Politechnika Wrocławska

Temat:

Analiza botów z Twittera

Metody AI w badaniu zagrożeń w systemach komputerowych

Dr inż. Tomasz Walkowiak

Wtorek 07:30

Wykonali:
Urszula Warmińska, 249060
Arkadiusz Kotynia, 249038
Katarzyna Kowalczyk, 200926

Spis treści

1. Cel pracy

Celem projektu jest sprawdzenie algorytmów uczenia maszynowego, a następnie zaimplementowanie i w konsekwencji wybranie najlepszego, mającego największą dokładność w weryfikacji czy dany post został napisany przez użytkownika czy przez bota.

2. Dane użyte w projekcie

Dane użyte w projekcie pochodziły z publicznej strony (https://www.kaggle.com/competitions/twitter-spam/data) i są możliwe do pobrania przez każdą zainteresowaną osobę. Strona docelowo została postawiona 4 lata temu w celu wyłonienia zwycięzcy konkursu, który w oparciu o dane dostępne do pobrania obliczyłby najbardziej zbliżony do aktualnych danych wynik. Konkurs wciąż trwa i jest możliwe przesłanie swoich wyników, warto więc zaznaczyć, że wyniki porównywane przy pomocy dzielenia zbioru danych na dane testowe i treningowe nieznacznie różnią się od wyników przesyłanych do autorów konkursu.

Dane testowe należało zmodyfikować i dostosować tak, aby mogły być użyte w uczeniu maszynowym. W tym celu usunięto kilka rekordów, które w niepoprawny sposób były rozkładane na kolumny, tym samym dane, które znajdowały się w kolumnach nie były jednolite i nie pasowały do wartości z innych wierszy. Kolumny w pliku:

a. Tweet - jest to tekst, który został opublikowany na Twitterze.
b. obserwujący - liczba osób, które konto, które tweetowało, obserwuje
c. followers - liczba osób śledzących konto, które tweetowało
d. actions - Łączna liczba polubień, odpowiedzi i retweetów danego tweeta.
e. is_retweet - wartość binarna [0,1]: Jeśli 0 to nie jest to retweet, jeśli 1 to jest to retweet
f. location - samodzielnie napisana lokalizacja podana przez użytkownika w jego profilu, może nie istnieć, być "Unkown", a także nie jest znormalizowana! Mogą być to wszystkie miejsca np. Wola, Ursynów, Kozanów, Wrocław, Polska itp.
g. Typ – nie spam lub spam

```
display(df)
```

	Tweet	following	followers	actions	is_retweet	location	Type	Unnamed: 7
0	Good Morning Love @LeeBrown_V	0.0	0.0	0.0	0.0	Pennsylvania, USA	Quality	NaN
1	'@realDonaldTrump @USNavy RIP TO HEROES'	42096.0	61060.0	5001.0	0.0	South Padre Island, Texas	Spam	NaN
2	Haven't been following the news but I understa...	0.0	0.0	NaN	0.0	Will never be broke ever again	Quality	NaN
3	pic.twitter.com/dy9q4ftLhZ What to do with pap...	0.0	0.0	0.0	0.0	Mundo	Quality	NaN
4	#DidYouKnow ► Mahatma Gandhi made a brief visi...	17800.0	35100.0	NaN	0.0	Nottingham, England	Quality	NaN
...	...	...	...	...	...	...	...	...
14894	#AllWentWrongWhen I told my hair stylist to "g...	695.0	533.0	868.0	1.0	United States	Spam	NaN
14895	They don't have to like you, and you don't hav...	0.0	0.0	0.0	0.0	NaN	Quality	NaN
14896	#Miami Graham Nash Live at Parker Playhouse #...	5647.0	15091.0	5823.0	0.0	United States	Spam	NaN
14897	@bethannhamilton is in the business of one-upp...	0.0	0.0	NaN	0.0	Southgate, MI	Quality	NaN
14898	Chasing Success by Space Cadetz Listen up...	1219.0	957.0	4077.0	1.0	United States	Spam	NaN

14899 rows × 8 columns

Zrzut ekranu 1 Dane udostępnione do uczenia maszynowego przed ich usystematyzowaniem.

3. Użyte narzędzia – ogólny opis

a. Jupyter Notebook

Jupyter Notebook to interaktywne środowisko programistyczne, które umożliwia tworzenie i udostępnianie dokumentów zawierających kod, tekst, obrazy, wizualizacje i wiele innych elementów. Narzędzie jest często używane w dziedzinie analizy danych, uczenia maszynowego, eksploracji danych i innych obszarach związanych z nauką danych.

Składa się z komórek, w których pisany jest kod (np. w języku Python, Scala, R), tekst, komentarze. Każda komórka może być wykonywana oddzielnie, co pozwala na uruchamianie części kodu i sprawdzanie wyniku uruchamianego kodu. Pozwala na wyświetlanie obrazów, wizualizacji danych (np. wykresów) i multimediów (np. wideo, dźwięku) bezpośrednio w dokumencie.

Możliwe jest zapisywanie dokumentu (kodu) w formacie .ipynb i ponowne jego wczytanie, aby kontynuować pracę. Narzędzie umożliwia integracje z bibliotekami używanymi w analizie danych, takimi jak NumPy, Pandas, Matplotlib, TensorFlow, scikit-learn itp., co ułatwia eksplorację danych i implementację metod AI. [7]

b. NumPy (Numerical Python)

To biblioteka do obliczeń numerycznych w języku Python. Zapewnia efektywne operacje na dużych tablicach wielowymiarowych, które są podstawowym typem danych w analizie danych i uczeniu maszynowym. NumPy oferuje szeroki zakres funkcji do manipulacji i przetwarzania danych numerycznych, takich jak operacje matematyczne, transpozycje, indeksowanie,

agregacje, obliczenia statystyczne itp. Jest często używany jako podstawowa biblioteka do manipulacji danymi w innych bibliotekach i narzędziach do analizy danych. [8]

c. Scikit-learn

To biblioteka oferująca wiele algorytmów uczenia maszynowego, takich jak klasyfikacja, regresja, grupowanie, redukcja wymiarowości, selekcja cech, uczenie nienadzorowane i nadzorowane. Scikit-learn zapewnia łatwy w użyciu interfejs API do budowania modeli, oceny wydajności, doboru hiperparametrów i wielu innych zadań związanych z uczeniem maszynowym. Biblioteka ta ma również narzędzia do przetwarzania danych, wizualizacji wyników i ewaluacji modeli. [9]

d. Matplotlib

To biblioteka wizualizacji danych w języku Python. Jest wykorzystywana do tworzenia różnego rodzaju wykresów, diagramów, histogramów, map cieplnych, wykresów punktowych i innych wizualizacji danych. Matplotlib oferuje elastyczną kontrolę nad wyglądem i stylizacją wykresów oraz możliwość dostosowywania ich do indywidualnych potrzeb. Jest często używany w analizie danych, eksploracji danych, raportowaniu wyników i prezentacji wizualnych. [10]

e. Pandas

To biblioteka do manipulacji i analizy danych w języku Python. Oferuje wydajne i elastyczne struktury danych, takie jak DataFrames, które umożliwiają łatwe przechowywanie, przetwarzanie i analizowanie danych tabelarycznych. Pandas zapewnia narzędzia do importu i eksportu danych z różnych źródeł, operacji na danych (filtrowanie, sortowanie, grupowanie), obliczeń statystycznych, manipulacji struktur danych, usuwania duplikatów, obsługi brakujących danych i wiele innych. Pandas jest jednym z najpopularniejszych narzędzi w analizie danych i stanowi podstawę dla wielu innych bibliotek i narzędzi w ekosystemie Pythona. [11]

f. Colab

Google Colab (skrót od Collaboratory) to bezpłatne środowisko do pracy z językiem Python dostępne w chmurze. Colab pozwala na tworzenie i uruchamianie notebooków Jupyter bez konieczności instalowania i konfigurowania lokalnego środowiska programistycznego. Możliwe funkcjonalności:

Interaktywne środowisko: umożliwia pisanie, uruchamianie i eksperymentowanie z kodem Python w interaktywnych komórkach. Można dodawać komórki z kodem, tekstem i wynikami, a następnie je uruchamiać i obserwować wyniki bezpośrednio w notebooku.

