FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Benchmarking of algorithms for machine learning
Student:	Tom Svoboda
Supervisor:	Ing. Viktor Černý
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of winter semester 2021/22

Instructions

In an unnamed company selling SaaS products, we want to offer the best service possible to our customers. To add additional value to them we want to predict their needs based on their input data. For this purpose, we can use machine learning algorithms, which create a prediction model for each customer. We assume various algorithms and their configurations can have different success rates for each customer type. The goal of this thesis is to create a tool, that can automatically evaluate the quality of created prediction models.

- Create a methodology, which evaluates the quality of prediction for each model against expected results.
- Apply this methodology in a tool, which automatizes the evaluation of these models.
- The tool will provide an output as feedback for developers of machine learning algorithms in a way, that will improve the quality of said models.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 24, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Benchmarking of algorithms for machine learning

Tom Svoboda

Department of Software Engineering
Supervisor: Ing. Viktor Černý

January 7, 2021

Acknowledgements

I want to thank Ing. Viktor Černý for his guidance, supervision, and help with the thesis. Also, my family and close friends for support and neverending belief in me.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on January 7, 2021 . . . .

Citation of this thesis

Svoboda, Tom. *Benchmarking of algorithms for machine learning.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Cílem práce je vytvořit metodiku pro hodnocení modelů strojového učení. Následně použít tuto metodiku v nástroji, který automatizuje hodnocení modelů a dává zpětnou vazbu jejich vývojářům.

Výsledkem práce je popsaná metodika pro hodoncení modelů, která je využitelná i bez automatizace. Nástroj byl implementován jako distribuovaný systém, který lze použít samostatně nebo lze napojit na další systémy.

Klíčová slova hodnocení modelů strojového učení, porovnání prediktivního modelování, metodika hodnocení modelů, systém pro vyhodnocení modelů, strojové učení, umělá inteligence

Abstract

The goal of this work is to create a methodology to evaluate machine learning models. Then use this methodology in a tool, to automate the evaluation of models and provide feedback to their developers.

The result of this work is a described methodology for model evaluation. The methodology can be used on its own with no automation. The tool was implemented as a distributed system that can be used as a standalone solution or integrated into other services.

Keywords machine learning model evaluation, benchmark predictive modeling, model evaluation methodology, model evaluation system, machine learning, artificial intelligence

Contents

List of Figures

Introduction

Motivation

In an unnamed company selling Software as a Service (SaaS) products, we want to offer our customers the best service possible. To add additional value to them, we want to predict their needs based on their input data. For this purpose, we can use machine learning algorithms, which create a prediction rule for each customer. We assume various algorithms and their configurations can have different success rates for each customer type.

The motivation for this thesis is to help with the development of such algorithms and provide a tool for selecting the best one for each customer.

The goal is not to compare machine learning algorithms in general but to compare their usability on a selected problem.

This work is focused only on supervised machine learning.[1]

Aim of the Thesis

This thesis aims to create a tool that can automatically evaluate the quality of a machine learning algorithm for prepared data sets.

The initial goal is to create a methodology that evaluates the quality of each model to predict against expected results. The consequent goal is to apply this methodology within a tool that automates the evaluation of these models. The final goal is to provide an output as feedback for developers of machine learning algorithms to improve the quality of said models.

Thesis structure

The first part of the thesis is analytical. The basics of machine learning and predictive modeling are described in Chapter 1. The process of development

[1]This choice is explained in Section 1.1

and evaluation of supervised learning models is covered in Chapter 2. Existing technology and services that implement or support predictive modeling are covered in Chapter 3.

The second part is practical. In Chapter 4, it is described how to automate the methodology and provide feedback.

Chapter 1

Machine Learning

“Machine learning is a set of methods that can automatically detect patterns in data and then use the uncovered patterns to predict future data or perform other kinds of decision-making under uncertainty.”[1]

Input data is a vector of input measurements, also known as ”question”

Label is an output measurement, also known as ”answer”

Model is a mathematical model for predicting labels for input data

Observation is input data with its true label

Supervised learning fit model to match observations (predictive)

Unsupervised learning find interesting patterns in the input data or observations themselves (descriptive)[1, 2]

Reinforcement learning teach an agent new behavior through trial-and-error interactions with a dynamic environment[3]

1.1 Choosing an area of machine learning

This work is focused only on supervised learning. It can solve relevant problems that do not have existing industry solutions. It has a less complicated evaluation process.

1.1.1 Considered aspects

- type of problems that can be solved for a SaaS company
- feasibility - existing solutions, complexity

All learning methods are useful in a SaaS environment. In supervised learning, we can think of business predictions such as customer churn or possible product features such as automating user actions. There is automatic content processing for unsupervised learning, such as cluster analysis or finding outliers in customer behavior. For reinforced learning, it could be preventing security threats or predicting infrastructure load for better auto-scaling.

The problems solvable with reinforced learning are common across SaaS companies. They already have existing industry solutions that can be used.[4, 5, 6] Spending resources to develop solutions for them within an organization might not be feasible.

Both supervised and unsupervised learning can solve many problems for SaaS companies that have the company-specifics to them.

Evaluating unsupervised learning is challenging. The goal of unsupervised learning is to find interesting patterns. Assessing how much the finding is interesting depends on each particular problem. That makes it harder to generalize the evaluation process.

Evaluation of supervised learning is easier to automate.[2] As supervised learning can solve relevant problems that do not have existing industry solutions and is the most feasible, this work focuses on this machine learning method.

[2]The methodology of evaluating supervised learning is explained in Chapter 4.

Chapter **2**

Supervised learning

2.1 Types of supervised learning

There are two types of problems supervised learning solves.[7]

Regression predicting continuous scalar value for input data (predicting a person's height from their sex and age).

Classification to classify input data (predicting illness from the patient's symptoms).

Classification can be further split into the following types:[8]

Binary input is classified into one of 2 classes

Multi-class input is classified into one of the l classes

Multi-label input is classified into several of the l classes

2.2 Process of developing models

We have clearly defined a prediction problem that is solvable with supervised learning. We need a data-set containing observations from which the learning algorithm will learn. If we have only input data, we need to add their labels manually. We need to establish an evaluation protocol (section 2.3) and choose a metric (section 2.4) appropriate for the problem. We should assess the success threshold. Now we select or develop the learning algorithm (section 2.5) that is suitable for the problem. Then the evaluation protocol is executed to assess the quality. The models that surpass the success threshold are candidates for usage.[9, 10]

2.3 Evaluation protocol

With machine learning, we create models that will process data that do not exist or are not available. For that, we can not measure the real performance but only predict it. We can use the existing data to measure the performance with varied success. We should not test the model accuracy on the data it has trained on, as it would promote over-fitting. We can first put aside some data for validation and use the rest for training the model.[9, 11]

2.3.1 Holdout Validation

This method consists of setting apart some portion of the data as the validation set. It is common to use $\frac{1}{5}$ to $\frac{1}{3}$ of the data for validation. This method gives a pessimistic estimation of the accuracy as the estimate is biased by the selection.[12]

2.3.2 Iterated K-Fold Validation with Shuffling

We can split the data into k folds (random, mutually exclusive and with the same size), use each fold as a validation set, and learn the model from the rest. Then we can compute the performance by averaging the performance of the k models.

