CATAPULT Digital | PAPIs.io

# Machine Learning Platforms

Using, extending and creating platforms to accelerate machine learning efforts and generate growth

Research report 2019

# Report contents

**Choosing machine learning platforms and filling the gaps**

# Introduction

Machine learning (ML) platforms reduce the time and cost of integrating ML into software applications, for organisations of all levels of ML maturity. ML platforms can be ready-made or custom-built.

This report provides a survey of ready-made platforms, grouped into different types and presented from starting with those that are aimed at general users, all the way to platforms that enable machine learning experimentation and optimisation. It shows how startups and larger organisations can benefit from utilising these platforms, and what can be their limitations.

This report can also inspire the design of custom-built platforms, which can serve two different purposes:

1. Internal usage to accelerate ML efforts

2. Selling access to generate economic growth

**ABOUT MACHINE LEARNING PLATFORMS**

Gartner defines ML platforms as "software products that data scientists use to help them develop and deploy their own data science and machine-learning solutions"[1]. For the purpose of this report this definition is extended to include products that can be used by software developers too. This report focuses on the integration of ML into software, for the purpose of creating machine intelligence, rather than on building scientific knowledge from data.

ML platforms can come in different formats, as they can be based on open source or commercial software, and can run on-premises or in the cloud. Some of these platforms are made available "as a service", which Wikipedia[2] defines as "a category of cloud computing services[3] that provides a platform[4] allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app". This model has the following advantages:

- There are no set-up costs (which is ideal for startups and for pilot projects)
- Users pay for the functionality they are interested in (not for hosting), according to usage
- There's nothing to install
- Users do not need to worry about where the application program interface (API) and the models are running, for example, which processor to use (CPU/GPU), how to scale up or down as the volume of prediction requests evolves

"We believe most people will begin their AI journeys using cloud services rather than creating and managing million-node neural networks in their own data centres. Even if you eventually outgrow a publicly available AI cloud service, your experience using it will help inform the design and scaling of your own AI infrastructure."

**Andreessen Horowitz** — AI Playbook[5]

# Introduction

## WHO IS THE REPORT WRITTEN FOR?

This paper has been written predominantly for users and potential users of ML platforms, including:

– Software developers and product managers, who will be adding machine intelligence superpowers to their applications

– Data scientists, who will be shipping ML models in production, and extending toolkits to run ML experiments

– Data and ML engineers, extending these platforms or creating new ones

– Domain experts, exploring ideas of new ML use cases, assessing their feasibility, and building prototypical solutions to business problems

This report will also be of interest to managers, analysts, and consultants who have heard that machine learning can help improve products, software applications, and processes, who are trying to figure out the right way for their organisation to make the most of ML, but who do not have the time to go through all the options.

## OUTLINE OF THIS REPORT

The different types of ML platforms are presented in the order below:

– Pre-trained models as a service: platforms that give access to predictions from pre-trained models, aimed at product teams (including domain experts and software developers)

– Vertical ML as a service: platforms that allow the creation and deployment of custom predictive models for a specific type of business problem, aimed at product teams

– Semi-specialised machine learning as a service (MLaaS): platforms that allow the creation and deployment of custom predictive models for a specific type of input and task, aimed at product teams

– ML development platforms: platforms that include an experimentation component facilitating the creation of predictive models, used by data scientists as well as experts in the domain where ML is applied

– ML deployment platforms: platforms that simplify the deployment of models created by data scientists, used by data engineers and data scientists

This report ends with recommendations on how to choose the best platforms for individual use cases, and how to fill their gaps in the creation of production machine learning systems.

# Introduction

## SCOPE OF THIS REPORT

VentureScanner segments US AI startups into 13 groups, including Machine Learning Platforms, which had 357 startups at the end of Q1 2019, and Computer Vision Platforms, which had 234 startups in the same period.

The objective of this report is not to provide an exhaustive list of platforms, but to give a broad overview of the types of products and services that are available. It provides examples of platforms with different characteristics, from which to extract general principles, ideas and recommendations. The examples that are discussed here were chosen for illustration purposes, and this report is not intended to provide a benchmark. It does, however, provide advice to create individual benchmarks and to help readers to find or create the best ML platform, for their own needs or those of their customers.

The number of ML platforms quoted above could be even bigger, if vertical ML platforms are considered. Indeed, some of these were listed by VentureScanner in the Machine Learning Applications group, including Sift[6], a startup that allows users to build custom fraud detection models and to access them in a production application via a prediction API.

The number of vertical ML platforms is expected to continue to grow. David Kelnar of MMC Ventures remarks that nine in ten AI startups in Europe are vertical[7], and it is anticipated that many of them could benefit from offering platforms for others to build on. Bradford Cross of Data Collective Venture Capital sees vertical AI startups as providing fully-integrated solutions to business problems[8], which are powered by predictive models. However, the authors of Prediction Machines[9] argue that it could be more advantageous in some cases to give access to the models' raw predictions, as illustrated with an example in the medical sector. The vertical platforms that are presented here provide both business solutions and access to raw predictions.

This report is written to inspire organisations not only to use existing ML platforms, but to build on them and create their own vertical platforms, for their customers to use and to build on.

**Contents**

# 1. Pre-trained models as a service platforms

# 1. Pre-trained models as a service platforms

This example relates to developers of a software application that can be improved with predictive features. This could be a food delivery app, for instance (think UberEats), to which predicted delivery times could be added. Predictions would be based on a model that maps a given order, plus its context to a delivery time. Creating this model would require analysis of historical data. Assume that a data scientist has already built such a model. What would be the best way to make it accessible to application developers?

To add further context and examples: imagine developing a customer support application for a global company. It wants to improve the app to reduce the average response time and increase customer satisfaction. When receiving a new support ticket, the first step towards an answer is to assign a representative based on the customer's language. This step could be automated with a language detection model. Another idea of improvement would be to use a sentiment model to identify the most negative tickets, and to assign them to the most experienced reps.

The standard practice to make predictive models available to developers is to create a prediction API. This section explains how APIs work, and shows an example where models made available by the Indico platform are used to improve the routing of customer support tickets. It provides examples of pre-trained models that anyone can re-use in their own applications, and of platforms as a service that provide access to their predictions.

## 1. Pre-trained models as a service platforms

### GIVING MACHINE INTELLIGENCE SUPERPOWERS TO SOFTWARE VIA APIS

An API is a set of protocols, functions and definitions for building software. It is made of “endpoints” that each define a software functionality, via its expected inputs and outputs. The implementations of these functionalities (that is the code to run) would reside in a library — which we don’t necessarily need access to. If the API was to be used from a Python computing environment, it is known as a Python API, as shown in examples later in this report. For now, we do not want to favour any specific programming environment (our app may not be written in Python), so the report will be focusing on http APIs, that can be “called” by sending http requests. These requests can be sent from any programming environment chosen, for example, Python, JavaScript, Go, C++, Swift, Java, or the command line.

Consider another example to illustrate how http APIs work. Imagine developing an application used for customer support by a team of representatives (think Zendesk). The business serves customers globally, and wants to improve the app so it allows the team of reps to provide faster answers to support tickets. The first step to provide an answer is to detect the language spoken by the customer. The developer may want to route the ticket to the best rep who speaks that language. One idea is to assign tickets with negative sentiment to the best available reps.

There are cloud services that provide access to pre-trained models for language detection and sentiment analysis, via http APIs. Language detection and sentiment analysis are the same for everyone, so it makes sense to reuse models built by others.

The idea of calling an http API is the same as when typing an http address in the browser: a request to a machine (server) is sent to provide the data back in the form of a webpage. There are just a couple of differences here:

– When sending some content along with the request, for example, for the input to pass the model (note that this is similar to submitting an online form in the browser). Most APIs use JSON[10] as a standard to pass data back and forth, which are made of key-value pairs. Here, only one key is needed: “input”

– The response will not be a webpage, but a JSON data structure, where one key is “prediction”

Here is an example API call sent from the command line, using a utility called curl[11], to request a sentiment prediction for this piece of text: “I will never stay in this hotel again”.

```
$ curl https://apiv2.indico.io/sentiment
-H ‘X-ApiKey: PASTE_YOUR_API_KEY_HERE’
-d ‘{“data”: “I will never stay in this
hotel again”}’
```

## 1. Pre-trained models as a service platforms

**Some more comments:**

- $ is the command line invite (not to be typed)
- https is the protocol used for the request
- apiv2.indico.io is the address of the API server
- sentiment is the API endpoint we are interested in
- -H is used to specify request "headers"; here, an API key is sent that allows us to authenticate to the server
- -d is used to send data along with the request

**The following response would be provided:**

```
{"prediction": 0.10827194878055087}
```

Sentiment values are given between 0 and 1, where 0 means extremely negative, and 1 means extremely positive.

This example API is served by Indico[12], being accessed via the internet. However, internet connectivity is not always required to use http APIs, Indico for instance would be able to access the API on the intranet. There are cases where the http API would be served on the same machine that is also calling for predictions ("localhost"). For instance, in internet of things (IoT) applications, a connected device equipped with a sensor could run a Node.js application that would read sensor values and react to them, and it could call a local http API in order to access a predictive model that was created in another programming environment (for example, Python).

APIs are powerful as they allow IT systems to decompose into interoperating services. Here, APIs allow app developers to focus on collecting data from the app, and integrating predictions into the app, and can be served locally, on-premise, or on the cloud.

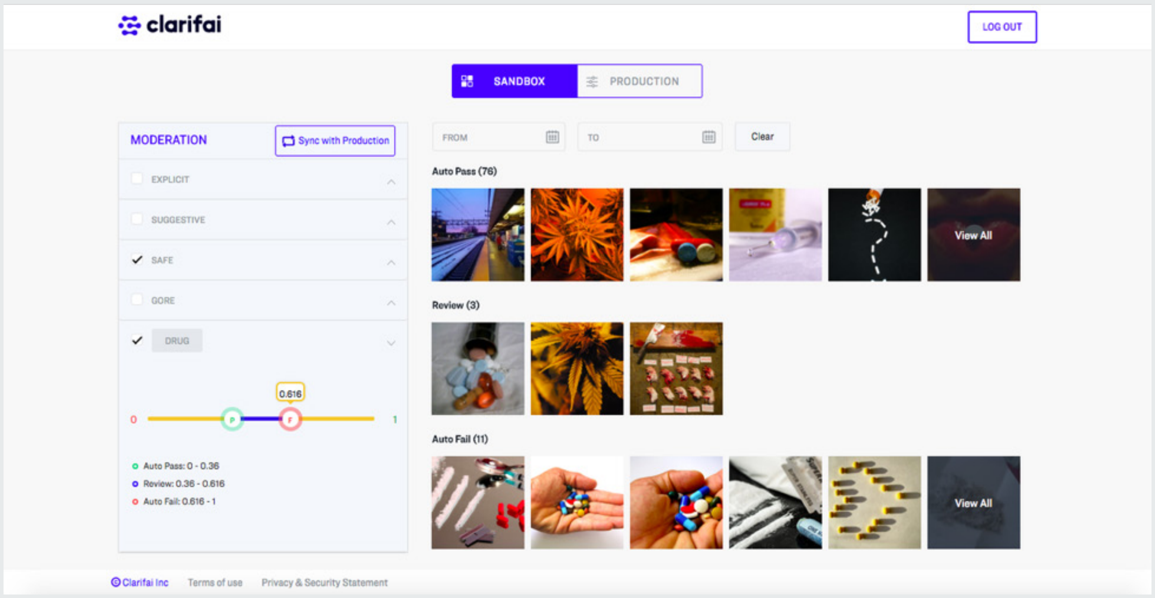
### PLATFORMS AS A SERVICE TO LEVERAGE PRE-TRAINED MODELS

It is possible to think of many other examples of pre-trained models on text data, that can be reused by anyone. If the app allows its users to send messages and photos to each other, it could improve by automatically filtering out bad language, nudity in images, or by automatically detecting scenes and objects in images and tagging them accordingly (for example "beach", "mountain", "city", "dog", "phone", etc.).

If developing an app for a connected security camera, intelligence could be added to it by analysing images captured from the camera upon detecting movement, detecting humans, locating faces and finding if the face is familiar or not, using a face recognition API like that of Kairos[13].

Clarifai[14] is another platform that provides access to several predictive models for images, via its API. It includes for example a 'Moderation' model (that detects concepts such as gore, drugs, explicit nudity, or suggestive nudity), a 'Travel' model (that recognises specific features of residential, hotel, and travel-related properties), a 'General' model that recognises over 11,000 different concepts including objects, themes, mood, and many more models.

## 1. Pre-trained models as a service platforms



Auto Pass: 0 - 0.36
Review: 0.36 - 0.616
Auto Fail: 0.616 - 1

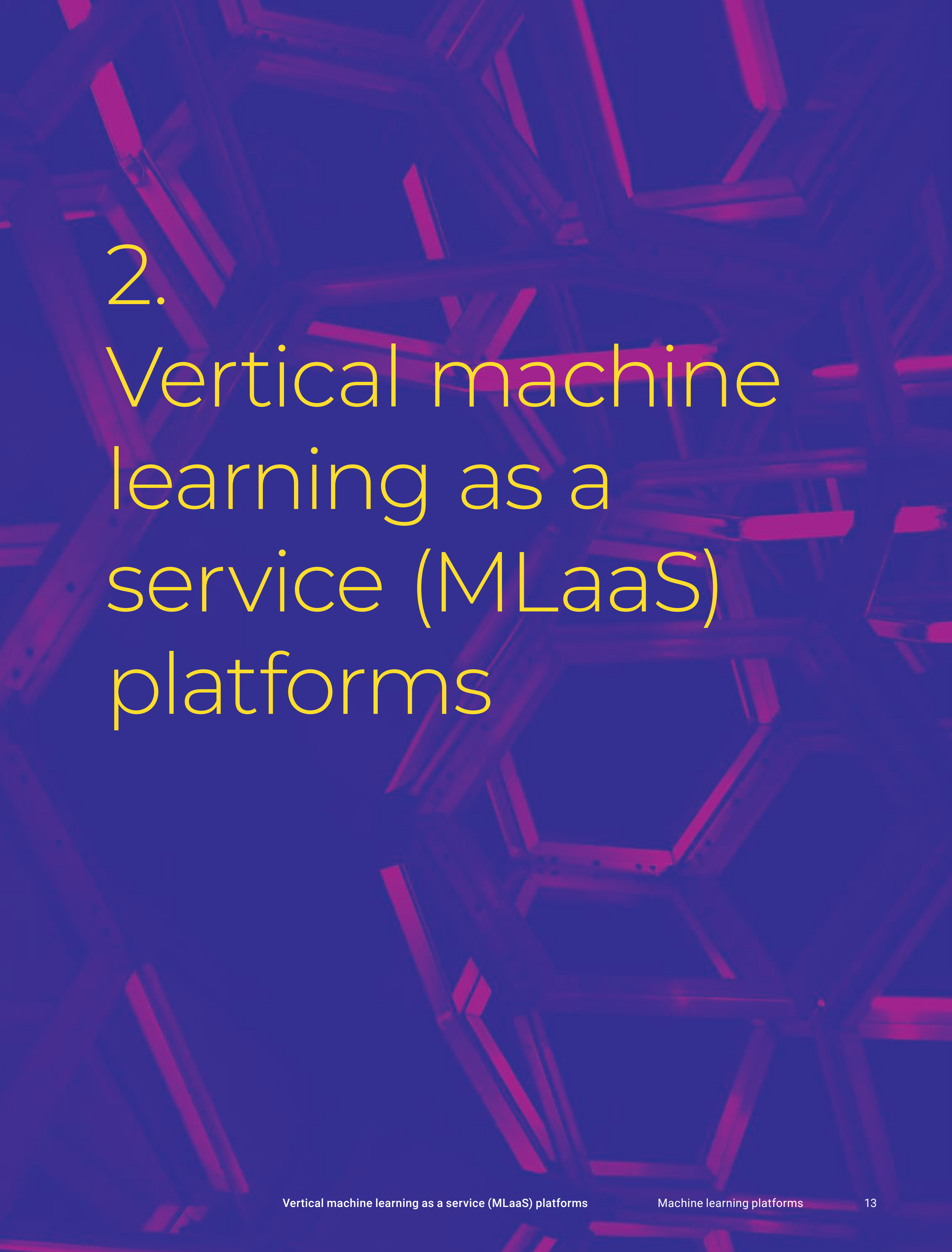
Accessing Clarifai's Moderation models from a web interface[15]

Other language models provided by Indico include text summarisation, emotion detection, personality detection and political orientation detection. In addition to images, videos and written language, there are APIs for speech, that allow detection of spoken language, transcription of speech to text, or speaker recognition, for example. For generic, difficult problems such as those mentioned above, and many more besides it often doesn't make sense to develop proprietary new technologies, but instead to leverage the result of significant research and development by others.

These solutions are readily available as APIs offered by platforms such as Google Natural Language API[16] and Google Vision API[17], and Microsoft Cognitive Services[18].

These are referred to as 'pre-trained models as a service' platforms. They typically target developers and product teams. Organisations may not need to hire data scientists to get started, but as referenced later in this report in the section about platform choice, thoughtful usage of these platforms is likely to require help from data engineers.

## Contents



# 2. Vertical machine learning as a service (MLaaS) platforms

# 2. Vertical machine learning as a service (MLaaS) platforms

Similarly to pre-trained models as a service platforms, vertical machine learning as a service platforms provide ready-made ML solutions to a prediction problem that others also have, but with one notable difference: the solution is based on an organisation's own data. They will be creating a predictive model that is created from the data we provide, for a specific ML task and a specific use case, hence the word "vertical". Google calls them "AI (pre-packaged) solutions".

One example of a vertical AI solution is Landing Light[19]: it allows users to detect defects in manufactured objects, based on images of those objects. A manufacturer would need to provide training data, consisting of example images of objects with and without defects. The solution relies on ML algorithms tuned for this particular type of classification problem and for this particular type of input (images of a single object against a monochromatic background), in a way that proved successful on different training sets provided by different manufacturers. These algorithms would run on the new training dataset provided by the manufacturer resulting in the creation of a model that is unique to them.

By focusing on a specific ML use case, and by utilising a solution, that has already had success for various organisations, with similar business problems, vertical platforms are able to drastically reduce the time for others to create custom models built from their own data. It also reduces the uncertainty in the effectiveness of these models, compared to tackling new types of ML use cases; the uncertainty that remains is only due to the quantity and quality of our data.

In this section, a case study of the Sift[20] fraud detection API shows how to query fraud risk scores and how to pass new observations for the risk model to update itself. This is followed by examples of platforms that target prediction problems for customer relationship management (CRM) (including support ticket allocation), and other examples of vertical platforms in various domains.

In addition to leveraging the best algorithms for the job, these platforms provide an edge on the data preparation side, with problem-specific data pre-processing (for example, de-saturating and resizing images of manufactured objects), data augmentation techniques (for example, flipping or rotating or cropping images), and data enrichment (for example, adding features extracted from social networks to better represent customers. When making predictions on platforms or adding their company information extracted from a curated database). These platforms can also apply domain-specific logic on top of predictions (for example, removing out-of-stock items from e-commerce recommendations, and enforcing product variety).

### CASE STUDY: BUILDING A CUSTOM FRAUD DETECTION MODEL WITH THE SIFT API

Vertical platforms that can be used instantly, and that provide APIs, are targeted at product teams and developers. The APIs include, as usual, an endpoint to query predictions, as well as endpoints to provide training data. This can be a single data point, or a multitude of data points, to add to or to replace an existing training set. The expectation is that sending additional data should automatically trigger model updates.

Sift is a fraud prevention service based on ML. It makes it very easy for businesses that process credit card payments to solve the problem of fraudulent payments. This is based on predictions of how likely fraud is, for a given transaction. These predictions are given as scores, between 0 and 1, and this is how scores are queried:

#### Example request

```
$ curl https://api.sift.com/v205/events?
return_score=true&abuse_types=payment_
abuse,promotion_abuse
  -d '{
    "$type"           : "$create_order",
    "$api_key"        : "YOUR_API_KEY",
    "$user_id"        : "billy_jones_301",
    "$session_id"     : "gigtleqddo84l8cm15qe4il",
    "$order_id"       : "ORDER-28168441",
    "$user_email"     : "bill@gmail.com",
    "$amount"         : 115940000, // means $115.94
    "$currency_code"  : "USD"
  }'
```

#### Example response

```
HTTP/1.1 200 OK
Content-Type: text/json;charset=UTF-
Connection: keep-alive

{
  "body": {
    "status": 0,
    "error_message": "OK",
    "request": "body_of_the_request_you_sent",
    "time": 1454517138,
    "score_response": {
      "status": 0,
      "error_message": "OK",
      "user_id": "billy_jones_301",
      "scores": {
        "payment_abuse": {
          "score": 0.898391231245,
          "reasons": [
            {
              "name": "UsersPerDevice",
              "value": 4,
              "details": {
                "users": "a, b, c, d"
              }
            }
          ]
        },
        "promotion_abuse": {
          "score": 0.472838192111,
          "reasons": []
        },
      },
      "latest_labels": {
        "payment_abuse": {
          "is_bad": true,
          "time": 1352201880,
          "description": "received a chargeback"
        },
        "promotion_abuse": {
          "is_bad": false,
          "time": 1362205000
        }
      }
    }
  }
  "http_status_code": 200
}
```

## 2. Vertical machine learning as a service (MLaaS) platforms

For Sift to build and update the model, it would need to identify whenever a transaction was successful and when it was fraudulent. This is done using the decisions endpoint. As an example, a decision could be "Accept Order" or "Block Order", for instance, if the payment processor sends a notification to an analyst that a payment did not go through, and the analyst manually reviews the order, and decides to block it, we would pass this data to Sift via the following call:

```
$ curl  https://api.sift.com/v3/accounts/
{accountId}/orders/{orderId}/decisions
  -u YOUR_API_KEY
  -d '{
        "decision_id" : "block_order_payment_abuse",
        "source"      : "MANUAL_REVIEW",
        "analyst"     : "analyst@example.com"
      }'
```

This allows Sift to automatically learn from new data points, and continuously improve the accuracy of risk scores.

