sciendo

Int. J. Appl. Math. Comput. Sci., 2021, Vol. 31, No. 1, 135–153
DOI: 10.34768/amcs-2021-0010

amcs

ENSEMBLE LEARNING TECHNIQUES FOR TRANSMISSION QUALITY CLASSIFICATION IN A PAY&REQUIRE MULTI–LAYER NETWORK

DARIUSZ ŻELASKO [a,*], PAWEŁ PŁAWIAK [a,b]

[a]Department of Information and Communications Technology
Cracow University of Technology
ul. Warszawska 24, 31-155 Kraków, Poland
e-mail: {dzelasko,plawiak}@pk.edu.pl

[b]Institute of Theoretical and Applied Informatics
Polish Academy of Sciences
ul. Bałtycka 5, 44-100 Gliwice, Poland
e-mail: plawiak@iitis.pl

Due to a continuous increase in the use of computer networks, it has become important to ensure the quality of data transmission over the network. The key issue in the quality assurance is the translation of parameters describing transmission quality to a certain rating scale. This article presents a technique that allows assessing transmission quality parameters. Thanks to the application of machine learning, it is easy to translate transmission quality parameters, i.e., delay, bandwidth, packet loss ratio and jitter, into a scale understandable by the end user. In this paper we propose six new ensembles of classifiers. Each classification algorithm is combined with preprocessing, cross-validation and genetic optimization. Most ensembles utilize several classification layers in which popular classifiers are used. For the purpose of the machine learning process, we have created a data set consisting of 100 samples described by four features, and the label which describes quality. Our previous research was conducted with respect to single classifiers. The results obtained now, in comparison with the previous ones, are satisfactory—high classification accuracy is reached, along with 94% sensitivity (overall accuracy) with 6/100 incorrect classifications. The suggested solution appears to be reliable and can be successfully applied in practice.

Keywords: Pay&Require, ensemble learning, machine learning, resource allocation, QoS.

1. Introduction

A constantly growing number of techniques used in computer networks has been observed. Networks are currently the basis for the functioning of computer systems. It is hard to imagine the era of computers without the possibility of data transmission over a distance. Thanks to computer networks, communication, access to banking, e-mail and various other services are possible. End users gain access to the network through Internet service providers (ISPs). The user usually pays a monthly fee so that he or she can use the network. Computer networks usually operate on a best-effort basis. Therefore, the user pays a fee for access to the network without a guarantee of any parameters regarding the quality of transmission. These parameters are delay, bandwidth, packet loss ratio and jitter. They affect the quality of data transmission over the network.

Because transmission quality is not guaranteed, the end user often pays for the maximum attainable throughput, which will never be achieved. The inability to achieve the expected throughput is often caused by the volume of traffic on the paths through which data are being sent. Ensuring the quality of service seems to be a very important issue. For this reason, various quality assurance techniques have been defined. One of them are quality of service (QoS). This covers are generally measurable quality parameters that affect the overall quality of the transmission.

*Corresponding author

Usually, the desired transmission quality is obtained by means of applying traffic classification and prioritization. Each administrator can define the QoS rules. For example, it is possible to give higher priority to sensitive data, and lower priority to data that are not crucial from the network functioning point of view. QoS is the general characteristic of service (data transmission) and an attempt to provide the quality at the level expected by the customer (ITU, 2008). QoS can be used to guarantee parameter levels such as constant delay, bandwidth, no packet loss, no jitter (Perez *et al*., 2006; Elnaka and Mahmoud, 2013; Li *et al*., 2019; Ruiz and Finke, 2019).

Software-defined networking (SDN) has recently become a very popular technique in computer networks (IRTF, 2015; Kreutz *et al*., 2015; Nunes *et al*., 2014; Xia *et al*., 2015; Hu *et al*., 2014). SDN is a fully open architecture and it enables dynamic network management. SDN introduces a model consisting of several layers, including (a) a logical layer, in which rules defining network operation are set, and (b) a physical layer, in which network devices operate. The former is responsible for controlling the latter. In SDN, the network is managed centrally. The administrator configures the rules, and then these are used to determine how the entire network should work. The administrator does not have to configure each network node separately. This approach allows defining QoS rules and applying them to the entire network where SDN is used.

Pay&Require (Żelasko *et al*., 2016) is a technique created mainly with respect to the assurance of the quality of data transmission. It is an approach to transmission quality assurance in computer networks and can be used as an alternative to the currently available techniques. The main purpose of using Pay&Require is to guarantee parameters which describe the quality of transmission. To achieve this goal, it is necessary to monitor transmission quality parameters in real time. When the parameters, such as improvement and deterioration, change, network reconfiguration is performed.

Software agents used in Pay&Require are responsible for monitoring and reconfiguring the network (if necessary). They are widely applied in different areas. Sakarkar and Thakar (2009) propose an autonomous counseling agent within educational organization. In turn Sakarkar and Shelke (2009) focus especially on the autonomy of agents in information technology. Agents are also applied in e-commerce (Shuang *et al*., 2007). A general approach to evaluation and comparison is offered by Slhoub *et al*. (2019). Another viewpoint is presented by Tello Leal *et al*. (2014). Agents can also be found in computer networks. A route optimization problem is discussed by Chen *et al*. (2017). There are also works on the use of agents in sensor networks (e.g., Chen *et al*., 2017; Hussain *et al*., 2006). QoS seems to be a very important topic in computer networks. Agents are also used in this field (Acosta and Avresky, 2005; Chowdhury and Neogy, 2012). Other instances are presented by Patri *et al*. (2019) and Grzonka *et al*. (2019).

When employing computer networks, the user usually pays for a certain service (throughput), and the price is fixed. Another approach is provided in Pay&Require, i.e., market based methods of purchasing the transmission quality. The general assumption of Pay&Require is that the user pays for a certain quality (transmission parameters), which is guaranteed. The user gets what he or she pays for. Usually a different approach is used in computer networks—the user pays for the maximum attainable parameters, which are usually not achieved. To guarantee the transmission quality, this term should be well defined. The transmission quality is described by such parameters as delay, bandwidth, packet loss ratio and jitter. These should be guaranteed along the entire path from source to destination. To make this possible, in Pay&Require, the transmission paths must be differentiated. Differentiation is possible thanks to using the appropriate routing protocol in combination with the agent technique.

Based on the parameters describing the transmission quality, it is possible to define certain quality levels. To ensure the quality of transmission at the expected level, it is very important to monitor changes in the quality parameters. If the parameters change, it is necessary to carry out the network reconfiguration process, which means adjusting the transmission parameters to the ones expected by the customers. Software agents are used to monitor the quality of transmission and perform reconfiguration processes. Agents are responsible for verifying if the current parameters meet the customers' expectations. Each customer can expect the quality of transmission on a different level. Differentiation of transmission quality is possible due to that of transmission paths. To perform a proper reconfiguration, it is necessary to specify the transmission quality of all paths. After choosing the paths whose parameters meet the customers' expectations, the routing tables are configured on network devices. This approach enables the service to be provided at a constant level.

A different routing approach is used in Pay&Require. When applying traditional routing protocols, the selection of a specific transmission path depends only on the target network. Policy based routing (PBR) applied in Pay&Require allows path selection based not only on the target network but also on the source of transmission. In this way, differentiation of paths depending on the customer can be achieved. Figure 1 presents an example network consisting of four network devices (PD=$\{R_1, R_2, R_3, R_4\}$), three customers (C=$\{C_1, C_2, C_3\}$) and seven links, four of which are relevant to the concept analysis (L=$\{L_1, L_2, L_3, L_4\}$. Links between network devices

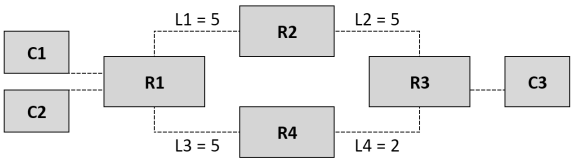

Fig. 1. Sample network.

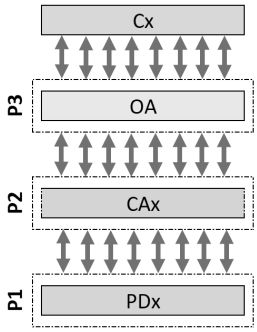

Fig. 2. Pay&Require layer model.

have a specific quality of transmission. The quality of transmission is specified within the range [1, 5]. In the presented example, it is assumed that the customer C_1 expects the quality of transmission 5, and the C_2 customer accepts the transmission quality 2. Two transmission paths are possible, i.e., $TP_1=\{R_1, L_1, R_2, L_2, R_3\}$ and $TP_2=\{R_1, L_3, R_4, L_4, R_3\}$.

The overall path assessment is influenced by single link assessments provided by the bottleneck method, i.e., the final path evaluation is the worst link assessment. This approach is justified because the worst link will affect the overall transmission parameters. Therefore, the path TP_1 has the rating of 5 and the path TP_2 has the rating of 2. In the example shown, both customers, i.e., C_1 and C_2, want to send data to the customer C_3. In the case of traditional routing, transmission would follow the same path regardless of the transmission source. In the case of Pay&Require, C_1–C_3 transmission will take place using the TP_1 path ($Tr_{C1–C2}=\{C_1, TP_1, C_3\}$), which meets the customer's expectations regarding the quality of the transmission. In turn, C_2–C_3 transmission will take place using the TP_2 path ($Tr_{C2–C3}=\{C_2, TP_2, C_3\}$). The selection of paths is made automatically by agents, based on the expectations of single customers. The agents are responsible for carrying out all necessary operations so that the transmission takes place along the right paths.

Pay&Require has a layered structure with plane separation. Figure 2 shows the structure that can be found in Pay&Require. There are three planes, i.e.,

1. Plane 1: network devices (PDx) work in this layer;
2. Plane 2: this layer is created by agents responsible for monitoring and reconfiguring the network;
3. Plane 3: this layer contains an agent responsible for interacting with the customer or his representative (OA). This layer is responsible for handling the transmission quality purchase process.

Cx is a representation of customers. The use of layer separation enables decentralization of network control and management. Decentralization is a good solution because it increases the level of security. Three types of agents have been suggested to enable the decentralization of the solution:

1. *Monitoring agent.* This type of agent is found in every network device. The agent's task is to monitor the transmission quality. To determine, the agent must check transmission parameters such as delay, bandwidth, packet loss ratio and jitter. These parameters must be mapped onto transmission quality on a certain scale—from 1 to 5. In order to do it, first we used translation tables. Unfortunately, they proved to be too complicated to define. Therefore, we decided to apply machine learning (ML) techniques in order to translate measured parameters into transmission quality assessment. The agent knows the needs, and when the transmission parameters change and do not fully meet the customer's expectations, a decision is made to reconfigure the network. This type of agent operates in Plane 2.

2. *Reconfiguration agent.* The agent's task is to perform the reconfiguration process. When the monitoring agent informs about the need for reconfiguration, the agent responsible for this process carries out the necessary steps. First the agent verifies all available paths from source to destination. Then the path evaluation is compared with the customer's expectations and the appropriate paths are selected. The reconfiguration agent works in Plane 2.

