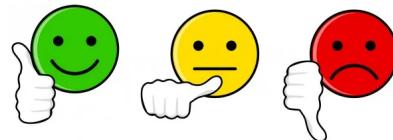


# Wprowadzenie do przetwarzania języka naturalnego

Dokumentacja końcowa

*Wykrywanie wydźwięku (sentiment analysis) opinii o produktach*



Prowadzący  
dr inż. Piotr Andruszkiewicz

Jakub Firlej  
Wojciech Gierulski  
Jan Kaniuka

Warszawa 2023

<b>1. Definicja problemu</b>	<b>3</b>
<b>2. Studia literaturowe</b>	<b>3</b>
2.1 Rule-Based Sentiment Analysis	3
2.2 Bag of Words Vectorization-Based Models	4
2.3 Transformer-Based Models	4
2.4 Different classifiers types	4
2.5 LSTM (Long Short Term Memory) Neural Networks	4
<b>3. Opis rozwiązania</b>	<b>5</b>
3.1 Wybrany model	5
3.2 Dotrenowanie istniejącego modelu	6
<b>4. Implementacja</b>	<b>7</b>
4.1 Język programowania	7
4.2 Web scraping	7
4.3 Wstępna analiza zebranych danych	8
4.4 Klasyfikacja	9
4.5 Opis wykorzystywanych bibliotek	10
<b>5. Instrukcja obsługi</b>	<b>11</b>
<b>6. Testy dla różnych klasyfikatorów</b>	<b>12</b>
6.1 Dane treningowe i walidacyjne	13
6.2 Testy dla zbioru niebilansowanego	13
6.3 Zbiór zbilansowany	17
6.4 Zbiór zbilansowany testowany na najlepszych dostrojonych klasyfikatorach	21
6.5 Zbiór zbilansowany z uwzględnieniem bigramów	22
<b>7. Obserwacje i wnioski</b>	<b>23</b>
7.1 Najczęstsze słowa	23
7.2 Najczęstsze bigramy	24
<b>8. Dotrenowanie modelu BERT</b>	<b>25</b>
<b>9. Porównanie wyników z innym modelem</b>	<b>29</b>
<b>10. Wnioski</b>	<b>30</b>
<b>11. Literatura</b>	<b>30</b>

# 1. Definicja problemu

Temat naszego projektu to "**Wykrywanie wydźwięku (sentiment analysis) opinii o produktach**". Projekt polega na pobraniu opinii o produktach takich jak **proszki do prania, tabletki do zmywarki i kubki termiczne** oraz zrealizowaniu modelu wykrywającego wydźwięk: **pozytywny, neutralny, negatywny**. Ponadto opinie mają być w **języku niemieckim**.



Zadanie projektowe można zdekomponować na mniejsze podzadania. Podział ten kształtuje się następująco:

1. Pobranie opinii z portali internetowych i przygotowanie korpusu.
2. Stworzenie modelu.
3. Stworzenie drugiego modelu wykorzystującego dane dostępne w literaturze w ramach dotrenowania wykorzystywanego pretrenowanego modelu, np. *BERT*.

## 2. Studia literaturowe

Pracę nad projektem rozpoczęliśmy od przeglądu literatury/aktualnego stanu wiedzy w dziedzinie wykrywania wydźwięku. Rezultatem tego przeglądu było poznanie różnych metod, które stosuje się w analizie wydźwięku - ich możliwości oraz ograniczeń. Ten rozdział dokumentacji wstępnej zostanie poświęcony opisowi tychże właśnie metod.

### 2.1 Rule-Based Sentiment Analysis

W tym podejściu opinia jest klasyfikowana na podstawie określonych zasad. Zbiór tych zasad określa się jako słownik/leksykon - stąd to podejście nazywa się *Rule-based approach* albo *Lexicon based approach* [1]. Metody wykorzystujące to podejście to m.in: *TextBlob*, *VADER*, *SentiWordNet*.

W *TextBlob* każdemu ze słów składowych opinii przypisuje się ocenę na podstawie słownika. Korzysta się z predefiniowanego słownika z sklasyfikowanymi słowami pozytywnymi i negatywnymi. Kiedy każde słowo ma już przypisaną ocenę stosuje się wybraną miarę np. średnią ocen wszystkich słów i na podstawie tego wylicza wydźwięk opinii. *TextBlob* oprócz wydźwięku opinii (*polarity*) zwraca także miarę zwaną *subjectivity* - określa ona, czy tekst jest faktem, czy opinią. *VADER* i *SentiWordNet* to podobne podejścia do *TextBlob*. *VADER* najczęściej wykorzystuje się do badania tekstu z social mediów.

## 2.2 Bag of Words Vectorization-Based Models

To podejście mówi o sposobach reprezentacji języka naturalnego w formie numerycznej. Jedno z podejść to **Count Vectorizers** [2], czyli sposób na konwersję danego zestawu ciągów znaków na reprezentację częstotliwościową. Drugie podejście to **TF-IDF** (*Term Frequency - Inverse Document Frequency*). Jest to statystyka, która opiera się na częstotliwości występowania danego słowa w korpusie, ale również stanowi liczbową reprezentację tego, jak ważne jest dane słowo dla analizy statystycznej.

## 2.3 Transformer-Based Models

Jest to jedna z najbardziej zaawansowanych technik przetwarzania języka naturalnego. Wykorzystuje ona architekturę opartą na *koderze-dekoderze* [3]. Chociaż zawsze można zbudować model transformatorowy od podstaw, jest to dość żmudne zadanie. Możemy więc użyć wstępnie wytrenowanych modeli transformatorów dostępnych w Internecie.

## 2.4 Different classifiers types

Wykrywanie wydźwięku to tak naprawdę zadanie klasyfikacji. Każda opinia zostaje przyporządkowana do jednej z trzech klas: *positive*, *negative*, *neutral*. Po wybraniu cech i ustaleniu formy reprezentacji każdej z opinii można zastosować jeden z wielu klasyfikatorów. Klasyfikatory powszechnie stosowane przy *sentiment analysis* to m.in: klasyfikator k-najbliższych sąsiadów, drzewo decyzyjne, las losowy, klasyfikator *AdaBoost* [4].

## 2.5 LSTM (Long Short Term Memory) Neural Networks

LSTM to wariant rekurencyjnej sieci neuronowej [5]. Sieci LSTM dobrze radzą sobie z danymi w postaci szeregów czasowych i pozwalają uchwycić tzw. *long-term dependencies* (zależności między słowami, które są oddalone od siebie).

### 3. Opis rozwiązania

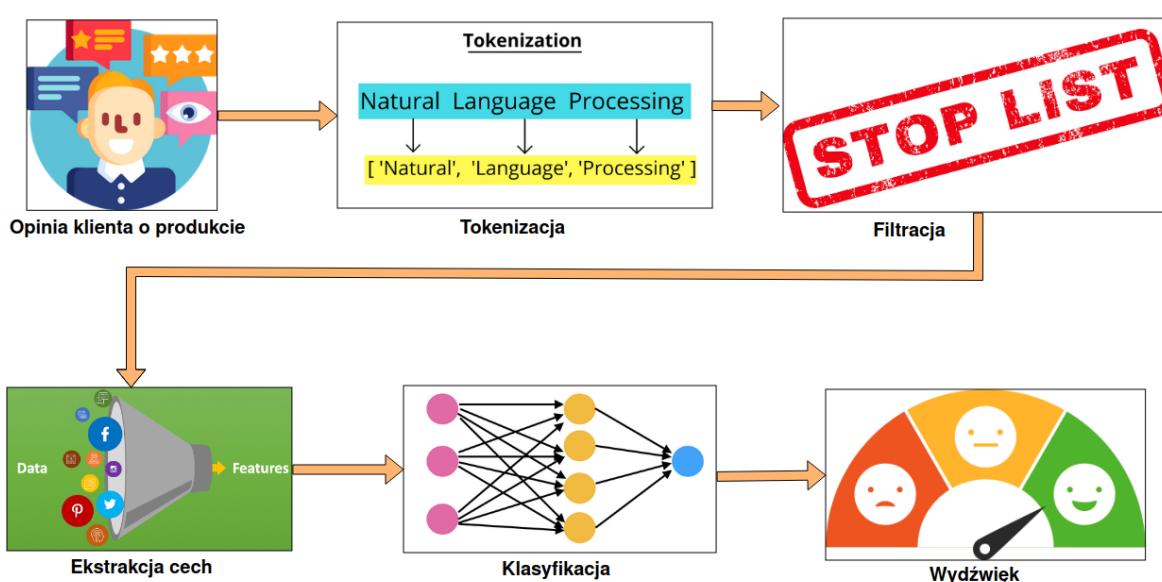
#### 3.1 Wybrany model

W celu wytrenowania modelu wykorzystano klasyczne podejście w uczeniu maszynowym:

1. Ekstrakcja cech
2. Klasyfikacja na podstawie cech

Pierwszym krokiem było podjęcie decyzji o wyborze cech, które będą reprezentować zebrane próbki danych (opinie). Została stworzona lista najpopularniejszych słów (\*i bigramów) dla opinii negatywnych, neutralnych i pozytywnych. Wektor cech dla konkretnej opinii koduje występowanie wybranych słów/bigramów z listy. Następnie przetestowano różne typy znanych klasyfikatorów (np. naiwny klasyfikator Bayesa, drzewa decyzyjne), żeby wybrać ten, który najlepiej radzi sobie z tym zdaniem.

W opisanym podejściu niezbędne było także **przefiltrowanie** opinii z wykorzystaniem gotowej *stop listy* dla języka niemieckiego, tak aby popularne słowa (niemające wyraźnego wydźwięku) nie zdominowały listy najczęściej występujących słów. Do przefiltrowania niezbędne okazało się też wykonanie wcześniej tokenizacji. Sekwencję czynności przedstawiono na poniższym rysunku.



## 3.2 Dotrenowanie istniejącego modelu

Jako model do dotrenowania wybraliśmy **BERT** (*Bidirectional Encoder Representations from Transformers*), ze względu na jego popularność. Wykorzystana została bibliotekę **transformers** udostępniona przez organizację *HuggingFace*. Biblioteka udostępnia duży zbiór gotowych modeli do pobrania oraz interfejs do ich trenowania. W ramach projektu zaplanowano dotrenowanie następujących modeli:

- **german-sentiment-bert** - model do wykrywania wydźwięku dla języka niemieckiego. Model był uczony na opiniach pochodzących z *Twittera*, *Facebooka* oraz różnych portali z opiniami o filmach i hotelach.
- **bert-base-german-cased** - ogólny model języka. Był trenowany bez konkretnego docelowego zastosowania oraz bez etykiet oznaczanych przez ludzi. Do trenowania wykorzystano teksty w języku niemieckim z następujących stron internetowych:
  - *Wikipedia*
  - *Open Legal Data*

## 4. Implementacja

### 4.1 Język programowania



Językiem programowania wykorzystywanym w projekcie jest **Python w wersji 3.9**. Wszystkie wykorzystywane w projekcie biblioteki umieszczone w pliku `requirements.txt` co znacząco ułatwia skonfigurowanie środowiska do uruchomienia projektu.

### 4.2 Web scraping

W celu pozyskania opinii o produktach po niemiecku wraz z ich wydźwiękiem, wykorzystano technikę **web-scraping'u**, czyli zautomatyzowanego wyciągania danych wprost ze strony HTML. Użyto do tego biblioteki *BeautifulSoup*.

