Stacking with Dynamic Weights on Base Models

Biswaroop Mookherjee and Abhishek Halder

1 Introduction

This paper is of methodological development of two new ways of stacking. Stacking is an ensemble technique which combines different classification techniques aiming correct classification rate higher than what the base techniques provide. Suppose, we have applied two classification techniques named CT1 and CT2 on a dataset where the objective is binary classification. Say, the base techniques CT1 and CT2 provide correct classifications of $x\%$ and $y\%$, respectively. Then, stacking is expected to provide correct classification higher than maximum of $x\%$ and $y\%$. We have proposed two new methods of stacking which perform better than the conventional way.

2 Literature Review

Stacking appears in the papers by Wolpert [1] and Breiman [2]. It is widely used by machine learning practitioners to get better classification by creating ensemble of multiple models based on different classification techniques.

In the conventional way, stacking is done by running a classification technique with outputs of the base (or primary) learners as independent variables. The target variable is kept same. Hence, one can think the overall structure as function of functions. Classification techniques are run on the dataset, and prediction of the classes is obtained. Such predicted classes go as independent variables in another model while the target variable remains same. Usually, the second-level modelling

B. Mookherjee (✉) · A. Halder
Tata Consultancy Services (TCS), Kolkata, India
e-mail: biswaroop9000@gmail.com

A. Halder
e-mail: abhishek.halder171@gmail.com

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2021

A. K. Laha (ed.), *Applied Advanced Analytics*, Springer Proceedings in Business and Economics, https://doi.org/10.1007/978-981-33-6656-5_15

is done using logistic regression. Finally, classification happens by the whole nested structure.

Stacking does not necessarily do well always because of its rigid structure of applying same set of weights on the base learners in all parts of the data. The weights are obtained from the second level of model. We have not found any procedure of stacking where the weights given on base learners vary in different parts of the data considering the performance of the base learners in different parts. Hence, we have developed methods which are narrated in Sects. 4 and 5.

3 Stacking by Conventional Way

In this conventional way, different models are prepared using different techniques. Then, the predicted classes by different models are used as predictors along with the observed class as target to run the upper-level model.

3.1 Steps of Stacking by Conventional Way

Step 1 Run k number of classification techniques $T1, T2, \ldots., Tk$ and build models.
Step 2 Take predicted classes as predictors along with the observed classes as values of the target variable
Step 3 Run a classification technique which is usually logistic regression on the dataset prepared in Step 2 to get a model
Step 4 The new observations are passed through the models prepared in Step 1 and 'k' number of outputs are obtained for each observation as 'k' number of techniques are there.
Step 5 The k outputs obtained from Step 4 are passed through the model prepared at Step 3 to get the final class prediction.

The *R* programming language functions that are used are 'glm' for logistic regression, 'lda' for linear discriminant analysis, rpart for decision tree, trainControl and train for k-nearest neighbors. Stacking by the conventional way is done by logistic regression.

4 Proposed Method I: Stacking Using Neighbourhood-Based Dynamic Weights

We propose a new way of stacking where the base learners do not get weights from the second level model. Such weights are applied on all points over the whole dataset. Our method is dynamic as it gets different set of weights at different parts of the data.

The method does not need a second level of model. It defines a neighbourhood and checks number of correct classifications done by different base learners. The weights provided to different base learners come from the number of correct classifications done by those base learners in the neighbourhood of the point to be classified. The detailed calculations can be understood in the Sect. 4.1.

4.1 Steps of Stacking Using Neighbourhood-Based Dynamic Weights

Step 1 Run k number of classification techniques $T1, T2, \ldots, Tk$ and build models.

Step 2 Get the first observation $O1$ from the new dataset.

Step 3 Using the chosen distance measure—(say, Euclidean if the independent variables are continuous) find out distance of all of the points in the training dataset from $O1$.

Step 4 Choose 'n' nearest neighbours of $O1$ using the distances found.

Step 5 Check correct classifications done by different base learners in the neighbourhood of $O1$.