3. *Trader agent.* This agent's task is to supervise the transmission quality purchase process. In the simplest case, the customer pays a certain amount of money for a specific quality of transmission. The separation of layers and the application of agents allows you to use other purchase approaches—it is possible, for example, to apply market based methods so that dynamic price shaping can be achieved.

In Pay&Require, the transmission quality is assessed based on the quality parameters describing the transmission, i.e., delay, bandwidth, packet loss ratio and jitter. Translating parameters to a certain scale is a difficult process. The presented solution adopts a scale of grades 1–5. Using this type of scale, it is easy to show to the customer what he or she pays for. Creating reliable

translation rules was complicated; static tables did not work because the combination of parameters could not be translated into quality. Therefore, the decision was made to apply ML techniques. This article presents several algorithms used for the purpose of translation, as well as the results concerning accuracy. It seems that ML is becoming an increasingly popular technique. It is used in a wide spectrum of solutions, i.e., personalized learning systems (Ross *et al*., 2013), underwriting processes (Tan and Zhang, 2005), stock market analysis (Pahwa and Agarwal, 2019), medicine (Tadeusiewicz, 2015; Szaleniec *et al.*, 2013; Huang *et al.*, 2004; Jaworek-Korjakowska and Tadeusiewicz, 2014; 2013; Augustyniak and Tadeusiewicz, 2006; Ogiela *et al.*, 2006; Ogiela and Tadeusiewicz, 2000), chemistry (Szaleniec *et al*., 2008), cybersecurity (Feng *et al*., 2017), credit scoring (Pławiak *et al*., 2020). Also in the case of scheduling, ML can be applied (Memeti *et al*., 2018; Kołodziej *et al*., 2013).

Unfortunately, this technique is still not popular in computer networks. Several papers on ML use in the network traffic classification can be found (Nguyen and Armitage, 2008; Li *et al*., 2007; Yuan *et al*., 2010; Dong *et al*., 2012; Liu *et al*., 2018).

This article presents various algorithms for determining the transmission quality in combination with ML and describes the method of obtaining samples necessary for the learning process. A number of popular classifiers and more complex classification methods are tested. Our contributions are as follows:

- the creation of a data set by obtaining samples from test users,
- presentation of the ML methodology for transmission quality classification,
- different single classifiers and six new ensembles of classifiers coupled with cross-validation and the genetic algorithm (GA) are proposed.

A decision was made to use ML because this solution seems to be reliable and very effective. The problem described in this article is very important in the field of quality assurance in computer networks, especially when the Pay&Require method is applied.

The obtained test data was based on the experience of test users. 100 samples were collected from four test users. The phrase 'one sample' refers to video streaming and website loading recorded with various parameters which describe the quality. This article presents six new ensembles of classifiers that can be used to solve the specified problem, i.e., the assessment of the transmission quality. Each of the algorithms represents a different combination of classifiers and a different approach to the final classification. This is an innovative solution. The best result obtained in the previous studies (single classifiers) was SEN = 89% (Żelasko, 2020). In this research paper, the best result was higher. It was obtained for the algorithm in which three and four classification layers were used. Each layer employs classifiers whose effectiveness was assessed in the previous studies. Other algorithms did not obtain such good results, but they are also interesting with respect to their construction. The obtained results are promising: however, some improvement should be considered in future research. This paper presents a new approach to determining the quality of data transmission over the network, with the application of the Pay&Require method.

The rest of the paper is organized as follows. The overall concept, models and proposed algorithms are described in Section 2. Section 3 presents the analysis of the conducted experiments and their results. Section 4 includes the discussion concerning the concept, the results and conclusions. The paper ends with Section 5. It consists of a summary and plans for further research.

2. Materials and methods

This section presents details concerning the data set used in the ML process, the description of the classification methods and the algorithms applied.

2.1. Data set. The ML method is used to classify the transmission quality. Therefore, obtaining reliable samples is very important. The samples are referred to as subjective assessments of transmission quality obtained from test users. A system was created thanks to which it was possible to show users (a) video streaming, (b) a website with an emphasis on its loading time. To ensure the repeatability of the presented cases, they were recorded and played back to the test users. Each user rated 100 samples. One sample is one case of (a) and (b). The user rated the quality on a scale of 1–5. In this scale, 1 means the lowest and 5 the highest quality. Each sample was registered in a network operating with different transmission quality parameters. Different levels of network traffic were simulated. Trex, the open source traffic generator, was used for this purpose.

The samples were evaluated by means of the quality of experience (QoE) method (Stankiewicz and Jajszczyk, 2011). They were used at the ML training stage. Therefore, it was very important for the test users not to be randomly selected. They had experience in using the computer network. The users rated the quality of website loading and video streaming, while the measured parameters describing the quality were used to create individual samples. One sample stands for the measured parameters (delay, bandwidth, jitter, packet loss ratio) rated by the test user based on the displayed movie and website. The samples were recorded, and then shown to the test users. People who rated the quality use the

Table 1. Details of the data set.

Test users: 4	
Samples: 100	
Parameters	
Input	**Output**
Bandwidth [Mb/s]	Quality [1-5]
Delay [ms]	
Jitter [ms]	
Packet loss ratio [%]	
Classes	
Class	**Samples**
1	15
2	20
3	26
4	20
5	19
Total	100

computer network on a daily basis for work and for private purposes. They come from the age group between 20 and 60. We decided to test non-expert users, but not without experience in the computer network. After obtaining the grades, the single samples evaluation was verified for differences. No major discrepancies were observed, so the final sample grade was determined on the basis of the average grades given by the test users. Details concerning the obtained samples are presented in Table 1.

2.2. Methods. For the purpose of the research, first we applied the classifiers available in the `sklearn` library. Then we used combinations of classifiers and genetic algorithms. The performance assessments can be conducted by means of various measures, e.g., the F1 score, accuracy, specificity, sensitivity or precision. The problem presented in this article is multi-class—we decided to apply the overall accuracy (O_ACC), which means sensitivity (SEN) (Sammut and Webb, 2010). In the conducted SEN studies, we used the accuracy score function from the `sklearn` library. The following parameters were employed to evaluate single algorithms and to select the best solution:

1. confusion matrices—for assessment of classification errors (Sammut and Webb, 2010),
2. F1 score—as an adaptation function (Sasaki, 2007; *F1 Score,* 2021*)*,
3. sensitivity (SEN)—the overall accuracy ratio (Sammut and Webb, 2010).

The best algorithm was chosen on the basis of the F1 score. The achieved results were presented by means of SEN and confusion matrices.

The following classifiers were used in the research:

- Nu-SVC (Cortes and Vapnik, 1995; Pławiak *et al*., 2019; Chang and Lin, 2011),
- kNN (Altman, 1992),
- random forest (Breiman, 2001),
- radius neighbors (Altman, 1992),
- nearest centroid (Tibshirani *et al*., 2002),
- extra trees (Geurts *et al*., 2006),
- linear SVC (Fan *et al*., 2008),
- C-SVC (Cortes and Vapnik, 1995; Pławiak *et al*., 2019; Chang and Lin, 2011),
- stacking classifier (Wolpert, 1992),
- voting classifier (Yong *et al*., 2014; Onan *et al*., 2016),
- logistic regression (Hsiang-Fu *et al*., 2011).

The parameters of single classifiers used in the research are presented in Table 2. The meaning of the individual parameters is as follows:

1. Nu-SVC (*NuSVC*, 2021):
   - kernel—specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'RBF', 'sigmoid', 'precomputed';
   - nu—an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors;
   - degree—the degree of the polynomial kernel function ('poly'). Ignored by all other kernels;
   - gamma—kernel coefficient for 'RBF', 'poly' and 'sigmoid'.
2. kNN (*KNeighborsClassifier*, 2021):
   - neighbors—the number of neighbors to be used by default for neighbors queries,
   - weights—the weight function used in prediction,
   - algorithm—the algorithm used to compute the nearest neighbors,
   - leaf_size—the leaf size passed to BallTree or KDTree.
3. Random forest (*RandomForestClassifier*, 2021):
   - n_estimators—the number of trees in the forest,
   - max_depth—the maximum depth of the tree,

- random_state—controls both the randomness of the bootstrapping of the samples used when building trees,
- max_samples—the number of samples to draw from X to train each base estimator.

4. Radius neighbors (*RadiusNeighborsClassifier*, 2021):
   - radius—the range of parameter space to use,
   - algorithm—the algorithm used to compute the nearest neighbors,
   - leaf_size—the leaf size passed to BallTree or KDTree,
   - weights—the weight function used in prediction.
5. Extra trees (*ExtraTreesClassifier*, 2021):
   - n_estimators—the number of trees in the forest.
6. C-SVC (*C-SVC*, 2021):
   - kernel—specifies the kernel type to be used in the algorithm,
   - coef0—an independent term in the kernel function. It is only significant in 'poly' and 'sigmoid',
   - degree—the degree of the polynomial kernel function,
   - gamma—the kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
7. Linear SVC (*LinearSVC*, 2021)
   - random_state—controls the pseudo random number generation for shuffling the data for the dual coordinate descent,
   - tol—tolerance for stopping criteria,
   - c—the regularization parameter; the strength of the regularization is inversely proportional to c.

The same parameter ranges for each classifier were used for each algorithm. The chromosome model is presented in Fig. 3. This type of chromosome appeared in all presented algorithms. Depending on the individual algorithms, the hyperparameters in the chromosome are different. Each estimator uses different parameters and their values (Table 2). At the beginning of the learning process, initial hyperparameters are randomly selected (from the ranges specified in Table 2) and a specific chromosome is created containing only those parameters which will be used in specific estimators depending on the algorithm applied. The GA was used to optimize the hyperparameters of estimators, with the following parameters: the population of 100 individuals and of 100 generations, with the probability of crossover at 0.4, and the mutation 0.9 (Table 3); the F1 score or accuracy as the fitness function. We used CxTwoPoint crossover type available in the `sklearn` library. We employed our own function to perform mutation. The function used for the mutation at each mutation randomizes the new value of one of the hyperparameters in the chromosome. We applied 10-fold stratified cross validation coupling with genetic optimization of hyperparameters. The data were divided into testing and training sets, the learning process was carried out on training sets, and the classification accuracy was assessed by means of testing sets. In the following subsections the classification algorithms are presented.

2.3. Algorithm 1. The first algorithm uses voting classifier available in the `sklearn` library in combination with several classifiers. This process is shown in Fig. 4. It was divided into the following steps:

1. *Preprocessing:* Several different rescalling methods were tested for preprocessing, i.e., MaxAbsScaler, MinMaxScaler, RobustScaler, Normalizer and StandardScaler. The purpose of rescalling was to get data in a specific range of values.
2. *Cross-validation:* We applied 10-fold stratified cross-validation. This means that 10 combinations of training and testing data collections were created. It was tested with all possible combinations of preprocessing.
3. *Classification:* The voting classifier was used in combination with random forest, linear SVC, the nearest centroid, C-SVC, kNN, extra trees as estimators.
4. *Parameter optimization:* the GA was applied to optimize the classifier parameters.

