



AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Praca dyplomowa

*Metody selekcji cech w analizie dokumentów za pomocą
przetwarzania języka naturalnego*

*Feature selection algorithms in natural language
processing of documents*

Autor:

Norbert Sak

Kierunek studiów:

Automatyka i Robotyka

Opiekun pracy:

dr hab. inż. Jerzy Baranowski

Kraków, 2022

Spis treści

1. Wprowadzenie	4
1.1. Cele pracy	4
1.2. Zawartość pracy	5
2. Analiza sentymentów i uczenie maszynowe	6
2.1. Przetwarzanie języka naturalnego	6
2.2. Analiza sentymentu	6
2.3. Uczenie maszynowe	7
2.4. Algorytmy klasyfikacyjne	8
3. Metody selekcji cech	11
3.1. Selekcja cech	11
3.2. Podział metod selekcji cech	11
3.3. Chi-kwadrat	13
3.4. ANOVA	13
3.5. Rekurencyjna eliminacja cech	14
3.6. Regularyzacja L1	14
4. Użyte technologie	15
4.1. Git	15
4.2. Flask	15
4.3. Heroku	15
4.4. Testy jednostkowe	16
4.5. Ciągła integracja	17
4.6. Dokumentacja kodu	18
5. Implementacja	19
5.1. Projekt uczenia maszynowego	19
5.1.1. Gromadzenie danych	19
5.1.2. Preprocessing danych	20

5.1.3. Ekstrakcja cech	21
5.1.4. Selekcja cech.....	22
5.1.5. Klasyfikacja.....	23
5.2. Aplikacja webowa.....	24
6. Opracowanie wyników	26
6.1. Wyniki klasyfikacji	26
6.1.1. Brak selekcji cech	27
6.1.2. Chi-kwadrat.....	28
6.1.3. ANOVA.....	31
6.1.4. RFE	34
6.1.5. Regularyzacja L1	37
7. Podsumowanie i wnioski.....	40
Bibliografia	42

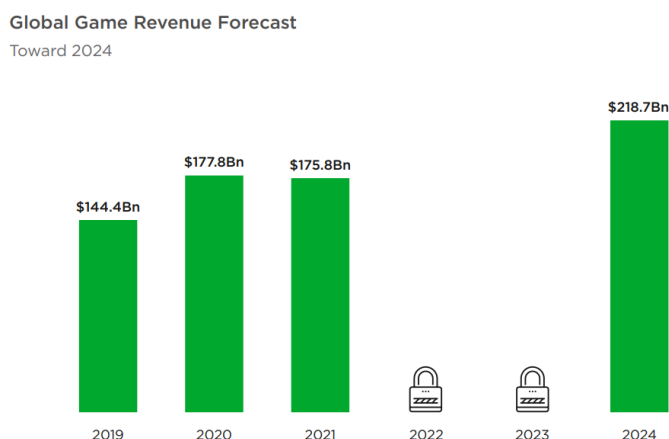
1. Wprowadzenie

W rozdziale przedstawiono cel pracy oraz opis zawartości poszczególnych rozdziałów.

1.1. Cele pracy

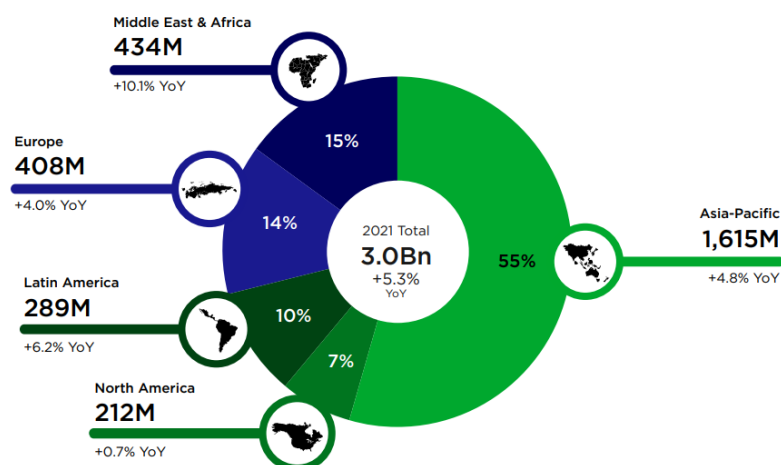
Celem poniższej pracy jest porównanie różnych metod selekcji cech podczas procesu analizy sentymentu (ang. sentiment analysis), czyli określenia nastawienia osoby piszącej przy pomocy przetwarzania języka naturalnego w uczeniu maszynowym.

W pracy jako zbiór danych zostały użyte recenzje gier w języku polskim napisane przez użytkowników jednej z największych platform dystrybucji cyfrowej gier na świecie, czyli Steam. Wybrany został taki obszar merytoryczny, ponieważ rynek gier jest jednym z najbardziej rosnących na świecie. Wartość rynku gier wynosi obecnie około 175 miliardów dolarów i rośnie z roku na rok. Przewiduje się, że w 2024 roku osiągnie wartość bliską 220 miliardów dolarów, co można zaobserwować na rysunku 1.1. Również liczba graczy na świecie rośnie w wysokim tempie. W Europie w 2021 roku wyniosła ona 408 milionów osób, co przekłada się na 4% wzrost w stosunku do roku poprzedniego co wynika z rysunku 1.2.



Rys. 1.1. Wzrost wartości rynku gier na świecie w latach 2019-2024. Przewiduje się, iż do roku 2024 wartość rynku gier na świecie wzrośnie o prawie 25% i osiągnie wartość około 220 miliardów dolarów [1].

2021 Global Players
Per Region



Rys. 1.2. Liczba graczy na świecie z podziałem na regiony w 2021 roku. Liczba graczy na całym świecie w roku 2021 wzrosła o ponad 5%. Jest to potwierdzenie tego, iż rynek gier z roku na rok staje się coraz większy [1].

Proces analizy sentymentu jest bardzo istotny, szczególnie w dzisiejszych czasach, gdzie każda firma stara się jak najbardziej polepszać jakość swoich usług i produktów. Z tego właśnie powodu w poniższej pracy zostały przeanalizowane różne metody selekcji cech za pomocą przetwarzania języka naturalnego w celu usprawnienia tego procesu.

Finalnym efektem poniższej pracy jest porównanie wyników klasyfikacji sentymentu recenzji gier dla różnych metod selekcji cech i wskazanie metody optymalnej, czyli tej, która osiągnie najlepszy wynik dla zbioru testowego oraz stworzenie aplikacji webowej z zaimplementowanym najlepszym modelem.

1.2. Zawartość pracy

Struktura pracy jest następująca. W rozdziale 2 został przedstawiony problem analizy sentymentu oraz uczenia maszynowego. Zostały także w krótki sposób omówione zastosowane algorytmy klasyfikacyjne zastosowane w pracy. W rozdziale 3 omówiona została selekcja cech, jej cele, podział oraz metody zastosowane w poniższej pracy. Technologie zastosowane w realizowanym projekcie zostały przedstawione w rozdziale 4. Rozdział 5 zawiera opis implementacji poszczególnych elementów projektu uczenia maszynowego oraz aplikacji webowej. W rozdziale 6 zaprezentowane i omówione zostały wyniki dokładności, specyficzności, czułości oraz czasu klasyfikacji modeli. W ostatnim rozdziale został podsumowany rezultat całej pracy.

2. Analiza sentymentów i uczenie maszynowe

W rozdziale tym zostanie omówiony proces przetwarzania języka naturalnego oraz analizy sentymentu. Zostanie także omówione uczenie maszynowe i klasyfikatory użyte w projekcie.

2.1. Przetwarzanie języka naturalnego

Przetwarzanie języka naturalnego (ang. Natural Language Processing, NLP) jest to metoda obróbki danych tekstowych, która daje maszynom możliwość wydobywania informacji sformułowanych za pomocą języka. Każda wypowiedź ludzka zawiera bardzo dużą dawkę informacji, a dzięki uczeniu maszynowemu jesteśmy w stanie przekazać te informacje komputerom i wykorzystać je do analizy danych. Przetwarzanie języka naturalnego bardzo szybko się rozwija między innymi dzięki poprawie dostępu do danych oraz zwiększaniu się mocy obliczeniowej. NLP znajduje swoje zastosowanie między innymi w obszarach takich jak finanse, media czy opieka zdrowotna [2]. Przykładowe zastosowania przetwarzania języka naturalnego to:

- pomoc w wykrywaniu fałszywych wiadomości publikowanych w mediach,
- detekcja spamu oraz podejrzanych wiadomości mailowych,
- chat-boty pomagające klientom w sklepach internetowych,
- zbieranie informacji co klienci sądzą o produkcie, usłudze badając informacje ze źródeł takich jak media społecznościowe poprzez analizę sentymentu.

2.2. Analiza sentymentu

Termin analiza sentymentu (ang. sentiment analysis) jest używany w odniesieniu do automatycznego przetwarzania opinii i nastrojów. Analiza sentymentu jest bardzo istotną dziedziną badań w przetwarzaniu języka naturalnego, ponieważ w połączeniu z uczeniem maszynowym pozwala nam na wydobycie opinii i nastroju z zestawu dokumentów [3]. Analiza sentymentu może być badana na różnych poziomach:

- analiza na poziomie całego dokumentu,

- analiza na poziomie jednego zdania,
- analiza na poziomie aspektu.

W tej pracy poruszony został problem analizy sentymentu na poziomie całego dokumentu, czyli ustalenie całościowej opinii, nastroju w dokumencie. Proces ten niesie ze sobą jednak wiele wyzwań i problemów, a niektóre z nich to:

- kontekstowa niejednoznaczność polaryzacji (opinia zawarta w zdaniu często zależy od kontekstu),
- wykrywanie sarkazmu,
- obsługa negacji.

2.3. Uczenie maszynowe

Uczenie maszynowe to proces, w którym program komputerowy lub system jest w stanie się uczyć. Na bardzo podstawowym poziomie uczenie maszynowe wykorzystuje algorytmy do znajdowania wzorców [4]. Uczenie maszynowe możemy podzielić na trzy typy:

- uczenie nadzorowane - zbiór danych dostarczonych maszynie do uczenia zawiera również wyjście (etykietę),
- uczenie nienadzorowane - dostarczone dane posiadają jedynie dane wejściowe (nie zawierają etykiety),
- uczenie przez wzmacnianie - system działa w środowisku zupełnie mu nie znanym, brak zarówno danych wejściowych jak i wyjściowych.

Natomiast algorytmy uczenia maszynowego możemy podzielić na trzy główne rodzaje. Algorytmy klasyfikacyjne, czyli takie, które pozwalają przypisać dane do odpowiednich kategorii (na przykład określenie rasy psa na podstawie różnych jego cech). Kolejnym typem są algorytmy regresyjne, w których zmienna wyjściowa przyjmuje wartości ciągłe (na przykład przewidywanie ceny domu). Istnieją także algorytmy grupujące, czyli takie, których zadaniem jest podzielenie danych na grupy na podstawie ich podobieństw (na przykład podział klientów sklepu na grupy).

W pracy tej użyte zostało uczenie nadzorowane wraz z algorytmami klasyfikacyjnymi. Projekt uczenia maszynowego zawierający przetwarzanie języka naturalnego zwykle składa się z następujących kroków.

1. Zebranie danych.

2. Preprocessing danych.
3. Ekstrakcja cech.
4. Selekcja cech.
5. Uczenie modelu.
6. Klasyfikacja
7. Ewaluacja wyników.

Wymienione powyżej etapy zostały dokładnie omówione w rozdziale 5.