Podstawowym źródłem danych były recenzje *per produkt* z niemieckiego serwisu zakupowego *Amazon*. Algorytm polega na odczytywaniu autora, treści oraz oceny (w postaci od jednej do pięciu gwiazdek) zawartych w recenzji, iterując po kolejnych stronach opinii użytkowników. Należało ręcznie wyselekcjonować i skategoryzować produkty i umieścić ich ID w pliku konfiguracyjnym. Na ich podstawie odpowiednio preparowany był link:

```
https://www.amazon.de/product-reviews/{product_id}/ref=cm_cr_ap_d_paging_btm_next_2?ie=UTF8&reviewerType=all_reviews&pageNumber={page_number}
```

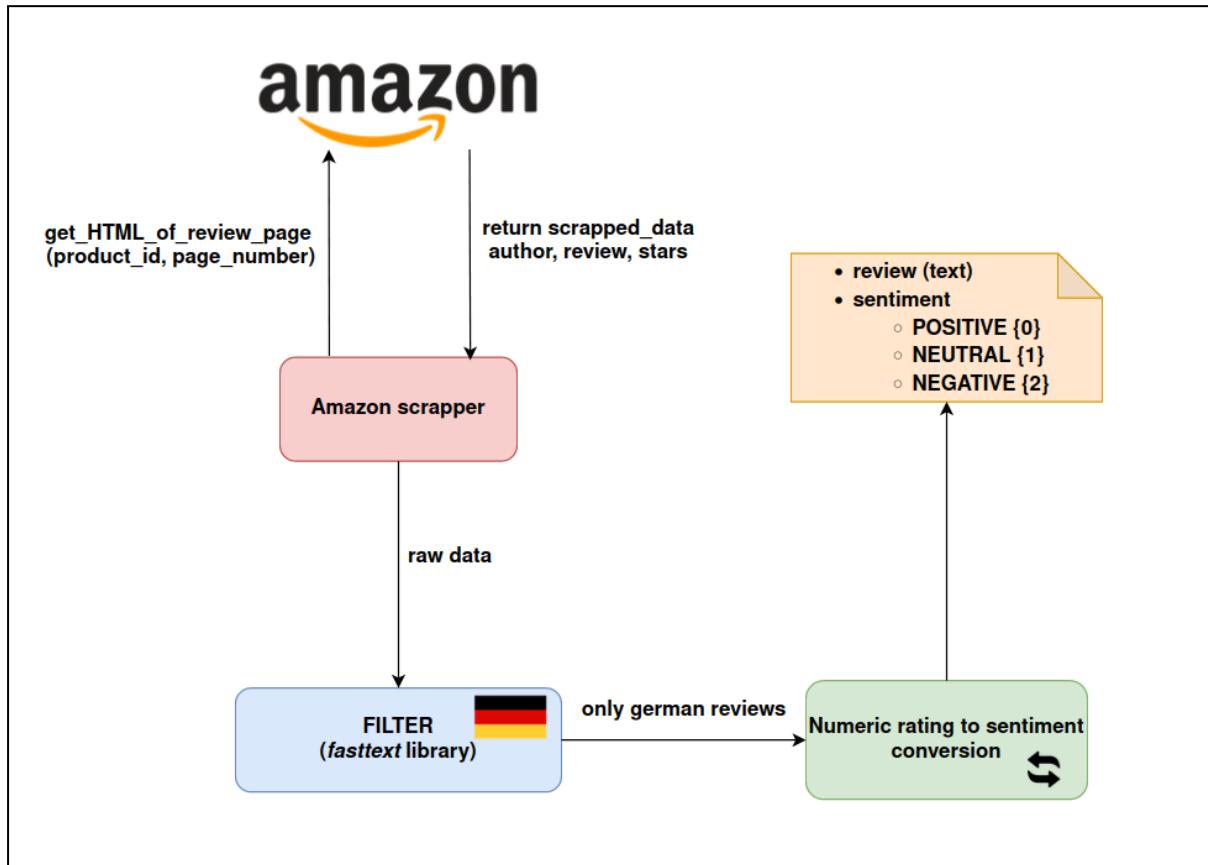
gdzie **{product\_id}** to ID produktu, a **{page\_number}** to rosnący (dopóki istnieje) numer strony opinii użytkowników.

Dane były filtrowane poprzez wykrywanie języka klasyfikatorem dostarczonym przez bibliotekę **fasttext**. Jeżeli wykrytym językiem był język niemiecki, dane były uwzględniane. Jeżeli wykryty język był inny lub nie wykryto języka w ogóle (treść recenzji to często sam emotikon, np. kciuk w góre ), takie dane były wtedy odrzucane. Dane zostały również zabezpieczone przed duplikatami, a raz zapisane zachowują swoją kolejność.

Ocena w postaci gwiazdek jest rzutowana do postaci etykiet:

- 1-2 gwiazdek skutkuje sentymentem “**NEGATIVE**”
- 3 gwiazdki skutkują sentymentem “**NEUTRAL**”
- 4-5 gwiazdek skutkuje sentymentem “**POSITIVE**”

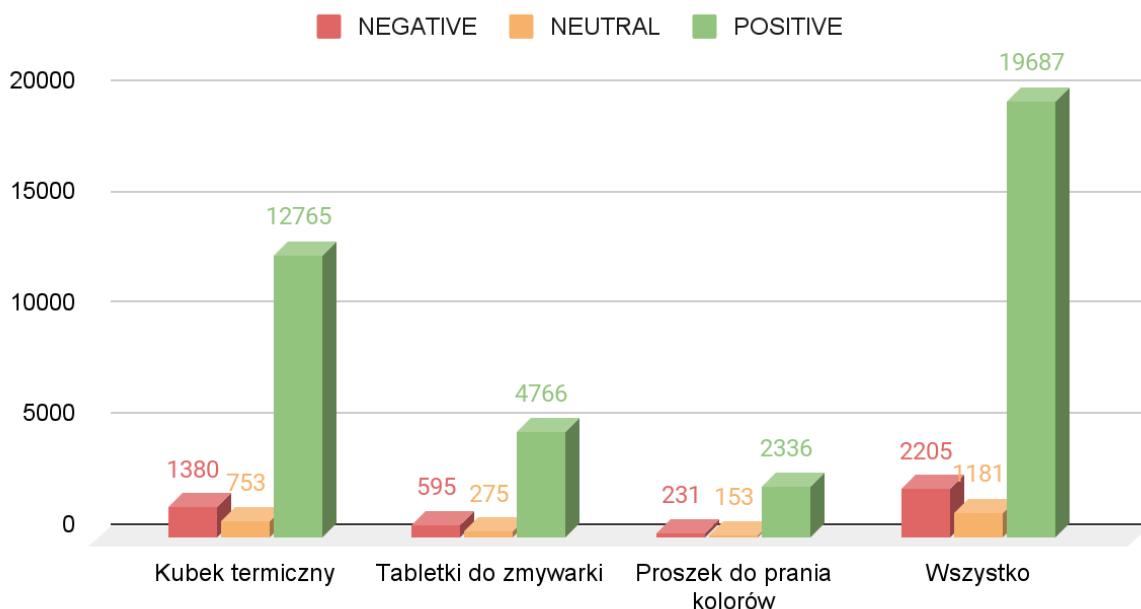
Cały proces web-scraping'u przedstawiono na poniższym diagramie:



### 4.3 Wstępna analiza zebranych danych

Ostatecznie, otrzymano następującą statystykę recenzji dla poszczególnych typów produktów:

Liczba sklasyfikowanych recenzji dla produktów



Niezależnie od typu produktu, najwięcej recenzji jest zawsze pozytywnych. Stosunek ilości recenzji o wydźwięku negatywnym do recenzji o wydźwięku neutralnym jest różny zależnie od typu produktu, jednak ogólnie ich liczba jest do siebie zbliżona.

Najwięcej recenzji znaleziono dla kubka termicznego, następnie dla tabletek do zmywarki i najmniej dla proszku do prania kolorowych ubrań. Sumarycznie znaleziono:

- 14898 recenzji dla kubków termicznych
- 5636 recenzji dla tabletek do zmywarki
- 2720 recenzji dla proszków do prania
- 23073 recenzji łącznie dla wszystkich trzech typów produktów

W celu zbilansowania klas należało odpowiednio odrzucić pewną część recenzji ze zbioru, aby osiągnąć w przybliżeniu równomierny rozkład klas.

## 4.4 Klasyfikacja

Przed rozpoczęciem eksperymentów koniecznym krokiem był **podział zebranych danych na zbiór treningowy i walidacyjny**. Do wykonania podziału wykorzystano dedykowaną funkcję `train_test_split()` z biblioteki `sklearn`. 75% danych stanowiło zbiór treningowy, a pozostałe 25% postużyły do walidacji modelu. Przygotowano dwa zestawy zbiorów treningowego i walidacyjnego.

- Pierwszy zestaw (pliki `train.csv` i `test.csv`) utworzono na bazie wszystkich pozyskanych danych.
- Drugi zestaw (pliki `train_balanced.csv` i `test_balanced.csv`) utworzono po odrzuceniu znacznej części opinii pozytywnych. Pozwoliło to na stworzenie zbioru treningowego zawierającego równoliczne zbiory opinii pozytywnych, neutralnych i negatywnych.

Następnie na podstawie zbioru treningowego **wygenerowano trzy korpusy** i umieszczono je w katalogu `/corpora_nlp22z`. Każdy z korpusów (w formacie `.txt`) zawierał jedynie opinie o jednakowym wydźwięku. Drugi zestaw korpusów przygotowano na podstawie zbioru treningowego zbalansowanego.

Kolejnym krokiem była **ekstrakcja cech**, które będą reprezentować zebrane próbki danych (opinie). Przed ekstrakcją należało odpowiednio przefiltrować dane i usunąć ich duplikaty:

- W ramach filtracji najpierw usunięto słowa ze **stoplisty** dla języka niemieckiego oraz wszystkie rzeczowniki (**POS tag** = “NN” oraz “NNP”(ang. *proper noun*) - nazwa własna)). Przed usunięciem rzeczowników najczęściej występującym słowem w każdym zbiorze było niemieckie słowo Becher (kubek).
- Na podstawie rozkładów częstościowych (`nltk.FreqDist()`) utworzono listy najczęściej występujących słów w każdym z korpusów. Pojawiło się wiele duplikatów w trzech listach, dlatego wykonano procedurę

usuwania duplikatów poprzez sprawdzenie, w której liście dane słowo występuje najczęściej i usunięcie tego słowa z dwóch pozostałych. Ostatecznie osiągnięto trzy listy stu najczęstszych słów w każdym korpusie.

Każda opinia jest reprezentowana przez 300-elementowy słownik (Python `dict()`). Pierwsze 100 kluczy to najpopularniejsze słowa występujące w opiniach pozytywnych, kolejne 100 to najczęstsze słowa z opinii neutralnych, a ostatnie 100 kluczy to najpopularniejsze słowa z opinii negatywnych. Wartości dla poszczególnych kluczy odpowiadają liczności słów kluczy w analizowanej opinii.

Przetestowane zostały również alternatywne **reprezentacje wektora cech**, gdzie zastosowano:

- 3-elementowy wektor cech → wartość pierwszego elementu była równa liczbie wystąpień słów z listy 100 najpopularniejszych słów dla opinii pozytywnych (kolejne elementy analogicznie).
- 30-elementowy wektor cech → w tym przypadku listy najczęściej występujących słów miały długość 10 elementów, a nie 100.
- 330-elementowy wektor cech → 300-elementowy wektor cech rozszerzono o 30 elementów. Na te 30 dodatkowych elementów składają się 3 listy 10 najczęściej występujących **bigramów**.

Do eksperymentów wybrano następujące **klasyfikatory** z biblioteki sklearn: `KNeighborsClassifier`, `DecisionTreeClassifier`, `RandomForestClassifier`, `AdaBoostClassifier`, `LogisticRegression`, `MLPClassifier`.

## 4.5 Opis wykorzystywanych bibliotek

Poniżej opisano nie wszystkie, lecz jedynie kluczowe z punktu widzenia całości projektu biblioteki:

- `requests` - przesyłanie żądania *HTTP* w celu pobrania opinii ze strony internetowej sklepu,
- `beautifulsoup4` - biblioteka ułatwiająca scrapowanie danych z sieci *Web*,
- `fasttext` - wykrywanie języka w celu odrzucenia recenzji napisanych w języku innym niż niemiecki ,
- `pandas` - obsługa plików *CSV* z danymi treningowymi i walidacyjnymi,
- `nltk` - tworzenie rozkładów częstościowych, wykorzystanie `stoplisty`, oznaczanie bigramów/trigramów,
- `spacy` - oznaczanie słów w celu wykrycia rzeczowników (*POS tagging*)
- `sklearn` - klasyfikacja, generowanie macierzy pomyłek i raportu klasyfikacji,
- `matplotlib` - graficzna prezentacja zebranych danych.

## 5. Instrukcja obsługi

Instrukcja uruchomienia utworzonych w ramach projektu programów znajduje się w pliku *ReadMe* repozytorium projektowego w serwisie GitHub. Umieszczone tam również listę wymaganych bibliotek, które należy zainstalować przed uruchomieniem programu.



Link do repozytorium GitHub: [LINK](#)

Wszystkie parametry programów zostały umieszczone w osobnym pliku **config.ini**, aby móc je sprawnie modyfikować.

