A Classification Model for Modeling Online Articles

Rula Alhalaseh[1], Ali Rodan[2], and Azmi Alazzam[2(✉)]

[1] Department of Information Technology, University of Jordan, Amman, Jordan
rula.alhalaseh@gmail.com
[2] Computer Information Science, Higher Colleges of Technology,
Al Ain Women's Campus, Al Ain, Abu Dhabi, UAE
{arodan,aalazzam}@hct.ac.ae

Abstract. Due to the constant evolvement of the web and the viral spread of online news on social media, predicting the popularity of a news article became a topic of interest to many categories of people ranging from marketing personnel to politicians. In this paper, we focus on comparing four classification algorithms on a dataset consisting of 39000 news articles taken from Mashable website. The articles were classified into two classes: Popular and not popular. Four different machine learning algorithms were used for classification of the data (KNN, Naïve bayes, Adaboost, and decision tree). Finally, the four classification methods were compared with each other.

Keywords: Ada Boost · KNN · Naïve Bayes · Decision Tree

1 Introduction

In recent years, with the viral evolvement of social media, the sharing, commenting on and reading of various kinds of articles, including news and articles of social or political nature, has become the center of people's daily entertainment. As tons of news, rumors and stories are published on a daily basis, there comes a need for predicting whether a news piece can go viral before it is published. Predicting news popularity became a trendy field of research for researchers, authors and advertisers to build their strategies as well as make it reach as many individuals as possible. This will also help in extracting and implementing the features contributing to a viral article outspread. Also some politicians are concerned of the influence of news articles on the population and the effects from spreading such news. In this paper, the dataset used is a real-world dataset taken from UCI Machine Learning Repository [1] that collected over than 39000 articles from Mashable [2] website.

The dataset has various informative features [3], we also intend to compare and analyze the performance of several machine learning algorithms to predict the popularity of news articles. Measurement of popularity is known as the number of times an article gets shared, liked and commented on. For the popularity measure we adopt a common binary task to classify the articles into popular and unpopular and then use the machine

C. Brito-Loeza et al. (Eds.): ISICS 2020, CCIS 1187, pp. 65–77, 2020.
https://doi.org/10.1007/978-3-030-43364-2_7

learning algorithms to build a classification model that can used to classify new articles based on some features.

There are two main prediction approaches to measure popularity [4].

The first approach is to use features that are only known and observed after publishing an article and the second approach does not use these features. The first approach is more common. An Example of the first approach can be found in [5] where the evolution of user generated content popularity is discussed. Another proposed methodology for predicting online contents popularity in a more precise manner rather than attempting to infer the possibility that a content will be popular can be found in [6].

The statistical analysis of the time of user reaction to a newly opened a discussion thread online which was made on the popular news website Slashdot [7, 8]. It also performed a characterization that enabled predicting intermediate and long-term user behavioral pattern with and acceptable result of precision.

Predicting the popularity of online content was elaborated in [9, 10]. In [11] the authors proposed a framework for modeling and predicting the popularity of online contents that aimed to infer the likelihood with which the content will be popular.

Since the prediction task is easier to implement, higher accuracies in prediction are often achieved. Popularity prediction of articles that do not use features which is not a commonly used approach as a low performance in prediction is expected. And moreover using the features as in the first approach are said to improve the content before having it published.

2 Literature Review

The work mentioned in [3] consists of a robust evaluation of five state of the art models for classification of around 39 thousand articles that were labeled and collected from Mashable website [2]. Experiments on Random forest in [3] conducted the best result having a discrimination power of 73% for binary classification.

A research to address the prediction task both as a regression and a classification problem as well as to predict the number of news tweets was discussed in [12]. The paper illustrated that even though predicting the exact number of tweets may have a high error percentage, there is a possibility for predicting ranges of popularity on news tweets with 84% overall accuracy. Furthermore, it considered four types of features (news source, category of the article, subjectivity language used, and names mentioned in the article) to predict the tweets number that has the article mentioned in them. Three popularity classes were studied which ranged between 1–20 tweets, 20–100 tweets and more than 100, discarding the articles with no tweets.

