Przegląd metod próbkowania danych

Stanisław Kaźmierczak

Motywacja

- Stwórzmy wiele zestawów danych uczących, na każdym z nich wytrenujmy model i zagregujmy rezultaty.
- Różnorodność danych treningowych zmniejsza korelację modeli.
- Niska (relatywnie) korelacja modeli jest obok siły poszczególnych modeli składowych, głównym czynnikiem wpływającym na końcową jakość zespołu modeli.
- Uwaga: inną motywację stanowi techniczna potrzeba samplowania danych strumieniowych.

Bootstrap aggregating

- **b**ootstrap **agg**regat**ing** → **bagging**
- D – zbiór treningowy, n – rozmiar zbioru
- Tworzymy m nowych zestawów treningowych D_i losując z powtórzeniami n' obserwacji.
- W klasycznej próbie bootstrapowej $n' = n$.
- Subagging: $n' < n$.
- Cel: znalezienie kompromisu między zróżnicowaniem zbiorów oraz ilością unikalnych obserwacji uczących w poszczególnych zbiorach.

Liczba unikalnych elementów (1)

- Prawdopodobieństwo nie wybrania danego elementu w żadnym z n losowań:

$$\left(1 - \frac{1}{n}\right)^n$$

- Prawdopodobieństwo wybrania danego elementu przynajmniej raz:

$$1 - \left(1 - \frac{1}{n}\right)^n$$

- Oczekiwana liczba unikalnych obserwacji:

$$E(U) = n \cdot \left[1 - \left(1 - \frac{1}{n}\right)^n\right]$$

Liczba unikalnych elementów (2)

- Wartości oczekiwane dla wybranych n:

$$n = 10\text{: } E(U) \approx 6.51$$

$$n = 100\text{: } E(U) \approx 63.40$$

$$n = 1000\text{: } E(U) \approx 632.30$$

$$n = 10000\text{: } E(U) \approx 6321.39$$

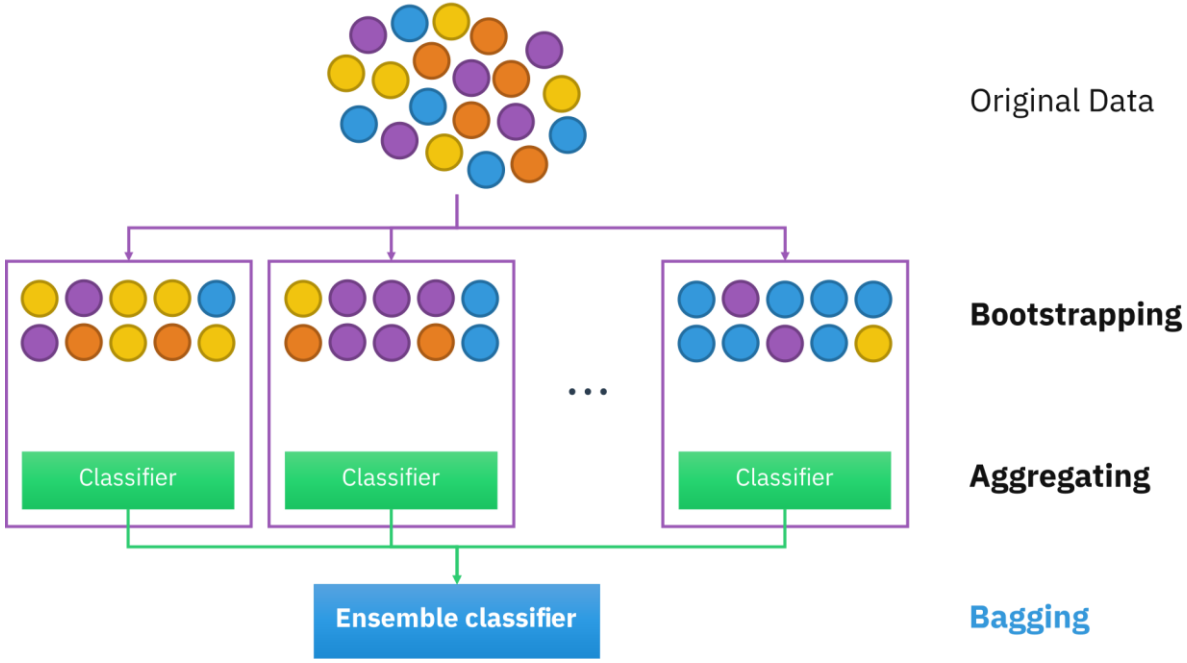
- Przybliżenie dla dużych n:

$$\lim_{n\to\infty} \left(1 - \tfrac{1}{n}\right)^n = e^{-1}$$

$$E(U) \approx n \cdot \left(1 - \frac{1}{e}\right) = n \cdot 0.632$$

Procedura baggingu

Najbardziej znaną realizacją baggingu jest las losowy (zbiór drzew decyzyjnych uczonych na próbach bootstrap), ale w ogólności może być stosowany do dowolnych algorytmów składowych.



[1]

Zbiór *Out-of-bag* (OOB)

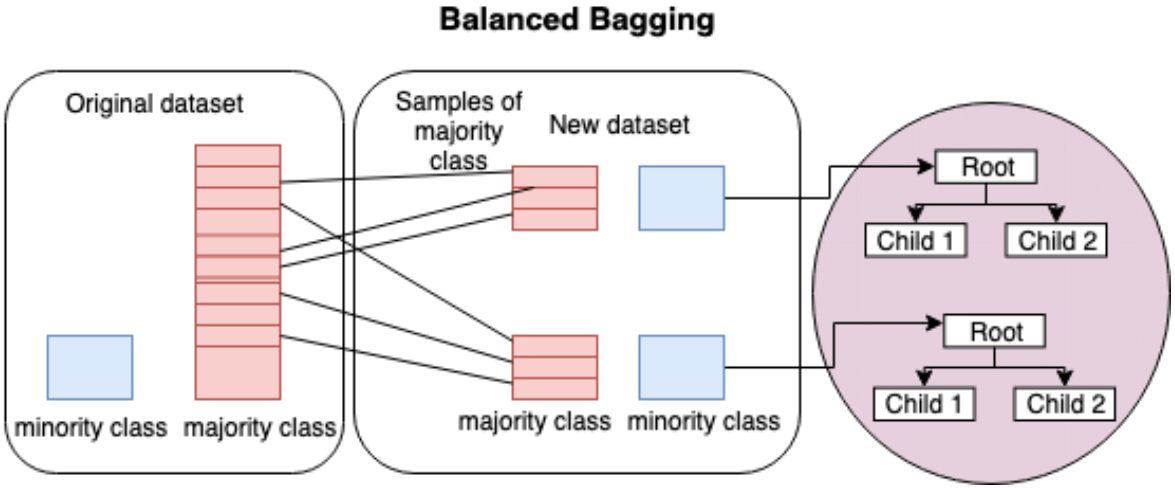
- OOB dla danego drzewa jest zbiorem obserwacji nie użytych do jego trenowania.
- Służą one obliczenia tzw. *OOB Error.*
- Każda obserwacja predykowana jest przez drzewa, do treningu których nie została użyta (takich drzew jest ok. 36%).
- Finalna predykcja dla danej obserwacji realizowana jest przy pomocy głosowania większościowego powyższych drzew.
- Dzięki temu nie potrzeba wydzielać dodatkowego zbioru testowego/walidacyjnego.
- Spodziewamy się, że błąd całego lasu losowego będzie nie większy niż błąd OOB.

Pasting

- W odróżnieniu od baggingu, próbki są losowane bez powtórzeń.
- Mniej próbek uczących, szybszy trening.
- Różnorodność ograniczona jedynie do podzbiorów unikalnych obserwacji.
- Wyniki na zbiorach benchmarkowych nie faworyzują baggingu nad pastingiem (lub odwrotnie)
  - Są istotnie zależnie od zbiorów danych i konfiguracji eksperymentów.

Balanced bagging

- W każdej próbce bootstrapowej klasy są zrównoważone.
- Exactly balanced bagging: pobierane są wszystkie dostępne obserwacje klasy mniejszościowej, undersampling klasy większościowej.
- Over-bagging: klasyczny bootstrap sampling dla klasy większościowej i oversampling klasy mniejszościowej.