Bezproblemowa konfiguracja: jest dostępny online i nie wymaga konfiguracji lokalnej.

Obliczenia na GPU i TPU: umożliwia wykorzystanie mocy obliczeniowej GPU (jednostki przetwarzania grafiki) i TPU (jednostki przetwarzania tensorów) w celu przyspieszenia obliczeń, zwłaszcza w dziedzinie uczenia maszynowego. Można łatwo skonfigurować środowisko, aby korzystać z tych zasobów obliczeniowych.

Dostęp do zewnętrznych danych: umożliwia łatwe importowanie danych z różnych źródeł, takich jak dysk Google, GitHub czy pliki lokalne.

Współpraca i udostępnianie: umożliwia współpracę z innymi osobami w czasie rzeczywistym. Można udostępniać notebooki, zapraszać innych do edycji i komentowania. Istnieje również możliwość udostępnienia notebooków jako statycznych dokumentów HTML lub PDF.

Integracja z Google Drive: jest ściśle zintegrowany z Google Drive. Można przechowywać notebooki w folderach, synchronizować je między urządzeniami i udostępniać zespołowi lub publicznie.

4. Użyte algorytmy

Na danych użyto następujących działań: zostały zliczone poszczególne wartości w kolumnie „Type” klasyfikujące danego tweeta do jednej z dwóch kategorii „Quality” lub „Spam”.

```
In [10]: df.value_counts("Type")

Out[10]: Type
         Quality         7454
         Spam            7443
         South Dakota       2
         dtype: int64
```

Rysunek 1 Zliczenie poszczególnych wartości

Zliczono również wartości w kolumnie „location”:

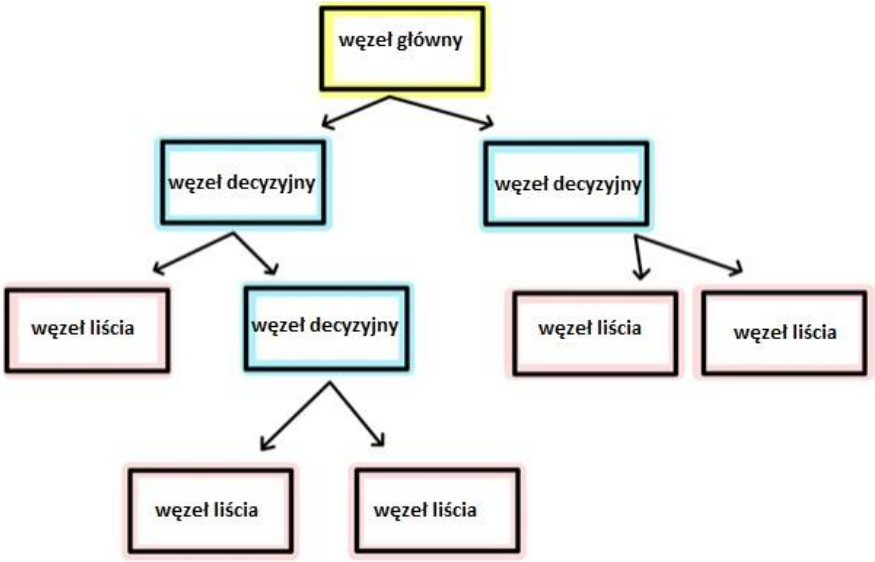
```
In [5]: df.value_counts("location")

Out[5]: location
        United States          4166
        United Kingdom           78
        Washington, DC           49
        Los Angeles, CA          49
        New York, NY             48
                                ...
        Germany, Potsdam          1
        The Castle                1
        Gilberdyke, England       1
        The Bay Area              1
        Upper East Side           1
        Length: 3437, dtype: int64
```

Rysunek 2 Zliczenie wartości w kolumnie "location"

Następnie podzielono dane na dane testowe i treningowe(X_tr, X_te) i użyto poszczególnych klasyfikatorów stosując oznaczenia cl0, cl1, cl2, … , cl7. Gdzie:

cl0[1] - klasyfikator Decission Tree, czyli drzewo decyzyjne to nieparametryczny algorytm uczenia nadzorowanego. Ma hierarchiczną, drzewiastą strukturę, w skład której wchodzi węzeł główny, gałęzie, węzły wewnętrzne i węzły liści. Parametry możliwe do zdefiniowania przez użytkownika to: **criterion** (możliwe wybory to „gini", „entropy", „log_loss"; domyślne ustawienie to „gini". Umożliwia zmierzenie jakości podziału danych); **splitter** (możliwe wybory to „best" i „random". Domyślna wartość to „best".); **max_depth** (wartość integer, domyślna wartość to none. Oznacza maksymalną głębokość drzewa. Jeżeli przyjmuje wartość none, wtedy wyliczany jest, aż wszystkie liście są czyste lub wszystkie liście zawierają wartość mniejszą niż min_samples_split.); **min_samples_split** (przyjmuje wartość integer lub float; domyślnie wartość 2. Oznacza minimalną liczbę próbek potrzebną do podziału wewnętrznego węzła); **min_samples_leaf** (domyślnie 1, przyjmuje wartość integer lub float); min_weight_fraction_leaf(domyślnie 0.0**);** **max_features** (domyślnie none, inne możliwe wartości to wartość integer, float lub „auto", „sqrt", „log2"); **random_state** (domyślnie none, kontroluje randomowość estymatora); **max_leaf_nodes** (przyjmuje wartość integer, domyślnie none); **min_impurity_decrease** (wartość float; domysłna wartość 0.0); **class_weight** (domyślna wartość none); **ccp_alpha** (domyślna wartość 0.0, inne wartości niezerowa zmienna float). [1]

