Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka stosowana (IST)**
Specjalność: **Inżynieria oprogramowania (IO)**

PRACA DYPLOMOWA MAGISTERSKA

Predykcja współtwórców z długookresowym udziałem w projektach GitHub z wykorzystaniem technik uczenia maszynowego

Krzysztof Szafraniak

Opiekun pracy
dr hab. inż. Krzysztof Brzostowski

Uczenie maszynowe, Klasyfikacja, Długookresowy udział w projektach, GitHub

WROCŁAW (2022)

Streszczenie

Predykcja współtwórców z długookresowym udziałem w projektach GitHub z wykorzystaniem technik uczenia maszynowego

Obecny w dzisiejszych czasach nacisk na ekonomiczność i efektywność rozwijania oprogramowania sprawia, że wiele przedsiębiorstw decyduje się na wykorzystanie zewnętrznych narzędzi i bibliotek dostępnych publicznie m.in. na platformie GitHub. Jednym z najważniejszych kryteriów wyboru jest długość udzielanego wsparcia przez programistów.

W tym badaniu sprawdzono czy można skutecznie klasyfikować członków projektu, którzy przez długi czas będą pracować na rzecz danej biblioteki jednocześnie gwarantując im wsparcie przez ten okres. Celem pracy jest poprawa efektywności metody predykcji względem wcześniejszych badań.

Spośród tysiąca najczęściej obserwowanych repozytoriów GitHub wyselekcjonowano 798, które stanowiły bazę do utworzenia zbioru obserwacji. Każda z nich reprezentuje dane członka i repozytorium do którego dołączył oraz zawiera informacje o prawidłowej klasyfikacji twórcy do jednej z grup: posiadającej status długookresowego członka projektu lub nie. W większości pracy taką etykietę przypisywano twórcy, który aktywnie uczestniczy w projekcie przez minimum 3 lata. W trakcie badań wykorzystano 29 cech, z których pięć powstało w ramach niniejszej pracy. W poszukiwaniu najlepszej metody klasyfikacji przetestowano siedem różnych klasyfikatorów, a dwa z nich poddano procesowi dostrajania hiperparametrów. Głównym kryterium oceny modeli była metryka MCC, ale wykorzystywano również cztery inne, które pozwały lepiej zrozumieć własności poszczególnych klasyfikatorów. W trakcie badań sprawdzono istotność nowych predyktorów oraz porównano osiągnięte wyniki z tymi z publikacji bazowej.

Przy użyciu dziesięciokrotnej walidacji krzyżowej przetestowano najlepszy z opracowanych w niniejszej pracy modeli. Losowy las decyzyjny (ang. Random Forest) z dostrojonymi hiperparametrami w metrykach: precyzja, czułość, F1, AUC, MCC osiągnął wyniki odpowiednio 0,437; 0,443; 0,437; 0,857; 0,385 co względem bazowego modelu stanowi zmianę o -37%, +461%, + 212%, -6%, +70%.

Istotność nowych cech okazała się kluczowa dla modeli predykcji i wraz z optymalizacją zmiennych klasyfikatora pozwoliła osiągnąć znaczny postęp w jakości prognoz. Opracowany model również nie wymaga miesięcznego okresu oczekiwania na zebranie danych po dołączeniu twórcy do projektu oraz zachowuje niską, a nawet jeszcze nieco zmniejszoną liczbę predyktorów.

Abstract

Prediction of contributors with long-term participation in GitHub projects using machine learning techniques.

The current emphasis on the economy and effectiveness of software development makes many companies decide to use external tools and libraries available to the public, including on the GitHub platform. One of the most important selection criteria is the time of their developer support.

In this study, it was checked whether it is possible to effectively classify project members who will work for a given library for a long time and will be the guarantee of their support for this period. The study aims to improve the effectiveness of the prediction method from previous studies.

From among the thousand most-watched repositories in GitHub, 798 were selected, which were the basis for creating a set of observations. Each of them represents the data of the member and the repository he joined and contains information about the correct classification of the creator into one of the groups: having the status of a long-term contributor or not. In most of the work, this label was assigned to a member who has been actively participating in the project for a minimum of 3 years. During the research, 29 features were used, five of which were developed in this study. Seven different classifiers were tested to find the best classification method, and on two of them, the hyperparameter tuning process was applied. The main criterion for assessing the models was the MCC metric, but four others were also used, which allowed for a better understanding of the properties of individual classifiers. During the research, the significance of new predictors was checked and the achieved results were compared with those from the base publication.

The best models developed in this study were tested using ten-fold cross-validation. Random Forest with tuned hyperparameters in the metrics: precision, recall, F1, AUC, MCC achieved the results of 0.437, 0.443, 0.437, 0.857, 0.385 respectively which is a change of - 37%, +461%, +212%, -6%, +70% compared to the base model.

The significance of the new features turned out to be of key importance for the prediction models and together with the optimization of the classifier variables, allowed to achieve significant progress in the quality of predictions. The developed model also does not require a month waiting period for collecting data after the contributor joins the project and maintains a low and even slightly reduced number of predictors.

Spis treści

1. Wprowadzenie

Przedmiotem pracy jest opracowanie modelu predykcji współtwórców projektów na platformie GitHub, którzy przez długi okres będą wspierać ich rozwój. Nieliczne dotąd badania w tym obszarze pokazują, że jest to tematyka w której można osiągać satysfakcjonujące rezultaty, ale jednocześnie pozostawiają jeszcze wiele możliwości do ich poprawy. W trakcie realizacji tematu zdefiniowano nowe cechy oraz dodatkowe klasyfikatory, które mają pomóc w osiągnięciu tego celu. Niniejsza praca opisuje proces wytwórczy nowego modelu, porównuje jego możliwości z osiągnięciami opublikowanymi w dotychczasowej literaturze naukowej oraz bada istotność poszczególnych modyfikacji w kontekście otrzymanych rezultatów. Termin „długookresowy udział w projektach GitHub" jest określeniem subiektywnym, dlatego w trakcie badań wykonano eksperymenty, które pokażą wpływ różnych wariantów wartości progowych na jakość modelu predykcyjnego.

1.1. Geneza pracy

Szybko rozwijająca się branża informatyczna oraz rosnąca konkurencja na rynku producentów oprogramowania sprawia, że w dzisiejszym świecie tylko nieliczni na komercyjnym rynku mogą sobie pozwolić na rozwijanie w pełni samodzielnych narzędzi i oprogramowania. Rozwój nowoczesnych systemu w sposób efektywny wymaga od architektów korzystania z gotowych rozwiązań. Jednocześnie rosnąca konkurencja wśród przedsiębiorstw powoduje, że utrzymanie się na rynku i oferowanie atrakcyjnych kosztowo rozwiązań, wymaga rozsądnego zarządzania czasem i formą realizacji projektów. Od wielu lat hasło „reużywalność kodu" (ang. code reusability) przewodzi inżynierom w podejmowaniu decyzji dotyczących rozwiązań architektonicznych i jest ono ciągle tym czego rynek potrzebuje. Dziś jest istotne, aby się do niego dostosować.

Obecnie nie przeznacza się czasu na wielokrotną implementację tych samych rozwiązań, a wykorzystuje się wcześniejsze implementacje. Ma to wiele zalet. Zaoszczędzony w ten sposób czas można przeznaczyć na optymalizację gotowego rozwiązania, ulepszenia innych części systemu lub obniżenie kosztu finalnego produktu. Architekci mogą budować własny zbiór bibliotek, ale często korzystają z zewnętrznych źródeł. Dostępnych jest wiele publicznie dostępnych rozwiązań: implementacji algorytmów, „frameworków" - rozwijanych przez wielkie korporacje jak Google i Microsoft, jednak jeszcze więcej jest rozwiązań otwarto źródłowych od mniej znanych wydawców. Podstawowymi problemami w podjęciu decyzji o wykorzystywaniu zewnętrznej biblioteki są: zagwarantowanie wysokiej jakości rozwiązania, sprawnego naprawiania błędów i podatności oraz utrzymanie kompatybilności z nowszymi wersjami środowisk. Wszystkie wymienione kwestie są zależne od jednego czynnika – wsparcia projektu przez programistów. W przypadku projektów za którymi stoją wielkie korporacje najczęściej można oczekiwać planów wsparcia, jednak sytuacja jest zupełnie inna, gdy weźmie się pod uwagę projekty niedochodowe lub otwarto źródłowe (ang. Non-Profit, Open Source). W takim przypadku oszacowanie wsparcia projektu jest trudne, twórcy nie są w żaden sposób zobowiązani do pracy nad projektem i jest to zależne tylko od ich woli. Mimo to istnieją pewne cechy dotyczące zarówno projektów jak i twórców, które mogą niejawnie wykazywać potencjał na długotrwałe zaangażowanie twórców w projekt.

Związki pomiędzy cechami opisującymi projekty i twórców, a okresem czasu pracy na rzecz danego rozwiązania często nie są oczywiste. Wykrycie korelacji i wzorców pomiędzy nimi umożliwiają różne techniki uczenia maszynowego. Ten podzbiór prężnie rozwijającej się sztucznej inteligencji stanowi ważny element dziedziny analityki wielkich zbiorów danych. Na podstawie dostarczanych danych modele są w stanie samodzielnie się doskonalić i podejmować najlepsze decyzje oraz formułować prognozy. Ta technika od lat jest stosowana w mechanizmach rekomendacji, marketingu i doskonale sprawdza się gdy zanany jest jasno określony cel.

1.2. Analiza stanu sztuki

Analiza stanu sztuki w obszarze działań pozwoliła na wyselekcjonowanie artykułu bazowego, który stanowi punkt odniesienia dla przeprowadzanych badań. Wyróżniono również kilka związanych z pracą publikacji, które pomogą w podejmowaniu decyzji. Przegląd literatury wykonano przy użyciu dwóch wyszukiwarek: Scopus oraz Google Scholar. Scopus to baza bibliograficzna prowadzona przez wydawnictwo Elsevier indeksująca wyłącznie recenzowane publikacje naukowe, doniesienia konferencyjne i książki. W pierwszym etapie analizy stanu sztuki wyznaczono listę haseł wyszukiwania treści naukowych:

- GitHub LTC,
- GitHub Long Term Contributor,
- Assessing prediction performance
- Binary classification
- Machine learning algorithms

Odpowiedni dobór fraz pozwolił zawęzić rezultaty wyszukiwania do tych odpowiadających na potrzeby niemniejszej pracy. Dodatkowo każdy z kwalifikowanych artykułów musiał spełniać określone kryteria:

- Opublikowany w języku angielskim.
- Opublikowany nie wcześniej niż w 2007r.
- Publikacja z obszaru informatyki.

Zebrany zbiór publikacji poddano następnemu etapowi selekcji. Bazując na tytułach oraz streszczeniach wybrano 15 artykułów związanych z tematyką badań. Po bardziej szczegółowym zapoznaniu się z treściami publikacji odrzucono kolejnych 7 pozycji, których zawartość nie spełniała oczekiwań. Finalnie ustalono listę 8 prac naukowych, wśród których: trzy są bezpośrednio związane z predykcją współtwórców z długookresowym udziałem w projektach GitHub, trzy inne dotyczą selekcji metryk do oceny jakości modeli predykcyjnych, a pozostałe dwie publikacje omawiają problematykę selekcji cech oraz niezbalansowania danych.

Opublikowany w 2021 roku artykuł jest pierwszą z spójnych tematycznie prac dotyczących predykcji twórców [1]. Jej celem jest opracowanie modelu pozwalającego określić czy dany programista stanie się długookresowym współtwórcą projektu, na podstawie danych zebranych niedługo po jego dołączeniu do repozytorium. Czas na podstawie którego dokonywana jest klasyfikacja jest okresem pomiędzy pierwszym oraz ostatnim wprowadzeniem zmiany w projekcie przez programistę. W publikacji zbadano

predykcje dla trzech różnych progów czasowych: 1, 2 i 3 lat oraz czterech klasyfikatorów. Rezultatem pracy było opracowanie modelu klasyfikatora losowego lasu decyzyjnego (ang. Random Forest), dla którego współczynnik korelacji Matthewsa – metryka, która jest najistotniejsza dla prowadzonych badań - wyniósł 0,219.

Niniejsza praca dyplomowa bazuje na publikacji autorstwa V.K. Eluri opublikowanej w 2021r. [2]. Autorzy na podstawie ponad 70 tysięcy obserwacji opracowują model predykcji klasyfikujący twórców na tych z prognozowanym długookresowym wsparciem dla repozytorium (ang. Long-time contributors - LTC) i tych u których nie przewiduje się takiego zachowania (ang. non-LTC). Paca jest rozwinięciem badań z wcześniej omówionej publikacji. Każda z obserwacji zawiera zredukowaną z 63 do 31 liczbę cech programisty oraz repozytorium do którego dołączył. Autorzy donoszą, że dodatkowo pozbyli się warunku pozwalającego na predykcję dopiero po miesiącu od momentu dołączeniu do projektu. Mimo mniejszej liczby cech oraz usunięcia ograniczenia czasowego udało im się osiągnąć znaczącą poprawę w otrzymanych wynikach.

