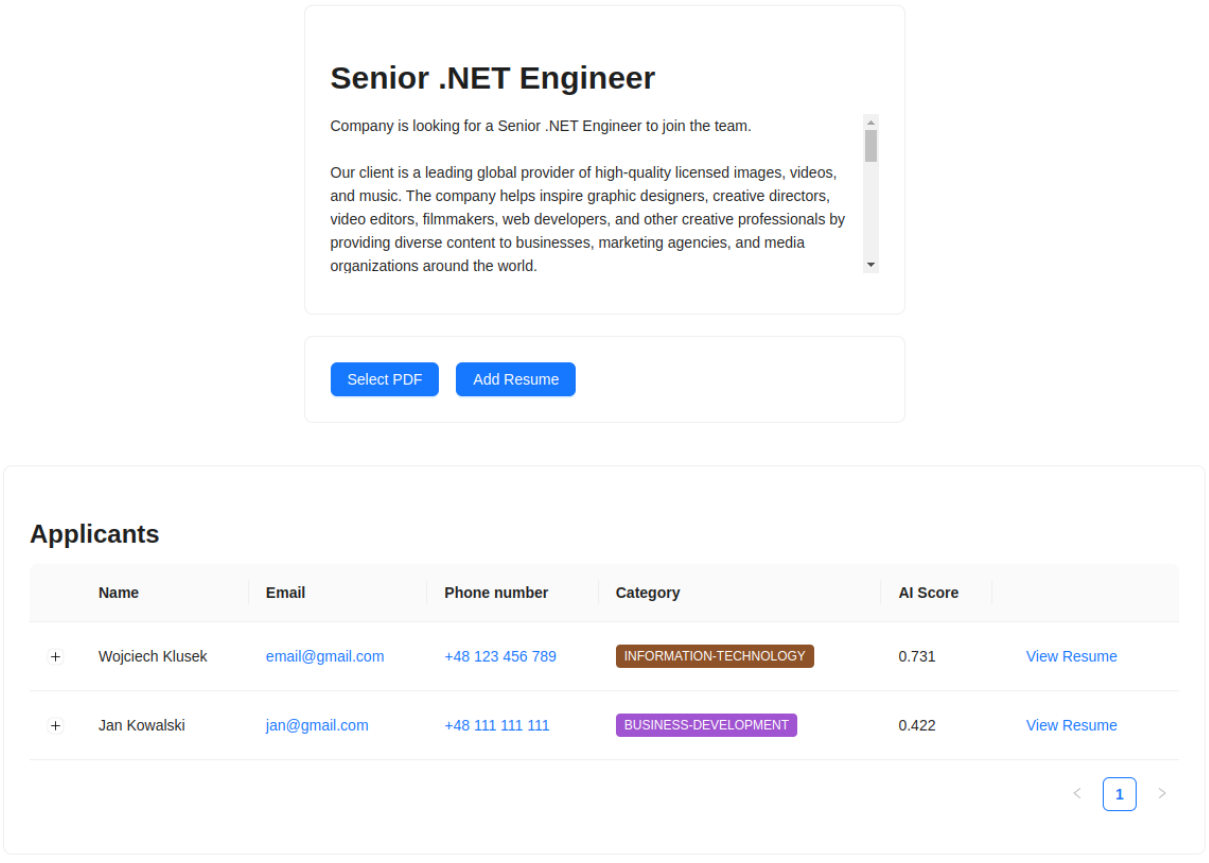


Rysunek 3.2: Szczegóły wybranej oferty pracy.

Jak pokazano na rysunku 3.2, po kliknięciu na daną ofertę z listy rozwija się widok szczegółowy, który wyświetla pełen opis pracy. Dzięki temu rekruter ma dostęp do wszystkich istotnych informacji dotyczących danej pozycji, co ułatwia ocenę i dalsze zarządzanie ofertą.

3.1.3. Zarządzanie CV kandydatów

Trzeci ekran systemu umożliwia zarządzanie CV kandydatów przesłanych na daną ofertę pracy. Na rysunku 3.3 przedstawiono widok szczegółów oferty pracy wraz z listą CV, które zostały na nią przesłane.



Rysunek 3.3: Zarządzanie CV kandydatów dla wybranej oferty pracy.

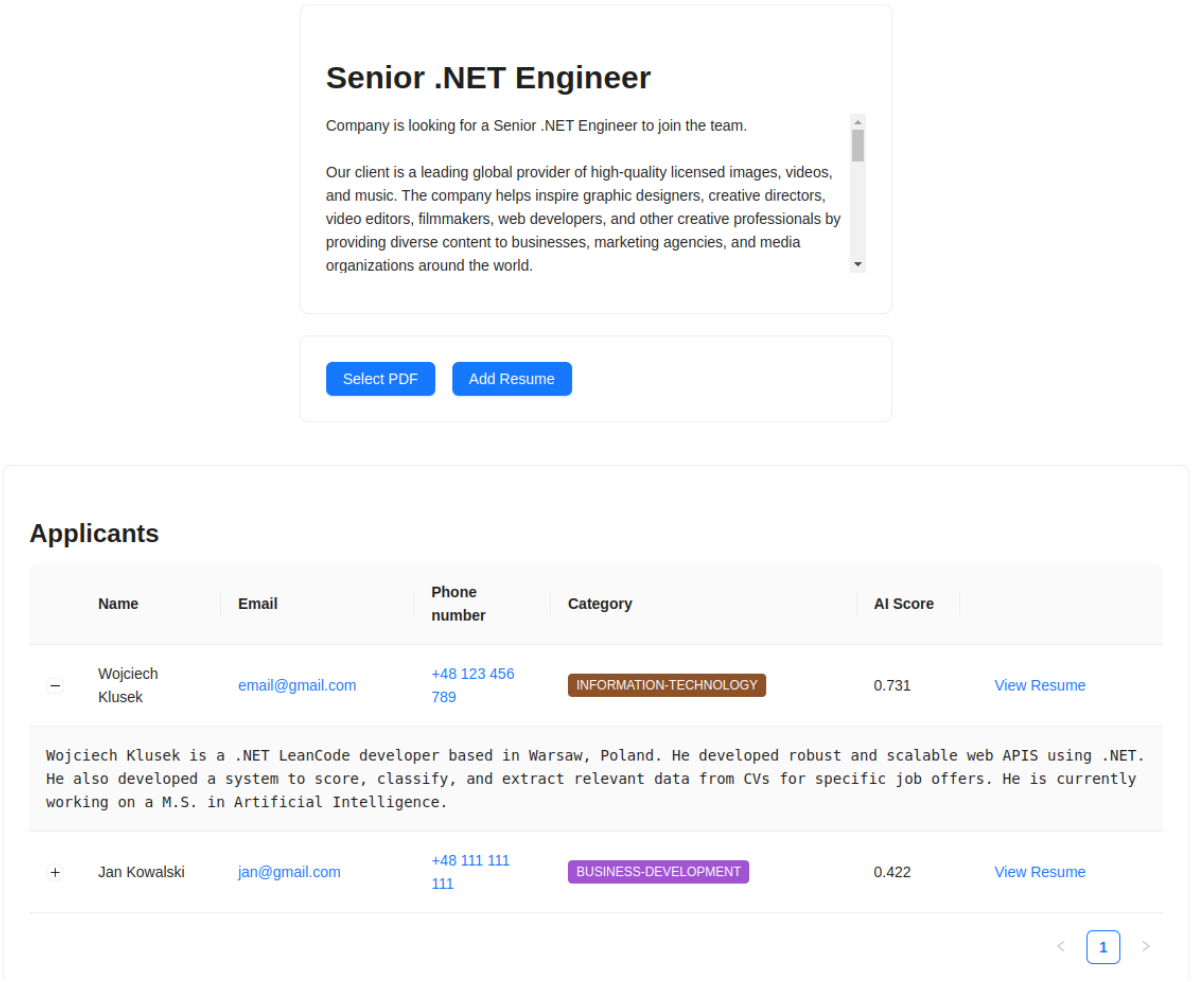
Na rysunku 3.3 użytkownik może zobaczyć listę wszystkich przesłanych CV na wybraną ofertę pracy. System umożliwia również dodanie nowych CV poprzez wybranie pliku PDF z dysku i automatycznie przetwarza plik zawierający CV, ekstrakując kluczowe informacje:

- Imię i nazwisko kandydata za pomocą techniki rozpoznawania nazw własnych (NER),
- Adres email oraz numer telefonu z wykorzystaniem wyrażeń regularnych (Regex),
- Kategorię CV, która jest określana przy użyciu klasyfikatora opartego na algorytmie XGBoost,
- Współczynnik dopasowania kandydata do danej oferty pracy, wyliczany z wykorzystaniem siei neuronowej opartej na architekturze LSTM.

W ten sposób system pozwala na efektywne zarządzanie aplikacjami kandydatów oraz integrację nowych aplikacji bezpośrednio z poziomu panelu administracyjnego.

3.1.4. Podsumowanie CV

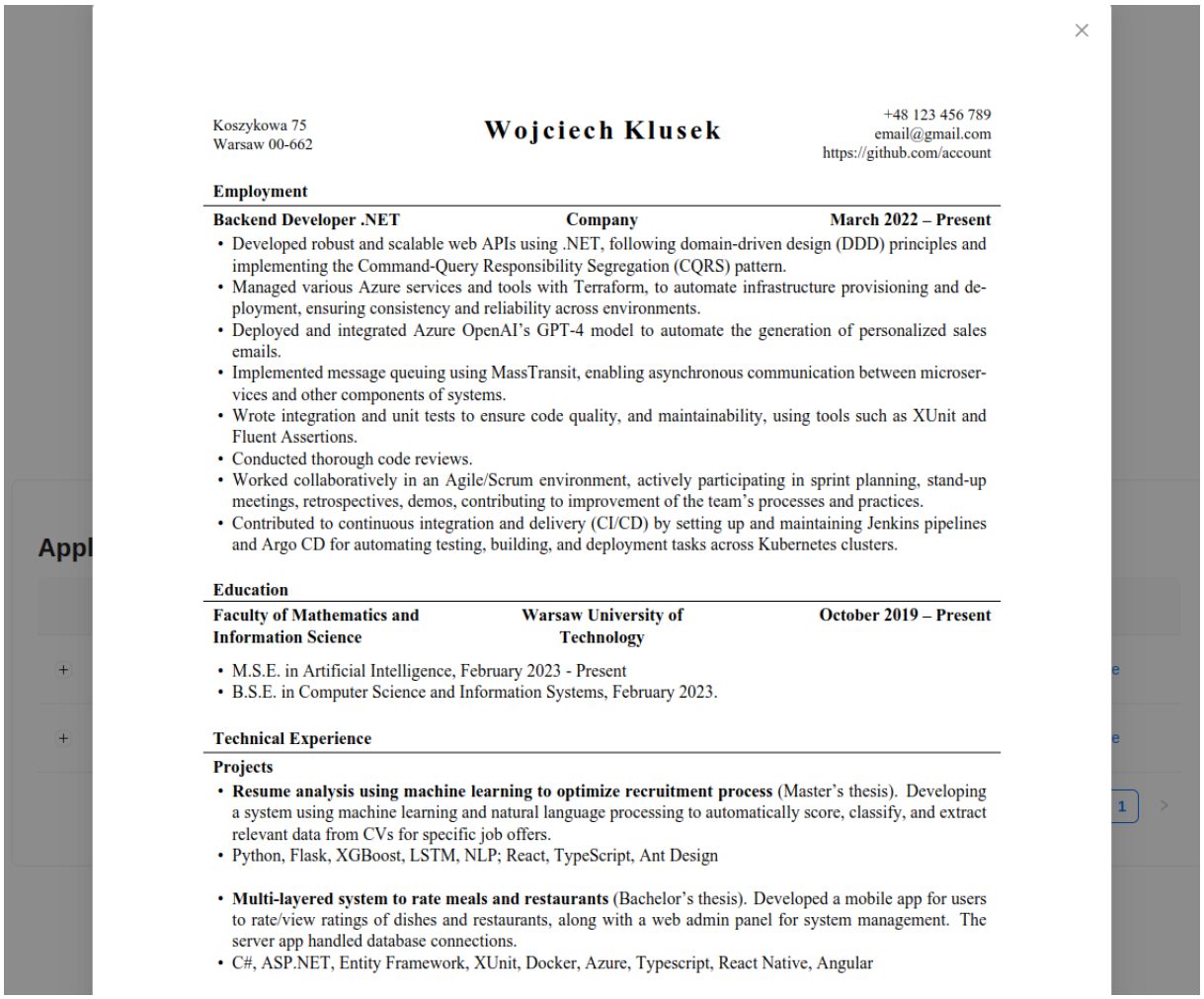
Po rozwinięciu pojedynczego rekordu z listy przesłanych CV użytkownik ma dostęp do szczegółowego podsumowania danego CV. Rysunek 3.4 przedstawia ten widok.



Rysunek 3.4: Podsumowanie wybranego CV.

