

CLOUD BENCHMARK

Comparing the offerings of
Machine Learning as a Service



**DR. CHRISTOPH
WINDHEUSER**



**MATTHIAS
KAINER**

Published in 2019



MLAAS VERSUS MACHINE LEARNING FRAMEWORKS 05

THE OFFERINGS OF AMAZON, GOOGLE AND MICROSOFT 07

A PRACTICAL EXAMPLE 10

AZURE MACHINE LEARNING STUDIO 13

AMAZON MACHINE LEARNING 19

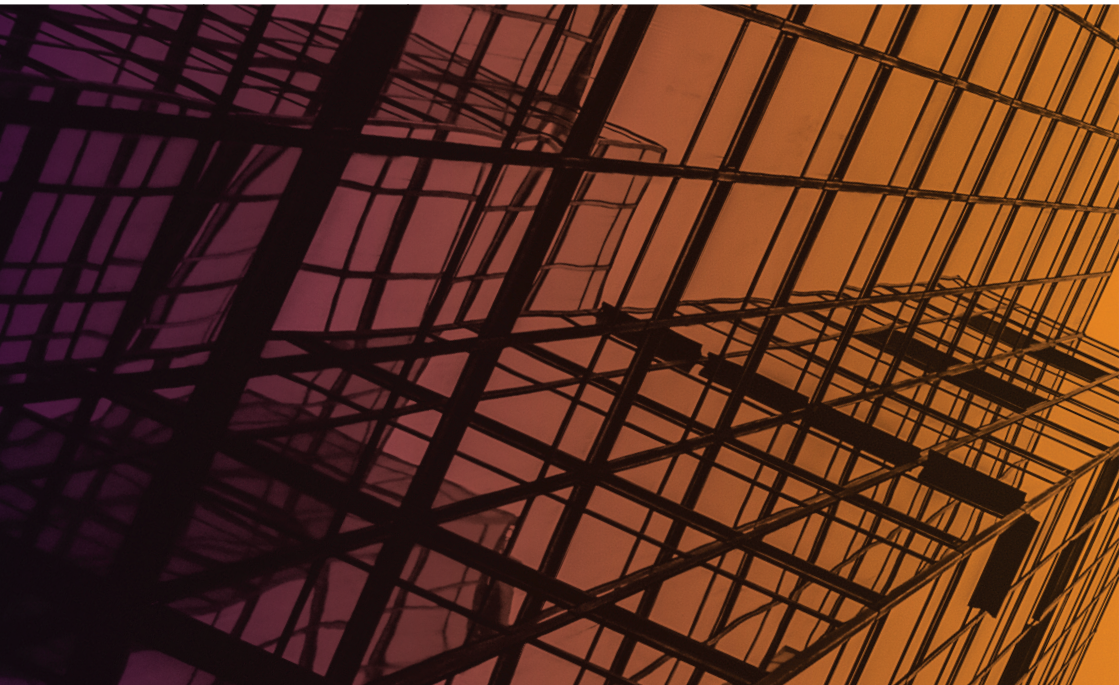
AMAZON SAGEMAKER 26

MICROSOFT AZURE MACHINE LEARNING SERVICE 32

GOOGLE CLOUD MACHINE LEARNING ENGINE 36

CONCLUSION 42

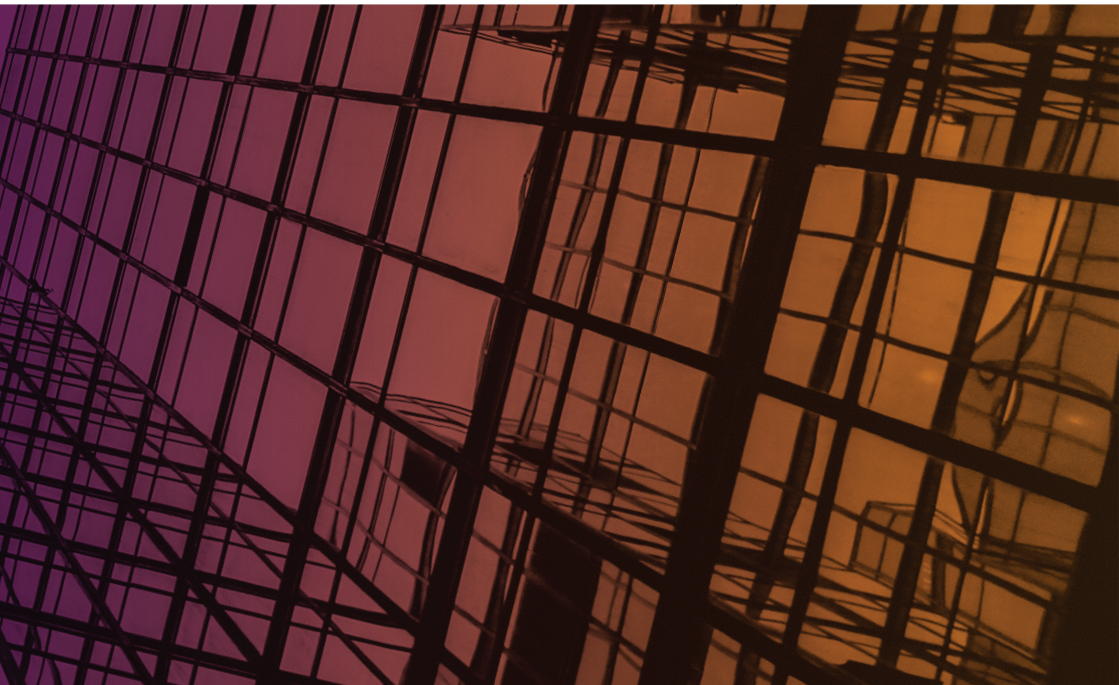
To deploy machine learning algorithms successfully, you'll need both large data sets and powerful hardware to train them. During the development phase, it pays to have access to as much processing power as you can muster — because that enables experimentation systematically with different hyperparameter values. As a result, you may find that it makes sense to utilize cloud resources for a limited period.



Cloud providers were quick to recognize a potential new market, one they've badged 'machine learning as a service' (MLaaS). This term refers to cloud services for the storage and preprocessing of training and test data, a selection of machine learning models, and services for training, optimizing and testing the algorithms and interfaces (APIs) to other programs (also in the cloud or on the premises) – all integrated into a cloud environment. Today, the big three cloud providers — Amazon Web Services (AWS), Google Cloud Platform (GCP) and Microsoft Azure — are locked in a MLaaS feature race, each seeking to tie customers to their platform through the provision of unique functions. The services offered differ considerably in terms of range of functions, target groups and application scenarios. For the end users, that makes it difficult to choose the best MLaaS solution.

In this ebook, we'll outline the principal services offered by Amazon Web Services, Google Cloud Platform and Microsoft Azure and describe their typical application scenarios. To ensure we were making valid comparisons, we used the same machine learning project for each offering. We'll look at factors such as user-friendliness, range of functions, performance, scalability and price. The aim is to help you understand which MLaaS solution is appropriate for your applications, skill sets and budget.

