



École Polytechnique de l'Université de Tours

64, Avenue Jean Portalis

37200 TOURS,

FRANCE

(33)2-47-36-14-14

www.polytech.univ-tours.fr

Projet libre

Mise en place de MLOps dans un projet existant de Machine Learning

Étudiant

Théo BOISSEAU

Encadrant

Nicolas RAGOT

Table des matières

I. Introduction	3
A. Objectifs.....	3
B. Organisation.....	3
II. Description générale.....	3
A. Environnement du projet.....	3
B. Structure générale du système	5
III. Mise en œuvre	6
A. Travail réalisé	6
IV. Analyse des résultats obtenus	12
RESSOURCES UTILES	19

I. Introduction

A. Objectifs

Ce projet dont le sujet a été librement choisi par l'étudiant vise à compléter son enseignement sur le Machine Learning abordé de façon universitaire lors de son cursus par une sensibilisation aux besoins spécifiques de l'industrie et du monde de l'entreprise.

L'étudiant doit donc s'approprier de nouvelles connaissances et de nouveaux outils sur l'MLOps et mûrir sa réflexion sur les enjeux de tels projets.

L'objectif principal de ce projet est de s'inspirer d'un projet de Machine Learning habituel en université pour proposer une démarche permettant de livrer rapidement et de façon fiable des modèles de Machine Learning en production à travers les concepts clés de l'MLOps : automatisation, collaboration, monitoring et évolutivité.

B. Organisation

Les exigences principales de ce projet ont été bien définies, claires et partagées en amont :

- Une phase **Data Science** lors de laquelle un projet de Data Science classique issu d'un Jupyter Notebook conçu par l'étudiant ou adapté d'une plateforme de concours de Machine Learning telle que Kaggle devra être validé et simplifié pour entraîner un réseau de neurones sur le dataset d'images de nombres MNIST.
- Une phase **Cloud** lors de laquelle le code de ce Notebook devra alors être repris pour être utilisé dans le Cloud sur Azure Machine Learning et découvrir la plateforme.
- Une phase **Production** lors de laquelle le modèle entraîné devra être déployé dans un environnement de production selon les bonnes pratiques de l'MLOps.

Il n'y a pas de chevauchement entre les phases, car le résultat d'une phase sert d'entrée à la phase suivante. Le projet n'étant pas complexe et les phases du développement étant claires, on se propose d'adopter une méthode de gestion de projet en cascade.

Le gestionnaire de configuration est GitHub et il contient tout le code source du projet ainsi que ses jeux de données dans un dépôt nommé *mlops-demo*.

Dans la suite de ce rapport, on se référera à Azure Machine Learning comme AML.

