Implementacja algorytmu oversamplingu ADASYN

Klaudia Maciaszek - 259704 i Jakub Szuper - 259695

Politechnika Wrocławska

1 Wstęp i przegląd literatury

1.1 Wprowadzenie do tematu projektu, rys teoretyczny

ADASYN [1] - Adaptive Synthetic Sampling - Adaptacyjne Syntetyczne Próbkowanie

Klasyfikacja z niezbalansowanymi danymi może stanowić problem, ponieważ większość algorytmów używanych do klasyfikacji została zaimplementowana przy założeniu równej liczby przykładów dla każdej klasy. Rozwiązaniem tej komplikacji jest między innymi oversampling. Działanie to polega na sztucznym dodaniu danych, w celu zrównoważenia liczności próbek obu klas.

ADASYN wykorzystuje w swoim działaniu oversampling - idea tej metody polega na generowaniu sztucznych próbek z klasy o mniejszej liczności na podstawie stopnia niezbalansowania, ilości sąsiadów z klasy większej oraz wagi próbki. Algorytm ADASYN poprawia uczenie się, odnosząc się do rozkładu danych, zmniejszając błąd wynikający z nierównowagi klas i adaptując się do danych warunków.

Celem projektu jest implementacja algorytmu oversamplingu ADASYN oraz wskazanie i porównanie z innymi metodami referencyjnymi.

1.2 Przegląd literatury

Istnieją także inne metody wykorzystujące oversampling [2].

Jedną z nich jest SMOTE - Synthetic Minority Oversampling Technique. Algorytm ten polega na [3] tworzeniu nowych próbek między istniejącymi na podstawie ich gęstości, które były wprowadzone jako dane wejściowe.

Alernatywną metodą jest ASMOTE [6] - Adaptive Synthetic Minority Oversampling Technique. Ta technika działa z dodatkową adaptacyjnością do rozmiaru mniejszej klasy, co może znacznie poprawiać wydajność modeli, dostarczając syntetyczny zestaw danych do szkolenia.

Kolejną metodą jest BorderlineSMOTE [4]. Opiera się ona na metodzie SMOTE. Wyróżnia się tym, że generowanie próbek odbywa się przy granicach decyzyjnych. W odróżnieniu od SMOTE, BorderlineSMOTE przepróbkowuje lub umacnia granicę wraz z punktami klasy mniejszościowej.

Następną metodą jest SMOTEBoost [5]. Algorytm ten łączy się z algorytmem SMOTE na zasadzie generowania próbek z klasy o mniejszej liczebności oraz procesu klasyfikacji opartego na algorytmie boostingowym. Tworzy on syntetyczne próbki, zmieniając aktualizowane wagi oraz wyrównując nieprawidłowe rozkłady.

Z kolei algorytm RamOBoost [2] generuje nowe próbki na podstawie wag, określających trudność w klasyfikacji danego przykładu. Wagi próbek z klasy mniejszości zostają dostosowane zgodnie z ich rozkładem.

Metoda MAHAKIL [7] wykorzystuje pojęcie odległości Mahalanobisa [8], jako miara podobieństwa między dwoma obiektami. Dodatkowo algorytm ten może łączyć się z grupowaniem K-means, które polega na dzieleniu danych wejściowych na określoną liczbę klas.

W projekcie metodą, do której porównywane będą wyniki, będzie zaimportowany ADASYN, SMOTE oraz BorderlineSMOTE.

2 Metoda

Jednym z głównych etapów projektu było zaimplementowanie metody Adasyn, sugerując się przy tym istniejącymi już implementacjami i funkcjami. W języku Python stworzono klasę kompatybilną z danym stosem technologicznym korzystając z BaseEstimator z biblioteki scikit-learn. Wykorzystano wygenerowane dane syntetyczne, w celu wykonania danej implementacji, na którą składały się takie etapy jak znalezienie i stworzenie listy klasy mniejszościowej oraz użycie algorytmu NearestNeighbors, wykorzystując przy tym odpowiednie biblioteki.

Implementacja wstępnego eksperymentu opierała się na uruchomieniu zaimplementowanej metody zgodnie z odpowiednim protokołem eksperymentalnym. Wstępnymi wynikami było wypisanie w konsoli liczbę klas mniejszościowej i większościowej oraz Accuracy dla danej metody.