2.4. Algorithm 2. In our previous research (Żelasko, 2020), the stacking classifier yielded a very good result, therefore it was decided to extend this method by adding more estimation layers. The whole process is shown in Fig. 5. It consists of the following stages:

1. *Preprocessing:* several different rescalling methods were tested for preprocessing, i.e., MinMaxScaler, MaxAbsScaler, StandardScaler, RobustScaler, and Normalizer. The purpose of using rescalling was to get data values in a specific range.
2. *Cross-validation:* 10-fold stratified cross-validation.
3. *Classification:* we utilized two layers of estimators. In Layer 1, linear SVC, the nearest centroid, kNN

Table 2. Classifier parameters.

Classifier	Parameter 1	Parameter 2	Parameter 3	Parameter 4
Nu-SVC	kernel=(linear, rbf, poly, sigmoid)	nu=(0.001 - 0.5)	degree=(1 - 5)	gamma=(0.001 - 2)
kNN	neighbors=(1 - 7)	weights=(uniform, distance)	algorithm=(ball_tree, kd_tree, brute, auto)	leaf_size=(15 - 45)
Random Forest	n_estimators=(50 - 200)	max_depth=(1 - 20)	random_state=(0 - 20)	max_samples=(1 - 64)
Radius Neighbors	radius=(5 - 15)	algorithm=(auto, ball_tree, kd_tree, brute)	leaf_size=(15 - 45)	wights=(uniform, distance)
Nearest Centroid	-	-	-	-
Extra Trees	n_estimators=(50 - 200)	-	-	-
C-SVC	kernel=(linear, rbf, poly, sigmoid)	coef0=(0 - 0.5)	degree=(1 - 6)	gamma=(0.001 - 2)
Linear SVC	random_state=(0 - 50)	tol=(1e-6 - 1e-4)	c=(0.5 - 2.0)	-

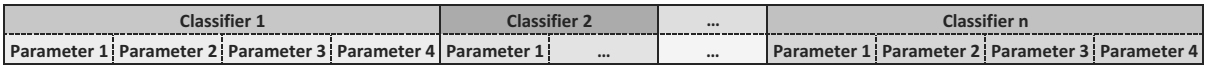

Fig. 3. Chromosome model.

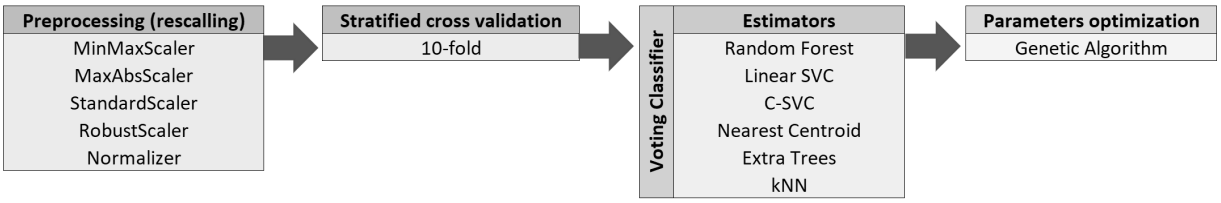

Fig. 4. Flowchart of ML for Algorithm 1.

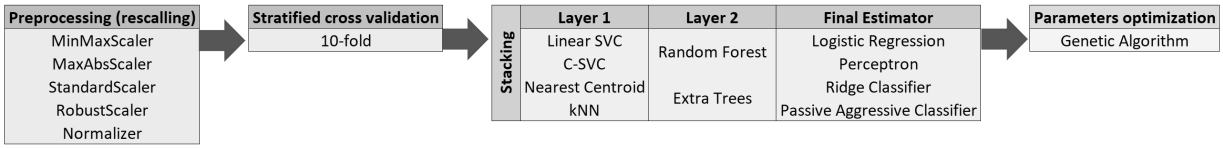

Fig. 5. Flowchart of ML for Algorithm 2.

and C-SVC were used as estimators. Layer 2 contains a random forest and extra trees. One method was used in the final estimation. Four methods were available so it was verified how each of them would handle the final estimation.

4. *Parameter optimization:* the GA was applied to optimize the classifier parameters.

2.5. Algorithm 3. The third algorithm is divided into three different approaches differing in the number of classification layers. Variant 1 (Fig. 6) uses two layers—Layer 1 consists of 5 classifiers. First, preprocessing is performed for each of them. The choice of preprocessing methods is not accidental—it results from previous studies (Żelasko, 2020). The preprocessing methods for which single classification methods produced the best results were selected. The preprocessing stage is followed by 10-fold stratified cross-validation.

Data prepared in this way are given to classifiers in order to teach them. Layer 2 is the final classification. At the input of Layer 2, the classification results for each of the classifiers used in Layer 1 appear. In Layer 2, 10-fold stratified cross validation is also used. The final classifier is logistic regression. The last step is the optimization of parameters performed by the GA. What is important, the GA optimizes the parameters of all classifiers simultaneously, i.e., both in Layer 1 and Layer 2.

Variant 2 differs from Variant 1 in that an additional classification layer is added (Fig. 6). The principle of operation of Layer 1 remains unchanged, while Layer 3 in this variant works on the same principle as Layer 2 in Variant 1. In this variant, Layer 2 uses three classifiers. At the entry to Layer 2, the results of the classification from Layer 1 appear. Then preprocessing is performed; also in

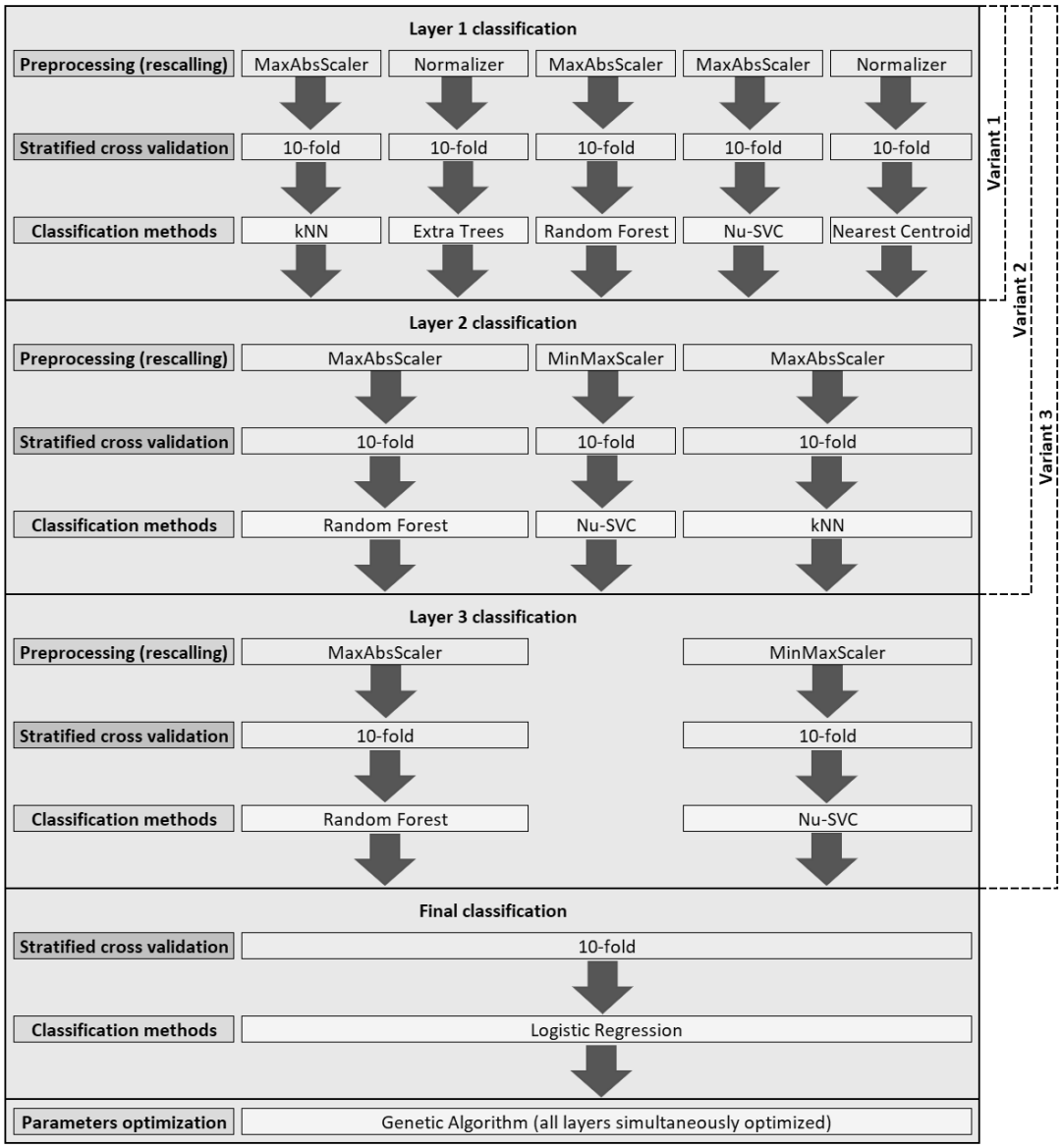

Fig. 6. Flowchart of ML for Algorithm 3.

this case the selection of methods is not accidental—it is based on the experience from previous research. The next step is 10-fold stratified cross-validation. The last stage in this layer is classification. The result from this layer is given at the entry to the next layer.

Variant 3 is an extension of Variant 2 with another classification layer. Layer 3 was added with two classifiers. The order of the steps in this layer is the same as in the previous layer, i.e., preprocessing, cross-validation and classification. Classification results from this layer are given as the input of the layer performing the final classification.

For all variants, the GA is responsible for optimizing the parameters of all layers.

2.6. Algorithm 4. Two layers of classification are used in this algorithm (Fig. 7). Layer 1 uses five different classifiers. The first step is preprocessing—for each classifier the method was chosen based on our previous research (Żelasko, 2020). The next step is cross-validation. Then, classification is carried out. Single classifiers are evaluated on the basis of the F1 score. After classification, the GA verifies its result and performs optimization of parameters for each classifier separately. Layer 2 receives Layer 1 classification results as the input, and cross-validation is carried out. Then the final classification is made. Several different classifiers were tested, namely, logistic regression, perceptron, the ridge classifier and the passive aggressive classifier. The quality

of the entire classification was determined on the basis of SEN.

2.7. Algorithm 5. Two layers of classification are used in this algorithm (Fig. 8). In Layer 1, classifiers are trained to classify all classes, but with the assumption that they should specialize in one class: kNN classifies Class 1, extra trees–Class 2, and random forest—Class 3, respectively. In turn, Nu-SVC specializes in classifying samples in Class 4, and the nearest centroid in Class 5. Preprocessing is performed for each classifier—the selection of methods is based on the previous studies (Żelasko, 2020). Then, 10-fold stratified cross-validation is performed. The parameters of each classifier are separately optimized by the GA. It is done for the purpose of specializing the classifier to recognize a particular class. The classification of a single classifier is evaluated on the basis of the weighted F1 score. The weight for a sample belonging to the class in which a given classifier is to specialize has a much higher value than for the others.