This method helps overcome selection bias and over-fitting. It is excellent for smaller data-sets as it maximizes the use of data as only $\frac{1}{k}$ is used for validation while improving the accuracy of the performance metric.[12] For bigger data-sets learning $k - 1$ additional models can be expensive.

2.4 Evaluation metrics

Learned models are approximations of reality. Therefore, there can be an error in their predictions. For regression models, we can measure the size of the error. Classifiers produce either true or a false classification.

To evaluate the correctness of the model, we need to choose an appropriate evaluation metric. The evaluation metric quantifies the extent to which the predicted labels for a given input-data are close to these observations' true labels.[1]

Each metric can cover different aspects of trained models. They can be sensitive to outliers, respect, or account for the distribution of training data. Some metrics are better for subjective labeling, and others are for data with outliers that cannot be explained from provided data. Which properties are important depends on the problem itself, so there is no single best metric.[8]

2.4.1 Confusion matrix

Confusion matrix A stores counts of matches and mis-matches of n-class classifier in an experiment. The value of element $A_{i,j}$ is a number of predictions in an experiment, where class$_i$ was predicted and class$_j$ was the true label.

Following example is a confusion matrix for a binary classifier.

$$\begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix} \tag{2.1}$$

Where TP is number of true positives, FP is number of false positives, FN is number of false negatives and TN is number of true negatives.

Sometimes it can be useful to analyse performance for individual classes. We can convert multi-class confusion matrix M to a set of binary confusion matrices for each class. Each will have 2 classes: $class_i$ and *not* $class_i$, where *not* $class_i$ encapsulates the other classes.

$$\begin{pmatrix} TP_i & FP_i \\ FN_i & TN_i \end{pmatrix} \tag{2.2}$$

Where TP_i is number of true predictions to $class_i$, FP_i is number of false predictions to $class_i$, FN_i is number of false predictions to *not* $class_i$ and TN_i is number of true predictions to *not* $class_i$.

2.4.2 Accuracy

Accuracy is ratio of correct predictions to all predictions. For binary classification we can use values in confusion matrix:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

Average accuracy for n-class classification:

$$Accuracy = \frac{1}{n} \sum_{i=0}^{n} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{2.4}$$

2.4.3 F1 score

It is the harmonic mean of recall r and precision p. Can be used when both recall and precision are equally important. It can take on values between $[0; 1]$, where higher value is better.

$$F = \frac{2pr}{p + r} \tag{2.5}$$

$$p = \frac{TP}{TP + FP} \tag{2.6}$$

$$r = \frac{TP}{TP + FN} \tag{2.7}$$

For multi-class classifiers, we can compute F1 score for each class.[3]

2.4.4 Matthews correlation coefficient

Matthews correlation coefficient (MCC) is used over pure accuracy for unbalanced data as it accounts for distribution of data within the experiment. Opposed to accuracy, a bigger error on a less represented class will be more noticeable. MCC can take on values between $[-1; 1]$, where a higher value is better.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.8}$$

MCC was also generalized for the multi-class case.[13]

$$MCC = \frac{\sum_k \sum_l \sum_m C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\sum_k (\sum_l C_{kl})(\sum_{k'|k' \neq k} \sum_{l'} C_{k'l'})}\sqrt{\sum_k (\sum_l C_{lk})(\sum_{k'|k' \neq k} \sum_{l'} C_{l'k'})}} \tag{2.9}$$

2.4.5 Mean absolute error

Mean absolute error (MAE) is used to quantify size of error for regression models. It can take on values between $[0; \infty)$, where lower value is better. The value of error is in the same dimension as the values have. This allows the result to be interpreted more easily.[14]

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{f}(x_i)| \tag{2.10}$$

Where $\hat{f}(x_i)$ is the prediction of label for observation$_i$ and y_i is the true label of the observation$_i$.

2.4.6 Mean squared error

Mean squared error (MSE) is also used to quantify size of error for regression models. It is more useful than MAE if we want to prevent outliers with huge error. It can take on values between $[0; \infty)$, where lower value is better.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 \tag{2.11}$$

[3] See 2.4.1

Where $\hat{f}(x_i)$ is the prediction of label for observation$_i$ and y_i is the true label of the observation$_i$.

2.5 Algorithms

There are many algorithms and their variants that have different results on different kinds of problems. What they have in common is their interface. On input, they require a labeled data-set. On output, they produce a model that can predict labels for additional data.

The choice of the algorithm does not affect the evaluation protocol. For reference, there are some examples of such algorithms:

- Artificial neural network
- Decision tree learning
- Linear regression
- Naive Bayes

To achieve better results, we can include in the algorithm other techniques, such as pre-processing.

Chapter 3

Existing technology and services

Using machine learning to solve tasks is a complex problem with multiple steps. It involves understanding the problem. The necessary part of that is to gather and study the data describing the problem. To help with that, we can use visualization tools and tools that streamline data processing. Usually, we need to prepare the data needs for machine learning algorithms to be successful. To give an example: making predictions on a text - as ML methods do not work well directly with language, but rather numbers, we can use natural language processing methods to convert it to the machine-readable values.

With the pre-requisites satisfied, we can build the model with machine learning methods. Then we can evaluate the models and approve them for usage. We can use this evaluation to guide us to better solutions and even automate the selection of the best method.

Once we have a successful model, we can decide to use it to solve real-world tasks. That includes choosing the channel for providing the model and how to maintain it. We may need to monitor their real performance and react if it would degrade. The degradation happens when the environment of the problem changes and the models gets outdated. Large scale applications can include continuous learning to avoid this issue.

As the machine learning industry progresses, there are several existing solutions that we can use. We can use most of them in the way that can complement each other although some do overlap. The solutions range from solving individual step in the process to providing full end-to-end solution with ranging level of automation.

We can use conventional database management systems, big data solutions like Apache Hadoop, or a simple file system to store the data.[15] We can manage the data by hand on a case-by-case basis or use tools to streamline the process.[16]

Machine learning engineers can use existing libraries that implement methods described in Chapter 1. Namely, Scikit-learn, PyTorch, and SciPy.[17, 18, 19] They are open-sourced and well documented. To use them, we need to integrate them into a process outlined above.

There are existing machine learning development tools that integrate with existing cloud service providers: H2O.ai, Azure Machine Learning, Amazon SageMaker, or Cloud AutoML from Google.[20, 21, 22, 23] These services provide end-to-end solutions.