3.1.5. Podgląd pliku PDF

Ostatni ekran systemu pozwala na podgląd pełnej treści pliku PDF zawierającego CV bezpośrednio w przeglądarce. Rysunek 3.5 ilustruje ten widok.



Rysunek 3.5: Podgląd pliku PDF z CV wewnątrz przeglądarki.

Jak pokazano na rysunku 3.5, system umożliwia przeglądanie pliku PDF bez konieczności pobierania go na lokalne urządzenie. Taka funkcjonalność znacznie ułatwia analizę pełnej treści dokumentu i pozwala na szybkie przejrzenie aplikacji bez opuszczania interfejsu systemu.

3.2. Ekstrakcja danych

W tym rozdziale zaprezentowany zostanie proces automatycznej ekstrakcji danych z CV, który bazuje na przetwarzaniu języka naturalnego oraz uczeniu maszynowym. Na początku omówione zostanie podejście do identyfikacji kluczowych informacji, takich jak imię i nazwisko, numer telefonu oraz adres e-mail z treści dokumentu CV. Następnie opisane zostaną techniki i algorytmy zastosowane w ekstrakcji danych, które umożliwiają skuteczne podsumowania CV.

3.2.1. Ekstrakcja imienia i nazwiska

Do ekstrakcji imienia i nazwiska z tekstu zastosowano Named Entity Recognition z biblioteki `spaCy` [14]. W modelu `en_core_web_lg`, który został wykorzystany w tym celu, NER jest odpowiedzialny za identyfikację i klasyfikację jednostek nazwanych, takich jak osoby (etykieta `PERSON`), organizacje, lokalizacje i inne kategorie. W kontekście analizy CV, model NER został użyty do wyodrębnienia pełnych imion i nazwisk. Tekst dokumentu jest przetwarzany przez model, który identyfikuje fragmenty tekstu będące imionami i nazwiskami, a następnie zwraca je jako wynik analizy.

3.2.2. Ekstrakcja numeru telefonu i adresu e-mail

Ekstrakcja numeru telefonu oraz adresu e-mail z treści CV została zrealizowana przy użyciu wyrażeń regularnych. Dla numerów telefonów zastosowano następujący wzorzec:

```
(?:\+?\d{1,3}[\s\-]?)?
(?:\(?\d{2,4}\)?[\s\-]?)?
\d{2,4}[\s\-]?\d{2,4}[\s\-]?\d{2,4}(?=\s|$)
```

Ten wzorzec umożliwia rozpoznanie różnych formatów numerów telefonów, takich jak międzynarodowe prefiksy, nawiasy, myślniki, spacje, a także kropki. Obsługuje on opcjonalny międzynarodowy prefiks, który może zawierać znak plus oraz kod kraju składający się z 1 do 3 cyfr, z możliwością oddzielenia go spacjami lub myślnikami. Wzorzec pozwala również na obecność numeru kierunkowego, który może być zapisany w nawiasach (np. (22) dla Warszawy), ale nawiasy są opcjonalne. Po numerze kierunkowym mogą występować spacje lub myślniki. Główna część numeru telefonu składa się z grup cyfr, z możliwością rozdzielenia ich spacjami, myślnikami lub innymi separatorami, co pozwala na różne formaty, takie jak 123 456 789, 123-456-789 czy 123456789. Dzięki swojej elastyczności, wzorzec jest w stanie rozpoznać numery telefonów o różnych długościach i formatach, zarówno krajowe, jak i międzynarodowe.

W przypadku adresów e-mail zastosowano poniższe wyrażenie regularne, które zostało opracowane na podstawie wzorca dostępnego na stronie `https://emailregex.com/` [9].
Wyrażenie to jest zgodne z oficjalnym standardem RFC 5322 [22], który definiuje poprawny format adresu e-mail, uwzględniając takie elementy jak nazwa użytkownika, domeny główne i subdomeny, a także dozwolone znaki specjalne. Dzięki temu wyrażenie obsługuje większość praktycznych przypadków adresów e-mail.

```
(?:[a-z0-9!#$%&\'*+/=?^_`{|}~-]+
  (?:\.[a-z0-9!#$%&\'*+/=?^_`{|}~-]+)*
  |"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]
    |\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@
(?:
  (?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+
  [a-z0-9](?:[a-z0-9-]*[a-z0-9])?
  |\[(?:
    (?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}
    (?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?
    |[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]
    |\\[\x01-\x09\x0b\x0c\x0e-\x7f])+
    )\]
)
```

Wzorzec ten dopasowuje standardowe formaty adresów e-mail, uwzględniając różnorodne kombinacje liter, cyfr oraz znaków specjalnych. W rezultacie umożliwia precyzyjne wydobycie adresów e-mail z tekstu CV.

3.3. Podsumowanie CV

W celu generowania zwięzłych podsumowań zawartości CV wykorzystano pretrenowany model `BART`, dostępny w ramach biblioteki `Hugging Face Transformers`, zaimplementowany przez zespół Facebook AI. BART, czyli Bidirectional and Auto-Regressive Transformers, jest modelem typu seq2seq z dwukierunkowym enkoderem (podobnym do `BERT`) oraz autoregresywnym dekoderem (podobnym do `GPT`). Model ten został przetrenowany na korpusie języka angielskiego, a następnie dostrojony na zbiorze danych CNN Daily Mail, co czyni go szczególnie skutecznym w zadaniach generowania tekstu, takich jak podsumowywanie. W pracy ten model został zastosowany do generowania streszczeń CV, umożliwiając szybkie i efektywne przetwarzanie dużej liczby dokumentów. Model BART był pretrenowany poprzez wprowadzenie szumu do tekstu, a następnie trenowany do odtwarzania oryginalnej treści. Dzięki tej metodzie BART jest wyjątkowo skuteczny w zadaniach związanych z generowaniem tekstu, jak również w zadaniach zrozumienia tekstu, takich jak klasyfikacja tekstu czy odpowiadanie na pytania.

3.4. Klasyfikacja CV

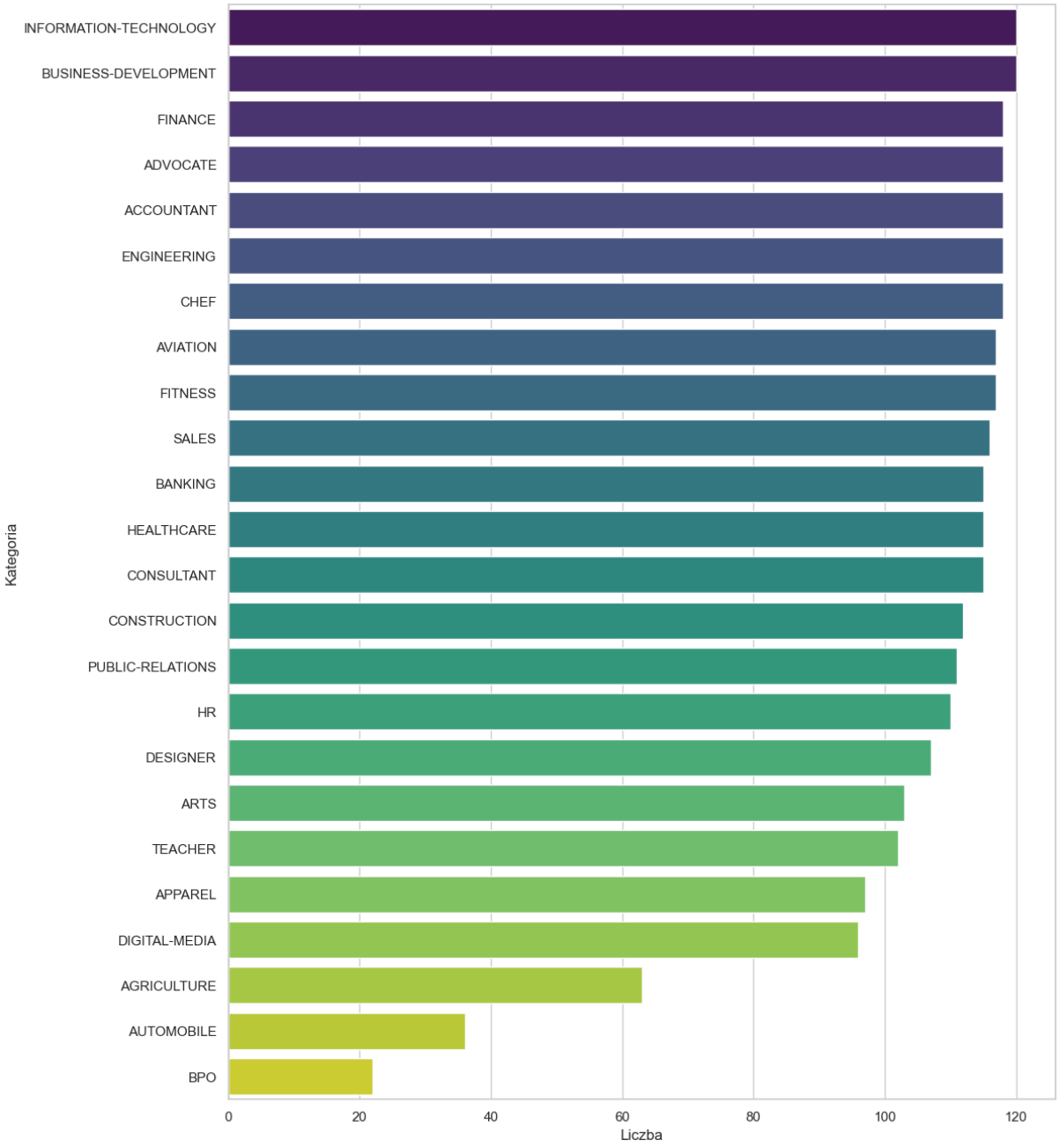
W tym rozdziale przedstawiony zostanie proces klasyfikacji CV, który obejmuje kilka kluczowych etapów. Najpierw omówiony zostanie zbiór danych, na którym przeprowadzono analizy. Następnie opisane zostaną metody wstępnego przetwarzania tekstu, które umożliwiły przekształcenie surowych danych w formę odpowiednią do modelowania. Kolejno, zostanie zaprezentowany model klasyfikacyjny wykorzystany do przypisywania CV do odpowiednich kategorii zawodowych. Na koniec przedstawione zostaną wyniki osiągnięte przez model, które pozwolą ocenić jego skuteczność w realizacji zadania.

3.4.1. Zbiór danych

Zbiór danych użyty do klasyfikacji CV pochodzi z serwisu Kaggle i jest dostępny pod adresem: `https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset`. Zawiera on ponad 2400 zanonimizowanych CV zebranych ze strony livecareer.com. Dane zostały przygotowane w dwóch formatach: pliki PDF oraz ich tekstowe reprezentacje w pliku CSV. Plik CSV zawiera następujące kolumny: identyfikator (`ID`), tekst CV (`Resume_str`), dane w formacie HTML (`Resume_html`) oraz przypisaną kategorię zawodową (`Category`). Pliki PDF są zorganizowane w katalogach odpowiadających poszczególnym kategoriom zawodowym, co dodatkowo wspiera możliwość ręcznej eksploracji danych.

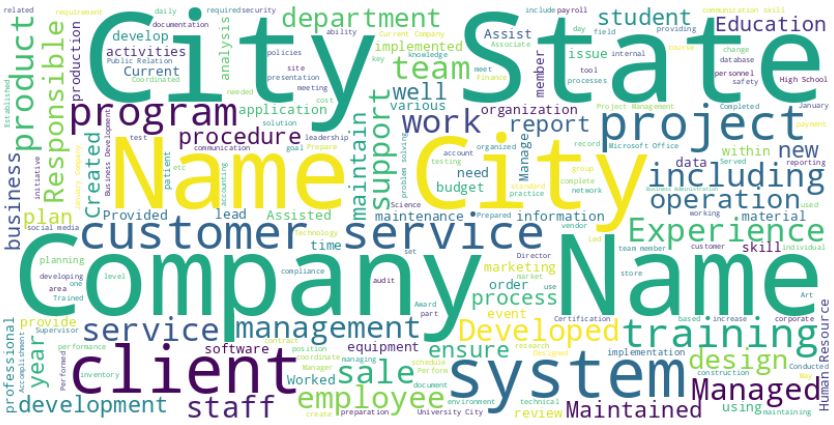
Zbiór danych obejmuje szeroką gamę kategorii zawodowych, takich jak: information technology, business development, finance, advocate, accountant, engineering, chef, aviation, fitness, sales, agriculture, automotive oraz bpo. Każda kategoria zawodowa posiada przypisaną liczbę CV, co obrazuje poniższy wykres (3.6) przedstawiający rozkład liczby CV w poszczególnych kategoriach. Najwięcej CV przypisano do kategorii takich jak information technology oraz business development, które zawierają po 120 przykładów. Kategorie takie jak finance, advocate oraz accountant posiadają podobną liczbę przykładów, wynoszącą ponad 100 CV. Mniej licznie reprezentowane są kategorie engineering, chef, aviation, fitness i sales, choć ich liczność wciąż pozostaje wystarczająca dla skutecznego uczenia modelu. Wyjątkiem są trzy kategorie o najmniejszej liczbie przykładów: agriculture, automotive oraz bpo.