II. Description générale

A. Environnement du projet

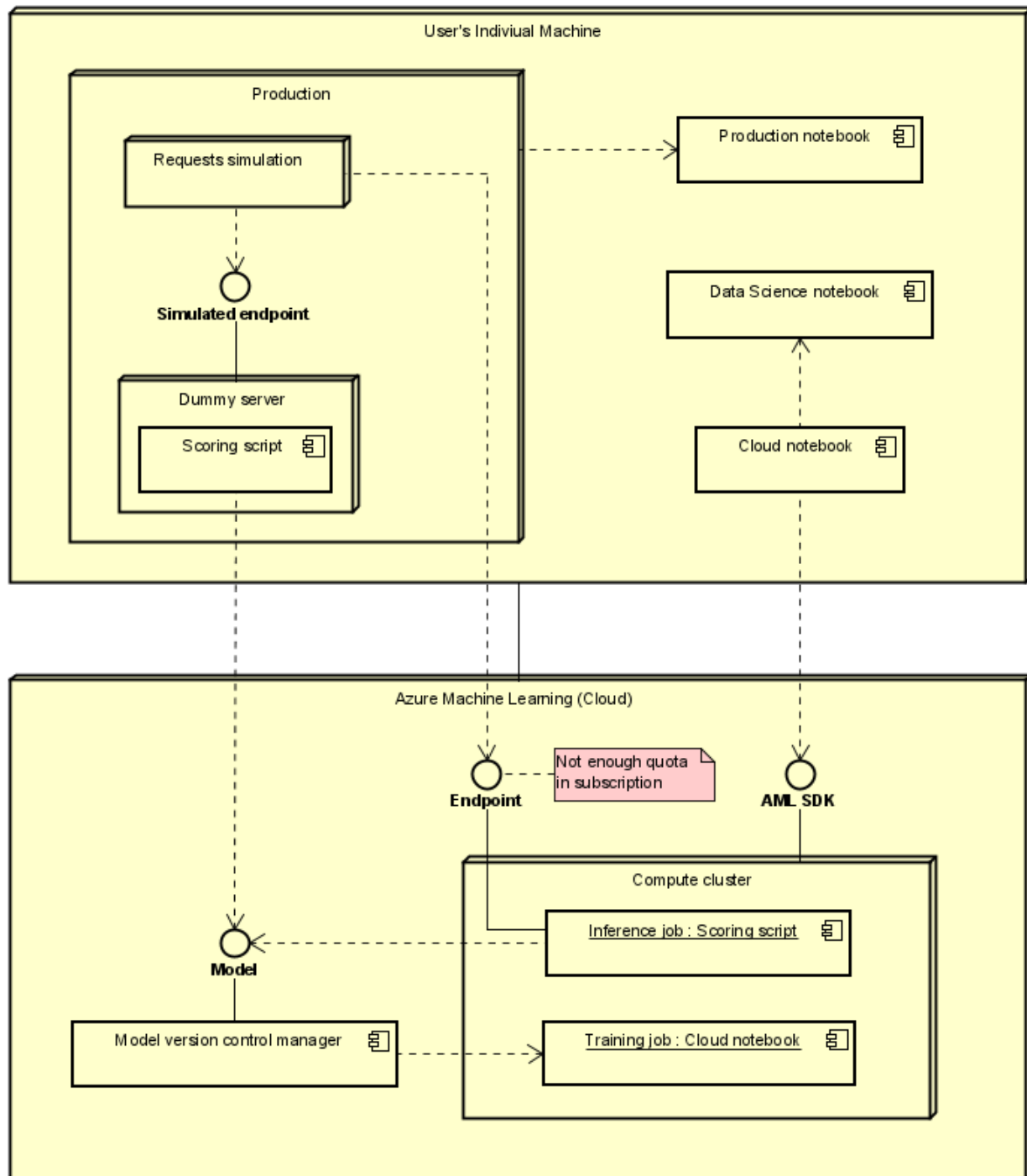
Le projet est développé intégralement en python mais chaque phase du projet est développée dans un environnement différent.

La phase de **Data Science** utilise Conda comme gestionnaire de librairies et développe le réseau de neurones à l'aide du Framework Keras. Il peut être utilisé sans connexion internet.

La phase **Cloud** utilise la plateforme en ligne AML mais l'utilisateur du projet est libre de configurer ses identifiants et d'installer le SDK2 d'AML pour travailler dans environnement Python quelconque tant qu'il dispose d'une connexion internet. Le code utilisé est tout de même dans le dépôt GitHub. Attention, l'utilisation d'AML engendre des coûts qui doivent être contrôlés notamment avec une gestion des données stockées et des machines virtuelles allouées dans le cloud par l'utilisateur.

La phase **Production** utilise le gestionnaire de packages Conda dans l'environnement spécifié par la sortie du job d'entraînement du modèle d'AML. Il peut être exécuté sans connexion internet.

B. Structure générale du système



Un projet de Data Science classique est mené en local. Celui-ci est adapté pour le Cloud et est destiné à lancer un *job d'entraînement*, soit l'entraînement d'un nouveau modèle, sur un cluster de calcul (une VM) d'Azure. Le modèle produit à l'issue du *job d'entraînement* est alors sérialisé avec son environnement et ses métadonnées dans Azure.

Ce modèle peut ainsi être récupéré, monté dans l'environnement spécifié par le *job d'entraînement* puis utilisé pour l'inférence. Pour déployer le modèle, un *job d'inférence* peut alors être lancé dans un cluster de calcul et un *endpoint* (associé à un URL) y est attribué. Celui-ci fonctionne avec un *scoring script* qui, à partir de la requête reçue sur l'*endpoint*, la traite, la fait passer par le modèle puis renvoie sa réponse.