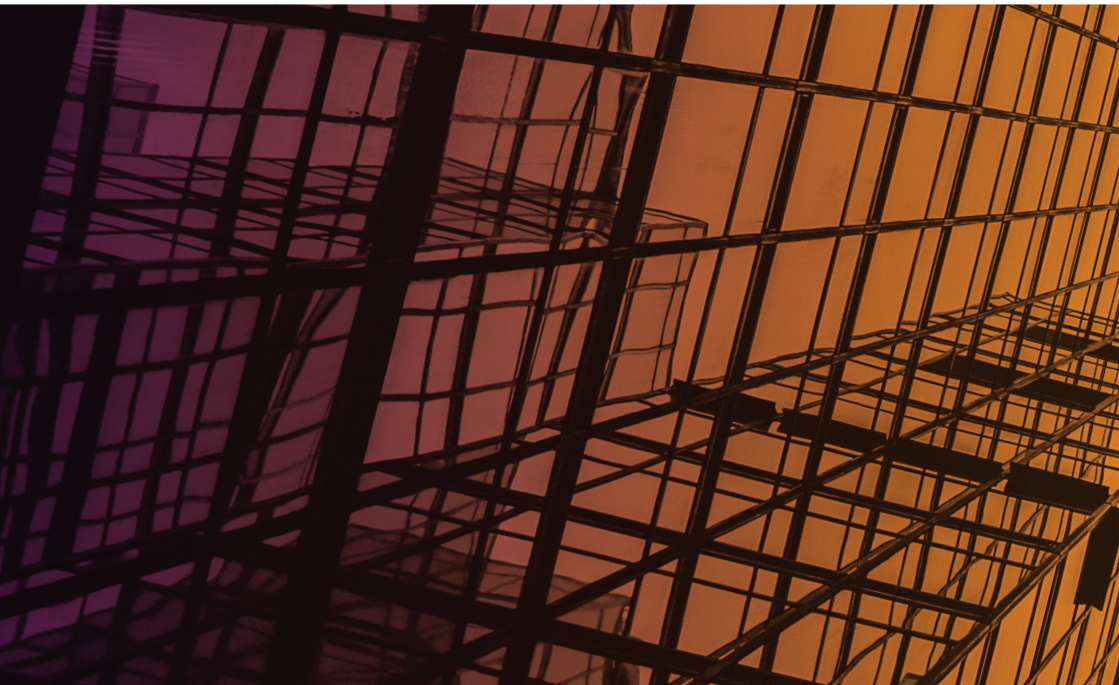
MLAAS VERSUS MACHINE LEARNING FRAMEWORKS



Of course, not everyone will want to go down the MLaaS path. For many, using a machine learning framework is preferable. A large number of free, open-source frameworks (such as TensorFlow, Keras, Caffe, PyTorch and scikit-learn) are available and can be installed either on the users' own hardware (on the premises) or in the cloud. These frameworks have the advantages of enabling users to choose the hardware and software they use and of lowering costs when hardware is used at capacity for extended periods. The disadvantage: considerably more knowledge and effort is necessary to install and operate on-premise solutions.

MLaaS is designed to make applications as simple as possible for the user. Some services use graphical interfaces and wizards, enabling machine learning newcomers to get started quickly. MLaaS can also be beneficial in the development phase, because you only pay for your hardware when it is actually used. However, you will face restrictions on algorithms and parameters you're able to use, which can be a disadvantage. Also, if you have high-performance requirements or large amounts of training data, you'll need a good internet connection.

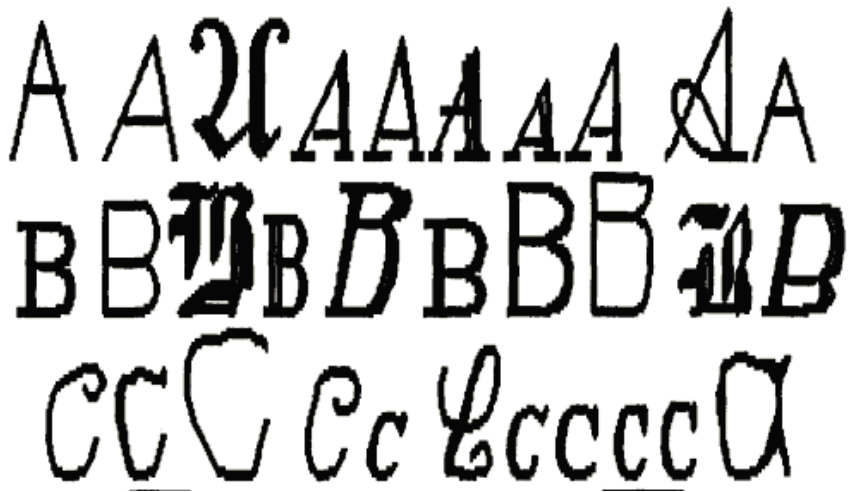
THE OFFERINGS OF AMAZON, GOOGLE AND MICROSOFT



Amazon offers two different machine learning frameworks in the cloud for two different target groups: Amazon Machine Learning and Amazon SageMaker. The former is designed for beginners and allows models to be created and trained without the need to write code. The models can be selected and configured via a console in the usual AWS way. In addition, APIs permit users to connect the model to external systems, allowing them to integrate it into existing IT applications.

Amazon SageMaker offers a much larger range of functions. It enables data scientists to create and host their beloved Jupyter Notebooks on AWS, and many models are preconfigured. Users can also integrate their own models or other machine learning frameworks via a Docker container.

As may be expected, Google's Cloud Machine Learning Engine is the most convenient all-in package for development teams committed to TensorFlow. For this purpose, Google offers a command line interface (CLI) for installation, which can be done in whatever your favored operating systems (Linux, macOS, Windows). This simplifies operation and automation. Good testability in the local environment makes it easy to start a development process, so you have the certainty that projects can be developed sustainably, even if they're likely to run over several years.

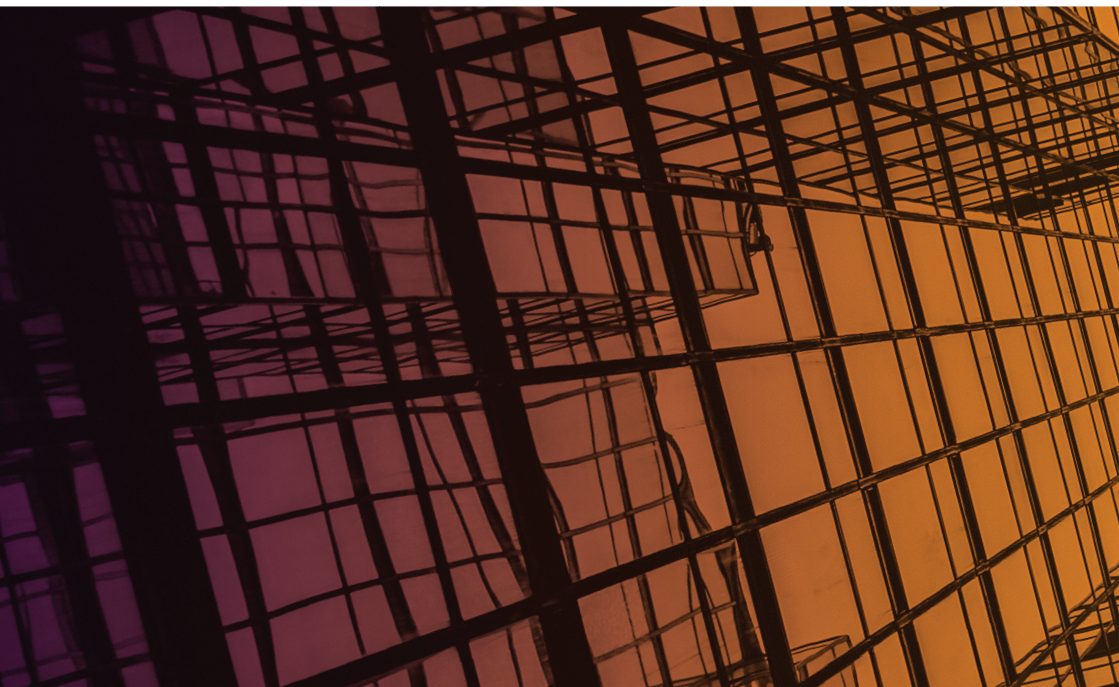


Various letters from the training set (Fig. 1)

For Azure, Microsoft also has two offers directed at two user groups. Azure Machine Learning Studio is designed for beginners or data science departments that haven't had much experience in software development. It provides an easy introduction to the creation and training of models as well as the rapid publishing of an endpoint, which includes documentation for use in existing IT applications of the models.

By contrast, Azure Machine Learning Services are directed towards experienced development departments. A well-guided start and a command line interface allows a user to create projects sustainably in teams. A workbench makes it easier for less experienced users to start using the software. One thing that sets Azure MLS apart is the way it simplifies the creation of Docker images and deployment on Azure IoT edge devices.

A PRACTICAL EXAMPLE



To simplify comparisons, the authors tested the same application with all solutions on offer. They chose a compact, freely available training example that can be implemented and trained; but it's also one that's not completely trivial. The goal was to recognize printed capital letters of the English alphabet in different fonts (see Fig. 1). The database consists of 20,000 patterns of capital letters from the English alphabet. Each pattern of a character was converted from the pixel image into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. The first 16,000 items were used for training and the remaining 4,000 were used for testing the model. See [a] for more details.