3 Projekt eksperymentów

Celem realizowanych eksperymentów jest porównywanie jakości klasyfikacji zbiorów niezbalansowanych dla wybranych metod.

Do wykonania eksperymentów zostanie użyty język Python 3.10.11, korzystając z IDE PyCharm Community Edition 2022.3.2. Używanymi bibliotekami

będą: scikit-learn, imbalanced-learn oraz NumPy.

Na początku zostanie zaimplementowany algorytm ADASYN, a następnie zostaną zaimportowane algorytmy ADASYN, SMOTE oraz BorderlineSMOTE. Będą wykonane dwa eksperymenty na danych syntetycznych oraz danych rzeczywistych. Kolejno zostaną obliczone wartości metryk oraz zostanie wykonany test statystyczny.

Dodatkowo wykonany zostanie sprawdzian krzyżowy k-foldowy z liczbą foldów k=5 (wariant walidacji krzyżowej), który jest standardową procedurą oceny wydajności. Polega on na podzieleniu zbioru danych na zbiór uczący oraz testowy k-razy.

3.1 Metryki oceny wyników

Do oceny wydajności algorytmu zostanie wykorzystane kilka wskaźników oceny [9], takich jak Accuracy, Precision, F1 oraz Recall z biblioteki scikit-learn.

3.2 Testy statystyczne

Do przetestowania normalności rozkładu zostanie wykorzystany test t-Studenta jako funkcja ttest_ind() z biblioteki SciPy. Oblicza on test dla średnich wyników z dwóch niezależnych prób wyników. Używa się go podczas obserwacji dwóch niezależnych prób z tej samej lub różnych populacji.

3.3 Sposób generowania danych syntetycznych

Dane syntetyczne zostaną wygenerowane za pomocą funkcji make_classification z biblioteki scikit.learn. Zostanie wygenerowane 200 próbek. Liczba informatywnych atrybutów będzie wynosiła domyślnie - 2. Tak samo pozostałe parametry są ustawione domyślnie. Parametr 'weights' określa proporcje próbek przypisane do każdej klasy. Aby stworzyć niezbalansowany zbiór danych, do tego parametru zostanie przypisane: weights=[0.1, 0.9].

3.4 Sposób pozyskania zbioru danych rzeczywistych

Dane rzeczywiste zostały przedstawione na stronie KEEL - Knowledge Extraction based on Evolutionary Learning [10] oraz udostępnione do pobrania na stronie Kaggle [11]. Został wybrany dataset związany z identyfikacją szkła. Liczba wszystkich przypadków tego zbioru wynosi 214, Jest to zbiór niezbalansowany: zawiera 13,55% pozytywnych przypadków oraz 86,45% negatywnych przypadków. Atrybutami są pierwiastki chemiczne.

4 Wyniki eksperymentów

Rozdział zawiera wyniki oraz ich analizę na podstawie wykresów i tabel.

4.1 Implementacja oraz przeprowadzenie eksperymentów

Po zaimplementowaniu metody Adasyn użyto zaimportowanych metod takich jak SMOTE i BorderlineSMOTE oraz gotową metodę ADASYN.

Wszystkie eksperymenty przeprowadzono na danych syntetycznych oraz danych rzeczywistych. Otrzymano wyniki dla każdych z wymienionych metryk (Accuracy, Precision, F1, Recall). Zapewniono powtarzalność wyników, co zrealizowano przy użyciu mechanizmów pseudolosowości opartych na zadanym ziarnie losowości. W przypadku danych eksperymentów użyto random_state, który został wykorzystany przy walidacji krzyżowej k-foldowej (StratifiedKFold z biblioteki scikit-learn) oraz przy make_classification. Otrzymano wyniki dla każdej metody i dla każdego kolejnego folda.

4.2 Implementacja kodu analitycznego

Kolejnym krokiem była prezentacja jakości według metryk, co przełożyło się na obliczenie dla każdego folda każdej metody wartości średniej oraz odchylenia standardowego.

Należało także przeprowadzić testy statystyczne, gdzie w przypadku projektu wykorzystano test t-Studenta (skorzystano z funkcji ttest_ind() z biblioteki SciPy).