Another study that used news tweets proposed the passive aggressive algorithm to predict how many times a news link tweet will be retweeted [13]. The study discussed also the some social features like how many users are following the user tweeting, which can determine the number of times an article will be retweeted. Moreover, this research noticed that the number of urls and hashtags could also boost the tweet to make it get to as many people as possible.

Xuandong et al. researched the topic of predicting whether a mashable news article will be viral or not by addressing two dimensions of the problem: multidimensional classification and numerical [14]. They applied linear regression, polynomial regression,

GAM with smoothing splines and Lasso to predict the exact amount of shares of a news article. In the paper, GAM with smoothing splines gave the best CV error which was 0.7649. In their paper, they used SVM, Random Forest and Bagging to predict popularity of the news, resulting into four categories for each news article and with Random forest giving best result of 50.4% accuracy in prediction.

The research work in [15] tested two binary classification tasks for prediction: popular and unpopular as well as appealing and non-appealing when compared to articles that were published on the same day used 10 English news outlets that related with one year. The paper used bag of words of the title and description, keywords and characteristics such as date of publishing combined with Support Vector Machine (SVM). The appealing task gave better accuracy results of 62–86% when compared with the popular and unpopular task which gave results ranging from 51–62%.

3 Methodology

In this section, the methodology of the classification algorithms is discussed. There are four classification algorithms that are discussed and implemented to classify the data collected from different articles. The goal of this study is to build a classification model with high accuracy that can be eventually used to predict the popularity of articles before a decision is made whether to publish them or not. The next four Subsect 3.1–3.6 will discuss and elaborate the methodology in details.

3.1 Dataset

The dataset obtained from [1] consists of over 39 thousand articles from Mashable [2] which is one of the largest well known news website. The data was retrieved and prepared by [3] on a 2 years period, from January, 7 2013 and January, 7 2015. Special occasion articles were discarded which was only a small portion of the dataset and did not follow the general structure of HTML as the processing of each occasion would require a specific parser. The collected data was donated to the UCI Machine Learning Repository [1] for public use. The processing and collection process of the Mashable [2] data in [3] was implemented in Python, while we are going to use Weka tool for our work. After preprocessing of the dataset the resulting articles were a total of 39 thousand data points with 60 features.

We summarized the work done on the Mashable [2] dataset before proceeding with the classification. The dataset classification considered 47 features in total that were extracted from html code. The features of the dataset are shown in Table 1 [3]. These features have different types: number-integer value, ration within [0, 1], a bool that can be either a 0 or 1 and a nominal value. Columns with (#) indicate the number of variables within each feature. The number of shares attribute which we will be working on in our paper was concluded in [3] by selecting a large list of characteristics that describe different aspects of the article and that were considered possibly relevant to influence the number of shares as we have the minimum, maximum and the average number of shares in various social networks in the dataset.

In this paper, a binary classifier is used. Two classes are considered popular and unpopular. If the article has more than 1400 shares it is considered as a popular article,

and otherwise it is considered as unpopular article. Thus, the classification algorithm will use the existing data to predict the classes based on the 47 attributes of this data.

The data will be divided into two parts: two thirds will be used for building (training the model) and one third for validation in order to avoid over fitting.

In the next subsections we discuss the data labeling process and the different algorithms that were used in building the classification model.

Table 1. Statistical measures of the articles in Mashable dataset [3]

Feature	Type (#)
Words	
-Number of words in the title	Num (1)
-Number of words in the article	Num(1)
-Average word length	Num (1)
-Rate of non-stop words	ratio (1)
-Rate of unique words	ratio (1)
-Rate of unique non-stop words	ratio (1)
Links	
-Number of links	Num (1)
-Number of Mashable article links	Num (1)
-Minimum, average and maximum number of shares of Links	Num(3)
Digital Media	
-Number of images	Num (1)
-Number of Videos	Num (1)
Time	
-Day of the Week	Nom (1)
-Published on a weekend?	Bool(1)

