The Developing of the System for Automatic Audio to Text Conversion

Vladyslav Tsap, Nataliya Shakhovska, Ivan Sokolovskyi

Lviv Polytechnic National University, 12 Bandera str., Lviv, 79013, Ukraine

Abstract

The paper is explored the AdaBoost algorithm, one of the most popular ensemble boosting methods. The main ensemble methods and their advantages and disadvantages are considered and focused on the AdaBoost algorithm. The dataset of the University of Belgium compiled is used. AdaBoost algorithm was experimentally applied to the data set, and its effectiveness was tested. This algorithm can slightly improve the result compared to "strong" classifiers. However, there are cases when even a difference of 3-5% is significant.

Keywords[1]

machine learning, ensemble, Adaboost, performance, Keras, accuracy

1. Introduction

There are many tasks in our life that require a lot of time, and it is not always possible to find the optimal solution by human forces. That is why to find an answer to this kind of problem using machine learning (ML), the purpose of which is to predict the result from the input data. In this case, you can quickly find a solution that will also be accurate and effective [1].

There are several types of machine learning [2]:

- Classical machine learning is used in the case when the task is simple, the data are quite simple, and the features are known;
- Reinforced learning - used when there is no processed, ready-made data, but there is an environment with which you can interact;
- Ensembles - used when the quality and accuracy of the result is critical;
- Neural networks and deep learning - often used when the data is complex, and the signs are not clear or to improve the already known model of MN.

The paper will be devoted to the study of ensembles, and more specifically, Adaptive Boosting.

The task will be to analyze the basic principles of the AdaBoost boosting ensemble (abbreviated form from Adaptive Boosting), find the areas where this ensemble is already used, and its software implementation a specific example.

2. Literature review

At this stage of machine learning development, systems are being developed for the management and interaction of Internet of Things technology, the concept of a "smart city", self-driving cars, and more. These industries are extremely promising because the direction in which large technology companies are moving, as Apple, Amazon, Facebook, Google, and Microsoft are already using MN in products with apparent benefits.

However, some tasks are so complex that no machine learning algorithm can handle them independently to give a sufficiently accurate answer. And at this point in the development of MN, ensembles that combine

MoMLeT+DS 2021: 3rd International Workshop on Modern Machine Learning Technologies and Data Science, June 5, 2021, Lviv-Shatsk, Ukraine

EMAIL: vladyslav.tsap.kn.2017@lpnu.ua (V. Tsap); nataliya.b.shakhovska@lpnu.ua (N. Shakhovska); sokolovskyi.vani@gmail.com (I.Sokolovskyi).

ORCID: 0000-0002-8062-0079 (O. V.Tsap); 0000-0002-6875-8534 (N. Shakhovska); 0000-0002-0112-8466 (I.Sokolovskyi)

several algorithms will help us a lot to learn and correct mistakes simultaneously, thus improving the accuracy of the results many times over other algorithms.

Due to their high accuracy and stability, ensembles are very common among giant IT companies, as it is crucial for them to quickly process large amounts of data and at the same time a reasonably accurate result. It is no secret that ensembles are now actively competing with neural networks, as both are quite effective in the same tasks.

Interestingly, ensembles often use unstable algorithms that are unstable to the input data to get better results. It is then that these algorithms can go beyond the usual framework of solutions and at the same time correct each other's errors to get the correct answer. In fact, classification algorithms are combined in an ensemble to increase accuracy by creating a strong classifier from a number of weak classifiers. An example of such unstable algorithms is Regression or Solution Tree because one strong anomaly is enough to break the whole model. However, the very combination of these two algorithms in the ensemble gives an excellent result. For example, algorithms that do not fit are the Bayesian or KNN algorithm because they are very stable.

There are three time-tested ways to make ensembles: stacking, bagging, and boosting [3].

In short, the peculiarity of stacking is that we teach several different algorithms and pass their results to the input of the last, who makes the final decision. The critical difference is different algorithms because if you teach the same algorithm on the same data, it will not matter. Regression is usually used as the final algorithm. However, stacking is rarely used among ensembles, as the other two methods are generally more accurate [4, 5].

Its peculiarity is that we train one algorithm many times on random samples from the source data for boosting. In the end, we average all the results, and, in this way, we get the answer. The most famous example of bagging is the Random Forest algorithm, which we can observe when we open the camera on a smartphone and see how the program has circled people's faces with yellow rectangles. Neural networks would be too slow in a particular task, and bagging is ideal here because it can count trees in parallel on all shaders of the video card. It is the possibility of paralleling that gives bugging an advantage over other ensembles [6].