Ostatnim artykułem dotyczącym predykcji współtwórców, ale jednocześnie najwcześniej opublikowanym (w 2018r.) jest artykuł z konferencji ACM [3]. Autorzy proponują model predykcyjny oparty na zdolnościach programisty i jego możliwościach wniesienia wkładu w momencie dołączenia do repozytorium. Zauważono, że przyszli długo okresowi współtwórcy zwykle w ciągu pierwszego miesiąca są bardziej aktywni i wykazują bardziej zorientowane na społeczność nastawienie niż pozostali użytkownicy. Wykorzystując dziesięciokrotną walidację krzyżową osiągnięto średnie wyniki AUC, precyzji oraz czułości na poziomach odpowiednio: 0,807; 0,500; 0,022.

Istotną wiedzę dla procesu dobierania metryk wnoszą trzy publikacje omawiające problematykę oceny jakości klasyfikatorów. W artykule „Assessing software defection prediction performance: Why using the Matthews correlation coefficient matters" autorzy na podstawie przeanalizowanych danych dotyczących przewidywania defektów porównują dwie metryki [4]. Przeprowadzone badania wyjaśniają dlaczego wykorzystanie F1 jako przewodzącej metody oceny jakości wprowadza w błąd oraz sprawdzają w jakim stopniu jej wykorzystanie prowadzi do otrzymania niewiarygodnych rezultatów. Dwa kolejne artykuły również skupiają się na porównywaniu metryk [5][6]. Poza miarą, którą odnotowano przy opisie poprzedniego artykułu, sprawdzono też wskaźniki takie jak współczynnik korelacji Mathewsa (ang. Matthews correlation coefficient), precyzja (ang. precision), czułąość (ang. Recall) czy dokładność (ang. Accuracy). Z badań wynika, że wszystkie sposoby oceny jakości klasyfikatorów, które nie pokrywają w pełni macierzy błędów – mogą zostać zmanipulowane, a ich wyniki mogą nie być rzetelne

Wszystkie trzy wymienione wyżej publikacje posiadają dwa wspólne mianowniki. Wskazują na problem popularności metryki F1 w publikowanych pracach badawczych, jako jedynego kryterium oceny klasyfikatorów. Ignorowanie prawdziwie negatywnych predykcji sprawia, że przy jego użyciu zarówno dobre jak i słabe klasyfikatory mogą otrzymać niski rezultat. Drugi wspólny punkt tych publikacji stanowi recenzja wskaźnika - współczynnika korelacji Mathewsa. Autorzy podkreślają, że jest to metryka niepodatna na niezbalansowane dane, a otrzymanie wysokiego wyniku jest możliwe tylko gdy model jest w stanie dobrze zaklasyfikować większość zarówno pozytywnych jak i negatywnych jednostek danych.

Opublikowany przez IEEE w 2009r. artykuł „Learning from imbalanced data" autorstwa He H. oraz Gracia E.A i pochodzący z instytutu technologii w Hoboken, omawia

problematykę uczenia maszynowego na niezbalansowanych danych [7]. Ilość twórców z długookresowym czasem wsparcia poszczególnych projektów drastycznie różni się od ilości tych którzy wspierają je tylko przez krótki czas. Jak potwierdzono w trakcie realizacji pracy, główny zbiór danych zawiera jedynie ok. 8,64% wierszy zakwalifikowanych jako pozytywne, dlatego omawiana w przedstawionej publikacji tematyka jest ściśle związana z niniejszą pracą. W artykule bada się naturę problemu oraz stan ówczesnych rozwiązań. Zwraca się również uwagę na problematyczność oceny wydajności szkolonych modeli. Autorzy wskazują, że duże różnice w liczności różnie zaklasyfikowanych danych sprawiają, że rezultaty otrzymywane z poszczególnych metryk mogą prezentować zakrzywiony obraz rzeczywistości. Na przykładzie podkreślają istotność dobrania właściwych algorytmów do przeprowadzania ocen. Przyjęli wzorcowy zbiór z wierszami binarnie zakwalifikowanymi do różnych grup, których rozmiary określono na 5% i 95% całkowitej liczby wierszy. Podkreślają, że korzystając wyłącznie z wskaźnika dokładności (ang. Accuracy) i przyjmując bezspornie niepoprawny model testowy klasyfikujący wiersze tylko do grupy bardziej licznej, otrzymamy bardzo dobry rezultat na poziomie 0,95. Podkreślając zerową skuteczność w wykrywaniu elementów z mniej licznej klasy, wynik z pewnością nie jest adekwatny do jego ogólnych możliwości. Autorzy polecają wykorzystywanie metryk pokrywających w całości macierz błędów, taką jak średnia geometryczna, a metody takie jak: czułość, precyzja i dokładność jako uzupełnienie opisu modelu.

Dobre praktyki dotyczące wyboru cech i zdobywania danych zaczerpnięto z artykułu „Feature selection: A data perspective" opublikowanego w 2017r. na łamach dziennika „ACM Computing Surveys" [8]. Jego zawartość stanowi kompleksowy przegląd rozwiązań z zakresu budowania czystszych, prostszych i bardziej zrozumiałych zbiorów danych oraz poprawy wydajności eksploracji informacji. Autorzy wskazują na istotność redukcji skorelowanych cech co pozwoli zredukować rozmiar zbioru danych oraz poprawi jego czytelność.

1.3. Przegląd algorytmów

W tym podrozdziale zostały omówione algorytmy istotne w kontekście niniejszej pracy. Przegląd rozpoczyna się od weryfikacji różnych metod ocen modeli predykcyjnych. Wyjaśniono czynniki wpływające na jakość metryk oraz przedstawiono zastosowania i wady poszczególnych rozwiązań. W dalszej części omówiono klasyfikatory oraz sposoby optymalizacji (dostrajania) modeli.

1.3.1. Ocena jakości klasyfikatorów binarnych

Ocena jakości predykcji binarnych wykonywanych przez klasyfikatory obywa się z wykorzystaniem metryk. Te opierają się na ilościach predykcji zakwalifikowanych do poszczególnych grup macierzy błędów (Tabela 1). Tablica pomyłek rozróżnia prognozy według czterech stanów. Do oddzielnych grup trafiają predykcje: prawidłowo określające rekord danych jako pozytywny (TP), prawidłowo określające rekord danych jako negatywny (TN), nieprawidłowo określające rekord danych jako pozytywny (FP) oraz nieprawidłowo określające rekord danych jako negatywny (FN). Uwzględnienie lub nie, poszczególnych ćwiartek macierzy w metryce sprawia, że staje się ona bardziej uniwersalna lub specjalistyczna, a wybór pomiędzy nimi jest zależny od potrzeb badania. Czerpiąc z odnotowanej w bibliografii literatury w dalszej części opisano wybrane z dostępnych metryk [4][5][6][9].

Tabela 1. Macierz błędów

	Rzeczywistość **Pozytywna**	Rzeczywistość **Negatywna**
Prognoza **Pozytywna**	Prawdziwie pozytywna **(TP)**	Fałszywie pozytywna **(FP)**
Prognoza **Negatywna**	Fałszywie negatywna **(FN)**	Prawdziwie negatywna **(TN)**

Precyzja (ang. precision) określa jaka część wyników wskazanych przez klasyfikator jako pozytywna jest rzeczywiście pozytywna. Jest miarą pewności dodatnich wyników i może być odpowiednim wyborem do badań, w których przyjęcie pozytywnej klasy jest ryzykowne.

$$precision = \frac{TP}{TP + FP} \quad (1)$$

Stopa prawdziwie dodatnich predykcji inaczej nazywana czułością (ang. sensivity, recall, true positive rate, TPR) jest metryką, która określa jaką część dodatnich wyników wykrył klasyfikator. Jest miarą prawidłowego wykrycia wszystkich dodatnich wartości, ale nie gwarantuje, że prognozy oznaczone jako pozytywne są takie w rzeczywistości. Czułość może być odpowiednim wyborem do badań, w których wszystkie pozytywne wartości muszą zostać wykryte mimo ryzyka dużej liczby predykcji fałszywie pozytywnych.

$$recall = \frac{TP}{TP + FN} \quad (2)$$

Metryką F1 nazywa się średnią harmoniczną z *precyzji* oraz *czułości*, równomiernie uwzględniając obie te miary. Jej algorytm skupia się na ocenie prognoz dla wartości pozytywnych. Otrzymując maksymalny rezultat można mieć pewność, że wszystkie pozytywne wartości zostały wykryte oraz, że wśród pozytywnych predykcji nie ma żadnych niewłaściwych klasyfikacji. Metryka F1 jest dobrym wyborem jeżeli istotne jest wykrycie tylko dodatnich wartości, a prawdziwie negatywne prognozy nie powinny wpływać na rezultat.

$$F1 = \frac{2(precision \times recall)}{precisin + recall} \quad (3)$$

AUC (ang. area under the curve) czyli pole pod krzywą ROC jest metryką bazującą na wszystkich czterech grupach macierzy błędów. Krzywa ROC to jedna z metod wizualizacji jakości klasyfikacji, pokazująca zależność wskaźników FPR (ang. False Positive Rate) oraz TPR (ang. True Positive Rate) inaczej nazywana czułością. AUC jest miarą prawidłowych prognoz, wrażliwą na niezbalansowane dane.

$$FPR = \frac{FP}{TN + FP} \tag{4}$$

Podobnie jak AUC współczynnik korelacji Matthewsa (MCC) obejmuje wszystkie ćwiartki macierzy błędów i jednocześnie jest mniej wrażliwa na niezbalansowane dane. Współczynnik może być wykorzystywany jako uniwersalna miara jakości modeli predykcyjnych.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

W tabeli (Tabela 2) zebrano wszystkie wyżej wymienione metryki oraz opisano je w zwięzłej formie przedstawiając dla każdej z nich pokrycie macierzy błędów, zastosowania oraz wady.

Tabela 2. Podsumowanie własności metryk

Metryka	Pokrycie macierzy	Zastosowanie	Wady
Precision	2	Tam, gdzie istotna jest tylko pewność pozytywnych predykcji	Nie ocenia negatywnych prognoz. Możliwy maksymalny wynik mimo wielu fałszywie negatywnych prognoz. Możliwy minimalny wynik dla wielu prawdziwie negatywnych prognoz.
Recall	2	Tam, gdzie istotne jest wykrycie wszystkich pozytywnych wartości	Nie ocenia prognoz dla rzeczywiście negatywnych wartości. Możliwy maksymalny wynik mimo wielu fałszywie pozytywnych prognoz. Możliwy minimalny wynik dla wielu prawdziwie negatywnych prognoz.
F1	3	Tam, gdzie istotne jest wykrycie z pewnością wszystkich pozytywnych wartości.	Nie ocenia prawdziwie negatywnych prognoz. Możliwy minimalny wynik dla wielu prawdziwie negatywnych prognoz.
AUC	4	Tam, gdzie istotna jest poprawność każdej predykcji dla zbalansowanych danych lub gdzie celowo ignoruje się różnoliczność klas w zbiorze danych.	Wrażliwa na niezbalansowane dane jednocześnie „dyskryminując” jedną z klas.
MCC	4	Tam, gdzie istotna jest poprawność każdej predykcji niezależnie od danych.	

1.3.2. Klasyfikatory binarne

Zadaniem klasyfikatorów binarnych jest przyporządkowanie rekordom danych jednej z dwóch możliwych klas, często opisywanych jako pozytywna i negatywna. Ten cel osiąga się różnymi metodami. Część z klasyfikatorów wykorzystuje strukturę drzew, do których zalicza się: drzewo decyzyjne (ang. Decision Tree), losowy las decyzyjny (ang. Random Forest) oraz Gradient Boosting, inne mogą wykorzystywać rachunek prawdopodobieństwa jak: naiwny klasyfikator Bayesa z rozkładem Gaussa (ang. Gaussian Naive Bayes) lub Complement Naive Bayes, jeszcze inne np.: sieci neuronowe.

Drzewa decyzyjne (ang. Decision Tree) są jednymi z najprostszych metod klasyfikacji, które opierają się na podziale przestrzeni wartości cech na podprzestrzenie w taki sposób, aby każda z nich zawierała rekordy danych tylko jednej klasy. Podział dokonuje się sekwencyjnie budując drzewo decyzji. W każdym z węzłów znajduje się warunek dotyczący jednej z cech, który dzieli przestrzeń na dwie podprzestrzenie w taki sposób, aby ich zróżnicowanie było jak najmniejsze. Każda z nich jest ponownie dzielona do czasu w którym będzie zawierać rekordy tylko jednej klasy. Utworzonym w ten sposób liściom przypisuje się klasę zgodną z klasami rekordów, które koncentrują (gdy podział drzewa nie jest przeprowadzany do końca, liść przyjmuje klasę, która dominuje wśród zawieranych rekordów). Klasyfikacja nowego obiektu rozpoczyna się w korzeniu drzewa. Porównując jego cechy z warunkami w węzłach, po krawędziach dociera się do liścia, któremu przypisana jest decyzja o klasyfikacji.