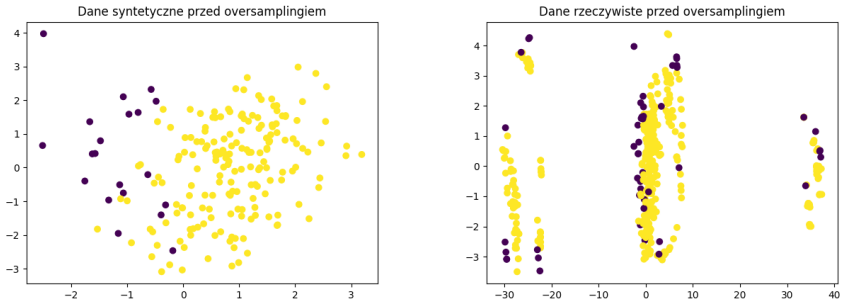
Końcowo wygenerowano wykresy, potrzebne do wykonania analizy dla każdej z metod.

4.3 Analiza wyników

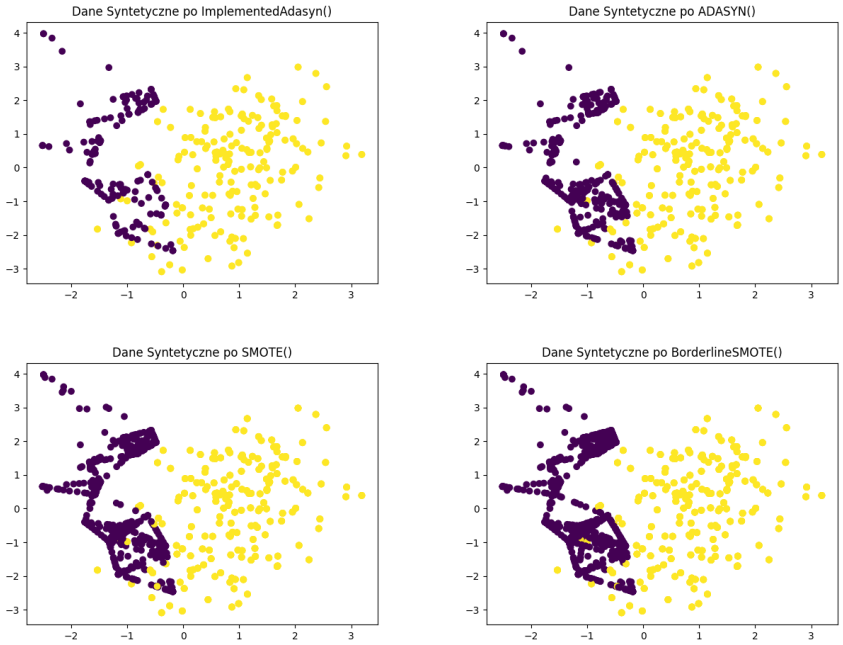
Na rysunku 1 przedstawiono, jak wyglądały dane syntetyczne i rzeczywiste przed zastosowaniem algorytmów oversamplingu. Wyraźnie widać dysproporcję w liczności klasy mniejszościowej (kolor fioletowy) w stosunku do większościowej (kolor żółty).

Wszystkie dane rzeczywiste prezentowane w dokumencie zostały zwizualizowane za pomocą metody t-SNE (redukcja wymiarowości). Pozwala ona na zobrazowanie, w jaki sposób wszystkie dane rozmieszczone są w przestrzeni. Dodatkowo jeden z parametrów w danych rzeczywistych - Rl przyjmował znacznie większe wartości niż pozostałe. Dla uzyskania odpowiednich wyników zlogarytmowano wszystkie wartości w ten kolumnie.

