
Prevision.io Documentation

Gerome Pistre

Feb 14, 2022

REFERENCE DOCUMENTATION

1	Value proposition	1
2	Requirements	3
3	Conditions	5
4	contacts	7
5	Getting started	9
5.1	Account creation	9
5.2	Connection	9
5.3	Cloud & freetrial limitations	9
5.3.1	Studio	11
5.3.1.1	Projects	11
5.3.1.2	Data	16
5.3.1.3	Experiments	36
5.3.1.4	Pipelines	85
5.3.1.5	Deployments	97
5.3.1.6	Contributors	101
5.3.1.7	Notebooks	103
5.3.1.8	Settings	104
5.3.2	API	105
5.3.2.1	Using The API	105
5.3.3	SDK	106
5.3.3.1	Using the Python SDK	106
5.3.3.2	Using the R SDK	109
5.3.3.3	Using the Prevision Quantum NN SDK	109
5.3.4	Guides and How-to	109
5.3.4.1	Datascience Guide	109
5.3.4.2	A list of some useful Dataset to explore Machine Learning	175

VALUE PROPOSITION

[Prevision.io](#) brings powerful AI management capabilities to data science users so more AI projects make it into production and stay in production. Our purpose-built AI Management platform was designed by data scientists for data scientists and citizen data scientists to scale their value, domain expertise, and impact. The platform manages the hidden complexities and burdensome tasks that get in the way of realizing the tremendous productivity and performance gains AI can deliver across your business

REQUIREMENTS

Prevision.IA is a SAAS platform optimized for Firefox and Chrome navigators. The cloud version can be accessed online at <https://cloud.prevision.io>, or it can be deployed on-premise or in your private cloud. Please visit us at <https://prevision.io> if you have any questions regarding our deployment capabilities.

CONDITIONS

Please read the general terms and conditions available on following link : <https://cloud.prevision.io/terms>

CONTACTS

If you have any questions about using Prevision.io platform please contact us using the chat button on the Prevision.io store interface or by email at the following contact address :

support@prevision.io

GETTING STARTED

There are two ways to get a Prevision.io account :

1. Go to <https://cloud.prevision.io> and open an account (or link one of your OAuth2 enabled account)
2. Deploy a dedicated computing service on Google Cloud

In order to deploy Prevision.io on Google Cloud, go to the dedicated Guide

5.1 Account creation

By clicking to the following address, you will land on the connection page which allows you to create an account or sign in if you already have a Prevision.io account.

<https://cloud.prevision.io>

In order to create a new account, just click on the sign up button next to the log in button. You will access the following account creation form.

Once you have filled the needed information, you will have a 30 days free but limited access to the Prevision.io platform. In order to upgrade your free trial account into a full access one, please contact us on following email (support@prevision.io)

Once done you can follow our *[complete guide to release a model](#)*

5.2 Connection

Once your account has been created, you will be able to access the prevision's Studio and Store and start creating models and deploying them.

Please note that SSO using your google/linkedin/GitHub account is available.

5.3 Cloud & freetrial limitations

If you are using our [cloud platform](#) using a free trial account, some limitations are set up. Here's a quick view of limitations for free accounts:

[English](#)

LOG IN. **SIGN UP.**

Register for a free 30-days trial period

First name

Last name

Country

Job title

Enterprise

Sector

Email

Password

Confirm password

☐ Je ne suis pas un robot



reCAPTCHA
Confidentialité - Conditions

Fig. 1: Signup screen

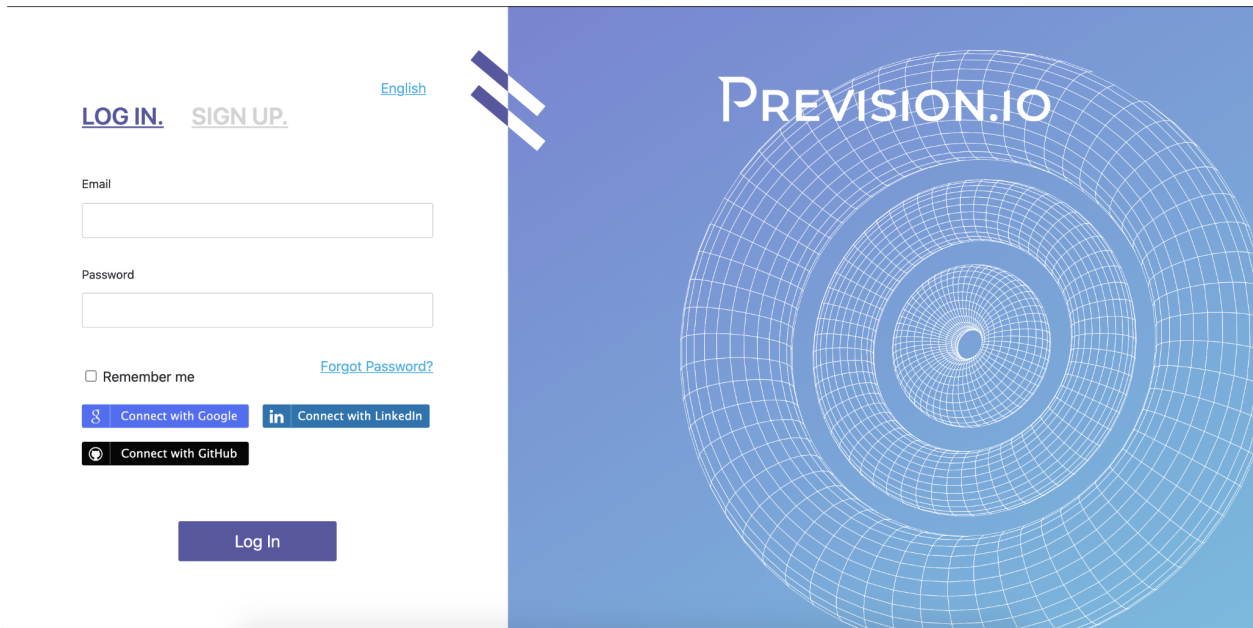


Fig. 2: login with SSO

Table 1: Freetrial Limitation

Entity	Action	Limitation
PROJECT	Create Project	Free trial users can create 2 Limited Projects
DATASETS	Add dataset from file / datasource	10 Datasets max + 1GB per dataset
IMAGE FOLDER	Add / Update / Delete image Folder in project	1 Image Folder
DATA SOURCE	Add / Update / Delete datasource in project	1 Datasource max
CONNECTOR	Add / Update / Delete connector in project	1 Connector max
USECASE	Add / Update / Delete experiment in project	5 Use Cases max
USECASE VER-SION	Add / Update experiment version in project	3 Concurrent experiment versions
PREDICTION	Add / Update prediction in experiment version	2 Concurrent predictions
STORE APP	Deploy Apps	5 Concurrent deployed apps

5.3.1 Studio

5.3.1.1 Projects

In Prevision.io studio, resources, such as datasets or models, are scoped by project in order to structure your work and collaborate easily with people inside a project.

A project is a collection of :

- *Datas* : for importing and exporting your data from and to external database or files
- *Experiments* : to build model, evaluate them and compare them
- *Pipelines* : to set up and schedule datascience pipelines
- *Deployments* : to push models to production and monitor it

- *Collaborators* : to add and manage users

List my Projects

All your projects are available on the homepage of your server. You can reach it with the mosaic icon on the upper-left corner of each screen

The screenshot shows the Prevision.io interface for a project named 'Sales prediction'. A red box highlights the mosaic icon in the top-left corner of the sidebar. The main content area displays three sections: Datasets, Pipelines, and Experiments, each with a table of project items.

Datasets							
NAME	CREATED AT	ROWS	COLUMNS	SIZE	SOURCE	STATUS	
predicted sales	10/14/2021, 08:47	127,438	3	3.93 MB	Pipeline output	Ready to be used	
development_predict_regression_21-10-14 06:39:29	10/14/2021, 06:39	127,438	17	14.27 MB	Pipeline intermediate file	Ready to be used	
predicted sales	10/14/2021, 03:18	127,438	3	3.93 MB	Pipeline output	Ready to be used	

Pipelines					
NAME	DESCRIPTION	NODES	CREATED AT	CREATED BY	USED IN RUN
Weekly Sales Forecast		3	10/11/2021, 15:33	arnold.zepher	Yes
write folds		3	10/11/2021, 09:42	arnold.zepher	Yes
build fold		2	10/11/2021, 09:25	arnold.zepher	No

Experiments									
NAME	SOURCE	VERSION	CREATED AT	DATA TYPE	TRAINING TYPE	SCORE	MODELS	PREDICTIONS	STATUS
az	AutoML		10/11/2021, 10:30	Tabular	Regression				
Sales forecast	AutoML	3	10/11/2021, 10:01	Tabular	Regression	3,450 (mae)	17	2,166,446	

Fig. 3: Go to the list of projects

You can switch from card view

To List view :

You will find the following information on the cards :

- project name
- created by and creating at
- description (if available)
- number of datasets/pipelines/use cases
- list of collaborators into the project and their associated role into this project

If your role has been set up as admin into a project, an action button on top right of each card will be available. By clicking on this button you will be able to edit the project information and delete the project. Please note that deleting a project will also delete all sub project's items, such as pipelines or datasets, created on the project.

Note: Tips : you can filter projects by their names using the search bar on top right of the projects view

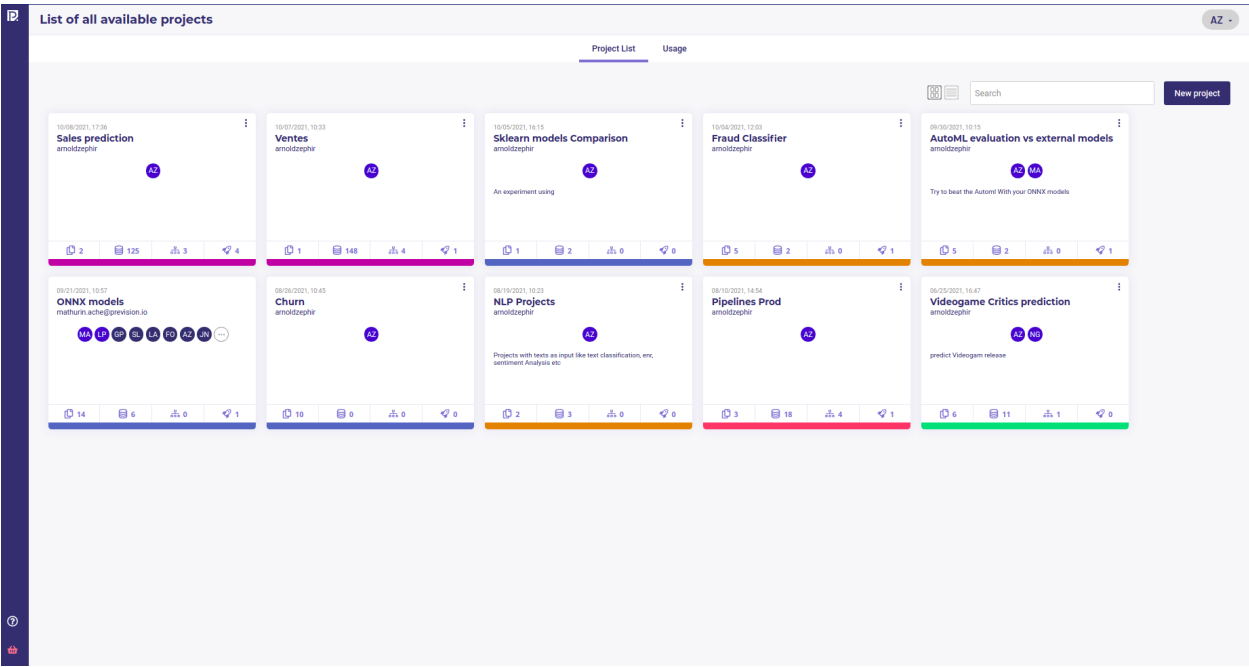


Fig. 4: List of projects

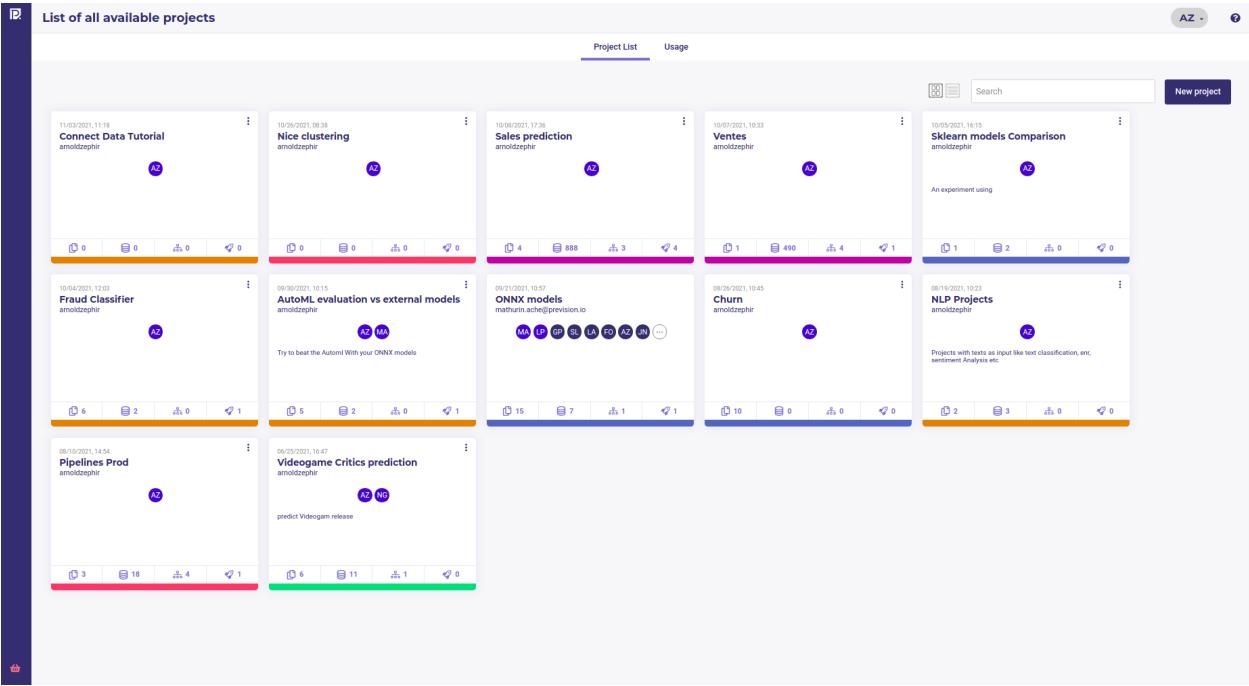


Fig. 5: List of projects

Create a new project

All your datascience project assets belong to a project. In order to Experiment, import models, deploy or monitor you need to create a new project.

When viewing the list of project on [your homepage](#), you can create a new project, by clicking the “new project” button on top right of the “my projects” view.

You will access to the following interface :



The screenshot shows a modal window titled "New project" with a close button (X) in the top right corner. Inside the modal, there are three input fields: "Project's name" with a character count of "0 / 40 characters", "Project's description (optional)" with a character count of "0 / 210 characters", and "Project's color" which displays a row of ten colored circles (blue, light blue, green, orange, yellow, red, purple, magenta, cyan, and dark blue). At the bottom of the modal, there are two buttons: a grey "Cancel" button on the left and a dark blue "Create" button on the right.

Fig. 6: Create a Project

In order to create your project you have to fulfill at least a color and a project name. You can also add a description of your project.

Please note that you can at any moment, if admin role into a project has been setted up for your account, change these information by clicking on “settings” into the project menu.

All your projects will be displayed in the “my projects” view. Two different displays are list view and cards view and you can switch between one view and another by clicking on the view button you prefer next to the search bar.

Project navigation

By entering a project, you will first be redirected to the project homepage. The following sections, including the 3 latest entries for each section, are displayed :

- Datasets : last uploaded dataset
- Pipelines : last pipeline templates
- Usecases : last experiments

Under each section you will also find a link to the dedicated page, also available through the left project main menu, and, for pipelines and experiment, a shortcut to create new ones.

Into a project you can load data, create a pipeline in order to automate some task or data transformations and train models into use cases menu. Once you enter a project by clicking on its card or on the list, the project menu will be loaded on the left navigation bar.

- Home : you will find here last items from datasets, pipelines and use cases created into the project.
- Usecases : you will find all your use cases trained into the selected project
- Data : you will find here your datasets and the connectors setted up on the project

Axel's Project

Datasets

NAME	CREATED AT	ROWS	COLUMNS	SIZE	DATA SOURCE	STATUS
regression_house_80_lite	06/23/2021, 17:14	399	21	39.67 KB		Ready to be used
french_tweets_lite	06/23/2021, 15:21	404,769	2	39.42 MB		Ready to be used
series-trafics	06/23/2021, 15:20	99,999	3	3.29 MB		Ready to be used

[See all Datasets](#)

Pipelines

No data for table

[See all Pipelines](#) [Create Pipeline](#)

Usecases

NAME	VERSION	CREATED AT	DATA TYPE	TRAINING TYPE	SCORE	MODELS	PREDICTIONS	STATUS
classif_edf_80_lite	1	06/24/2021, 10:28	Tabular	Classification	-	0	0	🟢

[See all Usecases](#) [Create Usecase](#)

[Aide](#)

- Pipelines : you will be able thanks to pipeline to augment your data and automate some actions
- Deployment : deploy and monitor deployed models and applications
- Collaborators : list of all project collaborators and their associated project role
- Settings : if your project role is admin, this menu is available and allows you to edit the project informations

collaboration into a project

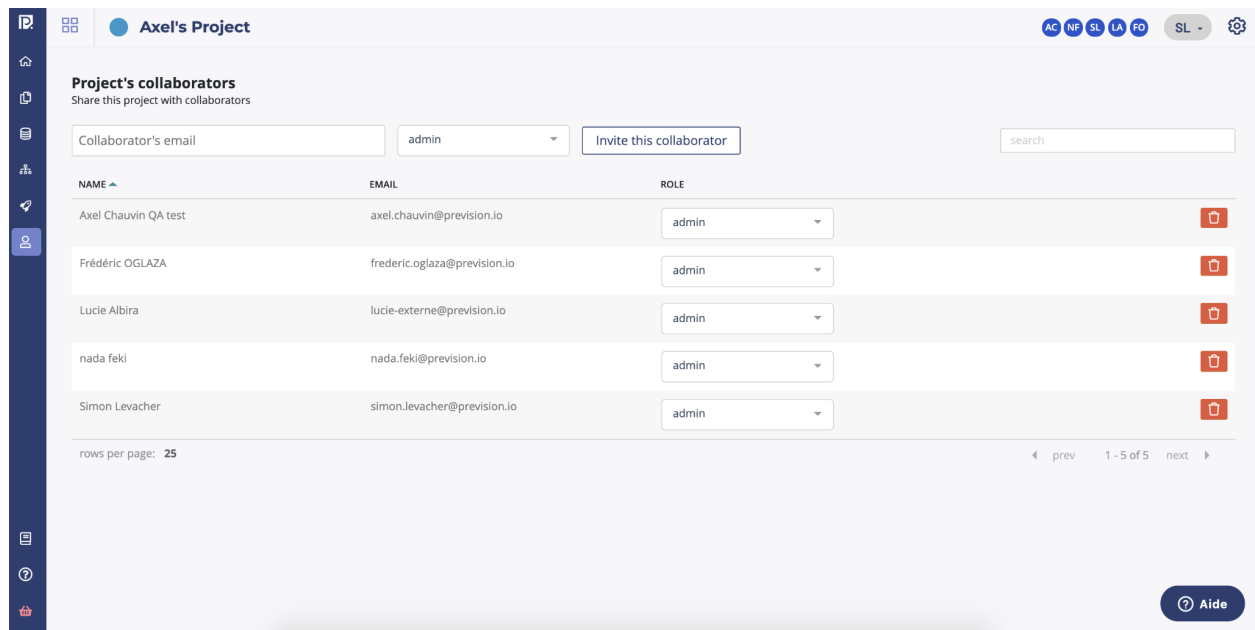
Prevision.io studio is built in order for our users to collaborate within the projects. To do that, into the “collaborators” menu of a selected project you can manage, if your role is admin on the project, the collaborators and the right inside the project.

- Add a user : by enter the email address of a Prevision.io platform registered user you can add a collaborator
- role : if your role level is admin into the project you will be able to edit user roles
- by clicking the delete button on the right side of a user, you can disable the access to the selected user to the project

Project roles

Into a project there are 3 levels of roles :

- End-user : in this project, the user can only access to the list of deployed models and applications and make predictions
- Viewer : you can navigate into all ressources of a project and visualize information but you can't download or create ressources
- Contributor : Viewer rights + you can create and manage resources of the project
- Admin : Contributor rights + you can manage project users and project settings



Edit a project

You can change the following parameters of your project by clicking on settings on the main navigation of your project or, on the list/card view of your project by clicking on the action button.

- Name of the project
- Description of the project
- Color displayed on the card of the project

Delete a project

If a project is no longer useful, you can delete it by clicking on the action button on the card/list projects view.

Warning : all ressources created into the project will be deleted with the suppression of the project with no possibility of back-up. If deleted, a project and its resources are no longer available for you but also for all users previously added to this project.

5.3.1.2 Data

Data is the raw material for all experiments. All your data are scoped to a *project* and you can access them from the “Data” section on the collapsing sidebar

The datas section holds 5 kind of assets :

- datasets : input for train, predictions and pipelines
- Image folders : folders containing image
- Data sources : information about location (database, table, folder) or remote dataset
- Exporters : information about remote table for exporting prediction

NAME	ROWS	COLUMNS	SIZE	CREATED AT	CREATED BY	SOURCE	STATUS	ACTIONS
dataset	2,000	13	123.96 KB	08/24/2021, 15:53	arnold zephir	File upload	Ready to be used	Compute embeddings
dataset	8,000	13	494.88 KB	08/24/2021, 15:53	arnold zephir	File upload	Ready to be used	Explore embeddings

Fig. 7: Data section

Datasets

Datasets are tabular data resulting from a flat file upload, a Datasource or a *pipeline execution*. They are input for *training*, *predicting* and *pipeline execution*

Datasets

- may be created from file upload
- may be created from data source
- may be created from pipeline output
- may be downloaded
- may be exported with exporters
- may be used as pipeline input

When you click on the datasets tabs, you see a list of your dataset with :

- filter checkbox for origin of the dataset (*pipeline output*, *Datasource*, *File upload*, *Pipeline intermediate file*)
- search box filtering on the name of datasets
- name of the datasets and information about them
- status. A dataset could be unavailable a short time after its creation due to parsing.
- a button to compute embeddings or explore them if already computed (see [the guide about exploring data](#))

NAME	ROWS	COLUMNS	SIZE	CREATED AT	CREATED BY	SOURCE	STATUS	ACTIONS
test_intermarche	86,940	23	12.07 MB	08/25/2021, 13:54	arnold zephir	File upload	Ready to be used	PT: Compute embeddings
train_intermarche	350,470	24	47.06 MB	08/25/2021, 13:53	arnold zephir	File upload	Ready to be used	PT: Compute embeddings
titanic_train	891	13	45.64 KB	07/01/2021, 18:35	piere Nowak	File upload	Ready to be used	PT: Compute embeddings
titanic_train	891	13	45.64 KB	07/01/2021, 18:35	piere Nowak	File upload	Ready to be used	PT: Compute embeddings
titanic_test	418	12	21.12 KB	07/01/2021, 18:35	piere Nowak	File upload	Ready to be used	PT: Compute embeddings
titanic_train	891	13	45.64 KB	07/01/2021, 18:35	piere Nowak	File upload	Ready to be used	PT: Compute embeddings
sales_timeseries	543,850	5	18.6 MB	05/26/2021, 16:06		File upload	Ready to be used	PT: Compute embeddings
sales	543,850	5	18.6 MB	05/25/2021, 17:44		File upload	Ready to be used	PT: Compute embeddings
output_2	402,423	4	12.34 MB	05/25/2021, 15:32		File upload	Ready to be used	Explore embeddings
ventes_products	32,793,018	7	2.49 GB	05/24/2021, 14:15		File upload	Ready to be used	Explore embeddings
mock_kaggle	937	4	21.71 KB	05/20/2021, 15:59		File upload	Ready to be used	PT: Compute embeddings

Fig. 8: Dataset list

Create a new dataset

In order to create new *experiments* you need a dataset. They can be created by clicking on the “import dataset” button. Dataset are always created from tabular data (database table or files). You can import data from a previously created *datasource* or from a flat csv file.

For data coming from file (upload, ftp, bucket, S3,...) you could input the columns and decimals separator but the auto detect algorithm will work in most of case.

When you click on the “save dataset” button, the dataset will immediately be displayed in your list of dataset but won’t be available for a few seconds. Once a dataset is ready, its status will change to “Ready to be used” and you can then compute embedding and use it from training and predicting.

Analyse dataset

Once a dataset is ready, you got access to a dedicated page with detail about your dataset

- **General Information** : Dataset summary, feature distribution and correlation matrix of its features
- **Columns** : Information about its columns. You can click on a column to get more information about it, its distribution for example
- **Sample** : a very short sample of the dataset

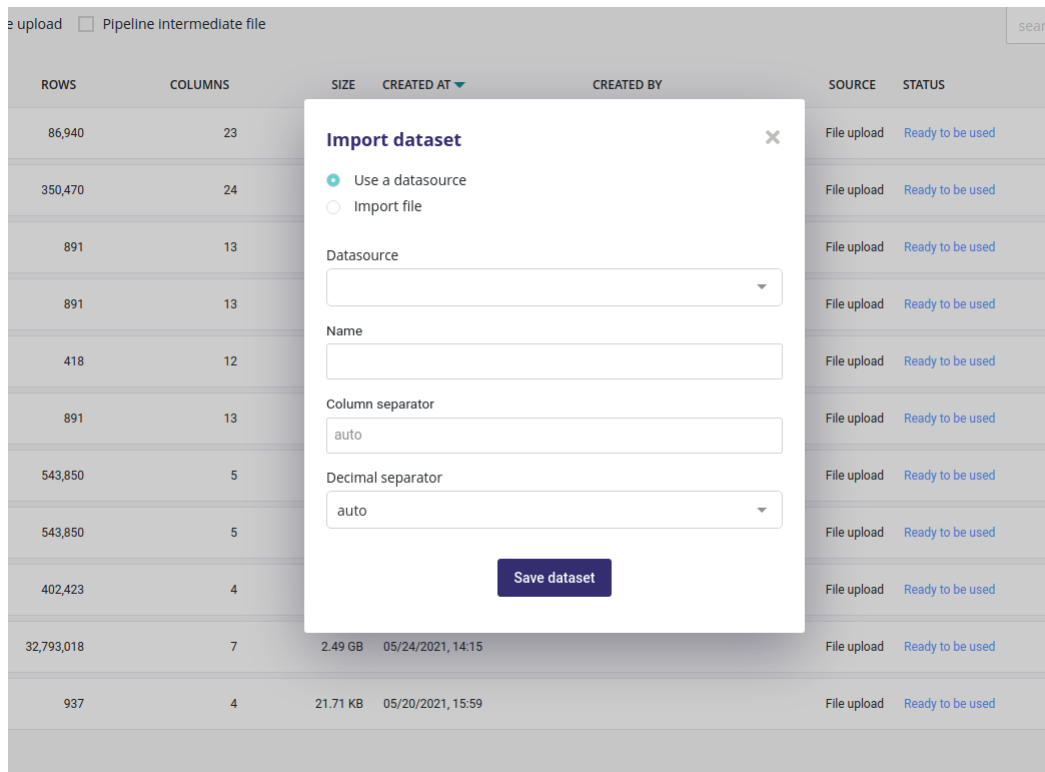


Fig. 9: Dataset import

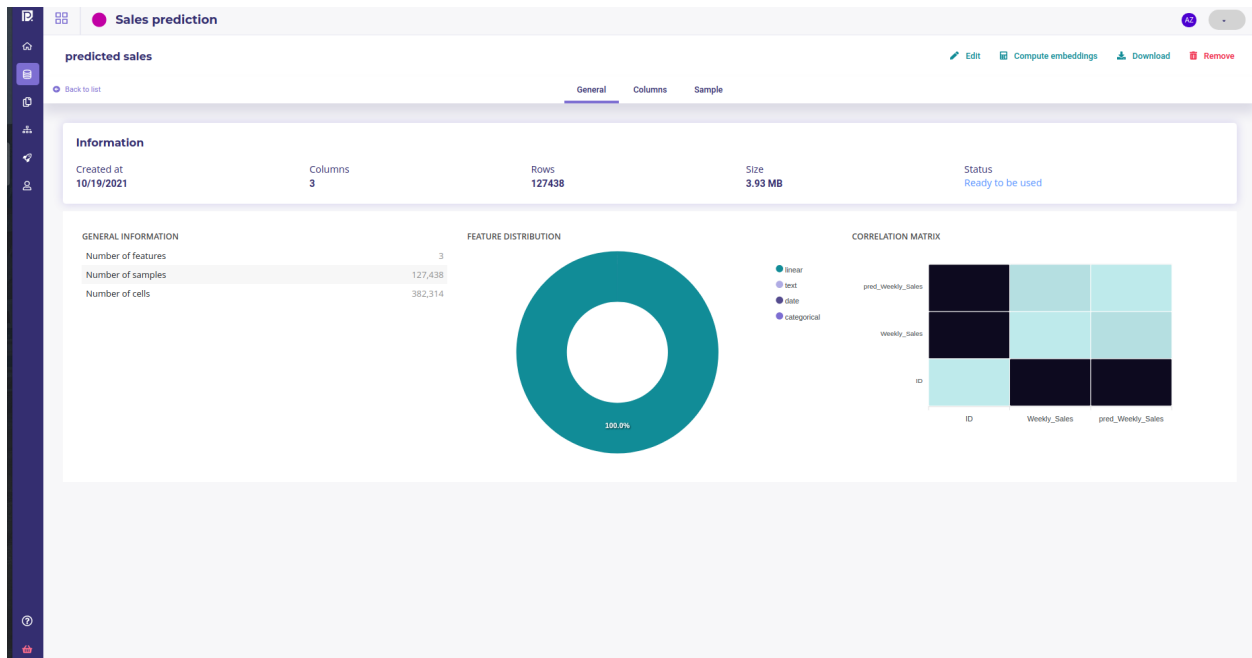


Fig. 10: Dataset detail

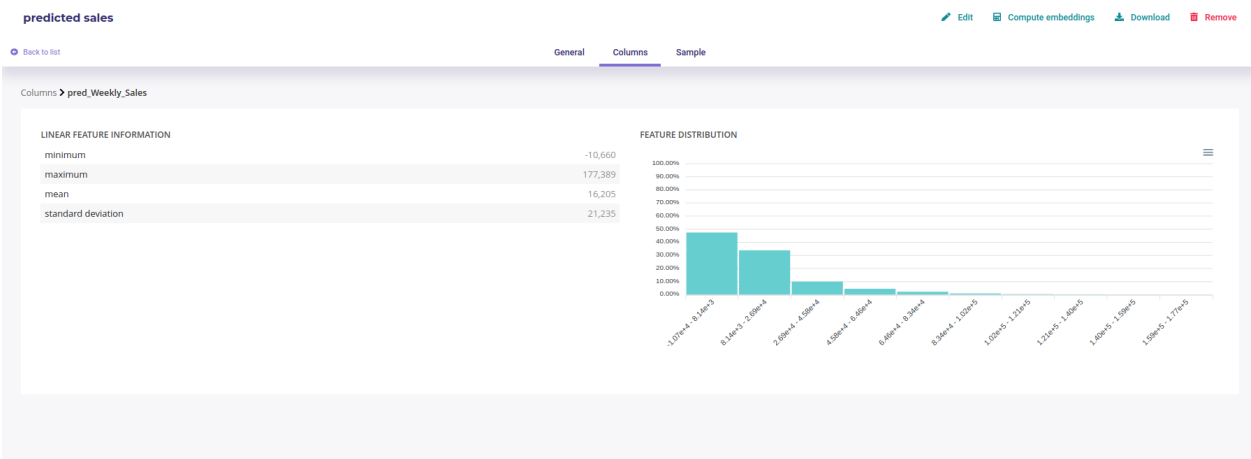


Fig. 11: Detail of a column

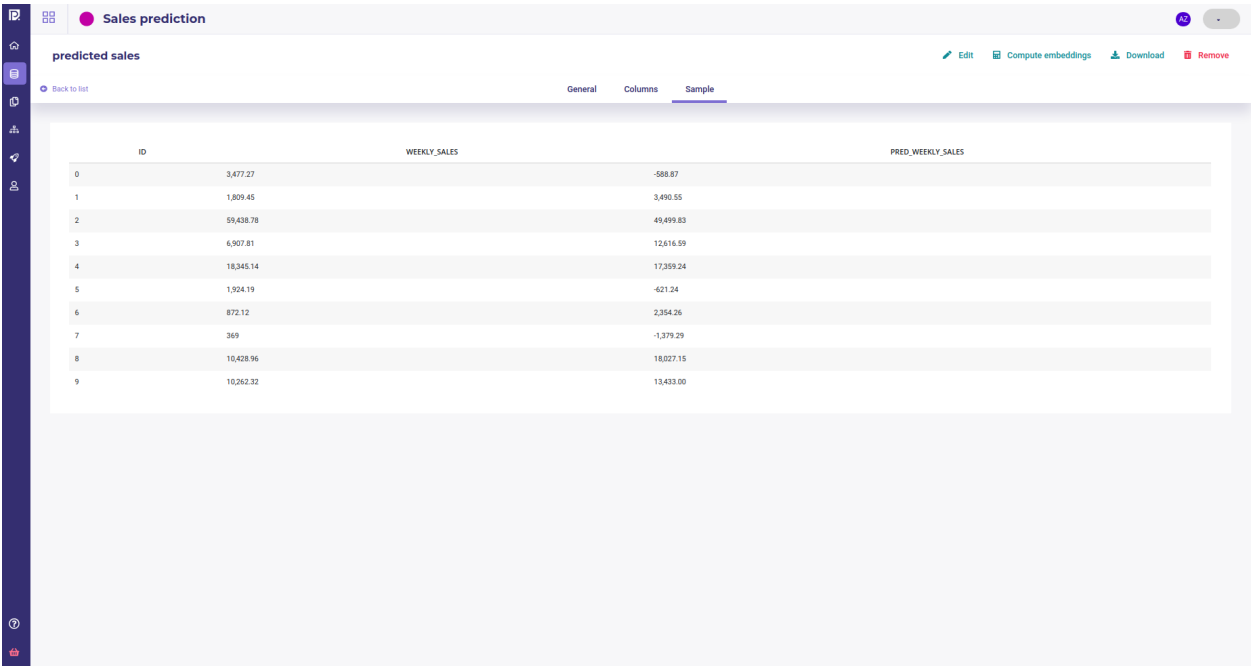


Fig. 12: Sample of a dataset

Edit and delete

You can edit name of a dataset, download it or remove it of your storage either in the top nav menu of the dataset details page or from the dot menu in the list of your dataset.

<input type="checkbox"/>	NAME	ROWS	COLUMNS	SIZE	CREATED AT ▼	CREATED BY	SOURCE	STATUS	ACTIONS
<input type="checkbox"/>	multiclassif_wines	1,483	12	77.54 KB	10/19/2021, 15:38	arnold zeghir	File upload	Ready to be used	<div> <div>Compute embeddings</div> <div>...</div> </div>
<input type="checkbox"/>	predicted_sales	127,438	3	3.93 MB	10/19/2021, 15:32	arnold zeghir	Pipeline output	Ready to be used	<div> <div>Compute embeddings</div> <div> <div>Edit</div> <div>Download</div> <div>Use an exporter</div> <div>Remove</div> </div> </div>
<input type="checkbox"/>	deployment_credit_recession_21-10-19 13:29:13	127,438	17	14.27 MB	10/19/2021, 15:29	arnold zeghir	Pipeline intermediate file	Ready to be used	<div> <div>Compute embeddings</div> </div>
<input type="checkbox"/>	predicted_sales	127,438	3	3.93 MB	10/19/2021, 14:19	arnold zeghir	Pipeline output	Ready to be used	<div> <div>Compute embeddings</div> </div>

Fig. 13: Edit and remove dataset

When you remove a dataset, local data are completely removed. Data source data are left untouched.

Compute embeddings

See the : [Complete Guide for exploring data](#)

Image folders

Image folders are storage for your image. It is source material for image experiments (classification, object detector, ...). For Images experiments you need an image folder.

Image folders

- may be creating from file upload
- can not use datasource or connectors
- can be downloaded
- can not be exported

Create a new image folder

When clicking on “Upload Image Folder” button, you can upload a zip file either from drag and dropping it or from selecting it from your local file browser. The zip file must contain only image but they can be organized into folders.

After having given a name, just click on “upload image folder” and wait. Your images will be available for *experiments* in a few seconds.

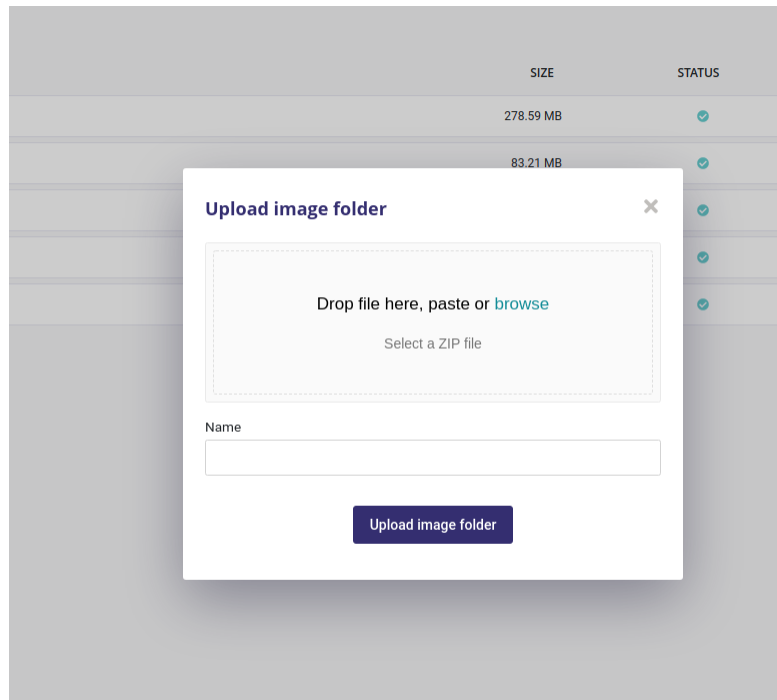


Fig. 14: Upload a folder of image

Edit and remove

You can edit the name of of your folder from the list of image folder, using the three-dots menu on the right. Removing the image folder from your storage is available from this menu too.

Connectors

Connectors are used to hold credentials used to access external databases or filesystems. You need to create a connector first to use Datasources and Exporters.

connectors

- may be used for creating data source
 - may be used for creating exporter
-

There are two kind of connectors :

- Connectors to database table (Any SQL Database)
- Connectors to storage (FTP, SFTP, Amazon S3 or Google Cloud Platform) that contain dataset

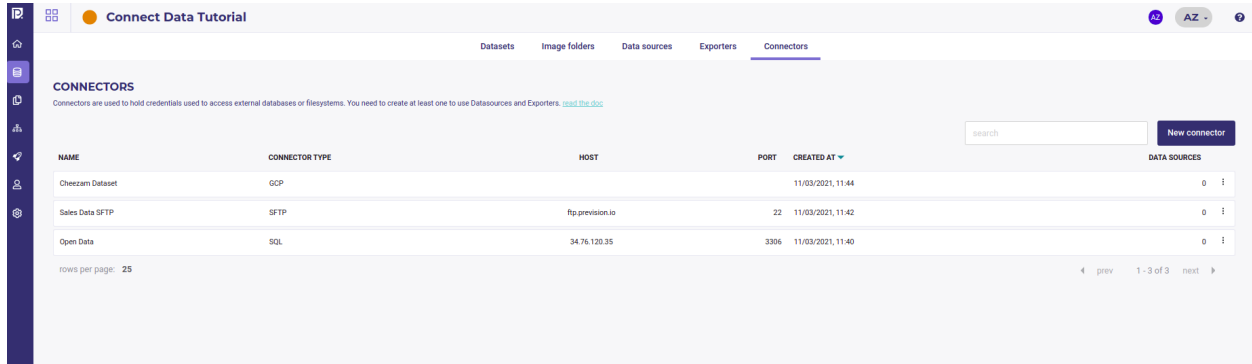
In the Prevision.io platform you can set up connectors in order to connect the application directly to your data sources and generate datasets.

The general logic to import data in Prevision.io is the following:

- Connectors hold credentials & connection information (url, etc.)
- Datasources point to a specific table or file

- Datasets are imported from datasource

You can see all your existing Connectors in the [data section](#) or your project, in the **Connectors tab**



The screenshot shows the 'Connectors' tab in the Prevision.io interface. It features a sidebar with navigation icons and a main content area with a table of connectors. The table has columns for NAME, CONNECTOR TYPE, HOST, PORT, and CREATED AT. There are three connectors listed: 'Cheezam Dataset' (GCP), 'Sales Data SFTP' (SFTP), and 'Open Data' (SQL). A 'New connector' button is visible in the top right corner of the table area.

NAME	CONNECTOR TYPE	HOST	PORT	CREATED AT
Cheezam Dataset	GCP			11/03/2021, 11:44
Sales Data SFTP	SFTP	ftp.prevision.io	22	11/03/2021, 11:42
Open Data	SQL	34.76.120.35	3306	11/03/2021, 11:40

Fig. 15: Connector List

Create Connector

By clicking on the “new connector” button, you will be able to create and configure a new connector. You will need to provide information depending on connector’s type in order for the platform to be able to connect to your database/file server.

Note: TIPS : you can test your connector when configured by clicking the “test connector” button.

SQL

You can connect any SQL Database by providing standard information :

- an host url
- a port
- a username
- a password

FTP and SFTP

FTP and SFTP use the same informations as an SQL database :

- an host url
- a port
- a username
- a password

When creating an FTP or SFTP connector, remember that the connector will open to the root folder of your ftp server so you must use the complete path for datasrouce created from this connectors.

New connector ×

SQL FTP SFTP S3 GCP

Connector name

Host

Port

Login

Password

Test Connector

Save Connector

Fig. 16: Create a new sql connector

New connector ×

SQL FTP SFTP S3 GCP

Connector name

Host

Port

Login

Password

Test Connector

Save Connector

Fig. 17: Create a new ftp connector

Amazon S3

You can get any dataset hosted on Amazon S3 storage by using an access key (see [The Amazon Guide](#) to get yours)

New connector ✕

SQL FTP SFTP **S3** GCP

Connector name

Access key id

Secret access key

Test Connector Save Connector

Fig. 18: Create a new S3 connector

GCP

If you have data hosted on Google Cloud Bucket, you can connect your bucket with the GCP connector.

GCP connector required a json file with your bucket credentials. [See here how to get them](#). Your key should look like that :

```
{
  "type": "service_account",
  "project_id": "project-id",
  "private_key_id": "key-id",
  "private_key": "-----BEGIN PRIVATE KEY-----\nprivate-key\n-----END PRIVATE KEY-----\n",
  "client_email": "service-account-email",
  "client_id": "client-id",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://accounts.google.com/o/oauth2/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/service-
  ↪account-email"
}
```

Once you got your GCP json credentials file, just upload it to connect your Bucket

Once connectors are added, you will find them in the list of all your connectors. You can, by clicking on the action button :


New connector ×

SQL FTP SFTP S3 **GCP**

Connector name
Cheezam Dataset

Key file (json) upload

1 file selected

 cheezam_creds.json
2.25 KB

Test Connector Save Connector

Fig. 19: Create a new GCP connector

- test the connector
- edit the connector
- delete the connector

Once at least one connector is well configured, you will be able to use the data sources menu in order to create CSV from your database or file server.

Data sources

data sources

- need a connector
- may be used as input of a pipeline
- may be used to import dataset

Datasources represent “dynamic” datasets, whose data can be hosted on an external database or filesystem. Using a pre-defined connector, you can specify a query, table name or path to a file to extract the data.

Each datasource is one tabular dataset that can be import to Prevision. All your datasources are available from the datas section of you project, under the Data sources tab.

By clicking the **New datasource** button you can create a datasource from a connector

A datasource is created from a connector thus you always need to select one when creating a new datasource.

Create datasource instead of dataset allows to point to dynamic data and thus schedule train and prediction on always the same datasource while the datas are updated.

DATA SOURCES

Using datasources you can connect your external databases or filesystems and use it in the platform to train new models or to build prediction pipelines. You will first need to configure the access to your data sources through Connectors. [read the doc](#)

search New datasource

NAME	CONNECTOR	CONNECTOR TYPE	DB	TABLE	QUERY	FILE PATH	CREATED AT
Cheezam_labels	Cheezam Dataset	GCP				labels.csv	11/03/2021, 15:50
Labels img product	Sales Data SFTP	SFTP				labels.csv	11/03/2021, 15:50
headlines_massive	Open Data	SQL	prevision		✓		11/03/2021, 15:44
headlines	Open Data	SQL	prevision		✓		11/03/2021, 15:42

rows per page: 25 1 - 4 of 4

Fig. 20: View all my datasource

For example, you can create a “weekly_sales” datasource that is pointing to a database table where data about sales are loaded each Sunday and then schedule a predict from this datasource each Monday.

Table as a datasource

You can create a datasource from any table from an sql connector. When selecting an SQL connector, you will be prompted to input a database and table names :

Using this for train or predict will import all the table. Be warned that if the table is big, the import process may last long. In most of case you'd better use a query.

Query as a datasource

From an SQL connector you can create datasource with an sql query :

Any query can run as long as your source database support it

S/FTP server file as datasource

When using an FTP or SFTP server, you could set an csv file as datasource :

Note that :

- you could only import csv data
- the path to you file starts from the root of your ftp server

New datasource ✕

Data source name

headlines

Connector

Open Data

Connector type

SQL

database

prevision

☐ select by query

table

news

Test Data source

Save Data source

Fig. 21: Import a table from a database

Edit data source ✕

Data source name

headlines_massive

Connector

Open Data

Connector type

SQL

database

prevision

☒ select by query

query

SELECT * FROM news ORDER BY date_part DESC LIMIT 500000;

Test Data source

Save Data source

Fig. 22: Import a table from an SQL Query

New datasource ✕

Data source name

Labels img product

Connector

Sales Data SFTP

Connector type

SFTP

file path

realtime_labels.csv

Test Data source

Save Data source

Fig. 23: Import a file from FTP server

S3 Bucket

If you want to create a datasource from a file in Amazon S3 Storage, you must input name of the bucket and name of the file :

New datasource ✕

Data source name:

Connector:

Connector type: **S3**

bucket:

file path:

Fig. 24: Import a file from an S3 Bucket

Only CSV files are supported and you can not import images folder from S3 Bucket.

Datasource from GCP

GCP Connectors offer two options :

- either point to a file in a bucket
- or point to a table of a big Query database

If you use the storage method, you have to input name of your bucket and name of your file :

New datasource ✕

Data source name:

Connector:

Connector type: **GCP**

Storage: ☒ Storage ☐ BigQuery

bucket:

file path:

Fig. 25: Import a file from a GCP Bucket

When using BigQuery, you must input the name of the dataset and the name of the Table

Once you input all your information, click on **Test Data Source** to test it and then on **Save Data Source** when datasource is fine. The Data Source is created immediately and is displayed in your list of datasource.

Edit data source ✕

Data source name:

Connector:

Connector type: **GCP**

Storage:

BigQuery

Fig. 26: Import a file from a GCP Big Query Table

Note that data are not imported into your account.

You can either *build an experiment* by creating a dataset from your datasource or *build a pipeline* in order to *schedule predictions*

Import your datas

Once you created a datasource, you can import it as a dataset to start to experiment by going to the Datasets tab of your projects's data section :

DATASETS

Dataset are a central resource within the platform. Imported from your files, created by a pipeline or generated from your data sources, it is the starting point of any model. [read the doc](#)

☐ Pipeline output ☐ Datasource ☐ File upload ☐ Pipeline intermediate file

search

NAME	ROWS	COLUMNS	SIZE	CREATED AT	CREATED BY	SOURCE	STATUS	ACTIONS
<input type="checkbox"/> headline_huge				11/03/2021, 15:51	arnold zeghir	Datasource	Copy on going	<input type="button" value="Compute embeddings"/> <input type="button" value="i"/>
<input type="checkbox"/> headlines_massive	300,000	7	136.01 MB	11/03/2021, 15:45	arnold zeghir	Datasource	Ready to be used	<input type="button" value="Compute embeddings"/> <input type="button" value="i"/>
<input type="checkbox"/> headlines	300	7	134.27 KB	11/03/2021, 15:43	arnold zeghir	Datasource	Ready to be used	<input type="button" value="Compute embeddings"/> <input type="button" value="i"/>

rows per page: 25 prev 1 - 3 of 3 next

Fig. 27: List all your datasets

Clicking on the **import dataset** button opens a modal windows where you can select a datasource previously created. By clicking on **save**, the importation will start and your dataset and its status will be displayed in your dataset list. There are 3 status :

1. Copy on going : dataset is copied from remote datasource
2. Dataset Statistics pending : copy is done and dataset analysis is running
3. Ready to be used : you can start to experiment (or launch an embedding)

Import dataset

☒ Use a datasource

☐ Import file

Datasource

headline_huge

Name

headline_huge

Column separator

auto

Decimal separator

auto

Save dataset

Fig. 28: Import Datas from a datasource

Exporters

In the same way that Datasources are used to import data into Prevision.io, Exporters are used when you write the data generated in the platform to an external database or filesystem. They also require a connector, and have similar configuration options.

Once you had create an exporter, you may use it :

- to write the result of a pipeline
- to export one of your dataset

Exporters

- need a connector
 - may be use as output of a pipeline
 - may be used to export dataset
-

You can view all your exporters in the **Exporters** tab of your project's **Data** section

Using an exporter allows you to deliver your dataset transformations and prediction to your user or stakeholder either once or on a scheduled basis.

EXPORTER NAME	CONNECTOR NAME	CONNECTOR TYPE	CREATED BY	CREATED AT	LAST EXPORT DATE	LAST EXPORT STATUS
SFTP Photo	Sales Data SFTP	SFTP	arnold zeghir	11/04/2021, 16:24	11/04/2021, 16:25	●
Bad Exporter	Cheezam Dataset	GCP	arnold zeghir	11/04/2021, 16:22	11/04/2021, 16:22	●
label_cheezam	Cheezam Dataset	GCP	arnold zeghir	11/04/2021, 16:21	11/04/2021, 16:22	●
cheezam	Cheezam Dataset	GCP	arnold zeghir	11/04/2021, 16:18	11/04/2021, 16:19	●

Fig. 29: List Your exporters

Create an exporter

When clicking on the **new exporter** button inside the Exporters tabs, you will be prompted to enter some information to the location you want to export your data when using this exporter, depending on the connector used.

Whatever the connector you used, you need to select an overwrite mode :

- ☒ Add a timestamp
- ☐ Cancel export
- ☐ Overwrite file

Fig. 30: Exporters overwrite options

The options are :

- **Add a timestamp:** a timestamp will be added to the file name. For example *label_cheezam.csv* becomes *label_cheezam_2021-07-12T15:22:29.csv* (or the table name)
- **Cancel export:** if a file or a table with the same name already exists, cancel the export and do not overwrite it.
- **Overwrite File:** overwrite the file or table if exists.

Depending on your usecase, you may choose one of this options. Of course in order to works, your exporter must used a connectotr with write permissions.

FTP and SFTP

For FTP and SFTP, you only have to enter the path where to export your dataset. If it does not exist on the server, it won't be created.

New exporter ×

Exporter name

Connector

Sales Data SFTP × ▾

file path

☒ Add a timestamp

☐ Cancel export

☐ Overwrite file

Cancel

Save

Fig. 31: Create an ftp exporter

S3 and GCP

For Amazon S3 and google bucket, you need to input the name of the bucket and the name of a file for saving.

New exporter ✕

Exporter name
cheezam

Connector
Cheezam Dataset ✕ ▾

bucket
cheezam-feedback

file path
prediction-cheese.csv

☒ Add a timestamp
☐ Cancel export
☐ Overwrite file

Cancel Save

Fig. 32: Create a storage exporter

SQL Database

When exporting to a remote database, you must input both the database name and a table name.

Using an exporter - export your dataset

Typical examples for exporters are :

- delivering predictions to external system
- delivering transformed dataset to external system

Once you had created an exporter, you can use it on any datasets from your dataset list, with the **action button**

When selecting the “**Use an exporter**” action from a dataset, you only have to select the exporter to use and then click the button **export**

New exporter

Exporter name

predict_sales

Connector

Open Data

database

prevision

table

news

☐ Overwrite database content

☒ Add a timestamp

Cancel Save

Fig. 33: Create a db exporter

Connect Data Tutorial
IP AZ

Datasets | Image folders | Data sources | Exporters | Connectors

DATASETS

Dataset are a central resource within the platform. Imported from your files, created by a pipeline or generated from your data sources, it is the starting point of any model. [read the doc](#)

☐ Pipeline output
 ☐ Datasource
 ☐ File upload
 ☐ Pipeline intermediate file

Import dataset

<input type="checkbox"/>	NAME	ROWS	COLUMNS	SIZE	CREATED AT ▼	CREATED BY	SOURCE	STATUS	ACTIONS
<input type="checkbox"/>	headline	300	7	134.27 KB	11/03/2021, 17:21	arnold zephir	Datasource	Ready to be used	+ Compute embeddings
<input type="checkbox"/>	headlines_massive	300,000	7	136.01 MB	11/03/2021, 15:45	arnold zephir	Datasource	Ready to be used	+ Compute embeddings
<input type="checkbox"/>	headlines	300	7	134.27 KB	11/03/2021, 15:43	arnold zephir	Datasource	Ready to be used	+ Compute embeddings

rows per page: 25

< prev | 1 - 3 of 3 | next >

Edit
 Download
 Use an exporter
 Remove

Fig. 34: export form dataset List

Export

Export

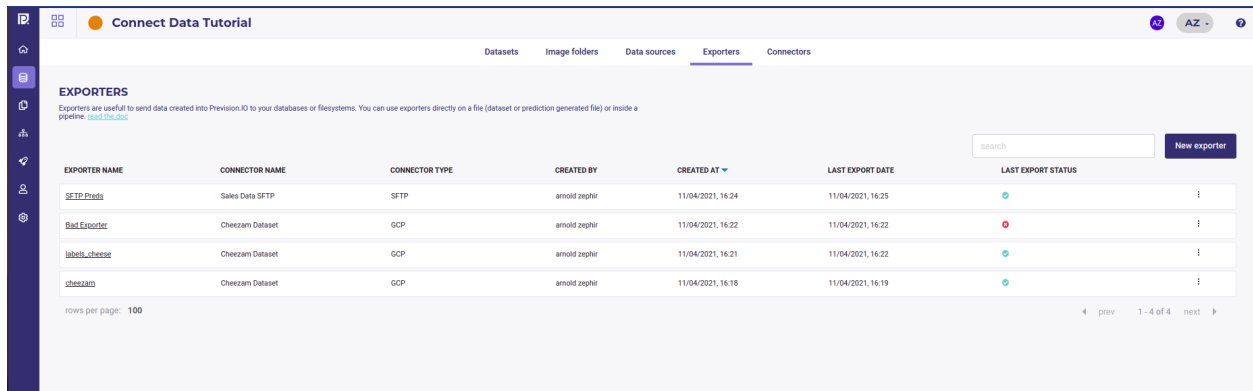
SFTP Prefs

Cancel

Export

Fig. 35: export a dataset

The export of your dataset will start to the location you enter when creating your exporter, with the name input in the exporter. Status of current exportation is available in the list of exporters :

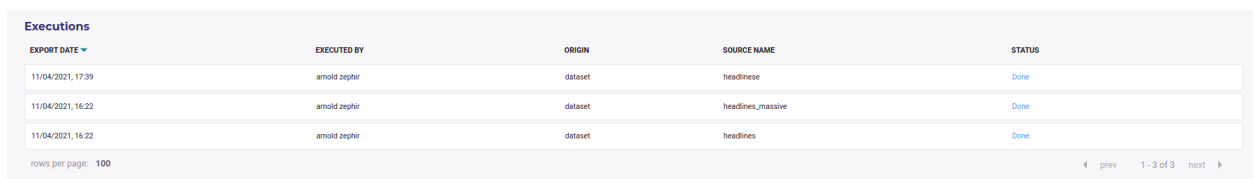


The screenshot shows the 'Exporters' tab in the Prevision.io interface. It displays a table with columns: EXPORTER NAME, CONNECTOR NAME, CONNECTOR TYPE, CREATED BY, CREATED AT, LAST EXPORT DATE, and LAST EXPORT STATUS. There are four exporters listed: 'SFTP Prelo', 'Bad Exporter', 'labels_cheese', and 'cheezam'. The 'Bad Exporter' has a red status icon, while the others have green icons. A 'New exporter' button is visible in the top right.