Losowy las (ang. Random Forest) jest dużo mniej wrażliwy na różnice w zestawach testowych w stosunku do danych szkoleniowych niż jego poprzednik. Klasyfikator jest konfigurowalny hiperparametrem K określającym liczbę drzew w lesie drzew decyzyjnych, który często domyślnie określony jest wartością 100. Na bazie wprowadzonych danych treningowych tworzonych jest K nowych i różnych od siebie zbiorów zachowujących oryginalną liczność. Osiągnięcie tego celu jest możliwe dzięki zezwoleniu na powtarzanie się tych samych krotek w powstających zestawach. Dobór rekordów odbywa się na zasadzie losowania ze zwracaniem z zbioru bazowego – treningowego. Na tak powstałych danych budowane są K drzewa decyzyjne. Proces klasyfikacji polega na przetestowaniu rekordu w każdym z nich, zebraniu wynikowych klas i podjęcie decyzji wskazującej na tą dominującą.

Gradient Boosting jest to kolejna metoda oparta na drzewach. Od ang. Random Forest wyróżnia ją metoda tworzenia drzew. Gdy losowy las buduje wiele niezależnych drzew decyzyjnych, Gradient Boosting wykorzystuje metodę nazywaną wzmacnianiem (ang. Boosting). Tworzy tzw. słabe klasyfikatory (drzewa często zawierające jeden warunek) sekwencyjnie, bazując na rezultatach poprzedniego i poprawiając jego błędy. Po dokonaniu podziału obserwacji przez pierwsze drzewo na dwie grupy, rekordom które należą do klasy mniejszościowej w swoim zbiorze (czyli te które zostały źle sklasyfikowane) nadawane są większe wagi, po czym wszystkie trafiają do następnego słabego klasyfikatora. Finalny klasyfikator jest kombinacją powstałych drzew decyzyjnych.

Klasyfikatorem z innej grupy jest algorytm K najbliższych sąsiadów (ang. K Nearest Neighbour). Każda obserwacja z zbioru treningowego posiadająca n-cech jest umieszczana w n wymiarowej przestrzeni. Klasyfikacja polega na umieszczeniu w niej nowego rekordu i odnalezieniu K najbliższych (najczęściej względem odległości euklidesowej) obserwacji. Następnie na podstawie klas przypisanych tym obserwacjom wybiera się tą dominującą. Metoda najbliższych sąsiadów jest wrażliwa na rozstęp wartości cech. Gdy predyktor A ma

większy rozstęp niż predyktor B, praktycznie jest mu przypisana mniejsza waga (odległość pomiędzy minimalną, a maksymalną wartością jest większa). Dlatego nie znając istotności poszczególnych cech zaleca się normalizację danych.

Naiwny klasyfikator Bayesa z rozkładem Gaussa (ang. Gaussian Naive Bayes) jest klasyfikatorem opartym na prawdopodobieństwach i gęstościach prawdopodobieństw. W pierwszym kroku obliczana jest szansa wystąpienia obserwacji klas pozytywnej oraz negatywnej wśród danych treningowych. Następnie zestaw jest dzielony na dwa podzestawy oddzielające obserwacje różnych klas. Dla każdej cechy i każdej podgrupy tworzona jest funkcja gęstości prawdopodobieństwa. Klasyfikacja odbywa się poprzez wybranie klasy dla której otrzymano wyższą wartość wskaźnika. Każdej funkcji gęstości należącej do danej klasy, przypisuje się jej wartość w punkcie wyznaczonym przez odpowiednią cechę sprawdzanego rekordu danych testowych. Prawdopodobieństwo danej klasy obliczone w pierwszym kroku, wymnaża się z otrzymanymi dla predykatorów gęstościami otrzymując wskaźnik. Otrzymana wartość byłaby bardzo mała dlatego na wartość nakłada się funkcję logarytm.

Complement Naive Bayes podobnie jak poprzedni klasyfikator opiera się na prawdopodobieństwach. Został opracowany, aby poprawić rezultaty predykcji przy pracy na niezbalansowanych danych. Stosuje się tu nieco odwróconą metodę polegającą na obliczaniu prawdopodobieństw nie należenia do poszczególnych klas. Testowany rekord klasyfikuje się do tej dla której otrzymano najniższy wynik.

MLPClassifier (ang. Multi-Layer Perceptron Classifier) czyli klasyfikator oparty na wielowarstwowej sieci. neuronowej jest najmniej interpretowalnym z pośród prezentowanych klasyfikatorów. Podstawowym elementem składowym jest neuron, czyli jednostka, która na podstawie danych wejściowych i własnych parametrów (wag krawędzi i ang. bias) oblicza swoją wartość. Sieć składa się z warstw: wejściowej, wyjściowej oraz warstw ukrytych w konfigurowalnej ilości. Wyjście z neuronów warstw wyższych stanowią wejście do tych z warstwy niższej, gdzie dokonywane są następne obliczenia. Szkolenie modelu odbywa się poprzez ocenę otrzymanego wyniku i użycia go do skorygowania parametrów neuronów.

Właściwości wymienionych klasyfikatorów z podziałem na poszczególne rodziny (mające wspólną bazę) w zwięzłej formie opisano w tabeli poniżej (Tabela 3).

Tabela 3. Podsumowanie przeglądu klasyfikatorów

Klasyfikator	Baza	Właściwości
Drzewo decyzyjne (ang. Decision Tree)	Struktura drzew	Prosta metoda klasyfikacji. Możliwe zbytnie dopasowanie (przetrenowanie).
Losowy las decyzyjny (ang. Random Forest)		Odporność na zmienność danych. Odporność na przeuczenie. Odporny na wartości odstające.
Gradient Boosting		Model budowany sekwencyjnie, uwzględniając błędy z wcześniejszych kroków.
Klasyfikator K najbliższych sąsiadów (ang. K Nearest Neighbour)	Odległość (euklidesowa)	Przejmuje klasę wzorując się na najbardziej podobnych do siebie obserwacjach. Zalecana normalizacja danych.
Naiwny klasyfikator Bayesa z rozkładem Gaussa (ang. Gaussian Naive Bayes)	Prawdopodobieństwo	Klasyfikacja oparta na gęstościach prawdopodobieństwa przynależności do danej klasy obliczonych dla poszczególnych cech.
Complement Naive Bayes		Niewrażliwy na niezbalansowanie danych. Stosuje odmienną metodę – sprawdza do której klasy nie należy.
MLPClassifier	Sieć neuronowa	Warstwowa budowa. Wyniki nie są interpretowalne. (Nie da się uzasadnić decyzji)

1.3.3. Walidacja krzyżowa

Istotnym problemem w szkoleniu i ocenie modeli uczenia maszynowego jest rzetelność i wiarygodność otrzymywanych wyników. Budowa modelu predykcji polega na jego dopasowywaniu się do przedstawionych mu danych. Klasyfikator wypada więc szczególnie dobrze, gdy dane testowe są identyczne z tymi na których został wyszkolony. Dlatego tak wykonany pomiar nie jest wiarygodny. Prawidłowo przeprowadzony test powinien wykorzystywać próbkę danych, która nie została wykorzystana podczas szkolenia modelu. Najprostszym rozwiązaniem jest podział danych na zbiór treningowy oraz testowy. W takim przypadku szkolenie i ocenę modelu przeprowadza się na oddzielnych zestawach. Rozwiązanie jest skuteczne, jednak może stwarzać problemy z wrażliwością wyników na sposób wykonania podziału danych. Umieszczenie w obu zbiorach, obiektów tej samej klasy ale skupiających pewne charakterystyczne cechy, spowoduje zaniżenie oceny jakości modelu. Aby zminimalizować problem braku równomiernego zróżnicowania danych wykorzystuje się walidację krzyżową.

Jest to algorytm konfigurowalny parametrem K oznaczającym ilość równolicznych grup na które zostanie podzielony zbiór. Po wyznaczeniu podzbiorów, każdy z nich kolejno staje się zestawem testowym, gdy wszystkie pozostałe są łączone w treningowy. W trakcie wykonywania algorytmu zostanie wyszkolonych K modeli, które będą testowane na K różnych danych. Każdy z rekordów jednokrotnie należy do zbioru testowego i K-1 krotnie do treningowego. W wyniku tych operacji otrzymuje się K rezultatów predykcji odpowiednio po jednym dla każdego z wyszkolonych modeli. Wyznaczenie finalnej oceny odbywa się poprzez ich uśrednienie.

1.3.4. Algorytmy dostrajania hiperparametrów

Przygotowanie zestawów danych oraz dobór odpowiednich metod klasyfikacji to jeszcze nie koniec możliwości poprawiania rezultatów predykcji. Większość z popularnych klasyfikatorów posiada zmienne, dzięki którym z zewnątrz możliwe jest sterowanie niektórymi właściwościami modelu predykcyjnego. Często w publikowanych badaniach korzysta się wyłącznie z domyślnych ustawień, co może okazać się błędem. Dostrajanie hiperparametrów, czyli dobieranie najlepszych wartości zmiennych dla danych, nierzadko pozwala osiągnąć przyrost efektywności rzędu 10-30% w porównaniu do domyślnej konfiguracji.

Jednak wykorzystanie tej metody optymalizacji ma też swoje wady. Zbyt precyzyjne dostrajanie może doprowadzić do przetrenowania modelu w efekcie czego nie uzyska się zadawalających efektów. Istotne jest też aby przeprowadzić badania rzetelnie. Dlatego należy pamiętać o separacji danych treningowych od testowych. Wykorzystując domyślne ustawienia klasyfikatora nie jest on w żaden sposób zoptymalizowany pod konkretne dane, dlatego można przyjąć niezależność hiperparametrów od próbki danych. Natomiast dostrajanie parametrów klasyfikatora odbywa się na danych, na których w dalszym etapie planowano wykonać pomiar. W takim podejściu otrzymane rezultaty nie będą wiarygodne. Możliwe rozwiązania są dwa: zdobycie dodatkowej próbki danych przeznaczonej wyłącznie do testów lub bardziej prawdopodobne rozwiązanie: dodatkowy podział na zbiory testowe i treningowe. Dodatkowy początkowy podział zbioru sprawia, że podczas testów mamy do dyspozycji zmniejszoną liczbę rekordów używanych w walidacji krzyżowej.

Istnieją dwie popularne metody dobierania hiperparametrów klasyfikatora [10][11]. Pierwszą z nich jest metoda siatki (ang. Grid Search). Definiuje się tablicę parametrów przeznaczonych do przetestowania, a następnie są wyznaczane ich wszystkie kombinacje. W następnym kroku algorytmu dla każdego zestawu hiperparametrów szkoli się model oraz zapisuje wyniki. Na koniec wybiera się kombinację parametrów dla których klasyfikator wypadł najlepiej. Jest to świetna metoda, jednak bardzo kosztowa. Przetestowanie 10 wartości dla 10 różnych parametrów z 10-krotną walidacją krzyżową wymaga wytrenowania i ocenienia 10^{11} modeli. Zakładając, że poświęcony na 1 model czas to 1s, wynik otrzymano by dopiero po 3171 latach.

Drugim sposobem optymalizacji klasyfikatorów poprzez dostrajanie ich hiperparametrów jest metoda losowania (ang. Random Search). Podobnie jak w (ang. Grid Search) algorytm rozpoczyna się od zdefiniowana siatki parametrów oraz wyliczania ich kombinacji. W dalszym kroku zamiast testować każdą z nich wybiera się n losowych konfiguracji i na nich przeprowadza testy. Liczba zestawów n jest konfigurowalna.

1.4. Cel pracy

Celem pracy jest opracowanie nowego modelu uczenia maszynowego, który będzie prezentował postęp w kontekście osiąganych przez dotychczasowe rozwiązania rezultatów. Otrzymywane z niego predykcje w jak najlepszy sposób powinny przewidywać potencjał współtwórców danego projektu w odniesieniu do długotrwałego wsparcia.

Osiągnięcie tego celu będzie opierać się na wynikach kilku przeprowadzonych w niniejszej pracy badań, które przyrostowo będą ulepszały model. Doświadczeniom będą podlegać różne klasyfikatory. Celem jest porównanie ich efektywności dla zadanego problemu badawczego i wyselekcjonowanie najbardziej dopasowanych rozwiązań. Osiągnięcie jeszcze lepszych wyników może umożliwić próba dostrojenia hiperparametrów poszczególnych modeli z zachowaniem pełnej izolacji zbiorów treningowych i testowych. Zbadana zostanie również istotność nowo zdefiniowanych cech dla opracowanych modeli uczenia maszynowego.

