LogForum

> Scientific Journal of Logistics <

2023, 19 (2), 263-275

http://doi.org/10.17270/J.LOG.2023.837

http://www.logforum.net p-ISSN 1895-2038 e-ISSN 1734-459X

ORIGINAL PAPER

THE APPLICATION OF ADABOOST.M1 BASED ON ANT COLONY OPTIMIZATION TO CLASSIFY THE RISK OF DELAY IN THE PHARMACEUTICAL SUPPLY CHAIN

Mateusz Wyrembek

Department of Logistics, Poznań University of Economics and Business, **Poland**

ABSTRACT. Background: The purpose of this article is to present the developed AdaBoost.M1 based on Ant Colony Optimization (hereby referred to as ACOBoost.M1 throughout the study) to classify the risk of delay in the pharmaceutical supply chain. This study investigates one research hypothesis, namely, that the ACOBoost.M1 can be used to predict the risk of delay in the supply chain and is characterized by a high prediction performance.
Methods: We developed a machine learning algorithm based on Ant Colony Optimization (ACO). The meta-heuristic algorithm ACO is used to find the best hyperparameters for AdaBoost.M1 to classify the risk of delay in the pharmaceutical supply chain. The study used a dataset from 4PL logistics service provider.
Results: The results indicate that ACOBoost.M1 may predict the risk of delay in the supply chain and is characterized by a high prediction performance.
Conclusions: The present findings highlight the significance of applying machine learning algorithms, such as the AdaBoost.M1 model with Ant Colony Optimization for hyperparameter tuning, to manage the risk of delays in the pharmaceutical supply chain. These findings not only showcase the potential for machine learning in enhancing supply chain efficiency and robustness but also set the stage for future research. Further exploration could include investigating other optimization techniques, machine learning models, and their applications across various industries and sectors.

Keywords: Ant Colony Optimization, AdaBoost.M1, Machine Learning, Supply Chain Risk management, Delay prediction

INTRODUCTION

In the modern era, the advancement of information and communication technology (ICT) has become critically important for enterprises. Technological progress has given rise to novel tools that facilitate more efficacious support for organizations in their operations. One of the most prominent trends in recent years is the escalating interest in artificial intelligence (AI) and its implementation in the business domain.

One of the principal fields of inquiry within the domain of artificial intelligence is that of machine learning [Hastiem Tibshirani, & Friedman, 2009; Mitchell, 1997; Russel & Norvig, 2021]. The most common task in machine learning is classification [Kozak, 2019], which assigns any description of an object's decision from a finite set of decisions. For example, there may be a need to classify certain deliveries by assigning them a decision of whether there is a risk of delay or not. A particular subtask of classification is called supervised learning. In this subtask, a finite set of objects (also called instances or cases) is provided, each with known decisions assigned to them. This set is called the training set; in the example considered here, it is the set of deliveries in the supply chain with assigned information about the risk of delay. The goal is to assign appropriate decisions (e.g., risk of delay or not) to new objects called test objects (e.g., new deliveries to be examined). One example of

Citation: *Wyrembek M. 2023. The application of AdaBoost.M1 based on Ant Colony Optimization to classify the risk of delay in the pharmaceutical supply chain LogForum 19 (2), 263-275,* http://doi.org/10.17270/J.LOG.2023.837
Received: 14.04.2023, Accepted: 01-06-2023, on-line: 02.07.2023

supervised learning is the AdaBoost.M1 algorithm[1].

The learning algorithm itself has parameters called hyperparameters. These parameters refer to the settings or configurations of the methods that can be freely selected within a certain range and have an impact on the performance or quality of the model [Bartz, Bartz-Beielstein, Zaefferer, & Mersmann, 2023]. Selecting hyperparameters manually is time-consuming, repetitive, and requires ad-hoc decisions by the practitioner [Hatıpoğlu, Tosun, & Tosun, 2022]. Metaheuristic methods, such as Ant Colony Optimization (ACO), are among the approaches used to search for optimal hyperparameters. Fig 1. presents an illustration of the problem of utilizing machine learning and ACO for the purpose of hyperparameter optimization.

One industry that can benefit from the potential of machine learning, for example, for supply chain risk management, which aims to protect supply chains from disruption (by predicting the presence of risk factors and mitigating their negative effects) [Wyrembek, 2023] is logistics. From the perspective of 4PL logistics services providers who are responsible for designing, planning, and optimizing the supply chain [Werner-Lewandowska Koliński, & Golinska-Dawson, 2023], it would be significant to predict delays with high prediction performance. It is even more important to predict delays in the pharmaceutical supply chain[2], which has an impact on patient care.