Feature	Type (#)
Words	
-Number of keywords	Num (1)
-Worst keyword (min./avg./max. shares)	Num (3)
-Average keyword (min./avg./max. shares)	Num (3)
-Best keyword (min./avg./max. shares)	Num (1)
-Article category (Mashable data channel)	Nom(1)
Natural Language Processing	
-Closeness to top 5 LDA topics	ratio (5)
-Title subjectivity	ratio (1)
-Article text subjectivity score and its absolute difference of 0.5	ratio (2)
-Title sentiment polarity	ratio (1)
-Rate of positive and negative words	ratio (2)
-Pos. words rate among non-neutral words	ratio (1)
-Neg. words rate among non-neutral words	ratio (1)
-Polarity of positive words (min./avg./max.)	ratio (3)
-Polarity of negative words (min./avg./max.)	ratio (3)
-Article text polarity score and its absolute difference to 0.5	ratio (2)

Target	Type (#)
-Number of article Mashable shares	Num (1)

3.2 Process of Labeling Classes and Evaluating Results

We added a new attribute named popularity and modified the dataset with the condition that articles having more than 1400 shares are to be labeled as popular 'pop' and other than that will be labeled as 'not-pop'. Excel function applied was:

Function=IF(BI2>1400, "pop", "not-pop")
We then observe that the classes are balanced on Weka as in the figure (Fig. 1)

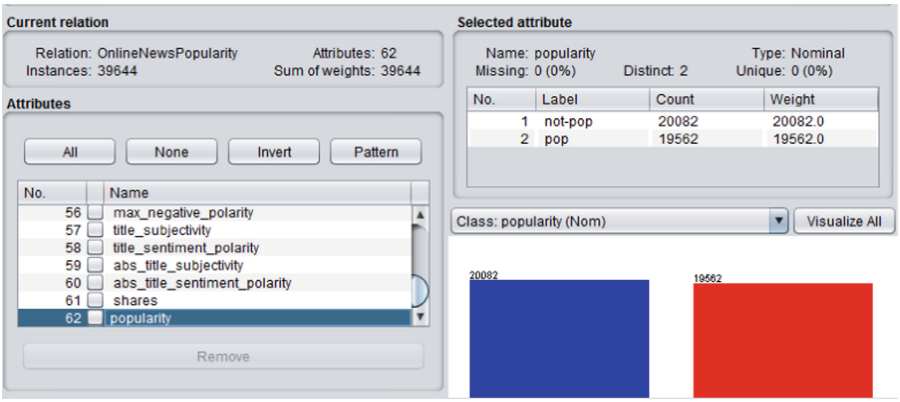

Fig. 1. Classified dataset based on popularity

The total number of attributes is 61 but we only used 47 as discussed before and also shown in Table 1. We used 10-fold cross-validation testing mode for all selected algorithms. 39644 class instances (rows) used in all algorithms.

3.3 AdaBoost

AdaBoost is short for "Adaptive Boosting", and defined as a machine learning meta-algorithm that was formulated by Yoav Freund and Robert Schapire [16]. Adaboost in conjunction with other learning algorithms are said to improve performance.

Weka AdaBoost M1 [20], which is a class for boosting nominal class classifier and can only tackle nominal class problems. Adaboost M1 was used in combination with J48 classifier.

The confusion matrix also known as an error matrix or a contingency table is a certain table layout which visualize the algorithm performance, every column stand for the instances in a forecasted class, while every row stands for the instances of the actual class. The confusion matrix for any classification algorithm is given by Table 2 [17].

Testing the models with just Accuracy and Sensitivity measures is not adequate to ensure that the classifications have reliable results, so we will use more measures for the model evaluation, which are as follow.