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli.

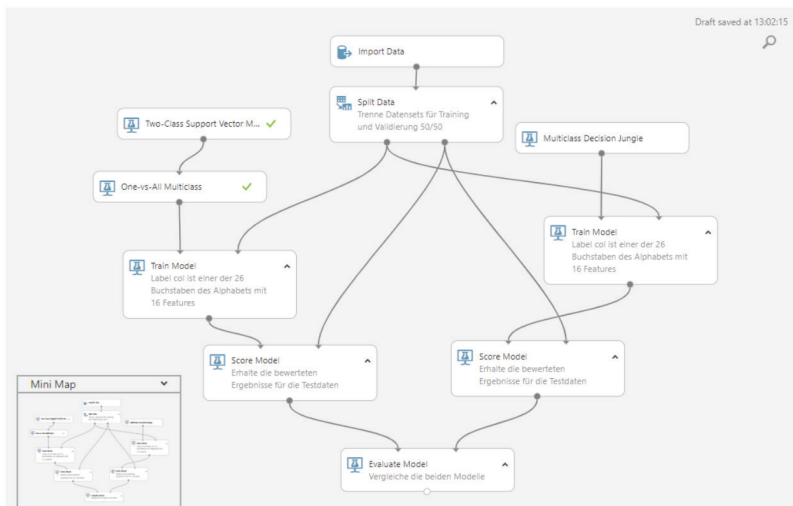
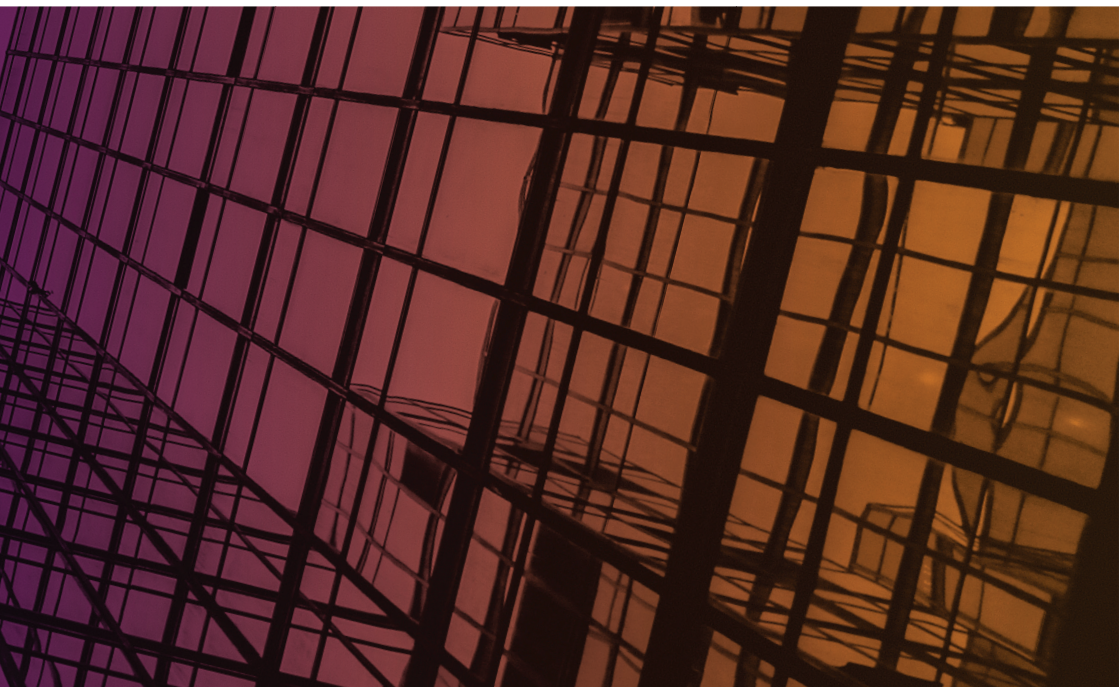


Figure 2: The experimental view in the Azure Machine Learning Studio is adequate for small projects but soon becomes confusing when used for larger ones.

The data from training example is available as a CSV file [a] and can be used for machine learning experiments without the need for further preprocessing. A good description of the data is to be found in the article entitled "Letter Recognition Using Holland-Style Adaptive Classifiers" [b]. This training set is 27 years old,, when it was new, rule-based learning techniques were capable of achieving recognition rates of 80%.

AZURE MACHINE LEARNING STUDIO



The user experience of Microsoft's Azure Machine Learning Studio [c] is reminiscent of other visually driven tools such as those known from the ETL area of Talend or Pentaho/Kettle. So if you're already familiar with those tools, you'll find it easier to make a start — although you can still expect some quirks. One surprise you may find is the location of Azure ML Studio, which isn't on the Azure portal but on a different page altogether.

A project usually follows a simple and intuitive workflow. Users can generate experiments for their training data in Learning Studio's web interface. This enables them to create the workflow via drag and drop using so-called "experiment items". Data input objects can be used to define any desired sources. A web URL is a possible input, and so is a hive table or a data pool within Azure. An OData feed is also easy to connect. ML Studio tries to derive the scheme from the data object autonomously, but it's possible to adjust it to your own requirements.

However, what works well for simple data sets can quickly become confusing when working with large tables or data quantities. So while the visualization tool is a boon initially, it can become a hindrance if project complexity grows. (see Fig. 2).

ML Studio also supports a handful of common data transformation tools, which enable imported data set to be prepared for analysis. These data transformations range from various filters and simple splitting operations for separating training data from test data, to objects and data normalizations similar to MapReduce. If something is missing, it can be copied using language modules (Python and R, at the time of this publication). Data management can only be done via a small text box within the Studio, which is less than ideal.

Finally, users can select their desired algorithm from the machine learning objects. The selection procedure seems very effective here. However, to select the correct data from a training object and then hand it over to classification, for example, it's necessary to know precisely what was transferred from the previous objects. As a result, you run the risk of training on an incorrect field, and sadly this is hard to debug via the web interface. What's more, if you want to experiment with different algorithms, there's no easy way of seeing whether the unaltered parameters are identical or not.

In contrast, the many preconfigured objects and standard tasks such as speech or text recognition are a good thing, and you can simply drag them over to their working area. It's also possible to drag several objects in succession in order to first recognize the language and then assess the objects in the correct one.

Finally, to compare algorithms, you can score and assess the model using other objects. This is easy to do and is also straightforward for simple models. But with larger projects, you rapidly reach a point where the experiment becomes confusing. For data scientists who prefer notebooks, Azure ML Studio allows the same result to be achieved using Jupyter Notebooks.

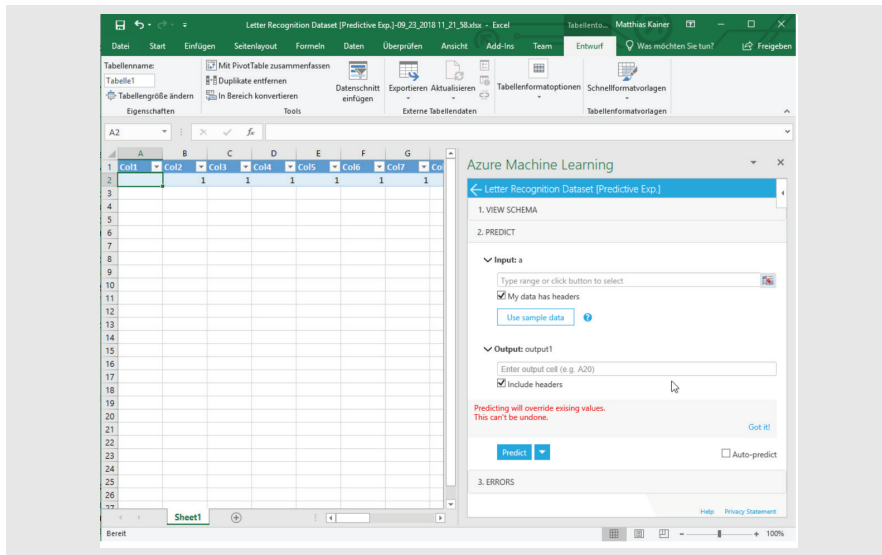


Figure 3: Machine Learning Studio supports the provision of its output to Excel.

