Badanie technik optymalizacji uczenia maszynowego klasyfikacji dla niezbalansowanych zbiorów danych

Jakub Krupiński

Politechnika Wrocławska

1 Wprowadzenie

1.1 Wstęp

Dzięki rozwojowi technologii, w tym np. internetu czy urządzeń IoT, z każdym rokiem możliwy jest dostęp do coraz większych ilości surowych danych, których liczba rośnie w niezwykle szybkim tempie. Stwarza to wiele możliwości związanych z inżynierią danych, ale niesie to ze sobą również wiele wyzwań. W przypadku surowych danych (zwłaszcza tych z heterogenicznych źródeł) można natknąć się na problem niezbalansowania danych, w którym liczba próbek części klas znacząco przewyższa liczbę próbek pozostałych klas.[6] Niezbalansowane dane mają tendencję do znacznego osłabiania większości standardowych algorytmów uczenia maszynowego, ponieważ zakładają one zrównoważony rozkład klas lub równych kosztów błędnej klasyfikacji - dlatego też, gdy prezentowane są one z danymi niezbalansowanymi, algorytmy te nie potrafią zapewnić odpowiedniej dokładności dla swoich predykcji[4], a standardowe metryki takie jak np. *accuracy* przestają być użyteczne[5].

1.2 Przegląd literatury

Problem niezrównoważonych zbiorów danych stał się powszechny w wielu dziedzinach. Sytuacje w których klasy mniejszościowe - często reprezentujące krytyczne informacje - prowadzi do zniekształcenia wydajności modelu, ze względu na swoją wybrakowaną reprezentację. Celem tego projektu jest zbadanie technik optymalizacji pod kątem ich skuteczności w pracy na niezrównoważonych zbiorach danych.

Wyzwania związane z danymi niezrównoważonymi obejmują:

1. Stronniczość klasyfikatorów: klasyfikatory mają tendencję do faworyzowania klasy większościowej, co prowadzi do wysokiej *dokładności*, ale niskiej *czułości* dla klas mniejszościowych.
2. Metryki oceny: tradycyjne metryki takie jak *dokładność* są mylące w kontekście danych niezrównoważonych. Zamiast tego bardziej informacyjne są metryki takie jak np. *F1-score* oraz *pole pod krzywą ROC-AUC*.

W literaturze wyróżnia się trzy główne podejścia do rozwiązania problemów związanych z niezbalansowaniem zbioru danych:

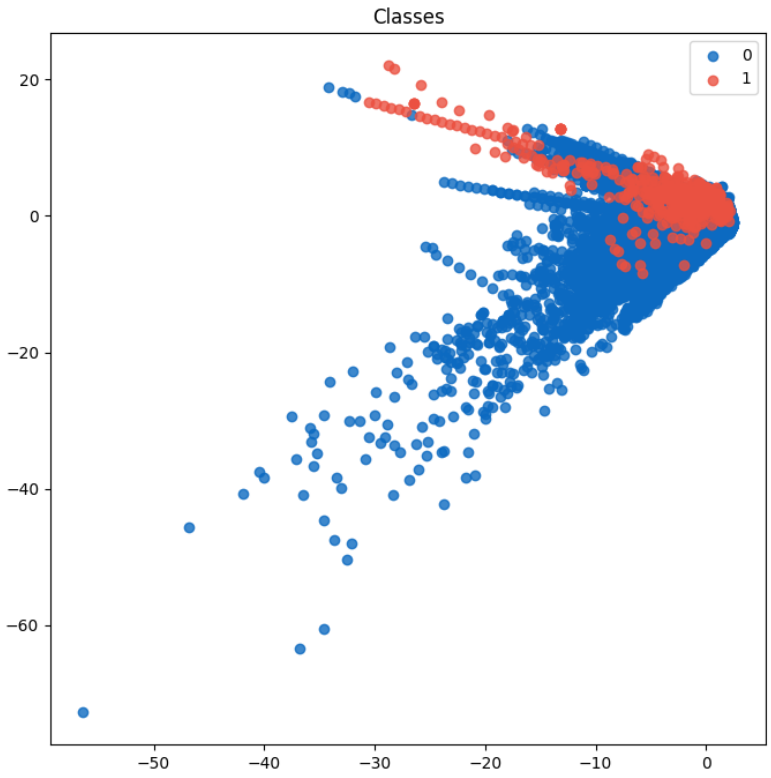
1. Metody próbkowania - zbiór metod opierających się na modyfikacji niezrównoważonego zestawu danych przez pewne mechanizmy w celu zapewnienia zrównoważonego rozkładu klas.[3]
   - Random oversampling - mechanizm operający się o dodawanie do zbioru danych początkowych zestawu losowo wybranych osobników klasy mniejszościowej. W ten sposób liczba całkowitych przykładów jest zwiększana, a równowaga dystrybucji klas odpowiednio dopasowana.
   - Random undersampling - mechanizm usuwający losowo wybrane dane klasy większościowej z oryginalnego zestawy danych, aby doprowadzić do równowagi między klasami.
   - Informed Undersampling - zbiór metod, których celem jest przezwyciężenie problemu utraty informacji pojawiającego się w klasycznym *losowym niedopróbkowaniu*. Przykładem tej metody jest wykorzystywana w ramach zaproponowanych przeze mnie eksperymentów metoda *EasyEnsamble*, który rozwija system uczenia się przez niezależne próbkowanie kilku podzbiorów z klasy większosciowej i rozwijanie wielu klasyfikatorów w oparciu o kombinację każdego podzbioru z danymi klasy mniejszościowej.
   - Próbkowanie syntetyczne z generowaniem danych - przykład tego podejścia to analizowana przeze mnie metoda *SMOTE*, która tworzy sztuczne dane w oparciu o podobieństwa przestrzeni cech między istniejącymi przykładami z klas mniejszościowych.
   - Adaptacyjne próbkowanie syntetyczne - zbiór metod, których celem jest zwalczenie problemu nadmiernej generalizacji nękającego algorytm *SMOTE*. Zaproponowano w tym celu metdy adaptacyjnego próbkowania takie jak m.in. *Borderline-SMOTE* i *Adaptive Synthetic Sampling (ADASYN)*.
   - Próbkowanie z technikami czyszczenia danych - metody stworzone w celu usunięcia się problemu nakładania się wynikającego z metod próbkujących. Przykładem jest np. technika *Tomek links*, której celem jest usunięcie przykładów, które mogą utrudniać działanie modelu i poprawienie jakości danych, co może prowadzić do lepszych wyników klasyfikacji. Polega to na znajdywanie par próbek, które należą do różnych klas i są swoimi najbliższymi sąsiadami w przestrzeni cech - następnie dla takiej pary, zależnie od obranej strategii, można np. usunąć próbkę należącą do klasy większościowej lub też usunąć obie, jako że mogą one być potencjalnym "szumem". Rozwiązanie to jest skuteczne, ponieważ w przypadku klas niezbalansowanych próbki klasy większosciowej na granicach w przestrzeni cech mogą dominować nad próbkami klasy mniejszościowej, co prowadzi do błędów klasyfikacji.
   - Metody próbkowania oparte na klastrach - zbiór algorytmów, które wykorzystują techniki klastrowania, co zapewnia im dodatkowy element elastyczności, którego nie ma w prostych syntetycznych algorytmach próbkowania. Przykładem dla tych metod jest *Cluster Based Oversampling (CBO)*, który wykorzystuje metodę klastrowania *K-średnich*