Layer 2 constitutes the final classification based on the results obtained in Layer 1. The classification is performed using weighted majority voting. It makes the final classification based on the credibility of Layer 1 classification, i.e., the final classification is influenced by the specialization of the classifier from Layer 1. If the kNN classifier (specialization of Class 1) indicates that a given sample belongs to Class 1, then such classification has higher weight than the indications of other classifiers. If the nearest centroid classifier indicates that the sample belongs to Class 5, then such classification is more reliable than the others. For this purpose, appropriate weights are utilized. If a classifier specializes in Class 5, the classification indicates a different class, but the weight of such an indication remains the same as in the others. Based on the obtained results—classification weights, weighted majority voting classifies samples into a specific class, i.e., one that is indicated by the majority of Layer 1 classifiers, when taking into account the specialization of classifiers. The quality of the entire classification is determined by SEN.

2.8. Algorithm 6. This algorithm is most complicated in terms of implementation and complexity. It uses three layers (Fig. 9). The first layer is divided into two parts. There are five classifiers in each of them. Each classifier specializes in classifying samples of a particular class. This means that for Class 1 in Layer 1 there are two classifiers specialized in classifying samples in this class. At first, preprocessing is performed (separately for each classifier), followed by cross-validation. Classification performance is determined by the weighted F1 score. The weights depend on the class in which the classifier specializes. The obtained result is verified by the GA and the parameters are optimized separately for each classifier.

At the same time, the second layer is taught. In this layer, each classifier specializes in a different classification variant. Variants mean specialization in the classification of samples in two different classes. Possible variants are

1. Classes 1 and 2,
2. Classes 1 and 3,
3. Classes 1 and 4,
4. Classes 1 and 5,
5. Classes 2 and 3,
6. Classes 2 and 4,
7. Classes 2 and 5,
8. Classes 3 and 4,
9. Classes 3 and 5,
10. Classes 4 and 5.

The purpose of the variants is to specialize classifiers to recognize differences between two classes. The learning process for each variant consists of preprocessing, cross-validation, classification and parameter optimization. Classification performance is determined by the weighted F1 score.

Layer 3 makes the final classification. An algorithm which gets as the input the results of classification from Layers 1 and 2 is applied. If the indication of classifiers in Layers 1.1 and 1.2 specialized in a given class is identical, it is assumed to be correct and final. If the indications are different, then the final classification is determined by the indication of the classifier of Layer 2. If the first layer classifier specified in Class 1 indicates 1 and the classifier specified in Class 2 indicates 2, the final classification is specified based on the indication of the Layer 2 classifier, Variant 1. In a situation, when Layer 1 classifiers do not make an indication in accordance with the specialization, the final classification is based on the majority of Layer 1 classifiers.

3. Results

Pay&Require is an approach which ensures the quality of transmission in computer networks, using agents to monitor and assess the quality of data transmission. The quality is provided at the level expected by the customer. This goal is achieved by using the differentiation of transmission paths and the possibility of their dynamic modification during network operation. We decided to use ML to ensure reliable transmission quality assessment

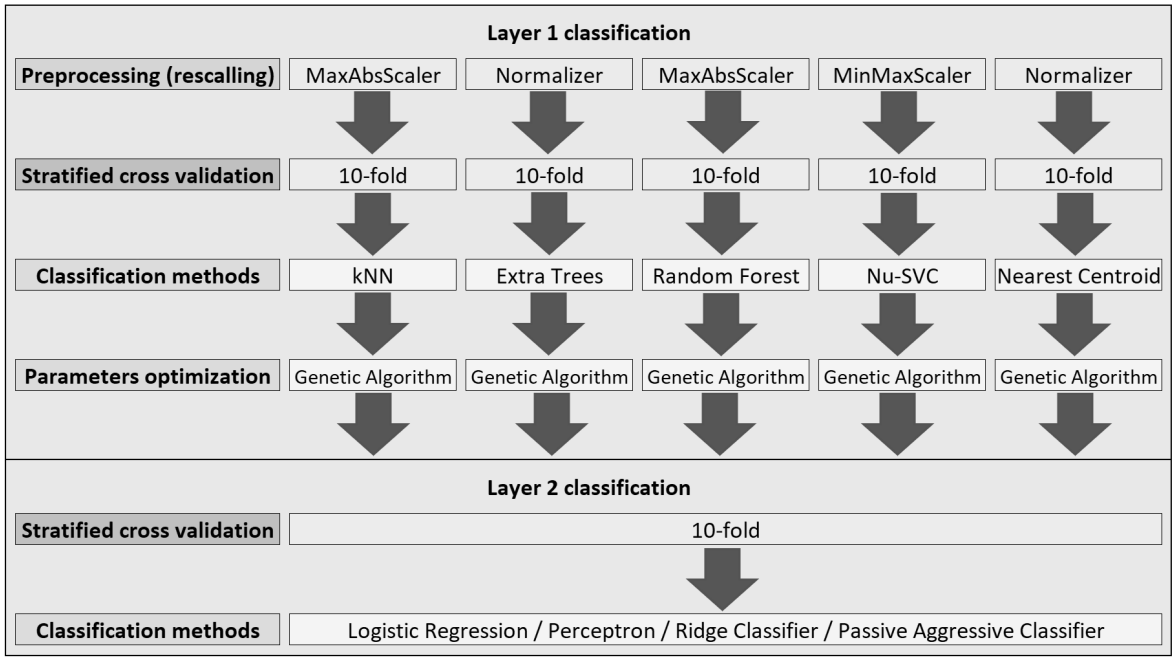

Fig. 7. Flowchart of ML Algorithm 4.

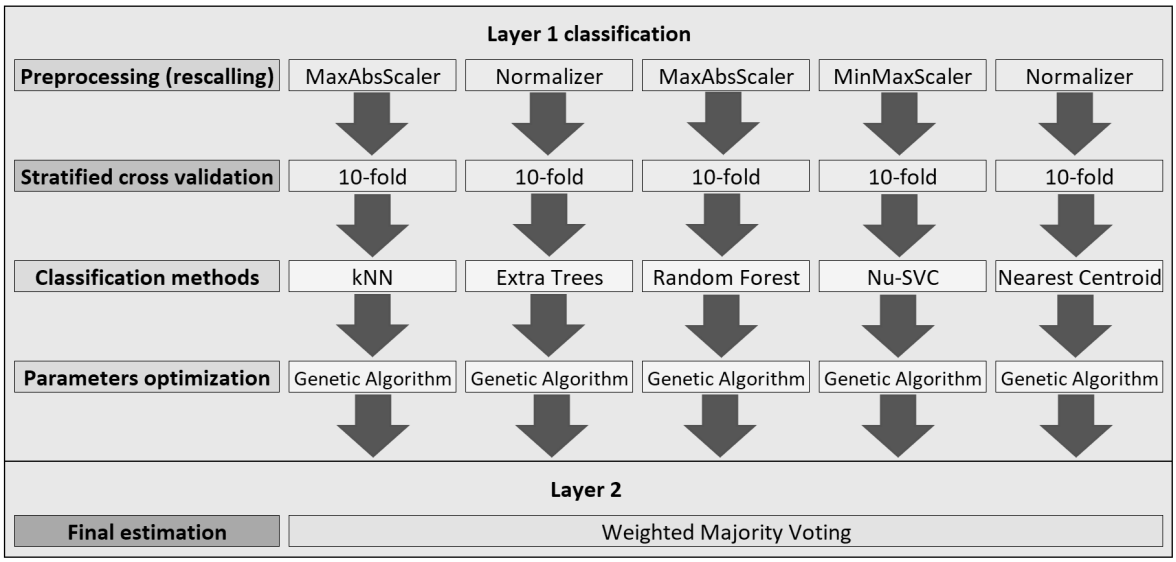

Fig. 8. Flowchart of ML Algorithm 5.

based on the parameters describing it. These parameters are measured by agents in time that can be considered real. The customer always receives the transmission service at the same expected level of quality.

The code was written in the Python programming language. The `sklearn` and `deap` libraries were used in the conducted research. The results are presented for the test data set. The selection of specific preprocessing or classification methods was not accidental and was based on the results obtained in our previous studies (Żelasko, 2020). Each of the algorithms was checked with the parameters depicted in Table 3. The impact of the presented parameters on the results in ML is significant. The selection of parameters was based on previously conducted research (Żelasko, 2020) and our own experience in machine learning, (e.g., Pławiak *et al.*, 2020; 2019). On this basis, a combination of parameters was chosen.

For each algorithm, we utilized 10-fold stratified cross-validation, and each of the algorithms operated on

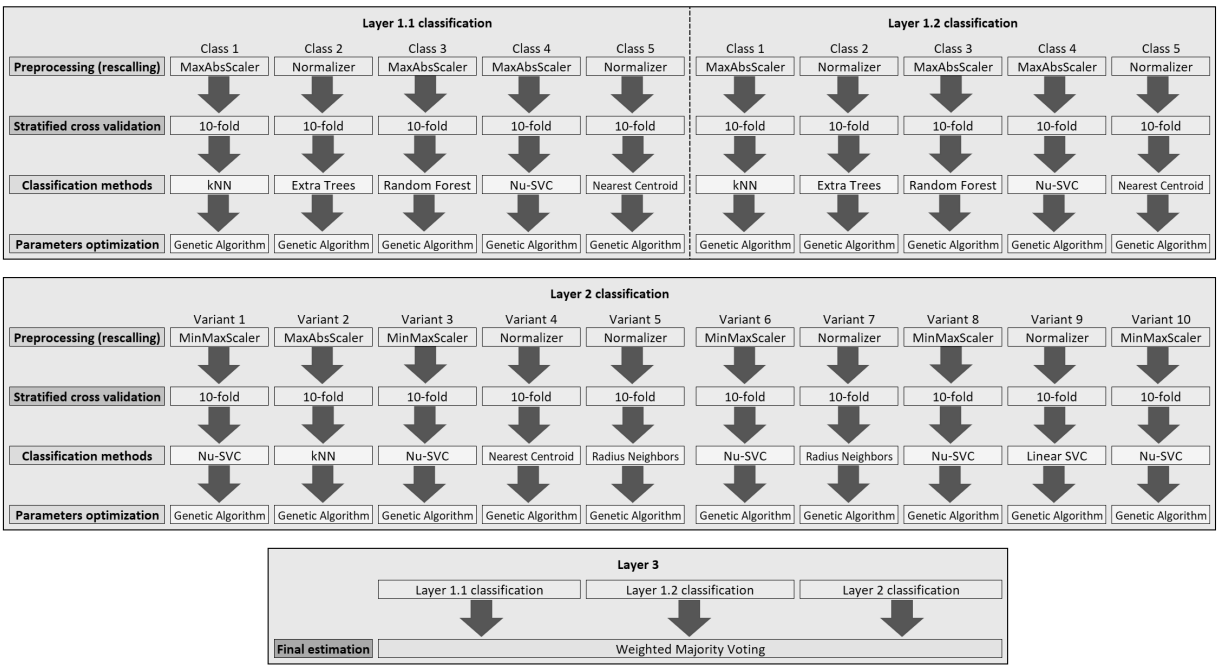

Fig. 9. Flowchart of ML for Algorithm 6.