It is possible to infer from these example requests that Sift performs a classification task on tabular data. Readers can learn more about the API via the developer portal (sift.com/developers/docs)[21].

### MLAAS PLATFORMS FOR PREDICTIVE CUSTOMER RELATIONSHIP MANAGEMENT

Infer[22] is a lead-scoring solution, the input to the ML models it builds is a prospective customer (lead), represented with tabular data, and the score represents how likely it is that the lead converts to an actual customer. Infer integrates with customer relationship management (CRM) systems, web analytics, and marketing automation tools, from which it can extract training data, and where it can give access to its predictions.

Salesforce Einstein[23] is a solution that allows prediction of several business outcomes, for a given customer input, such as churn or lifetime value. Einstein Prediction Builder allows users to create custom models on any Salesforce field or object, with clicks.

The Einstein Intent API[24] allows users to categorise unstructured text from emails, chats, or web forms, into custom labels to better understand what users are trying to accomplish. One type of business case that can be tackled with the Intent API is routing customer support tickets (textual inputs):

– Determining what products prospects are interested in and sending customer inquiries to the appropriate salesperson

– Routing service cases to the correct agents or departments, or provide self-service options

Google Contact Center AI[25] and Answer IQ[26] are two other platforms that provide similar solutions.

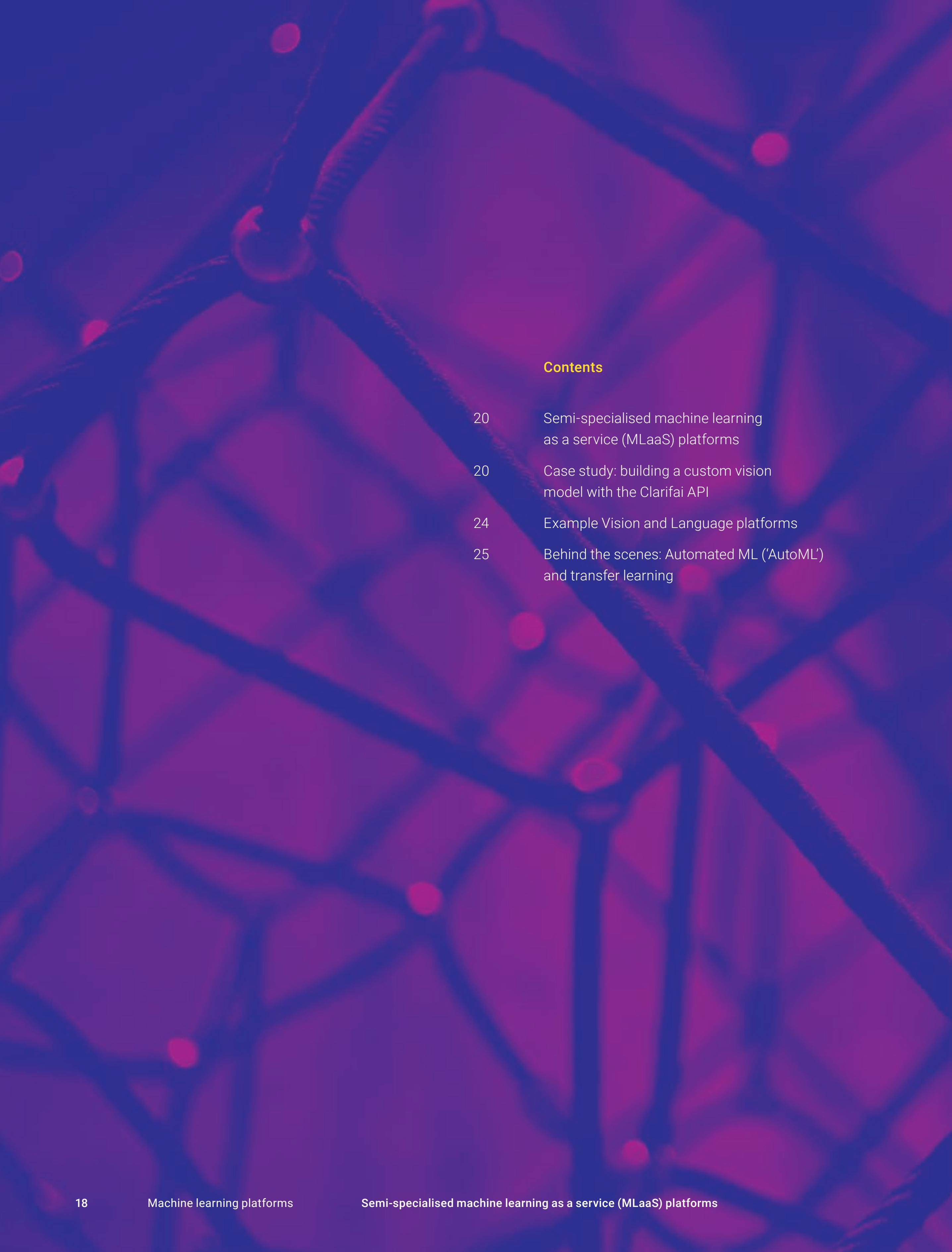
## 2. Vertical machine learning as a service (MLaaS) platforms

### OTHER EXAMPLE PLATFORMS

Google Cloud Talent[27] is a human resources solution, which allows users to score job candidates based on the job description (textual data) and the candidate's profile (tabular data).

Dialogflow[28] is a platform that builds models to detect the intent of a request/command to a chatbot or virtual assistant (VA). If the interaction between user and VA is via voice, a speech-to-text service can reduce this to a text classification problem, on inputs of a certain nature (that is requests). Custom intents can be defined, but we would need to provide example requests, their associated intents, and entities (for instance, the request "set a timer to five minutes" has a time entity, which is "five minutes").

Google Recommendations AI[29] is an e-commerce solution that can provide a set of product recommendations, for a given customer. Once data sources are connected (Google Tag Manager, Google Shopping, Google Cloud Storage, BigQuery), custom models are automatically created. Recommendations can be configured to either increase engagement, revenue, or conversions. The platform allows users to apply business rules to tune recommendations shown to customers by adding diversity among recommended products, filtering out unavailable products, etc. Recommendations can be added anywhere (product pages, shopping cart, order confirmation page, mobile app, email). The service scales automatically to match traffic to our e-commerce site.



## Contents

# 3. Semi-specialised machine learning as a service (MLaaS) platforms

# 3. Semi-specialised machine learning as a service (MLaaS) platforms

The platforms in this section specialise in inputs of a fixed type or nature. Similarly to vertical platforms, they allow users to build custom models from their own data. These platforms specialise in one type of ML task, for instance classification or concept detection, but are less specialised than vertical platforms, as they can be used to tackle any use case for that type of task.

In theory, a semi-specialised API could be used to build a vertical API. For instance, the API of a language platform could be used to classify customer support tickets, and build an intelligent routing solution (similar to Google Contact Center AI). Or a vision API can be used to classify images of manufactured objects, and build defect detection solutions (similar to Landing Light). Some of the techniques listed at the end of the previous section could be implemented: we would not have the ability to tune algorithms, but we could use the knowledge of the domain of application and of the problem to implement data pre-processing and augmentation techniques.

The case study of Clarifai's image classification API shows how to concretely use the API to build a defect detector, followed by a list of some other Vision platforms that specialise on image input data, and language platforms that specialise on text input data. Finally, some of the limitations these platforms can have are introduced by looking behind the scenes and discussing transfer learning and ML automation techniques.

## CASE STUDY: BUILDING A CUSTOM VISION MODEL WITH THE CLARIFAI API

Clarifai[30] is a platform that provides access to pre-trained models, which detect concepts from images (for instance, food items from pictures of dishes, down to the ingredient level). It also provides the ability to create our own models. The first step is to add images that contain the concepts the model should see. Note that an image can contain several concepts, which are not necessarily exclusive. In this example case, the solution sought is a binary classification model that detects defects in images of manufactured objects.

### Adding images with concepts

Data (in this case an image labelled as having a defect) can be sent to Clarifai as shown below:

```
$ curl https://api.clarifai.com/v2/inputs
  -H "Authorization: Key YOUR_API_KEY"
  -H "Content-Type: application/json"
  -d '{ "inputs": [
        { "data": {"image": {"base64": "'"$(base64
/home/user/object1.jpeg)"'"},
          "concepts": [ {"id": "defect", "value":
true} ] }
      ] }'
```

More than one example can be sent in a single API call.

## 3. Semi-specialised machine learning as a service (MLaaS) platforms

### Creating a model

Once all the examples have been added,
a model would be created as follows:

```
$ curl https://api.clarifai.com/v2/models
  -H "Authorization: Key YOUR_API_KEY"
  -H "Content-Type: application/json"
  -d '
  {
    "model": {
      "name": "defect_detector",
      "output_info": {
        "data": {
          "concepts": [
            {
              "id": "defect"
            }
          ]
        },
        "output_config": {
          "concepts_mutually_exclusive": true,
          "closed_environment": false
        }
      }
    }
  }'
```

A model id would be received in the response.
Let's call it model_id.

### Training the model

"Creating" a model just means that a model object has been initialised. The model needs to be trained from data in a separate call. When training a model, the system is told to look at all the images with the concepts provided, and to learn from them. This train operation is asynchronous — it may take a few seconds or minutes for our model to be fully trained and ready.

```
$ curl https://api.clarifai.com/v2/models/<model_
id>/versions
    -H "Authorization: Key YOUR_API_KEY"
```

The response will contain a model_version id.

### Predicting

Predictions are queried as follows:

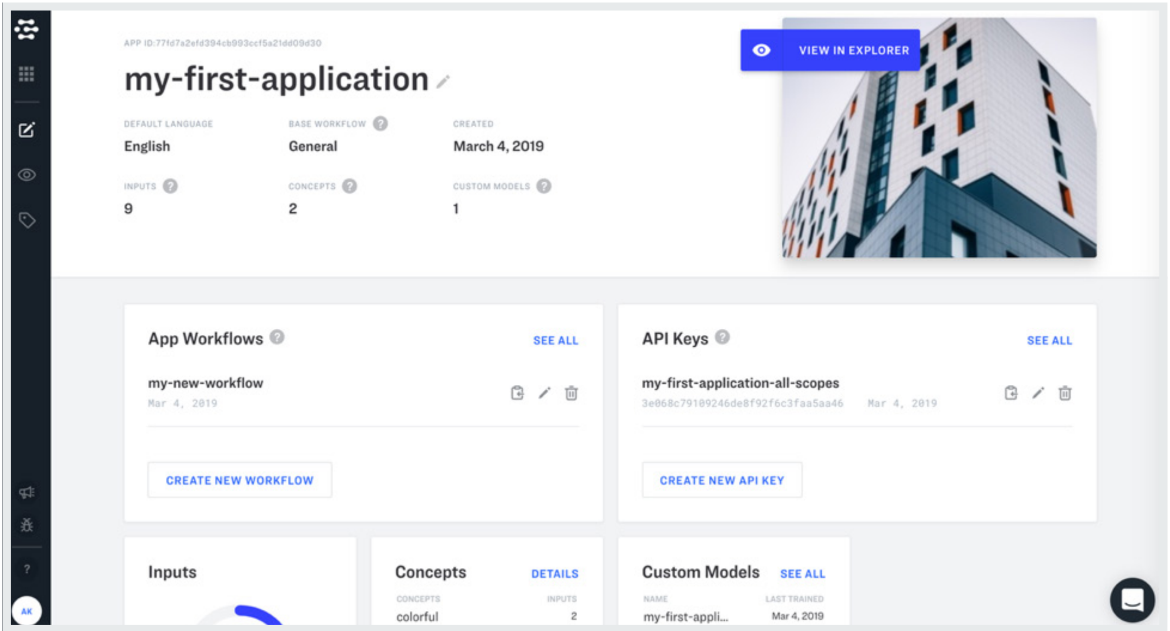
```
$ curl https://api.clarifai.com/v2/models/<model_
id>/outputs
    -H 'Authorization: Key YOUR_API_KEY'
    -H "Content-Type: application/json"
    -d '
    {
      "inputs": [
        {
          "data": {
            "image": {
              "base64": "'"$(base64 /home/user/
new_object.jpeg)"'"
            }
          }
        }
      ]
    }'
```

## 3. Semi-specialised machine learning as a service (MLaaS) platforms

If no model version id is specified, predictions will occur on the most recent version of the model. Predictions can be queried from a specific model version by changing the URL to

```
https://api.clarifai.com/v2/models/<model_id>/
versions/<model_version_id>/outputs
```

More images with concepts can be added, and the model can be re-trained, which would produce a new model version.


APP ID:77fd7a2efd394cb993ccf5a21dd09d30

my-first-application

DEFAULT LANGUAGE: English
BASE WORKFLOW: General
CREATED: March 4, 2019
INPUTS: 9
CONCEPTS: 2
CUSTOM MODELS: 1

App Workflows
my-new-workflow
Mar 4, 2019

API Keys
my-first-application-all-scopes
3e068c79109246de8f92f6c3faa5aa46    Mar 4, 2019

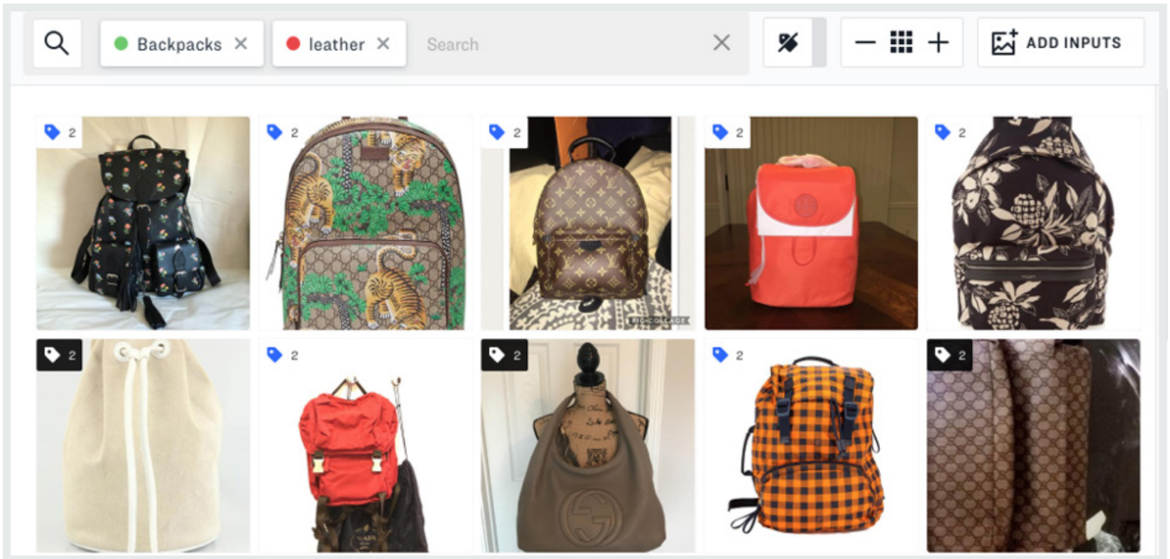
Inputs

Concepts
CONCEPTS: colorful
INPUTS: 2

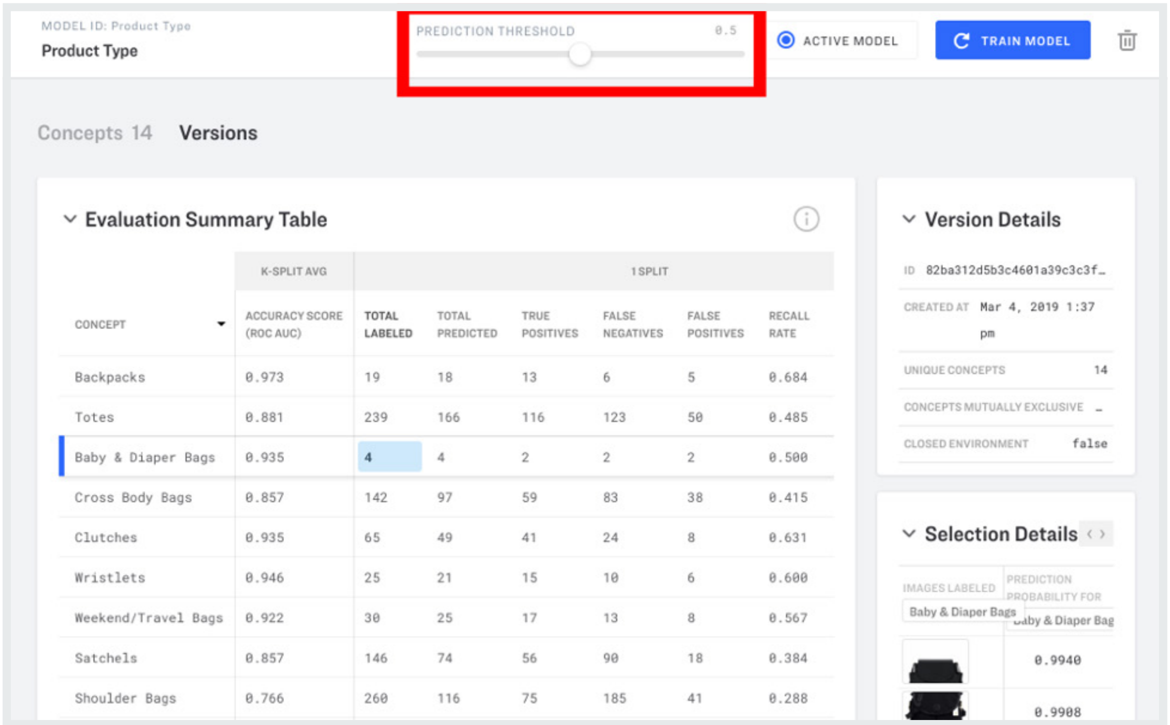
Custom Models
NAME: my-first-appli...
LAST TRAINED: Mar 4, 2019

Creating models from the Clarifai Portal[31]

## 3. Semi-specialised machine learning as a service (MLaaS) platforms



Preview custom models in the Clarifai Portal[32]



MODEL ID: Product Type

Product Type

PREDICTION THRESHOLD 0.5

ACTIVE MODEL

TRAIN MODEL

Concepts 14 Versions

Evaluation Summary Table

| | K-SPLIT AVG | 1 SPLIT | | | | | |
|---|---|---|---|---|---|---|---|
| CONCEPT | ACCURACY SCORE (ROC AUC) | TOTAL LABELED | TOTAL PREDICTED | TRUE POSITIVES | FALSE NEGATIVES | FALSE POSITIVES | RECALL RATE |
| Backpacks | 0.973 | 19 | 18 | 13 | 6 | 5 | 0.684 |
| Totes | 0.881 | 239 | 166 | 116 | 123 | 50 | 0.485 |
| Baby & Diaper Bags | 0.935 | 4 | 4 | 2 | 2 | 2 | 0.500 |
| Cross Body Bags | 0.857 | 142 | 97 | 59 | 83 | 38 | 0.415 |
| Clutches | 0.935 | 65 | 49 | 41 | 24 | 8 | 0.631 |
| Wristlets | 0.946 | 25 | 21 | 15 | 10 | 6 | 0.600 |
| Weekend/Travel Bags | 0.922 | 30 | 25 | 17 | 13 | 8 | 0.567 |
| Satchels | 0.857 | 146 | 74 | 56 | 90 | 18 | 0.384 |
| Shoulder Bags | 0.766 | 260 | 116 | 75 | 185 | 41 | 0.288 |

Version Details

| | |
|---|---|
| ID | 82ba312d5b3c4601a39c3c3f… |
| CREATED AT | Mar 4, 2019 1:37 pm |
| UNIQUE CONCEPTS | 14 |
| CONCEPTS MUTUALLY EXCLUSIVE | – |
| CLOSED ENVIRONMENT | false |

Selection Details

| IMAGES LABELED Baby & Diaper Bags | PREDICTION PROBABILITY FOR Baby & Diaper Bag |
|---|---|
| | 0.9940 |
| | 0.9908 |

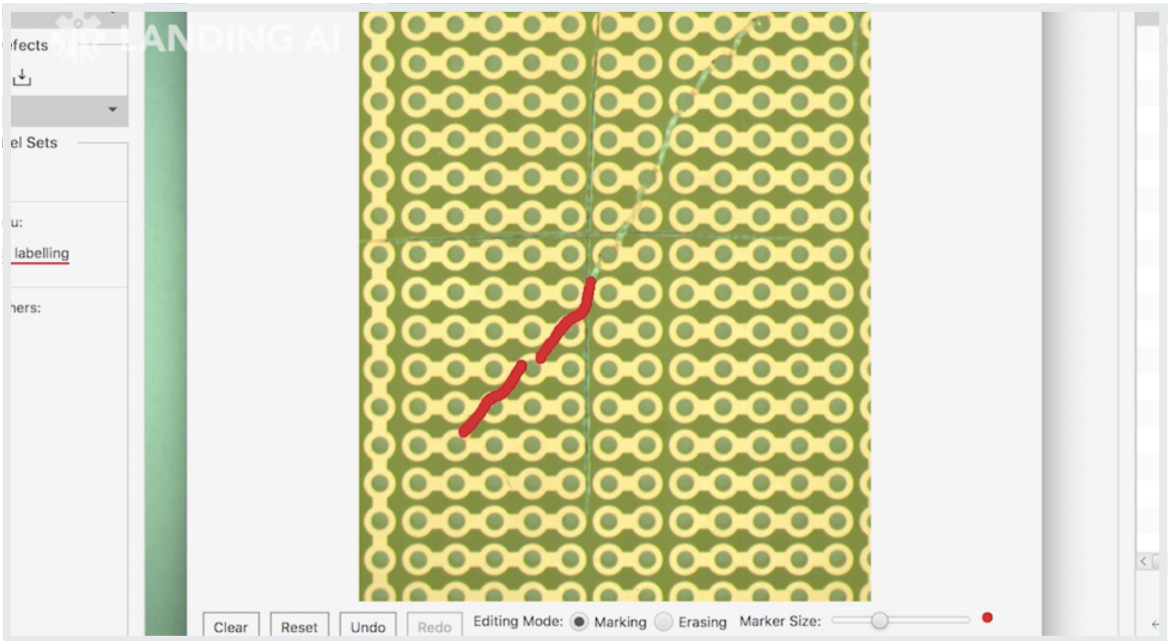
Model evaluation and threshold adjustment in the Clarifai Portal[33]

## 3. Semi-specialised machine learning as a service (MLaaS) platforms

### EXAMPLE VISION AND LANGUAGE PLATFORMS

Language platforms allow users to train custom text models from their own data. Inputs would be text in a given language, or images or video for Vision platforms; outputs would be labels that identify concepts. These could be references that are internal to our organisation, for example project names or team names. Indico, whose pre-trained models were presented in the first section of this report, is one such platform. Other examples include Amazon Comprehend[34], Google AutoML Natural Language[35], and MonkeyLearn[36]. Lateral[37] is another interesting platform, which includes a tag suggestion API (based on existing text documents and tags) and a recommendation API that recommends similar documents to a given text document. This uses a hybrid approach, where relevance is determined based on text content and user behaviour. For Vision, Amazon has Rekognition[38] and Google has AutoML Vision[39].

