JOURNAL OF MODERN SCIENCE
SPECIAL ISSUE
3/57/2024
www.jomswsge.com

DOI: doi.org/10.13166/jms/191218

DARIUSZ WOŹNIAK
Graduate School of Business – National-Louis University, Poland
ORCID iD: orcid.org/0000-0002-3580-0708

MICHAŁ WARSZYCKI
ProductSupply.AI, Poland
ORCID iD: orcid.org/0000-0002-3004-1909

JÓZEF STOKŁOSA
WSEI University in Lublin, Poland
ORCID iD: orcid.org/0000-0003-1266-6945

RAFAŁ SZRAJNERT
WSEI University in Lublin, Poland
ORCID iD: orcid.org/0000-0001-9045-1769

ROBERT CHMURA
WSEI University in Lublin, Poland

PREDICTIVE ALGORITHMS FOR SUPPLY CHAIN MANAGEMENT: A COMPREHENSIVE APPROACH TO FORECASTING DELIVERY TIMES AND MANAGING RISK

ALGORYTMY PREDYKCYJNE W ZARZĄDZANIU ŁAŃCUCHEM DOSTAW: KOMPLEKSOWE PODEJŚCIE DO PROGNOZOWANIA TERMINÓW DOSTAW I ZARZĄDZANIA RYZYKIEM

Abstract

The article delves into the challenges of delivery time management in today's business landscape. The authors underscore the need for precise delivery time forecasts, a key factor in maintaining a competitive edge and meeting customer expectations. They outline various methods for estimating the time of a selected commodity based on historical data, and stress the necessity of modern tools that can adapt to the intricate web of factors influencing delivery times and facilitate swift responses to changes.

The following article presents an innovative delivery time forecasting application that integrates advanced predictive algorithms with historical data, current data, and external factors affecting the delivery process. The application was developed to provide more accurate delivery time forecasts and optimize logistics processes. Through advanced technologies, it can consider even the most complex scenarios and changes, allowing companies to plan and manage their logistics operations more effectively.

Streszczenie

Artykuł omawia znaczenia efektywnego zarządzania czasem dostaw w dzisiejszym świecie biznesu. Autorzy podkreślają, że dokładna prognoza czasu dostaw jest kluczowa dla zachowania konkurencyjności i zadowolenia klientów. Zostały przedstawione różne metody do szacowania czasu wybranego towaru na podstawie danych historycznych, a także wskazuje na potrzebę rozwoju nowoczesnych narzędzi, które uwzględniałyby złożoność czynników wpływających na czas dostaw oraz umożliwiałyby szybką reakcję na zmiany.

W dalszej części artykułu przedstawiono nowatorską aplikację do prognozowania czasu dostaw, która integruje zaawansowane algorytmy predykcyjne z danymi historycznymi, aktualnymi i zewnętrznymi czynnikami wpływającymi na proces dostawy. Aplikacja ta została opracowana w celu zapewnienia dokładniejszych prognoz czasu dostaw oraz optymalizacji procesów logistycznych. Dzięki wykorzystaniu zaawansowanych technologii, aplikacja jest w stanie uwzględnić nawet najbardziej złożone scenariusze i zmiany, co pozwala firmom skuteczniej planować i zarządzać swoimi operacjami logistycznymi.

Keywords: *classification models, regression models, supply chain, forecasting delivery times, XGBoost, Random Forest, LightGBM*

Słowa kluczowe: *model klasyfikacyjny, model regresyjny, łańcuch dostaw, prognozowanie czasów dostarczania towarów, XGBoost, Las losowy, LightGBM*

Introduction

Forecasting delivery times is a critical element of effective supply chain management. With analyses based on machine learning models and algorithms, delivery delays can be predicted, which has many advantages and benefits:

- Inventory optimization – allows for better inventory management. By predicting delivery delays, shortages of goods or overstocking can be avoided.
- Increased operational efficiency—Accurate forecasting can optimize production schedules and processes, leading to better efficiency throughout the supply chain.
- Increase customer satisfaction – predicting delays allows you to better manage customer expectations by enabling you to inform them of actual delivery times. This translates into building trust and brand loyalty.
- Cost reduction—Minimizing delays avoids costs associated with emergency deliveries, additional storage charges, or lost customers due to dissatisfied service.
- Improving logistics planning: Forecasting delivery delays allows for better route and lead time planning, resulting in more efficient use of logistics resources.
- Use of historical data—Machine learning algorithms can analyze large amounts of historical data, considering various factors affecting delays. This allows for more accurate predictions than traditional methods.
- Adaptability and model improvement – machine learning-based systems can continuously improve, learning from new data and improving their forecasts as information increases.