Mimo mniejszych różnic w liczności pomiędzy poszczególnymi kategoriami, zbiór danych można uznać za zrównoważony, co jest istotnym czynnikiem wpływającym na jakość procesu klasyfikacji. Taka równomierność w rozkładzie danych umożliwia modelowi skuteczne uczenie się różnorodnych wzorców bez uprzywilejowania kategorii większościowych, co minimalizuje ryzyko błędów klasyfikacyjnych wynikających z niezbalansowanego zestawu treningowego.



Rysunek 3.6: Rozkład liczby CV w poszczególnych kategoriach zawodowych

Aby lepiej zrozumieć charakterystykę tekstu zawartego w CV, wygenerowano wizualizację, przedstawioną na rysunku 3.7. Obraz ten pokazuje najczęściej występujące słowa w tekstach CV, gdzie większe rozmiary czcionek oznaczają częstsze występowanie danego słowa. Dominującymi słowami są m.in. Name, Company, City, State, Client, Service oraz System. Ich obecność odzwierciedla typową strukturę CV, w której często pojawiają się dane osobowe, nazwy firm, lokalizacje, a także słowa związane z obowiązkami zawodowymi i kompetencjami jak: Sale, Product, Project czy Management.



Rysunek 3.7: Najczęściej występujące słowa w tekstach CV.

3.4.2. Wstępne przetwarzanie

Wstępne przetwarzanie danych jest kluczowym krokiem w procesie klasyfikacji CV, który umożliwia przekształcenie surowych danych tekstowych w formę, która może być efektywnie wykorzystana przez modele uczenia maszynowego. W klasyfikacji CV, przetwarzanie danych składa się z kilku istotnych kroków, które mają na celu oczyszczenie i standaryzację tekstu, co zwiększa skuteczność procesu klasyfikacji.

Najpierw, teksty CV są oczyszczane z wszelkich znaków nieliterowych, takich jak cyfry, znaki interpunkcyjne oraz specjalne symbole. Następnie, cały tekst jest konwertowany na małe litery, aby uniknąć problemów z rozróżnianiem wielkości liter, co mogłoby prowadzić do niepotrzebnych różnic w analizie tekstu.

W kolejnym etapie z tekstu są usuwane tzw. "stop words", czyli powszechnie występujące słowa, które nie niosą ze sobą istotnej wartości semantycznej w kontekście klasyfikacji (np. "and", "the", "in"). Proces ten pozwala skupić się na istotnych słowach kluczowych, które mają większe znaczenie w kontekście określenia kategorii zawodowej.

Ostatnim krokiem przetwarzania tekstu jest lematyzacja, która polega na sprowadzeniu słów do ich podstawowej formy (lematu). Dzięki temu słowa o różnych formach gramatycznych, ale tej samej podstawowej treści, są traktowane jako jedno, co pozwala na lepsze uogólnienie danych. Po wykonaniu tych operacji teksty CV są gotowe do dalszej analizy.

Przekształcony tekst jest następnie zamieniany na wektor cech przy użyciu metody TF-IDF (ang. Term Frequency-Inverse Document Frequency). TF-IDF mierzy częstość występowania słów w dokumencie, z uwzględnieniem ich unikalności w całym zbiorze danych. Wektor cech służy jako wejście do modelu klasyfikacyjnego.

Przeprowadzone wstępne przetwarzanie danych znacząco wpłynęło na poprawę jakości danych

wejściowych oraz skuteczność procesu klasyfikacji CV. Operacje takie jak oczyszczanie tekstu, usuwanie stop words i lematyzacja pozwoliły na redukcję szumów i standaryzację danych, co przełożyło się na lepsze dopasowanie modelu do zadania klasyfikacyjnego. Dodatkowo, zastosowanie metody TF-IDF do reprezentacji tekstu umożliwiło uchwycenie istotnych różnic między kategoriami zawodowymi, co zwiększyło zdolność modelu do prawidłowej klasyfikacji. W wyniku tych działań skuteczność klasyfikacji znacząco wzrosła, co zostało opisane w rozdziale 3.4.7.

3.4.3. Wykorzystany model

Proces klasyfikacji CV opiera się na wykorzystaniu Extreme Gradient Boosting Classifier, jednego z najbardziej efektywnych algorytmów klasyfikacji. Model jest trenowany na wcześniej przygotowanych wektorach cech, które powstały w procesie przetwarzania tekstu i został wybrany ze względu na jego zdolność do łączenia wielu słabszych klasyfikatorów w jeden silniejszy model, który jest w stanie osiągnąć wysoką dokładność.

3.4.4. Wyniki

Poniższe wyniki zostały uzyskane przy wykorzystaniu wybranych parametrów modelu oraz konfiguracji, które zostały zoptymalizowane w procesie eksperymentów. Dokładny opis procesu optymalizacji hiperparametrów został opisany w sekcji 3.4.5. Kluczowe parametry modelu oraz konfiguracja są przedstawione w tabeli 3.1.

Tabela 3.1: Parametry modelu i konfiguracja

Parametr	**Wartość**
Współczynnik uczenia (`learning_rate`)	0.3
Maksymalna głębokość drzewa (`max_depth`)	6
Minimalna waga liścia (`min_child_weight`)	1
Regularyzacja L2 (`lambda`)	5
Regularyzacja L1 (`alpha`)	0
Liczba iteracji boostingowych (`num_boost_round`)	100
Wielkość zbioru treningowego	80%
Wielkość zbioru testowego	20%
Maksymalna liczba cech TF-IDF	10,000
Zakres n-gramów	1-5
Ziarno generatora liczb losowych	1

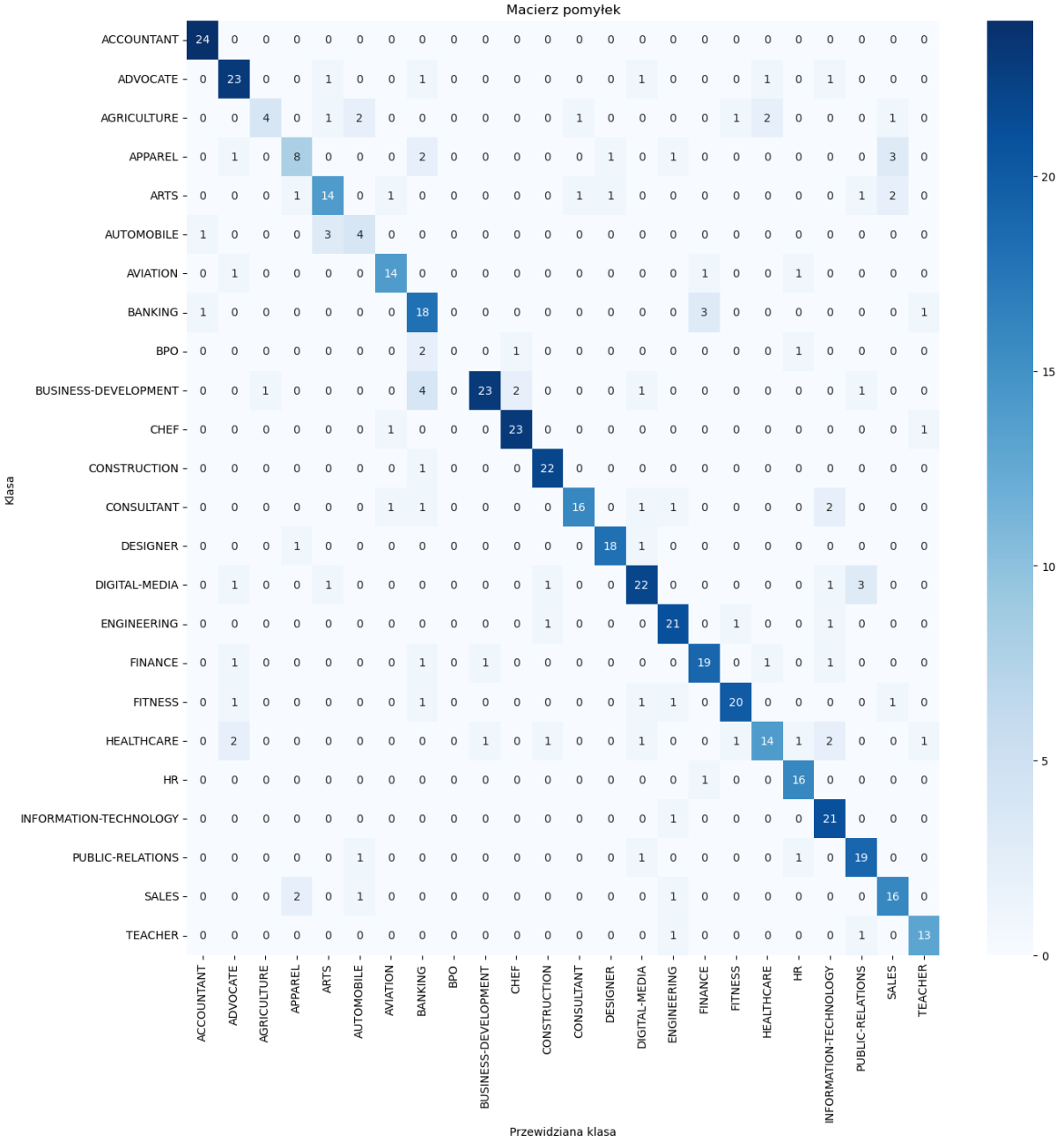
Wyniki modelu, zaprezentowane w tabeli 3.2, pokazują zróżnicowaną skuteczność w klasyfikacji poszczególnych klas zawodowych. Wysokie wartości precyzji i F1-Score dla klas takich jak accountant i construction wskazują na dobrą efektywność modelu w rozpoznawaniu tych kategorii. Z kolei znacznie niższe wyniki dla klas takich jak: agriculture, automobile i bpo sugerują trudności w ich klasyfikacji, co prawdopodobnie wynika z niewielkiej liczby przykładów w zbiorze danych — są to bowiem najrzadziej reprezentowane klasy.

Wartość średnia ważona (ang. weighted average) dla wszystkich miar skuteczności wynosi odpowiednio 0.79 dla precyzji, 0.79 dla recall i 0.78 dla F1-Score, co sugeruje stabilność modelu w klasyfikacji CV, dla różnych kategorii zawodowych. Pomimo zróżnicowanej skuteczności w poszczególnych klasach, średnie wartości wskazują, że model jest względnie spójny i skuteczny w ocenie różnych klas zawodowych.

Tabela 3.2: Wyniki modelu na zbiorze testowym

Klasa	**Precyzja**	**Recall**	**F1-Score**	**Liczność**
ACCOUNTANT	0.92	1.00	0.96	24
ADVOCATE	0.77	0.82	0.79	28
AGRICULTURE	0.80	0.33	0.47	12
APPAREL	0.67	0.50	0.57	16
ARTS	0.70	0.67	0.68	21
AUTOMOBILE	0.50	0.50	0.50	8
AVIATION	0.82	0.82	0.82	17
BANKING	0.58	0.78	0.67	23
BPO	0.00	0.00	0.00	4
BUSINESS-DEVELOPMENT	0.92	0.72	0.81	32
CHEF	0.88	0.92	0.90	25
CONSTRUCTION	0.88	0.96	0.92	23
CONSULTANT	0.89	0.73	0.80	22
DESIGNER	0.90	0.90	0.90	20
DIGITAL-MEDIA	0.76	0.76	0.76	29
ENGINEERING	0.78	0.88	0.82	24
FINANCE	0.79	0.79	0.79	24
FITNESS	0.87	0.80	0.83	25
HEALTHCARE	0.78	0.58	0.67	24
HR	0.80	0.94	0.86	17
INFORMATION-TECHNOLOGY	0.72	0.95	0.82	22
PUBLIC-RELATIONS	0.76	0.86	0.81	22
SALES	0.70	0.80	0.74	20
TEACHER	0.81	0.87	0.84	15
Accuracy	0.79 (dla 497 próbek)			
Macro avg	0.75	0.75	0.74	497
Weighted avg	0.79	0.79	0.78	497

Na poniższym obrazku 3.8 przedstawiono macierz pomyłek dla modelu. Wartości na przekątnej macierzy wskazują poprawne klasyfikacje, natomiast pozostałe wartości reprezentują błędy modelu.



Rysunek 3.8: Macierz pomyłek pokazująca wyniki klasyfikacji.

Jak widać powyżej na rysunku 3.8, macierz pomyłek przedstawia wyniki klasyfikacji dla różnych kategorii zawodów. Na macierzy, na osi pionowej widnieją rzeczywiste klasy, a na osi poziomej przewidziane przez model klasy.