A distinctive feature of the boosting ensemble is that we train our algorithms consistently, even though each subsequent one pays special attention to the cases in which the previous algorithm failed. Like in the running, we make samples from the source data, but now it's not entirely random. In each new sample, we take part of the data on which the previous algorithm worked incorrectly. In fact, we are learning a new algorithm from the mistakes of the previous one. This ensemble has a very high accuracy, which is an advantage over all other ensembles. However, there is also a downside - it is difficult to parallelize. It still works faster than neural networks, but slower than bugging. Also, boosting Sundays can be attributed to the fact that it can lead to the construction of cumbersome compositions, which consist of hundreds of algorithms. Such compositions eliminate the possibility of meaningful interpretation, require massive amounts of memory to store basic algorithms and spend a lot of time calculating classifications.

A well-known example of boosting is the problem of classifying objects in an image because simple classifiers based on some features, are usually ineffective in this classification. Using boosting methods for this task is combining weak classifiers with a special method to improve the overall classification capability. The classification of features is a typical computer vision task, where it is determined whether an image contains a certain category of objects or not. This idea is closely related to recognition, identification and analysis. Also, boosting is widely used in the task of ranking the issuance of search engines. This happened when this problem was considered in terms of the loss function, which penalizes for errors in the order of issuance, so it was quite convenient to implement gradient boosting (English gradient boosting) in the ranking.

One of the popular boosting algorithms is AdaBoost (short for Adaptive Boosting). It was proposed by Robert Shapiro and Joav Freud in 1996. AdaBoost was the basis for all subsequent research in this area.

Its main advantages include high speed, as the construction time of the composition is almost entirely determined by the learning time of basic algorithms, ease of implementation, good generalized property, which can be further improved by increasing the number of basic algorithms, and the ability to identify emissions - the most "heavy" objects xi, for which in the process of increasing the composition of the weight wi take the largest values.

Its disadvantages include the fact that AdaBoost is an algorithm with a convex loss function, so it is sensitive to noise in the data and is prone to overfitting compared to other algorithms. An example can be seen in Table 1; namely, we can draw a column error in the test - it first decreases, and then begins to grow, despite the fact that the error in learning is constantly decreasing.

Also, the AdaBoost algorithm requires large enough training samples. Other methods of linear correction, in particular, bugging, are able to build algorithms of comparable quality on smaller data samples.

The paper aimed to analyse “weak classifiers” and to choose the appropriative number of classisifers and theirs hyperparameters.

Table 1.
AdaBoost error rate on training and test data

Number of classifiers	Training error	Testing error
1	0.28	0.27
10	0.23	0.24
50	0.19	0.21
100	0.19	0.22
500	0.16	0.25
1000	0.14	0.31
10000	0.11	0.33

3. Materials and Methods

Stacking reinforces "weak" classifiers by uniting them in a committee. It has gained its "adaptability" because each subsequent classifier committee is built on objects that previous committees have misclassified. This is because correctly classified objects lose weight, and incorrectly classified objects gain more weight.

An example of the algorithm can be explained using Figure 1.

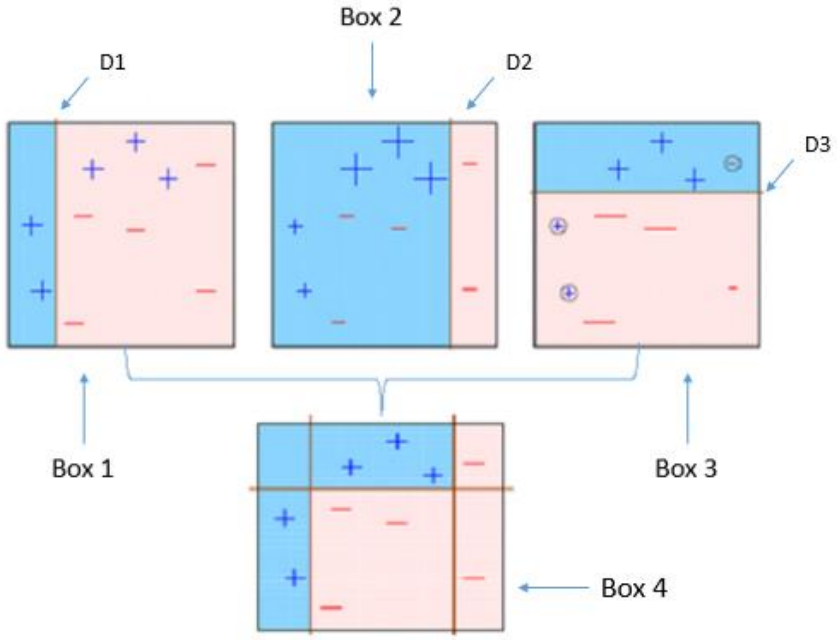

Figure 1. Diagram of the AdaBoost algorithm

In Box 1, we assign weight levels to all points and use a decision stump to classify them as “pluses” or “minuses”. This "weak" classifier generated a vertical line on the left (D1) to classify the points. We see that this vertical line incorrectly divided the three "pluses", classifying them as "minuses". In this case, we give these three "pluses" more weight and apply this classifier again.