[1] plik "Kod do projektu.ipynb" komórka 6/28



Rysunek 3 DecissionTree Classifier podział

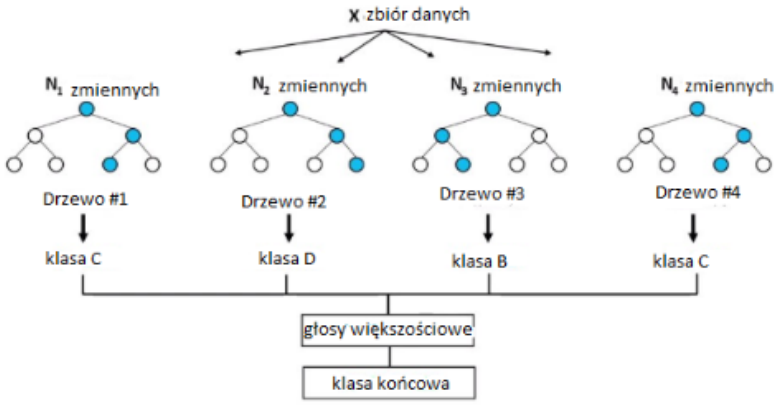
cl1[2] - klasyfikator KNeighborsClassifier, bazuje na k najbliższych sąsiadach danej próbki, która ma zostać sklasyfikowana. Numer „k" jest wprowadzany jako wartość liczbowa integer i może być modyfikowana przez użytkownika. To jeden z najczęściej używanych klasyfikatorów z biblioteki sklearn. Klasy użyte w bibliotece sklearn.neighbors mogą używać zarówno danych wprowadzanych jako pola z bliblioteki Numpy jak również z scipy.sparse. W naszym projekcie wprowadzane dane są sklasyfikowane bliblioteką Numpy. Parametry możliwe do wprowadzenia to: **x**(domyślnie none, inne wartości array-like, sparse matrix, **shape**(n_quaries, n_feauteres) lub (n_querries, n_indexed); **n_neighbors** (wartość integer, domyślnie none); **return_distance** (wartość bool, domyślnie true). [2]

cl2[3] - klasyfikator MLP to wielowartstwowy klasyfikator perceptron. Operuje na sieciach neuronowych. W odróżnieniu od innych klasyfikatorów operujących na sieciach neuronowych jak np. Naive Bayes, MLP opiera się na podstawowej sieci neuronowej w celu wykonania zadania klasyfikacji.

cl3[4] – klasyfikator SVM, Support Vector Machine to estymator nadzorowany uczenia w oparciu o procesy klasyfikacji i analizy regresji. Wykorzystuje algorytm optymalizacji w oparciu o maksymalizację marginesu hiperplanu. SVM dzięki funkcji SVC może mieć charakter nieliniowy.

[2] plik "Kod do projektu.ipynb" komórka 6/28
[3] plik "Kod do projektu.ipynb" komórka 6/28
[4] plik "Kod do projektu.ipynb" komórka 6/28

cl4[5] – klasyfikator Random Forest, czyli losowy las jest to algorytm nadzorowany uczenia maszynowego. Stosowany zarówno do problemów klasyfikacji jak i regresji. Buduje drzewa decyzyjne na różnych próbach i bierze ich większość głosów do klasyfikacji. W przypadku regresji używa średniej głosów. Algorytm najpierw wybiera losowe próbki z całości zadanych danych. Następnie algorytm tworzy drzewo decyzyjne dla każdej wybranej próbki. Głosowanie jest przeprowadzane dla każdej przewidywanej próbki.

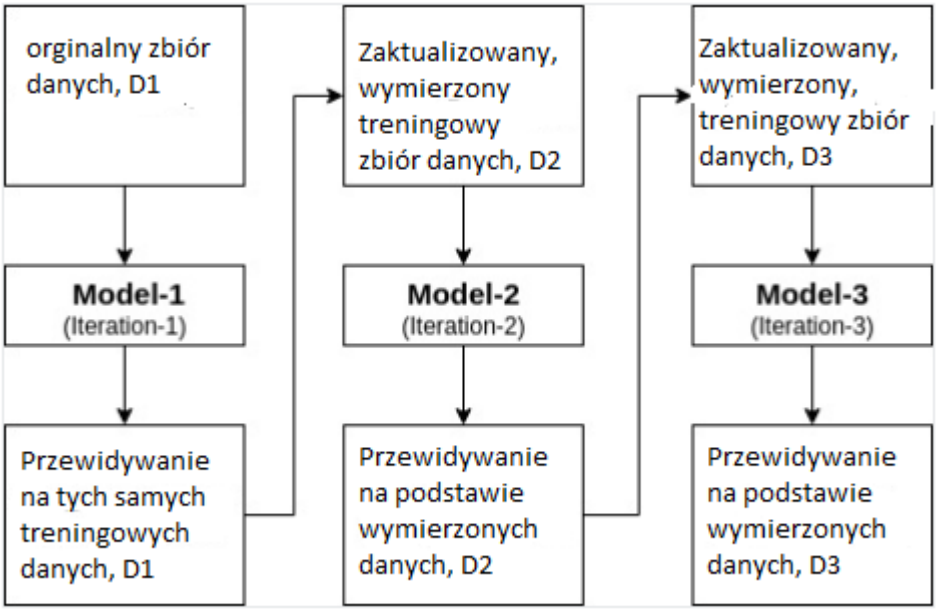


Rysunek 4 zasada działania klasyfikatora lasu przypadkowego

cl5[6] - klasyfikator AdaBoost, algorytm dzięki któremu z dużej liczby słabych klasyfikatorów można otrzymać jeden lepszy. AdaBoost w kolejnych iteracjach trenuje, a następnie mierzy błąd wszystkich wprowadzonych słabszych klasyfikatorów. Ważność źle zakwalifikowanych podziałów zbioru jest zwiększana, tak aby klasyfikatory zwracały na te dane większą uwagę. Klasyfikator najpierw dobiera odpowiedni klasyfikator do oryginalnego zbioru danych, a następnie dopasowuje dodatkowe kopie klasyfikatora zwiększając wagę źle dopasowanych danych, tak aby klasyfikatory skupiały się na trudniejszych przypadkach. Parametry możliwe do sprecyzowania przez użytkownika to: **estimator** (jeżeli ustawimy ten parametr jako none, zostanie użyty klasyfikator drzewa decyzyjnego z parametrem max_depth=1); **n_estimator** (wartość liczbowa wprowadzana jako integer, domyślnie 50**); learning_rate** (wprowadzane jako zmienna float, domyślnie 1.0); **algorithm** (możliwe SAMME lub SAMME.R. Zazwyczaj algorytm SAMME.R jest szybszym i jest domyślnie ustawionym algorytmem); **random_state** (wprowadzany jako zmienna integer, wartość domyślna none, odpowiada za wartość losową dodaną w każdej rosnącej iteracji); **base_estimator** (domyślna wartość to none; w przypadku

[5] plik "Kod do projektu.ipynb" komórka 6/28
[6] plik "Kod do projektu.ipynb" komórka 6/28

ustawienia wartości na none algorytm używa klasyfikatora drzewa losowego z parametrem max_depth=1). [3]



Rysunek 5 Zasada działania klasyfikatora AdaBoost

cl6[7] - klasyfikator GaussianNB. Możliwe parametry wprowadzone przez użytkownika to: **priors** (domyślna wartość None), **var_smoothing** (wartość float, domyślnie 1e-9 parametr do obliczenia stabiności). [4]

cl7[8] - klasyfikator kwadratowej analizy dyskryminacyjnej, bazuje na algorytmie kwadratowej analizy dyskryminacyjnej, jak wskazuje sama jego nazwa. Podobny klasyfikator to liniowa analiza dyskryminacyjna. QDA zakłada, że każda klasa ma rozkład Gaussa. Priorytet specyficzny dla klasy to proporcja punktów danych należących do klasy. Wektor średni to jest średnią ze zmiennych wejściowych należących do klasy. QDA służy do znajdowania nie-liniowej granicy między klasyfikatorami, w odróżnieniu od LDA. QDA pozwala na lepsze dopasowanie danych niż LDA, ale też potrzeba wprowadzić większą liczbę parametrów w przypadku tego klasyfikatora. [5]

5. Część praktyczna

Do wykonania algorytmów użyto narzędzia Jupyter Notebook i odpowiednich bibliotek. W celu wczytania danych zastosowano biblioteki takie jak: pandas, sklearn, numpy, matplotlib, os, graphviz, umap.