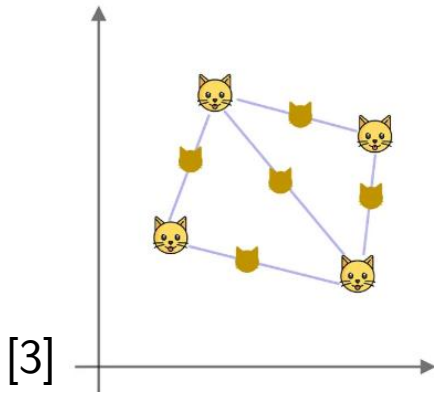
[2]

SMOTEBagging

- Algorytm
  1. Dla wylosowanego punktu A (z klasy mniejszościowej) znajdź k najbliższych sąsiadów tej samej klasy.
  2. Dla każdego ze znalezionych sąsiadów (ozn. B) wygeneruj obserwację C na odcinku AB:

$$C = A + \lambda \cdot (B - A)$$

  gdzie λ jest losową wartością z przedziału $(0, 1)$.
- Stosuje się modyfikacje, w których tworzone obserwacje nie muszą należeć do klasy mniejszościowej.
- Można spotkać też podejście, w którym interpolacja nie zachodzi między sąsiadami tej samej klasy (efektem są *smooth labels*)



[3]

Borderline-SMOTE Bagging (1)

Suppose that the whole training set is T, the minority class is P and the majority class is N, and

$$P = \{p_1, p_2, ..., p_{pnum}\}, N = \{n_1, n_2, ..., n_{nnum}\}$$

where $pnum$ and $nnum$ are the number of minority and majority examples. The detailed procedure of borderline-SMOTE1 is as follows.

Step 1. For every $p_i (i = 1,2,..., pnum)$ in the minority class P, we calculate its m nearest neighbors from the whole training set T. The number of majority examples among the m nearest neighbors is denoted by $m' (0 \le m' \le m)$.

Step 2. If $m' = m$, i.e. all the m nearest neighbors of p_i are majority examples, p_i is considered to be noise and is not operated in the following steps. If $m/2 \le m' < m$, namely the number of p_i 's majority nearest neighbors is larger than the number of its minority ones, p_i is considered to be easily misclassified and put into a set DANGER. If $0 \le m' < m/2$, p_i is safe and needs not to participate in the follows steps.

Step 3. The examples in DANGER are the borderline data of the minority class P, and we can see that $DANGER \subseteq P$. We set

$$DANGER = \{p'_1, p'_2, ..., p'_{dnum}\}, \quad 0 \le dnum \le pnum$$

For each example in *DANGER*, we calculate its k nearest neighbors from P.

[8]

Borderline-SMOTE Bagging (2)

Step 4. In this step, we generate $s \times dnum$ *synthetic positive examples from the data in DANGER, where s is an integer between 1 and k . For each* p'_i *, we randomly select s nearest neighbors from its k nearest neighbors in P. Firstly, we calculate the differences,* dif_j $(j = 1,2,...,s)$ *between* p'_i *and its s nearest neighbors from P , then multiply* dif_j *by a random number* r_j $(j = 1,2,...,s)$ *between 0 and 1, finally, s new synthetic minority examples are generated between* p'_i *and its nearest neighbors:*

$$synthetic_j = p'_i + r_j \times dif_j, \qquad j = 1,2,...,s$$

We repeat the above procedure for each p'_i in *DANGER* and can attain $s \times dnum$ synthetic examples.

[8]

- Wersja Borderline-SMOTE2 uwzględnia również sąsiadów klasy przeciwnej.
- Nowa obserwacja generowana jest na odcinku łączącym klasę mniejszościową z sąsiadem klasy przeciwnej w losowym miejscu pomiędzy obserwacją klasy mniejszościowej i połową odcinka.

ADASYN (1)

[Algorithm - ADASYN]

Input

(1) Training data set D_{tr} with m samples $\{\mathbf{x_i}, y_i\}$, $i = 1, ..., m$, where $\boldsymbol{x}_i$ is an instance in the n dimensional feature space $\boldsymbol{X}$ and $y_i \in Y = \{1, -1\}$ is the class identity label associated with $\boldsymbol{x}_i$. Define m_s and m_l as the number of minority class examples and the number of majority class examples, respectively. Therefore, $m_s \leq m_l$ and $m_s + m_l = m$.