   - Integracja próbkowania i wzmacniania - zbiór metod, które próbują łączyć techniki próbkowania i wzmacniania, takie jak np. *SMOTEBoost*, który łączy *SMOTE* oraz *AdaBoost.M2*.
2. Metody wrażliwe na koszty - zbiór metod, które zamiast skupiać się na równoważeniu rozkładów klas mniejszościowych i większościowych, biorą pod uwagę użycie macierzy kosztów związanych z błędną klasyfikacją dowolnego konkretnego przykładu danych.
3. Metody oparte na jądrze - rodzina metod, która skupia się głównie na mechanice samego algorytmu *SVM*. Jednym z jej przykładów jest algorytm konstrukcji klasyfikowatora jądra oparty na ortagonalnej selekcji do przodu i zregularyzowanym ortagonalnym estymatorze najmniejszych ważonych kwadratów (*ROWLS*). Optymalizuje on generalizację w modelu uczenia się opartym na jądrze wprowadzając dwa główne komponenty, które zajmują się niezrównoważonymi rozkładami dla dwuklasowych zestawów danych - pierwszy komponent integruje koncepcję walidacji krzyżowej typu *LOO* i metrykę oceny pola pod krzywą do opracowania funkcji obiektywnej *LOO-AUC*, a drugi komponent wykorzystuje wrażliwość kosztową funkcji kosztu szacowania parametru w algorytmie *ROWLS*, aby przypisać większą wagę przykładom błędnych danych w klasie mnejszościowej - możliwe modyfikacje dla tej rodziny metod obejmują m.in. techniki dostosowywania granicy klas *SVM*.
4. Aktywne uczenie - choć tradycyjne metody aktywnego uczenia używane są do rozwiązywania problemów związanych z nieoznakowanymi danymi treningowymi, w ostatnich latach badane były podejścia do klasyfikacji danych niezbalansowanch z użyciem tego typu metod. Rozwiązania te oparte na *SVM* osiągają zadowalające wyniki, gdzie *SVM* trenowany jest na wielu podzbiorach, po czym najbardziej informatywne instancje zostają ekstrahowane i formowane w nowy zestaw treningowy zgodny z opracowaną hiperpłaszczyzną.[4]

2 Metodologia

2.1 Opis problemu

W ramach przeprowadzanych badań analizuję zbiór danych "Credit Card Fraud Detection"który zawiera transakcje dokonywane kartami kredytowymi w ciągu dwóch dni we wrześniu 2013r. przez europejskich posiadaczy kart. We wspomnianym okresie doszło do 492 oszustw z 284807 transakcji - stopień niezbalansowania[7] wynosi więc około 578, co jest bardzo wysokim wynikiem, dane te są więc idealne na potrzeby moich badań.
Zbiór składa się z cech liczbowych (V1, V2, ..., V28), które są wynikiem transformacji PCA a także kolumn "Czas", "Kwota"oraz "Klasa", która jest zmienną odpowiedzi i przyjmuje wartość "1"w przypadku oszustwa. Kolumna "Czas"została usunięta przeze mnie ze zbioru danych, ponieważ uznałem, że nie będzie ona przydatna w przypadku mojego problemu (wskazywała ona czas w sekundach od dokonania pierwszej transakcji, która znajduje się w zbiorze danych).



Rysunek 1. Przestrzeń cech dla zbioru danych: "Credit Card Fraud Detection"

Jak pokazano na rysunku 1, przestrzeń cech dla zbioru danych "Credit Card Fraud Detection"charakteryzuje się tym, że klasy mniejszościowe i większościowe są szeroko rozproszone i w dużym stopniu nakładają się na siebie. Przykłady oszustw i prawidłowych transakcji nie są wyraźnie oddzielone, co będzie stanowiło duże wyzwanie w kontekście rozróżniania ich w procesie klasyfikacji[7].

2.2 Opis algorytmu

Podstawowym zadaniem tego projektu była własna implementacja jednego z estymatorów - w moim przypadku był to AdaBoostClassifier. Po wykonaniu tego zadania przeprowadzone zostały badania, które obejmują dziesięciokrotne wykonanie każdego z trzech opracowanych przeze mnie eksperymentów:

1. Eksperyment 1 skupia się na analizie porównawczej przygotowanego zestawu klasyfikatorów na podstawie zbioru danych, który nie został jeszcze poddany żadnej z metod balansujących. Zaproponowane klasyfikatory to: *GaussianNB*, *LogisticRegression*, *RandomForest*, *AdaBoostClassifier* oraz *własna implementacja estymatora AdaBoost*.
2. Eksperyment 2 skupia się na analizie wpływu metod balansowania zbioru danych na wyniki uzyskiwane w zadaniu klasyfikacji. Trenowany jest tu jeden z zaproponowanych w eksperymencie klasyfikatorów (*LogisticRegression* w połączeniu z czterema różnymi strategiami balansowania zbioru: *RandomOverSampler*, *SVMSMOTE*, *RandomUnderSampler* oraz *TomekLinks*.
3. Eksperyment 3 poddaje analizie porównawczej zestaw czterech klasyfikatorów zbiorczych zapewnionych przez bibliotekę *imbalanced-learn*: *EasyEnsamble*, *RUSBoost*, *BalancedBagging* oraz *BalancedRandomForest*.

Dokładne parametry każdego z klasyfikatorów zostały dobrane na drodze eksperymentalnej - każdy z nich został uruchomiony dla pięciu różnych ustawień hiperparametrów, a w ostatecznych badaniach użyte zostały te ustawienia klasyfikatorów, które osiągnęły najlepsze wyniki podczas tego procesu.

3 Badania

3.1 Pytania badawcze i plan eksperymentu

1. Pytania badawcze
   (a) Które algorytmy klasyfikacyjne osiągają najlepsze wyniki w przypadku pracy z niezbalansowanymi danymi?
   (b) Jaki wpływ na skuteczność klasyfikatorów dla danych niezbalansowanych mają techniki balansowania: oversampling oraz undersampling?
   (c) Czy metody zbiorcze pozwalają na osiągnięcie lepszych wyników dla problemu klasyfikacji w kontkeście danych niezbalansowanych od klasycznych rozwiązań, które nie zawierają żadnych metod balansowania zbioru danych?
2. Plan eksperymentu
   (a) Przygotowanie zbioru danych
      – Wczytanie zbioru danych
      – Wstępne przetworzenie zbioru danych
   (b) Przeprowadzenie eksperymentu 1 - Analiza porównawcza efektywności klasyfikatorów bez zastosowania technik balansowania zbioru danych.
   Wybrane algorytmy klasyfikacji
      – Naiwny klasyfikator Bayesa
      – Regresja logistyczna
      – Las losowy
      – AdaBoost
      – Własna implementacja klasyfikatora AdaBoost

(c) Przeprowadzenie eksperymentu 2 - Analiza wpływu technik oversamplingu i undersamplingu na działanie klasyfikatorów dla niezbalansowanego zbioru danych.
Wybrane klasyfikatory:
- Naiwny klasyfikator Bayesa
- Własna implementacja klasyfikatora AdaBoost

Wybrane techniki balansowania:
- SMOTE (oversampling)
- RandomUnderSampler (undersampling)

(d) Przeprowadzenie eksperymentu 3 - Analiza porównawcza efektywności metod zbiorczych dla niezbalansowanego zbioru danych.
Wybrane klasyfikatory:
- EasyEnsamble
- BalancedBagging
- RUSBoost
- BalancedRandomForest

(e) Analiza wyników dla przeprowadzonych eksperymentów

3.2 Opis środowiska badawczego

1. Opis zbioru danych
2. Środowisko programistyczne
   Eksperymenty przeprowadzono w języku Python 3.11.0 korzystając z bibliotek: *pandas*, *numpy*, *matplotlib*, *scikit-learn*, *imbalanced-learn* i *xgboost*.
3. Ustawienia eksperymentu
   (a) 5-krotna walidacja krzyżowa
   (b) Ustawienia poszczególnych klasyfikatorów:
   - Naiwny klasyfikator Bayesa: ustawienia domyślne
   - Regresja logistyczna: *penalty*="l2", *C*=1.0, *solver*="lbfgs", *class_ weight*=None
   - Las losowy: *n_ estimators*=100, *max_ depth*=15, *min_ samples_ split*=2, *class_ weight*=None
   - AdaBoost: *n_ estimators*=150, *learning_ rate*=1.0, *algorithm*="SAMME"
   - Własna implementacja klasyfikatora AdaBoost: *n_ estimators*=25
   - EasyEnsemble: *n_ estimators*=10, *sampling_ strategy*=0.5
   - RUSBoost: *n_ estimators*=100, *learning_ rate*=0.1, *sampling_ strategy*=0.5
   - BalancedBagging: *n_ estimators*=10, *sampling_ strategy*=0.7, *max_ samples*=0.5
   - BalancedRandomForest: *n_ estimators*=100, *max_ depth*=5, *sampling_ strategy*=0.4