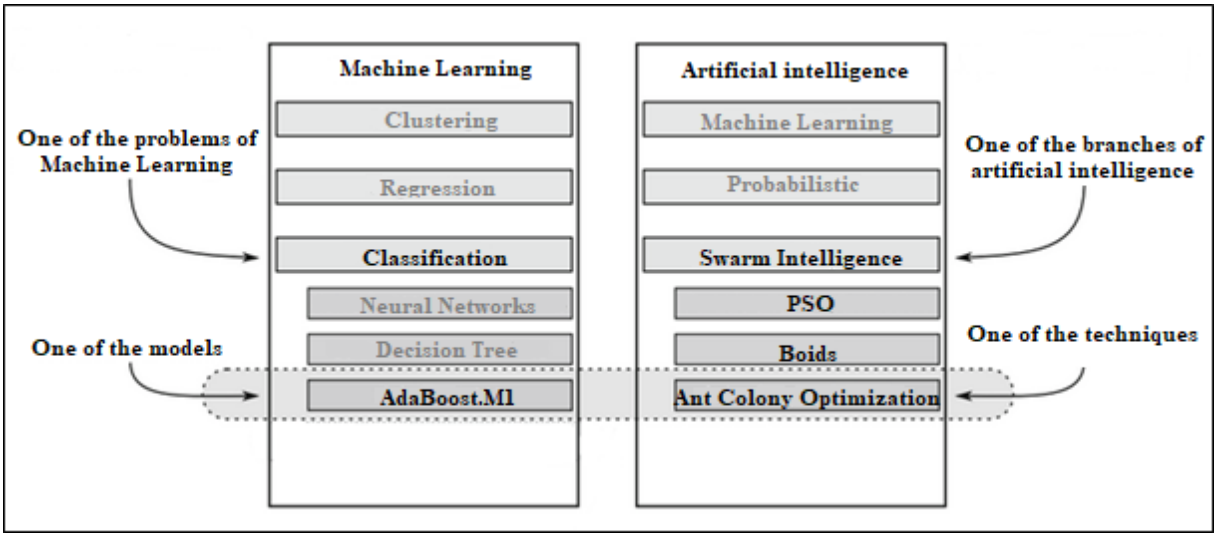

Fig. 1 Problem positioning *Source: Own work based on [Kozak, 2019]*

The aim of this study is to present the developed AdaBoost.M1 based on Ant Colony Optimization (hereby referred to as ACOBoost.M1 throughout the study) to classify the risk of delay in the pharmaceutical supply chain. Furthermore, the research hypothesis has been stated as follows:

[H1] ACOBoost.M1 makes it possible to predict the risk of delay in the supply chain and is characterized by a high prediction performance.

The research hypothesis is justified by works [Baryannis, Dani, & Antoniou, 2019; Trajkovski, & Madjarov, 2022], in which Baryannis et al. [2019] train classifiers to determine whether a delivery is late or not. Following Trajkovski & Madjarov's study [2022] it was demonstrated that optimizing hyperparameters of machine learning models using ACO resulted in high predictive performance.

[1] The terms "algorithm", "model", and "method" will be used interchangeably in this work

[2] The term „Pharmaceutical Supply Chain" is defined in this work as the supply chain of drug.

Literature Review on Machine Learning and predicting delays in the supply chains

Machine learning and supply chains delay started to be studied not long ago. This is the reason for the scarcity of publications on this topic, indicating an existing research gap that remains open. In the body of literature regarding late deliveries in the supply chain, studies by [Baryannis et al., 2019; Biazon de Oliveira et al., 2021; Steinberg, Burggräf, Wagner, & Heinbach, 2022; Wyrembek, 2022; Steinberg et al., 2023] have been found.

First, Baryannis et al. [2019] and Wyrembek [2022] face difficulties in handling data with high dimensionality, which is commonly referred to as the curse of dimensionality. If these machine learning projects were implemented in business operations, they would likely be deemed unsuccessful from a practical standpoint. Baryannis et al. [2019] compare three models: Support Vector Machine, Unrestricted Decision Trees, and Restricted Decision Trees. The best model was SVM with an accuracy of 94%. When it comes to Wyrembek [2022], Logistic Regression was used with AUC on the level 0,96. In these works [Biazon de Oliveira et al., 2021; Steinberg et al., 2022; Steinberg et al., 2023], regression was used instead of classification. Biazon de Oliveira et al. [2021] compare five models: Linear Regression, Linear Support Vector Machine, Random Forest, K-Nearest Neighbors, and Multi-Layer Perceptron. The best algorithm to forecast the lead time was SVM. They focused only on analyzing a selected link, rather than the entire pharmaceutical supply chain. The studies [Steinberg et al., 2022; Steinberg et al., 2023] are conducted only on one application case and may not be representative for other industries.

METHODOLOGY

In this section, the proposed approach is introduced.

Adaboost.M1 as an example of Boosting

Boosting is a machine learning technique that involves constructing multiple weak classifiers using pseudo-samples and employing them for classification through voting. The probability distribution of selecting elements from the learning set is not uniform but depends on the classification errors of the individual classifiers in the ensemble. Each element in the learning set is assigned a weight that determines its probability of being chosen for the pseudo-samples. After generating a pseudo-sample, a classifier is built and evaluated, and the weights of incorrectly classified elements are increased to raise their likelihood of being selected for subsequent pseudo-samples. This process is repeated for each classifier in the ensemble, leading to a balanced set of classifiers with greater accuracy than a single classifier [Kozak, 2019]. AdaBoost.M1 (the method was invented in 1995 by Freund and Schapire [Chengsheng, Huacheng, & Bing, 2017]) is a particular type of boosting that modifies the weights of objects in the learning set based on the classification errors of the individual classifiers already in the ensemble [Wyrembek, 2023]. Weight modification is dependent on the total weight of wrongly classified objects [Kozak, 2019]:

$$\varepsilon(j) = \sum_{x_i} \quad we_i\left[c_i \neq c_i^j\right] \qquad [1]$$

where we_i is the weight of object x_i, and c_i^j is the decision class of the analyzed object.