Procedure

(1) Calculate the degree of class imbalance:

$$d = m_s/m_l \tag{1}$$

where $d \in (0,\ 1]$.

(2) If $d < d_{th}$ then (d_{th} is a preset threshold for the maximum tolerated degree of class imbalance ratio):

(a) Calculate the number of synthetic data examples that need to be generated for the minority class:

$$G = (m_l - m_s) \times \beta \tag{2}$$

Where $\beta \in [0,\ 1]$ is a parameter used to specify the desired balance level after generation of the synthetic data. $\beta = 1$ means a fully balanced data set is created after the generalization process.

[4]

ADASYN (2)

(b) For each example $\boldsymbol{x_i} \in minorityclass$, find K nearest neighbors based on the Euclidean distance in n dimensional space, and calculate the ratio r_i defined as:

$$r_i = \Delta_i / K, \quad i = 1, ..., m_s \tag{3}$$

where Δ_i is the number of examples in the K nearest neighbors of $\boldsymbol{x}_i$ that belong to the majority class, therefore $r_i \in [0,\ 1]$;

(c) Normalize r_i according to $\hat{r}_i = r_i / \sum_{i=1}^{m_s} r_i$, so that $\hat{r}_i$ is a density distribution ($\sum_i \hat{r}_i = 1$)

(d) Calculate the number of synthetic data examples that need to be generated for each minority example $\boldsymbol{x}_i$:

$$g_i = \hat{r}_i \times G \tag{4}$$

where G is the total number of synthetic data examples that need to be generated for the minority class as defined in Equation (2).

(e) For each minority class data example $\boldsymbol{x}_i$, generate g_i synthetic data examples according to the following steps:

Do the **Loop** from 1 to g_i:

(i) Randomly choose one minority data example, $\boldsymbol{x}_{zi}$, from the K nearest neighbors for data $\boldsymbol{x}_i$.

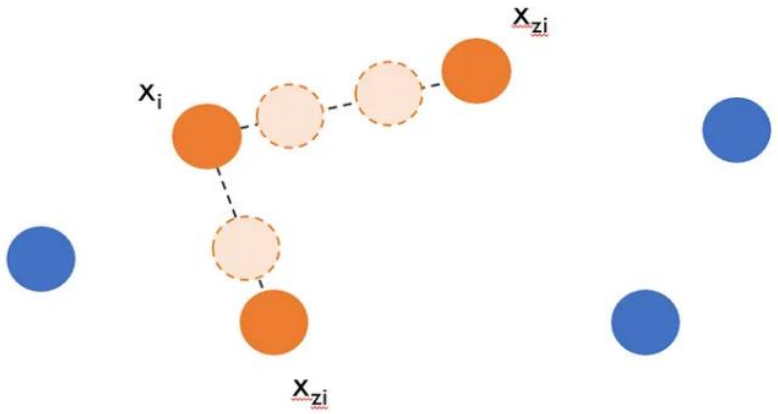
(ii) Generate the synthetic data example:

$$\boldsymbol{s_i} = \boldsymbol{x_i} + (\boldsymbol{x_{zi}} - \boldsymbol{x_i}) \times \lambda \tag{5}$$

where $(\boldsymbol{x_{zi}} - \boldsymbol{x_i})$ is the difference vector in n dimensional spaces, and λ is a random number: $\lambda \in [0,\ 1]$.