1.5. Zakres pracy

Zakres prac skupia się wokół przygotowania modelu uczenia maszynowego oraz wpływie poszczególnych czynników na osiągane rezultaty. W ramach niniejszej pracy zrealizowane zostały następujące zadania:

1. Zdefiniowanie zestawu cech dla modelu uczenia maszynowego.
2. Przygotowanie skryptów SQL przetwarzających surowe dane.
3. Przygotowanie kilku zestawów danych wejściowych.
4. Implementacja oprogramowania badawczego.
5. Implementacja oprogramowania testującego
6. Konteneryzacja oprogramowania.
7. Wytrenowanie modeli uczenia maszynowego.
8. Przegląd rezultatów na tle literatury bazowej.

Postawiono również pięć problemów badawczych na które w trakcie przeprowadzonych doświadczeń udzielono odpowiedzi:

Pyt.1 Jaki wpływ na dokładność predykcji ma wykorzystanie nowego zestawu cech ?

Pyt.2 Czy wykorzystanie nowych klasyfikatorów wpłynie pozytywnie na otrzymywane rezultaty ?

Pyt.3 Jak wpływa przyjęty próg czasowy określający twórcę jako „długookresowego" na wyniki predykcji ?

Pyt.4 Czy strojenie (ang. tuning) hiperparametrów modelu poprawi otrzymywane rezultaty ?

Pyt.5 Które z cech są najistotniejsze dla opracowanych modeli ?

2. Przygotowanie danych

W tym rozdziale przedstawiono sposób pozyskania wykorzystywanych w pracy zbiorów danych. Opisano różne dostępne źródła, wybrany zestaw cech, sposób przetwarzania surowych informacji oraz wykorzystując podstawowe metryki omówiono powstałe zbiory.

2.1. Źródło danych

GitHub jest serwisem hostingowym gromadzącym projekty/repozytoria wykorzystujące system kontroli wersji Git. Według danych zebranych w lutym 2022r. usługa zrzesza ponad 73 miliony programistów, ponad 4 miliony organizacji oraz przeszło 200 milionów repozytoriów [12]. Jest obecnie największą platformą tego typu. Konkurencyjny serwis GitLab posiada o połowę mniej użytkowników (ponad 30 milionów). Poza swoim głównym przeznaczaniem udostępnia wiele różnych dodatkowych funkcjonalności, gromadząc wiele metadanych. GitHub ze względu na swoją popularność oraz posiadanie wielu informacji opisujących użytkowników i repozytoria stanowi dobrą bazę dla przeprowadzonych badań.

Dostęp do danych serwisu można uzyskać na kilka sposobów, ale nie wszystkie pozwalają efektywnie pozyskać ich masową ilość. Tak jest w przypadku GitHub REST API i jego następcy GraphQL API. Są to oficjalne interfejsy udostępniające zasoby portalu, które mogą zostać wykorzystane przez jego użytkowników przy pomocy darmowego „żetonu" (ang. token). Wiąże się z nim jednak limit określony w dokumentacji ograniczający liczbę danych które można pozyskać w jednostce czasu, co ma na celu zapobieganie nadużyciom.

Innym rozwiązaniem jest wykorzystanie projektu GHTorrent [13]. Jego celem jest budowanie baz danych SQL oraz NoSQL, bazując na omówionym wcześniej interfejsie. Projekt jest nieoficjalny dlatego i jego również dotyczą ograniczenia związane z limitem pobieranych danych w czasie. Autorzy rozwiązania opierają swój projekt na doktrynie Fair Use. W zamian za korzystanie z zgromadzonych danych proszą użytkowników o udostępnienie im swojego żetonu. Zrzuty baz danych są publicznie dostępne jednak problem stanowi ich rozmiar, który często przekracza 10TB.

Ostatnią z metod dostępu do danych jest wykorzystanie BigQuery do pracy na opublikowanych w nim bazach GHTorrent. Jest to narzędzie pochodzące od Google służące do przechowywania i przetwarzania ogromnych ilości danych [14]. Dzięki silnikowi SQL użytkownicy mogą wykonywać polecenia w chmurze nie martwiąc się o swoje zasoby sprzętowe. Usługa jest płatna jednak oferuje też darmowy pakiet, który jest wystarczający do przeprowadzanych badań.

Ze względu na ograniczenia pozostałych metod w niemniejszej pracy wykorzystano usługę BigQuery. Jest to rozwiązanie, które pozwoli ominąć niewystarczające zasoby maszyny wykorzystanej do badań. Podjęcie decyzji o wykorzystaniu usługi chmurowej umożliwi dostęp do bazy danych, a także do udostępnianego silnika SQL. Pojawia się więc możliwość wykonania wstępnego przetwarzania informacji w usłudze. Dane po przetworzeniu w chmurze będą posiadały znacznie zredukowany rozmiar co pozwoli na dalszą pracę nad nimi na maszynie lokalnej.

Schemat wykorzystywanej bazy danych zawiera dwadzieścia encji. Korzystając z dokumentacji zawartej na stronie projektu, opracowano zwięzły opis ośmiu z nich (Tabela 4), które zostały wykorzystane w niniejszym badaniu [13].

Tabela 4. Opis wykorzystywanych tabel bazy danych

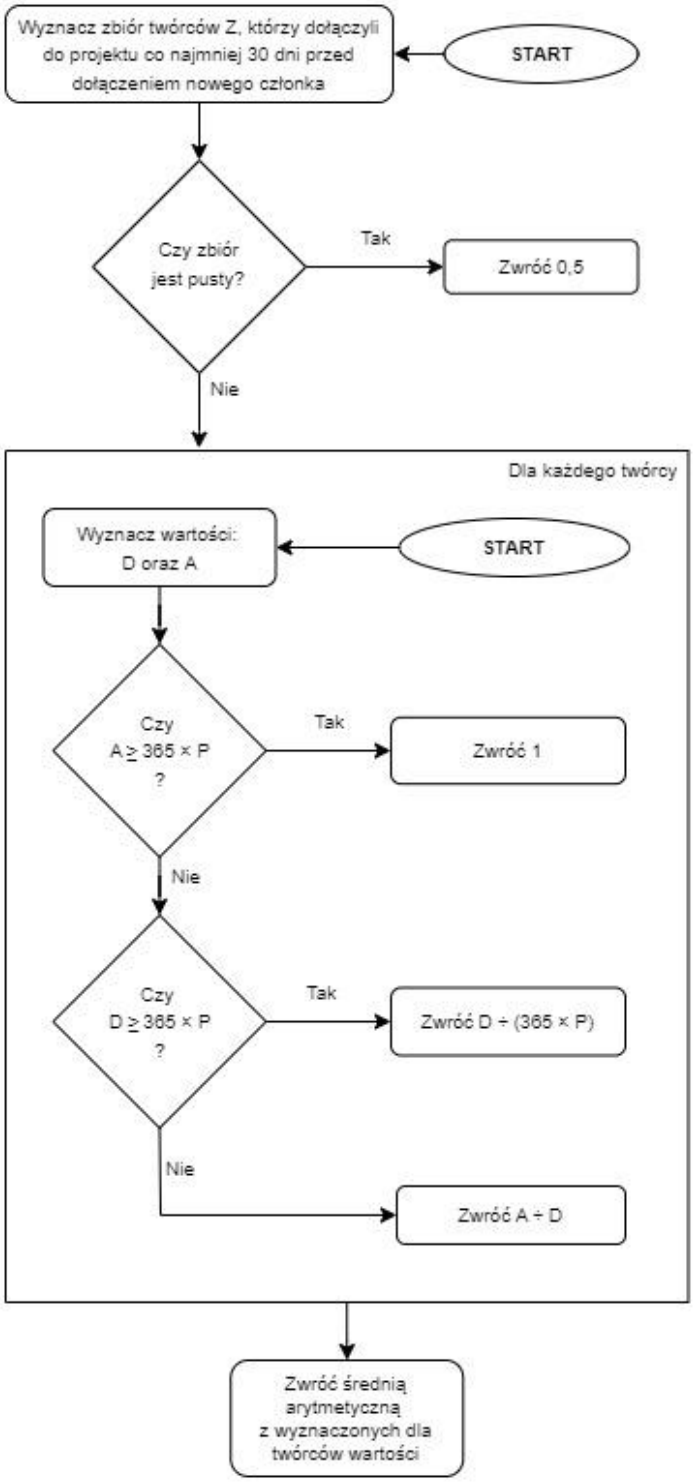
Nazwa tabeli	Opis
commits	Tabela opisująca kto, kiedy oraz do którego repozytorium wprowadził zmiany.
followers	Tabela opisująca kto, kogo i kiedy zaczął obserwować.
issues	Tabela opisująca dodane zgłoszenia.
projects	Tabela opisująca repozytorium (url, język, stan, itp.)
pull_request_history	Tabela przedstawiająca historię wykonania akcji w kontekście zdarzeń „Pull Request”.
pull_requests	Tabela zawierająca podstawowe informacje o zdarzeniach „Pull Request”.
users	Tabela opisująca użytkowników usługi (data dołączenia, typ, stan, itp.).
watchers	Tabela skrzyżowań łącząca użytkownika, który śledzi repozytorium z dodatkowym polem – datą rozpoczęcia.

2.2. Zestaw cech

Dobór cech do przeprowadzanych badań rozpoczęto od przeglądu informacji zawartych w bazie danych. Następnie, analizie poddano 31 predyktorów na których oparto badania w publikacji z Uniwersytetu w Waszyngtonie. Dużą część z nich wykorzystano ponownie, jednak zaszły też zmiany. Oznaczenie cech jest charakterystyczne. Literą „r” rozpoczynają się te dotyczące repozytorium, literą „c” te opisujące użytkownika/programistę/współtwórcę (‘c’ od ang. contributor).

Z pierwotnego zbioru cech usunięto siedem pozycji, które ze względu na brak precyzyjnego opisu metody ich określania rodziły wątpliwości. Brak dołączonych źródeł uniemożliwiał ustalenie dokładnego algorytmu, a część z predyktorów budziła obawę, że zostały zebrane nie w momencie dołączenia programisty do repozytorium, a w momencie wykonania migawki bazy danych. Jest wśród nich m.in. cecha „c_is_Deleted” mająca określać czy repozytorium było usunięte w chwili dołączenia. Obawa wynika z tego, że w bazie danych nie są zawarte informacje na temat zmian tego parametru w czasie. Źródłem tej informacji mogła być kolumna „deleted” w tabeli „projects” jednak opisuje ona jedynie stan na dzień wykonania migawki. Nie odnaleziono również wystarczających informacji, aby potwierdzić, że wersja plików Markdown na których oparto niektóre z tych cech pochodziły z właściwego momentu w historii repozytorium.

Wprowadzono natomiast 5 nowych cech, które mogą być istotne dla klasyfikatorów w badanym obszarze. Pierwszą jest „r_Avg_Contributor_Involvement” określająca poziom zaangażowania członków danego repozytorium i skalowana wartością ustalonego progu. Metoda polega na wyznaczeniu średniej z wskaźników zaangażowania wszystkich członków w projekt, którzy dołączyli do niego przed nowym współtwórcą. Algorytm wyznaczania tej cechy zobrazowano na schemacie blokowym (Rysunek 1) oraz opisano poniżej.



Rysunek 1. Schemat blokowy algorytmu wyznaczania cechy „r_Avg_Contributor_Involvement”.

Oznaczenia:

L	Zbiór twórców, którzy dołączyli do projektu co najmniej 30 dni przed momentem dołączenia nowego członka.
P	Przyjęta liczba lat określająca twórcę jako długookresowego
D	Liczba dni od wprowadzenia pierwszej zmiany w repozytorium do momentu dołączenia nowego członka.
A	Liczba dni aktywnego udziału w projekcie do momentu dołączenia nowego członka. (Różnica pomiędzy datą pierwszej i ostatniej wprowadzonej zmiany)

Ideą wskaźnika jest promowanie repozytoriów, których obecni członkowie są lub jest prawdopodobne, że staną się długookresowymi współtwórcami w projekcie. Cecha przyjmuje tym większą wartość, im członkowie projektu dłużej pracują na rzecz repozytorium i tym mniejszą im krócej. Poniższy wzór opisuje główną myśl powyższego algorytmu.

$$w = \begin{cases} \frac{\sum_{i \in L} w_i}{|L|} & , |L| > 0 \\ \frac{1}{2} & , |L| = 0 \end{cases}$$

$$w_i = \begin{cases} 1 & , A_i \geq 365 \times P \\ \frac{A_i}{365 \times P} & , A_i < 365 \times P \wedge D_i \geq 365 \times P \\ \frac{A_i}{D_i} & , w\ p.p \end{cases} \quad (6)$$

Drugą wprowadzoną cechą jest „c_Avg_Repository_Involvement”, która podobnie jak poprzednia również mierzy zaangażowanie. W tym przypadku nie jest to jednak cecha repozytorium, mierząca aktywność członków w danym repozytorium, a predyktor opisujący użytkownika platformy GitHub określający jego średnie zaangażowanie w projekty, w których brał udział. Algorytm jest bardzo zbliżony do zaprezentowanego powyżej, główną różnicą jest to, że tym razem iteruje po repozytoriach nowego użytkownika, a nie po wczesnych członkach repozytorium. Zadaniem wskaźnika jest określenie czy programista mocno angażuje się w projekty do których dołącza. Gdy w wszystkich projektach z którymi miał styczność wprowadził tylko po jednej zmianie jego wskaźnik wyniesie 0.