If the classification error is less than or equal to 0.5, the weights are modified accordingly. Otherwise, the weights are multiplied by a coefficient [2] and then normalized [Kozak, 2019].

$$\kappa(j) = \frac{1-\varepsilon(j)}{\varepsilon(j)} \qquad [2]$$

In the case of boosting, there is voting with weights, in which a single classifier gets the weight [Kozak, 2019][3]:

$$dDF(x) := \underset{y \in Y}{argmax} \sum_{j=1}^{J} \left(log\frac{1}{\kappa_j}\right) h_j(x,y).$$

The pseudocode of AdaBoost.M1 is presented in Fig. 2. Example hyperparameters of AdaBoost include the learning rate and n_estimators [Gao, & Liu, 2020].

Algorithm 1: The AdaBoost.M1 algorithm

```
1:  weight_of_objects[1..n] ← 1/n;
2:  ensemble ← NULL;
3:  weight_of_objects ← NULL;
4:  for number_of_classifiers do
5:     {Construction of the classifier with a weighted vote}
6:     data_set_classifier ← choose_objects(data_set, weight_of_objects);
7:     new_classifier ← build_classifier(data_set_classifier);
8:     new_classifier.determine_the_weight_of_voting();
9:     ensemble.add(new_classifier);
10:    {Calculate the new weight of objects}
11:    for number_of_objects do
12:       κ ← classifies_object(current_object, ensemble) {by Eq. [2]}
13:       weight_of_objects[current_object] ← weigh_of_objects[current_object] · κ
14:    end for
15: end for
16: result ← ensemble
```

Fig. 2. The pseudocode of AdaBoost.M1 algorithm *Source: Own work based on [Kozak, 2019]*

Ant Colony Optimization

The Ant Colony Optimization algorithm (ACO) was first presented in the early 1990s in relation to combinatorial optimization problems, particularly the traveling salesman problem and job shop scheduling [Colorni, Dorigo, & Maniezzo, 1991; Colorni, Dorigo, Maniezzo, & Trubian, 1994] and is a form of artificial intelligence called swarm intelligence [Boryczka & Kozak, 2010].

Their idea is based on observing the behavior of ants in the ecosystem [Boryczka & Kozak, 2016]. Research by Dorigo in his PhD Thesis has shown that ants have limited abilities when acting individually, but their efficiency greatly increases when they act as a group [Socha, & Blum, 2006]. In the experiment, the behavior of ants during their search for food and where they left pheromones on their path was observed, and subsequent ants followed the amount of pheromones on each path. Shorter paths were found to be more preferred as more ants could pass through them in a unit of time, leaving more pheromones on them. However, the entire process was not fully determined as ants could choose a different path despite the presence of pheromones on a given path [Jadczak, 2019]. Figure 3 presents the pseudocode of the ACO metaheuristics algorithm.

Algorithm 2: Ant Colony Optimization

```
1: procedure ANT_COLONY_OPTIMIZATION
2:     initialization()
3:     while (termination_conditions_not_met) do
4:         construct_ant_solutions()
5:         apply_local_search()                    ▷ Optional
6:         update_pheromones()
7:     end while
8: end procedure
```

Fig. 3. The pseudocode of ACO metaheuristics algorithm *Source: Own work based on [Kozak, 2019]*

One iteration of the loop involves all ants constructing solutions, optionally improving them using a local search algorithm, and updating the pheromone levels. The subsequent sections provide a more detailed explanation of the algorithm [Kozak, 2019]:

- **initialization()** - At the beginning of the algorithm, the approach parameters are set and all pheromone variables are initialized to a value τ0, which is a parameter of this algorithm [Kozak, 2019; Jadczak, 2019].
- **construct_ant_solution()** - The choice of a solution is made probabilistically in each step of construction. The node transition rule is modified to explicitly allow for exploration. An ant *k* positioned at an analyzed node *i* chooses node *j* to move to and thus follows the rule [Boryczka & Kozak, 2010; Kozak, 2019; Jadczak 2019][4]:

$$j = \begin{cases} argmax_{u \in J_i^k} \{[\tau_{iu}(t)] \cdot [\eta_{iu}]^{\beta}\} & ifq \leq q_0 \\ J & ifq > q_0 \end{cases},$$

where the pheromone amount τiu(t) currently present on the edge (*i, u*) at time *t* is multiplied by the heuristic value ηiu between nodes *i* and *u*. A random variable *q*, which is uniformly distributed between 0 and 1, is used to determine the probability of a certain node being selected as the next node to be traversed. The parameter q_0, which can be adjusted, is used to control the balance between exploration and exploitation. Finally, $J \in J^k_i$ represents the node being analyzed, and it is chosen randomly with a probability determined by:

$$p_{iJ}^k(t) = \frac{\tau_{iJ}(t) \cdot [\eta_{iJ}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}(t)] \cdot [\eta_{il}]^{\beta}} \qquad [5]$$

- **apply_local_search() - optional** - After the construction of solutions, further steps may be needed before updating the pheromones, which are often referred to as daemon actions [Kozak & Boryczka, 2016]. One common daemon action is to apply local search to the solutions, using the locally optimized solutions to determine which pheromone values to update in the matrix [Socha, & Blum, 2006].
- **update_pheromones()** - The goal of updating pheromones is to enhance pheromone values linked with good or potentially good solutions while reducing ("punishing") those related to bad solutions. Typically, this is accomplished by decreasing all pheromone values through pheromone evaporation and increasing the pheromone levels of selected good solutions [Kozak, 2019]. Pheromone evaporation is needed to avoid a too rapid convergence of the algorithm [Boryczka & Kozak, 2016].