EXPORTER NAME	CONNECTOR NAME	CONNECTOR TYPE	CREATED BY	CREATED AT	LAST EXPORT DATE	LAST EXPORT STATUS
SFTP Prelo	Sales Data SFTP	SFTP	arnold zephir	11/04/2021, 16:24	11/04/2021, 16:25	Success
Bad Exporter	Cheezam Dataset	GCP	arnold zephir	11/04/2021, 16:22	11/04/2021, 16:22	Failure
labels_cheese	Cheezam Dataset	GCP	arnold zephir	11/04/2021, 16:21	11/04/2021, 16:22	Success
cheezam	Cheezam Dataset	GCP	arnold zephir	11/04/2021, 16:18	11/04/2021, 16:19	Success

Fig. 36: Last export status

And the status all all your export done ever done with one particular exporter is available by clicking on its name in the exporter list :



The screenshot shows the 'Executions' tab, which displays a table of export executions. The columns are: EXPORT DATE, EXECUTED BY, ORIGIN, SOURCE NAME, and STATUS. Three executions are listed, all with a 'Done' status.

EXPORT DATE	EXECUTED BY	ORIGIN	SOURCE NAME	STATUS
11/04/2021, 17:39	arnold zephir	dataset	headlines	Done
11/04/2021, 16:22	arnold zephir	dataset	headlines_massive	Done
11/04/2021, 16:22	arnold zephir	dataset	headlines	Done

Fig. 37: Exporter history

Usage in a Pipeline

Any exporter can be used in a *pipeline* as an output node, meaning that each time the pipeline will be executed, the exporter will be used to write the result of the pipeline execution on a remote location

The exporter to used for your pipeline will be input when configuring a schedule run, in the configuration screen :

Then each time your pipeline is executed, the exporter will write the pipeline output to the location you use as exporter parameter when creating it (either a file location or a database table)

5.3.1.3 Experiments

Note: An experiment represents a collection of trained *Models*, that were created either by the *AutoML Engine*, or from *External Model*. By *Versionning your experiments*, you will /studio/experiments/evaluating and compare different models against different metrics with details on predictive power of each *Features*.

This models may come from the Prevision AutoML engine or be imported in the Project as an *External Model*.

Any model of your experiments can then be used for *Deployments* as REST APIs or included in a *Pipelines*

Regarding the problematic and the data type you have, several training possibilities are available in the platform :

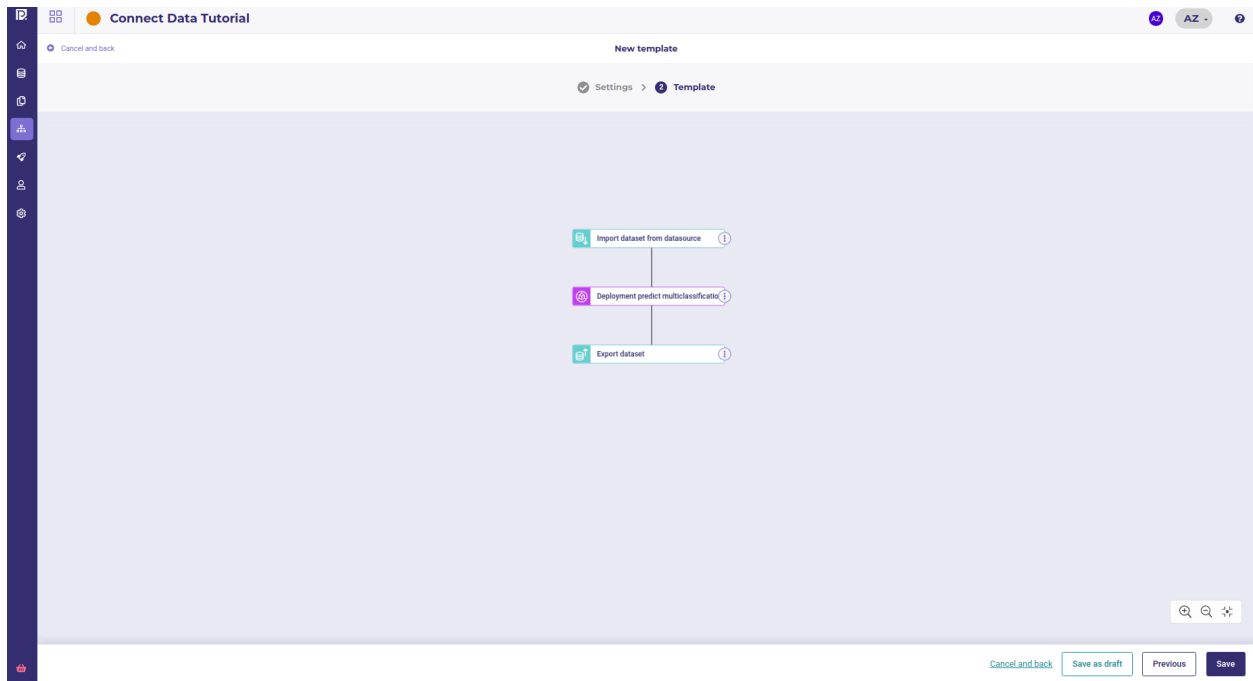


Fig. 38: A typical pipeline : import data, make a prediction and export data

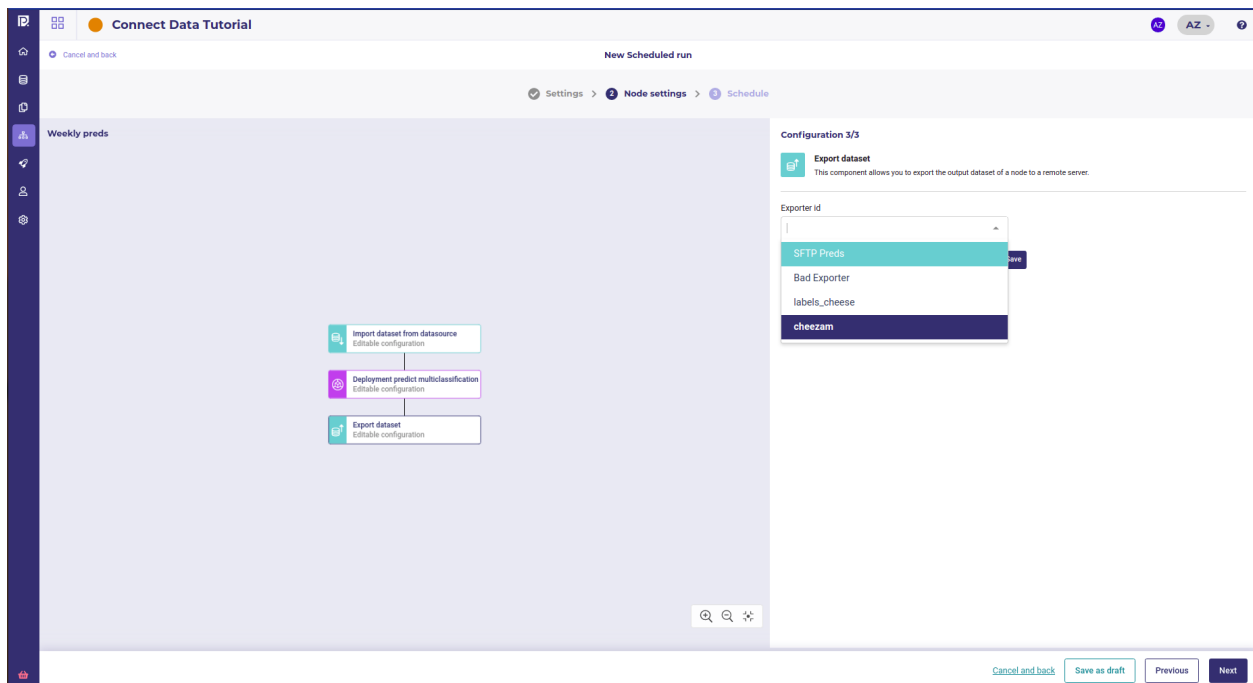


Fig. 39: Using a user defined exporter in a schedule run

Table 2: Experiments list and restrictions

Training type / Data type	Tabular	Time-series	Images	Available for External Model ?	Definition	Exemple
Regression	Yes	Yes	Yes	Yes	Prediction of a quantitative feature	2.39 / 3.98 / 18.39
Classification	Yes	No	Yes	Yes	Prediction of a binary quantitative feature	« Yes » / « No »
Multi Classification	Yes	No	Yes	Yes	Prediction of a qualitative feature whose cardinality is > 2	« Victory » / « Defeat » / « Tie game »
Object Detection	No	No	Yes	No	Detection from 1 to n objects per image + location	Is there a car in this image ?
Text Similarity	Yes	No	No	No	Estimate the similarity degree between two text	

List your experiments

All your experiments are related to a *project* so in order to Create a new experiment, you need first to create a *project* .

On the homepage of your project, you can see a summary of your project resources and access to the experiments dashboards from the left sidebar

<

which allows you to navigate and filter all your project's experiments from the experiments table

Each row gives you :

- the name of your experiment, that links to the experiment dashboard
- the source of the models of your experiment, AutoML or external models
- the latest version of your experiment

The screenshot shows the Prevision.io interface for a project named 'Ventes'. On the left is a dark sidebar with navigation icons. The main area displays a table of experiments. The table has columns: NAME, SOURCE, VERSION, CREATED AT, CREATED BY, DATA TYPE, TRAINING TYPE, SCORE, MODELS, PREDICTIONS, and STATUS. A search bar and a 'New experiment' button are at the top right. The table contains one row with the following data:

NAME	SOURCE	VERSION	CREATED AT	CREATED BY	DATA TYPE	TRAINING TYPE	SCORE	MODELS	PREDICTIONS	STATUS
experiment_12345	AutoML	1	10/07/2021, 10:56	arnold zephir	Tabular	Regression	-	0	0	

Below the table, it says 'rows per page: 25'. At the bottom right of the table area, there are navigation links: 'prev', '1 - 1 of 1', and 'next'.

- the creation date and time of the experiment
- its creator (see [Contributors](#))
- the datatype ([Structured Data](#), [Computer Vision Model](#) or [Time series](#))
- the training type (Regression, Classification, Multi-classification, Object Detection or [Natural Language Processing](#))
- score : the choosen metrics of the last version, the type of metrics and a 3-stars evaluation
- the number of models built into the last version of your experiment
- the numbers of predictions done over the last version
- and the status (running, paused, failed or done)

Create a new experiment

Note: Experiment is a way to group several modelisation under a common target in order to compare them and track progress. An experiment may have one or more [Versionning your experiments](#) and you can change any parameter you want from version to version (Trainset, features used, metrics,...). The only constant between experiment are :

- the target used. Once you had selected your target, you cannot change it and muste create a new experiment if you want to try a new one
- the Engine used, AutoML or [External Model](#)

Once you had created a project, you can create a new experiment from the project Dashboard or the list of experiments, by clicking on the “Create experiment” in the bottom left corner of project dashboard or on the “New experiment” in the experiments list.

AutoML Engine and ONNX import

The first things asked will be to choose between AutoML or external model and to give a name to your experiment. AutoML use Prevision.io engine to choose and built the best model without human intervention from the shape and type of your data. External model is the mode for importing models saved as onnx file. Both allow to evaluate, deploy and monitor your model.

The screenshot shows the 'New experiment' page in the Prevision.io interface. The top bar includes the Prevision.io logo, the name 'Ventes', and user information 'AZ'. The main area is titled 'New experiment' and contains a form with the following sections:

- Model Selection:** Two tabs, 'AutoML' (selected) and 'External model'.
- Name:** A text input field with the placeholder 'Experiment name'.
- Data type:** Three buttons: 'Tabular' (selected), 'Timeseries', and 'Images'.
- Training type:** Four buttons: 'Regression' (selected), 'Classification', 'Multi-classification', and 'Text similarity'.

On the right side, there is an 'AUTOML' section with the following text:

AUTOML

The Prevision.io AutoML engine allows you to quickly benchmark and optimize a range of open source algorithms to get highly performant models.

What do I need?

You just need data, imported into Prevision.io as a Dataset. Your dataset just needs to contain a target column (and a temporal one, if you are working with time series), and you are good to go.

What's next?

Once the experiment and the models are created, you can analyze, version, and deploy them to create a prediction API endpoint, or use it with pipelines to schedule batch predictions.

At the bottom right, there are two buttons: 'Cancel and back' and 'Create Experiment'.

You can then select your Data type and problem type remembering this restrictions :

Table 3: Experiments list and restrictions

Training type / Data type	Tab-ular	Time-series	Im-ages	Available for External Model ?	Definition	Exemple
Regression	Yes	Yes	Yes	Yes	Prediction of a quantitative feature	2.39 / 3.98 / 18.39
Classification	Yes	No	Yes	Yes	Prediction of a binary quantitative feature	« Yes » / « No »
Multi Classi-fication	Yes	No	Yes	Yes	Prediction of a qualitative feature whose cardinality is > 2	« Victory » / « De-feat » / « Tie game »
Object Detec-tion	No	No	Yes	No	Detection from 1 to n objects per image + location	Is there a car in this image ?
Text Similar-ity	Yes	No	No	No	Estimate the similarity degree between two text	

You can learn more about AutoML and external models on their dedicated section :

AutoML

Prevision.io platform can train model based on your experiment parameters. The AutoML Engine make analysis of your dataset and :

- builds the best feature engineering given your datatype (for example : convert text and images to embedding or build lags auto on time serie)
- choose the fittest algorithm given your data
- choose the best parameters for each algorithm
- may blend and combined many model to get performance

When choosing AutoML, you can tune some configuration but the most important are those on the “Basics” tabs :

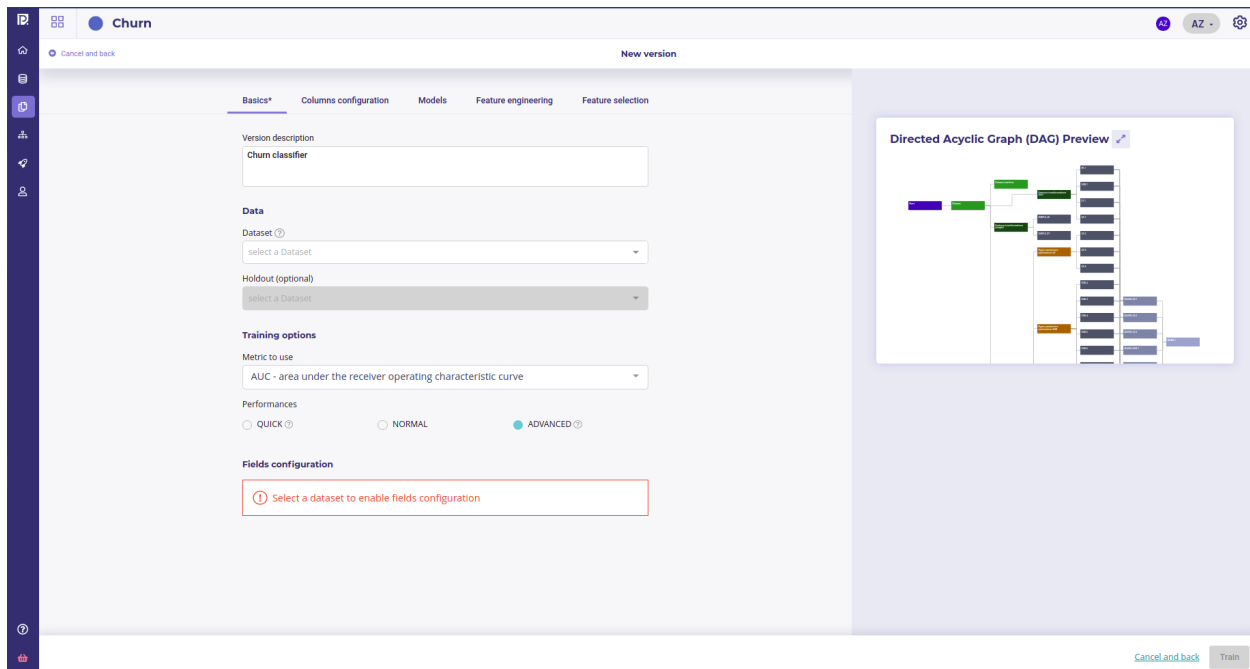


Fig. 40: Basic configuration

- the Metric to optimise (see [our guide on how many time you should spend on choosing the metrics](#))
- the performances profil :
 - Fast: get a result as fast as possible
 - Normal : Spend more time in hyper opt
 - Advanced : get the best result (spent many time son hyper opt and blending models)

General advice is to start project with a “Fast” profile and go for advanced train when the problem and features are completely defined.

Most of the configuration is common accross the training type and Data types except the metrics that depends on the type of problem, but some of them have specificities, especially on the metrics and the available models

Note that when you create a new experiment, you will be prompted to create a *first version*

See more about each type of training on dedicated section :

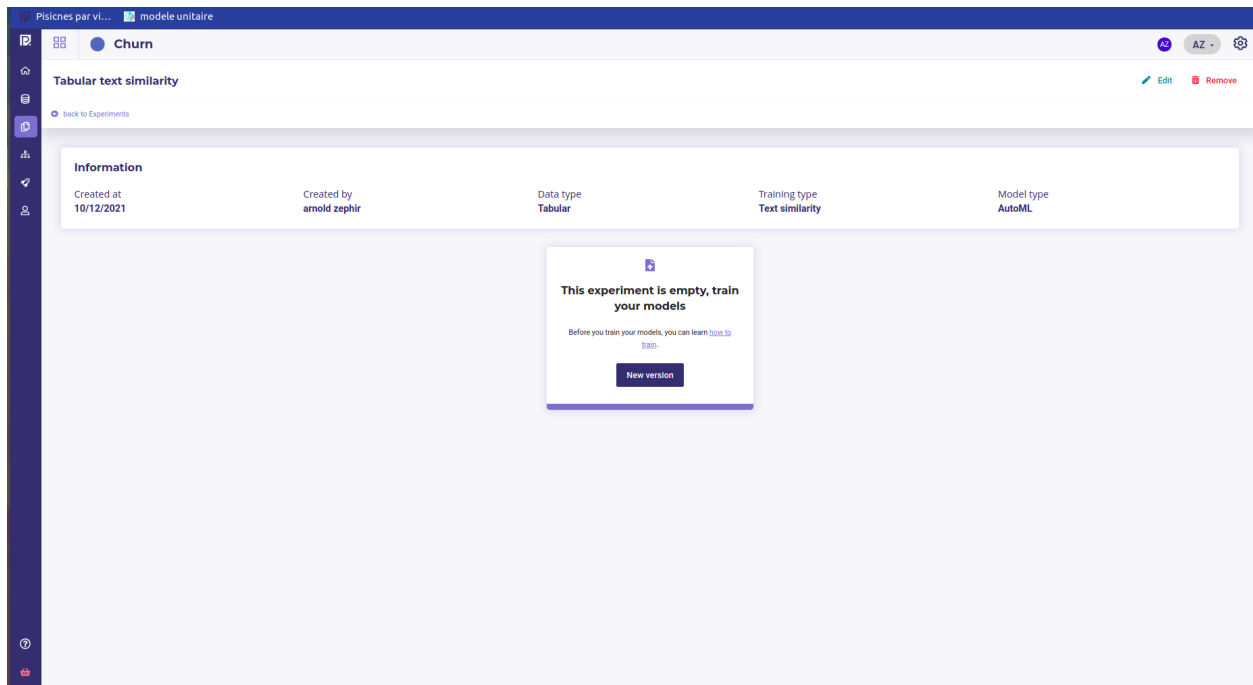


Fig. 41: Empty experiment

Structured Data

Tabular data use data in columns to build model. There 3 kind of probleme type, Regression, classification, multi classification and a special one, *text similarity*

This section explains the parameters for regression, classification and multi classification. For text similarity, see the *dedicated section*

Basics tabs

The basics tab group all the important parameters :

- dataset : the *dataset* from your datas assets to train on. This parameter is mandatory for automl
- holdout : holdout is another *dataset* that will be used to evaluate your model. It should have the same features and target than the trainset but with sample never seen in the trainset. It is optional but strongly recommended, especially at the end of your experiments, to check your model stability
- metrics : the options depends on your training type but you may choose one of the standard Machine Learning metrics for objective
- Performances : quick for qui result, Advanced for best result. Be aware that advanced option could lead to more than 24 hours of training if the trainset is big. Quick get result in less than 3 hours in mot of case.
- Target : set the column of your trainset to predict. Note that the platform filter available target based on your problem type (example : it expects the target too have 2 modalities only if the problem is a classification)
- ID Column : you can set a column as an id. It must have only unique values The column set as an id will not be used as a feature and will be repeat on subsequent prediction to serve as a join column. If you do not set an ID column, an index will be generated.

The screenshot shows the 'ONNX models' interface with the 'Basics*' configuration tab selected. The interface includes a 'Version description' field, a 'Data' section with 'Dataset' (validation_churn_bank) and 'Holdout (optional)' (select a Dataset) dropdowns, a 'Training options' section with 'Metric to use' (RMSE - root mean squared error) and 'Performances' (QUICK, NORMAL, ADVANCED) radio buttons, and a 'Fields configuration' section with 'Target column', 'ID column (optional)', 'Weight (optional)', and 'Fold (optional)' dropdowns. A 'Directed Acyclic Graph (DAG) Preview' is shown on the right. The bottom right corner has 'Cancel and back' and 'Train' buttons.

Fig. 42: Basics configuration

- **Weight** : you can set a column to be used as a Weight column for sample. The sample with large weight will be favored during the training. If you do not set weight, the system apply a balanced trainign, meaning it applies larger weight to the less frequent target modalities.
- **Fold** : Fold colum will be used to generate cross validation. If you do not know how to generate correct fold, leave this empty. Otherwise, you may put a fold number (integer) in this column and the Cross Validation will be run by splitting the dataset against this folds.

The two mandatory parameters are Dataset and Target. Once had set them up you can proceed and launch train by clicking on the **trian** button in the right-lower corner.

Columns configuration

The second bottom left panel allows you to ignore some columns of your dataset. To do it, just deselect unwanted features. Please note that you can search by features name the columns you want to unselect and use the “select/unselect all” checkbox to apply your choice to the selection.

Why drop features ?

In most of case, you should not drop any features. The Prevision Platform can handle up to several thousands of features and will keep those meaningful to build its models. Yet, there is two ase you would drop some features :

- Some Features has too much importance in the model and you suspect that it is in fact a a covector of the target, or that you have an important leakage. If you see that your simplest model (Simple LR) perform as well as complex one and that some feature as a big feature importance, drop it
- you want to get some result fast. Dropping features allows for faster training. If you suspect that some features bears low signal, drop it at the start of your project to iterate faster

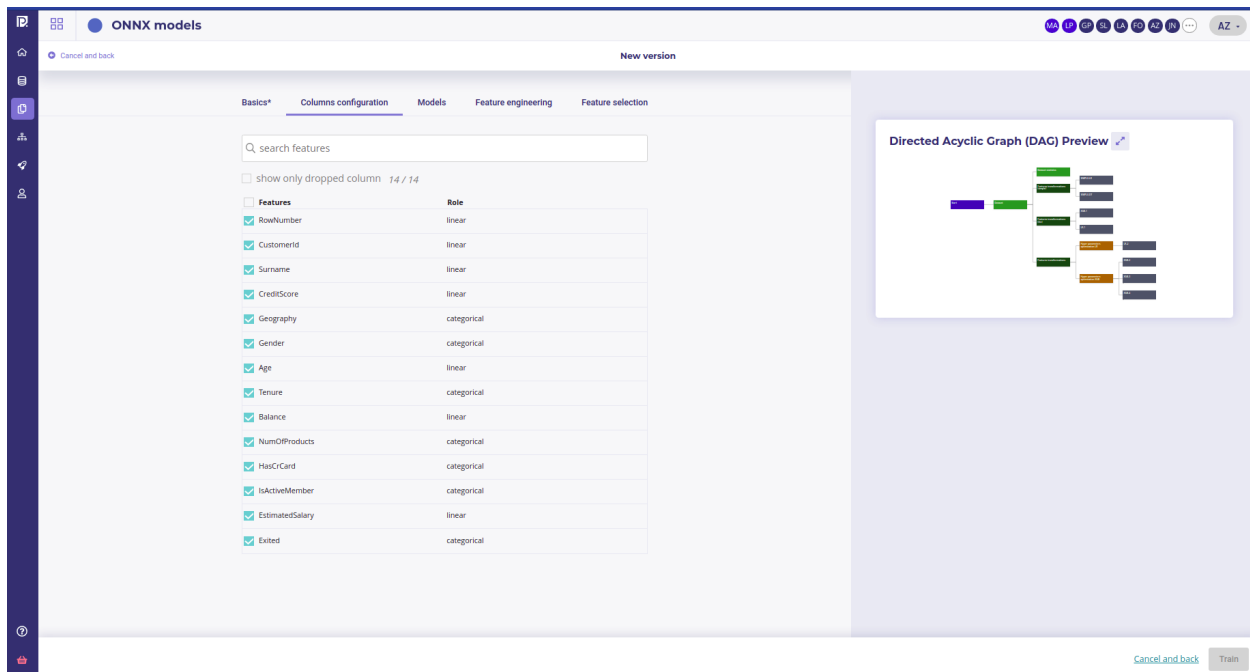


Fig. 43: Basics configuration

Models

The model selection area allows you to select the type of model you want to train. You got 3 sections.

Simple Models

Simple models are models done with no complexe optimisation and using simple algorithm, a linear regression and Random Forest with less than 5 split. They allow to check if the problem is treatable with very simple algorithm instead of fancy Machine Learning. Moreover, simple model generate :

- a Chart that explain model and is human readable
- python code to implement it
- SQL Code to implement it

You can unselect simple model but it is recommended to keep them when starting a project and watch how good they perform vs more complex model. If a simple Decision Tree performs as good as a Blend of XGBoost, or only marginally worst, favor the simple model.

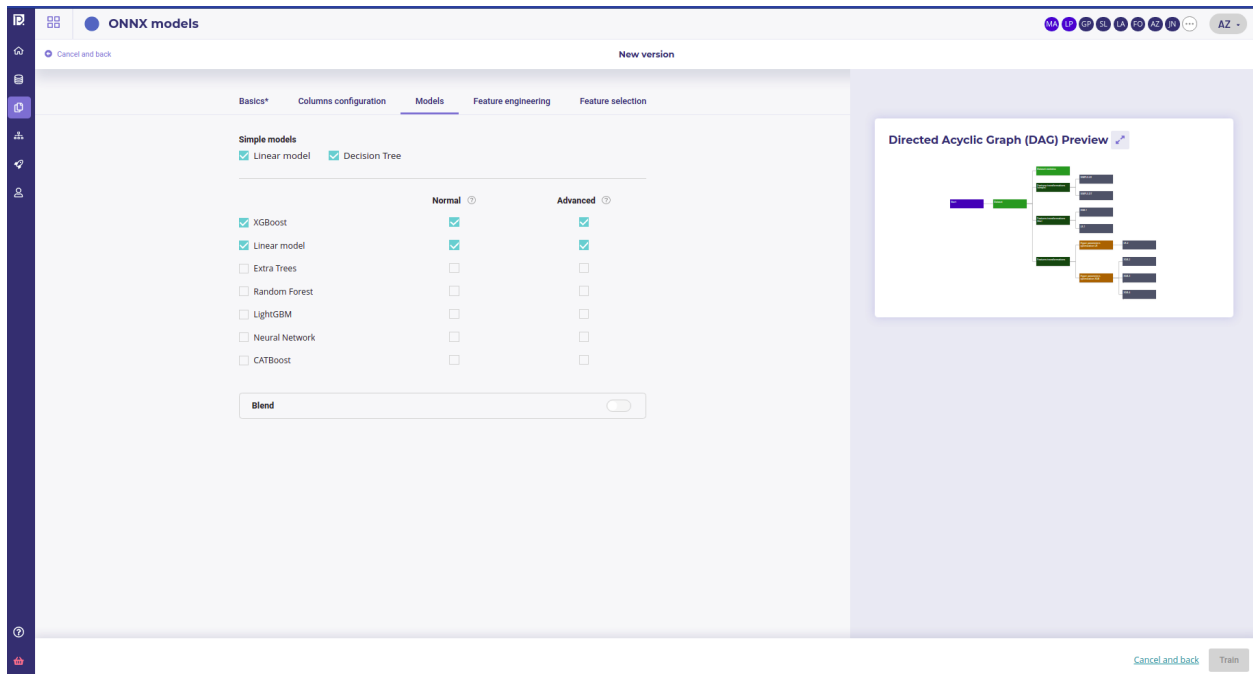


Fig. 44: Model tab

Model Selection

You can choose the algorithm type that the automl engine will use. The more you select, the longer is the train. All the selected algorithm will be used in blend

Note that only if :

- there is at least one column with text
- the probleme type is a classification or a multi classification

you can select Naive Bayes Model

Blend

Blending model is a powerful technique to get the most performance yet it can be quite long to train. Blend use a model over all the others to merge and combine them in order to get the best performance. Switch the option if you want to blend your models but be aware that resulting train will last very long.

Feature engineering

In this section, you could select more feature engineering (or unselect some).

Four kinds of feature engineering are supported by the platform. :

- Date features : dates are detected and operations such as information extraction (day, month, year, day of the week, etc.) and differences (if at least 2 dates are present) are automatically performed
- Textual features :

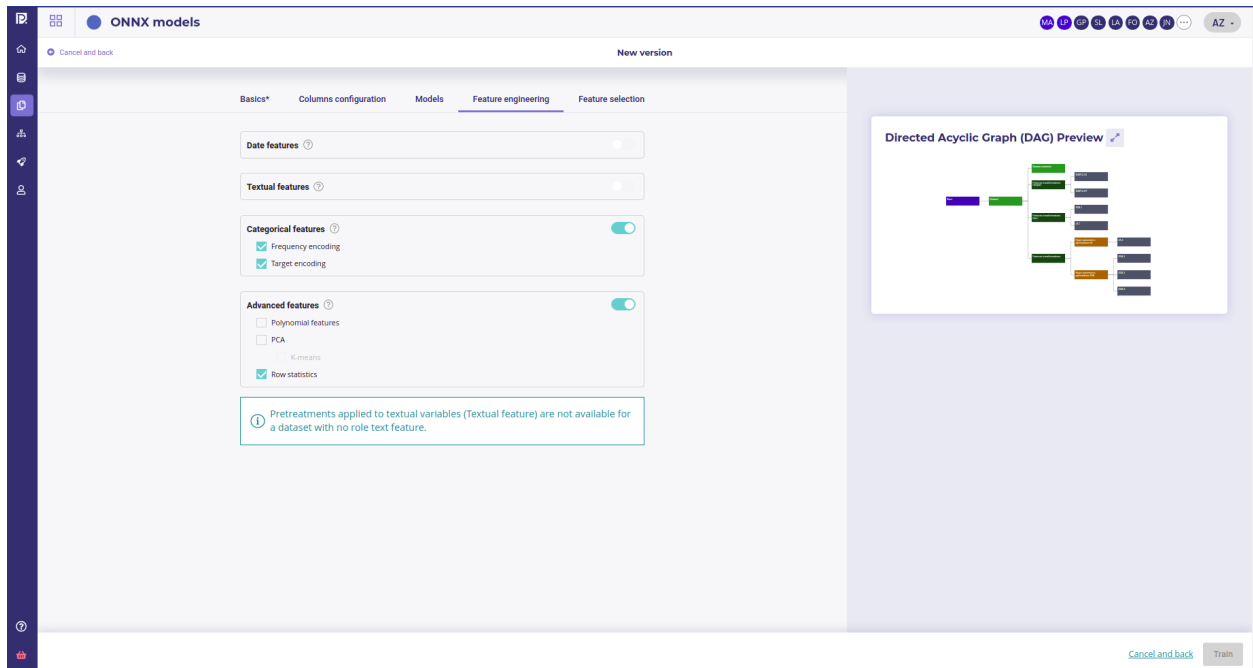


Fig. 45: Model tab

- Statistical analysis using Term frequency–inverse document frequency (TF-IDF). Words are mapped to numerics generated using tf-idf metric. The platform has integrated fast algorithms making it possible to keep all uni-grams and bi-grams tf-idf encoding without having to apply dimension reducing. More information about TF-IDF on <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>
- Word embedding approach using Word2Vec/Glove. Words are projected a dense vector space, where semantic distance between words are : Prevision trains a word2vec algorithm on the actual input corpus, to generate their corresponding vectors. More information about Word embedding on https://en.wikipedia.org/wiki/Word_embedding
- Sentence Embedding using Transformers approach. Prevision has integrated BERT-based transformers, as a pre-trained contextual model, that captures words relationships in a bidirectional way. BERT transformer makes it possible to generate more efficient vectors than word Embedding algorithms, it has a linguistic “representation” of its own. To make a text classification, we can use these vector representations as input to basic classifiers to make text classification. Bert (base/uncased) is used on english text and Multi Lingual (base/cased) is used on french text. More information about Transformers on [https://en.wikipedia.org/wiki/Transformer_\(machine_learning_model\)](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)). The Python Package used is Sentence Transformers (https://www.sbert.net/docs/pretrained_models.html)
- Categorical features:
 - Frequency encoding: modalities are converted to their respective frequencies
 - Target encoding: modalities are replaced by the average (TARGET, grouped by modality) for a regression and by the proportion of the modality for the target’s modalities in the context of a classification
- Advanced features:
 - Polynomial features: features based on products of existing features are created. This can greatly help linear models since they do not naturally take interactions into account but are less usefull on tree based models
 - PCA: main components of the PCA
 - K-means: Cluster number coming from a K-means methode are added as new features

- Row statistics: features based on row by row counts are added as new features (number of 0, number of missing values, ...)

Please note that if you don't have a feature of one of these feature types in your train dataset, the corresponding feature engineering toggle button will be disabled. Also please note that textual features pretreatments only concern advanced models and normal Naive Bayes model

Feature selection

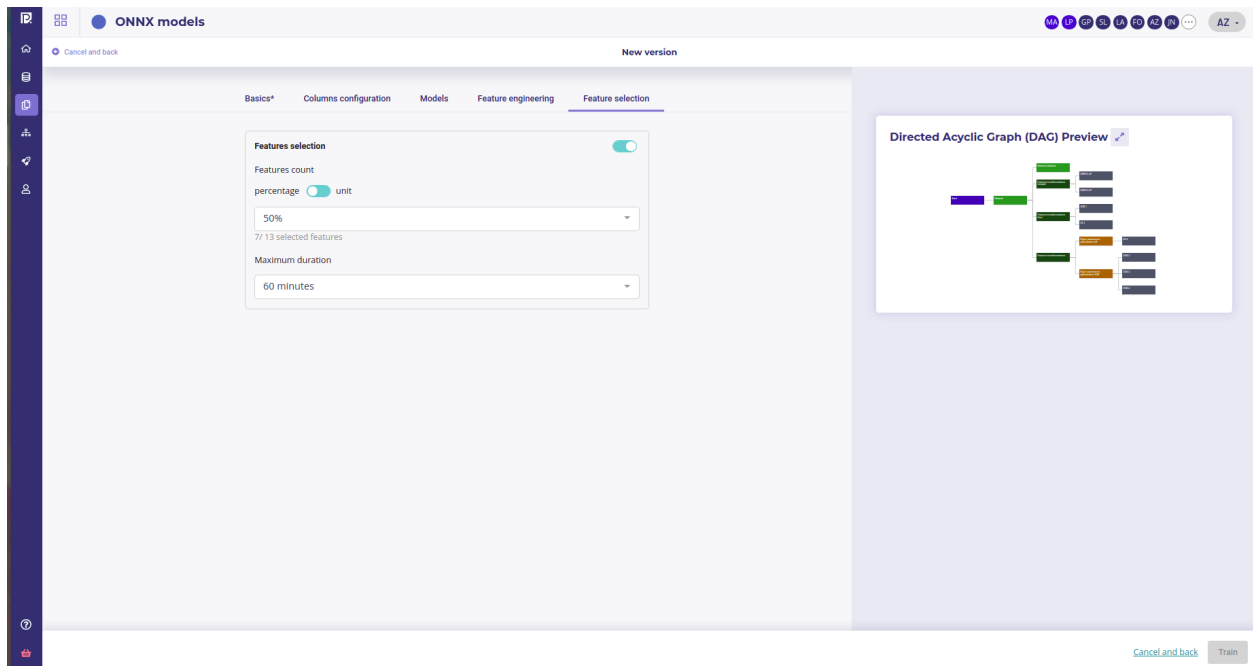


Fig. 46: Feature Selection

In this part of the screen you can choose to enable feature selection (off by default).

This operation is important when you have a high number of features (a couple hundreds) and can be critical when the number of features is above 1000 since the full Data Set won't be able to hold in RAM.

You can choose to keep a percentage or a count of feature and you can give a time budget to Prevision.io's to perform the search of optimal features given the TARGET and all other parameters. In this time, Prevision.io will subset the feature of the Data Set then start the classical process.

The variable selection strategy in Prevision.io is hybrid, depends on the characteristics of the dataset and the time available.

1. It is hybrid because it combines both so-called filtering methods, encapsulation methods and integrated methods. The filtering methods perform the selection of entities independently of the construction of the classification / regression model. Encapsulation methods iteratively select or eliminate a set of entities using the metric of the classification / regression model. In built-in methods, feature selection is an integral part of the classification / regression model.
2. It depends on the characteristics of the dataset and the time allotted. In fact, depending on the volume of the dataset, a small data strategy is applied for a dataset of less than 8 GB, fully in memory. Otherwise, a big data strategy is applied.
3. In a small data situation, a first filtering approach is carried out consisting in filtering the variables of zero variance, the duplicated variables, the intercorrelated variables beyond 99% and the variables correlated to

the target variable beyond 99% . Depending on the time remaining available, a second so-called encapsulation method is carried out using a LASSO-type regularization on the entire dataset by cross validation with the aim of optimizing the metric selected when the use case is launched.

4. In a big data situation, as time permits, several row and column samplings are carried out and the stages of filtering, encapsulation method and integrated methods completed by a reinforcement learning strategy are successively launched. . The variables are then ranked in order of priority according to the different approaches tested and the top variables, at the threshold defined by the user, are sent to the various algorithms of Prevision.io.

Natural Language Processing

Fig. 47: Text Similarity parameters

Even if considered as a training type for tabular data type, text similarity experiments are particular and need specific training options. Text similarity models allow to retrieve textual documents from a query. For example, from “Red shoes for girls” query, your model should return a corresponding item.

Creating a Text similarity Model

In order to train a text similarity model you **must have** a trainset (dataset dropdown menu) with :

- a description column : some column with text that describes items you want to query (*Description column* dropdown menu)
- an id column : only column with unique ID could be selected (*ID column*)

To get better evaluation you **should have** a query dataset (queries dropdown menu) with :

- a textual column containing user queries that should have match with some item description (*query column* dropdown menu)

- a column with the id of the item whose description should have match the query (*Matching ID column in the description dataset dropdown menu*)

Your queries dataset **could have** its own ID column (*ID Column dropdown menu*)

Note that the drop down to select column only appear when you had selected Dataset and/or a Queries

quora_items_short

Back to list

General Columns Sample

ITEM_DESC	ITEM_ID
A direct competitor of mine in a geographically based service market approached me to buy their business. What should I think of this?	251,113
Who is the best TV actor/actress?	77,217
Which is better in term of Performance & Power, laptop i7 or desktop i7 for Graphic designing & Website Designing?	116,204
Why does Ladder Logic fall short when tasks become complex and why isn't it the best for modular PLC programming?	185,384
Was Hitler a Nazi?	138,632
What is my vocal type? I'm a male and my vocal range is A2-C6?	480,557
Is it possible to buy a 30 lakh car for a 24 year old with a current salary of 15k?	258,133
When you are an adult, what do you call a girlfriend?	246,590
How can I reduce breast size?	217,402
Does 1,000,000 exist?	328,290

Fig. 48: A dataset with items and their description

You then have to select a metric :

- **Accuracy at k:** Is the real item corresponding to a query present in the search result, among the k items returned? The value is a percentage calculated on a set of queries.
- **Mean Reciprocal Rank (MRR) at k:** Similar to accuracy at k. However the score for each query is divided by the rank of appearance of the corresponding item. Example: If for a query the corresponding item appears in third position in the returned list, then the score will be $\frac{1}{3}$. If it appears in second position the score will be $\frac{1}{2}$, in first position the score will be 1, etc. https://en.wikipedia.org/wiki/Mean_reciprocal_rank
- **K results :** the number of query like items that the tool must return during a search. Value between 1 and 100.

Text similarities models options

Text similarity module has its own modeling techniques and is composed of 2 kinds of models :

- embedding model to make a vector representation of your data
- search models to find proximity between your queries and product database

Embedding model / word vectorization

Term Frequency - Inverse Document Frequency (TF-IDF): Model representing a text only according to the occurrence of words. Words not very present in the corpus of texts will have a greater impact. <https://fr.wikipedia.org/wiki/TF-IDF>

Transform: Model representing a text according to the meaning of words. In particular, the same word will have a different representation according to the other words surrounding it. [https://en.wikipedia.org/wiki/Transformer_\(machine_learning_model\)](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model))

Transformer feature-based: Transformer that has been trained upstream on a large volume of data, but has not been re-trained on the corpus in question.

nlp

quora_queries

Back to list

GeneralColumnsSample

TRUE_ITEM_ID	QUERY
241,981	Is a PhD in chemistry worth pursuing?
96,178	Who is the most badass footballer ever?
429,697	What are things which I can export from India?
532,049	What is the best question and answer platform?
184,530	What are some good books for web developers to learn about design?
95,514	What are some activities that people (adults) with various disabilities can all enjoy?
395,794	Why is my Yahoo! sign in failing?
67,925	What are some examples of controversial topics in special education?
150,661	How many times do you have to smoke weed before you get high?
404,641	Can I do an MBA after BE?

Fig. 49: A dataset with user queries and the item id that should have match

Cancel and back

New version

Basics*ModelsFeature engineering

	TF-IDF	Transformers	Transformers fine-tuned
Brute force	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Cluster pruning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
IVF OPQ	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
LSH	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
HKM	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Combination of greyed out algorithms is not possible

Directed Acyclic Graph (DAG) Preview

Cancel and back

Train

Fig. 50: Available algorithm for text similarity models

Fine-tuned transform: A transform that has been trained on a large volume of data and then re-trained on the text corpus in question.

Search models

Brute Force: Exhaustive search, i.e. each query is compared to the set of item descriptions.

Locality sensitive hashing (LSH): exhaustive search. Vectors are compressed to speed up distance calculations. https://fr.wikipedia.org/wiki/Locality_sensitive_hashing

Cluster Pruning: non-exhaustive research. Item descriptions are grouped by cluster according to their similarity. Each query is compared only to the queries of the closest group. <https://nlp.stanford.edu/IR-book/html/htmledition/cluster-pruning-1.html>

Hierarchical k-means (HKM): non-exhaustive research. The idea is the same as for the previous model, but the method used to group the items is different.

Inverted File and Optimized Product Quantization (IVFOPQ): non-exhaustive search. The idea is the same as for the two previous models, but the method used to group the items is different. Vectors are also compressed to speed up distance calculations.

Please note that in order to guarantee the performance of IVF-OPQ models, a minimum of 1000 unique IDs in the train dataset is required.

Text similarity Preprocessing

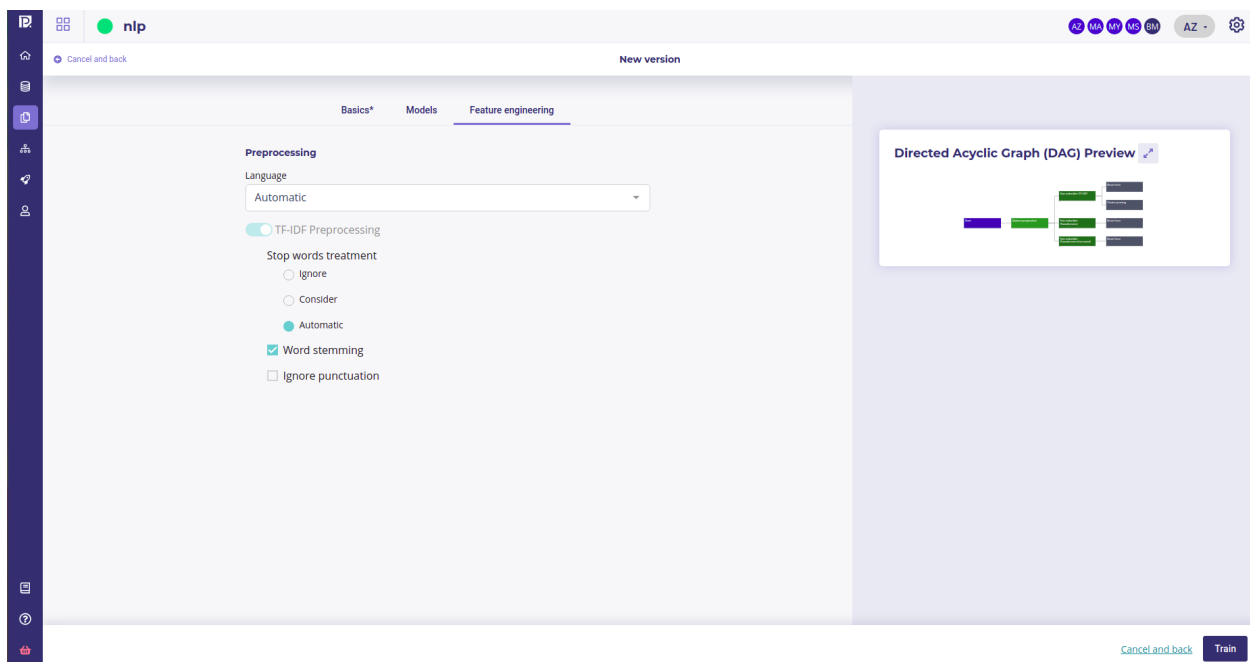


Fig. 51: Feature engineering for text similarities

Several preprocessing options are available :

- Language : you can force the training dataset language to english or french or, let the platform determines by itself between these two languages

- Stop words treatment : you can choose if the platform has to ignore or consider the stopwords during the training. As for the language, you can also let the system makes it own decision by selecting “automatic”
- Word stemming : **stemming** is the process of reducing inflected (or sometimes derived) words to their **word stem**, base or **root** form—generally a written word form.
- Ignore punctuation : by activating this option, the punctuation will not be considered during the training

Watching my Text similarity experiments

Text similarity experiments will be available in the same way that standard tabular data experiments, by clickin on it the list of your experiments yet it has some specificities.

First one, when you select a model, is the model evaluation chart. It shows the performance evolution along the expected rank

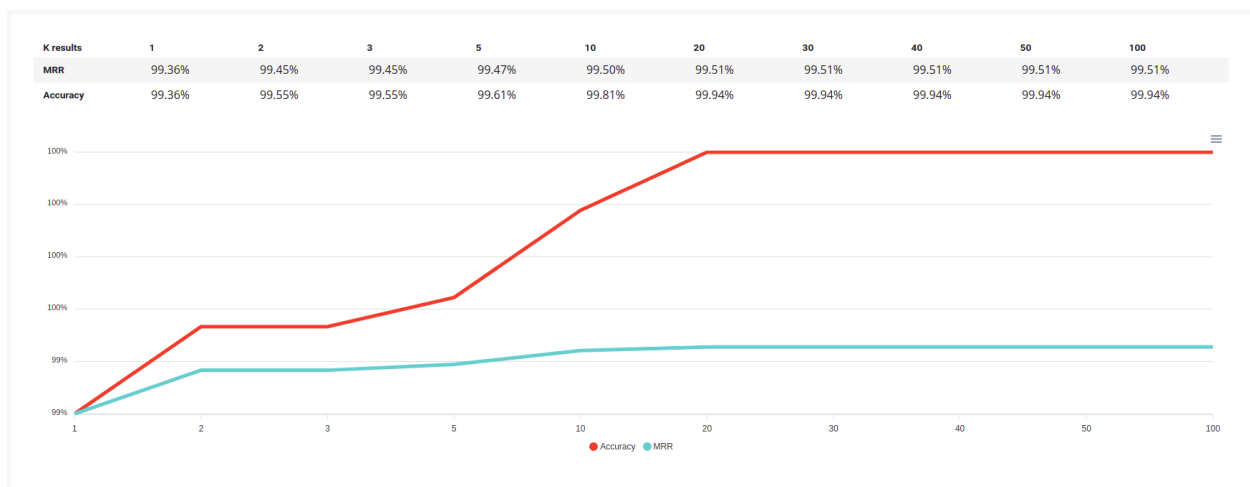


Fig. 52: Text similarity models

Second difference is the prediction tab, which is slightly different from other :

Prediction Bulk

SELECT A MODEL: brute_force-tf_idf (0.9994)

SELECT A DATASET:

K RESULTS: 10

DESCRIPTION COLUMN:

MATCHING ID COLUMN (OPTIONAL):

ID COLUMN (OPTIONAL):

Launch (bulk) prediction

User Generated Predictions


NAME	CREATED AT	MODEL	SCORE	VALIDATION SCORE	ROW COUNT	DURATION	CREATED BY	STATUS
Netflix_Catalog	10/19/2021, 10:51	brute_force-tf_idf	0.9994				arnold zephir	

rows per page: 25

Fig. 53: Txt similarity predictions

Time series

In the prevision.io platform you have the possibility to train time series experiment in order to do forecasting predictions. By selecting in the new experiment screen the timeseries data type you will access the timeseries experiment configuration.



studio/experiments/automl/img/slected_ts.png

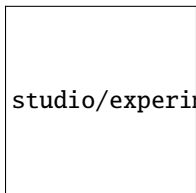
Timeserie experiment configuration

Time series is very similar to tabular experiment except:

- There is no hold out
- There is no weight
- There is no fold (in this case, Prevision.io use temporal stratification)

However, you will find some new notions:

- Temporal column: the feature that contains the time reference of the time series. Since date formats can be complex, Prevision.io supports ISO 8601 (https://fr.wikipedia.org/wiki/ISO_8601) as well as standard formats (e.g. DD/MM/YYYY or DD-MM-YYYY hh:mm).
- Time step: period between 2 events (within the same group) from the temporal column (automatically detected)
- Observation window: illustrate the period in the past that you have for each prediction * Start of observation window: the maximum time step multiple in the past that you'll have data from for each prediction (inclusive, 30 by default) * End of the observation window: the last time step multiple in the past that you'll have data from for each prediction (inclusive, 0 by default that means that the immediate values before the prediction time step is known)
- Prediction window: illustrate the period in the future that you want to predict * Start of the prediction window: the first time step multiple you want to predict (inclusive, 1 by default which means we will predict starting at the next value) * End of the prediction window: the last time stamp multiple you want to predict (inclusive, 10 by default which means we will predict up to the 10th next value)
- A priori features: features whose value is known in the future (customer number, calendar, public holidays, weather...)
- Group features: features that identify a unique time serie (e.g. you want to predict your sales by store and by product. If you have 2 stores selling 3 products, there are 6 time series in your file. Selecting features « store » and « product in the group column allows Prevision.io to take into account these multiple series)



studio/experiments/automl/img/ts_fields.png

Please note that advanced options work the same way than for tabular experiments. Please read the corresponding readthedoc section in order to configure your time series experiment.

Computer Vision Model

In the prevision.io platform, you can train models using images for :

- regression
- classification (either simple one with 2 class or multiclassification with more than 2 class)

And a Special one :

- image detection

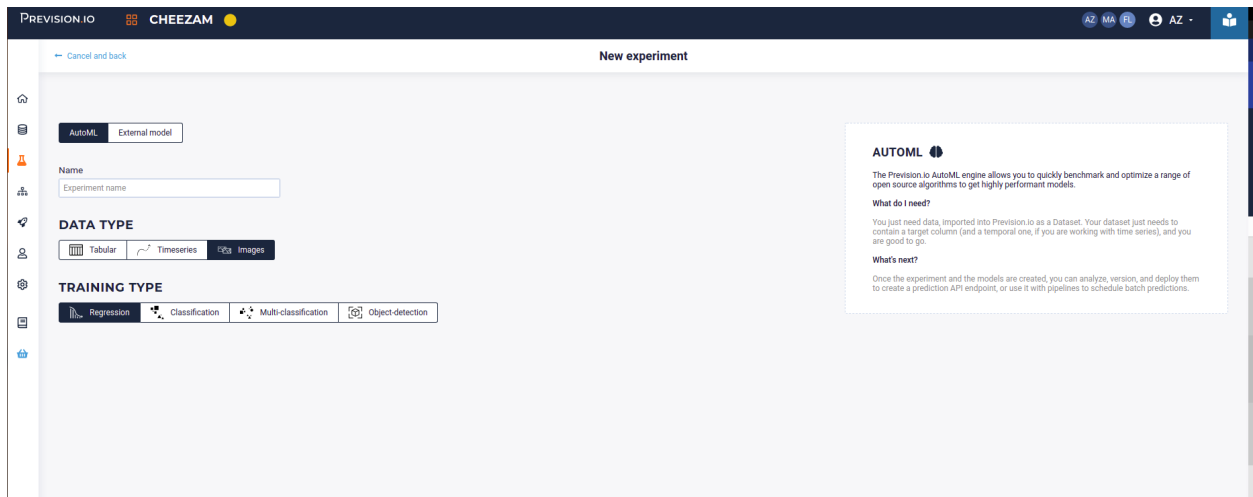


Fig. 54: When selecting an image usecase, you can train a classification, a multiclassification, a regression or an object detector

Note that your models can be trained on mixed dataset : you can have a dataset with tabular datas, columns of text, categorical datas and linear one.

For example, this dataset is valid :

Table 4: A dataset with mixed data

id	path_image	description	screen-shot_day	nbclick	cat	TARGET
1	games/action/screenshot_1.png	brutal top-down [...]	2020-03-12	4555	ACTION	offensive
2	games/action/screenshot_2.png	playstation 3 compete [...]	2020-05-12	12298	ACTION	neutral
3	games/simulation/screenshot_1.png	an [...]	2020-06-04	10024	SIMULATION	offensive

Setup

When training an image use case, you always need 2 assets :

- a csv dataset with path to the images and different info (and the target)
- a zip of images uploaded in *your image folder storage space*

The csv dataset is a classic dataset imported with connectors or uploaded via interface while the zip is a zip file of image set up on your local environment and uploaded to your image folder storage.

More over, for Object Detector training, you need to draw bounding box onto your image and export them to Pascal Voc format.

Regression

A regression on images is a model that predict a number from an image, for example number of click from a thumbnail or a screenshot.

Classification

A classification is a model that predict class of an image. There could be 2 class or more. For example “smiling”, “crying” or “thinking” from the image of a face.

The screenshot shows a web interface for configuring a classification model. It is divided into three main sections:

- Data**: Contains two dropdown menus. The first, labeled 'Dataset', has the value 'detailcheezam3000'. The second, labeled 'Image Folder', has the value 'cheezam3000'.
- Training options**: Contains a dropdown menu labeled 'Metric to use' with the value 'Categorical cross entropy loss'. Below this are three radio buttons for 'Performances': 'QUICK' (selected), 'NORMAL', and 'ADVANCED'.
- Fields configuration**: Contains five dropdown menus. The first is 'Target column'. The second is 'Image path'. The third is 'ID column (optional)'. The fourth is 'Weight (optional)'. The fifth is 'Fold (optional)'.

Fig. 55: A classification use case with Image path input

Object detection

An object-detection use case is a model that from an image return zero, one or more bounding box with label.