End **Loop**

[4]



[6]

ADASYNBagging

Algorithm 1 ADASYNBagging Algorithm

Input: Imbalanced Data, $D_{Imbalance}$, No. of Iterations: k, & C4.5 as base classifier.
Output: Ensemble model, M^*
Method:

1: divide $D_{Imbalance}$ into $D_{Majority}$ and $D_{Minority}$;
2: **for** i = 1 to k **do**
3: create bootstrap sample, $D_{sampled_Majority}$ from $D_{Majority}$, while $D_{Minority}$ remains untouched;
4: create $D_{over-sampled-minority}$ from $D_{Minority}$ using ADASYN;
5: marge $D_{over-sampled-minority}$ with $D_{sampled_Majority}$ to get $D_{balanced-over-sampled}$;
6: build a classifier, M_i from $D_{balanced-over-sampled}$;
7: **end for**

Classify a new instance, x_{New} using ensemble model, M^*:
Classify x_{New} by each $M_i \in M^*$;
Return the majority vote;

[7]

RSYNBagging

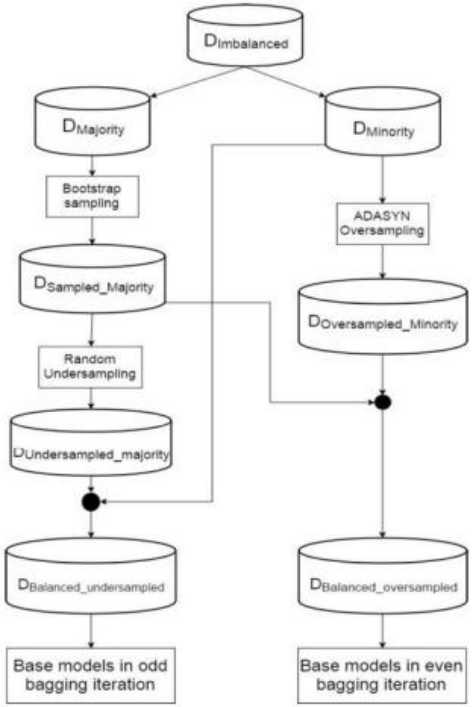
- Główna wada oversamplingu: duplikaty zwiększające prawdopodobieństwo przeuczenia.
- Główna wada undersamplingu: ryzyko utraty informacji.
- Koncept: połowa estymatorów uczona z użyciem ADASYN, a połowa z użyciem undersamplingu.

Algorithm 2 RSYNBagging Algorithm

Input: Imbalanced Data, $D_{Imbalance}$, No. of Iterations: k, & C4.5 as base classifier.
Output: Ensemble model, M^*
Method:

1: divide $D_{Imbalance}$ into $D_{Majority}$ and $D_{Minority}$;
2: **for** i = 1 to k **do**
3: create bootstrap sample, $D_{sampled_Majority}$ from $D_{Majority}$, while $D_{Minority}$ remains untouched;
4: **if** i is odd **then**
5: create $D_{under-sampled-majority}$ from $D_{sampled_Majority}$ using Random Under Sampling;
6: marge $D_{under-sampled-majority}$ with $D_{Minority}$ to get $D_{balanced-under-sampled}$;
7: build a classifier, M_i from $D_{balanced-under-sampled}$;
8: **else**
9: create $D_{over-sampled-minority}$ from $D_{Minority}$ using ADASYN;
10: marge $D_{over-sampled-minority}$ with $D_{Majority}$ to get $D_{balanced-over-sampled}$;
11: build a classifier, M_i from $D_{balanced-over-sampled}$;
12: **end if**
13: **end for**

Classify a new instance, x_{New} using ensemble model, M^*:
Classify x_{New} by each $M_i \in M^*$;
Return the majority vote;



[5, 7]

Wyniki ilościowe dla SMOTE-, ADASYN-, RSYNBagging

TABLE II: Average AUROC Comparison

Dataset	Under Bagging	SMOTE Bagging	ADASYN Bagging	RSYN Bagging
pageblocks-13vs4	0.989	0.973	0.984	**0.994**
ecoli-0-3-4-7_vs_5-6	0.887	0.845	0.886	**0.914**
poker-9_vs-_7	**0.621**	0.547	0.617	0.597
yeast-0-2-5-7-9_vs_3-6-8	0.885	0.882	0.890	**0.895**
ecoli-0-3-4-_vs_5	0.893	0.881	0.912	**0.934**
spambase	0.915	0.911	0.912	**0.924**
new-thyroid1	0.977	**0.985**	0.983	0.982
glass-0-1-5-_vs_2	**0.691**	0.639	0.681	0.652
vehicle2	0.952	0.951	0.957	**0.962**
pima	0.659	0.661	0.656	**0.671**
abalone19	0.776	0.787	**0.798**	0.784