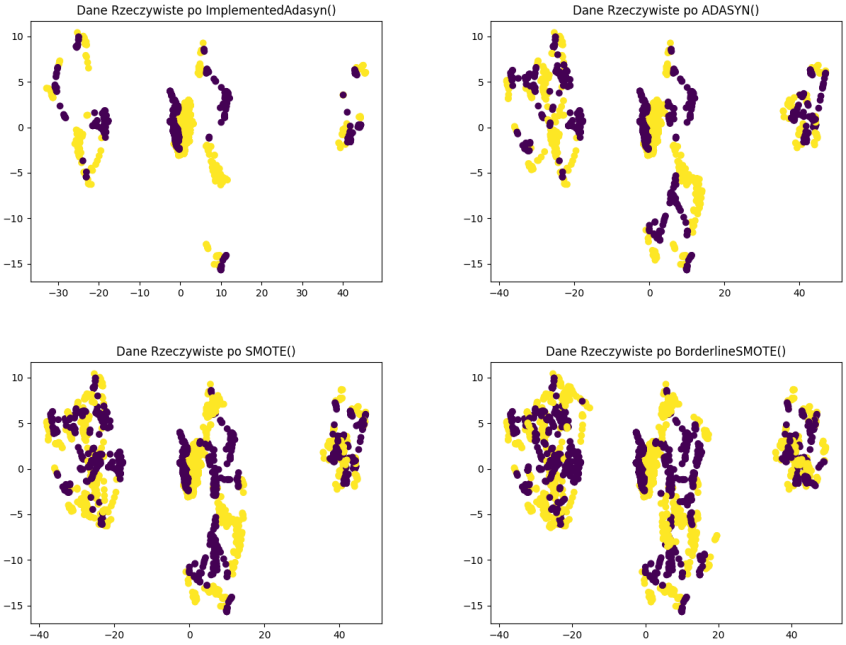
Na kolejnym rysunku (rysunek 2) widać jak zastosowanie algorytmów wpłynęło na liczność klasy mniejszościowej w przypadku danych syntetycznych, natomiast na rysunku 3 dla danych rzeczywistych.



Rysunek 1. Dane przed oversamplingiem



Rysunek 2. Dane syntetyczne po różnych algorytmach oversamplingu



Rysunek 3. Dane rzeczywiste po różnych algorytmach oversamplingu

ImplementedAdasyn()	Dokladnosc	Precyzja	F1	Recall
Fold 1	0.8605	0.9429	0.9167	0.8919
Fold 2	0.8140	0.9143	0.8889	0.8649
Fold 3	0.8605	0.8974	0.9211	0.9459
Fold 4	0.8605	0.9429	0.9167	0.8919
Fold 5	0.7857	0.9667	0.8657	0.7838

Tabela 1. Metryki dla każdego folda zaimplementowanego algorytmu Adasyn

Tabela 1 przedstawia wyniki metryk dla każdego z foldów dla zaimplementowanego algorytmu Adasyn. Tabele 2 oraz 3 przedstawiają uśrednione wyniki z odchyleniem standardowym z 5 foldów dla każdej z metod i metryk dla danych syntetycznych, oraz rzeczywistych.

Metody/Metryki	Dokladnosc	Precyzja	F1	Recall
Zaimplementowany Adasyn	0.8950 ± 0.0187	0.9720 ± 0.0296	0.9398 ± 0.0107	0.9111 ± 0.0272
Zaimportowany Adasyn	0.8850 ± 0.0325	0.9820 ± 0.0147	0.9328 ± 0.0124	0.8889 ± 0.0248
BorderlineSMOTE	0.9100 ± 0.0255	0.9720 ± 0.0296	0.9490 ± 0.0137	0.9278 ± 0.0136
SMOTE	0.8950 ± 0.0187	0.9768 ± 0.0215	0.9394 ± 0.0110	0.9056 ± 0.0222

Tabela 2. Uśrednione wyniki z odchyleniem standardowym z 5 foldów dla każdej z metod i metryk dla danych syntetycznych

Metody/Metryki	Dokladnosc	Precyzja	F1	Recall
Zaimplementowany Adasyn	0.8362 ± 0.0310	0.9328 ± 0.0243	0.9018 ± 0.0214	0.8757 ± 0.0530
Zaimportowany Adasyn	0.8456 ± 0.0325	0.9333 ± 0.0252	0.9081 ± 0.0208	0.8865 ± 0.0465
BorderlineSMOTE	0.8691 ± 0.0190	0.9349 ± 0.0253	0.9235 ± 0.0109	0.9135 ± 0.0265
SMOTE	0.8363 ± 0.0218	0.9324 ± 0.0259	0.9023 ± 0.0136	0.8757 ± 0.0324

Tabela 3. Uśrednione wyniki z odchyleniem standardowym z 5 foldów dla każdej z metod i metryk dla danych rzeczywistych

Jak widać w tabelach najwyższą wartość dokładności osiągnął BorderlineSMOTE. Zwykły SMOTE natomiast osiąga nieco niższe wyniki.