## 6. Testy dla różnych klasyfikatorów

Wykonano testy przy użyciu sześciu klasyfikatorów z biblioteki *sklearn* z domyślnymi wartościami hiperparametrów:

- K-Neighbors

```
k_neighbors_kwargs = {
    "n_neighbors": 5, "weights": "uniform", "algorithm": "auto", "leaf_size": 30,
    "p": 2, "metric": "minkowski", "metric_params": None, "n_jobs": None
}
```

- Decision Tree

```
decision_tree_kwargs = {
    "criterion": "gini", "splitter": "best", "max_depth": None, "min_samples_split": 2,
    "min_samples_leaf": 1, "min_weight_fraction_leaf": 0.0, "max_features": None,
    "random_state": None, "max_leaf_nodes": None, "min_impurity_decrease": 0.0,
    "class_weight": None, "ccp_alpha": 0.0
}
```

- Random Forest

```
random_forest_kwargs = {
    "n_estimators": 100, "criterion": "gini", "max_depth": None, "min_samples_split": 2,
    "min_samples_leaf": 1, "min_weight_fraction_leaf": 0.0, "max_features": "sqrt",
    "max_leaf_nodes": None, "min_impurity_decrease": 0.0, "bootstrap": True, "oob_score": False,
    "n_jobs": None, "random_state": None, "verbose": 0, "warm_start": False, "class_weight": None,
    "ccp_alpha": 0.0, "max_samples": None
}
```

- Logistic Regression

```
logistic_regression_kwargs = {
    "penalty": "l2", "dual": False, "tol": 1e-4, "C": 1.0, "fit_intercept": True,
    "intercept_scaling": 1, "class_weight": None, "random_state": None, "solver": "lbfgs",
    "max_iter": 100, "multi_class": "auto", "verbose": 0, "warm_start": False, "n_jobs": None,
    "l1_ratio": None
}
```

- MLP

```
mlp_kwargs = {
    "hidden_layer_sizes": (100,), "activation": "relu", "solver": "adam", "alpha": 0.0001,
    "batch_size": "auto", "learning_rate": "constant", "learning_rate_init": 0.001, "power_t": 0.5,
    "max_iter": 200, "shuffle": True, "random_state": None, "tol": 1e-4, "verbose": False,
    "warm_start": False, "momentum": 0.9, "nesterovs_momentum": True, "early_stopping": False,
    "validation_fraction": 0.1, "beta_1": 0.9, "beta_2": 0.999, "epsilon": 1e-8, "n_iter_no_change": 10,
    "max_fun": 15000
}
```

- Ada Boost

```
ada_boost_kwargs = {
    "estimator": None, "n_estimators": 50, "learning_rate": 1.0, "algorithm": "SAMME.R",
    "random_state": None, "base_estimator": "deprecated"
}
```

W celu umożliwienia porównania jakości klasyfikacji różnych klasyfikatorów, za każdym razem generowano **macierz pomyłek** oraz wyliczano odpowiednie **metryki** (*accuracy, recall, precision* etc.).

## 6.1 Dane treningowe i walidacyjne

Przeprowadzono testy dla danych niezbilansowanych:

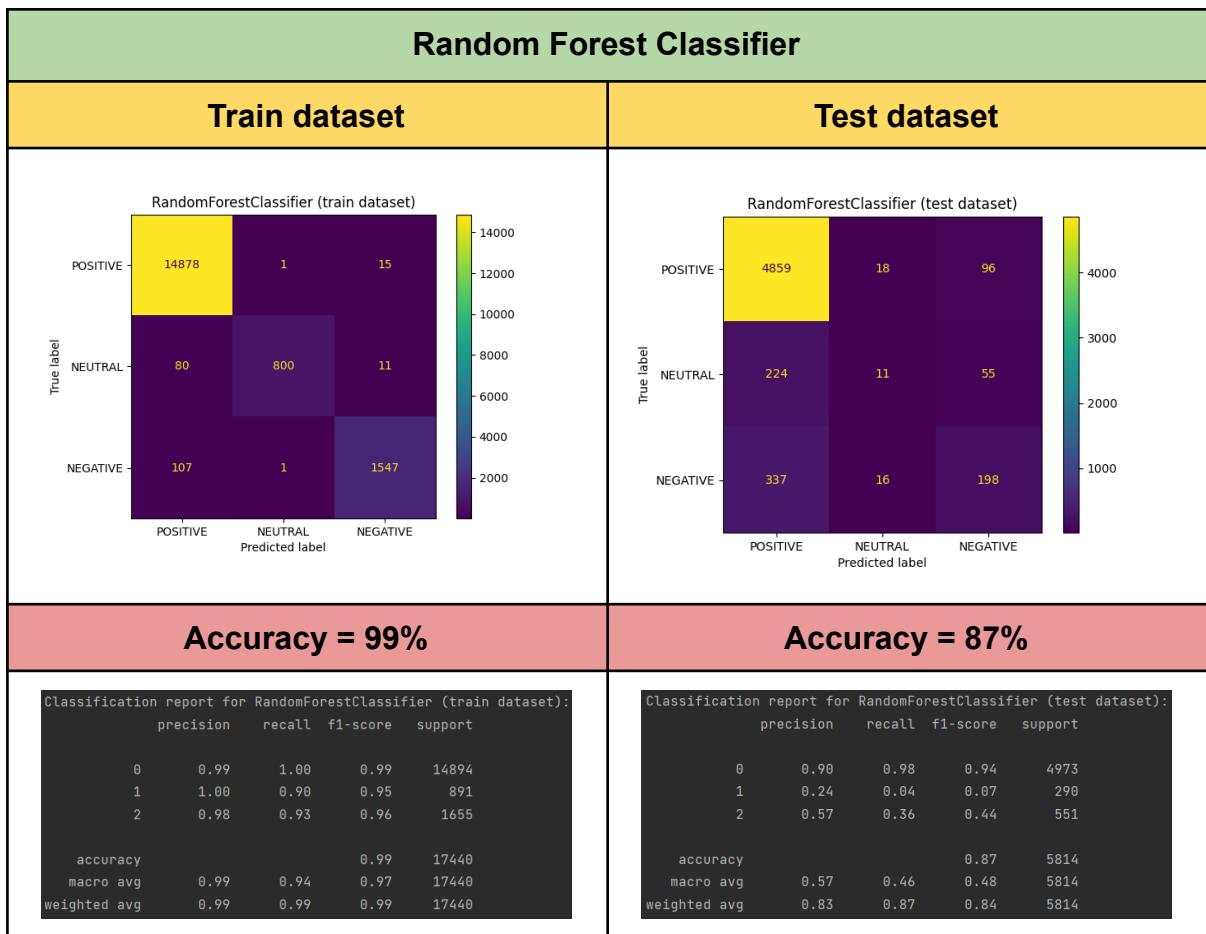
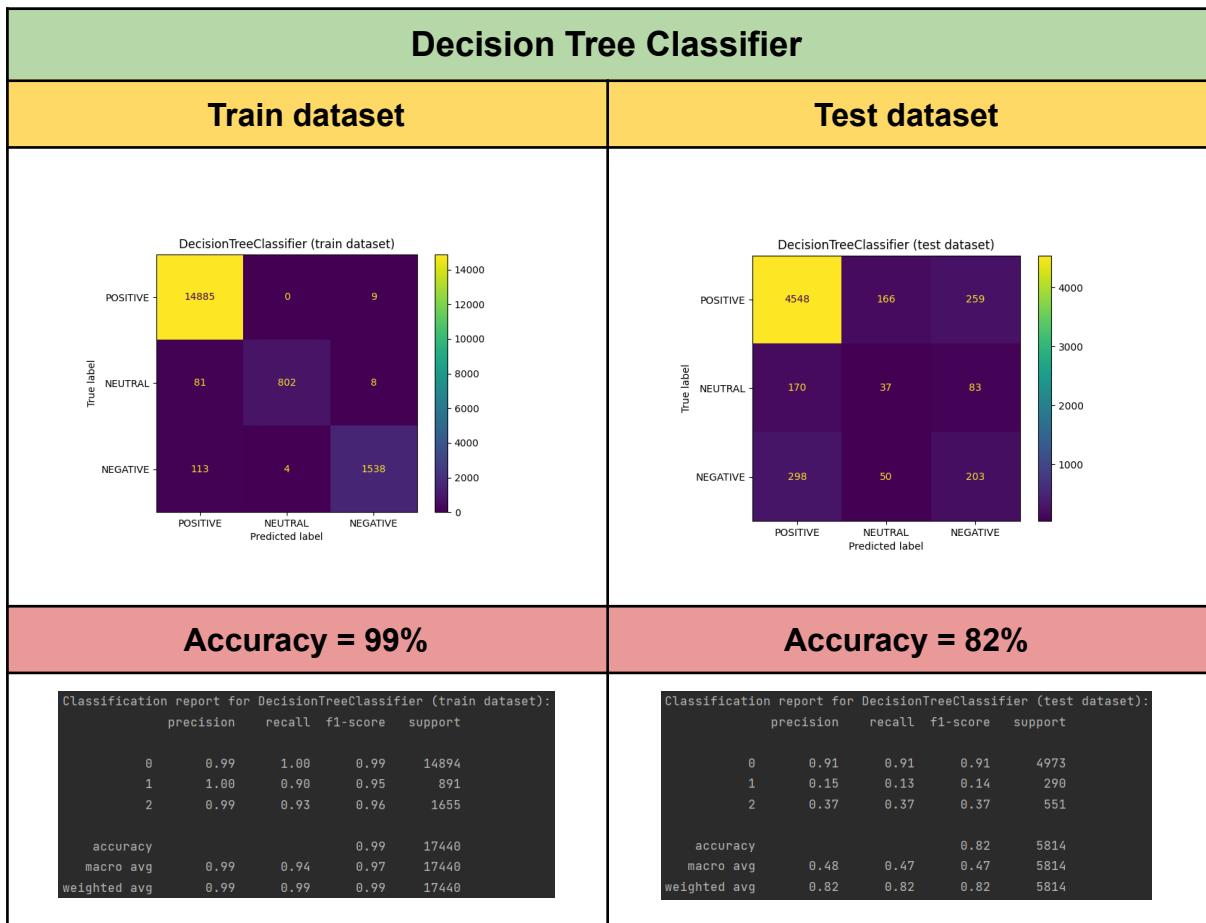
- Opinie pozytywne - 14894 w danych treningowych, 4973 w danych walidacyjnych
- Opinie neutralne - 891 w danych treningowych, 290 w danych walidacyjnych
- Opinie negatywne - 1655 w danych treningowych, 551 w danych walidacyjnych

Następnie przeprowadzono testy dla danych zbilansowanych:

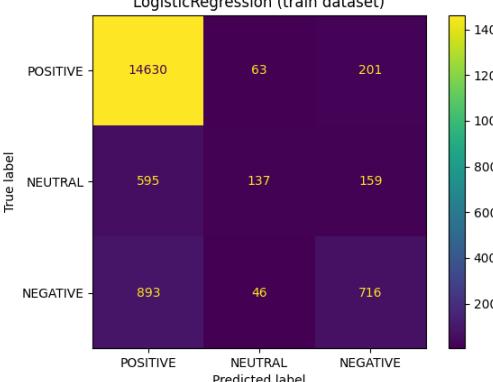
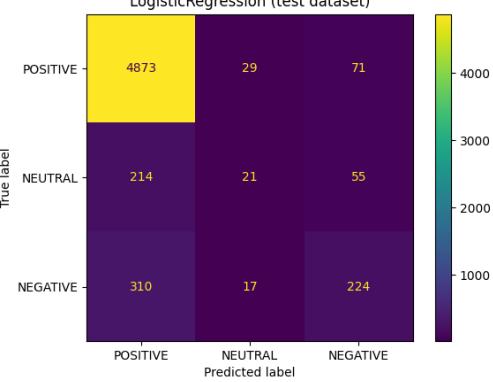
- Opinie pozytywne - 1000 w danych treningowych, 290 w danych walidacyjnych
- Opinie neutralne - 891 w danych treningowych, 290 w danych walidacyjnych
- Opinie negatywne - 1000 w danych treningowych, 290 w danych walidacyjnych