Step 6 Derive the weight of the base learner Ti $[i = 1, 2, 3, \ldots, k]$ as the ratio of 'correct classifications done by Ti in the neighbourhood of $O1$' to 'sum of correct classifications done by all of the techniques' in the neighbourhood of O1. Say, the weight for base learner Ti $[i = 1, 2, 3, \ldots, k]$ in the neighbourhood of $O1$ is $w_{1i}[i = 1, 2, 3, \ldots, k]$.

Step 7 Do a dot-product of the weights of the base learners with the probabilities of event they give for $O1$. Say, the probability of event given by base learner Ti $[i = 1, 2, 3, \ldots, k]$ for $O1$ is $p_{1i}[i = 1, 2, 3, \ldots, k]$. So, we have to get

$$\sum_{i=1}^{k} w_{1i} p_{1i}$$

Step 8 The number obtained at Step 7 can be considered as the probability of event for $O1$. Thereafter, classification is done based on a cut-off.

Step 9 Repeat same procedure for other observations from the new dataset like $O2$, $O3$, etc. The neighbourhood for different observations can be different, and hence, the weights they get for different base learners would be different.

Step 10 Hyperparameter tuning: Run the whole procedure for different values of n to get optimal value of n. Please note that n is the number of nearest neighbours as mentioned in Step 4.

5 Proposed Method II: Stacking Using Distance-Based Dynamic Weights

This method is a variant of the method described in proposed method I. Here, all points in the neighbourhood do not get same importance. Within the neighbourhood, distance of a point from the new observation, is considered to provide importance to the point while calculating weights of the base learners. Adjustment factors are calculated which are higher for a closer point to the new observation in comparison to a distant point. These adjustment factors are used to get an adjusted count of correct classifications by different base learners. The weights provided to different base learners come from the number of adjusted correct classifications done by those base learners in the neighbourhood of the new observation to be classified. The detailed calculations can be understood in Sect. 5.1.

5.1 Steps of Stacking Using Distance-Based Dynamic Weights

Step 1. Run k number of classification techniques $T1, T2, \ldots, Tk$ and build models.
Step 2. Get the first observation $O1$ from the new dataset.
Step 3. Using the chosen distance measure, find out distance of all of the points in the training dataset from $O1$.
Step 4. Choose 'n' nearest neighbours of $O1$ using the distances found.
Step 5. Find out the maximum distance a point has with $O1$ within O1s defined neighbourhood. Say the mentioned maximum distance is M.
Step 6. Get vector of adjustment factors **A** consisting of adjustment factors Aj [$j = 1, 2, 3, \ldots, n$] for each of the n points as
1—(distance of the point from $O1/M$)
The point with distance M from $O1$ will have 0 as adjustment factor. So, one of the n values in the vector **A** will be 0.
Step 7. Define Cij as 1 if technique Ti has done correct classification of the observation j in the defined neighbourhood of $O1$; otherwise, Cij is 0. [$i = 1, 2, 3, \ldots, k$; $j = 1, 2, 3, \ldots, n$].
Get matrix **C** of order $k \times n$.
Step 8. Get the matrix $\mathbf{U} = \mathbf{C} \cdot \mathbf{A}$. **U** is a matrix of order $k \times 1$
Step 9. Get the matrix $\mathbf{W} = \mathbf{U}$/sum of all elements of **U**. **W** is a matrix of order $k \times 1$. The elements of **W** in serial are weights to be used on the k base learners, respectively.
Step 10. Do a dot-product of the weights of the base learners with the probabilities of event they give for $O1$
Step 11. The number obtained at Step 10 can be considered as the probability of event for $O1$. Thereafter, classification is done based on a cut-off.

Step 12. Repeat same procedure for other observations from the new dataset like $O2$, $O3$, etc. The neighbourhood for different observations can be different, and hence, the weights they get for different base learners would be different.
Step 13. Hyperparameter tuning: Run the whole procedure for different values of n to get optimal value of n.