[7]

Roughly balanced bagging (1)

- Różnicuje liczbę obserwacji klasy większościowej i mniejszościowej w poszczególnych samplach.
  - Choć uśredniona liczba obserwacji po wszystkich samplach jest taka sama.
- Zwiększa to różnorodność modeli
- Negatywny rozkład dwumianowy modeluje prawdopodobieństwo wystąpienia m porażek przed wystąpieniem n-tego sukcesu; q – prawdopodobieństwo sukcesu
- Dla $q = 0.5$, wartość oczekiwana wynosi m.

$$p(m|n) = \binom{m+n-1}{n} q^n (1-q)^m$$

[8]

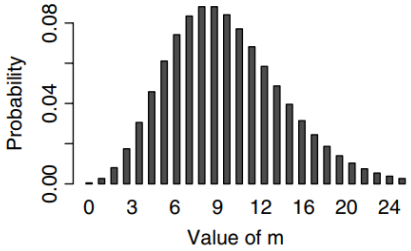


Fig. 2 Distribution of the negative binomial distribution ($q = 0.5, n = 10$).

Roughly balanced bagging (2)

- Inputs:
  D is the training data set
  L is the algorithm for base learners
  K is the number of base learners
  x_i is an example drawn from the test set

- Build Roughly Balanced Bagging Model (D, L, K):
  Divide D into a negative set D^{neg} and a positive set D^{pos}
  Set N^{pos} as the size of D^{pos} (i.e. $|D^{pos}|$)
  For $k = 1$ to K
    Draw N_k^{neg} from the negative binomial distribution (3.1) with $n = N^{pos}$ and $q = 0.5$
    Let D_k^{neg} be N_k^{neg} examples sampled from D^{neg} with or without replacement
    Let D_k^{pos} be N^{pos} examples sampled from D^{pos} with or without replacement
    Build a base learner model $f^k(x)$ by L based on $D_k^{neg} \cup D_k^{pos}$
  Combine all $f^k(x)$ into the aggregated model $f^A(x)$
  Return $f^A(x)$

- Predict ($f^A(x)$, x_i, y):
  Calculate $p^A(y|x_i) = \frac{1}{K} \sum_{k=1}^{K} p^k(y|x_i)$ for all y
  Let $\hat{y} = \text{argmax}_{y \in \mathcal{L}}\, p^A(y|x_i)$
  Return $\hat{y}$

[8]

Roughly balanced bagging (3)

Table V. The experimental results on the benchmark data sets (taken from the UCI repository).