Koleją wprowadzoną cechą jest „r_Days_From_Last_Commit” odwzorowująca liczbę dni, które upłynęły od ostatnio wprowadzonej zmiany w danym repozytorium. Duża wartość może oznaczać, że projekt został porzucony co zmniejszałoby prawdopodobieństwo na dołączanie nowych aktywnych członków.

Predyktor „r_Number_of_Pull_Requests” jest liczbą zgłoszonych zdarzeń typu „Pull Request” w danym projekcie. Są to m.in. mechanizmy informowania innych użytkowników o gotowości i chęci wprowadzenia zmian w zawartości repozytorium wspierające również wykonywanie przeglądu kodu. Wiele takich zdarzeń może oznaczać, że w projekcie są stosowane dobre praktyki rozwijania oprogramowania oraz, że wprowadzanie zmian w bibliotekach jest nadzorowane. To natomiast prawdopodobnie wpływa na chęć przystąpienia do współtworzenia danego rozwiązania przed długi czas.

Ostatnią z wprowadzonych cech jest „r_Number_of_Issues”. Jest ona odzwierciedleniem liczby wątków przypiętych do danego repozytorium. Użytkownicy za ich pomocą zgłaszają błędy, pomysły na ulepszenia lub dzielą się wiedzą na temat funkcjonalności biblioteki i ich zachowań. Ta metryka może odwzorowywać rozmiar społeczności zainteresowanej lub korzystającej z rozwiązań dostarczanych przez dany projekt. Popularność bibliotek potencjalnie wpływa na chęci twórców do rozwijania produktów. Większą motywacje do pracy będzie wykazywać ten twórca który jest świadom, że jego praca nie przepadnie, a wręcz przeciwnie posłuży ogromnej społeczności.

Finalnie ustalono zbiór 29 predyktorów, które zostały wykorzystane do badań w dalszej części pracy. Tabela 5 jest zestawieniem wszystkich z tych cech łącząc ich nazwy z zwięzłym opisem. Dodatkowo pogrubioną czcionką wyróżniono nowe predyktory.

Tabela 5. Opis zdefiniowanych predyktorów. Wyróżniono cechy, które nie były wykorzystane w wcześniejszych badaniach

Nazwa cechy	Opis
r_Language	Główny język programowania wykorzystywany w repozytorium.
r_Age	Liczba dni od powstania repozytorium do momentu dołączenia współtwórcy.
r_Number_of_Watchers	Liczba użytkowników usługi GitHub śledząca repozytorium.
r_Number_of_Contributors	Liczba współtwórców projektu.
r_Min_All_Contributor_Commits	Najmniejsza liczba zmian wprowadzonych przez jednego członka.
r_Max_All_Contributor_Commits	Największa liczba zmian wprowadzonych przez jednego członka.
r_Min_All_Contributor_Stay	Najkrótszy całkowity okres aktywności jednego członka liczony w dniach.
r_Max_All_Contributor_Stay	Najdłuższy całkowity okres aktywności jednego członka liczony w dniach.
r_Avg_Contributor_Involvement	Średnie zaangażowanie współtwórców w repozytorium.
r_Days_From_Last_Commit	Liczba dni od ostatniej wprowadzonej zmiany.
r_Number_of_Issues	Liczba zarejestrowanych zgłoszeń.
r_Number_of_Pull_Requests	Liczba zarejestrowanych zdarzeń typu „Pull Request”
c_is_Fake	Czy konto jest fałszywe (np. należące do bota).
c_Age	Liczba dni od zarejestrowania do momentu dołączenia do repozytorium.
c_is_Watching_Project	Czy nowy członek w momencie dołączania do repozytorium je obserwował.
c_Own_Repos_Lang	Liczba posiadanych repozytoriów w języku programowania zgodnym z projektem do którego dołącza.
c_Watch_Repos_Lang	Liczba obserwowanych repozytoriów w języku programowania zgodnym z projektem do którego dołącza.
c_Contribute_Repos_Lang	Liczba repozytoriów do których należy jako współtwórca w języku programowania zgodnym z projektem do którego dołącza.
c_History_Commits_Lang	Liczba zmian wprowadzonych w języku programowania zgodnym z projektem do którego dołącza.
c_History_Pull_Request_Lang	Liczba zarejestrowanych zdarzeń typu „Pull Request” w języku programowania zgodnym z projektem do którego dołącza.
c_History_Issues_Lang	Liczba zarejestrowanych zgłoszeń w języku programowania zgodnym z projektem do którego dołącza.
c_History_Followers	Liczba obserwujących nowego członka projektu.
c_History_Following	Liczba obserwowanych przez nowego członka projektu.
c_Number_of_Repos,	Liczba repozytoriów posiadanych przez nowego członka projektu.
c_Min_All_Repos_Commits	Najmniejsza liczba zmian wprowadzonych w projektach w których brał udział.
c_Max_All_Repos_Commits	Największa liczba zmian wprowadzonych w projektach w których brał udział.
c_Min_All_Repos_Stay	Najkrótszy całkowity okres aktywności w jednym repozytorium liczony w dniach.
c_Max_All_Repos_Stay	Najdłuższy całkowity okres aktywności w jednym repozytorium liczony w dniach.
c_Avg_Repository_Involvement	Średnie zaangażowanie w rozwój projektów w których użytkownik brał udział.

2.3. Przetwarzanie wstępne

Procedura pozyskiwania danych bazowych jest wzorowana na tych opisanych w artykułach [1][2]. Podstawą do utworzenia zbiorów danych jest tysiąc najczęściej obserwowanych repozytoriów w usłudze GitHub z pewnymi włączeniami.

Wiersze z tabeli „watchers" pogrupowano względem identyfikatora repozytorium i posortowano według liczby elementów w każdej z nich, jednocześnie ustalając kolejność projektów od tych z największą liczbą obserwujących do tych z najmniejszą. W przedstawionych w artykułach metodach sortowane odbywa się z wykorzystaniem liczby gwiazdek. Ogwaizdkowanie repozytorium i jego zaobserwowanie nie jest tą samą czynnością, jednak tym co w artykułach bazowych rozumiano poprzez liczbę gwiazdek jest w rzeczywistości liczba obserwatorów. Można tak wnioskować z opisu tabeli „watchers", w którym przedstawiono ją jako zbiór zdarzeń ogwiazdkowania projektu przez użytkowników. W praktyce są to jednak różne akcje, różniące się przede wszystkim otrzymywaniem powiadomień. Aby podążać za wzorcowym schematem pozyskiwania danych do ustalenia kolejności projektów wykorzystano liczbę obserwujących, jednoczenie wyjaśniając różnice w pojęciach i nie powtarzając błędnego określenia.

W następnych kilku krokach odfiltrowano część z tysiąca wybranych projektów, które z przedstawionych w dalszej części powodów, nie byłyby dobrą podstawą do przeprowadzenia badań. Rozpoczęto od odrzucenia repozytoriów, dla których nie zdefiniowano języka programowania. Takie repozytoria w kolumnie „language" nie posiadały wartości. Są to przypadki, w których w repozytorium nie jest przechowywany kod źródłowy, a inne materiały np. pliki PDF. Następnie wykluczono te nie korzystajcie z „Issue Trackera". W praktyce takie miano otrzymuje repozytorium z zerową liczbą przypisanych wątków. Odrzucono również te projekty, które są rozgałęzieniem/kopią. Taką informację pozyskano z tabeli „projects", w której obecność wartości w kolumnie „forked_form" oznaczała, że takie repozytorium nie powstało z czystego projektu. Ostatnim wykorzystanym filtrem było wybranie tylko tych z nich, które nie zostały oznaczone jako usunięte.

Zdarzają się przypadki w których data pierwszej wprowadzonej zmiany w repozytorium jest wcześniejsza niż data jego założenia na platformie GitHub. Taka sytuacja może zajść m.in. gdy projekt został utworzony lokalnie, a dopiero później wysłany na platformę lub gdy został przeniesiony z innej usługi. Aby skorygować to nienaturalne zjawisko w danych przyjęto, że datę utworzenia repozytorium będzie stanowić data wcześniejsza z pierwszej zarejestrowanej zmiany lub daty jego utworzenia.

Dla tak przygotowanej listy projektów przygotowano finalne zestawy danych, w których każdy wiersz zawiera informacje o repozytorium, twórcy oraz wyliczonych dla nich cechach. Co ważne wszystkie wartości predyktorów zostały obliczone na danych z przed momentu dołączenia programisty do projektu. Jednocześnie badaniom podlegali tylko Ci, którzy dołączyli do niego odpowiednio wcześnie (w zależności od ustalonego progu), czyli tak, aby mieli możliwość zostać tym kogo określa się twórcą z długookresowym udziałem.

Budując główny zestaw danych zmierzono liczności wierszy na poszczególnych etapach jego doskonalenia. Jest nim zbiór będący odpowiednikiem tego, wykorzystywanego w wcześniejszych badaniach, czyli opartego na migawce z 1 września 2017 roku oraz z progiem przyjętym na poziomie 3 lat. Z wybranych 1000 najczęściej obserwowanych repozytoriów usunięto wiersze bez zdefiniowanego języka, w wyniku czego pozostały 852

projekty. Wykluczenie tych nie korzystających z „Issue Trackera” zredukowało tę liczbę o 1. Sześć z repozytoriów okazało się rozgałęzieniami innych, które również odrzucono. Finalnie wyselekcjonowano 798 projektów, które powstały po wykluczeniu tych oznaczonych jako usunięte.

2.4. Opracowane zbiory

Poza głównym zbiorem danych wejściowych przygotowano jeszcze 7 zestawów w innej konfiguracji. Finalnie powstał jeden zbiór dla każdego z progów 1, 2, 3 oraz 4 lata w kombinacji z dwoma bazami danych z dni 1 września 2017r i 1 kwietnia 2018r. W znajdującej się poniżej tabeli (Tabela 6) przedstawiono opis poszczególnych zbiorów wraz z licznościami wierszy określonych jako LTC (ang. Long-Time Contributor).

Tabela 6. Podstawowe metryki wygenerowanych zbiorów danych

Data migawki	Próg LTC (lata)	Liczba wierszy	LTC
01/09/2017	1	52 720	7 469 (14,17%)
	2	38 620	3 574 (9,25%)
	3	21 491	1 856 (8,64%)
	4	13 006	992 (7,63%)
01/04/2018	1	57 326	8 383 (14,62%)
	2	45 240	4 123 (9,11%)
	3	29 430	2 182 (7,41%)
	4	14 706	1 160 (7,89%)

Liczba wierszy zmniejsza się wraz z długością progu ponieważ jest znacznie mniej twórców, którzy są częścią projektu od 4 lat niż od 1 roku (przyp. aby programista został włączony do badanego zbioru musi należeć minimum wartość progu czasu, aby móc go prawidłowo zakwalifikować jako LTC lub Non-LTC).

3. Metodyka przeprowadzania badań

Opisane w poprzednim rozdziale zestawy danych zostały pobrane z usługi BigData w formacie CSV i stanowią wejście do opracowanych skryptów, w których rozwiązywane są przedstawione w wprowadzeniu problemy badawcze. W tym rozdziale została przedstawiona metodyka przeprowadzania badań i poszukiwania najlepszego modelu predykcji. Opisano również wykorzystane rozwiązania technologiczne, które pomogły w realizacji celu.

3.1. Rozwiązania technologiczne

Implementację skryptów wykonano w języku Python z wykorzystaniem specjalistycznych bibliotek. Numpy oraz Pandas są zastosowane do odczytu danych z pliku oraz manipulacji zestawami danych. Natomiast pakiet Sklearn dostarcza niezbędnych narzędzi związanych z uczeniem maszynowym.

Aby przeprowadzenie doświadczeń było możliwe, konieczne było zakodowanie jedynej tekstowej cechy jakościowej „r_langauge" na postać liczbową, która jest wymagana przez wykorzystywane implementacje klasyfikatorów. Do tego celu użyto narzędzie LabelEncoder z pakietu Sklearn w tym przypadku mapującego języki programowania na przypisane im numery. Dodatkowo przeprowadzono normalizację danych, która jest istotna m.in. dla klasyfikatora N Najbliższych Sąsiadów. Wykonano ją metodą skalowania według wartości minimalnej i maksymalnej z użyciem narzędzia MinMaxScaller pochodzącego również z pakietu Sklearn.