Once a model is trained, you can publish its trained functions as a web or batch service. To do this, all you need to do is create a simple dashboard, along with what Microsoft calls "New Web Services Experience." The new functions are still in the preview phase, but they're already superior to the dashboard used previously. In addition to the endpoints, the new Experience allows data scientists to generate documentations in Swagger (a tool for the documentation and automatic generation of source code for REST APIs) and even examples of code in R, Python and C#. Development teams can then share this information. As you may expect from Microsoft, the data generated by the model can be put to immediate use in Excel in order to run forecasts etc. And this can be achieved with just a few clicks during initial setup (see Fig. 3).

An API key is needed to authenticate a user itself with regard to the API. However, these keys are limited to two per service. To audit accesses more precisely and revoke them if necessary, therefore, users should consider other methods of publishing in the corporate environment, for example with one service endpoint per team. A dashboard for monitoring the endpoint rounds off the range of functions provided. The versioning of newly trained models at the endpoint allows users to make the teams of developers who develop with regard to the endpoints aware of new versions of the models.

On the whole, it's extremely easy to start working with Azure ML Studio. This is true for handling as well as costs. The latter are charged per session and per hour of experimentation.

For web services, there are various different packages consisting of defined transactions and computing hours. Small projects can even be realized completely free of charge.

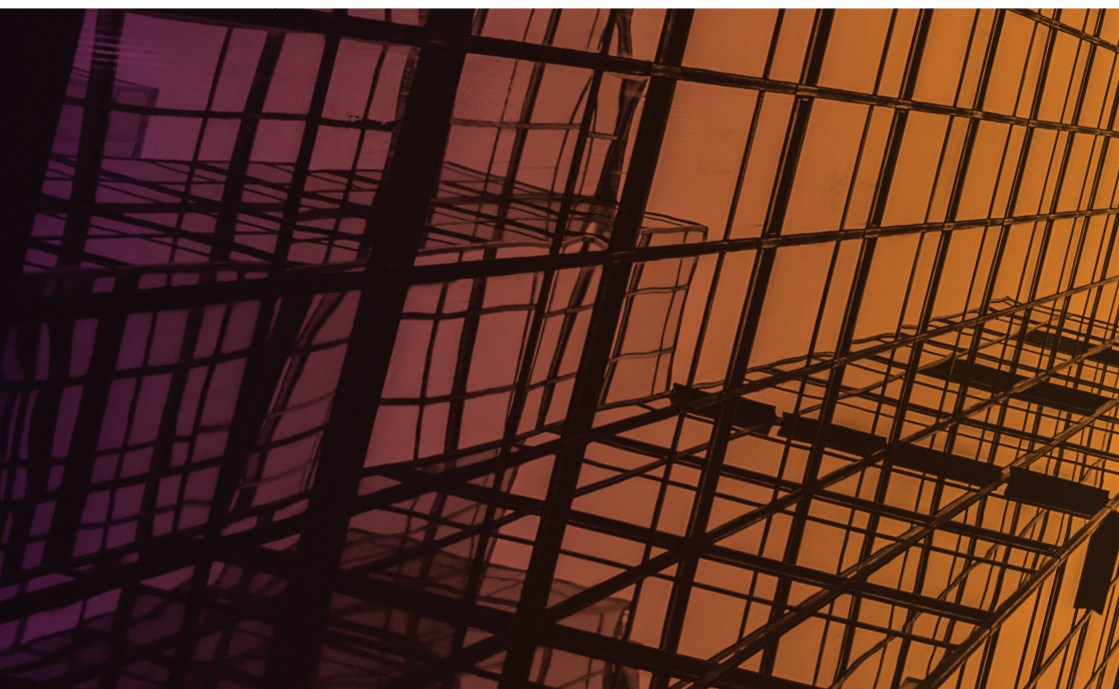
Trained models can be executed Parallelization options for the rapid and efficient realization of the models enable the trained models to be executed efficiently in production. Objects can be executed in parallel in the models, meaning that as soon as the data has been loaded, the various parts of the process can be executed simultaneously.

Thanks to the good preconfiguration, we achieved a recognition rate of 80–85% for the character recognition test, using the default values for model and hyperparameters. Depending on the algorithm selected, CPU times range between 2–5 minutes. There are no methods for the further optimization of the hyperparameters. This makes it difficult to achieve an additional optimization of the recognition rate.

Thanks to the good publishing options for the web services and the ease of setup, Azure Machine Learning Studio is ideal for those new to machine learning scenarios. For business-critical applications, however, users should turn to one of the other applications, which provide better support for long-term maintainability and further development of models

One other note of caution: at the time of writing, Azure Machine Learning Studio is only hosted in US data centers. Any organization that has concerns around data protection and General Data Protection Regulation (GDPR) compliance should proceed with caution.

AMAZON MACHINE LEARNING



Amazon Machine Learning [d] provides an environment that enables beginners to be successful quickly. Even users with no programming experience can load training data, train ML models and ultimately use an API endpoint to provide the functions learned (such as classification) to other programs. Amazon Machine Learning is restricted to algorithms for binary classification, multi-class classification and regression. In practice, a wide range of applications can be covered by the three approaches.

To start, you tell Amazon Machine Learning where the training and test data are located by creating so-called data sources. The data can be stored in CSV files in AWS S3 or within the Amazon Redshift database. If the data is in an AWS S3 bucket, the relevant rights must be established on the bucket according to the machine learning algorithms. This happens by means of a "bucket policy" that users can establish in S3 under "Permissions". You'll need to write a short JSON script for this purpose. For non-programmers, Amazon's documentation provides an example script, and you can adapt it to the current bucket and path names

The selection of ML algorithms is limited. One each is available for binary classification, multi-class classification and regression, respectively. Amazon Machine Learning selects the algorithm automatically with the help of the training data. When selecting the parameters, you can either take over the default values suggested or select ones of your choosing. After selecting the default values, training can begin immediately. The preset values are usually effective and well selected. With them, we achieved a very good recognition rate for the example of letter recognition.

S3 data access

Tell Amazon ML how to access your data and give it permission to access it.

S3 location *

Enter the path to a single file or folder in Amazon S3. You need to grant Amazon ML permission to read this data. [Learn more.](#)

If you already have a schema for this data, provide it in a file at s3://<path-of-input-data>.schema. If you don't have a schema, Amazon ML will help you create one on the next page. ⓘ

Datasource name

The validation is successful. To go to the next step, choose Continue

Datasource name training data

Data location s3://ix-article/datafiles/letter-recognition-all.csv

Data format CSV

Schema source Auto generated

Number of files 1

Total size 695.9 KB

* Required

Figure 4: Amazon ML checks the data source (in an S3 bucket here) and automatically generates a data scheme.

The parameters can be set by hand, but they're restricted to the essential functions. The training data can also be transformed using so-called "recipes" (JSON scripts). Simple functions such as normalization and lower-case transformations are available, and so are statistical functions such as N-gram transformations. In addition, users can set the maximum number of parameters, the number of training runs, the training and test data, the random mixing of the training data and the regularization type and value. Once all parameters have been determined, the model is trained and then evaluated.

CLOUD WORKSHOP

The screenshot shows the Amazon S3 console interface. At the top, the breadcrumb navigation indicates 'Amazon S3 > ix-article'. Below this is a tabbed interface with three tabs: 'Overview', 'Properties', and 'Permissions'. The 'Permissions' tab is currently selected. Underneath the tabs are three buttons: 'Access Control List', 'Bucket Policy', and 'CORS configuration'. The 'Bucket Policy' button is highlighted. Below the buttons, the text 'Bucket policy editor' is followed by the ARN 'arn:aws:s3:::ix-article'. A note states: 'Type to add a new policy or edit an existing policy in the text area below.' To the right of this text are three buttons: 'Delete', 'Cancel', and 'Save'. The main area is a text editor containing a JSON policy document. The policy has a version of '2008-10-17' and two statements. The first statement allows the 'machinelearning.amazonaws.com' principal to perform the 's3:GetObject' action on the resource 'arn:aws:s3:::ix-article/datafiles/*'. The second statement allows the same principal to perform the 's3:ListBucket' action on the resource 'arn:aws:s3:::ix-article', with a condition that the 's3:prefix' must be 'datafiles/*'.

```
1 {
2   "Version": "2008-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Principal": {
7         "Service": "machinelearning.amazonaws.com"
8       },
9       "Action": "s3:GetObject",
10      "Resource": "arn:aws:s3:::ix-article/datafiles/*"
11    },
12    {
13      "Effect": "Allow",
14      "Principal": {
15        "Service": "machinelearning.amazonaws.com"
16      },
17      "Action": "s3:ListBucket",
18      "Resource": "arn:aws:s3:::ix-article",
19      "Condition": {
20        "StringLike": {
21          "s3:prefix": "datafiles/*"
22        }
23      }
24    }
25  ]
26 }
```

Figure 5: After loading, users have to specify the target for the learning algorithm. Amazon uses the target to recognize which learning algorithm has to be used (binary classification, multi-class classification or regression) (Fig. 5).