   (c) Ustawienia technik balansowania zbioru danych:
   - RandomOverSampler: *sampling_ strategy*={0: 284315, 1: 2000}
   - SVMSMOTE: *sampling_ strategy*="minority"
   - RandomUnderSampler: *sampling_ strategy*=0.7
   - TomekLinks: *sampling_ strategy*="majority"

   (d) Kryteria oceny: AUC-ROC oraz F1

3.3 Wyniki badań

Kolory w tabelkach oznaczają:

- zielony - najwyższa wartość metryki lub najniższa wartość jej odchylenia standardowego spośród wszystkich klasyfikatorów w eksperymencie
- czerwony - najniższa wartość metryki lub najwyższa wartość jej odchylenia standardowego spośród wszystkich klasyfikatorów w eksperymencie
- jasnozielony - ten kolor został użyty do tego, aby oznaczyć te wartości metryk oraz ich odchyleń standardowych uległy poprawie po użyciu danej metody balansowania zbioru danych
- jasnoczerwony - ten kolor został użyty do tego, aby oznaczyć te wartości metryk oraz ich odchyleń standardowych uległy pogorszeniu po użyciu danej metody balansowania zbioru danych
- jasnoniebieski - ten kolor został użyty aby zwiększyć przejrzystość macierzy istotności statystycznych, oznaczając wartości pozytywne w tych macierzach

Eksperyment 1:

Mean Scores - comparing classifiers (01)					
	Naive Bayes	Logistic Regression	Random Forest	AdaBoost	Custom AdaBoost
Mean F1	0.115368	0.721543	0.846578	0.744762	0.486845
Mean AUC	0.960658	0.97405	0.974526	0.976272	0.952353

Rysunek 2. Średnie wartości F1 i AUC dla klasyfikatorów w *Eksperymencie 01*

Standard Deviations - comparing classifiers (01)					
	Naive Bayes	Logistic Regression	Random Forest	AdaBoost	Custom AdaBoost
Std Dev F1	0.00358913	0.0253905	0.0164006	0.0218804	0.160442
Std Dev AUC	0.0047188	0.00704424	0.00615916	0.00449639	0.0185808

Rysunek 3. Średnie odchylenia standardowe metryk F1 i AUC dla klasyfikatorów w *Eksperymencie 01*

Statistical Significance Matrix for F1 (1: Significant, 0: Not Significant) - comparing classifiers (01)					
	Naive Bayes	Logistic Regression	Random Forest	AdaBoost	Custom AdaBoost
Naive Bayes	0	1	1	1	1
Logistic Regression	1	0	1	1	1
Random Forest	1	1	0	1	1
AdaBoost	1	1	1	0	1
Custom AdaBoost	1	1	1	1	0

Rysunek 4. Macierz istotności statystycznej dla wyników metryki F1 w kontekście klasyfikatorów wykorzystanych w *Eksperymencie 01*

Statistical Significance Matrix for AUC (1: Significant, 0: Not Significant) - comparing classifiers (01)					
	Naive Bayes	Logistic Regression	Random Forest	AdaBoost	Custom AdaBoost
Naive Bayes	0	1	1	1	1
Logistic Regression	1	0	0	1	1
Random Forest	1	0	0	1	1
AdaBoost	1	1	1	0	1
Custom AdaBoost	1	1	1	1	0

Rysunek 5. Macierz istotności statystycznej dla wyników metryki AUC w kontekście klasyfikatorów wykorzystanych w *Eksperymencie 01*

Eksperyment 2:

Mean Scores and Standard Deviations - comparing balancing strats (02)					
	None	RandomOverSampler	SVMSMOTE	RandomUnderSampler	TomekLinks
Mean F1	0.720117	0.78776	0.351695	0.107323	0.726687
Std Dev F1	0.0235994	0.0159396	0.0960088	0.0205837	0.0240499
Mean AUC	0.973969	0.976554	0.957853	0.974951	0.974178
Std Dev AUC	0.00639435	0.00460083	0.0100601	0.0060349	0.0059798

Rysunek 6. Średnie wartości F1 i AUC oraz średnie odchylenia standardowe tych metryk dla klasyfikatorów w *Eksperymencie 02*

Statistical Significance Matrix for F1 (1: Significant, 0: Not Significant) - comparing balancing strats (02)					
	None	RandomOverSampler	SVMSMOTE	RandomUnderSampler	TomekLinks
None	0	1	1	1	1
RandomOverSampler	1	0	1	1	1
SVMSMOTE	1	1	0	1	1
RandomUnderSampler	1	1	1	0	1
TomekLinks	1	1	1	1	0

Rysunek 7. Macierz istotności statystycznej dla wyników metryki F1 w kontekście klasyfikatorów wykorzystanych w *Eksperymencie 02*

Statistical Significance Matrix for AUC (1: Significant, 0: Not Significant) - comparing balancing strats (02)					
	None	RandomOverSampler	SVMSMOTE	RandomUnderSampler	TomekLinks
None	0	1	1	0	0
RandomOverSampler	1	0	1	1	1
SVMSMOTE	1	1	0	1	1
RandomUnderSampler	0	1	1	0	0
TomekLinks	0	1	1	0	0

Rysunek 8. Macierz istotności statystycznej dla wyników metryki AUC w kontekście klasyfikatorów wykorzystanych w *Eksperymencie 02*

Eksperyment 3:

Mean Scores - comparing ensamble methods (03)				
	EasyEnsamble	RUSBoost	BalancedBagging	BalancedRandomForest
Mean F1	0.156921	0.32522	0.146762	0.513321
Mean AUC	0.97806	0.969344	0.967637	0.975141

Rysunek 9. Średnie wartości F1 i AUC dla klasyfikatorów w *Eksperymencie 03*

Standard Deviations - comparing ensamble methods (03)				
	EasyEnsamble	RUSBoost	BalancedBagging	BalancedRandomForest
Std Dev F1	0.0138739	0.0686132	0.0269939	0.0450571
Std Dev AUC	0.00384822	0.00495233	0.00643375	0.00459834

Rysunek 10. Średnie odchylenia standardowe metryk F1 i AUC dla klasyfikatorów w *Eksperymencie 03*

Statistical Significance Matrix for F1 (1: Significant, 0: Not Significant) - comparing ensamble methods (03)				
	EasyEnsamble	RUSBoost	BalancedBagging	BalancedRandomForest
EasyEnsamble	0	1	1	1
RUSBoost	1	0	1	1
BalancedBagging	1	1	0	1
BalancedRandomForest	1	1	1	0

Rysunek 11. Macierz istotności statystycznej dla wyników metryki F1 w kontekście klasyfikatorów wykorzystanych w *Eksperymencie 03*

Statistical Significance Matrix for AUC (1: Significant, 0: Not Significant) - comparing ensamble methods (03)				
	EasyEnsamble	RUSBoost	BalancedBagging	BalancedRandomForest
EasyEnsamble	0	1	1	1
RUSBoost	1	0	1	1
BalancedBagging	1	1	0	1
BalancedRandomForest	1	1	1	0

Rysunek 12. Macierz istotności statystycznej dla wyników metryki AUC w kontekście klasyfikatorów wykorzystanych w *Eksperymencie 03*