Wszystkie wykonane implementacje w języku Python zostały opakowane w kontener Docker gwarantujący działanie na różnych środowiskach. Jego mały rozmiar w stosunku do maszyn wirtualnych oraz jednoczesne izolowanie zasobów w dużym zakresie rozwiązuje problemy z mobilnością zaimplementowanych skryptów.

3.2. Metoda oceny jakości predykcji

Do oceny jakości modeli predykcji posłużyła metryka MCC (ang. Matthews Correlation Coefficient), która stanowiła wyznacznik i wspólny punkt porównawczy pomiędzy opracowanymi w niniejszym badaniu klasyfikatorami oraz tymi z badań wcześniejszych. Jest to metryka posiadająca obecnie duże uznanie w świecie nauki, często stosowana i polecana w publikacjach, również w tych z przeglądu literatury. Dla każdego z opracowanych modeli wyznaczono dodatkowo precyzję, czułość, AUC oraz F1, które pozwolą lepiej opisać i zrozumieć ich właściwości.

Przedstawiane w pracy wyniki zostały uzyskane poprzez obliczenie średniej z 10 wyników będących rezultatem dziesięciokrotnej walidacji krzyżowej. Dzięki temu każdy wiersz danych bierze udział zarówno w części treningowej jak i w weryfikacyjnej i pozwala uniknąć błędów, wynikających z niewłaściwego podziału zbioru.

3.3. Klasyfikatory

Weryfikacja wpływu nowego zestawu cech na jakość predykcji wymaga zastosowania tych samych klasyfikatorów opisanych w artykule bazowym, dlatego w grupie tych wykorzystywanych w niniejszej pracy znalazły się: drzewo decyzyjne (ang. Decision Tree), las drzew losowych (ang. Random Forest), klasyfikator K najbliższych sąsiadów (ang. K Nearest Neighbours) oraz naiwny algorytm Bayesa z rozkładem Gaussa (ang. Gaussian Naive Bayes). Nie można jednak zagwarantować, że ostatni z nich jest tym zastosowanym w wcześniejszych badaniach, ponieważ nie sprecyzowano dokładnie, która wersja klasyfikatora Naive Bayes została użyta.

Ich pulę postanowiono rozszerzyć o trzy dodatkowe algorytmy klasyfikacji. Pierwszym z nich jest Gradient Boosting, czyli samodoskonalące się drzewo decyzyjne, które docelowo jest konkurentem dla algorytmu lasu drzew losowych (ang. Random Tree). Następnym Complement Naive Bayes, który dzięki swojej niewrażliwości na niezbalansowanie danych stanowi dużą konkurencję dla swojego odpowiednika w wersji wykorzystującej rozkład normalny. Ostatnim jest klasyfikator MLPClassifier oparty na odmiennej od pozostałych bazie – sieci neuronowej. Na etapie planowania można było spodziewać się lepszych rezultatów otrzymywanych dla nowych klasyfikatorów w porównaniu do swoich odpowiedników (w kontekście bazy/zasady działania) z pierwszej grupy. Weryfikacja tych oczekiwań nastąpiła podczas realizacji jednego z problemów badawczych, opisanych w dalszej części pracy.

3.4. Metodyka dostrajania hiperparametrów

Wykonanie dostrajania hiperparametrów wymaga dodatkowego podziału zbioru, aby zagwarantować separację zestawów testowych od treningowych. Pierwszy podział dzieli dane na 40% danych testowych oraz 60% danych treningowych, na których będzie wykonywana optymalizacja (ang. tunning). W ramach tego procesu dokonywane są kolejne podziały drugiego z zbiorów. Wynikają one z zastosowania walidacji krzyżowej, która wielokrotnie dzieli już zmniejszony zbiór na mniejsze - testowe i treningowe - odpowiednio w proporcjach 1:4.

Zastosowana metodyka dostrajania hiperparametrów opiera się na obu przedstawionych w wprowadzeniu metodach. Pierwsza faza wykorzystuje metodę losowania (ang. Random Search). Wprowadzono do niego siatkę wartości z szerokiego zakresu oraz polecono przetestowanie kilku tysięcy losowych kombinacji. Celem tego etapu jest ograniczenie zakresu wartość poszczególnych hiperparametrów do takiego w obrębie którego model spisuje się najlepiej. Metoda nie testuje wszystkich kombinacji, ale wyznaczony przez nią najlepszy zestaw zmiennych, może oznaczać, że w jego okolicach znajduje się ten optymalny dla badanego problemu oraz klasyfikatora.

W drugiej fazie konstruowana jest gęściejsza siatka hiperparametrów skupiona wokół wyników otrzymanych w pierwszej fazie. Przetestowanie tym razem wszystkich kombinacji przy użyciu metody siatki (ang. Grid Search) da nam pewność, że w przeszukiwanej przestrzeni hiperparametrów nie ma bardziej optymalnego zestawu niż rezultat pracy skryptu. W efekcie otrzymuje się wystarczająco dobrze wyselekcjonowane wartości zmiennych, których zastosowanie powinno pozytywnie wpłynąć na ocenę jakości modelu predykcji.

4. Problemy badawcze

W tym rozdziale opisano rozwiązania postawionych w wprowadzeniu problemów badawczych. Pomagają one zarówno stawiać kroki naprzód do osiągnięcia postawionego w niniejszej pracy celu oraz interpretować zachodzące zmiany. W konsekwencji tłumaczą co wpływa na poprawę wydajności opracowywanego modelu.

4.1. Jaki wpływ na dokładność predykcji ma wykorzystanie nowego zestawu cech ?

Pierwszym doświadczeniem jest zweryfikowanie jak dobór nowych predyktorów wpłynął na osiągane przez poszczególne modele rezultaty. Przeprowadzając to badanie wykorzystano klasyfikatory opisane w publikacji bazowej oraz zastosowano domyślne ustawienia klasyfikatorów.

Tabela 7. Zestawienie ocen jakości modeli predykcji otrzymanych po wprowadzeniu nowych cech do klasyfikatorów oraz wyników osiągniętych i opisanych w artykule bazowym

Klasyfikator		Precyzja	Czułość	F1	AUC	MCC
kNN	Bazowe	0,597	0,044	0,082	0,811	0,158
	Nowe cechy	0,416	0,146	0,216	0,708	**0,207 (+31%)**
DT	Bazowe	0,524	0,075	0,131	0,851	0,191
	Nowe cechy	0,278	0,304	0,289	0,615	**0,220 (+15%)**
RF	Bazowe	0,695	0,079	0,140	0,913	0,226
	Nowe cechy	0,580	0,205	0,303	0,851	**0,312 (+38%)**
(G)NB	Bazowe	0,146	0,395	0,210	0,812	0,210
	Nowe cechy	0,252	0,205	0,225	0,732	**0,162 (-23%)**

Przeglądając wyniki doświadczenia (Tabela 7) można zauważyć, że nowe cechy w trzech przypadkach znacznie poprawiają wartość współczynnika korelacji Matthewsa. W jednym przypadku odnotowano jego spadek. Przypominając jednak, że ze względu na brak odpowiedniej informacji nie można być pewnym, że porównywany klasyfikator z rodziny naiwnych algorytmów Bayesa jest właściwy, należy z ostrożnością interpretować ten wynik. Analiza pokazuje, że w wcześniejszych badaniach osiągano bardzo dobre wyniki AUC oraz precyzji za to czułość, F1 oraz MCC przeważnie są na dużo niższym poziomie. Bardzo niskie czułości wskazują, że klasyfikatory bardzo rzadko wykrywają prawdziwie pozytywne rezultaty. Klasyfikator K najbliższych sąsiadów prawidłowo kategoryzuje pozytywne wiersze nawet rzadziej niż raz na dwadzieścia prób. Podobnie jest z klasyfikatorami opartymi o drzewa decyzyjne. Tak duże różnice pomiędzy poszczególnymi wartościami świadczą, że model może sprawdzić się dobrze w jednych badaniach, a w innych wypaść tragicznie. Metryka AUC osiąga wysokie wartości (nawet powyżej 0,9) pomimo, że model słabo spisuje się w niektórych przypadkach. Właśnie dlatego zaleca się ocenianie jakości modeli z wykorzystaniem MCC, gdyż jest to miara, która osiąga dobre wyniki tylko wtedy, gdy ogólna sprawność modelu jest dobra i nie posiada wrażliwych punktów.

Wprowadzenie nowych predyktorów już na starcie pozwoliło istotnie poprawić najlepszy wynik modelu z artykułu bazowego. Wzrost wartości wskaźnika z 0,226 do 0,312 stanowi przyrost aż o 38%. Jest to efekt opracowania bardziej zbilansowanych modeli, które choć w niektórych aspektach wypadają słabiej to jednak w zdecydowanie większym stopniu uzupełniają najsłabsze strony bazowych modeli - m.in. nawet czterokrotnie poprawiając czułość.

4.2. Czy wykorzystanie nowych klasyfikatorów wpłynie pozytywnie na otrzymywane rezultaty ?

W tym podrozdziale sprawdzono czy wprowadzone nowe klasyfikatory pozwolą poprawić dotychczas najlepszy wynik lub przynajmniej zgodnie z oczekiwaniami wypadną lepiej niż swoje odpowiedniki z tej samej rodziny. Tabela 8 przedstawia oceny, zdobyte przez nowe modele predykcji. Dodatkowo dla dwóch z nich prezentuje ona procentową zmianę wskaźnika MCC w stosunku do odpowiadającego mu klasyfikatora z pierwszego problemu badawczego należącego do tej samej rodziny.

Tabela 8. Oceny jakości modeli predykcji opartych na nowych klasyfikatorach oraz przedstawienie procentowej zmiany wskaźnika w stosunku do klasyfikatora z tej samej rodziny z poprzedniego problemu badawczego

Klasyfikator	Precyzja	Czułość	F1	AUC	MCC
Complement NB	0,150	0,721	0,249	0,739	**0,191 (+18%)**
Gradient Boosting	0,624	0,218	0,322	0,857	**0,337 (+8%)**
Neural Network (MLPC)	0,577	0,175	0,267	0,829	**0,285**

Complement Naive Bayes został porównany z Gaussian Naive Bayes i jak oczekiwano osiągnięto lepszy o 18% rezultat. Poprawa jest wynikiem znacznie poprawionej czułości modelu, która wzrosła 3,5 krotnie. Odnotowano też mniej istotne poprawy w metrykach F1 i AUC oraz niższą wartość precyzji.

Model oparty na klasyfikatorze Gradient Boosting ma bezpośrednio konkurować z tym wykorzystującym las drzew losowych (ang. Random Forest). Zarejestrowano wyższe rezultaty dla każdej z wykorzystywanych metryk, a główny z nich poprawił się o 8% wynosząc 0,337 i jednocześnie ustanawiając nowy najlepszy rezultat.

Ostatnim jest klasyfikator oparty na sieci neuronowej. Nie posiada on swojego odpowiednika w wcześniejszych badaniach, dlatego warto sprawdzić jak poradzi on sobie w predykcji współtwórców z długookresowym udziałem w projektach GitHub. Rezultaty są zadawalające. Wynik na poziomie 0,285 jest do tej pory trzecim najlepszym rezultatem pokonując metodę najbliższych sąsiadów, czy też obie oparte na prawdopodobieństwach - Naive Bayes.

4.3. Jak wpływa przyjęty próg czasowy określający twórcę jako „długookresowego” na wyniki predykcji ?

Wymagany okres wsparcia danej biblioteki/projektu przez programistów jest zależny od indywidualnych potrzeb przedsięwzięcia i może różnić się dla każdego z nich. Dlatego w niniejszej pracy powtarza się zmienna nazywana progiem, która reguluje tę subiektywną własność. Domyślnym i głównym progiem przyjętym dla badań w niniejszej pracy są trzy lata. Kontynowanie pracy na rzecz projektu przez ten czas sprawia, że taki użytkownik otrzymuje etykietę długookresowego współtwórcy. Jednak nie jest to jedyna właściwa wartość, dlatego zbadano również, jak dwa - do tej pory - najlepsze klasyfikatory sprawdzają się dla różnych opcji tej zmiennej. Dodatkowo sprawdzono również ich efektywność dla danych z migawki bazy danych pochodzącej z innego dnia.