2.4. Algorytmy klasyfikacyjne

W poniższej pracy użyte zostały następujące algorytmy klasyfikacyjne:

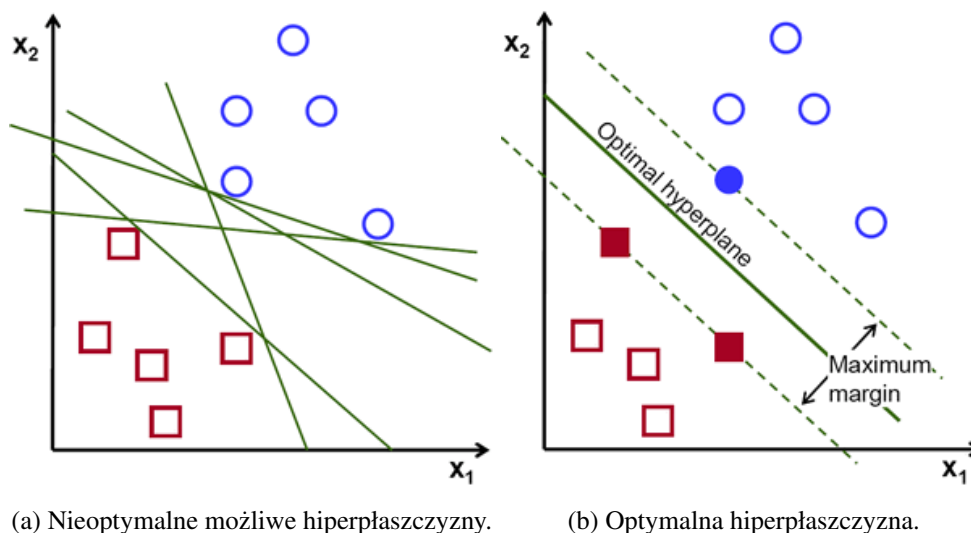
- naiwny klasyfikator Bayesa,
- maszyna wektorów nośnych,
- regresja logistyczna,
- las losowy.

Naiwny klasyfikator Bayesa (ang. Naive Bayes Classifier) to probabilistyczny model uczenia maszynowego używany do problemu klasyfikacji. Opiera się on na twierdzeniu Bayesa:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \quad (2.1)$$

gdzie y to klasa którą przewidujemy, a X reprezentuje poszczególne cechy. Korzystając z powyższego twierdzenia, możemy znaleźć prawdopodobieństwo wystąpienia klasy y , zakładając, że wystąpiła cecha X . Przyjmuje się tutaj założenie, że cechy są niezależne (obecność jednej nie wpływa na obecność drugiej). Dlatego właśnie klasyfikator ten nazywany jest naiwnym [5].

Maszyna wektorów nośnych (ang. Support Vector Machine, SVM) jest to algorytm uczenia maszynowego, którego celem jest znalezienie hiperpłaszczyzny w przestrzeni n -wymiarowej, gdzie n to liczba cech, która wyraźnie klasyfikuje punkty danych [6].



Rys. 2.1. Możliwe hiperpłaszczyzny w dwuwymiarowej przestrzeni cech. Na rysunku (a) zobrazowane zostały różne, nieoptymalne możliwości. Natomiast na rysunku (b) optymalna hiperpłaszczyzna szukana przez algorytm SVM [6].

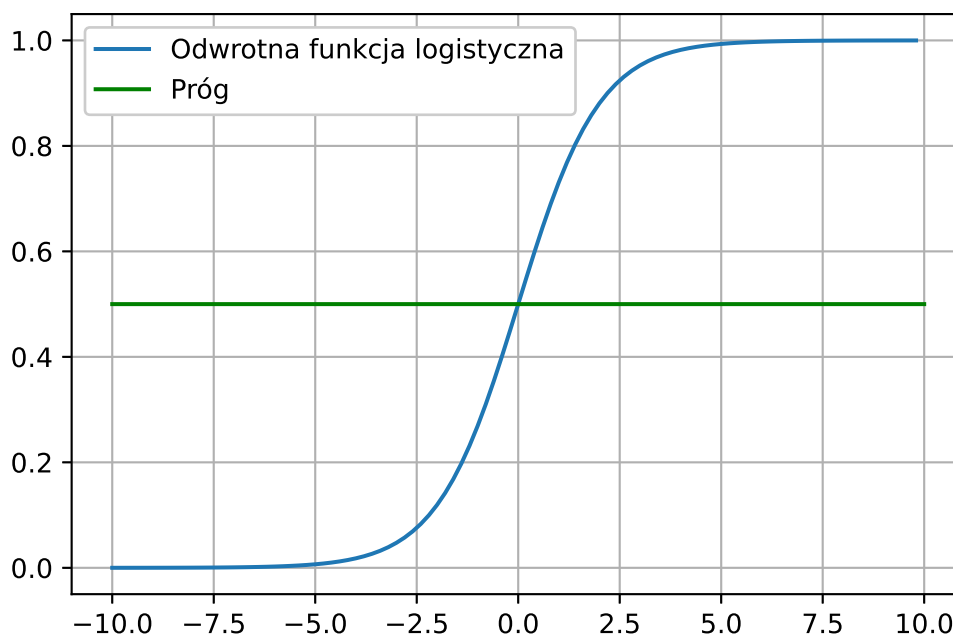
Na rysunku 2.1 można zauważyć przykład wyznaczania hiperpłaszczyzny dla dwóch cech x_1 i x_2 . Celem jest znalezienie takiej płaszczyzny, która ma maksymalny margines (ang. maximum margin), czyli maksymalną odległość pomiędzy punktami z obu klas. Dzięki tej maksymalizacji przyszłe punkty będą klasyfikowane z większą pewnością.

Regresja logistyczna (ang. logistic regression) jest algorytmem klasyfikacji używanym do przypisania obserwacji do dyskretnego zestawu klas. W przeciwieństwie do regresji liniowej, która generuje ciągłe wartości, regresja logistyczna przekształca swoje dane wyjściowe za pomocą odwrotności funkcji logistycznej, w celu otrzymania prawdopodobieństwa, z którego następnie możemy określić do której z klas przynależy dana obserwacja [7]. Odwrotna funkcja logistyczna określona jest następującym wzorem:

$$f(z) = \frac{1}{1 + e^{-z}}, \quad (2.2)$$

gdzie $f(z)$ to rezultat z zakresu od 0 do 1 (oszacowane prawdopodobieństwo), z to dane wejściowe do funkcji (na przykład funkcja liniowa), a e to podstawa logarytmu naturalnego. Dla problemu binarnego ustalany próg (ang. threshold) z zakresu od 0 do 1. Jeśli wartość

funkcji jest większa od progu to przewidywana klasą jest 1, a jeśli jest mniejsza to 0. Funkcja $f(z)$ została przedstawiona na rysunku 2.2 kolorem niebieskim, natomiast kolorem zielonym zaznaczony został próg.



Rys. 2.2. Odwrotna funkcja logistyczna z progiem, który może być zmieniany.

Las losowy (ang. random forest) składa się z dużej liczby pojedynczych drzew decyzyjnych, które działają jako zespół. Każde pojedyncze drzewo dokonuje predykcji, a klasa z największą liczbą głosów jest prognozą modelu. Drzewa chronią się nawzajem przed indywidualnymi błędami, ponieważ, gdy kilka drzew się pomyli wiele innych może mieć rację. Kluczem jest niska korelacja pomiędzy drzewami [8]. Drzewem decyzyjnym nazywamy graficzną metodę wspierania procesu decyzyjnego. Przypomina ono schemat blokowy i składa się węzłów, gałęzi oraz liści. Węzły drzewa tworzone są przez wybraną cechę (początkowy węzeł nazywamy korzeniem), natomiast poszczególne gałęzie reprezentują wartości tej cechy. Węzeł, z którego nie wychodzi żadna gałąź reprezentuje etykietę klasy i nazywany jest liściem [9].

3. Metody selekcji cech

W poniższym rozdziale został przedstawiony problem selekcji cech, podział oraz omówione poszczególne metody selekcji cech wykorzystane w projekcie.

3.1. Selekcja cech

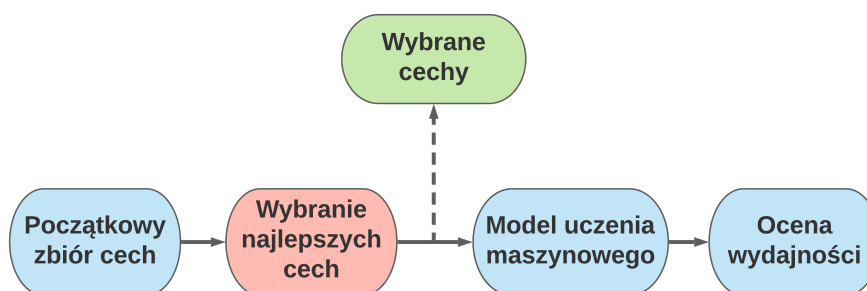
Cecha jest to indywidualna, mierzalna właściwość obserwowanego procesu. Algorytmy uczenia maszynowego wykorzystują zbiór cech, aby dokonać klasyfikacji. Selekcja cech pomaga w:

- zredukowaniu złożoności modelu, przez co jest on łatwiejszy w interpretacji,
- zmniejszeniu czasu uczenia modelu,
- uniknięcie zjawiska nadmiernego dopasowania (ang. overfitting),
- poprawy wydajności klasyfikatora, gdy zostały wybrane właściwe cechy.

Celem selekcji cech jest wybranie, z danych wejściowych, podzbioru cech, które skutecznie opisują te dane i jednocześnie redukują nieistotne cechy zapewniając porównywalnie dobre wyniki predykcji. Aby usunąć nieistotne cechy, potrzebne jest kryterium wyboru, które jest w stanie zmierzyć powiązanie każdej cechy z klasą wyjściową. Z perspektywy uczenia maszynowego jeśli model używa mało znaczących, to wykorzysta je również dla nowych danych, co prowadzi do zbytniego uogólnienia modelu [10].

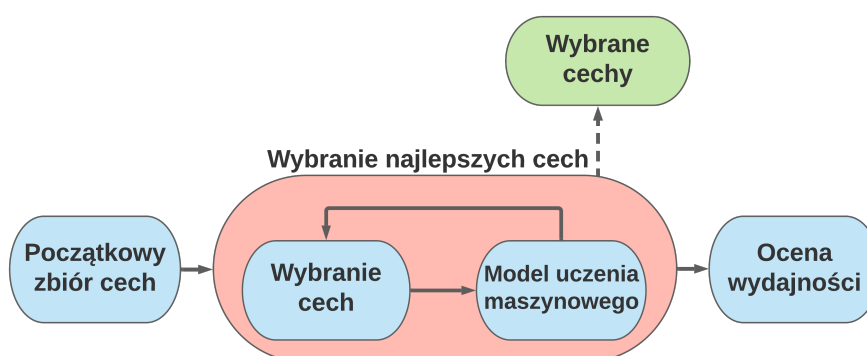
3.2. Podział metod selekcji cech

Metody selekcji cech możemy podzielić na trzy główne grupy. Pierwszą z nich są filtry, które autonomicznie podejmują decyzje na temat wybranych cech. Często korzystają z różnych metod statystycznych. Dla każdej z cech obliczany jest pewien współczynnik na podstawie danego kryterium i dla ustalonego progu wybierana jest odpowiednia liczba cech. Metody te są niezależne od klasyfikatora, szybkie i względnie proste [11, 12]. Graficzna wizualizacja metod filtrowych została przedstawiona na rysunku 3.1.



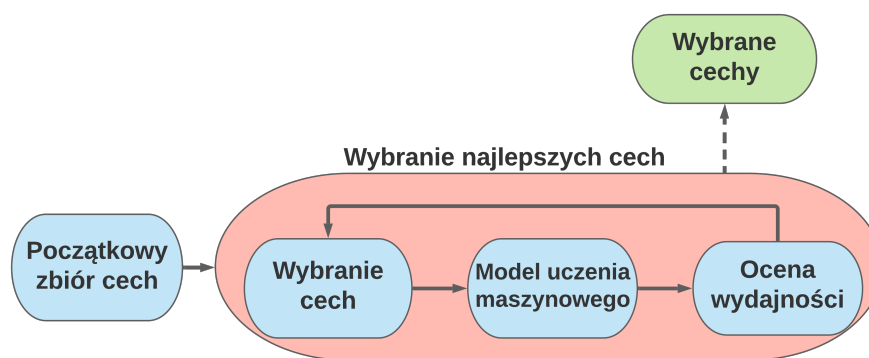
Rys. 3.1. Selekcja cech z użyciem filtrów. Do selekcji wykorzystywane są metody statystyczne, niezależne od modelu uczenia maszynowego, które autonomicznie podejmują decyzje na temat wybranych cech [12].

Kolejnym typem metod selekcji cech są wrappery. Wykorzystują one sprzężenie zwrotne pomiędzy modelem uczenia maszynowego, a elementem decyzyjnym. Wybieramy podzbiór cech i uczymy model z ich wykorzystaniem, a następnie korzystając z wyników dodajemy lub odejmujemy cechy z danego podzbioru. Metody te są zazwyczaj bardzo kosztowne obliczeniowo [11, 12]. Graficzna wizualizacja wrapperów została przedstawiona na rysunku 3.2.



Rys. 3.2. Selekcja cech z użyciem wrapperów. Do selekcji wykorzystywane jest sprzężenie zwrotne pomiędzy modelem uczenia maszynowego, a elementem decyzyjnym. W każdej iteracji dodawane lub odejmowane są kolejne cechy [12].