4 Wnioski

4.1 Odpowiedzi na pytania badawcze

1. Przeprowadzony przeze mnie pierwszy eksperyment pozwolił na zbadanie tego, jak standardowe estymatory poradzą sobie w sytuacji, w której zbiór

danych jest silnie niezbalansowany. Zgodnie z oczekiwaniami, zdecydowanie najgorzej poradził sobie naiwny klasyfikator Bayesa - opiera się on na prawdopodobieństwach klas, które dla klas mniejszościowych są bardzo niskie, stąd też tak niskie wyniki metryki F1, co widać na zdjęciu 2. Zdecydowanie najlepiej poradził sobie las losowy, dzięki swojej zdolności do kontrolowania nadmiernego dopasowania oraz wykrywania różnorodnych wzorców w danych dzięki wykorzystaniu wielu drzew decyzyjnych. AdaBoostClassifier również korzysta z drzew decyzyjnych, nie udało mi się jednak znaleźć konfiguracji, która pozwoliłaby na osiągnięcie lepszych wyników niż w przypadku lasów losowych, zarówno w wersji znajdującej się w bibliotece *scikit-learn*[2], jak i we własnej implementacji, która poradziła sobie wyraźnie gorzej ($\sim 48,7\%$) - wynika to zapewne z tego, że moja implementacja tego metaestymatora jest implementacją podstawową - jestem pewien, że możliwe byłoby dostrojenie tej metody tak, by radziła sobie z problemami niezbalansowanych danych znacznie lepiej, jest to jednak zadanie wykraczające poza zakres tego projektu.

2. Ze wszystkich metod balansowania zbioru danych zdecydowanie najlepiej poradził sobie *RandomOverSampler*, osiągając nie tylko najwyższe, ale też najbardziej stabilne wyniki dla obu badanych metryk, co jest widoczne na 6. Zwiększa on reprezentację klasy mniejszościowej, jednakże w sposób ograniczony[1], co pozwoliło na uniknięcie przeuczenia. Klasy większościowe i mniejszościowe nie są też dobrze rozdzielone i mają nakładające się na siebie obszary - losowe nadpróbkowanie pozwoliło więc skupić się nieco bardziej na cechach klas mniejszościowych. Wspomniane już słabe rozdzielenie cech tych klas w przestrzeni cech jest też powodem dlaczego metoda *SVMSMOTE* poradziła sobie tak bardzo słabo, ponieważ metoda ta tworzy syntetyczne próbki w niewielkich odległościach od próbki w stronę jednego z jej najbliższych sąsiadów, co sprawia, że próbki te są nierozróżnialne od niektórych reprezentatów klasy większościowej. Najgorszy pod kątem metryki F1 jest *RandomUnderSampler* - metoda podpróbkowania redukuję liczbę próbek klasy większościowej, co może doprowadzić do dużej straty informacji, co może również stanowić problem do pracującego na tych danych klasyfikatora (regresji logistycznej), który wymaga tych danych do poprawienia swojej własnej wydajności. Metoda *TomekLinks* poradziła sobie dobrze, nie wyróżniła się jednak na tle zdecydowanie lepiej działającego losowego nadpróbkowania - usuwa ona reprezentatów klasy większościowej, które są najbliżej elementów klasy mniejszościowej, co wydawało się być odpowiednim podejściem biorąc pod uwagę nachodzenie się próbek obu klas, podejrzewam jednak, że poza swoimi pozytywnymi efektami mogło to też doprowadzić do usunięcia takich przypadków, które mogłyby być kluczowe do rozróżniania elementów z obu tych klas, co mogło ograniczyć poprawę działania tej strategii.
   Ostatecznie udało mi się dowieść, że techniki balansowania mogą mieć bardzo pozytywne efekty podczas pracy z danymi niezbalansowanymi. Ważne jest, by brać pod uwagę to jaka jest charakterystyka naszych danych, jako że każdy problem jest inny i nie zawsze każda z tych strategii będzie skuteczna - tak jak w moim przypadku, po przeprowadzeniu odpowiedniej analizy możliwe

jest jednak znalezienie takiego rozwiązania, który pozwoli naszej metodzie znacznie poprawić swoją wydajność.