Table 3. GA parameters.

Parameter	Value
Population size	100
Number of generations	100
Crossover probability	0.4
Probability of mutation	0.9
Selection algorithm	Tournment (size=3)
Fitness function	Accuracy or F1 score

Table 4. Results obtained in previous research.

Classifier	O_ACC = SEN
Stacking Classifier	89.00%
Nu-SVC	87.00%
Random Forest	87.00%
Extra Trees	85.00%
kNN	84.00%
Decision Tree	84.00%
C-SVC	83.00%
Radius Neighbors	83.00%
Nearest Centroid	81.00%
Linear SVC	75.00%

all four features, i.e., delay, bandwidth, packet loss ratio and jitter. This article presents the best results for each algorithm. In order to determine the effectiveness of the classification and to verify how algorithms deal with each class, we applied SEN and confusion matrices.

The results from the previous research are presented in Table 4. These were obtained for single classifiers and for a stacking classifier, using several single classifiers. The best result was obtained for the stacking classifier and it was SEN=89%.

Table 5 presents parameters used for the best result of Algorithm 1, where SEN equal to 85.00% was obtained. In turn, Fig. 10 presents the confusion matrix for Algorithm 1. It can be stated that Algorithm 1 did the best with Class 5, with SEN = 100.00%. A good result was also obtained for Class 1 (93.33%). The worst result, 60.00% was obtained in the case of Class 4.

In turn, the parameters of Algorithm 2 are presented in Table 6. SEN=90.00% was obtained. Two classification layers were used for this algorithm. Figure 11 presents the confusion matrix for the algorithm. The classification was highest for Class 1, i.e., SEN = 100.00%. Satisfactory results were also obtained for Classes 3 and 5. The worst result was obtained for Class 4, i.e., 70.00%.

The next tested algorithm was Algorithm 3. It has three variants. The parameters for all variants are shown in Table 7. For Variant 1 SEN = 91.00% was achieved. For Variant 2 SEN was equal to 94.00% and for Variant 3 SEN result of 94.00% was also obtained. Figure 12 presents the confusion matrices. It can be seen that Variant 1 of Algorithm 3 worked very well for Classes 1 and 5, i.e., SEN = 100.00%. For Classes 2 and 3 it was above 90.00%. The worst result, 75.00%, was obtained for Class 4. In the case of Variant 2 the best result was

Table 5. Parameters used for the best classification result of Algorithm 1.

Rescalling	Estimator	Parameter 1	Parameter 2	Parameter 3	Parameter 4	O_ACC = SEN
Normalizer	Random Forest	n_estimators=64	max_depth=20	random_state=2	max_samples=56	85.00%
	Linear SVC	random_state=29	tol=6.53463e-06	c=1.378117402	-	
	C-SVC	kernel='rbf'	coef0=0.256643826	degree=4	gamma=0.358487459	
	Nearest Centroid	-	-	-	-	
	Extra Trees	n_estimators=67	-	-	-	
	kNN	n_neighbors=1	weights='distance'	algorithm='brute'	leaf_size=33	

Actual \ Predicted	1	2	3	4	5
1	14	1	0	0	0
2	2	17	1	0	0
3	0	1	23	2	0
4	0	0	5	12	3
5	0	0	0	0	19

	1		2		3		4		5	
14	1	93.33% 6.67%	17	3	85.00% 15.00%	23	3	88.46% 11.54%	12	8	60.00% 40.00%	19	0	100.00% 0.00%
2	83	97.65% 2.35%	2	78	97.50% 2.50%	6	68	91.89% 8.11%	2	78	97.50% 2.50%	3	78	96.30% 3.70%
87.50% 12.50%	98.81% 1.19%	**93.33%** 6.67%	89.47% 10.53%	96.30% 3.70%	**85.00%** 15.00%	79.31% 20.69%	95.77% 4.23%	**88.46%** 11.54%	85.71% 14.29%	90.70% 9.30%	**60.00%** 40.00%	86.36% 13.64%	100.00% 0.00%	**100.00%** 0.00%

Fig. 10. Confusion matrix: Algorithm 1, best result (SEN = 85%). SEN was calculated for individual classes.

Table 6. Parameters used for the best classification result of Algorithm 2.

Rescalling	Layer	Estimator	Parameter 1	Parameter 2	Parameter 3	Parameter 4	O_ACC = SEN
MaxAbsScaler	1	Linear SVC	random_state=21	tol=4.83839e-05	c=0.982870391	-	90.00%
		C-SVC	kernel='poly'	coef0=0.006357180	degree=1	gamma=1.331759915	
		Nearest Centroid	-	-	-	-	
		kNN	n_neighbors=3	weights='uniform'	algorithm='kd_tree'	leaf_size=25	
	2	Random Forest	n_estimators=75	max_depth=17	random_state=14	max_samples=32	
		Extra Trees	n_estimators=154	-	-	-	

Actual \ Predicted	1	2	3	4	5
1	15	0	0	0	0
2	0	18	2	0	0
3	0	0	25	0	0
4	0	0	3	14	3
5	0	0	0	0	18

	1		2		3		4		5	
15	0	100.00% 0.00%	18	2	90.00% 10.00%	25	1	96.15% 3.85%	14	6	70.00% 30.00%	18	1	94.74% 5.26%
0	85	100.00% 0.00%	0	80	100.00% 0.00%	5	69	93.24% 6.76%	2	78	97.50% 2.50%	3	78	96.30% 3.70%
100.00% 0.00%	100.00% 0.00%	**100.00%** 0.00%	100.00% 0.00%	97.56% 2.44%	**90.00%** 10.00%	83.33% 16.67%	98.57% 1.43%	**96.15%** 3.85%	87.50% 12.50%	92.86% 7.14%	**70.00%** 30.00%	85.71% 14.29%	98.73% 1.27%	**94.74%** 5.26%

Fig. 11. Confusion matrix: Algorithm 2, best result (SEN = 90%). SEN was calculated for individual classes.

obtained for Classes 1 and 5, i.e., 100.00%. In turn, for Classes 2 and 3 a result above 95.00% was obtained. Also in this case the worst classification result was obtained for Class 4, it was SEN = 80.00%. In Variant 3 the best SEN was obtained for Classes 1 and 5, i.e., 100.00%, and the worst for Class 4 (80.00%).

The parameters of the next algorithm, i.e., Algorithm 4, are shown in Table 8. The best obtained result was SEN = 85.00%. For Classes 1, 2 and 3 a result above 90.00% was achieved as shown in Fig. 13. The worst result was 65.00%, which was obtained for Class 4.

In the case of Algorithm 5, the SEN result of 87.00% was obtained for the parameters presented in Table 9. The confusion matrix is shown in Fig. 14. It can be seen that the algorithm achieved the best result for Classes 1 and 5 (100.00%), and the worst for Class 4 (65.00%).

The last algorithm is Algorithm 6. For the best classification case, i.e., SEN = 86.00%, the parameters are presented in Table 10. In turn, the confusion matrix is shown in Fig. 15. The algorithm coped best with Classes 3 and 5 (100.00%); for Classes 1 and 2 the result was around 85.00%. The worst result was obtained for Class 4 (55.00%).

In Fig. 16, the results obtained for single classifiers and stacking classifier from the previous research are presented. For the purpose of the present research, a summary of the SEN for all of the algorithms in individual classes is shown in Fig. 17. The summary of the results (SEN) in the previous research and those presented in this article are depicted in Fig. 18.

Measurement is not a continuous process, it can be done less or more frequently depending on the system configuration. It should be emphasized that verification of the quality parameters is a process that, depending on the size of the network, lasts a certain amount of time. Therefore, the speed of classification is not important. During the study it was found that the classification takes a maximum of a few seconds. Complexity is also unimportant because it has an impact on the learning process, but not on the final classification process. Therefore, no time and complexity results were presented—insignificant from the system point of view.

Generally, classification of the quality parameters is very important when we use Pay&Require. Based on the quality parameters, the overall quality of service is determined. The user pays a certain amount of money when using the network at the expected quality level. The amount payable varies with respect to the quality level.

Table 7. Parameters used for the best classification result of Algorithm 3.

Variant / Layer	Estimator	Rescalling	Parameter 1	Parameter 2	Parameter 3	Parameter 4	O_ACC = SEN
1 / 1	kNN	MaxAbsScaler	n_neighbors=3	weights='uniform'	algorithm='brute'	leaf_size=45	91.00%
	Extra Trees	Normalizer	n_estimators=154	-	-	-	
	Random Forest	MaxAbsScaler	n_estimators=174	max_depth=12	random_state=15	max_samples=18	
	Nu-SVC	MaxAbsScaler	kernel='rbf'	nu=0.270444645	degree=5	gamma=2.619916864	
	Nearest Centroid	Normalizer	-	-	-	-	
2 / 1	kNN	MaxAbsScaler	n_neighbors=3	weights='uniform'	algorithm='kd_tree"	leaf_size=36	94.00%
	Extra Trees	Normalizer	n_estimators=172	-	-	-	
	Random Forest	MaxAbsScaler	n_estimators=111	max_depth=13	random_state=14	max_samples=29	
	Nu-SVC	MaxAbsScaler	kernel='poly'	nu=0.435555251	degree=3	gamma=2.384872936	
	Nearest Centroid	Normalizer	-	-	-	-	
2 / 2	Random Forest	MaxAbsScaler	n_estimators=193	max_depth=4	random_state=14	max_samples=9	
	Nu-SVC	MinMaxScaler	kernel='poly'	nu=0.069852823	degree=1	gamma=2.295198665	
	kNN	MaxAbsScaler	n_neighbors=3	weights='distance'	algorithm='kd_tree'	leaf_size=33	
3 / 1	kNN	MaxAbsScaler	n_neighbors=4	weights='distance'	algorithm='kd_tree'	leaf_size=43	94.00%
	Extra Trees	Normalizer	n_estimators=126	-	-	-	
	Random Forest	MaxAbsScaler	n_estimators=175	max_depth=8	random_state=20	max_samples=63	
	Nu-SVC	MaxAbsScaler	kernel='rbf'	nu=0.228686519	degree=4	gamma=0.110472750	
	Nearest Centroid	Normalizer	-	-	-	-	
3 / 2	Random Forest	MaxAbsScaler	n_estimators=57	max_depth=9	random_state=10	max_samples=14	
	Nu-SVC	MinMaxScaler	kernel='rbf'	nu=0.084774299	3	gamma=0.944700151	
	kNN	MaxAbsScaler	n_neighbors=7	weights='distance'	algorithm='brute'	leaf_size=17	
3 / 3	Random Forest	MaxAbsScaler	n_estimators=196	max_depth=7	random_state=7	max_samples=39	
	Nu-SVC	MinMaxScaler	kernel='rbf'	nu=0.467701715	degree=3	gamma=0.312846365	
All / Last	LogisticRegression	-	-	-	-	-	

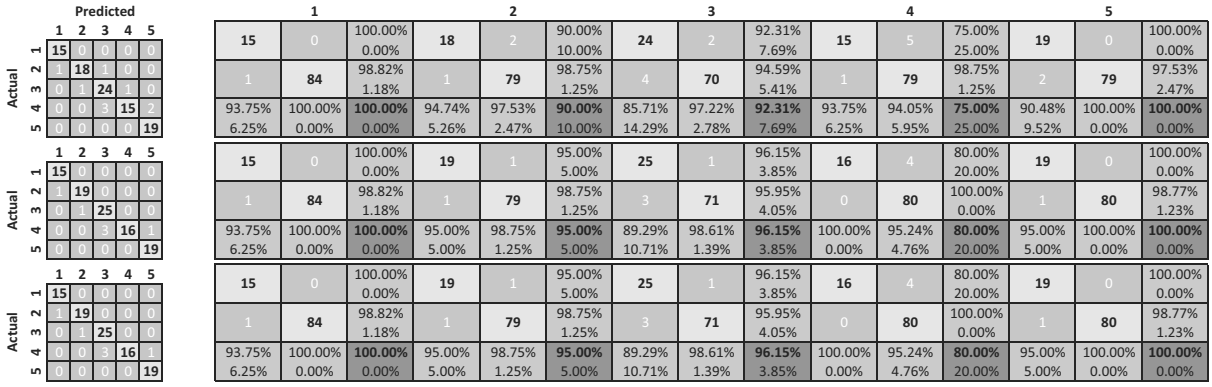

Fig. 12. Confusion matrices: Algorithm 3, best results. SEN was calculated for individual classes.