[7] plik "Kod do projektu.ipynb" komórka 6/28
[8] plik "Kod do projektu.ipynb" komórka 6/28

Poniższy kod został napisany w celu wyznaczenia dokładności uczenia każdego algorytmu. Zawarta jest w nim wektoryzacja danych. W tej części zastosowano wektoryzację worek słów – zlicza liczbę wystąpień każdego wyrazu w tekście i przedstawia za pomocą wektora składającego się z liczb naturalnych. W kodzie dokonany jest w nim podział na dane testowe i treningowe, a także zaimplementowane są algorytmy, których zadaniem jest uczenie maszynowe. Cały proces odbywa się w pętli, aby zautomatyzować wpisywanie stosunku liczby danych testowych. Wyświetlana jest dokładność każdego algorytmu w zależności od tego stosunku, a następnie zapisywana jest ona do zmiennej, a w rezultacie do tabeli.

```
In [37]: from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(max_features=50)

X_train_bow = vectorizer.fit_transform(X_train).toarray()
X_test_bow = vectorizer.transform(X_test).toarray()

pd.DataFrame(X_train_bow, columns=vectorizer.get_feature_names_out())
```

Out[37]:

	about	all	and	are	as	at	be	but	by	can	...	trump	twitter	up	was	we	what	will	with	you	your
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5953	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
5954	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
5955	0	0	0	0	0	0	1	0	0	1	...	0	0	0	0	0	0	0	0	0	0
5956	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5957	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5958 rows x 50 columns

Zrzut ekranu 2 Wektoryzacja worek słów – implementacja oraz dane zwektoryzowane

```
pd.DataFrame(X_train_tfidf, columns=vectorizer.get_feature_names_out())

X_tr = X_train_tfidf
X_te = X_test_tfidf

cl0 = tree.DecisionTreeClassifier().fit(X_tr, y_train)
cl1 = KNeighborsClassifier().fit(X_tr, y_train)
cl2 = MLPClassifier(max_iter=500).fit(X_tr, y_train)
cl3 = SVC().fit(X_tr, y_train)

cl4 = RandomForestClassifier().fit(X_tr, y_train)
cl5 = AdaBoostClassifier().fit(X_tr, y_train)
cl6 = GaussianNB().fit(X_tr, y_train)
cl7 = QuadraticDiscriminantAnalysis().fit(X_tr, y_train)

y_cl0_pred = cl0.predict(X_te)
y_cl1_pred = cl1.predict(X_te)
y_cl2_pred = cl2.predict(X_te)
y_cl3_pred = cl3.predict(X_te)
y_cl5_pred = cl5.predict(X_te)
y_cl4_pred = cl4.predict(X_te)
y_cl6_pred = cl6.predict(X_te)
y_cl7_pred = cl7.predict(X_te)

print("Accuracy (cl0 DecisionTreeClassifier):",metrics.accuracy_score(y_test, y_cl0_pred))
print("Accuracy (cl1 KNeighborsClassifier):",metrics.accuracy_score(y_test, y_cl1_pred))
print("Accuracy (cl2 MLPClassifier):",metrics.accuracy_score(y_test, y_cl2_pred))
print("Accuracy (cl3 SVC):",metrics.accuracy_score(y_test, y_cl3_pred))
print("Accuracy (cl4 RandomForestClassifier):",metrics.accuracy_score(y_test, y_cl4_pred))
print("Accuracy (Cl5 AdaBoostClasifier):",metrics.accuracy_score(y_test, y_cl5_pred))
print("Accuracy (Cl6 Naive_BayesClaasifier):",metrics.accuracy_score(y_test, y_cl6_pred))
print("Accuracy (Cl7 QuadraticDiscriminantAnalysis):",metrics.accuracy_score(y_test, y_cl7_pred))

cl0_DecisionTreeClassifier=metrics.accuracy_score(y_test, y_cl0_pred)
cl1_KNeighborsClassifier=metrics.accuracy_score(y_test, y_cl1_pred)
cl2_MLPClassifier=metrics.accuracy_score(y_test, y_cl2_pred)
cl3_SVC=metrics.accuracy_score(y_test, y_cl3_pred)
cl4_RandomForestClassifier=metrics.accuracy_score(y_test, y_cl4_pred)
Cl5_AdaBoostClasifier=metrics.accuracy_score(y_test, y_cl5_pred)
Cl6_Naive_BayesClaasifier=metrics.accuracy_score(y_test, y_cl6_pred)
Cl7_QuadraticDiscriminantAnalysis=metrics.accuracy_score(y_test, y_cl7_pred)

np_array = np.array([cl0_DecisionTreeClassifier,cl1_KNeighborsClassifier,cl2_MLPClassifier,cl3_SVC,cl4_RandomForestClassi
result_table[str(str(j+1) + " " + str(i))] = pd.DataFrame(np_array)
print(j)
```

Zrzut ekranu 3 Użyty kod w celu wyliczenia dokładności (1)

```
print("Accuracy (cl0 DecisionTreeClassifier):",metrics.accuracy_score(y_test, y_cl0_pred))
print("Accuracy (cl1 KNeighborsClassifier):",metrics.accuracy_score(y_test, y_cl1_pred))
print("Accuracy (cl2 MLPClassifier):",metrics.accuracy_score(y_test, y_cl2_pred))
print("Accuracy (cl3 SVC):",metrics.accuracy_score(y_test, y_cl3_pred))
print("Accuracy (cl4 RandomForestClassifier):",metrics.accuracy_score(y_test, y_cl4_pred))
print("Accuracy (Cl5 AdaBoostClasifier):",metrics.accuracy_score(y_test, y_cl5_pred))
print("Accuracy (Cl6 Naive_BayesClaasifier):",metrics.accuracy_score(y_test, y_cl6_pred))
print("Accuracy (Cl7 QuadraticDiscriminantAnalysis):",metrics.accuracy_score(y_test, y_cl7_pred))

cl0_DecisionTreeClassifier=metrics.accuracy_score(y_test, y_cl0_pred)
cl1_KNeighborsClassifier=metrics.accuracy_score(y_test, y_cl1_pred)
cl2_MLPClassifier=metrics.accuracy_score(y_test, y_cl2_pred)
cl3_SVC=metrics.accuracy_score(y_test, y_cl3_pred)
cl4_RandomForestClassifier=metrics.accuracy_score(y_test, y_cl4_pred)
Cl5_AdaBoostClasifier=metrics.accuracy_score(y_test, y_cl5_pred)
Cl6_Naive_BayesClaasifier=metrics.accuracy_score(y_test, y_cl6_pred)
Cl7_QuadraticDiscriminantAnalysis=metrics.accuracy_score(y_test, y_cl7_pred)

np_array = np.array([cl0_DecisionTreeClassifier,cl1_KNeighborsClassifier,cl2_MLPClassifier,cl3_SVC,cl4_RandomForestClassi
result_table[str(str(j+1) + " " + str(i))] = pd.DataFrame(np_array)
print(j)

display(result_table)
print(" ")
print(" ")

result_table.to_excel('results1.xlsx')
```