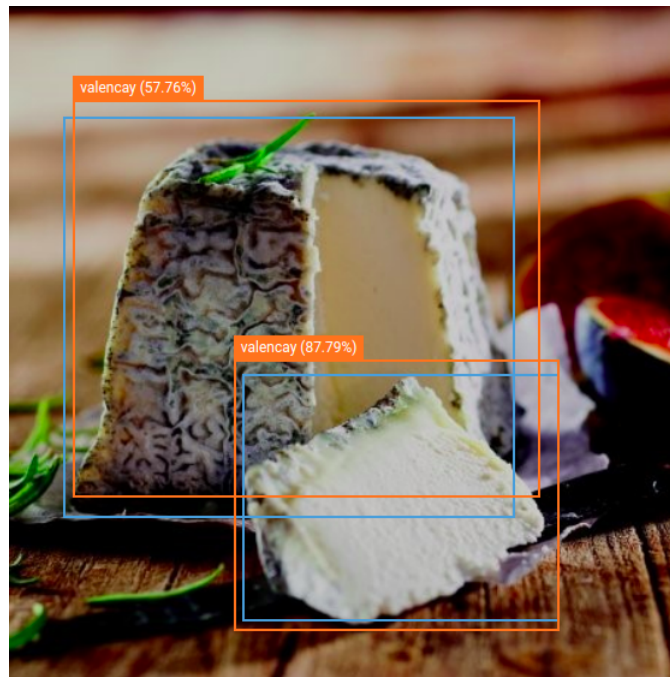


Fig. 56: An image with its training bounding box (*blue*), the predicted bounding box (*orange*) with predicted label and probability

Image Path

When there are image in a dataset, the path **to each image** from **the root of the image folder** should be put in a column and this column must then be selected as the “Image path” column.

For example if your folder of image has this architecture :

You should zip it from the root :

Upload the zip to your image folder Storage

and your dataset should have a column with the following info

Table 5: path info are from the root

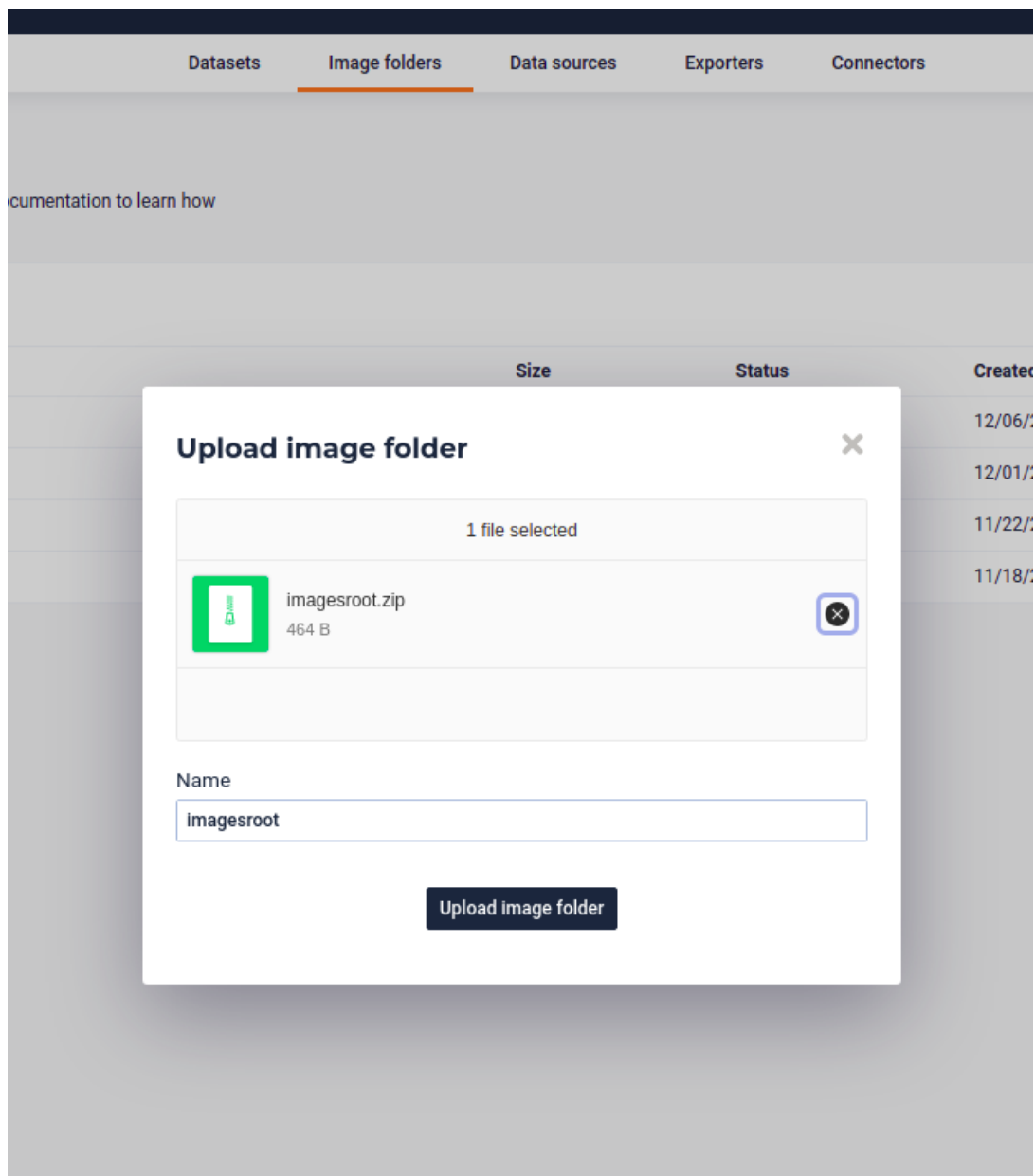
path	class	otherinformation
gruyere/image_20210806_110725_16780.jpg	gruyere	yellow
gruyere/image_20210806_105914_1559.jpg	gruyere	yellow
mimolette/image_20210806_105914_1553.jpg	mimolette	orange
mimolette/image_20210806_105914_1558.jpg	mimolette	orange
mimolette/image_20210806_105914_1551.jpg	mimolette	orange
neufchatel/image_20210806_110935_20864.jpg	neufchatel	white

When setting the experiment, this path from the zip root should be mapped to the ‘image path’ input.


```
tutoprompt >tree
.
├── gruyere
│   ├── image_20210806_110725_16780.jpg
│   ├── image_20210806_110725_16781.jpg
│   ├── image_20210806_110725_16782.jpg
│   ├── image_20210806_110725_16783.jpg
│   ├── image_20210806_110725_16784.jpg
│   ├── image_20210806_110725_16785.jpg
│   ├── image_20210806_110725_16786.jpg
│   ├── image_20210806_110725_16787.png
│   ├── image_20210806_110725_16788.jpg
│   ├── image_20210806_110725_16789.jpg
│   ├── image_20210806_110935_20860.jpg
│   ├── image_20210806_110935_20861.jpg
│   ├── image_20210806_110935_20862.jpg
│   ├── image_20210806_110935_20863.jpg
│   ├── image_20210806_110935_20864.jpg
│   ├── image_20210806_110935_20865.png
│   ├── image_20210806_110935_20866.jpg
│   ├── image_20210806_110935_20867.jpg
│   ├── image_20210806_110935_20868.jpg
│   └── image_20210806_110935_20869.jpg
├── nimolette
│   ├── image_20210806_105914_1550.jpg
│   ├── image_20210806_105914_1551.jpg
│   ├── image_20210806_105914_1552.jpg
│   ├── image_20210806_105914_1553.jpg
│   ├── image_20210806_105914_1554.jpg
│   ├── image_20210806_105914_1555.jpg
│   ├── image_20210806_105914_1556.jpg
│   ├── image_20210806_105914_1557.jpg
│   ├── image_20210806_105914_1558.jpg
│   └── image_20210806_105914_1559.jpg
└── neufchatel
    ├── image_20210806_110935_20860.jpg
    ├── image_20210806_110935_20861.jpg
    ├── image_20210806_110935_20862.jpg
    ├── image_20210806_110935_20863.jpg
    ├── image_20210806_110935_20864.jpg
    ├── image_20210806_110935_20865.png
    ├── image_20210806_110935_20866.jpg
    ├── image_20210806_110935_20867.jpg
    ├── image_20210806_110935_20868.jpg
    └── image_20210806_110935_20869.jpg

3 directories, 40 files
tutoprompt >
```

```
3 directories, 40 files
tutoprompt >zip imagesroot.zip *
  adding: gruyere/ (stored 0%)
  adding: mimolette/ (stored 0%)
  adding: neufchatel/ (stored 0%)
tutoprompt >ls
gruyere imagesroot.zip mimolette neufchatel
tutoprompt >
```



Label or class

For classification, Multiclassification and Object Detection, you must set a column with the name of a class, or label

Table 6: A label column

path	label	otherinformation
gruyere/image_20210806_110725_16780.jpg	gruyere	yellow
gruyere/image_20210806_105914_1559.jpg	gruyere	yellow
mimolette/image_20210806_105914_1553.jpg	mimolette	orange
mimolette/image_20210806_105914_1558.jpg	mimolette	orange
mimolette/image_20210806_105914_1551.jpg	mimolette	orange
neufchatel/image_20210806_110935_20864.jpg	neufchatel	white

You can call it anything you want but when setting your experiment, you need to select it in the ‘class’ input selector

Bounding box

If you want to train a model to detect object on a image, you must run an Object Detection training and you **must provide** a dataset with bounding box for training

New version

Data

Dataset ?

detailcheezam3000

Image Folder

cheezam3000

Training options

Performances

☒ QUICK ?
☐ NORMAL
 ☐ ADVANCED ?

Fields configuration

Object class column

Image path

Top

Bottom

Left

Right

This bounding box is described in a dataset in csv format that can be *imported or uploaded to your data section*. Bounding box always use 4 columns to be described but there are two format for bounding box :

- Pascal Voc bounding box : x-top left, y-top left,x-bottom right, y-bottom right
- Yolov bounding box : x-top left, y-top left, width, height (*of the bounding box*)

Prevision Platform use the Pascal Voc Bounding box format in csv. So to describe a bounding box , you must provide 4 more columns and map them in the input interface when setting your object detection experiment :

Table 7: A complete object detector dataset

path	category	xleft	ytop	xright	ybottom	imgname	labels
image_20210806_105845_425.jpg	patemol-jeune routelavee	228	634	846	984	image_20210806_105845_425.jpg	vieuxboulogne
image_20210806_105845_428.jpg	patemol-jeune routelavee	83	99	302	329	image_20210806_105845_428.jpg	vieuxboulogne
image_20210806_105845_434.jpg	patemol-jeune routelavee	0	4	188	145	image_20210806_105845_434.jpg	vieuxboulogne
image_20210806_105846_446.jpg	patemol-jeune routelavee	36	8	244	215	image_20210806_105846_446.jpg	vieuxboulogne

Note that for object detection, there can be several object in one image. So your dataset may have many row with same image path but different bounding box and label :

Table 8: Several bounding box on one image

path	xleft	ytop	xright	ybottom	labels
image_20210806_105845_425.jpg	228	634	846	984	vieuxboulogne
image_20210806_105845_425.jpg	455	877	565	984	mimolette
image_20210806_105845_425.jpg	28	64	754	800	mimolette
image_20210806_105845_425.jpg	22	34	600	900	cabecou

Object Segmentation

Prevision.io does not provide Object Segmentation models.

Launch train

Once you had mapped each of the input to the corresponding columns, and given a name to your experiment, you can click on the train button and the training will start.

For Regression, Classification and multiclassification, you will find the standard metrics and chart. For Object Detection, the metrics is Mape and represent the accuracy of the bounding box.

External Model

When running some *Experiments* you can either use the Automl engine over the data connected or upload your own pretrained models.

Both will benefit from same Prevision.io features :

- evaluation,
- batch prediction,
- deployment,

- pipeline integration,
- monitoring

External model import uses the [Standardized ONNX Format](#) and most of the standard ML library have module for export :

Table 9: Onnx exporter for standard lib

library	exporter	link
sklearn	sklearn-onnx	http://onnx.ai/sklearn-onnx/
Tensor-flow	tensorflow-onnx	https://github.com/onnx/tensorflow-onnx/
Pytorch	torch-onnx	https://pytorch.org/docs/stable/onnx.html
XGBoost	sklearn-onnx	http://www.xavierdupre.fr/app/sklearn-onnx/helpsphinx/auto_tutorial/plot_gexternal_xgboost.html
XGBoost	onnxmltools	https://github.com/onnx/onnxmltools
Light-GBM	skl2onnx	http://www.xavierdupre.fr/app/sklearn-onnx/helpsphinx/auto_tutorial/plot_gexternal_lightgbm.html
Light-GBM	onnxmltools	https://github.com/onnx/onnxmltools
CatBoost	onnxmltools	https://github.com/onnx/onnxmltools

If you prefer reading code than document to kickstart your project, you may use this [provided boilerplate](#) . This code builds a basic classifier and creates the needed files to use with Prevision Platform :

- a Trainset for trying Automl
- an holdout file to evaluate each iteration of your experiments
- an onnx file
- a yaml configuration file (see below)

```
git clone https://github.com/previsionio/prevision-onnx-templates.git
cd prevision-onnx-templates
python3.8 -m venv env
source env/bin/activate
pip install -r requirements.txt
python sktoonnx.py
```

Constraints

Current version of Prevision Platform only supports *Structured Data* and only for classification, regression and multi-classification (no *Natural Language Processing* nor *Computer Vision Model*)

Prerequisites

For importing a model as an experiments you should at least provide :

- an holdout file to evaluate the model. This holdout must have the same features and target column than the model was trained on.
- the onnx file of the model (it must be a classification, a regression or a multiclassification)
- a config file in the yaml format

You could provide a trainset too. The trainset is going to be used for computing drift (and thus, expected distribution of features and target in production) if you deploy your model. If you provide both a trainset and holdout :

- score is computed from the holdout
- drift is computed from trainset

Note: You could, and probaly should, import many onnx models in one experiment in order to compare them side by side.

How to get the Onnx File ?

To get you onnx file, you need first to build a model, wth standard Machine Learning Frameworks like sklearn or XGBoost, and then export them with exporter modules.

You can get any onnx file from anybody or any tools as long as :

- a config file describing the inputs is provided
- the output of the model is an array of string (binary classification or multi classification) , an array of numbers (regression) of both (a multi classification)

For example :

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType

clf = RandomForestClassifier(max_depth=50, verbose=1, n_estimators=200, max_features=1)
clf.fit(X_train, y_train)

initial_type = [('float_input', FloatTensorType([None, np.array(X_train).shape[1]]))]
onx = convert_sklearn(clf, initial_types=initial_type)

with open(join(OUTPUT_PATH, "classif_fraud.onnx"), "wb") as f:
    f.write(onx.SerializeToString())
```

How to make the yaml config File ?

The config file is a standard yaml file.

You **must** always provide the list of the names of the inputs and, if the model is a classifier, you must provide the name of the class too.

Beware that the name of the class will be cast as a string so if your class are 1.0 and 0.0 (because some int has been converted), the name of the class must be "1.0" and "0.0"

You **could** provide your model hyper parameter, like the algorithm used, by providing :

- an *algorithm* field with a string as value (that can be anythin).
- a list of key/value i the *hyperparams* field :

```
---
class_names:
- A
- B
input:
- DeviceType_desktop
- DeviceType_mobile
- "Code Produit_C"
- "Code Produit_H"
- "Code Produit_R"
- "Code Produit_S"
# optional metadata
algorithm:
- random-forest
hyperparams:
- n_trees: 30
- seed: 42
```

Any name can be used and. **This metadata will only be used for display in the model page.** The model are not retrained and yaml purpose is only to provide informations about your models. You could refer to the [Prevision.io Github](#) to see how to generate an yaml config file from your model and training data.

Importing an External Model

Once you got all the required assets (holdout file, onnx file and yaml file), you can launch a new experiment in your project.

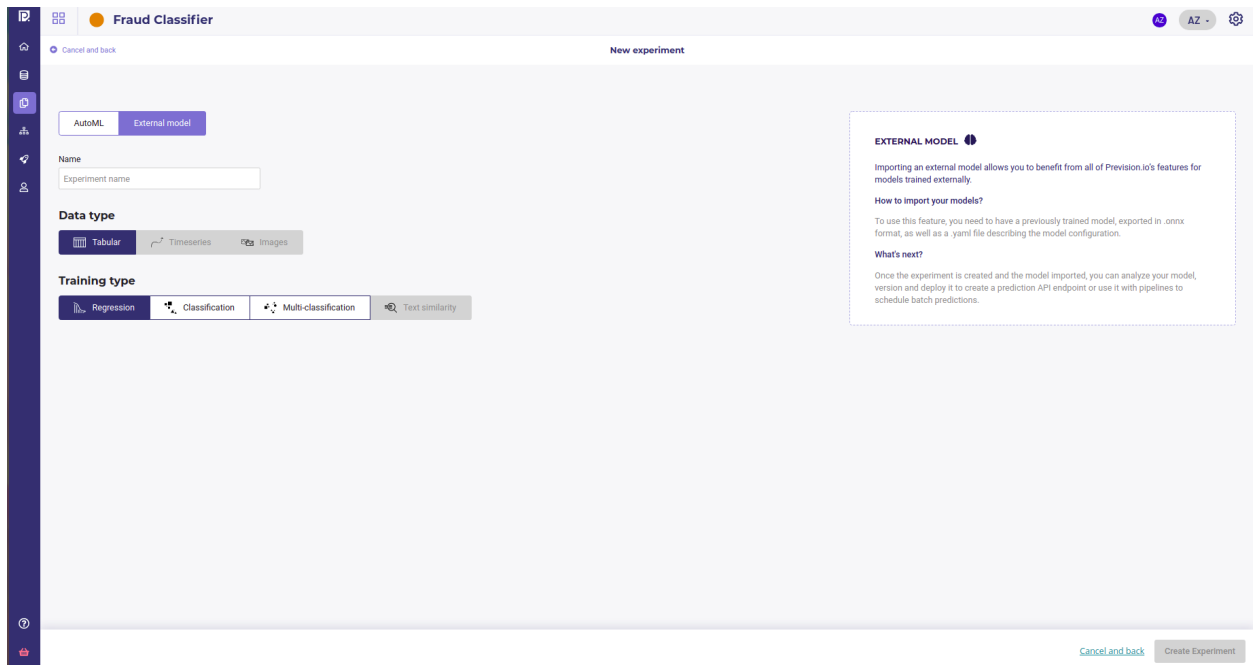
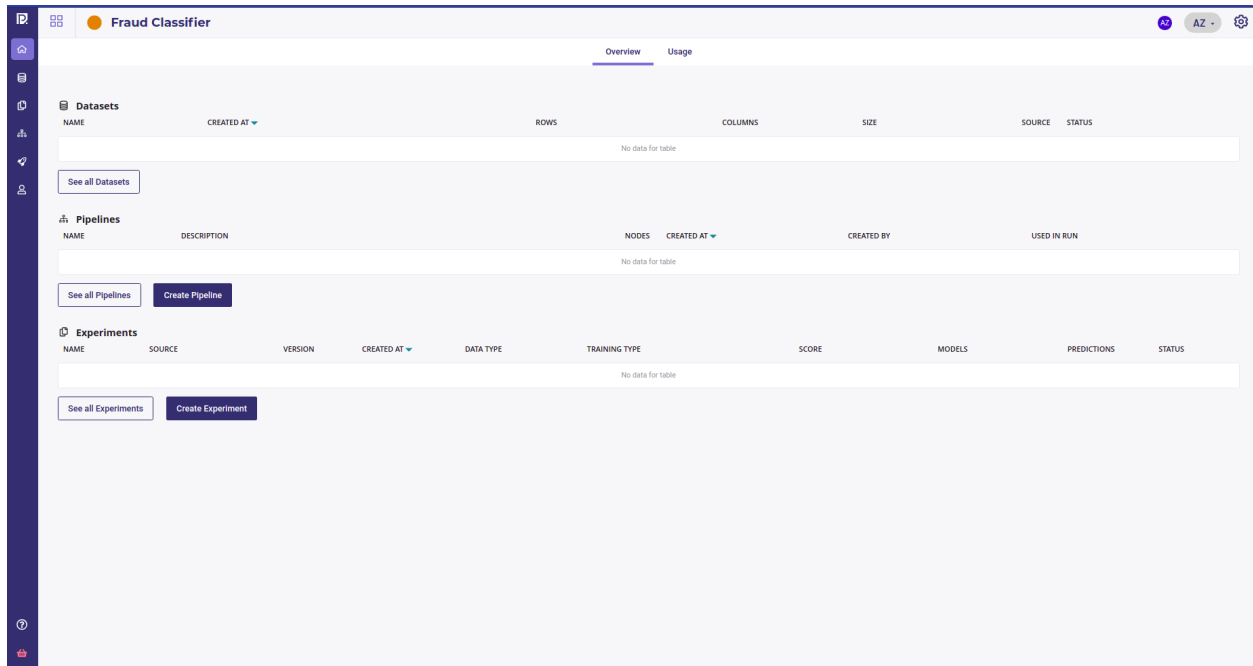
Go to your project homepage and create a new experiment with the "Create experiment" button :

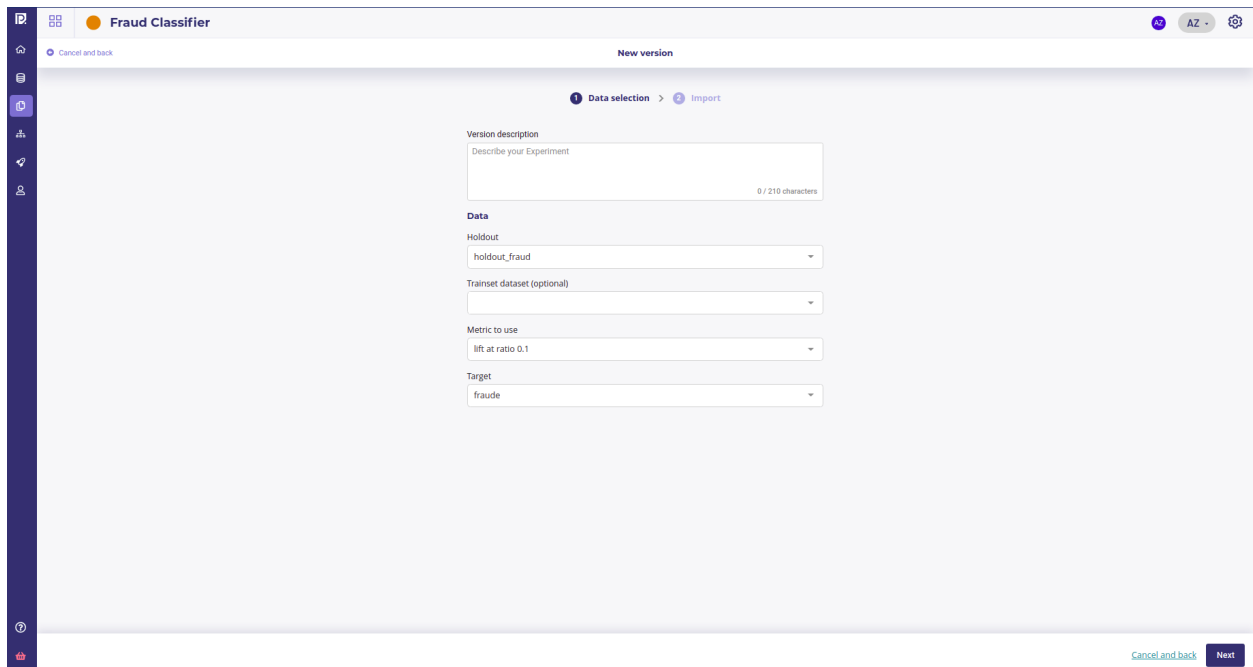
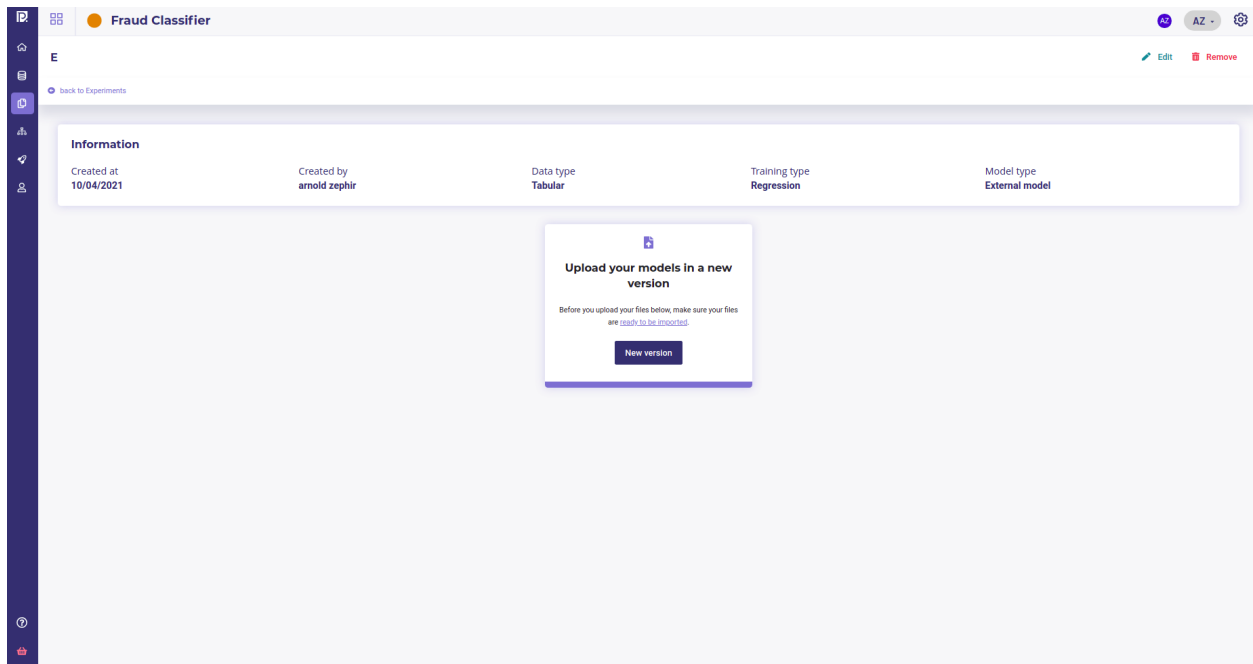
In order to import your model, select "External model". The Timeseries, Images and Text similarity options will be muted as they are not supported yet. Assign a name to your experiment and click on "Create experiment"

When you create an experiment, you need to create a first version (see [Versionning your experiments](#))

Describe your version and select :

- the holdout file to evaluate
- the metrics to use for evaluation
- the target (only when you create the first version. Target will always be the same in all version of your experiment)





On the next screen you will be asked to provide an onnx file and a yaml file **for each model your upload**. You can upload as much model as you want as long as they have the same target.

The screenshot shows the 'Import' step in the Prevision.io interface for a 'Fraud Classifier' model. The interface is titled 'Import trained models and their configuration files'. It features two slots for importing models. The first slot is filled with a model named 'sklearn_rf_classifier'. It has a text input for the model name, a 'Drop file here, paste or browse' button for the 'Import configuration file (.YAML)', and another 'Drop file here, paste or browse' button for the 'Import train model (.ONNX)'. There is an 'Upload' button to the right of the ONNX input. The second slot is empty and shows the same layout. At the bottom right, there are three buttons: 'Cancel and back', 'Previous', and 'Create'.

When all resources are uploaded, you can click on the create button on the bottom right of the screen. All the uploaded models will be evaluating on the holdout dataset upon the metrics you selected before.

In a few minutes, the experiments should be available on the experiment page.

Now you can `/studio/experiments/evaluating` and proceed to [Deployments](#)

Data type

Data type is, obviously, the type of your data :

Hint: Most of nlp problems should be considered as tabular data whom one or more columns are text. If any columns is detected as a textual one, Prevision AutoML engine will apply a set of standard NLP embedding technics (tf/idf, Transformers, seq to seq...). The only limitation is that you cannot yet build generative model (so no automatic summarisation) but if you want to classifiante or rate docs or email, tabular data is the way to go.

As far as that goes, image are used in a tabular way too, except for the object detector. When choosing data type image, you will used a dataset whom on feature is a path to some image uploaded in your image folder. You can run Classification or Regression on Image !

- tabular : data from csv, sql database, hive database, ... suitable for classification and regression
- timeseries : when target depends on time, use a timeserie. Note that you should have data with constant timestep as much as possible and only regression are possible with timeseries
- images : if you want to build an image-based model. Note that you can mix images and standard features in the same experiment. Image has a special probleme type (Object Detection)

	NAME	SOURCE	VERSION	CREATED AT ▼	CREATED BY	DATA TYPE	TRAINING TYPE	SCORE	MODELS	PREDICTIONS	STATUS
<input type="checkbox"/>	sklearn_v1	External model	1	10/04/2021, 12:05	arnold zephr	Tabular	Classification	4.167 (lfl_at_0.1) ★★☆☆	1	100	●

rows per page: 25 < prev 1 - 1 of 1 next >

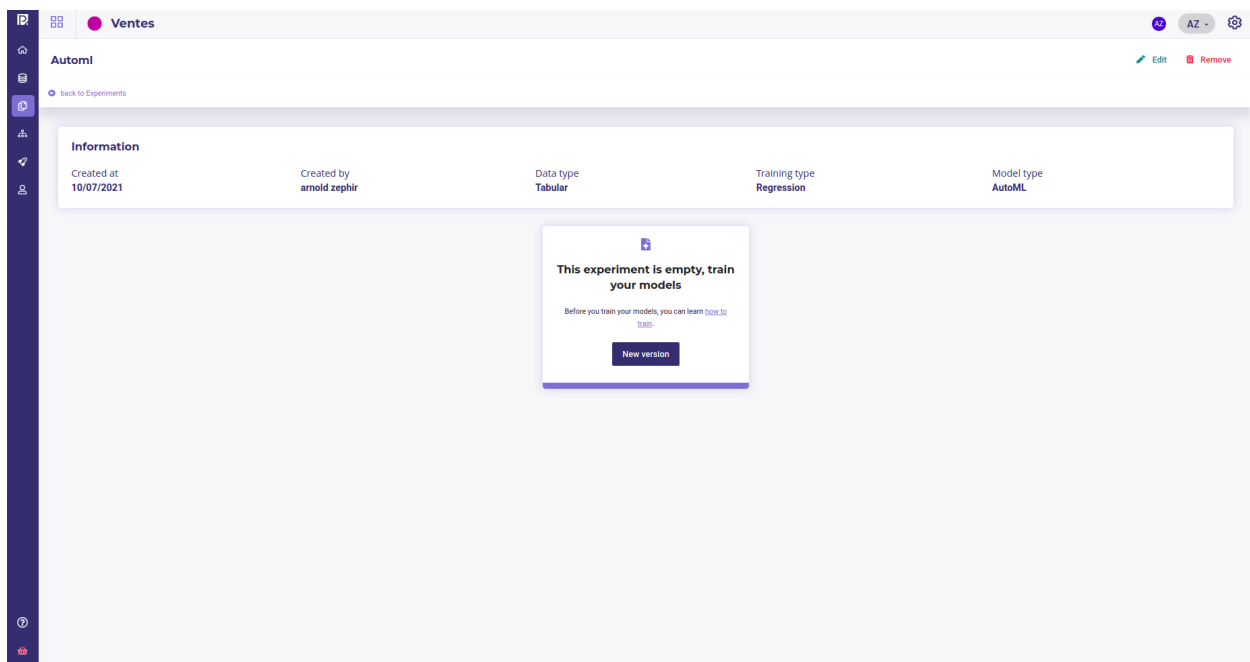
Training type

Training type is the kind of problem you want to solve :

- Regression : when you need to predict a continuous value. Suitable for sales forecasting, price estimation, workforce management, ... can be used for image and text.
- classification : when target has only 2 modalities, choose classification. For example fraud detection, churn prediction, Risk management,...
- multi-classification : if your target has more than one modality. Standard example are product cross sale, Transport Mode detection, Evaluation prediction, email classification, sentiment analysis ...
- Text similarity : this training type is dedicated to retrieve doc from query. The input is tabular data with at least one column of docs (text) and the model will be trained to attribute later query to one of this original doc. It's useful for searching item from their description or build chatbot to answer to user questions
- Object-detection : Object detection train a model to detect some object on image, attribute a class and return a bounding box. For example you can [detect pools on satellite image](#) or [type of french cheese on a photo](#)

And get a more details about each Problem type :

Whatever your choice, when you create a new experiment, you will be prompt to create the first *version* of it.



When clicking on **new version**, you will enter the configuration screen, that depends on the engine you choose

For the first version only, you need to fill the target field to set the common target for all versions of your experiment.

Once every mandatory parameters are fill (see each training type doc for an explanation of parameters), you can click on “train” to launch a train and start modelisation.

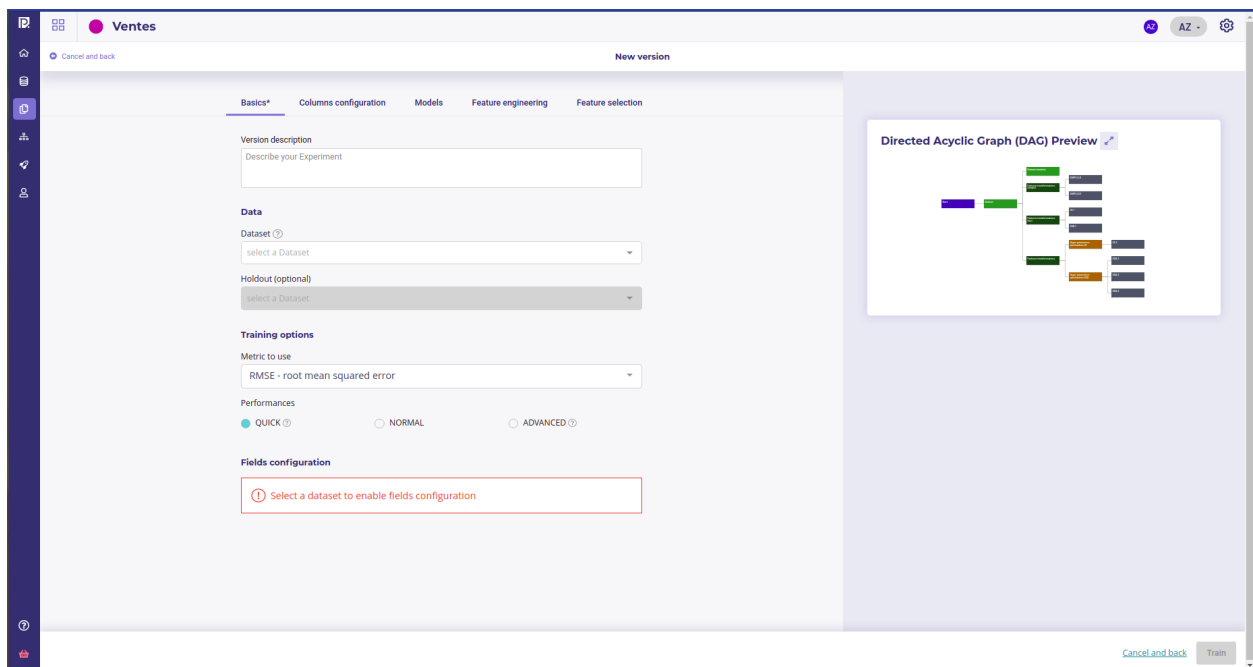


Fig. 57: automl configuration screen, with the graph of tasks that is gonna be executed

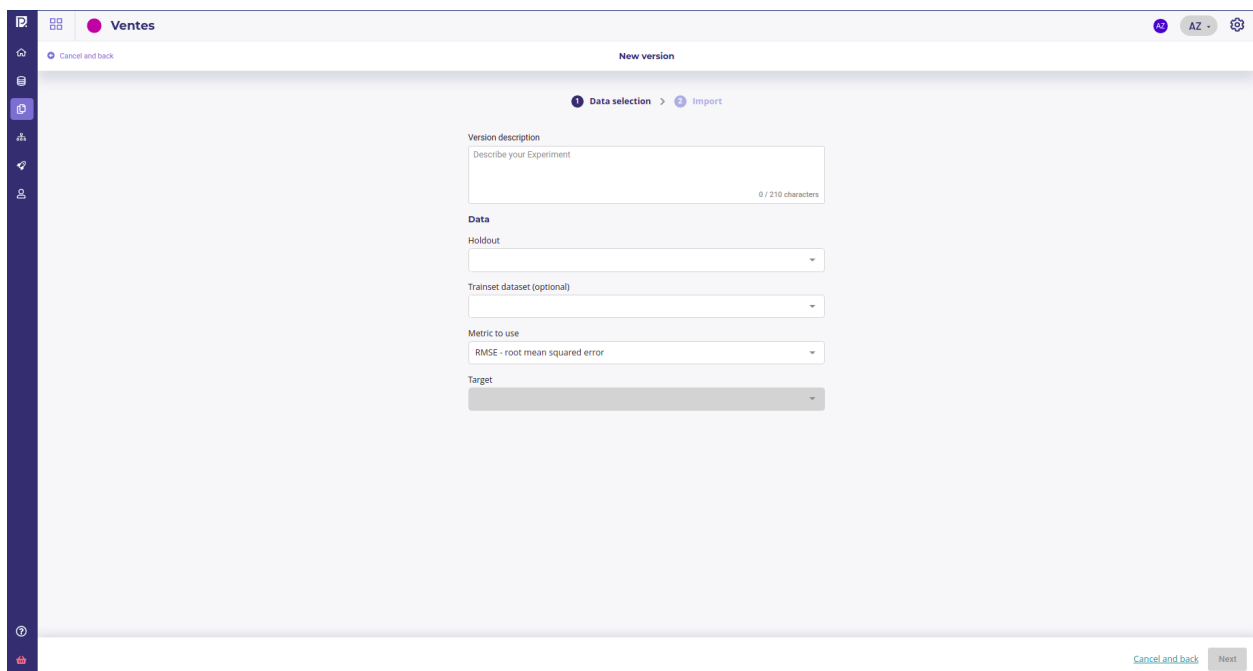


Fig. 58: The *external model* configuration screen

Import External models into your experiment

If you already have models built from other frameworks, so called *external models* you can import them to benefit from Prevision.io evaluation and monitoring tools.

See the [dedicated page](#)

Inspect your experiment and evaluate your models

Once an experiment has at least one version, you can get some details about it on its corresponding dashboard by clicking on its name in the list of experiments.

Models

By clicking on the models menu of top experiment navigation, you will access the model list trained for this experiment version. you will also, at the bottom of the page, find information regarding the model selected for this train.

NAME	TECHNOLOGY	TYPE	SCORE	TRAINING DURATIONS	PREDICT DURATIONS	ARRIVAL TIME
CB-1 <i>Best performance</i>	CATBoost	Base	7,911 ± 2,516	12m 34.6s	48ms	10/20/2021, 14:22:02
LGB-1	LightGBM	Base	8,008 ± 2,454	1m 10.4s	93ms	10/20/2021, 14:12:08
XGB-1	XGBoost	Base	8,204 ± 2,473	4m 10.5s	81ms	10/20/2021, 14:18:04
XGB-2	XGBoost	Base	8,424 ± 1,040	8m 0.2s	311ms	10/20/2021, 14:26:23
LGB-2	LightGBM	Base	8,456 ± 1,069	3m 34.5s	255ms	10/20/2021, 14:18:32
RF-1	Random Forest	Base	8,544 ± 2,603	4m 37.7s	224ms	10/20/2021, 14:15:45
XGB-3	XGBoost	Base	8,577 ± 1,361	9m 50s	292ms	10/20/2021, 14:42:53
CB-4	CATBoost	Base	8,588 ± 1,468	23m 11.9s	262ms	10/20/2021, 16:09:14
XGB-4	XGBoost	Base	8,729 ± 628	9m 28s	237ms	10/20/2021, 14:39:21
LGB-4	LightGBM	Base	8,785 ± 1,116	4m 8.2s	273ms	10/20/2021, 14:24:45
CB-3	CATBoost	Base	8,902 ± 1,476	24m 43s	481ms	10/20/2021, 16:05:36
LGB-3	LightGBM	Base	8,951 ± 1,031	6m 54.9s	248ms	10/20/2021, 14:26:03

Fig. 59: Model List

By clicking on the model name in the list, you will be redirected to the model detail page. Please note that a toggle button is available on the right side of the list for each model. This toggle allows you to tag a model as deployable. In order to know how to deploy a model, please go to the dedicated section.

Each model page is specific to the datatype/training type you choose for the experiment training. Screens and functionality for each training type will be explained in the following sections. You can access a model page by two ways :

- by clicking on a graph entry from the general experiment page
- by clicking on a list entry from the models top navigation bar entry

Then you will land on the selected model page splitted in different parts regarding the training type.

For each kind of tabular training type, the model general information will be displayed on the top of the screen. Three sections will be available.

Models > XGB-3		Download CV	
MODEL INFORMATION		HYPERPARAMETERS Download	
technology	XGB	colsample_bytree	0.8451704283003703
score	8,577	eta	0.05
metric	rmse	eval_metric	rmse
metric standard deviation	1,361	max_depth	7
train duration	9m 50s	min_child_weight	15
predict response time	292ms	objective	reg:linear
deployable	Yes	reg_lambda	0.5236229827587133
model used for blend	No	silent	1
arrival time	10/20/2021, 14:42:53	subsample	0.843057816759945
		feature_selected	["freq", "lin"]
		seed	33479
		num_boost_round	1079
		SELECTED FEATURE ENGINEERINGS	
		✓ Frequency encoding	
		✓ Linear feature scaling ?	

Fig. 60: Model detail

- Model information : information about the trained model such as the selected metric and the model score
- Hyperparameters : downloadable list of hyperparameters applied on this model during the training
- Selected feature engineerings (for regression, classification & multi-classification) : features engineerings applied during the training
- Preprocessing (for text similarity experiments) : list of pre-processing applied on textual features

Please note that for following experiments types, the general information parts is different than from others :

- Image detection experiments : no feature engineering
- text similarity experiments : preprocessing are displayed instead of feature engineering

Cross Validation File

Each time a Model is built, a Cross Validation is performed on trainset. All score and charts displayed on this page, except those explicitly built on holdout files, are computed by performing a k-fold cross Validation on your trainset with k=5 as default values.

If you selected a fold column when *configuring your experiment*, the fold are built using those you provided. You can read *the guide about having a good cross validation strategy*.

You can download the cross validation file for any model by clicking on the upper right button ****Download CV***. Inside it you will find prediction for each fold when training is done on the other folds.

For example, the line :