4. Discussion

This paper presents the application of the Pay&Require method in determining the quality of data transmission over the network. Quality assurance is achieved by means of using a multi-agent system (to monitor and reconfigure the network), and by using PBR—a technique in which packets can be transmitted through different paths depending on the network configuration. The basic assumption of the Pay&Require method is differentiation of the path. The transmission quality is described by four parameters: delay, bandwidth, packet loss ratio and jitter. These parameters affect the quality of transmission perceived by the customer.

In Pay&Require the user pays for a specific transmission quality and this quality is guaranteed. This quality assurance is achieved by choosing the appropriate transmission paths—those through which transmission is performed with a certain quality. In traditional computer networks, the choice of the transmission path relies on the destination and not on the transmission source. In the Pay&Require method, the selection of a specific path depends not only on the destination, but also on the source, which allows path differentiation.

It may be difficult for the customer to understand how the parameters describing the quality of transmission will translate into the service he receives. Therefore, we should define a simple scale that allows the quality of transmission to be assessed in a way that customer understands. The scale [1–5] was adopted; however, the translation of parameters describing transmission quality (delay, bandwidth, packet loss ratio and jitter) on a certain scale turned out to be a complicated problem. The use of translation tables was difficult and not very effective. For this reason, we decided to use ML to translate parameters describing the quality of transmission on a certain scale. To carry out the ML process, it was necessary to obtain the test data. Ratings based on QoE

Table 8. Parameters used for the best classification result of Algorithm 4.

Estimator	Rescalling	Parameter 1	Parameter 2	Parameter 3	Parameter 4	O_ACC = SEN
kNN	MaxAbsScaler	n_neighbors=3	weights='uniform'	algorithm='brute'	leaf_size=15	85.00%
Extra Trees	Normalizer	n_estimators=74	-	-	-	
Random Forest	MaxAbsScaler	n_estimators=113	max_depth=2	random_state=13	max_samples=27	
Nu-SVC	MinMaxScaler	kernel='sigmoid'	nu=0.196159703	degree=7	gamma=2.363264651	
Nearest Centroid	Normalizer	-	-	-	-	

Actual \ Predicted	1	2	3	4	5
1	14	1	0	0	0
2	1	18	1	0	0
3	0	1	24	1	0
4	0	0	3	13	4
5	0	0	0	3	16

	1			2			3			4			5		
1	14	1	93.33% 6.67%	18	2	90.00% 10.00%	24	2	92.31% 7.69%	13	7	65.00% 35.00%	16	3	84.21% 15.79%
2	1	84	98.82% 1.18%	2	78	97.50% 2.50%	4	70	94.59% 5.41%	4	76	95.00% 5.00%	4	77	95.06% 4.94%
3	93.33% 6.67%	98.82% 1.18%	**93.33%** 6.67%	90.00% 10.00%	97.50% 2.50%	**90.00%** 10.00%	85.71% 14.29%	97.22% 2.78%	**92.31%** 7.69%	76.47% 23.53%	91.57% 8.43%	**65.00%** 35.00%	80.00% 20.00%	96.25% 3.75%	**84.21%** 15.79%

Fig. 13. Confusion matrix: Algorithm 4, best result (SEN = 85%). SEN was calculated for individual classes.

Table 9. Parameters used for the best classification result of Algorithm 5.

Estimator	Rescalling	Parameter 1	Parameter 2	Parameter 3	Parameter 4	O_ACC = SEN
kNN	MaxAbsScaler	n_neighbors=3	weights='uniform'	algorithm='brute'	leaf_size=21	87.00%
Extra Trees	Normalizer	n_estimators=102	-	-	-	
Random Forest	MaxAbsScaler	n_estimators=72	max_depth=2	random_state=16	max_samples=14	
Nu-SVC	MinMaxScaler	kernel='poly'	nu=0.303701374	degree=4	gamma=2.388222064	
Nearest Centroid	Normalizer	-	-	-	-	

Actual \ Predicted	1	2	3	4	5
1	15	0	0	0	0
2	2	16	2	0	0
3	0	1	24	1	0
4	0	0	3	13	4
5	0	0	0	0	19

	1			2			3			4			5		
1	15	0	100.00% 0.00%	16	4	80.00% 20.00%	24	2	92.31% 7.69%	13	7	65.00% 35.00%	19	0	100.00% 0.00%
2	2	83	97.65% 2.35%	1	79	98.75% 1.25%	5	69	93.24% 6.76%	1	79	98.75% 1.25%	4	77	95.06% 4.94%
3	88.24% 11.76%	100.00% 0.00%	**100.00%** 0.00%	94.12% 5.88%	95.18% 4.82%	**80.00%** 20.00%	82.76% 17.24%	97.18% 2.82%	**92.31%** 7.69%	92.86% 7.14%	91.86% 8.14%	**65.00%** 35.00%	82.61% 17.39%	100.00% 0.00%	**100.00%** 0.00%

Fig. 14. Confusion matrix: Algorithm 5, best result (SEN = 87%). SEN was calculated for individual classes.

come from four users. Test samples were prepared with various transmission quality parameters. 100 samples were collected. One sample stands for movie streaming and the website loading process. The test users rated the samples on a scale from 1 to 5, based on their own experience. The results we obtained were used in the ML process. Classification results were presented in this article.

This paper presented six new ensemble learning methods. The obtained results are better in most cases in comparison to the previous studies (Żelasko, 2020). In the previous research, the best result was SEN=89.00%. In the current research, Algorithms 1, 4, 5 and 6 produced worse results compared to the previous studies. Algorithm 1 there the use of the voting classifier in combination with several estimators. In the case of three classes, the result was below 90.00%, which is not satisfactory. Algorithm 4 is the use of two classification layers; the first employs five different classifiers. The worst result was obtained in two classes: 4 and 5. In the other classes the result was slightly above 90.00%. Algorithm 5 also utilizes two layers of classification, with Layer 2 using weighted majority voting for the final classification. This algorithm had a very good result for Classes 1 and 5 (100.00%); unfortunately, for Classes 2 and 4 the result was below 80.00% which is unsatisfactory. Another algorithm, i.e., Algorithm 6, produced a very good result for Classes 3 and 5, i.e., 100.00%. The worst result was obtained for Class 4: 55.00%.

Algorithms that did better than the best single classifier in previous studies are Algorithms 2 and 3 (in all variants). Algorithm 2 utilizes stacking classifier in combination with several layers of classification. Fairly good results were obtained for all classes (over 90.00%), except for Class 4, which got 70.00%. In turn, Algorithm 3 had several variants. Each variant has a different number of classification layers. Variant 1 uses two layers of classification: the first layer employs 5 classifiers, the second—logistic regression. Variant 2 adds one intermediate classification layer, Variant 3 expands Variant 2 with another classification layer. For Variant 2 (three layers) and Variant 3 (four layers) exactly the same result was obtained, i.e., SEN = 94.00%; both variants dealt with Classes 1 and 5 successfully, while Class 4 had the worst results.

Figures 16 and 17 show classification results for individual classes using single methods obtained in the previous and current studies. It can be noticed that all classifiers were the worst in the case of Class 4 classification, which could mean that the samples obtained

Table 10. Parameters used for the best classification result of Algorithm 6.

Layer	Estimator	Rescalling	Parameter 1	Parameter 2	Parameter 3	Parameter 4	O_ACC = SEN
L1.1	kNN	MaxAbsScaler	n_neighbors=3	weights='uniform'	algorithm='brute'	leaf_size=17	86.00%
	Extra Trees	Normalizer	n_estimators=99	-	-	-	
	Random Forest	MaxAbsScaler	n_estimators=158	max_depth=2	random_state=3	max_samples=44	
	Nu-SVC	MinMaxScaler	kernel='sigmoid'	nu=0.241815544	degree=2	gamma=2.124638721	
	Nearest Centroid	Normalizer	-	-	-	-	
L1.2	Linear SVC	Normalizer	random_state=29	tol=6.10873e-05	c=1.290368551	-	
	Nu-SVC	MinMaxScaler	kernel='linear'	nu=0.178280458	degree=4	gamma=2.048582813	
	kNN	MaxAbsScaler	n_neighbors=4	weights='uniform'	algorithm='brute'	leaf_size=15	
	Random Forest	MaxAbsScaler	n_estimators=92	max_depth=4	random_state=10	max_samples=43	
	Radius Neighbors	Normalizer	radius=6.872890495	algorithm='ball_tree'	leaf_size=30	weights='distance'	
L2.1	Nu-SVC	MinMaxScaler	kernel='linear'	nu=0.176140558	degree=5	gamma=3.438696991	
L2.2	kNN	MaxAbsScaler	n_neighbors=4	weights='uniform'	algorithm='kd_tree'	leaf_size=27	
L2.3	Nu-SVC	MinMaxScaler	kernel='linear'	nu=0.178148738	degree=1	gamma=1.161476259	
L2.4	Nearest Centroid	Normalizer	-	-	-	-	
L2.5	Radius Neighbors	Normalizer	radius=7.179069875	algorithm='ball_tree'	leaf_size=36	weights='distance'	
L2.6	Nu-SVC	MinMaxScaler	kernel='rbf'	nu=0.009248084	degree=7	gamma=0.051971845	
L2.7	Radius Neighbors	Normalizer	radius=5.109086106	algorithm='brute'	leaf_size=45	weights='distance'	
L2.8	Nu-SVC	MinMaxScaler	kernel='sigmoid'	nu=0.302767450	degree=4	gamma=2.735249768	
L2.9	Linear SVC	Normalizer	random_state=0	tol=4.40873e-05	c=1.802774393	-	
L2.10	Nu-SVC	MinMaxScaler	kernel='poly'	nu=0.375384705	degree=4	gamma=0.645935745	

Actual \ Predicted	1	2	3	4	5
1	**13**	2	0	0	0
2	1	**17**	2	0	0
3	0	0	**26**	0	0
4	0	0	3	**11**	6
5	0	0	0	0	**19**

	1			2			3			4			5		
13	2	86.67% 13.33%	**17**	3	85.00% 15.00%	**26**	0	100.00% 0.00%	**11**	9	55.00% 45.00%	**19**	0	100.00% 0.00%
1	**84**	98.82% 1.18%	2	**78**	97.50% 2.50%	5	**69**	93.24% 6.76%	0	**80**	100.00% 0.00%	6	**75**	92.59% 7.41%
92.86% 7.14%	97.67% 2.33%	**86.67%** 13.33%	89.47% 10.53%	96.30% 3.70%	**85.00%** 15.00%	83.87% 16.13%	100.00% 0.00%	**100.00%** 0.00%	100.00% 0.00%	89.89% 10.11%	**55.00%** 45.00%	76.00% 24.00%	100.00% 0.00%	**100.00%** 0.00%

Fig. 15. Confusion matrix: Algorithm 6, best result (SEN = 86%). SEN was calculated for individual classes.

in this class should be verified. It is possible that more test users would provide more reliable samples, and that could positively affect the classification result.