In Box 2 we see that the size of the three incorrectly classified "pluses" is larger than the other points. In this case, the threshold classifier of solutions (D2) will try to predict them correctly. And really: now the vertical line (D2) correctly classified the three incorrectly classified "pluses". However, this has led to other classification problems - three "minuses" are incorrectly classified. Let's perform the same operation as in the previous time - assign more weight to incorrectly classified points and apply the classifier again.

In Box 3, the three "minuses" have more weight. To correctly distribute these points, we again use the threshold classifier (D3). This time, a horizontal line is formed to classify the “pluses” and “minuses” based on the greater weights of the misclassified observation.

In Box 4, we combine the results of classifiers D1, D2, and D3 to form a strong prediction that has a more complex rule than the individual rules of the “weak” classifier. And as a result, in Box 4, we see that the AdaBoost algorithm classified observations much better than any single "weak" classifier.

We have a binary classification task with labels $y \in \{-1, +1\}$. In general, the task is given as $h(x) = h(x|y), h(x) \in \{-1, +1\}$, where classificatory is defined as $\hat{y}(x) = sign\left\{h_0(x) + \sum_{i=1}^{N} c_i \cdot h_i(x)\right\}$, and loss function $\mathfrak{L}(h(x), y) = e^{-y \cdot h(x)}$.

The algorithm consists of the following steps:

- Input: training dataset (x_i, y_i), $i = \overline{1,N}$; basic algorithm $h(x) \in \{-1, +1\}$ trained by weighted dataset; M is the number of iterations.
- The weight initialization: $w_i = \frac{1}{N}, i = \overline{1,N}$.
- For $m = 1, 2, \dots, M$:
  - To train $h^m(x)$ on training dataset with weights w_i, $i = \overline{1,N}$
  - To calculate weighted error $E_m = \frac{\sum_{i=1}^{N} w_i \cdot \mathbb{I}[h^m(x_i) \neq y_i]}{\sum_{i=1}^{N} w_i}$
  - If $E_m > 0.5$ or $E_m = 0$: stop
  - To calculate $c_m = \frac{1}{2} \cdot ln \frac{1-E_m}{E_m}$.
- To increase all weights where the basic algorithm was wrong: $w_i := w_i \cdot e^{2 \cdot c_m}, i \in \{i: h^m(x_i) \neq y_i\}$
- Output: ensemble $F(x) = sign\{\sum_{m=1}^{M} c_m \cdot h^m(x)\}$

The coefficient for weights looks like the following: $c_m = \frac{1}{2} \cdot ln \frac{1-E_m}{E_m}$, where E_m or $Error$ is the total number of incorrectly classified points for this training set divided by the size of our dataset.

The coefficient c or $Alpha$ distribution based on Error rate is given on Figure 2.

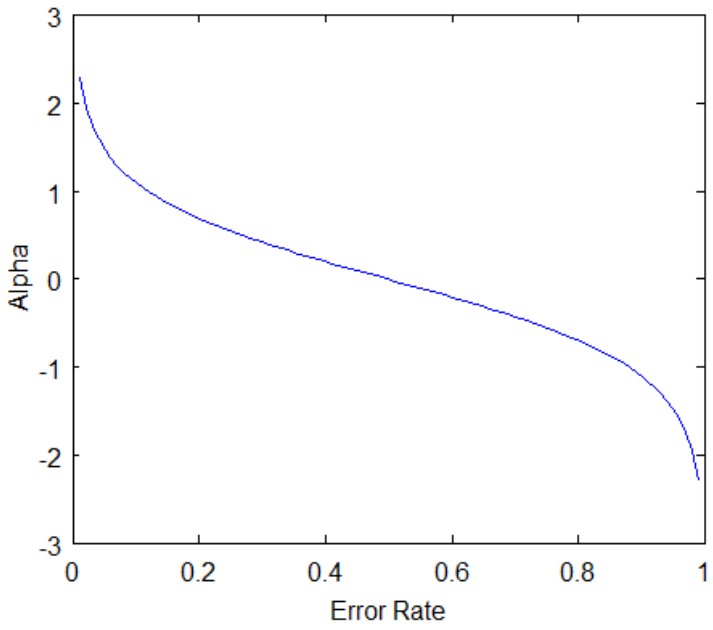

Figure 2. Graph the value of the coefficient c from the error value

Note that when the basic algorithm works well or does not have the wrong classification, it leads to an error of 0 and a relatively high value of c. When the base classifier classifies half of the points correctly, and half incorrectly, the value of the coefficient c will be equal to 0 because this classifier is no better than random assumptions with a probability of 50%. And in the case where the primary classifier constantly gives incorrect results, the value of the coefficient c will become quite negative.