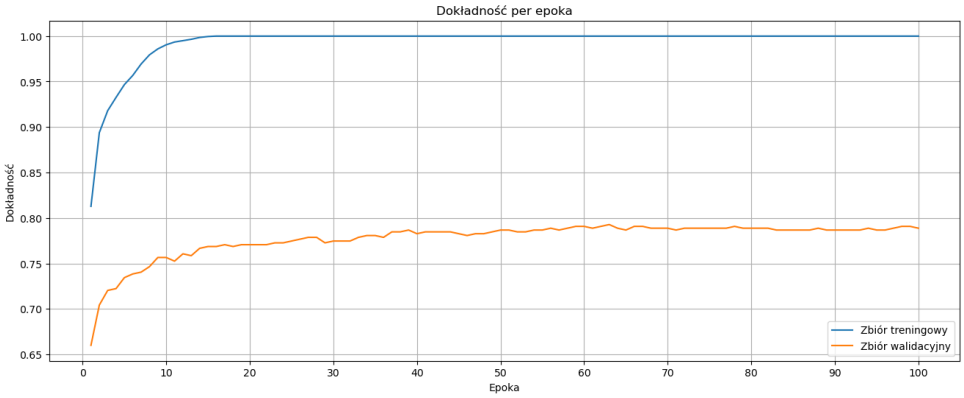
Największą liczbę poprawnych klasyfikacji (elementy na przekątnej) odnotowano dla klasy accountant z 24 prawidłowymi klasyfikacjami. Inne klasy, jak: chef, construction, czy business

development, również uzyskały wysokie wartości poprawnych klasyfikacji, co świadczy o skuteczności modelu w tych kategoriach.

Zauważyć można pewne pomyłki, np. zawód business development został kilkukrotnie sklasyfikowany błędnie jako banking. Podobnie zawód bpo (ang. business process outsourcing) został błędnie przypisany do kategorii banking, a banking jako finance. Wskazuje to na trudności modelu w rozróżnianiu pewnych klas, szczególnie w przypadkach, gdzie zawody mogą mieć cechy wspólne lub podobne.

Poniżej znajduje się rysunek 3.9 przedstawiający dokładność (ang. accuracy) modelu w poszczególnych epokach podczas treningu. Jak pokazuje wykres, dokładność na zbiorze treningowym rośnie szybko w początkowych epokach, osiągając niemal 100% już około 15-tej epoki, po czym utrzymuje się na tym poziomie do końca treningu. Sugeruje to, że model szybko nauczył się rozpoznawać wzorce w zbiorze treningowym.

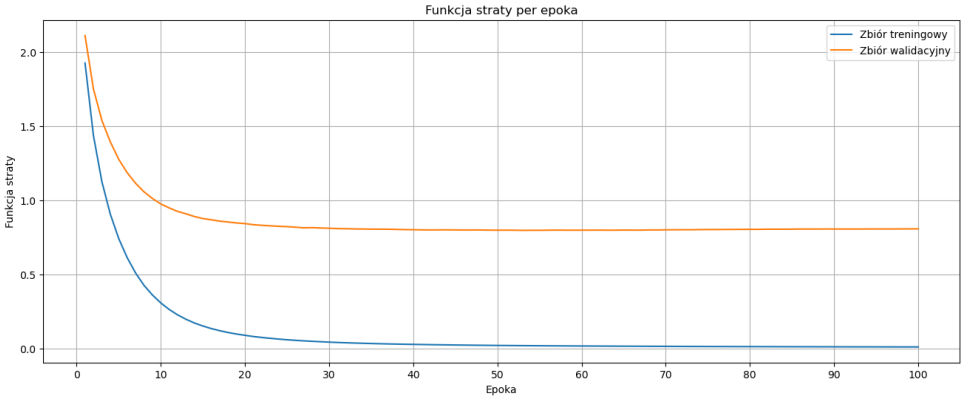
Jednakże dokładność na zbiorze walidacyjnym wykazuje inny trend. Początkowo, podobnie jak w przypadku zbioru treningowego, dokładność rośnie, ale w znacznie wolniejszym tempie. W okolicach 50-tej epoki można zauważyć, że dokładność na zbiorze walidacyjnym zaczyna się stabilizować i oscyluje w granicach 75-80%. To sugeruje, że model mógł ulec przeuczeniu (ang. overfitting), co prowadzi do bardzo wysokiej dokładności na zbiorze treningowym, ale mniejszej skuteczności na nowych, niewidzianych wcześniej danych (zbiór walidacyjny).



Rysunek 3.9: Dokładność per epoka.

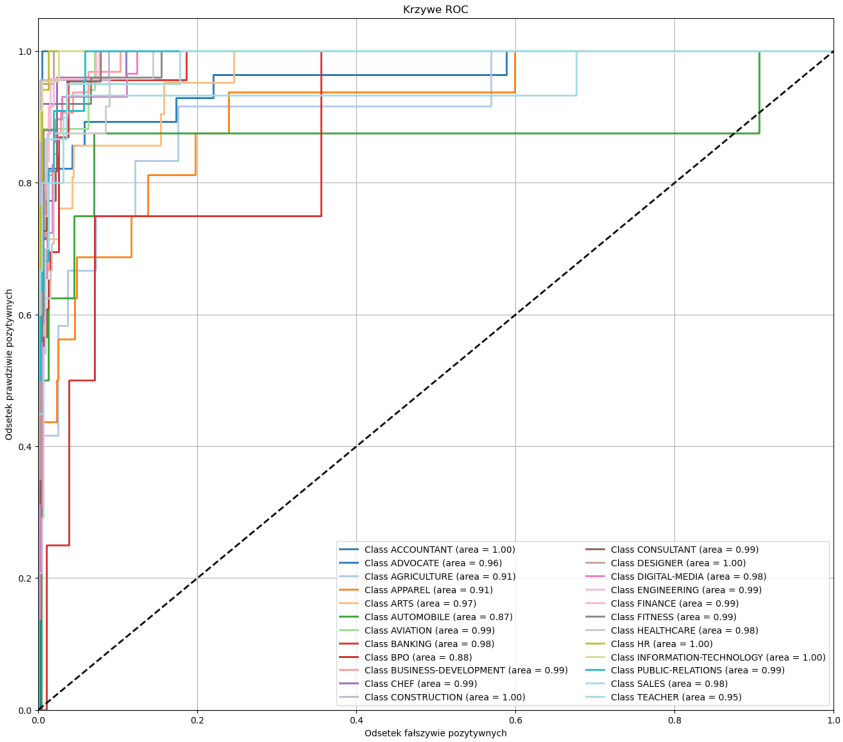
Poniżej znajduje się rysunek 3.10 ilustrujący wartość funkcji straty w poszczególnych epokach treningu. Jak widać na wykresie, wartość funkcji straty dla zbioru treningowego (niebieska linia) gwałtownie spada w początkowych epokach, osiągając bardzo niskie wartości już około 40-tej epoki. W kolejnych epokach funkcja straty w zbiorze treningowym maleje coraz wolniej, aż ostatecznie osiąga wartości bliskie zera, co wskazuje na to, że model bardzo dobrze dopasował się do danych treningowych.

W przypadku zbioru walidacyjnego (pomarańczowa linia) wartość funkcji straty również spada, jednakże w znacznie wolniejszym tempie. Po około 30-tej epoce funkcja straty zaczyna się stabilizować na poziomie wyraźnie wyższym niż w zbiorze treningowym.



Rysunek 3.10: Funkcja straty per epoka.

Krzywe ROC (ang. Receiver Operating Characteristic) przedstawione na rysunku 3.11 poniżej obrazują wydajność klasyfikatora w odniesieniu do wszystkich klas. Wartości pola pod krzywą (ang. Area Under Curve) dla tych klas opisane są w legendzie na wykresie. Każda krzywa ROC ilustruje zależność pomiędzy odsetkiem prawdziwie pozytywnych przypadków (ang. True Positive Rate) a odsetkiem fałszywie pozytywnych przypadków (ang. False Positive Rate).



Rysunek 3.11: Krzywe ROC dla różnych klas.

Klasyfikator o idealnej wydajności miałby krzywą ROC przechodzącą przez punkt $(0, 1)$, co odpowiadałoby wartości AUC równej 1. W analizowanym przypadku najlepsze wartości AUC wyniosły 1 i zostały osiągnięte dla następujących klas: accountant, construction, designer, hr oraz information technology. Taka wartość wskazuje na doskonałą zdolność klasyfikatora do odróżnienia tych klas od pozostałych. Dodatkowo, wartość wskaźnika Gini, wyrażona jako:

$$\text{Gini} = 2 \cdot \text{AUC} - 1$$

gdzie wskaźnik ten przyjmuje wartości od 0 (brak zdolności predykcyjnej) do 1 (doskonała zdolność predykcyjna), w przypadku metody *One-vs-Rest* wyniosła 0.94. Wskaźnik Gini jest miarą nierówności i różnorodności, a w kontekście oceny modeli klasyfikacyjnych odzwierciedla zdolność modelu do poprawnej klasyfikacji przypadków. Wysoka wartość wskaźnika ($Gini = 0.94$) potwierdza dobrą jakość klasyfikatora, podkreślając jego skuteczność w rozróżnianiu klas.

3.4.5. Optymalizacja hiperparametrów

W procesie optymalizacji hiperparametrów dla modelu XGBoost wykorzystano siatkę parametrów zawierającą wszystkie kombinacje poniższych wartości:

Tabela 3.3: Siatka parametrów modelu.

Parametr	**Zakres wartości**
`learning_rate`	[0.05, 0.1, 0.3]
`max_depth`	[6, 8]
`min_child_weight`	[1, 5]
`lambda (Regularyzacja L2)`	[1, 5]
`alpha (Regularyzacja L1)`	[0, 1]
`num_boost_round`	[50, 100]

Łącznie uwzględniono 96 kombinacji hiperparametrów. Dla każdej konfiguracji przeprowadzono trening oraz ewaluację modelu w celu zidentyfikowania najlepszych wartości parametrów w oparciu o stratę oraz dokładność na zbiorze testowym. Wyniki dla 10 najlepszych kombinacji hiperparametrów przedstawiono w tabeli 3.4, a dla 10 najgorszych w tabeli 3.5.

Tabela 3.4: Najlepsze kombinacje hiperparametrów.

learning_rate	max_depth	min_child_weight	lambda	alpha	num_boost_round	Train Loss	Test Loss	Train Acc	Test Acc
0.30	6	1	5	0	100	0.012	0.808	1.000	0.789
0.30	6	1	5	0	50	0.022	0.798	1.000	0.787
0.30	6	1	1	0	50	0.011	0.797	1.000	0.785
0.30	8	5	5	1	50	0.095	0.786	0.999	0.783
0.10	6	5	5	0	100	0.104	0.771	0.999	0.783
0.30	8	1	1	0	100	0.007	0.817	1.000	0.783
0.30	6	1	1	0	100	0.007	0.814	1.000	0.783
0.30	8	5	5	1	100	0.066	0.785	1.000	0.780
0.30	6	1	1	1	50	0.042	0.782	1.000	0.780
0.30	6	1	1	1	100	0.042	0.782	1.000	0.780

Tabela 3.5: Najgorsze kombinacje hiperparametrów.

learning_rate	max_depth	min_child_weight	lambda	alpha	num_boost_round	Train Loss	Test Loss	Train Acc	Test Acc
0.10	8	1	5	1	50	0.474	1.128	0.985	0.736
0.05	8	1	5	1	100	0.167	0.898	0.999	0.740
0.05	8	1	1	1	50	0.345	1.063	0.998	0.744
0.10	8	1	1	1	50	0.104	0.872	1.000	0.744
0.10	6	1	5	0	50	0.122	0.890	1.000	0.744
0.10	6	1	5	0	50	0.132	0.863	1.000	0.746
0.05	8	1	5	0	100	0.124	0.890	1.000	0.744
0.10	8	1	5	0	100	0.063	0.818	1.000	0.744
0.10	6	1	5	0	50	0.165	0.900	0.999	0.746
0.05	6	1	5	0	50	0.500	1.094	0.978	0.744

W procesie optymalizacji zauważono, że niektóre hiperparametry miały znaczący wpływ na wyniki modelu, co zostało określone na podstawie analizy dokładności na zbiorze testowym oraz częstotliwości występowania określonych wartości hiperparametrów w konfiguracjach uznanych za najlepsze i najgorsze. Dokładność na zbiorze testowym była kluczowym wskaźnikiem pozwalającym na ocenę zdolności modelu do generalizacji i przewidywania na danych niewidzianych w procesie treningowym. Natomiast analiza częstotliwości występowania wartości hiperparametrów w topowych oraz najgorszych konfiguracjach umożliwiła identyfikację trendów i zależności między konkretnymi wartościami parametrów a wynikami modelu. Dzięki temu możliwe było wskazanie, które wartości hiperparametrów w największym stopniu wpływały pozytywnie na wydajność modelu, a które prowadziły do pogorszenia rezultatów:

- **Największy pozytywny wpływ:**

  - `learning_rate` = 0.30: Wyższa wartość współczynnika uczenia pozwoliła na szybsze zbieżności i osiągnięcie wyższej dokładności.

