Politechnika Wrocławska

Uczenie maszyn

Sprawozdanie projektowe

Klasyfikacja binarna danych niezbalansowanych z wykorzystaniem metod zespołowych

Autorzy
Bartosz Rodziewicz 226105
Radosław Skorulski 241294

Termin zajęć:
Środa 18.55

2 lutego 2022

1 Wstęp

1.1 Zadanie klasyfikacji

Zadanie rozpoznawania wzorców, inaczej klasyfikacja polega na przypisaniu danego obiektu do określonych kategorii. Podział na kategorie (klasy) dokonywany jest na podstawie wartości wybranych cech określających obiekt będący klasyfikowanym [1]. W ramach tego projektu badana jest klasyfikacja binarna, co oznacza że istnieją tylko dwie kategorie do których obiekt może być sklasyfikowany.

1.2 Dane niezbalansowane

Mianem danych niezbalansowanych w kontekście klasyfikacji określa się zbiory w których wystąpień pewnej klasy jest znacznie mniej niż wystąpień innej klasy [2]. Dla zadania klasyfikacji binarnej, która jest przedmiotem prowadzonych badań niezbalansowanie zbioru można określić jako iloraz liczby wystąpień klasy większościowej i liczby wystąpień klasy mniejszościowej. Przykładowo w pewnym zbiorze liczba próbek klasy większościowej to 100, a liczba próbek klasy mniejszościowej to 50. Iloczyn tych wartości, a za razem niezbalansowanie zbioru to 2,0.

1.3 Metody zespołowe

Na metodę zespołową (ensemble) składa się zbiór klasyfikatorów bazowych, których predykcje są łączone w celu dokonania zadania klasyfikacji. Według wielu badań użycie metod zespołowych pozwala osiągnąć większą dokładność niż w przypadku pojedynczego klasyfikatora [3]. Wyróżniamy metody zespołowe homogeniczne i heterogeniczne. W metodach homogenicznych wszystkie klasyfikatory wchodzące w skład zespołu są generowane za pomocą tego samego typu klasyfikatora, natomiast w heterogenicznych, używane są różne typy klasyfikatorów. W tej pracy badano jedynie klasyfikatory homogeniczne.

2 Cel i plan eksperymentu

2.1 Cel eksperymentu

Celem eksperymentu jest zbadanie i porównanie skuteczności wybranych metod zespołowych w zadaniu klasyfikacji binarnej przeprowadzanym na zbiorze danych niezbalansowanych.

2.2 Plan eksperymentu

Każdy z badanych zbiorów zostanie podzielony w ramach 5-cio krotnej podwójnej stratyfikowanej walidacji krzyżowej. Zbiory dzielone są w taki sam sposób dla każdej z badanych metod, aby można było je później porównać. Po podzieleniu zbiory uczące zostaną następnie zbalansowane algorytmem SMOTE [4], który wykorzystuje oversampling. W odróżnieniu od klasycznego oversamplingu metoda SMOTE nie duplikuje obiektów klasy mniejszościowej, a tworzy nowe obiekty, podobne do znanych wystąpień klasy mniejszościowej. Zbiory testowe nie będą oversamplowane.

Badane metody zespołowe to kolejno: AdaBoostClassifier, BaggingClassifier oraz RandomSubspaceClassifier. Każda z powyższych metod zespołowych będzie składać się z wielu instancji tych samych klasyfikatorów bazowych. Do tego celu zostaną wykorzystane klasyfikatory: decisionTree, logisticRegression i gaussianNB. Implementacja wybranych metod zespołowych oraz klasyfikatorów zostanie zaczerpnięta z biblioteki scikit-learn [5]. Dla każdego z badanych zbiorów danych planujemy użyć każdej z powyższych metod zespołowych.

W celu zmierzenia skuteczności różnych metod zostanie wykorzystana metryka F1 [6]. Jest to średnia harmoniczna dwóch parametrów:

- precyzji, czyli jak dobrze dana klasa jest poprawnie klasyfikowana pośród wszystkich obiektów tej klasy w zbiorze testowym,
- recall, czyli jak dobrze dana klasa jest poprawnie klasyfikowana pośród wszystkich obiektów sklasyfikowanych jako ta klasa.

Oczywiście metryka F1 będzie mierzyła jakość klasyfikacji obiektów klasy mniejszościowej. Szukając informacji na temat metryki F1 można natknąć się na informacje, że metryka ta nie jest dobra do porównywania klasyfikatorów, gdyż w zależności od badanego zbioru któryś z dwóch parametrów wchodzących w skład F1 może okazać się ważniejszy. Jednakże w badanych zbiorach nie określono, który z parametrów lepiej mierzyłby skuteczność klasyfikacji, dlatego postanowiono na metrykę F1, aby jednocześnie brać pod uwagę oba parametry.