They provide the necessary functionality and infrastructure to apply the methods described in this document. Some have provided specialized solutions for the type of machine learning usage that this work supports. Microsoft has Many Models Solution Accelerator, which automates building multiple models on Azure Machine Learning.[24] Amazon has Multi-Models Endpoint providing functionality to serve multiple models from one server to cut down operating costs.[25]

There are tools that help with model evaluation such as Neptune and Guild AI.[26, 27]

Neptune provides a system of records for executed experiments. We can use their library in existing model training scripts to track model evaluations and parameters used to get them. The results are available to explore in provided dashboard.

Guild AI instead provides a CLI tool that can execute provided the scripts. It automatically declares new experiments in its system and tracks results as well as used parameters to get them. Thanks to that we can later analyze individual runs.

My solution is focused on running the experiments on multiple models for single problem easier.

Chapter **4**

Automating evaluation

The goal is to make the development of models easier. The development of models is an iterative process. What works for one kind of problem and a set of data may not apply to others. To understand the success of a solution, we need to evaluate it. As the evaluation is a repetitive process, it is a great candidate for automation.

4.1 Methodology

Before we can start, we need to understand the problem we want to tackle. We need to gather all information related to the problem that may impact the outcome we want to predict. We must structure the information so that the machine can process it. We need to have enough instances of the problem recorded for the learning algorithms to be successful. Every recorded instance used in a supervised learning algorithm must have defined the expected outcome (label).

The next step is to establish the evaluation process. The flow of this process is illustrated in figure 4.1.

The **first step** is to design the evaluation protocol. We need to decide which protocol to use. We could use Holdout or K-Fold validation. See Section 2.3 for more information.

The **second step** is to select the evaluation metric for the protocol The type of the problem indicates which metric we can use. Also depending on the problem, we need to identify what is essential.

We should set the success threshold of the problem for the metric. We can derive the value from an existing solution (e.g., we want to improve on an older model or some custom solution). There can be business requirements that imply minimal value. We can use this threshold to filter out solutions that are not usable. Note that solutions passing a metric may still not be great as the metric quantifies complex problems into a single number. Also, the quality of the learning data limits the ability of the model.

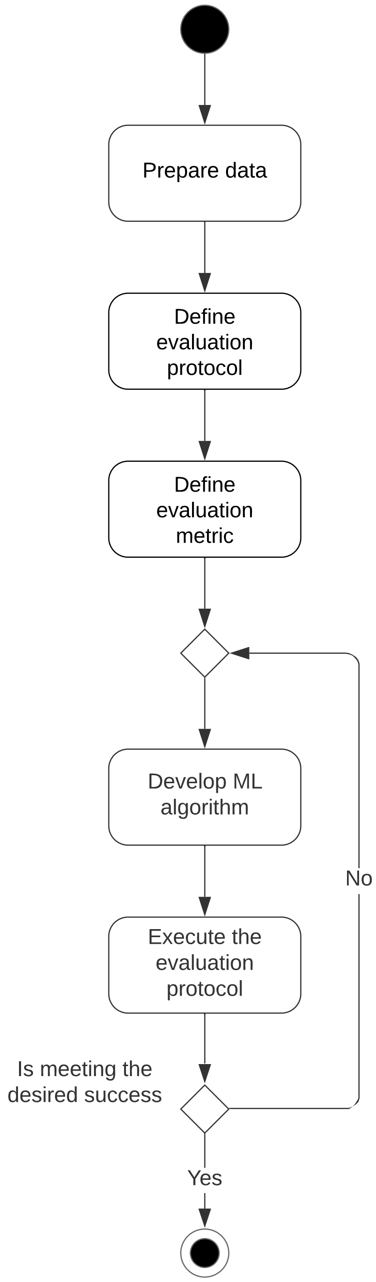

Figure 4.1: Evaluation activity diagram

The **third step** is to prepare the data for training and evaluation according to the evaluation protocol. We need to ensure the task's data do not change to achieve reproducible results and make them directly comparable. To achieve that, we will persist both the data and the selection of data.

The **fourth step** is to execute the evaluation protocol to get an evaluation.

4.2 Requirements

- Define evaluation metrics for a particular problem only once and use it for all customers and different learning algorithms
- Results for different algorithms are reproducible and directly comparable for the same instance of a problem
- For other tasks in machine learning we can use 3rd party services - e.g., training the models

4.3 Domain model

The domain of evaluation is illustrated in figure 4.2. The primary entity is the *problem* itself. The problem has a title and optional description, where we can track additional information. It has assigned some *evaluation protocol* and an *evaluation metric*. Each problem is also described by a set of data of customers (*customer data*). The customer data is specific to the problem and will be used for training. Each problem can be solved by some machine learning (ML) algorithm (*ML Algorithm*). Such ML algorithms can be used in an *experiment* to be evaluated for each customer. Each customer is evaluated using their data in a *customer experiment*.

4.4 Design

The evaluation protocol, success metrics, and ML algorithms are procedures that a data scientist would develop. The challenge is that we need to use them in the automated process and allow flexibility in their implementation.

I was considering three options while accounting for the requirements in Section 4.2:

The first was to define the algorithm in an interpreted language (Python). It would use a provided library that would be configured by the system through environment variables. The user could then upload this algorithm to the server, which would then execute.

The second option was to make the algorithm part of the system itself. The user would then need to contribute to the system to add new algorithms.

The third option was to provide a library that would serve as an automation helper. The user would set up problems in the system and then use the

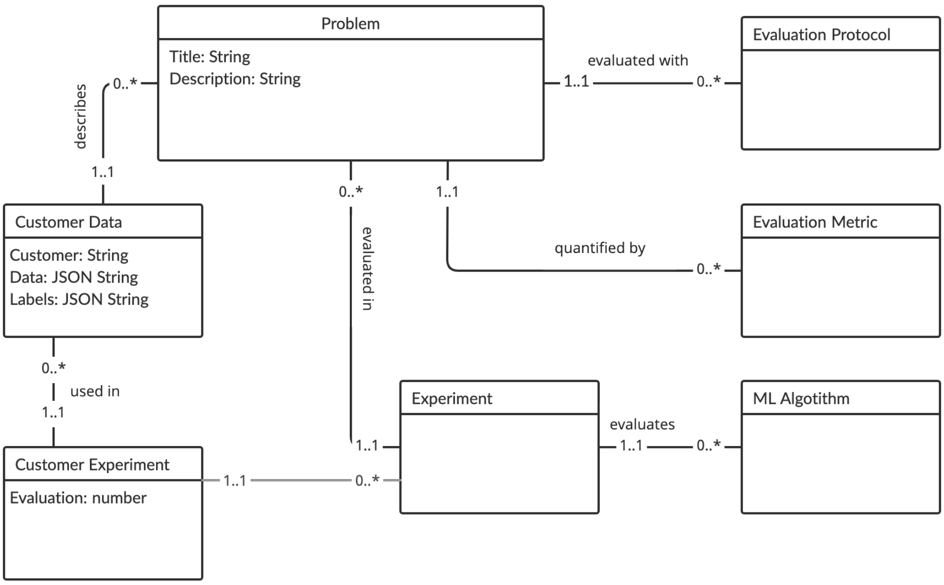

Figure 4.2: Evaluation domain model

library. It would provide the data, evaluate the model, and save the results in the system.