There are two cases of alpha function:

- The value of c is positive when the predicted and actual results coincide, i.e. the point has been classified correctly. In this case, we reduce the point's weight because the work is going in the right direction.

- A value of c is negative when the predicted result does not match the actual result, ie the point was misclassified. In this case, it is necessary to increase the point's weight so that the same erroneous classification is not repeated in the next classification.

In both cases, the basic classifiers are dependent on the result of the previous one.

4. Results

We will use the dataset of the University of Belgium compiled in 2017 [13]. This dataset contains information on electricity consumption throughout the house and temperature and humidity in the house and weather conditions (temperature, humidity, pressure, wind speed, etc.) on the street.

The structure of dataset is given below.

```
'data.frame':  19735 obs. of  29 variables:
 $ date       : chr  "2016-01-11 17:00:00" "2016-01-11 17:10:00" "2016-01-11 17
:20:00" "2016-01-11 17:30:00" ...
 $ Appliances : int  60 60 50 50 60 50 60 60 60 70 ...
 $ lights     : int  30 30 30 40 40 40 50 50 40 40 ...
 $ T1         : num  19.9 19.9 19.9 19.9 19.9 ...
 $ RH_1       : num  47.6 46.7 46.3 46.1 46.3 ...
 $ T2         : num  19.2 19.2 19.2 19.2 19.2 ...
 $ RH_2       : num  44.8 44.7 44.6 44.6 44.5 ...
 $ T3         : num  19.8 19.8 19.8 19.8 19.8 ...
 $ RH_3       : num  44.7 44.8 44.9 45 45 ...
 $ T4         : num  19 19 18.9 18.9 18.9 ...
 $ RH_4       : num  45.6 46 45.9 45.7 45.5 ...
 $ T5         : num  17.2 17.2 17.2 17.2 17.2 ...
 $ RH_5       : num  55.2 55.2 55.1 55.1 55.1 ...
 $ T6         : num  7.03 6.83 6.56 6.43 6.37 ...
 $ RH_6       : num  84.3 84.1 83.2 83.4 84.9 ...
 $ T7         : num  17.2 17.2 17.2 17.1 17.2 ...
 $ RH_7       : num  41.6 41.6 41.4 41.3 41.2 ...
 $ T8         : num  18.2 18.2 18.2 18.1 18.1 18.1 18.1 18.1 18.1 18.1 ...
 $ RH_8       : num  48.9 48.9 48.7 48.6 48.6 ...
 $ T9         : num  17 17.1 17 17 17 ...
 $ RH_9       : num  45.5 45.6 45.5 45.4 45.4 ...
 $ T_out      : num  6.6 6.48 6.37 6.25 6.13 ...
 $ Press_mm_hg: num  734 734 734 734 734 ...
 $ RH_out     : num  92 92 92 92 92 ...
 $ Windspeed  : num  7 6.67 6.33 6 5.67 ...
 $ Visibility : num  63 59.2 55.3 51.5 47.7 ...
 $ Tdewpoint  : num  5.3 5.2 5.1 5 4.9 ...
 $ rv1        : num  13.3 18.6 28.6 45.4 10.1 ...
 $ rv2        : num  13.3 18.6 28.6 45.4 10.1 ...
```

The idea is to classify high and low electricity consumption depending on the weather and learn to predict what the consumption will be depending on the weather. Since this is actual data, it can be used to forecast electricity costs; it will be possible to consider the feasibility of buying energy-efficient light bulbs, for example, or just think about how to reduce electricity consumption.

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	RH_5	T6	RH_6	T7	RH_7	T8	
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	45.566667	17.166667	55.20	7.026667	84.256667	17.200000	41.626667	18.2	48.
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	45.992500	17.166667	55.20	6.833333	84.063333	17.200000	41.560000	18.2	48.
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	45.890000	17.166667	55.09	6.560000	83.156667	17.200000	41.433333	18.2	48.
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	45.723333	17.166667	55.09	6.433333	83.423333	17.133333	41.290000	18.1	48.
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	45.530000	17.200000	55.09	6.366667	84.893333	17.200000	41.230000	18.1	48.