Note that some of these platforms also offer data annotation services, to help create proprietary models (for instance, Google has a crowdsourced Data Labeling Service[40]). If the platform being used does not have such a service, developers can also use a data annotation platform such as Figure Eight[41]. If data annotation requires knowledge of the domain of application and of the problem, developers may want to develop their own data annotation interface, as Landing Light did.



Landing Light's annotation interface allows to indicate pixels of interest for classifying images[42]

## 3. Semi-specialised machine learning as a service (MLaaS) platforms

### BEHIND THE SCENES: AUTOMATED ML (AUTOML) AND TRANSFER LEARNING

Semi-specialised ML platforms use pre-configured ML algorithms, and would automatically tune some of their parameters based on the data provided. This is where the name AutoML comes from, in the Google Cloud Platform products ("Automated ML"), but it is not specific to Google. This automation is extremely useful, but for it to work well, it should use an evaluation procedure that makes sense for the application.

Imagine building a model to detect fake reviews posted on websites such as Tripadvisor or Booking.com that should generalise to any website where reviews are posted. It is important to ensure that the "validation data", which is used to evaluate models, comes from different websites than the training data. However, most platforms would choose this validation data randomly among the data provided. If this is an issue for the application, as in the case of fake review detection, a platform that allows users to specify validation data explicitly should be sought.

In addition to using AutoML techniques, semi-specialised platforms wouldn't ordinarily create models from scratch, but they would adapt existing models by using "transfer learning" techniques. This works by extracting powerful numerical representations of inputs, from pre-trained neural network models. Such representations are found in neural networks as intermediate layers of neurons. The closer they are to the output of the network (that is the thing to predict), the more specific they are to the task that the network was trained on. The closer they are to the input of the network (that is the original, raw representation of inputs) the more general these representations are.

Transfer learning may not work if the nature of inputs on which the new model is trained is too different from the nature of inputs to the pre-trained models. For instance, all of Clarifai's models are made for images from everyday life. Therefore, it is not clear if transfer learning would work on medical images such as fMRI scans, or on satellite images. Another approach may be needed to build models that detect concepts on such images – for example anomalies on medical images, or solar panels on satellite images.

## Contents

# 4. Machine learning development platforms

# 4. Machine learning development platforms

ML development platforms (also called "workbenches") give more flexibility for building custom models. Some of these platforms also have the capacity to deploy models as APIs for production usage, however this report will focus on deployment in the next section.

The platforms presented allow users to experiment with different learning techniques, algorithms and parameters, to evaluate them, and to create models with those that work best on the data at hand. Their objective is to make it easier and faster to create ML models, and to remove the difficulty in using a mix of open source libraries, connecting various software components, and making it run on the appropriate infrastructure. They make experienced teams more efficient in their ML efforts, and they also democratise ML to a wider audience, including (very) small and medium-sized businesses.

Platforms do this by either providing a web-based "studio", or an integrated development environment (IDE), with collaboration features (so that each person in the team has access to the same environment with their individual account) and experiment tracking (functionalities to record all experiments, browse, filter and sort them, and to view results for each). A studio is a graphical user interface (UI) for experimenting and building ML models, without necessarily writing code. It gives access to proprietary software with higher-level functionalities than open source, such as AutoML. Model building can also be done via an API (usually http or Python) that provides the same functionality as the studio.

The studio is mostly intended for experimentation, whereas the API can also be used for continuous delivery of models and for their deployment. Once a learning technique and parameters that work have been found through experimenting in the studio, it can be used programmatically via the API. It would be called by developers to query predictions, or to trigger model creation. This can be useful to automatically update an existing model with fresher data, or to create a dedicated model per user of our application, that learns from their own data (for instance, a priority filtering use case could require different models for different end-users).

Platforms can be varied in their flexibility and ease of use: the highest level platforms are the easiest to use, but the least flexible; they are also the most restricted in the types of ML tasks that can be tackled with built-in algorithms. This report starts by presenting high-level platforms as a service, then presents self-hosted platforms with studios built on top of common open source libraries. All the studios mentioned in this section are focused on tabular data; they allow users to deal with classification and regression problems, unless indicated otherwise. Note that they could also be used with mixed data including image and text, after "embedding them": the image and text representations would be transformed to numerical features, and thus be considered tabular data; Indico has two API endpoints for that: Text Features[43] and Image Features[44].

The report progresses to present cloud-based IDEs that are more flexible, and also make developing with common open source ML libraries easier. Finally the issue of developing ML pipelines that include a featurisation component is discussed.

# 4. Machine learning development platforms

**HIGH-LEVEL PLATFORMS AS A SERVICE**
Platforms as a service are among the easiest to use, as there's nothing to install and no need to worry about infrastructure.

**High-level features include:**

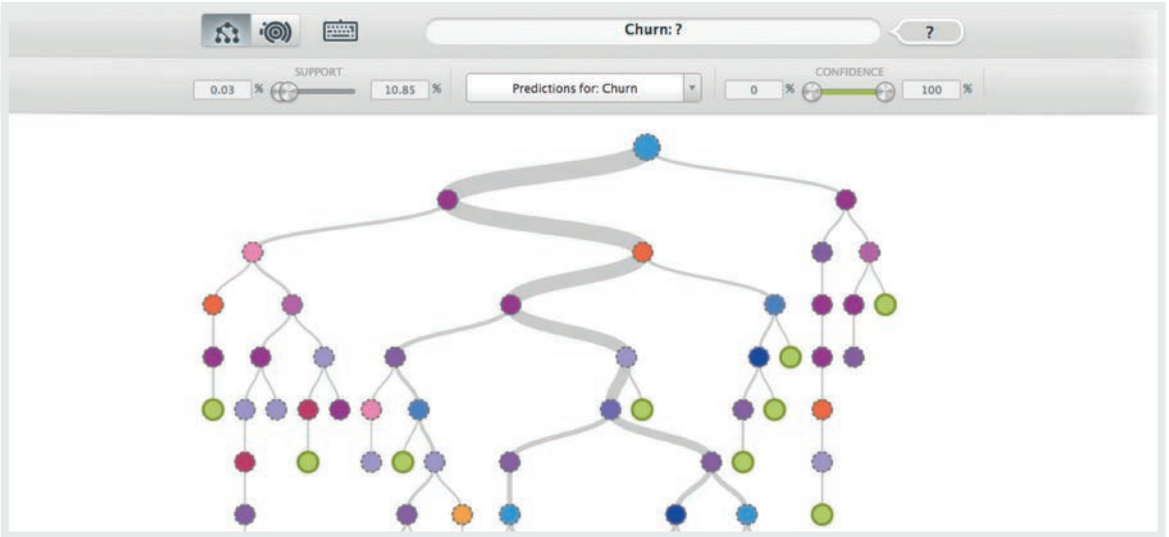
- Automatic detection of the type of problem (for example, classification or regression)
- Automatic preparation of data (for example, encoding of categorical variable, normalisation, feature selection and so on)
- Automation configuration of the learning algorithm (AutoML with meta-learning and smart search of hyper-parameters)

These platforms are particularly useful for those with less knowledge of ML algorithms, and they can be accessible to domain experts or software developers. Arguably, they are also useful for data scientists, as they allow for faster experimentation and fewer errors.

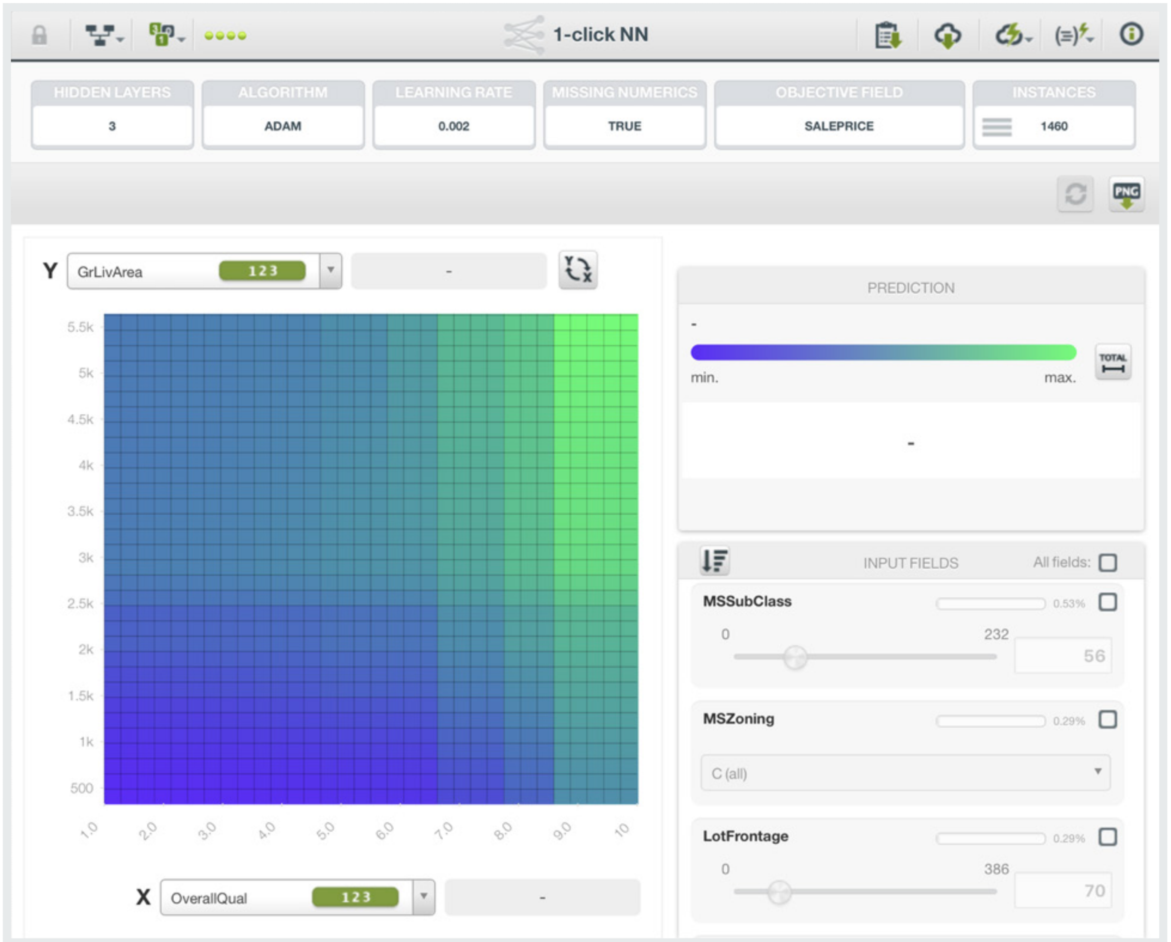
This, in turn, allows focus on everything that takes place before and after learning from data — which is what matters most in practice: gathering ML-ready data, evaluating predictions, and using them in software.

BigML[45] is a platform that provides a variety of built-in algorithms for classification and regression (Decision trees, random forests, boosted trees, neural networks, linear and logistic regression) that supports numerical features and categorical features, as well as text features.
It also gives access to unsupervised learning algorithms (clustering with K-means and G-means, anomaly detection with Isolation Forests, PCA, topic modelling) and time series forecasting (exponential smoothing). Its "OptiML" functionality implements AutoML techniques that find the best model for a given training set, validation/test set, and performance metric, in a given time budget. Its "Fusions" functionality creates model ensembles.

The BigML Studio has great interactive visualisations, see some examples below.



BigML's Decision Tree interactive visualisation



| HIDDEN LAYERS | ALGORITHM | LEARNING RATE | MISSING NUMERICS | OBJECTIVE FIELD | INSTANCES |
|---|---|---|---|---|---|
| 3 | ADAM | 0.002 | TRUE | SALEPRICE | 1460 |

Y: GrLivArea 123
X: OverallQual 123

MSSubClass 0.53% — 0 to 232 — 56
MSZoning 0.29% — C (all)
LotFrontage 0.29% — 0 to 386 — 70

Inspecting a neural network created automatically by BigML, with an interactive partial dependence plot

## 4. Machine learning development platforms

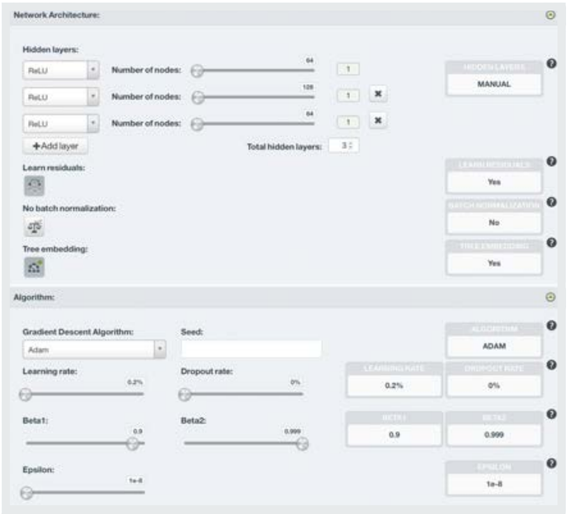
The BigML API allows users to query predictions and trigger model training at scale on the BigML cloud infrastructure. Let's assume that BIGML_AUTH is an environment variable that contains the BigML username and API key. Here is how the API would be called to use OptiML for a given training and test datasets (identified by their ids), in a way that maximises the area under the receiver operating characteristic (ROC) curve (AUC):

```
$ curl https://bigml.io/optiml?$BIGML_AUTH
       -d '{
            "dataset": "<training_dataset_id>",
            "test_dataset": "<test_dataset_id>",
            "metric": "area_under_roc_curve",
            "max_training_time": 3600
          }'
```

The request is asynchronous and will return an OptiML id. After one hour (3600 seconds), the API can be hit to get information about the OptiML object created, in particular the summary that contains the ID of the best model that was found for the dataset. Predictions can then be requested from this model:

```
$ MODEL_ID = curl https://bigml.io/optiml/
<optiml_id>?$BIGML_AUTH | jq -r ".optiml.
summary.model.best"
$ curl https://bigml.io/predict?$BIGML_AUTH
       -d '{
             "model": "'"$MODEL_ID"'",
             "input_data": {"text": "I will
never stay in this hotel again"}
           }'
```

BigML is probably the easiest platform to use among those presented in this section, but also the least flexible. ML practitioners will find missing functionalities, such as the ability to plot learning curves and to use a custom performance metric. However, BigML allows users to execute scripts on their platform in a language they created and called WhizzML, which is a high-level programming language specific to machine learning. This extends the functionality of the platform. It is also one of the most closed platforms here, as it does not play well with open source, so is limited in the format in which models can be exported. It is not possible to export modelling scripts — however it is possible to see which algorithm and parameters were used to create a model.


Network Architecture:

Hidden layers:
ReLU Number of nodes: 64 1
ReLU Number of nodes: 128 1
ReLU Number of nodes: 64 1
+Add layer
Total hidden layers: 3
Learn residuals:
No batch normalization:
Tree embedding:

MANUAL
Yes
No
Yes

Algorithm:

Gradient Descent Algorithm: Adam
Seed:
Learning rate: 0.2%
Dropout rate: 0%
Beta1: 0.9
Beta2: 0.999
Epsilon: 1e-8

ADAM
0.2%
0%
0.9
0.999
1e-8

Inspecting the neural network parameters found by BigML when creating an optimal model for a given training set and cross-validation method

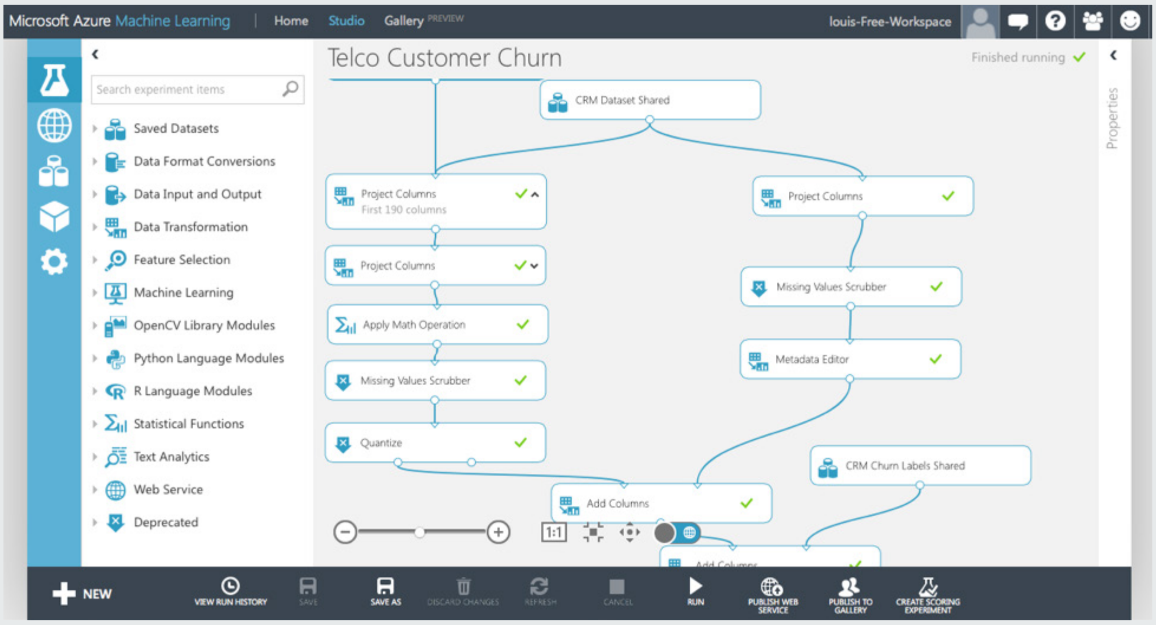
## 4. Machine learning development platforms

Google AutoML Tables[46] is a beta product that bears similarities to BigML, but is not yet intended for usage in critical applications. Its UI is simpler, and the product seems more targeted at developers than domain experts. Google Cloud Inference API[47] focuses on time series prediction. It is an alpha product that doesn't include a UI, but is more geared towards deployment in production.

Craft.ai[48] is a fully automated MLaaS platform for experimentation and deployment, with a focus on explainability and on the use of decision trees.

Microsoft Azure ML[49] is an MLaaS platform offering a studio, AutoML functionality, and the ability to turn models into prediction APIs that scale automatically.

The studio has a component that was not present in the other platforms presented: the "interactive canvas". It allows users to view and visually edit sequences of operations on data (loading, preparation, modelling, evaluation). A sequence of operations that take data in and apply ML algorithms to produce (and evaluate) a predictive model is called an ML "pipeline". The canvas makes it easier to understand pipelines, but also to create them, by avoiding potential errors that are caught by the interface (for example, missing inputs to the data operations, or forbidden connections).