## 6.2 Testy dla zbioru niezbilansowanego

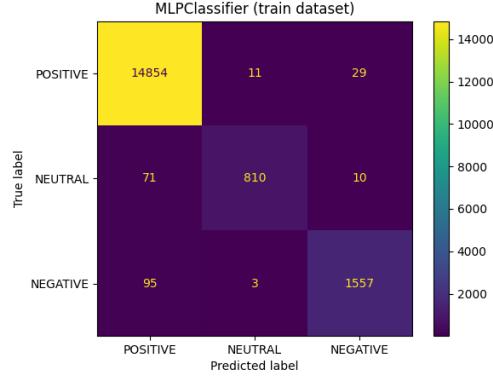
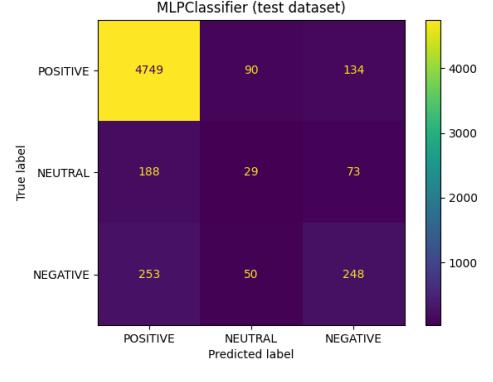


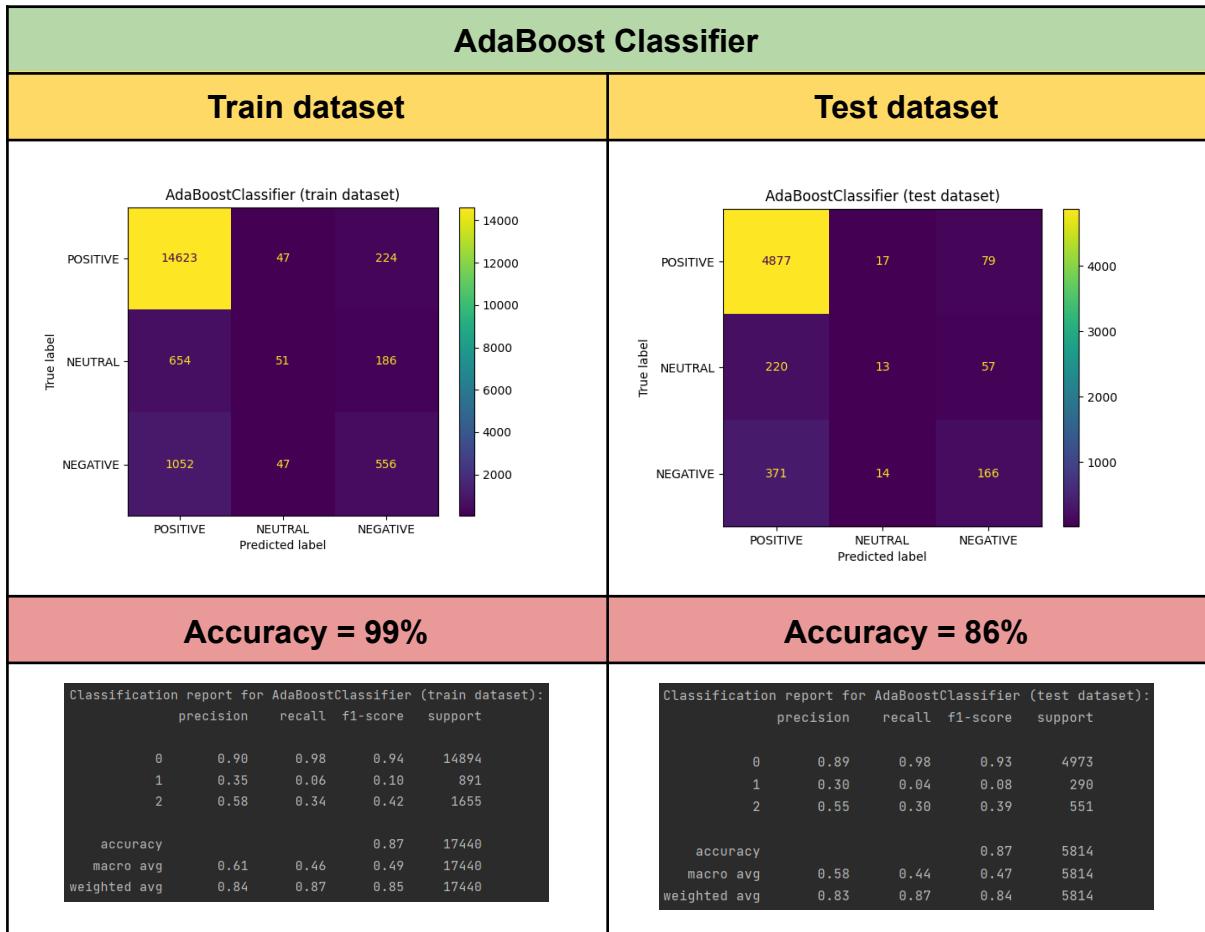


## Logistic Regression

Train dataset			Test dataset																																																																								
<b>LogisticRegression (train dataset)</b>  <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>POSITIVE</td><td>14630</td><td>63</td><td>201</td></tr> <tr><td>NEUTRAL</td><td>595</td><td>137</td><td>159</td></tr> <tr><td>NEGATIVE</td><td>893</td><td>46</td><td>716</td></tr> </table>			POSITIVE	14630	63	201	NEUTRAL	595	137	159	NEGATIVE	893	46	716	<b>LogisticRegression (test dataset)</b>  <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>POSITIVE</td><td>4873</td><td>29</td><td>71</td></tr> <tr><td>NEUTRAL</td><td>214</td><td>21</td><td>55</td></tr> <tr><td>NEGATIVE</td><td>310</td><td>17</td><td>224</td></tr> </table>			POSITIVE	4873	29	71	NEUTRAL	214	21	55	NEGATIVE	310	17	224																																														
POSITIVE	14630	63	201																																																																								
NEUTRAL	595	137	159																																																																								
NEGATIVE	893	46	716																																																																								
POSITIVE	4873	29	71																																																																								
NEUTRAL	214	21	55																																																																								
NEGATIVE	310	17	224																																																																								
<b>Accuracy = 89%</b>			<b>Accuracy = 88%</b>																																																																								
Classification report for LogisticRegression (train dataset): <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr> </thead> <tbody> <tr><td>0</td><td>0.91</td><td>0.98</td><td>0.94</td><td>14894</td></tr> <tr><td>1</td><td>0.56</td><td>0.15</td><td>0.24</td><td>891</td></tr> <tr><td>2</td><td>0.67</td><td>0.43</td><td>0.52</td><td>1655</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.89</td><td>17440</td></tr> <tr><td>macro avg</td><td>0.71</td><td>0.52</td><td>0.57</td><td>17440</td></tr> <tr><td>weighted avg</td><td>0.87</td><td>0.89</td><td>0.87</td><td>17440</td></tr> </tbody> </table>				precision	recall	f1-score	support	0	0.91	0.98	0.94	14894	1	0.56	0.15	0.24	891	2	0.67	0.43	0.52	1655	accuracy			0.89	17440	macro avg	0.71	0.52	0.57	17440	weighted avg	0.87	0.89	0.87	17440	Classification report for LogisticRegression (test dataset): <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr> </thead> <tbody> <tr><td>0</td><td>0.90</td><td>0.98</td><td>0.94</td><td>4973</td></tr> <tr><td>1</td><td>0.31</td><td>0.07</td><td>0.12</td><td>290</td></tr> <tr><td>2</td><td>0.64</td><td>0.41</td><td>0.50</td><td>551</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.88</td><td>5814</td></tr> <tr><td>macro avg</td><td>0.62</td><td>0.49</td><td>0.52</td><td>5814</td></tr> <tr><td>weighted avg</td><td>0.85</td><td>0.88</td><td>0.86</td><td>5814</td></tr> </tbody> </table>				precision	recall	f1-score	support	0	0.90	0.98	0.94	4973	1	0.31	0.07	0.12	290	2	0.64	0.41	0.50	551	accuracy			0.88	5814	macro avg	0.62	0.49	0.52	5814	weighted avg	0.85	0.88	0.86	5814
	precision	recall	f1-score	support																																																																							
0	0.91	0.98	0.94	14894																																																																							
1	0.56	0.15	0.24	891																																																																							
2	0.67	0.43	0.52	1655																																																																							
accuracy			0.89	17440																																																																							
macro avg	0.71	0.52	0.57	17440																																																																							
weighted avg	0.87	0.89	0.87	17440																																																																							
	precision	recall	f1-score	support																																																																							
0	0.90	0.98	0.94	4973																																																																							
1	0.31	0.07	0.12	290																																																																							
2	0.64	0.41	0.50	551																																																																							
accuracy			0.88	5814																																																																							
macro avg	0.62	0.49	0.52	5814																																																																							
weighted avg	0.85	0.88	0.86	5814																																																																							

## MLP Classifier

Train dataset			Test dataset																																																																								
<b>MLPClassifier (train dataset)</b>  <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>POSITIVE</td><td>14854</td><td>11</td><td>29</td></tr> <tr><td>NEUTRAL</td><td>71</td><td>810</td><td>10</td></tr> <tr><td>NEGATIVE</td><td>95</td><td>3</td><td>1557</td></tr> </table>			POSITIVE	14854	11	29	NEUTRAL	71	810	10	NEGATIVE	95	3	1557	<b>MLPClassifier (test dataset)</b>  <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>POSITIVE</td><td>4749</td><td>90</td><td>134</td></tr> <tr><td>NEUTRAL</td><td>188</td><td>29</td><td>73</td></tr> <tr><td>NEGATIVE</td><td>253</td><td>50</td><td>248</td></tr> </table>			POSITIVE	4749	90	134	NEUTRAL	188	29	73	NEGATIVE	253	50	248																																														
POSITIVE	14854	11	29																																																																								
NEUTRAL	71	810	10																																																																								
NEGATIVE	95	3	1557																																																																								
POSITIVE	4749	90	134																																																																								
NEUTRAL	188	29	73																																																																								
NEGATIVE	253	50	248																																																																								
<b>Accuracy = 99%</b>			<b>Accuracy = 86%</b>																																																																								
Classification report for MLPClassifier (train dataset): <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr> </thead> <tbody> <tr><td>0</td><td>0.99</td><td>1.00</td><td>0.99</td><td>14894</td></tr> <tr><td>1</td><td>0.98</td><td>0.91</td><td>0.94</td><td>891</td></tr> <tr><td>2</td><td>0.98</td><td>0.94</td><td>0.96</td><td>1655</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.99</td><td>17440</td></tr> <tr><td>macro avg</td><td>0.98</td><td>0.95</td><td>0.97</td><td>17440</td></tr> <tr><td>weighted avg</td><td>0.99</td><td>0.99</td><td>0.99</td><td>17440</td></tr> </tbody> </table>				precision	recall	f1-score	support	0	0.99	1.00	0.99	14894	1	0.98	0.91	0.94	891	2	0.98	0.94	0.96	1655	accuracy			0.99	17440	macro avg	0.98	0.95	0.97	17440	weighted avg	0.99	0.99	0.99	17440	Classification report for MLPClassifier (test dataset): <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr> </thead> <tbody> <tr><td>0</td><td>0.92</td><td>0.95</td><td>0.93</td><td>4973</td></tr> <tr><td>1</td><td>0.17</td><td>0.10</td><td>0.13</td><td>290</td></tr> <tr><td>2</td><td>0.55</td><td>0.45</td><td>0.49</td><td>551</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.86</td><td>5814</td></tr> <tr><td>macro avg</td><td>0.54</td><td>0.50</td><td>0.52</td><td>5814</td></tr> <tr><td>weighted avg</td><td>0.84</td><td>0.86</td><td>0.85</td><td>5814</td></tr> </tbody> </table>				precision	recall	f1-score	support	0	0.92	0.95	0.93	4973	1	0.17	0.10	0.13	290	2	0.55	0.45	0.49	551	accuracy			0.86	5814	macro avg	0.54	0.50	0.52	5814	weighted avg	0.84	0.86	0.85	5814
	precision	recall	f1-score	support																																																																							
0	0.99	1.00	0.99	14894																																																																							
1	0.98	0.91	0.94	891																																																																							
2	0.98	0.94	0.96	1655																																																																							
accuracy			0.99	17440																																																																							
macro avg	0.98	0.95	0.97	17440																																																																							
weighted avg	0.99	0.99	0.99	17440																																																																							
	precision	recall	f1-score	support																																																																							
0	0.92	0.95	0.93	4973																																																																							
1	0.17	0.10	0.13	290																																																																							
2	0.55	0.45	0.49	551																																																																							
accuracy			0.86	5814																																																																							
macro avg	0.54	0.50	0.52	5814																																																																							
weighted avg	0.84	0.86	0.85	5814																																																																							