  - `max_depth` = 6: Mniejsze głębokości drzewa skutkowały lepszą generalizacją i mniejszą stratą na zbiorze testowym.

- **Największy negatywny wpływ:**

  - `learning_rate` = 0.05/0.1: Niska wartość współczynnika uczenia prowadziła do wolniejszej zbieżności i wyższej straty na zbiorze testowym.

  - `max_depth` = 8: Większa głębokość drzewa zwiększała ryzyko nadmiernego dopasowania, co skutkowało niższą dokładnością na zbiorze testowym.

Powyższe wyniki wskazują, że dla danych tego modelu kluczowe znaczenie ma odpowiednie zbalansowanie szybkości uczenia i głębokości drzewa w celu osiągnięcia optymalnej wydajności. Najwyższą skuteczność zaobserwowano w przypadku modeli charakteryzujących się wyższą szybkością uczenia oraz mniejszą głębokością drzewa.

3.4.6. Porównanie modeli

W ramach oceny efektywności różnych algorytmów klasyfikacyjnych przetestowano kilka popularnych modeli: Las Losowy, Regresję Logistyczną, Maszyny Wektorów Nośnych, Perceptron Wielowarstwowy oraz XGBoost. Modele korzystały z parametrów przedstawionych w tabeli 3.6. Każdy z modeli został oceniony pod kątem dokładności na zbiorze testowym, a uzyskane wyniki zostały zestawione w tabeli 3.7.

Tabela 3.6: Parametry modeli.

Model	**Parametry**
Las Losowy	`n_estimators=100, criterion="gini", max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features="sqrt", bootstrap=True`
Regresja Logistyczna	`penalty="l2", C=1.0, solver="lbfgs", max_iter=100, multi_class="auto"`
Perceptron Wielowarstwowy	`hidden_layer_sizes=(100,), activation="relu", solver="adam", alpha=0.0001, learning_rate="constant", max_iter=200`
Maszyny Wektorów Nośnych	`C=1.0, kernel="rbf", gamma="scale", degree=3, probability=False`
XGBoost	`num_boost_round=100, learning_rate=0.3, min_child_weight=1, max_depth=6, lambda=5, alpha=0, objective="multi:softprob", eval_metric="mlogloss"`

Tabela 3.7: Porównanie dokładności różnych modeli klasyfikacyjnych.

Model	**Accuracy**
XGBoost	0.7887
Las Losowy	0.7324
Regresja Logistyczna	0.6318
Perceptron Wielowarstwowy	0.6318
Maszyny Wektorów Nośnych	0.6217

Wyniki pokazują, że najlepszą skuteczność osiągnął model XGBoost z dokładnością 0.7887, co czyni go najskuteczniejszym w tym zadaniu klasyfikacyjnym. Drugie miejsce zajął Las Losowy, uzyskując wynik 0.7324, co również wskazuje na jego dobrą zdolność generalizacji w analizowanym problemie. Modele oparte na prostszych podejściach, takich jak Regresja Logistyczna (0.6318) oraz Perceptron Wielowarstwowy (0.6318), wykazały porównywalną skuteczność, jednak były wyraźnie gorsze od bardziej zaawansowanych metod. SVM osiągnął najniższą dokładność (0.6217), co sugeruje, że ten algorytm może być mniej efektywny w przypadku analizowanych danych.

3.4.7. Wpływ wstępnego przetwarzania na wyniki

Wstępne przetwarzanie danych miało istotny wpływ na poprawę wyników klasyfikacji. Aby ocenić jego wpływ, porównano dokładność modeli testowanych na danych przed i po wstępnym przetwarzaniu. Wyniki przedstawiono w tabeli 3.8.

Tabela 3.8: Porównanie dokładności modeli ze wstępnym przetwarzaniem i bez.

Model	**Bez przetwarzania**	**Z przetwarzaniem**
XGBoost	0.7606	0.7887
Las Losowy	0.6982	0.7324
Regresja Logistyczna	0.6217	0.6318
Perceptron Wielowarstwowy	0.6258	0.6318
Maszyny Wektorów Nośnych	0.5996	0.6217

Zestawienie pokazuje, że wprowadzenie wstępnego przetwarzania danych poprawiło dokładność wszystkich testowanych modeli. Największy wzrost dokładności zaobserwowano dla Lasu Losowego, który zwiększył swój wynik z 0.6982 do 0.7324. XGBoost również wykazał znaczącą poprawę, osiągając wzrost z 0.7606 do 0.7887. Modele prostsze, takie jak Regresja Logistyczna i Perceptron Wielowarstwowy, odnotowały mniejsze, ale wciąż istotne wzrosty, wynoszące odpowienio 0.0101 i 0.006. Maszyny Wektorów Nośnych, mimo stosunkowo niskiej początkowej dokładności, również skorzystały na wstępnym przetwarzaniu, co przełożyło się na wzrost wyniku z 0.5996 do 0.6217.

Powyższe wyniki potwierdzają, że odpowiednie przekształcenie danych wejściowych poprzez wstępne przetwarzanie jest kluczowe dla zwiększenia skuteczności klasyfikacji. Działania takie jak oczyszczanie tekstu, usuwanie stop words, lematyzacja oraz wykorzystanie TF-IDF pozwalają na lepsze uogólnienie danych i wydobycie istotnych cech, co skutkuje lepszym dopasowaniem modeli do zadania klasyfikacyjnego.

3.5. Ranking CV

W tym rozdziale zaprezentowany zostanie proces tworzenia rankingu CV. Na początku omówiony zostanie zbiór danych, który stanowił podstawę przeprowadzonych analiz. Następnie szczegółowo opisane zostaną techniki wstępnego przetwarzania tekstu, które były kluczowe dla przygotowania danych wejściowych do modelu. W dalszej części przedstawiona zostanie architektura modelu, służącego do oceny CV w kontekście wybranej oferty pracy. Po tym zaprezentowane

zostaną wyniki działania modelu, które umożliwią ocenę jego skuteczności. Na koniec omówiony zostanie proces optymalizacji hiperparametrów oraz przeprowadzona analiza porównawcza z innymi modelami.

3.5.1. Zbiór danych

Zbiór danych użyty w tym projekcie pochodzi z platformy Hugging Face i jest dostępny pod adresem: `https://huggingface.co/datasets/cnamuangtoun/resume-job-description-fit`. Zbiór ten zawiera 8000 wierszy, co zapewnia wystarczającą ilość danych do przeprowadzenia analizy i trenowania modeli uczenia maszynowego. Każdy wiersz reprezentuje pojedynczy rekord łączący CV oraz ofertę pracy, wraz z etykietą określającą wynik wstępnego przeglądu CV.

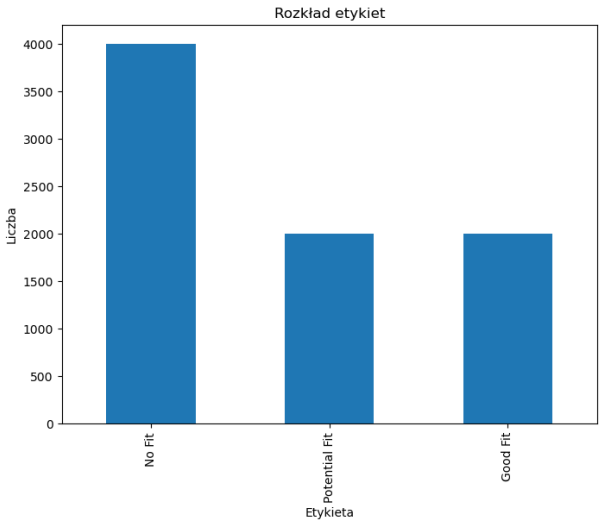
Każdy wiersz w zbiorze danych zawiera trzy kolumny:

- `resume_text` - tekst CV kandydata,
- `job_description_text` - tekst oferty pracy,
- `label` - wynik wstępnego przeglądu CV, który przyjmuje jedną z trzech wartości: no fit, potential fit, good fit.

Kolumna `label` zawiera wynik wstępnego screeningu, który określa, jak dobrze CV pasuje do opisu oferty pracy. Etykieta ta może przyjmować trzy wartości:

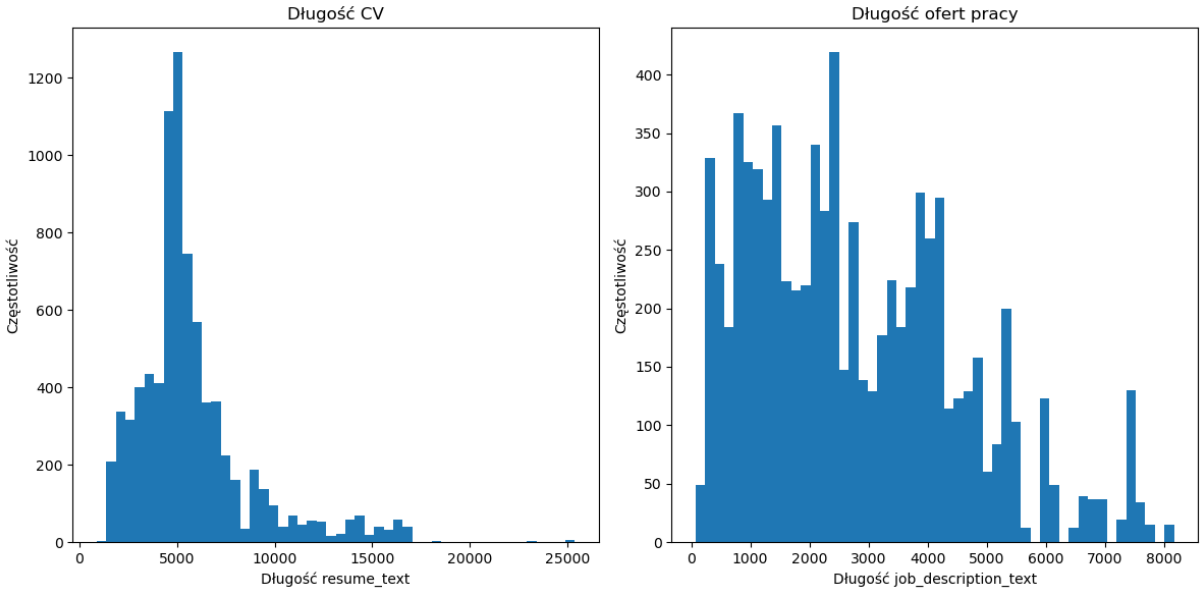
- no fit - CV nie odpowiada wymaganiom oferty pracy,
- potential fit - CV częściowo odpowiada wymaganiom oferty pracy, istnieje potencjalna zgodność,
- good fit - CV odpowiada wymaganiom oferty pracy.

Na rysunku 3.12 przedstawiono podstawowe statystyki dotyczące rozkładu etykiet w zbiorze danych:



Rysunek 3.12: Rozkład etykiet w zbiorze danych.

Rysunek 3.13 przedstawiaja rozkład długości CV oraz ofert pracy w analizowanym zbiorze danych. Wykres po lewej stronie pokazuje, że większość CV ma długość od 1000 do 5000 znaków, natomiast wykres po prawej stronie ilustruje, że długości ofert pracy są bardziej rozproszone, z wartościami rozciągającymi się od około 500 do 8000 znaków. Te różnice mogą wskazywać na różne standardy i praktyki w pisaniu CV oraz ofert pracy.



Rysunek 3.13: Długość CV, ofert pracy.

Aby lepiej zrozumieć charakterystykę tekstów zawartych zarówno w CV, jak i w ofertach pracy, wygenerowano osobne wizualizacje najczęściej występujących słowów w CV oraz ogłoszeniach o pracę, przy czym większe rozmiary czcionek odpowiadają wyższym częstotliwościom występowania danego słowa.

Rysunek 3.14 prezentuje najczęściej używane słowa w tekstach CV. Wśród nich dominują terminy techniczne i zawodowe, takie jak SQL, Server, Test Case, Team Member, Data Analysis oraz Software Development. Ich obecność sugeruje, że kandydaci często podkreślają swoje doświadczenie techniczne oraz kompetencje związane z analizą danych i programowaniem.

Rysunek 3.14: Najczęściej występujące słowa w tekstach CV.

Z kolei rysunek 3.15 przedstawia najczęściej występujące słowa w opisach ofert pracy.
W tym przypadku widoczne są takie terminy, jak Software Development, Ability, Team Member, Business Analyst, Experience, Looking, Computer Science oraz Software Engineer.
Słowa te odzwierciedlają wymagania stawiane kandydatom oraz typowe kwalifikacje poszukiwane przez pracodawców.

Rysunek 3.15: Najczęściej występujące słowa w opisach ofert pracy.