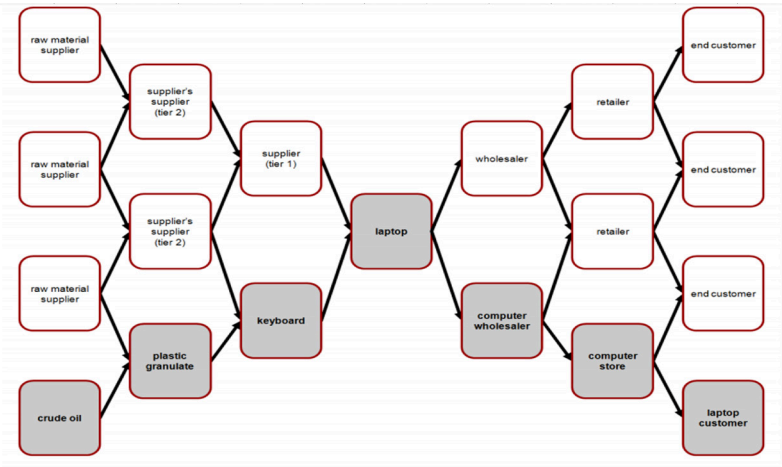
Predicting delivery delays using machine learning models and algorithms is a critical element of supply chain optimization that can significantly improve supply chain efficiency, reduce costs, and increase customer satisfaction. The module presented here uses a two-step approach. First, we build a classification model based on a user-defined number of days of order processing, after which we consider a shipment to be delayed. This is applied because we may not have information in the available data about whether the fulfillment of a particular order

has been delayed. Based on the historical data provided, the model can determine the probability of delay for new shipments. The next step is to apply a regression model to help us approximate the expected lead time for an order. A more detailed description of the methods used for these tasks will be presented in the following section. As the implemented module fits into the adopted model of creating independent services (microservices) in Docker containers, this will also be retained here. The created service will use FastAPI to communicate with the client application. When a request is sent, analyses selected by the user will be performed (learning models, prediction for new orders, grouping of goods), and the results will be sent back and displayed on the created client application.

According to the user's choices in the client application, this data is transmitted to the analysis service as a query. This service, in turn, connects to the database to obtain the relevant information according to the user's choices. Finally, the results are returned to the user panel. Due to the two-stage approach to the proposed analyses, classification and regression models are trained. The variables identified in the previous tasks were used for modeling. Descriptions for selected models used in the proposed solution will be presented below.

The figure below shows the process of managing a supply chain of demand and supply networks on a product-specific basis.

Figure 1. *Supply chain management of an example product (Jaggi 2016)*



Classification models

The task of the classification models is to assign a given order to one of the groups – delayed or non-delayed. Based on the data concerning the dates of placing and final fulfillment of orders (delivery to the customer), we can determine the number of days of fulfillment for individual orders in the historical data set. This approach, combined with the user's definition of a cut-off threshold – the number of days above which a shipment is considered delayed – allows us to build precisely the classification models that can tell us the probability of delayed delivery (Hastie, 2009).

Random Forest

Random forest is an example of an aggregated model algorithm that combines multiple weak classifiers (decision trees in the case of random forest) (Schoniau 2020) to produce a single outcome model with high predictive power. The result is obtained by a ‚voting' process where each classifier predicts the resulting class, and then a moda is extracted from all the results.

$$C = \text{argmax}_i(n_i)$$

where n_i is the result of the i-th tree.

Random forest is a generalization of the bagging method, improving the technique of building single classifiers by randomly drawing a subset of the predictors that make the tree most often the root of all available columns used.

$$m = \sqrt{p}$$

XGBoost

Like Random forest, XGBoost is an aggregated model based on a decision tree classifier. Unlike Random Forest, however, XGBoost is a serial structure that uses a gradient-boosting algorithm. The XGBoost algorithm, by modifying individual trees in each pass, improves the errors of the previous classifier using the gradient descent method to minimize the loss function.