ACOBoost.M1

The ACO method, in conjunction with the AdaBoost.M1 model, serves to optimize the hyperparameters of the classification model. In the context of hyperparameter optimization, ants traverse the hyperparameter space in order to find the best set of parameters for the boosting model.

In the algorithm, each ant represents a potential set of hyperparameters. The ants explore the hyperparameter space, taking into account the attractiveness of individual values. Attractiveness is calculated based on pheromones and cross-validation error. Pheromones are updated in each iteration, allowing for dynamic exploration of the hyperparameter space.

The ACO algorithm is executed for a specified number of iterations, and in each iteration, ants traverse the hyperparameter space, evaluate the cross-validation error, and update the pheromones. Based on the pheromones and attractiveness, ants select the subsequent hyperparameter values, leading to the ultimate optimal set of hyperparameters.

In this particular task, the AdaBoost.M1 model is optimized with regard to two hyperparameters: the number of estimators ('n_estimators') and the learning rate ('learning_rate'). The ACO algorithm searches the hyperparameter space to find the best values for these parameters, which result in the lowest cross-validation error.

Upon completion of the ACO algorithm and identification of the best hyperparameters, the AdaBoost.M1 model is trained on the training data set using these optimal parameters. As a result, we obtain a model with higher accuracy and improved generalization ability on novel data. Fig. 4 presents the pseudocode of the ACOBoost.M1 algorithm.

Algorithm 3: ACOBoost.M1 Algorithm

```
initialize best_hyperparameters;
set_ACO_parameters;
set best_error := infinity
set best_parameters := null
for i = 1 to max_iterations do
    for j = 1 to num_hyperparameters do
        for a = 1 to num_ants do
            generate_parameters_list
        end for
    end for
    for j = 1 to num_hyperparameters do
        for a = 1 to num_ants do
            calculate_error
            if current_error < best_error then
                best_error := current_error
                best_parameters := current_parameters
            end if
        end for
        calculate_attractiveness
    end for
    for j = 1 to num_hyperparameters do
        for a = 1 to num_ants do
            update_pheromone_trail
        end for
    end for
end for
return best_parameters
train_AdaBoost_model(ACO_best_parameters, X_train, y_train)
predict_and_evaluate(X_test, y_test)
if new_classifier_performance > best_classifier_performance then
    best_classifier := new_classifier
end if
report_performance_metrics
```

Fig. 4. The pseudocode of ACOBoost.M1 algorithm

CASE STUDY

For this case study, we chose a 4PL logistics services provider that designed and manages a pharmaceutical supply chain. To develop a model, we applied the established CRISP-DM procedure model (presented in Fig. 5) [Sarkar, Bali, & Sharma, 2018]. As a result, this section will proceed according to the phases of CRISP-DM, including business understanding, data understanding, data preparation, modeling, evaluation, and deployment. As we primarily focused on model development, we excluded the last phase, namely, Deployment. The case study was carried out on a computer with an 8-core CPU, 8-core GPU, and 8 GB RAM.

Business Understanding

The Business understanding phase typically includes a description of the business problem and a transfer of the business problem into concrete requirements and objectives for further data analysis [Steinberg et al., 2023].

As the business problem is delays in pharmaceutical supply chains, our goal is to classify whether a delivery is delayed or not using ACOBoost.M1. We assume that the risk of delay is real and has been classified into binary categories, where 0 indicates on-time delivery and 1 indicates delayed delivery.

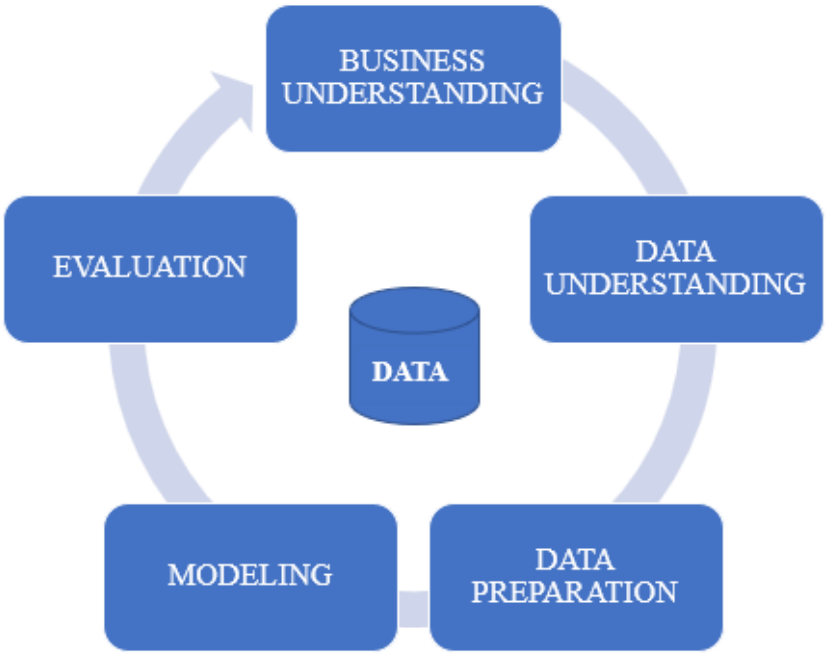

Fig. 5 CRISP-DM model *Source: Own work based on [Sarkar, Bali, & Sharma, 2018]*

Data Understanding

The dataset contains 6711 instances, where a single instance represents a delayed delivery in the supply chain. Each instance is described by eleven input attributes described in Table 1. The target variable is **Risk**.