3. Zaproponowane przeze mnie metody zbiorcze nie osiągnęły niestety zadowalających wyników. Choć powinny one zdecydowanie lepiej radzić sobie w kontekście niezbalansowanych danych, nie udało mi się znaleźć odpowiednich konfiguracji, które pozwoliłyby na poprawę wyników - wyniki widoczne na 9 osiągnięte przez te algorytmy są gorsze od prawie wszystkich innych estymatorów badanych w eksperymencie pierwszym. Wyjątkami są jedynie moja własna implementacja *AdaBoostClassifier*, która okazała się o niemal 3 punkty procentowe mniej wydajna w kontekście metryki F1 niż *BalancedRandomForest* oraz naiwny klasyfikator *Bayesa*, który wciąż pozostaje najmniej skutecznym ze wszystkich estymatorów). Niewątpliwie możliwe jest znalezienie takicy konfiguracji hiperparametrów, które osiągałyby lepsze wyniki dla badanych przeze mnie metryk, wymagałoby to jednak przeprowadzenia dalszych dokładnych eksperymentów w tej dziedzinie.

4.2 Podsumowanie

Podczas analizy wyników moich eksperymentów skuteczność modeli oceniana była za pomocą metryk F1 i AUC. Zauważyłem jednak, że metryka AUC, choć powszechnie stosowana, okazała się mało przydatna w kontekście mojego problemu, jako że wyniki AUC były bardzo zbliżone dla wszystkich estymatorów, co utrudniało rzetelne porównanie i ocenę ich efektywności. Z tego względu metryka F1, która lepiej równoważy precyzję i czułość okazała się bardziej adekwatna w mojej analizie.

Ostatecznie przeprowadzone przeze mnie eksperymenty zostały w większości zakończone sukcesem. Udało mi się w większości odpowiedzieć na postawione przeze mnie pytania badawcze i dojść do ciekawych wniosków odnośnie tego jak poszczególne estymatory oraz metody balansowania danych pozwalają radzić sobie ze specyficznym problemem wykonywania zadania klasyfikacji na zbiorze o dużym współczynniku niezbalansowania. Analiza pokazała jak istotne jest indywidualne podejście do każdego problemu i odpowiednie dobranie metod, jako że różnice potrafią być bardzo wyraźne, zależnie od natury problemu. Estymatory takie jak *RandomForest*, *BalancedRandomForest* i *AdaBoostClassifier* oraz metody balansowania takie jak *RandomOverSampler* i *TomekLinks* wydają się być najbardziej skuteczne dla badanego przeze mnie zbioru danych, nie jest to jednak odpowiedź uniwersalna, która musi być prawdziwa również dla innych zestawów.

Literatura

1. imbalanced-learn documentation (2024), https://imbalanced-learn.org/stable/index.html, mIT-licensed library. Last accessed: 2024-11-24
2. scikit-learn Machine Learning in Python (2024), https://imbalanced-learn.org/stable/index.html, mIT-licensed library. Last accessed: 2024-11-24
3. Haixiang, G., Yijing, L., Junjie, S., Mingyun, G., Yuanyue, H., Bing, G.: Learning from class-imbalanced data: Review of methods and applications. Expert Systems with Applications **73**, 220–239 (2017). https://doi.org/10.1016/j.eswa.2016.12.035
4. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering **21**(9), 1263–1284 (2009). https://doi.org/10.1109/TKDE.2008.239
5. Kaur, H., Pannu, H., Malhi, A.: A systematic review on imbalanced data challenges in machine learning. ACM Computing Surveys **52**, 1–36 (2019). https://doi.org/10.1145/3343440
6. Kumar, P., Bhatnagar, R., Gaur, K., Bhatnagar, A.: Classification of imbalanced data: Review of methods and applications. IOP Conference Series: Materials Science and Engineering **1099**, 012077 (2021). https://doi.org/10.1088/1757-899x/1099/1/012077
7. Vuttipittayamongkol, P., Elyan, E., Petrovski, A.: On the class overlap problem in imbalanced data classification. Knowledge-Based Systems **212**, 106631 (2021). https://doi.org/10.1016/j.knosys.2020.106631