Table 2. Confusion matrix structure

	Actual value		
Predicted value		Positives	Negatives
	Positives	TP (True Positive)	FP (False Positive)
	Negatives	FN (False negative)	TN (True Negative)

To check the classification performance of the algorithm, different measures can be used. Some of the most common measures that can be calculated from the confusion matrix are:

Classification accuracy: The True rate of the model and it's measured as the summation of number for the correct classes divided by the total number.
Sensitivity: The true positive rate, measures the ratio of the actual positives. Sensitivity $= \text{tp}/(\text{tp} + \text{fn})$
specificity: also known as the true negative rate, measures the ratio of negatives. Specificity $= \text{tn}/(\text{tn} + \text{fp})$
Precision: measures the provided accuracy of a certain class that has been forecasted. Precision $= \text{tp}/(\text{tp} + \text{fp})$ [12]

The resulting confusion matrix, where 'A' stands for not-pop class and 'B' stands for popular class is shown in Table 3. Results of running Adaboost and remaining algorithms is detailed in the next section for comparison. It can be seen from the confusion Matrix of the AdaBoost algorithm that 23397 data points were correctly classified. This will give an accuracy of 0.59.

Table 3. Confusion matrix for AdaBoost classification

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	11878	8207
	Negatives(A)	8061	11501

3.4 K Nearest Neighbor K-NN

K-NN is a supervised learning algorithm and an easy algorithm that saves all available instances, and classifies them based on distance functions (similarity measure). K-NN has been used in estimation of statistics and recognition pattern as a non-parametric method. The K-NN classifies each instance based on its neighbors, and assigns each data point to the class with the most similarity gauged by a distance function like the

Euclidian distance function [18]. In our work we used K-NN with K = 1, where the data instance is simply assigned to the class of that single nearest neighbor. K-NN gave us the worst result among all the other tested algorithms when the value of K was 1. Experimenting further with K-NN by increasing the value of k eventually gave the best result among all the tested algorithms. However having a high value of k eventually started to give lower accuracy rate.

Weka KNN classifier scheme used is as follows with different values of K. The results for the performance measures are shown in Table 4. We can see from the table that the best results were obtained when K = 37. The confusion matrix for each value of is shown in Tables 15 and 16.

Table 4. K-NN performance measures for different value of K

K	Correctly classified	Incorrectly classified	Precision	Recall	Time(s) to build model	RMS
1	22681	16963	.572	.572	.01	.6541
3	23339	16305	.589	.589	.25	.5376
5	23860	15784	.602	.602	.08	.5098
7	24193	15451	.609	.646	.09	.4977
11	24514	15130	.619	.618	.11	.4882
15	24571	15073	.620	.620	.07	.4834
33	24881	14763	.628	.628	.01	.4775
35	24900	14744	.628	.628	.13	.4772
36	24894	14750	.628	.626	.12	.4769
37	**24918**	**14726**	**.629**	**.629**	**.01**	**.477**
38	24894	14750	.629	.628	.12	.4441
40	24843	14801	.628	.627	.01	.4768
50	24868	14776	.628	.625	.01	.4763
66	24867	14777	.628	.627	.06	.4758

Table 5. Confusion matrix for K-NN (K = 1)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	11838	8244
	Negatives (A)	8719	10843

Table 6. Confusion matrix for K-NN (K = 3)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	12357	7725
	Negatives (A)	8580	10982

Table 7. Confusion matrix for K-NN (K = 5)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	12743	7339
	Negatives (A)	8445	11117

Table 8. Confusion matrix for K-NN (K = 7)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13207	6875
	Negatives (A)	8255	11307

Table 9. Confusion matrix for K-NN (K = 11)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13207	6875
	Negatives (A)	8255	11307

Table 10. Confusion matrix for K-NN (K = 13)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13292	6790
	Negatives (A)	8283	11279

Table 11. Confusion matrix for K-NN (K = 33)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13419	6663
	Negatives (A)	8100	11462

Table 12. Confusion matrix for K-NN (K = 35)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13413	6669
	Negatives (A)	8075	11487

Table 13. Confusion matrix for K-NN (K = 37)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13404	6678
	Negatives (A)	8048	11514

Table 14. Confusion matrix for K-NN (K = 40)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13816	6266
	Negatives (A)	8510	11052

Table 15. Confusion matrix for K-NN (K = 50)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13701	6381
	Negatives (A)	8420	11142

Table 16. Confusion matrix for K-NN (K = 66)

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	13567	6515
	Negatives (A)	8262	11300

The accuracy as calculated from Table 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16 is shown in Fig. 2. We can see how the accuracy changes when we increase the value of K in K-NN, best accuracy was at K = 37. The X-Axis showing the k value and the y-Axis showing the accuracy.