The data was collected in the first quarter of 2023. As shown in Figure 6, we can observe that 17.1% of deliveries were delayed and 82.9% were on time. In the dataset, there are 1847 missing values for the **Month** and **Kilometers** categories, respectively. Figure 7 presents a missingness map, which shows where the missing values occurred exactly.

Table 1. Attributes

Attribute	Description	Type
Supplier	ID of the Supplier	Object
Month	Month of delivery	Float64
Kilometers	Kilometers to place of delivery	Float64
ID_ODB_T	Binary recipient category - 0 represents pharmacy, 1 represents hospital pharmacy.	int64
Weight	Weight of the parcel	Float64
Freight	Cost of delivery	Float64
Parcel	Number of packages in the delivery	int64
Type of delivery	Type of delivery. Classic - delivery within 72 hours, premium – delivery within 24 hours	Object
Status	Status of the delivery	Object
Country	Country of the delivery	Object
Carrier	Carrier that delivered the parcel	Object
Risk	Binary category – 0 – represents on time delivery, 1, represents delayed delivery	int64

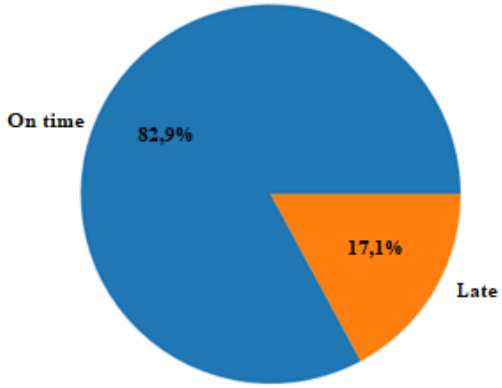

Fig. 6. On-time and delayed deliveries as a percentage.

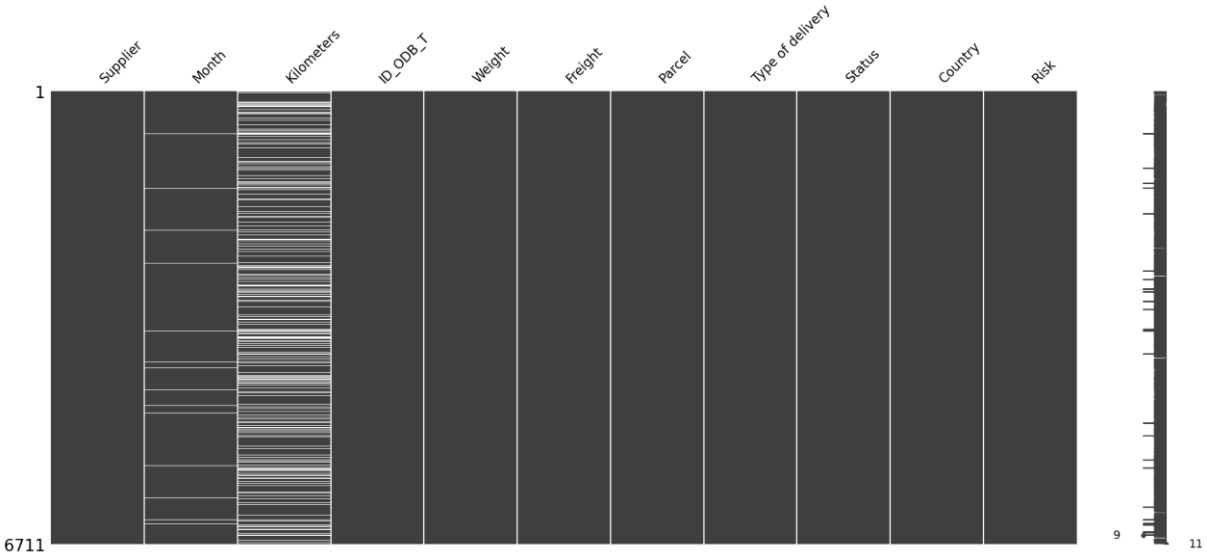

Fig. 7. Missingness map

Data Preparation

Based on the knowledge and insight acquired from the dataset, a standard pre-processing phase was applied, including cleaning and removal of duplicates and corrupted data, outlier detection, manipulation of missing values, correlation analysis, Cramér's V, and Pearson correlation coefficient [Ristoski, & Paulheim, 2016; Kuhn, & Johnson, 2016].

As we did not find any duplicates in our dataset, we handled missing values. As was shown in the previous section, there are 1847 missing values for the **Month** and **Kilometers** categories. Following Emmanuel et al. [2021], we use a simple imputation method such as a mean to fill the missing values. After removing outliers and using correlation analysis, Cramér's V, and Pearson correlation coefficient, we chose the following variables: **Month**, **Kilometers**, **ID_ODB_T**, **Weight, Parcel**, **Type of delivery**, **Country**, **Carrier**, and **Risk**. The next step was to make a dummy variable from the **Carrier**, **Type of delivery,** and **Country** categories. The final step of the pre-proccesing was data normalization [Dong, & Liu, 2018].