Model prediction in the classification task is achieved by weighted majority voting and applying a logistic function on the sum of probabilities.

$$p_i = \frac{1}{1 + \exp\left(-\sum_{k=1}^{K} f_k(x_i)\right)}$$

where $f_k(x_i)$ is the score of the k-th tree for the i-th observation and p_i is the probability of the resulting class for the i-th observation.

LightGBM

LightGBM is also a gradient enhancement technique based on decision trees, but it was created to increase model performance and reduce memory consumption(Guolin 2017). It uses two innovative techniques:

- Gradient-based one-side sampling (GOSS) – different data instances play a distinct role in calculating information gain. Instances with more significant gradients (i.e., untrained instances) will contribute more to information gain. GOSS keeps those instances with large gradients (e.g., more important than a predefined threshold). It randomly removes only those instances with small gradients to maintain the accuracy of the information gain estimation. This approach can lead to more accurate gain estimation than uniformly random sampling, with the same target sampling frequency, especially when the information gain value has an extensive range.
- Exclusive feature bundling (EFB): Multivariate data is usually very sparse, which allows us to design a nearly lossless approach to reducing the number of features.

In particular, many features are mutually exclusive in the sparse feature space, i.e., they never simultaneously take on non-zero values. Exclusive variables can be safely combined into a single feature (a package of exclusive features). Hence, the complexity of building the histogram changes from O(data × feature) to O(data × package), while package<<feature . Therefore, the speed of the training structure is improved without compromising accuracy. These techniques meet the limitations of the histogram-based algorithm,

which is mainly used in all GBDT (gradient enhancement decision tree) methods. The two techniques GOSS and EFB form the characteristics of the LightGBM algorithm. They combine to make the model perform efficiently and give it an advantage over other GBDT methods.

Regression models

The task of regression models is to estimate the delivery time of a selected good based on learning from historical data. Such an estimate, combined with the probability of delay determined by classification methods, can provide meaningful delivery time information for new shipments and, combined with a significance analysis, determine how individual variables affect the prediction. The user will then be informed that the delivery time depends on, for example, its weight or the supplier's country.

Random Forest

A random forest for regression is a model that aggregates multiple weak regressors (usually regression trees) to form a single vital regression function. The model is based on a ‚voting' process, where each regressor predicts a value, and the final value is usually predicted as the mean or median of all predictions:

$$y = \frac{1}{N}\sum_{i=1}^{N} y_i,$$

where y is the predicted value and y_i is the result of the i¬th regressor.

Random forest improves the building of single regressors by randomly selecting a subset of the predictors that make the tree. The most commonly used is the root of all available columns , which affects the trees' diversity, reducing the model's variance.

XGBoost

XGBoost for regression is also a tree-based aggregation model but uses a gradient enhancement algorithm (Tianqi, 2016). The XGBoost algorithm improves the errors of the previous regressor by modifying individual trees in each pass, using a gradient descent method to minimize the loss function. A weighted sum of tree predictions obtains the model prediction in the regression task:

$$y_i = \sum_{i=1}^{K} f_k(x_i),$$

where y_i is the predicted value for the i-th observation and f_k (x_i) is the result of the k-th tree for the i-th observation.

LightGBM

LightGBM for regression is a gradient enhancement technique based on regression trees optimized for performance and minimal memory consumption. It also works in analogy with classification. GOSS allows different data instances to have different meanings when calculating information gain. Instances with higher gradients (e.g., under-trained) have a more significant impact on information gain. EFB, on the other hand, allows sparse features to be combined into exclusive bundles, reducing the complexity of model building without sacrificing accuracy. These techniques enable LightGBM to perform efficiently and outperform other GBDT (Gradient Boosting Decision Trees) methods regarding performance and prediction accuracy in regression tasks.

Creating an API

As mentioned in the previous descriptions, an application performing analyses of order fulfillment and delays has been built in Python and run as a Fast API service to which we can send queries. To create the FastAPI service, classes define the input and output parameters and functions that perform specific tasks. The necessary libraries are, of course, imported at the outset. Most activities (e.g., creating and training models, making predictions, creating descriptions) are performed using functions in the functions.py file.