Kolejnym krokiem będzie wykonanie parowych testów statystycznych. Każda z trzech metod zespołowych zostanie zestawiona z pozostałymi w ramach jednego klasyfikatora bazowego. Zostaną wykonane testy t-Studenta w celu uzyskania statystycznych różnic między metodami zespołowymi dla każdego zbioru danych. Dodatkowo na podstawie uśrednionej oceny ze wszystkich zbiorów zostanie przeprowadzony test Wilcoxona, na podstawie którego będzie można stwierdzić przewagę metod nad innymi dla ogółu badanych zbiorów. Dla obu testów statystycznych został przyjęty próg ufności na poziomie 0,05.

2.3 Wykorzystane zbiory danych

Eksperymenty zostały przeprowadzone dla 20 zbiorów. Dane pochodzą ze strony KEEL-dataset repository [7]. Wykorzystane zbiory zostały zaprezentowane w tabeli 1. Dla każdego ze zbiorów określono liczebność oraz miarę niezbalansowania. Niezbalansowanie badanych zbiorów waha się w przedziale od 1,82 do 32,73, ze średnim niezbalansowaniem wynoszącym 10,05.

Tabela 1: Wykorzystane zbiory

l.p.	nazwa zbioru	liczebność	liczba atrybutów	niezbalansowanie
1	ecoli1	336	7	3.36
2	glass0	214	9	2.06
3	glass1	214	9	1.82
4	glass4	214	9	15.47
5	glass6	214	9	6.38
6	haberman	306	3	2.78
7	iris0	150	4	2
8	new-thyroid1	215	5	5.14
9	page-blocks-1-3_vs_4	472	10	15.86
10	pima	768	8	1.87
11	segment0	2308	19	6.02
12	vehicle0	846	18	3.25
13	vehicle3	846	18	2.99
14	vowel0	988	13	9.98
15	winequality-red-4	1599	11	29.17
16	wisconsin	683	9	1.86
17	yeast-1-4-5-8_vs_7	693	8	22.1
18	yeast3	1484	8	8.1
19	yeast4	1484	8	28.1
20	yeast5	1484	8	32.73

3 Opis środowiska testowego

Podczas realizacji projektu został wykorzystany edytor tekstowy Atom [8] oraz język skryptowy Python [9] w wersji 3.10. Kod źródłowy został podzielony na oddzielne pliki aby łatwo oddzielić fragmenty kodu odpowiedzialne za różne zadania.

W projekcie zostały wykorzystane następujące biblioteki:

- scikit–learn [5] – do implementacji selekcji oraz klasyfikacji danych,
- imblearn [10] – do balansowania danych za pomocą algorytmu SMOTE,
- scipy [11] – do przeprowadzenia testów t-Studenta i Wilcoxona,
- numpy [12] – do ładowania i operacji na danych i wynikach,
- pandas [13] – do przekształcania danych,
- tabulate [14] – do prezentacji wyników w konsoli.

Klasyfikator RandomSubspaceClassifier został utworzony za pomocą modyfikacji implementacji klasyfikatora BaggingClassifier. Parametry użytych w pracy implementacji metod zespołowych są następujące:

- AdaBoostClassifier – n_estimators = 50, learning_rate = 1.0
- BaggingClassfier – n_estimators = 50, random_state = const
- RandomSubspaceClassifier –n_estimators = 50, random_state = const, bootstrap = False, max_features = 1/2 liczby cech w zbiorze.

4 Wyniki i wnioski

Wyniki zostały przedstawione w trzech tabelach, każda tabela odpowiada jednemu rodzajowi klasyfikatora bazowego. W tabeli 2 przedstawiono wyniki uzyskane dla klasyfikatora bazowego decisionTree, w tabeli 3 zawarto wyniki klasyfikacji przy użyciu bazowego klasyfikatora logisticRegresion, a w tabeli 4 znajdują się wyniki klasyfikacji klasyfikatora bazowego gaussianNB. Wiersze tych tabeli reprezentują poszczególne zbiory danych, natomiast kolumny to metody zespołowe. W komórkach przedstawiono wynik F1 klasyfikacji oraz informację od których metod zespołowych zawartych w tabeli dana metoda jest statystycznie znacząco lepsza. Informacja o przewadze metody nad inną została uzyskana za pomocą testu t-Studenta. W ostatnim wierszu przedstawiono średnią rangę osiągniętą przez metodę zespołową dla wszystkich zbiorów oraz informację od których metod zespołowych dana metoda jest średnio statystycznie znacząco lepsza. Średnia przewaga metod została wyznaczona za pomocą testu Wilcoxona.