Data set	Algorithm	AUC	MSE	ISE	F-measure	G-mean	Accuracy
Diabetes	RB Bagging (K = 100)	83.5	0.162	0.0490*	69.2	76.2	76.3
	RB Bagging w/replace (K = 100)	**83.7**	0.161	**0.0487**	**70.4**	**77.2**	**77.5**
	Exact Balanced Bagging (K = 100)	82.7	0.173	0.0541	67.5	74.7	74.1
	Original bagging (K = 100)	83.4	**0.157**	0.0513	63.8	71.2	76.4
	C4.5 (pruned)	76.6	0.193	0.106	59.0	67.6	73.3
	AdaBoost (K = 100)	79.6	0.244	0.239	63.7	71.4	75.1
	AdaBoost (K = 200)	78.9	0.255	0.251	63.0	70.9	74.1
	RIPPER (Optimize = 2)	69.5	0.193	0.0854	58.6	67.1	73.6
	RIPPER (Optimize = 10)	71.2	0.188	0.0885	61.5	69.5	74.9
Breast	RB Bagging (K = 100)	**98.8**	0.0359	0.0182	94.1	95.7	95.8
	RB Bagging w/replace (K = 100)	98.7*	0.0358	**0.0181**	94.1	95.7	95.8
	Exact Balanced Bagging (K = 100)	98.6	0.0413	0.0257	93.1	95.1	95.1
	Original bagging (K = 100)	98.3	0.0374	0.0198*	93.7	95.4	95.6
	C4.5 (pruned)	96.4	0.0467	0.0332	92.2	94.1	94.6
	AdaBoost (K = 100)	98.3	**0.0293**	0.0288	**95.7**	**97.0**	**97.0**
	AdaBoost (K = 200)	98.2	0.0301*	0.0301	**95.7**	**97.0**	**97.0**
	RIPPER (Optimize = 2)	92.7	0.0619	0.0525	89.6	91.5	93.0
	RIPPER (Optimize = 10)	92.9	0.0574	0.0522	90.8	92.3	93.8
German	RB Bagging (K = 100)	77.3	0.188	0.0431	77.7	**70.1**	71.1
	RB Bagging w/replace (K = 100)	**78.1**	0.186	**0.0415**	77.3	69.9*	70.8
	Exact Balanced Bagging (K = 100)	76.0	0.208	0.0566	71.8	67.9	65.9
	Original bagging (K = 100)	76.9	**0.173**	0.0566	82.5	60.4	74.0
	C4.5 (pruned)	66.2	0.227	0.146	80.2	56.3	70.8
	AdaBoost (K = 100)	71.0	0.249	0.245	**83.0**	62.6	74.9*
	AdaBoost (K = 200)	70.0	0.249	0.248	82.9*	63.4	**75.0**
	RIPPER (Optimize = 2)	63.5	0.194	0.0723	81.5	58.4	72.6
	RIPPER (Optimize = 10)	63.9	0.197	0.0758	80.6	59.9	71.8
E-Coli-4	RB Bagging (K = 100)	94.7	0.0877	0.0365	62.7	**89.3**	87.5
	RB Bagging w/replace (K = 100)	**95.7**	0.0871	0.0365	61.2	88.9	86.9
	Exact Balanced Bagging (K = 100)	94.0	0.103	0.0516	58.5	88.3	85.7
	Original bagging (K = 100)	94.3	**0.0460**	**0.0215**	**65.3**	74.1	93.8
	C4.5 (pruned)	81.7	0.0523	0.0449	63.7*	69.5	**94.4**
	AdaBoost (K = 100)	93.7	0.0680	0.0679	62.2	70.1	93.2
	AdaBoost (K = 200)	93.3	0.0775	0.0756	55.4	65.7	92.0
	RIPPER (Optimize = 2)	77.1	0.0619	0.0478	57.3	67.8	92.6
	RIPPER (Optimize = 10)	78.8	0.0671	0.0513	61.6	74.7	91.7

[8]


Q&A

Źródła (1)

1. https://en.wikipedia.org/wiki/Bootstrap_aggregating#/media/File:Ensemble_Bagging.svg
2. Barros, T. M., Souza Neto, P. A., Silva, I., & Guedes, L. A. (2019). Predictive models for imbalanced data: A school dropout perspective. *Education Sciences*, 9(4), 275.
3. Iriawan, N., Fithriasari, K., Ulama, B. S. S., Suryaningtyas, W., Pangastuti, S. S., Cahyani, N., & Qadrini, L. (2018, November). On The Comparison: Random Forest, SMOTE-Bagging, and Bernoulli Mixture to Classify Bidikmisi Dataset. In *2018 International Conference on Computer Engineering, Network and Intelligent Multimedia* (CENIM) (pp. 137-141). IEEE.
4. He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* (pp. 1322-1328). IEEE.

Źródła (2)

5. Dey, I., & Pratap, V. (2023). A comparative study of SMOTE, borderline-SMOTE, and ADASYN oversampling techniques using different classifiers. In 2023 *3rd international conference on smart data intelligence* (ICSMDI) (pp. 294-302). IEEE.
6. https://medium.com/@ruinian/an-introduction-to-adasyn-with-code-1383a5ece7aa
7. Ahmed, S., Mahbub, A., Rayhan, F., Jani, R., Shatabda, S., & Farid, D. M. (2017). Hybrid methods for class imbalance learning employing bagging with sampling techniques. In *2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)* (pp. 1-5). IEEE.
8. Hido, S., Kashima, H., & Takahashi, Y (2009). Roughly balanced bagging for imbalanced data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(5-6), 412-426.
9. Han, H., Wang, W. Y., & Mao, B. H. (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing* (pp. 878-887). Berlin, Heidelberg: Springer Berlin Heidelberg.