Figure 3. The first 5 records from the dataset

We see that there is data that we do not need - then we will clear it. These include, for example, a timestamp. Also, check our dataset for empty values. Checking for Outliers and removing extreme 1% of the data is provided.

First, explaratory analysis is given. Distribution graphs of sampled columns is shown on Fig. 4.

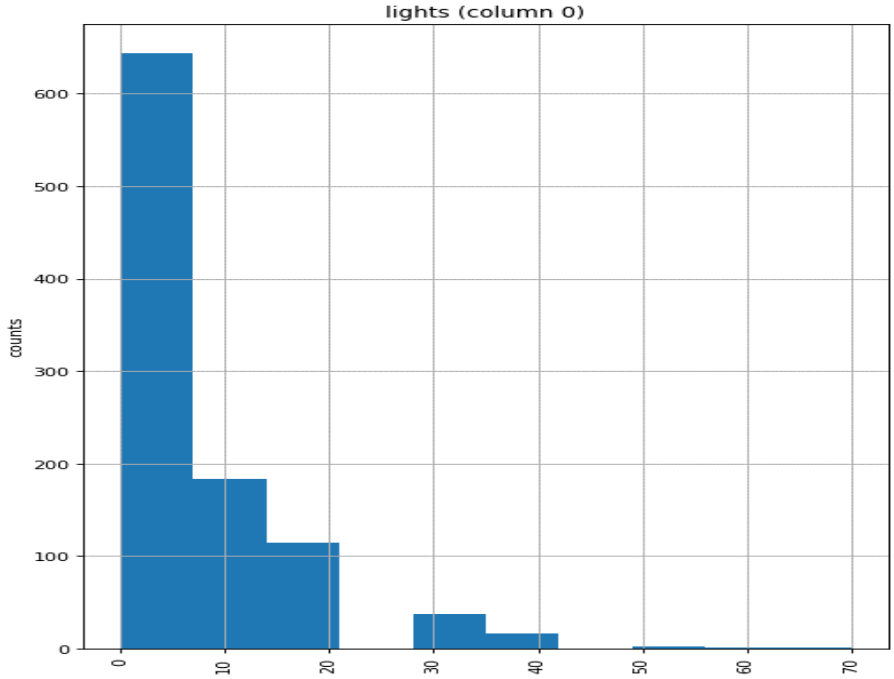

Figure 4. Distribution graphs of sampled columns

Corellation matrix shows us the dependencies between columns. Here the dependencies between T1 and T2, T6 and T_out are found (Fig. 5).