Tabele od 4 do 11 przedstawiają obliczone wartości pvalue, które uzyskano za pomocą wykonanego testu t-Studenta. Test, po kolei dla każdej z metryk oraz każdego typu danych, obliczał wartość pvalue między dwiema metodami, każda z każdą. Tak uzyskane wyniki mogą wskazać, czy odrzucić hipotezę zerową. Jeżeli pvalue jest poniżej poziomu istotności, to przyjmujemy na korzyść hipotezę alternatywną.

Przy hipotezie zerowej zakłada się, że jeżeli mamy dwie populacje, to ich średnie są takie same (nie ma istotnych różnic statystycznych). Hipoteza alternatywna zakłada, że średnia tych dwóch populacji jest wystarczająco różna od siebie (różnice są istotne).

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.4860	0.3706	1.0000
Zaimportowany Adasyn	0.4860	1.0000	0.1614	0.4860
BorderlineSMOTE	0.3706	0.1614	1.0000	0.3706
SMOTE	1.0000	0.4860	0.3706	1.0000

Tabela 4. Wyniki pvalue dla accuracy_score przy wykorzystaniu testu t-Studenta dla danych syntetycznych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.5636	1.0000	0.8002
Zaimportowany Adasyn	0.5636	1.0000	0.5636	0.7012
BorderlineSMOTE	1.0000	0.5636	1.0000	0.8002
SMOTE	0.8002	0.7012	0.8002	1.0000

Tabela 5. Wyniki pvalue dla precision_score przy wykorzystaniu testu t-Studenta dla danych syntetycznych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.4171	0.3210	0.9613
Zaimportowany Adasyn	0.4171	1.0000	0.1174	0.4478
BorderlineSMOTE	0.3210	0.1174	1.0000	0.3076
SMOTE	0.9613	0.4478	0.3076	1.0000

Tabela 6. Wyniki pvalue dla f1_score przy wykorzystaniu testu t-Studenta dla danych syntetycznych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.2623	0.3052	0.7599
Zaimportowany Adasyn	0.2623	1.0000	0.0252	0.3466
BorderlineSMOTE	0.3052	0.0252	1.0000	0.1265
SMOTE	0.7599	0.3466	0.1265	1.0000

Tabela 7. Wyniki pvalue dla recall_score przy wykorzystaniu testu t-Studenta dla danych syntetycznych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.6862	0.1081	0.9955
Zaimportowany Adasyn	0.6862	1.0000	0.2474	0.6474
BorderlineSMOTE	0.1081	0.2474	1.0000	0.0532
SMOTE	0.9955	0.6474	0.0532	1.0000

Tabela 8. Wyniki pvalue dla accuracy_score przy wykorzystaniu testu t-Studenta dla danych rzeczywistych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.9808	0.9095	0.9818
Zaimportowany Adasyn	0.9808	1.0000	0.9298	0.9636
BorderlineSMOTE	0.9095	0.9298	1.0000	0.8948
SMOTE	0.9818	0.9636	0.8948	1.0000

Tabela 9. Wyniki pvalue dla precision_score przy wykorzystaniu testu t-Studenta dla danych rzeczywistych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.6833	0.1085	0.9665
Zaimportowany Adasyn	0.6833	1.0000	0.2266	0.6550
BorderlineSMOTE	0.1085	0.2266	1.0000	0.0414
SMOTE	0.9665	0.6550	0.0414	1.0000

Tabela 10. Wyniki pvalue dla f1_score przy wykorzystaniu testu t-Studenta dla danych rzeczywistych

Metody	Zaimplementowany Adasyn	Zaimportowany Adasyn	BorderlineSMOTE	SMOTE
Zaimplementowany Adasyn	1.0000	0.7668	0.2371	1.0000
Zaimportowany Adasyn	0.7668	1.0000	0.3420	0.7128
BorderlineSMOTE	0.2371	0.3420	1.0000	0.1083
SMOTE	1.0000	0.7128	0.1083	1.0000

Tabela 11. Wyniki pvalue dla recall_score przy wykorzystaniu testu t-Studenta dla danych rzeczywistych

5 Wnioski

Na rysunkach 2 i 3 widać, że wygenerowane datasety znacząco nie różnią się od siebie. W przypadku zaimplementowanego Adasyna widocznych próbek jest mniej niż w przypadku pozostałych algorytmów. Najbardziej widać to, porównując zaimplementowanego Adasyna z BorderlineSMOTE. Nieduże różnice w wyglądzie wykresów przekładają się na podobne wyniki metryk.