```
customerid,target,pred_target_1,__fold__
15712807,1,0.23583529833568478,2
```

Means that the sample whom customerid is 15712807 has received the fold number "2" and when the model was trained on a dataset with only folds 1,3,4 and 5, it predicted 0.23583 for this sample.

By downloading the Cross Validation file, you can perform further investigation on performance of your model for specific groups or range of values.

Note that the detail of metrics for each fold is not displayed on the page. Each metrics is infact the average of each fold

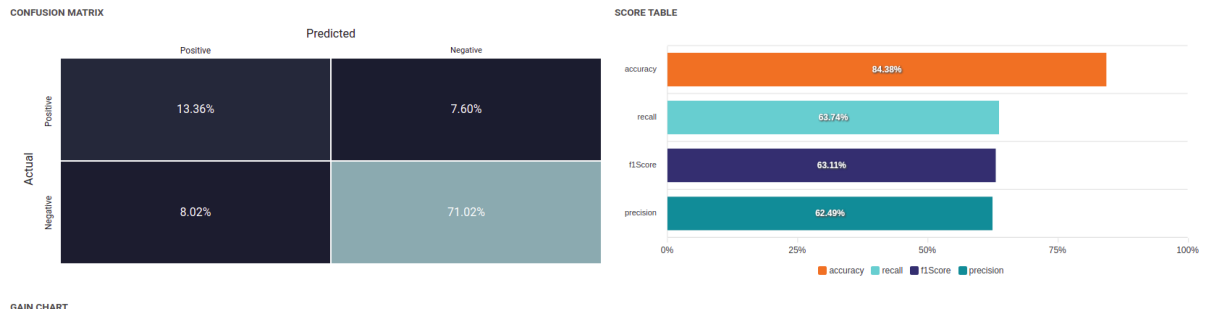


Fig. 61: All this metrics are average of metrics for each fold

Model page - graphs explanation

In order to better understand the selected model, several graphical analyses are displayed on a model page. Depending on the nature of the experiment, the displayed graphs change. Here an overview of displayed analysis depending on the experiment type.

Table 10: Type of Training. Tabular Data

chart	re-gres-sion	clas-sifi-ca-tion	multi-classification	text-simi-larity	Time se-ries	Image regres-sion	Image classifi-cation	Image multi-classification	Image detec-tion
Scatter plot graph	Yes	No	No	No	Yes	Yes	No	No	No
Residual errors distribution	Yes	No	No	No	Yes	Yes	No	No	No
Score table (textual)	Yes	No	No	No	Yes	Yes	No	No	No
Residual errors distribution	No	No	No	No	No	No	No	No	No
Score table (overall)	No	No	Yes	No	No	No	No	Yes	No
Cost matrix	No	Yes	No	No	No	No	Yes	No	No
Density chart	No	Yes	No	No	No	No	Yes	No	No
Confusion matrix	No	Yes	Yes	No	No	No	Yes	Yes	No
Score table (by class)	No	Yes	Yes	No	No	No	Yes	Yes	No
Gain chart	No	Yes	No	No	No	No	Yes	No	No
Decision chart	No	Yes	No	No	No	No	Yes	No	No
lift per bin	No	Yes	No	No	No	No	Yes	No	No
Cumulated lift	No	Yes	No	No	No	No	Yes	No	No
ROC curve	No	Yes	Yes	No	No	No	Yes	Yes	No
Accuracy VS K results	No	No	No	Yes	No	No	No	No	No

Please note that you can download every graph displayed in the interface by clicking on the top right button of each graph and selecting the format you want.

Scatter plot graph

This graph illustrates the actual values versus the values predicted by the model. A powerful model gathers the point cloud around the orange line.

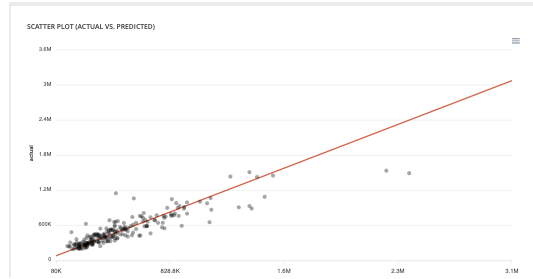
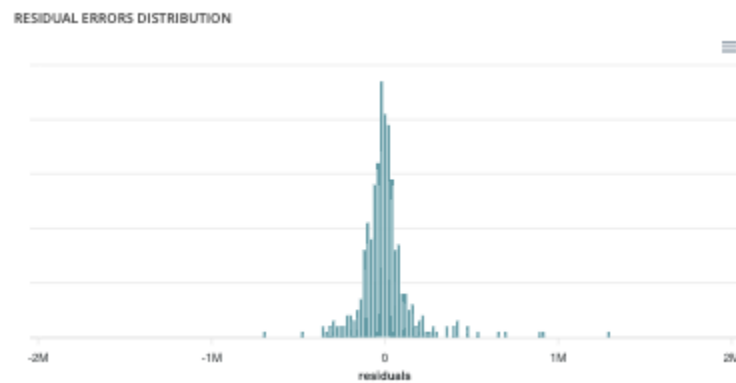


Fig. 62: Scatter plot

Residual errors distribution

This graph illustrates the dispersion of errors, i.e. residuals. A successful model displays centered and symmetric residues around 0.



Score table (textual)

Among the displayed metrics, we have:

- The mean square error (MSE)
- The root of the mean square error (RMSE)
- The mean absolute error (MAE)
- The coefficient of determination (R2)
- The mean absolute percentage error (MAPE)

SCORE TABLE	
Mean squared error	27,223,899,929
Root mean squared error	164,997
Mean absolute error	94,770
R2	99.95%
Mean absolute percentage error	18.55%

Slider

For a binary classification, some graphs and scores may vary according to a probability threshold in relation to which the upper values are considered positive and the lower values negative. This is the case for:

- The scores
- The confusion matrix
- The cost matrix

Thus, you can define the optimal threshold according to your preferences. By default, the threshold corresponds to the one that minimizes the F1-Score. Should you change the position of the threshold, you can click on the « back to optimal » link to position the cursor back to the probability that maximizes the F1-Score.



Cost matrix

Provided that you can quantify the gains or losses associated with true positives, false positives, false negatives, and true negatives, the cost matrix works as an estimator of the average gain for a prediction made by your classifier. In the case explained below, each prediction yields an average of €2.83.

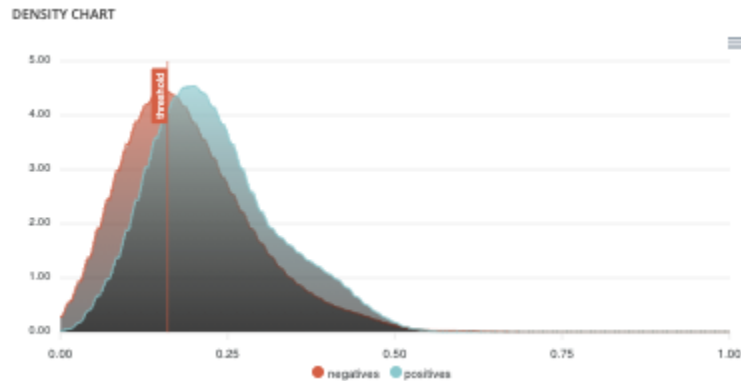
COST MATRIX

value = true/1	predict = true/1	gain = 10	expected = 0.248
	predict = false/0	gain = -5	expected = -0.804
value = false/1	predict = true/1	gain = -5	expected = -0.088
	predict = false/0	gain = 5	expected = 3.984

The matrix is initiated with default values that can be freely modified.

Density chart

The density graph allows you to understand the density of positives and negatives among the predictions. The more efficient your classifier is, the more the 2 density curves are disjointed and centered around 0 and 1.



Confusion matrix

The confusion matrix helps to understand the distribution of true positives, false positives, true negatives and false negatives according to the probability threshold. The boxes in the matrix are darker for large quantities and lighter for small quantities.

CONFUSION MATRIX

		Predicted	
		Positive	Negative
Actual	Positive	31.20%	7.18%
	Negative	17.51%	44.11%

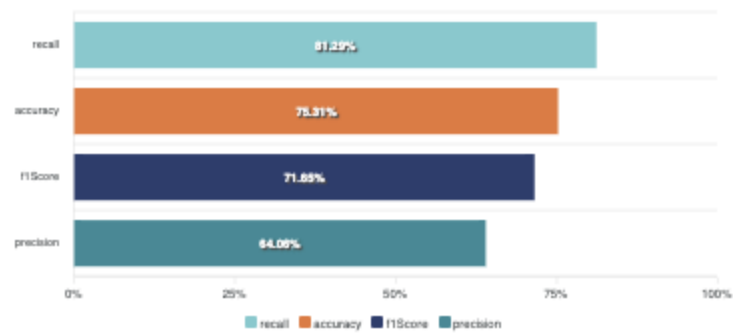
Ideally, most classified individuals should be located on the diagonal of your matrix.

Score table (graphical)

Among the displayed metrics, we have:

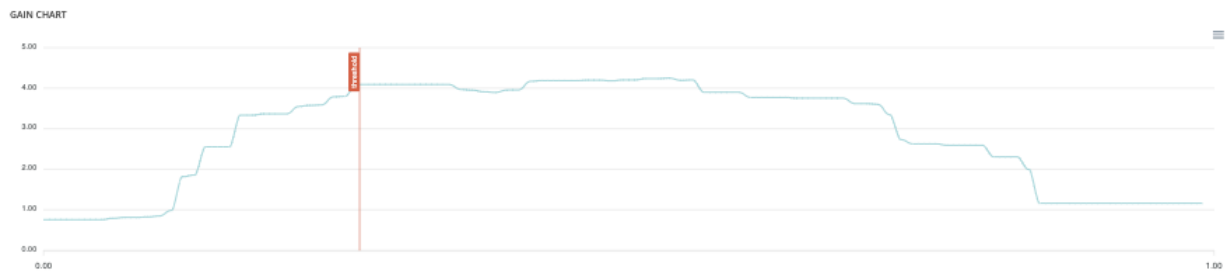
- Accuracy: The sum of true positives and true negatives divided by the number of individuals
- F1-Score: Harmonic mean of the precision and the recall
- Precision: True positives divided by the sum of positives
- Recall: True positives divided by the sum of true positives and false negatives

SCORE TABLE



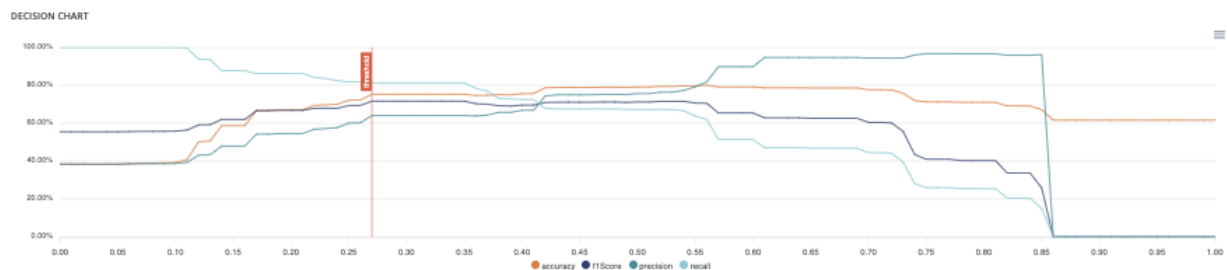
Gain chart

The gain graph allows you to quickly visualize the optimal threshold to select in order to maximise the gain as defined in the cost matrix.



Decision chart

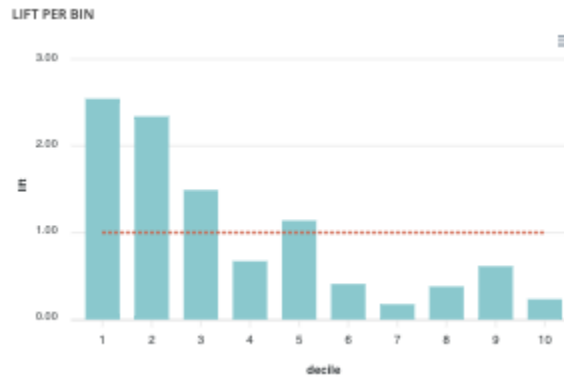
The decision graph allows you to quickly visualize all the proposed metrics, regardless of the probability threshold. Thus, one can visualize at what point the maximum of each metric is reached, making it possible for one to choose its selection threshold.



It should be noted that the discontinuous line curve illustrates the expected gain by prediction. It is therefore totally linked to the cost matrix and will be updated if you change the gain of one of the 4 possible cases in the matrix.

Lift per bin

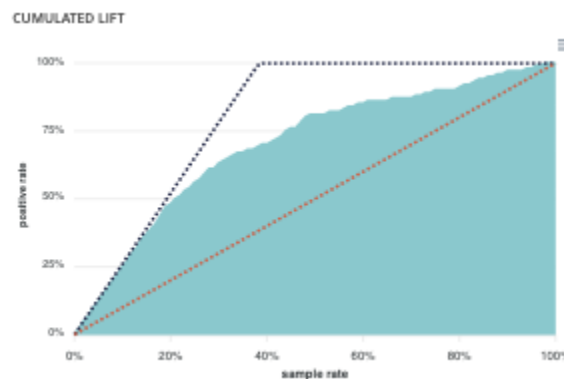
The predictions are sorted in descending order and the lift of each decile (bin) is indicated in the graph. Example: A lift of 4 means that there are 4 times more positives in the considered decile than on average in the population.



The orange horizontal line shows a lift at 1.

Cumulated lift

The objective of this curve is to measure what proportion of the positives can be achieved by targeting only a subsample of the population. It therefore illustrates the proportion of positives according to the proportion of the selected sub-population.

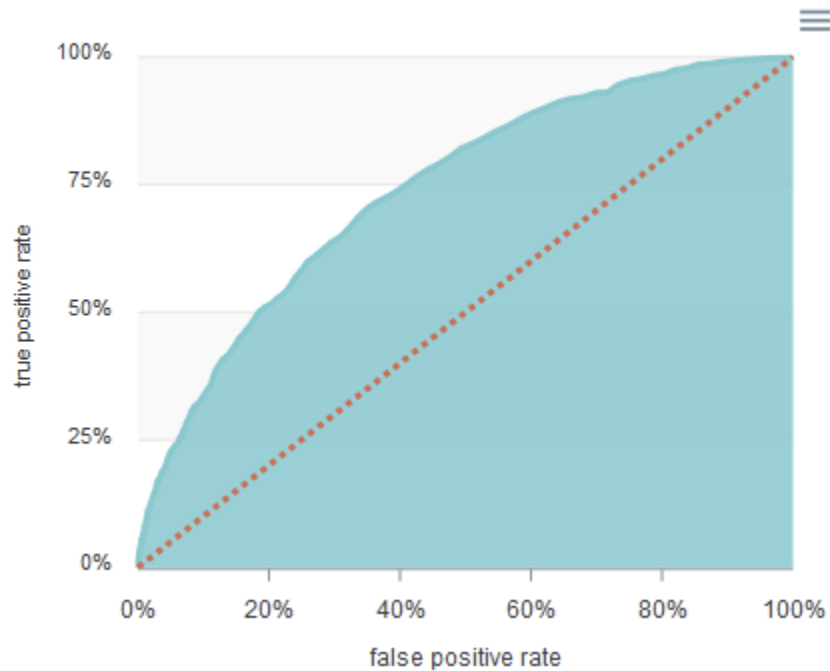


A diagonal line (orange) illustrates a random pattern (= x % of the positives are obtained by randomly drawing x % of the population). A segmented line (blue) illustrates a perfect model (= 100% of positives are obtained by targeting only the population's positive rate).

ROC curve

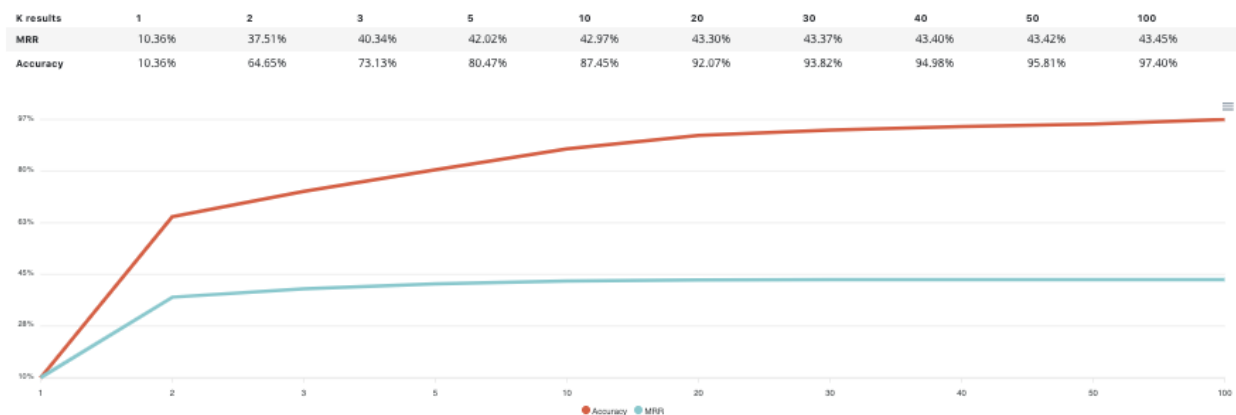
The ROC curve illustrates the overall performance of the classifier (more info: https://en.wikipedia.org/wiki/Receiver_operating_characteristic). The more the curve appears linear, the closer the quality of the classifier is to a random process. The more the curve tends towards the upper left side, the closer the quality of your classifier is to perfection.

ROC CURVE (AUC = 0.7358)



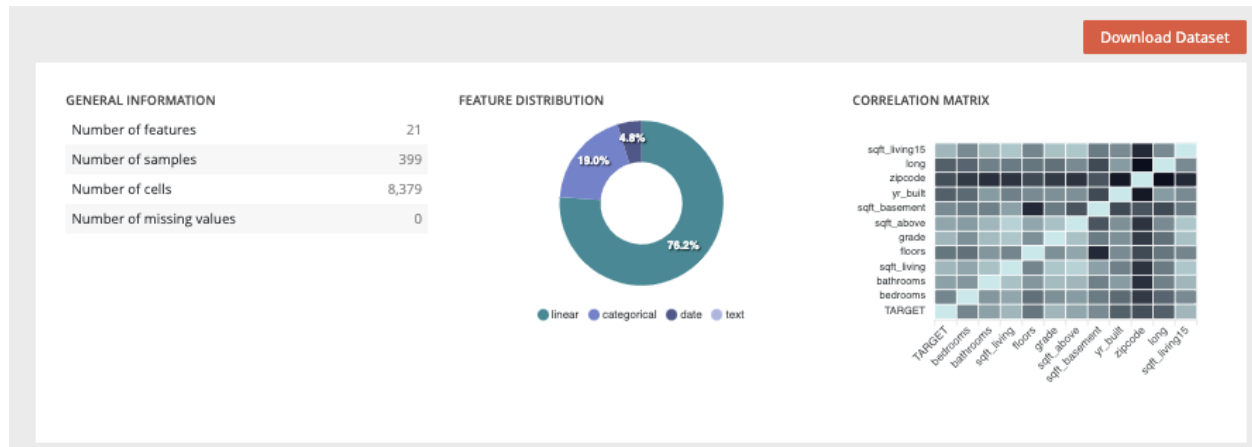
Accuracy VS K results

this graph shows the evolution of accuracy and MRR for several value of K results



Features

In this section you will find any information relative to the dataset used during the train.



On the top panel, you will find generic information about the dataset used during the train such as the number of columns, number of samples and number of cells or the usecases using this dataset.

You can also download the dataset used for the training by clicking on the “download dataset” button on top of the page.

Two graph are also displayed showing :

- the feature distribution regarding the feature type (linear, categorical, date or text). This distribution is automatically calculated when uploading a dataset into the platform
- correlation matrix showing the correlation coefficients between variables. Each cell in the table shows the correlation between two variables

Under this top panel, two tabs are available :

- Features analysis : table displaying features information calculated after the upload of the dataset such as the % of missing value.
- Dropped features : In this tab, you will find a list of all features you dropped for the usecase training during the usecase configuration

By clicking on a feature name you will be redirected to feature detail page

Predictions

The predictions menu allows you to do bulk predictions using a previously loaded [dataset](#) and see holdout predictions made during training.

In order to do a new prediction, you have to first select a model from the dedicated dropdown list and then a dataset uploaded on the project. Then, by clicking on the “launch prediction” button, the system will compute and generate a prediction file downloadable by clicking on the right side button on the prediction list below.

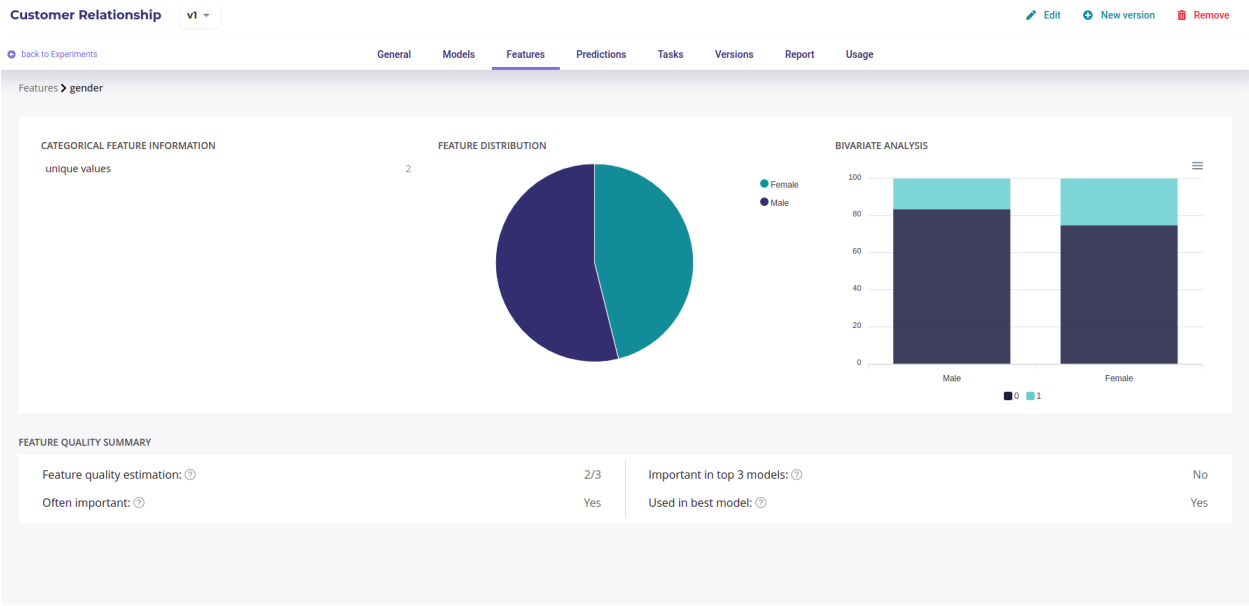


Fig. 63: Feature detail

The feature detail has some statistics chart about the features, its relation to the target and its signal power for the submitted problem

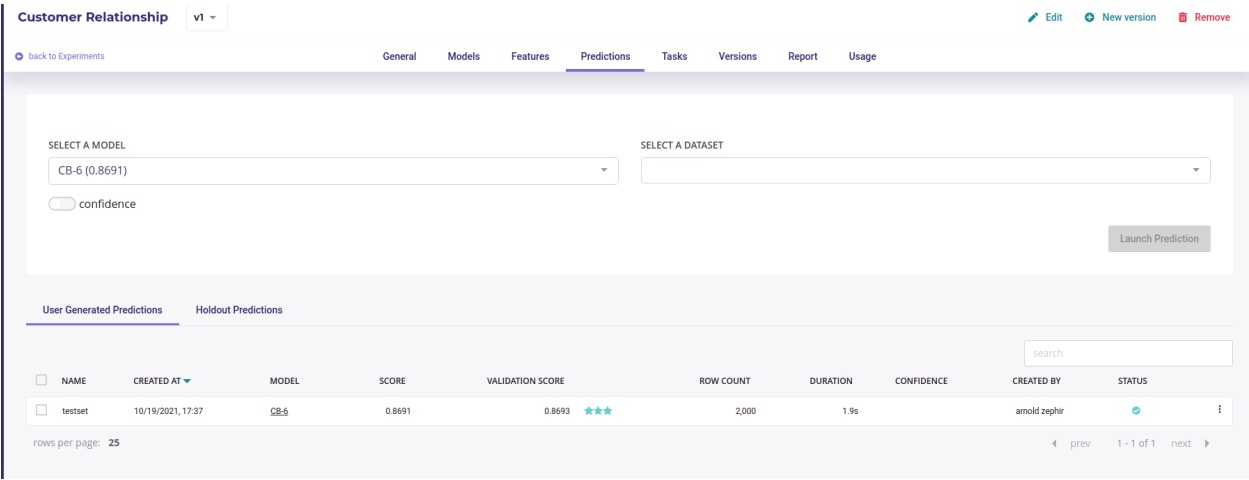


Fig. 64: Make and see predictions

Versioning your experiments

In the Prevision.io platform you can iterate versions of your experiments.

All the version keep one common thing : the target of your training, that defines your experiment. Yet from one version to the others you can change any parameters to see how it impacts performance and stability of your model.

The screenshot displays the 'Basics*' configuration tab for an experiment. It includes sections for 'Version description', 'Data', 'Training options', and 'Fields configuration'. The 'Data' section has a 'Dataset' dropdown set to 'trainset' and a 'Holdout (optional)' dropdown set to 'testset'. The 'Training options' section has a 'Metric to use' dropdown set to 'AUC - area under the receiver operating characteristic curve' and 'Performances' radio buttons with 'NORMAL' selected. The 'Fields configuration' section has a 'Target column' dropdown set to 'target' and an 'ID column (optional)' dropdown set to 'customerid'.

Fig. 65: The target stays the same and cannot be changed along versions of an experiment

To do that, three possibilities :

- On the experiment list of a project, by clicking on the right side action button of the experiment you want to iterate and select “new version”
- On any pages of a experiment by clicking on the top right “actions” button and select new version
- On the “Version” menu of a experiment, by clicking on the action button right side of a version listed and select “new version”

Then, on the version menu of a experiment, you will find the list of all trained versions for this experiment. By clicking on the version number, left side of this list, you will access the selected experiment version page. You can also navigate through versions by using the dropdown list top left of the information banner on any page of a experiment.

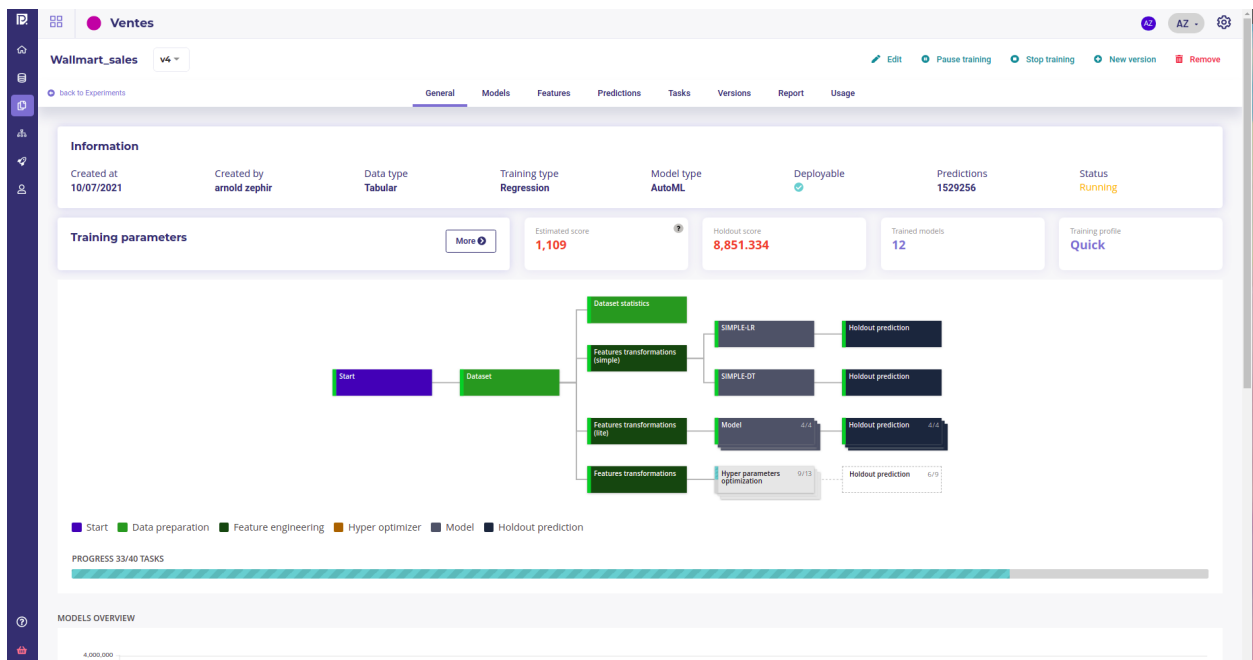
After clicking on a new version button, you will be redirected to the experiment configuration menu. The experiment version configuration you selected for your iteration will be automatically loaded in order for you to know what configuration was done and select the changement you want to apply.

Note: when creating a new experiment or a new version, add a description to your experiment at the first screen of new experiment configuration. It will help you finding the version you want to work with later.

VERSION	DESCRIPTION	CREATED AT	CREATED BY	SCORE	MODELS	PREDICTIONS	STATUS
V2		06/28/2021, 16:06	Simon Levacher	-	0	0	
V1		06/25/2021, 14:42	Axel Chauvin QA test	0.9587 (auc) ★★	8	0	

rows per page: 25

The default dashboard is those of the **last version** of your experiment. If you have many *version of your experiment*, you can change it with the dropdown menu on the top left corner.



The front page of experiment dashboard shows you :

- General : general information and comparison of your models in terms of performances
- Models : list view of the created models and information about the *trained models*
- Features : information about the way *the features* are used for the training and the configuration of the feature engineering
- Prediction : create *bulk predict* using CSV files and view all bulk predictions done for this usecase
- Task : Graph and listing of all operations done during training
- Versions : list of *all version* of the selected usecase
- Report : generate PDF reports explaining the models/usecases

The information header gives you the important information regarding your usecase. You can navigate through the versions using the dropdown list on the left side of this panel.

- Action button : on the top-right corner of the page are the actions buttons allowing you to :
 - edit the name and description of the experiment version

- create a new *version*
- delete the experiment
- Under the information panel, cards displaying information regarding your usecase are displayed. Please note that the holdout score card will be displayed only if a holdout was selected during training configuration
- Two graphs are displayed on the general page of a usecase showing : - The models ranked by score. By clicking on a model chart bar, you can access to the selected model details - Models score vs. estimated prediction time

Please note that for object detection, the general screen is quite different from the other use cases types. On the image detection general menu you will find a sample of images used during the train in orange, the predicted bounding boxes using cross validation and in blue, the true bounding boxes.

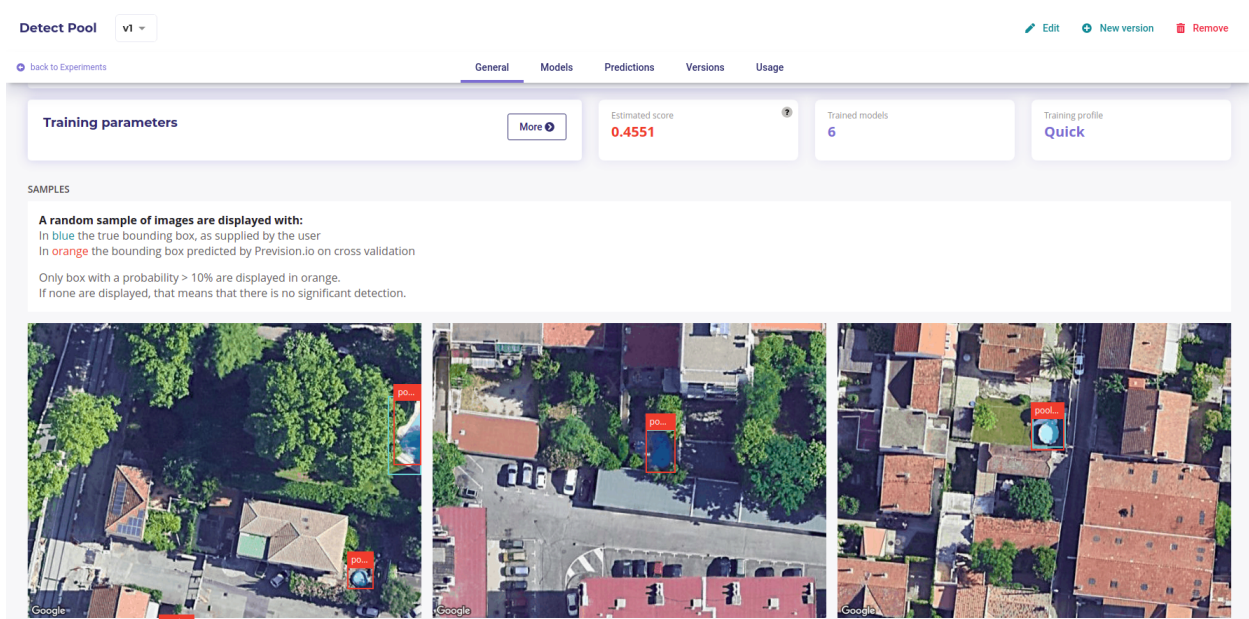


Fig. 66: Object detector dashboard

Tasks

In this menu you will find an overview of all tasks made by the platform during the usecase training and their status. The aim of this screen is to help you to better understand the operations made during the training and, if errors occurred, at which level it happened. When a task failed, you can access logs by clicking on the **logs button** that appears.

Two views are available :

- Liste view : list all single operations done
- DAG view : graphical view of single operations and their relation

You can switch between these views by clicking on the execution graph / tasks board switch.

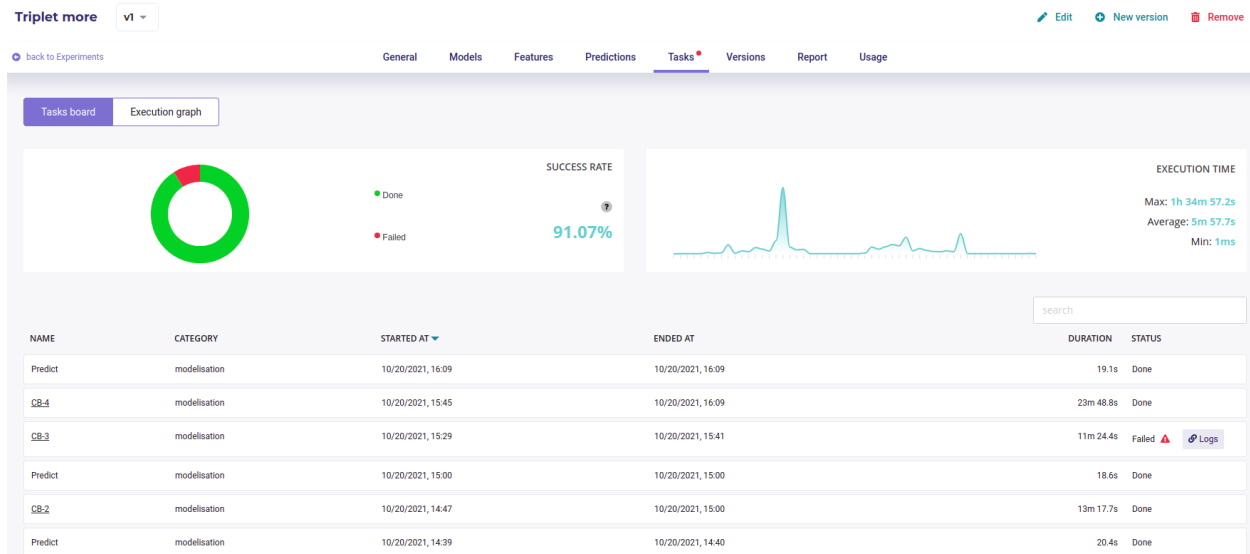


Fig. 67: All the tasks executed

Reports

In this menu, you can generate PDF reports regarding models from the usecase. To do that, once on the dedicated model menu, you will have to choose from the drop down the models you want to appear in the generated report and the feature importance count. You also can select explanations by check/uncheck the show explanation checkbox. Then, by clicking on the generate button, you will get an overview of the report. By clicking on the “print” button on the top of the overview, you will download the PDF report.

5.3.1.4 Pipelines

In Prevision.io platform you can automate several actions using the pipeline editor. We will see in this chapter the possibilities of the pipelines and its components and how the editor works. In order to execute a pipeline, several requirements need to be fulfilled :

- first, you have to create your own template using the pipeline editor. This template includes generic components with no configuration required in this step. This allows you to create a generic template and apply it several times on different experiments by configuring the component.
- then, you will be able to configure the pipeline run by choosing an already created pipeline template and by configuring the nodes to your experiment. You also can choose to run the pipeline manually or automatically by using the scheduler.

Optionally, you can create and load into the platform your own components and use them into pipelines.

Before going into detail about the creation of pipelines itself, let's have a look at the pipeline components existing in the platform and their purpose. This overview will help you to better understand the possibilities of Prevision.io pipelines

Pipeline components

Pipeline components can be considered as steps, or nodes, of the pipeline. Several categories of components are available in order to easily find them when building a pipeline template.

Some Components are provided on the shelf when you subscribe a Prevision.io plan.

NAME	NODE TYPE	CREATED BY	CREATED AT	DESCRIPTION
Import dataset	Import	Prevision	08/09/2021, 13:46	This component will import a dataset, previously load into the dataset s...
Import dataset from datasource	Import	Prevision	08/09/2021, 13:46	This component allows you to generate a dataset from an external sour...
Export dataset	Export	Prevision	08/12/2021, 15:02	This component allows you to create and save into the platform an Out...
deployment predict on regression	Predict	Prevision	08/09/2021, 13:46	This component allows you to automatize the generation of prediction ...
deployment predict multiclassification	Predict	Prevision	08/09/2021, 13:46	This component allows you to automatize the generation of prediction ...
deployment predict classification	Predict	Prevision	08/09/2021, 13:46	This component allows you to automatize the generation of prediction ...
Add country key	Prevision components	Prevision	08/09/2021, 13:46	This component will add different KPI for countries
Concatenate 2 columns	Prevision components	Prevision	08/09/2021, 13:46	This component will concatenate 2 columns using a selectable character
Sentiment Analysis	Prevision components	Prevision	08/09/2021, 13:46	This component will compute text sentiment analysis probability
Add siren siret	Prevision components	Prevision	08/09/2021, 13:46	This component will add siren and siret static features for each siret
Add POI data	Prevision components	Prevision	08/09/2021, 13:46	This component will add the number of electric charging stations within...
Add weather	Prevision components	Prevision	08/09/2021, 13:46	This component will add weather features depending on temporality an...
Sample	Prevision components	Prevision	08/09/2021, 13:46	This component will sample the input dataset according to the selectabl...
Add special days	Prevision components	Prevision	08/09/2021, 13:46	This component will add special days in country for each date
Pad column	Prevision components	Prevision	08/09/2021, 13:46	This component will pad column with "char" to "length".
Filter outliers	Prevision components	Prevision	08/09/2021, 13:46	This component will filter rows where values in numerical columns fall o...
Sentiment Product Review	Prevision components	Prevision	08/09/2021, 13:46	This component will compute text sentiment product review probability (...)
Add weather city day	Prevision components	Prevision	08/09/2021, 13:46	This component will add weather features in cities for each date
Sentiment Recognition	Prevision components	Prevision	08/09/2021, 13:46	This component will compute text sentiment analysis class
Add siren KPI	Prevision components	Prevision	08/09/2021, 13:46	This component will add yearly accounts for companies, based on date ...

- Import : All component relative to the import of data
- Export : All component relative to the export of data
- Prevision component : Various component developed by Prevision.io datascientists for their various projects
- Custom component : Components developed by you or your team
- Predict : All components relative to the automatisisation of the predictions
- Retrain : All components relative to the automatisisation of model training

Each component has a description helping you to choose the ones suitable for your needs. You can access all components by clicking on the pipelines menu on the side project's menu and, navigate to the pipeline components menu.

NAME	NODE TYPE	CREATED BY	CREATED AT	DESCRIPTION
Import dataset	Import	Prevision	08/09/2021, 13:46	This component will import a dataset, previously load into the dataset s...
Import dataset from datasource	Import	Prevision	08/09/2021, 13:46	This component allows you to generate a dataset from an external sour...
Export dataset	Export	Prevision	08/09/2021, 13:46	This component allows you to create and save into the platform an Out...

Building you own component

You can build and use your own component for custom dataset transform.

A boilerplate with more guide is available on the [Prevision.io public repo](#)

To use your own component you need a gitlab or a github account (and it needs to be setup in your account page)

Once done, your component may be use in any Pipeline Template.

Pipeline templates

Pipeline Template are a tool for describing operations to schedule. In a pipeline template, you do not input any parameters. Instead your build a pipeline by linking nodes together and setting some placeholder.

Your pipeline template will then be used in the schedule step, where you, or someone else, is going to fill the placeholder with their inputs.

For example, you can define a template that “add an integer to the age column”. The value of this integer will fill in when a scheduler use your template.

A template may be used in different scheduled run with different parameters each time.

In order to create a new pipeline template, you have to navigate through the pipeline template menu.

NAME	DESCRIPTION	NODES	CREATED AT	CREATED BY	USED IN RUN
add_siren_kpi_template	test component add_siren_kpi	3	08/11/2021, 15:32	gerome pistre	true
bulk	bulk prediction	3	08/10/2021, 09:58	gerome pistre	
pipeline0	first one !	2	08/09/2021, 14:54	gerome pistre	
Test fred Draft	La descr de fred	4	08/09/2021, 14:45	Frédéric OGLAZA	

rows per page: 25

You will then access the pipeline template liste of the project. By clicking on the “new template” button, you will access to the pipeline template editor.

The first step is to set up the name of your pipeline template and, optionally, a description.

New template

1 Settings > 2 Template

Name

0 / 40 characters

Description (optional)

0 / 210 characters

Once (and only once) the required information is fulfilled, you will be able to reach the next step by clicking on the next button bottom right.

Then, the pipeline graphical editor will be visible and you will be able to start the creation of your pipeline template. In order to add your first node, you have to click on the “+” button in the center of the graphical area.

Cancel and back

New template

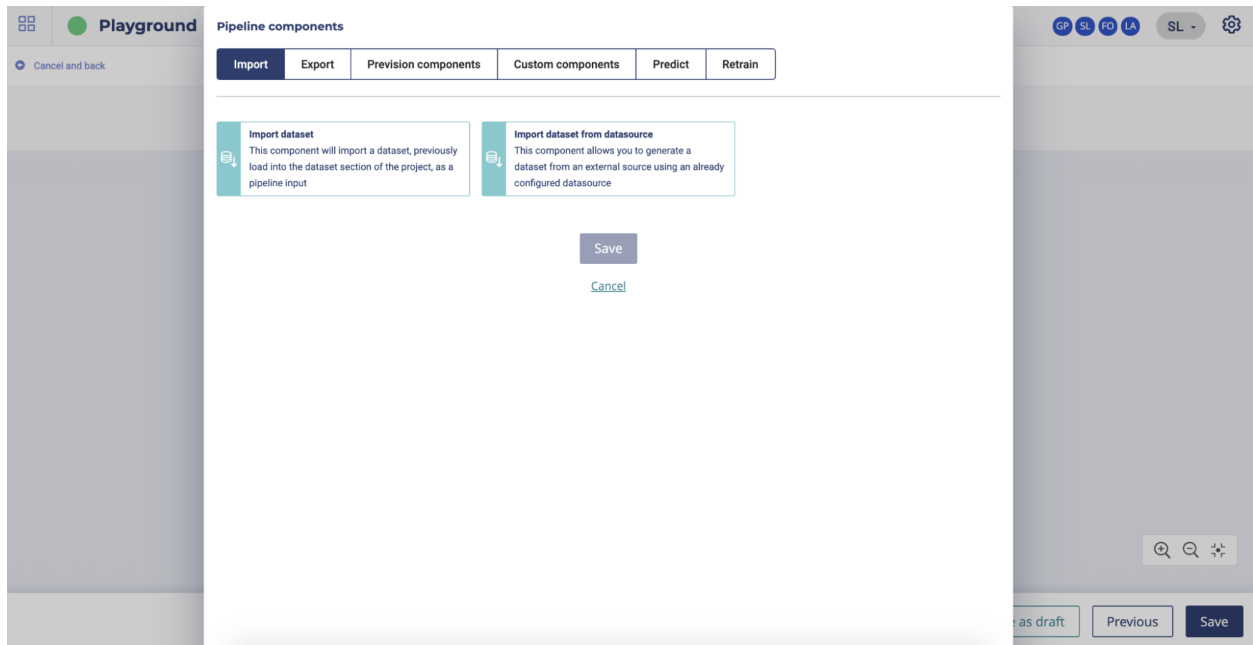
✓ Settings > 2 Template

+

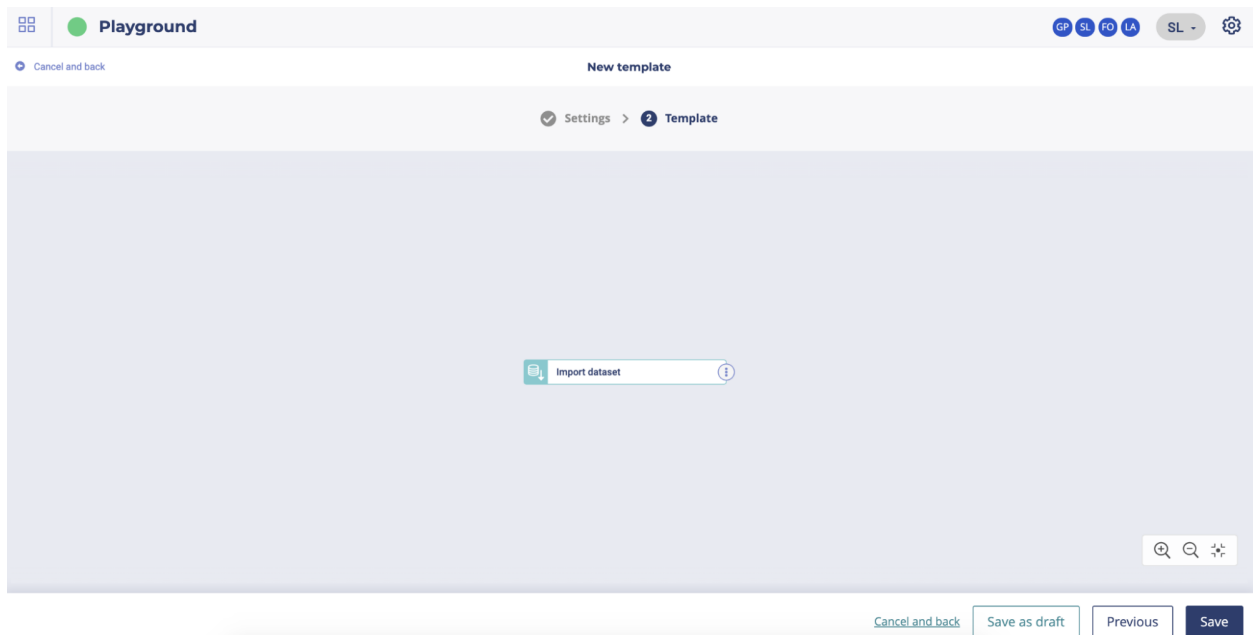
Cancel and back Save as draft Previous Save

Then a popup window will appear allowing you to select the first node. In this popup, the components are classified in several categories. You can navigate through nodes categories by clicking on the top bar menu. In order to add a node, you simply have to click on it and save. Please note that the eligible nodes are colored unless ineligible ones that are in gray. This will help you through the creation of your pipeline in order to be sure that the final result is conform to

what the platform is expected.



The selected node will be visible in the graphical view. Please note that you can save as draft your pipeline template in order to finish it later by clicking on the bottom right “save as draft” button.



Several actions are possible on the newly added node. You can access to the possible actions by clicking on the more action button on the right side of the node.



Four actions are possible :

- add a node after the selected one
- switch this node for another
- view settings of the node
- delete the node

Please note that some special nodes can have limited actions.

Once your pipeline template is finished, you can save it by clicking on the bottom right “save” button. You will be then redirected to the pipeline template list.

Scheduled runs

A scheduled run is the combination of a pipeline template and a specific schedule. It allows to trigger some pipeline at regular interval.

Once you had defined a pipeline template, you can use schedule it for running :

- Once
- periodically forever
- periodically for a defined period of time

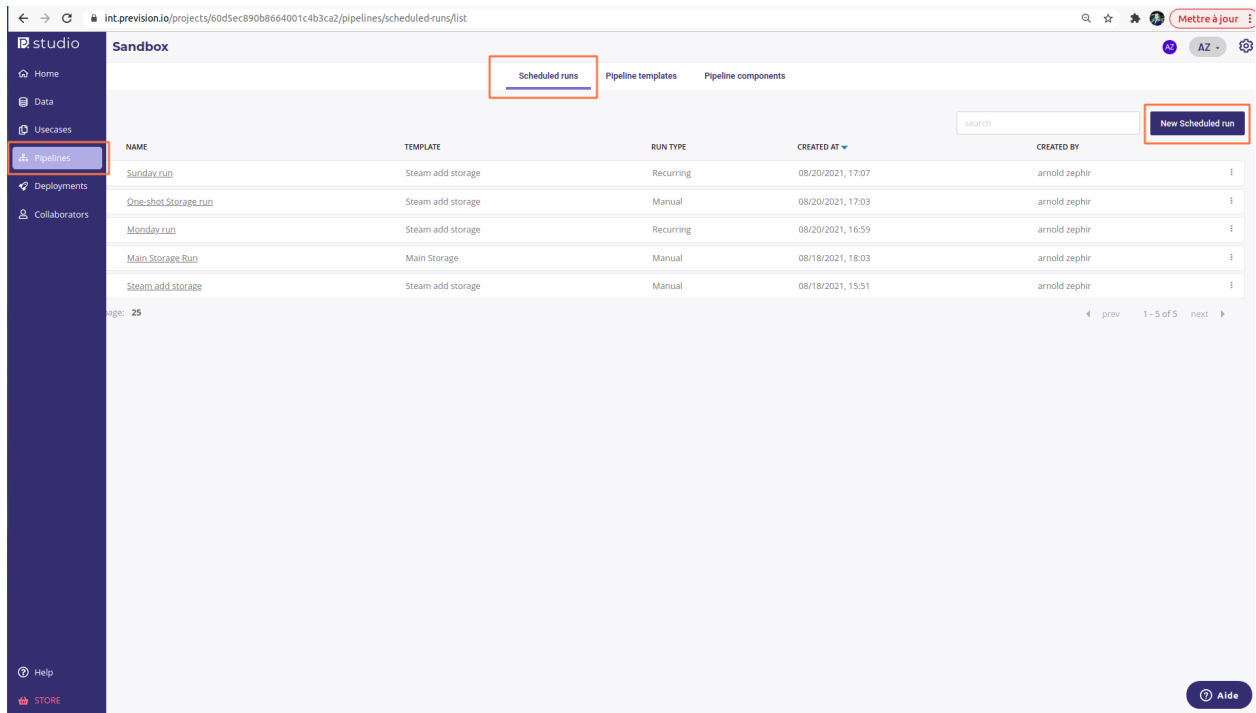
When to used scheduled runs ?

In most of case, scheduled runs are used :

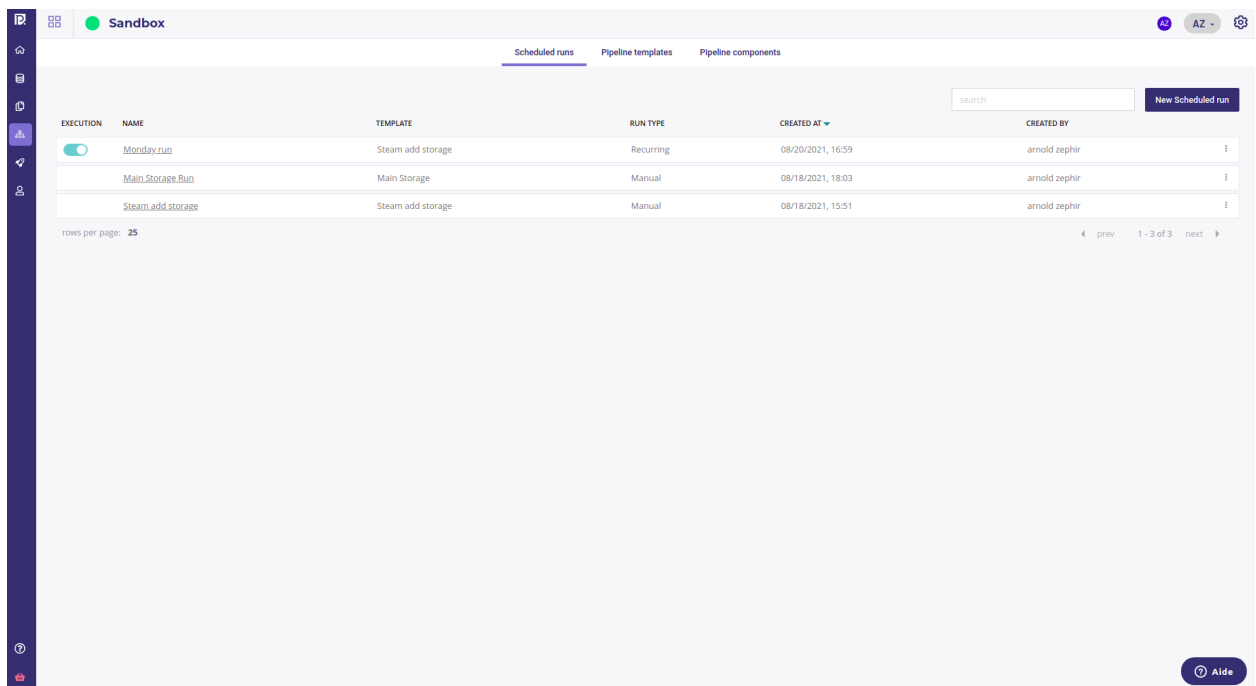
- for delivering prediction on a periodic schedule. For example sending a list of churner to sales team each monday
- for retraining a model, for example each first day of a quarter
- for computing engineered features and pushing them to others teams on a regular basis

Create a new scheduled running

Scheduled run are available in the Pipeline section of a project, available under the “Scheduled runs” tab.



From the main page, you can view a list of all your scheduled runs and create a new one by clicking on “New Scheduled run” button



First step is to give a name and some description and select a pipeline template.

Select the template you want to fill in and click on “next” on the right lower corner.

Note: Pipeline templates were created in a previous step, with some input inside nodes to fill in. “Scheduled runs” is the place where you are to fill them up.

Cancel and back

New Scheduled run

Settings > Node settings > Schedule

Name
One-shot Storage run 20 / 40 characters

Description (optional)
Add the storage requirement to the main dataset, extracted from Text 68 / 210 characters

Template
Steam add storage

Cancel and back Save as draft Next

The next screen is where you fill all the parameters of your pipeline. For each node with one or more parameter to provide, a *to configure* yellow label is displayed on the node.

Cancel and back

New Scheduled run

Settings > Node settings > Schedule

Steam add storage

Import dataset
To configure

Create the output path
To configure

Export dataset
Editable configuration

Configuration 1/3

Create the output path
Get storage

req_col
minimum_requirements

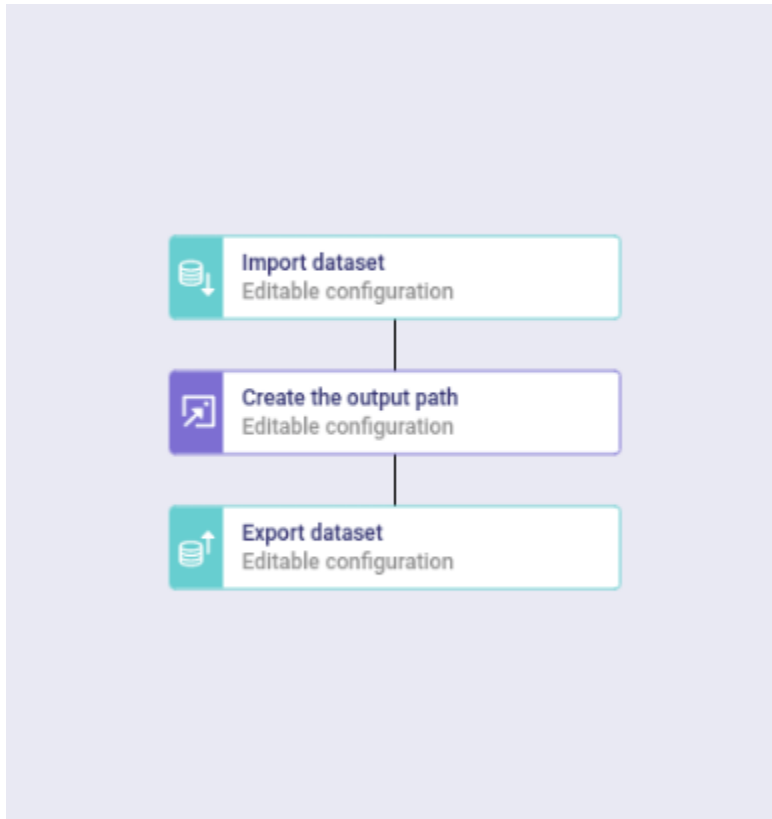
Save

Cancel and back Save as draft Previous Next

To enter a parameter, click on each node and fill the input. **Click on save to save them** (parameters are not taken into account till you click on save). To know more about a parameter, you may go to the “pipeline components” tab and

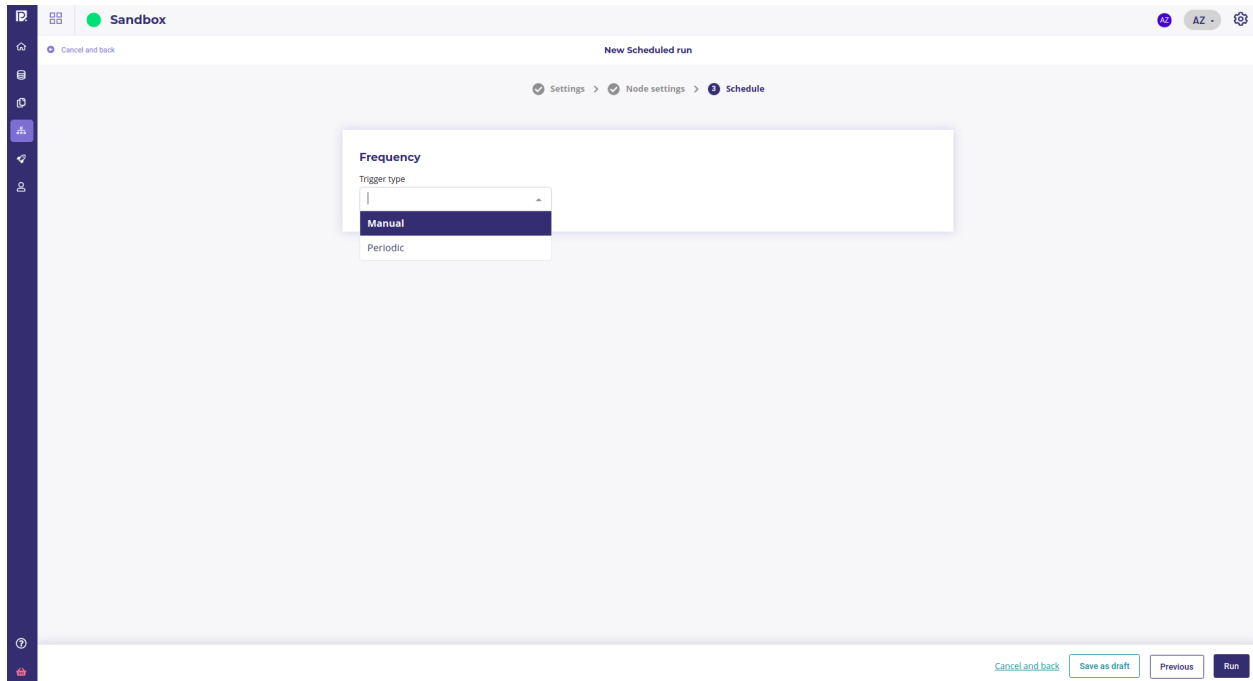
click on the component to get input description.

Once you filled all the node parameters, there should be no more yellow label :



You can click on “next” and choose the Trigger type :

- Manual : to run your pipeline once (useful for testing). The pipeline will be run as soon as you click on run (it can be run as much as you want later)
- Periodic : to run your pipeline at a given period for some duration



If you choose “periodic” you will be prompt to input some information :

- hour (and minutes) for hourly run. The pipeline will be launch every day, every x hours. for Example, if you input 2 hours, the pipeline will be ran twelve time a day (every 2 hours)
- day period, hour and minutes of execution for daily run.

Note: Note that “Day” is the number of day between each run. If you input “5”,the pipeline will run every 5 days.

- day of the week (at least one), hour and minut for a weekly period
- months period, day, hour and minute for monthly period
- a crontab expression (see [crontab guide](#) for syntax) if you select advanced mode

Frequency

Trigger type
Periodic

Period

Hourly
Daily
Weekly
Monthly
Advanced

Hour 12am Minute 0

Duration (optional)

Cancel and back Save as draft Previous Run

Once period input, you may select a start and end of run. The pipeline will only run on between the date you selected.

Frequency

Trigger type
Periodic

Period

Hourly
Daily
Weekly
Monthly
Advanced

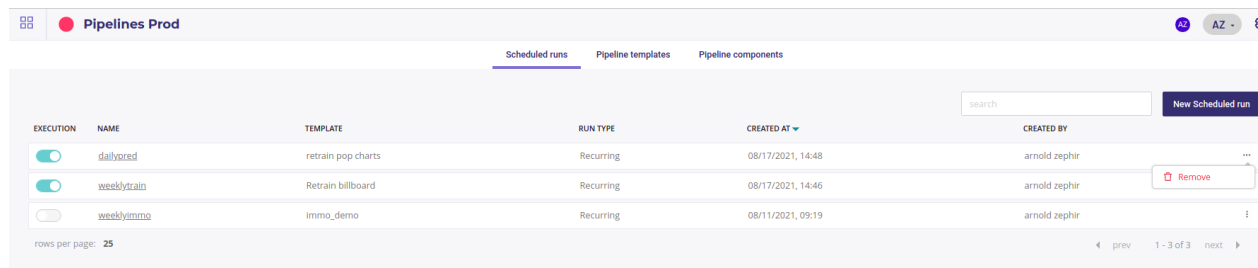
Hour 12am Minute 0

2021-09-15

Cancel and back Save as draft Previous Run

See all your scheduled run

All your scheduled run are available under the Scheduled run tab (in the pipeline section) :



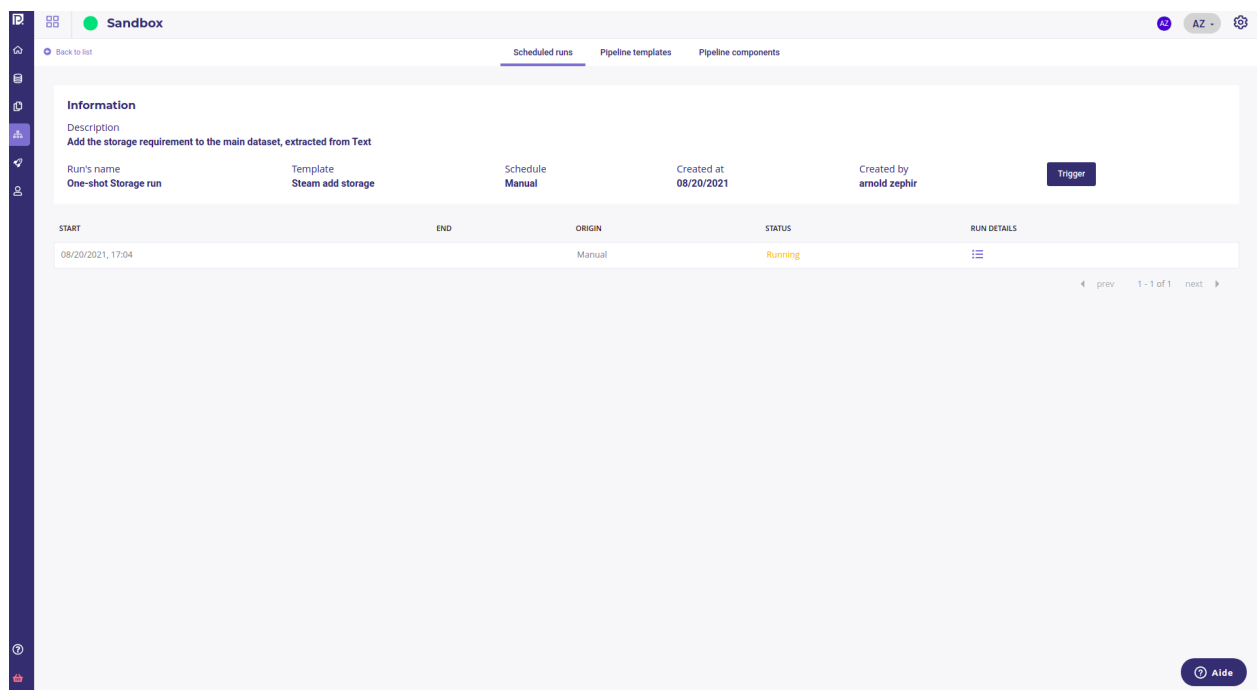
The screenshot shows the 'Pipelines Prod' interface with the 'Scheduled runs' tab selected. It displays a table of scheduled runs with columns for Execution, Name, Template, Run Type, Created At, and Created By. There are three rows of data, each with an execution toggle switch. The first two rows have their toggles turned on, while the third is turned off. A 'New Scheduled run' button is in the top right, and a 'Remove' button is next to the second row. Pagination shows 'rows per page: 25' and '1 - 3 of 3'.

EXECUTION	NAME	TEMPLATE	RUN TYPE	CREATED AT	CREATED BY
<input checked="" type="checkbox"/>	dailyprod	retrain pop charts	Recurring	08/17/2021, 14:48	arnold zephir
<input checked="" type="checkbox"/>	weeklytrain	Retrain billboard	Recurring	08/17/2021, 14:46	arnold zephir
<input type="checkbox"/>	weeklymmo	mmo_demo	Recurring	08/11/2021, 09:19	arnold zephir

The scheduled run with a periodic schedule have a switch in order to disable or enable them. A disabled Run will obviously not run.

The “Manual” Scheduled run does not have Execution switch. Instead you need to trigger them manually.

See run state, trigger a run



The screenshot shows the 'Sandbox' interface with the 'Scheduled runs' tab selected. It displays details for a specific run, including a description, run name, template, schedule, created at, and created by. A 'Trigger' button is visible. Below this is a table showing the run's execution details, including start, end, origin, status, and run details. The status is 'Running'.

START	END	ORIGIN	STATUS	RUN DETAILS
08/20/2021, 17:04		Manual	Running	

When clicking on a Scheduled Run in the list, you get details about its executions :

- either previous executions, with a “done” status or a “failed” status
- current execution status (“running”)

If an execution failed, you can get more informations by clicking on the “run detail” icon. For all execution that succeeded, the dataset produced are available in your “data” Section with the tag “pipeline output”

Each Scheduler Run can be trigger to run once and immediately by using the “Trigger” button.

Edit a run

You cannot edit a scheduled run, you have to create a new one.

Delete a Scheduled run

In the list of Scheduled Run , you can use the “More actions” icons to remove a scheduled run. Note that instead of removing it, you can disabled it by using the “execution” switch.

5.3.1.5 Deployments

Deployments are the last step of a *Machine Learning Project* and are used to

- share model across your organisation
- start monitoring you models

Deployments need *experiments with at least one model*. You can deploy any model of any version of your experiment.

Once an experiment is deployed...

- you can *schedule batch prediction*
 - you can monitor performance of one main model and one challenger
 - external users can use your model for unit prediction from an url
 - external application can call your model from a REST API
-

Deployments are scoped to a project and available from the **deployments section** on the collapsing sidebar. When entering the deployment section, you will see a list of each of your deployment and two status :

- deployed : are models built and available over API
- running : does API reponses to request

The url column links to a page where human can call the model over a simple form in order to test it.

Create a new Deployments

Creating a new deployment is done by clicking on the **deploy a new experiment** button under the **deployments experiments** tab.

In addition to give it a name and a description, you must :

- select one of your experiment
- select a version
- select a model from this version of your experiment

and you could select another version and another model of the same experiment taht will be deployed as Challenger model. This one will be called each time you main model is called and it's response will be recorded next to those of the main model in order to compare them and maybe be switch them.

When deploying a new experiment you need to grant access :

- public : everybody can call your model

Deploy a new experiment

Deployment name

Description (optional)

Select a experiment

Select a version

Select a tagged model to deploy

Select a challenger model (Optional)

Deploy

ACCESS RIGHTS

☒ Public ☐ Instance collaborators ☐ Project collaborators

Fig. 68: Create a new deployment

- Instance collaborators : everybody on the instance can call your model (note : every user on cloud.prevision.io share the same instance)
- Project collaborators : only your project's collaborators can call your model

Once you click on the deploy button, the model you chose will be deployed. You can check its status in the list of deployments or in the deployment page.

Inspect and monitor a deployed experiment

The deployment page is available as soon as the experiment is deployed. On each deployment page lie five sections.

General

DEPLOYED MODELS

MODEL	EXPERIMENT VERSION	SCORE	TYPE
CB-6		0.8691	MAIN
XGB-5		0.8631	Challenger

API KEYS

API keys allow you to access your application or model through a third-part app. Access is made through OAuth2. You can create as many keys as you want per application, and revoke them on demand.

[generate new key](#)

Client Id	Client secret	Actions
api-key_churn_68831249-5570-4e9e-b4f0-ea38e04305ea	copy	copy revoke

Fig. 69: General info about your deployment

The general tab will show you :

- Creation date of your deployment
- Current version of your deployment
- A link for using the model with a simple form
- deployment status
- main model status
- challenger model status
- a link to the documentation of model API
- the access you granted

And

- a drift chart : a chart showing distribution of the training data vs distribution of the data predicted since deployed
- a summary of the models deployed
- an API key generator than you can share to others applications in order to call your model

Monitoring

The monitoring section will show you a selection of chart of :

- the data sent to your models
- the prediction done by your models, both main and challenger if you deployed a challenger model
- the drift of your input data



Fig. 70: Model monitoring

Monitoring this chart let you decide when your model becomes obsolete and you should schedule a retrain.

Predictions

VERSION	CREATED AT	ROWS	TYPE	MODEL	SCORE	DURATION	STATUS	DOWNLOAD
V1	10/11/2021, 15:41	127438	Main	LGB-3	9,725	8s	Done	Download
			Challenger	XGB-4	8,970	5.8s	Done	Download
V1	10/11/2021, 16:01	127438	Main	LGB-3	9,725	7.4s	Done	Download
			Challenger	XGB-4	8,970	5.7s	Done	Download
V1	10/11/2021, 17:02	127438	Main	LGB-3	9,725	7.6s	Done	Download
			Challenger	XGB-4	8,970	6s	Done	Download
V1	10/11/2021, 18:03	127438	Main	LGB-3	9,725	7.3s	Done	Download
			Challenger	XGB-4	8,970	5.3s	Done	Download
V1	10/11/2021, 19:01	127438	Main	LGB-3	9,725	7.7s	Done	Download
			Challenger	XGB-4	8,970	6.2s	Done	Download
V1	10/11/2021, 20:05	127438	Main	LGB-3	9,725	7.4s	Done	Download
			Challenger	XGB-4	8,970	6.5s	Done	Download
V1	10/11/2021, 21:05	127438	Main	LGB-3	9,725	7.7s	Done	Download
			Challenger	XGB-4	8,970	6.6s	Done	Download

Fig. 71: Predictions list

The Predictions tab is where you will find all the predictions scheduled in a *pipeline* done with your deployed experiment. When an experiment is deployed, it can be used in a pipeline and this, can be schedule, to be deliver in an external database each monday for example. Each time a prediction is ran in a pipeline, a file is generated both for the main model and the challenger model and they can be downloaded for further inspection from this section.

Versions

At any moment you can change the model of your experiment that is called, picking it in any version of your experiment. By doing so, you will replace the models called when someone use your deployed experiment without anyone noticing or having to change its client call.

When creating a new version you can change both the main model and the challenger model. You can rollback to a previous version of your deployment just by clicking on the Rollback link. The version clicked will then be used again to do predictions.

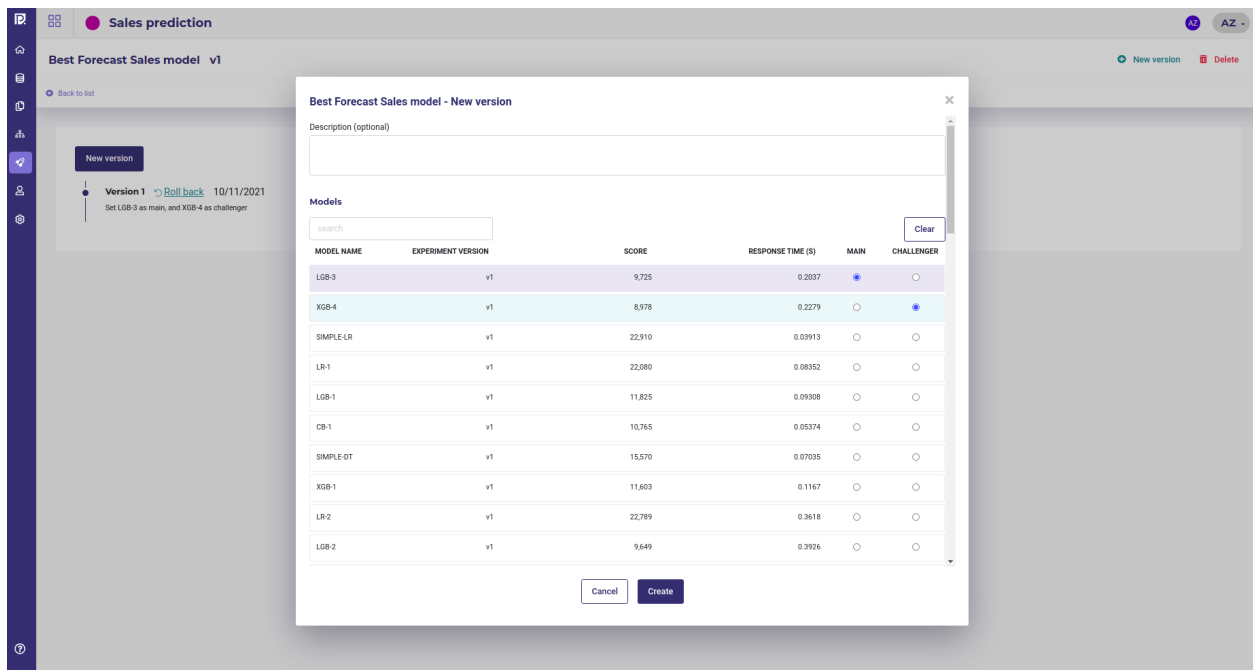


Fig. 72: Managing version of a deployment

Usage

The usage section displays CPU and memory usage of your deployed experiment.

Edit and remove a deployment

You can not edit a deployment but you can remove it either from the list of deployment, with the 3-dots menu on the right of the list items, or on the individual page of each deployment with the Delete button.

5.3.1.6 Contributors

By clicking on contributors on the main menu of a project, you will access the list of all users of the project. If you have sufficient rights ("Admin") on this project, you will be able to add & delete users from the project and modify users' rights.

Roles & rules

- Viewer : you can access to all pages (except project settings) with no possibility of creation or edition
- Contributor : viewer rights + you can edit and create resources inside the project
- Admin : contributor rights + you can manage users and modify project properties

Add & delete

If you are admin in a project, you can manage users into your project.

Project's collaborators

Share this project with collaborators

admin

Invite this collaborator

NAME	EMAIL	ROLE	
Axel Chauvin QA test	axel.chauvin@prevision.io	admin	
Simon Levacher	simon.levacher@prevision.io	admin	

To add a user, you have to enter the collaborator email into the top left field, set his right using the dropdown menu and click on the “invite this collaborator”. Please note that you can only invite collaborators that already have a prevision.io account.

To change the rights of a user into the project, you just have to select the new role using the dropdown. To be sure that the project and users properties can be managed, at least one collaborator have to be admin of the project.

To remove a collaborator from the project, use the trash button on the left side of the list.

Project settings

If you are admin on a project, the project settings button is enabled and, by clicking on it, you will access the project setting page.

Project's settings

Edit the project's settings

SETTINGS

Project's name
 24 / 40 characters

Project's description (optional)
 0 / 210 characters

Cancel

Save new project's settings

Delete the project

Set a color to this project

Change color

PREVIEW

QA - Multiclassification

axel.chauvin@prevision.io

05/18/2021, 11:06

2

4

Collaborators: AC SL UU

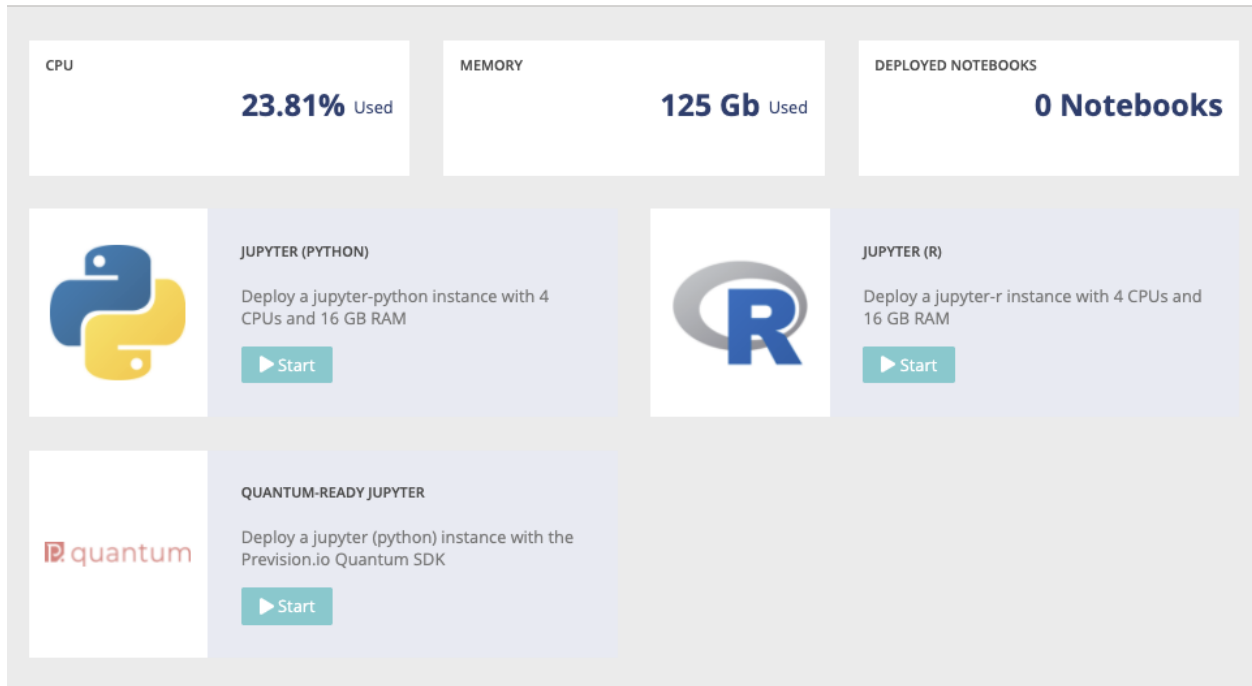
On this page, you can :

- update name, description and color of your project
- delete the project. Please note that if you delete a project, all resources linked to the project will be deleted (experiments, datasets, deployed models, etc.)

5.3.1.7 Notebooks

Introduction

Prevision.io offers various tools to enable data science use cases to be carried out. Among these tools, there are notebooks and production tools. Notebooks are not scoped into projects. You can access notebooks by clicking on the notebook button on the left main menu. Then you will be redirected to the following page.



Jupyter (python)

For Python users, a JUPYTERLAB environment (<https://github.com/jupyterlab/jupyterlab>) is available in Prevision.io

Note that a set of packages is preinstalled on your instance (list: https://previsionio.readthedocs.io/fr/latest/_static/ressources/packages_python.txt), particularly the previsionio package that encapsulates the functions using the tool's native APIs. Package documentation link: <https://prevision-python.readthedocs.io/en/latest/>

Jupyter (R studio)

For R users, a R STUDIO environment (<https://www.rstudio.com>) is available in Prevision.io

Note that a set of packages is preinstalled on your instance (list: https://previsionio.readthedocs.io/fr/latest/_static/ressources/packages_R.txt), particularly the previsionio package that encapsulates the functions that use the tool's native APIs. Package documentation link: https://previsionio.readthedocs.io/fr/latest/_static/ressources/previsionio.pdf

5.3.1.8 Settings

Most of your settings are in the profile section, from the upper-right menu

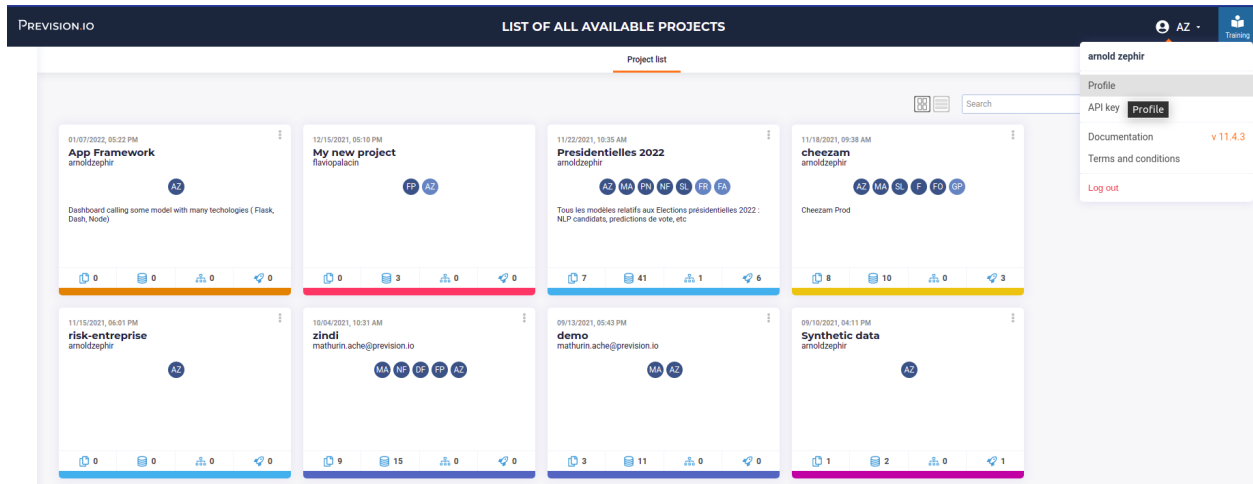


Fig. 73: Profile Section is un the upper right menu

Here you can find you master Token (“API”) and you security information for authentication with 3rd party service.

Connecting my code

For a lot of Prevision feature (components, app deployment,...) you may connect your git repo, either Gitlab or github.

This takes place in the Federated identity tab :

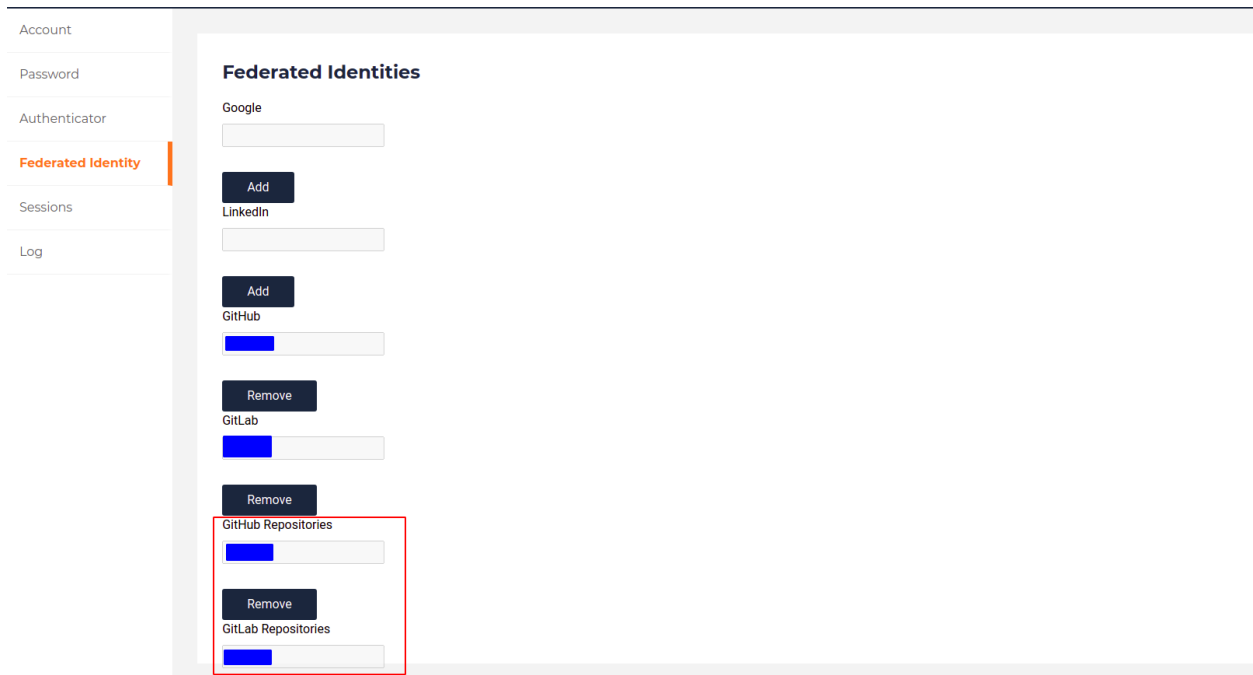


Fig. 74: Connecting GIT repo

Account and repositories

Github and gitlab offer both OAuth2 authentication and git repo hosting. the *github* and *gitlab* fields are for OAuth2 service, the *github repositories* and *gitlab repositories* are for connecting your code repo.

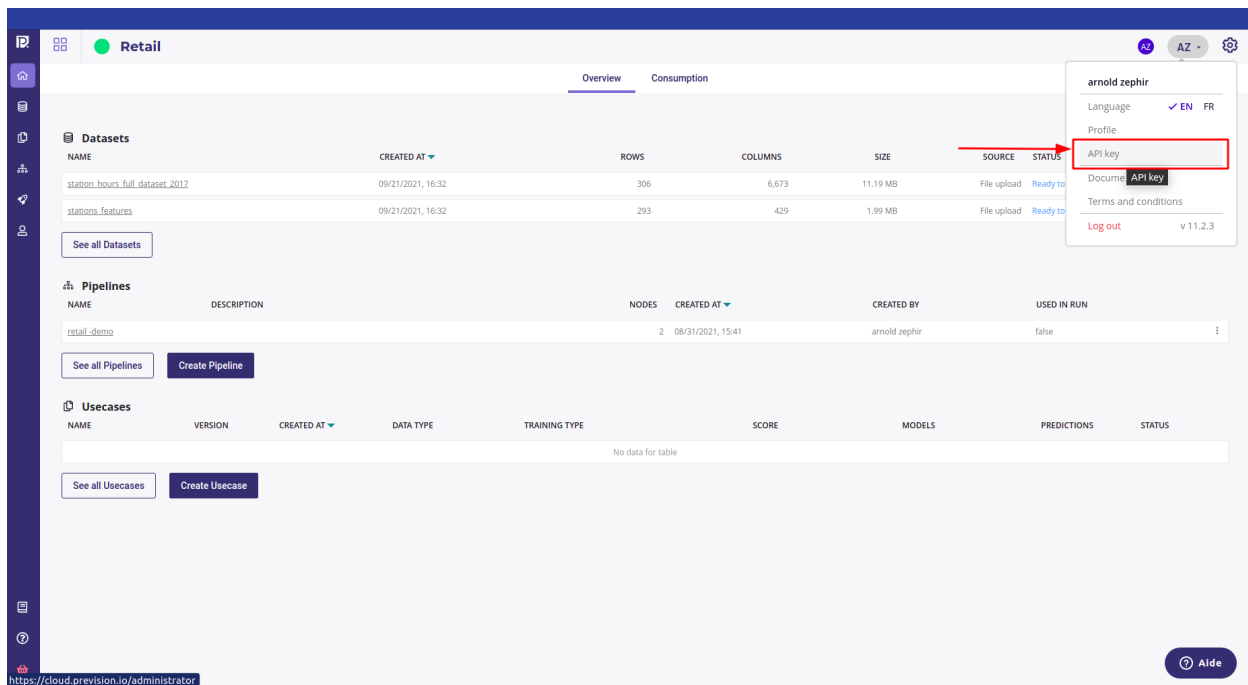
Just click on the Add button and enter your credentials. Your code repo will now be available in the *Components* section and *Deployments* section of the Prevision Studio.

5.3.2 API

5.3.2.1 Using The API

You can access every object generated by Prevision Platform from the *API*.

To run the Code below you need your Master Token. It is available in the API Key page of your user settings :



You can use the native urllib module to parse API. This page show how to use API with raw python but we suggest to use the SDK (*Python* and *R*) for an higher level of abstraction.

First, import native python 3 urllib.request and set up your Token (warning : if you have an on promise server or custom dedicated domain, you need to replace the url "cloud.prevision.io" with your own)