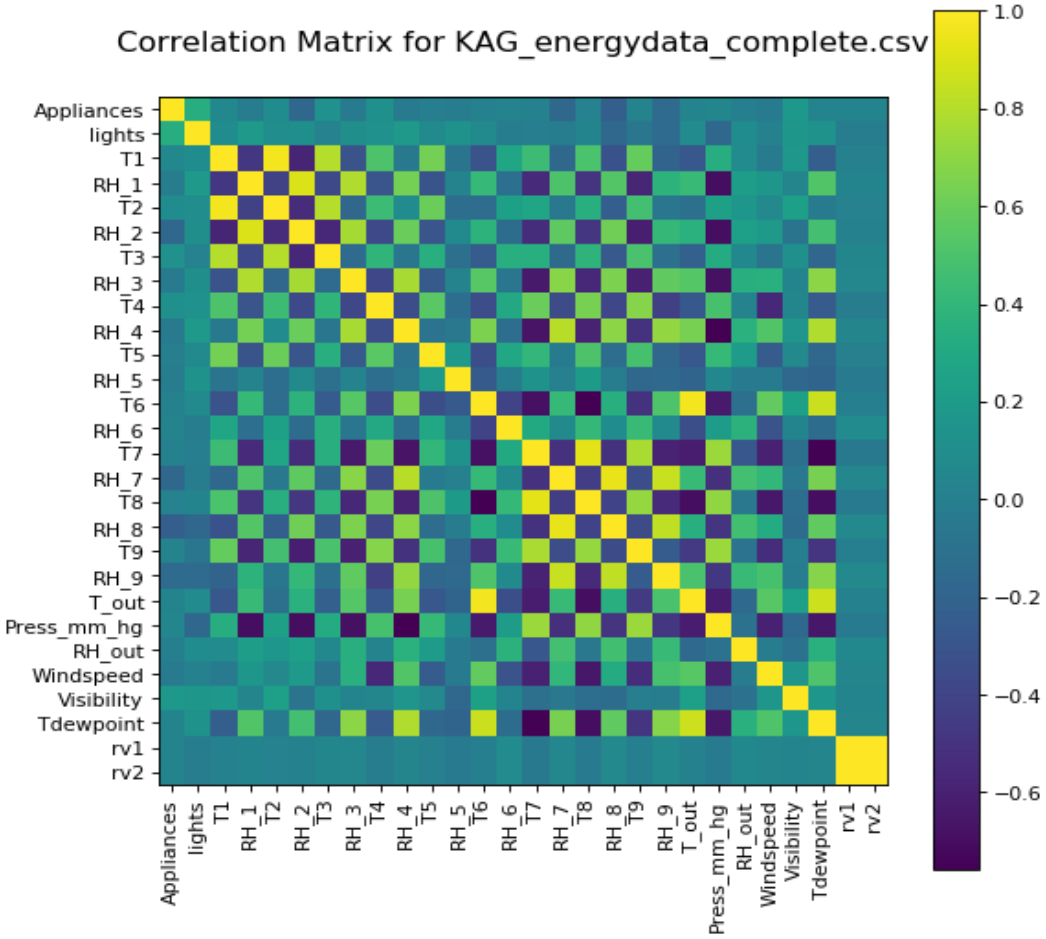

Figure 5. Corellation matrix

Scatter and density plots are presented on Fig. 6.

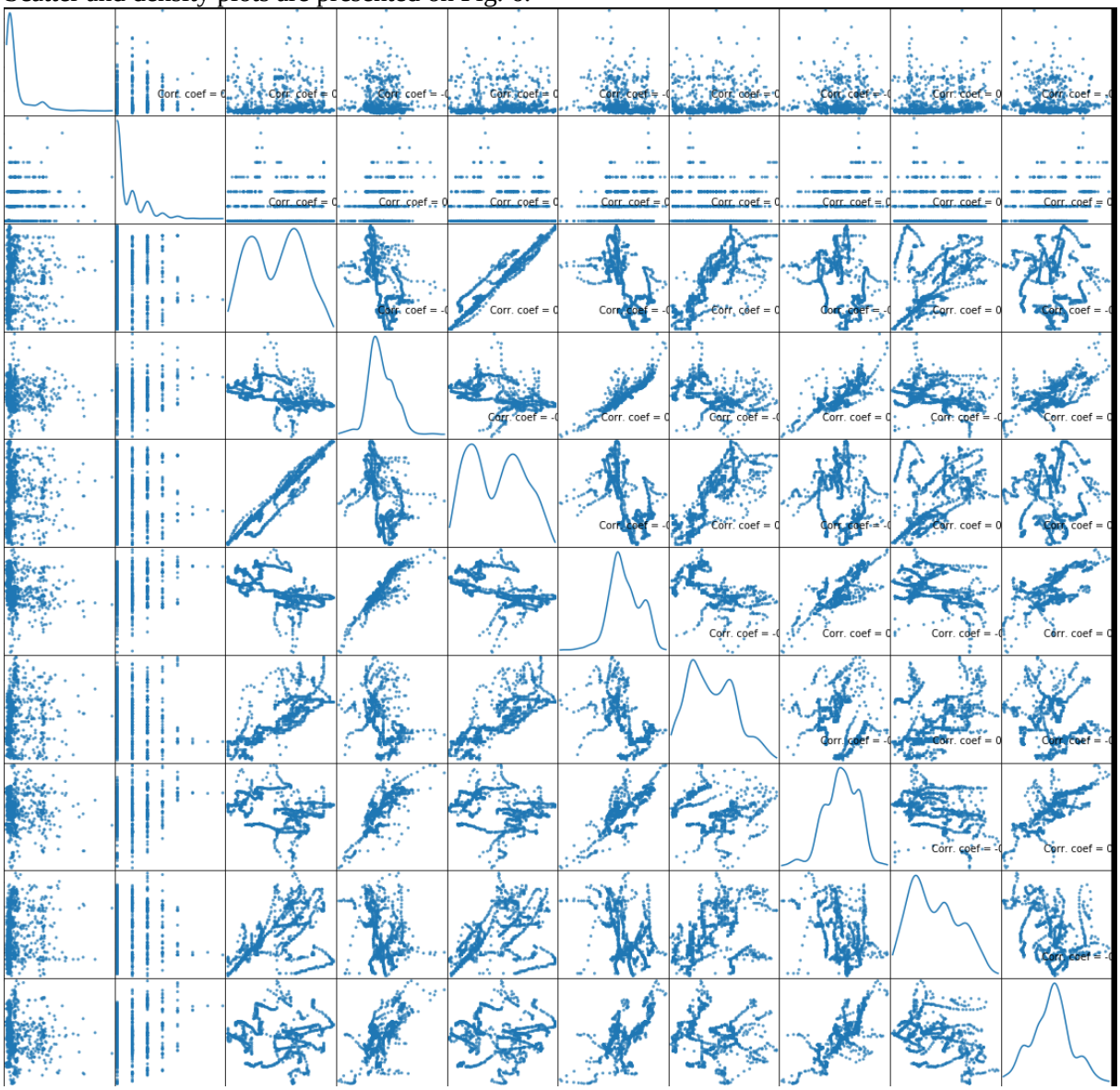

Figure 6. Scatter plots for all variables.