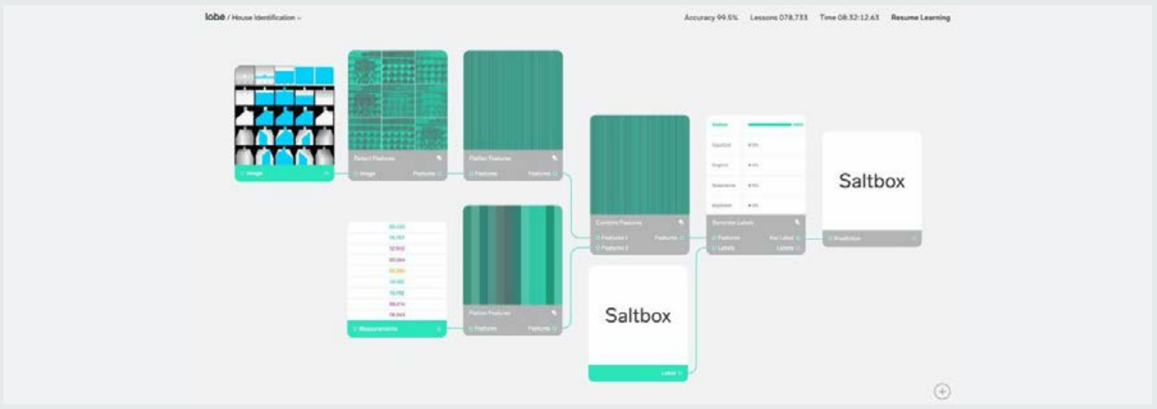


Microsoft Azure ML's Studio — accessed July 2015

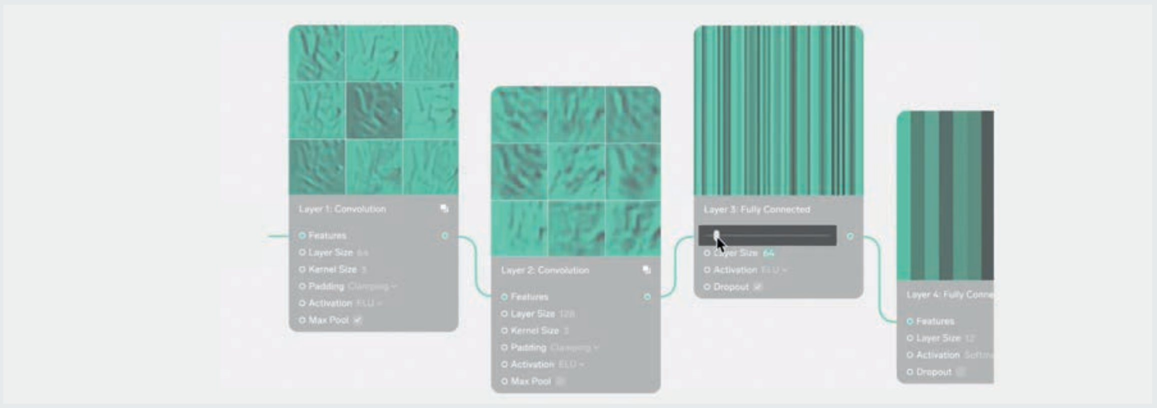
## 4. Machine learning development platforms

Lobe[50] is a service that Microsoft acquired in 2018, which provides an interactive canvas and AutoML functionality, but also allows users to deal with image features. It provides an easy-to-use environment to automatically build neural network models, via a visual interface.

Models are made of building blocks that can be fully controlled (Lobe is built on top of TensorFlow and Keras); some building blocks are pre-trained, which allows for transfer learning. Training can be monitored with real-time, interactive charts. Trained models can be made available via the Lobe Developer API, or exported to Core ML and TensorFlow files to run on iOS and Android devices.



Visual editing of a sequence of operations leading up to an ML model on image and numerical features, in Lobe[51]



Customising neural network models in Lobe (here, a CNN)[52]

## SELF-HOSTED STUDIOS

Several other studios offer a canvas, under different names — DataRobot[53] calls it “Blueprint”, Rapidminer[54] and Dataiku[55] call it “Workflow” — and AutoML functionality. The ML development platforms mentioned here are seen as lower-level than the previous ones, since they are not provided “as a service” and need to be installed and hosted on our own (cluster of) machines. However, they have interesting capabilities, which were not found in the MLaaS offerings.

One common aspect that these platforms share is that they are based on standard open source ML libraries, and allow users to use custom libraries and custom code. They let users export ML workflows/pipelines as Python scripts, and to export trained models to various open formats that avoid lock-in. Note that some of the vendors mentioned also provide separate deployment and model serving solutions.

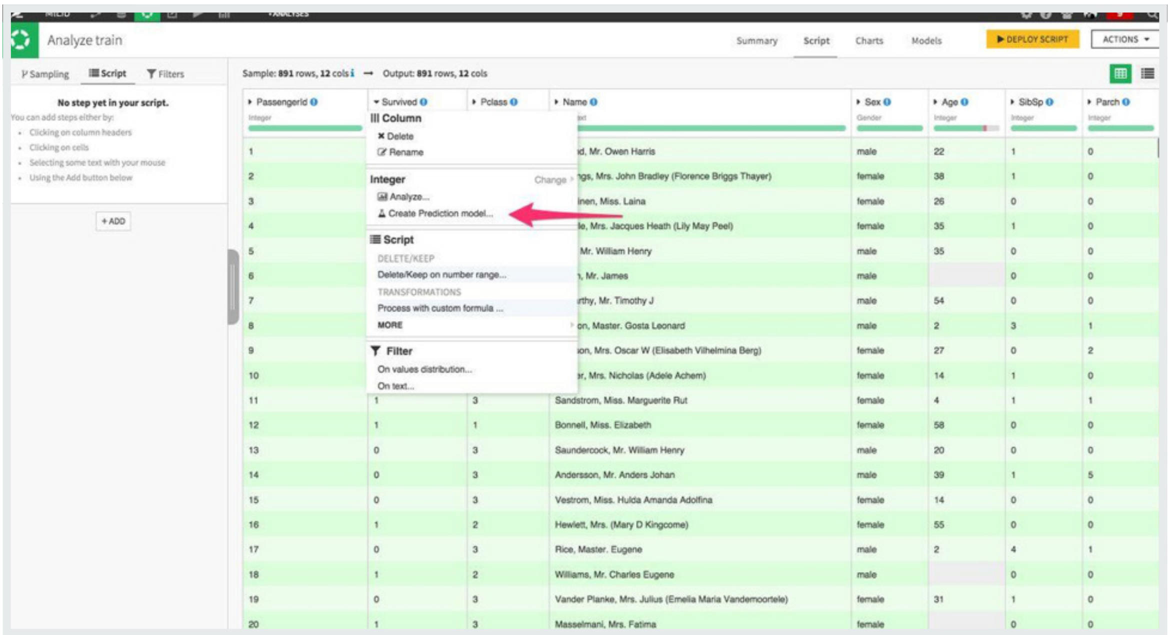
**Dataiku’s Data Science Studio (DSS) presents the following advantages:**

– Connectors to many types of databases. Data sources can be changed from a CSV file to a Hadoop File System (HDFS) Uniform Resource Identifier (URI) for instance, without having to change the rest of our ML pipeline

– Visual data wrangling, feature engineering and feature enrichment, to improve data and help prepare it before usage in learning algorithms

– Different ML back-ends providing built-in algorithms. The back-end can be changed from scikit-learn to Spark MLlib for instance, without having to change the rest of the ML pipeline

– Built-in clustering and anomaly detection algorithms



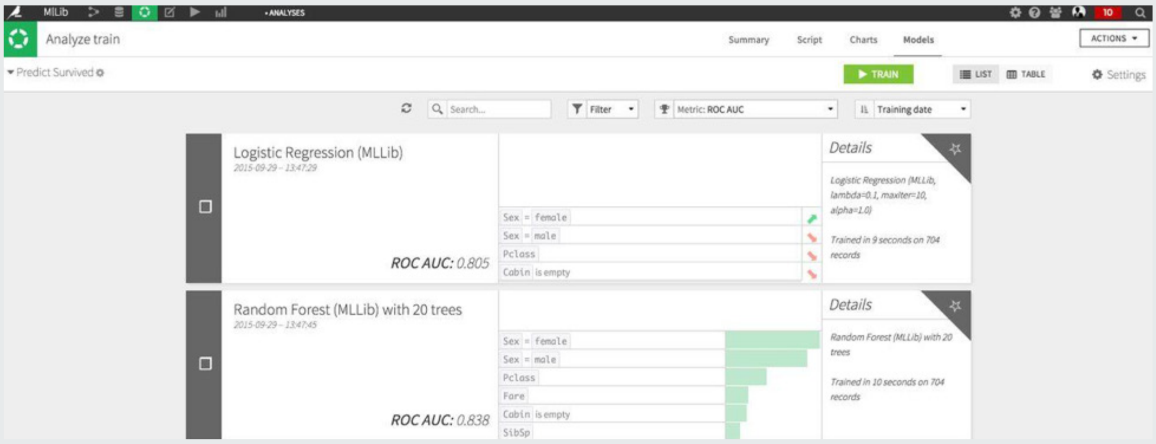
No step yet in your script.
You can add steps either by:
- Clicking on column headers
- Clicking on cells
- Selecting some text with your mouse
- Using the Add button below

Sample: 891 rows, 12 cols → Output: 891 rows, 12 cols

| PassengerId (Integer) | Survived | Pclass | Name (text) | Sex (Gender) | Age (Integer) | SibSp (Integer) | Parch (Integer) |
|---|---|---|---|---|---|---|---|
| 1 | | | d, Mr. Owen Harris | male | 22 | 1 | 0 |
| 2 | | | ngs, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 |
| 3 | | | inen, Miss. Laina | female | 26 | 0 | 0 |
| 4 | | | le, Mrs. Jacques Heath (Lily May Peel) | female | 35 | 1 | 0 |
| 5 | | | Mr. William Henry | male | 35 | 0 | 0 |
| 6 | | | , Mr. James | male | | 0 | 0 |
| 7 | | | rthy, Mr. Timothy J | male | 54 | 0 | 0 |
| 8 | | | on, Master. Gosta Leonard | male | 2 | 3 | 1 |
| 9 | | | on, Mrs. Oscar W (Elisabeth Vilhelmina Berg) | female | 27 | 0 | 2 |
| 10 | | | r, Mrs. Nicholas (Adele Achem) | female | 14 | 1 | 0 |
| 11 | 1 | 3 | Sandstrom, Miss. Marguerite Rut | female | 4 | 1 | 1 |
| 12 | 1 | 1 | Bonnell, Miss. Elizabeth | female | 58 | 0 | 0 |
| 13 | 0 | 3 | Saundercock, Mr. William Henry | male | 20 | 0 | 0 |
| 14 | 0 | 3 | Andersson, Mr. Anders Johan | male | 39 | 1 | 5 |
| 15 | 0 | 3 | Vestrom, Miss. Hulda Amanda Adolfina | female | 14 | 0 | 0 |
| 16 | 1 | 2 | Hewlett, Mrs. (Mary D Kingcome) | female | 55 | 0 | 0 |
| 17 | 0 | 3 | Rice, Master. Eugene | male | 2 | 4 | 1 |
| 18 | 1 | 2 | Williams, Mr. Charles Eugene | male | | 0 | 0 |
| 19 | 0 | 3 | Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele) | female | 31 | 1 | 0 |
| 20 | 1 | 3 | Masselmani, Mrs. Fatima | female | | 0 | 0 |

Training an MLlib model in Dataiku’s Data Science Studio[56]

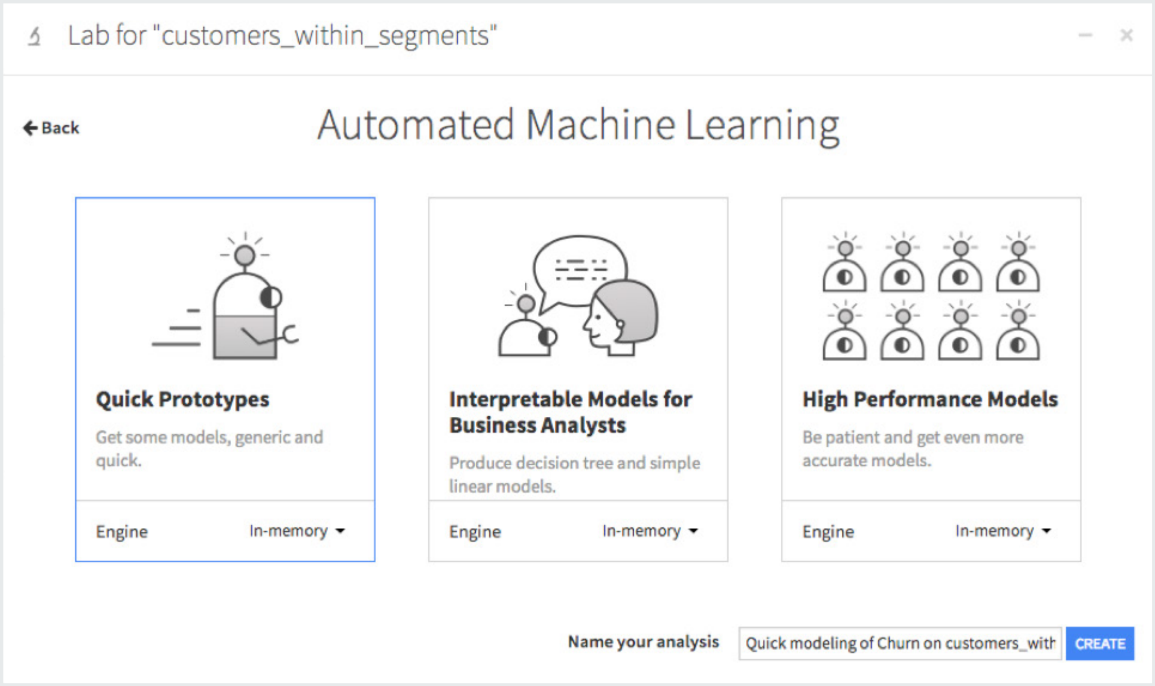
## 4. Machine learning development platforms



Logistic Regression (MLLib)
2015-09-29 – 13:47:29
ROC AUC: 0.805
Sex = female
Sex = male
Pclass
Cabin is empty
Logistic Regression (MLLib, lambda=0.1, maxiter=10, alpha=1.0)
Trained in 9 seconds on 704 records

Random Forest (MLLib) with 20 trees
2015-09-29 – 13:47:45
ROC AUC: 0.838
Sex = female
Sex = male
Pclass
Fare
Cabin is empty
SibSp
Random Forest (MLLib) with 20 trees
Trained in 10 seconds on 704 records

Comparing MLlib models’ performance in Dataiku’s Data Science Studio[57]


Lab for "customers_within_segments"
Back
Automated Machine Learning
Engine
In-memory
Engine
In-memory
Engine
In-memory
CREATE

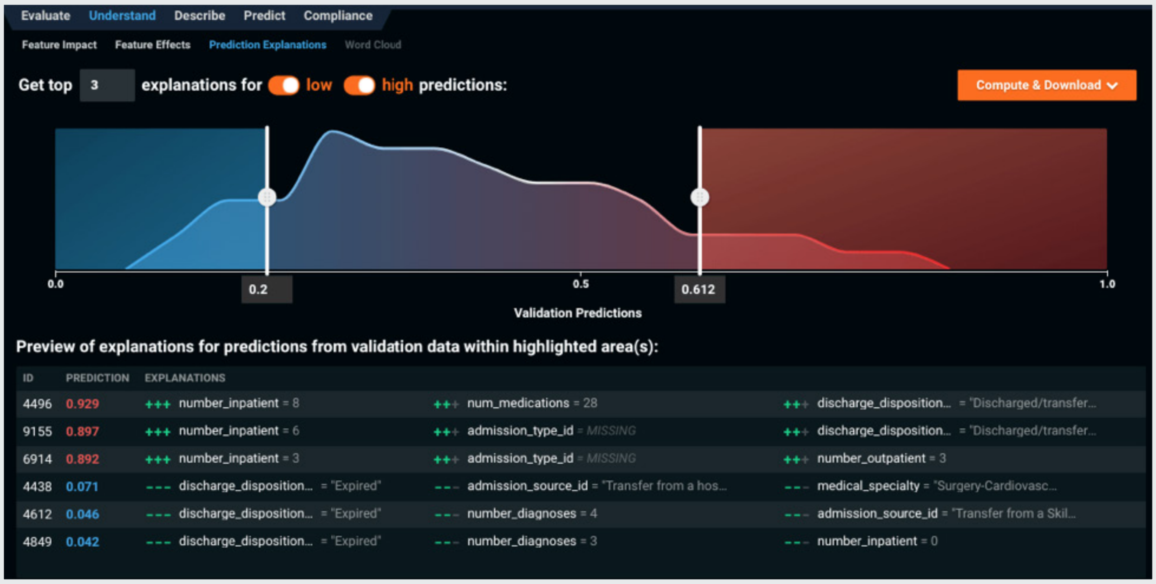
Quick Prototypes
Get some models, generic and quick.

Interpretable Models for Business Analysts
Produce decision tree and simple linear models.

High Performance Models
Be patient and get even more accurate models.

Name your analysis: Quick modeling of Churn on customers_with

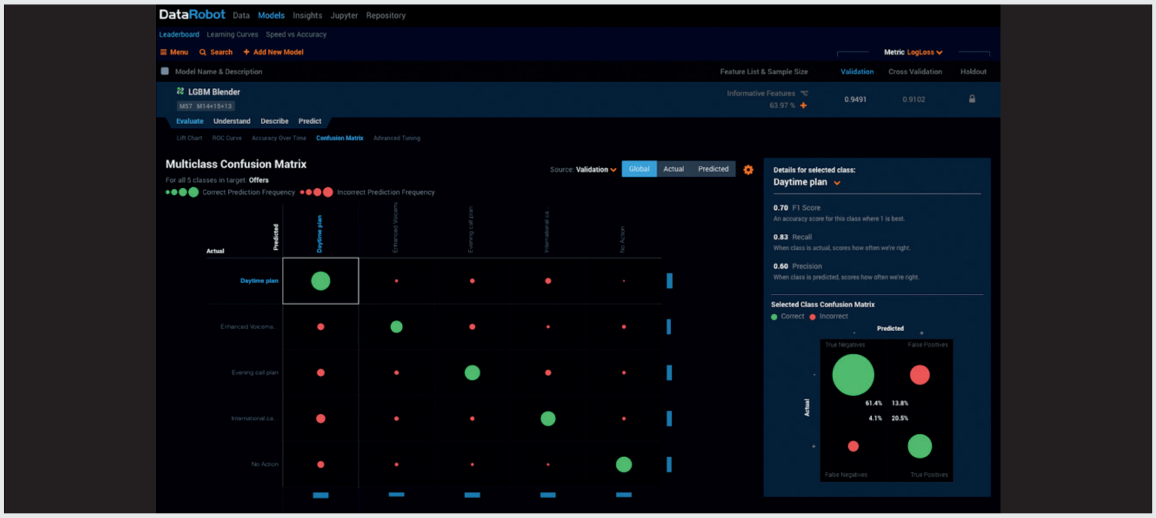
Visual AutoML in Dataiku’s Data Science Studio[58]



Preview of explanations for predictions from validation data within highlighted area(s):

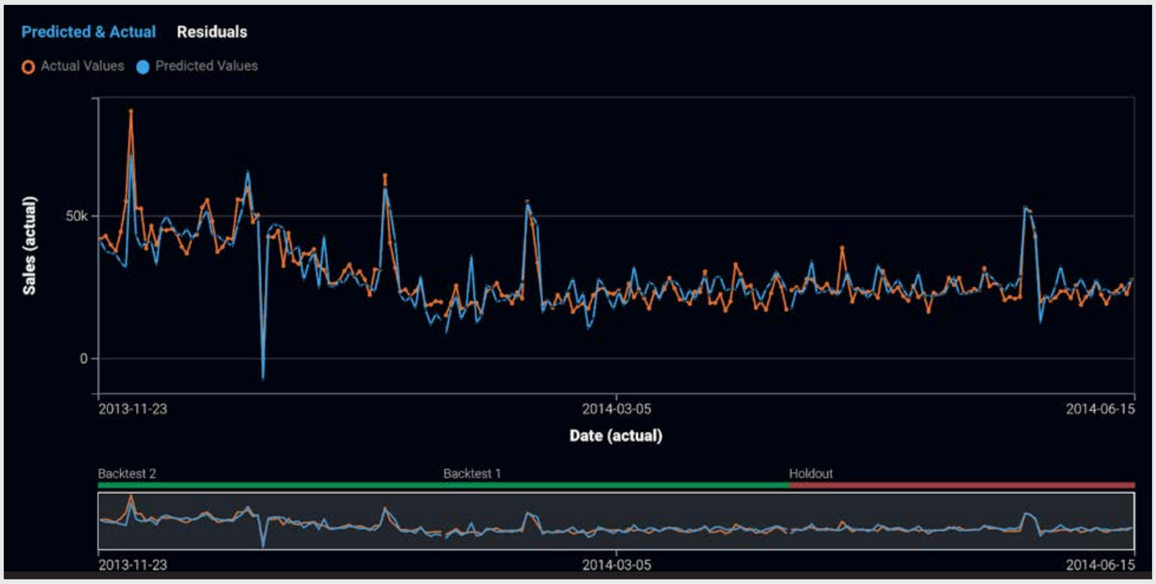
| ID | PREDICTION | EXPLANATIONS | | |
|---|---|---|---|---|
| 4496 | 0.929 | +++ number_inpatient = 8 | +++ num_medications = 28 | +++ discharge_disposition... = "Discharged/transfer... |
| 9155 | 0.897 | +++ number_inpatient = 6 | +++ admission_type_id = MISSING | +++ discharge_disposition... = "Discharged/transfer... |
| 6914 | 0.892 | +++ number_inpatient = 3 | +++ admission_type_id = MISSING | +++ number_outpatient = 3 |
| 4438 | 0.071 | --- discharge_disposition... = "Expired" | --- admission_source_id = "Transfer from a hos... | --- medical_specialty = "Surgery-Cardiovasc... |
| 4612 | 0.046 | --- discharge_disposition... = "Expired" | --- number_diagnoses = 4 | --- admission_source_id = "Transfer from a Skil... |
| 4849 | 0.042 | --- discharge_disposition... = "Expired" | --- number_diagnoses = 3 | --- number_inpatient = 0 |

Prediction explanations in DataRobot[59]



Confusion matrix in DataRobot[60]

## 4. Machine learning development platforms



Time series in DataRobot[61]

**DataRobot has the following advantages:**

- Automatic feature engineering
- Model inspection features and visualisations of prediction explanations
- Advanced time series forecasting: several built-in algorithms (ARIMA, Facebook Prophet, Gradient Boosting), backtesting evaluation method, automatic detection of stationarity, seasonality, and time series feature engineering

H2O's Driverless AI[62] is a similar product to DataRobot. Both have a Python API. Below is a call to the start_experiment_sync method of H2O's Python API, which is similar in spirit to the OptiML method of BigML's http API:

```
params = h2oai.get_experiment_tuning_suggestion(
         dataset_key=train.key,
         target_col=target,
         is_classification=True,
         is_time_series=False,
         config_overrides=None)
experiment = h2oai.start_experiment_sync(
         dataset_key=train.key,
         testset_key=test.key,
         target_col=target,
         is_classification=True,
         scorer='AUC',
         accuracy=params['accuracy'],
         time=params['time'],
         interpretability=params['interpretability'],
         enable_gpus=True,
         seed=1234, # for reproducibility
         cols_to_drop=['ID'])
```

# 4. Machine learning development platforms

## CLOUD MACHINE LEARNING INTEGRATED DEVELOPMENT ENVIRONMENTS (IDES)

The last class of model development platforms can be thought of as IDEs for machine learning, hosted on the cloud. The platforms do not offer the high-level functionality of the ML Studios previously presented, but they benefit from cloud computing. It is common practice among machine and deep learning practitioners to use powerful, GPU-equipped virtual machines (VMs), provided by cloud platforms, for their experiments.