### Komentarz zbiorczy dla eksperymentów na zbiorze niezbilansowanym:

Po dokładniejszej analizie innych metryk, okazało się, że pomimo dość wysokiego *accuracy* model w rzeczywistości działa niezbyt dobrze. Zdecydowanej większości opinii przypisano klasę *POSITIVE*, która była klasą dominującą. Modele szczególnie słabo radzą sobie dla klasy *NEUTRAL*. Może to wynikać z tego, że opinie z trzema gwiazdkami (czyli te oznaczone jako neutral) nie zawsze są w rzeczywistości neutralne. Przykładowe opinie oznaczone jako neutral:

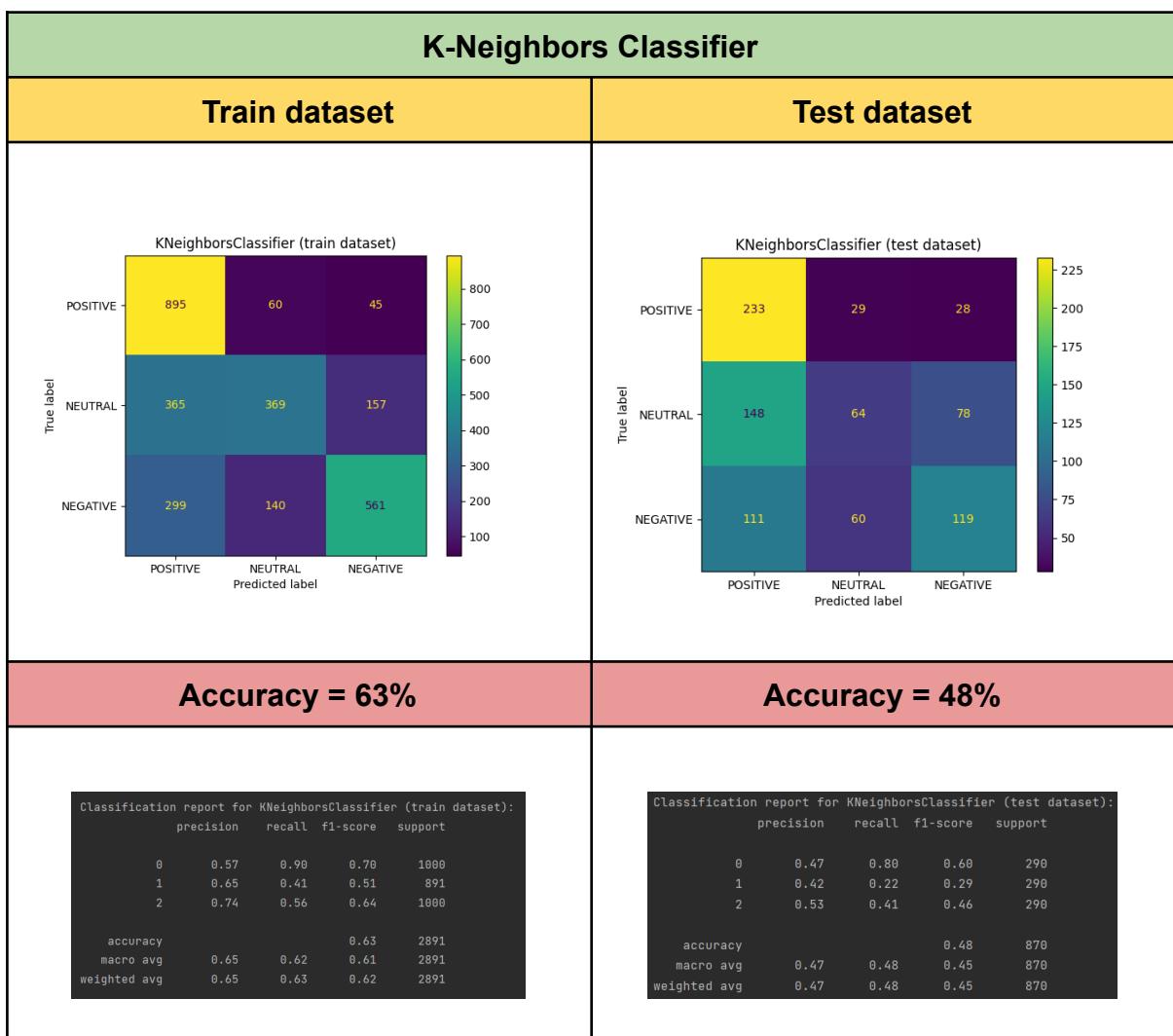
- **Das Produkt ist gut in der Wärmeleistung** (*Produkt dobrze trzyma ciepło*) - klasyfikowana jako *POSITIVE*, ze względu na słowo "gut", znajdujące się na szczytce listy słów ze zbioru opinii pozytywnych.
- **An sich ist das Produkt top. Nur leider war diesmal die Verpackung beschädigt sodass Waschpulver schon aus dem Karton gerieselt kam!** (*Produkt sam w sobie jest topowy. Niestety, tym razem opakowanie było uszkodzone tak, że proszek do prania już wyszedł z pudełka!*) - klasyfikowane jako *POSITIVE*, opinia zawiera zarówno cechę pozytywną jak i negatywną, więc do prawidłowej interpretacji wymagana jest analiza kontekstu. Testowany model tego nie potrafi.
- **Der Becher hält die Hitze wirklich hervorragend, er ist absolut dicht und sieht in cranberry toll aus. Leider ist das Plastik nicht gegen Abnutzung gefeit. Nach genau zwei Jahren war ein Teil im Deckel so rund geschliffen, dass er nicht mehr zu ging. Ersatzdeckel kostet mehr als ein ganzer Becher (der im Moment bei 23€ liegt). Wirklich schade, denn an sich ist der Becher sehr gut.** (*Kubek naprawdę dobrze trzyma ciepło, jest absolutnie szczelny i świetnie wygląda w*

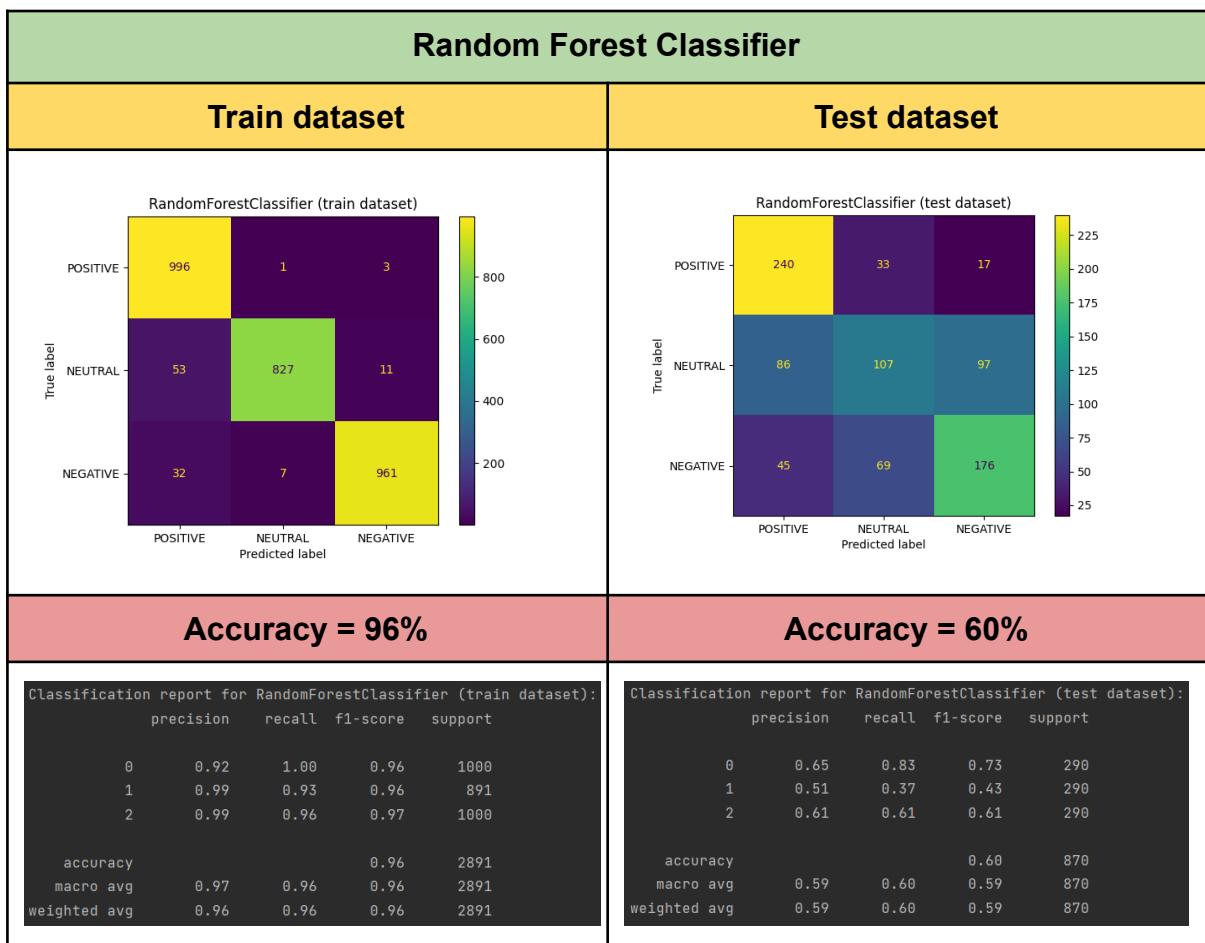
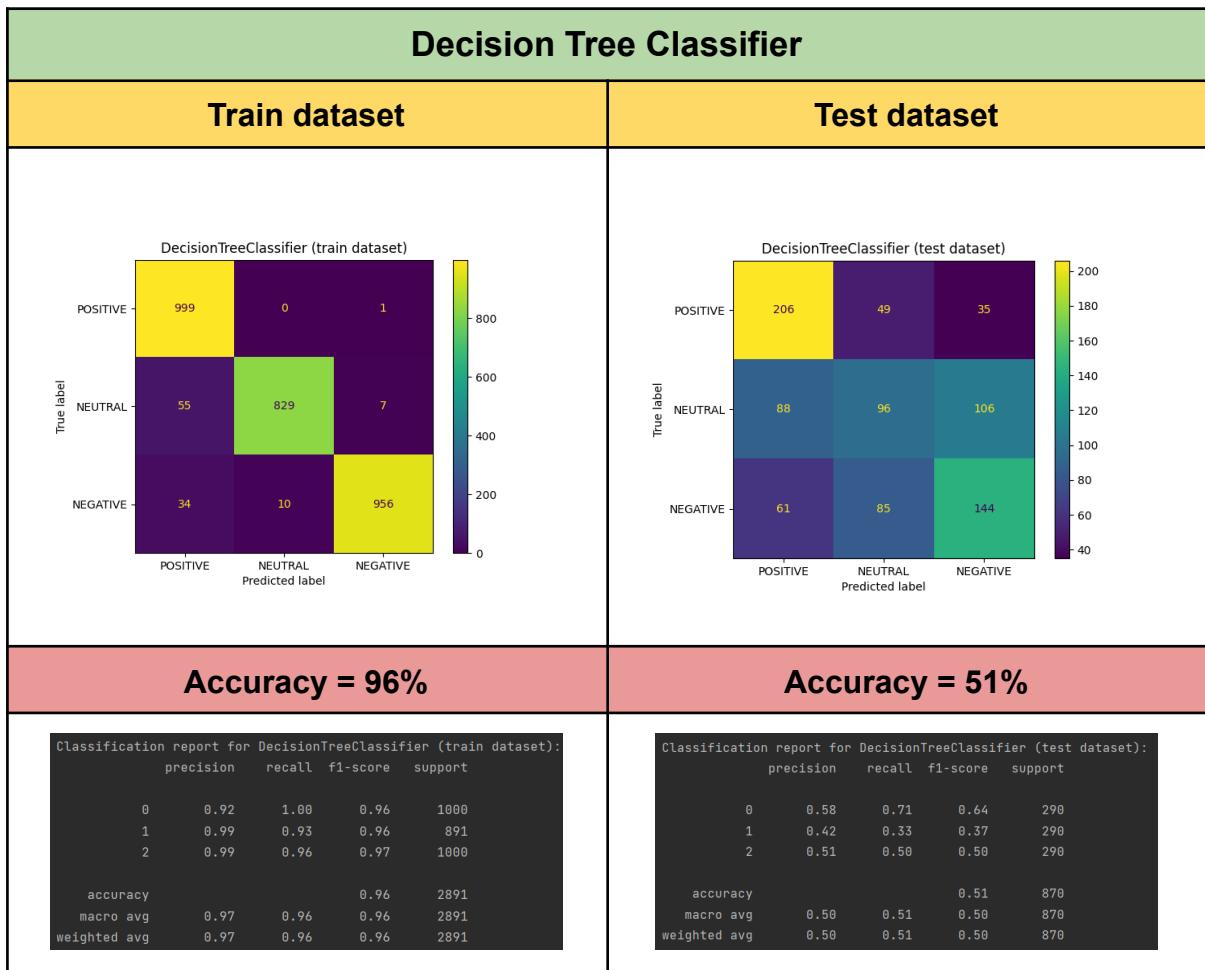
kolorze żurawinowym. Niestety plastik nie jest odporny na zużycie. Dokładnie po dwóch latach część w pokrywie była tak zaokrąglona, że już się nie zamykała. Wymienne wieczka kosztują więcej niż cały kubek (czyli obecnie 23€). Naprawdę szkoda, bo sam kubek jest bardzo dobry.) - klasyfikowane jako *POSITIVE*, sytuacja podobna, jak w poprzednim przypadku. Występuje kilka bardzo popularnych pozytywnych słów (*gut, hält, dicht, wirklich*), jedno popularne słowo neutralne (*sieht*) i dwa popularne negatywne (*leider, mehr*).