The first option has security implications. If someone accessed the Application Programming Interface (API) of the system, they would be able to execute their scripts. The second option is self-contained and most secure. We can accept only approved contributions to the system - to prevent unauthorized scripts executed in a protected environment. We could restrict the access to the learning data and expose only the results on them - if we would need to protect sensitive data. The third option is the most versatile. It is the approach used by existing services. It is not readily usable if we do not want to use the 3rd party services to execute the learning algorithm or have existing infrastructure. In case we would use the 3rd party service - it is easier to stay in the ecosystem and use the provided APIs to build this solution.

For the above reasons, this work implements the second option: a self-contained system.

4.5 System design

Evaluating machine learning algorithms is a computationally demanding operation. The part of the system responsible for evaluation will often change to include new algorithms we want to test. We can load data into the system from other services so that we can automate this process.

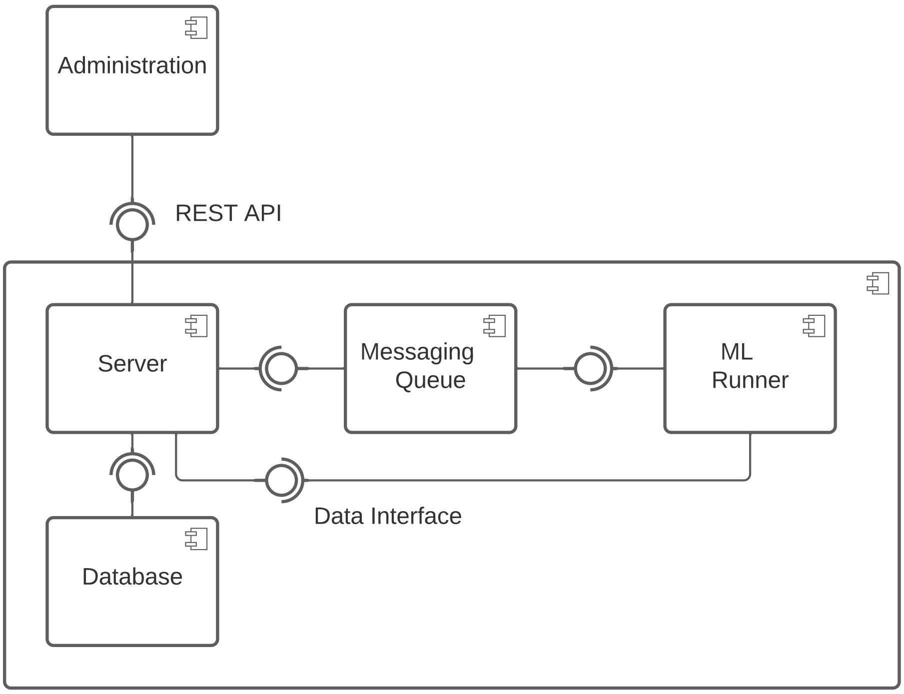

Figure 4.3: Components of the evaluation system

The system is distributed into three services to work under these aspects. Their relation is captured in figure 4.3.

Server is the leading service that manages data and evaluations. It provides an API for the Administration or other internal services.

ML Runner service is executing evaluations.

Administration is an optional service that provides user interface (UI) for the server.

We can use Administration or provided server APIs to declare problems and execute experiments with available algorithms. The *Administration* is using the provided API from Server to send requests to it. The *Server* has a connection to the database, where it can store data provided through API. The *Server* can execute experiments on said data by scheduling evaluation jobs to *ML Runner* through *Messaging Queue*. *ML Runner* is processing evaluation requests from the *Messaging Queue*. Runner sends the results back through the queue. Thanks to that, we can update *ML Runner* with new algorithms while keeping the main Server available and queue further evaluation requests. The *ML Runner* could be scaled to run in multiple instances to speed up the evaluation queue processing.

4.6 Implementation and used technologies

To make the deployment of our service more manageable, we use technology for containerization of the applications. It manages the required dependencies and environment for the applications. In this work, we use Docker.[28] To orchestrate the services we use docker-compose.[29]

To broker messages between Server and Runner, we use RabbitMQ.[30]

PostgreSQL is used as a database management system.[31]

4.6.1 ML Runner

This service is responsible for the execution of evaluations. It is the place to define new algorithms.

We will support Python for developing the algorithms. This programming language is popular among data scientists.[32] All major ML frameworks and services covered in Chapter 3 provide APIs for Python.

The ML Runner service is implemented in Python to simplify the usage of developed ML algorithms.

To communicate with the messaging queue, we will use a library called Pika.[33]

We can execute evaluation for each customer separately to allow processing of the evaluation in parallel when we have multiple ML Runner instances. The evaluation request messages must contain all information needed to execute the experiment on a customer. We use JSON to encode the information in the message.[34] It must contain identifiers for the following algorithms: protocol, metric, and learning algorithm. Also, it must have an identifier of the experiment and customer that will be evaluated. The last requirement is data for the learning algorithm.

To identify the algorithms, we will use the concept of catalogs. By design, the user needs to define their identifiers to register them in a catalog. Each catalog is enforcing the algorithms to use a particular interface. The interface for each type is described at the end of this section.

Identifiers of the experiments and customers are defined in the database.

The amount of data can be large, so we cannot send them in the message.[35] Instead, the system needs to expose the resource to the Runner through HTTP GET method.[36] The address to this resource is part of the message so that the Runner can load the data upon processing.

The algorithms may use some 3rd party Python packages. To allow contributors to add new, we need a package manager with easy configuration.

The package manager of choice is Conda.[37] With Conda, we can manage the dependencies in a YAML configuration file.[38]

Interface of algorithms

ML algorithm creates the model. The input of the ML algorithm is training data to learn on. The output is a model for making predictions.

Evaluation metric evaluates the model. The evaluation metric's input is a list of results in an experiment and a list of actual results.

Evaluation protocol defines how the evaluation is executed. On input, it takes data of a customer, ML algorithm, and evaluation metric. It outputs the evaluation of the algorithm for the provided data.

4.6.2 Server

The system must provide API for other services to interact with it (including *Administration*). We will use REST API as it is the standard for communication between web services.[39]

This service is written in TypeScript.[40] It is an extension of JavaScript providing a type system.

The sources are executed with NodeJS.[41] We use Yarn as a package manager.[42]

The framework of choice is NestJS.[43] It supports object oriented programming (OOP) patterns, including dependency injection (DI).