Zrzut ekranu 4 Użyty kod w celu wyliczenia dokładności (2)

	Klasyfikator drzewa decyzyjnego	**K-najbliższych sąsiadów**	**Klasyfikator MLP**	**SVC**	**Klasyfikator losowego lasu**	**Klasyfikator AdaBoost**	**Klasyfikator Naive Bayes**	**Kwadratowa analiza dyskryminacyjna**
Stosunek danych testowych	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]	Średnia wyniku dokładności [%]
0.1	82,38 ± 0,18	80,27 ± 0	83,78 ± 0,45	84,63 ± 0	83,77 ± 0,19	84,36 ± 0	72,62 ± 0	76,24 ± 0
0.2	82,72 ± 0,18	82,62 ± 0	83,62 ± 0,42	85,3 ± 0	83,87 ± 0,19	85,07 ± 0	73,46 ± 0	76,85 ± 0
0.3	83,28 ± 0,17	82,64 ± 0	83,79 ± 0,22	85,06 ± 0	84,38 ± 0,1	84,21 ± 0	75,39 ± 0	78,03 ± 0
0.4	83,38 ± 0,07	80,85 ± 0	83,78 ± 0,23	84,76 ± 0	84,81 ± 0,11	84,68 ± 0	75,85 ± 0	78,57 ± 0
0.5	83,6 ± 0,12	83,06 ± 0	84,37 ± 0,26	85,25 ± 0	85,15 ± 0,11	84,62 ± 0	76,25 ± 0	79,1 ± 0
0.6	83,53 ± 0,17	82,69 ± 0	83,96 ± 0,19	85,07 ± 0	85,21 ± 0,07	84,25 ± 0	73,55 ± 0	63,49 ± 0
0.7	82,9 ± 0,04	82,34 ± 0	83,89 ± 0,14	84,75 ± 0	84,92 ± 0,15	84,13 ± 0	72,73 ± 0	76,34 ± 0
0.8	82,98 ± 0,04	80,35 ± 0	83,22 ± 0,16	84,23 ± 0	84,47 ± 0,13	83,37 ± 0	72,32 ± 0	75,72 ± 0
0.9	83,37 ± 0,14	80,79 ± 0	83,54 ± 0,11	83,75 ± 0	84,28 ± 0,25	83,73 ± 0	71,61 ± 0	76,24 ± 0

Tabela 1 Przedstawia wyniki dokładności przy wektoryzacji – worek słów

Tabela przedstawia wynik działania zastosowanego kodu. Wartość po znaku ± to odchylenie standardowe. Została stworzona do zobrazowania dokładności uczenia maszynowego. Najlepsze algorytmy w tym procesie to SVC, klasyfikator losowego lasu, klasyfikator AdaBoost. W wielu algorytmach wartość odchylenia standardowego jest równa 0, co oznacza że wyniki są stabilne i za każdym razem uzyskiwany jest ten sam wynik. Najwyższa wartość dokładności to 85,% - algorytm SVC.

Kolejnym krokiem była zmiana metody wektoryzacji. Zastosowano wektoryzację tf-idf (ważenie częstością termów – odwrotna częstość w dokumentach) – metoda liczenia wagi terminów w oparciu o ich częstość w dokumencie oraz ich rozkład w całym korpusie. Wagi tf-idf faworyzują słowa występujące w niewielu dokumentach, ponieważ mają większą siłę dyskryminacyjną. *Czym rzadsze wystąpienie terminu w dokumentach, tym wartość idf będzie większa.* W poniższych zrzutach ekranu możliwe jest sprawdzenie użytego w tym celu kodu oraz wyniku działania.

```
vectorizer = TfidfVectorizer(max_features=50)

X_train_tfidf = vectorizer.fit_transform(X_train).toarray()
X_test_tfidf = vectorizer.transform(X_test).toarray()

pd.DataFrame(X_train_tfidf, columns=vectorizer.get_feature_names_out())
```

Zrzut ekranu 5 Wektoryzacja tf-idf - implementacja

	about	all	and	are	as	at	be	but	by	can	...	trump	twitter	up	was	we	what	will	with	you	your
0	0.0	0.0	0.000000	0.0	0.0	0.392986	0.00000	0.0	0.0	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
1	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
2	0.0	0.0	0.361991	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
3	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.307101	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
4	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.460322	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5953	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.614542	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
5954	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.265373	0.0
5955	0.0	0.0	0.000000	0.0	0.0	0.000000	0.37948	0.0	0.0	0.432253	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
5956	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0
5957	0.0	0.0	0.000000	0.0	0.0	0.000000	0.00000	0.0	0.0	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0

5958 rows × 50 columns

Zrzut ekranu 6 Zwektoryzowane dane algorytmem tf-idf

	Klasyfikator drzewa decyzyjnego	**K-najbliższych sąsiadów**	**Klasyfikator MLP**	**SVC**	**Klasyfikator losowego lasu**	**Klasyfikator AdaBoost**	**Klasyfikator Naive Bayes**	**Kwadratowa analiza dyskryminacyjna**
STOSUNEK DANYCH TESTOWYCH	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]	ŚREDNIA WYNIKU DOKŁADNOŚCI [%]
0.1	82,62 ± 0,23	78,93 ± 0	83,95 ± 0,39	84,5 ± 0	83,29 ± 0,27	83,83 ± 0	76,31 ± 0	79,13 ± 0
0.2	83,4 ± 0,16	80,94 ± 0	83,87 ± 0,2	84,77 ± 0	84,19 ± 0,23	84,56 ± 0	77,18 ± 0	79,73 ± 0
0.3	83,49 ± 0,12	79,11 ± 0	83,81 ± 0,25	84,68 ± 0	84,85 ± 0,16	84,56 ± 0	78,93 ± 0	80,36 ± 0
0.4	83,13 ± 0,08	79,24 ± 0	84,09 ± 0,24	85,13 ± 0	85,01 ± 0,18	84,81 ± 0	78,99 ± 0	80,57 ± 0
0.5	83,55 ± 0,18	79,78 ± 0	84,38 ± 0,19	86,01 ± 0	85,46 ± 0,13	85,37 ± 0	79,7 ± 0	81,66 ± 0
0.6	83,67 ± 0,17	80,48 ± 0	84,17 ± 0,2	85,8 ± 0	85,35 ± 0,12	84,74 ± 0	76,05 ± 0	75,86 ± 0
0.7	83,18 ± 0,08	80,46 ± 0	84,11 ± 0,14	85,69 ± 0	85,29 ± 0,13	84,52 ± 0	74,95 ± 0	73 ± 0
0.8	82,75 ± 0,11	79,68 ± 0	83,27 ± 0,18	84,74 ± 0	84,39 ± 0,12	83,89 ± 0	74,97 ± 0	75,36 ± 0
0.9	82,56 ± 0,1	77,66 ± 0	83,1 ± 0,26	84,42 ± 0	83,92 ± 0,21	83,11 ± 0	72,68 ± 0	71,78 ± 0