Do metod selekcji cech należą również metody embedded. Są one zaimplementowane za pomocą algorytmów, które posiadają wbudowane metody selekcji cech. Selekcja dokonywana jest na etapie uczenia modelu. Jedną z metod embedded jest regularyzacja wykorzystująca funkcje celu, które minimalizują błędy dopasowania a także obniżają wagę nieistotnych cech. Elementy z wagą bliską zero są eliminowane [11, 12]. Graficzna wizualizacja metod embedded została przedstawiona na rysunku 3.3.



Rys. 3.3. Selekcja cech z użyciem metod embedded. Do selekcji wykorzystywane są algorytmy z wbudowanymi metodami selekcji cech, a selekcja dokonywana jest na etapie uczenia modelu [12].

3.3. Chi-kwadrat

Metoda chi-kwadrat jest statystyczną metodą filtrową, która mierzy powiązanie pomiędzy cechą a jej klasą. Określa jak bardzo zależne są od siebie dana cecha i klasa. Im bardziej zależna jest ich relacja, tym bardziej dana cecha jest użyteczna w predykcji danej klasy. Na poniższym wzorze A to liczba wystąpień f (ang. feature - cecha) i c (ang. class - klasa) razem, B to liczba wystąpień f bez c , C to liczba wystąpień c bez f , D to liczba przypadków, gdy f i c nie występują, a N to całkowita liczba analizowanych dokumentów [13].

$$\chi^2(f, c) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \quad (3.1)$$

χ^2 jest wartością znormalizowaną, której wartość można porównywać z z cechami z tej samej kategorii. Jednak, gdy cecha występuje z niską częstotliwością w danym dokumencie to wspomniana normalizacja przestaje występować. Stąd możemy wywnioskować, iż ta metoda nie jest odpowiednia dla cech o niskiej częstotliwości. Jednak pomimo tego metoda chi-kwadrat jest uważana za jedną z dwóch najlepszych filtrowych metod selekcji cech wraz z metodą Information Gain [13].

3.4. ANOVA

Analiza wariancji (ang. analysis of variance, ANOVA) jest kolejną z metod filtrowych. Polega na sprawdzeniu istotności różnic pomiędzy dwiema lub większej liczbie próbek. Metodę tę można podsumować jako obliczenie wielu średnich oraz wariancji i finalne określenie istotności ich stosunku. F-test służy do porównywania czynników sumy odchylenia, a formuła wygląda następująco:

$$F_{\text{values}} = \frac{MSG}{MSE} = \frac{SSG/df_g}{SSE/df_e}, \quad (3.2)$$

gdzie MSG to różnica pomiędzy zabiegami, MSE to wariancja w obrębie zabiegów, SSG oznacza grupę sumy kwadratów, SSE sumę błędu kwadratowego, a df_g i df_e są ich stopniami swobody [14].

3.5. Rekurencyjna eliminacja cech

Rekurencyjna eliminacja cech (ang. recursive feature elimination, RFE) jest wrapperem i polega na użyciu zewnętrznego estymatora, który każdej z cech przypisuje wagę (na przykład współczynniki modelu liniowego). Celem rekurencyjnej jest wybranie cech poprzez rekurencyjne uwzględnienie coraz mniejszych podzbiorów cech. Estymator szkolony jest na początkowym zestawie cech, a waga każdej z cech uzyskiwana jest poprzez dowolny określony atrybut (na przykład atrybut `coef_`). Następnie najmniej istotne cechy są usuwane, a estymator uczony jest na nowym zestawie cech. Procedura ta powtarzana jest do momentu, aż osiągniemy pożądaną liczbę cech [15].

3.6. Regularyzacja L1

Regularyzacja L1 jest metodą embedded, która korzysta z regresji Lasso, czyli modelu liniowego wykorzystującego następującą funkcję kosztu:

$$\frac{1}{2N} \sum_{i=1}^N (y_{\text{real}}^{(i)} - y_{\text{pred}}^{(i)})^2 + \alpha \sum_{j=1}^N |a_j|, \quad (3.3)$$

gdzie N jest liczebnością zbioru treningowego, $y_{\text{real}}^{(i)}$ i $y_{\text{pred}}^{(i)}$ to odpowiednio klasa prawdziwa i klasa przewidziana przez model. Ostatni składnik powyższej funkcji nazywany jest karą L1, α jest hiperparametrem, który pozwala dostosować intensywność kary, a a_j jest współczynnikiem j -tej cechy. Im wyższy współczynnik cechy, tym wyższa wartość funkcji kosztu [16].

Podczas selekcji cech regresja Lasso, w celu zminimalizowania funkcji kosztu, automatycznie wybierze przydatne cechy, a zbędne wyeliminuje. Odrzucenie obiektu powoduje, że jego współczynnik będzie równy zero. Regresja Lasso jest dopasowywana do zbioru danych i brane są pod uwagę tylko cechy, które mają współczynnik różny od zera. Oczywiście do poprawnego działania selekcji wymagane jest odpowiednie dostrojenie parametru α .

4. Użyte technologie

W tym rozdziale opisane zostały technologie użyte podczas realizacji projektu takie jak: Git, Flask, Heroku, testy jednostkowe, CI oraz dokumentacja kodu.

4.1. Git

Git to system kontroli wersji, czyli oprogramowanie służące do śledzenia zmian w kodzie źródłowym. W projekcie użyty został najpopularniejszy serwis internetowy wykorzystujący system kontroli wersji Git, czyli GitHub. Głównymi jego zaletami jest możliwość śledzenia wszystkich dokonywanych zmian w kodzie w czasie oraz udostępnianie swojego kodu innym osobom w łatwy i przyjemny sposób.

4.2. Flask

Flask jest to mikro framework do tworzenia aplikacji webowych w języku Python. Stosowany jest on do tworzenia stosunkowo prostych aplikacji internetowych. W projekcie został użyty do stworzenia prostej aplikacji webowej z zaimplementowanym modelem uczenia maszynowego przewidującego recenzje gier. Zaletą Flaska jest względna prostota w porównaniu do, na przykład, frameworka Django.

4.3. Heroku

Heroku to platforma chmurowa, której działanie skupia się na kontenerach wykorzystywanych do hostowania aplikacji webowych. W projekcie wykorzystany został do opublikowania aplikacji webowej napisanej przy pomocy Flaska. Głównymi zaletami Heroku jest prostota oraz możliwość bezpłatnego korzystania z usług tej platformy.

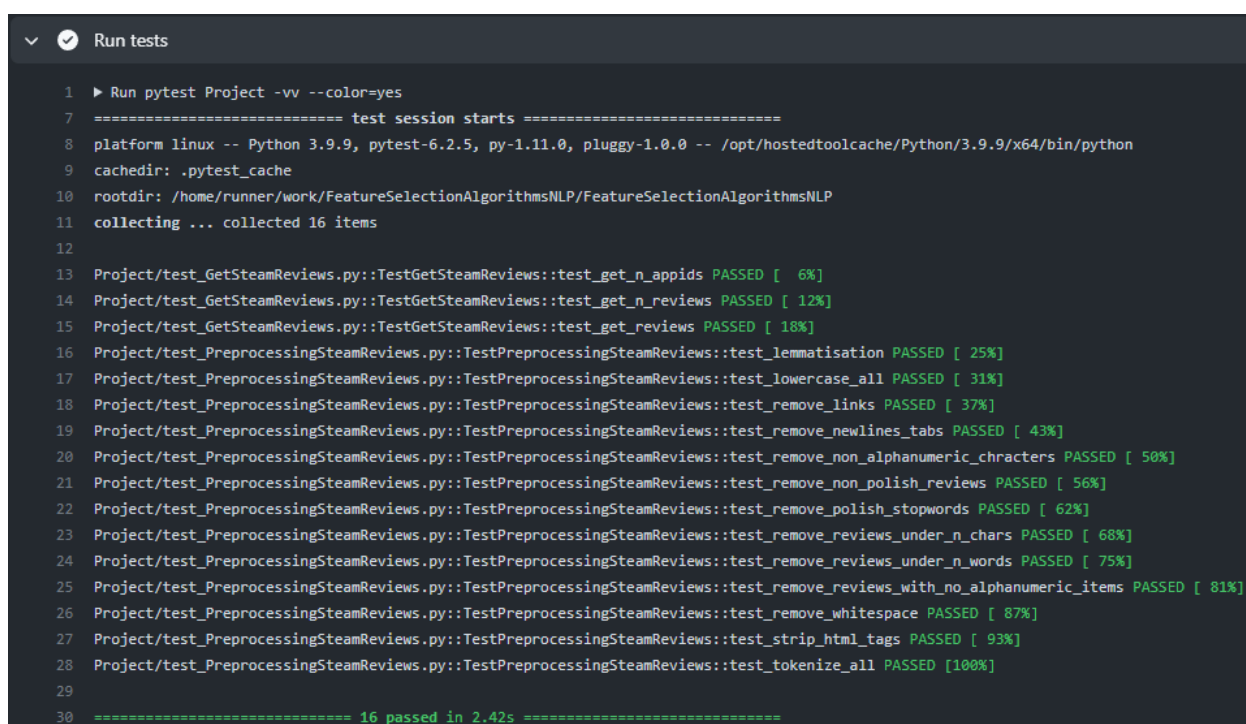
4.4. Testy jednostkowe

Testy jednostkowe (ang. unit tests) jest to jedna z metod testowania oprogramowania polegająca na wykonywaniu testów sprawdzających poprawne działanie pojedynczych elementów kodu (na przykład funkcji, metod klasy). W projekcie testy jednostkowe zostały napisane dla metod klasy odpowiedzialnej za preprocessing recenzji gier oraz dla funkcji do zbierania recenzji z API Steam. Do napisania testów wykorzystany został wbudowany w język Python framework *unittest*. Przetestowane została poprawność działania metod i funkcji odpowiedzialnych za:

- zbieranie ID gier z API platformy Steam,
- zdobywanie informacji o pojedynczej recenzji z API platformy Steam,
- zbieranie określonej liczby recenzji z pojedynczej strony w API platformy Steam,
- usuwanie z recenzji elementów języka HTML,
- eliminacje znaków nowej linii oraz tabulacji z recenzji,
- usuwanie nadmiernych znaków spacji z recenzji,
- eliminacje z recenzji znaków, które nie są alfanumeryczne,
- usuwanie z recenzji polskich stopwordów,
- lematyzacje recenzji
- eliminacje linków z recenzji,
- ujednolicenie liter w recenzji na małe,
- tokenizowanie recenzji,
- usuwanie recenzji napisanych w języku innym niż polski,
- eliminacje recenzji, które nie zawierają żadnych znaków alfanumerycznych,
- usuwanie recenzji, które zawierają liczbę znaków poniżej określonego progu,
- eliminacje recenzji, które zawierają liczbę słów poniżej określonego progu.

4.5. Ciągła integracja

Ciągła integracja (ang. Continuous Integration, CI) jest to jedna z praktyka używana podczas tworzenia oprogramowania polegająca na każdorazowej weryfikacji zmian w kodzie poprzez wykonanie testów oraz ewentualnego zbudowanie projektu. W projekcie CI zostało wdrożone przy pomocy sekcji *Actions* na GitHubie i stworzenia workflowu odpalającego testy jednostkowe przy każdym pushu do repozytorium. Na rysunku 4.1 przedstawiony został rezultat testów puszczanych podczas wrzucania zmian w projekcie.



```
Run tests

1 ▶ Run pytest Project -vv --color=yes
7 ===== test session starts =====
8 platform linux -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0 -- /opt/hostedtoolcache/Python/3.9.9/x64/bin/python
9 cachedir: .pytest_cache
10 rootdir: /home/runner/work/FeatureSelectionAlgorithmsNLP/FeatureSelectionAlgorithmsNLP
11 collecting ... collected 16 items
12
13 Project/test_GetSteamReviews.py::TestGetSteamReviews::test_get_n_appids PASSED [ 6%]
14 Project/test_GetSteamReviews.py::TestGetSteamReviews::test_get_n_reviews PASSED [ 12%]
15 Project/test_GetSteamReviews.py::TestGetSteamReviews::test_get_reviews PASSED [ 18%]
16 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_lemmatisation PASSED [ 25%]
17 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_lowercase_all PASSED [ 31%]
18 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_links PASSED [ 37%]
19 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_newlines_tabs PASSED [ 43%]
20 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_non_alphanumeric_characters PASSED [ 50%]
21 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_non_polish_reviews PASSED [ 56%]
22 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_polish_stopwords PASSED [ 62%]
23 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_reviews_under_n_chars PASSED [ 68%]
24 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_reviews_under_n_words PASSED [ 75%]
25 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_reviews_with_no_alphanumeric_items PASSED [ 81%]
26 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_remove_whitespace PASSED [ 87%]
27 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_strip_html_tags PASSED [ 93%]
28 Project/test_PreprocessingSteamReviews.py::TestPreprocessingSteamReviews::test_tokenize_all PASSED [100%]
29
30 ===== 16 passed in 2.42s =====
```

Rys. 4.1. Rezultat testów jednostkowych na GitHubie.