In turn, Fig. 18 presents a summary of all the results. The use of ensemble learning led to expected results, i.e., better outcomes in terms of the overall accuracy. Therefore, it is worth conducting further research in order to obtain better results. Current outcomes are good and look promising. The presented problem is a multi-class issue; classification methods for binary cases cannot be used. Confusion matrices are very useful when analyzing the results. They allow us to answer the question of how algorithms perform in the case of different classes.

5. Conclusions

In this paper we presented the following:

- the use of classic ML and ensemble learning methods in transmission quality parameter translation,
- a method of collecting samples used in the ML process and based on QoE,
- a comparison of classic ML methods and six new ensembles of classifiers.

The conducted research indicates the possibility of using ML in the case of transmission quality translations. The advantages of this approach are the flexibility of the solution, simpler evaluation by users and reliable translation. There are also disadvantages, one of which is the need to obtain reliable samples. Samples are significant because the whole ML process is based on supervised training. Another drawback is the number of samples—in the case of ML, 100 samples are not many; however, from the point of view of computer networks, generating 100 different combinations of parameters affecting the quality of transmission is difficult. It would be worth obtaining more samples which could improve the quality of the classification process.

Further research should incorporate more test users evaluating the samples. It is also advisable to collect more samples. Furthermore, other classification methods that may give better results should be considered. The possibility of using deep learning to solve the problem is also taken into account. Summing up, the results obtained so far are satisfactory and promising, but for the purpose of future studies it is worth testing more algorithms, which can improve the final result.

References

Acosta, J.R. and Avresky, D.R. (2005). Dynamic network reconfiguration in presence of multiple node and link failures using autonomous agents, *2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, San Jose, USA*.

Altman, N. (1992). An introduction to kernel and nearest neighbor non parametric regression, *The American Statistician* **46**(3): 175–185.

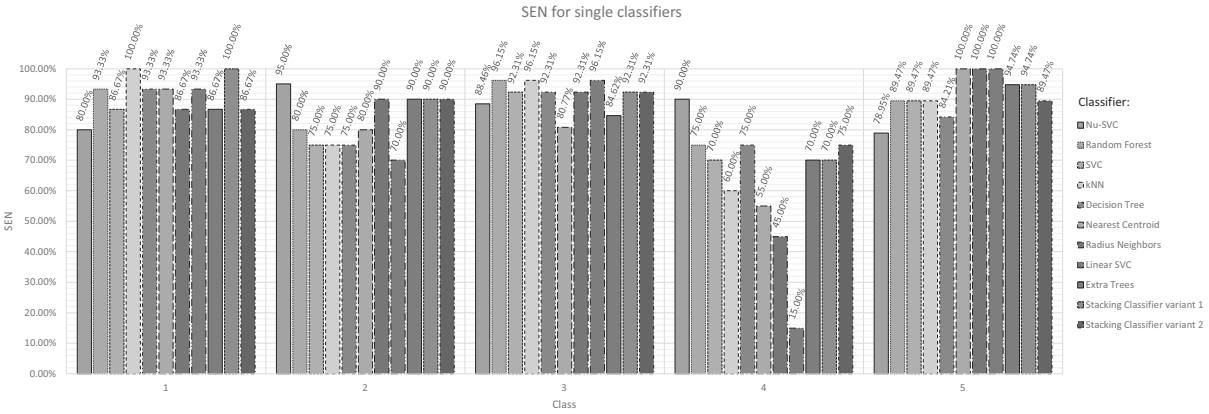

Fig. 16. SEN for single classifiers in different classes.

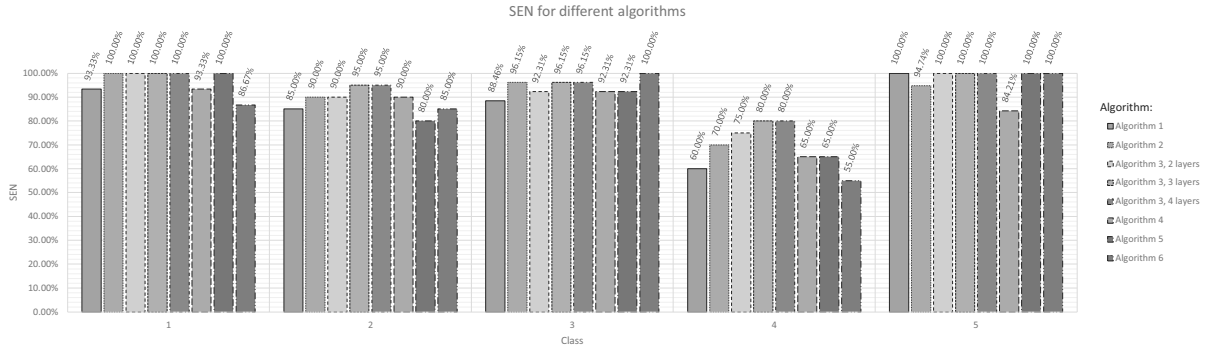

Fig. 17. SEN for different algorithms (ensembles of classifiers) in different classes.

Augustyniak, P. and Tadeusiewicz, R. (2006). Assessment of electrocardiogram visual interpretation strategy based on scanpath analysis, *Physiological Measurement* **27**(7): 597–608.

Breiman, L. (2001). Random forests, *Machine Learning* **45**: 5–32.

Chang, C.C. and Lin, C.J. (2011). LibSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology* **2**(3).

Chen, J., Wang, X., Cheng, Z. and Qin, J. (2017). On wireless sensor network mobile agent multi-objective optimization route planning algorithm, *2017 IEEE International Conference on Agents, Beijing, China*, pp. 101–103.

Chowdhury, C. and Neogy, S. (2012). Reliability of mobile agent system in QoS mobile network, *2012 4th International Conference on Communication Systems and Networks, Bangalore, India*, pp. 1–2.

Cortes, C. and Vapnik, V. (1995). Support-vector networks, *Machine Learning* **20**: 273–297.

C-SVC (2021). `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.

Dong, S., Zhou, D. and Ding, W. (2012). The study of network traffic identification based on machine learning algorithm, *2012 4th International Conference on Computational Intelligence and Communication Networks, Mathura, India*, pp. 205–208.

Elnaka, A.M. and Mahmoud, Q.H. (2013). Real-time traffic classification for unified communication networks, *2013 International Conference on Selected Topics in Mobile and Wireless Networking, Montreal, Canada.*

ExtraTreesClassifier (2021). `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html`.

Fan, R.E., Chang, K.W., Hsieh, C., Wang, X. and Lin, C.J. (2008). Liblinear: A library for large linear classification, *Journal of Machine Learning Research* **9**(9): 1871–1874.

Feng, C., Wu, S. and Liu, N. (2017). A user-centric machine learning framework for cyber security operations center, *2017 IEEE International Conference on Intelligence and Security Informatics, Beijing, China*, pp. 173–175.

F1 Score (2021). `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`.

Geurts, P., Ernst, D. and Wehenkel, L. (2006). Extremely randomized trees, *Machine Learning* **63**: 3–42.

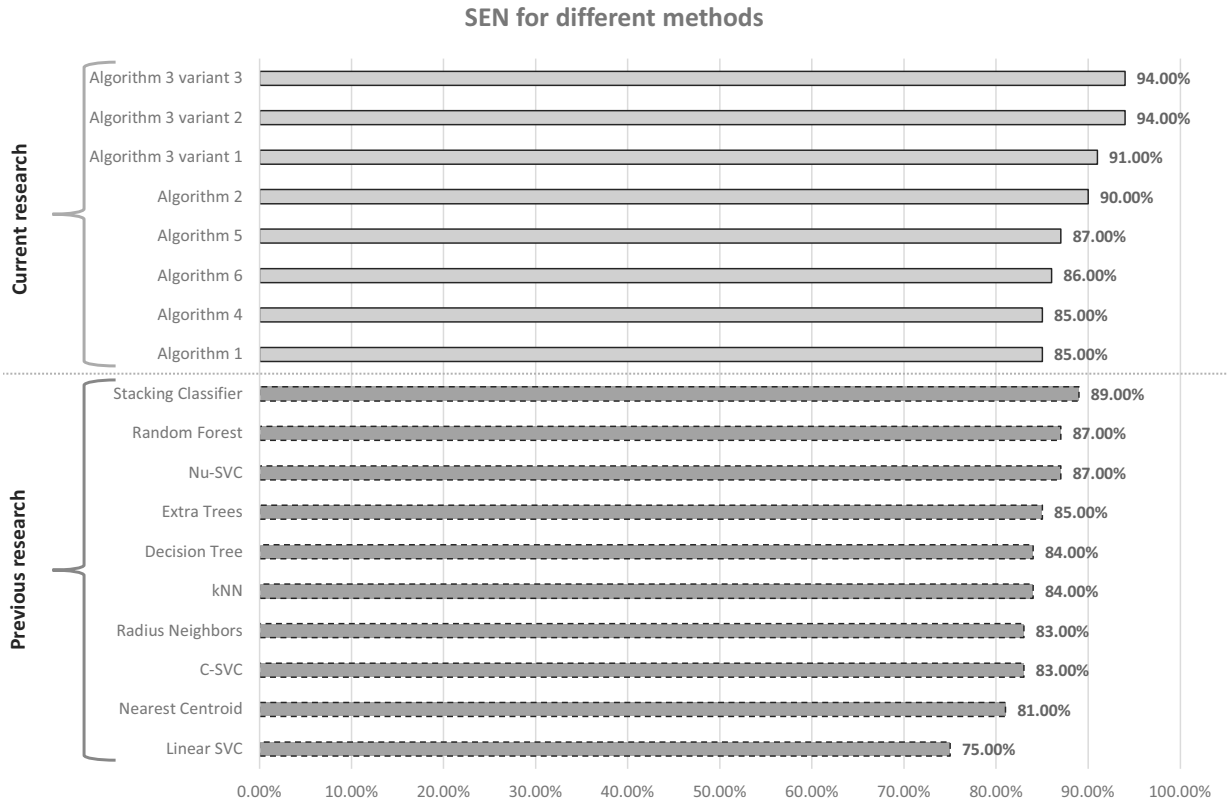

Fig. 18. SEN for different methods.