Tabela 2 Przedstawia wyniki dokładności przy wektoryzacji – tf-idf

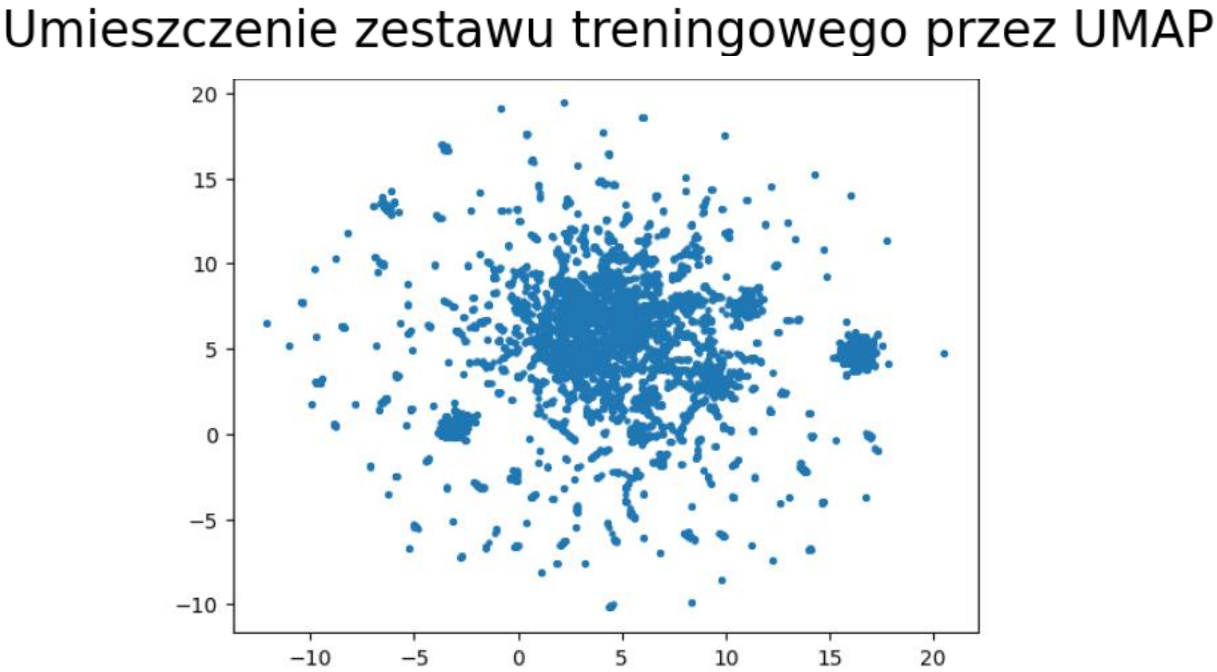
Powyższa tabela przedstawia wyniki dokładności przy zastosowaniu wektoryzacji tf-idf. Wyniki okazały się wyższe niż w poprzedniej próbie. Najwyższy uzyskany wynik to 86.01% (algorytm SVC). Podobnie jak w przypadku poprzednio zastosowanej wektoryzacji (worek słów) w kilku algorytmach pojawiły się odchylenia standardowe wynoszące 0%, co oznacza stabilność algorytmu.

W trakcie realizacji projektu wykonano również uczenie maszynowe za pomocą algorytmu Berta. Za jego pomocą uzyskano najlepszy wynik dokładności – aż 93,79%. Należy jednak wspomnieć, że wykonywanie się tego algorytmu było procesem czasochłonnym.

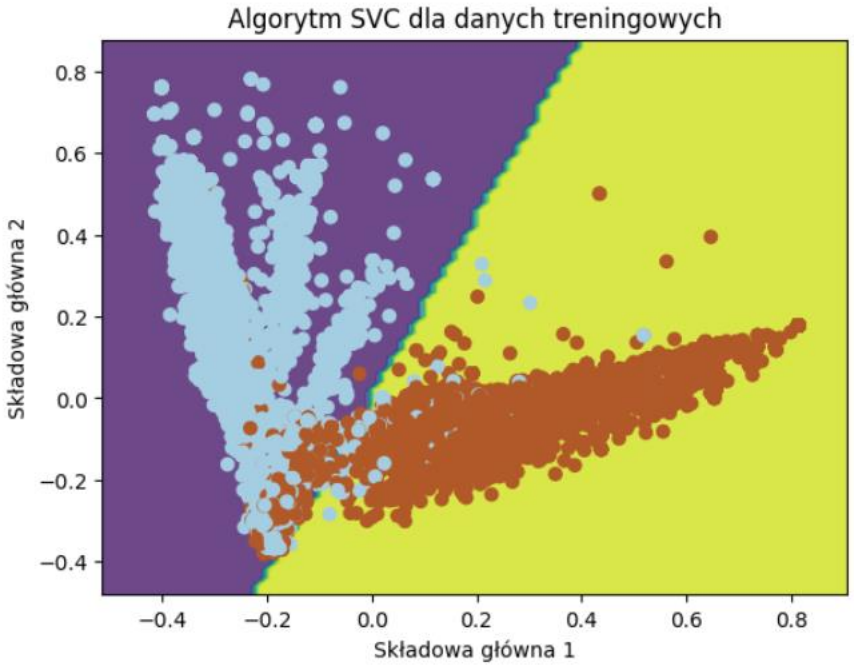
```
Epoch 1/5 - Average Loss: 0.40791468824297084
Epoch 2/5 - Average Loss: 0.33606365276907807
Epoch 3/5 - Average Loss: 0.33938601710802047
Epoch 4/5 - Average Loss: 0.3869346690222772
Epoch 5/5 - Average Loss: 0.2813947348177012
Accuracy: 0.9379194630872483
```

Zrzut ekranu 7 Wynik dokładności uzyskany za pomocą zastosowaniu algorytmu Berta.