After preparing the dataset, we will review the distribution of electricity use (Fig. 7, Fig. 8).

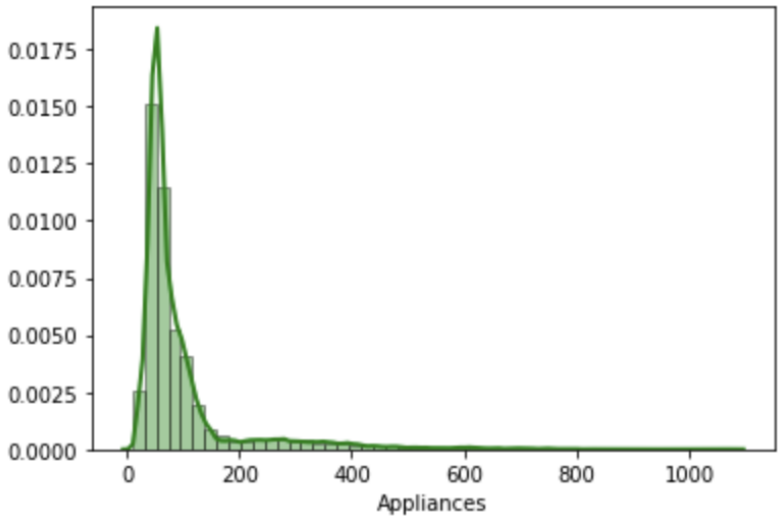

Figure 7. Distribution of electricity consumption.

	Appliances
count	19735.000000
mean	97.694958
std	102.524891
min	10.000000
25%	50.000000
50%	60.000000
75%	100.000000
max	1080.000000

Figure 8. Basic metrics for the energy consumption column

We see that the average value is 100 units, which is why we will divide our dataset into 2 parts: low (up to 100) and high consumption (more than 100).

Classification will take place on 25 parameters, and the type of consumption will be defined.

We first start the Decision Tree - the primary classifier (Fig. 9).

```
[[2956  107]
 [ 783  101]]
              precision    recall  f1-score   support

           0       0.79      0.97      0.87      3063
           1       0.49      0.11      0.18       884

    accuracy                           0.77      3947
   macro avg       0.64      0.54      0.53      3947
weighted avg       0.72      0.77      0.72      3947

Accuracy Score:  77.5
Cross-Validation Score:  78.7
```

Figure 9. The results of the classification of the Decision Tree.

Then create an AdaBoost classifier with the following structure:

- base_estimator ;
- n_estimators - the maximum number of ratings;
- learning_rate - influence on the weight of the classifier.

As a "weak" classifier we will use the Decision Tree, which we used in the previous classification, the number of classifiers n_estimators will be set to 200, and the effect on learning_rate weights - 1, which is equal to the default value. Run the AdaBoost algorithm (Fig. 10).

```
[[2866  197]
 [ 286  598]]
              precision    recall  f1-score   support

           0       0.91      0.94      0.92      3063
           1       0.75      0.68      0.71       884

    accuracy                           0.88      3947
   macro avg       0.83      0.81      0.82      3947
weighted avg       0.87      0.88      0.88      3947

Accuracy Score:  87.8
Cross-Validation Score:  89.0
```

Figure 10. The results of the AdaBoost classifier.

We see that the prediction is more accurate, which means that AdaBoost is appropriate and profitable to use in this case. In general, the improvement is more than 10%, which is a very good result.

The comparison with other well-known classifiers is given too.

```
LinearRegression()
Average Error       : 0.3065 degrees
Variance score R^2  : 23.88%
Accuracy            : 83.00%

SVR()
Average Error       : 0.2764 degrees
Variance score R^2  : 23.67%
Accuracy            : 84.02%

RandomForestRegressor(random_state=1)
Average Error       : 0.1932 degrees
Variance score R^2  : 67.16%
Accuracy            : 85.71%

LGBMRegressor(n_estimators=200, num_leaves=41)
Average Error       : 0.2009 degrees
Variance score R^2  : 65.26%
Accuracy            : 85.54%

XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
Average Error       : 0.2180 degrees
Variance score R^2  : 61.44%
Accuracy            : 85.11%
```

To summarise, the classifier accuracy for Adabost is appropriated

5. Conclusion

The paper is explored the AdaBoost algorithm, which is the most popular ensemble boosting method.