Modeling

After understanding the data and defining the features of our ACOBoost.M1 model, we train it. To train a model we use the Python programming language utilizing the scikit-learn, numpy, pandas, and random libraries.

Firstly, we set up Ant Colony Optimization parameters. The ACO algorithm parameters used in our study include the number of ants set to 100, the number of iterations set to 50, and the evaporation coefficient set at 0.7. The algorithm searched for the best hyperparameters for 23 hours, 25 minutes, and 10 seconds. Upon completion of the optimization process, the best hyperparameters for the ACOBoost.M1 model were identified as ('n_estimators': 54, 'learning_rate': 1.0).

Figure 8 represents a pheromone map created when ACO finished running. The pheromone map displays the quantity of pheromones for each combination of hyperparameters (in this case, 'n_estimators' and 'learning_rate') and ants. Each column represents an ant, while each row corresponds to a hyperparameter. The values in the pheromone map indicate the attractiveness of a given hyperparameter for the ants. The greater the pheromone value, the higher the likelihood that an ant will select that particular hyperparameter. The ACO algorithm relies on an iterative process in which ants update the pheromone map based on the quality of the solutions they have found. As a result, higher pheromone values indicate better solutions. [Kozak, & Boryczka, 2016; Kozak, 2019]. The map provides an insight into the collective exploration and search behavior of the ants as they traverse the hyperparameter space in pursuit of optimal solutions for the ACOBoost.M1 model.

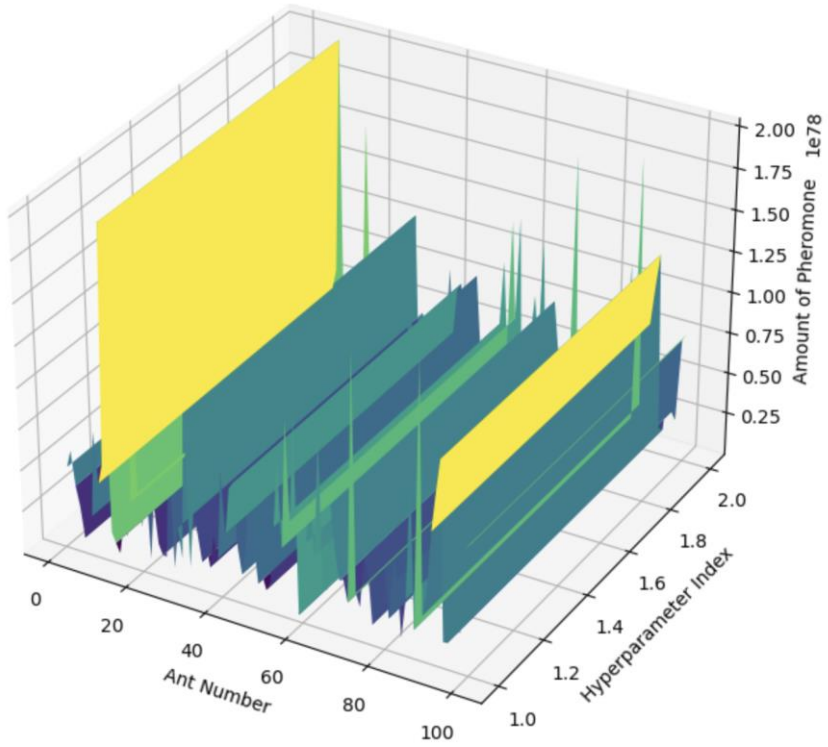

Fig. 8. Pheromone map created when ACO finished running

In the pheromone map shown in Figure 8, it can be observed that the pheromone values differ between ants but are generally quite high. These values signify the appeal of hyperparameter combinations for the ants. We can deduce from this that the ACO algorithm has identified certain hyperparameter combinations as more attractive than others. Attention should be paid to the hyperparameter combinations with the highest pheromone values. These values suggest that a given hyperparameter combination yielded better results during the optimization process.

Evalutation

In the context of binary classification problems, such as the one under consideration, instances can be labeled as either positive or

negative. Taking this dichotomy into account, the classification outcomes can be organized into four distinct categories, which create the confusion matrix: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Both TP and TN represent instances where the classification has been performed correctly. FP is a false alarm, also called a Type I error, and FN represents miss-classified ones, also called a Type II error [Hatıpoğlu, 2022; Wyrembek, 2023].

Derived from the confusion matrix, several performance measures can be computed for the model, encompassing [Iwendi et al, 2022; Wyrembek, 2023]:

- Accuracy - the proportion of correctly classified instances among the total number of instances, which includes true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).
- Precision - a metric reflecting the model's effectiveness, signifying the ratio of true positives (TP) to the combined count of true positives (TP) and false positives (FP).
- Recall - the ratio of true positives (TP) to the sum of true positives (TP) and false negatives (FN), indicating the model's ability to identify positive instances correctly.
- F1 Score - this measure represents the harmonic mean of precision and recall, offering a balanced perspective on the model's performance.

Table 2 contains a summary of performance metrics and classification. The confusion matrix reveals that the model successfully identified 1129 instances with a risk of delay (TP) and 30 instances without a risk of delay (TN). However, it also misclassified 181 instances with a risk of delay (FP) and 3 instances without a risk of delay (FN).