```

1 import urllib.request
2 import pandas as pd
3 import ssl
4 import json
5
6 MASTER_TOKEN="<YOUR_MASTER_TOKEN>"
7
8 BASE_PATH ="https://cloud.prevision.io/ext/v1"
```

(continues on next page)

(continued from previous page)

```

9
10 projectsurl = f"{BASE_PATH}/projects"
11 request = urllib.request.Request(projectsurl)
12 request.add_header('Authorization', MASTER_TOKEN )
13 # Disable SSL check
14 projectslist = urllib.request.urlopen(request, context=ssl.SSLContext()).read()
15 projectslist = json.loads(projectslist)

```

5.3.3 SDK

The following parts contain information about Prevision.io SDKs, which allow to access to the platform's features programmatically.

If your looking for the full documentation of the software's APIs you will find it [here](#).

5.3.3.1 Using the Python SDK

Standard Machine Learning Workflow with Prevision SDK

Here is a list of standard and common workflows you can acheive with Prevision Python SDK. You may read the [Prevision Public github](#) or the [API reference](#) too.

External Model

You can get code for runing this guide on the [Getting started guide](#)

First import all the modules

```

1 import previsionio as pio
2 import yaml
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.pipeline import make_pipeline
5 from sklearn.preprocessing import OrdinalEncoder
6 from sklearn.neighbors import KNeighborsClassifier
7 from skl2onnx import convert_sklearn
8 from skl2onnx.common.data_types import FloatTensorType
9 import numpy as np
10 import logging

```

Setup your token account (see [Using The API](#)) and some parameter for your project, like its name, the name of the datasets...

Note that you always must create a Project for hosting datasets and experiments.

```

1 import os
2 from os.path import join
3 from dotenv import load_dotenv
4
5 load_dotenv()
6

```

(continues on next page)

(continued from previous page)

```

7 PROJECT_NAME="Sklearn models Comparison"
8 TRAINSET_NAME="fraud_train"
9 HOLDOUT_NAME="fraud_holdout"
10 INPUT_PATH=join("data","assets")
11 TARGET = 'fraude'
12
13
14 pio.client.init_client(
15     token=os.environ['PIO_MASTER_TOKEN'],
16     prevision_url=os.environ['DOMAIN'])

```

Create a New project, or reuse an existing one

```

1 projects_list = pio.Project.list()
2 # Create a new Project or using the old one
3
4 if PROJECT_NAME not in [p.name for p in projects_list] :
5     project = pio.Project.new(name=PROJECT_NAME, description="An experiment using ")
6 else :
7     project = [p for p in projects_list if p.name==PROJECT_NAME] [0]

```

Add the dataset to the projects or get the existing one if already uploaded (the dataset will be automatically uploaded to your account when you create them)

```

1 datasets_list = project.list_datasets()
2 for d in datasets_list:
3     if TRAINSET_NAME in [d.name for d in datasets_list] :
4         train = [d for d in datasets_list if d.name==TRAINSET_NAME] [0]
5     else :
6         train = project.create_dataset(file_name=join(INPUT_PATH,"trainset_fraud.csv"),
7         ↪name='fraud_train')
8
9     if HOLDOUT_NAME in [d.name for d in datasets_list] :
10         test = [d for d in datasets_list if d.name==HOLDOUT_NAME] [0]
11     else :
12         test = project.create_dataset(file_name=join(INPUT_PATH,"holdout_fraud.csv"),
13         ↪name='fraud_holdout')

```

Beware to converting the data to the right type before makgin your dataset

```

1 train_data = train.data.astype(np.float32)
2 test_data = test.data.astype(np.float32)
3
4 X_train = train_data.drop(TARGET, axis=1)
5 y_train = train_data[TARGET]

```

Then train some classifiers (you may upload many models at once) and create an yaml file to hodi the models configuration.

```

1 classifiers=[ {
2     "name":"lrsklearn",
3     "algo":LogisticRegression(max_iter=3000)
4 },

```

(continues on next page)

(continued from previous page)

```

5         {
6             "name": "knnsk",
7             "algo": KNeighborsClassifier(3)
8         }
9     ]
10
11 initial_type = [('float_input', FloatTensorType([None, X_train.shape[1]]))]
12
13
14 config={}
15 config["class_names"] = [str(c) for c in set(y_train)]
16 config["input"] = [str(feature) for feature in X_train.columns]
17 with open(join(INPUT_PATH, 'logreg_fraude.yaml'), 'w') as f:
18     yaml.dump(config, f)

```

Sklearn Pipeline are supported so you may build any pipeline you want as long as you provide the right config file. Convert each of your model to an onnx file once fitted :

```

1 for clf in classifiers :
2     logging
3     clr = make_pipeline(OrdinalEncoder(), clf["algo"])
4     clr.fit(X_train, y_train )
5
6     onx = convert_sklearn(clr, initial_types=initial_type)
7     with open(join(INPUT_PATH, f'{clf["name"]}_logreg_fraude.onnx'), 'wb') as f:
8         f.write(onx.SerializeToString())

```

And last, use the [Project create_external_classification method](#) to upload all your models at once in the same experiment

Note: You can upload many onnx file in the same experiment in order to becnhmark them. To do that you must provide a list of tuple, one for each onnx file with :

- a name
- the path to your onnx file
- the path to your config file (often the same for each model

```

1 external_models=[(clf["name"], join(INPUT_PATH, f'{clf["name"]}_logreg_fraude.onnx'),
2 ↪ join(INPUT_PATH, 'logreg_fraude.yaml')) for clf in classifiers ]
3 exp = project.create_external_classification(experiment_name=f'churn_sklearn_{clf["name"]}',
4 ↪ "]",
5
6         dataset=train,
7         holdout_dataset=test,
8         target_column=TARGET,
9         external_models = external_models
10     )

```

5.3.3.2 Using the R SDK

- [Getting started](#)
- [API reference](#)
- [Source code](#)

5.3.3.3 Using the Prevision Quantum NN SDK

Prevision-quantum-nn is a library that allows to handle automatically quantum variational circuits. Its main page is available [here](#).

5.3.4 Guides and How-to

5.3.4.1 Datascience Guide

How and when to do data embedding exploration

What is this guide about ?

Data exploration is an important step of Data modelisation and Machine Learning projects lifecycle. Even if not needed for supervised modelisation, as algorithm are now powerful enough to build the best model without human insight, Data Exploration may be useful when starting the project :

- to check data quality and integrity (even if “no data” is still an important insight)
- to check modelisation feasibility with visual hint
- and, more important, to onboard the line of business user and formalize intuition and goal of the project !

The last point is probably the most important. In every Datascience project, first and most important step is to define clear objective that serve purpose. Exploring data with visual tools oftent allow to get insight from business expert and get them involved in the project

What are we learning in this Guide ?

This guide is splitted in 4 sections :

- How does data exploration takes place in the Machine Learning Pipeline ?
- The principles of vector embedding anything
- How to make data embedding in Prevision Platform ?
- What to do with embedding ?

Data exploration in the Machine Learning pipeline

What's data exploration ?

We define data exploration as any process that take raw data and produce indicators and charts for human to analyse. Statistics is a kind of data exploration. Scatter plot and histogram are an other kind.

Sometimes, Humans can build very basic models from statistical indicators and get rules-based model like *if age > 40 then wants_motobike = true*.

In a Machine Learning project pipeline, Data exploration may serve the 3 following purpose.

Getting insight

Before any modelisation, the first step of any machine learning is , or at least, should be, data exploration.

Before Big Data and Machine Learning advent, most of data analysis where done visually with and data Insight were extracted by human from statistical indicator and charts.

Data Analysis is getting replaced with Artificial Intelligence and Machine Learning for understanding phenomena and building models but human mind is still great at getting insight from visual clue.

Exploring data may still help to build the target, decide modelisation type or find an innovative feature engineering. Sometimes it serves to detect underrepresented category and add some weigh to the the data.

Warning: Do not build segment for Machine Learning and AI problems ! Segmentation was a great way to build basic models but Big Data and Machine Learning tools do not need Segmentation anymore as they works on individual sample. Only use segmentation to build basic rules-based model or explore data and understand problem. But if the project need performance, use supervised learning of IA unsupervised technics.

Checking the data quality for modeling

Even if building a model only from data exploration is not the best way to get performance, data exploration can serve as a show stopper as it can highlight two main issue from your data before going on the modelisation step :

- random or noisy data
- unbalanced data

There are specific indicators for noisy data or signal/noise ratio and this can be seen from some specific visual representation. Looking after a too low signal/noise ratio is a good way to avoid poor modelisation due to poor data quality.

About unbalanced data, it's still possible to get good models if the low rate target has some very specific features , wich will probably appear in the data exploration process, but as a rule of thumbs, looking for general shape of data and some under-representation of sur-representation for planning some kind of weighting is considered as good practice.

Talking with the Line of Business

Most important output of running data exploration, especially with visual tools, is to onboard the line of business manager into the datascience project.

Success for a datascience project often rely on building the good target, that serves a true business purpose. By running a data exploration phase with someone from the business, you can, as Data Scientist Practitioner :

- get insight to build your metrics and objectives , and thus optimize the model for R.O.I
- get the Lob manager involved and build a relationship to build the fittest model for business.

Data exploration technic focus : Data Embedding

Data exploration often relies on the 3 following methods :

- build statistics for each feature (average, median, minimum value, number of occurrence, mode, ...)
- build charts (histogram, pie chart,...)
- build chart about some relation between features (bivariate analysis, correlation matrix,)

In Prevision.io platform, statistics and charts are produced on the dedicated features page

Home > [my project] > [my experiment] > Features

and

Home > [my project] > [my experiment] > Features > [my feature]

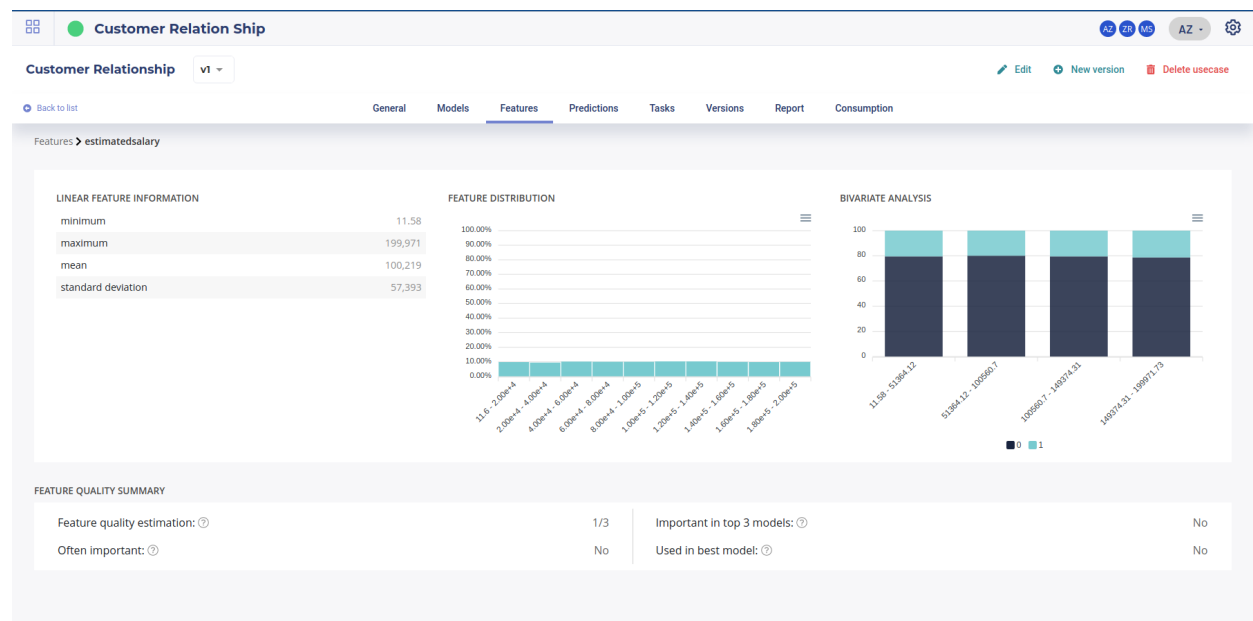


Fig. 75: Statistics available on experiment only (note the bivariate analysis related to the target)

or

Home > [my project] > [my dataset] > General

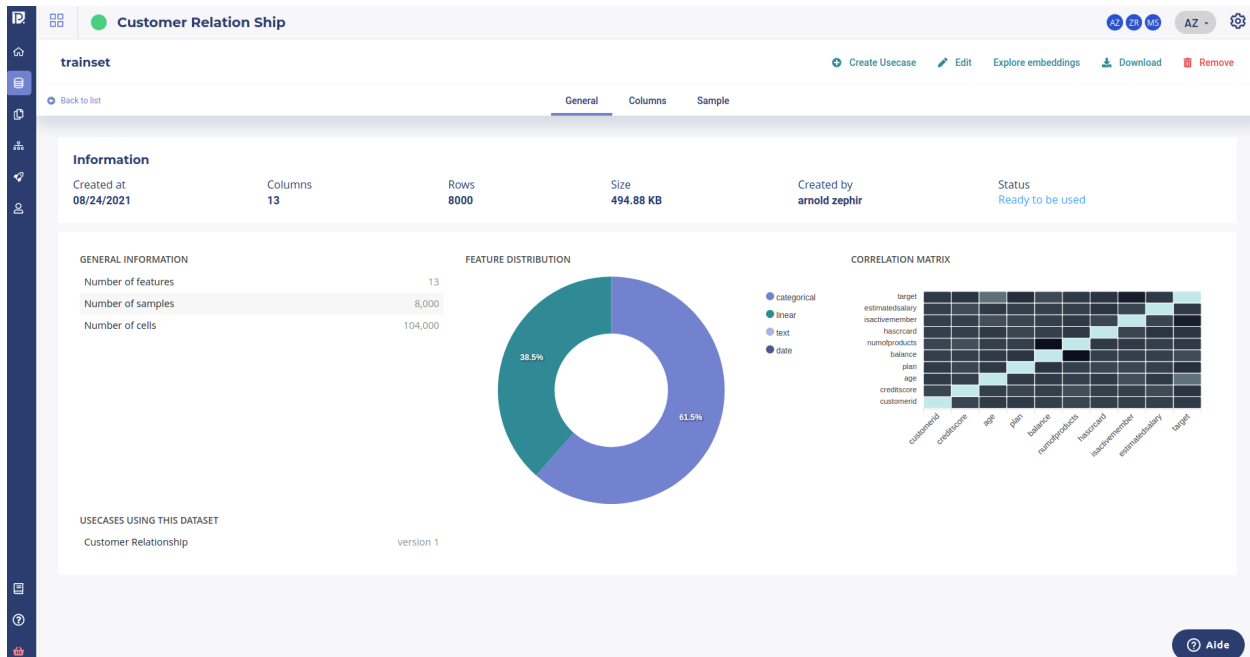


Fig. 76: Statistics available on each new dataset (automatic computing)

Note: Some statistics and chart are built only once the use case (see [Experiments](#)) has been defined because bivariate chart related to the target are built. Some others are available few seconds after you created new [Data](#)

Basic statistical indicators help to better understand the dataset yet more powerful technics and tools exist based on continuous vector built upon the data.

Embedding technics are various way to transform data such that you go from a discrete (categorical and such) representation of data to a continuous one with mathematical vectors. It is a very important method as it allow to run mathematical operations on all kind of data (cosine similarity, difference, addition) while preserving relationship between features. When embedding data, each sample of the dataset is transformed into a vector with fewer dimensions. This vector may be used to build chart, compute similarities between sample, cluster data or detect outliers.

Note: Four usage of Data Embedding :

- visualize cluster
- compute similarities between sample
- detect outliers
- visualize segment relative weights

There are many technics to build embedding but here are the most common

PCA

PCA (Principal Common Analysis) is based on Matrix eigen vector and eigen values. When applied on a dataset, it find eigen values and eigen vector of the data and resulting vectors can be interpreted as “axes of greater variance”. It often put emphasis on feature correlation and is used as a dimension reduction algorithm

Note: Let’s say that you got a dataset with 10 samples of 5 features

X1	X2	X3	X4	X5
fr	43	10	50	5
fr	43	4	20	3
en	13	7	35	3.5
en	12	20	100	10
en	12	34	170	17
en	13	18	90	9
fr	41	18	90	9
en	12	20	100	10
fr	43	32	160	16
fr	43	64	320	32

Even if there are 5 features, a PCA wil show that the samples are in fact variation of 2 vectors :

- a first one highly correlated with (X1,X2) features
- another one correlated with (X3,X4,X5)

Thus this 10 sample may in fact be written as two dimensionnal features :

V1	V2
0	0.1562
0	0.0625
1	0.1094
1	0.3125
1	0.5312
1	0.2813
0	0.2813
1	0.3125
0	0.5
0	1

Where :

- $X1 = \text{“fr” if } V1 == 0 \text{ else “en”}$
- $X2 = -30 * V1 + 43$

and

- $X3 = 64 * V1$
- $X4 = 320 * V1$
- $X5 = 32 * V1$

PCA may be used as some kind of feature importante

VAE

Variational Auto Encoding is a more powerful technic trying to compress data with less features than the original dataset. For example, if a dataset has 300 features but the compression algorithm can build a dataset with 30 features that is able to reconstruct the original dataset without losing too many signals, the theory says that the analysis can be done on the 30 features without loss of meaning or significance.

Variational Auto Encoding often uses a Neural Network that is tasked to generate to output the vector presented in input but with few neurons (for example, only 4).

The signal in the deep layer of this Network may be interpreted as a vector representation of the sample, called embedding, and has the interesting property that you can build a distance metric such that *similar samples have a small metric distance*.

Given this property you can build your analysis on the embedding space.

This is this technology that Prevision.io platform uses for data exploration

Data Embedding exploration in Prevision Platform

Prevision platform offers two features for data exploration :

- building the embedding
- a tool for exploring the embedding (“The Explorer”)

Building the Embedding

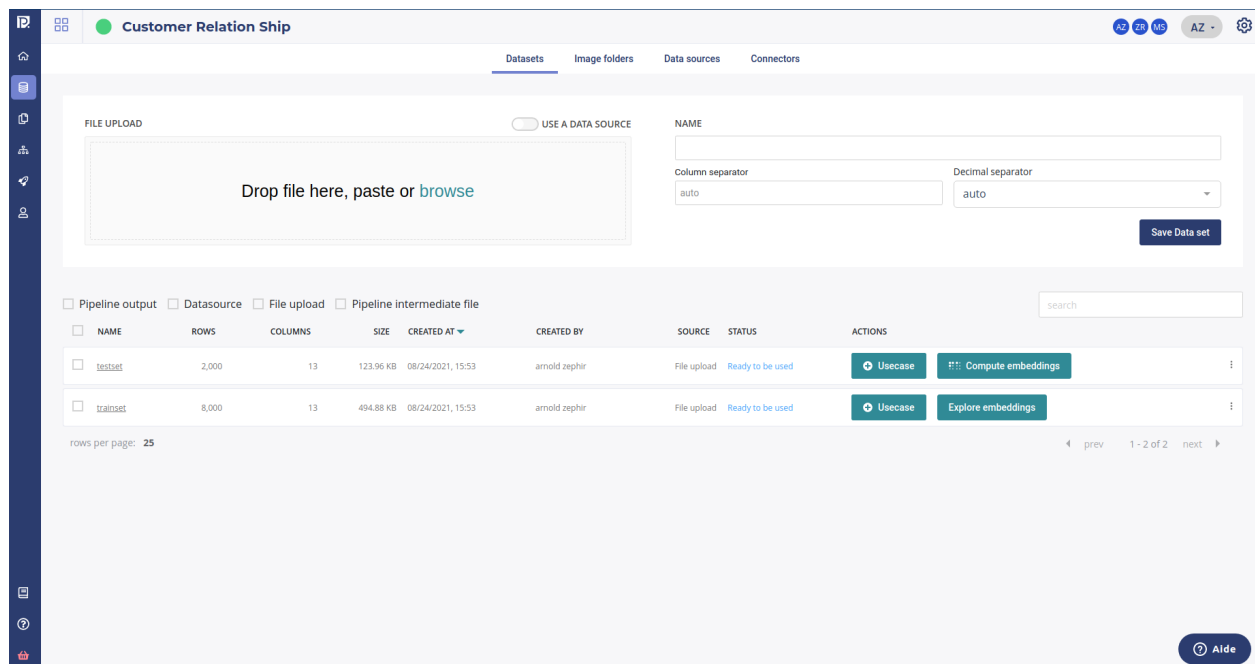


Fig. 77: The datascreen, where to launch an embedding

The “Compute Embeddings” button is available in the *Data* section of the platform. When you Upload or import a new dataset, it becomes available when the data importation is done.

As this is a cpu intensive algorithm, user must explicitly launch it by clicking on the button. It lasts about a few minutes and once done, you can explore your data with the “Explore Embeddings” button.

The Data Explorer

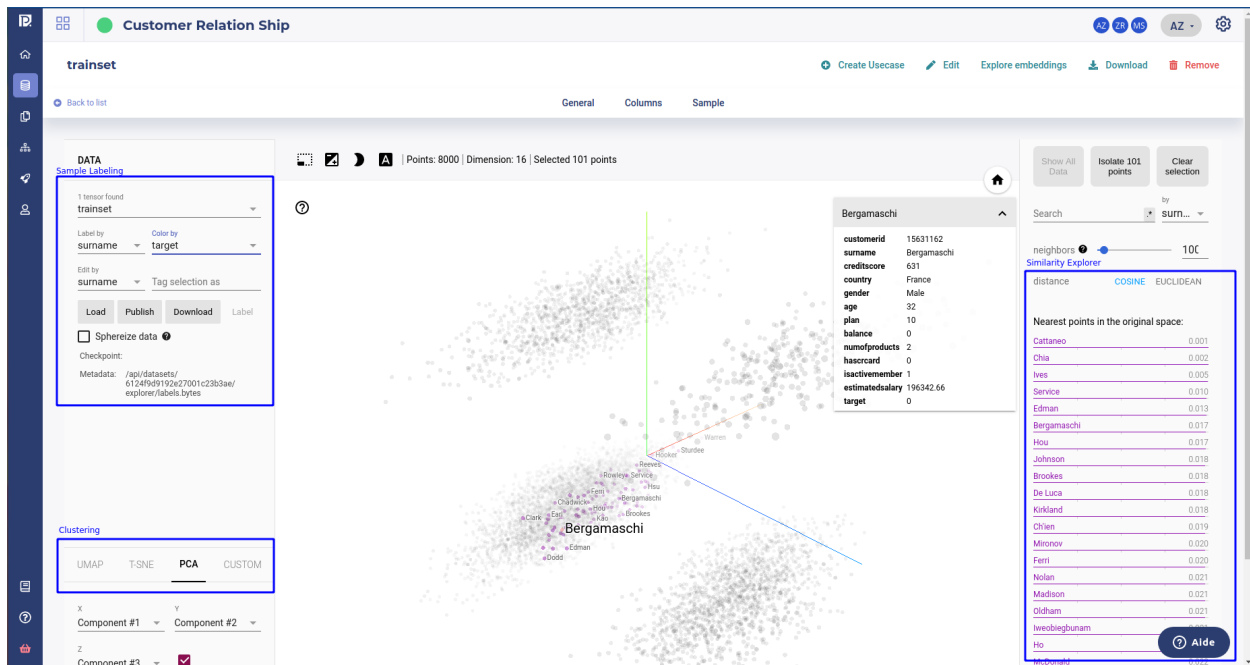


Fig. 78: The data explorer. Draw embedding vector on a 3D charts

The data explorer is an interface to handle data rows embedding, or at least a subsample (5000) of it. It projects the data onto a 3D or 2D vector space and give some tools for exploring data :

- the search and similarity sidebar, on the left, displays nearest neighbors for 2 metrics, cosine similarities and Euclidean distance, when clicking on a point. The number of nearest neighbors is a parameters that user can change. Note that you can isolate a point and its neighbors for further investigation
- the labeling box on the top left corner allows to assign labels and color to sample along some feature. You can display modalities (or values) of features as a label or as a color.
- the clustering box on the bottom left let user launch a clustering :
 - PCA : fastest but do not respect distance
 - TSNE : better but slow and complex to use
 - UMAP : less good than TSNE in respecting distance but fastest and a little bit easier to use

The central window displays a navigatio interface when you can pan, rotate and zoom.

Getting and Working with the Embedding Vector

Visualize cluster

To build cluster, and segment, you just need to launch a clustering computing by using the clustering section. There are 3 methods of 'clustering' :

- PCA
- TSNE
- UMAP

Each methods has its own set of parameters that build cluster more or less constrained.

Note: This are not methods of clustering *per se* as the clustering algorithms assigns a class (cluster number) to each point of the dataset. Here are algorithm that visually bring together data rows that share similarities (take whom distances are small). It allows visual inspection by a human mind, which is is often better to make generalities than algorithm.

PCA

PCA is a very cheap method but fast to compute. It sometimes highlights very simple variance axes and give some insight but do not expect much. The interesting thing in PCA algorithm is that it computes the explained variance for each component :

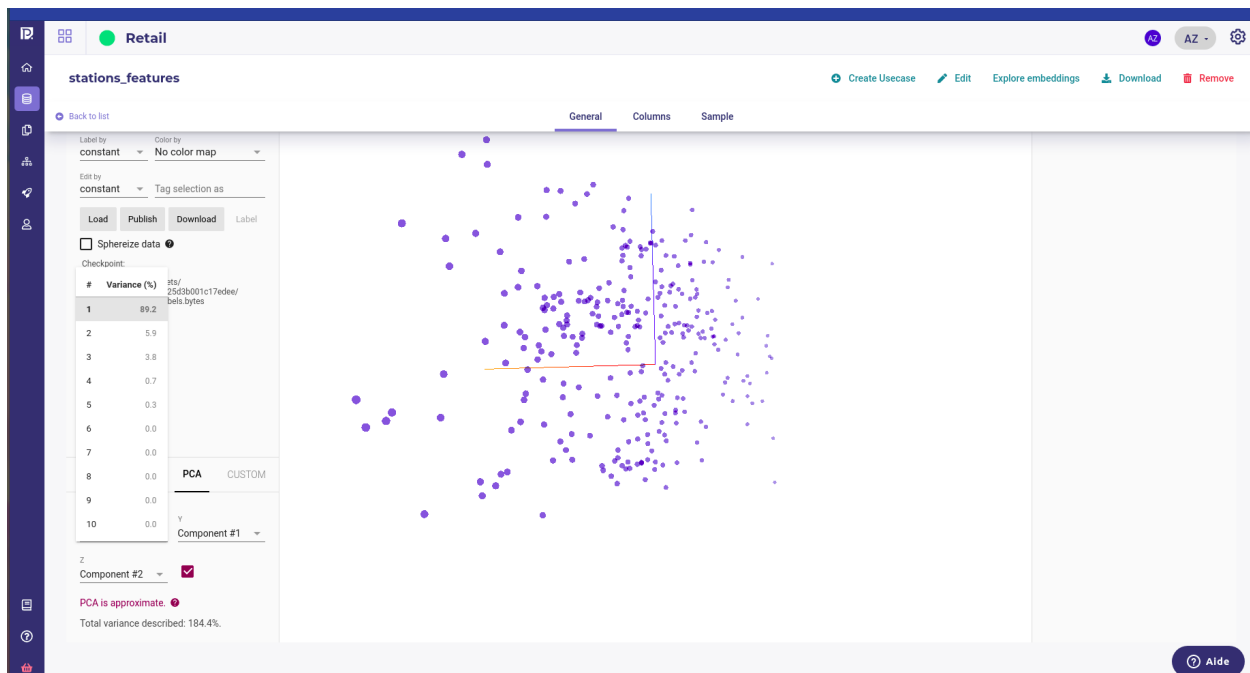


Fig. 79: PCA Component explained variance

Thus, you can quickly see if your data has in fact a low dimensionnality.

On the example above, we see that one component hold 89% of the data variance so even if the original dataset has 428 features, there seem to share a lot of information.

TSNE

TSNE is the most interesting clustering method as it can lead to very defined, and separated, segment when they exists. Yet it can be quite long to compute and hard to tune.

As a rule of thumb :

- the higher the perplexity, the more defined the shape but the longer to converge.
- with a smaller learning rate, convergence takes longer but point are well placed

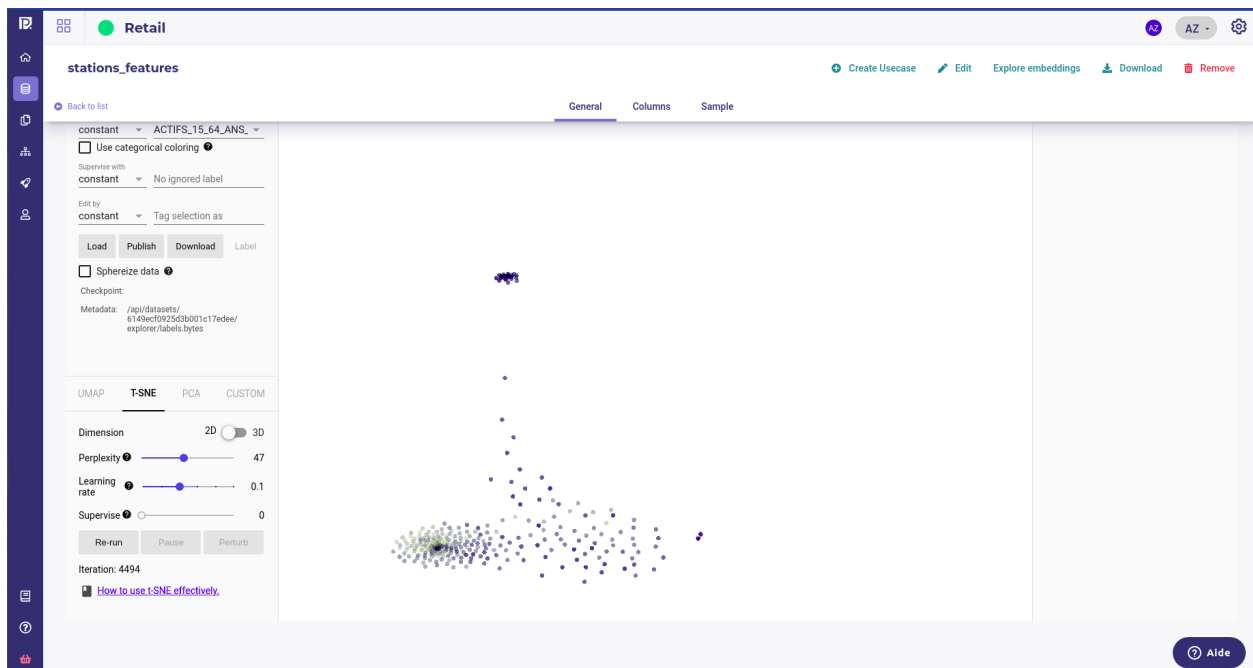


Fig. 80: Clear cluster of Paris Subway station (residential, job and tourism)

Note that you can constraint cluster to a feature, to force the algorithm to split the cluster along this features but do it only if you want to confirm some intuition.

UMAP

UMAP is way faster than TSNE and get good result to put most similar point together but is less able to generate well blocked shape. Yet, it's often good to use it to start exploration and then switch to tsne to validate some hypothesis.

UMAP has only one parameter, that is number of Neighbors. The more neighbors you allow, the larger and more inclusive are the structure of cluster.

Once you got some visual cluster, it's time to explore points and their similarities but let's talk about similarities, distance and proximity in 3D spaces.

Note: As we said before, the algorithms used by Prevision build vector space where you can build distances metrics such that similar sample have a small distance. Yet, in most of case, the list of similar points displayed on the right

will not always be grouped together on the central window after a clustering run because the visual explorer is only 2D or 3D. Similarities distance, cosine similarity or Euclidean distance, are computed on the whole dimension of the embedding space as UMAP and TSNE are computed so that similar points are near together in the 2D/3D space. Yet is not always possible to respect every constraints and sometimes, some points with a small distance will be far in the 3D space. When it happens, always keep the similarities distance as the truth.

Explore sample and look after similarities between samples

Whatever the representation you used, PCA, TSNE or UMap, you can always click on a point, which represent a row of your dataset.

When you click on a point :

- its features are displayed in a small dropdown windows
- a set of similar samples are highlighted and displayed on the similarities windows on the right sidebar.

In this window you can :

- select the number of neighbors displayed
- change the distance used, cosine (dot product) or euclidean distance
- search and highlight for specific value on specific feature (Note : you can use regexp for filters)

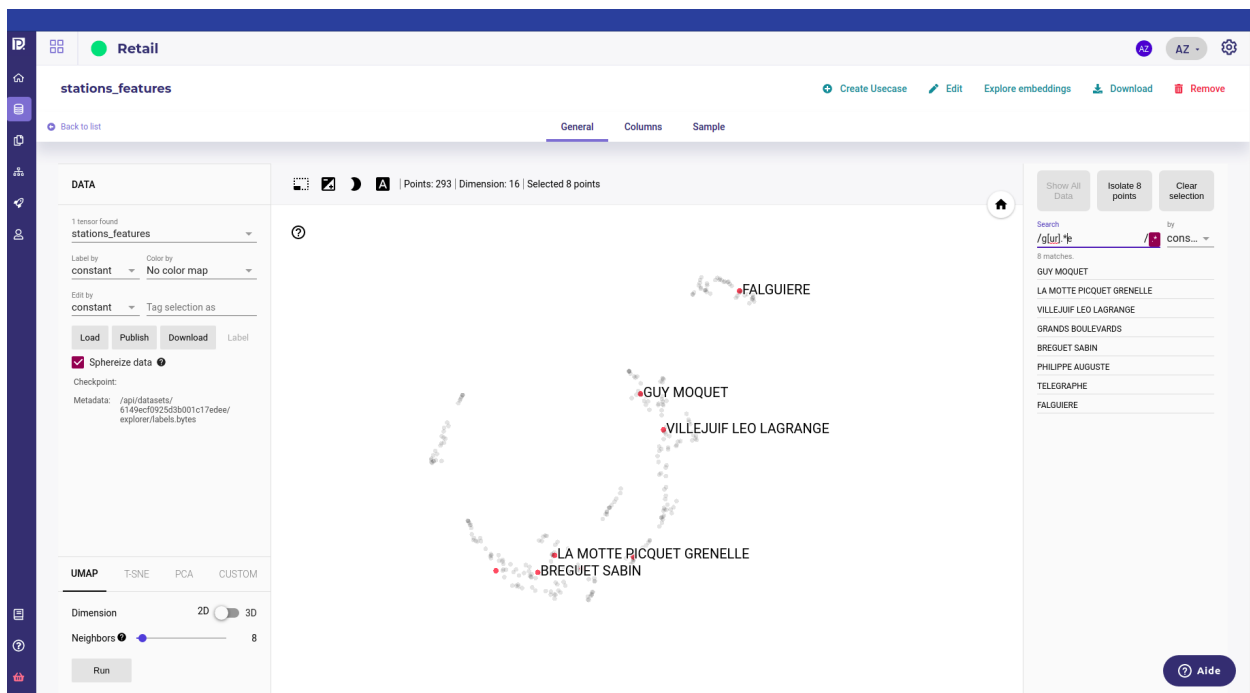


Fig. 81: filtering all the station whom name match `g[ur].*e`

- isolate for analysis all the highlighted points
- clear the selection

Once you have isolated some sample, the cluster then run only on this selection, allowing to target your analysis on a specific segment. Thanks to the sample labeling box on the left you can label and color your sample along a feature.

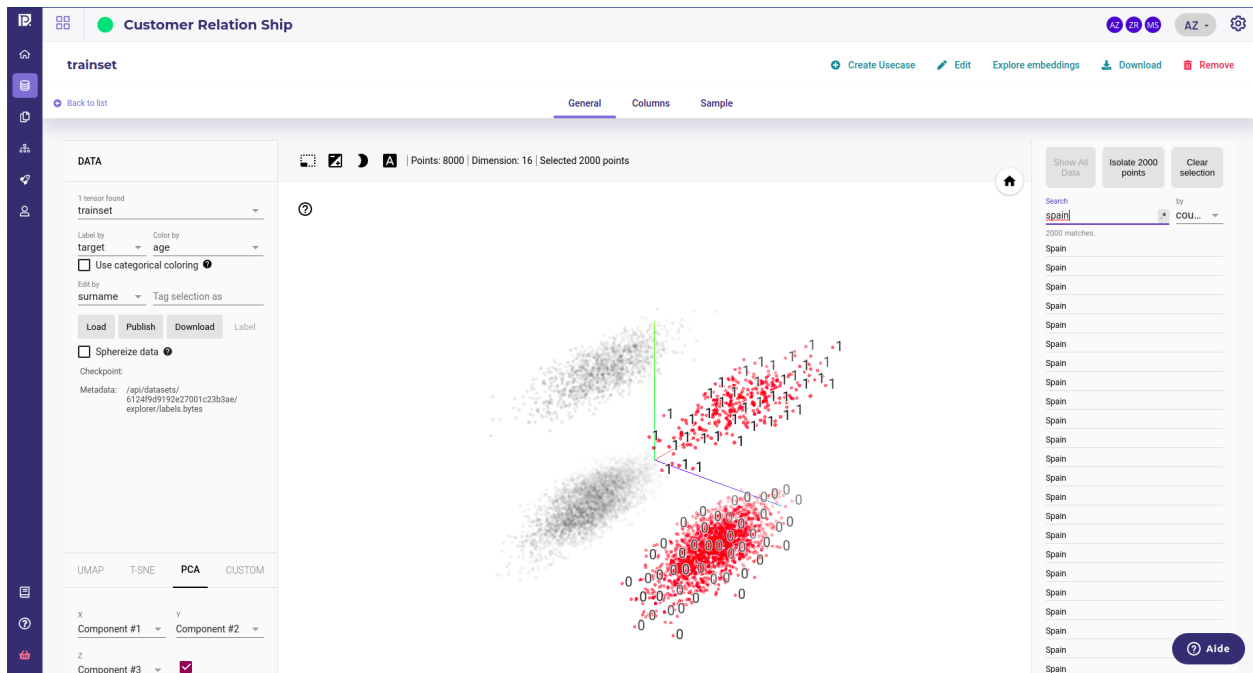


Fig. 82: In this dataset exploration, user has highlighted all the customers from Spain and siplay the target.

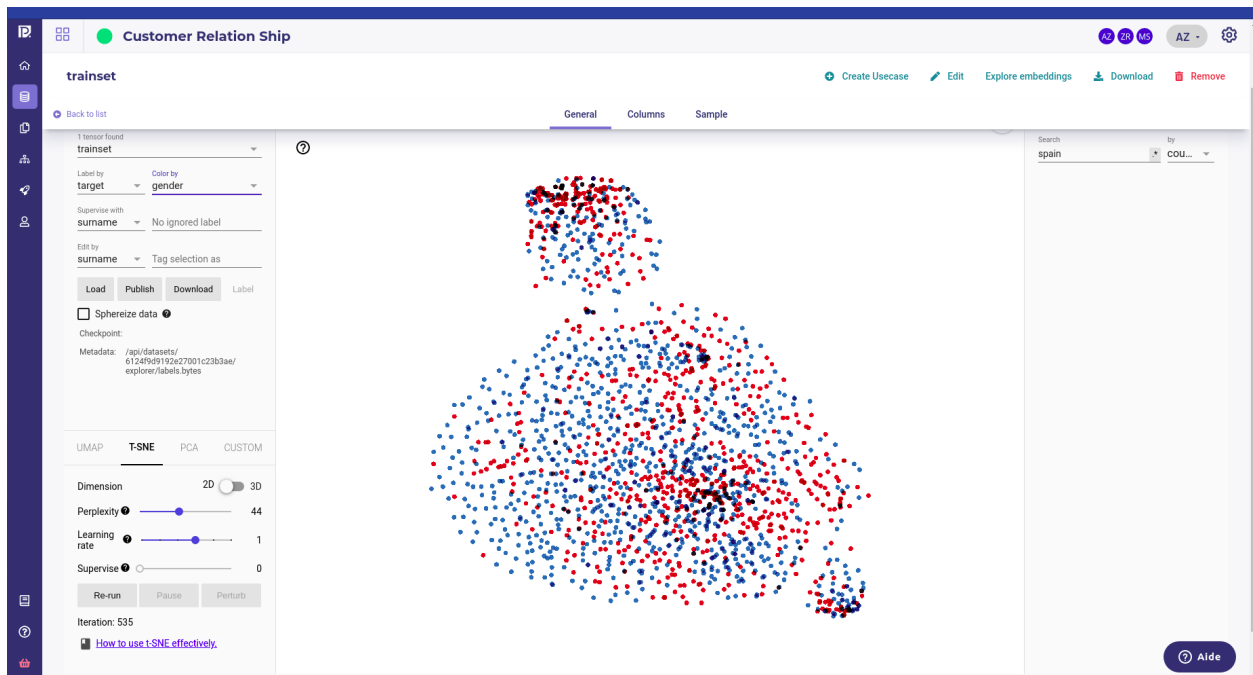


Fig. 83: The spanish user have been isolated and a cluster ran on this segment only. The gender is used has coloring showing that gender is not a main concer in churner split

Detect outliers

One of the positive byproduct of Embedding technics is that you can easily detect outliers visually. As the vector are build so that sample grouped by similar feature distribution and covariance, points that are visually outside any shape can be interpreted as outlier.

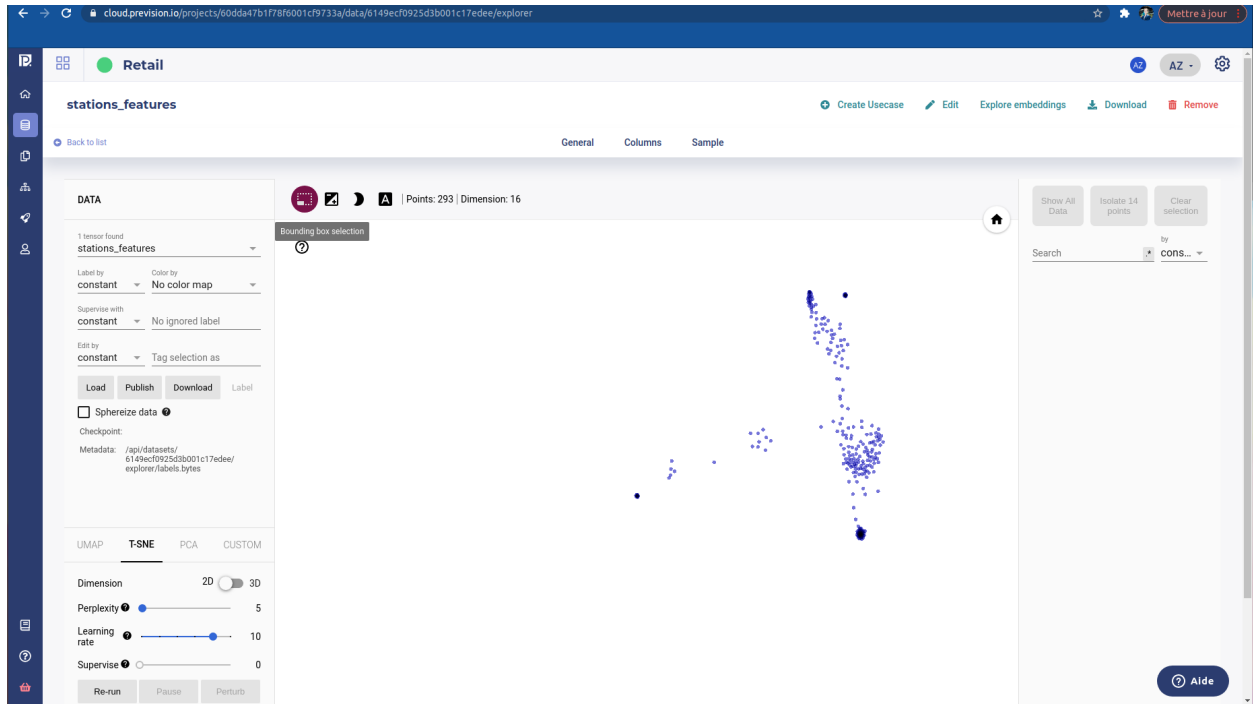


Fig. 84: A group of isolated outliers

Using a small perplexity for tsne isolate outliers. You can then click on any point of this outlier group to select it and its neighbors, if some, and then isolate them

Once isolated, you can color or select each point in order to understand why this samples stands out.

Going Further and use the API

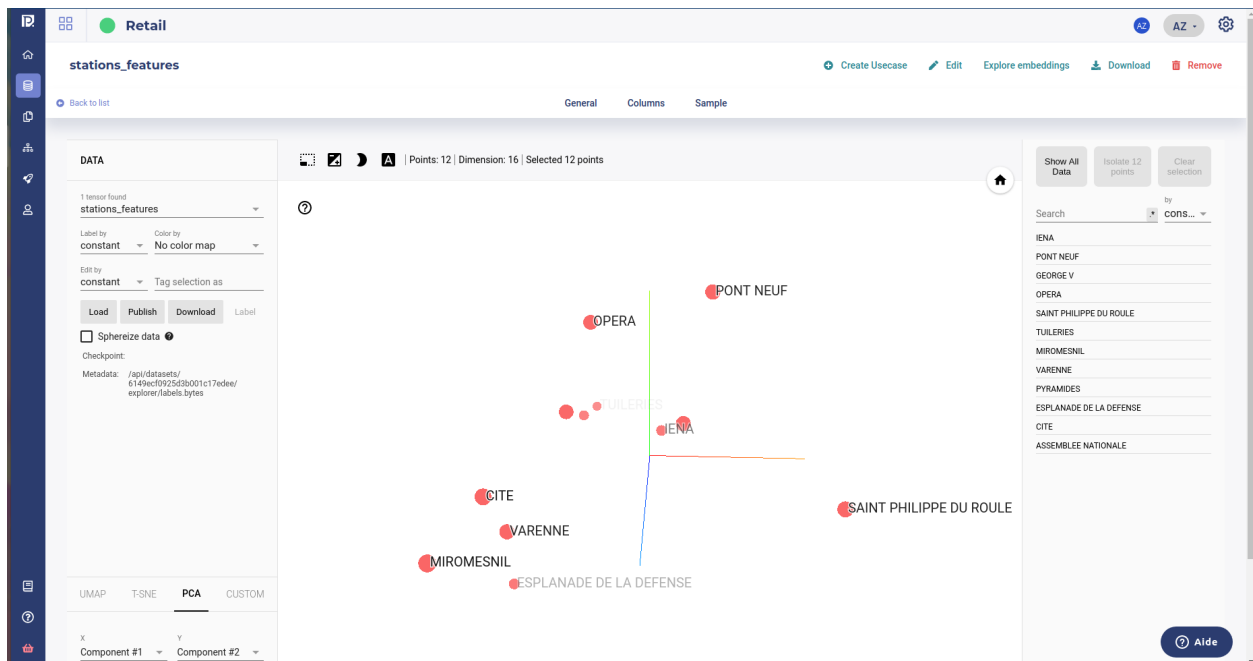
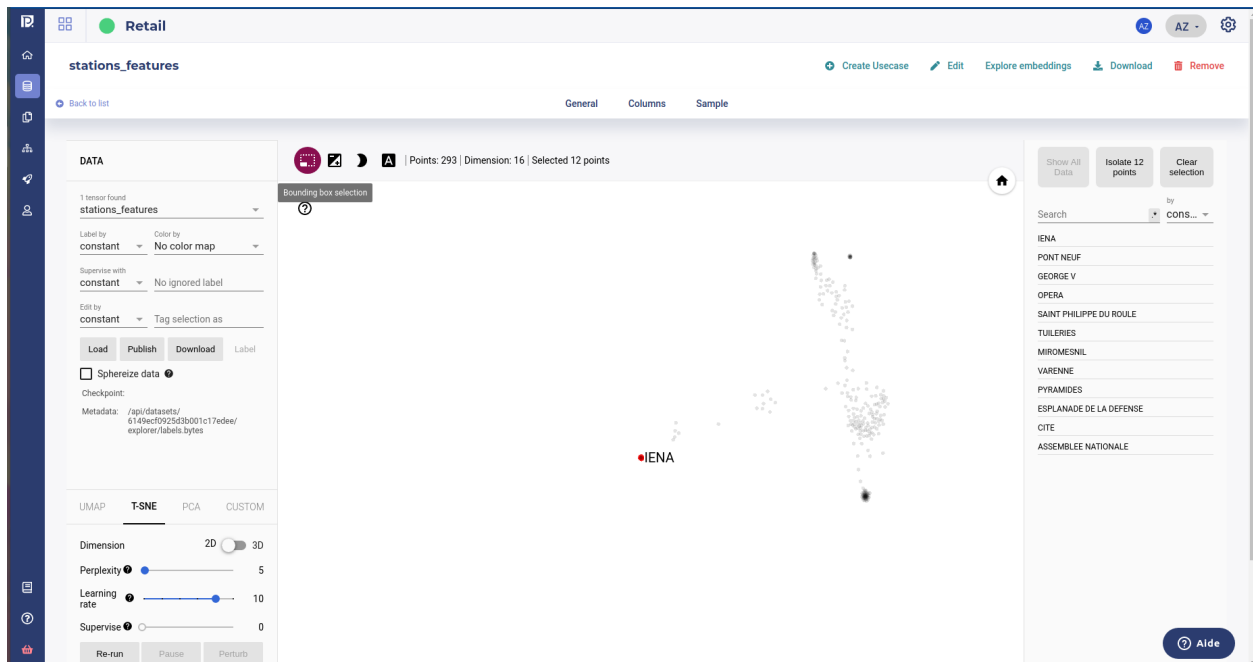
Like every feature of Prevision Studio, you can get the embedding over the [API](#).

The embedding are saved as numpy32 float and is an array. From them you can do many things :

- rerun your own clustering technics
- make histogram and stats along each axes in order to understand the meaning
- anything you want

To run the Code below you need :

- your Master Token. It is available in the API Key page of your user settings :
- The Id of your dataset embedding, wich is available in the url or in the



The screenshot shows the Prevision.io dashboard for a project named 'Retail'. The 'Overview' tab is active, displaying a table of datasets. The 'stations_features' dataset is highlighted. A dropdown menu for the user 'arnold zephir' is open, showing options for 'API key' and 'Document'. The 'API key' option is highlighted with a red box and a red arrow.

NAME	CREATED AT	ROWS	COLUMNS	SIZE	SOURCE	STATUS
stations_full_dataset_2017	09/21/2021, 16:32	306	6,673	11.19 MB	File upload	Ready to use
stations_features	09/21/2021, 16:32	293	429	1.99 MB	File upload	Ready to use

NAME	DESCRIPTION	NODES	CREATED AT	CREATED BY	USED IN RUN
retail_demo		2	08/31/2021, 15:41	arnold zephir	false

NAME	VERSION	CREATED AT	DATA TYPE	TRAINING TYPE	SCORE	MODELS	PREDICTIONS	STATUS
No data for table								

The screenshot shows the 'stations_features' dataset explorer in the Prevision.io interface. The 'General' tab is active, displaying a scatter plot of data points. The 'Metadata' field is highlighted with a red box and a red arrow, showing the value '6149ecf0925d3b001c17dede'.

1 tensor found
stations_features

Label by: constant | Color by: No color map

Edit by: constant | Tag selection as:

Load | Publish | Download | Label

☐ Spheroize data

Checkpoint:

Metadata: 6149ecf0925d3b001c17dede

UMAP | T-SNE | PCA | CUSTOM

X: Component #1 | Y: Component #2

Z: Component #3

You can use the native urllib module to parse API but you need pandas and numpy to use data.

First, import native python 3 urllib.request and set up your Token and url built from dataset id (warning : if you have an on promise server or custom dedicated domain, you need to replace the url “cloud.prevision.io” with your own)