These cloud VMs have been available before ML-specific cloud platforms were created. The platforms make it faster to run experiments, and easy to do 24/7 if desired. The platforms also make it possible to scale experiments with CPUs with many cores, powerful GPUs with a lot of RAM, and clusters that are already configured (for example, for distributed learning, tuning hyper-parameters in parallel, and using deep neural networks). Users only pay for what they use; there are no upfront costs to acquire costly hardware.

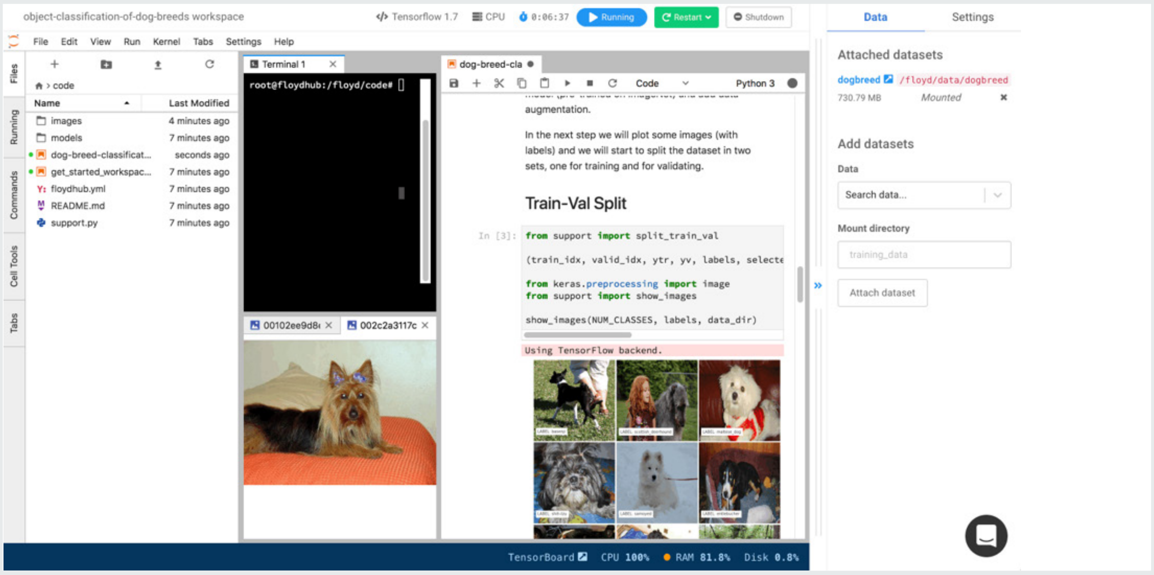
The new ML-specific cloud platforms give access to Jupyter[63] Lab environments, as web-based IDEs, on preconfigured infrastructure. They provide cloud VMs with all the common open source ML libraries. TensorBoard[64] web servers are usually included, for monitoring the progress of TensorFlow-based experiments. Cloud ML IDEs aim at making VMs available faster than other cloud services (typically from several minutes to seconds), and at making it more convenient to persist work done on these (short-lived) VMs.

Floyd[65] is a great platform to start with, as it is less complex to use than competitors, but still provides a few different options to run ML experiments. It gives access to two types of CPUs and two types of GPUs, to choose from depending on our needs. There are two different ways in which experiments can be run:

- Workspace, which is an IDE based on Jupyter Lab, for interactive experimentation. It also gives access to TensorBoard, and to a similar feature called Metrics, that can be used with any ML library (not necessarily TensorFlow) to monitor the progress of model training
- Jobs, for running longer experiments as scripts. Jobs are started from a command line interface (CLI) tool installed on our own machine, in the same way as executing a script locally, but they are run on the Floyd platform. The CLI also allows users to tag jobs and to download outputs. The ability to browse the history of jobs, to filter with tags, and to see result summaries in the history, fulfils the role of an experiment tracker

Another useful feature is the ability to store large datasets on the cloud platform and to share with the whole team, so there is no need to download the datasets locally and to keep them in sync.

The Floyd infrastructure is built on top of Amazon's North American public cloud, but Floyd can also be installed on private clouds and on-premise, which is one way it differentiates itself from Google and Amazon products. For European companies, Faculty.ai[66] provides an alternative based in the UK.



| Name | Last Modified |
|---|---|
| images | 4 minutes ago |
| models | 7 minutes ago |
| dog-breed-classificat... | seconds ago |
| get_started_workspac... | 7 minutes ago |
| floydhub.yml | 7 minutes ago |
| README.md | 7 minutes ago |
| support.py | 7 minutes ago |

```
root@floydhub:/floyd/code#
```

augmentation.

In the next step we will plot some images (with labels) and we will start to split the dataset in two sets, one for training and for validating.

Train-Val Split

```
In [3]: from support import split_train_val
        (train_idx, valid_idx, ytr, yv, labels, selecte
        from keras.preprocessing import image
        from support import show_images
        show_images(NUM_CLASSES, labels, data_dir)
```

```
Using TensorFlow backend.
```

Attached datasets

dogbreed /floyd/data/dogbreed
730.79 MB Mounted

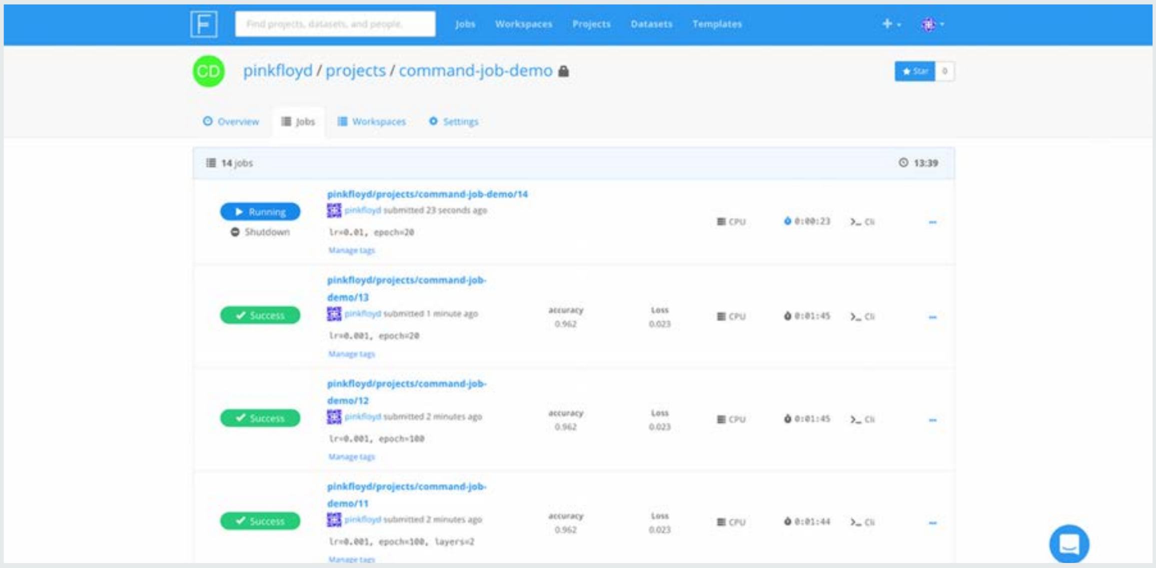
Add datasets

Data: Search data...

Mount directory: training_data

Attach dataset

Floyd Workspace running Jupyter Lab[67]



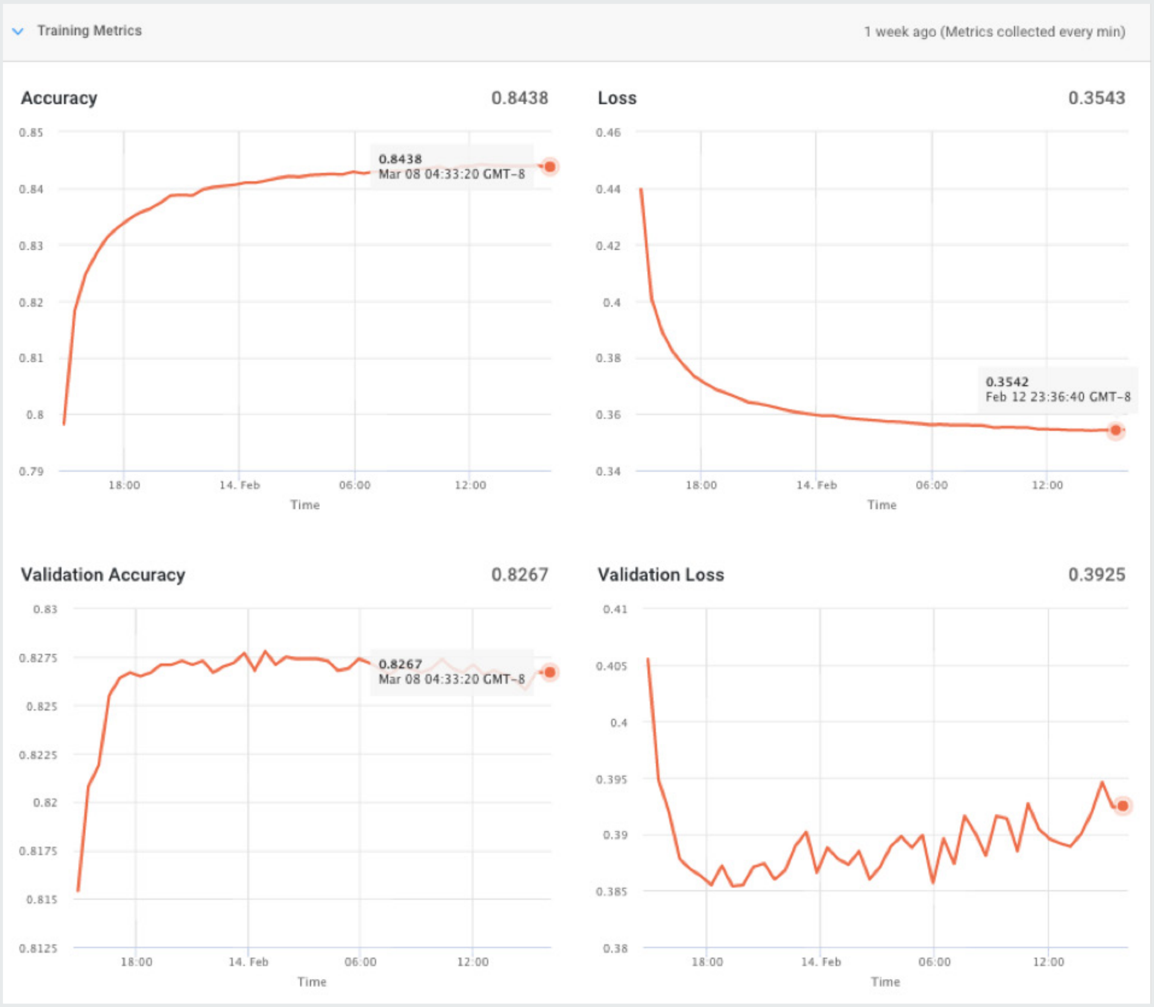
pinkfloyd / projects / command-job-demo

14 jobs 13:39

| | | | | | | |
|---|---|---|---|---|---|---|
| Running / Shutdown | pinkfloyd/projects/command-job-demo/14<br>pinkfloyd submitted 23 seconds ago<br>`lr=0.01, epoch=20`<br>Manage tags | | | CPU | 0:00:23 | CLI |
| Success | pinkfloyd/projects/command-job-demo/13<br>pinkfloyd submitted 1 minute ago<br>`lr=0.001, epoch=20`<br>Manage tags | accuracy 0.962 | Loss 0.023 | CPU | 0:01:45 | CLI |
| Success | pinkfloyd/projects/command-job-demo/12<br>pinkfloyd submitted 2 minutes ago<br>`lr=0.001, epoch=100`<br>Manage tags | accuracy 0.962 | Loss 0.023 | CPU | 0:01:45 | CLI |
| Success | pinkfloyd/projects/command-job-demo/11<br>pinkfloyd submitted 2 minutes ago<br>`lr=0.001, epoch=100, layers=2`<br>Manage tags | accuracy 0.962 | Loss 0.023 | CPU | 0:01:44 | CLI |

Floyd Jobs – Source[68]

## 4. Machine learning development platforms



Floyd Metrics – Source[69]

## 4. Machine learning development platforms

Google AI Platform Notebooks[70] is similar to Floyd Workspace. It uses the platform's Deep Learning VM Image[71] and its Cloud TPUs[72] (tensor processing unit). It has built-in Git support and integrates with Google AI Hub, a product that helps discovery of what others have built within the organisation (such as notebooks, pipelines and models) that one should check before starting a new development. Since Notebooks is a Google Cloud product, it gives access to pre-configured/pre-installed Google Cloud Platform libraries such as Dataflow and Dataproc for data wrangling. GPUs can be added to or removed from the cloud VMs used by the platform, whereas Floyd is limited to 1 GPU per VM.

SageMaker[73] is Amazon's ML platform, which offers a similar development environment and facilitates the use of cluster computing for distributed model training and distributed hyper-parameter tuning experiments. SageMaker allows users to define model training jobs via its Python API and a dictionary data structure where the datasets are defined to use for training and validation, hyper-parameter values for the training algorithm, and resources for running the job.

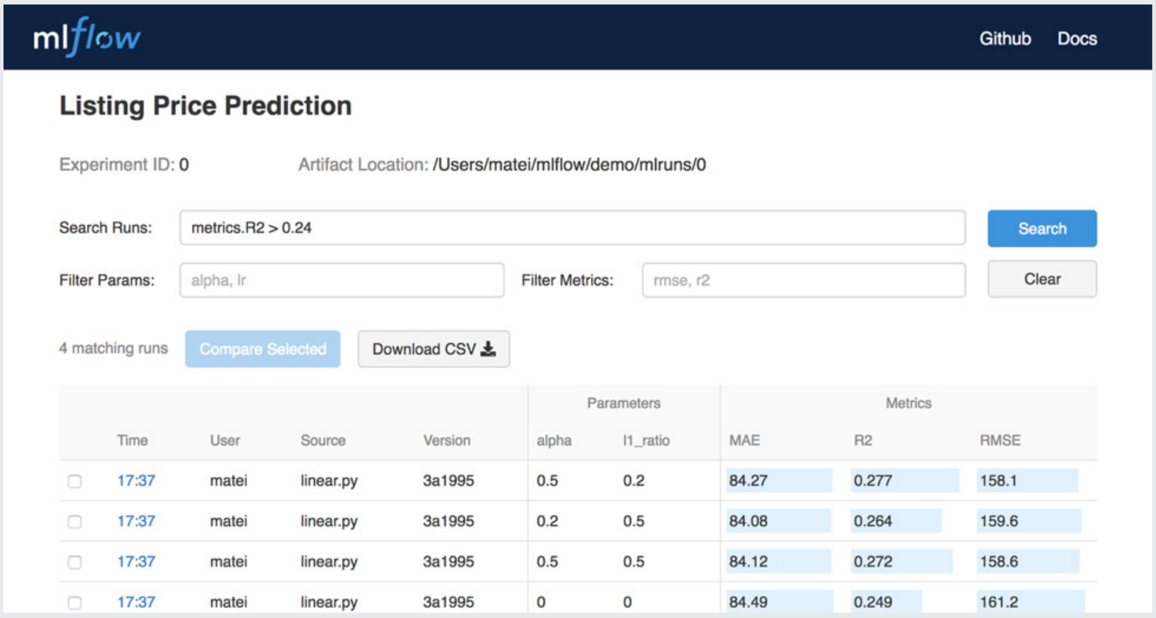
```
training_params = \
{
    "AlgorithmSpecification": {
        "TrainingImage": image, # specify the
training docker image
        "TrainingInputMode": "File"
    },
    "RoleArn": role,
    "OutputDataConfig": {
        "S3OutputPath": 's3://{}/{}/output'.
format(bucket)
    },
    "ResourceConfig": {
        "InstanceCount": 1,
        "InstanceType": "ml.p3.2xlarge",
        "VolumeSizeInGB": 50
    },
    "TrainingJobName": "model_a",
    "HyperParameters": {
        "image_shape": "3,224,224",
        "num_layers": "18",
        "num_training_samples": "15420",
        "num_classes": "257",
        "mini_batch_size": "128",
        "epochs": "2",
        "learning_rate": "0.2",
        "use_pretrained_model": "1"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 360000
    },
    "InputDataConfig": [
        {
          "ChannelName": "train",
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "S3Prefix",
               "S3Uri": 's3://{}/train/'.format(bucket),
                    "S3DataDistributionType":
"FullyReplicated"
                }
            },
            "ContentType": "application/x-recordio",
            "CompressionType": "None"
        },
        {
            "ChannelName": "validation",
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "S3Prefix",
                    "S3Uri": 's3://{}/validation/'.
format(bucket),
                    "S3DataDistributionType":
"FullyReplicated"
                }
            },
            "ContentType": "application/x-recordio",
            "CompressionType": "None"
        }
    ]
}
sagemaker.create_training_job(**training_params)
```

## 4. Machine learning development platforms

The platform also provides project templates for various ML tasks, including advanced ones such as sequence-to-sequence learning, and reinforcement learning.

Databricks Unified Analytics Platform[74] allows users to access cluster computing on Amazon or Microsoft's cloud platforms. Databricks maintains Apache Spark, a leading open source cluster-computing framework.

The platform also gives access to Databricks' other open source framework, MLflow, which has an experiment tracking component. MLflow is designed to scale to large data sets, large output files (for example, models), and large numbers of experiments. It supports launching multiple runs in parallel (for example, for hyperparameter tuning) and executing individual runs on Spark. It can take input from, and write output to, distributed storage systems.