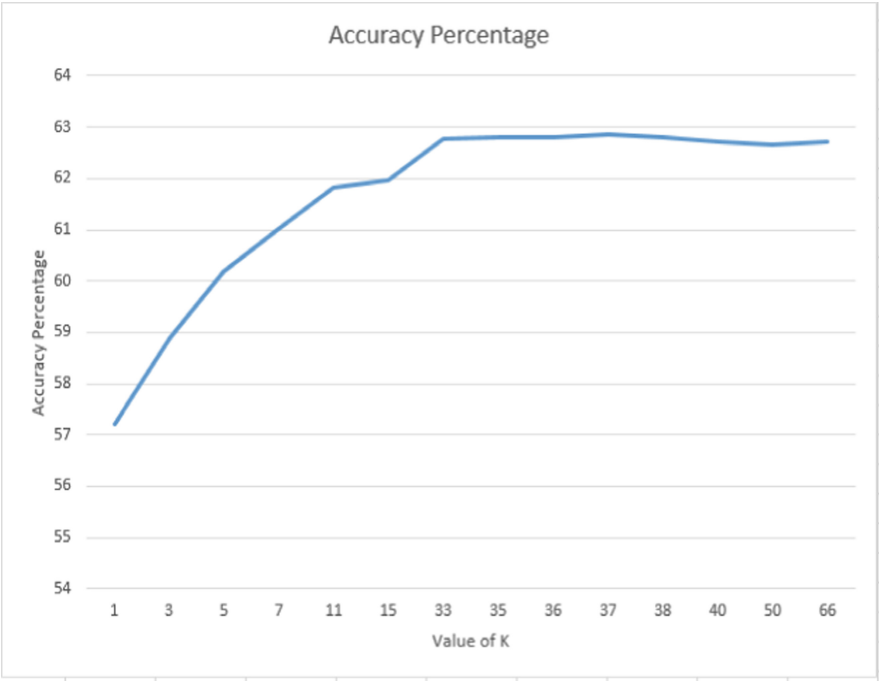

Fig. 2. K-NN accuracy for different values of K

3.5 J48 Decision Tree (Pruned Tree)

Decision Tree structure is a tree-like flowchart in which the internal node perform a test on the attribute. The branch stands for a consequence of the test, and the leaf node represents the class label. The root node in a tree is the topmost node. The Decision tree uses different algorithms. J48 is an algorithm for generating a decision tree developed

by Ross Quinlan [19]. J48 is usually used for classification. The J48 was implemented using Weka.

We can see by the resulting confusion matrix (Table 17) that the accuracy was similar as when we used Adaboost with J48 to classify popularity:

Table 17. Confusion matrix for J48 decision tree

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	11878	8204
	Negatives (A)	8061	11501

3.6 Naïve Bayes

Which is a Machine Learning Algorithm that need to be trained for supervised learning tasks like classification and prediction or for unsupervised learning tasks like clustering.

The resulting confusion matrix, where 'A' stands for not-popular class and 'B' stands for popular class. We notice a relatively high error rate Table 18.

Table 18. Confusion matrix for Naïve Bayes

	Actual value		
Predicted value		Positives (B)	Negatives (A)
	Positives (B)	18714	1368
	Negatives (A)	17201	2361

4 Results and Summary

Four different Algorithms have been used to classify the articles' data based on their popularity. Table 19 shows a comparison of these algorithms.

From this comparison we can see that we got the same number of correctly classified and incorrectly classified instances when running Adaboost using J48 classifier and J48 alone. While the AdaBoost gave slightly less root mean squared error and took more time to build.

Naïve Bayes gave the worst result amongst all with the least number of correctly classified instances and highest root mean squared error.

KNN gave the best result among all the algorithms when $K = 37$ with accuracy which proved better than the random forest and SVM in a previous work [3]. K-NN was also the fastest when comparing time it took to build the model. KNN also gave the least root mean squared error value among the other algorithms compared.