The FastAPI() application is then defined, and CORS-related settings are passed on. Cross-source resource sharing (CORS) refers to the situation where the front end running in the browser has JavaScript code communicating with the backend, and the backend has a different ‚origin' than the front end. The following lines define individual classes relating to error handling and input/output handling for particular tasks. The first of these concerns the training and selection process of the optimal model, for which the input is the number of days above which a shipment is considered delayed. At the same time, the output is a description of the model and a list of graphs regarding the quality of its fit. The following class concerns forecasting lead times for a given commodity. The input is a set of parameters, including weight, order value, supplier and recipient country, and the commodity group to which the ordered product belongs. The output is a description of the prediction results, a table of the relevance of the individual variables to the generated result, and a description of these relevances.

The last classes refer to allocating the user-selected commodity to the clusters created from the data about the lead time. After specifying the commodity ID, the user is provided with a description of the clustering, a box plot of the values in the individual clusters, a clustering table, a description of the assignment of the commodity to a given cluster, and the ID of this cluster. In addition, a class is defined that allows the user to check whether already trained models are currently available. Asynchronous functions are then described as reading the input data, calling the individual analysis process, and returning the output data according to the specified parameters.

Containerization

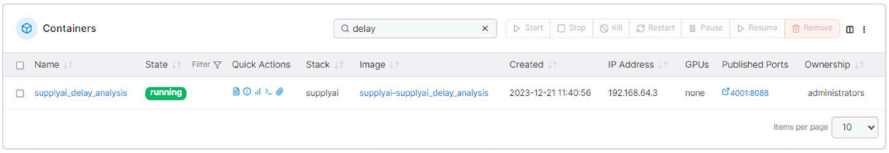
The next step is the containerization process of our service, which will allow it to be immutable and run on different platforms. Docker was used for containerization, as in the previous module. We first specify that we want to use Python 3.8 as the base image, then download the necessary libraries and create a configuration file to connect to the SQL Server database using the FreeTDS driver. In the following steps, the files are copied, and the individual

Python libraries are installed. The most critical libraries from the point of view of the model learning process and the operation of the application, together with their versions, are presented below.

yellowbrick==1.5
scikit-learn==1.2.1
xgboost==2.0.0
lightgbm==4.1.0
pyodbc==4.0.31
dalex==1.6.0
uvicorn==0.20
fastapi==0.91

Once the libraries are installed, the FastAPI application inside the container is run. At this point, we have a built image. A docker-compose.yml file has been created to run it and in the future for consistency in running all services. Once the container is built and running, we can observe its activity, for example, in a tool such as Portainer, which is used to manage containers. The following figure shows that the created service is active and ready to run.

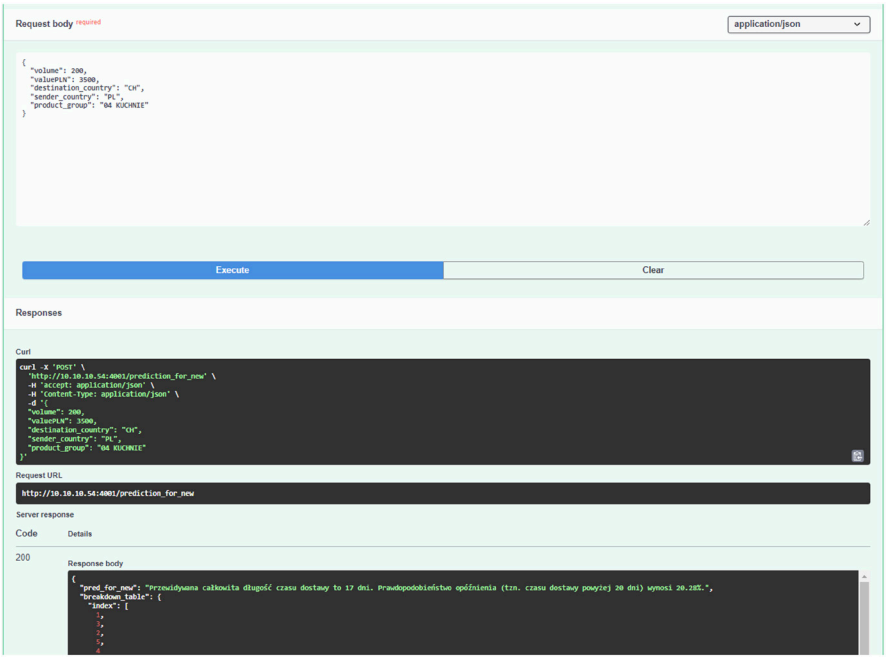
Figure 2. *Running container from the Portainer tool*



Application

We can demonstrate the service's operation in several ways, including from the automatically created documentation. The following figure shows just such a callout.