To work with the database, we use object–relational mapping (ORM) - for our case library TypeORM has support for NodeJS.[44] TypeORM integrates into NestJS framework through @nestjs/typeorm module.[45]

We can use @nestjsx/crud to build the APIs, which provides methods to build CRUD API for TypeORM repositories with @nestjsx/crud-typeorm package.[46] We document the API with OpenAPI Specification standard.[47] We can generate the API specification using @nestjs/swagger module.[48] With that, the documentation is always up to date and available on the service itself. It is accessible on the server on path /api.

4.6.3 Administration

To implement the UI for the Server, we also use TypeScript. The library for building components is React.[49] To scaffold the UI for our Create, Read, Update, Delete (CRUD) APIs we can use @FusionWorks/ra-data-nest-crud.[50] To visualize data in graphs, we use recharts.[51]

4.7 Experiment

Because of privacy and security concerns regarding customers' data, we cannot use them in this thesis without their consent. In the end, we could not get the consent, given there was still outstanding work to be done to follow security protocols regarding the usage of their data.

Instead, we use generated data for a proof of concept. The consequence of this approach is we cannot evaluate the real usage. For example, answer whether different machine learning methods work great for some customers' segments. On the other hand, this approach does not affect the validity of this solution overall. We can simulate the types of problems that will be solved. The system must be able to process them and provide evaluations we would expect.

In the experiment, we use the kind of problem, which properties are similar to problems we have identified for our application.

To give an example: say we have an organization where we have a queue of tasks. Each task has an owner that solves it. Some people are specializing in certain types of tasks. It can be by domain, involved business partner, or any other aspect the task can have. Every person and different organizations may have different approaches to assign ownership, so we cannot create a custom rule for this problem that would serve all cases. Given we have comprehensive metadata about the task and have a history of ownership within the organization, we can create a model to predict the owners.

4.7.1 Preparing data

The problem in the example is a multi-classification problem.

At first, we will declare the problem in Administration of the System (fig. 4.4). After providing the name and brief description, we need to configure the evaluation. For the evaluation protocol, we select Hold Out. For the evaluation metric, we select MCC as it is suitable for multi-classification. We cannot set a success threshold, but we know that values close to 0 mean similar success as random classifier would have. A score of 1 is for perfect models, and -1 means the model is always wrong. The expectation is that the models will perform better than random. Both the protocol and the metric are predefined in the Runner's catalog as they are likely to be used for other problems.

To simulate the problem we can generate data with a selected number of classes. The data will take form of a hyper-cube. The dimension of hyper-cube is the number of attributes we have in the metadata. Each point in the hyper-cube is an instance of the problem with attributes. The points for each class form a cluster that is spread in each dimension using noise. The clusters can overlap. Some attributes contain random values and have no correlation with the classification.

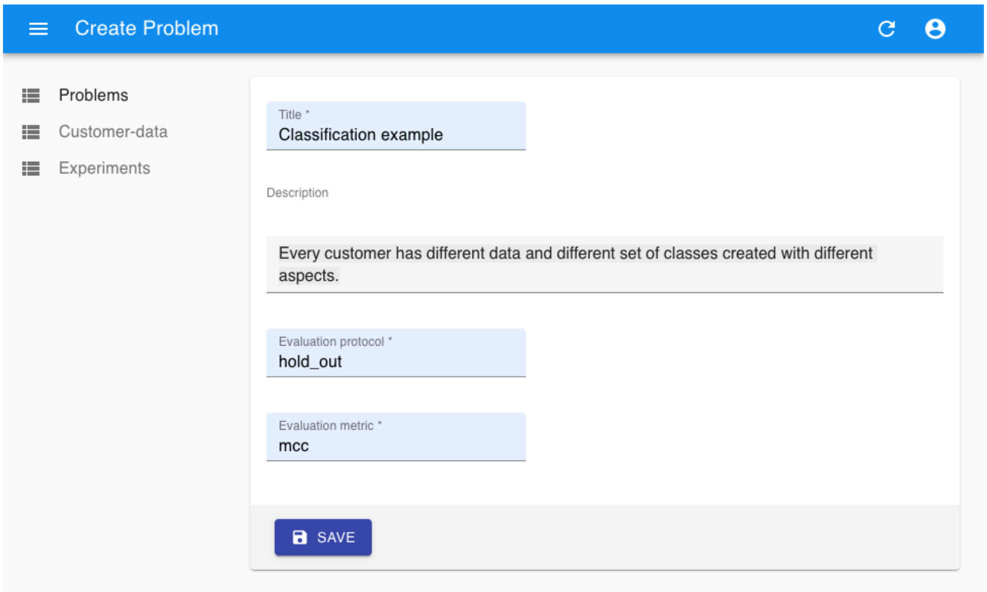

Figure 4.4: Defining a problem in Administration

To generate the data, we are using the Scikit-learn library with a simple Python script. To simulate different customers, we use a different number of classes, amount of data, overall noise, and overlap between the classes. Using the API, we can upload the data to the *Server* and assign them to the problem.

4.7.2 Evaluating algorithms

We will use this data on two machine learning methods (Neural Network and Decision Tree) and a random classifier for reference.

Then we need to develop the algorithm for the random classifier. In Runner, we would create a new entry in the learning catalog and implement its interface. In this case, we will extract unique labels from the training data and make predictions by selecting a random label from them. We need to restart the Runner service to have the algorithm available.

In Administration we can create a new experiment for the problem and use the new algorithm (fig. 4.5). All the data attached to the problem is automatically evaluated. As the customers get evaluated, their results will be displayed there. We can reload the results to update the displayed content.

The results are plotted in a graph and also displayed in a list (fig. 4.6). The results of a random classifier are close to zero, as we would expect for MCC.

Next, we follow the same process with the algorithm for Neural Network. The results (fig. 4.7) are better with values in the range between 0.15 and 0.6.