4.6. Dokumentacja kodu

W projekcie dokumentacja kodu została zapewniona przy użyciu docstringów w Pythonie. Docstring jest to ciąg znaków określony w kodzie źródłowym, używany podobnie jak komentarz, do udokumentowania określonego fragmentu kodu (na przykład metody klasy). Taki dokument staje się specjalnym atrybutem `__doc__` obiektu. Docstringi napisane zostały dla metod klasy odpowiedzialnej za preprocessing recenzji gier oraz dla funkcji do zbierania recenzji z API Steam. Implementacja jednego z docstringów została przedstawiona na rysunku 4.2, natomiast na rysunku 4.3 został przedstawione wywołanie atrybutu `__doc__`.

```
def strip_html_tags(self, text: str) -> str:
    """
    Metoda usuwająca wszystkie elementy składni języka HTML z recenzji.
    Parameters:
        text (str): Recenzja wejściowa
    Returns:
        str: Przeprosesowana recenzja
    """
    soup = BeautifulSoup(text, "html.parser")
    stripped_text = soup.get_text(separator=" ")
    return stripped_text
```

Rys. 4.2. Przykładowy docstring (kolor zielony) dla funkcji `strip_html_tags`.

```
Metoda usuwająca wszystkie elementy składni języka HTML z recenzji.
Parameters:
    text (str): Recenzja wejściowa
Returns:
    str: Przeprosesowana recenzja
```

Rys. 4.3. Wygląd atrybutu `__doc__` dla powyższego przykładu.

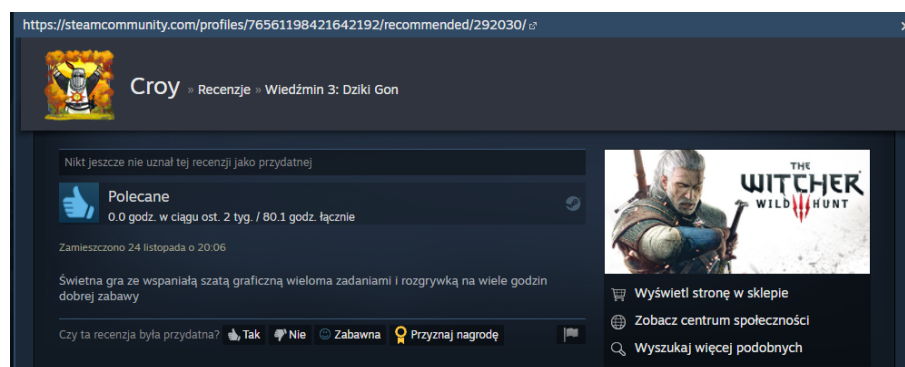
5. Implementacja

W poniższym rozdziale zostały przedstawione kolejne kroki implementacji realizowanego projektu uczenia maszynowego oraz aplikacji webowej.

5.1. Projekt uczenia maszynowego

5.1.1. Gromadzenie danych

Dane, czyli recenzje gier użytkowników platformy Steam zostały zebrane przy pomocy API powyższej platformy. Wybrane zostały recenzje 25 aktualnie najbardziej popularnych gier w języku polskim z kategorii RPG (ang. role-playing game), czyli gier fabularnych. Recenzje zostały zawężone do jednej kategorii aby użyte w recenzjach słownictwo nie różniło od siebie w znaczący sposób. Na rysunku 5.1 można zaobserwować przykładowy wygląd recenzji z platformy Steam. Pod uwagę bierzemy jedynie treść recenzji oraz etykietę, czyli informację czy gra jest polecana przez użytkownika. Referencyjny sentyment recenzji określany był na podstawie tego, czy użytkownik polecił grę (jeżeli polecił to recenzja została uznana za pozytywną, a w przeciwnym wypadku recenzji przypisywana była klasa negatywna).



Rys. 5.1. Przykładowa recenzja z platformy Steam [17]. Wykorzystywanymi w pracy elementami jest treść recenzji oraz informacja, czy gra jest polecana przez użytkownika.

Do analizy zostały wyselekcjonowane recenzje o długości minimum 100 znaków, aby wyeliminować opinie krótkie, których treść często jest niezwiązana z wystawioną oceną i posiadają wyłącznie charakter żartobliwy co mogłoby negatywnie wpłynąć na jakość modelu uczenia maszynowego.

5.1.2. Preprocessing danych

Kolejnym krokiem było przetworzenie zebranych danych (ang. preprocessing), tak aby były one w jak najlepszej formie do zastosowania w uczeniu maszynowym. Głównym zadaniem przetwarzania było usunięcie zbędnych i niepotrzebnych elementów, które mogłyby jedynie spowodować problemy podczas uczenia oraz jak największe uproszczenie dokumentów, tak aby zawierały one jedynie najbardziej istotne słowa. Preprocessing składał się z następujących kroków.

1. Usunięcie wszystkich znaków modyfikacji (ang. escape characters) odpowiedzialnych na przykład za znak nowej linii czy tabulatora. W Pythonie są to znaki `\n` i `\t`. Znaki te w żaden sposób nie byłyby pomocne podczas analizy sentymentu tekstu.
2. Eliminacja z analizowanych dokumentów elementów składni języka HTML, takich jak między innymi `<h>`, `
` czy `<p>`. Elementy te często zostają zebrane podczas gromadzenia danych, jednak nie wnoszą one niczego podczas analizy sentymentu. Mogą jedynie mieć negatywny wpływ na działanie modelu.
3. Usunięcie zbędnych znaków spacji, tak aby nigdzie nie występował więcej niż jeden znak spacji.
4. Pozbycie się wszystkich znaków interpunkcyjnych oraz pozostawienie znaków tylko i wyłącznie z polskiego alfabetu, ponieważ praca zakłada analizę dokumentów w języku polskim. Pozwoli to także na wyeliminowanie z tekstu różnego rodzaju emotikonów i tym podobnych elementów, które mogłyby być bardzo ciężkie do poprawnego przeanalizowania podczas analizy sentymentu.
5. Eliminacja linków do różnego rodzaju stron z zebranych danych, które nie byłyby pomocne podczas analizy sentymentu recenzji.
6. Pomimo użycia filtru, aby zebrać tylko i wyłącznie recenzje w języku polskim z API Steam, pojawiły się również nieliczne recenzje w innych językach, głównie angielskie. Ponieważ projekt zakłada analizę recenzji polskich, istotnym krokiem było usunięcie dokumentów w innym języku. Do tego kroku wykorzystana została funkcja `detect` z biblioteki

`textttlangdetect` w Pythonie odpowiedzialna za detekcję języka, który dominuje w danym tekście.

7. Zamienienie wszystkich liter na małe (ang. lowercase) przy pomocy wbudowanej w Pythonie funkcji `lower`. Celem tego kroku jest to, aby przykładowo słowa *dobry* i *Dobry* nie były traktowane jako dwa osobne słowa.
8. Eliminacja stopwords, czyli słów o małym znaczeniu takich jak spójniki (na przykład: a, i, oraz) czy słowa popularne, które nie wpływają na identyfikację analizowanego tekstu.
9. Lematyzacja analizowanych dokumentów, czyli sprowadzenie każdego słowa do jego formy podstawowej, co zostało zademonstrowane w tabeli 5.1.

Tabela 5.1. Przykład lematyzacji.

Przed lematyzacją	Po lematyzacji
zagrałem	zagrać
zagrał	zagrać
zagrali	zagrać
zagrać	zagrać

Do lematyzacji użyty został analizator i generator fleksyjny Morfeusz 2 [18], a dokładnie jego biblioteka w Pythonie `morfeusz2`.

Poniżej w tabeli 5.2 zaprezentowany został wynik opisanego powyżej wstępnego przetwarzania danych na przykładzie jednego zdania.

Tabela 5.2. Przykładowy rezultat przetwarzania zebranych danych.

Przed przetwarzaniem	Po przetwarzaniu
Ukończyłem już ponad połowę gry i jestem bardzo zadowolony z zakupu.\n	ukończyć połowa zadowolić zakup

Finalnie z 75 tysięcy zebranych recenzji po przetworzeniu danych otrzymaliśmy około 5 tysięcy recenzji nadających się do użycia w dalszej części projektu.

5.1.3. Ekstrakcja cech

Następnym krokiem jest ekstrakcja cech z przetworzonych danych przy pomocy funkcji `TfidfVectorizer` z najpopularniejszej biblioteki przeznaczonej do uczenia maszynowego jaką jest `scikit-learn`. Ekstrakcja cech polega na przekonwertowaniu danych tekstowych

na wektory cech. Na samym początku wykonywana jest wektoryzacja danych, czyli zamiana tekstu na macierz zliczeń poszczególnych słów. Dzięki temu otrzymujemy dane, które mogą być już interpretowane przez model uczenia maszynowego. W tym przypadku użyty został podział słów na unigramy (pojedyncze słowa) i bigramy (pary słów). Następnie dokonywane jest przekształcenie macierzy przy pomocy transformacji TFIDF (ang. TF – term frequency, IDF – inverse document frequency), która oblicza wagi poszczególnych słów na podstawie liczby ich wystąpień. TFIDF obliczane jest ze wzoru:

$$TF \cdot IDF = \frac{n_{i,j}}{n_{k,j}} \cdot \log \left(\frac{N}{m_i} \right), \quad (5.1)$$

gdzie:

TF - częstotliwość słowa i w dokumencie j ,

IDF - odwrotna częstotliwość słowa i we wszystkich dokumentach,

$n_{i,j}$ - liczba wystąpień słowa i w dokumencie j ,

$n_{k,j}$ - liczba wystąpień wszystkich słów k w dokumencie j ,

N - liczba wszystkich dokumentów,

m_i - liczba dokumentów, które zawierają co najmniej jedno słowo i .

Oczywiście przed użyciem funkcji `TfidfVectorizer` dane zostały podzielone przy pomocy funkcji `train_test_split`, gdzie 70% to dane treningowe. Następnie wykonywana jest metoda `fit_transform` na danych treningowych aby uniknąć przetrenowania modelu. Dopiero na końcu wykonana zostaje osobno metoda `transform` dla danych testowych. W tym momencie zbiór treningowy posiada ponad 116 tysięcy kolumn, czyli cech, które będą redukowane do minimum przy pomocy ich selekcji w dalszej części pracy.

5.1.4. Selekcja cech

W końcu przyszła pora na kluczowy element owej pracy, czyli selekcje cech. W pracy użyte zostały metody opisane w rozdziale trzecim, czyli:

- Chi-kwadrat - zaimplementowana przy pomocy funkcji `SelectKBest` z biblioteki `scikit-learn` z parametrem `score_func=chi2`, która wybiera k najlepszych cech. W tym przypadku użyte zostały k należące do zbioru {500, 1000, 1500, 2000, 2500}.
- ANOVA - również zaimplementowana przy pomocy funkcji `SelectKBest` z biblioteki `scikit-learn` z parametrem `score_func=f_classif`, która wybiera k najlepszych cech. W tym przypadku użyte zostały k należące do zbioru {500, 1000, 1500, 2000, 2500}.

- RFE - zaimplementowana przy pomocy funkcji `RFE` z biblioteki `scikit-learn`, która stopniowo eliminuje cechy z krokiem `step=20000` (wartość kroku oznacza liczbę cech eliminowanych podczas każdej iteracji), aż osiągniemy pożądaną liczbę cech `k` należące do zbioru `{500, 1000, 1500, 2000, 2500}` przy pomocy estymatora regresji logistycznej.
- Regularyzacja L1 - zaimplementowana przy pomocy funkcji `SelectFromModel` z biblioteki `scikit-learn` z użyciem estymatora `LinearSVC` z karą `penalty=l1` (regresja Lasso) oraz parametrem `C` (parametr regularyzacji, moc regularyzacji jest odwrotnie proporcjonalna do wartości `C`) odpowiednio w zakresie od 0.5 do 5 z krokiem 0.2 co przekłada się na liczbę wybranych cech od 143 do 1375.