Table 19. Comparison of the 4 classification algorithms

Algorithm	Correctly classified	Incorrectly classified	Precision	Recall	Time to build model (S)	RMS error
AdaBoost using j48 classifier	23379	16265	0.590	0.590	42.61	0.6114
KNN K = 37	24918	14726	0.629	0.629	0.01	0.477
J48	23379	16265	0.590	0.590	39.53	0.6149
NB	21075	18569	0.576	0.532	0.59	0.6689

5 Conclusion

In this paper, we have introduced and discussed four machine learning algorithms that are usually used in supervised learning. The four algorithms were: Adaptive boosting, K-NN, Decision Trees and Naïve Bayes. The data classified in this work is related to some articles that have 47 attributes and one target class (popular or unpopular). The article that had more than 1400 shares was considered as popular. We have seen that the K-NN with k = 37 has the best performance among the four algorithms.

For future work we will consider splitting the articles into three target classes and perform the comparison between the mentioned and more classification algorithms such as SVM and Artificial Neural Network.

References

1. UCI machine learning repository. https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity. Accessed 16 Sept 2019
2. Mashable. http://mashable.com. Accessed 20 Sept 2019
3. Fernandes, K., Vinagre, P., Cortez, P.: A proactive intelligent decision support system for predicting the popularity of online news. In: Pereira, F., Machado, P., Costa, E., Cardoso, A. (eds.) EPIA 2015. LNCS (LNAI), vol. 9273, pp. 535–546. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23485-4_53
4. Tatar, A., Amorim, M., Fdida, S., Antoniadis, P.: A survey on predicting the popularity of web content. J. Internet Serv. Appl. **5**(1), 1–20 (2014)
5. Ahmed, M., Spagna, S., Huici, F., Niccolini, S.: A peek into the future: predicting the evolution of popularity in user generated content. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 607–616. ACM (2013)
6. Lee, J., Moon, S., Salamatian, K.: An approach to model and predict the popularity of online contents with explanatory factors. In: ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Canada, pp. 623–630 (2010)
7. Kaltenbrunner, A., Gomez, V., Lopez, V.: Description and prediction of Slashdot activity. In: Web Conference, LA-WEB 2007, pp. 57–66. IEEE, Latin American (2007)
8. SlashdotMedia: Slashdot: News for nerds, stuff that matters (2016). https://slashdot.org/. Accessed 11 Sept 2019
9. Szabo, G., Huberman, B.: Predicting the popularity of online content. Commun. ACM **53**(8), 80–88 (2010)

10. Tatar, A., Antoniadis, P., De Amorim, M., Fdida, S.: From popularity prediction to ranking online news. Soc. Network Anal. Min. **4**(1), 1–12 (2014)
11. Lee, J., Moon, S., Salamatian, K.: Modeling and predicting the popularity of online contents with cox proportional hazard regression model. Neurocomputing **76**(1), 134–145 (2012)
12. Roja, B., Asur, S., Huberman, B.: The pulse of news in social media: forecasting popularity. In: Proceedings of the 6th International AAAI Conference on Weblogs and Social Media, ICWSM (2012)
13. Sasa, P., Osborne, M., Lavrenko, V.: RT to Win! Predicting message propagation in Twitter. In: ICWSM, Spain(2011)
14. Xuandong, L., Hu, X., Fang, H.: Is your story going to spread like a virus? Machine learning methods for news popularity prediction. In: CS229 (2015)
15. Hensinger, E., Flaounas, I., Cristianini, N.: Modelling and predicting news popularity. Pattern Anal. Appl. **16**(4), 623–635 (2013)
16. Freund, Y., Schapire, R.: Decision-Theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
17. Stehman, S.: Selecting and interpreting measures of thematic classification accuracy. Remote Sens. Environ. **62**(1), 77–89 (1997)
18. Altman, N.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
19. Quinlan, J.: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco (1993)
20. Weka 3 - Data Mining with Open Source Machine Learning Software in Java, Cs.waikato.ac.nz (2016). http://www.cs.waikato.ac.nz/~ml/weka/. Accessed 14 Sept 2019