```

1 import urllib.request
2 import numpy as np
3 from io import BytesIO
4 import pandas as pd
5 import ssl
6 import json
7
8 MASTER_TOKEN="<YOUR_MASTER_TOKEN>"
9
10 BASE_PATH = "https://cloud.prevision.io/ext/v1/datasets"
11 DATASET_ID="<dataset_id>"
12
13 meta_url      =f"{BASE_PATH}/{DATASET_ID}/explorer"
14 labels_url    =f"{meta_url}/labels.bytes"
15 dataset_url   = f"{meta_url}/tensors.bytes"

```

Then, get the meta information of the embedding, especially the shape of the tensors generated

```

1 # Meta info
2 request = urllib.request.Request(meta_url)
3 request.add_header('Authorization',MASTER_TOKEN )
4 # Disable SSL check
5 meta = urllib.request.urlopen(request, context=ssl.SSLContext()).read()
6 meta = json.loads(meta)
7 tensorShape = meta["embeddings"][0]["tensorShape"]

```

If needed for readability, you can get the originals labels

```

1 # Labels
2 request = urllib.request.Request(labels_url)
3 request.add_header('Authorization',MASTER_TOKEN )
4 # Disable SSL check
5 labels = urllib.request.urlopen(request, context=ssl.SSLContext()).read()
6 labels = pd.read_csv(BytesIO(labels), sep="\t")

```

And last, get you embeddings, that are float32, and reshape them according to the meta information you got before.

```

1 # Tensors
2 request = urllib.request.Request(dataset_url)
3 request.add_header('Authorization',MASTER_TOKEN )
4 # Disable SSL check
5 vec = urllib.request.urlopen(request, context=ssl.SSLContext()).read()
6 vec = np.frombuffer(BytesIO(vec).read(), dtype="float32")
7 vec = vec.reshape(tensorShape[0],tensorShape[1])
8
9
10 df = labels.join(pd.DataFrame(vec))

```

Hence, you have a dataframe containing your original data and their embedding value, that may be use for any operation.

Full ML pipeline: From data collection to deploying using Prevision.io

How to release a model across all your organisation in one morning (and stop spending 24 man-month on a model that will never go into production) ?

What is this Guide about ?

This guide is walkthrough for delivering (very) quickly a complete Machine Learning Project by using the [Prevision.io platform](#)

The guide details each standard step of a Machine Learning project, from data to model usage across the organisation, and shows how to accomplish them in the platform.

We use historical sales data and intend to build a sales forecasting model.

What's in this guide ?

Starting point

This guide assumes that :

- you got a [Prevision.io platform](#)
- the IT Teams put some historical sales data in a database and gave you access (but if not, csv files are provided below for the sake of this guide)
- An objective has been defined by the Line of Business.

Steps

The steps of our guide will be :

Table 11: Steps of a Machine Learning Project

Step	Name	Goal	LoB	IT	Data lab	Output	time spent
1	Data acquisition	Get the historical data for training Machine Learning Model	No	Yes	Yes	dataset	5mn
2	Feature engineering	Prepare the dataset	No	No	Yes	Holdout and validation strategy (fold)	20mn
3	Define the problem	Define a metrics that reflects LoB process	Yes	No	Yes	A consensus	As Much as possible
4	Experiment	Train models	No	No	Yes	~100 Models	25mn
5	Evaluate	Get the fittest model	Yes	No	Yes	A selection of 3 to 4 models that go in production	As Much as possible
6	Deploy	Share the model accross the organisation	No	No	Yes	Webapp for human, API for machine	5mn
7	Schedule	Schedule predictions	Yes	Yes	Yes	Prediction delivered each Monday a 9:00 am in CRM software	20mn
8	Monitor	Track the model in situ	Yes	No	Yes	Dashboard	As Much as possible
TO-TAL							1h15

For each of them, the guide explains what to expect from this steps and how we accomplish it.

Results

At the end of this guide :

- LoB will get a weekly sales forecast each Monday at 09:00 AM
- LoB will get a simulator for testing hypothesis over the model
- Applicative team will get an API for calling the model in their own Application
- IT Team will get a dashboard to monitor model Quality of Service

So what's the job of a datascientist anyway ?

Datascientist is not a developer, even if her or his main tools is code.

Datascientist salary (and scarcity) are quite high. If your datascientist spends most time coding models, you should get a developer.

The time spent of a datascientist in an organisation should be spent with the Business owner on the 3th and 5th steps :

- DEFINE METRICS THAT REFLECTS A TRUE BUSINESS ISSUE
- CHOOSE THE MODEL THAT SOLVES THE PROBLEM THE BEST

This are the two most important points for a [project to deliver R.O.I](#) and using tools enables to focus on what matter.

You can [open a free account](#) to practice the following steps. When your account is ready, create a Project to host the assets

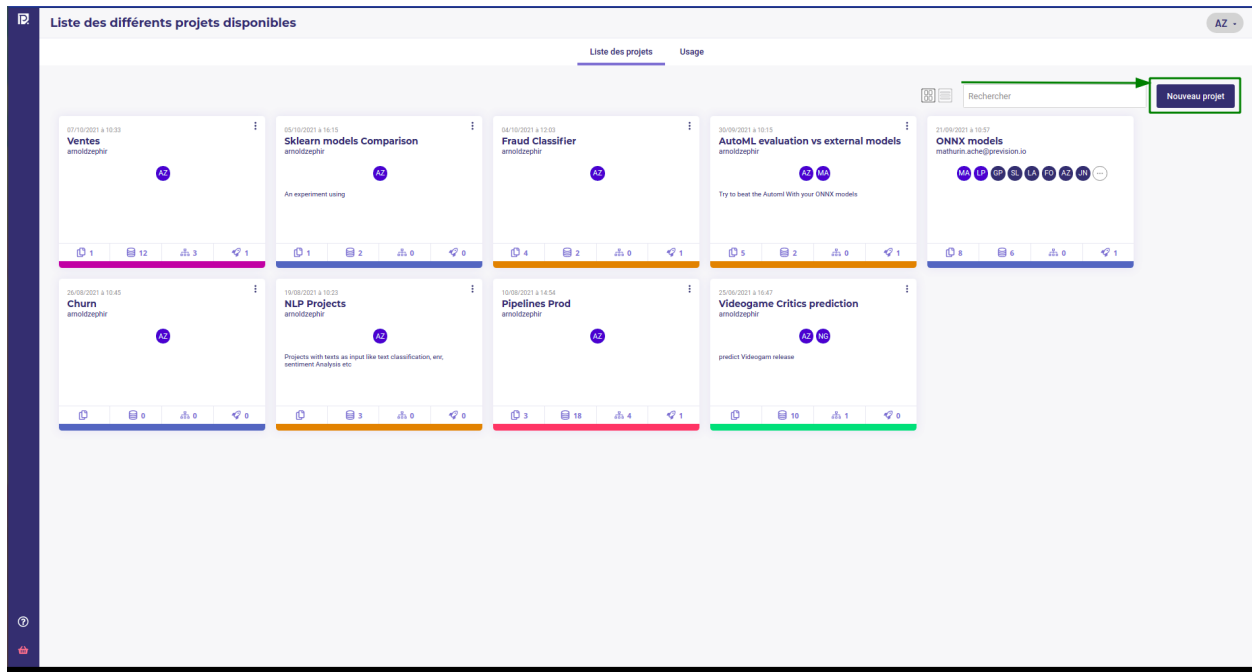


Fig. 85: Create a new project

Data acquisition

First step to any project is getting historical data in order to train our Algorithm. As the name implies, Machine learning is all about reading historical data and let a computer model learns to predict a target.

The datas should have been loaded into a database by the IT Team and they have generated credentials for you. Once you have created your project, and selected it :

- go to the data section (sidebar on the left)
- create a new connector and provided the credentials
- create a new datasource from the db and table of past sales
- Import it as a dataset

If available, you could import recent sales as an holdout dataset in order to validate and check stability of your model :

Data acquisition is done, you can now start to model

Table 12: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn

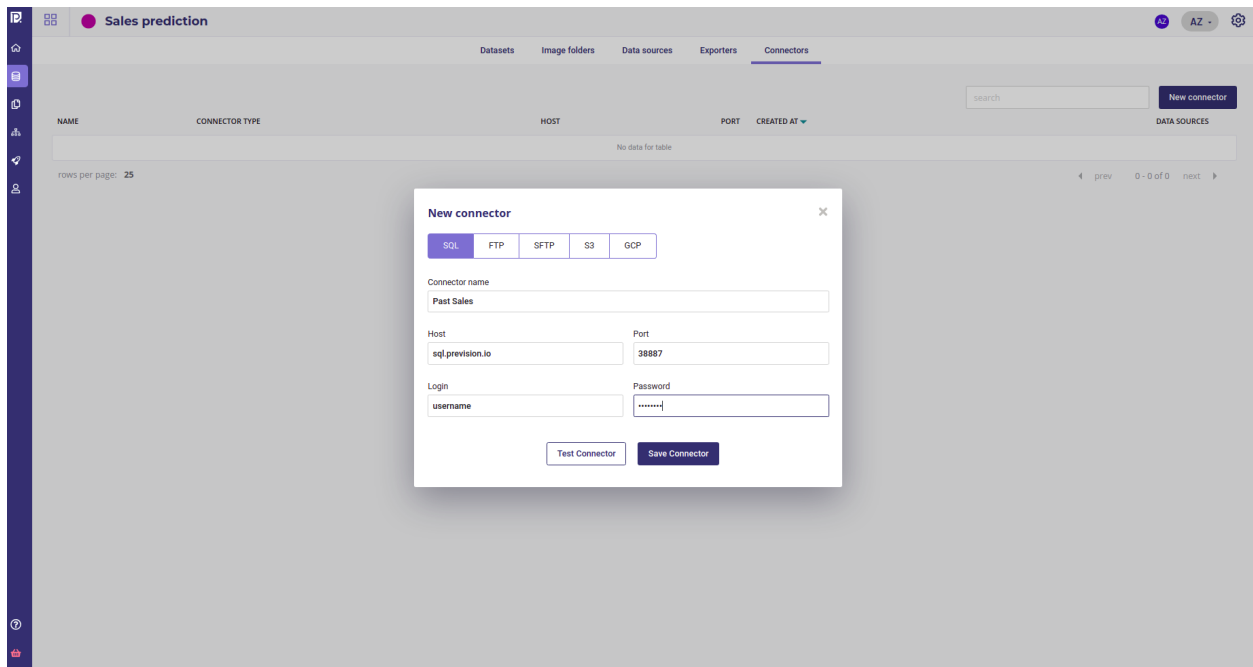


Fig. 86: Create a new connector

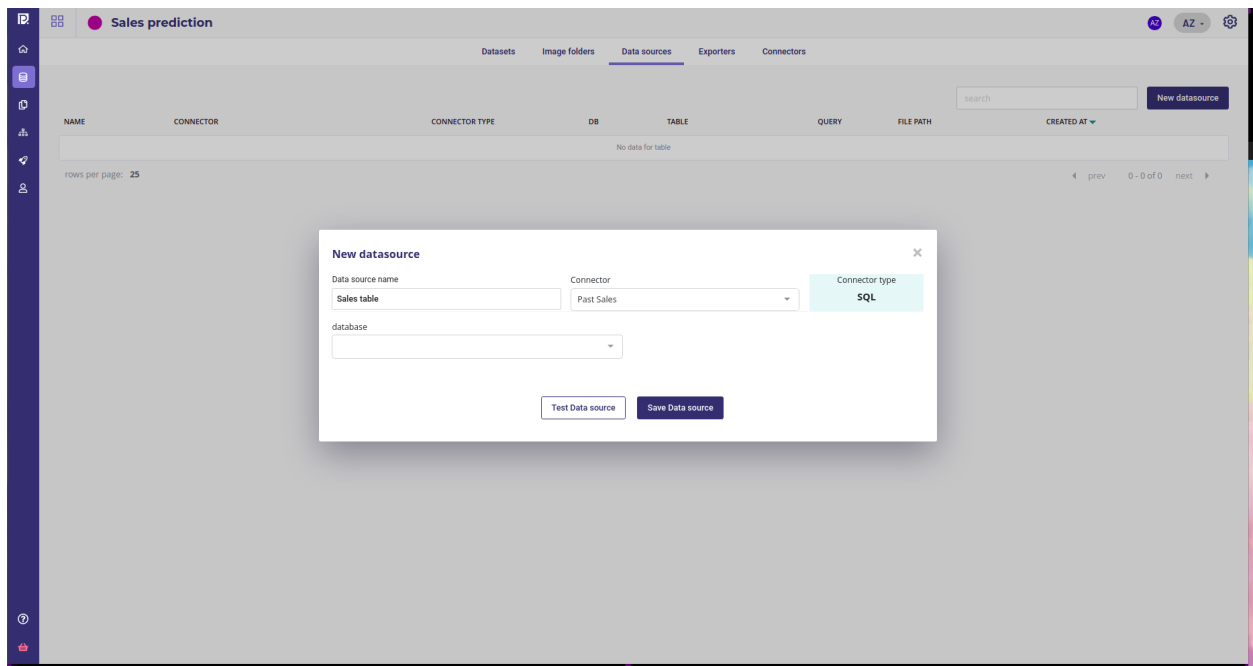


Fig. 87: Create a new datasource

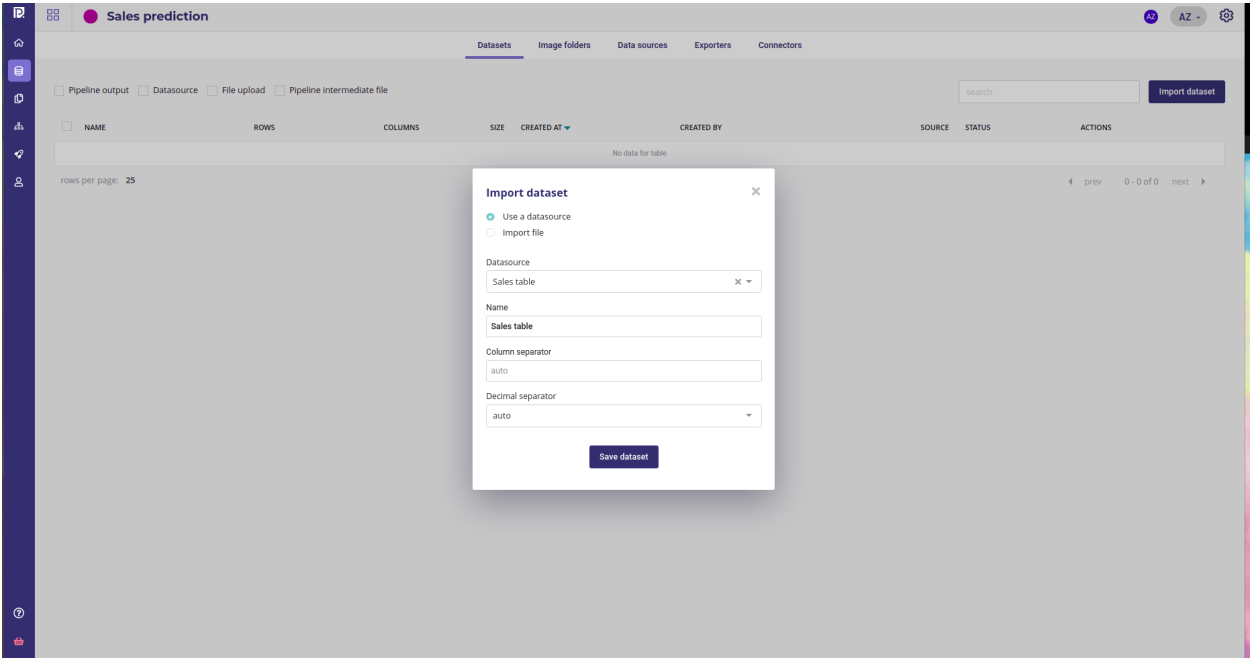


Fig. 88: Import dataset

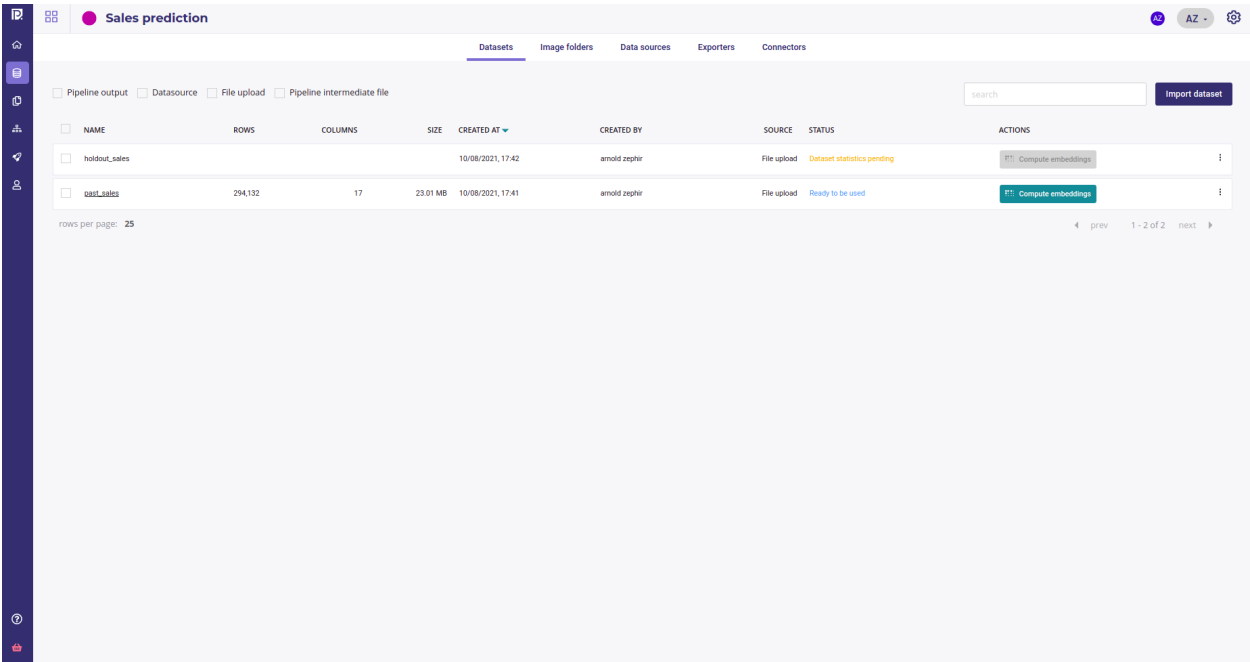


Fig. 89: You got two datasets.

Download data for testing by yourself

if you don't have database credentials, you can use the following files. Just import file instead of using a datasource when importing dataset.

- the trainset.
- the holdout.

Feature engineering

Feature engineering is the addition or transformation of one or more features to create new features from the original dataset. In Prevision Platform, and most of the modern tools, feature engineering are done with *components and pipelines* yet in most of case you don't need to add features as the AutoML engine makes all of the standard feature engineering by itself.

Here we are going to add a fold column on the date features in order to properly evaluate our model stability. A specific component has been developed by the datascience team [starting from the Prevision Boilerplate](#) and pushed on a private repo.

The component may now be integrated into the component library of the project.

Go to the pipelines section of your project and under the Pipeline Components tab, click **New pipeline Component**

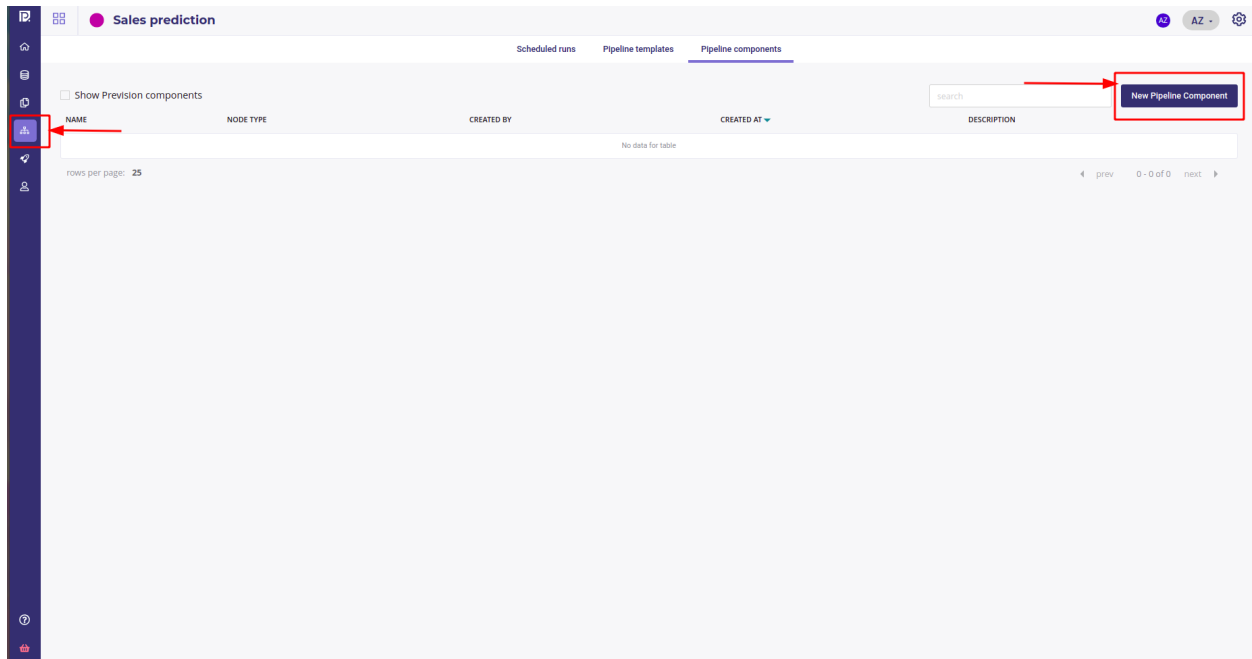


Fig. 90: Create a new component

And select your repo and branch

Once the component is built, its status will be ok and we can use it in a *pipeline*. Create a new pipeline template with three nodes :

- an import dataset, to read the trainset
- the newly created component (“build fold”)
- a **save dataset** node to save the feature engineered dataset into you Data

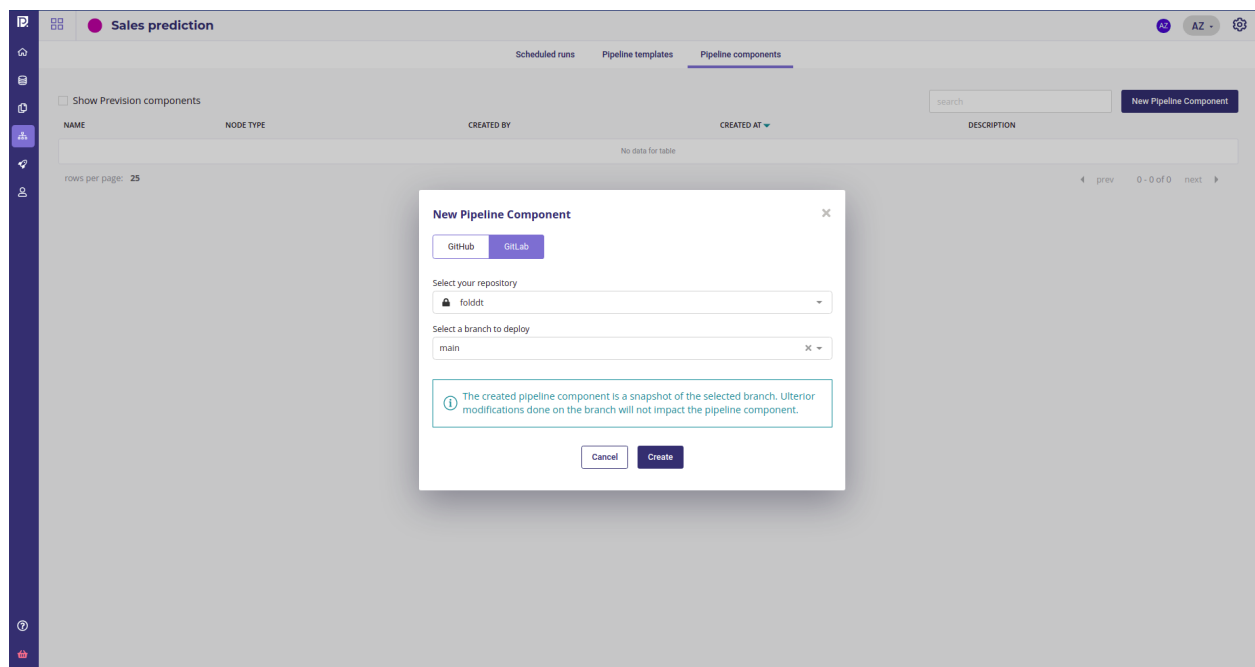


Fig. 91: Import component from your repo

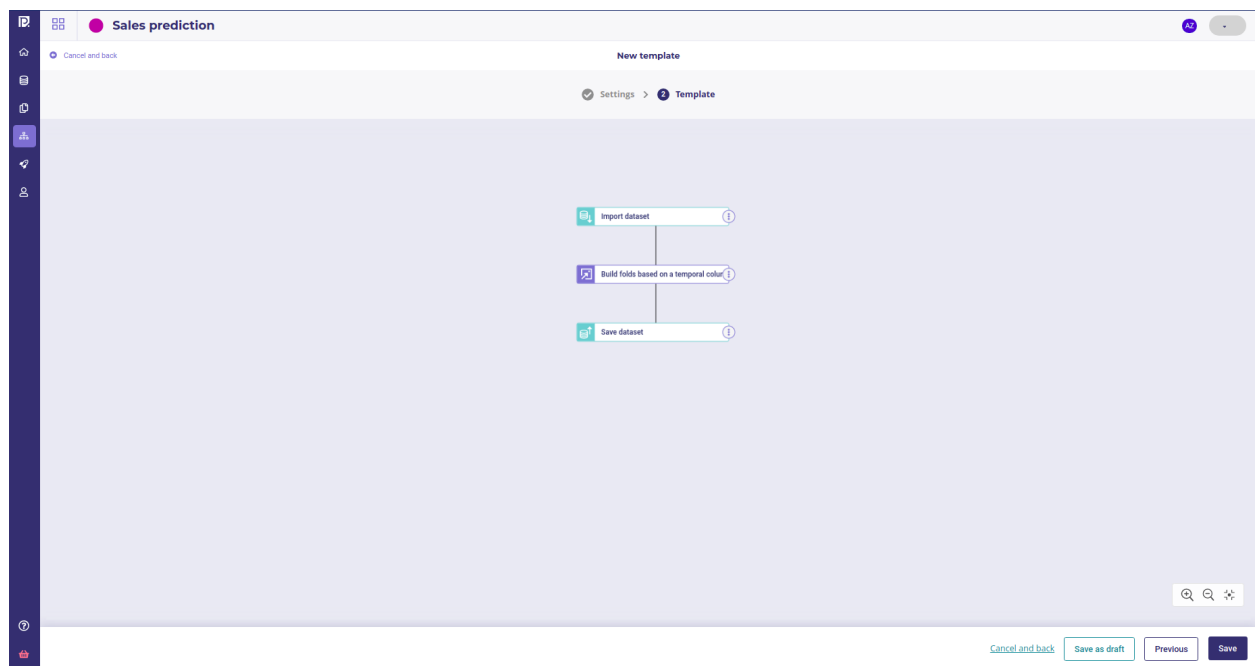


Fig. 92: A simple feature engineering pipeline

Then create a new *schedule run* that you gonna execute manually once on your trainset.

Fig. 93: Create a new scheduled run

Once you did the configuration, select “Manual” as the trigger and run your Schedule run. In a few seconds, a new dataset should be available in your data section as a **pipeline output** with a new **fold** column

You now got a dataset with features for training model and an holdout to validate your models.

Table 13: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn

Build your own

For the sake of this guide, we built a very basic feature engineering pipeline but you can add as many transformation as you want and build very complex pipeline.

Here we only have one component that adds a fold column, which is the year modulo 4. You can make the feature engineering on your local machine with the following code. Yet, if you want to build your own component you can follow [this guide](#) or some others

```

1 def addfold(df: pd.DataFrame, dtcol: str="dt", foldon:str="year", nfolds:int=3) -> pd.
  ↳ DataFrame:
2     if nfolds <=0 :
3         nfolds=3

```

(continues on next page)

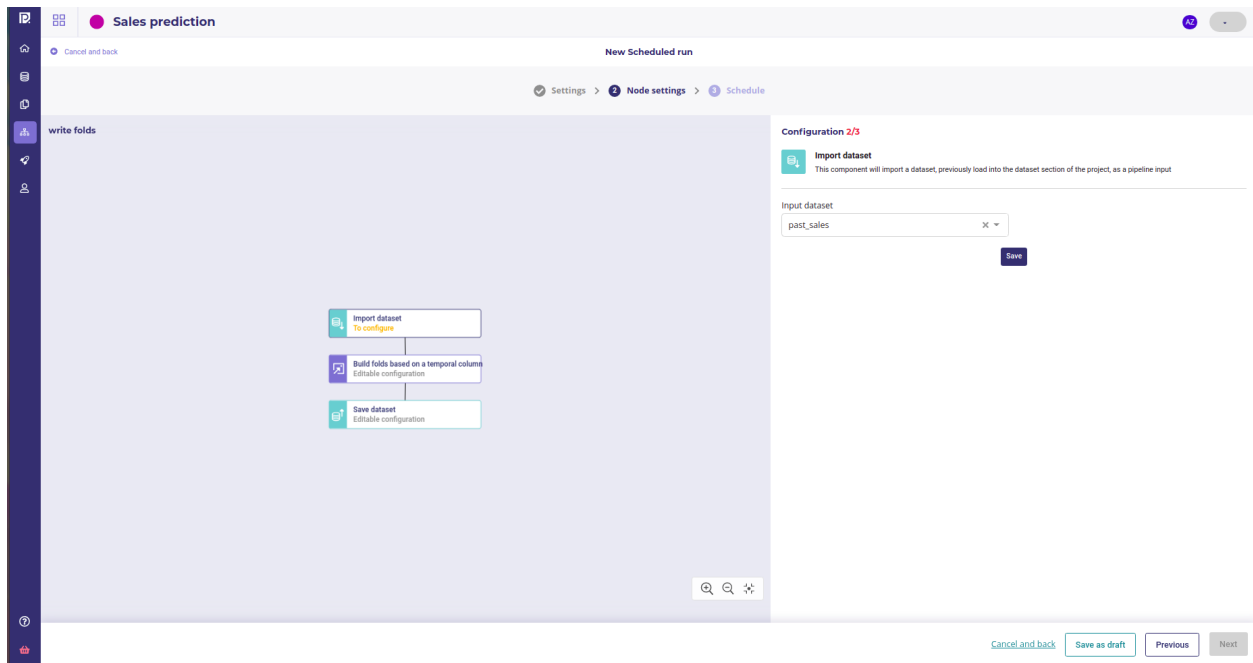


Fig. 94: Set your trainset as the input dataset

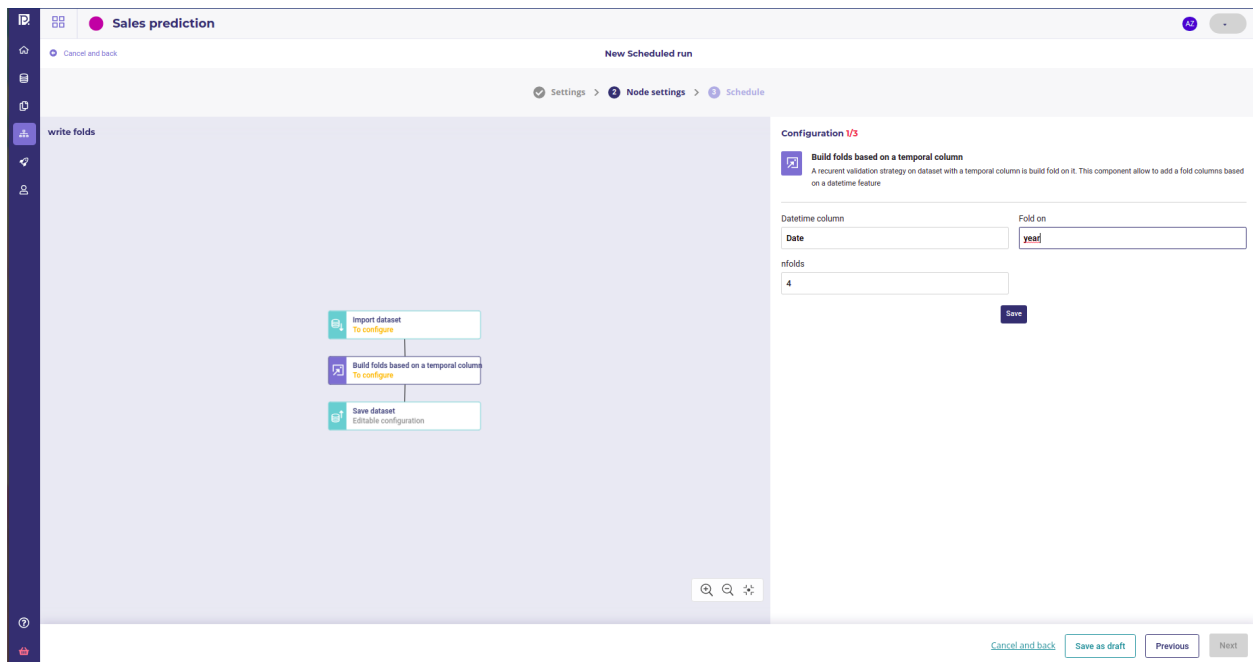


Fig. 95: configure your fold component parameters

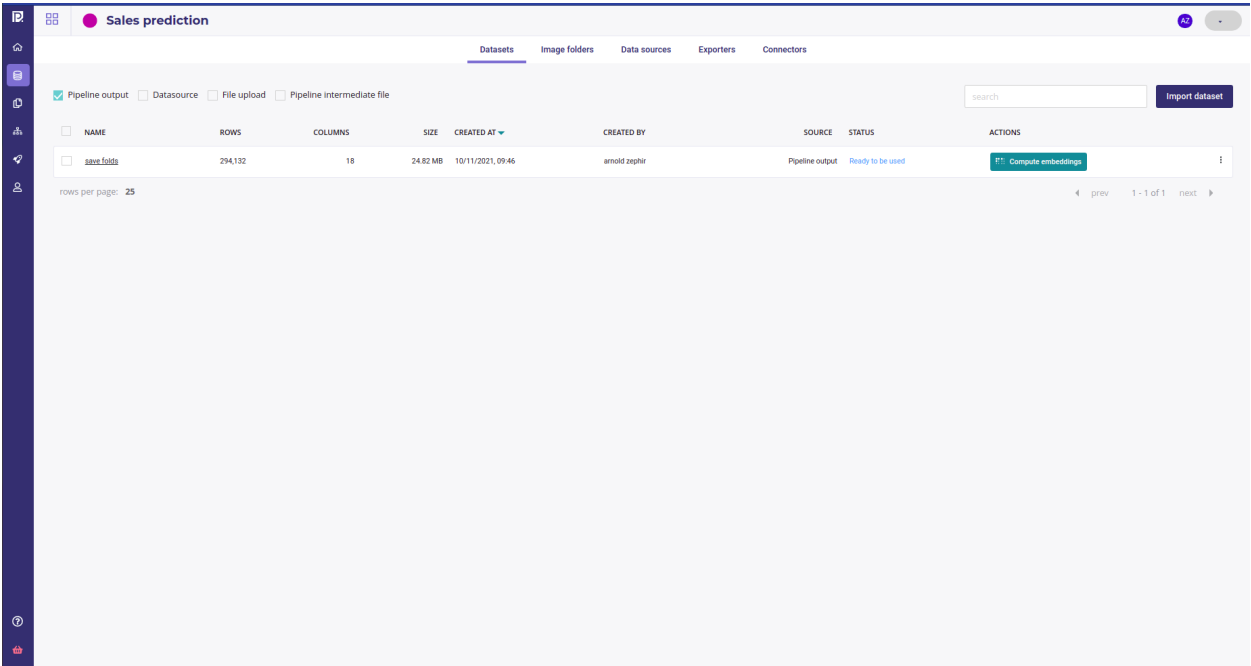


Fig. 96: Pipeline output dataset

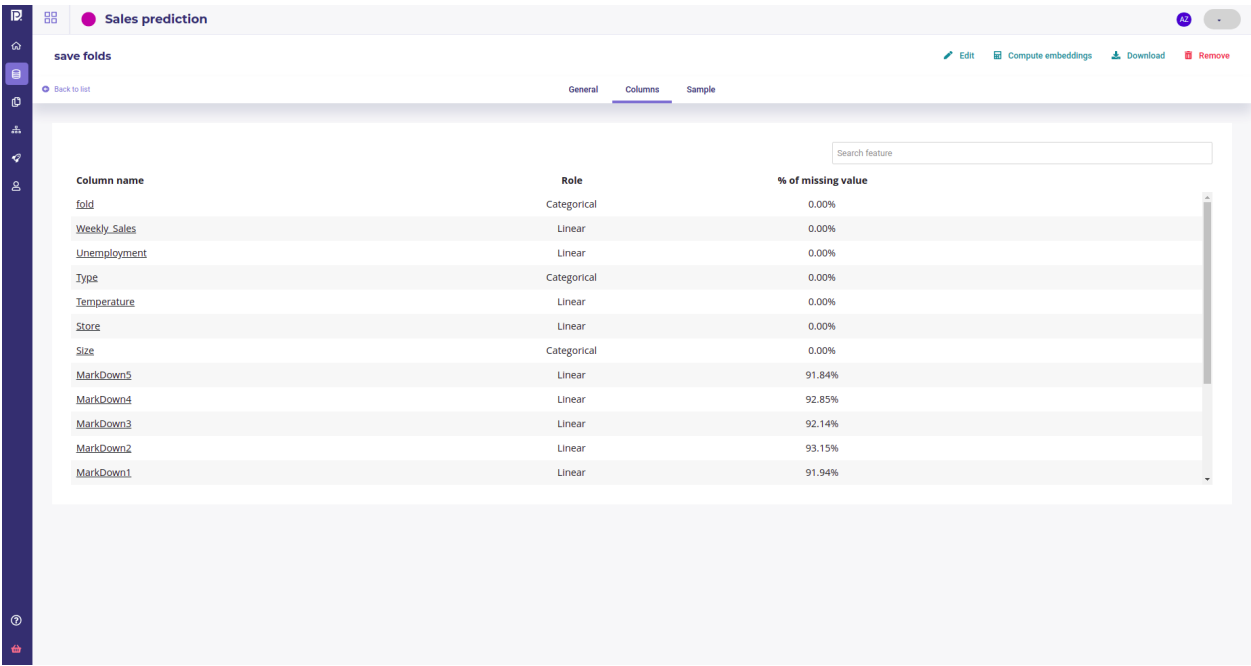


Fig. 97: Pipeline output dataset

(continued from previous page)