Listing Price Prediction

Experiment ID: 0 Artifact Location: /Users/matei/mlflow/demo/mlruns/0

Search Runs: metrics.R2 > 0.24

Filter Params: alpha, lr Filter Metrics: rmse, r2

4 matching runs

| | | | | Parameters | | Metrics | | |
|---|---|---|---|---|---|---|---|---|
| Time | User | Source | Version | alpha | l1_ratio | MAE | R2 | RMSE |
| 17:37 | matei | linear.py | 3a1995 | 0.5 | 0.2 | 84.27 | 0.277 | 158.1 |
| 17:37 | matei | linear.py | 3a1995 | 0.2 | 0.5 | 84.08 | 0.264 | 159.6 |
| 17:37 | matei | linear.py | 3a1995 | 0.5 | 0.5 | 84.12 | 0.272 | 158.6 |
| 17:37 | matei | linear.py | 3a1995 | 0 | 0 | 84.49 | 0.249 | 161.2 |

Experiment tracking with Databrick's MLflow[75]

## ADDING FEATURISATION TO MACHINE LEARNING PIPELINES

There are cases where the full numerical representation of inputs is not readily available (as it would be for text or image inputs), but needs to be computed before it can be passed to a model. For a customer input, for instance, who would be identified by an id, some features would be already stored in database (for example, date of birth), and others would require computation. This would be the case for behavioural features that describe how the customer interacted with the product over a certain period of time: they would be computed by querying and aggregating data that logged customer interactions with the product. If, by nature, features do not change too often, they could be computed in batches. But in ML use cases such as UberEats' Expected Time of Delivery, we could have "hot" features that would change rapidly and would need to be computed in real-time; for instance, the aggregate delivery time of a given restaurant over the last X minutes. For simplicity, real-time featurisation for now has not been considered, focusing instead on batch featurisation.

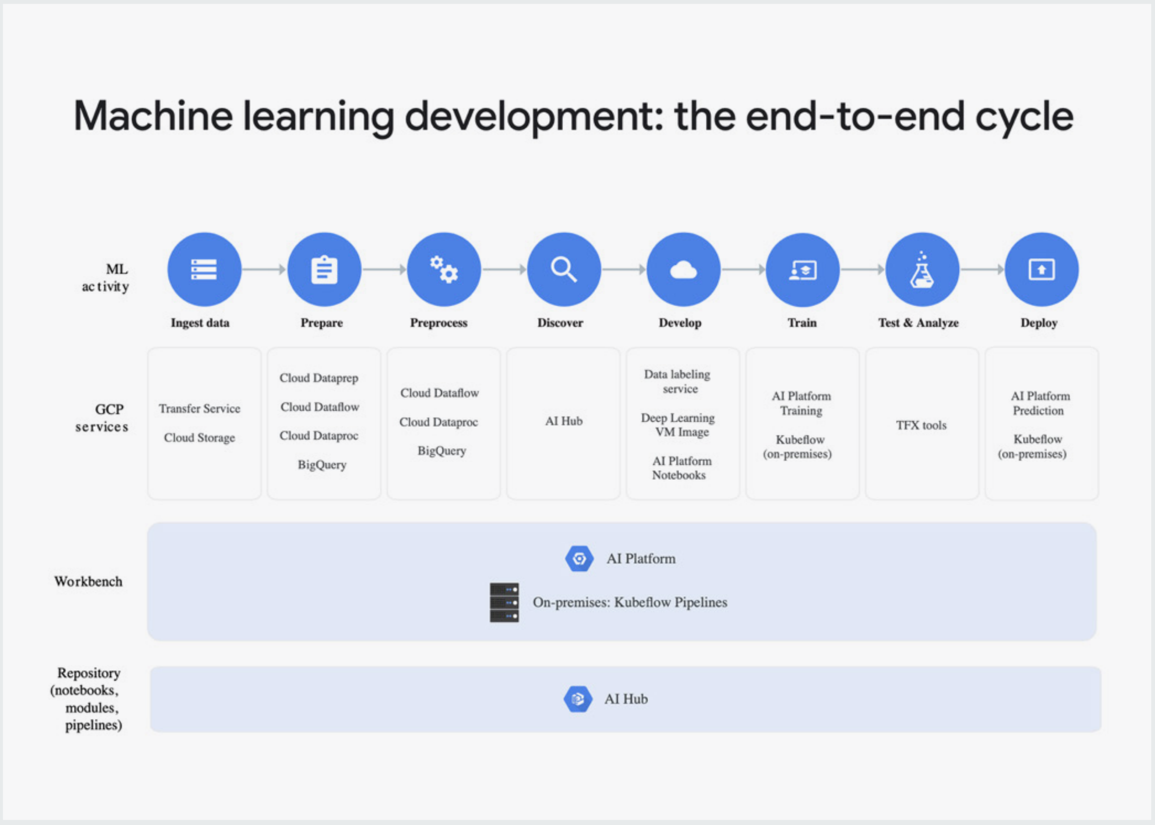
The pipeline is getting more complex, as it now involves a "featuriser" that accesses and queries various databases, and performs various aggregations and treatments on the queried data. Similarly to the other components of the ML pipeline, this featuriser has parameters (such as the number of minutes X in the example above), which have an impact on the performance of the whole pipeline. One difference with other data preparation components is that featurisation typically feeds from multiple data sources.

Featurising is typically done using different software libraries and computing environments than for pre-processing and modelling. When using platforms and software as a service, the featurisation could be done with a data wrangling tool such as Trifacta[76], and the modelling with one of the ML development platforms previously discussed.

Pipelines need to be executed by an "orchestrator", which would give commands to the featuriser and to the model builder, and connect the output of one to the input of the other. A common practice is to save featurised data to (and read from) Amazon S3, Google Cloud Storage, or a similar service. The orchestrator could be a human, but to update models frequently, or to tune (hyper-)parameters of the featuriser and of the modeller jointly, the orchestrator will need to be a computer program. This will make it possible to call the orchestrator programmatically, and to use a scheduler to update models at a certain frequency. Infrastructure is also needed to run the orchestrator. If we use our own, our ML solution will not be 100% as a service any more, even though it would be using platforms as a service for data wrangling and ML development.

An alternative is to use a single platform as a service to define, orchestrate and run full pipelines. Google AI Platform allows users to do that: creating a featuriser that uses Google Cloud data products such as Dataprep[77] (an integrated partner service provided by Trifacta), Dataflow[78] (a simplified stream and batch data processing tool), Dataproc (a managed way to run Spark) and BigQuery[79] (a serverless cloud data warehouse), and to define a training application[80] based on TensorFlow or built-in algorithms (for example XGBoost).

## 4. Machine learning development platforms



Google Cloud Platform, AI and machine learning products, AI Platform[81]

# 4. Machine learning development platforms

Another option to define, orchestrate and run full ML pipelines in the same environment, is to use Spark. It is a popular choice for processing important volumes of data, and it includes a machine learning library (MLlib).

ML pipelines can also be created with open source software exclusively, and run on cloud compute platforms. Steps of the pipeline could be defined and executed in different Docker containers (for example, the featurisation could be done with Pandas in one container, and the modelling with scikit-learn or TensorFlow in another container).

Kubernetes[82] is one of the most popular open source container orchestration systems among ML practitioners.

- Kubeflow[83] and Seldon Core[84] are open source tools that allow users to describe ML pipelines and turn them into Kubernetes clustered applications. This can be done in a local environment, and the application can be run on a Kubernetes cluster, which could be installed on-premise or provided in a cloud platform — Google Kubernetes Engine[85] for instance, which is used by Google AI Platform, or Azure Kubernetes Service[86], or Amazon EKS[87]. Amazon also provides an alternative to Kubernetes with Fargate[88] and ECS[89]
- Apache Airflow[90] is another open source workflow management tool, originally developed by Airbnb, which can serve as orchestrator of ML pipelines. Airflow has become a popular way to coordinate the execution of general IT tasks, including ML ones, and it also integrates with Kubernetes

## Contents

# 5. Machine learning deployment platforms

# 5. Machine learning deployment platforms

Assume that a model has been built and evaluated on machine A (using an ML development platform, or maybe an a programming environment on the individual's own machine). The next step is to use this model for predictions in a production app, or in a sandbox that mimics a production setting and allows to further test the model. This will be on a different machine, machine B.

Firstly, the two variants to packaging models so that they can be "moved" from A to B, when machine B is a server that we control, are outlined, followed by a discussion about another way that can be used when B is the end-user's device or a mobile device (in an IoT application).

Client/server architectures are the focus of the rest of this section. Deployment platforms include model servers that turn packaged models into production-ready http APIs. Presented here are two deployment case studies, with sample code: one where the Floyd platform is used with the Flask open source serving library to deploy a model to production, and the other one where the SageMaker platform is used to deploy and A/B test models in production. Both platforms provide cloud VMs that serve predictions. Another way to serve models, called "serverless", where VMs are abstracted away, is also introduced. Finally, this section presents a platform with interesting features for managing models in production: Seldon Deploy.

"It's hard to understate how nascent the field of production in machine learning is, and that means the tools supporting this ecosystem are only starting to be fully developed."[91]

## PACKAGING MODELS

### Saving model object and programming environment

One approach to packaging a model is to persist with the model object that lives in the memory of the programming environment that created it. This is usually quite straightforward. In Python for example, an object can be saved to a file by using the Pickle or the Joblib library. For neural networks, the preferred method consists in saving the model structure to JSON format, and its weights to an HDF file.

Loading these files on machine B requires that A and B both use the same libraries. One way to ensure that is to create a requirements file, which simply lists the names and versions of all the libraries in machine A's Python environment,
and to use this file to recreate the same environment on machine B. This is the approach used by MLflow's Pyfunc format[92] for model persistence, which allows users to load models created with H2O, Keras, PyTorch, Scikit-learn, MLlib or TensorFlow, and to package them for deployment to Microsoft Azure ML or Amazon SageMaker.

Another, more general approach, is to use a programming environment that lives in a Docker container on machine A. Thus a model would be "packaged" by saving it to file(s) in the same way as discussed before, along with the Docker image associated to the Docker container. This would capture non-Python dependencies, such as Java libraries. There would be no need to recreate the environment on machine B, as it would already exist in the Docker image.

However, these solutions assume that organisations will be making predictions on a machine that can run Python, or Docker. What if predictions need to be made on a small device where we cannot expect this to be true — on a smartphone for instance, or on an IoT device?

# 5. Machine learning deployment platforms

**Exporting neural network models for client-side (a.k.a. on-device) predictions**
Apple uses the Core ML format to load neural network models in iOS applications. TensorFlow has its own SavedModel format, and the "Lite" version of the framework may be installed on a small device. Amazon SageMaker Neo[93] is a service and open source project that can ingest models from many different ML libraries, and that allows them to be optimised them for the specific processor on which predictions will be run, or to be implement on a field-programmable gate array (FPGA).

AWS IoT Greengrass[94] is another Amazon platform, which is useful for the deployment of models to the edge, for example, for pushing new models to a fleet of IoT devices, so they can be used locally (and without internet connectivity). Model versions can be easily updated or rolled back. Microsoft IoT Edge[95] has a similar offering.

**Cross-framework model export formats**
Some of the most common ML model types can be exported to cross-framework, open formats such as PMML[96] and PFA[97]. One use case would be to save a Scikit-learn model to PMML and to load it in Spark to perform batch predictions on clusters of machines (Spark is particularly efficient at processing large volumes of data in parallel, which could be the case if predictions on many inputs are requested in one go).

ONNX[98] is a cross-framework format for neural networks. One potential use case would be to experiment with PyTorch on machine A, and to make predictions with TensorFlow Lite (on mobile or embedded devices) or TensorFlow.js (in the browser) on machine B.

## CASE STUDY: SERVING PREDICTIONS WITH FLASK ON FLOYD

For most applications, the best practice to integrate predictions is to use a client/server architecture, where predictions are computed on dedicated infrastructure and are served via an http API. This allows separation of concerns, and in practice, it makes it possible for data scientists to use the best language and tools for their ML needs, and app developers to use the best language and tools to create their application. Platforms can be useful to provide infrastructure to run predictions, at scale.

Some of the platforms mentioned for ML development provide model serving options. Floyd Serve uses the Flask open source API server, which allows to run arbitrary Python code when API requests are received. Here is an example Flask app, whose code would be written in an app.py file and would load a Scikit-learn model saved with Joblib:

```
from flask import Flask, request, jsonify
from joblib import load
app = Flask(__name__)
model = load('models/model1.joblib')
@app.route('/', methods=['POST'])
def predict(path):
    text=request.get_json()['input']
    return jsonify(input=text, prediction=model.
predict_proba([text]))
```

An instruction is then sent to Floyd to serve a new API, from the CLI:

```
$ floyd run --mode serve
```

This would upload the contents of the local directory to Floyd (code, Python requirements file, model files), which would start and provide a VM, then start a Flask application based on the contents of app.py. VM uptime is then paid for.

Note that BigML also allows to "APIfy" any custom function, as long as it is written in WhizzML.

**CASE STUDY: A/B TESTING MODELS IN PRODUCTION WITH AMAZON SAGEMAKER**

Amazon SageMaker can also directly make models available as http APIs. It is actually possible to attach several models to the same API endpoint, with different weights and different instance configurations. This allows the user to implement strategies that test and compare different models in production: canary testing, to validate on a small fraction of end-users that a new model does not have a negative impact, and A/B testing, to see which out of two models has the most positive impact. It also allows for fast model update and rollback.

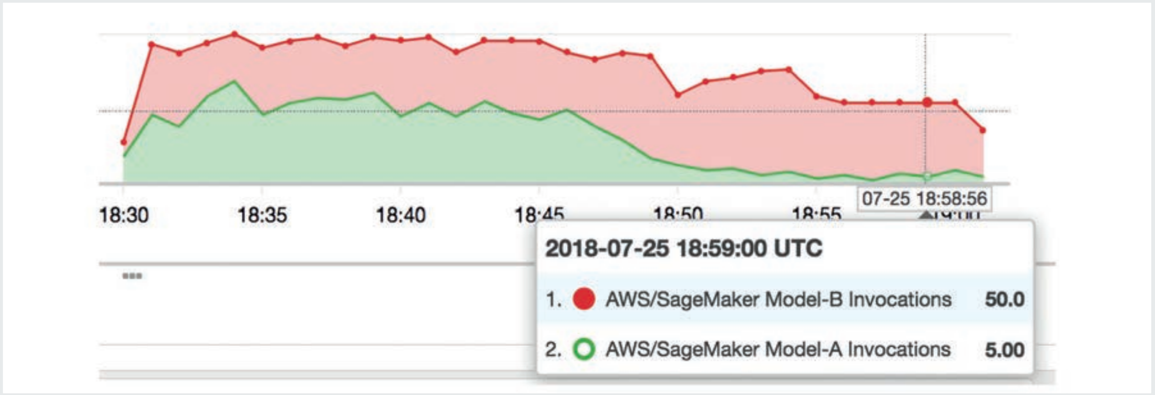
To pick up from the model_a training job that was defined in section 4:

```
info = sagemaker.describe_training_
job(TrainingJobName='model_a')
sagemaker.create_model(ModelName='model_a',
   ExecutionRoleArn=sagemaker.get_execution_role(),
   PrimaryContainer={'Image': image, # docker image
   'ModelDataUrl': info['ModelArtifacts']
['S3ModelArtifacts']})
```

Assuming that model_b has been defined similarly, an endpoint configuration can be created with two production variants (A and B) that have equal weights[99]:

```
endpoint_config_response = sagemaker.create_
endpoint_config(
    EndpointConfigName = 'my_endpoint',
    ProductionVariants = [
        {
        'InstanceType': 'ml.m4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': 'model_a',
        'VariantName': 'Model-A',
        'InitialVariantWeight': 1
        },
        {
        'InstanceType': 'ml.m4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': 'model_b',
        'VariantName': 'Model-B',
        'InitialVariantWeight': 1
        }
    ])
```

CloudWatch[100] provides API metrics out-of-the-box. The weights defined above can be updated programmatically (UpdateEndpointWeightAndCapacities[101] API) or via the AWS Console. CloudWatch would allow us to monitor how traffic gets routed.



2018-07-25 18:59:00 UTC

1. ● AWS/SageMaker Model-B Invocations 50.0

2. ○ AWS/SageMaker Model-A Invocations 5.00

Evolution of traffic when changing B's weight to 9 (instead of 1), monitored in CloudWatch[102]

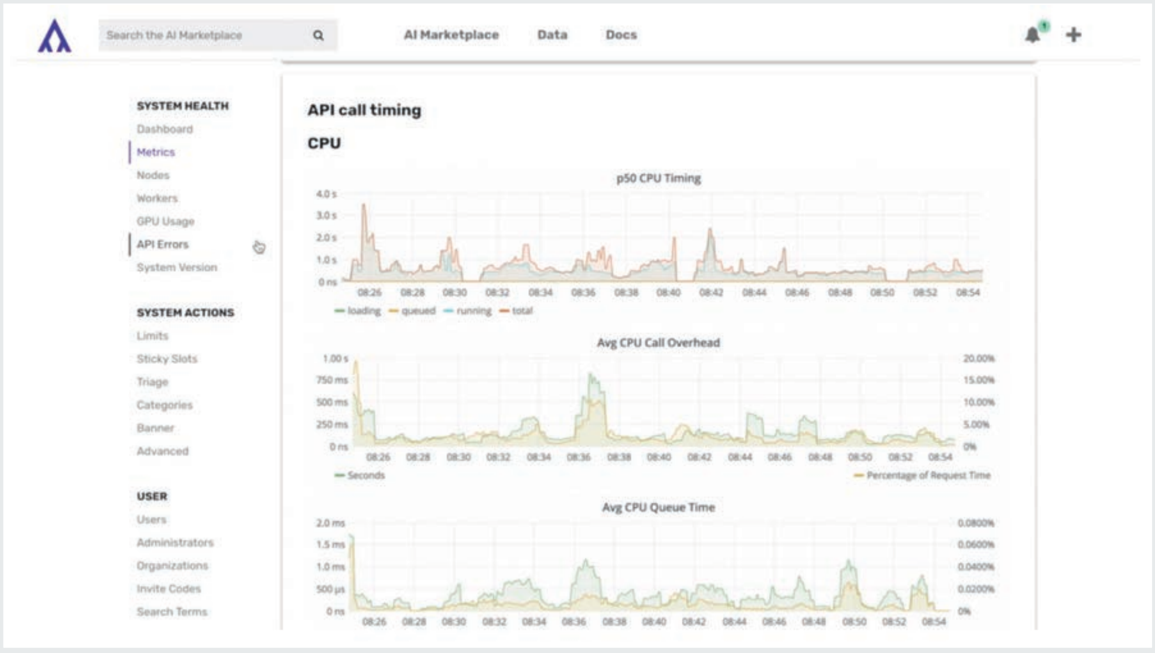
## SERVERLESS MODEL SERVING PLATFORMS

Algorithmia[103] is a cloud platform focused on model serving, that one may call "serverless": it focuses on turning code into APIs and hides away the fact that the code is run on server VMs (which requires scaling of VMs up and down automatically, based on traffic).

This translates into a business model where you pay for compute time, and not for VM uptime.

Here are some other advantages of the Algorithmia platform:

- API monitoring (similar to CloudWatch)
- The "Marketplace", where you can find pre-trained models made by others (for example sentiment analysis), or sell access to your own models
- The "ML portfolio", which is a catalogue of all your models that makes it easier to make models available to everyone in your organisation. This is interesting for organisations that have many model authors who produce many models: it allows users to keep track of where and when they are being used, and allows tagging, categorisation, and search. The ML portfolio is similar in spirit to Google's AI Hub



API monitoring on Algorithmia[104]

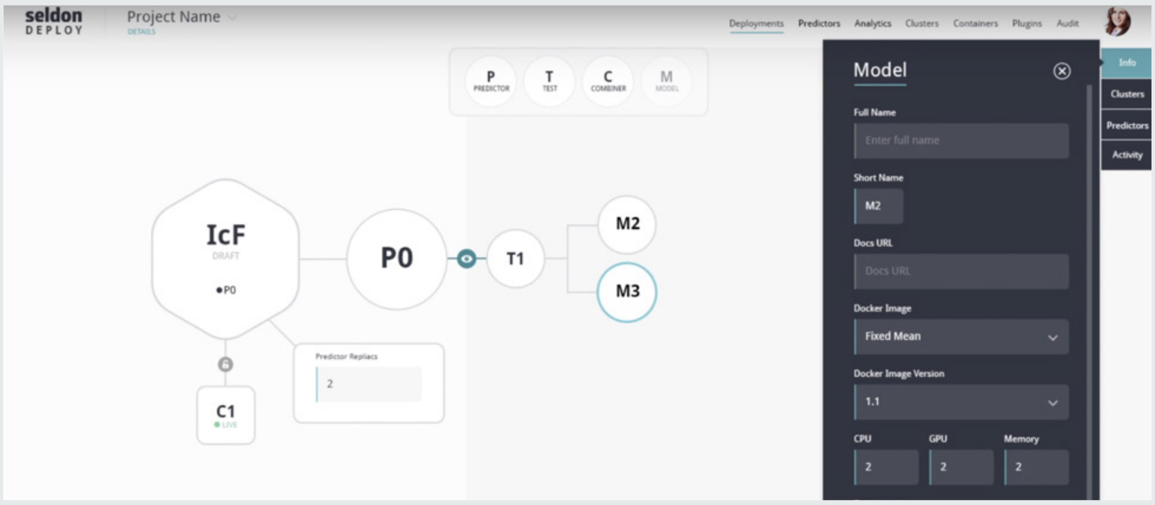
Amazon Lambda[105] and Google Cloud Functions[106] are two other serverless platforms, which turn functions into auto-scaling APIs. The process remains the same: code and dependencies are packaged into a .zip archive, with a single entry point function. Lambda applications can be written with Chalice, a Python library that feels very similar to Flask. Note that these platforms can have limitations on the size of the archive, duration of function execution and available memory. They can also be limited to CPU usage.