Tabela 9. Prezentacja ocen jakości modeli predykcyjnych dla różnych danych oraz przyjętych progów

Klasyfikator	Dane	Próg	Precyzja	Czułość	F1	AUC	MCC
Random Forest	2017	1	0,703	0,266	0,386	0,816	**0,384 (+23%)**
		2	0,652	0,233	0,343	0,844	**0,357 (+14%)**
		3	0,580	0,205	0,303	0,851	**0,312 (base)**
		4	0,550	0,205	0,295	0,853	**0,303 (-3%)**
	2018	1	0,714	0,282	0,404	0,822	**0,398 (+23%)**
		2	0,651	0,220	0,329	0,849	**0,347 (+7%)**
		3	0,597	0,208	0,208	0,864	**0,324 (base)**
		4	0,566	0,211	0,304	0,863	**0,312 (-4%)**
Gradient Boosting	2017	1	0,711	0,239	0,358	0,819	**0,365 (+8%)**
		2	0,680	0,224	0,336	0,850	**0,359 (+7%)**
		3	0,624	0,218	0,322	0,857	**0,337 (base)**
		4	0,586	0,206	0,303	0,871	**0,317 (-6%)**
	2018	1	0,727	0,260	0,383	0,826	**0,386 (+16%)**
		2	0,659	0,209	0,317	0,854	**0,340 (+2%)**
		3	0,622	0,208	0,311	0,868	**0,333 (base)**
		4	0,581	0,192	0,287	0,870	**0,304 (-9%)**

Analizując dane zawarte w tabeli (Tabela 9) można zauważyć pewien wzór. Im próg jest mniejszy tym osiągano lepsze rezultaty. Możliwym powodem jest wielkość zestawów danych, która również rośnie wraz z skracaniem przyjmowanego okresu (Tabela 6). Mniejszy próg sprawia również, że łatwiej zdobyć etykietę długookresowego współtwórcy, przez co ich stopa rośnie, a sam zbiór staje się bardziej zbalansowany.

4.4. Czy strojenie hiperparametrów modelu poprawi otrzymywane rezultaty ?

W ramach tego problemu badawczego wybrano dwa do tej pory najlepsze klasyfikatory (ang. Random Forest oraz Gradient Boosting) i poddano je procesom strojenia ich hiperparametrów. Podczas przeprowadzania pierwszej fazy optymalizacji liczba przetestowanych kombinacji w metodzie losowej (ang. Random Search) wynosiła 5000 dla lasu drzew losowych oraz 3000 dla Gradient Boosting, co stanowiło odpowiednio 17% oraz 46% wszystkich możliwych kombinacji.

Tabela 10. Opis hiperparametrów klasyfikatora Random Forest wykorzystanych w procesie dostrajania modelu

Nazwa	Opis
bootstrap	Czy losowe próbkowanie z wymianą jest włączone.
n_estimators	Liczba drzew decyzyjnych w lesie.
max_depth	Maksymalna głębokość drzewa decyzyjnego.
max_features	Maksymalna liczba cech decydująca o podziale węzła.
max_leaf_nodes	Maksymalna liczba liści drzewa decyzyjnego.
min_samples_leaf	Minimalna liczba próbek wymagana w liściach drzewa.
min_samples_split	Minimalna liczba próbek wymagana do podziału węzła.
class_weight	Metoda ważenia klas w drzewie decyzyjnym.
criterion	Funkcja mierząca jakość podziału. / Kryterium podziału.

Tabela 11. Opis hiperparametrów klasyfikatora Gradient Boosting wykorzystanych w procesie dostrajania modelu

Nazwa	Opis
loss	Optymalizowana funkcja straty.
n_estimators	Liczba etapów doskonalenia\wzmacniania modelu.
max_depth	Maksymalna głębokość drzewa decyzyjnego.
max_features	Maksymalna liczba cech decydująca o podziale węzła.
max_leaf_nodes	Maksymalna liczba liści drzewa decyzyjnego.
min_samples_leaf	Minimalna liczba próbek wymagana w liściach drzewa.
min_samples_split	Minimalna liczba próbek wymagana do podziału węzła.
learning_rate	Tępo uczenia się / poprawiania błędów.

W powyższych tabelach (Tabela 10 i Tabela 11) opisano hiperparametry, które zostały poddane optymalizacji. Tabela 12 prezentuje natomiast rezultaty osiągnięte podczas dostrajania. W drugiej kolumnie obliczono bazowe wartości wskaźnika MCC do których należy porównywać osiągane rezultaty. Zostały one obliczone z wykorzystaniem

domyślnych ustawień klasyfikatorów oraz zmniejszonego zbioru testowego (patrz 3.4). Trzecia kolumna prezentuje wartości jakie udało się uzyskać podczas szkolenia modeli na danych treningowych. Są to również wartości dla których pobrano konfiguracje hiperparametrów, które zostały zaaplikowane do testów na zbiorze testowym. Finalne rezultaty prezentuje ostatnia kolumna.

Tabela 12. Rezultaty modeli przed i po wykonaniu dostrajania hiperparametrów

Klasyfikator	MCC - bazowy	MCC – zbiór treningowy	MCC – zbiór testowy
Random Forest	0,318	0,392 (+23%)	**0,385 (+21%)**
Gradient Boosting	0,357	0,358 (+0%)	**0,342 (-4%)**

W przypadku klasyfikatora lasu drzew losowych (ang. Random Forest) optymalizacja bardzo się opłaciła. Udało się uzyskać wzrost wskaźnika o powyżej 20%, jednocześnie znacznie przebijając dotychczasowy najlepszy wynik. Różnica wyników na zbiorach testowych oraz treningowych jest akceptowalna i zrozumiała z oczywistych względów.

Inaczej jest natomiast z drugim z klasyfikatorów - Gradient Boosting. Jak można zauważyć na zbiorze treningowym uzyskano marginalny i pomijalny przyrost, całkowicie nie adekwatny do kosztów (m.in. czasu) poniesionych w trakcie procesu. Co istotne wynik, który naprawdę się liczy (przyp. osiągnięty na zbiorze testowym) pogorszył się w stosunku do bazowej wartości wskaźnika o ponad 4%. W sytuacji, gdy wartość podstawowa oraz ta uzyskana na zbiorze treningowym są bardzo zbliżone można wnioskować, że nie było tu miejsca na efektywną optymalizacje. Osiągnięty minimalny przyrost prawdopodobnie skutkował zbytnim dopasowaniem modelu do danych, w efekcie czego, wynik na zbiorze testowym wypadł istotnie gorzej niż pozostałe.

4.5. Które z cech są najistotniejsze dla opracowanych modeli ?

Dla przygotowanych w rozwiązaniu poprzedniego problemu badawczego modeli dokonano weryfikacji istotności poszczególnych predyktorów w predykcji współtwórców. Dzięki temu rozpoznano cechy, które w największym stopniu mają wpływ na wynik klasyfikacji oraz te, które ze względu na niski udział mogłyby zostać pominięte. Zweryfikowano też najciekawsze zagadnienie - gdzie w rankingu istotności predyktorów znajdują się te nowe - wprowadzone w niniejszej pracy. To doświadczanie pozwoliło ocenić trafność doboru cech. Tabela 13 przedstawia zestawienie wszystkich predyktorów posortowanych według obliczonej dla nich istotności zarówno dla klasyfikatora lasu drzew losowych (ang. Random Forest) jak i Gradient Boosting. Ważności cech wyrażane są procentowo, a suma ich wartości wynosi 100%. Wytłuszczonym drukiem wyróżniono te z nich, które zostały opracowane na potrzeby niniejszej pracy.

Tabela 13. Istotności poszczególnych predyktorów dla modeli po optymalizacji hiperparametrów

	Random Forest		Gradient Boosting	
#	Cecha	Istotność	Cecha	Istotność
1	**r_Number_of_Issues**	**13,48%**	c_History_Following	9,08%
2	c_History_Following	11,00%	**r_Avg_Contributor_Involvement**	**8,19%**
3	**r_Avg_Contributor_Involvement**	**8,47%**	**r_Number_of_Issues**	**8,14%**
4	**r_Number_of_Pull_Requests**	**8,21%**	**r_Number_of_Pull_Requests**	**6,90%**
5	r_Number_of_Watchers	6,42%	r_Max_All_Contributor_Commits	5,94%
6	r_Max_All_Contributor_Commits	4,91%	r_Number_of_Watchers	5,61%
7	r_Max_All_Contributor_Stay	4,84%	r_Max_All_Contributor_Stay	5,27%
8	r_Number_of_Contributors	3,98%	r_Number_of_Contributors	4,51%
9	r_Age	3,95%	r_Age	4,49%
10	c_Age	3,02%	c_Age	4,24%
11	r_Language	2,73%	c_History_Commits_Lang	3,43%
12	**c_Avg_Repository_Involvement**	**2,62%**	c_is_Fake	3,02%
13	c_Contribute_Repos_Lang	2,61%	**c_Avg_Repository_Involvement**	**2,99%**
14	c_History_Commits_Lang	2,53%	c_Max_All_Repos_Commits	2,99%
15	c_Max_All_Repos_Commits	2,47%	c_History_Followers	2,95%
16	c_is_Fake	2,46%	c_Max_All_Repos_Stay	2,86%
17	c_History_Followers	2,40%	r_Language	2,52%
18	c_Max_All_Repos_Stay	2,21%	c_Contribute_Repos_Lang	2,45%
19	c_Number_of_Repos	2,14%	c_Number_of_Repos	2,28%
20	c_Watch_Repos_Lang	1,72%	r_Min_All_Contributor_Commits	2,04%
21	c_History_Issues_Lang	1,69%	c_Watch_Repos_Lang	1,88%
22	c_Own_Repos_Lang	1,58%	c_History_Issues_Lang	1,83%
23	r_Min_All_Contributor_Commits	1,10%	c_Own_Repos_Lang	1,69%
24	**r_Days_From_Last_Commit**	**0,94%**	**r_Days_From_Last_Commit**	**1,44%**
25	c_History_Pull_Request_Lang	0,88%	c_History_Pull_Request_Lang	1,08%
26	c_Min_All_Repos_Commits	0,64%	c_Min_All_Repos_Commits	0,85%
27	c_is_Watching_Project	0,46%	c_Min_All_Repos_Stay	0,58%
28	c_Min_All_Repos_Stay	0,34%	r_Min_All_Contributor_Stay	0,45%
29	r_Min_All_Contributor_Stay	0,21%	c_is_Watching_Project	0,30%

Analiza przedstawionych w tabeli rezultatów, kieruje do spostrzeżenia, że w obu rankingach istotności cech, ich kolejność jest bardzo zbliżona. Łatwo można zauważyć te z nich, które znajdują się na tej samej lokacie. Pozostałe różnią się najczęściej o 1-3 pozycje, a pojedyncze przypadki o 4 lub maksymalnie 5.

Wśród pięciu nowych cech jedna okazała się mało przydatna dla modeli. Jest nią liczba dni od wprowadzenia ostatniej zmiany do repozytorium. Okazało się, że mimo, iż może stanowić przesłankę o porzuceniu projektu lub przeciwnie - dużej aktywności w nim, nie ma ona większego wpływu na predykcje długotrwałego zaangażowania współtwórców w projekt.

Na drugim końcu zestawienia znalazły się aż trzy cechy znajdujące się na czterech najwyższych lokatach w rankingach dla obu modeli. Suma ich istotności dla modelu opartego na lesie drzew losowych (czyli najlepszego opracowanego modelu) wynosi ponad 30% stanowiąc dużą część całości w kontekście 26 innych predyktorów. Jak się okazało aktywność społeczności tworzącej wątki w projekcie, intensywność korzystania z mechanizmu „Pull Request" oraz ocena historycznego zaangażowania twórców w dany projekt ma kluczowe znaczenie dla jakości predykcji. Warto również odnotować, że na drugiej pozycji znalazła się cecha „c_History_Following" pochodząca z zestawu wykorzystanego w publikacji bazowej, która uzupełnia ostatnią wolną pozycję wśród czterech najważniejszych predyktorów.

Ostatnia z dodatkowych cech „c_Avg_Repository_Involvement" zajęła miejsce nieco powyżej środka rankingu. Z wynikiem na poziomie 2,62% można uznać że jest ona istotna dla modelu jednak nie kluczowa jak opisane wyżej trzy predykatory. Interesująca okazała się różnica pozycji pomiędzy tą cechą, a jej odpowiednikiem dotyczącym repozytorium, który okazał się ponad 3 krotnie istotniejszy. Wynika z tego, że większy wpływ na długotrwały udział programisty w projekcie ma nie ocena średniego zaangażowania nowego twórcy w dotychczasowe projekty, a średnie zaangażowanie obecnych współtwórców w projekt do którego dołącza.