III. Mise en œuvre

A. Travail réalisé

```
mlops-demo
├─ data_science/                # code pour la phase de data science
│   ├── input/                  # données d'entrée
│   │   ├── test.csv           # données de test
│   │   └─ train.csv           # données d'entraînement
│   └─ working/                 # code de travail
│       ├── data_science_digits_model.ipynb # carnet Jupyter du modèle de data science
│       └─ main.py              # script principal
└─ cloud_AML_digits_model.ipynb # carnet Jupyter pour la phase de cloud

├─ production/                  # code pour la phase de production
│   ├── foreign_data/           # données externes
│   │   ├── test.csv           # données de test
│   │   └─ train.csv           # données d'entraînement
│   └─ model/                   # modèle entraîné issu d'Azure Machine Learning
│       ├── data/               # données de modèle
│       │   └─ model/           # modèle sérialisé
│       │       ├── variables/  # variables du modèle
│       │       ├── keras_metadata.pb # fichier de métadonnées Keras
│       │       └─ saved_model.pb # fichier de modèle sauvegardé
│       ├── keras_module.txt    # fichier texte de module Keras
│       └─ save_format.txt      # fichier texte du format de sauvegarde
│   ├── MLmodel                 # modèle MLFlow
│   ├── _summary.txt            # résumé du modèle
│   ├── conda.yaml              # environnement Conda
│   ├── python_env.yaml         # environnement Python
│   └─ requirements.txt         # dépendances Python
│   └─ working/                 # code de travail
│       ├── data_simulation.py  # script de simulation de données
│       ├── dummy_server.py     # script du serveur factice
│       ├── score.py            # script de scoring
│       └─ visualization.py     # script de visualisation
└─ production.ipynb            # carnet Jupyter pour la phase de production

├─ .gitignore                  # fichiers à ignorer lors de la confirmation dans Git
└─ README.md
```

Pour illustrer la suite du rapport, on se propose de présenter brièvement quels éléments appartiennent à quelles phases du projet à partir de la structure du dépôt GitHub.

C'est dans le dossier *data_science* qu'est présent le code des phases **Data Science** et **Cloud**. Le dossier *input* contient les données du dataset MNIST et le dossier *working* contient le Notebook d'un projet classique de Data Science adapté aux besoins du projet. Le Notebook *cloud_AML_digits_model.ipynb* contient les instructions en python destinées à AML et le fichier *main.py* dans le dossier *working* correspond au code qu'il a généré et déployé sur la plateforme. Pour le cloud, tout le dossier *data_science* doit être envoyé sur AML.

Le dossier *production* contient dans le sous-dossier *foreign_data* deux datasets étrangers au dataset MNIST occidental : un de nombres écrits par des indiens et des népalais, et un autre écrit par des japonais. Il contient également la sortie du job d'entraînement d'AML : un artifact *model* contenant l'environnement nécessaire pour faire fonctionner le modèle, le modèle sérialisé et des méta-données. Le fichier *production.ipynb* est un Notebook dans lequel est simulé un environnement de production et qui présente un exemple de déploiement du modèle. Dans *working* est le fichier *score.py* (un script de scoring) utilisé pour l'inférence et généré par le Notebook *production.ipynb*, ainsi que le reste du code utilisé pour simuler la production.

Le contenu de la phase Data Science étant très classique et surtout hors-sujet, on se propose d'aborder dès à présent la phase **Cloud**.

Le notebook *cloud_AML_digits_model.ipynb* s'identifie auprès d'AML, crée un cluster de calcul, un environnement virtuel, un script de scoring (automatiquement généré) et enfin un job d'entraînement lors duquel on crée les sources de données. Il est important de noter que chaque élément précédemment créé est enregistré et versionné dans la plateforme, ceci afin de favoriser l'automatisation et la collaboration qui sont deux concepts importants d'MLOps.