6 Findings

Four classification techniques, namely logistic regression, linear discriminant analysis, decision tree and k-nearest neighbors are run on four datasets freely available. Thereafter, we ran stacking by conventional way as well as both of the proposed ways where weights vary. Performance of the models are judged by the following metrices.

AUC: Area under the receiver operating characteristics (ROC) curve.

Accuracy: Percentage of correct classifications.

Precision: Percentage of truly being '1' (event) out of the total number of '1' predicted. Suppose, a model has classified $m1$ number of observations to the class of '1' while only m out of m1 are correctly classified as '1'. Then, precision of the model equals to $m/m1$.

Recall: Percentage of observed '1' (event) predicted as '1'. Suppose, the number of observations belonging to class '1' in the data is $m2$ out of which m are correctly classified as '1' by a model. Then, recall of the model equals to $m/m2$.

The datasets are split randomly in 80:20 ratio where distribution of the categories of the target variable are kept same. Models are built on the 80% of the dataset, and the remaining 20% is treated as if it is the set of new observations where the models are to be applied. Performance of models on 20% of the dataset is of more importance and are compared. Henceforth, the 80% chunk is referred as training data and the 20% hold-out sample is referred to as test data.

The findings based on the experimentation done on four datasets are given below. The probability cut-off for classifying event or non-event is set at the proportion of event in the whole dataset. The proposed methods of stacking are done using different values of 'n' which is the number of neighbours, and the optimum value on 'n' is chosen based on accuracy.

6.1 Wholesale Customer Data

The data is of clients of a wholesale distributor [3]. It is of 440 observations. The data has information about how much clients spent in a year on different categories, namely fresh products, milk products, grocery products, frozen products, paper products and delicatessen products. Annual spent on different categories are continuous variables. Channel is the target variable, which is a binary nominal variable having

Table 1 Evaluation of the results on the training dataset of wholesale customer data

Techniques	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	90	82	90
Linear discriminant analysis	85	94	58
Decision tree	94	93	88
K-nearest neighbor ($K = 15$)	92	87	89
Stacking by conventional way	94	93	89

Chosen value of K is the one where highest accuracy obtained when different values of K are tried

Table 2 Evaluation of the results on the test dataset of wholesale customer data

Techniques	AUC	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	0.97	92	84	93
Linear discriminant analysis	0.97	88	95	64
Decision tree	0.94	92	84	93
K-nearest neighbor ($K = 15$)	0.97	92	84	93
Stacking by conventional way	0.96	92	84	93

categories 'Horeca (hotel/restaurant/cafe)' and 'Retail'. There is one more variable which is on regions; we have not used it.

We have used different techniques aiming classification of the categories of the target variable 'Channel'. The models provide probability of being 'Retail' (Tables 1 and 2).

Though 'precision' is found to be better in training dataset in most of the techniques, 'recall' is found to be better in test data due to increase in correct classification of the events '1'. The only technique which did not perform at par with others is 'linear discriminant analysis'. We applied the new ways of stacking, accuracy, precision and recall on the test data which are 93%, 84% and 96%, respectively, when stacking is done using neighbourhood-based dynamic weights with size of neighbourhood $n = 5$ (found optimum among the values of $n = 2, 3, 4, \ldots, 25$). Stacking using distance-based dynamic weights with size of neighbourhood $n = 6$ (found optimum among the values of $n = 2, 3, 4, \ldots, 25$) also provide accuracy, precision and recall of 93%, 84% and 96%, respectively, on the test data. So, both of the new methods of stacking increases recall; as recall becomes 96% from 93%.