Porównanie obu wizualizacji ujawnia pewne podobieństwa, takie jak powtarzające się słowa związane z technologią i współpracą zespołową (Software Development, Team Member). Jednocześnie widać różnice, ponieważ CV częściej akcentują specyficzne umiejętności techniczne (SQL, Server, Test Case), z kolei oferty pracy kładą większy nacisk na ogólne pojęcia związane z rekrutacją i rolą zawodową, takie jak Ability, Experience czy Looking, które odzwierciedlają sposób komunikacji oczekiwań przez pracodawców. Warto również zauważyć, że w tym zbiorze

danych teksty CV są bardziej techniczne i szczegółowe w porównaniu do poprzedniego użytego do klasyfikacji CV, co wskazuje na większe skupienie kandydatów na podkreślaniu konkretnych umiejętności oraz doświadczeń zawodowych.

3.5.2. Wstępne przetwarzanie

Wstępne przetwarzanie danych w jest procesem analogicznym do tego, który jest stosowany w klasyfikacji CV, z jedną różnicą dotyczącą sposobu reprezentacji tekstu. Po przeprowadzeniu standardowych kroków przetwarzania tekstu, takich jak oczyszczenie z nieliterowych znaków, konwersja na małe litery, usunięcie stop words oraz lematyzacja, teksty CV są przekształcane do formy numerycznej, która może być bezpośrednio użyta przez modele uczenia maszynowego.

W przypadku rankingu CV zastosowany jest inny wektoryzator niż w klasyfikacji. Zamiast TF-IDF, który mierzy częstość występowania słów z uwzględnieniem ich unikalności, wykorzystywana jest warstwa wektoryzacji tekstu oparta na tokenizacji i konwersji słów do sekwencji liczbowych. Wektoryzator ten przekształca tekst do jednorodnych sekwencji o stałej długości, które mogą być bezpośrednio podane na wejście sieci neuronowej.

Przeprowadzone wstępne przetwarzanie danych, dostosowane do wymagań modelu rankingu CV, odegrało ważną rolę w poprawie wyników klasyfikacji. Standardowe operacje przetwarzania tekstu, takie jak oczyszczanie, usuwanie stop words oraz lematyzacja, skutecznie zredukowały szum w danych. Dzięki temu model lepiej radził sobie z różnorodnymi tekstami CV, co bezpośrednio przełożyło się na wzrost skuteczności klasyfikacji, co zostało opisane w rozdziale 3.5.7.

3.5.3. Architektura sieci

Sieć neuronowa składa się z następujących komponentów:

1. **Wejście:** Model przyjmuje dwa różne wejścia tekstowe:
   - `resume_text`: tekst CV.
   - `job_description_text`: tekst opisu stanowiska.
2. **Warstwa wektoryzacji tekstu:** Oba wejścia są przekształcane przez warstwę `TextVectorization`, która zamienia tekst na sekwencje liczb całkowitych reprezentujących indeksy słów w słowniku o rozmiarze 10 000. Sekwencje te są następnie przycinane lub wypełniane do długości 100 tokenów.
3. **Warstwa osadzeń (*Embedding*):** Wektoryzowane sekwencje są dalej przekształcane za pomocą warstwy `Embedding`, która mapuje każde słowo na wektor o wymiarze 128.

4. **Warstwa LSTM:** Oba przekształcone wejścia przechodzą przez wspólną warstwę `LSTM` o 64 jednostkach, która zwraca pełne sekwencje.

5. **Agregacja:** Wyjścia z warstwy LSTM są sumowane za pomocą warstwy `Lambda`, co daje wektor o wymiarze 64 dla każdego z dwóch wejść.

6. **Normalizacja:** Każdy z agregowanych wektorów jest następnie normalizowany za pomocą warstwy `Lambda`, która używa normalizacji L2, aby zapewnić, że wektory mają jednostkową długość.

7. **Podobieństwo cosinusowe:** Warstwa `Lambda` oblicza podobieństwo cosinusowe pomiędzy dwoma znormalizowanymi wektorami.

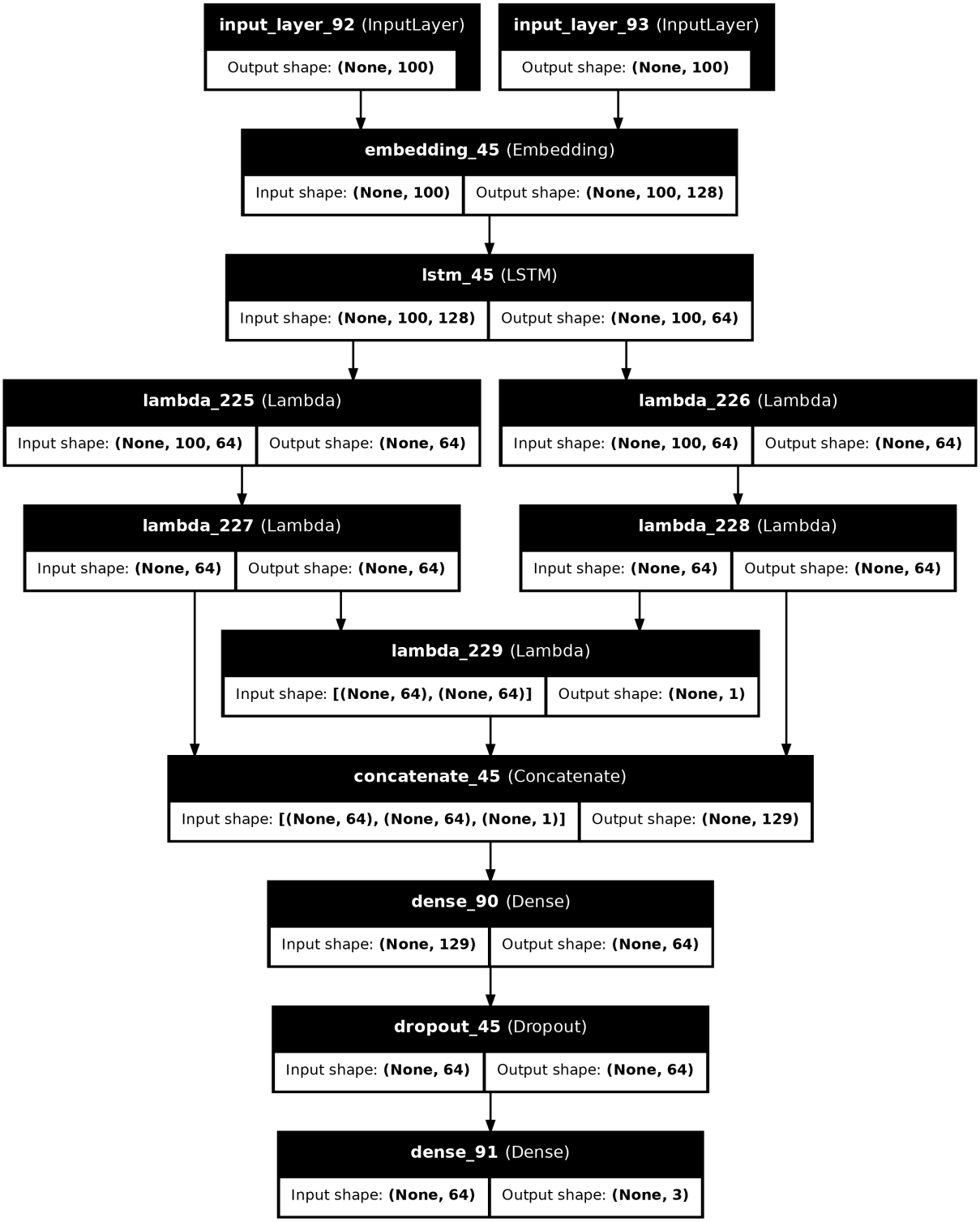
8. **Połączenie:** Trzy wektory (dwa znormalizowane wektory oraz ich podobieństwo cosinusowe) są łączone za pomocą warstwy `Concatenate`, tworząc wektor o wymiarze 129.

9. **Warstwa gęsta:** Połączony wektor przechodzi przez gęstą warstwę (`Dense`) o 64 jednostkach z funkcją aktywacji `ReLU` oraz regularizacją $L2$ z parametrem $\lambda = 0.001$.

10. **Warstwa dropout:** Następnie stosowana jest warstwa `Dropout` z prawdopodobieństwem 0.4, aby zapobiec przeuczeniu.

11. **Wyjście:** Ostatecznie wektor przechodzi przez gęstą warstwę wyjściową (`Dense`) o 3 jednostkach z funkcją aktywacji `softmax`, co daje rozkład prawdopodobieństw dla trzech możliwych klas.

Model jest trenowany za pomocą optymalizatora `Adam` ze współczynnikiem uczenia 0.001 oraz funkcji straty `sparse_categorical_crossentropy`. Wizualizacja modelu znajduje się na rysunku 3.16 poniżej.



Rysunek 3.16: Architektura modelu do rankingu CV.

3.5.4. Wyniki

Poniższe wyniki zostały uzyskane przy wykorzystaniu wybranych parametrów modelu oraz konfiguracji, które zostały zoptymalizowane w procesie eksperymentów. Dokładny opis procesu optymalizacji hiperparametrów został opisany w sekcji 3.5.5. Kluczowe parametry modelu oraz konfiguracja są przedstawione w tabeli 3.9.

Tabela 3.9: Parametry modelu i konfiguracja

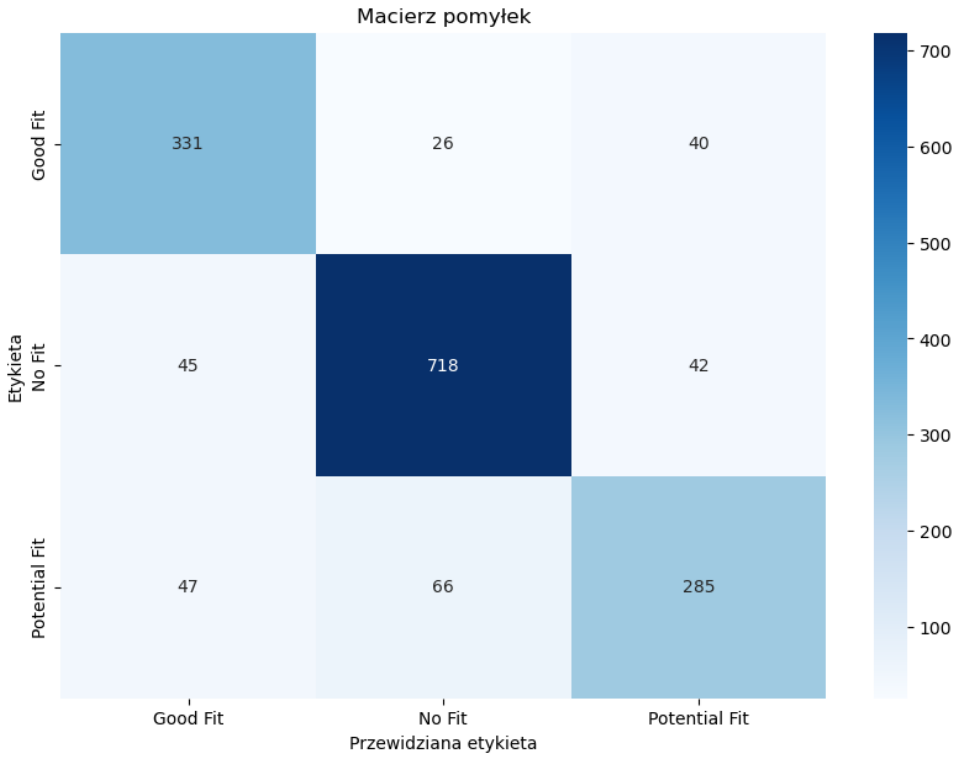
Parametr	**Wartość**
Rozmiar warstwy embedding (`embedding_dim`)	128
Liczba jednostek LSTM (`lstm_units`)	64
Wartość dropout (`dropout_rate`)	0,4
Współczynnik uczenia (`learning_rate`)	0.001
Liczba epok (`epochs`)	15
Rozmiar batcha (`batch_size`)	8
Wielkość zbioru treningowego	80%
Wielkość zbioru testowego	20%
Rozmiar słownika	10,000
Maksymalna długość sekwencji	100
Rozmiar warstwy Dense	64
Funkcja aktywacji w warstwie Dense	ReLU
Funkcja aktywacji w warstwie wyjściowej	Softmax
Funkcja straty	Sparse Categorical Crossentropy
Optymalizator	Adam
Podział na walidację	20%
Wartość regularizacji L2	0.001
Ziarno generatora liczb losowych	1

Wyniki modelu, przedstawione w tabeli 3.10, wskazują, że najlepszą skuteczność osiągnięto dla klasy no fit, z wartością f1-score równą 0.89. Klasy good fit i potential fit uzyskały zbliżone wyniki f1-score, odpowiednio 0.81 i 0.75. Wartość średnia ważona (ang. weighted average) dla wszystkich miar skuteczności również wynosi 0.83, co sugeruje stabilność modelu w klasyfikacji różnych typów dopasowania CV.