Toutes ces étapes peuvent également être données et/ou suivies sans code via l'interface web. Chaque étape est respectivement effectuée dans la plateforme web :

Compute

Compute instances		Compute clusters		Kubernetes clusters	Attached computes	
+ New	Refresh	Delete	Edit columns	Reset view	View quota	
Search		State		Location	All filters	Clear all
Name	☆	State	Size	Location	Created on ↓	Active runs
cpu-cluster		✓ Succeeded (0 nodes)	STANDARD_DS1_V2	francecentral	Apr 2, 2023 4:56 PM	0

Gestion/création d'un cluster de calcul dans la partie *Manage* de AML

Environments

Curated environments **Custom environments**

Custom environments are user defined environments created from a Docker image, a Docker build context, and a conda specification with Docker image. [Learn more about custom environments](#)

+ Create Refresh Archive Edit columns Reset view | ☒ Show latest version only ☐ Include archived

Search

Tags Registry All filters Clear all

Showing 1-1 of 1 environments Page size: 25

Name	Source	Version	Created ↓	Created by	Tags
cv_dli:35	This workspace	35	Mar 24, 2023 11:12 PM	Theo Boisseau	tensorflow : 2.6.0

Enregistrement d'un environnement virtuel dans la partie Assets de AML

Data

Data assets Datastores Dataset monitors

Data assets are references to your data. You can create data assets from datastores, local files, public URLs, or Open Datasets. Data assets can be versioned and easily referenced and reused for machine learning tasks. [Learn more about data assets](#)

+ Create Refresh Archive Edit columns Reset view | ☒ Show latest version only ☐ Include archived

Search

Registry All filters Clear all

Showing 1-2 of 2 data assets Page size: 25

Name	☆	Source	Version	Created on ↓	Modified on	Type	Properties	Created by
MNIST-test		This workspace	1	Mar 25, 2023 3:09 PM	Mar 25, 2023 3:09 PM	File		Theo Boisseau
MNIST-train		This workspace	1	Mar 25, 2023 3:09 PM	Mar 25, 2023 3:09 PM	File		Theo Boisseau

Enregistrement de sources de données dans la partie Assets de AML

Jobs

All experiments **All jobs** All schedules

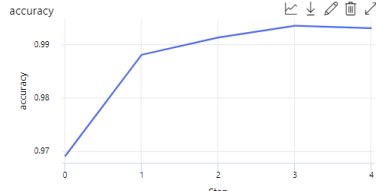
+ Create job (preview) Add chart Refresh Compare (preview) Edit columns Cancel Delete | Current view: default Save view Edit view Share view

Search

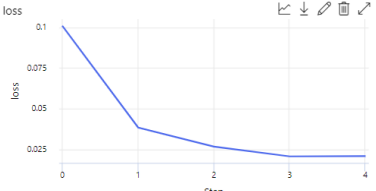
Status Created by ☒ Include child jobs ☐ View only my jobs All filters Clear all