5.1.5. Klasyfikacja

Następnie po selekcji i ekstrakcji cech zbudowane zostały klasyfikatory opisane w rozdziale drugim z użyciem funkcji `GridSearchCV` z biblioteki `scikit-learn`, która odpowiada za dopasowanie najlepszych parametrów modelu. Podawane są parametry, które chcemy przetestować. `GridSearchCV` przechodzi przez wszystkie kombinacje parametrów i tworzy najlepszą z nich w oparciu o wybraną metrykę punktacji (na przykład dokładność). Dodatkowo dla każdego układu parametrów wykonywany jest sprawdzian krzyżowy (ang. *cross validation*), czyli podział danych na kilka podzbiorów i przeprowadzenia analizy na niektórych z nich (domyślnie jest ich 5 i taka też liczba została użyta w projekcie). Użyte klasyfikatory i ich parametry do przeszukiwania siatki (ang. *grid search*) to:

- `MultinomialNB` - wielomianowy naiwny klasyfikator Bayesa z parametrem `alpha` (parametr wygładzania addytywnego, zero oznacza brak) ze zbioru `{0.001, 0.01, 0.1, 1, 10, 100}`,
- `SVC` - klasyfikator wektorów nośnych z parametrami `C` (parametr regularyzacji) ze zbioru `{1, 10, 100, 1000}` oraz `gamma` (współczynnik jądra) należące do zbioru `{1, 0.1, 0.01, 0.001}`,
- `LogisticRegression` - regresja logistyczna z parametrami `C` (parametr regularyzacji) ze zbioru `{0.001, 0.01, 0.1, 1, 10, 100, 1000}` oraz `penalty` (określa normę kary) należące do zbioru `{'l1', 'l2', 'elasticnet'}`,
- `RandomForestClassifier` - klasyfikator lasu losowego z parametrem `n_estimators` (liczba drzew w lesie) ze zbioru `{100, 200, 400, 600}`.

Każdy z klasyfikatorów został wytrenowany dla najlepszego zestawu parametrów po użyciu `GridSearchCV` na zbiorze treningowym. Następnie dokonano klasyfikacji dla zbioru testowego, by porównać osiągi modeli. Proces ten został wykonany dla każdego ze zbioru wybranych cech opisanych w rozdziale 5.1.4. Ewaluacja wyników poszczególnych klasyfikatorów

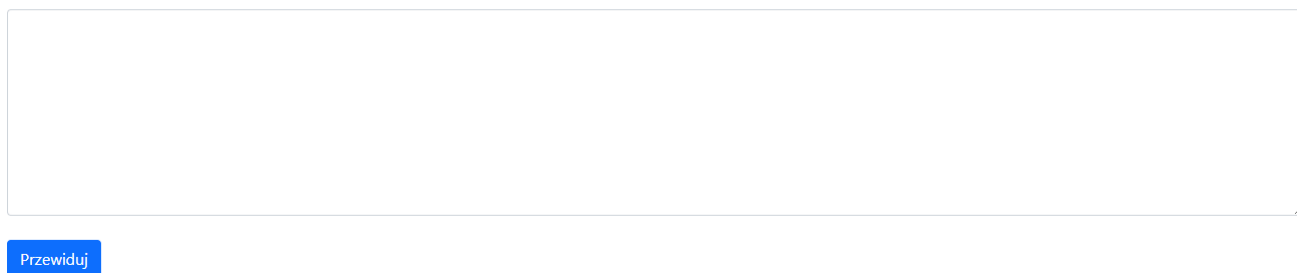
(dokładność, specyficzność i czułość) oraz czas klasyfikacji zostały zaprezentowane i opisane w rozdziale 6.

5.2. Aplikacja webowa

W ramach projektu zaimplementowana również została aplikacja webowa przy pomocy mikro frameworka Flask w Pythonie oraz wdrożona przy pomocy platformy Heroku. Do implementacji strony został również użyty framework Bootstrap ułatwiający tworzenie graficznego interfejsu stron internetowych.

Do aplikacji zaimplementowany został najlepszy uzyskany model uczenia maszynowego. Na stronie znajduje się pole tekstowe do wpisania recenzji przez użytkownika oraz przycisk do wykonania predykcji. Zostało to zobrazowane na rysunku 5.2.

Predykcja sentymentu recenzji gier z zastosowaniem selekcji cech



Rys. 5.2. Pole do wprowadzania recenzji przez użytkownika.

Po wprowadzeniu i zatwierdzeniu recenzji otrzymujemy informacje takie jak:

- recenzje wejściową wpisaną przez użytkownika,
- recenzje po preprocessingu,
- wynik predykcji,
- tabele zawierającą do dziesięciu najbardziej znaczących cech podczas wykonywania predykcji przez model wraz z ich istotnością (wartość od 0 do 1).

Działanie aplikacji dla przykładowej recenzji zostało przedstawione na rysunku 5.3.

Recenzja wejściowa

Gra od samego początku chwyciła mnie za serce. Bardzo dobra fabuła, której towarzyszy grafika na bardzo wysokim poziomie. Jedyny minus to lagi podczas ładowania lokacji. Zakup zaliczam do udanych.

Recenzja po preprocessingu

początek chwycić serce dobra fabuła towarzysz wysoki poziom jedyny minus laga ładować lokacja zakup zaliczać udać

Wynik predykcji → **POZYTYWNA**

Cecha	Istotność (od 0 do 1.0)
jedyny minus	0.23747024692352567
ładować	0.22541680622538834
serce	0.21933390112399256
wysoki poziom	0.2120740798941144
laga	0.20931054149711772
jedyny	0.19202606744423276
udać	0.19144006520643053
wysoki	0.165413834018017
minus	0.1544693024280755
dobra	0.1542291958855427

Rys. 5.3. Przykładowe działanie aplikacji webowej.

6. Opracowanie wyników

W poniższym rozdziale zaprezentowane zostały wyniki klasyfikacji. Pod uwagę wzięte zostały miary takie jak dokładność, specyficzność i czułość. Porównane również zostały czasy klasyfikacji danych testowych.

6.1. Wyniki klasyfikacji

Wyniki klasyfikacji można przedstawić za pomocą macierzy błędów (ang. confusion matrix). Jest to najpopularniejszy sposób oceny jakości klasyfikacji, a jej wygląd został przedstawiony w tabeli 6.1.

Tabela 6.1. Macierz błędów.

		Wynik faktyczny	
		pozytywny	negatywny
Przewidywany wynik	pozytywny	TP	FP
	negatywny	FN	TN

TP – (ang. True Positive) liczba przewidywań fałszywie pozytywnych

TN – (ang. True Negative) liczba przewidywań prawdziwie negatywnych

FP – (ang. False Positive) liczba przewidywań fałszywie pozytywnych

FN – (ang. False Negative) liczba przewidywań fałszywie negatywnych

Pierwszą z zastosowanych miar jest dokładność (ang. accuracy), która określa jaka część ze wszystkich klasyfikowanych obserwacji została zaklasyfikowana poprawnie. Jest to najbardziej popularny wskaźnik jakości klasyfikacji. Zostanie ona porównana z wynikiem otrzymanym bez użycia żadnej z analizowanych metod selekcji cech. Można ją obliczyć ze wzoru:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \quad (6.1)$$

Kolejną z miar zastosowanych w niniejszej pracy jest specyficzność (ang. specificity), często zapisywana jako TNR (ang. True Negative Rate). Jest to miara pokrycia określająca w jakim stopniu klasa faktycznie negatywna została przewidziana jako negatywna. Jest ona opisana wzorem:

$$TNR = \frac{TN}{TN + FP}. \quad (6.2)$$

Ostatnią z zastosowanych miar oceny jakości modelu jest czułość (ang. sensitivity), często zapisywana jako TPR (ang. True Positive Rate). Jest to miara pokrycia określająca w jakim stopniu klasa faktycznie pozytywna została przewidziana jako pozytywna. Można ją obliczyć ze wzoru:

$$TPR = \frac{TP}{TP + FN}. \quad (6.3)$$

Dla każdej z metod selekcji cech wykonane zostały również pomiary czasu selekcji dla zbioru testowego, aby zobrazować zysk czasowy po zmniejszeniu liczby cech.

6.1.1. Brak selekcji cech

Na samym początku zaprezentowane zostaną wyniki uzyskane bez użycia żadnej z omawianych metod selekcji cech. W tym przypadku liczba cech wynosiła ponad 116 tysięcy. Wyniki te będą potrzebne, aby porównać je z tymi uzyskanymi dla zastosowanych metod selekcji cech. W tabeli 6.2 zostały zaprezentowane wyniki dokładności, natomiast w tabeli 6.3 czasowe wyniki klasyfikacji dla zbioru testowego.

Tabela 6.2. Dokładność klasyfikacji modeli bez zastosowania selekcji cech.

	Dokładność
MultinomialNB	0.86
SVC	0.84
LogisticRegression	0.84
RandomForestClassifier	0.83

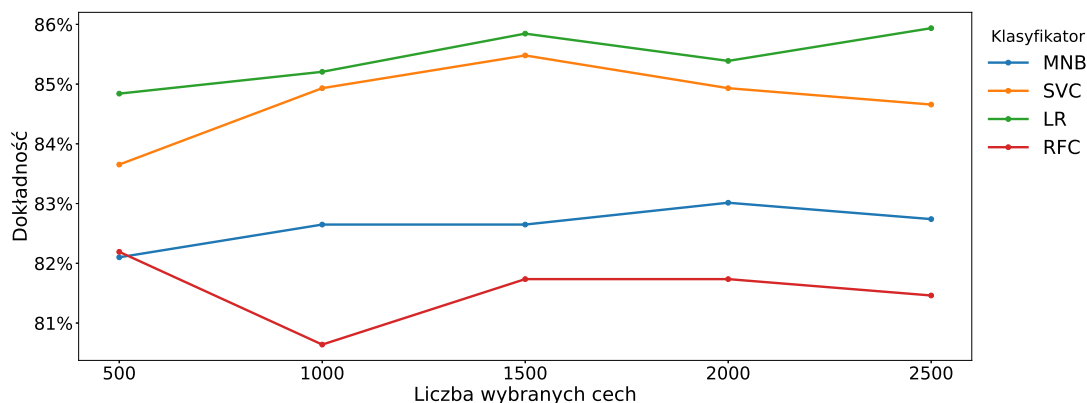
Tabela 6.3. Czas klasyfikacji modeli dla zbioru testowego bez zastosowania selekcji cech.

	Czas klasyfikacji [s]
MultinomialNB	1.13
SVC	550
LogisticRegression	1.01
RandomForestClassifier	1.25

Najlepsza dokładność została osiągnięta dla klasyfikatora MultinomialNB i wyniosła ona 86 procent. Jednakże dla wynik ten dla pozostałych klasyfikatorów był tylko minimalnie gorszy. Natomiast czas klasyfikacji dla klasyfikatorów MultinomialNB, LogisticRegression, RandomForestClassifier wyniosły lekko ponad 1 sekundę, natomiast dla SVC było to aż 550 sekund. Czas obliczeniowy dla klasyfikatora wektorów nośnych rośnie znacznie wraz ze zwiększeniem liczby wektorów.

6.1.2. Chi-kwadrat

Pierwszą z użytych metod selekcji cech była metoda filtrowa chi-kwadrat. Na rysunku 6.1 zobrazowany został wykres dokładności modeli względem liczby wybranych cech.

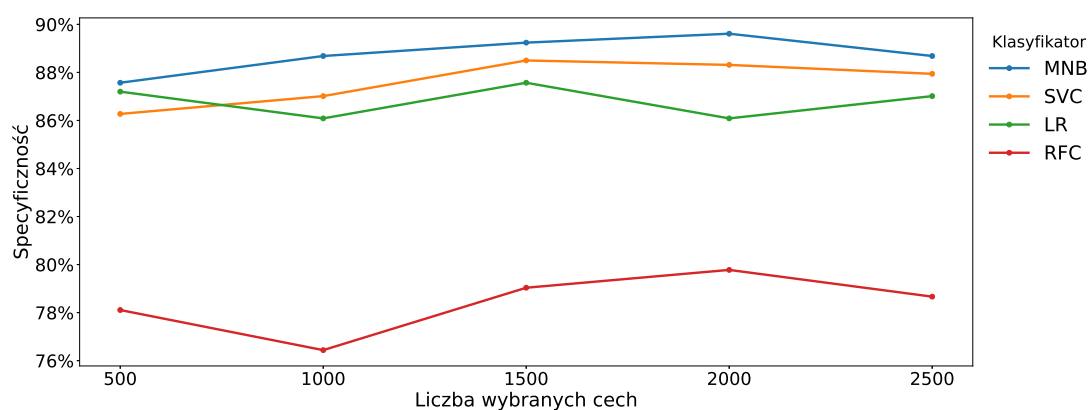


Rys. 6.1. Wykres dokładność modeli w zależności od liczby wybranych cech metodą chi-kwadrat. Rozrzut wartości jest bardzo mały i zawiera się w zakresie 5%. Różnice są więc marginalne, a najlepiej wypada klasyfikator regresji logistycznej z 2500 wybranymi cechami osiągając wynik 86%.