Tabela 3.10: Wyniki modelu na zbiorze testowym

Klasa	**Precyzja (Precision)**	**Recall**	**F1-Score**	**Liczność (Support)**
Good Fit	0.78	0.83	0.81	397
No Fit	0.89	0.89	0.89	805
Potential Fit	0.78	0.72	0.75	398
Accuracy	0.83 (dla 1600 próbek)			
Macro avg	0.82	0.81	0.81	1600
Weighted avg	0.83	0.83	0.83	1600

Na poniższym obrazku 3.17 przedstawiono macierz pomyłek dla modelu.

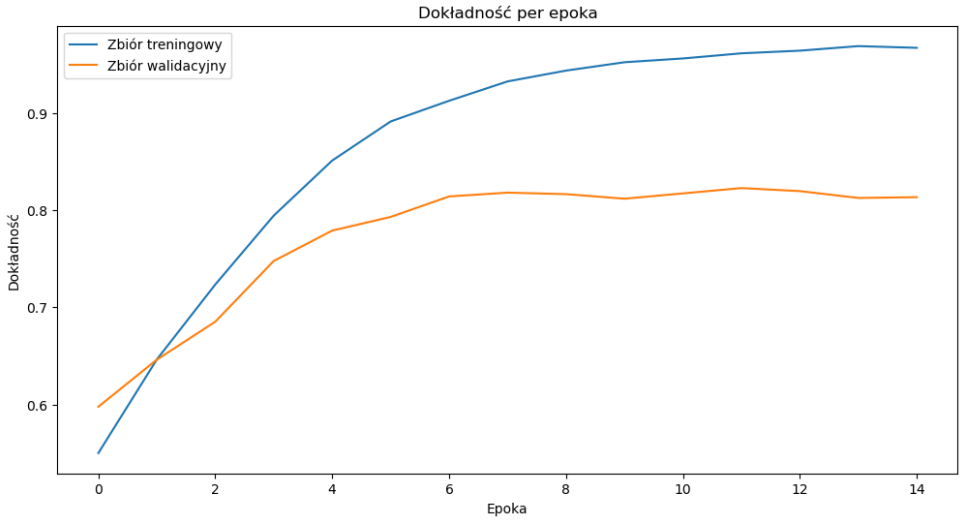


Rysunek 3.17: Macierz pomyłek pokazująca wyniki klasyfikacji.

Jak widać powyżej 3.17, model najczęściej prawidłowo klasyfikuje etykiety jako no fit z najmniejszą liczbą pomyłek w stosunku do liczby przykładów, co stanowi największą część poprawnych przewidywań. Model ma również dobre wyniki w kategoriach good fit i potential fit, aczkolwiek występują w tych kategoriach pewne pomyłki.

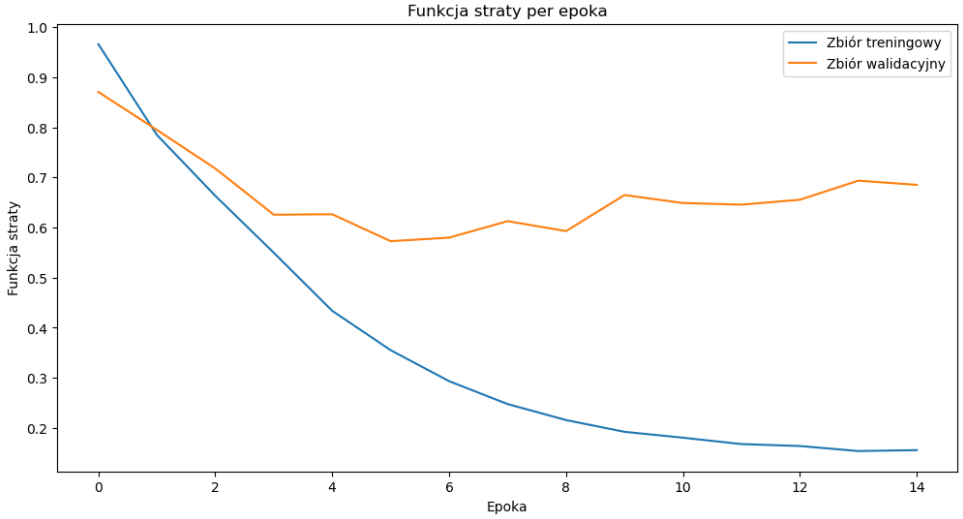
Poniżej znajduje się wykres 3.18 przedstawiający dokładność (ang. accuracy) modelu w poszczególnych epokach podczas treningu. Jak pokazuje wykres, dokładność modelu na zbiorze walidacyjnym (stanowiącym 20% zbioru treningowego) osiągnęła wysoką wartość w 6 epoce,

a później nieznacznie się zmieniała do końca. Dokładność na zbiorze treningowym była o około 15% wyższa niż na zbiorze walidacyjnym.



Rysunek 3.18: Dokładność per epoka.

Poniżej znajduje się wykres 3.19 ilustrujący wartość funkcji straty w poszczególnych epokach treningu. Jak widać na wykresie, funkcja straty osiągnęła najniższą wartość dla zbioru walidacyjnego w 6 epoce. Wartość funkcji straty na zbiorze walidacyjnym była o około 0.5 wyższa niż na zbiorze treningowym, co sugeruje lepszą efektywność modelu na danych treningowych w porównaniu do danych walidacyjnych.

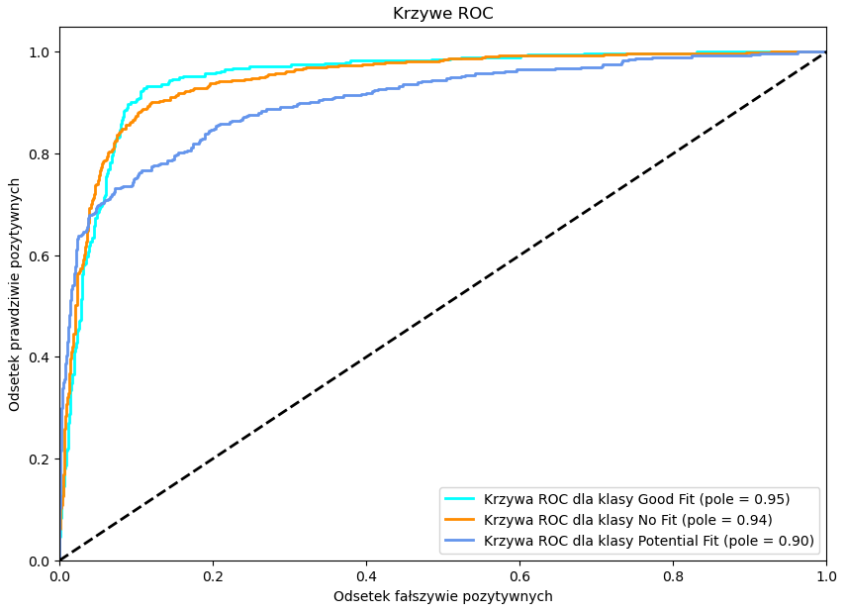


Rysunek 3.19: Funkcja straty per epoka.

Krzywe ROC (ang. Receiver Operating Characteristic), przedstawione na Rysunku 3.20, obrazują jakość klasyfikatora w odniesieniu do klas: good fit, no fit oraz potential fit. Pole pod krzywą ROC (ang. Area Under Curve) dla klas good Fit, no fit oraz potential fit wynoszą odpowiednio

0.95, 0.94 oraz 0.90. Wysoka wartość AUC dla klasy good fit sugeruje, że model bardzo dobrze rozpoznaje tę klasę. Klasy no fit i potential fit mają nieco niższe, lecz wciąż wysokie wartości AUC, co potwierdza skuteczność modelu również dla tych klas.

Dodatkowo, wartość wskaźnika Gini, dla metody *One-vs-rest* wyniosła 0.865. Jest to stosunkowo wysoka wartość, co potwierdza dobrą jakość klasyfikatora i jego zdolność do efektywnego rozróżniania pomiędzy klasami.



Rysunek 3.20: Krzywe ROC dla różnych klas.

3.5.5. Optymalizacja hiperparametrów

W procesie optymalizacji hiperparametrów dla sieci LSTM wykorzystano siatkę parametrów zawierającą wszystkie kombinacje poniższych wartości:

Tabela 3.11: Siatka parametrów.

Parametr	**Zakres wartości**
`learning_rate`	[0.001, 0.01, 0.1]
`batch_size`	[8, 16]
`dropout_rate`	[0.3, 0.4]
`lstm_units`	[64, 128]
`epochs`	[10, 15]
`embedding_dim`	[64, 128]

Łącznie uwzględniono 96 kombinacji hiperparametrów. Dla każdej konfiguracji przeprowadzono trening oraz ewaluację modelu w celu zidentyfikowania najlepszych wartości parametrów w oparciu o stratę oraz dokładność na zbiorze testowym. Wyniki dla 10 najlepszych kombinacji hiperparametrów przedstawiono w tabeli 3.12, a dla 10 najgorszych w tabeli 3.13.

Tabela 3.12: Najlepsze kombinacje hiperparametrów.

learning_rate	batch_size	dropout_rate	lstm_units	epochs	embedding_dim	Train Loss	Test Loss	Train Acc	Test Acc
0.001	8	0.4	64	15	128	0.156	0.620	0.967	0.834
0.001	8	0.3	64	10	128	0.220	0.581	0.947	0.828
0.001	8	0.3	64	15	64	0.175	0.674	0.961	0.821
0.001	16	0.3	128	10	128	0.261	0.568	0.934	0.819
0.001	8	0.3	64	10	64	0.228	0.603	0.943	0.818
0.001	8	0.4	128	15	64	0.183	0.680	0.958	0.814
0.001	16	0.4	64	15	64	0.162	0.696	0.965	0.811
0.001	8	0.3	128	10	64	0.261	0.629	0.925	0.810
0.001	8	0.4	128	10	128	0.289	0.585	0.918	0.810
0.001	8	0.4	128	15	128	0.166	0.671	0.965	0.806

Tabela 3.13: Najgorsze kombinacje hiperparametrów.

learning_rate	batch_size	dropout_rate	lstm_units	epochs	embedding_dim	Train Loss	Test Loss	Train Acc	Test Acc
0.100	8	0.400	128	15	128	1.051	1.043	0.501	0.503
0.010	8	0.300	128	15	64	1.041	1.038	0.501	0.503
0.100	16	0.300	64	10	64	1.048	1.042	0.501	0.503
0.100	16	0.300	128	10	64	1.048	1.042	0.501	0.503
0.100	8	0.300	128	10	128	1.051	1.043	0.501	0.503
0.010	16	0.300	128	15	64	1.040	1.038	0.501	0.503
0.100	16	0.300	64	15	64	1.048	1.042	0.501	0.503
0.100	16	0.300	128	15	64	1.048	1.042	0.501	0.503
0.100	16	0.300	64	10	128	1.048	1.042	0.501	0.503
0.100	16	0.400	128	15	128	1.048	1.042	0.501	0.503

W procesie optymalizacji zauważono, że niektóre hiperparametry miały znaczący wpływ na wyniki modelu, co określono na podstawie dokładności na zbiorze testowym oraz częstotliwości występowania wartości hiperparametrów w najlepszych/w najgorszych wynikach.

- **Największy pozytywny wpływ:**
  - `learning_rate` = 0.001: Niska wartość współczynnika uczenia pozwoliła na bardziej stabilne dopasowanie modelu, skutkując niższą stratą i wyższą dokładnością na zbiorze testowym.
  - `batch_size` = 8: Mniejszy rozmiar batcha sprzyjał dokładniejszemu dostosowaniu gradientów, co pozytywnie wpłynęło na wyniki modelu.
- **Największy negatywny wpływ:**
  - `learning_rate` = 0.1: Wyższa wartość współczynnika uczenia prowadziła do problemów z konwergencją i wysokiej straty na zbiorze testowym.
  - `batch_size` = 16: Wyższy rozmiar batcha negatywnie wpływał na precyzję dopasowania modelu, skutkując gorszymi wynikami.

Powyższe wyniki wskazują, że kluczowym czynnikiem wpływającym na wydajność modelu jest odpowiednie połączenie szybkości uczenia i rozmiaru batcha. Najlepsze rezultaty uzyskano przy `learning_rate` = 0.100 i `batch_size` = 8, co zapewniło szybszą konwergencję i wyższą dokładność. Z kolei większy rozmiar batcha (`batch_size` = 16) w połączeniu z wyższym `learning_rate` pogorszył wyniki.

3.5.6. Porównanie modeli

W ramach oceny efektywności różnych algorytmów klasyfikacyjnych przetestowano kilka popularnych modeli: Las Losowy, Regresję Logistyczną, Maszyny Wektorów Nośnych, Perceptron Wielowarstwowy, XGBoost oraz autorski model LSTM. Modele korzystały z parametrów przedstawionych w tabeli 3.14. Każdy z modeli został oceniony pod kątem dokładności na zbiorze testowym, a uzyskane wyniki zostały zestawione w tabeli 3.15.

Tabela 3.14: Parametry modeli.