Running: 0 Completed: 1 Failed: 0 Canceled: 0 Queued: 0 Other: 0

accuracy



loss

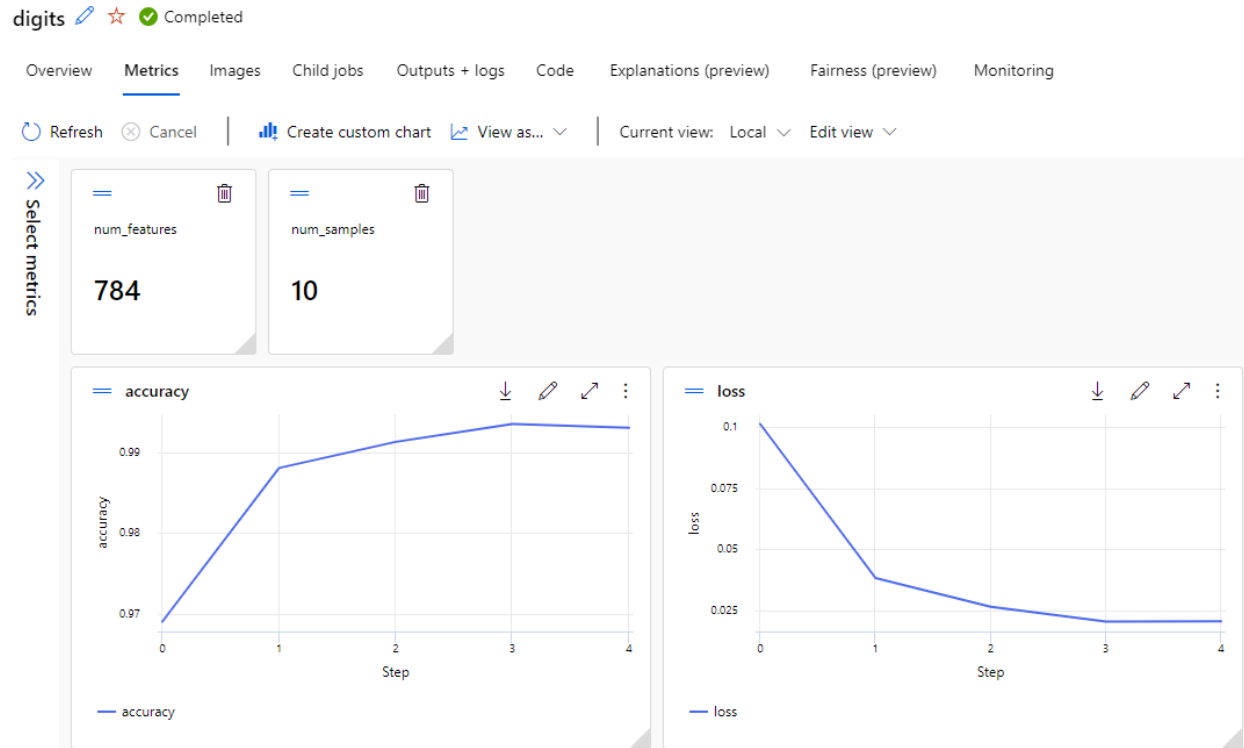


Show only selected rows (0 selected) Page Size: 25

Display name	☆	Experiment	Status	Created on ↓	Duration	Created by	Compute target	Job type	Last(loss)	Last(accura...)	Tags
digits		train_model_mnist	Completed	Mar 24, 2023 11:12 PM	7m 7s	Theo Boisseau	cpu-cluster	Command	0.02073022	0.9930714	

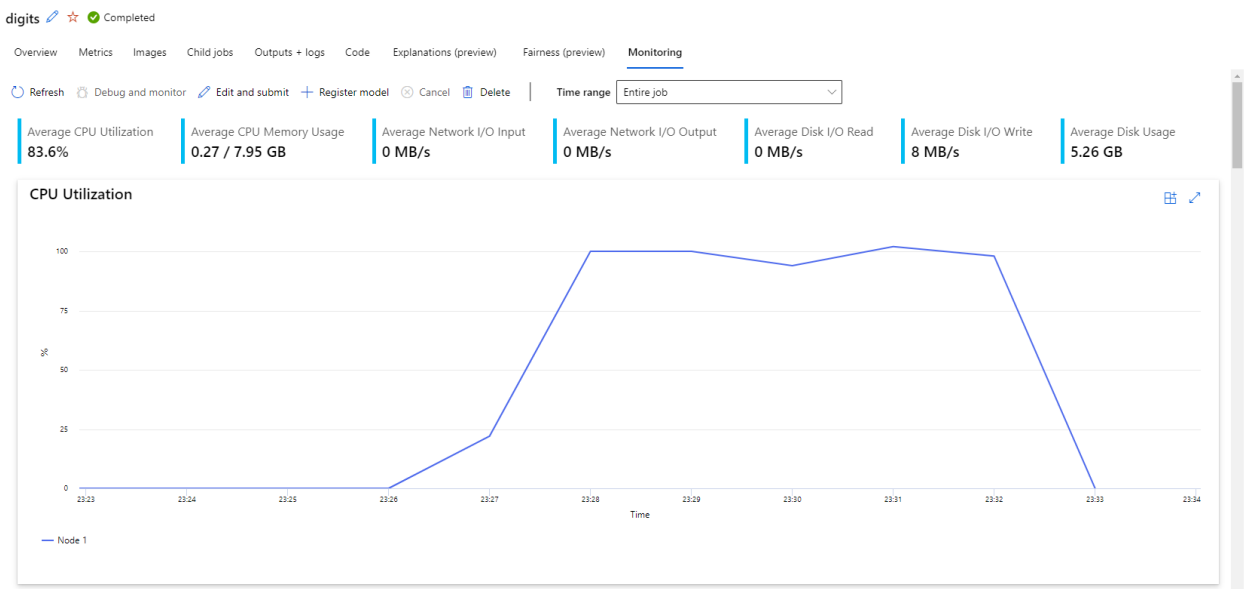
Création d'un job d'entraînement dans la partie Assets de AML