Figure 3. *How to call the service*



However, since a client application has already been developed to enable user interaction with the module under construction, we will show the service's operation using an example. The figure shows the appearance of the application embedded in the web page.

Figure 4. *Client application – Delay threshold configuration*

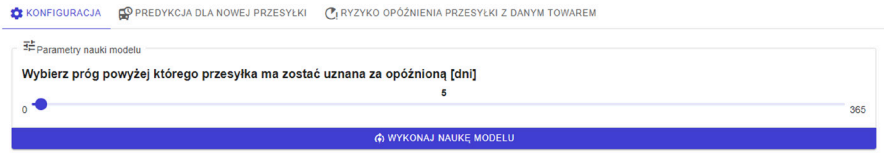


Figure 5. *Client application – Regression model learning results (top), learning outcomes of classification models (down)*

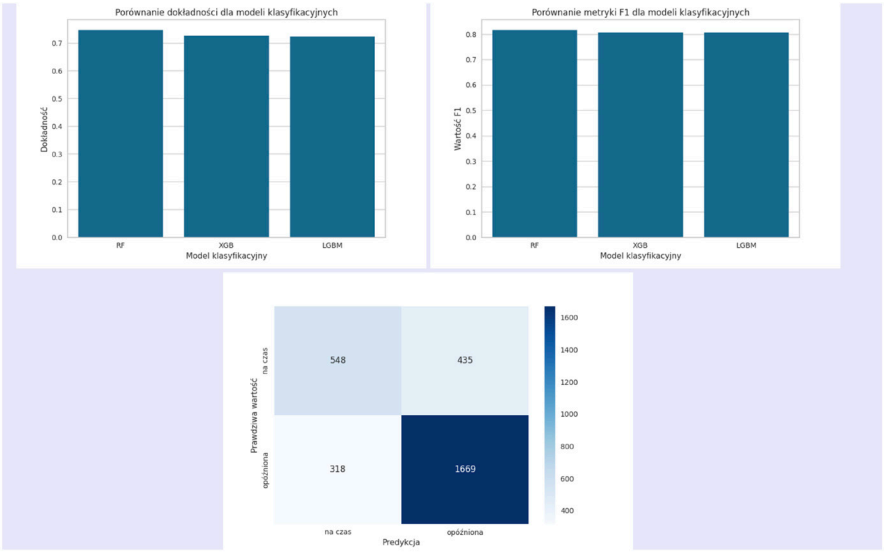


Figure 6. *Client application – prediction for a new shipment*

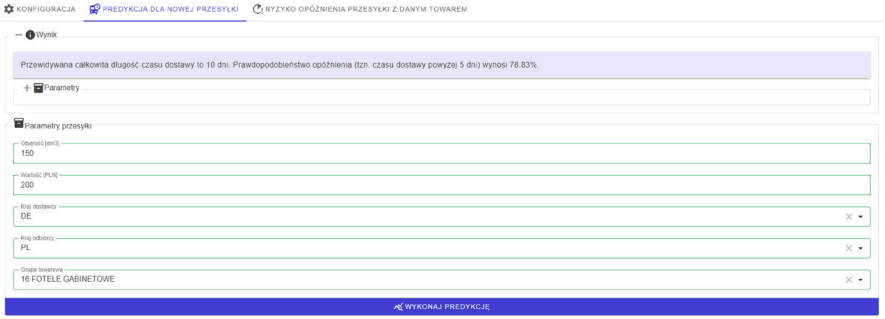
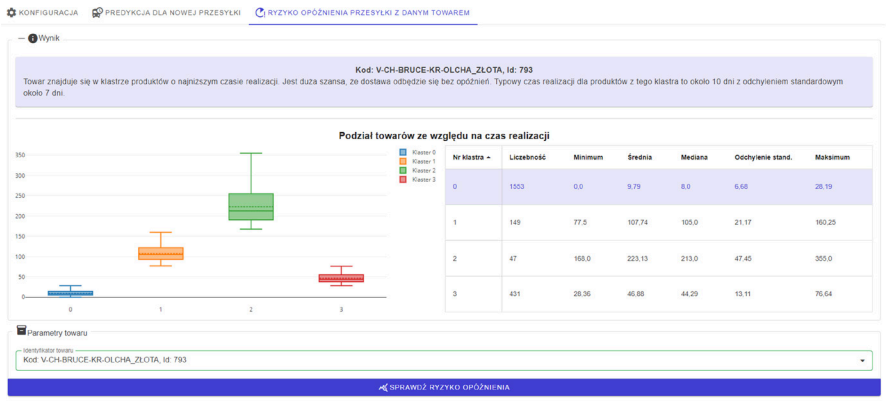