Pierwszym badanym klasyfikatorem bazowym dla którego badano metody zespołowe było drzewo decyzyjne (decisionTree). Metoda AdaBoost dla trzech zbiorów okazała się statystycznie znacząco gorsza od metody RandomSubspace. Bagging zyskał statystycznie znaczącą przewagę nad AdaBoost w połowie badanych przypadków, oraz okazał się być lepszy od RandomSubspace w 7 przypadkach. RandomSubspace był lepszy od AdaBoost dla siedmiu zbiorów i lepszy od Baggingu w 4 przypadkach. Ogólnie ujmując spośród tych trzech metod, Bagging okazał się lepszy od metody AdaBoost. Choć wystąpiły różnice pomiędzy pozostałymi parami metod, nie okazały się one znaczące.

Podczas badania bazowego klasyfikatora logisticRegresion, wykazano że dla trzech przypadków metoda AdaBoost daje wyniki lepsze od RandomSubspace. Bagging okazał się lepszy od AdaBoost dla 7 przypadków i lepszy od RadomSubspace również w 7 przypadkach. Metoda RandomSubspace była lepsza od AdaBoost dla 7 przypadków oraz lepsza od Baggingu w przypadku dwóch zbiorów. Patrząc na całość, podobnie jak dla drzewa decyzyjnego, metoda Baggingu okazała się lepsza od AdaBoost. Metoda RandomSubspace dla badanych zbiorów także okazała się średnio statystycznie lepsza od AdaBoost.

Bazowy klasyfikator gaussianNB był ostatnim klasyfikatorem pod kątem którego badano skuteczność metod zespołowych. Metoda AdaBoost okazała się lepsza od Baggingu i metody RandomSubspace w przypadku 4 zbiorów(tych samych). Bagging wykazał przewagę nad AdaBoost w przypadku 7 zbiorów i przewagę nad RandomSubspace w przypadku 3 zbiorów. Metoda RandomSubspace wykazała się przewagą nad AdaBoost w 7 przypadkach, oraz przewagą nad Baggingiem dla jednego zbioru. Dla naiwnego klasyfikatora bayesowskiego, żadna spośród trzech badanych metod zespołowych nie wykazała statystycznie znaczącej przewagi nad pozostałymi.

Można także zauważyć, że skuteczność klasyfikacji mierzona miarą F1 w zbiorach zbiorach winequality-red-4 oraz yeast-1-4-5-8_vs_7 odbiega od skuteczności klasyfikacji innych zbiorów, niezależnie od użytych metod zespołowych i klasyfikatorów bazowych. Być może wynika to ze specyfiki danych tych dwóch zbiorów - obiekty przypisywane do obu klas mogły mieć podobne wartości atrybutów, przez co metoda SMOTE przy oversamplingu zbioru mniejszościowego tworzyła obiekty o wartościach atrybutów zbyt bliskich do tych posiadanych przez obiekty klasy większościowej. Oczywiście taką hipotezę należałoby zbadać.

Z kolei skuteczność klasyfikacji dla zbioru danych iris0, dla wszystkich badanych przypadków z wyjątkiem połączenia metody AdaBoost z klasyfikatorem logisticRegression, była możliwie najlepsza – wszystkie obiekty z klasy mniejszościowej zostały poprawnie zakwalifikowane.

Tabela 2: Wyniki – klasyfikator bazowy decisionTree

zbiór danych	AdaBoost – 1	Bagging – 2	RandomSubspace – 3
ecoli1	0.759 3	0.778 3	0.682 —
glass0	0.679 —	0.737 1	0.743 1
glass1	0.653 —	0.715 —	0.725 1
glass4	0.595 —	0.586 —	0.724 2
glass6	0.791 —	0.841 1	0.851 1
haberman	0.415 3	0.409 3	0.203 —
iris0	1.000 —	1.000 —	1.000 —
new-thyroid1	0.937 —	0.927 —	0.924 —
page-blocks-1-3_vs_4	0.939 —	0.939 —	0.938 —
pima	0.586 —	0.645 1,3	0.611 1
segment0	0.976 —	0.981 —	0.986 1,2
vehicle0	0.861 —	0.901 1	0.921 1,2
vehicle3	0.511 —	0.571 1,3	0.537 —
vowel0	0.893 —	0.915 —	0.957 1,2
winequality-red-4	0.105 —	0.129 —	0.110 —
wisconsin	0.922 —	0.944 1	0.957 1,2
yeast-1-4-5-8_vs_7	0.095 —	0.104 —	0.054 —
yeast3	0.713 3	0.762 1,3	0.491 —
yeast4	0.277 —	0.357 1,3	0.238 —
yeast5	0.647 —	0.711 1,3	0.598 —
średnia ranga	1.65 —	2.35 1	2 —