Table 2 Summary of Performance measures and classification

Performance measures				**Classification**			
Accuracy	Precision	Recall	F1 Score	TP	FN	FP	TN
0.862	0.861	0.997	0.924	1129	3	181	30

The model's accuracy of approximately 86.3% demonstrates its ability to correctly classify instances in the majority of cases. With an F1 score of roughly 0.924, the model demonstrates a robust equilibrium between recall and precision in detecting instances involving a risk of delay. The high recall of approximately 99.7% indicates that the model is adept at identifying instances with a risk of delay, as it manages to capture nearly all of the actual positive instances. A precision of approximately 86.2% signifies that 86.2% of the instances predicted as having a risk of delay by the model are indeed at risk of delay.

DISCUSSION

Based on the results obtained, it can be concluded that the ACOBoost.M1 model effectively predicts the risk of delay in the pharmaceutical supply chain, thereby supporting Hypothesis H1. The model's high predictive performance, with an accuracy of about 86.3%, substantially exceeds the levels of accuracy forecasted in the study by Baryannis et al. [2019]. This suggests that ACOBoost.M1 might outperform traditional classification models such as SVM in predicting delay risk.

Our findings confirm and extend those of Trajkovski and Madjarov [2022], who demonstrated that hyperparameter optimization of machine learning models using ACO resulted in high predictive performance. The model's F1 score of approximately 0.924 indicates a strong balance between recall and precision. These results, coupled with a high recall of nearly 99.7% and a precision of about 86.2%, demonstrate that the ACOBoost.M1 model is effective at identifying and predicting instances with a risk of delay.

However, our study, like others [Baryannis et al., 2019; Biazon de Oliveira et al., 2021; Steinberg et al., 2023; Wyrembek, 2022], has certain limitations. Our results are based on a specific dataset, and the model's performance may vary when applied to different datasets or contexts.

Future research should focus on applying the ACOBoost.M1 model to different datasets and contexts to assess its versatility and potential for improvement. In line with the results of previous studies, our results provide strong evidence that the ACOBoost.M1 model is a powerful tool for predicting the risk of delay in the pharmaceutical supply chain.

CONCLUSION

In today's world, the importance of applying machine learning is continually growing and can be utilized across numerous industries, potentially in all of them. For instance, the medical industry can leverage these algorithms to manage risk in the supply chain to protect against disruptions (by predicting the occurrence of risk factors).

In this article, we presented a different approach to the classification problem in machine learning. The Ant Colony Optimization algorithm was employed for hyperparameter tuning to construct an AdaBoost.M1 model, which classifies the risk of delay in the pharmaceutical supply chain.

The ACOBoost.M1 model has demonstrated its effectiveness in managing the risk of delay in the pharmaceutical supply chain. Its high prediction performance, as evidenced by its accuracy, F1 score, recall, and precision, indicates that the model can reliably identify instances with a risk of delay. This ability is crucial for decision-makers in the pharmaceutical industry to implement timely mitigation strategies and maintain supply chain efficiency.

In light of these findings, the ACOBoost.M1 model can serve as a valuable tool for organizations operating in the pharmaceutical supply chain. By incorporating this model into their risk management strategies, companies can proactively address potential delays, optimize their logistics processes, and ultimately improve overall supply chain performance. Future research and model refinement, including further exploration of hyperparameters, will help enhance the model's predictive capabilities and its applicability to various pharmaceutical supply chain scenarios.

ACKNOWLEDGEMENTS

I would like to thank Prof. J. Kozak for inspiring me to write this article. Thanks to his advice, I was able to develop ACOBoost.M1. Additionally, I would like to express my gratitude to my mentor and supervisor, Prof. M. Szymczak, who continuously supports me with valuable advice and constantly shows me how to be a good researcher.

REFERENCES

Bartz, E., Bartz-Beielstein, T., Zaefferer, M., Mersmann, O. (Eds.). (2023). Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide. Springer. https://doi.org/10.1007/978-981-19-5170-1

Baryannis, G., Dani, S., & Antoniou, G. (2019). Predicting supply chain risks. Using machine learning: The trade-off between performance and interpretability. Future Generation Computer Systems, 101, 993–1004. https://doi.org/10.1016/j.future.2019.07.059

Biazon de Oliveira, M., Zucchi, G., Lippi, M., Farias, C., Rosa da Silva, N., & Iori, M. (2021). Lead time forecasting with machine learning techniques for a pharmaceutical supply chain. Paper presentation. Proceedings of the 23nd International Conference on Enterprise Information Systems on International Conference on Enterprise Information Systems (ICEIS). 26–28.04.2021. https://www.scitepress.org/Papers/2021/104344/104344.pdf

Boryczka, U., & Kozak, J. (2010). Ant colony decision trees—a new method for constructing decision trees based on ant colony optimization. In J.S. Pan, S.M. Chen, & N.T Nguyen (Eds.), Computational Collective Intelligence. Technologies and Applications (pp. 373 - 382). Springer. https://doi.org/10.1007/978-3-642-16693-8_39

Boryczka, U., & Kozak, J. (2016). Adaptive Ant Clustering Algorithm with Pheromone. In N.T. Nguyen, B. Trawiński, H. Fujita, T.P. Hong (Eds.), Intelligent Information and database Systems (pp. 117-126). Springer. https://doi.org/10.1007/978-3-662-49390-8_11

Chengsheng, T., Huacheng, L., & Bing, X. (2017). AdaBoost typical Algorithm and its application research. Paper presentation. Proceedings of the 3rdInternational Conference on Mechanical, Electronic and Information Technology Engineering (ICMITE 2017). 16-17.12.2017. https://doi.org/10.1051/matecconf/201713900222

Colorni, A., Dorigo, M., Maniezzo, V. (1991). Distributed optimization by ant colonies. In F.J. Varela & P. Bourgine (Eds.), Proceedings of European Conference Artificial Life (ECAL '91) (pp. 134 - 142). Elsevier Publishing.