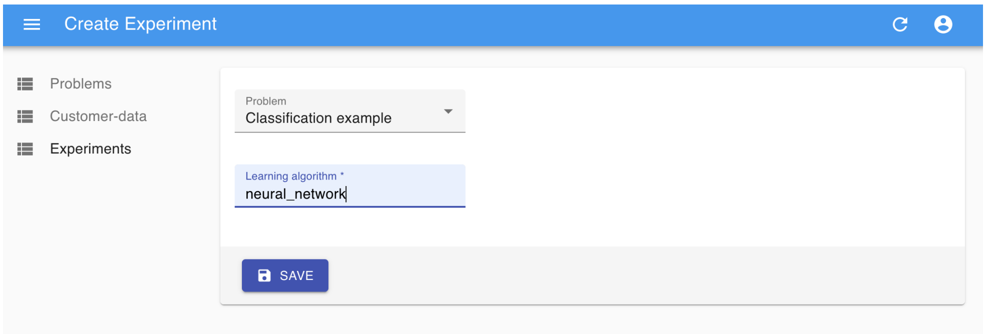

Figure 4.5: Defining an experiment in Administration

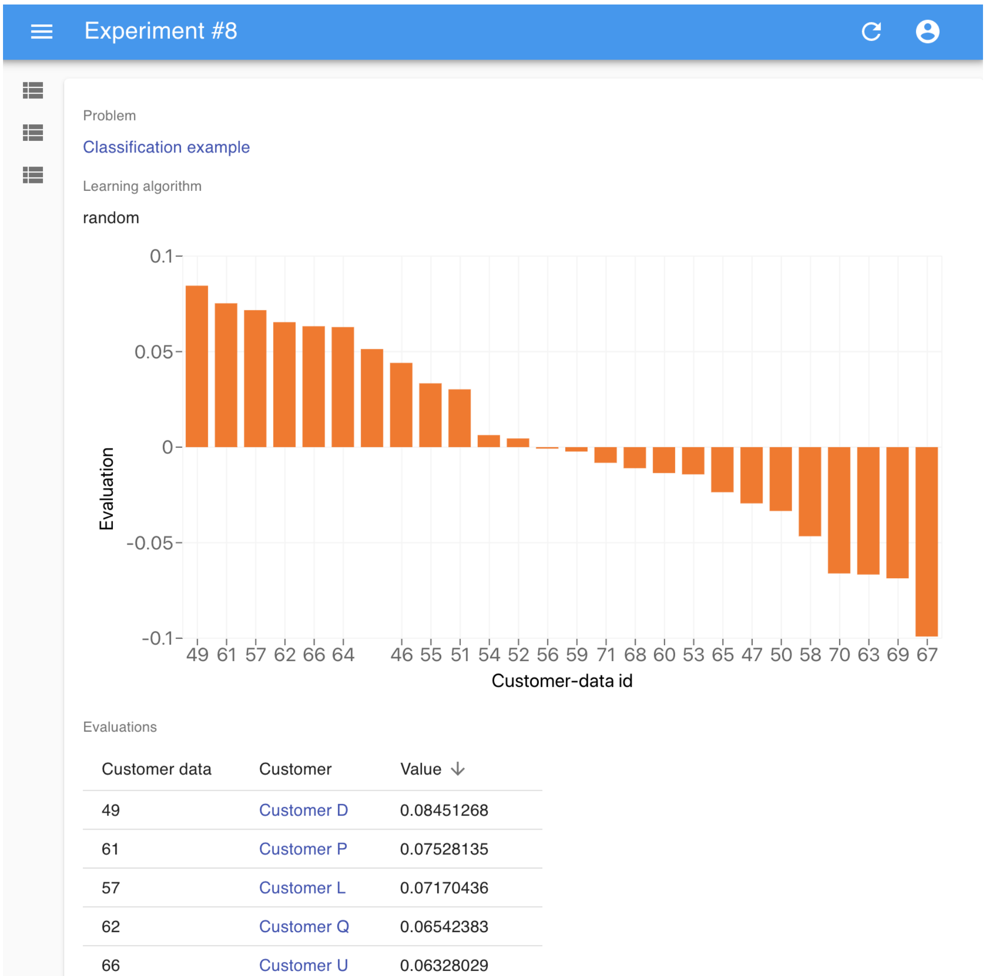

Figure 4.6: Results of the experiment for random classifier

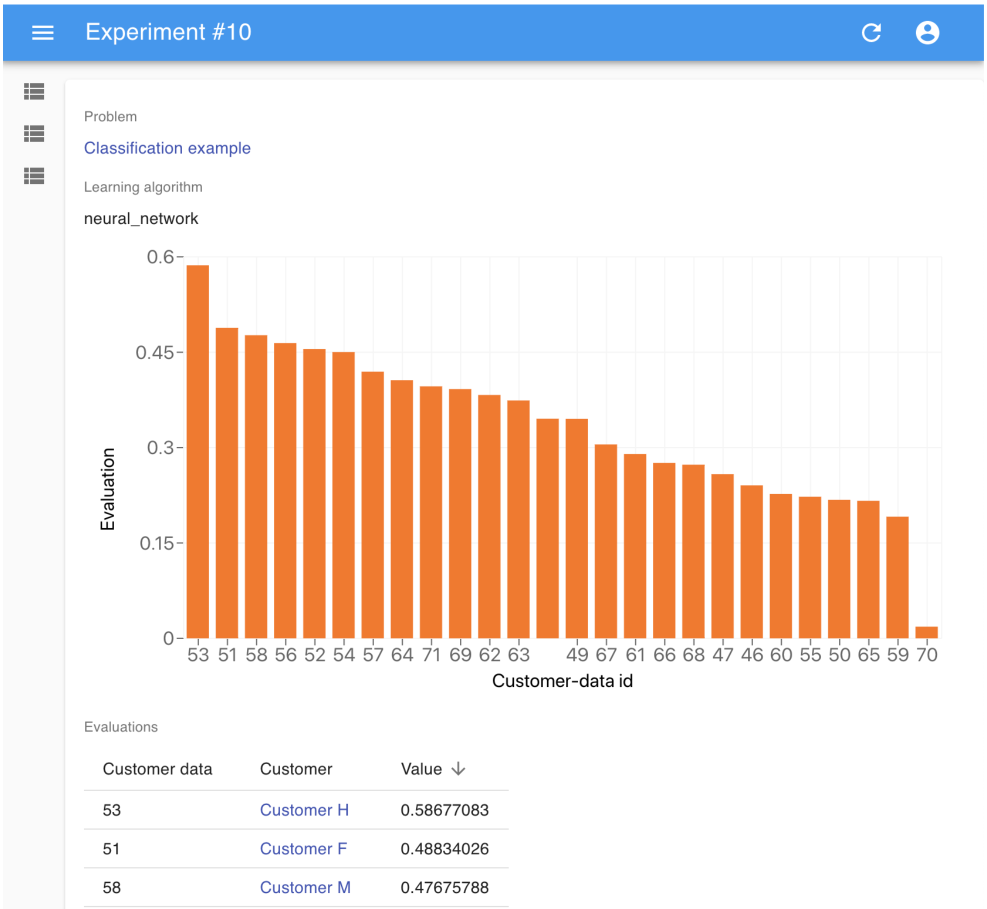

Figure 4.7: Results of the experiment for Neural Network classifier

There is one outlier with a value close to 0.

And with Decision Tree as well. The results (fig. 4.8) are similar to the ones for Neural Network, but trailing slightly.

We can see that the machine learning algorithms are working as have better results than random solution. Overall the models for neural network algorithm performed slightly better than the decision tree on this data set. But at the cost of the time of evaluation as the creation of the model is slower. We can observe a few interesting cases: From the results, we can see that customer 70 for all methods has the same results as a random classifier would have. The generated clusters have too much noise and are close to each other there. Customer 66 has a performance score more than two times higher for the neural network algorithm than for the decision tree. On the contrary: customer 58 has a value higher by 0.1 for the decision tree.