5. Najefektywniejszy model predykcji

Osiągnięcie postawionego w wprowadzeniu celu pracy było możliwe dzięki wykonanej sekwencji czynności. Zawarte w niej kroki stopniowo doprowadziły do wyznaczenia finalnego modelu predykcji, a najistotniejszymi z nich były:

1. Opracowanie nowego zestawu cech oraz algorytmów ich wyznaczania.
2. Wyznaczenie metody dostrajania hiperparametrów klasyfikatorów będącej kombinacją metod siatki i losowej.
3. Wyszkolenie oraz pomiar efektywności modeli opartych na klasyfikatorach wykorzystywanych w bazowej publikacji z wykorzystaniem nowego zestawu cech.
4. Wyznaczenie klasyfikatorów potencjalnie bardziej dopasowanych do charakterystyki analizowanych danych niż klasyfikatory z artykułu bazowego uwzględniając m.in. niezbalansowanie danych.
5. Wyszkolenie oraz pomiar efektywności modeli opartych na nowych klasyfikatorach z wykorzystaniem nowego zestawu cech
6. Analiza dotychczasowych rezultatów modeli oraz optymalizacja hiperparametrów wykonana dla dwóch najefektywniejszych z nich.

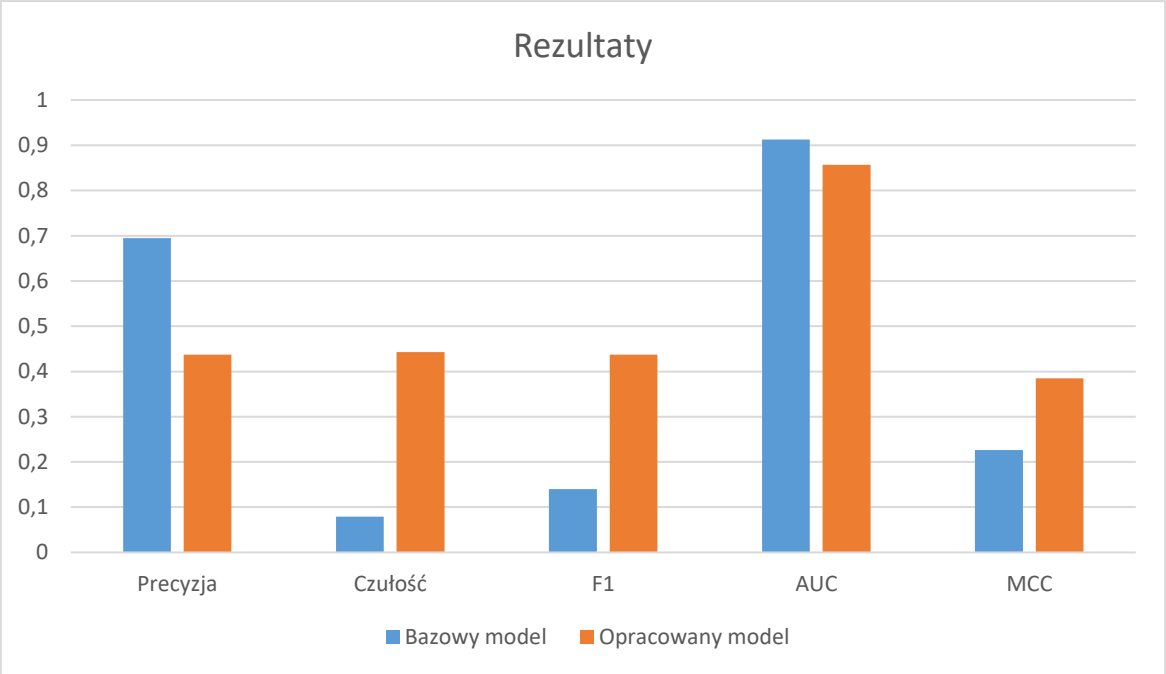
Już rozwiązanie pierwszego z wyznaczonych problemów badawczych pozwoliło osiągnąć cel: „opracowanie nowego modelu uczenia maszynowego, który będzie prezentował postęp w kontekście osiąganych przez dotychczasowe rozwiązania rezultatów". Z czasem wraz z przeprowadzaniem kolejnych badań, zdobyto wiedzę o mechanice i czynnikach wpływających na jakość modeli predykcji wraz z którą odnotowywano również postęp w dokładności klasyfikacji. Znaczący wpływ na rozwój i szybkie osiągnięcie celu miał trafny dobór nowych cech, które jak później się okazało, miały kluczowe znaczenie dla opracowanych modeli predykcji.

Osiągnięte rezultaty dodatkowo poprawiono w drugim z badań, w którym sprawdzono jak na nie wpływa dobór innych – możliwe, że bardziej dopasowanych do charakterystyki danych – klasyfikatorów. Dwa z nich (Gradient Boosting oraz Complement Naive Bayas), które posiadały swojego konkurenta pochodzącego z tej samej rodziny klasyfikatorów (mających wspólną bazę) w pierwszym badaniu, wykazały postęp przesuwając granicę dotychczas najlepszego rezultatu. Należy też zaznaczyć, że trzeci klasyfikator oparty na sieci neuronowej wygenerował dobre rezultaty, lepsze od tych z bazowego modelu jednak gorsze od rozwiązania opartego o drzewa decyzyjne.

Rozwiązaniem trzeciego z problemów badawczych w którym można było oczekiwać kolejnej poprawy rezultatów, było przeprowadzenie operacji dostrajania hiperparametrów. Dzięki dopasowaniu odpowiednich wartości zmiennych do klasyfikatora lasu drzew losowych (ang. Random Forest) osiągnięto ponad 20% przyrost współczynnika korelacji Matthewsa.

Tabela 14. Zestawienie najlepszych modeli predykcji opracowanych w tym badaniu oraz w publikacji bazowej

Model	Precyzja	Czułość	F1	AUC	MCC
Bazowy - RF	0,695	0,079	0,140	0,913	0,226
Opracowany - RF	**0,437 (-37%)**	**0,443 (+461%)**	**0,437 (+212%)**	**0,857 (-6%)**	**0,385 (+70%)**



Rysunek 2. Rezultaty osiągnięte przez najlepszy z opracowanych modeli w kontekście bazowego modelu

Jak wynika z informacji zawartych w tabeli (Tabela 14) - zobrazowanych na rysunku (Rysunek 2) - powstały w niniejszej pracy model predykcji, aż o 70% poprawia wartość głównej metryki MCC. Mimo akceptowalnych spadków precyzji (o 37%) oraz AUC (o 6%) pojawiają się znaczące przyrosty czułości oraz F1 liczone w setkach procent, które wcześniej osiągały niskie poziomy. Opracowane rozwiązanie jest bardziej zbalansowane i uniwersalne. Nowy model wykrywa znacznie więcej pozytywnych wartości, które wcześniej były fałszywie klasyfikowane jako negatywne. Wzrost metryki F1 skupiającej się na wartościach pozytywnych prezentuje niejako bilans zysków i strat zmian czułości i precyzji. Ogół działań pozwolił osiągnąć duży wzrost kluczowej z obserwowanych metryk.

6. Podsumowanie

Podjęcie właściwej decyzji o włączeniu danej biblioteki do zestawu narzędzi wykorzystywanych do rozwoju nowego oprogramowania nie jest łatwym procesem. Jednym z czynników mających na nią wpływ jest okres wsparcia projektów, który w tym badaniu pośrednio starano się przewidzieć. Powstałe w ramach niniejszej pracy modele pozwalają prognozować, czy członkowie wybranego projektu pozostaną i będą brać w nich aktywny udział przez długi czas.

Wybór metryki MCC jako głównego wyznacznika jakości okazał się właściwą decyzją, co potwierdzały przeprowadzone analizy. Jak niejednokrotnie można było zauważyć metryka oparta również na pełnej macierzy błędów – AUC, mimo niskich ocen czułości oraz F1 często utrzymywała wysokie rezultaty.

W trackie rozwiązywania wyznaczonych problemów badawczych potwierdzono skuteczność i istotność procesu dostrajania hiperparametrów klasyfikatorów. Badania obejmujące zachowanie modeli predykcyjnych przy założeniach różnych progów czasowych, pokazują zależność, w której wraz z skracaniem wymaganego okresu rośnie dokładność klasyfikacji. Nie bez znaczenia okazał się właściwy dobór klasyfikatorów do badań. Przemyślany wybór tych dostosowanych do niezbalansowanych danych oraz posiadających mechanizmy wewnętrznego samodoskonalenia jak Gradient Boosting czy ten bazujący na sieci neuronowej, dał możliwość osiągnięcia lepszych rezultatów. Wszystkie przedstawione wyniki osiągnięto dzięki wymienionym metodom jednak kluczowy okazał się właściwy dobór cech. Odrzucenie kilku cech budzących wątpliwości i wprowadzenia nowych trafnych predyktorów dało już na starcie dużą przewagę. Jak wynika z piątego doświadczenia 3 z 4 kluczowych dla modeli cech stanowią te opracowane na potrzeby niniejszej pracy.

Ostatecznie osiągnięto główny cel niemniejszej pracy – opracowanie skuteczniejszej metody klasyfikacji współtwórców z potencjałem na długotrwałą pracę na rzecz projektu GitHub. Uzyskany znaczny wzrost wartości współczynnika korelacji Matthewsa pokazał, że obszar badań nie został jeszcze w pełni wyczerpany. Istotnym faktem jest zachowanie zalet bazowego modelu podczas opracowywani nowego. Zbiór cech pozostał niewielki względem 63 cech z jednego z artykułów z przeglądu literatury, a nawet go zmniejszono. Ponadto do osiągnięcia wymienionych rezultatów podobnie jak w artykule bazowym nie wykorzystywano danych z początkowego okresu po dołączeniu programisty do projektu. Dlatego również nie zostało nałożone ograniczenie - minimum miesięcznego członkostwa twórcy w danym projekcie.

Bibliografia

[1] L. Bao, X. Xia, D. Lo, G.C. Murphy, "A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects", *IEEE Transactions on Software Engineering*, 2021.

[2] V.K. Eluri, T.A. Mazzuchi, S. Sarkani, "Predicting long-time contributors for GitHub projects using machine learning", *Information and Software Technology*, 2021.

[3] T. Wang, Y. Zhang, Y. Yin, H. Wang, "Who will become a long-term contributor? A prediction model based on the early phase behaviors", *ACM International Conference Proceeding Series*, New York, 2018

[4] J. Yao, M. Shepperd, "Assessing software defection prediction performance: Why using the Matthews correlation coefficient matters", *ACM International Conference Proceeding Series*, New York, 2020

[5] D. Chicco, G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation", *BMC Genomics*, 2020

[6] D. Chicco, N. Tötsch, G. Jurman, "The matthews correlation coefficient (Mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation", *BioData Mining*, 2021

[7] H. He, E.A. Garcia, "Learning from imbalanced data", *IEEE Transactions on Knowledge and Data Engineering*, 2009

[8] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, "Feature selection: A data perspective", *ACM Computing Surveys*, 2017

[9] W. Jaworski , "Miary jakości", 2022. [Online]. Available: https://www.mimuw.edu.pl/~wjaworski/SU/SU04_miary_jakosci.pdf

[10] M. Mamczur, "Czym są hiperparametry i jak je dobrać?", 2022. [Online]. Available: https://miroslawmamczur.pl/czym-sa-hiperparametry-i-jak-je-dobrac

[11] G. Malato, "Hyperparameter tuning: Grid search and random search", 2022. [Online]. Available: https://github.com/

[12] "GitHub", 2022. [Online]. Available: https://github.com/

[13] "The GHTorrent project", 2022. [Online]. Available: https://ghtorrent.org

[14] "BigQuery", 2022. [Online] Available: https://cloud.google.com/bigquery

Załączniki

Do niniejszego dokumentu dołączono płytę CD z plikiem „załączniki.zip” zawierającym niezbędne elementy do reprodukcji badań. Poniżej opisano kilka najważniejszych z nich.

- **predictor/inputData/*.csv**

Osiem plików dopasowujących się do powyższego wzorca stanowią zbiory obserwacji omówione w rozdziale 2.4 Opracowane zbiory.

- **predictor/inputData/BigQuery _SQL_queries.txt**

Dokument tekstowy będący zbiorem wszystkich zapytań SQL wykorzystanych do selekcji cech i przygotowania plików CSV. W jego treści etykietą <<LTC_TRESHHOLD>> oznaczano miejsca, w które wprowadza się liczbę. Oznacza ona przyjęty próg czasowy, po którym programista jest określany mianem długookresowego współtwórcy w projekcie.

- **predictor/outputData**

Jest to katalog do którego w postaci plików zapisywane są dane wyjściowe z działania skryptów badawczych.

- **predictor/docker-compose.yml**

Plik konfigurujący narzędzia Docker Compose pozwalający na budowę oraz uruchomienie infrastruktury rozwiązującej problemy badawcze w kontenerze. Domyślna konfiguracja wykonuje wszystkie z pięciu zdefiniowanych doświadczeń, jednak jego niewielka edycja - polegająca na dodaniu argumentu do uruchamianego skryptu powłoki - umożliwia uruchomienie pojedynczego, wybranego zadania.

- **predictor/src**

Jest to katalog zawierający implementacje skryptów badawczych, testujących, pomocniczych, a także pliki konfigurujące kontener i środowisko, takie jak Dockerfile oraz requirements.txt.