In our experiments, we achieved a recognition rate of 84% on the evaluation data (20 percent of the data set). Changing the model parameters didn't improve the recognition rate — which suggests that the default values for the algorithms are pretty useful.

We'd initially allocated one minute of CPU time for training but found we need considerably longer (in the range of minutes). One possible explanation is that the hardware instances have to be allocated and instantiated in the AWS cloud and the data quantities have to be loaded from the S3 bucket and then transformed.

Several options are available for using the model trained:

- If you want to enter individual patterns into the model, there's an input mask that enables manual entry of input pattern values. Entire input patterns can be conveniently entered using copy and paste.
- If you want to batch-process a file with input patterns, this file can be read in via S3. The costs required for processing the file are then displayed. The algorithm is output via S3 too.

To address the model using other IT systems, an endpoint can be generated. The resulting costs are displayed in a straightforward way. Prediction requests are transferred to the endpoint using requests in JSON format. The endpoint also answers the request in JSON format with the relevant prediction.

1. Input data 2. ML model settings 3. Recipe **4. Advanced settings** 5. Evaluation 6. Review

Advanced settings

You can use the advanced settings to fine-tune the accuracy of your ML model. [Learn more.](#)

Maximum ML model Size MB (2000 MB maximum)
The total size, in MB, of patterns that Amazon ML creates during the training of an ML model.

Maximum number of data passes 100 maximum
The number of passes Amazon ML will make over your data to discover patterns.

Shuffle type for training data ☐ None ☒ Auto
If you haven't already shuffled your data, choose Auto. ⓘ

Regularization type ☐ None ☐ L1 ☒ L2
You can control the amount of regularization by using the regularization parameter. ⓘ

Regularization amount

[Cancel](#)

[Previous](#)

[Continue](#)

Figure 6: The hyperparameters of the learning algorithm can be set under "Advanced Settings".

Summing up, we're confident that even non-programmers can successfully use Amazon Machine Learning for predictions (either category or value predictions). It only took a few hours to set up and train the models for the purposes of the example. The Amazon documentation is exemplary, and there are comprehensive links to the relevant help guides. The recognition rates were good, even without laborious optimization of the hyperparameters. The performance was good too, even though this was only a small example application and the users had no say on the hardware used.

ML model settings

You can use the automatically suggested ML model settings, or you can choose to customize.

ML model type MULTICLASS ⓘ

ML model target _Target_

ML model name
(Optional) ML model: Letter-recognition-all.csv

**Select training and
evaluation settings**

Recipes and training parameters control the ML model training process. You can select these settings for your ML model or use the defaults provided by Amazon ML. In either case, you can choose to have Amazon ML reserve a portion of the input data for evaluation. [Learn more](#).

☒ **Default (Recommended)**

- Generate a default recipe
- Use default training parameters
- Set aside 30% of your training data to evaluate the training
- Split the evaluation data sequentially ⓘ

☐ **Custom**

- Modify the recipe Amazon ML generates
- Modify training parameters
- Randomly or sequentially split your evaluation data ⓘ

Evaluation Name Evaluation: ML model: Letter-recognition-all.csv

[Cancel](#)

[Previous](#)

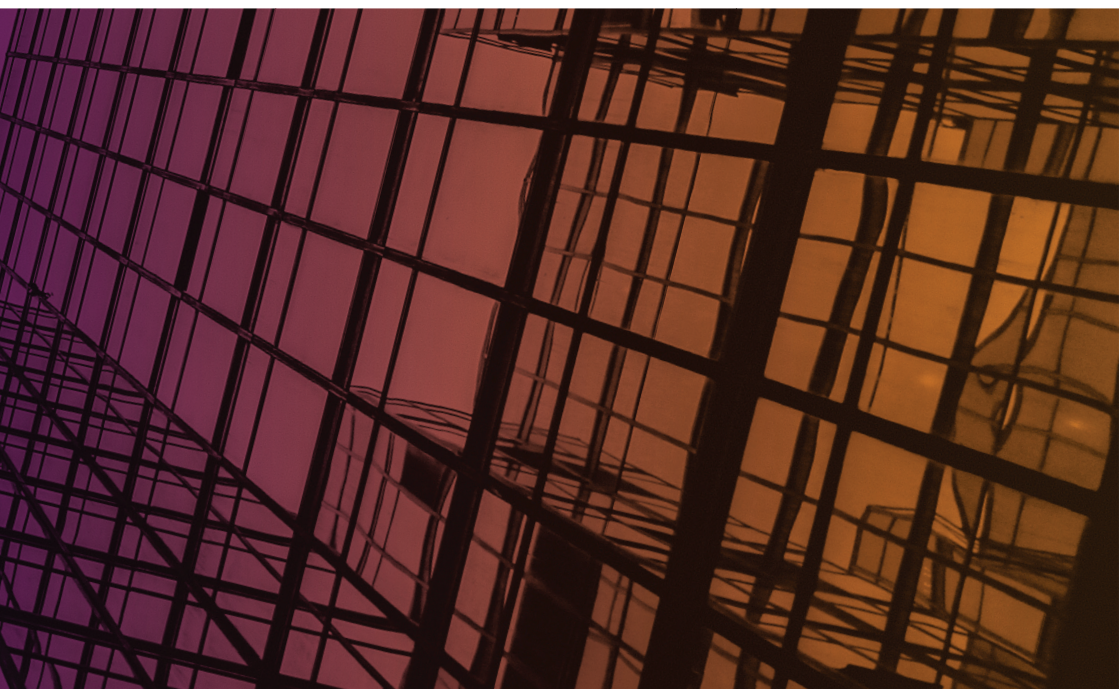
[Review](#)

Figure 7: When selecting the parameters of the ML model, users can choose the default values set by Amazon or set values of their own .

Available API endpoints allow Amazon Machine Learning to be used in a production IT environment, although perhaps only to a limited extent. The output of the endpoints is restricted (to 200 requests per second in our case) and retraining of the model results in a new model with new endpoints, which must then be manually adjusted in the calling systems. It cannot be used to realize contemporary development approaches with continuous delivery and automatic pipelines

However, Amazon Machine Learning is a good solution allowing a quick start and initial experiments as well as relatively small productive applications.

AMAZON SAGEMAKER



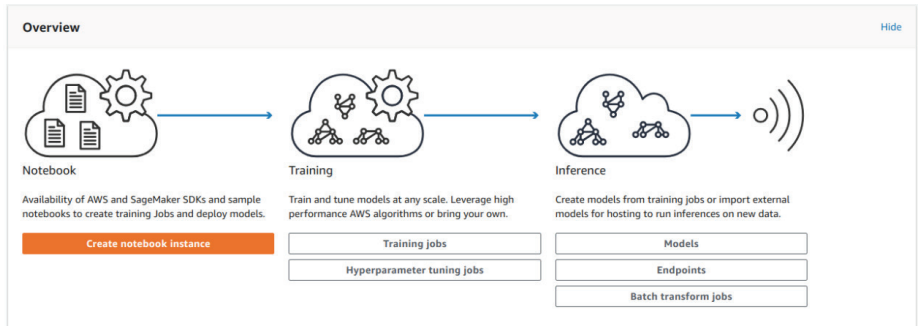


Figure 8: The SageMaker dashboard with links to the individual cockpits

Amazon SageMaker [e] focuses on the Jupyter Notebooks that data scientists love so much. All machine learning models are programmed in Python here. Even though SageMaker comes with many completed functions and libraries, users need detailed knowledge of Python to use SageMaker.