Figure 7. *Client application – Grouping of goods*



Conclusions

Forecasting changes based on analysis of historical data from the system and real-time data acquired from devices. The system design is based on a template referred to as Lambda, consisting of two subsystems: based on historical data stored in databases and analyzed in batch mode and real-time analysis using streaming data analysis methods. Due to its nature, risk estimation of executing complex orders/orders with large amounts of data is carried out using Monte Carlo simulations and artificial neural networks. In today's business environment, ever-sharper and frequent natural and artificial disasters make supply chains more vulnerable. Supply chain disruptions appear to occur more frequently and have more severe consequences. Companies can lose revenue during and after supply chain disruptions and incur high recovery costs. If supply chain managers were more able to measure supply chain vulnerability, they could reduce the number of disruptions and their impact. Quantifying supply chain vulnerability helps managers assess the vulnerability of their supply chains and compare the effectiveness of different risk mitigation strategies.

The functioning of the supply chain, whether in terms of planning or actual operations, is always subject to a certain degree of uncertainty, resulting from hazards and disruptions along the way, often beyond our control.

Risks to the secure supply chain operation can refer to various events that disrupt individual processes, such as the flow of goods, inventory management, delivery, or simply problems in information flow (Szymonik, 2014; Khojasteh, 2018). Such events occurring individually and jointly generate dangerous situations for all participants forming a given supply chain. Therefore, an essential aspect is skillful risk management, which often allows both the identification of potential sources of risk and the determination of its level. In this way, we can prevent risky situations and mitigate risk or minimize its effects when a particular event is unavoidable (Myszak, 2016).

Supply chain management includes managing the transportation or flow of goods and services, including warehousing, shelf life, analysis of goods purchased and goods sold, logistics, etc. Supply chain management helps in planning and executing various activities in the supply chain of a particular organization to build the net worth of the organization by identifying the current market trend related to the demand and supply of any goods or services and synchronizing it to measure the performance of the organization (Jaggi, 2016).

References

Szymonik, A. (2014). Funkcjonowanie łańcucha dostaw w sytuacjach zagrożeń," Logistyka, no. 6

Myszak, J. M., Sowa, M. (2016). The Risk Management in the Aspect of Supply Chain Zeszyty Naukowe Uniwersytetu Szczecińskiego Problemy Transportu i Logistyki, vol. 36, pp. 185–192, doi: 10.18276/ptl.2016.36-19.

Jaggi, H. S., Kadam, S. S. (2016). Integration of Spark framework in Supply Chain Management, Procedia Computer Science, vol. 79, pp. 1013–1020, doi: 10.1016/j.procs.2016.03.128.

Hastie, T., Tibshirani, R., Friedman, J. (2009). The elements of statistical learning, *Springer-Verlag* New York Inc.

Khojasteh, Y. Ed.,(2018). Supply Chain Risk Management. Singapore: Springer Singapore,

Schonlau, M., Zou, R.Y. (2020). The random forest algorithm for statistical learning. Stata J., 20, 3–29.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. (2017). LightGBM: a highly efficient gradient-boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 3149–3157.

Tianqi Chen, and Carlos Guestrin. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ,16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785