Google Cloud Run[107] is a serverless containers as a service product that removes some of the limitations found in Google Cloud Functions. It supports any language and libraries as long as developers can package the prediction app into a Docker image for Linux (x86-64), and GPUs are available. Cloud Run is built on the Knative[108] open-source project, enabling portability across platforms. Knative uses clusters of Docker containers that are orchestrated by Kubernetes. The product is not ML-specific, but makes it very easy to set up prediction APIs that auto-scale and has the advantage of charging only for time spent processing requests.

### EXAMPLE MODEL MANAGEMENT PLATFORM: SELDON DEPLOY

Seldon Deploy[109] is a serving platform in private beta, with a user interface that makes it easier to manage several models in production, to test and to update them. It offers A/B testing, model ensembling, and it implements a multi-armed bandit algorithm that intelligently directs traffic to models: it learns about each model's performance in production and focuses on the most promising ones (exploration/exploitation trade-off). Seldon Deploy also gives access to audit trails (saving which model a prediction came from), model explanations, outlier detection (to trigger fallback modes), and bias detection.

## 5. Machine learning deployment platforms



Predictor Replicas: 2

Model

Full Name: Enter full name

Short Name: M2

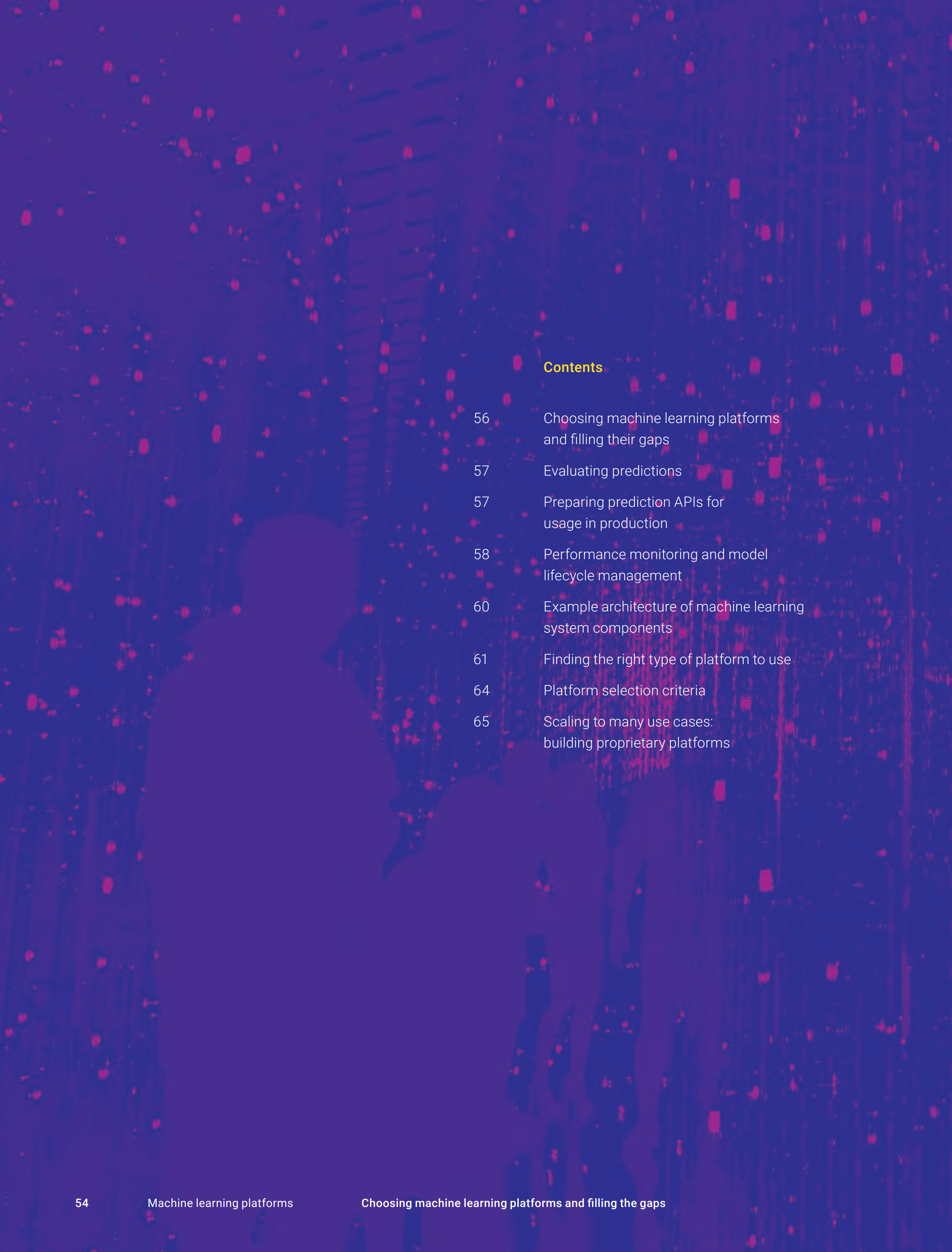
Docs URL: Docs URL

Docker Image: Fixed Mean

Docker Image Version: 1.1

| CPU | GPU | Memory |
|---|---|---|
| 2 | 2 | 2 |

Defining models to test in production in Seldon Deploy[110]



## Contents

# 6. Choosing machine learning platforms and filling the gaps

# 6. Choosing machine learning platforms and filling the gaps

Whatever type of ML platform is chosen, it's important to start with a definition of how to evaluate the planned ML system. As well as measuring prediction accuracy, evaluation of short-term and long-term impact via application-specific performance metrics, and system metrics such as lag and throughput is desirable. This will allow comparison of predictive models, but also to continuously monitor performance in production, in order to check if the models behave well through time and that they keep having a positive impact on the application.

Assuming one or several (baseline) models are already available as APIs, but that they have not yet been integrated into the application, this report will discuss how they can be evaluated on a test dataset, how to prepare their usage in production with the creation of a 'front-end', and how to implement a model performance 'monitor' with the creation of a database for production data, a 'ground truth' API, and a dashboard.

We would monitor models' performance on production data, and decide whether to integrate one in the application accordingly. When a new model version becomes available, API traffic would be moved progressively to the new model's API, via the front-end. This would be done for an increasing number of end-users, while monitoring performance and checking that the new model is not breaking anything.

The report moves on to present a recap of all the components to be found in real-world ML systems, via an example architecture that shows how they are connected. Ready-made ML platforms do not systematically provide all of the components needed in real-world ML systems (such as front-end and monitor). However, these platforms can greatly accelerate the creation of prediction APIs (or of packaged models that are ready to be deployed as APIs). Advice is provided to identify the right type of platform to use, based on the nature of the ML use case, and what to complement it with. The report also provides some selection criteria to help determine the best platform for your organisation.

Finally, the report discusses how to scale your usage of ML to many use cases, and how to streamline the creation of several new ML systems by building your own platform.

## EVALUATING PREDICTIONS

There are two important objectives behind model evaluation: comparing models, and deciding whether it is safe to integrate a model into an application.

Evaluation can be performed on a predetermined set of test cases, for which it is known what the prediction should be — called the 'ground truth'. For each test case, the error between the prediction and the ground truth is computed. Individual predictions, or the error distribution, could also be examined, or errors could be aggregated. This can be done in an evaluation program that has access to the test set's ground truth, takes a predictions file in input, and returns performance metrics based on aggregates of the prediction errors.

The evaluation program can be used on predictions made by a baseline model, to provide a reference. Baseline models are usually heuristics that are based on input characteristics — for example, features. They can be hand-crafted rule sets; for instance, a simple heuristic for churn prediction would be to say that if a customer logged in less than three times in the last 30 days, they are likely to churn.

The next step towards deciding if a model can be integrated into an application is to use the model's API on the inputs encountered in production (called 'production data'), in a production-like setting. This could be done with the API that exposes our baseline model, but it requires some preparation first.

## PREPARING PREDICTION APIS FOR USAGE IN PRODUCTION

When designing a prediction API, the decision needs to be made as to what the API should take as input. For example, when making predictions about customers, should the input be the full feature representation of the customer or just the customer id? In the latter case, it would need to create a featurisation microservice that would extract information about inputs based on their ids.

It is then fairly easy to deploy a (baseline) model as an API with a serverless platform that runs on the cloud or on a local computer.

### Refining the API exposed to the client via a front-end

It would be useful to create a front-end microservice, on top of this API, that would send its inputs and outputs for storage in a database. If the baseline is a hand-crafted decision tree, the path through the tree could also be saved, as an explanation of the baseline prediction. The front-end could refine its output by providing an anomaly score for the input (via a simple anomaly detector such as an isolation forest), which would allow the app that calls the front-end to trigger a fallback mode when encountering anomalous inputs.

When a new model trained with a machine learning algorithm would be available, the front-end could query predictions from this new model, use a blackbox model explainer (for example, SHAP or Influence Functions) to provide a prediction explanation similar to those of Indico's API[111], and send results for storage to the production database. However, the front-end would need to keep returning predictions made by the old model.

## PERFORMANCE MONITORING AND MODEL LIFECYCLE MANAGEMENT

**Storing production inputs and predictions for several model versions, via the front-end**
Evaluations can be taken further by testing and comparing models on production data, and checking the behaviour of models through time. Inputs would be saved to database by the front-end along with predicted outputs. We would need to also obtain ground truths, in order to compute prediction errors and performance metrics. There are two ways this can be done:

– Observing user behaviour: the app developer observes whether a past prediction was correct or incorrect, or the amount of error, by observing user behaviour. For example, for churn prediction, they would observe that the user proceeded to cancel their subscription; for price prediction, they would observe transactions on a marketplace. Also, some apps use feedback buttons to invite end-users to share if a prediction was correct or not (Google Mail's Priority Inbox for example).

– Provide ground truths: examine a subset of production data (this would be the case for spam detection for instance) to establish ground truths. There are dedicated services for outsourcing this (for example, Figure Eight and Google Data Labeling Service).

This evaluation on production data should be done continuously. The front-end would return only one prediction, but it could send production inputs to several models to be tested, which would allow performance metrics to be plotted over time for all of these models.

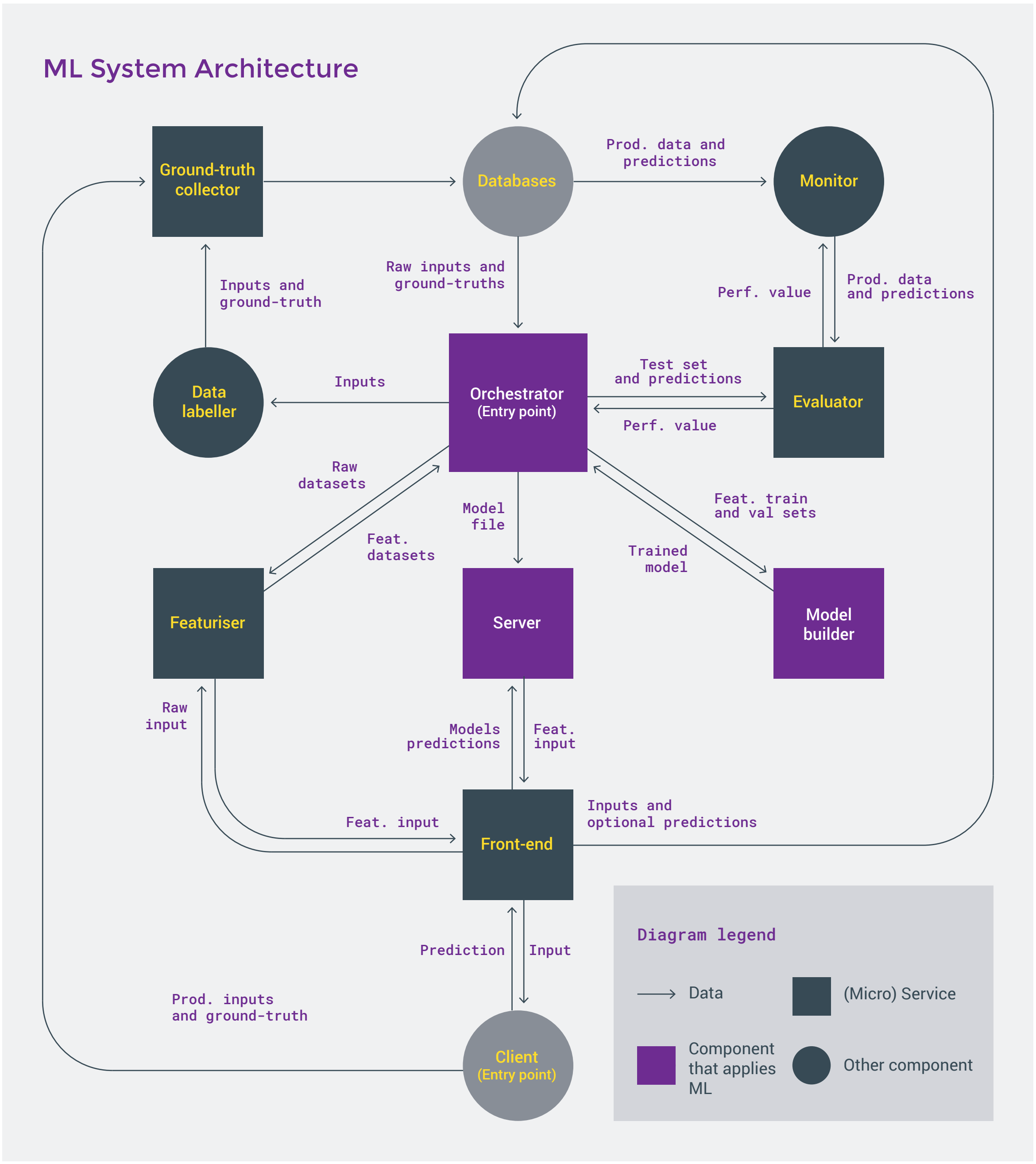
**Ground truth API and performance monitor**
Computing and monitoring performance metrics requires a database to store production inputs, predictions and ground truths, and an API for developers to send ground truths. The performance monitor would consist of a program that reads from that database, and a dashboard that shows the metrics computed by that program.

It could be augmented with a data monitor that has visualisation widgets to view distributions on production data — to make sure they are as expected, or to detect drift.

In addition to monitoring, the ground truth API can be useful to provide new training data to update models. Vertical and semi-specialised platforms usually provide this, and they may provide performance monitoring. However, it is recommended to build your own ground truth API and monitor, so as to compare competing platforms on production data.

**Testing new models on production inputs**
If a new candidate model seems to be performing better than the current one, it is possible to test its actual impact on the application by having the front-end return this model's predictions for a small fraction of our application's end-users (canary testing). This requires implementation of application-specific performance metrics. Test users could be taken from a list, or they could be chosen by one of their attributes, by their geolocation, or purely randomly. When monitoring performance and getting confident that the new model is not breaking anything, developers can gradually increase the proportion of test users and perform an A/B test to further compare the new model and the old model. If the new model is confirmed to be better, the front-end would simply "replace" the old model by always returning the new model's prediction. If the new model ends up breaking things, it is also possible to implement rollback via the front-end.

This report saw in Section 5 that canary and A/B testing could be done with SageMaker or Seldon Deploy. TensorFlow Serving[112] is an open source model server library that can be used in Kubeflow pipelines and that provides part of a solution, with the ability to assign labels to model versions (for example 'canary' and 'stable'). However, traffic still needs to be directed to them, which can be done by the front-end micro-service.


ML System Architecture
Ground-truth collector
Databases
Prod. data and predictions
Monitor
Inputs and ground-truth
Raw inputs and ground-truths
Perf. value
Prod. data and predictions
Data labeller
Inputs
Orchestrator (Entry point)
Test set and predictions
Perf. value
Evaluator
Raw datasets
Model file
Feat. datasets
Feat. train and val sets
Trained model
Featuriser
Server
Model builder
Raw input
Models predictions
Feat. input
Feat. input
Inputs and optional predictions
Front-end
Prediction
Input
Prod. inputs and ground-truth
Client (Entry point)
Diagram legend
Data
(Micro) Service
Component that applies ML
Other component

## EXAMPLE ARCHITECTURE OF MACHINE LEARNING SYSTEM COMPONENTS

The following diagram is proposed to visualise the components of a typical ML system and how they are connected to each other.

The components in light grey would already exist prior to the creation of the ML system. The others, in dark grey, are new components to be built. Those that apply ML models are represented in purple.
The squares are used to represent components that are expected to provide micro-services, accessed via representational state transfer (REST) APIs, that would typically run on serverless platforms.

There are two 'entry points' to the ML system: the client requesting prediction(s), and the orchestrator creating/updating models. The client represents the application used by the end-user who will be impacted by the ML system. The orchestrator would usually be a program called by a scheduler — so that models could be updated periodically, for example every week — or called via an API so that it could be part of a continuous integration / continuous delivery pipeline.

A more detailed view of the steps in the ML pipeline executed by the orchestrator:

- Extract - transform - load and split (raw) data into training, validation, test sets
- Send training/validation/test sets for featurisation (if any)
- Prepare featurised training/validation/test sets
  - Augment training data (for example rotate/flip/crop images)
  - Pre-process training/validation/test sets
    - Data cleaning/sanitisation (so that it can be safely used for modelling or predicting)
    - Problem-specific preparation (for example, de-saturate and resize images)
- Send URIs of prepared train/validation sets, along with metric to optimise, to model builder
- Get optimal model, apply to test set, and send predictions to evaluator
- Get performance value and decide if model is OK to be pushed to server (for canary-testing on production data, for instance)

The model builder is in charge of providing an optimal model. For this, it trains various models on the training set and evaluates them on the validation set, with the given metric, in order to assess optimality. Note that this is identical to the OptiML example explored earlier in this report; BigML automatically makes the model available via its API, but with other platforms we would typically package the model, save it as a file, and have the server load that file. The front-end requests predictions from all models that were successfully tested by the orchestrator, when it receives new production inputs, and it decides which prediction to return to the client.

The owner of the ML system and the owner of the client application would be accessing the monitor's user interface and dashboard on a regular basis.

Domain experts may be expected to access a data labeller, where they would be shown inputs and would be asked to label them. These labels would be stored in a database, and would then be available to the orchestrator for usage in training/validation/test data. The choice of which inputs to present for labelling could be made manually, or programmed in the orchestrator (for instance, looking at production inputs where the model was the least confident).

# 6. Choosing machine learning platforms and filling the gaps

## FINDING THE RIGHT TYPE OF PLATFORM TO USE

The choice of the best platform to create a prediction API to integrate into an application depends on the nature of the ML use case. Are other organisations likely to have the same prediction problem? Does the resolution of this problem require data that is unique to the company? If not (for example, language detection), one should turn to pre-trained models as a service. Otherwise, if tackling an industry-wide problem (for example, churn prediction), a vertical platform may provide a solution from which custom models can be created, once data sources have been connected.