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M. (1994), Ant system for job-shop scheduling, ORBEL Belgian Journal of Operations Research. Statistics and Computer Science, 34, 39–53.

Dong, G., & Liu, H. (Eds.). (2018). Feature engineering for machine learning and data analytics. CRC Press. https://doi.org/10.1201/9781315181080

Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Banyatsang, M., & Tabona, O. (2021). A survey on missing data in machine learning. Journal of Big data, 8(140). https://doi.org/10.1186/s40537-021-00516-9

Gao, R., & Liu, Z. (2020). An Improved AdaBoost Algorithm for Hyperparameter Optimization. Journal of Physics: Conference Series, 1631(2020). https://doi.org/10.1088/1742-6596/1631/1/012048

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.

Hatıpoğlu, I., Tosun, Ö., Tosun, N. (2022). Flight delay prediction based with machine learning. LogForum, 18(1), 97-107. https://doi.org/10.17270/J.LOG.2022.655

Iwendi, C., Mahboob, K., Khalid, Z., Javed, A. R., Rizwan, M., & Ghosh, U. (2022). Classification of COVID-19 individuals using adaptive neuro-fuzzy inference system. Multimedia systems, 28(4), 1223–1237. https://doi.org/10.1007/s00-530-021-00774-w

Jadczak, R. (2019). Vehicle routing in supply chain management. Models, methods, and applications. Wydawnictwo Uniwersytetu Łódzkiego.

Kozak, J., & Boryczka, U. (2016). Collective Data Mining in The Ant Colony Decision Tree Approach. Information Sciences, 372, 126-147. https://doi.org/10.1016/j.ins.2016.08.051

Kozak, J. (2019). Decision Tree and Ensemble Learning Based on Ant Colony Optimization. Springer. https://doi.org/10.1007/978-3-319-93752-6

Kuhn, M., Johnson, K. (2016). Applied predictive modeling (5th ed.). Springer.

Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

Sarkar, D., Bali, R., & Sharma, T. (2018). Practical Machine Learning with Python. A Problem-Solver's Guide to Building Real-World Intelligent Systems. Apress. https://doi.org/10.1007/978-1-4842-3207-1

Socha, K., Blum, C. (2006). Ant Colony Optimization. In E. Alba & R. Martí (Eds.), Metaheuristic Procedures for Training Neural Networks (pp. 153 - 180). Springer. https://doi.org/10.1007/0-387-33416-5_8

Steinberg, F., Burggräf, P., Wagner, J., & Heinbach, B. (2022). Impact of material data in assembly delay prediction—a machine learning-based case study in machinery industry. The International Journal of Advanced Manufacturing Technology, 120, 1333-1346. https://doi.org/10.1007/s00170-022-08767-3

Steinberg, F., Burggräf, P., Wagner, J., Heinbach, B., Saßmannshausen, T., Brintrup, A. (2023). A novel machine learning model for predicting late supplier deliveries of low-volume-high-variety products with application in a German machinery industry. Supply Chain Analytics, 1. https://doi.org/10.1016/j.sca.2023.100003

Ristoski, P. & Paulheim, H. (2016). Semantic web in data mining and knowledge discovery: A comprehensive survey. Journal of Web Semantics, 36, 1–22. https://doi.org/10.1016/j.websem.2016.01.001

Russell, S. J., & Norvig, P. (2021). Artificial intelligence: A modern approach (4th ed.). Pearson Education.

Trajkovski, A., & Madjarow, G. (2022). Model Hyper Parameter Tuning using Ant Colony Optimization. Paper presentation. Proceedings on the 19th Intrnational Conference on Informatics and Information Technologies – CIIT 2022. https://repository.ukim.mk/bitstream/20.500.12188/25676/1/CIIT_2022_7.pdf

Werner-Lewandowska, K., Koliński, A., & Golinska-Dawson, P. (2023). Barriers to electronic data exchange in the supply chain – result from empirical study. LogForum, 19(1), http://doi.org/10.17270/J.LOG.2023.804

Wyrembek, M. (2022). The use of Big Data technology to predict risk of delay in supply chain. Material Economy and Logistics, 6, 29-36. https://doi.org/10.33226/1231-2037.2022.6.4

Wyrembek, M. (2023). The use of machine learnings methods to predict the risk of damage to goods. Material Economy and Logistics, 1, 58-66, https://doi.org/10.33226/1231-2037.2023.1.7

Mateusz Wyrembek ORCID ID: https://orcid.org/0000-0002-7946-948X
Department of Logistics,
Institute of International Business and Economics,
Poznań University of Economics and Business, Poznań, **Poland**
e-mail: mateusz.wyrembek@phd.ue.poznan.pl