Najlepsze wyniki (patrząc na F1-Score dla poszczególnych klas) uzyskano dla klasyfikatorów:

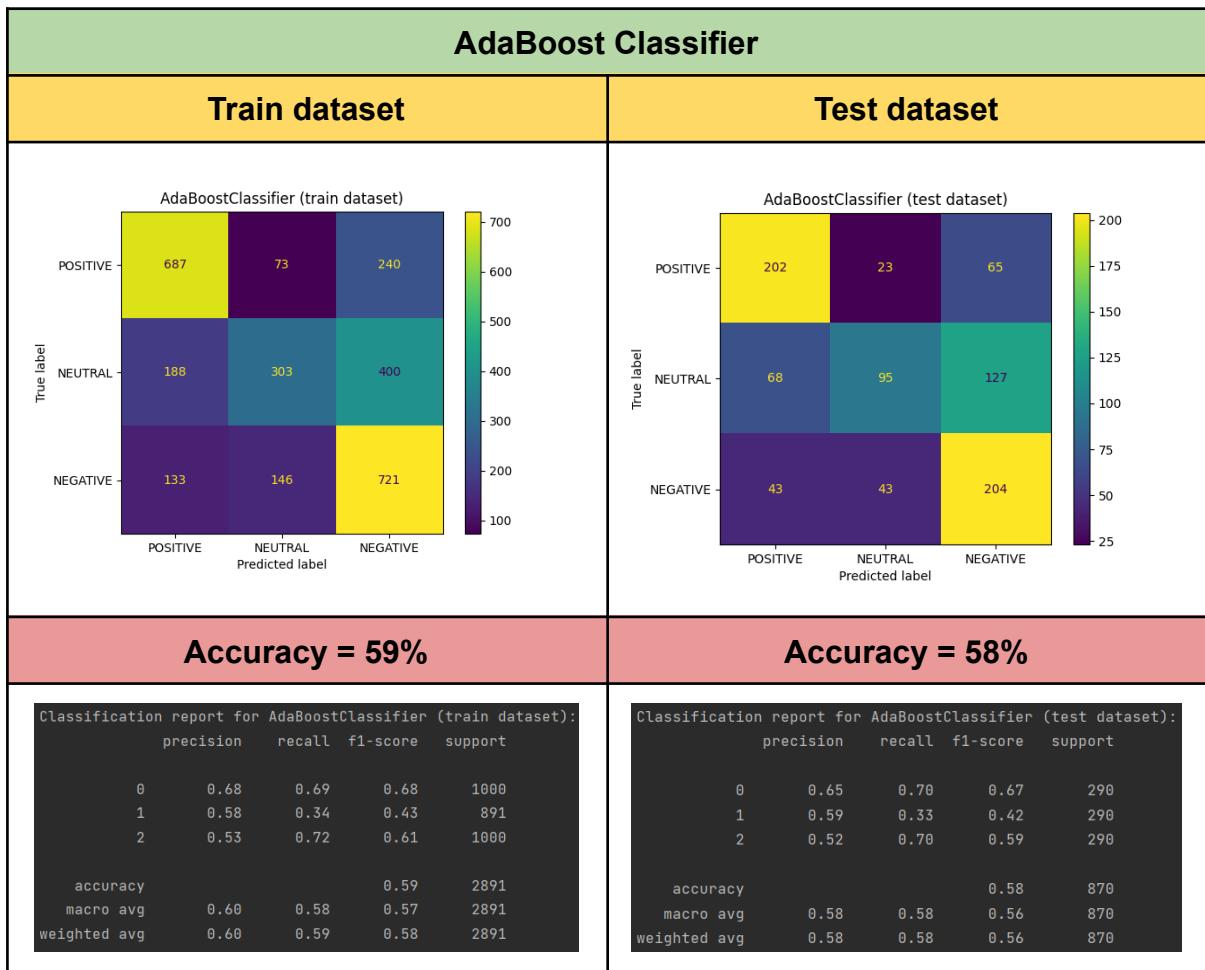
- MLP Classifier
- Logistic Regression

## 6.3 Zbiór zbilansowany









### Komentarz zbiorczy dla eksperymentów na zbiorze zbilansowanym:

Zbilansowanie zbiorów pozwoliło na uzyskanie bardziej równomiernych metryk dla każdej klasy. Ogólna dokładność spadła, jednak tym razem wynik odzwierciedla faktyczną skuteczność modelu. Najlepszy wynik osiągnięto dla klasyfikatorów *Logistic Regression* i *Random Forest*. Pierwszy z nich uzyskał lepszą ogólną dokładność, współczynnik *recall* i precyzyję. Widoczny jest również nieznacznie lepszy *f1-score* dla opinii neutralnych i takie same wartości tego współczynnika dla opinii pozytywnych i negatywnych, co w *Random Forest*. Oba klasyfikatory generują bardzo podobną macierz pomyłek, z najlepszym rozpoznaniem opinii pozytywnych i nieco gorszym dla opinii negatywnych.

W dalszym ciągu opinie neutralne są w każdym przypadku najgorzej rozpoznanie, co szczególnie widać na uzyskanych macierzach pomyłek.

## 6.4 Zbiór zbilansowany testowany na najlepszych dostrojonych klasyfikatorach

Zaplanowano wykonanie dodatkowych testów dla dostrojonych klasyfikatorów **Random Forest** oraz **Logistic Regression**. Spośród dwóch wymienionych udało się poprawić dokładność tego pierwszego, dodając poniższe ustawienia parametrów:

```
"RandomForestClassifier": RandomForestClassifier(n_estimators=1000, max_features="log2",
                                                 min_samples_leaf=1, min_samples_split=8,
                                                 max_depth=500),
```

Wszystkie parametry:

```
random_forest_kwargs = {
    "n_estimators": 1000, "criterion": "gini", "max_depth": 500, "min_samples_split": 8,
    "min_samples_leaf": 1, "min_weight_fraction_leaf": 0.0, "max_features": "log2",
    "max_leaf_nodes": None, "min_impurity_decrease": 0.0, "bootstrap": True, "oob_score": False,
    "n_jobs": None, "random_state": None, "verbose": 0, "warm_start": False, "class_weight": None,
    "ccp_alpha": 0.0, "max_samples": None
}
```

W przypadku *Logistic Regression* niezależnie od wartości i liczby zmienionych parametrów (*solver*, *penalty*, *C*) otrzymane wyniki są bardzo zbliżone do tych dla domyślnych parametrów, jeśli nie znacząco gorsze.

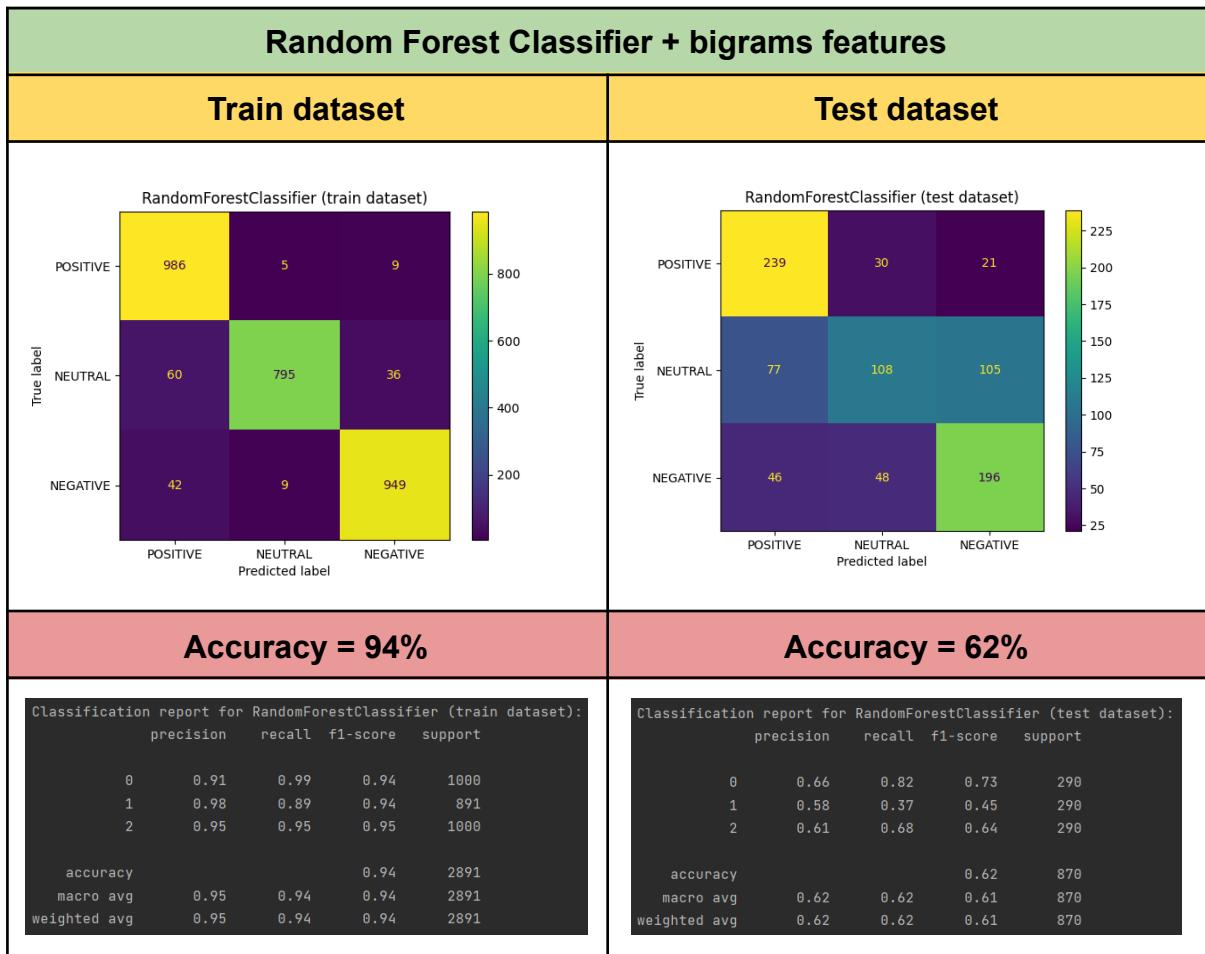
Poniżej rezultat dostrojenia hiperparametrów *Random Forest*, skutkującego wzrostem dokładności o 2% dla zbioru walidacyjnego:



Otrzymano stabilny (tzn. lepszy przy każdym uruchomieniu) wzrost dokładności algorytmów, jednak nieduży. Być może lepsze wyniki można osiągnąć dla lepszej metody wyznaczania cech (np. uwzględnienie bigramów) bądź dla zbioru danych lepszej jakości niż opinie z serwisu Amazon.