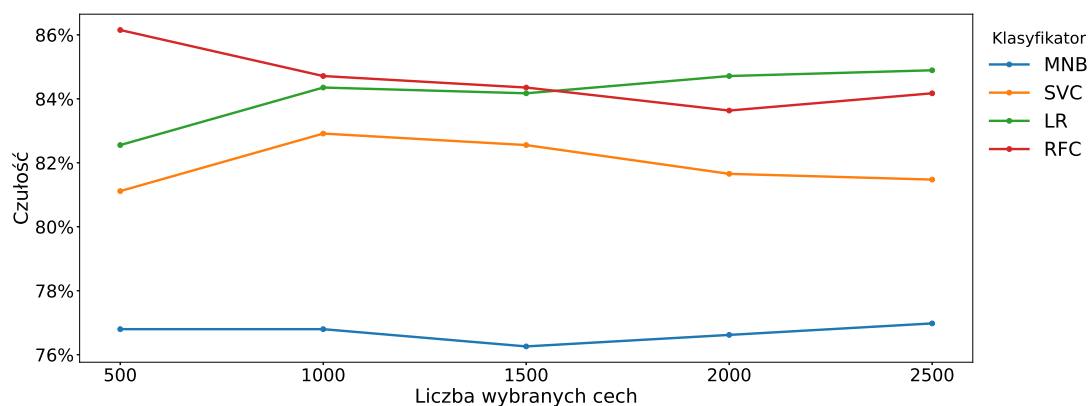
Jak można zauważyć, klasyfikatory LogisticRegression i SVC poradziły sobie lepiej osiągając wyniki z przedziału od 84 do 86%. Natomiast RandomForestClassifier oraz MultinomialNB

poradziły sobie nieco gorzej osiągając wyniki z zakresu od 81 do 83%. Najlepszy wynik dokładności otrzymany został dla 2500 cech przy użyciu klasyfikatora regresji logistycznej i wyniósł on 86%. Jest on minimalnie lepszy od rezultatu uzyskanego podczas braku selekcji cech. Jest to bardzo dobry wynik ponieważ pomimo znacznego zmniejszenia liczby cech, dokładność klasyfikacji nie spadała.

Na rysunkach 6.2 i 6.3 zostały przedstawione kolejno wykresy specyficzności i czułości od liczby wybranych cech dla metody selekcji chi-kwadrat.



Rys. 6.2. Wykres specyficzności modeli w zależności od liczby wybranych cech metodą chi-kwadrat. Rozrzut wartości dla klasyfikatorów MNB, SVC oraz LR jest bardzo mały i zawiera się w zakresie 4%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy negatywnej wypada naiwny klasyfikator Bayesa. Jedynie klasyfikator RFC osiągnął rezultat odbiegający od reszty od około 8%.



Rys. 6.3. Wykres czułości modeli w zależności od liczby wybranych cech metodą chi-kwadrat. Rozrzut wartości dla klasyfikatorów RFC, SVC oraz LR jest bardzo mały i zawiera się w zakresie 5%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy pozytywnej wypada klasyfikator lasu losowego. Jedynie klasyfikator MNB osiągnął rezultat lekko odbiegający od reszty.

W przypadku naiwnego klasyfikatora Bayesa wyniki specyficzności (pomiędzy 0.88 a 0.9) były znacznie lepsze od czułości (pomiędzy 0.76 a 0.77). Oznacza to, iż dużo lepiej radził on sobie w przewidywaniu recenzji negatywnych. W przypadku klasyfikatora wektorów nośnych również lepiej radził on sobie z przewidywaniem klasy negatywnej jednak różnica pomiędzy specyficznością (pomiędzy 0.86 a 0.88) a czułością (pomiędzy 0.81 a 0.83) była już mniejsza. Podobną sytuację można zaobserwować dla regresji logistycznej. W tym przypadku specyficzność waha się od 0.86 do 0.88, a czułość od 0.83 do 0.85. Odwrotna sytuacja jest w przypadku klasyfikatora lasu losowego, który lepiej radzi sobie w przewidywaniu klasy pozytywnej. W jego przypadku specyficzność zawiera się pomiędzy 0.76 a 0.81, natomiast czułość pomiędzy 0.83 a 0.87.

W tabeli 6.4 podane zostały wartości średnie czasu selekcji dla badanej liczby cech w zależności od zastosowanego klasyfikatora. W kolumnie spadek podana została krotność redukcji czasu klasyfikacji wynikająca z zastosowania selekcji cech.

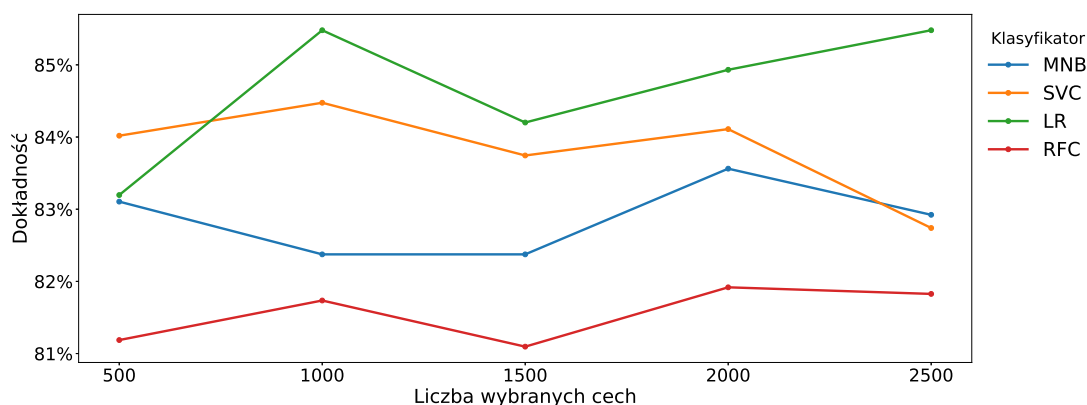
Tabela 6.4. Wyniki czasowe klasyfikacji podczas zastosowania metody selekcji cech chi-kwadrat.

	Średni czas klasyfikacji [s]	Spadek [krotność]
MultinomialNB	0.0041	276
SVC	1.94	284
LogisticRegression	0.0027	374
RandomForestClassifier	0.18	7

Największy zysk czasowy uzyskany został dla klasyfikatora regresji logistycznej. W tym przypadku czas zmalał 374 krotnie, co jest bardzo dobrym rezultatem. Natomiast najmniejszy zysk (7 krotność) uzyskany został dla lasu losowego.

6.1.3. ANOVA

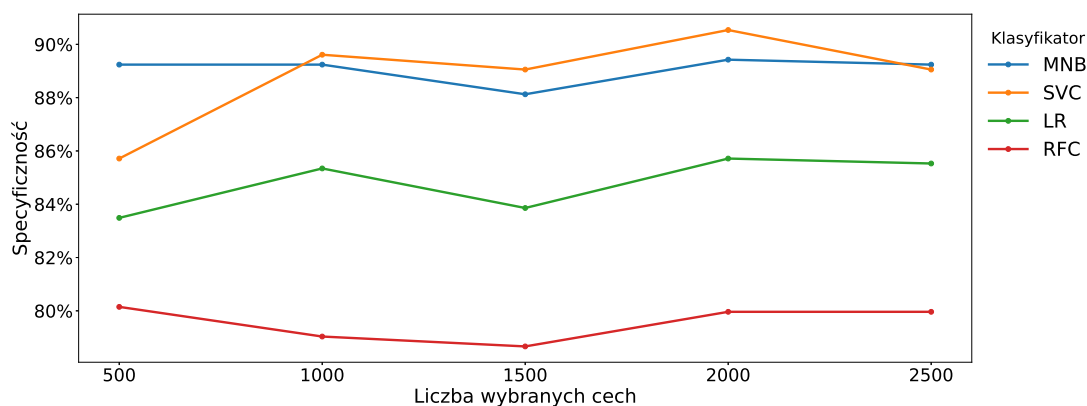
Drugą z metod filtrowych była metoda analizy wariancji. Na rysunku 6.4 znajduje się wykres dokładności zastosowanych modeli od liczby wybranych cech.



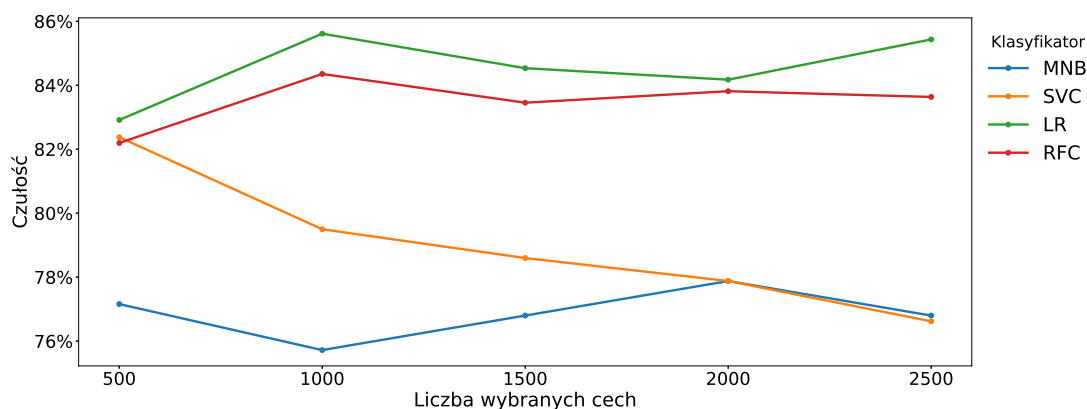
Rys. 6.4. Wykres dokładność modeli w zależności od liczby wybranych cech metodą ANOVA. Rozrzut wartości jest bardzo mały i zawiera się w zakresie 45%. Różnice są więc marginalne, a najlepiej wypada klasyfikator regresji logistycznej z 1000 wybranymi cechami osiągając wynik 85%.

Najlepiej poradził sobie klasyfikator regresji logistycznej osiągając wyniki dokładności od 83 do ponad 85%. Natomiast najmniejszy wynik osiągnął klasyfikator lasu losowego, którego dokładność w zależności od liczby wybranych cech wahała się w okolicach 81-82%. Naiwny klasyfikator Bayesa oraz klasyfikator wektorów nośnych osiągnęły wyniki pomiędzy dwoma wspomnianymi powyżej. Najlepsza dokładność osiągnięta została dla 1000 cech oraz klasyfikatora LogisticRegression i wyniósł on ponad 85%. Jest to wynik lekko lepszy od tego osiągniętego bez selekcji cech. Ponownie pomimo znacznego zmniejszenia liczby cech do aż 1000, dokładność klasyfikacji nie zmniejszyła się.

Na rysunkach 6.5 i 6.6 zostały przedstawione kolejno wykresy specyficzności i czułości od liczby wybranych cech dla metody selekcji cech ANOVA.



Rys. 6.5. Wykres specyficzności modeli w zależności od liczby wybranych cech metodą ANOVA. Rozrzut wartości dla klasyfikatorów MNB, SVC oraz LR jest bardzo mały i zawiera się w zakresie 6%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy negatywnej wypada naiwny klasyfikator Bayesa. Jedynie klasyfikator RFC osiągnął rezultat minimalnie odbiegający od reszty.



Rys. 6.6. Wykres czułości modeli w zależności od liczby wybranych cech metodą ANOVA. Rozrzut wartości dla klasyfikatorów RFC oraz LR jest bardzo mały i zawiera się w zakresie 4%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy pozytywnej wypada klasyfikator lasu losowego. Natomiast klasyfikatory MNB i SVC osiągnęły rezultaty minimalnie odbiegające od reszty.