Notebook instance settings

Notebook instance name

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

Search

- ml.t2.medium
- ml.t2.large
- ml.t2.xlarge
- ml.t2.2xlarge
- ml.m4.xlarge
- ml.m4.2xlarge
- ml.m4.4xlarge
- ml.m4.10xlarge
- ml.m4.16xlarge

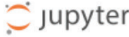
Accelerated computing

- ml.p2.xlarge
- ml.p2.8xlarge
- ml.p2.16xlarge
- ml.p3.2xlarge
- ml.p3.8xlarge
- ml.p3.16xlarge

Figure 9: When users install the Jupyter Notebook, they have to specify the cloud hardware on which the notebook is to run.

In SageMaker, you start by installing a Jupyter Notebook instance — there are wizards available that make this easy. Next, you have to specify the server hardware that will run the notebook in the AWS cloud (see Fig. 9). The important thing is for the notebook to have access to S3 — this allows it to read training data and save results and parameters. For this to happen, users need the relevant rights (IAM.FullAccess). Then a machine learning model must be selected and implemented. In contrast to Amazon Machine Learning, SageMaker can theoretically provide an unlimited number of models. Options include:

- Use of the algorithms predefined by SageMaker;
- Use of the algorithms provided by Apache Spark;
- Creation of Python code that uses TensorFlow or Apache MXNet machine learning frameworks;
- Creation of your own algorithms, which are then provided to SageMaker using a docker container.



Quit

Files
Running
Clusters
SageMaker Examples
Conda

A collection of Amazon SageMaker sample notebooks. ↻

Introduction to Amazon Algorithms	▼
DeepAR-Electricity.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
Image-classification-fultraining.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
Image-classification-1st-format.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
Image-classification-transfer-learning.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
LDA-Introduction.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
SageMaker-Seq2Seq-Translation-English-German.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
blazingtext_hosting_pretrained_fasttext.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
blazingtext_text_classification_dbpedia.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
blazingtext_word2vec_subwords_text8.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
blazingtext_word2vec_text8.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
deepar_synthetic.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
factorization_machines_mnist.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
k_nearest_neighbors_covtype.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
linear_learner_mnist.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
ntm_synthetic.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
object_detection_image_json_format.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
object_detection_recordio_format.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
pca_mnist.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
random_cut_forest.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
xgboost_abalone.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>
xgboost_mnist.ipynb	<div style="display: inline-block; border: 1px solid black; padding: 2px 5px;">Preview</div> <div style="display: inline-block; background-color: #ff7f0e; color: white; padding: 2px 5px; margin-left: 5px;">Use</div>

Figure 10: Python example programs for the different learning algorithms

SageMaker has a large number of well-optimized algorithms, covering a wide range of tasks for machine learning applications, available. For example, there are numerous algorithms for classification, regression, image recognition and text recognition. In the notebook, these models are called in Python using an API. We decided in favor of the predefined algorithm XGBoost (eXtreme Gradient Boosting), a supervised learning algorithm based off the classical decision tree model.

Compared to Amazon Machine Learning, the SageMaker documentation is somewhat lacking. For example, XGBoost is not explained; instead, a link to a scientific paper is given. The hyperparameters for calling the algorithm are described, but there's no mention of how to use it and how to make the call itself. For non-English speakers, localization options may lead to more confusion. (For example, in German, "training" is always translated as "Schulung," which is incorrect). Instead, basic examples are presented in the notebook in Python. From these examples, the calling structure of the algorithms soon becomes clear.

Training data must be provided in an S3 bucket. As SageMaker runs on servers of the region eu-west-1 (Ireland) in Europe, the S3 buckets which will be accessed by SageMaker will also need to be created in that region, otherwise you'll get an error in the training process. You can specify the region of your S3 buckets during the creation process.

You'll also need to specify the hardware for the training instance in the training algorithm's configuration parameters. We selected the following configuration (derived from the examples):

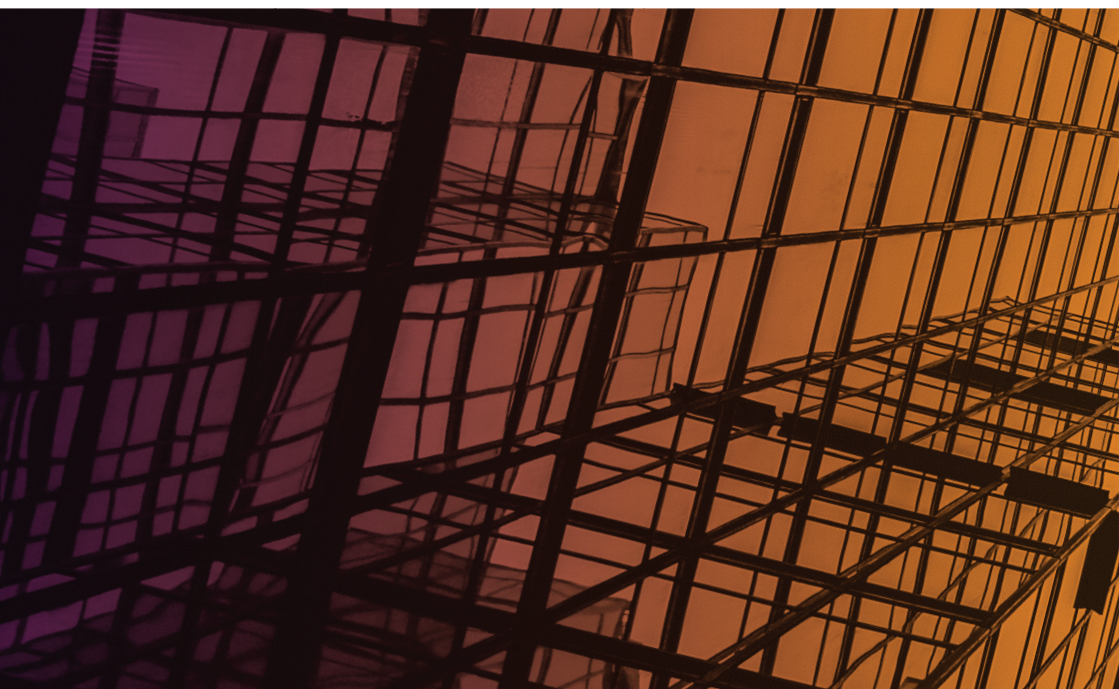
```
"ResourceConfig": {  
  "InstanceCount": 1,  
  "InstanceType":  
    "ml.m4.10xlarge",  
  "VolumeSizeInGB": 5  
}
```

The CPU time for training on this hardware was 40 milliseconds.

To test the model trained, you have to install an endpoint for the model, which can be done from the Python code in your notebook. Without tuning the main parameters, we immediately achieved a recognition rate of 84% on a test set — something that speaks favorably for the quality of XGBoost and the preconfiguration of the hyperparameters. The total costs for training the model, installing the endpoint and testing at AWS amounted to \$0.56 USD.

As all important parameters of a machine learning algorithm are concentrated in a notebook and notebooks are easy to import and export, development, testing and operation as well as version management for the different development versions are easy to organize and automate in SageMaker.

MICROSOFT AZURE MACHINE LEARNING SERVICE



The Azure ML Services [f] are directed towards advanced users who operate the algorithms they develop in the cloud and want to make them available there too. A project is set up using a CLI that you install on your local computer. Python is required here. Alternatively, you can install the Azure Workbench (for Linux, macOS and Windows), and it carries out Python setup. It was rather difficult to get the Azure CLI to run on a Mac; the installer seems to have had a problem with the virtualenv tool. In this case, a functioning environment couldn't be executed until the workbench was installed.

Once setup has been mastered, you'll want to create a project. Microsoft provides a number of options here. Creation can be performed either in the usual Azure portal using an assistant, the workbench or CLI. To ensure reproducibility, CLI is recommended here. If a user hasn't had much experience with Azure, it can be helpful to use the UI at first.