## 6.5 Zbiór zbilansowany z uwzględnieniem bigramów

Wykonano dodatkową klasyfikację z uwzględnieniem dodatkowych 30 cech. Na te dodatkowe cechy składają się 3 listy 10 najczęściej występujących **bigramów**. Jeżeli dany bigram występuje w opinii, jest to odnotowywane poprzez inkrementację licznika dla danego bigramu w ramach wektora cech. Niestety dodanie bigramów tylko nieznacznie pogarsza rezultaty wszystkich klasyfikatorów. Poniżej przedstawiono wynik dla dostrojonego **Random Forest**:



## 7. Obserwacje i wnioski

### 7.1 Najczęstsze słowa

Pozytywne 😊	Neutralne 😐	Negatywne 😞
<b>gut</b> (dobry)	<b>beim</b> (przy)	<b>leider</b> (niestety)
<b>hält</b> (trzyma [np. ciepło])	<b>allerdings</b> (wprawdzie)	<b>mehr</b> (więcej [np. oczekiwano])
<b>warm</b> (ciepły)	<b>ganz</b> (całkiem)	<b>mal</b> ([np. kilka] razy)
<b>lange</b> (długo [np. trzyma ciepło])	<b>sieht</b> (wygląda)	<b>undicht</b> (nieszczelny)
<b>immer</b> (zawsze)	<b>jedoch</b> (jednakże)	<b>gekauft</b> (zakupiono)
<b>heiß</b> (gorący)	<b>finde</b> (uważam)	<b>läuft</b> (rozpuszcza się [np. tabletka do zmywarki])
<b>zufrieden</b> (zadowolony)	<b>schön</b> (piękny)	<b>richtig</b> (prawidłowy)
<b>wirklich</b> (naprawdę)	<b>daher</b> (dlatego)	<b>zurück</b> ([np. odesłany] z powrotem)
<b>einfach</b> (prosty [np. w obsłudze])	<b>besser</b> ([np. mógłby być] lepszy)	<b>ab</b> (odtąd)
<b>dicht</b> (szczelny)	<b>hätte</b> ([np. gdyby kubek] ... miał)	<b>wurde</b> (czasownik używany w stronie biernej)

**LEGENDA** (poprawność kategoryzacji wydźwięku słowa w opinii autorów dokumentacji):

poprawnie skategoryzowany pozytywny
poprawnie skategoryzowany neutralny
poprawnie skategoryzowany negatywny
nie pasuje do kategorii

## 7.2 Najczęstsze bigramy

Pozytywne 😊	Neutralne 😐	Negatywne 😞
<b><i>lange warm</i></b> (długo [trzyma] ciepło)	<b><i>gut Hand</i></b> (dobrze [leży] w ręce)	<b><i>Becher undicht</i></b> (kubek [nie jest] szczelny)
<b><i>lange heiß</i></b> (długo [trzyma] gorąco)	<b><i>Becher gut</i></b> (dobry kubek)	<b><i>Leider Becher</i></b> (niestety kubek [...])
<b><i>hält lange</i></b> (długo trzyma [np. ciepło])	<b><i>ganz dicht</i></b> (całkiem szczelny)	<b><i>Becher leider</i></b> (kubek niestety [...])
<b><i>Becher hält</i></b> (kubek trzyma [...])	<b><i>Becher sieht</i></b> (kubek wygląda [...])	<b><i>leider undicht</i></b> (niestety [nie jest] szczelny)
<b><i>Travel Mug</i></b> (kubek na podróż)	<b><i>sieht gut</i></b> (wygląda dobrze)	<b><i>leider mehr</i></b> (niestety więcej)
<b><i>leicht reinigen</i></b> (łatwo się czyści)	<b><i>ganz gut</i></b> (całkiem dobry)	<b><i>Tabs lösen</i></b> (tabletki rozpuszczają się)
<b><i>hält wirklich</i></b> (naprawdę trzyma)	<b><i>kurzer Zeit</i></b> (krótki czas)	<b><i>zurück geschickt</i></b> (odesłany z powrotem)
<b><i>absolut dicht</i></b> (całkowicie szczelny)	<b><i>mehr erwartet</i></b> (oczekiwano więcej)	<b><i>beim Trinken</i></b> (podczas picia)
<b><i>Stunden warm</i></b> (godzinami [trzyma] ciepło)	<b><i>liegt gut</i></b> (dobrze leży [np. w ręce])	<b><i>richtig sauber</i></b> (prawidłowo oczyszczone)
<b><i>Hält lange</i></b> (długo trzyma [np. ciepło])	<b><i>Becher super</i></b> (super kubek)	<b><i>mehr dicht</i></b> (bardziej szczelny)

**LEGENDA** (poprawność kategoryzacji wydźwięku bigramu w opinii autorów dokumentacji):

poprawnie skategoryzowany pozytywny
poprawnie skategoryzowany neutralny
poprawnie skategoryzowany negatywny
nie pasuje do kategorii

## 8. Dotrenowanie modelu BERT

Model **bert-base-german-uncased** [7] został udostępniony przez organizację *HuggingFace*. Jest to ogólny model języka niemieckiego trenowany na danych ze stron:

- *wikipedia*,
- *OpenLegalData*

Do uruchomienia modelu wykorzystano bibliotekę *transformers*. Każdy trening wykonywany był przy maksymalnym możliwym rozmiarze mini-pakietu na jaki pozwalał sprzęt równym 16. Z pierwszego uruchomienia modelu na całych zbiorze danych wynikło, że optymalną liczbą epok treningu wydaje się być 2. Pokrywa się to z zaleceniami autorów (sugerują oni 2-4 epoki). Po drugiej epoce strata na danych walidacyjnych zaczyna rosnąć, co jest typową oznaką przetrenowania.

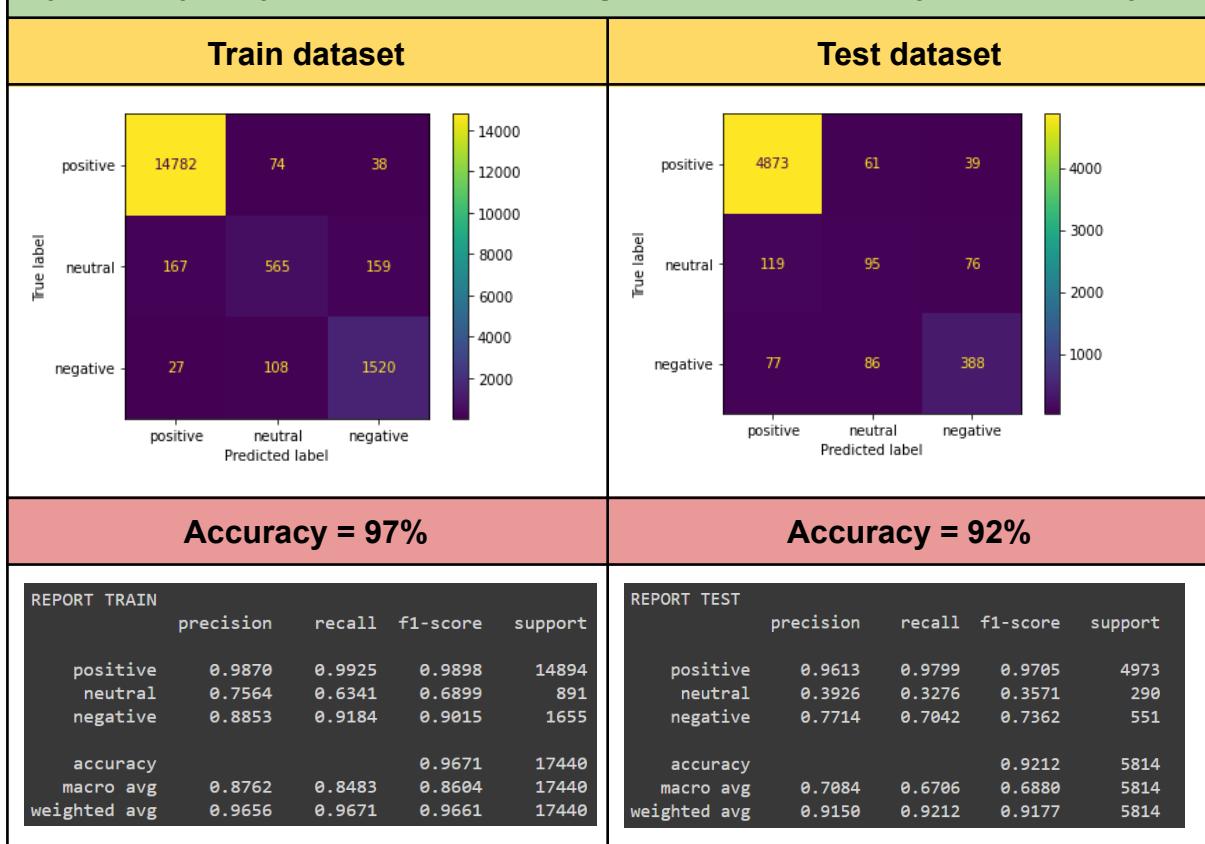
Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.271100	0.256272	0.917097	0.699199	0.591436	0.577334
2	0.211800	0.244151	0.911249	0.684845	0.680741	0.681350
3	0.136100	0.302927	0.920193	0.702524	0.676597	0.688423
4	0.081800	0.385838	0.913141	0.688968	0.676112	0.680937

Statystyki treningu z domyślnymi parametrami

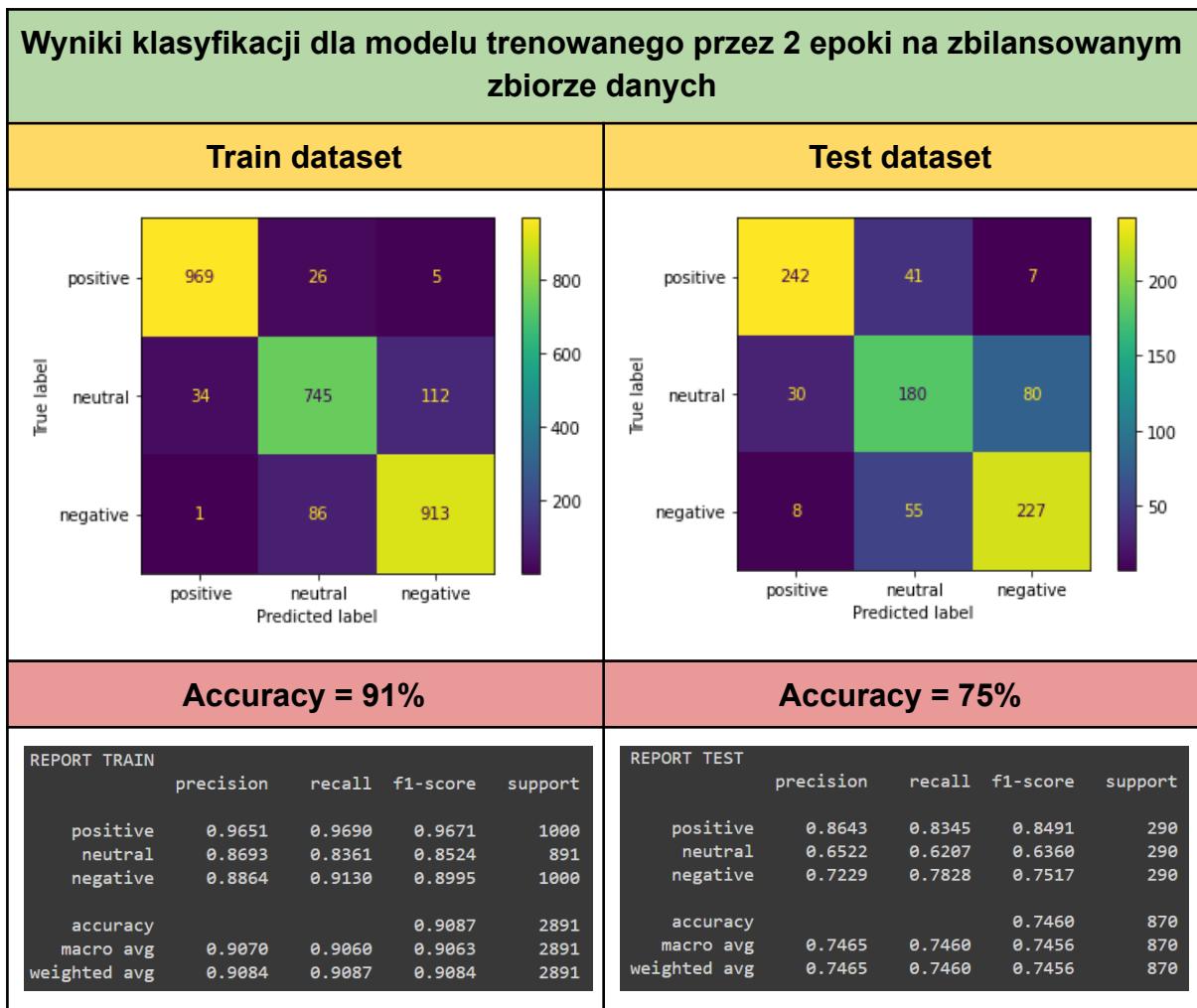
### Wyniki klasyfikacji modelu trenowanego przez 4 epoki na całym zbiorze danych