W tabelach dotyczących testu t-Studenta można zauważyć, że wyniki, w większości przypadków przekraczają poziom istotności równy 0,05. Sytuacje, w których wyniki są niższe od tego poziomu to przy danych syntetycznych wartość pvalue dla recall_score przy porównaniu zaimportowanego ADASYNA oraz BordelineSMOTE (tabela 7) oraz w tabeli 10 przy danych rzeczywistych wartość pvalue dla f1_score przy porównaniu SMOTE i BorderlineSMOTE. Analizując tabele 2 i 3, faktycznie można zauważyć, że wartości średnich danych algorytmów dla danych metryk posiadają widoczne różnice.

Patrząc na tabele z metodami i metrykami (tabela 2 oraz 3) widać, że wartości dla metody BordelineSMOTE dla wszystkich metryk (dokładność, precyzja, f1 i recall), zarówno dla danych syntetycznych jak i rzeczywistych są nieco wyższe niż pozostałe. W metrykach f1 oraz recall można zauważyć nawet większą, bardziej znaczącą różnicę. Wynika to z tego, że BorderlineSMOTE jest bardziej zaawansowanym algorytmem - opiera się na metodzie SMOTE, ale wyróżnia się tym, że generowanie próbek odbywa się blisko przy granicach decyzyjnych. W odróżnieniu od SMOTE, BorderlineSMOTE przepróbkowuje lub umacnia granicę wraz z punktami klasy mniejszościowej.

Dodatkowo przeanalizowano, jak poradziły sobie metody z liczebnością próbek. Dla danych syntetycznych wyszło 180 próbek klasy większościowej. Zaimportowany ADASYN, SMOTE oraz BorderlineSMOTE poradziły sobie bez problemu, generując po próbkowaniu, w takiej samej ilości próbki klasy mniejszościowej. Zaimplementowany Adasyn wykazał 169 próbek, co też jest dobrym wynikiem. Podobna sytuacja była przy danych rzeczywistych.

Podsumowując, zaimplementowany Adasyn poradził sobie dobrze ze sztucznym wygenerowaniem potrzebnych danych w celu zrównoważeniu liczności próbek, na co przekładają się jego wyniki dla danych metryk oraz dla testu statystycznego.

Literatura

1. Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li, ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning, 2008
2. Miss. Mayuri S. Shelke, Dr. Prashant R. Deshmukh, Prof. Vijaya K. Shandilya, A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique, 2017

3. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, 2002
4. Hui Han1, Wen-Yuan Wang1, and Bing-Huan Mao2, Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning, 2005
5. Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, Kevin Bowyer, SMOTEBoost: Improving Prediction of the Minority Class in Boosting
6. V. Tra, B. -P. Duong and J. -M. Kim, "Improving diagnostic performance of a power transformer using an adaptive over-sampling method for imbalanced data," in IEEE Transactions on Dielectrics and Electrical Insulation, vol. 26, no. 4, pp. 1325-1333, Aug. 2019, doi: 10.1109/TDEI.2019.008034.
7. Y. Zhang, T. Zuo, L. Fang, J. Li and Z. Xing, "An Improved MAHAKIL Oversampling Method for Imbalanced Dataset Classification," in IEEE Access, vol. 9, pp. 16030-16040, 2021, doi: 10.1109/ACCESS.2020.3047741.
8. Zygmunt Kaczmarek, Stanisław Czajka, Elżbieta Adamska, Propozycja metody grupowania obiektów jednoi wielocechowych z zastosowaniem odległości Mahalanobisa i analizy skupień
9. Rand Kouatly, Ietezaz Ul Hassan, Raja Hashim Ali, Zain Ul Abideen, Talha Ali Khan, Significance of Machine Learning for Detection of Malicious Websites on an Unbalanced Dataset
10. https://sci2s.ugr.es/keel/dataset.php?cod=143&fbclid=IwAR3GC6uS1LGu9Wn2_-u7658PBNA1uD5ZbwSKPPCBYlusAU8T8qsQwMRzePY#sub1
11. https://www.kaggle.com/datasets/baguspurnama/glass-imbalanced?resource=download&fbclid=IwAR1-JgWLgFk64MhlOj-1TKF1rXldHU8AtxF0_QerPuOAMqyZhJlWpRAtDCQ