6.2 Pima Indians Diabetes Data

The data is of female patients of at least 21 years old of Pima Indian heritage [4]. It is of 768 observations. The data has information of patients about number of pregnancies, plasma glucose concentration at 2 h in an oral glucose tolerance test,

Table 3 Evaluation of the results on the training dataset of Pima Indians diabetes data

Techniques	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	74	60	80
Linear discriminant analysis	77	73	56
Decision tree	85	84	71
K-nearest neighbor ($K = 17$)	79	76	58
Stacking by conventional way	85	80	75

Chosen value of K is the one where highest accuracy obtained when different values of K are tried

Table 4 Evaluation of the results on the test dataset of Pima Indians diabetes data

Techniques	AUC	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	0.87	77	61	91
Linear discriminant analysis	0.87	77	69	65
Decision tree	0.77	75	65	59
K-nearest neighbor ($K = 17$)	0.80	72	63	50
Stacking by conventional way	0.81	75	65	65

diastolic blood pressure, triceps skinfold thickness (mm), 2 h serum insulin (mu U/ml), body mass index (weight in kg/(height in m)^2), diabetes pedigree function and age (years). Diabetes (Class) is the target variable, which is a binary nominal variable having categories 'diabetic' and 'non-diabetic'.

We have used different techniques aiming classification of the categories of the target variable 'diabetes (class)'. The models provide probability of being 'diabetic' (Tables 3 and 4).

Here, logistic regression performs much better than other techniques. Surprisingly, performance of logistic regression is much better in the test data set than training dataset. In case of linear discriminant analysis, its precision is 8% higher than the precision of logistic regression, but recall is 26% lower than recall of logistic regression. Performance of other techniques are not satisfactory, and thus, conventional stacking does not cause any benefit to the results when compared with logistic regression. However, both of the new methods of stacking does much better than the conventional way of stacking in recall though some compromise is there in precision. Accuracy, precision and recall on the test dataset in case of stacking using neighbourhood-based dynamic weights with size of neighbourhood $n = 4$ (found optimum among the values of $n = 2, 3, 4, \ldots, 25$) are 74%, 60%, 81%, respectively, while such measures on the test dataset when stacking using distance-based dynamic weights with size of neighbourhood $n = 6$ (found optimum among the values of $n = 2, 3, 4, \ldots, 25$) is used are 76%, 61%, 87%, respectively. So, we see that stacking using distance-based dynamic weights performed better than stacking using neighbourhood-based dynamic weights where accuracy, precision and recall got increased by 2%, 1% and 6%, respectively.

Table 5 Evaluation of the results on the training dataset of bank note authentication data

Techniques	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	99	98	99
Linear discriminant analysis	97	95	100
Decision tree	98	97	99
K-nearest neighbor ($K = 9$)	100	100	100
Stacking by conventional way	100	100	100

Chosen value of K is the one where highest accuracy obtained when different values of K are tried

Table 6 Evaluation of the results on the test dataset of bank note authentication data

Techniques	AUC	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	0.99	99	98	99
Linear discriminant analysis	0.99	99	97	100
Decision tree	0.98	99	98	99
K-nearest neighbor ($K = 9$)	1.00	100	100	100
Stacking by conventional way	1.00	100	100	100

6.3 Bank Note Authentication Data

The data is of authentication of bank notes [5]. It is of 1372 observations. Data were extracted from images that were taken from genuine and forged banknote-like specimens. Wavelet transform tool was used to extract features from images. The data has information about variance of wavelet transformed image, skewness of wavelet transformed image, kurtosis of wavelet transformed image, entropy of image which are continuous variables. Class is the target variable, which is a binary nominal variable having categories ‘authenticate’ or ‘non-authenticate’.

We have used different techniques aiming classification of the categories of the target variable ‘Class’. The models provide probability of being ‘authenticate’ (Tables 5 and 6).

Here, recall of all of the base techniques is between 99% and 100%. Here, all of the methods of stacking provide 100% in all three measures, namely accuracy, precision and recall. The size of neighbourhood n used for both of the proposed methods is 3.