Model	**Parametry**
Las Losowy	`n_estimators=100, criterion="gini", max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features="sqrt", bootstrap=True`
Regresja Logistyczna	`penalty="l2", C=1.0, solver="lbfgs", max_iter=100, multi_class="auto"`
Perceptron Wielowarstwowy	`hidden_layer_sizes=(100,), activation="relu", solver="adam", alpha=0.0001, learning_rate="constant", max_iter=200`
Maszyny Wektorów Nośnych	`C=1.0, kernel="rbf", gamma="scale", degree=3, probability=False`
XGBoost	`num_boost_round=100, learning_rate=0.3, min_child_weight=1, max_depth=6, lambda=5, alpha=0, objective="multi:softprob", eval_metric="mlogloss"`
LSTM	`learning_rate=0.001, batch_size=8, dropout_rate=0.4, lstm_units=64, epochs=15, embedding_dim=128`

Tabela 3.15: Porównanie accuracy różnych modeli klasyfikacyjnych.

Model	**Dokładność**
LSTM	0.8338
XGBoost	0.6913
Las Losowy	0.6656
Maszyny Wektorów Nośnych	0.5975
Perceptron Wielowarstwowy	0.5269
Regresja Logistyczna	0.5238

Najlepszy wynik uzyskany został przez sieć opartą na LSTM'ie, który osiągnął dokładność na poziomie 0.8338. Kolejnym najlepszym modelem okazał się XGBoost z wynikiem 0.6913, co wskazuje na jego dobrą skuteczność w zadaniach klasyfikacyjnych. Pozostałe modele, takie

jak Las Losowy (0.6656) oraz Maszyny Wektorów Nośnych (SVM) (0.5975), także osiągnęły przyzwoite wyniki, ale znacznie gorsze niż LSTM. Modele oparte na prostszych technikach, takie jak Perceptron Wielowarstwowy (MLP) (0.5269) oraz Regresja Logistyczna (0.5238), osiągnęły najniższe wartości dokładności, co sugeruje, że bardziej złożone architektury, jak LSTM, radzą sobie lepiej z analizą danych tekstowych w tym przypadku.

3.5.7. Wpływ wstępnego przetwarzania na wyniki

Wstępne przetwarzanie danych miało istotny wpływ na poprawę wyników klasyfikacji. Aby ocenić jego wpływ, porównano dokładność modeli testowanych na danych przed i po wstępnym przetwarzaniu. Wyniki przedstawiono w tabeli 3.16.

Tabela 3.16: Porównanie dokładności modeli w rankingu CV przed i po wstępnym przetwarzaniu.

Model	**Bez przetwarzania**	**Z przetwarzaniem**
LSTM	0.7869	0.8338
XGBoost	0.6813	0.6913
Las Losowy	0.6594	0.6656
Maszyny Wektorów Nośnych	0.6006	0.5975
Perceptron Wielowarstwowy	0.5238	0.5269
Regresja Logistyczna	0.5312	0.5238

Wyniki wskazują, że proces wstępnego przetwarzania danych znacząco poprawił wyniki najbardziej zaawansowanego modelu, LSTM, który osiągnął wzrost dokładności z 0.7869 do 0.8338, co stanowi największą poprawę w zestawieniu. XGBoost również zyskał na wstępnym przetwarzaniu, zwiększając dokładność z 0.6813 do 0.6913. Las Losowy zanotował mniejszy wzrost, z 0.6594 do 0.6656, co wskazuje na ograniczoną, choć widoczną poprawę wyników. Prostsze modele, takie jak Perceptron Wielowarstwowy i Regresja Logistyczna, wykazały minimalne zmiany dokładności – odpowiednio wzrost o 0.0031 oraz spadek o 0.0074. Co ciekawe, Maszyny Wektorów Nośnych odnotowały niewielki spadek z 0.6006 do 0.5975.

Analiza wyników wskazuje, że wstępne przetwarzanie, obejmujące operacje takie jak oczyszczanie tekstu, usuwanie stop words oraz lematyzacja, ma kluczowe znaczenie dla modeli głębokiego uczenia, takich jak LSTM, oraz wspiera bardziej zaawansowane algorytmy klasyczne, jak XGBoost. Jednak jego wpływ na prostsze modele może być ograniczony lub neutralny w kontekście oceny CV.

4. Wnioski

Z przeprowadzonych badań oraz eksperymentów wynika, że zaprojektowany system automatycznego przeglądu kandydatów aplikujących na oferty pracy, oparty na nowoczesnych technologiach przetwarzania języka naturalnego oraz uczenia maszynowego, może znacząco usprawnić proces rekrutacyjny. W szczególności, wdrożenie algorytmów klasyfikacji, oceny dopasowania kandydatów do ofert pracy oraz podsumowywania CV wykazało potencjał w zakresie automatyzacji oraz optymalizacji tego procesu.

Zastosowany model XGBoost w zadaniu klasyfikacji CV uzyskał 79% dokładności na zbiorze testowym, co jest wynikiem zbliżonym do najlepszych wartości osiąganych na tym zbiorze danych. Najlepsze modele, osiągające 85% dokładności na tym zbiorze, wykorzystywały jednak pretrenowane transformery do wektoryzacji. Najlepsze modele, które nie korzystały z pretrenowanych transformerów, osiągały około 80% dokładności, co dodatkowo świadczy o wysokiej efektywności zastosowanego rozwiązania. Wysoką skuteczność odnotowano w przypadku klas takich jak accountant oraz construction, gdzie precyzja oraz miary F1-Score były na poziomie ponad 0.9. Z kolei niższe wyniki, takie jak dla klas agriculture, automobile i bpo, sugerują, że model miał trudności z klasyfikowaniem zawodów o mniejszej liczbie próbek w zbiorze danych.

W przypadku modelu bazującego na LSTM, który służył do rankingu kandydatów według dopasowania do ofert pracy, uzyskano dokładność na poziomie 83%. Analiza wyników wskazuje, że model ten wypadł najlepiej na tle innych popularnych metod stosowanych w badaniu, takich jak XGBoost, Las Losowy czy Maszyny Wektorów Nośnych. Chociaż nie znaleziono innych wyników na tym samym zbiorze danych, uzyskane rezultaty potwierdzają wysoką skuteczność zaproponowanego rozwiązania. Model osiągnął najlepsze wyniki w klasyfikacji kandydatów do klasy no fit, z wysoką wartością F1-Score wynoszącą 0.89. Dla klas good fit i potential fit wartości F1-Score wyniosły odpowiednio 0.81 i 0.75, co również wskazuje na dobrą skuteczność modelu w rozróżnianiu kandydatów o różnych poziomach dopasowania do ofert pracy. Podczas analizy macierzy pomyłek zauważono, że model najlepiej radził sobie z klasyfikowaniem kandydatów do klasy no fit, co jest szczególnie istotne w kontekście efektywności systemu rekrutacyjnego. Algorytm skutecznie identyfikuje kandydatów, którzy w ogóle nie spełniają wymagań stanowiska, co pozwala na ich łatwe odrzucenie. Tego rodzaju filtracja jest kluczowa w systemach rekruta-

cyjnych, ponieważ umożliwia szybkie wykluczenie osób, które nie mają szans na dalszy etap rekrutacji, oszczędzając czas zarówno rekruterom, jak i kandydatom.

Optymalizacja hiperparametrów miała duże znaczenie dla uzyskania najlepszych wyników. W przypadku modelu XGBoost zauważono, że najlepszą skuteczność uzyskano przy wyższej wartości współczynnika uczenia (`learning_rate` = 0.3) oraz mniejszej głębokości drzewa (`max_depth` = 6), co pozwoliło na osiągnięcie lepszej generalizacji modelu. Z kolei w przypadku modelu bazującego na LSTM najważniejszymi parametrami okazały się wartość `learning_rate` = 0.001 oraz mniejszy rozmiar batcha (`batch_size` = 8), co zapewniło bardziej stabilne dopasowanie i lepsze wyniki na zbiorze testowym.

Wpływ wstępnego przetwarzania danych na wyniki eksperymentów był znaczący zarówno dla klasyfikacji CV, jak i rankingu kandydatów. W obu przypadkach zastosowanie technik takich jak oczyszczanie tekstu, usuwanie stop words czy lematyzacja pozwoliło na wydobycie istotnych cech z danych wejściowych, co przełożyło się na zauważalne poprawy dokładności modeli. Modele bardziej zaawansowane, takie jak XGBoost, Las Losowy w klasyfikacji czy LSTM w rankingu, wykazały największe korzyści, co potwierdza kluczowe znaczenie jakości danych wejściowych dla bardziej skomplikowanych algorytmów. Również prostsze modele, mimo ograniczonego zakresu poprawy wyników, zyskały na wstępnym przetwarzaniu, choć jego wpływ był mniej znaczący lub neutralny w kontekście bardziej podstawowych algorytmów.

Opracowany system stanowi skuteczne i nowoczesne narzędzie wspierające procesy rekrutacyjne. Uzyskane wyniki potwierdzają, że może on znacząco przyczynić się do automatyzacji i usprawnienia selekcji kandydatów, co przekłada się na oszczędność czasu oraz zwiększenie efektywności pracy rekruterów. Dzięki zastosowaniu zaawansowanych metod uczenia maszynowego oraz przetwarzania języka naturalnego, system wyróżnia się wysoką skutecznością i może realnie wspierać podejmowanie decyzji w procesie rekrutacji.

Bibliografia

[1] Berke Akkaya, Ersin Sener, and Cem Gursu. A comparative study of heart disease prediction using machine learning techniques. In *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–8, 2022.

[2] Vimala Balakrishnan and Lloyd-Yemoh Ethel. Stemming and lemmatization: A comparison of retrieval performances. *Lecture Notes on Software Engineering*, 2:262–267, 01 2014.

[3] S Bharadwaj, Rudra Varun, Potukuchi Sreeram Aditya, Macherla Nikhil, and G. Charles Babu. Resume screening using nlp and lstm. In *2022 International Conference on Inventive Computation Technologies (ICICT)*, pages 238–241, 2022.

[4] Luis Adrián Cabrera-Diego, Marc El-Bèze, Juan-Manuel Torres-Moreno, and Barthélémy Durette. Ranking résumés automatically using only résumés: A method free of job offers. *Expert Systems with Applications*, 123:91–107, 2019.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016.

[6] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA, February 1988. Association for Computational Linguistics.

[7] Chirag Daryani, Gurneet Chhabra, Harsh Patel, Indrajeet Chhabra, and Ruchi Patel. An automated resume screening system using natural language processing and similarity. pages 99–103, 01 2020.

[8] Christopher Dozier, Ravikumar Kondadadi, Marc Light, Arun Vachher, Sriharsha Veeramachaneni, and Ramdev Wudali. *Named Entity Recognition and Resolution in Legal Text*, pages 27–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[9] Email regex. `https://emailregex.com/`, 2024. Dostęp: 2024-12-04.

[10] Inc. Facebook. *React - A JavaScript library for building user interfaces*. `https://reactjs.org/`.

[11] Ant Financial. *Ant Design - A UI Design Language and React UI library*, 2023. `https://ant.design/`.

[12] D. Richard Hipp. *SQLite - SQL Database Engine*. `https://www.sqlite.org/`.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[14] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[15] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 3rd edition, 2024.

[16] Tian X. Pavur R. Han H. Zhang L. A machine learning-based human resources recruitment system for business process management: using lsa, bert and svm. *Business Process Management Journal*, pages vol. 29, no. 1, pp. 202–222., 2023.

[17] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[18] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[19] Microsoft. *TypeScript - JavaScript with Syntax for Types*. `https://www.typescriptlang.org/`.

[20] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[21] Abdul Hanan Minhas, Mohammad Daniyal Shaiq, Saad Ali Qureshi, Musa Dildar Ahmed Cheema, Shujaat Hussain, and Kifayat Ullah Khan. An efficient algorithm for ranking candidates in e-recruitment system. In *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–8, 2022.

[22] Rfc5322. `https://datatracker.ietf.org/doc/html/rfc5322`, 2024. Dostęp: 2024-12-04.

[23] Armin Ronacher. *Flask - A lightweight WSGI web application framework*. `https://flask.palletsprojects.com/`.

[24] Pradeep Roy, Sarabjeet Chowdhary, and Rocky Bhatia. A machine learning approach for automation of resume recommendation system. *Procedia Computer Science*, 167:2318–2327, 01 2020.

[25] Lokesh. S, Mano Balaje. S, Prathish. E, and B. Bharathi. Resume screening and recommendation system using machine learning approaches, 2022.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[27] Ebin Babu Thomas. Understanding lstm: An in-depth look at its architecture, functioning, and pros/cons. `https://www.linkedin.com/pulse/understanding-lstm-in-depth-look-its-architecture-pros-babu-thomas/`, 2023. Dostęp: 2024-11-28.

[28] Guido van Rossum and Fred L. Drake. *Python 3.10 Documentation*. `https://docs.python.org/3/`.

Spis rysunków

Spis tabel