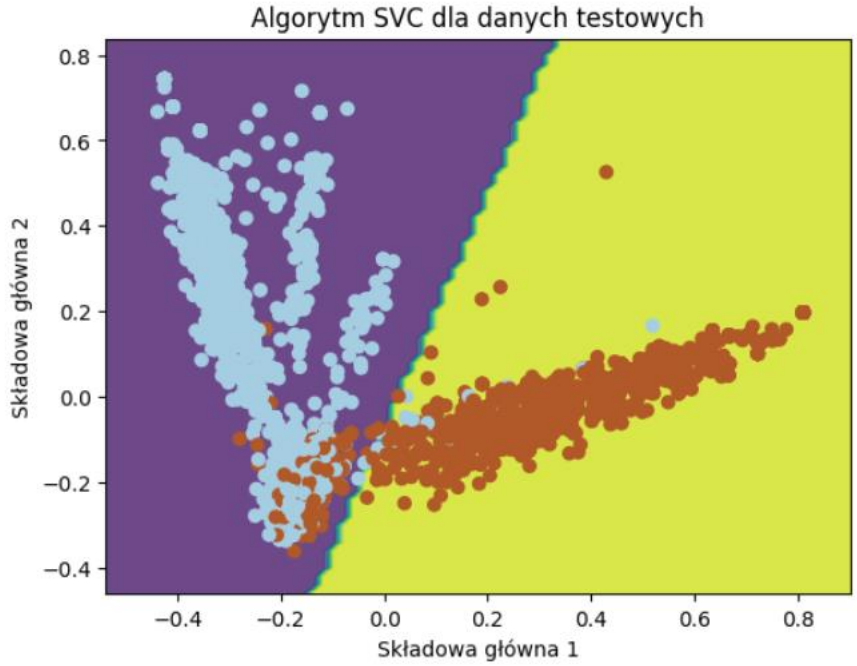
6. Wykresy



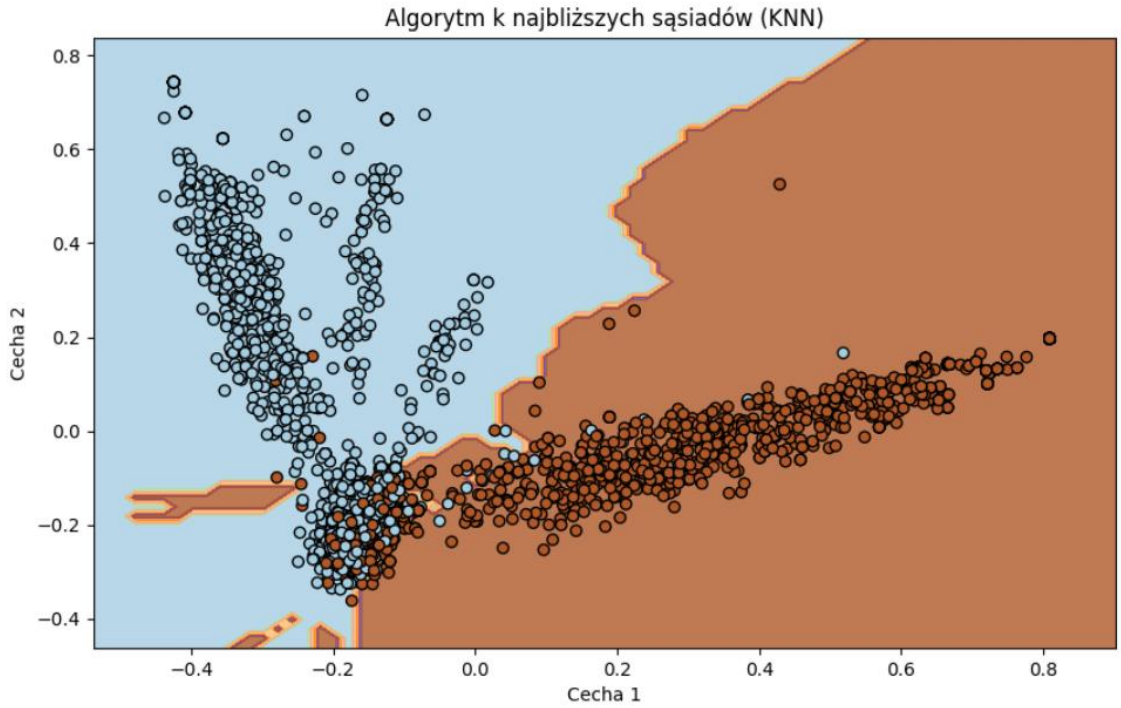
Na wykresie UMAP, W tym wykresie, każdy punkt odpowiada jednemu punktowi danych. Odległość między punktami na wykresie pokazuje, jak bardzo te punkty są podobne do siebie w oryginalnych danych. Pozwala nam zobaczyć, jak dane są zorganizowane i czy istnieją grupy lub klastry punktów, które są podobne do siebie. Na podstawie tego wykresu możemy dowiedzieć się, jak różne punkty danych są podobne lub różnią się od siebie. Na powyższym wykresie można zidentyfikować kilka skupisk, przedstawiających kilka grup.



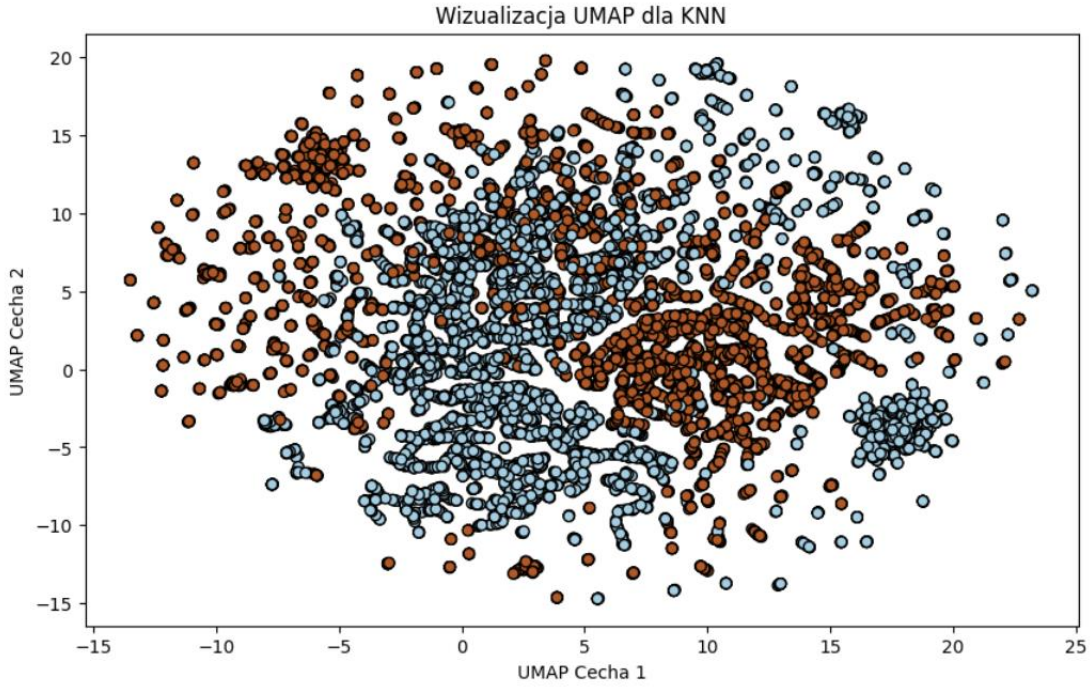
Na wykresie algorytmu SVC, punkty danych są przedstawiane jako punkty na płaszczyźnie, podobnie jak na wykresie UMAP. Jednak, w przypadku wykresu SVC, dodatkowo rysowane są linie lub hiperpłaszczyzny, które są granicami decyzyjnymi między klasami. Z wykresu wynika, że algorytm jasno podzielił dane, miał problem z nieliczną ilością danych.