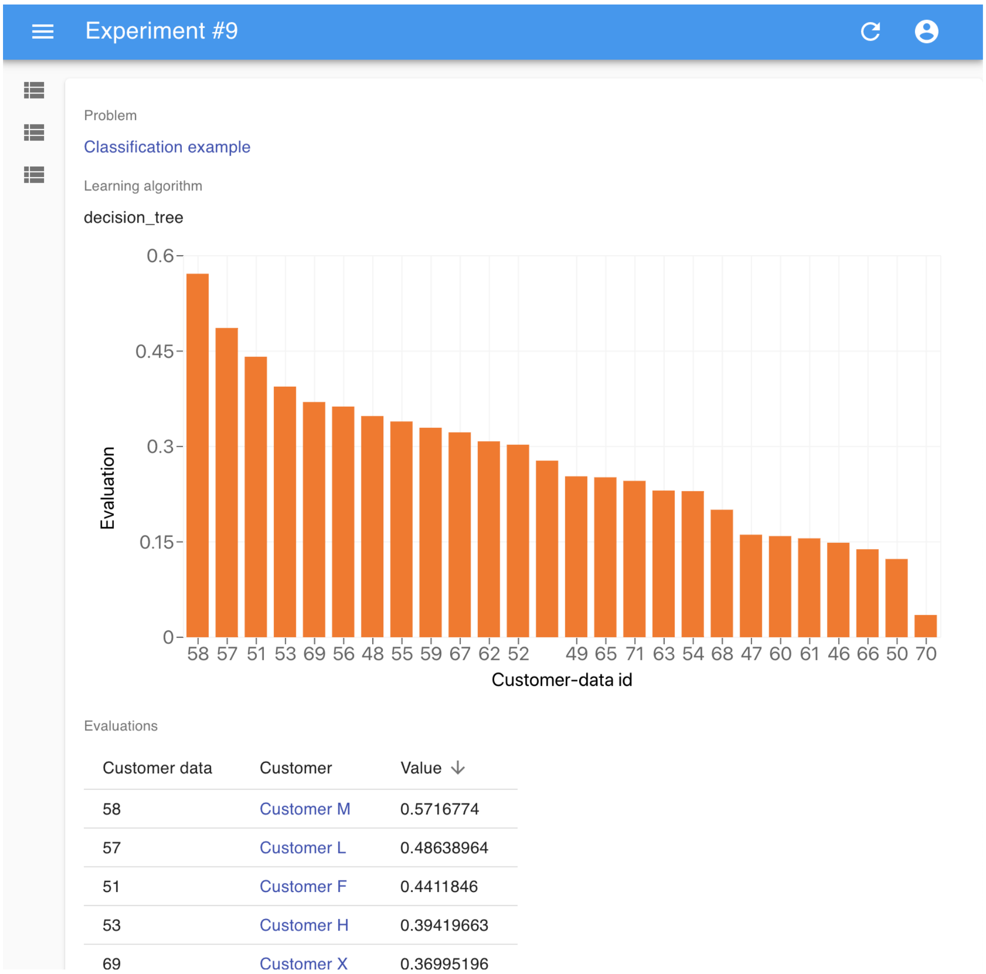

Figure 4.8: Results of the experiment for Decision Tree classifier

4.8 Evaluation

This work was based on the assumption that algorithms can have different success rates for each customer. Based on the experiment, this hypothesis seems valid.

Creating distributed system was a great choice. The system was responsive even when we filled it with a high amount of evaluation requests. The capability to add new algorithms into the system with no downtime is a neat attribute of this architecture.

We could improve a few interactions in the Administration to make the system more comfortable to use. When users are declaring a new problem or experiment, they need to know what keys are available in the catalog. It would be nice to select them from a list. Another issue is that users do not know whether there are still any customer experiments to be evaluated. Showing the status of the process would improve the transparency of the system.

Chapter 5

Conclusion

This thesis aimed to create a tool that automatically evaluates machine learning algorithms for data sets. This goal is fulfilled for problems solvable with supervised learning. The system can accept any learning algorithm, and even algorithms can use another service. This provides freedom in the usage of this service.

The initial goal was to define the methodology of evaluation. As there are many areas of machine learning, this work is focused on supervised learning problems only.

The consequent goal was to use the methodology to automate evaluation. The built tool is capable of automating the machine learning algorithms only with some initial configuration required. The methodology proved to be easy to automate.

The final goal was to provide feedback on the models. This topic proved to be broad and dependent on what is being solved. That is why the system provides feedback on them only with the evaluations of the model, which is the most crucial aspect. It will be interesting to measure and compare models for real customers, but unfortunately, it was impossible to run the experiments before the deadline of this thesis.

Future work could extend the methodology and the tool with unsupervised learning to enable a broader range of problems to be solved more efficiently. The tool could provide guidelines for selecting the best evaluation protocol and metric. This system could evolve into a Model Management Inventory and Governance service. We could persist the models and provide an endpoint to make predictions. The service then could track the health of models by measuring their real performance.

Bibliography

[1] James, G.; Witten, D.; et al. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics, Springer New York, 2013, ISBN 9781461471387.

[2] Shalev-Shwartz, S.; Ben-David, S. *Understanding machine learning from theory to algorithms*. 2014, ISBN 9781107298019 9781107057135, oCLC: 945459331. Available from: https://doi.org/10.1017/CBO9781107298019

[3] Kaelbling, L. P.; Littman, M. L.; et al. Reinforcement Learning: A Survey. *CoRR*, volume cs.AI/9605103, 1996. Available from: https://arxiv.org/abs/cs/9605103

[4] IBM QRadar Security Intelligence — IBM. https://www.ibm.com/security/security-intelligence/qradar, (Accessed on 10/28/2020).

[5] What Is Machine Learning in Security? - Cisco. https://www.cisco.com/c/en/us/products/security/machine-learning-security.html, (Accessed on 10/28/2020).

[6] New – Predictive Scaling for EC2, Powered by Machine Learning — AWS News Blog. https://aws.amazon.com/blogs/aws/new-predictive-scaling-for-ec2-powered-by-machine-learning/, (Accessed on 10/29/2020).

[7] Murphy, K. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series, MIT Press, 2012, ISBN 9780262018029.

[8] Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, volume 45, no. 4, 2009: pp. 427 – 437, ISSN 0306-4573,

doi:https://doi.org/10.1016/j.ipm.2009.03.002. Available from: http://www.sciencedirect.com/science/article/pii/S0306457309000259

[9] Goodfellow, I.; Bengio, Y.; et al. *Deep Learning*. Adaptive Computation and Machine Learning series, MIT Press, 2016, ISBN 9780262035613. Available from: https://books.google.cz/books?id=Np9SDQAAQBAJ

[10] Roman, V. How To Develop a Machine Learning Model From Scratch. https://towardsdatascience.com/machine-learning-general-process-8f1b510bd8af, December 2018, (Accessed on 11/17/2020).