Grzonka, D., Kołodziej, J. and Jakóbik, A. (2019). Agent-based monitoring of the task scheduling in computational clouds, *Journal of Intelligent & Fuzzy Systems* **37**(9): 1–12.

Hsiang-Fu, Y., Fang-Lan, H. and Chih-Jen, L. (2011). Dual coordinate descent methods for logistic regression and maximum entropy models, *Machine Learning* **85**(1–2): 41–75.

Hu, F., Hao, Q. and Bao, K. (2014). A survey on software-defined network and openflow: From concept to implementation, *IEEE Communications Surveys Tutorials* **16**(4): 2181–2206.

Huang, C.-J., Liu, M.-C., Chu, S.-S. and Cheng, C.-L. (2004). Application of machine learning techniques to web-based intelligent learning diagnosis system, *4th International Conference on Hybrid Intelligent Systems, Kitakyushu, Japan*, pp. 242–247.

Hussain, S., Shakshuki, E. and Matin, A.W. (2006). Agent-based system architecture for wireless sensor networks, *20th International Conference on Advanced Information Networking and Applications, Vienna, Austria*, Vol. 2.

IRTF (2015). *Software-Defined Networking (SDN): Layers and Architecture Terminology*, `https://tools.ietf.org/html/rfc7426`.

ITU (2008). *Definitions of Terms Related to Quality of Service*, `https://www.itu.int/rec/T-REC-E.800-200809-I`.

Jaworek-Korjakowska, J. and Tadeusiewicz, R. (2013). Assessment of dots and globules in dermoscopic color images as one of the 7-point check list criteria, *International Conference on Image Processing, Melbourne, Australia*, Vol. 3, pp. 1456–1460.

Jaworek-Korjakowska, J. and Tadeusiewicz, R. (2014). Determination of border irregularity in dermoscopic color images of pigmented skin lesions, *36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Chicago, USA*, Vol. 2014, pp. 6459–62.

KNeighborsClassifier (2021). `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`.

Kołodziej, J., Szmajduch, M., Wang, L., Chen, D. and Khan, S. (2013). Genetic-based solutions for independent batch scheduling in data grids, *27th European Conference on Modelling and Simulation, Ålesund, Norway*, pp. 504–510.

Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S. (2015). Software-defined networking: A comprehensive survey, *Proceedings of the IEEE* **103**(1): 14–76.

Li, S., Zhang, Y., Wang, Y. and Sun, W. (2019). Utility optimization-based bandwidth allocation for elastic and inelastic services in peer-to-peer networks, *International Journal of Applied Mathematics and Computer Science* **29**(1): 111–123, DOI: 10.2478/amcs-2019-0009.

Li, Z., Yuan, R. and Guan, X. (2007). Accurate classification of the internet traffic based on the svm method, *IEEE International Conference on Communications, Glasgow, UK*.

LinearSVC (2021). https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

Liu, C.-C., Chang, Y., Tseng, C.-W., Yang, Y.-T., Lai, M.-S. and Chou, L.-D. (2018). SVM-based classification mechanism and its application in SDN networks, *10th International Conference on Communication Software and Networks, Chengdu, China*.

Memeti, S., Pllana, S., Binotto, A., Kołodziej, J. and Brandic, I. (2018). A review of machine learning and meta-heuristic methods for scheduling parallel computing systems, *LOPAL'18: Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications, Córdoba, Spain*, pp. 1–6.

Nguyen, T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning, *IEEE Communications Surveys & Tutorials* **10**(4): 56–76.

Nunes, B.A.A., Mendonca, M., Nguyen, X., Obraczka, K. and Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks, *IEEE Communications Surveys Tutorials* **16**(3): 1617–1634.

NuSVC (2021). https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html.

Ogiela, L., Tadeusiewicz, R. and Ogiela, M. (2006). Cognitive approach to visual data interpretation in medical information and recognition systems, *in* N. Zheng *et al.* (Eds), *Advances in machine Vision, Image Processing, and Pattern Analysis*, Springer, Berlin/Heidelberg, pp. 244–250.

Ogiela, M. and Tadeusiewicz, R. (2000). Syntactic pattern recognition for x-ray diagnosis of pancreatic cancer, *IEEE Engineering in Medicine and Biology Magazine* **19**(6): 94–105.

Onan, A., Korukoglu, S. and Bulut, H. (2016). A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification, *Expert Systems with Applications* **62**: 1–16.

Pahwa, K. and Agarwal, N. (2019). Stock market analysis using supervised machine learning, *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, Faridabad, India*, pp. 197–200.

Patri, S.K., Grigoreva, E., Kellerer, W. and Mas Machuca, C. (2019). Rational agent-based decision algorithm for strategic converged network migration planning, *IEEE/OSA Journal of Optical Communications and Networking* **11**(7): 371–382.

Pławiak, P., Abdar, M. and Acharya, U. (2019). Application of new deep genetic cascade ensemble of SVM classifiers to predict the Australian credit scoring, *Applied Soft Computing* **84**: 105740.

Pławiak, P., Abdar, M., Pławiak, J., Makarenkov, V. and Acharya, U. (2020). DGHNL: A new deep genetic hierarchical network of learners for prediction of credit scoring, *Information Sciences* **516**: 401–418.

Perez, J.A., Zarate, V.H. and Cebrera, C. (2006). A network and data link layer QoS model to improve traffic performance, *International Conference on Embedded and Ubiquitous Computing*.

RadiusNeighborsClassifier (2021). https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.RadiusNeighborsClassifier.html.

RandomForestClassifier (2021). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

Ross, M., Graves, C.A., Campbell, J.W. and Kim, J.H. (2013). Using support vector machines to classify student attentiveness for the development of personalized learning systems, *2013 12th International Conference on Machine Learning and Applications, Miami, USA*, Vol. 1, pp. 325–328.

Ruiz, D. and Finke, J. (2019). Lyapunov-based anomaly detection in preferential attachment networks, *International Journal of Applied Mathematics and Computer Science* **29**(2): 363–373, DOI: 10.2478/amcs-2019-0027.

Sakarkar, G. and Shelke, N.M. (2009). A new classification scheme for autonomous software agent, *2009 International Conference on Intelligent Agent Multi-Agent Systems, Chennai, India*, pp. 1–2.

Sakarkar, G. and Thakar, V.M. (2009). Autonomous software agent for localization, *2009 International Conference on Intelligent Agent Multi-Agent Systems, Chennai, India*, pp. 1–4.

Sammut, C. and Webb, I.G. (2010). *Encyclopedia of Machine Learning*, Springer, Boston.

Sasaki, Y. (2007). The truth of the F-measure, https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf.

Shuang, Z., Qinghe, H. and Dingwei, W. (2007). Application of software agent to e-commerce consumer buying support, *2007 2nd IEEE Conference on Industrial Electronics and Applications, Harbin, China*, pp. 2500–2503.

Slhoub, K., Carvalho, M. and Nembhard, F. (2019). Evaluation and comparison of agent-oriented methodologies: A software engineering viewpoint, *2019 IEEE International Systems Conference (SysCon), Orlando, USA*, pp. 1–8.

Stankiewicz, R. and Jajszczyk, A. (2011). A survey of QoE assurance in converged networks, *Computer Networks* **55**(7): 1459–1473.

Szaleniec, J., Wiatr, M., Szaleniec, M., Składzień, J., Tomik, J., Oleś, K. and Tadeusiewicz, R. (2013). Artificial neural network modelling of the results of tympanoplasty in chronic suppurative otitis media patients, *Computers in Biology and Medicine* **43**(1): 16–22.

Szaleniec, M., Tadeusiewicz, R. and Witko, M. (2008). How to select an optimal neural model of chemical reactivity?, *Neurocomputing* **72**(1–3): 241–256.

Tadeusiewicz, R. (2015). Neural networks as a tool for modeling of biological systems, *Bio-Algorithms and Med-Systems* **11**(3): 135–144.

Tan, Y. and Zhang, G.-J. (2005). The application of machine learning algorithm in underwriting process, *2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China*, Vol. 6, pp. 3523–3527.

Tello Leal, E., Chiotti, O. and David Villarreal, P. (2014). Software agents for management dynamic inter-organizational collaborations, *IEEE Latin America Transactions* **12**(2): 330–341.

Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression, *Proceedings of the National Academy of Sciences of the United States of America* **99**: 6567–6572.

Wolpert, D.H. (1992). Stacked generalization, *Neural Networks* **5**: 241–259.

Xia, W., Wen, Y., Foh, C.H., Niyato, D. and Xie, H. (2015). A survey on software-defined networking, *IEEE Communications Surveys Tutorials* **17**(1): 27–51.

Yong, Z., Hongrui, Z., Jing, C. and Binbin, Y. (2014). A weighted voting classifier based on differential evolution, *Artificial Intelligence and Data Mining* **2014**:1–6.

Yuan, R., Li, Z., X., G. and Xu, L. (2010). An SVM-based machine learning method for accurate internet traffic classification, *Information Systems Frontiers* **12**: 149–156.

Żelasko, D. (2020). Simulation of transmission quality classification in Pay&Require multi-agent managed network by means of machine learning techniques, *Simulation Modelling Practice and Theory* **103**: 102–106.

Żelasko, D., Cetnarowicz, K., Wajda, K. and Koźlak, J. (2016). Pay&Require as concept of variable cost routing in dynamically reconfigured networks, *Technical Transactions* **113**(16): 201–214.

Dariusz Żelasko was born in 1987 in Katowice, Poland. He received his BEng degree in information technology at the College of Information Technology in 2010 in Katowice. In 2012 he received his MSc degree in information technology at the Cracow University of Technology, Poland. His research interests include computer networks, QoS, security, IoT, mobile networks, machine learning, and ensemble learning.

Paweł Pławiak was born in 1984 in Ostrowiec, Poland. He received his BEng and MSc degrees in electronics and telecommunications and his PhD degree with honors in biocybernetics and biomedical engineering at the AGH University of Science and Technology in Kraków, Poland, in 2012 and 2016, respectively. In 2020, he received his DSc degree in computer science at the Silesian University of Technology in Gliwice, Poland. He is the head of the Department of Information and Communications Technology and an associate professor at the Cracow University of Technology and in the Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences in Gliwice. He has published more than 20 papers in refereed international SCI-IF journals. His research interests include machine learning and computational intelligence (e.g., artificial neural networks, genetic algorithms, fuzzy systems, support vector machines, k-nearest neighbors, and hybrid systems), ensemble learning, deep learning, evolutionary computation, classification, pattern recognition, signal processing and analysis, data analysis and data mining, sensor techniques, medicine, biocybernetics, and biomedical engineering.

Received: 26 June 2020
Revised: 9 September 2020
Re-revised: 18 October 2020
Accepted: 6 November 2020