If a vertical platform cannot be found for the intended ML use case, start by considering the nature of the inputs to the ML problem:

– If dealing with image or text inputs, it may be enough to use language or vision platforms as long as the nature of inputs treated by the platform's pre-trained models is similar in nature

  – A data collection strategy, a data augmentation and processing pipeline based on the knowledge of the application domain and of the problem should be implemented. This 'data-preparation pipeline' would be built outside of the language/vision platform, and executed before sending data to it. The combination of a problem-specific data-preparation pipeline and of a semi-specialised ML platform would constitute a vertical ML platform. Besides, data labelling platforms could also be helpful to collect output data

  – In the case where all the data cannot be collected for the ML use case, but that data lives on the end-users' devices, client-side model training will be needed. Examples include sensitive data such as fingerprint data (used by Touch ID on iOS) and text messages data (used by Smart Reply on Android). There are a growing number of open source libraries that allow for federated learning and privacy-preserving ML, such as TensorFlow Privacy[113], TensorFlow Federated[11]4 and PySyft[11]5, but none of the platforms that we are aware of directly leverages these technologies. They might however be used in the code that is deployed via edge platforms

– When dealing with input representations that are unique to an organisation (for example, variables whose meaning only makes sense for the company), a modelling pipeline will need to be built, in addition to a data-preparation pipeline as discussed above. This can be done with an ML development platform

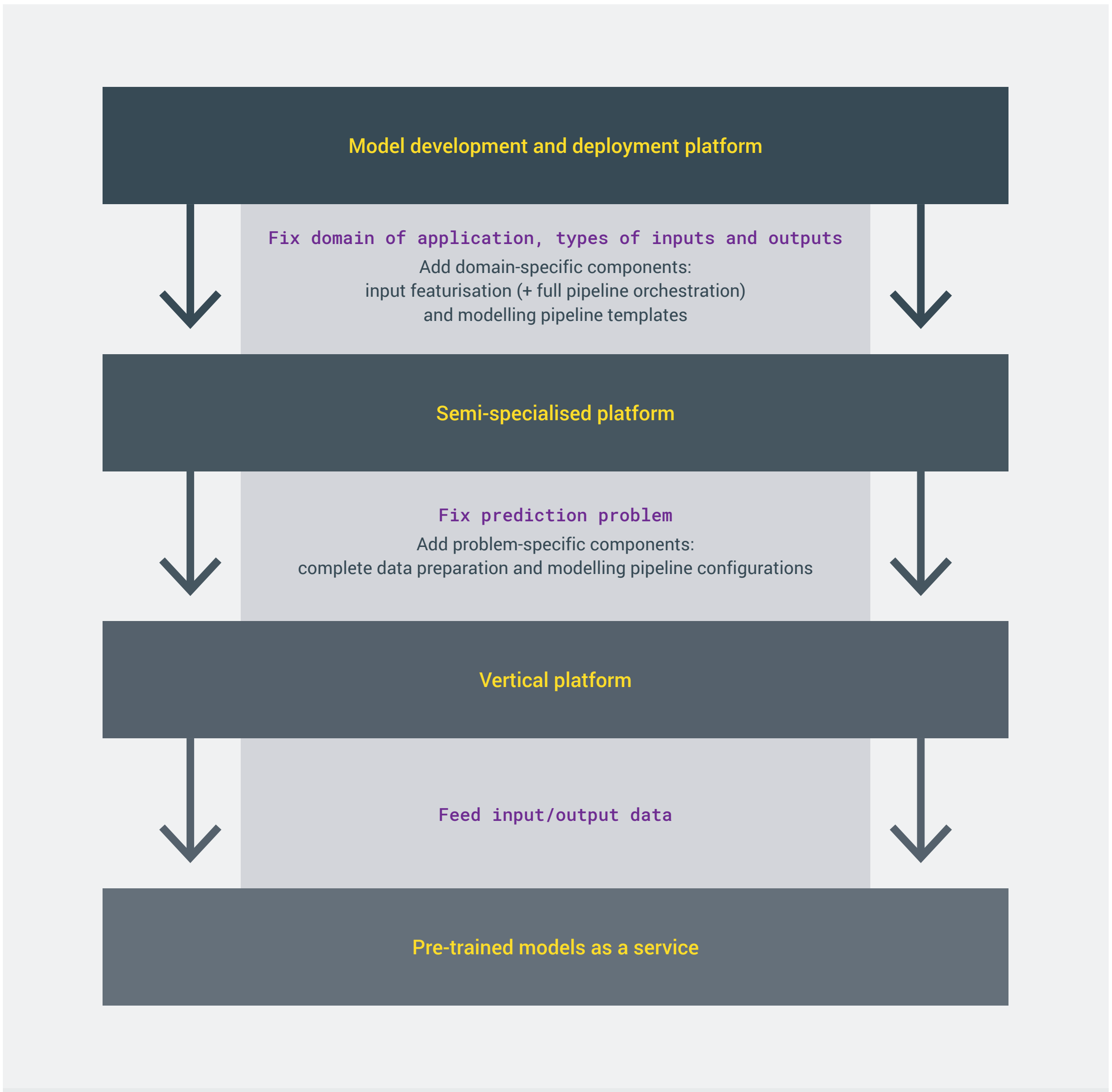
  – If starting from scratch, consider training domain experts in the basics of ML and having them start a project on a high-level MLaaS model development platform, in order to assess feasibility. If needed, have someone with more experience provide assistance. Let them pick the platform that they prefer and that allows them to get started the fastest, but pay attention to the ability to export work before investing too much time or money in that platform

  – If a modelling pipeline creation has already started with open source libraries, ML studios and cloud IDEs may be a preferred choice. They can help to be more efficient in ML experiments

  – When results are good enough, models can be turned into APIs and tested on production data. Deployment platforms will help, using the model APIs in the front-end, and monitoring performance. Also consider API system metrics to check how the API scales with traffic. If the solution does not use a serverless platform, monitoring traffic spikes and latency can be useful to adjust the parameters of scaling the solution; for instance, provision of a few more serving instances may be needed in order to deal with unexpected traffic spikes without affecting latency

  – All experimentation and deployment platforms allow for batch training, batch featurisation, and batch or real-time predictions. Some platforms are flexible enough to allow for online learning of models: new input-output pairs can be sent via an API request and passed to the model to learn from and update itself — assuming that it implements an online learning algorithm, for instance the partial_fit[116] method found in scikit-learn). However, most platforms are not made for ML use cases that require real-time featurisation

## 6. Choosing machine learning platforms and filling the gaps



Model development and deployment platform
Fix domain of application, types of inputs and outputs
Add domain-specific components:
input featurisation (+ full pipeline orchestration)
and modelling pipeline templates
Semi-specialised platform
Fix prediction problem
Add problem-specific components:
complete data preparation and modelling pipeline configurations
Vertical platform
Feed input/output data
Pre-trained models as a service

As additional help to identify the right type of ML platform to use to support ML efforts, the diagram above shows each platform type and visualises how they relate to each other.

# 6. Choosing machine learning platforms and filling the gaps

| | Deployment | Development | Semi-specialised | Vertical | Pre-trained |
|---|---|---|---|---|---|
| **Input** | Any | Any | Fixed (in ready-made platforms: image, video, text) | Fixed | Fixed |
| **Built-in featurisation** | No | No | Yes | Yes | Yes |
| **Output** | Any | Any | Any among fixed type(s) of output | Fixed | Fixed |
| **Model customisation** | Yes | Yes (using available algorithms) | Via configuration of ML pipeline, and via training/ validation data | Via training/ validation data | No |
| **Live monitoring** | System metrics | N/A | Prediction accuracy metrics | Problem-specific business metrics | No |

The characteristics of the different platform types are summarised in the table above.

Platform types are presented in increasing level of abstraction: it will be faster to integrate ML in an application with the use of higher-level platforms, such as pre-trained models as a service or a vertical platform; however, you reduce the scope of problems that can be tackled, and you lose flexibility, compared to using a model development and deployment platform.

When the nature of the input is fixed, platforms are able to provide built-in featurisation (via manually engineered features or via pre-trained models, thus providing transfer learning). Outputs are also increasingly constrained from left to right. Their type would be fixed in semi-specialised platforms, depending on the type of ML tasks that can be tackled (for example, classification or regression), and their nature would be fixed in vertical and pre-trained platforms, where the prediction tasks are fixed.

Even though the nature of the output to predict would be fixed in a vertical platform, custom models would be created, based on the input-output pairs that are sent to the ground truth collector. Further to the left of the table model customisation is increased: semi-specialised platforms may allow us to configure parts of the ML pipeline, and model development platforms let us customise it down to the choice of modelling algorithms and hyper-parameters to use.

Pre-trained models as a service platforms don't allow for model customisation. For this reason, there is no ground truth collector, hence no live monitoring of performance on production data. Only some semi-specialised and vertical platforms provide model performance monitoring. Deployment platforms are limited to monitoring of system metrics. 'N/A' is marked for 'not applicable' in the development column, since these platforms don't get to see production data.

# 6. Choosing machine learning platforms and filling the gaps

## PLATFORM SELECTION CRITERIA

ML systems are complex by nature, so it is recommended to use the simplest and highest level tools when starting ML efforts. Even if the user ultimately wants more flexibility, they can be useful to create an initial baseline, and can maximise the chances of success of pilot programmes. For example, when looking for a Cloud ML IDE, start simple with a platform such as Floyd. If the application ends up outgrowing it, it can still be switched to products such as SageMaker and leverage distributed learning, for instance. More comprehensive products are attractive because of the number of options they offer, but an assessment must be made as to whether the potential gains provided by these options would warrant the additional complexity for the team.

Using high-level platforms as a service for model development and deployment means there is less to configure and maintain. It also means more automation. Starting with an AutoML and auto-scaling solution helps get your priorities right: design and implement a performance metric that makes sense for the application, create training, validation, and test sets, work on the data, and work on a strategy to deploy predictions to end-users.

When researching platforms, one way to go beyond marketing material is to browse the developer portal and to go through the API documentation. Have a look at demo videos or screenshots of the UI, and compare functionalities with that of the API. Go into details such as the type of ML tasks, inputs, or data sources that are supported. For example, when looking at language platforms, check which languages are supported.

If the result is a list of platforms that look equivalent, just test them out and see which feels best for the application. At this stage, it is best to minimise commitment — does the platform have a free tier or a free trial, to start testing it instantly with a Kaggle dataset? When ready to consider paying or to use your own data, pay closer attention to service level agreements (SLAs), constraints, limitations, terms and conditions. In particular, when using semi-specialised and vertical platforms, make sure to be comfortable with how the platform is using data. Can data and models be exported? Who owns them? Could the platform vendor use your data to train models themselves? What would they be able to do with these models? What would happen to your models if the platform vendor stopped offering access to the platform?

A platform's pricing structure may depend on which compute is used — your own, or cloud CPU/GPU/FPGA/ASICs. If using the platform on own compute, pricing could be based on the characteristics of the machines where the platform is installed. If it is a platform in the cloud, pricing could be based on the number of predictions that will be made, on the number of models to be trained, on the volume of data used, or on compute time. A framework such as the Machine Learning Canvas[117] can help estimate these as it helps describe the ML system in detail, the data it will learn from, and how predictions will be used. These numbers will also be useful to anticipate the best option in the long term to create models (on-premise or cloud-based), the best option to make predictions (client-side or on-premise or cloud-based), and when to switch.

# 6. Choosing machine learning platforms and filling the gaps

## SCALING TO MANY USE CASES: BUILDING PROPRIETARY PLATFORMS

In his AI Transformation Playbook[118], Andrew Ng lists five steps to transform larger enterprises (with a market capitalisation from $500M to $500B). The first one is to execute pilot ML projects to gain momentum (which we suggest to do with the help of ready-made ML platforms). The second step is to build an in-house AI team. One of this team's responsibilities is to "develop company-wide platforms that are useful to multiple divisions/business units and are unlikely to be developed by an individual division". Uber, in particular, is famous for its internal MLaaS platform Michelangelo[119].

The objective of such a platform is to make it easier for all divisions in the organisation to experiment with new ML use cases and to develop new ML systems. An internal 'hub' or 'portfolio' (similar to those of Google AI Platform and Algorithmia) is a step towards this: it centralises ML assets (datasets, pipelines, feature stores, configurations, trained models, and APIs) and makes them accessible across the organisation.

Many components of an ML system can be reused from project to project:

- Databases: if the ML inputs for a new system are of the same nature as for previous systems, we would just need to add a new column to the ML input tables. Otherwise, new tables would need to created
- The evaluator, model builder, and server do not need to change. Models could be tagged with the name of the system they belong to, in the file system in which they are saved. This would allow the server to know which models are available for each ML system in production
- The ground truth collector would essentially remain the same, but it would need to know in which database and table to store ground truths it receives; it could be adapted so that this information would be passed to it, or it could infer it from the name of the ML output
- Monitor: users would want to reuse the monitor and its widgets, but would need it to be aware of the new use case and to use its performance metrics. This might be achieved with a configuration file for each use case
- The orchestrator's workflow/pipeline could also be made configurable; users would specify in its configuration where to get data from, how to split data, what to pass to the model builder (for example, which algorithms to use for pre-processing and modelling, which range of hyper-parameters to search), and which performance criterion to use for model deployment. Configurations could be reused across ML systems, and adapted
- The front-end can be quite specific to the ML use case, in the input that it takes and the output it returns. For instance, if there are many concepts to detect in an input image, then users might want to serve requests to get the top K concepts only. However, the saving of production inputs and predictions, and the model management features (test, update, rollback) would remain very similar. This could motivate splitting the front-end into two components: a "post-processor" applied to the raw prediction, and a "model manager". The latter could be reused across use cases and would just need to know where to store predictions (which database/table)
- Featuriser:
  - If the ML inputs of the new system are different from those in previous systems, a new featuriser should be created. One of the key functionalities of Michelangelo is its Feature Store, to make it easier to reuse previously created features and to share work across the organisation's teams
  - If the inputs are the same (the company's customers, for example), the previous featuriser could be reused, and we could think of the ML platform as a semi-specialised one. If the previous ML system used a neural network model, the featuriser could be augmented with one of the last hidden layers of that network, which would allow for transfer learning across use cases

# Conclusions

# Conclusions

Machine learning platforms promise to help make software or data science teams more productive, and free up time to spend on tasks that are as important for the success of ML projects as the actual ML: tasks such as gathering data, designing the right performance metrics, and working on the deployment strategy.

They can also help embed good engineering practices through documentation, versioning, collaboration and visualisations that make monitoring, audit, measurement and improvement easier. These are part of responsible ML development. Some also include libraries that are explicitly focused on responsible ML development, such as for evaluating bias or providing explanations.

These types of functions are expected to expand and become more widely available. See Digital Catapult's 'From What to How. An Overview of AI Ethics Tools, Methods and Research to Translate Principles into Practices'[120] for more detail. It may be more cost-effective and easier to use third-party functions than to develop them all in-house.

However, the choice of ML platforms is wide, and can be daunting. The ML platforms that are available and the features that they offer are subject to rapid change. This report has therefore chosen to highlight different types of ML platform, with examples given for illustration, rather than providing a snapshot of the whole current landscape. It is hoped that this will help organisations to identify whether to use a third-party platform, which is the right sort of platform for their needs, and how to build upon existing platforms to create their own, that will support scaling up to many ML use cases.

We have also provided some advice on criteria for choosing a specific platform. Relatively up-to-date lists of vendors can always be found on the internet.

# Footnotes

[1] https://www.gartner.com/reviews/customers-choice/data-science-machine-learning-platforms/May-2019

[2] https://en.wikipedia.org/wiki/Platform_as_a_service

[3] https://en.wikipedia.org/wiki/Cloud_computing#Service_models

[4] https://en.wikipedia.org/wiki/Computing_platform

[5] http://aiplaybook.a16z.com/docs/intro/survey-parameters

[6] https://www.welcome.ai/sift

[7] https://medium.com/mmc-writes/introducing-the-state-of-ai-2019-divergence-14d69cb3b16c

[8] http://www.bradfordcross.com/blog/2017/6/13/vertical-ai-startups-solving-industry-specific-problems-by-combining-ai-and-subject-matter-expertise

[9] https://www.predictionmachines.ai

[10] http://www.json.org/

[11] https://curl.haxx.se

[12] https://indico.io

[13] https://kairos.io

[14] https://www.clarifai.com/

[15] https://blog.clarifai.com/introducing-clarifais-first-end-to-end-solution-for-moderation (accessed April 2019)

[16] https://cloud.google.com/natural-language/

[17] https://cloud.google.com/vision/docs/

[18] https://www.microsoft.com/cognitive-services

[19] https://landing.ai/ai-solutions/

[20] ibid [5].

[21] https://sift.com/developers/docs/

[22] https://infer.com/

[23] https://www.salesforce.com/products/einstein/features/

[24] https://metamind.readme.io/docs/intro-to-einstein-language

[25] https://cloud.google.com/solutions/contact-center/

[26] https://www.answeriq.com/home

[27] https://cloud.google.com/solutions/talent-solution/

[28] https://dialogflow.com/

[29] https://cloud.google.com/recommendations/

[30] https://www.clarifai.com/

[31] https://community.clarifai.com/t/our-new-user-interface-is-now-live/1252 (accessed Apr 2019)

[32] https://clarifai.com/developer/guide/ (accessed Apr 2019)

[33] https://clarifai.com/developer/guide/ (accessed Apr 2019)

[34] https://aws.amazon.com/comprehend/

[35] https://cloud.google.com/natural-language/

## Footnotes

36 https://monkeylearn.com

37 https://lateral.io/tech

38 https://aws.amazon.com/rekognition/

39 https://cloud.google.com/vision/

40 https://cloud.google.com/data-labeling/docs/

41 https://www.figure-eight.com

42 https://landing.ai/ai-solutions/ (accessed Apr 2019)

43 https://indico.io/blog/docs/indico-api/text-analysis/text-features/

44 https://indico.io/blog/docs/indico-api/image-analysis/image-features/

45 https://bigml.com

46 https://cloud.google.com/automl-tables/

47 https://cloud.google.com/inference/

48 https://www.craft.ai

49 https://azure.microsoft.com/en-us/services/machine-learning-studio/

50 https://lobe.ai

51 https://lobe.ai/examples (accessed Apr 2019)

52 https://lobe.ai (accessed Apr 2019)

53 https://www.datarobot.com

54 https://rapidminer.com

55 https://www.dataiku.com

56 https://www.dataiku.com/learn/guide/spark/mllib/using-mllib-in-ui.html (accessed Apr 2019)

57 https://www.dataiku.com/learn/guide/spark/mllib/using-mllib-in-ui.html (accessed Apr 2019)

58 https://doc.dataiku.com/dss/latest/machine-learning/auto-ml.html (accessed Apr 2019)

59 https://www.datarobot.com/wiki/prediction-explanations/ (accessed Apr 2019)

60 https://www.datarobot.com/wiki/classification/ (accessed Apr 2019)

61 https://www.datarobot.com/product/time-series/ (accessed Apr 2019)

62 http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/examples/credit_card_default.html?highlight=api

63 https://jupyter.org

64 https://www.tensorflow.org/guide/summaries_and_tensorboard

65 http://floydhub.com

66 https://faculty.ai

67 https://blog.floydhub.com/workspaces/ (accessed Apr 2019)

68 https://blog.floydhub.com/advanced-features/ (accessed Apr 2019)

69 https://docs.floydhub.com/guides/jobs/metrics/

## Footnotes

(accessed Apr 2019)

70 https://cloud.google.com/ai-platform-notebooks/

71 https://cloud.google.com/deep-learning-vm/

72 https://cloud.google.com/tpu/

73 https://aws.amazon.com/sagemaker/

74 https://databricks.com/product/unified-analytics-platform

75 https://databricks.com/blog/2018/06/05/introducing-mlflow-an-open-source-machine-learning-platform.html (accessed Apr 2019)

76 https://www.trifacta.com

77 https://cloud.google.com/dataprep/

78 https://cloud.google.com/dataflow/

79 https://cloud.google.com/bigquery/

80 https://cloud.google.com/ml-engine/docs/tensorflow/training-overview

81 https://cloud.google.com/ai-platform/ (accessed Apr 2019)

82 https://kubernetes.io

83 http://kubeflow.org

84 https://www.seldon.io/open-source/

85 https://cloud.google.com/kubernetes-engine/

86 https://azure.microsoft.com/services/kubernetes-service/

87 https://aws.amazon.com/eks/

88 https://aws.amazon.com/fargate/

89 https://aws.amazon.com/ecs/

90 https://airflow.apache.org

91 https://blog.algorithmia.com/how-to-version-control-your-production-machine-learning-models/

92 https://www.mlflow.org/docs/latest/python_api/mlflow.pyfunc.html#pyfunc-filesystem-format

93 https://aws.amazon.com/sagemaker/neo/

94 https://aws.amazon.com/greengrass/ml/

95 https://azure.microsoft.com/en-us/services/iot-edge/

96 https://en.wikipedia.org/wiki/Predictive_Model_Markup_Language

97 https://en.wikipedia.org/wiki/Portable_Format_for_Analytics

98 https://onnx.ai/

99 https://gitlab.com/juliensimon/dlnotebooks/blob/master/sagemaker/05-Image-classification-two-models.ipynb

100 https://aws.amazon.com/cloudwatch/

101 https://docs.aws.amazon.com/sagemaker/latest/dg/API_UpdateEndpointWeightsAndCapacities.html

102 https://medium.com/@julsimon/mastering-the-mystical-art-of-model-deployment-c0cafe011175 (accessed Apr 2019)

## Footnotes

[103] https://algorithmia.com

[104] https://algorithmia.com/admin/metrics (accessed Apr 2019)

[105] https://aws.amazon.com/lambda/

[106] https://cloud.google.com/functions/

[107] https://cloud.google.com/run/

[108] https://knative.dev/

[109] https://www.seldon.io/

[110] https://www.youtube.com/watch?v=jYZ8nlPrMFM

[111] https://indico.io/blog/docs/indico-api/custom-collections/explaining-predictions/

[112] https://www.tensorflow.org/tfx/guide/serving

[113] https://github.com/tensorflow/privacy

[114] https://www.tensorflow.org/federated

[115] https://github.com/OpenMined/PySyft

[116] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier.partial_fit

[117] http://www.machinelearningcanvas.com

[118] https://landing.ai/ai-transformation-playbook/

[119] http://proceedings.mlr.press/v67/li17a/li17a.pdf

[120] https://arxiv.org/abs/1905.06876

# About

## DIGITAL CATAPULT

Digital Catapult is the UK's leading advanced digital technology innovation centre. It drives the early adoption of digital technologies: to make UK businesses more competitive and productive, and to grow the country's economy. Its AI and machine learning stream consists of a team of applied technology specialists, and the provision of facilities and programmes that support innovation and facilitate collaboration across large organisations, startups and academia.

## PAPIS

PAPIs is the first series of international conferences dedicated to real-world machine learning applications and APIs. It provides a venue for leading ML practitioners to share their experiences in building ML systems, to discuss challenges, and to present innovative tools, frameworks and platforms that are taking the industry further. Previous editions took place in São Paulo, Boston, Sydney, Barcelona, Paris, Valencia, and London.

## AUTHOR

Louis Dorard is the author of the Machine Learning Canvas, general chair of PAPIs, and lead instructor at Microsoft AI School. As an independent consultant and ML coach, Louis helps corporations and startups integrate ML into their products. He has held workshops at major companies such as Airbus, Amazon, Deloitte, EDF, Intel, Konica Minolta. Previously he taught ML at UCL School of Management. Louis holds a PhD in ML from University College London.

You can get in touch with him at www.louisdorard.com.

Digital Catapult
101 Euston Road
London NW1 2RA

0300 1233 101

www.digicatapult.org.uk

www.papis.io