Ponownie w przypadku klasyfikatora MultinomialNB dużo lepiej radził on sobie z przewidywaniem recenzji negatywnych. W tym przypadku wyniki specyficzności wahały się pomiędzy 0.88 a 0.9, natomiast czułości od 0.76 do 0.78. Klasyfikator SVC także poradził sobie

lepiej z predykcją klasy negatywnej, jednak wyniki w zależności od liczby cech zawierały się w większych przedziałach. Specyficzność wyniosła od 0.86 do 0.9, a czułość od 0.77 do 0.82. W przypadku klasyfikatora LogisticRegression miary specyficzności (od 0.84 do 0.86) i czułości (od 0.83 do 0.86) były bardzo podobne. Ponownie w przypadku klasyfikatora RandomForestClassifier lepiej poradził on sobie w przewidywaniu klasy pozytywnej, jednak różnice były niewielkie. W jego przypadku czułość wynosiła od 0.82 do 0.84, a specyficzność od 0.78 do 82.

W tabeli 6.5 podane zostały wartości średnie czasu selekcji dla badanej liczby cech w zależności od zastosowanego klasyfikatora.

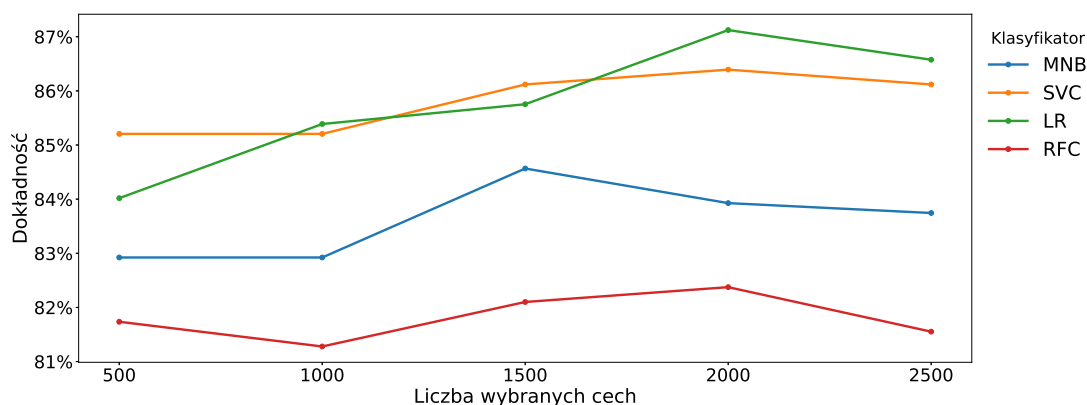
Tabela 6.5. Wyniki czasowe klasyfikacji przy użyciu metody selekcji cech ANOVA.

	Średni czas klasyfikacji [s]	Spadek [krotność]
MultinomialNB	0.0037	305
SVC	1.85	297
LogisticRegression	0.0034	297
RandomForestClassifier	0.11	11

Największy zysk czasowy uzyskany został dla naiwnego klasyfikatora Bayesa. W tym przypadku czas zmalał 305 krotnie, co jest bardzo dobrym rezultatem. Dla klasyfikatorów wektorów nośnych oraz regresji logistycznej zysk był tylko minimalnie mniejszy. Natomiast zdecydowanie najmniejszy zysk (11 krotność) ponownie został uzyskany dla lasu losowego.

6.1.4. RFE

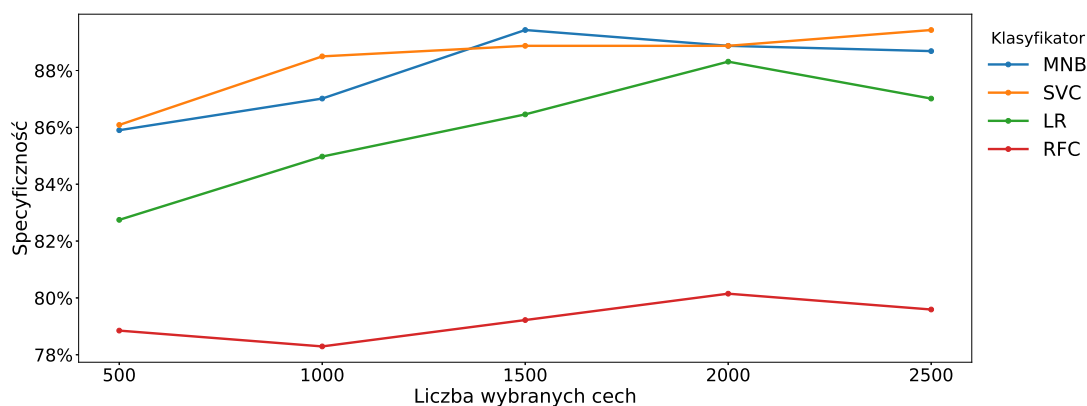
Kolejną metodą była metoda należąca do wrapperów, a konkretnie rekursywna eliminacja cech. Na rysunku 6.7 zobrazony został wykres dokładności modeli względem liczby wybranych cech.



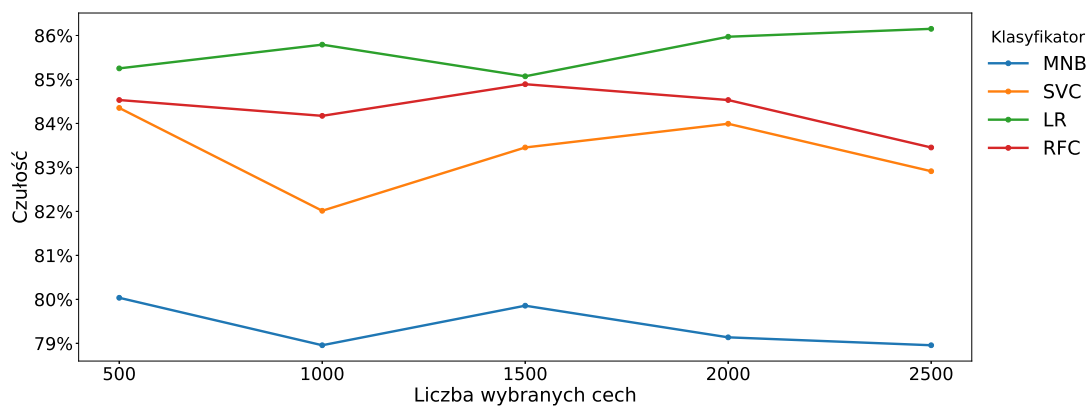
Rys. 6.7. Wykres dokładność modeli w zależności od liczby wybranych cech metodą RFE. Rozrzut wartości jest bardzo mały i zawiera się w zakresie 6%. Różnice są więc marginalne, a najlepiej wypada klasyfikator regresji logistycznej z 2000 wybranymi cechami osiągając wynik 87%.

Ponownie klasyfikatory LogisticRegression oraz SVC osiągnęły najlepsze rezultaty dokładności. W przypadku pierwszego z nich miara ta wyniosła od 84 do 87%, a w przypadku drugiego od 85 do 86%. Najmniejszy wynik osiągnął klasyfikator RandomForestClassifier (od 81 do 82%). MultinomialNB osiągnął dokładność pomiędzy wcześniej wspomnianymi klasyfikatorami. Najlepszy wynik został osiągnięty dla 2000 cech oraz klasyfikatora regresji logistycznej i wyniósł 87%. Jest to wynik minimalnie lepszy od wyniku bez zastosowania selekcji cech.

Na rysunkach 6.8 i 6.9 zostały przedstawione kolejno wykresy specyficzności i czułości od liczby wybranych cech dla metody selekcji cech RFE.



Rys. 6.8. Wykres specyficzności modeli w zależności od liczby wybranych cech metodą RFE. Rozrzut wartości dla klasyfikatorów MNB, SVC oraz LR jest bardzo mały i zawiera się w zakresie 6%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy negatywnej wypadł naiwny klasyfikator Bayesa oraz klasyfikator wektorów nośnych. Jedynie klasyfikator RFC osiągnął rezultat lekko odbiegający od reszty.



Rys. 6.9. Wykres czułości modeli w zależności od liczby wybranych cech metodą RFE. Rozrzut wartości dla klasyfikatorów RFC, SVC oraz LR jest bardzo mały i zawiera się w zakresie 4%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy pozytywnej wypada klasyfikator lasu losowego. Natomiast klasyfikator MNB osiągnął rezultat minimalnie odbiegający od reszty.

Po raz kolejny naiwny klasyfikator Bayesa lepiej poradził sobie z przewidywaniem klasy negatywnej. Specyficzność dla tego modelu wyniosła od 0.86 do 0.89, natomiast czułość od 0.79 do 0.8. Również klasyfikator wektorów nośnych poradził sobie lepiej z wykrywaniem recenzji negatywnych, jednak w jego przypadku różnica była mniejsza (specyficzność od 0.86 do 0.89, a

czułość od 0.82 do 0.84). W przypadku regresji logistycznej wyniki czułości (od 0.85 do 0.86) i specyficzności (od 0.83 do 0.88) było bardzo zbliżone. Klasyfikator lasu losowego ponownie lepiej poradził sobie z przewidywaniem klasy pozytywnej osiągając czułość na poziomie od 0.84 do 0.85, natomiast specyficzność od 0.78 do 0.8.

W tabeli 6.6 podane zostały wartości średnie czasu selekcji dla badanej liczby cech w zależności od zastosowanego klasyfikatora.

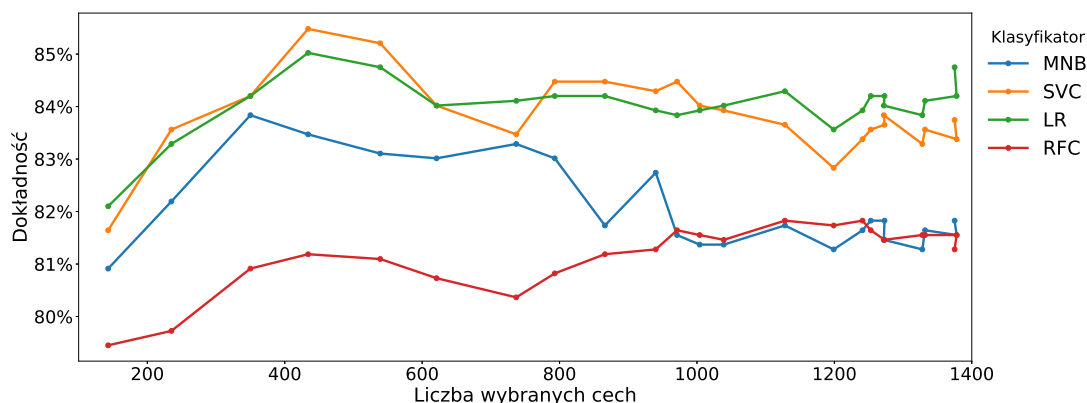
Tabela 6.6. Wyniki czasowe klasyfikacji podczas zastosowania metody selekcji cech RFE.

	Średni czas klasyfikacji [s]	Spadek [krotność]
MultinomialNB	0.0037	305
SVC	2.45	224
LogisticRegression	0.0039	259
RandomForestClassifier	0.16	8

Największy zysk czasowy ponownie został uzyskany dla klasyfikatora regresji logistycznej. W tym przypadku czas zmalał 305 krotnie, co jest bardzo dobrym rezultatem. Natomiast zdecydowanie najmniejszy zysk (8 krotność) uzyskany został ponownie dla lasu losowego.

6.1.5. Regularyzacja L1

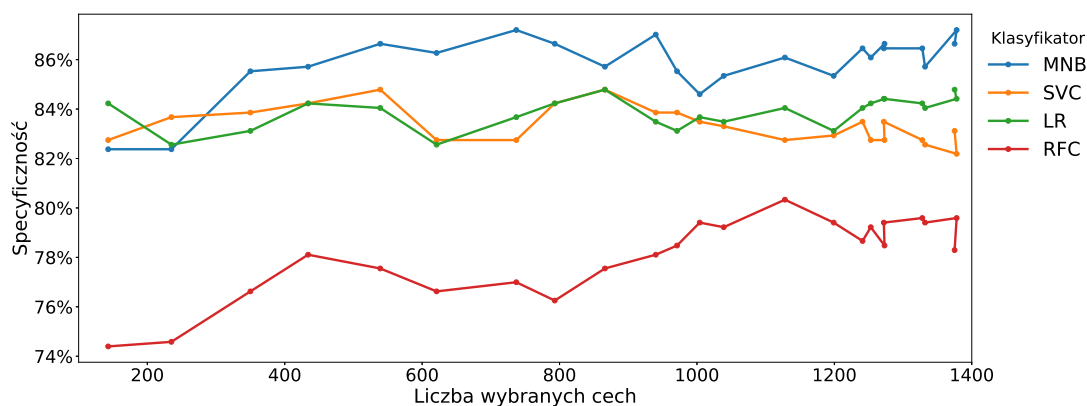
Ostatnią z badanych metod selekcji cech była metoda z użyciem regularyzacji L1. Na rysunku 6.10 zobrazony został wykres dokładności modeli względem liczby wybranych cech.