### Wyniki klasyfikacji dla modelu trenowanego przez 2 epoki na całym zbiorze danych



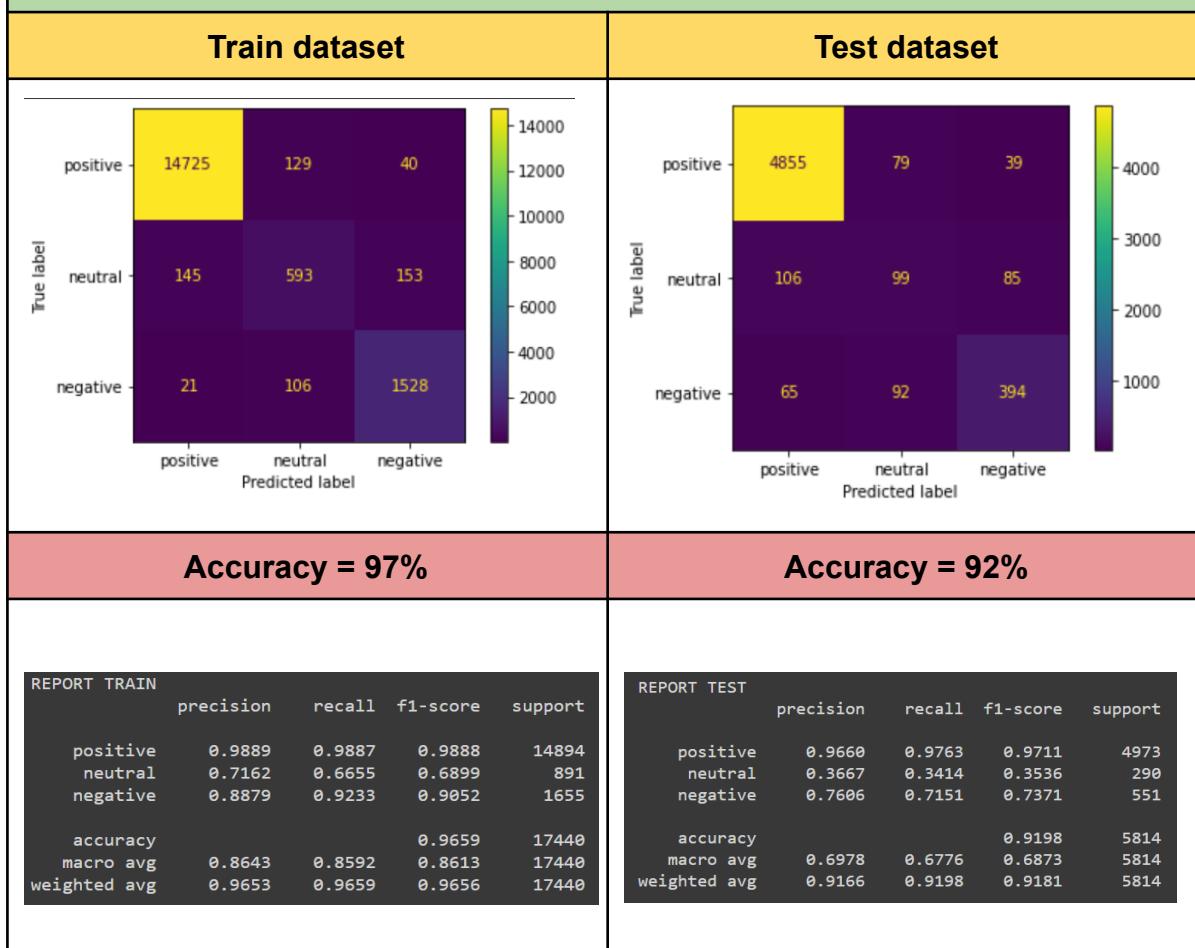
Pomimo niższej wartości funkcji straty przy trenowaniu przez 2 epoki wyniki nie poprawiły się.



Z przedstawionych powyżej danych wynika, że model dobrze radzi sobie z opiniami pozytywnymi, gorzej z negatywnymi, a najgorzej z neutralnymi. Aby poprawić wyniki wytrenowano model z inną niż domyślna funkcją straty. Zastosowano ważoną entropię krzyżową z większymi wagami dla opinii neutralnych i negatywnych:

- **waga opinii pozytywnych - 1,0**
- **waga opinii neutralnych - 5,0**
- **waga opinii negatywnych - 5,0**

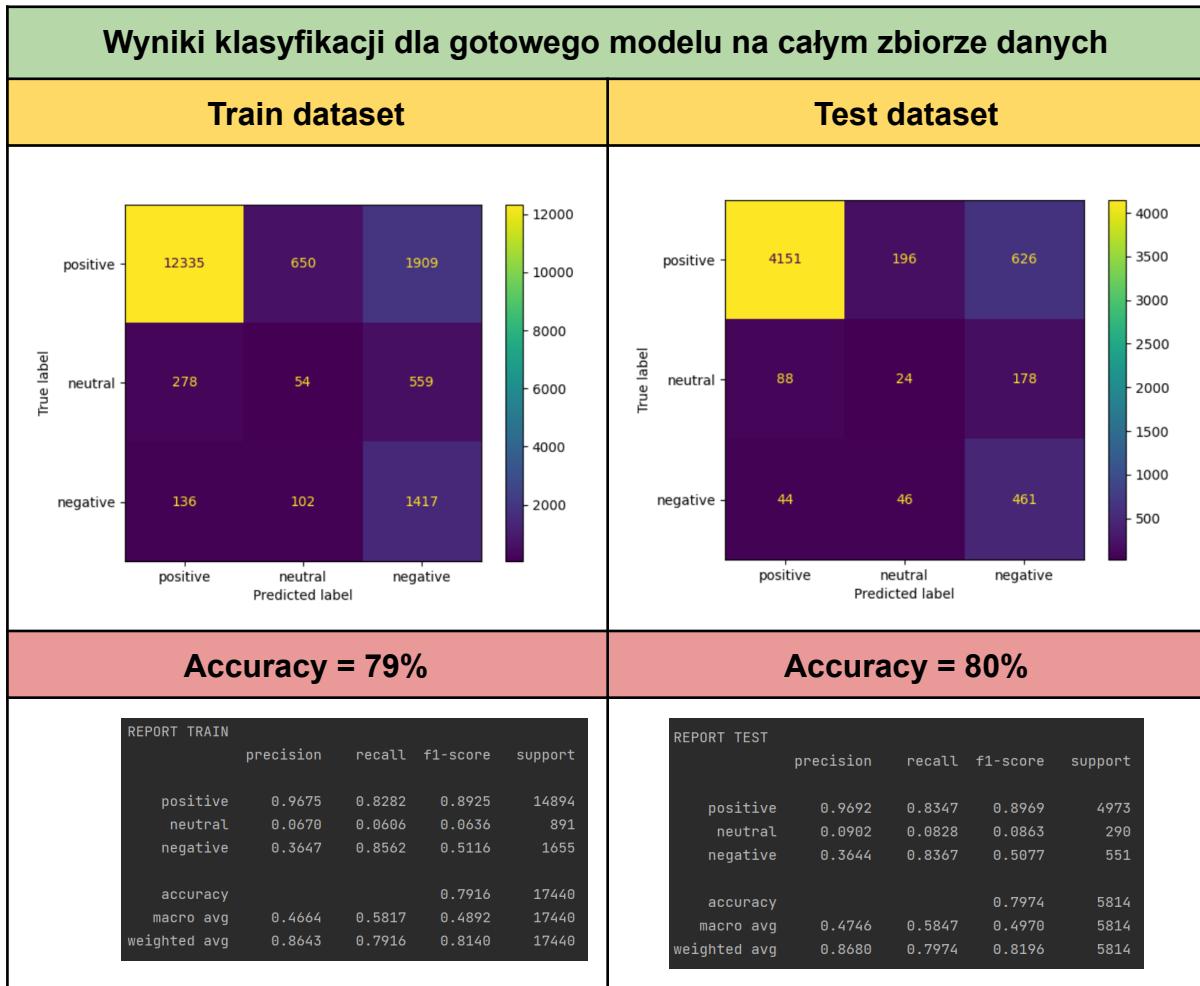
## Wyniki klasyfikacji dla modelu trenowanego przez 2 epoki na całym zbiorze danych, ze zmodyfikowaną funkcją straty



Modyfikacja funkcji straty niestety nie poprawiła wyników.

## 9. Porównanie wyników z innym modelem

Do porównania wyników klasyfikacji wykorzystano model **german-sentiment** opisany w pracy [6]. Został on wytrenowany na ponad 5 000 000 opinii pochodzących ze stron z recenzjami między innymi filmów i hoteli.



Wyniki są zauważalnie gorsze, zwłaszcza dla klasy opinii neutralnych. Biorąc pod uwagę, fakt, że model **german-sentiment** był trenowany na zdecydowanie większym i bardziej reprezentatywnym zbiorze danych, wskazuje to na przetrenowanie naszych modeli spowodowane zbyt mało reprezentatywnym zbiorem danych lub niską jakością danych (gwiazdki nie odpowiadają w rzeczywistości wydzielkowi opinii).

## 10. Wnioski

Modele bazujące na *bag of words* osiągnęły widocznie słabsze wyniki niż model oparty o sieć transformer. Powodem tego było to, że model *bag of words* nie bierze pod uwagę kontekstu, pozycji słowa w zdaniu i zawsze ma ograniczony rozmiar słownika. Przykładowo dla dwóch opinii:

- **The topic of this move is love** - wydźwięk neutralny,
- **I love a movie about this topic** - wydźwięk pozytywny,

otrzymamy ten sam wektor cech, przy różnym wydźwięku opinii. Z kolei dla opinii:

- **Ich mag diese Becher** - wydźwięk pozytywny,
- **Ich mag diese Becher nicht** - wydźwięk negatywny,

po ekstrakcji cech również otrzymujemy ten sam wektor cech, ponieważ słowo *nicht* jest wycinane przez filtrowania za pomocą stoplisty. Obie recenzje klasyfikowane są jako pozytywne. W analizie języka naturalnego kluczowa jest analiza kontekstu, a nie pojedynczych słów bez ich pozycji w zdaniu.

Trening modelu opartego o sieć transformer wymusza branie pod uwagę kontekstu. Taki model koduje też pozycję słów w zdaniu. Dodatkowo BERT był wstępnie trenowany na bardzo dużym zbiorze danych, co czyni go dobrym, ogólnym modelem języka. Jest to obecnie *state-of-the-art* w analizie języka naturalnego, podczas gdy *bag of words* w porównaniu do sieci transformer, to metoda przestarzała. Nie może, więc dziwić różnica wyników.

## 11. Literatura

[1] Harika Bonthu. 2022. [Rule-Based Sentiment Analysis in Python](#)

[2] Sheel Saket. 2020. [Count Vectorizer vs TFIDF Vectorizer | Natural Language Processing](#)

[3] Suvrat Arora. 2022. [Sentiment Analysis Using Python](#)

[4] Marius Mogyorosi. [Sentiment Analysis: First Steps With Python's NLTK Library](#)

[5] TechVidvan. [Sentiment Analysis using Python](#)

[6] Oliver Guhr, Anne-Kathrin Schumann, Frank Bahrmann, and Hans Joachim Böhme. 2020. [Training a Broad-Coverage German Sentiment Classification Model for Dialog Systems](#). In Proceedings of the Twelfth Language Resources and Evaluation Conference, pages 1627–1632, Marseille, France. European Language Resources Association.

[7] Branden Chan, Stefan Schweter, and Timo Möller. 2020. [German’s Next Language Model](#). In Proceedings of the 28th International Conference on Computational Linguistics, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.