6.4 Iris Data

The data is of different types of flowers [6]. It is of 150 observations. The data has information about sepal length, sepal width, petal length, petal width and species type. Species is the categorical variable, having categories ‘setosa’, ‘versicolor’ and

Table 7 Evaluation of the results on the training dataset of iris data

Techniques	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	73	58	75
Linear discriminant analysis	73	63	43
Decision tree	95	89	98
K-nearest neighbor ($K = 11$)	96	93	95
Stacking by conventional way	96	93	95

Chosen value of K is the one where highest accuracy obtained when different values of K are tried

Table 8 Evaluation of the results on the test dataset of iris data

Techniques	AUC	Accuracy (%)	Precision (%)	Recall (%)
Logistic regression	0.94	87	80	80
Linear discriminant analysis	0.95	80	83	50
Decision tree	1	100	100	100
K-nearest neighbor ($K = 11$)	1	100	100	100
Stacking by conventional way	1	100	100	100

'virginica'; other variables are continuous in nature which are used as explanatory variables. Spec_1 is our derived variable used as target variable which is binary in nature stating whether the species is 'versicolor' or not.

We have used different techniques aiming classification of the categories of the target variable 'Spec_1'. The models provide probability of being 'versicolor' (Tables 7 and 8).

Here, all of the methods of stacking provide 100% in all three measures, namely accuracy, precision and recall. The size of neighbourhood n used for both of the proposed methods is 3.

7 Conclusion

Both of the proposed methods of stacking with dynamic weights work better than the conventional way of stacking since the proposed methods are flexible. The performance of stacking by conventional way and the proposed methods is shown below (Table 9).

Table 9 Performance comparison

Dataset	Method	Accuracy (%)	Precision (%)	Recall (%)
Wholesale Customer Data	Stacking by conventional way	92	84	93
	Stacking by neighbourhood-based dynamic weights	93	84	96
	Stacking by distance-based dynamic weights	93	84	96
Pima Indians Diabetes Data	Stacking by conventional way	75	65	65
	Stacking by neighbourhood-based dynamic weights	74	60	81
	Stacking by distance-based dynamic weights	76	61	87
Bank Note Authentication Data	Stacking by conventional way	100	100	100
	Stacking by neighbourhood-based dynamic weights	100	100	100
	Stacking by distance-based dynamic weights	100	100	100
Iris Data	Stacking by conventional way	100	100	100
	Stacking by neighbourhood-based dynamic weights	100	100	100
	Stacking by distance-based dynamic weights	100	100	100

Though ‘accuracy’ and ‘precision’ are not seen to get improved by the proposed methods of stacking, but improvement in ‘recall’ is noticed. In the ‘wholesale customer data’, recall is as high as 93% by conventional stacking. Still by applying both of the proposed methods, we are able to increase recall by further 3%. In the Pima Indian Diabetes Data, stacking by neighbourhood-based dynamic weights has performed poorer in precision by 5% but has increased recall by 16% over stacking by conventional way. Results of stacking using distance-based dynamic weights is even more encouraging as we get 22% increase in recall over stacking by conventional way, and a compromise of 4% is there in precision. Performances of the proposed methods of stacking are same as the one by conventional way on other two datasets as

the conventional way did not leave any scope of improvement here by hitting 100% in all of the measures.

Hence, both of the proposed methods 'stacking by neighbourhood-based dynamic weights' and 'stacking by distance-based dynamic weights' are seen to outperform conventional way of stacking in recall.

Acknowledgements We hereby feel happy to acknowledge Siddhartha Mukherjee of Tata Consultancy Services, Kolkata and thank him for his help in coding.

References

1. Wolpert, D. (1992). Stacked generalization. *Neural Networks, 5*(2), 241–259.
2. Breiman, L. (1996). Stacked regression. *Machine Learning, 24*.
3. Cardoso, M. G. M. S. (2014). [Wholesale Customer Data Set] UCI Machine Learning Repository [https://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
4. Sigillito, V. (1990). [Pima Indians Diabetes Data Set] UCI Machine Learning Repository [https://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
5. Lohweg, V., & Doerksen, H. (2013). [Banknote Authentication Data Set] UCI Machine Learning Repository [https://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
6. Fisher, R. A., & Marshall, M. (1988). [Iris Data Set] UCI Machine Learning Repository [https://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.