[11] Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014, ISBN 9781107057135. Available from: https://www.cse.huji.ac.il/~shais/UnderstandingMachineLearning/

[12] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. volume 14, 03 2001.

[13] Jurman, G.; Riccadonna, S.; et al. A Comparison of MCC and CEN Error Measures in Multi-Class Prediction. *PLoS ONE*, volume 7, no. 8, Aug. 2012: p. e41882, ISSN 1932-6203, doi:10.1371/journal.pone.0041882. Available from: https://dx.plos.org/10.1371/journal.pone.0041882

[14] Willmott, C. J.; Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, volume 30, no. 1, 2005: pp. 79–82. Available from: https://www.int-res.com/abstracts/cr/v30/n1/p79-82/

[15] Apache Hadoop. https://hadoop.apache.org/, (Accessed on 01/07/2021).

[16] RapidMiner — Best Data Science & Machine Learning Platform. https://rapidminer.com/, (Accessed on 01/07/2021).

[17] scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation. https://scikit-learn.org/stable/index.html, (Accessed on 12/20/2020).

[18] PyTorch. https://pytorch.org/, (Accessed on 01/07/2021).

[19] SciPy.org — SciPy.org. https://www.scipy.org/, (Accessed on 01/07/2021).

[20] Home - Open Source Leader in AI and ML. https://www.h2o.ai/, (Accessed on 01/07/2021).

[21] Azure Machine Learning — Microsoft Azure. https://azure.microsoft.com/cs-cz/services/machine-learning/, (Accessed on 01/07/2021).

[22] Amazon SageMaker – Machine Learning – Amazon Web Services. https://aws.amazon.com/sagemaker/, (Accessed on 12/20/2020).

[23] Cloud AutoML Custom Machine Learning Models — Google Cloud. https://cloud.google.com/automl, (Accessed on 01/07/2021).

[24] microsoft/solution-accelerator-many-models. https://github.com/microsoft/solution-accelerator-many-models, (Accessed on 10/29/2020).

[25] Host Multiple Models with Multi-Model Endpoints - Amazon SageMaker. https://docs.aws.amazon.com/sagemaker/latest/dg/multi-model-endpoints.html, (Accessed on 01/07/2021).

[26] Neptune.ai — Experiment tracking tool for you and your team. https://neptune.ai/, (Accessed on 01/07/2021).

[27] Guild AI - Experiment tracking, ML developer tools. https://guild.ai/, (Accessed on 01/07/2021).

[28] Empowering App Development for Developers — Docker. https://www.docker.com/, (Accessed on 01/03/2021).

[29] Overview of Docker Compose — Docker Documentation. https://docs.docker.com/compose/, (Accessed on 01/03/2021).

[30] Messaging that just works — RabbitMQ. https://www.rabbitmq.com/, (Accessed on 01/03/2021).

[31] PostgreSQL: The world's most advanced open source database. https://www.postgresql.org/, (Accessed on 01/03/2021).

[32] 2019 Kaggle Machine Learning & Data Science Survey — Kaggle. https://www.kaggle.com/c/kaggle-survey-2019, (Accessed on 12/20/2020).

[33] pika/pika: Pure Python RabbitMQ/AMQP 0-9-1 client library. https://github.com/pika/pika, (Accessed on 01/04/2021).

[34] JSON. https://www.json.org/json-en.html, (Accessed on 01/04/2021).

[35] What is the message size limit in RabbitMQ? - CloudAMQP. https://www.cloudamqp.com/blog/2019-05-24-what-is-the-message-size-limit-in-rabbitmq.html, (Accessed on 01/04/2021).

[36] RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. https://tools.ietf.org/html/rfc7231#section-4.3.1, (Accessed on 01/04/2021).

[37] Conda — Conda documentation. https://docs.conda.io/en/latest/, (Accessed on 01/03/2021).

[38] The Official YAML Web Site. https://yaml.org/, (Accessed on 01/04/2021).

[39] Web Services Architecture. https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest, (Accessed on 01/04/2021).

[40] TypeScript: Typed JavaScript at Any Scale. https://www.typescriptlang.org/, (Accessed on 01/03/2021).

[41] Node.js. https://nodejs.org/en/, (Accessed on 01/03/2021).

[42] Home — Yarn - Package Manager. https://yarnpkg.com/, (Accessed on 01/03/2021).

[43] NestJS - A progressive Node.js framework. https://nestjs.com/, (Accessed on 01/03/2021).

[44] TypeORM - Amazing ORM for TypeScript and JavaScript (ES7, ES6, ES5). Supports MySQL, PostgreSQL, MariaDB, SQLite, MS SQL Server, Oracle, WebSQL databases. Works in NodeJS, Browser, Ionic, Cordova and Electron platforms. https://typeorm.io/#/, (Accessed on 01/03/2021).

[45] nestjs/typeorm: TypeORM module for Nest framework (node.js). https://github.com/nestjs/typeorm, (Accessed on 01/03/2021).

[46] nestjsx/crud: NestJs CRUD for RESTful APIs. https://github.com/nestjsx/crud, (Accessed on 01/04/2021).

[47] OpenAPI Specification. http://spec.openapis.org/oas/v3.0.3, (Accessed on 01/04/2021).

[48] nestjs/swagger: OpenAPI (Swagger) module for Nest framework (node.js). https://github.com/nestjs/swagger, (Accessed on 01/04/2021).

[49] React – A JavaScript library for building user interfaces. https://reactjs.org/, (Accessed on 01/03/2021).

[50] FusionWorks/react-admin-nestjsx-crud-dataprovider: Data provider which integrates React Admin with NestJS CRUD library. https://github.com/FusionWorks/react-admin-nestjsx-crud-dataprovider#readme, (Accessed on 01/03/2021).

[51] Recharts. `https://recharts.org/en-US/`, (Accessed on 01/07/2021).

Appendix A

Acronyms

API Application Programming Interface.

CLI Command Line Interface.

CRUD Create, Read, Update, Delete.

DI dependency injection.

JSON JavaScript Object Notation.

MAE mean absolute error.

MCC Matthews correlation coefficient.

ML machine learning.

MSE mean squared error.

OOP object oriented programming.

ORM object–relational mapping.

REST representational state transfer.

SaaS Software as a Service.

UI user interface.

YAML YAML Ain't Markup Language.

Appendix **B**

Contents of enclosed CD

```
├── thesis.pdf ............................ the thesis text in PDF format
├── thesis ............... the directory of LaTeX source codes of the thesis
└── src ................................... the directory of source codes
    ├── admin ......... the directory of source codes of Administration service
    ├── runner ........... the directory of source codes of ML Runner service
    ├── server ................ the directory of source codes of Server service
    ├── README.md ........................................ installation guide
    ├── .env.example ........................... example of environment file
    └── docker-compose.yml .......... configuration file for Docker Compose
```