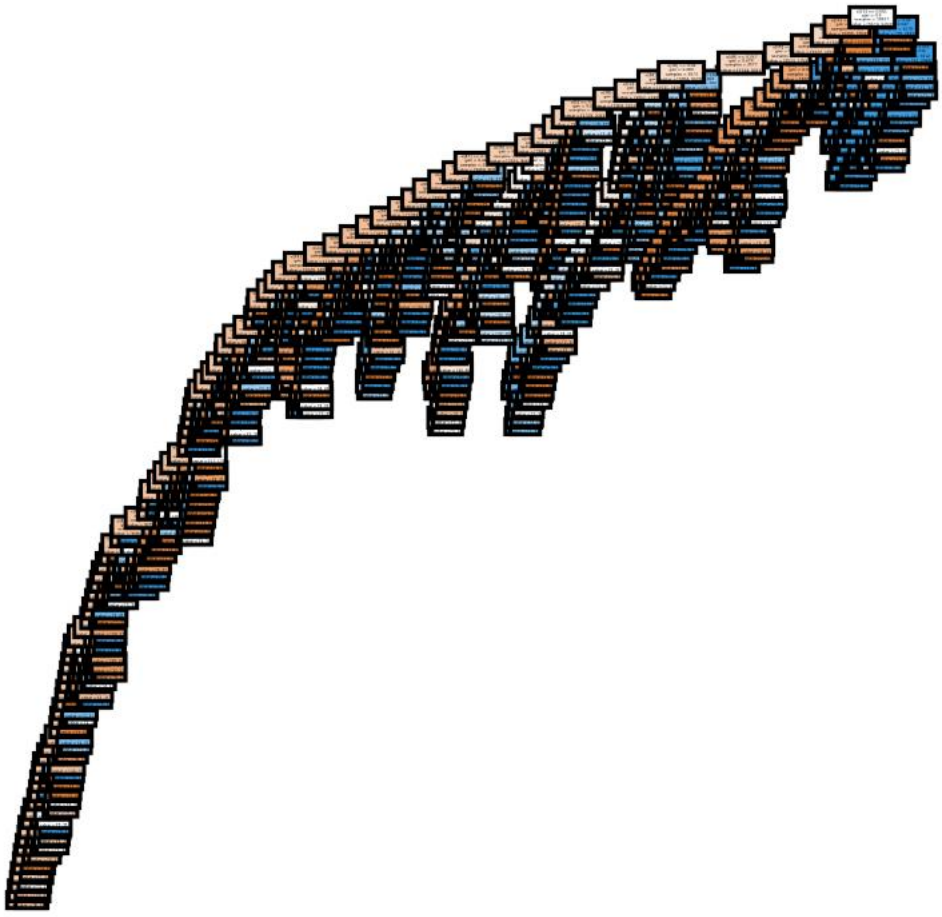
W przypadku danych testowych algorytm również nie miał większych problemów z przyporządkowaniem danych do konkretnych obszarów.



Algorytm k najbliższych sąsiadów prezentuje, jak punkty danych grupują się wokół siebie na podstawie podobieństwa. Punkty blisko siebie są bardziej podobne i mają większą szansę na przynależność do tej samej klasy. Wykres k-NN pomaga nam zobaczyć, jak algorytm k-NN klasyfikuje dane i jak dobrze radzi sobie z rozróżnianiem różnych klas. Możliwa jest również identyfikacja obszarów o wysokiej koncentracji punktów danej klasy. W powyższym wykresie widać, że dużo punktów umiejscowione jest w części wspólnej obu cech. Oznaczać to może, że algorytm nie radzi sobie idealnie z danymi, które otrzymał.



Wizualizacja UMAP dla algorytmu k-NN przedstawia sposób, w jaki dane są zorganizowane w przestrzeni na podstawie podobieństwa między nimi, przy użyciu metody redukcji wymiarowości UMAP. Na wykresie UMAP dla k-NN, punkty danych są reprezentowane jako punkty na płaszczyźnie, podobnie jak na zwykłym wykresie UMAP. Jednak w przypadku wizualizacji dla k-NN, dodatkowo oznaczane są sąsiedztwa punktów, które zostały znalezione przez algorytm k-NN. Na powyższym wykresie można zobaczyć skupiska punktów wskazujące na poszczególne grupy. Widać jak algorytm dzieli te punkty oraz przyporządkowuje je do odpowiednich grup.



Zrzut ekranu 8 Drzewo dezyzyjne

Powyższy zrzut ekranu przedstawia drzewo decyzyjne. Dodane w celu wizualizacji jego wyglądu.

Poniżej przedstawione są wyrazy ułożone od najbardziej wskazującego na SPAM. Jak widać na pierwszym miejscu jest słowo co, następnie com, news, http…

```
Ważność cechy co: 0.4797897498685614
Ważność cechy com: 0.11321585858685025
Ważność cechy news: 0.04596221413707394
```

```
Ważność cechy http: 0.0449201468846271
Ważność cechy to: 0.019914422200486104
Ważność cechy the: 0.019010390328662103
Ważność cechy in: 0.015834878318318785
Ważność cechy you: 0.014932141280725014
Ważność cechy and: 0.013120507286773586
Ważność cechy for: 0.012751933963212387
Ważność cechy is: 0.012048120664081544
Ważność cechy trump: 0.011984987954613 41
Ważność cechy of: 0.011062152275510674
Ważność cechy it: 0.010971702022644898
Ważność cechy not: 0.010196207734100984
Ważność cechy that: 0.009552888710143249
Ważność cechy on: 0.008829005197879816
Ważność cechy we: 0.008682783398666896
Ważność cechy my: 0.007970127982170777
Ważność cechy all: 0.007495046170507338
Ważność cechy will: 0.0071523299109526005
Ważność cechy at: 0.0064666747109369555
Ważność cechy from: 0.0064114030631580035
Ważność cechy have: 0.006234863496339671
Ważność cechy so: 0.005929258777414123
Ważność cechy this: 0.0058158637659929905
Ważność cechy be: 0.005652482059070848
Ważność cechy about: 0.005559244025537569
Ważność cechy can: 0.005446520505068791
Ważność cechy me: 0.0052307386418289255
Ważność cechy your: 0.005158845469654001
Ważność cechy https: 0.00507146768945335
Ważność cechy as: 0.005017566386790228
Ważność cechy they: 0.004958319399887426
Ważność cechy with: 0.004758541080338947
Ważność cechy by: 0.004725058960617622
Ważność cechy are: 0.004158465696372798
Ważność cechy just: 0.003807071561644599
Ważność cechy up: 0.0031277736860537016
Ważność cechy was: 0.0030897086162110367
Ważność cechy when: 0.0029813005341207508
Ważność cechy what: 0.0028247791487250175
Ważność cechy new: 0.002579429963606268
Ważność cechy do: 0.002547451250583344
Ważność cechy out: 0.0025383413633355014
Ważność cechy but: 0.002219404090144471
Ważność cechy twitter: 0.0010399065241950556
Ważność cechy like: 0.0008905890518700177
Ważność cechy pic: 0.0003613356044852834
```

7. Podsumowanie

W projekcie zdaniem było znalezienie algorytmu uczenia maszynowego, który najlepiej wyliczy dokładność w identyfikacji spamu. Całość projektu została napisana w języku programowania Python, główne użyte biblioteki to Pandas i Sklearn. Poszczególne etapy realizacji projektu zostały zapisane w formie zeszytu Jupyter Notebook, który był uzupełniany wraz z postępem realizacji. Pomocnym narzędziem okazało się narzędzie MS Office Excel, w

sposób szybki i klarowny pozwalał on na przeglądanie danych użytych do projektu. Algorytmy, które zostały zaimplementowane pozwoliły uzyskać wysokie dokładności. W pierwszej próbie najwyższy wynik uzyskano po zastosowaniu algorytmu SVC – wartości to 85,07% po wektoryzacji worek słów (przy stosunku danych testowych równym 0,2), 86,01% po wektoryzacji metodą td-idf (przy stosunku danych testowych równym 0.5). W drugiej próbie użyto algorytmu Bert, który pozwolił na uzyskanie najwyższego wyniku 93,79%. Według algorytmu 5 słów mających największy wpływ na weryfikację czy to spam to kolejno: co, com, news, http.

8. Bibliografia

[1] https://scikit-learn.org/stable/modules/tree.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.kneighbors

[3] https://scikit-learn.org/stable/modules/ensemble.html#adaboost

[4] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB

[5] https://scikit-learn.org/stable/modules/lda_qda.html#qda

[6] Wykłady

[7] https://jupyter.org

[8] https://numpy.org

[9] https://scikit-learn.org/stable/

[10] https://matplotlib.org

[11] https://pandas.pydata.org

Spis zrzutów ekranu

Spis tabel