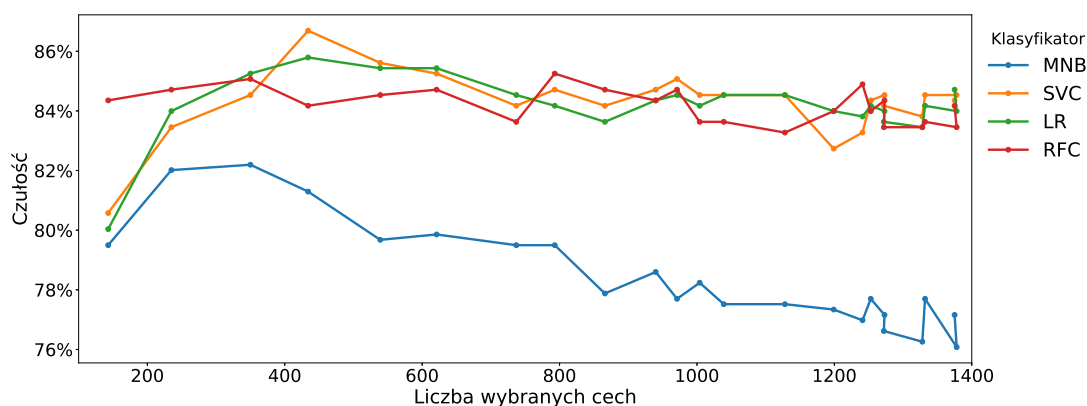
Rys. 6.10. Wykres dokładność modeli w zależności od liczby wybranych cech metodą z użyciem regularyzacji L1. Rozrzut wartości jest bardzo mały i zawiera się w zakresie 6%. Różnice są więc marginalne, a najlepiej wypada klasyfikator SVC z 434 wybranymi cechami osiągając wynik 85%.

Klasyfikatory SVC oraz LogisticRegression osiągnęły najlepszą dokładność zawierającą się w przedziale od 82 do 85%. Minimalnie gorzej poradził sobie klasyfikator MultinomialNB osiągając wyniki z zakresu od 81 do 84%. Ponownie najślabszą dokładność osiągnął RandomForestClassifier z wynikami od 79 do 82%. Najlepszy wynik dokładności uzyskany został dla 434 cech oraz klasyfikatora wektorów nośnych i wyniosła 85%. Po raz kolejny jest to wartość minimalnie lepsza od tej osiągniętej bez zastosowania selekcji cech.

Na rysunkach 6.11 i 6.12 zostały przedstawione kolejno wykresy specyficzności i czułości od liczby wybranych cech dla metody selekcji z wykorzystaniem regularyzacji L1.



Rys. 6.11. Wykres specyficzności modeli w zależności od liczby wybranych cech metodą z użyciem regularyzacji L1. Rozrzut wartości dla klasyfikatorów MNB, SVC oraz LR jest bardzo mały i zawiera się w zakresie 4%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy negatywnej wypadł naiwny klasyfikator Bayesa. Jedynie klasyfikator RFC osiągnął rezultat lekko odbiegający od reszty.



Rys. 6.12. Wykres czułości modeli w zależności od liczby wybranych cech metodą z użyciem regularyzacji L1. Rozrzut wartości dla klasyfikatorów RFC, SVC oraz LR jest bardzo mały i zawiera się w zakresie 4%. Różnice są więc marginalne, a najlepiej w przewidywaniu klasy pozytywnej wypada klasyfikator wektorów nośnych. Natomiast klasyfikator MNB osiągnął rezultat minimalnie odbiegający od reszty.

Ponownie naiwny klasyfikator Bayesa osiągnął lepszy wynik specyficzności (od 0.82 do 0.86) niż czułości (od 0.76 do 0.82), co przekłada się na to, iż lepiej przewidywał on klasę negatywną. Klasyfikatory wektorów nośnych oraz regresji logistycznej osiągnęły bardzo zbliżone

wyniki miar specyficzności i czułości. W znaczniej większości zawierały się one pomiędzy 0.82 a 0.86. Natomiast klasyfikator lasu losowego znowu lepiej poradził sobie z przewidywaniem klasy pozytywnej. W jego przypadku czułość wahała się od 0.83 do 0.86, a specyficzność od 0.74 do 0.81.

W tabeli 6.7 podane zostały wartości średnie czasu selekcji dla badanej liczby cech w zależności od zastosowanego klasyfikatora.

Tabela 6.7. Wyniki czasowe klasyfikacji podczas zastosowania metody selekcji cech z użyciem regularyzacji L1.

	Średni czas klasyfikacji [s]	Spadek [krotność]
MultinomialNB	0.0027	419
SVC	0.86	640
LogisticRegression	0.0022	459
RandomForestClassifier	0.2	6

W tym wypadku najlepszy zysk czasowy uzyskany został dla klasyfikatora SVC. Zmalał on 640 krotnie, co jest bardzo dobrym rezultatem. Natomiast ponownie, zdecydowanie najmniejszy zysk (6 krotność) uzyskany został dla lasu losowego.

Wnioski oraz podsumowanie wyników omówionych w powyższej części pracy zostaną omówione i przedstawione w następnym rozdziale.

7. Podsumowanie i wnioski

W rozdziale tym podsumowane zostaną wszystkie wyniki omówione w rozdziale 5 oraz badane metody selekcji cech.

Ze wszystkich badanych przypadków najlepszy wynik dokładności modelu został osiągnięty dla metody selekcji cech RFE, czyli rekursywnej eliminacji cech oraz klasyfikatora regresji logistycznej. Rezultat ten wyniósł 87%, a liczba wyselekcjonowanych cech 2000. W przypadku tego modelu również wyniki specyficzności i czułości były do siebie bardzo zbliżone i wynosiły one odpowiednio 86 i 88%. Jest to bardzo satysfakcjonujący rezultat, ponieważ obie klasy recenzji (negatywna i pozytywna) miały bardzo podobny procent ich przewidywania. W przypadku na przykład modeli naiwnego klasyfikatora Bayesa wyniki czułości i specyficzności różniły się w sposób znaczący, ze wskazaniem na klasę negatywną. Ten jak i całą resztę najlepszych wyników dokładności modeli dla każdej kombinacji zastosowanych metod selekcji cech oraz klasyfikatorów zostały przedstawione w tabeli 7.1. W nawiasach obok dokładności została podana liczba wybranych cech.

Tabela 7.1. Porównanie wszystkich najlepszych wyników dokładności dla każdego klasyfikatora i metody selekcji cech (w nawiasie obok wyniku znajduje się wybrana liczba cech).

	Brak	Chi-kwadrat	ANOVA	RFE	Reg. L1
MultinomialNB	0.86	0.83 (2000)	0.84 (2000)	0.85 (1500)	0.84 (350)
SVC	0.84	0.85 (1500)	0.84 (1000)	0.86 (2000)	0.85 (434)
LogisticRegression	0.84	0.86 (2500)	0.85 (1000)	0.87 (2000)	0.85 (434)
RandomForestClassifier	0.83	0.82 (500)	0.82 (2000)	0.82 (1500)	0.82 (1128)

Patrząc na wyniki ze strony użytej metody selekcji cech, to są one względnie zbliżone. Jednak mimo wszystko można stwierdzić, że najlepiej poradziła sobie metoda rekursywnej eliminacji cech. Natomiast względem użytych klasyfikatorów wyniki również nie odbiegają od siebie w sposób znaczny. Jednak można zaobserwować, iż najgorzej poradził sobie klasyfikator lasu losowego osiągając wyniki rzędu od 82 do 83%. Najlepsze wyniki osiągnął klasyfikator

regresji logistycznej, którego wyniki wahają się od 84 do 87%. Jednakże klasyfikatory SVC oraz MultinomialNB osiągnęły minimalnie gorsze wyniki.

Porównując wyniki uzyskane podczas zastosowania selekcji cech do rezultatów uzyskanych podczas jej braku można zauważyć, iż w żadnym przypadku selekcja cech nie pogorszyła dokładności w sposób znaczący. W przypadku regresji logistycznej oraz klasyfikatora wektorów nośnych wyniki uległy nawet minimalnemu polepszeniu na skutek wyeliminowania nieznaczących cech.

Patrząc na wyniki czasowe klasyfikacji, to czas ten uległ znacznemu polepszeniu dzięki zmniejszeniu liczby cech do minimum. Największa redukcja czasu klasyfikacji została uzyskana dla metody z użyciem regularyzacji L1 oraz klasyfikatora wektorów nośnych. Czas selekcji w tym przypadku został zmniejszony 640 krotnie. Najmniejsze spadki czasu selekcji cech, bo tylko kilkukrotne zostały uzyskane dla klasyfikatora lasu losowego.

Podsumowując, zastosowanie selekcji cech spełniło swoje zastosowanie znacznie zmniejszając złożoność czasową i obliczeniową modelu nie tracąc przy tym na dokładności predykcji. W niektórych przypadkach selekcja cech nawet zwiększyła dokładność modelu, a najlepiej poradziła sobie metoda RFE z klasyfikatorem regresji logistycznej osiągając dokładność na poziomie 87%.

Bibliografia

- [1] Newzoo. „*Newzoo Global Games Market Report 2021 | Free Version*”. <https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2021-free-version>. 2021.
- [2] Diego Lopez Ys. „*Your Guide to Natural Language Processing (NLP)*”. 2019. URL: <https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1> (term. wiz. 2021).
- [3] Salima Behdenna, Fatiha Barigou i Ghalem Belalem. „*Document Level Sentiment Analysis: A survey*”. W: *EAI Endorsed Transactions on Context-aware Systems and Applications* (2018).
- [4] WGU. „*Machine learning: definition, explanation, and examples*.” URL: <https://www.wgu.edu/blog/machine-learning-definition-explanation-examples2007> (term. wiz. 2021).
- [5] Rohith Gandhi. „*Naive Bayes Classifier*”. 2018. URL: <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c> (term. wiz. 2021).
- [6] Rohith Gandhi. „*Support Vector Machine — Introduction to Machine Learning Algorithms*”. 2018. URL: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47> (term. wiz. 2021).
- [7] ML Glossary documentation. „*Logistic Regression*”. URL: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html (term. wiz. 2021).
- [8] Tony Yiu. „*Understanding Random Forest*”. 2019. URL: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2> (term. wiz. 2021).
- [9] Wojciech Korona i Monika Czwakiel. „*Drzewo decyzyjne*”. URL: https://mfiles.pl/pl/index.php/Drzewo_decyzyjne (term. wiz. 2022).
- [10] Ferat Sahin. „*A survey on feature selection methods*”. W: *Computers and Electrical Engineering* (2014), s. 16–28.

- [11] Lai Hung, Rayner Alfred i Mohd Hijazi. „*A Review on Feature Selection Methods for Sentiment Analysis*”. W: *Advanced Science Letters* (2015), s. 2952–2956.
- [12] Sauravkaushik8 Kaushik. „*Introduction to Feature Selection methods with an example (or how to select the right variables?)*” URL: <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables> (term. wiz. 2021).
- [13] Lai Hung, Rayner Alfred i Mohd Hijazi. „*A Review on Feature Selection Methods for Sentiment Analysis*”. W: *Advanced Science Letters* (2015), s. 2952–2956.
- [14] Shida He i in. „*MRMD2.0: A Python tool for machine learning features ranking and reduction*”. W: *Current Bioinformatics* (2020).
- [15] scikit-learn. „*sklearn.feature_selection.RFE*”. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE (term. wiz. 2021).
- [16] Gianluca Malato. „*Feature selection in machine learning using Lasso regression*”. URL: <https://towardsdatascience.com/feature-selection-in-machine-learning-using-lasso-regression-7809c7c2771a> (term. wiz. 2021).
- [17] Steam. „*Wiedźmin 3: Dziki Gon - recenzje*”. URL: https://store.steampowered.com/app/292030/Wiedmin_3_Dziki_Gon/?l=polish (term. wiz. 2021).
- [18] Witold Kieraś i Marcin Woliński. „*Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego*”. W: *Język Polski XCVII.1* (2017), s. 75–83.