After creating the project, you have to generate a model that's the result of an experiment. In our example, a simple scikit-learn script with an Azure module handled this. Next, the project is published using an 'az' command. Training can be carried out either locally or in the cloud via 'az ml experiment submit'. You can then provide a model that includes all resources by creating a manifesto. With a model and a manifesto, you get a Docker container. This sounds simple, but you'll need to be aware that you can't generate a service without a resource group, and you can't generate a service (with which they provide the model as a Docker container using the manifesto) without an environment. Once this is mastered, everything is simple and can easily be automated using CLI.

A nice feature is that everything is "dockerized" and you can easily execute scripts on your own directly in Docker containers via the az-CLI. The host for Dockers requires ssh access. Once you've configured it, you can execute the build using the command 'az ml experiment prepare -c my_docker_host'.

Microsoft's idea is to operate these images in any desired environments either locally, in the cloud or on IoT devices. The company offers four deployments for this. The simplest option is to provide the model directly as an Azure container instance. This only takes a few minutes.

If you already have an existing Kubernetes instance in Azure, you're able to operate the model in it as a service. You can also carry out edge deployment for IoT use cases. For this purpose, the model is run on an edge device (a server available on the company premises or locally). This use scenario is just as suitable for real-time applications as it is for environments in which there's no connection to the internet. Finally, it's possible to realize the project as a Field Programmable Gate Array (FPGA), an option for the hardware-supported execution of the models with a low latency.

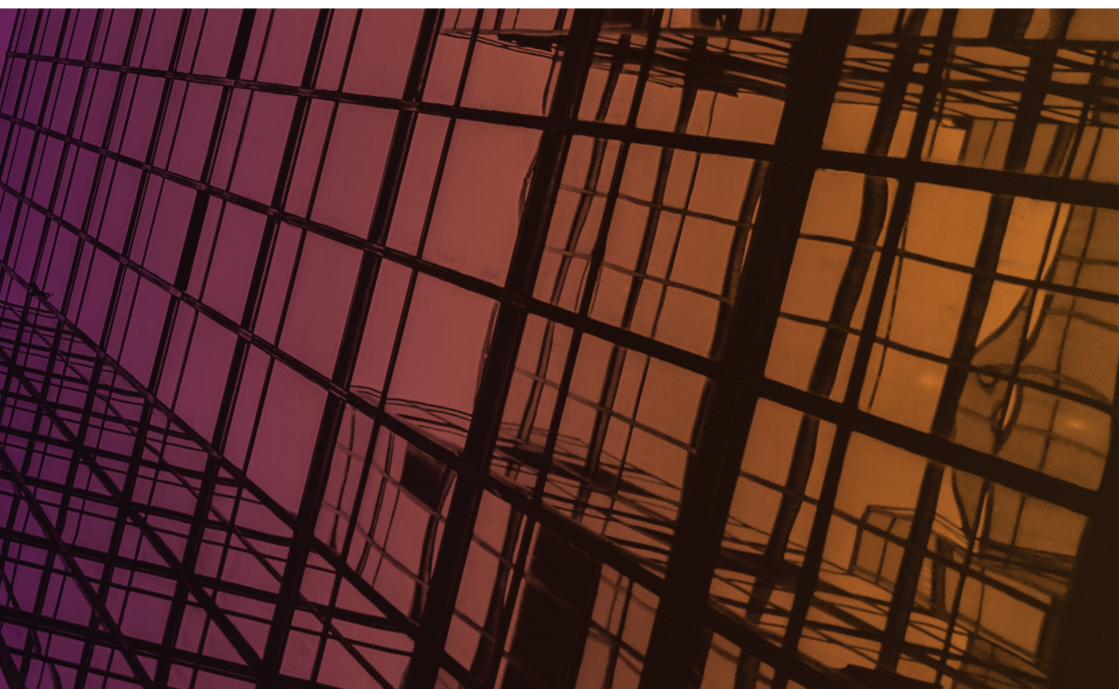
The UI is intuitive and the translation of the documentation is very good. We did, however, repeatedly encounter small problems with the local Python environment that ranged from difficulties during setup to problems and incompatibilities with local modules, for example with a previously installed miniconda. For inexperienced users, the workbench is a good start. Also, the web UI is a help in many places and offers an easy start, but not all tasks can be performed there.

A model can also be generated using TensorFlow (it can be done with any available Python library, in fact). The documentation doesn't say that this is possible and the above problems with Python modules cause a few problems to occur at times. In virtually all documentation, Microsoft only uses scikit-learn, which is included in the standard version of the workbench. Generating the test project for this article using scikit-learn was ultimately easier than successfully realizing a prepared TensorFlow project. The modules data scientists can easily use in the context of a project are helpful here. This means that you don't need to create a known task, such as image recognition, but can directly use a web service instead.

The Azure ML Services are a robust offering with weaknesses in particular in the extension of the standard az-cli with the ml tools. It seems that not every Python environment is compatible here, but further investigation would be needed to validate that. The default deployment in Docker containers is an exciting approach and one we'd recommend. A positive feature is that you can avoid vendor lock-in quite well if you install your Python scripts in a modular way.

The resources for the project can be saved in the data center of your choice, including ones in the EU. This ensures that data processing will be carried out within the EU — which is helpful for those with regulatory concerns. The same is true for the services provided, and users can select the region in which the service is to be provided. The speed of execution was very high in the experiments. With scikit learn's Decision Tree Classification (CART), recognition rates of 85% were achieved in about two minutes of CPU time without much experimentation; the same is true for "Machine Vision API" provided by Azure. The high degree of automation achieved here makes it worth exploring. For companies that are already using Azure or are considering building up a cloud for their machine learning activities, Azure ML Services are a very good choice.

GOOGLE CLOUD MACHINE LEARNING ENGINE



Although the Google Cloud Machine Learning Engine [g] supports a number of languages and frameworks, the focus of this environment is clearly on one framework: TensorFlow. Here Google offers a home for its framework as well as resources such as tutorials and templates, and it does so in a precise and well-designed way. When you start a new project, therefore, it makes sense to use the cloudml template [h] that Google provides as a template under an Apache license (see Fig. 11). This offers a number of starting points ranging from shell scripts (for Linux, macOS and Linux Subsystem for Windows) for build and deployment, to a relatively easily configurable TensorFlow project. It is possible to configure the data format using metadata and determine whether to choose classification or regression as an option. Thanks to skillful standardization, the input is automatically parsed and processed, the features are created and the model is trained. Further adjustments to suit your requirements are easy to perform.

CLOUD WORKSHOP

```
# *****
# YOU NEED TO MODIFY THE FOLLOWING METADATA TO ADAPT THE TRAINER TEMPLATE TO YOUR DATA
# *****

# Task type can be either 'classification', 'regression', or 'custom'
# This is based on the target feature in the dataset, and whether you use a canned or a custom estimator
TASK_TYPE = '' # classification | regression | custom

# A List of all the columns (header) present in the input data file(s) in order to parse it.
# Note that, not all the columns present here will be input features to your model.
HEADER = []

# List of the default values of all the columns present in the input data.
# This helps decoding the data types of the columns.
HEADER_DEFAULTS = []

# List of the feature names of type int or float.
INPUT_NUMERIC_FEATURE_NAMES = []

# Numeric features constructed, if any, in process_features function in input.py module,
# as part of reading data.
CONSTRUCTED_NUMERIC_FEATURE_NAMES = []

# Dictionary of feature names with int values, but to be treated as categorical features.
# In the dictionary, the key is the feature name, and the value is the num_buckets (count of distinct values).
INPUT_CATEGORICAL_FEATURE_NAMES_WITH_IDENTITY = {}

# Categorical features with identity constructed. if any. in process_features function in input.py module.
```

Figure 11: The Google Cloud Platform template offers a number of options for editing the metadata of the training data.

If you're using this template, you'll need some previous knowledge of TensorFlow, otherwise, it will be difficult to estimate the impact changes are likely to have. Bash scripts are provided so that you can work with the model presented. That said, you'll still need to make adjustments — so look at the comments in the files, which point to the relevant places. In addition, you'll have to install the "gcloud command line interface tool" and preconfigure your own Python environment, preferably using a virtualenv.