The main ensemble methods and their advantages and disadvantages are consideres, as well as focused on the AdaBoost algorithm.

In the second section, the AdaBoost algorithm was experimentally applied to the data set and its effectiveness was tested.

In conclusion, we can say that the effectiveness of ensemble boosting methods compete with neural networks, but they are somewhat more stable and understandable, which is why they have an advantage in choice. Also, the AdaBoost algorithm can slightly improve the result compared to "strong" classifiers. However, there are cases when even a difference of 3-5% is significant. This can be said in the example of Google, which uses AdaBoost in search results, as well as Apple, Amazon, Facebook and Microsoft, which also use ensemble methods, simply do not publish their algorithms. They do it because it is financially profitable. And then even a few percent improvement is valuable. That is why this algorithm will be further developed and used.

The limitations of the study are the following:

- The quality of the ensemble depends on the dataset. For an imbalanced dataset, the classification accuracy will be lower;
- The modeling of charged cases should be provided together with clustering analysis. The authors plan to model each separated cluster and compare the classification accuracy.

The future extention of current work is ensemble development from weak classifiers.

References

[1] Y.Kryvenchuk, N.Boyko, I.Helzynskyy, T.Helzhynska, R.Danel “Synthesis control system physiological state of a soldier on the battlefield”. CEUR. Vol. 2488. (2019): 297–306.

[2] E. Eslami, A.K. Salman, Y.Choi, A. Sayeed, Y. Lops. “A data ensemble approach for real-time air quality forecasting using extremely randomized trees and deep neural networks”. Neural Computing and Applications (2019): 1-17.
[3] S.Ardabili, A.Mosavi, A. R.Várkonyi-Kóczy. “Advances in machine learning modeling reviewing hybrid and ensemble methods”. In International Conference on Global Research and Education. Springer, Cham. (2019): 215-227.
[4] A.Alves. “Stacking machine learning classifiers to identify Higgs bosons at the LHC”. Journal of Instrumentation, 12(05). (2017): T05005.
[5] Y. Freund “A more robust boosting algorithm”. arXiv preprint (2009). arXiv:0905.2138.
[6] J.Dou, , Yunus, A. P., Bui, D. T., Merghadi, A., Sahana, M., Zhu, Z., ... & Pham, B. T. (2020). Improved landslide assessment using support vector machine with bagging, boosting, and stacking ensemble machine learning framework in a mountainous watershed, Japan. Landslides, 17(3), 641-658.
[7] C.Ying, M.Qi-Guang, L.Jia-Chen, G.Lin. “Advance and prospects of AdaBoost algorithm”. Acta Automatica Sinica, 39(6). (2013): 745-758.
[8] T.Hastie, S.Rosset, J.Zhu, H.Zou. “Multi-class adaboost”. Statistics and its Interface, 2(3) (2009): 349-360.
[9] J.Cao, S.Kwong, R.Wang. “A noise-detection based AdaBoost algorithm for mislabeled data”. Pattern Recognition, 45(12), (2012): 4451-4465.
[10] D. P.Solomatine, D. L.Shrestha. “AdaBoost. RT: a boosting algorithm for regression problems”. In 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541). Vol. 2, 2004: 1163-1168.
[11] Y.Wei, X.Bing, C.Chareonsak. “FPGA implementation of AdaBoost algorithm for detection of face biometrics”. In IEEE International Workshop on Biomedical Circuits and Systems. (2004): 1-6.
[12] F.Wang, Z.Li, F.He, R.Wang, W.Yu, F.Nie. “Feature learning viewpoint of AdaBoost and a new algorithm”. IEEE Access, 7, (2019). 149890-149899.
[13] O. M.Mozos, C.Stachniss, W.Burgard. “Supervised learning of places from range data using adaboost”. In Proceedings of the 2005 IEEE international conference on robotics and automation (2005): 1730-1735.
[14] N.Shakhovska, S.Fedushko, N.Melnykova, I.Shvorob, Y.Syerov. “Big Data analysis in development of personalized medical system”. Procedia Computer Science, 160, (2019): 229-234.
[15] R.Tkachenko, I.Izonin. “Model and Principles for the Implementation of Neural-Like Structures Based on Geometric Data Transformations”. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education. ICCSEEA 2018. Advances in Intelligent Systems and Computing, vol 754. Springer, Cham (2019)
[16] Dataset https://archive.ics.uci.edu/ml/machine-learning-databases/00374/energydata_complete.csv