Tabela 3: Wyniki – klasyfikator bazowy logisticRegression

zbiór danych	AdaBoost – 1	Bagging – 2	RandomSubspace – 3
ecoli1	0.753 —	0.750 —	0.753 —
glass0	0.654 —	0.642 —	0.657 —
glass1	0.542 —	0.568 1,3	0.548 —
glass4	0.591 —	0.593 —	0.602 —
glass6	0.788 —	0.798 —	0.827 1
haberman	0.466 —	0.480 —	0.476 —
iris0	0.990 —	1.000 1	1.000 1
new-thyroid1	0.902 —	0.935 —	0.964 —
page-blocks-1-3_vs_4	0.714 3	0.761 3	0.570 —
pima	0.656 —	0.661 —	0.656 —
segment0	0.985 —	0.988 —	0.991 1,2
vehicle0	0.933 3	0.934 3	0.916 —
vehicle3	0.613 —	0.616 3	0.580 —
vowel0	0.720 3	0.759 1,3	0.657 —
winequality-red-4	0.117 —	0.136 1	0.136 1
wisconsin	0.940 —	0.948 1	0.959 1,2
yeast-1-4-5-8_vs_7	0.133 —	0.136 —	0.136 —
yeast3	0.605 —	0.673 1,3	0.658 1
yeast4	0.283 —	0.277 —	0.281 —
yeast5	0.423 —	0.457 1,3	0.449 1
średnia ranga	1.45 —	2.325 1	2.225 1

Tabela 4: Wyniki – klasyfikator bazowy gaussianNB

zbiór danych	AdaBoost – 1	Bagging – 2	RandomSubspace – 3
ecoli1	0.432 —	0.601 —	0.572 —
glass0	0.368 —	0.609 1	0.609 1
glass1	0.482 —	0.602 1	0.602 1
glass4	0.425 —	0.380 —	0.390 —
glass6	0.779 —	0.787 —	0.809 —
haberman	0.352 —	0.445 —	0.427 —
iris0	1.000 —	1.000 —	1.000 —
new-thyroid1	0.933 2,3	0.902 —	0.887 —
page-blocks-1-3_vs_4	0.701 2,3	0.479 —	0.479 —
pima	0.483 —	0.654 1	0.656 1
segment0	0.853 2,3	0.615 3	0.548 —
vehicle0	0.641 —	0.563 —	0.600 —
vehicle3	0.350 —	0.495 1	0.490 1
vowel0	0.324 —	0.598 1	0.593 1
winequality-red-4	0.084 —	0.110 1	0.110 1
wisconsin	0.840 —	0.945 1	0.959 1,2
yeast-1-4-5-8_vs_7	0.069 —	0.091 —	0.091 —
yeast3	0.228 —	0.247 3	0.236 —
yeast4	0.106 —	0.075 —	0.073 —
yeast5	0.433 2,3	0.219 3	0.212 —
średnia ranga	1.75 —	2.3 —	1.95 —

Literatura

[1] M. Wozniak, (2014) Hybrid Classifiers, Methods of Data, Knowledge, and ClassifierCombination. Springer-Verlag Berlin Heidelberg

[2] Haibo He, & Garcia, E. A. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284.

[3] Opitz D. & Maclin R, (1999) Popular Ensemble Methods: An Empirical Study. Journal of Artificial Intelligence Research 11 169-198.

[4] N. V. Chawla, et al. (2002) SMOTE: Synthetic Minority Over-sampling Technique Journal Of Artificial Intelligence Research, Volume 16,321-357.

[5] Pedregosa et al. (2011), Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830.

[6] Chinchor, N. (1992). MUC-4 evaluation metrics. Proceedings of the 4th Conference on Message Understanding - MUC4 '92.

[7] J. Alcalá-Fdez, et al.(2011) KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 255-287

[8] Atom https://atom.io/ [dostęp 31.02.2022]

[9] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

[10] Lemaitre G. et. al. (2017) Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, Volume 18, Number 17, 1-5

[11] Pauli Virtanen, et al. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

[12] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020).

[13] Jeff Reback, et al. (2020). pandas-dev/pandas: Pandas 1.0.3 (Version v1.0.3). Zenodo.

[14] Tabulate. Pretty-print tabular data in Python, a library and a command-line utility. https://github.com/astanin/python-tabulate [dostęp 31.02.2022]