With that done, you can use the command `./local-train.sh` to train your model locally and put it into a GCP bucket using `./cloud111/-deploy-model.sh`. The command `./cloudml-submittrain-job.sh` enables you to run the training directly in the cloud too. The model is then stored in a bucket. Be aware: a saved model mustn't exceed 250MB. If the model is too large, you can try to quantify it. If this line of action is exhausted, you've reached the limits.

Once the model's in a bucket, an endpoint can be exposed, which can then be integrated into development projects and used there with the help of the "google api libraries". For this to happen, generate a model version from the model trained. This can be done in the console, but also using a CLI and REST endpoint. The model resource created can then be exposed as an HTTP and batch prediction. Simply put, the difference is that HTTP requests work synchronously while batch requests work asynchronously. This means that batch requests can be scaled better, whereas HTTP requests allow a result to be displayed directly, for example in an app.

Google Cloud Platform

ML Engine

Jobs

Models

Create version

To create a new version of your model, make necessary adjustments to your saved model file before exporting and store your exported model in Cloud Storage. [Learn more](#)

Name

character_recognition

Name cannot be changed, is case sensitive, must start with a letter, and may only contain letters, numbers, and underscores. 21 / 128

Description

Python version

3.5

Select the Python version you used to train the model

Model version with Python 3.0 and beyond can't be used for batch prediction jobs. Online prediction still works.

Framework

TensorFlow

Framework version

1.9

ML runtime version

1.9

Machine type

Single core CPU

Model URI *

gs://mlaas_default_bucket

BROWSE

Enter the Google Cloud Storage path where you uploaded your model. If your model uses scikit-learn or XGBoost, check out the [naming requirements](#) and enter the path of the Cloud Storage bucket.

Online prediction deployment

Scaling

Auto scaling

Minimum number of nodes (optional)

Keeping a minimum number of nodes running all the time will avoid dropping requests due to nodes initialization after the service has scaled down. This setting can increase cost, as you pay for the nodes even when no predictions are served.

SAVE

CLEAR

CANCEL

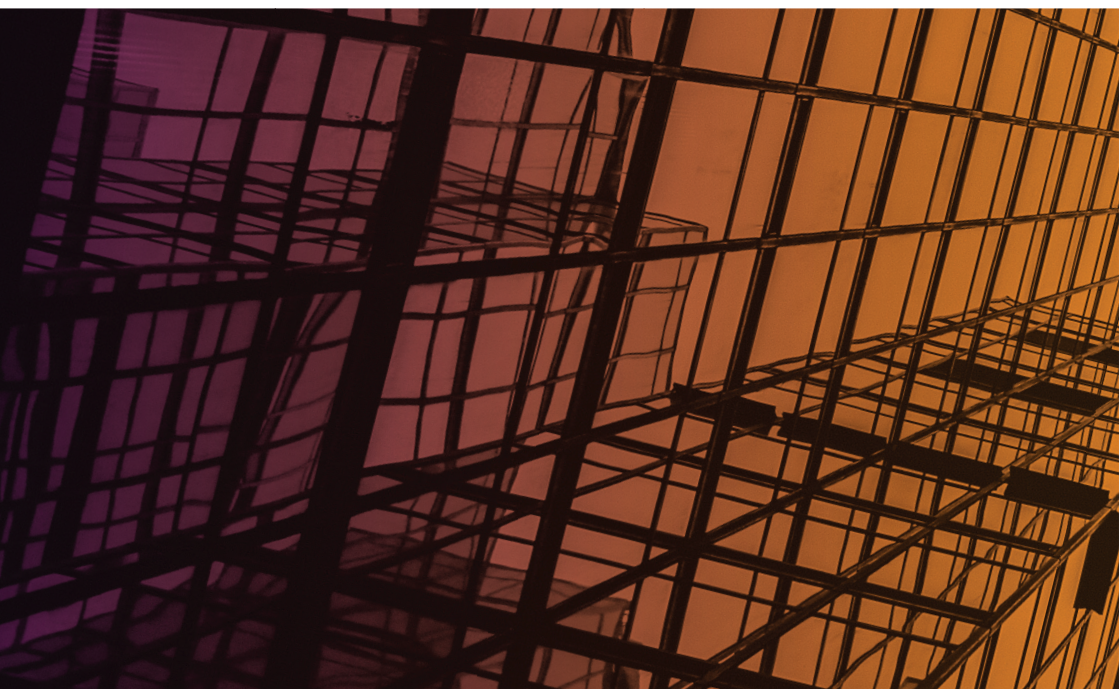
Figure 12: A model can be created either using the web console (as shown here) or the CLI or by means of an endpoint REST.

The Google Cloud Machine Learning Engine is an interesting proposition for developers and data scientists with a background in development. Starting is made easier by a cloud-ml template for TensorFlow, and this template generally offers a good, expandable ML platform, but it takes a little getting used to. The many tutorials and the good documentation are as convincing as the automation options provided by the intuitive CLI (see Fig. 12). In addition, Google provides a collection of endpoints for typical tasks (language, image recognition).

While you can determine the storage and server location for products on their own, in this case they agree to their data being executed and stored in US data centers.

We found execution speed was high. After a few configuration experiments with the cloud template, we achieved constant recognition rates of over 80% in a runtime of less than two minutes. The Google Cloud environment generally has a good reputation with developers and displays an intuitive concept here too. When working with the platform, you'll soon feel at home, and achieve reproducible results quickly and easily. Teams with a large proportion of developers will feel most at home with the platform, whereas for teams without a background in development, it will take a little longer to get used to. That said, some experienced operation employees may miss the adjustment options provided by AWS, such as the rolling of the cloud platform, which aren't available here.

CONCLUSION



Microsoft's Azure Machine Learning Studio is the ideal solution for companies who aren't very skillful in terms of development and operation but still want to create and share models at a reasonable price. The platform is a convincing proposition due to its simplicity – it really allows users to create models with just a few clicks and then use them. But it's ill-suited to cases where the models will be used in the long term or where the projects are likely to grow significantly. This is where the strongly UI-driven application reaches its limits.

The same is true for Amazon Machine Learning. Classification and regression tasks can be realized quickly and easily without prior knowledge here. The endpoints allow the machine learning algorithms to be integrated into the IT environment. In the case of relatively large projects, users quickly reach the limits set by the graphical interface and the lack of flexibility.

This is where Amazon SageMaker shows just how good it is: A large number of efficient machine learning algorithms are available for numerous use cases here. If this isn't enough, you can use machine learning algorithms via the Python interfaces of TensorFlow, Apache MXNet and Apache Spark. You can also develop algorithms on your own in Python. The use of SageMaker requires a good knowledge of Python. Due to the somewhat weak documentation, a few obstacles must be mastered at the beginning.

The Azure Machine Learning Services are directed towards companies that use experienced developers to create ML projects. In particular, the deployment options are convincing here.

Azure IoT Edge is one of the most exciting solutions available to companies wishing to work with offline devices or on a real-time basis. The simple deployment as a Kubernetes service is convincing too.

Google's Cloud Machine Learning Engine is the most convenient application; it enables you to work with TensorFlow without having to worry about infrastructure. With the help of the `cloudml-template` in particular, it's easy to get started. It allows teams of developers with no experience to start a project extremely rapidly and provide a model.

Christoph Windheuser is Global Head of Artificial Intelligence at ThoughtWorks Inc.

Matthias Kainer is Developer und Lead Consultant at ThoughtWorks Germany. He is an expert for machine learning and cloud computing
All links: ix.de/ix1815108.

Online sources

[a] David J. Slate; Letter Image Recognition Data
<https://archive.ics.uci.edu/ml/datasets/Letter+Recognition>

[b] Peter W. Frey, David J. Slate; Letter Recognition Using Holland-Style Adaptive Classifiers
<http://www.cs.uu.nl/docs/vakken/mpr/Frey-Slate.pdf>

[c] Machine Learning Studio
docs.microsoft.com/en-us/azure/machine-learning/studio/

[d] Amazon Machine Learning
aws.amazon.com/de/aml/

[e] Amazon SageMaker docs.aws.amazon.com/sagemaker/

[f] Azure ML Services
azure.microsoft.com/en-us/services/machine-learning-services/

[g] <https://cloud.google.com/ml-engine/docs/>

[h] cloudml-template <https://github.com/GoogleCloudPlatform/cloudml-samples/tree/master/cloudml-template>