```

4 df[dtcol] = pd.to_datetime(df[dtcol])
5 df["fold"] = df[dtcol].dt.month % nolds
6
7
8 if foldon=="year" :
9     df["fold"] = df[dtcol].dt.year % nolds
10 if foldon=="day" :
11     df["fold"] = df[dtcol].dt.day % nolds
12 if foldon=="hour" :
13     df["fold"] = df[dtcol].dt.hour % nolds
14 return df

```

Define the problem

This is the most important part and the one that should be allocated the most time.

In this step, you're going to define with the Line of business how to qualify the project as a success and you, as a datascientist, are gonna to translate this as datascience metrics.

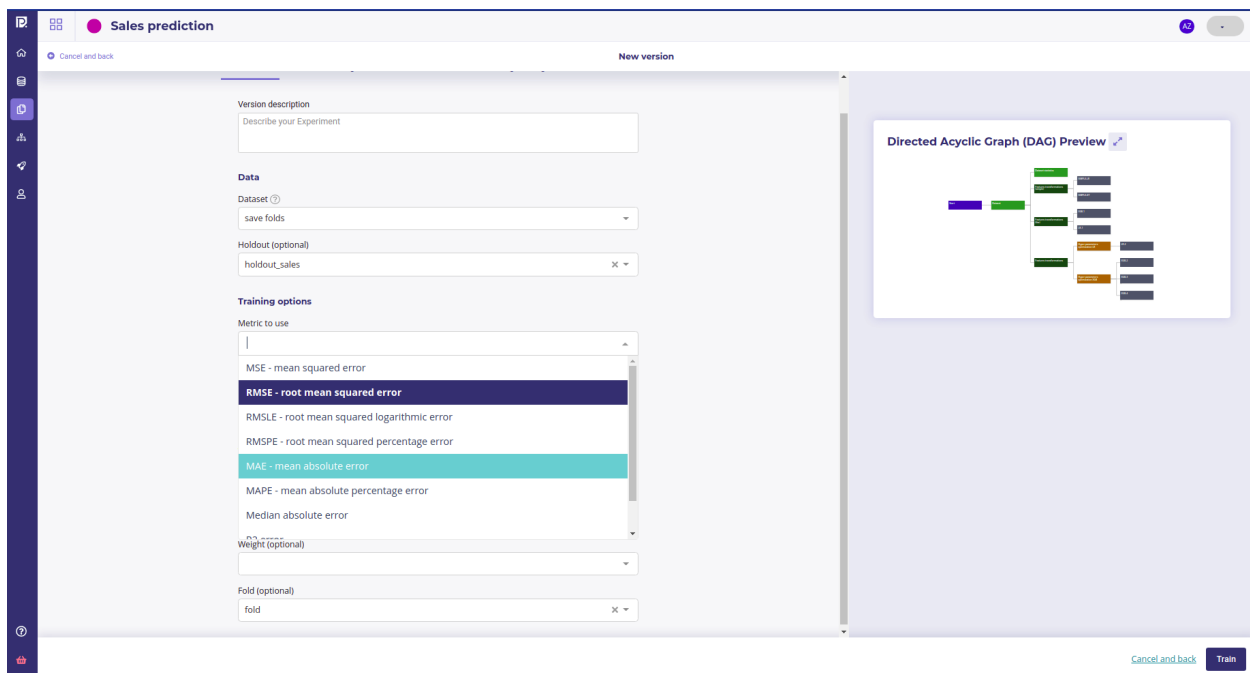


Fig. 98: Regression metrics

Choosing the best metrics is out of the scope of this document but you must spend time with your business teams and ask this kind of question :

- Imagine that I have the perfect model, does it make me gain something ?
- how many money do I lost if I forecast 110 sales instead of 100 ?
- how many money do I lost if I forecast 90 sales instead of 100 ?
- are all the predicted product equal ?

- Should I forecast the total number of item sold, the total amount of sales (in €), the total weight of my items or the total volume ?
- How much time before should I forecast ?
- etc ...

As a datascientist, by using an AutoML platform, your role is not to code in python or make dockerfile but to transcribe business problems to Machine Learning parameters.

However, in Prevision Platform, you can build what is called **experiment** to help refine your objectives

Table 14: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn
Define the problem	Done	a metric to validate the models	A week

Experiment

An experiment is a set of Model Building with slightly different parameters across version and a common Target on each version. On each experiment, many models will be automatically built evaluated in cross-validation and on the holdout dataset if you provide some.

In our case, the models will be trained on our engineered dataset **with a fold column** and evaluated on an holdout provided by IT Team.

It is very important to have a good validation strategy to guarantee that the model built in the experiment phase will stay stable on production. Here we choose to :

- build a fold column on the modulo of the year number so that we stay confident that the model learned some trends that stay stable over the year
- Validate on an holdout with sales from a year that was not in the trainset

Hence, if the holdout score is near to the cross validation score, we know that our model is going to stay good when launched in production and shared across the company

For creating a new experiment, go to the **experiments** section of your project and click **new experiment**. You could choose to import some *external models* if you have some but here we are using the AutoML Prevision Engine. As we want to forecast sales, choose “Tabular” and “Regression”. Give a name to your experiment and click “Create experiment”

When you create a new experiment, there is no version of the experiment existing so you will be prompted to create a new version. The next screen is where you setup all your experiment parameters :

- The train dataset : use the output of the Schedule run from the step 2 with engineered features
- The holdout dataset : use a dataset with same target than trainset but with data that **are not** in the trainset
- The metric : use the fittest metrics that solves the business objectives defined in the step 3. You can change it on each version of your experiment so run as much version as you need if you are not sure
- set your target (here we choose “Weekly Sales”)

The screenshot shows the 'Sales prediction' experiment setup page. The interface includes a sidebar with navigation icons, a top bar with the experiment title, and a main content area. The 'New experiment' section contains the following fields and options:

- Buttons:** 'Cancel and back' and 'New experiment'.
- Model Type:** 'AutoML' (selected) and 'External model'.
- Name:** A text input field with the placeholder 'Experiment name'.
- Data type:** 'Tabular' (selected), 'Timeseries', and 'Images'.
- Training type:** 'Regression' (selected), 'Classification', 'Multi-classification', and 'Text similarity'.

On the right side, there is an 'AUTOML' section with the following text:

AUTOML

The Prevision.io AutoML engine allows you to quickly benchmark and optimize a range of open source algorithms to get highly performant models.

What do I need?

You just need data, imported into Prevision.io as a Dataset. Your dataset just needs to contain a target column (and a temporal one, if you are working with time series), and you are good to go.

What's next?

Once the experiment and the models are created, you can analyze, version, and deploy them to create a prediction API endpoint, or use it with pipelines to schedule batch predictions.

At the bottom right, there are 'Cancel and back' and 'Create Experiment' buttons.

Fig. 99: Setting the experiment up

The screenshot shows the 'Sales prediction' experiment configuration page for a 'New version'. The interface includes a sidebar with navigation icons, a top bar with the experiment title, and a main content area. The 'New version' section contains the following fields and options:

- Buttons:** 'Cancel and back' and 'New version'.
- Version description:** A text input field with the placeholder 'Describe your Experiment'.
- Data:**
 - Dataset:** 'save folds' (selected).
 - Holdout (optional):** 'holdout_sales' (selected).
- Training options:**
 - Metric to use:** 'RMSE - root mean squared error' (selected).
 - Performances:** 'QUICK' (selected), 'NORMAL', and 'ADVANCED'.
- Fields configuration:**
 - Target column:** 'Weekly_Sales' (selected).
 - ID column (optional):** (empty).
 - Weight (optional):** (empty).
 - Fold (optional):** 'fold' (selected).

On the right side, there is a 'Directed Acyclic Graph (DAG) Preview' section showing a visual representation of the workflow. At the bottom right, there are 'Cancel and back' and 'Train' buttons.

Fig. 100: Experiment parameters

- and set the fold column up, using the column built during the feature engineering phase.

Note that you may go the models and feature engineering tabs to change some automl configuration but in most of case the default configuration is fine.

Once done, click on **train** to launch the training. The platform will immediately start to build and select models with the best hyper parameters. The models will stacks in the “models” tabs of your experiment :

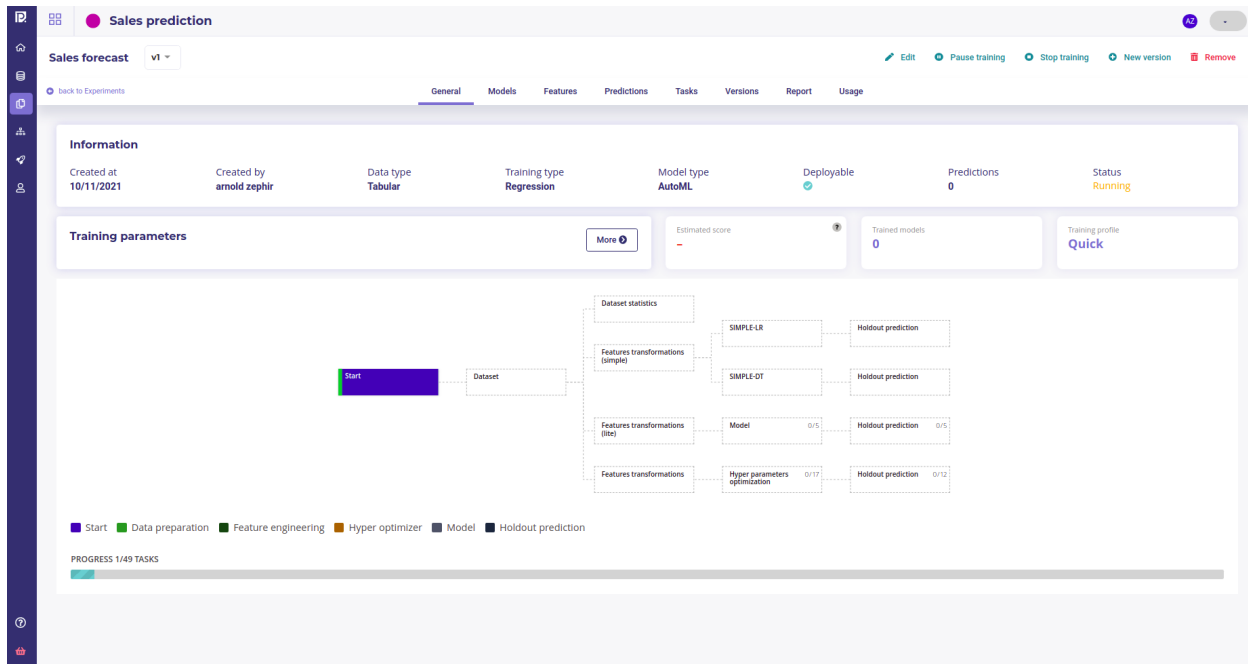


Fig. 101: The experiment dashboard

Note that you can launch another version of your experiment as soon as you want, for testing other metrics for example, by using the new version button in the top right corner.

If you have several version , the experiment dashboard will always display the last version but you can change to another version with the version dropdown menu or the **versions list** tabs

You can launch as much version as you want and they will run in parallel. You can now grab a coffee and wait till models are built ! Depending on the size of your dataset and the plan you subscribed, expect to wait from 10mn to 2 hours before having enough models to evaluate your experiment. In our case, we got our model in ~20mn

Table 15: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn
Define the problem	Done	a metric to validate the models	A week
Experiment	Done	~100 models	~20mn

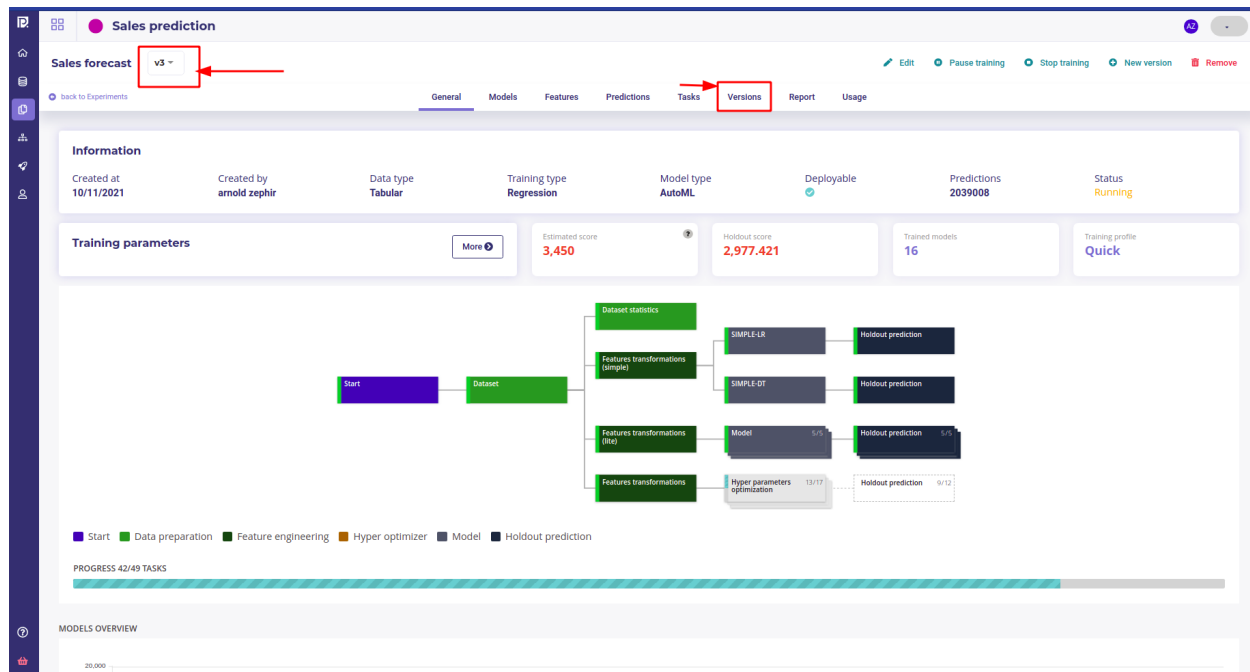


Fig. 102: The experiment dashboard

Evaluate

After a few minutes, you should have between 15 and 40 models for each version, depending on your option

This step is all about evaluating all the models produced and select 2 to 4 models to deploy for testing model in real conditions.

First, have a quick look at the list of versions below (tab **versions** of your experiment). There is a small 3-star evaluation that gives you informations about versions quality. In that case, the Version 3, that has been trained on Mean Absolute Error, looks the most promising. Click on the version to enter the version dashboard for deepest analysis.

On the Version dashboard, you got several indicator but the most important is the models comparator :

You can quickly see :

- performance of each model done , evaluated on the metrics you choose for this version.
- stability of each model (represented with a small error bar) computed on a cross validation of the trainset **using the fold column provided**

Dumb models

Prevision Platform always produces what we sometimes call “dumbs models”, a linear regression and a Random Forest of only 5 depth, called simple-LR and simple DT. It is always a good idea to watch performance of this models against the most complexe one and ask yourself if using them could be good enough for your problem.

Indeed, as they are very simple :

- they can be implement in sql (auto-generated code is even provided on the model analysis page)
- they often are more explainable and are more accepted from the Business teams, are they are easier to understand and use.

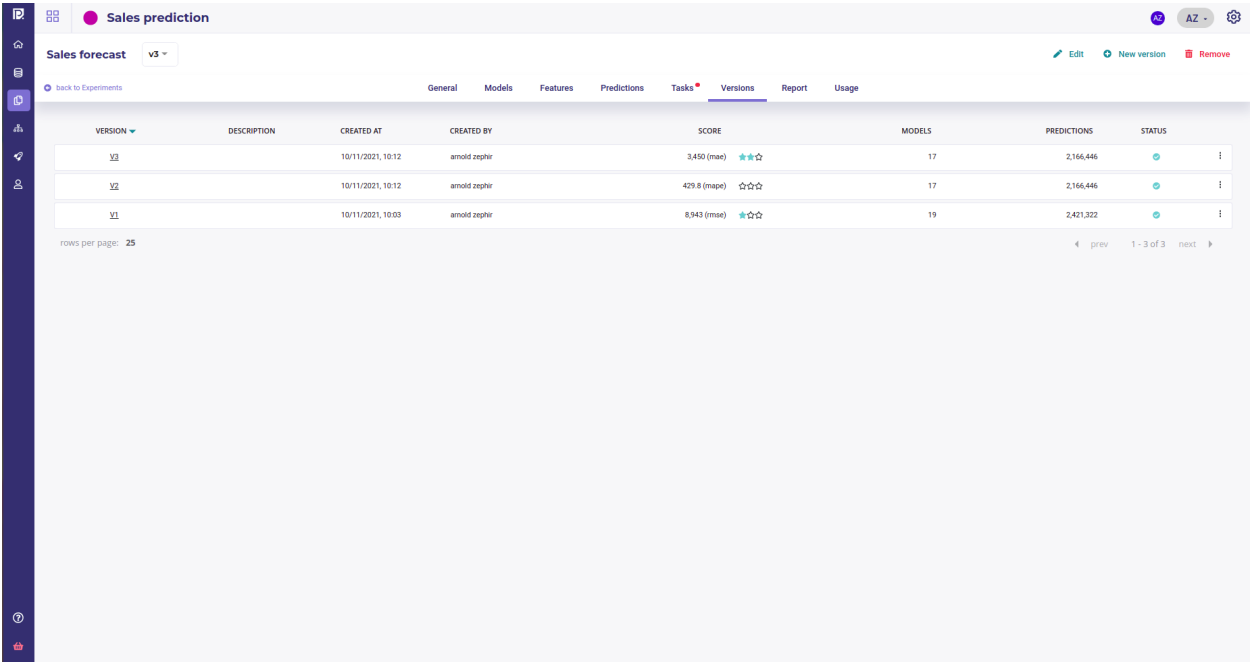


Fig. 103: List of experiment versions.

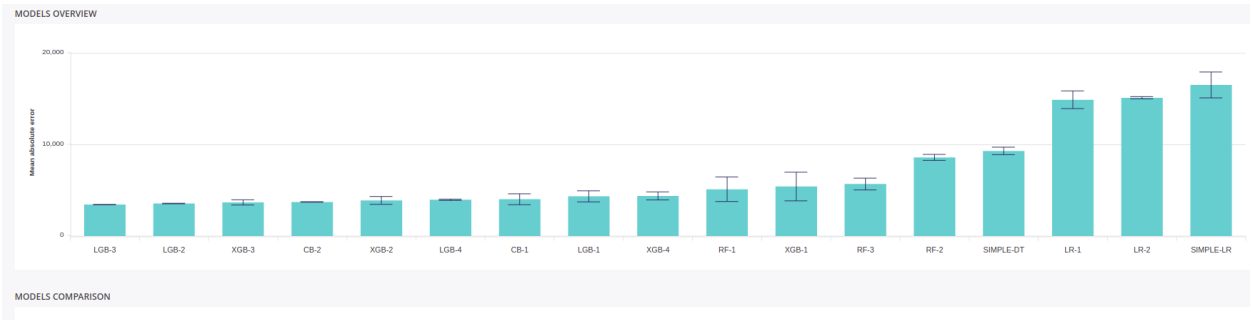


Fig. 104: List of models of a version

As a datascientist, accepting to use a simple if-else instead of complex Blend of Gradient Boosting if it solves the issue is your responsibility too !

On the experiment above, the :

- LGB-3
- XGB-4
- CB-2

Looks promising so we are going to have a closer look. Click on the model barplot to enter the detailed model analysis, CB-2 for example.

Here you got more detail about the models you select, like various metrics and the actual vs predicted Scatterplot

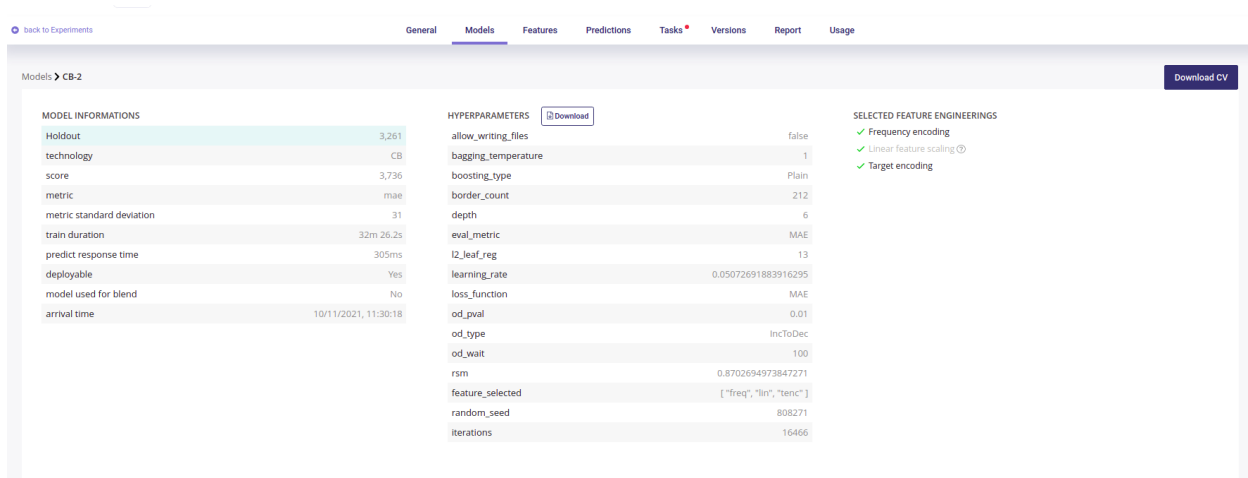


Fig. 105: All the metrics of the model

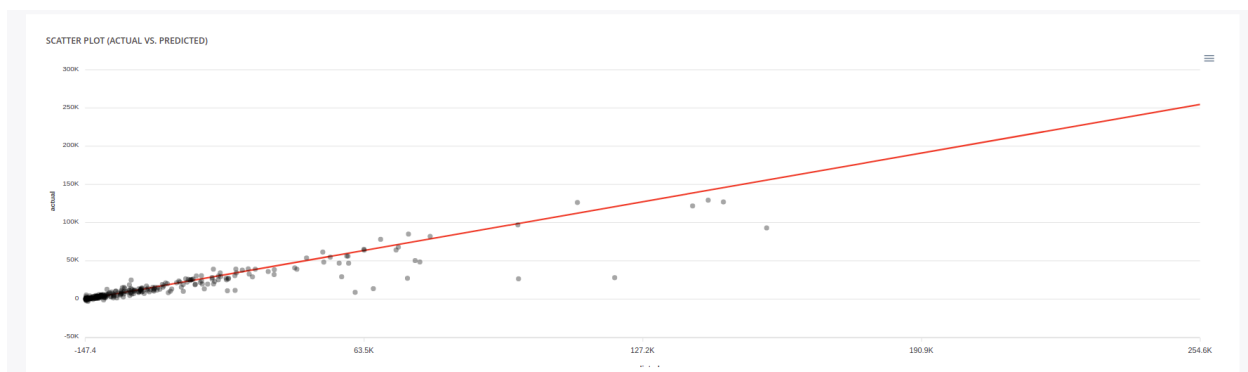


Fig. 106: Predicted vs actual

You can download the Cross validation file if you want to run your own evaluation. The CB2 is quite good but if we look at the Scatterplot, we see that performance fall in the range from 40k to 80k. If we go to the LGB-3 page, we see a more stable performance

Evaluating a model is out of the scope of this guide but be aware that it is another step where you **MUST** involve your business team and explain each metrics and chart to them so you choose together the model that solves their problem the best.

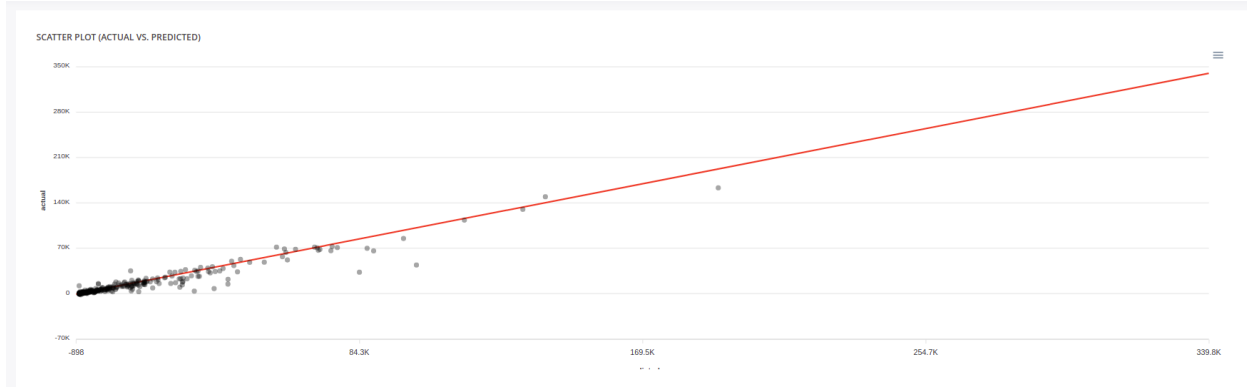


Fig. 107: Predicted vs actual (LGB-3)

The model analysis page is full of metrics to parse and you can run as much experiment as you want in order to find the model that fits the business problem the best.

After discussions with the LoB, we decided to keep the LGB-3 and the XGB-4, one because it performs well and the others because its performance is stable when evaluated on the holdout.

In order to refine this, we are now going to deploy both models and see how they perform in real world.

Table 16: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn
Define the problem	Done	a metric to validate the models	A week
Experiment	Done	~100 models	~20mn
Evaluate	Done	2 models selected with business team	A week

Deploy

In this step two models will be deployed in order to test them on real data and usage. While deployed, their performance will be closely monitored for deciding if they are good for production grade utilisation.

Go to the “Deployment” section of your project and click on **deploy a new experiment**. Select LGB-3 as main model and XGB-4 as a challenger in order to see which one performs best on real data.

The Main model will be used for prediction but each time you call it, a prediction will be done with the challenger model too and chart will be generated so you can compare them.

Wait a few minutes to get :

- a standalone webapp for human user to test (“Application link” url)
- a batch predictor available for scheduling prediction

Sales prediction

Deployments experiments | Deployments applications

Deploy a new experiment Deploy

Deployment name
Sales forecasting run

Description (optional)
<https://sales-forecasting-run.int.prevision.io/>

Select a experiment: Sales forecast X Select a version: 3 X Select a tagged model to deploy: LGB-3 X

Select a challenger model (Optional)

Select a experiment: Sales forecast X Select a version: 3 X Select a tagged model to deploy: XGB-4 X

ACCESS RIGHTS
☐ Public ☒ Instance collaborators ☐ Project collaborators

Fig. 108: Set your main and challenger

- a REST API for calling the model from others software (“Documentation API” link)

That’s all. Your model can now be called from any client of your company and all its requests will be logged for further monitoring. Yet, in order to send prediction each week to the sales team, you need to schedule them.

Table 17: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn
Define the problem	Done	a metric to validate the models	A week
Experiment	Done	~100 models	~20mn
Evaluate	Done	2 models selected with busines team	A week
Deploy	Done	Model available accross the organisation	5mn

The screenshot shows the 'Deploy a new experiment' form in the Prevision.io interface. The form includes the following fields and options:

- Deployment name:** Sales forecasting run
- Description (optional):** (empty text area)
- URL:** <https://sales-forecasting-run.int.prevision.io/>
- Select a experiment:** Sales forecast (dropdown with 'X' icon)
- Select a version:** 3 (dropdown with 'X' icon)
- Select a tagged model to deploy:** LGB-3 (dropdown with 'X' icon)
- Select a challenger model (Optional):** (empty dropdown with 'X' icon)
- Select a experiment:** Sales forecast (dropdown with 'X' icon)
- Select a version:** 3 (dropdown with 'X' icon)
- Select a tagged model to deploy:** XGB-4 (dropdown with 'X' icon)
- ACCESS RIGHTS:**
 - ☐ Public
 - ☒ Instance collaborators
 - ☐ Project collaborators

A 'Deploy' button is located in the top right corner of the form.

Fig. 109: Set your main and challenger

Schedule

Once any model is deployed, it can be used to schedule prediction. First step is to insert it into a pipeline template and then create a new Schedule using this template.

Note that you need helps from your IT team in this step, in order to define the name of the table where you will read the features from each week. You can use the same table that will be overwritten each week, for example “sales to predict” to read and “Sales predicted” to write, or a more complexe naming scheme.

First you need two create two new assets :

- a new datasource that gonna link to the Table were the IT team is going to put the features for prediction each week
- a new exporter to push the result

So you can use them in a new pipeline template with 3 nodes again :

- Import from the datasource, where the datasource is the table with all the weekly features
- a deployment predict regression node
- an export dataset node, that use the exporter above

Once you got your template, create a new Schedule based on it

And choose the Name of your deployment as the experiment deployment Id

And then, instead of the manual Trigger, use a periodic one , putting the configuration that fits your need the best (here, a weekly prediction each Monday at 7:00 AM)

Click run and wait few seconds. Your Prediction is now scheduled to run every Monday , from the table of “sales to predict” to the “Sales predicted” table of your databases.

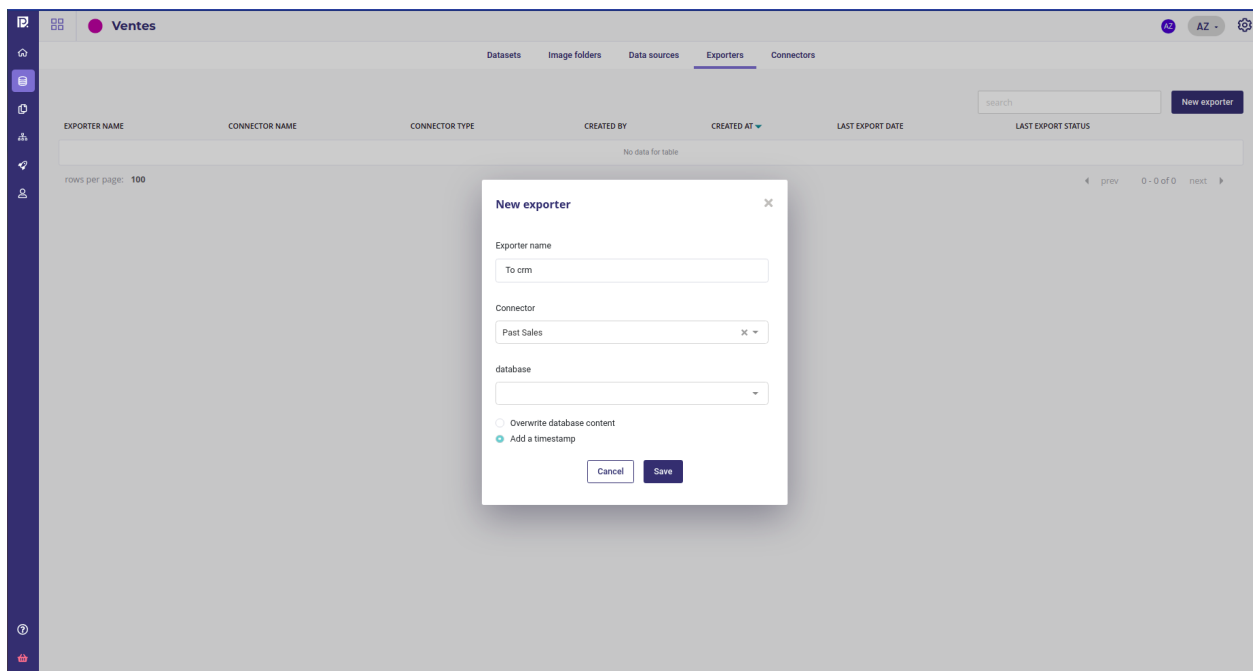


Fig. 110: Create an exporter to push data to your crm

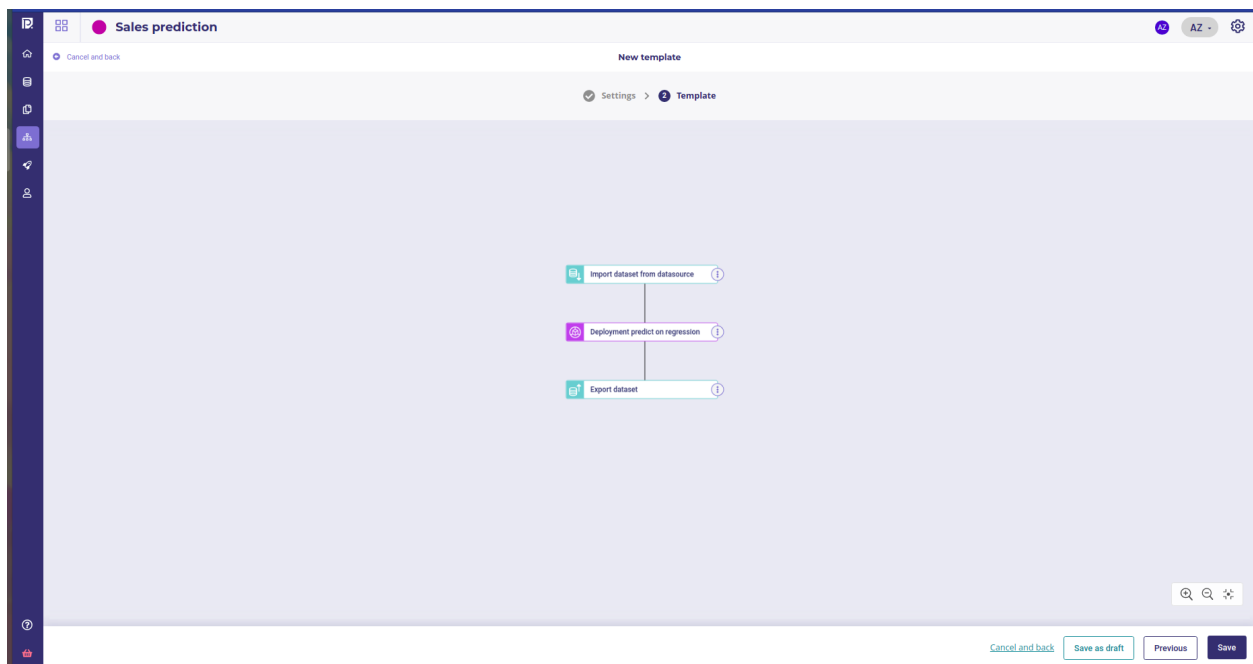


Fig. 111: Template

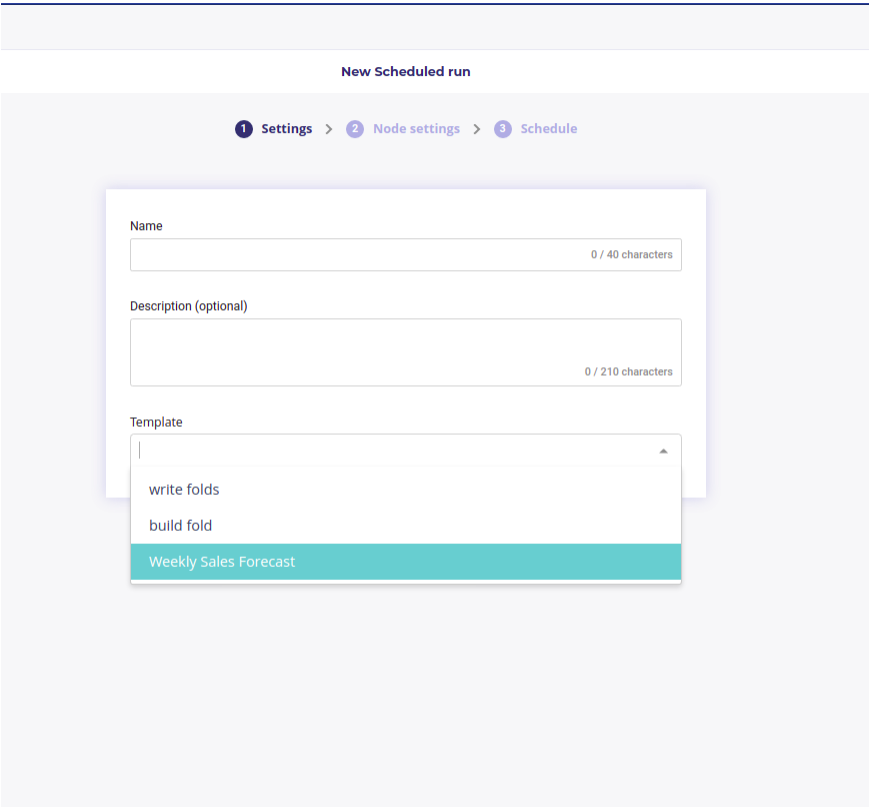


Fig. 112: Use your template in a schedule run

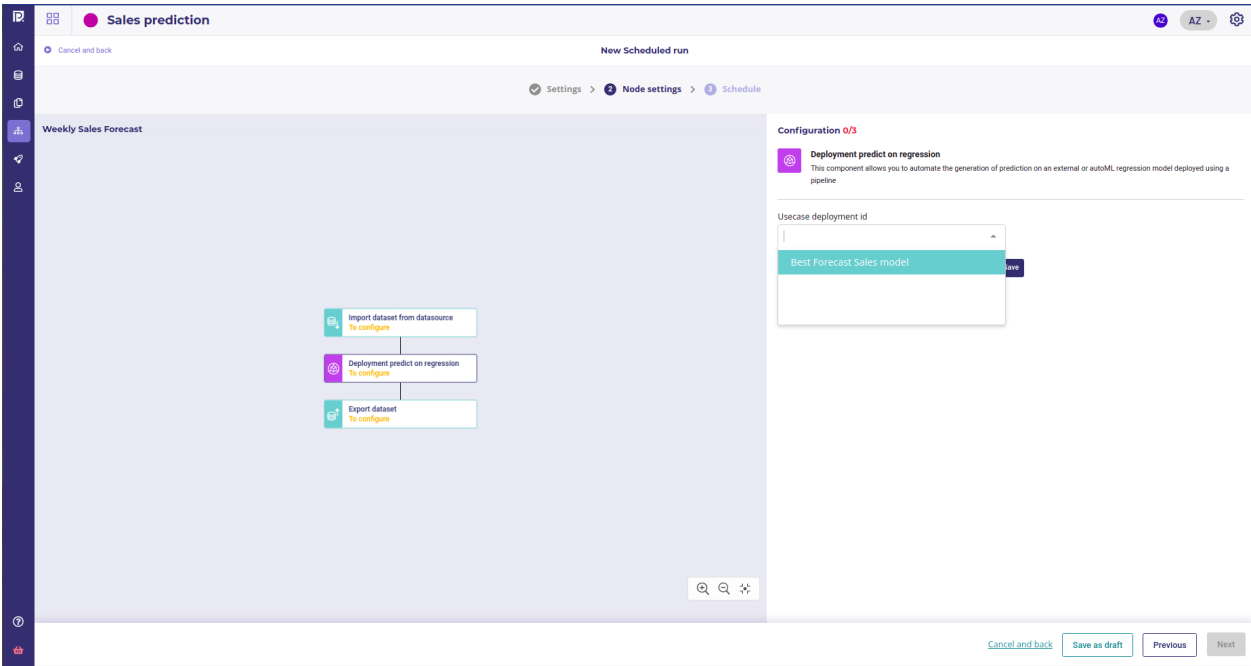


Fig. 113: Use your template in a schedule run

Cancel and back

New Scheduled run

Settings > Node settings > Schedule

Frequency

Trigger type
Periodic

Period
Weekly

Day
Monday

Hour
7am

Minute
0

At 07:00 AM, only on Monday

Duration (optional)

2021-10-13 ~ 2021-11-16

Cancel and back Save as draft Previous Run

Fig. 114: Scheduling a prediction each monday Morning

Table 18: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn
Define the problem	Done	a metric to validate the models	A week
Experiment	Done	~100 models	~20mn
Evaluate	Done	2 models selected with busines team	A week
Deploy	Done	Model available accross the organisation	5mn
Schedule	Done	Prediction in CRM each Monday at 09:00	20mn

Monitoring

Once a model is deployed, each call to it will be logged, being unit one or scheduled batch. You can track your model into the Deployments section of your project by clicking on a deployed experiment name in the list of experiments to access the deployment dashboard :

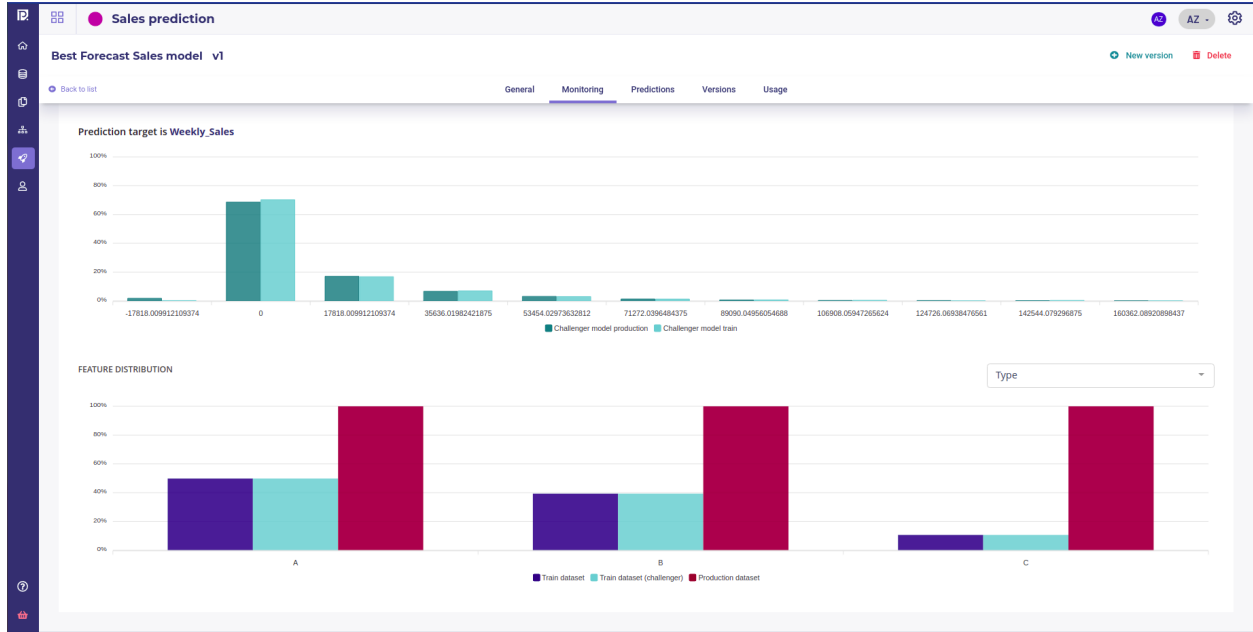


Fig. 115: Train and production distribution

You can watch the features distribution of the trainset compared to the feature distribution seen in production and check the drift. Target distribution of the Main Model and Challenger model are shown side-by-side with those of the production in order to evaluate performance in a real application.

Under the monitoring/usage tab sit some SLA statistics about number of call average response time and errors.

By tracking all this indicators for a month or more, you can evaluate how does your model lives in production and check that it behaves the way you expected while evaluating it in the experiment step.

Table 19: Status

Task	Status	Output	Time spent
Data acquisition	Done	one trainset, one holdout	5mn
Feature engineering	Done	one engineered dataset with features, one holdout	20mn
Define the problem	Done	a metric to validate the models	A week
Experiment	Done	~100 models	~20mn
Evaluate	Done	2 models selected with business team	A week
Deploy	Done	Model available across the organisation	5mn
Schedule	Done	Prediction in CRM each Monday at 09:00	20mn
Monitor	Done	Prediction in CRM each Monday at 09:00	A month

Conclusion

In this guide, you saw how to complete the whole datascience process in less than a morning and went from data to fully deployed model, shared across the company with full monitoring.

Using a tool to solve the technical issue of the datascience, like finding the best model, deploy a model or import the data, allow to spend more time on what truly matters : talk with the Line of Business team to translate their problem to datascience configuration and metrics.

Cross validating your machine learning models

There is a common trope about data science that beginner data scientists often outperform more experienced ones and get models with 99,9% accuracy, compared to experienced data scientists, who rarely get 80%. Yet, as the term implies, data science is all about data and science. And science means having a good validation protocol.

Cross Validation Strategies

Learning is a process that involves:

1. Gathering data
2. Making a model or hypothesis that explains the experimental data
3. Checking that hypothesis on never-seen data _ (See this excellent, if very technical article, about [learning by Scott Aaronson](#) .)

Like human learning, machine learning should follow this protocol. Most scientists are fully aware of how important the third part of the process is, because building models that work only on the data you got is not very useful. But for those just starting out in data science, it's a common mistake to neglect it.

Often, ignoring this step results in models that get very high performances while experimenting, but then collapse when deployed and used on real, never beforeseen data. This comes from two main phenomena:

- Data overfitting
- Data leakage

NAME	CREATED AT ▼	MODEL	SCORE	VALIDATION SCORE
walmart_sales_holdout	10/07/2021, 20:53	GB-4	88,976	346,703 ☆☆☆

Fig. 116: A poorly done modelization. Model scored 88,976 RMSE on training data, but a 346,703 RMSE on validation data.

As machine learning is very powerful, data overfitting can, and will, happen a lot. With thousands of parameters, you will always succeed in building a model that perfectly fits your experimental data, yet predicts few — or no — new data.

Another issue, data leakage, is when your model starts to memorize the relationship between samples and targets instead of understanding them. It's common with data that's split into groups, like categories of goods or people. The model might memorize the group's most frequent target instead of the relationship between its feature and the target.

When confronted with a new category of goods, the model will then be clueless.

Validation technique: Know about the future performance

Validation techniques are an important part of a solid strategy in order to keep models and hyperparameters performing their best on future data.

It's a set of techniques that helps guarantee that, in addition to getting a good model initially, the model will stay good when exposed to new data. In other words, the wonderful performance you got in the lab won't collapse when your company agrees to use your model worldwide for piloting all its vital processes.

Common techniques to prevent performance collapse

Luckily, the validation problem has been widely studied and we, as data scientists, benefit from several methods to get model performance as good in production as it was during experimentation. All of these methods rely on a common principle — put data aside when training, and evaluate models on the data put aside.

Yet, the art is all about knowing which data to put aside.

Level 0: Train test split

The first technique is a very basic one, but it works: Get our data, split it in two sets, usually 80% and 20% of the size of the original dataset, respectively called the trainset and holdout. Then, train on one set, and measure performance on the other one.

Even if it's not foolproof, this technique often prevents too much overfitting and is the least time-consuming. Split your data, make one train pass, and validate! Just be sure to use a sample without replacement when splitting your dataset, so none of the sample data in the trainset is memorized and then automatically predicted during validation (except if you want to cheat).

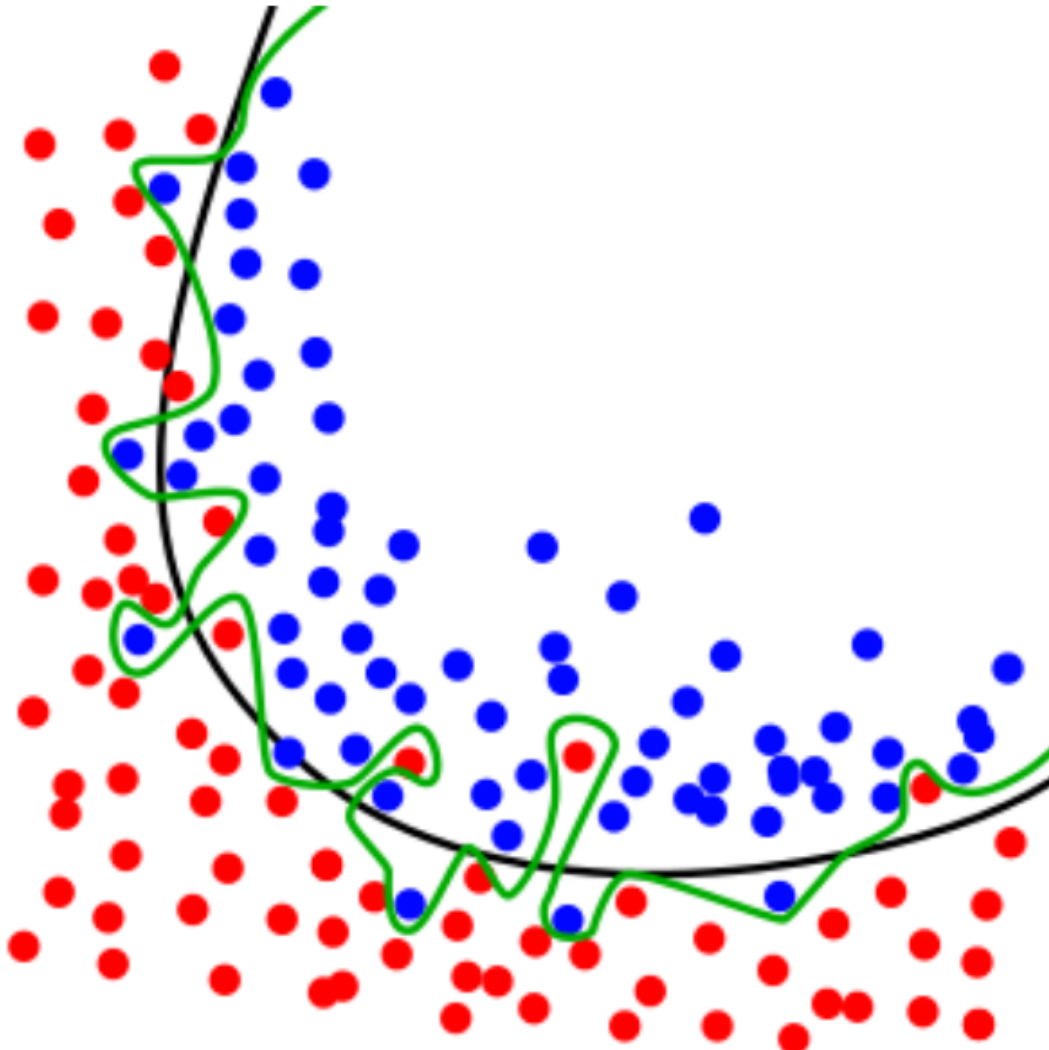


Fig. 117: A 100% accurate model, in green, and a good model, in black.



The screenshot displays the 'Basics*' tab of the Prevision.io configuration interface. At the top, there are five tabs: 'Basics*', 'Columns configuration', 'Models', 'Feature engineering', and 'Feature selection'. Below the tabs, the 'Version description' section contains a text box with the text 'Validate models on Holdout'. The 'Data' section includes a 'Dataset' dropdown menu set to 'trainset_fraud' and a 'Holdout (optional)' dropdown menu set to 'holdout_fraud'. The 'Training options' section features a 'Metric to use' dropdown menu set to 'AUC - area under the receiver operating characteristic curve'.

Basics* Columns configuration Models Feature engineering Feature selection

Version description

Validate models on Holdout

Data

Dataset ?

trainset_fraud

Holdout (optional)

holdout_fraud

Training options

Metric to use

AUC - area under the receiver operating characteristic curve

Fig. 118: Use two datasets: One for training, one for validation.

Level 0.5: Train test split with good normalization

Before moving on to the next technique, let's talk about a common error that's often underestimated. Data scientists frequently normalize, or apply some kind of transformation, to data before training. When doing that, always fit your transformation on the trainset, not the whole dataset.

cheat).

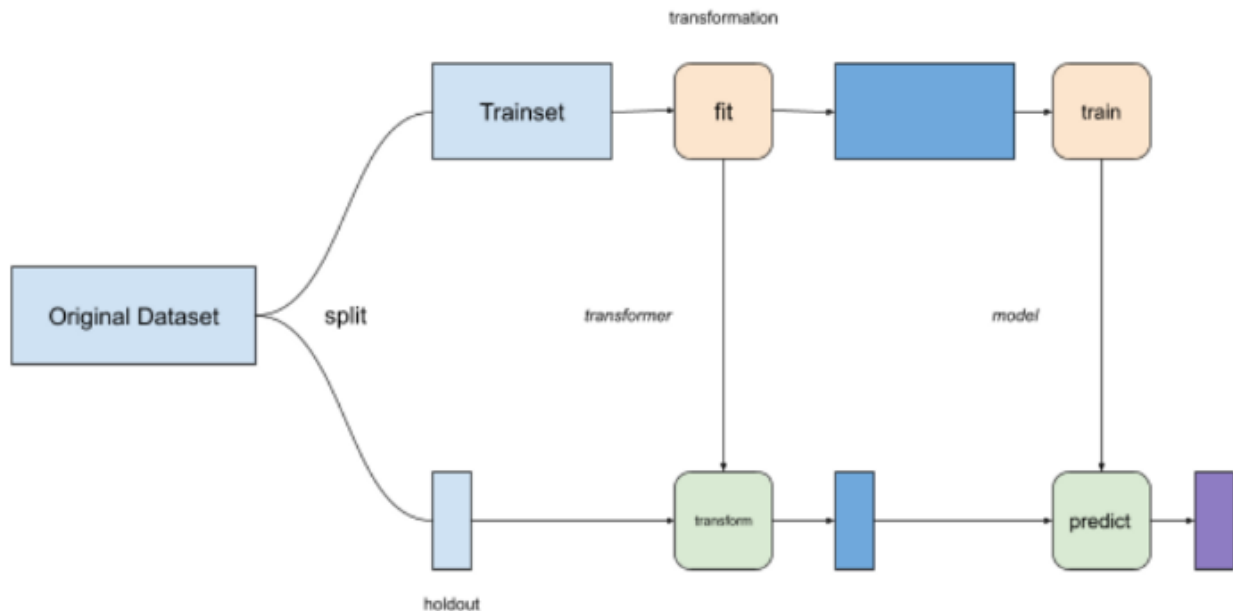


Fig. 119: Do

If you apply some kind of transformation, like normalization, to the whole dataset, some information will leak between the holdout step to the training step. You will get a little boost of performance from it, but that boost will disappear when the model goes into production.

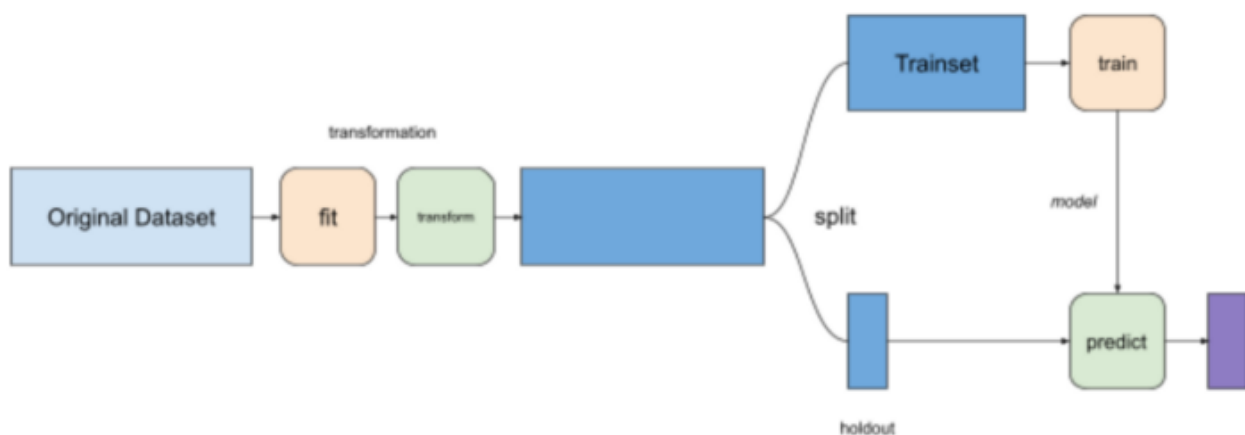


Fig. 120: Don't

Level 1: Folds

In the very basic technique of splitting a dataset, we did not address the topic of variance, or stability, of a model.



Fig. 121: Variance represented as error bar on each model performance.

Variance is a way to quantify how much your performance will change if you change the split of your model, even if you preserve the underlying distribution across each trial.

It is a very good indicator of generalization capability, as your performance should not change that much if you randomly sample training data from your original dataset.

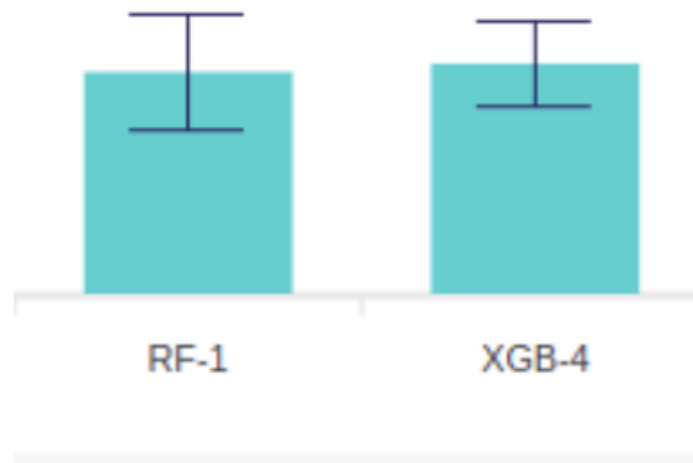


Fig. 122: Which one would you put to real use? RF-1 is a little bit better, but if you take its standard deviation into consideration, represented as an error bar, expect to be less confident about its performance on future data.

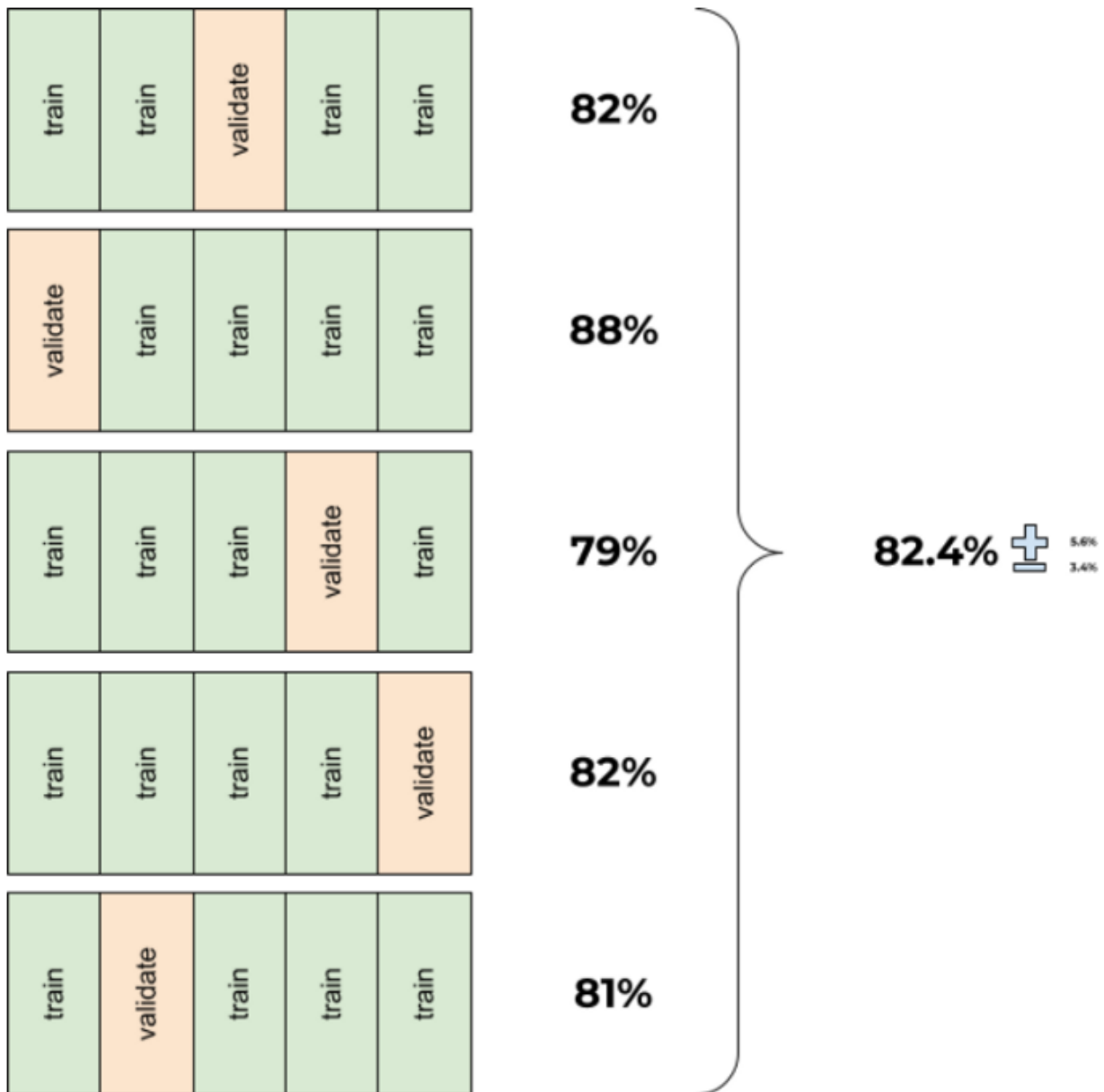
If the performance changes significantly when you change your training data, it means that the way you built your model is very unstable and, in most cases, it will be very unpredictable when put in production.

It's like reaching a high equilibrium on top of a very narrow mountain. You get to be the best, but any minor change in the data could throw everything off and plummet you to the bottom.

That's why when building models, you should adopt a K-fold validation strategy:

1. Choose an integer k (generally $k = 3$ or 5 works well).
2. Split your dataset into k parts.
3. Choose one part.
4. Train on the remaining $(k-1)$ parts.
5. Validate on the part you chose in step 3.

6. Start again from step 3, until you round around all of your k parts.



By splitting data in a round robin manner you got two results: - Average performance - Variance of this performance

There is no definitive way to solve the tradeoff of performance vs. variance, but as a data scientist, you must be aware of the consequences of selecting a high-variance model — and be able to explain that to your business coworkers.

In the Prevision.io platform, k-fold cross-validation is always done each time a model is trained, and variance is displayed on the model list of each version, so you can choose which one fits best with your team. (Or, more precisely, Prevision.io's platform uses a stratified k-fold strategy — check it out in the next section.)

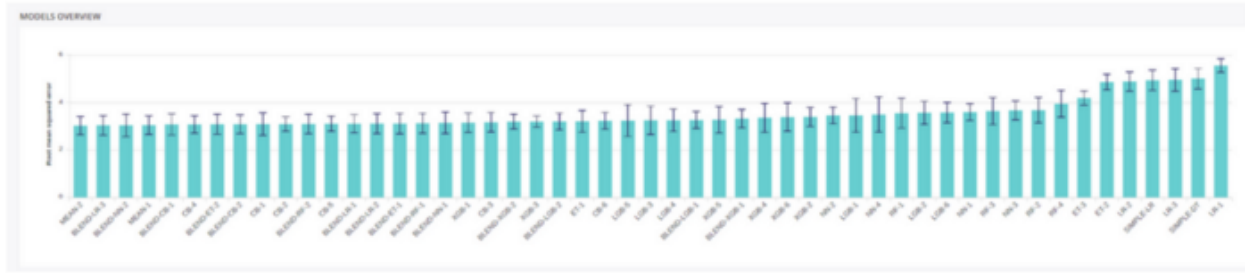


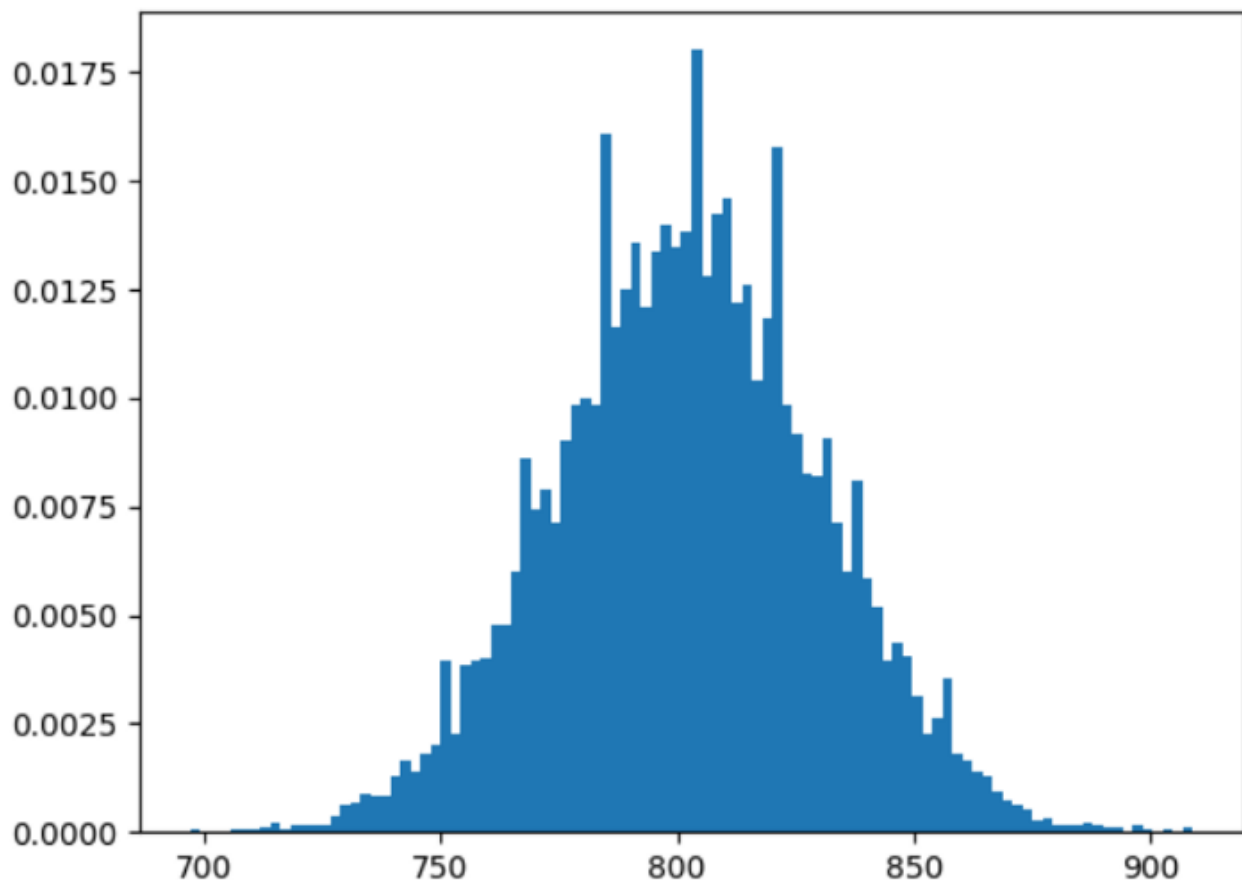
Fig. 123: Don't feel that good about this one...

Level 2: Stratify

Even when folding, another problem would arise when the data is very unbalanced.

Let's say that you are training a classifier on a dataset whose target rate is 1%, and you use a 5-fold strategy. On a dataset of 100,000, each fold will be 20,000 samples, and the training set on each round will be 80,000 samples. As the target rate is 1%, we expect to get 800 positives in the training set and 200 in the validation set.

Yet, if you just sample randomly, the distribution of your target rate will look this way.



In approximately 10% of your folding round, the target rate difference between the trainset and the validation set will be more than 10%!

In the worst case, for example, your trainset could get 720 out of 1,000 of the existing positives, that is a 0.9% rate,

while the validation set would get 280 positives, a 1,4% rate.

This Problem only increases with lower target rates. With a target rate of 0.1%, you can see a 10x factor between the target rate of the trainset and the validation set when splitting your fold with a basic random algorithm. One of the most common use cases where this problem arises is in building a fraud classifier, as it's very common that fraud target rates are low.

Cross-validating data with this method is a bad practice as the performance and variance you gain are built on poor foundations. Expecting consistent and reliable performance is not possible.

In order to avoid this, you need to use a stratified k-fold cross-validation strategy, which is a way of splitting the data so that the trainset and validation set always have the same target rate. Even on a balanced dataset, the impact of a stratified k-fold is clearly visible compared to a simple k-fold split:

To a stratified one where target rate is well distributed across folds :

Level 3: Custom Folds

The last common technique, and a very important one, solves the problems that arise on two types of datasets:

1. Dataset with time-indexed data
2. Dataset with columns that represent some kind of group, like goods or people

With this kind of data, machine learning models sometimes get some information from future leakage, or group leakage, and instead of deducing general rules about the relationship between the features and target, it memorizes the average target value of a group or a period of time.

Let's say, for example, that you want to build a model that predicts the prices that customers are willing to pay for fruits based on some of their features (e.g. weight, amount of sugar, origin, or color).

One of your columns probably is the category of fruit (apple, berries, etc.).

If you build a model that uses the category of fruit as a feature, you will probably get some good results. BUT, if a new kind of fruit comes in, your model will probably fail because instead of really learning what determined the price of a fruit, it instead memorized your existing catalog. So sometimes you need to build custom folds in order to know how your model will behave when a new item from an unknown category comes in. Custom folds depend entirely on your use case, and should be built manually with regards to the problem you want to solve in the context of your application.

Temporal data presents the same kind of trap. If you use just a random fold across your dataset, you're going to leak some data from the future that you cannot expect to benefit from when applying the model in real usage.

For example, if you build a trainset with some data from each year over a 10-year span, your model will probably learn some target statistics from each year. For example, if in 2020 the average price of a fruit is 2,3€, you cannot use that for a year that hasn't happened yet, e.g. 2023.

What you really want are general rules that stay true in the future, e.g. a price increase of 3% each year since 2000, or something like that.

Even if using stratified k-fold, you should not sample random data across the whole dataset, but instead build a new feature, often called "fold" that respects the unknown in your future data.

In the case of a fruit category, you could attribute an integer to each category of fruit and split your data regarding this number. In other words:

- Train model on banana and apples, validate it on citrus
- Train model on citrus and apples, validate on banana
- Train model on banana and citrus, validate on apple

```
...
SPLIT NO 1
TRAINING SET SIZE: 0.67 TEST SET SIZE: 0.33
PROPORTION OF TARGET IN THE TRAINING SET
0      0.395210
1      0.305389
2      0.299401
Name: target, dtype: float64
PROPORTION OF TARGET IN THE TEST SET
1      0.360360
2      0.345345
0      0.294294
Name: target, dtype: float64

SPLIT NO 2
TRAINING SET SIZE: 0.67 TEST SET SIZE: 0.33
PROPORTION OF TARGET IN THE TRAINING SET
1      0.353293
2      0.323353
0      0.323353
Name: target, dtype: float64
PROPORTION OF TARGET IN THE TEST SET
1      0.336336
2      0.333333
0      0.330330
Name: target, dtype: float64

SPLIT NO 3
TRAINING SET SIZE: 0.67 TEST SET SIZE: 0.33
PROPORTION OF TARGET IN THE TRAINING SET
2      0.36747
1      0.36747
0      0.26506
Name: target, dtype: float64
PROPORTION OF TARGET IN THE TEST SET
0      0.359281
1      0.329341
2      0.311377
Name: target, dtype: float64
```

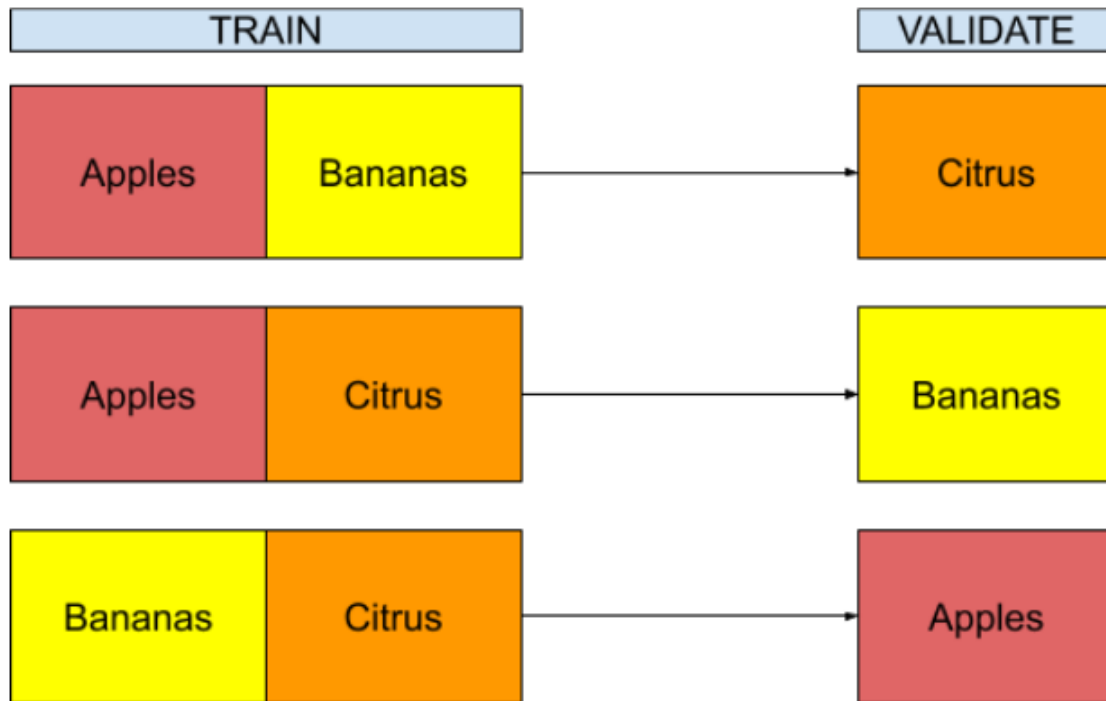
Fig. 124: k-fold

```
SPLIT NO 1
TRAINING SET SIZE: 0.67 TEST SET SIZE: 0.33
PROPORTION OF TARGET IN THE TRAINING SET
1    0.341317
2    0.329341
0    0.329341
Name: target, dtype: float64
PROPORTION OF TARGET IN THE TEST SET
1    0.342342
2    0.330330
0    0.327327
Name: target, dtype: float64

SPLIT NO 2
TRAINING SET SIZE: 0.67 TEST SET SIZE: 0.33
PROPORTION OF TARGET IN THE TRAINING SET
1    0.341317
2    0.329341
0    0.329341
Name: target, dtype: float64
PROPORTION OF TARGET IN THE TEST SET
1    0.342342
2    0.330330
0    0.327327
Name: target, dtype: float64

SPLIT NO 3
TRAINING SET SIZE: 0.67 TEST SET SIZE: 0.33
PROPORTION OF TARGET IN THE TRAINING SET
1    0.343373
2    0.331325
0    0.325301
Name: target, dtype: float64
PROPORTION OF TARGET IN THE TEST SET
1    0.341317
2    0.329341
0    0.329341
Name: target, dtype: float64
```

Fig. 125: stratified k-fold



The performance will probably decline from building folds this way, but you could expect it to stay stable if your grocery gets a new kind of fruit.

For temporal data, in addition to folding over a component of the date (year, month, week), you should train on past data and validate on future data. This strategy is called temporal blocks folding and should always be used when dealing with temporal data. It might look like:

- Training on 2010 and 2011, validating on 2012
- Training on 2010, 2011 and 2012, validating on 2013
- Training on 2010, 2011, 2012, 2013 through 2019, validating on 2020

These cross-validation strategies, both group folding and temporal block folding, are quite complex and you really need to understand your data well to implement them. Because building custom folds always leads to a dip in performance, you want to be sure that doing so serves a model stability purpose.

In the Prevision.io platform, you could define one of your columns as a custom fold and the cross-validation will be done along that fold. When you use the “timeseries” training type, the platform will automatically use a temporal block to compute its cross-validation score, so you don’t need to define blocks by yourself.

The impact of a cross-validation strategy

Let’s see how a cross-validation strategy may impact your performance.

We built a dataset of sales, with two categories (product and date/time), ranging over 3 years, and decided to measure the performance by holding out sales from 2012 and training models on sales from 2010 and 2011. The quality of our models will be measured as the RMSE on holdout, and we used 6 different folding strategies:

- No folds: We don’t input any folds, so Prevision.io will just use a random stratified k-fold along the trainset
- Random folds: We use a 7-fold, built randomly
- Trimester folds: Trimesters (4) are used as fold

Data

Dataset ?

616066c0119523001ca9ee53

Holdout (optional)

616066cb119523001ca9ee54

Training options

Metric to use

RMSE - root mean squared error

Performances

☒ QUICK ?

☐ NORMAL

☐ ADVANCED ?

Fields configuration

Target column

ID column (optional)

Weight (optional)

Fold (optional)

IsHoliday_x

Fuel_Price

Dept

Date

CPI

NAME	SOURCE	VERSION	CREATED AT ▼	CREATED BY	DATA TYPE	TRAINING TYPE	SCORE
Store folds	AutoML	1	10/20/2021, 10:15	arnold zephir	Tabular	Regression	13,300 (rmse)
triplet month folds	AutoML	1	10/20/2021, 10:13	arnold zephir	Tabular	Regression	7,609 (mse)
all month folds	AutoML	1	10/20/2021, 10:12	arnold zephir	Tabular	Regression	7,792 (rmse)
Trimester Folds	AutoML	1	10/20/2021, 10:11	arnold zephir	Tabular	Regression	9,734 (rmse)
Random Fold	AutoML	1	10/20/2021, 10:09	arnold zephir	Tabular	Regression	3,618 (rmse)
No fold	AutoML	1	10/20/2021, 10:08	arnold zephir	Tabular	Regression	3,635 (rmse)

- All month folds: Months (12) are used as fold
- Triplet month folds: The month number modulo 3 is used as fold (e.g, we train on January and February, and validate on March)
- Store folds: We use the store number as fold

Results

Here is a summary of the model's performance for each strategy. Remember, the lower the RMSE, the better.

Strategy	Trainset RMSE	Cross-validation score Deviation	Deviation in %	Holdout RMSE	Spread trainset → holdout
No folds	3635	195	5.3%	9401	159%
Random folds	3618	91	2.5%	9386	159%
Trimester folds	9734	3897	40.1%	6629	32%
All month folds	7792	2999	38.5%	5746	36%
Triplet month folds	7609	1852	24.4%	7025	7.6%
Store folds	13300	1547	11.6%	9197	30%

We see that there are 3 main types of input from our strategy:

- Strategies with no smart folds get very good models (no folds and random folds) in the trainset, as their RMSE are the best. But when applied on holdout, the performance totally collapses. You should never use models like that.
- Some models (trimester folds and all month folds) get the worst score, but remain more stable, and we get a similar performance in holdout. Yet the variance in cross-validation, and the fact that the holdout score is lower than the trainset score, must trigger attention. These models should probably be tested on more data to validate their stability or lack of stability.

- The triplet month fold strategy gets some good performance on the trainset and a very similar one on the holdout. The variance of cross-validation is still high, but could probably be lowered with more powerful models. Of the three, this fold strategy is probably the strongest, and more powerful modelisation algorithm could now be used to get better performance

Note that folding on the store columns get some interesting results, but this is probably due to the lack of temporal folds, which artificially decrease the RMSE and improve their performance.

Text Classification with Prevision.io Data Platform

In this post we will show how in just a few minutes the Prevision.io platform can carry out automated Natural Language Processing and text classification.

It is known that textual data is usually more tricky and harder to process than the linear or categorical features. In fact, the linear features sometimes need to be scaled. Categorical features are scalar straightly encoded, but transforming texts into machine readable format requires a lot of pre-processing and feature engineering. Moreover, there are many other challenges that have to be addressed: how to cover different languages? How is it possible to preserve the semantic relationship between the words' vocabulary? How to embed the sentence context in the words encoding?

Fortunately, Prevision.io provides an automated processing for the textual features, offering an array of textual transformers that will be automatically operate, while addressing all the challenges that we mentioned: language auto-detection, different types of encoding (metric based encoding, wor2vec encoding, and sentence embedding), as well as different type of models.

We will start by retrieving a kaggle dataset called [Real or Not? NLP with Disaster Tweets](#) and show how to launch a text classification “experiment” with Prevision.io, for the first time with the user interface, then using the Prevision.io SDK for “us coders”. The classification use case consists of predicting whether a given tweet is about a real disaster or not.

Text classification experiment with Prevision.io's UI

Create a new project or use an existing one

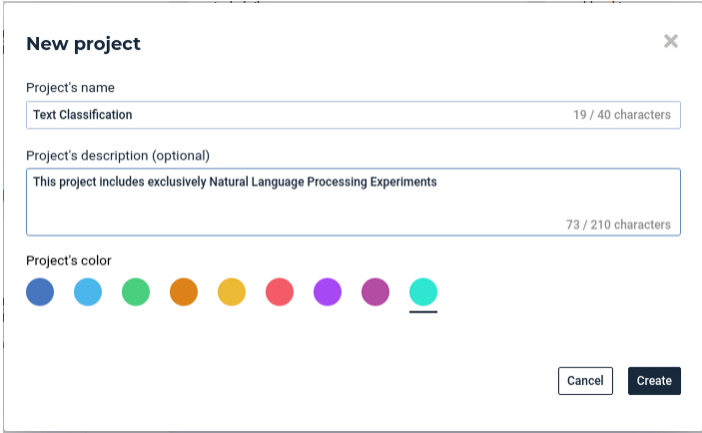


Fig. 126: Create a new project

Once you are connected on your Prevision.io instance, click on the button on the top right of the screen of the home page, to create a new project, you can set up the name of your project and add a small description (optional): Example:

Import your dataset

Once the project is created you upload your dataset by clicking on the datasets tab on the left vertical bar then click on Create Dataset button

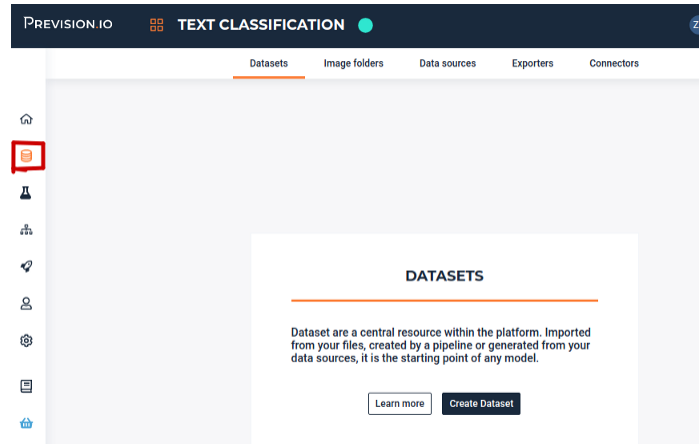


Fig. 127: Import your dataset

Then select Import Dataset option and upload you dataset from your machine:

The screenshot shows a modal dialog titled 'Import dataset' with a close button (X) in the top right corner. Inside the dialog, there are two radio buttons: 'Use a datasource' (unselected) and 'Import file' (selected). Below the radio buttons, a file selection area shows '1 file selected' and a preview of a file named 'train.csv' with a size of '965 KB'. Underneath the file selection, there are three input fields: 'Name' (containing 'train_tweet'), 'Column separator' (containing 'auto'), and 'Decimal separator' (a dropdown menu set to 'auto'). At the bottom of the dialog is a 'Save dataset' button.

Fig. 128: Import csv file

Launch a new Experiment

Once the dataset is loaded you can launch a new experiment by clicking on the experiment tab on the left vertical pallet, then set the following parameters:

The screenshot shows the 'New experiment' configuration window. On the left is a vertical sidebar with icons for home, datasets, experiments, users, settings, and help. The main area is titled 'New experiment' and contains the following fields:

- Model Type:** Two buttons, 'AutoML' (selected) and 'External model'.
- Name:** A text input field containing 'tweet-classification-experiment'.
- DATA TYPE:** Three buttons: 'Tabular' (selected), 'Timeseries', and 'Images'.
- TRAINING TYPE:** Four buttons: 'Regression', 'Classification' (selected), 'Multi-classification', and 'Text similarity'.

Fig. 129: Experiment parameters

Basic setting

For the basic settings, we have to set the essential settings: select the pre-uploaded dataset, add a short description (optional but recommended especially if it is a collaborative project) fix the target (here for a classification experiment it has to be a binary feature which tells whether it is a real disaster or not), fix if there is an ID feature or not (predictions will be applied by id defined by this feature)...

Columns config

In this section, features that were already selected are shaded but you unselect other ones.

Models

In this section you can select the model families that will be experimented. Notice that the Naive Bayes model is available only if we have textual features. In fact it's based on the popular Bayes' probability theorem, and is known for creating simple but well performing models, especially in text and document classification. As a first version, we will chose for examples these models:

The 'Basics*' tab is active, showing the following configuration options:

- Version description:** A text box containing "a first version of disaster tweets classification".
- Data:**
 - Dataset:** A dropdown menu showing "train_tweet".
 - Holdout (optional):** A dropdown menu showing "select a Dataset".
- Training options:**
 - Metric to use:** A dropdown menu showing "AUC - area under the receiver operating characteristic curve".
 - Performances:** Three radio buttons: "QUICK" (selected), "ADVANCED", and "NORMAL".
- Fields configuration:**
 - Target column:** A dropdown menu showing "target".
 - ID column (optional):** A dropdown menu showing "id".
 - Weight (optional):** A dropdown menu (partially visible).

Fig. 130: Basic settings

The 'Models' tab is active, showing the following model selection options:

- Simple models:**
 - ☐ Logistic model
 - ☐ Decision Tree
- Advanced models:**

	Normal	Advanced
<input type="checkbox"/> XGBoost	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Logistic model	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Extra Trees	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Random Forest	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> LightGBM	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Neural Network	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/> CATBoost	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Naive Bayes	<input checked="" type="checkbox"/>	<input type="checkbox"/>
- Blend:** A toggle switch currently turned off.

Fig. 131: Select models

Feature engineering

In this section we will choose the transformations that will be applied upstream from the modeling. The textual feature engineering supported by the platform are the following:

- **Statistical-analysis based transformation (TF-IDF)** : words are mapped to numerics using tf-idf. The platform has integrated fast algorithms making it possible to keep all uni-grams and bi-grams and trigrams tf-idf encoding then applies automatically an efficient dimension reduction algorithm allowing to speed up the modeling task
- **Word embedding transformation**: words are projected to a dense vector space, where semantic distance between words are preserved: Prevision.io trains a word2vec algorithm on the actual input corpus, to generate their corresponding vectors.
- **Sentence embedding**: Prevision.io has integrated BERT-based transformers, as a pre-trained contextual model, that captures word relationships in a bidirectional way. A BERT transformer makes it possible to generate more efficient vectors than word embedding algorithms, it has a linguistic “representation” of its own. Furthermore we have integrated multiple berts: the basic Bert for english, and the Multilingual Bert for other languages. Each word is represented by a vector that depends on the language and the context, then we use these context-capturing vector representations as input to basic classifiers to carry out an efficient text classification

Once the experiment is launched you can see your models displayed one by one. You can extract all the analysis that you need (feature importances, metrics, graphs) and also you can directly extract the cross validation, and make new predictions.

Text classification experiment with Prevision.io’s SDK

In this section we will see how we can reproduce exactly the same experiment via the SDK. It can be useful if you want to automate this operation, or integrate it as a component in a machine learning pipeline. Please take into account that this code sample is available with **previsionio** <<https://pypi.org/project/previsionio/> _ version 11.3.1.

All the UI based tasks described in the previous section can be translated to very few code lines as follows.

Connect to the instance

```
1 import previsionio as pio
2 import pandas as pd
3 URL = 'https://cloud.prevision.io'
4 TOKEN = my_token
5 # initialize client workspace
6 pio.prevision_client.client.init_client(URL, TOKEN)
```

Create the Prevision.io dataset from a pandas dataframe

```
1 # use an existing project from id
2 p = pio.Project.from_id(existing_project_id)
3 # please make sure to update with your project ID
4 # create a dataset from a local dataset
5 ## read csv
6 df = pd.read_csv('tweets_disaster.csv')
7 ## create prevision dataset
8 train = p.create_dataset('tweets_dataset_from_sdk', dataframe=df)
```

Configure and launch the experiment

The setting below is similar to what we chose with the user interface previously:

```

1 # experiment config
2 col_config = pio.ColumnConfig(target_column='target', id_column='id')
3 # config use case profile
4 experiment_config = pio.TrainingConfig(profile='quick',
5 simple_models=[], normal_models=['CB'], advanced_models=['CB'], with_blend=False,
6 ↪ features=["text_tfidf", "text_word2vec", "text_embedding", "tenc", "Counter"])
7 # launch a new experiment
8 exp = p.fit_classification(experiment_name='tweets_classification_sdk', metric='auc',
9 ↪ dataset=train, column_config=col_config, training_config=experiment_config)

```

Conclusion

Textual data is one of the most tedious data types to process, that's why it can be a blessing to have a tool that makes it straightforward and fully automated. Thank you for reading my post until the end ! I hope it was helpful! In the next post I will show you how to use the Prevision.io AutoML platform in a machine learning competition.

Synthetic Data for machine Learning : A guide

As often quoted, "Data is the new oil" and we, datascientist, clearly know the value of good data to train our model. However, we often face some problems with datas :

- we don't have access them because of confidentiality issues
- we do not have enough
- they are unbalanced and thus lead to biased model toward more represented class.

In this guide, we discuss the usage and performance of synthetic data generation. If you want to try it by yourself you can :

- Grab the dataset in our datapack section
- generate Fake data with [our synthetic data generation datascience service](#)

Usage of synthetic Data

A common problem in B2C industry, telecom, bank,... And healthcare is confidentiality of data. We want to build models upon some dataset but want to avoid showing any information about personal record.

For example, you may have a dataset whom each rows is age, gender, size, weight, place of residence,... Even if data are anonymized by removing name or email, it's often easy to find some particular person which unique combination of features, for example :

A 30 years old man who live near Gare du nord, earn 30 000€ a month , whom wife is engineer and had subscribed to sport magazine

would be unique enough to know a lot about him, even without identity data. So for some industry, exposing such sensitive data, even for modeling, is clearly a no go.

Second problem is lack of data or worst, lack of data for some segment of the population that will then be disadvantaged by models.

For example, let's say that we got a dataset with 10000 men and 1000 womens and we are using RMSE as Metric for our training. Let's say for sake of this article that every prediction for man have the same error *manError* and womens have a *womanError* constant error

The Score of a model will be :

$$\sqrt{\frac{10000 * manError^2 + 1000 * womanError^2}{11000}}$$

```
df['RMSE'] = ((10000*(df['manError']**2) + 1000*(df['womanError']**2))/(10000 + 1000))**.5
```

And the errors of each gender will weight as follows on the training metric :

Table 20: Gender error weight

FemaleError	100	200	1000	2000	10000	100000
MaleError						
100	100.0	112	316	610	3016	30151
200	193	200.0	356	632	3021	30151
1000	953	955	1000	1128	3162	30166
2000	1907	1907	1930	2000.0	3567	30211
10000	9534	9534	9539	9553	10000	31622
100000	95346	95346	95346	95348	95393	100000

We clearly see that making an error on woman segment weights much less on the optimized metric, as going from a 100 to 10000 woman error only goes from 100 to 3016 total error yet for man it goes from 100 to 9534.

As goal of machine learning model are to find minimum of givne metrics, a model trained on this data will advantage man Segment over woman Segment. Of course, if we had the same number of mens datas and womens datas, the error will be balanced and no bias toward some segment will happen. But adding more woman datas is not enough, we cannot just copy and paste more rows, as it will lack of diversity.

What we should do instead is adding “realistic” woman Datas by sampling from the underlying distribution of each woman feature

Yet, if we just do that the naive way, this could happen :

Table 21: A random sample from latent distribution

ID	age	weight	size	job	sport
random 1233	82	12	193	Salarymen	American Football

(Note: of course retired women may play american football yet weighting 12kg and playing american football when you are 82 years old is quite odd)

There are too tarpit when generating fake data :

- Assuming everything is a normal distribution and reducing a feature to its average and its deviation. In fact, most of real world distribution are skewed
- not caring about [joint probability distribution](#).

And this is were good synthetic Data Generator come to the rescue.

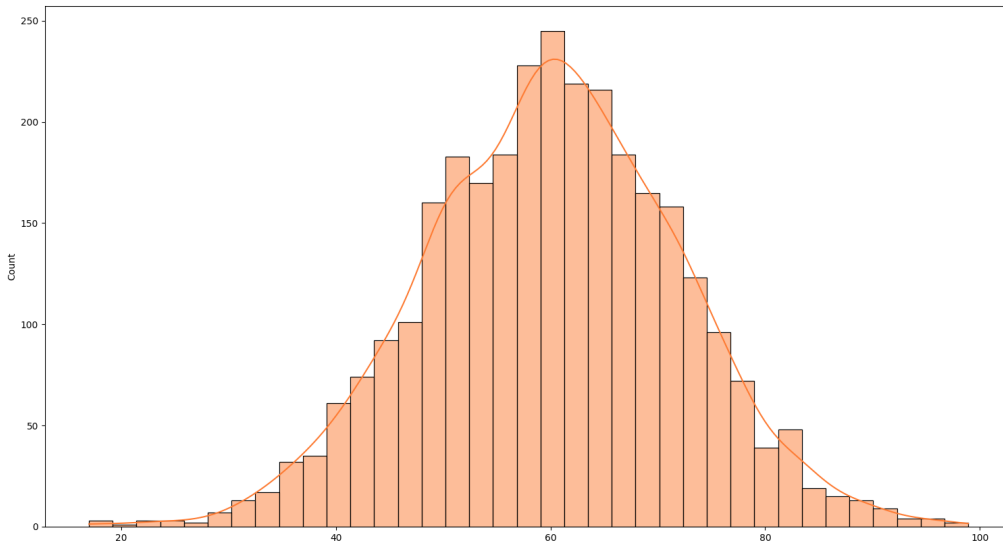


Fig. 132: Weight distribution for some segment

Performance of synthetic Data Generator evaluation

Like every datascience modelisation, synthetic data generation needs some metric to measure performance of different algorithm. This metrics should capture three expected output of synthetic data :

- distribution should “looks like” real data
- joint distribution too (of course, someone who weight 12kj should not be 1,93m tall) (*likelihood fitness*)
- Modelisation performance should stay the same (*machine learning efficiency*)

Statistical properties of Generated Synthetic Data

First obvious way to generate synthetic data is assume every feature follows a normal distribution, compute means and deviation and generate data from gaussian distribution (or discrete uniform distribution) with the following statistics :

Table 22: Dataset statistics

stats	price	bedrooms	bathrooms	sqft_living	sqft_lot	yr_built
mean	5.393670e+05	3.368912	1.747104	2078.563541	1.529471e+04	1971
std	3.647895e+05	0.937394	0.735760	917	4.261295e+04	29
min	7.500000e+04	0.000000	0.000000	290	5.200000e+02	1900
25%	3.220000e+05	3.000000	1.000000	1430	5.060000e+03	1952
50%	4.500000e+05	3.000000	2.000000	1910	7.639000e+03	1975
75%	6.412250e+05	4.000000	2.000000	2550	1.075750e+04	1997
max	7.062500e+06	33.000000	8.000000	13540	1.651359e+06	2015.

In python

```
[...]
nrm = pd.DataFrame()

for feat in ["price", "sqft_living", "sqft_lot", "yr_built"]:
    stats=src.describe()[feat].astype(int)
    mu, sigma = stats['mean'], stats['std']
    s = np.random.normal(mu, sigma, len(dst))
    nrm[feat] = s.astype(int)

# For discrete Features
for feat in ["bedrooms", "bathrooms"]:
    p_bath = (src.groupby(feat).count()/len(src))["price"]
    b=np.random.choice(p_bath.index, p=p_bath.values,size=len(dst))
    nrm[feat] = b
```

Yet this lead to bad feature distribution and joint distribution.

In the following plot, we show the distribution of features of a dataset (*blue*), distribution of a synthetic dataset generated with a CTGAN in quick mode (*orange*, only 100 epoch for fitting) and distribution of data assuming each feature is independant and follows a normal distribution (see code above)

For prices, we see that generated data perfectly captured the distribution of it, so good that we barely see the orange curve. Gaussian random data has same mean and deviation than original data but clearly does not fit the true data distribution. For yr_built, the synthetic data generator (*orange*) struggles to perfectly capture the true distribution (*blue*) but it looks better than the average

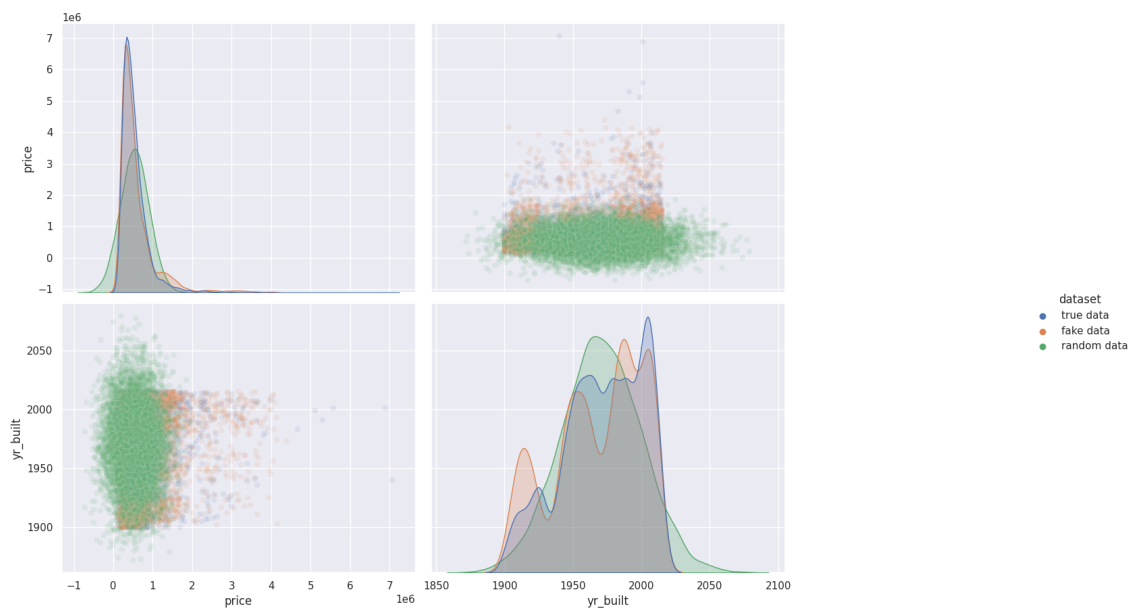


Fig. 133: prices and bedrooms distribution

Here below is same density chart for prices, zoomed

It's more difficult to see joint distribution on this chart but we can draw contourmap to get a better view :

The same conclusion goes for others features too. Random independant data do not fit distribution and joint distribution

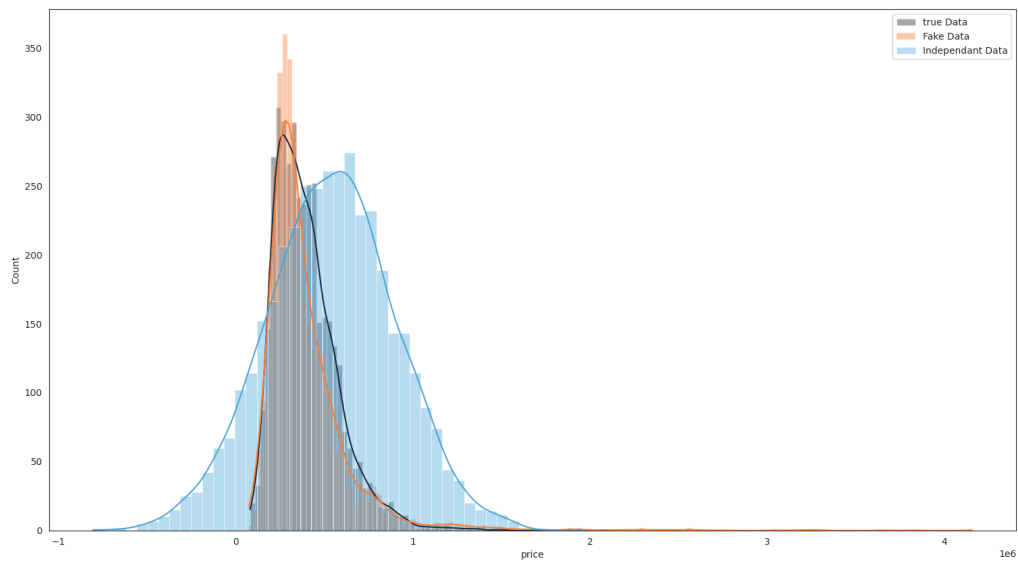


Fig. 134: Zoom on price distribution

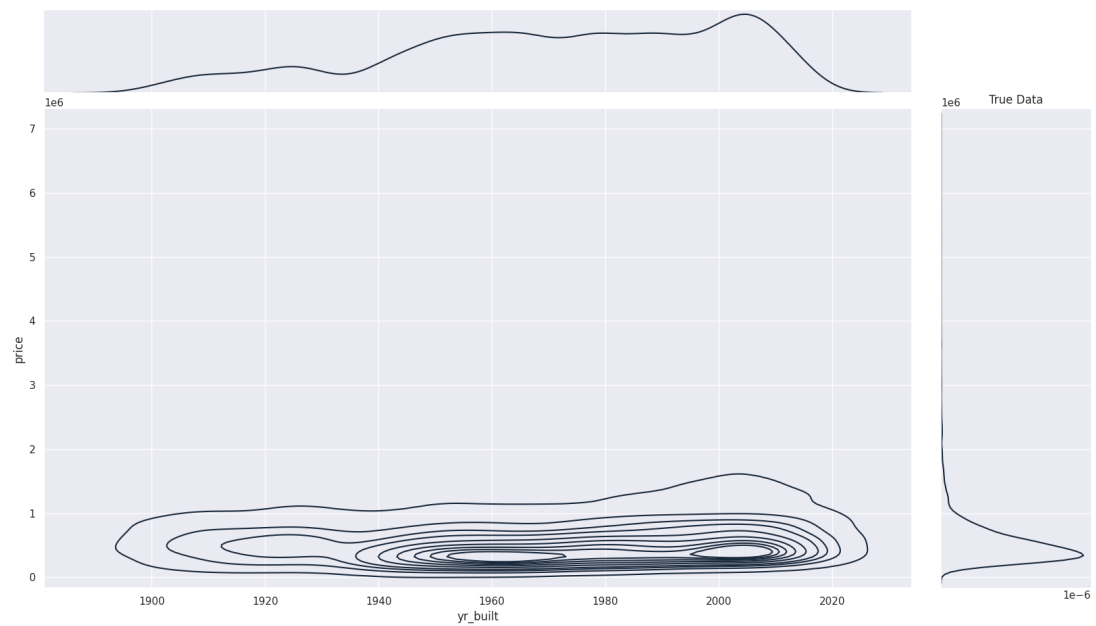


Fig. 135: Original true Data Joint distribution of prices and year constructed

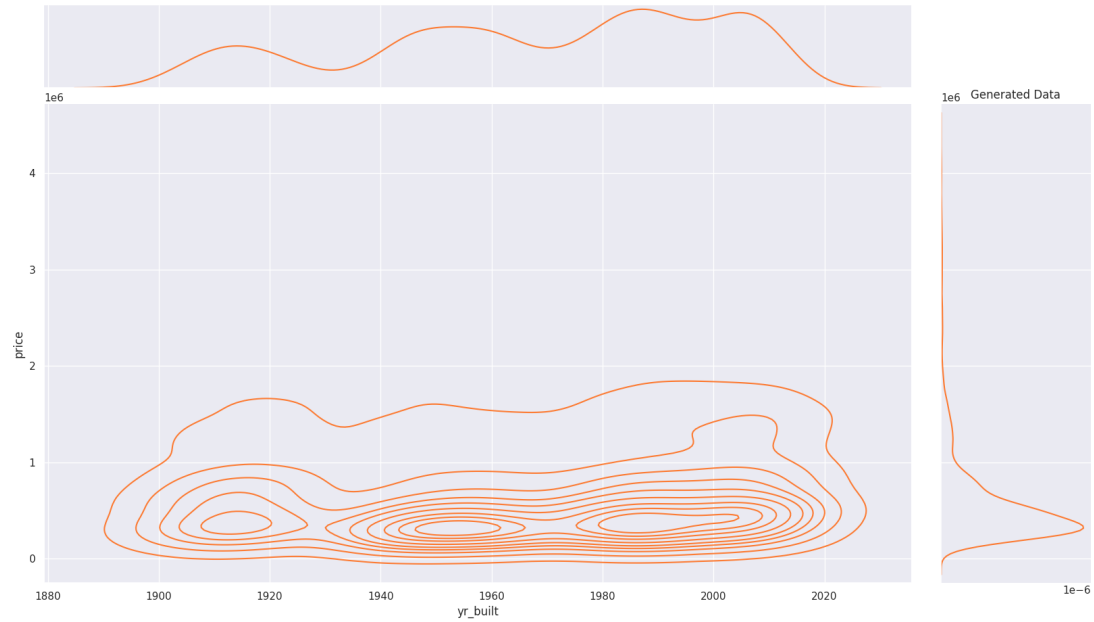


Fig. 136: Generated Data Joint distribution of same features. We see density start to reach those of real data

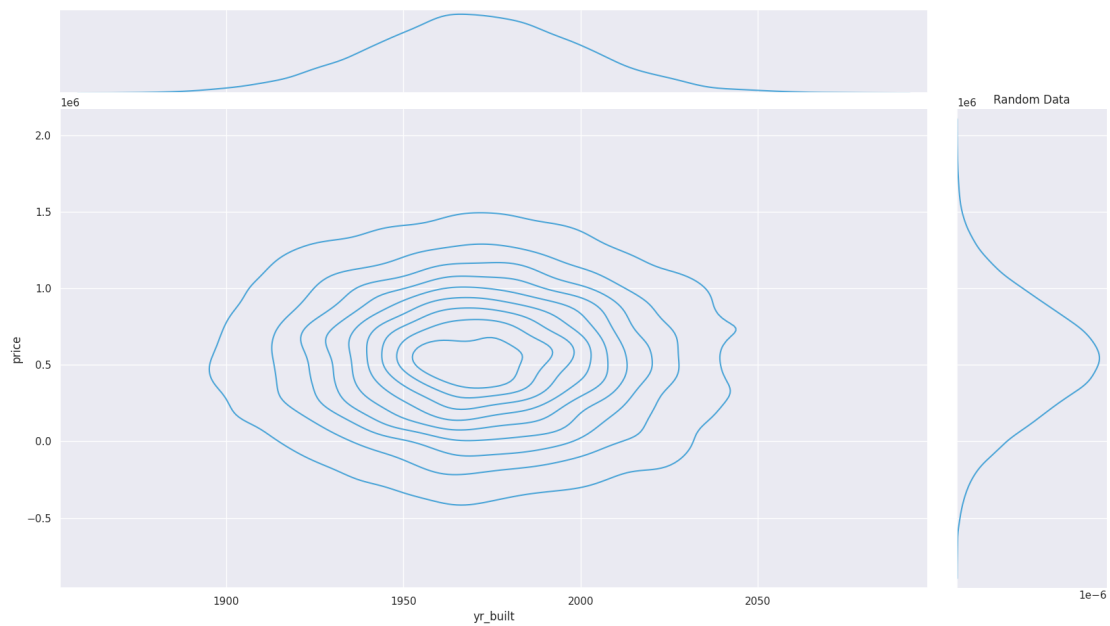


Fig. 137: Independant random variable distribution : jointplot density is totally wrong

as good as properly generated data :

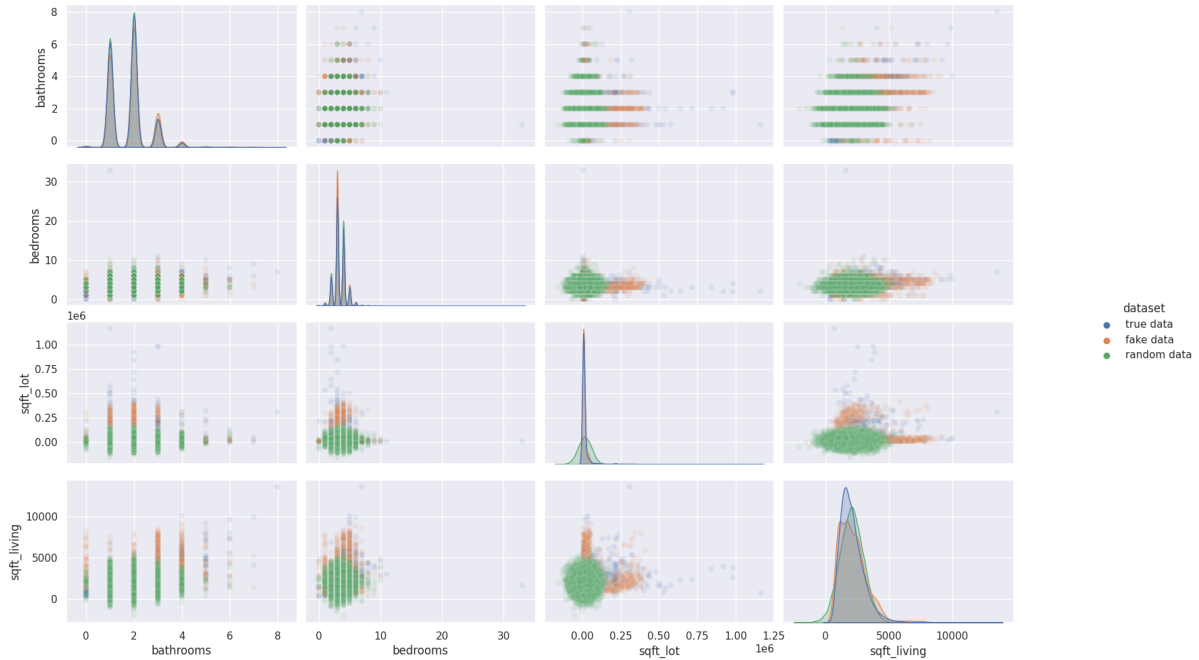


Fig. 138: each feature has its own distribution and distribution relative to others

Of course, better models than simple Normal distribution exist. You can build kernel density estimation, gaussian mixture model or use other distribution than a normal one yet doing so mean you start to use more and more complex model, till you 'll discover that going for Generative Adversarial Network is the way to go.

Generating good Synthetic Data

Like image before, Tabular data benefits from usage of Generative Adversarial Networks with Discriminator. In order to avoid the issue shown above, we can train a neural network on data whom input would be for each sampe a vector of :

- discrete probability for each mode of categorical variable
- probability of each mode of a gaussian mixture model

The network is then tasked to generate a “fake sample” output and a Discriminator then try to guess if this output comes from the Generator or from the the true datas.

By doing this, the Genelor will learn both distribution and joint distribution between each feature, converging to a generator that output sample “as true as the real one”.

The current best implementation of this idea is [CTGAN](#) that you can [use on our Marketplace](#)

Machine Learning efficiency

Of course, the best practical performance estimation of generated data is Machine Learning efficiency. We ran some performance test on [our automl platform](#) with the optimal parameters (all models tried, hyperoptimisation and Blending of model) on 3 datasets with the same target :

- Original dataset
- Synthetic dataset fitt on 800 epochs
- random Dataset with no joint probability

We expect that by using automl platform, only data quality would change the performance.

Here are the results :

Score table

Mean squared error	11,613,876,328
Root mean squared error	107,768
Mean absolute error	62,731
R2	100.00%
Mean absolute percentage error	12.06%

Fig. 139: Modelisation on true Data

Score table

Mean squared error	30,862,056,976
Root mean squared error	175,676
Mean absolute error	71,549
R2	99.99%
Mean absolute percentage error	16.10%

Fig. 140: Modelisation on Synthetic Data

We see that with CTGAN Synthetic Data, Machine Learning efficiency is preserved, which is not the case for independant data distribution. Moreover, this efficiency is good even for sgemnt of the original dataset, meaning you can generate synthetic samples for under represented category and keep their signal high enough to fix bias in your dataset.

Score table

Mean squared error	169,221,168,857
Root mean squared error	411,365
Mean absolute error	237,991
R2	99.99%
Mean absolute percentage error	43.46%

Fig. 141: Modelisation on random independant data

Going Further

You can read [the original paper for CTGAN](#) and get it [here](#).

The CTGAN implementation is available on [our Marketplace](#) and you can test it with one of [our public standard Dataset](#)

Remember that synthetic datas is very useful when you need to enforce data privacy or unbiase dataset. Training a Synthetic Data Generator for few hours build a generator that can be used to generate data in a few seconds and thus synthetic data should become part of your datascientist toolset too.

5.3.4.2 A list of some useful Dataset to explore Machine Learning

Tabular Dataset

Table 23: List of dataset

Number	Nom	Download link	Type	Industry	Target	Detail
1	churn	csv	Classification	B2B	churn	Churn usecase
2	Forecast Energy France trainset	csv	Regression	Energy	TARGET	Energy usecase
3	Forecast Energy France validation	csv	Regression	Energy	TARGET	Energy usecase
4	DNS Attacks Origins	csv	Multiclassification	Energy	Class	DNS usecase
5	Sales Forecasting	csv	Regression	Retail	Weekly_Sales	
6	Songs Hits	csv	Classification	Retail	target	
7	House pricing Regression - Trainset	csv	Regression	Retail	TARGET	House usecase
8	House pricing Regression - Holdout	csv	Regression	Retail	TARGET	House usecase
9	EDF Classification - Trainset	csv	Classification	Energy	TARGET	
10	EDF Classification - Holdout	csv	Classification	Energy	TARGET	
11	Sales Timeseries - Trainset	csv	Timeserie	Retail	Volume	
12	Sales Timeseries - Holdout	csv	Timeserie	Retail	Volume	

Images

Table 24: Images Folders

Number	Nom	Download link	Type	Industry
1	youtube train	You tube adds Trainset	Images	Ads
2	youtube train labels	You tube adds Trainset Labels	Images Labels	Ads
3	youtube test	You tube adds Testset	Images	Ads
4	youtube test Labels	You tube adds Testset Labels	Images Labels	Ads
5	Cheezam	Cheezam Images	Images	Ads
6	Cheezam Labels	Cheezam Labels	Images Labels	Ads

NLP

Table 25: NLP Folders

Num-ber	Nom	Download link	Type	Industry
1	Netflix catalog	Netflix movies with data	NLP	entertainment
2	French candidates multiclassif trainset sample	Tweets of French Presidential Candidates 2022 (train sample)	NLP	politics
3	French candidates multiclassif holdout	Tweets of French Presidential Candidates 2022 (holdout)	NLP	politics

Externals models

Table 26: Externals Models

Number	Nom	Download link	type	format
1	classification model	Files	Classification	onnx