Une fois le notebook cloud_AML_digits_model.ipynb exécuté sur AML, le job d'entraînement est lancé et on peut en visualiser les métriques sur le panel suivant :



Panneau *Metrics* du job d'entraînement AML d'un réseau de neurones sur le dataset MNIST

De plus, on peut obtenir de nombreuses informations en ce qui concerne le monitoring (le panneau étant déroulant):



Panneau Monitoring d'un job d'entraînement AML d'un réseau de neurones sur le dataset MNIST

Enfin, on peut surtout obtenir l'artifact du modèle : le modèle sérialisé, son environnement python et conda, ainsi que ses métadonnées, tous sauvegardés par MLFlow:

digits Completed

Overview Metrics Images Child jobs **Outputs + logs** Code Explanations (preview) Fairness (preview) Monitoring

Refresh Debug and monitor Edit and submit Register model Cancel Delete | Download all Enable log streaming Word wrap

<<

MLmodel

>>

azureml-logs

model

data

conda.yaml

MLmodel

python_env.yaml

requirements.txt

system_logs

tensorboard_logs

user_logs

model_summary.txt

```
1 artifact_path: model
2 flavors:
3   keras:
4     code: null
5     data: data
6     keras_module: tensorflow.keras
7     keras_version: 2.6.0
8     save_format: tf
9   python_function:
10    data: data
11    env: conda.yaml
12    loader_module: mlflow.keras
13    python_version: 3.8.15
14 mlflow_version: 1.26.1
15 model_uuid: 8494565dd6e54e4ea8e7ff14c61f92a6
16 run_id: khaki_parsnip_0nh4r3mgw2
17 utc_time_created: '2023-03-24 22:31:34.472117'
```

Panneau *Outputs + logs* d'un job d'entraînement AML d'un réseau de neurones sur le dataset MNIST

Toujours dans le panneau *Outputs + logs*, on peut enregistrer le modèle à l'aide du bouton *+ Register model*.

Ceci nous amène donc à la partie **Production**. Afin de simuler un environnement de production, on se propose de créer un dataset unique, composé du dataset MNIST, mais également des datasets étrangers tels que *Hindi/Devanagari MNIST-like dataset* et *Japanese Handwritten Digits (JHD) dataset*. L'objectif de cette partie du projet est de distribuer les éléments de chaque dataset en fonction de différentes lois de probabilité pour simuler des scénarios issus des concepts de data-drift.

Afin de déployer notre modèle en production, il nous faut créer un *endpoint* capable de récupérer une requête.

Pour cela, AML permet d'en créer un à partir d'un modèle enregistré. Il faut alors allouer une nouvelle machine virtuelle capable d'héberger le modèle, son environnement et son *scoring script* qui pilote la gestion des requêtes :

Deploy CNN-Digits:1



For the selected model, the scoring script and environment are auto generated for you.

[Learn More](#)

Virtual machine *

i 'Standard_DS1_v2' and 'Standard_F2s_v2' may be too small for bigger models and may lead to container termination due to insufficient memory, not enough space on the disk, or probe failure as it takes too long to initialize the container. [Learn more](#)

Standard_F2s_v2 2 Cores, 4 GB (RAM), 16 GB (Disk), \$0.10/hr

Instance count *

2

Endpoint

☒ New ☐ Existing

Endpoint name *

mlops-brndx

i An endpoint URL will be generated after creating an endpoint.

<https://mlops-brndx.francecentral.inference.ml.azure.com/score>

[Learn how to consume](#)

Deployment name *

cnn-digits-1

Deploy

Cancel

[More options](#)

Interface de déploiement d'un modèle de réseau de neurones dans le panneau Endpoints d'AML

Cependant, la souscription AML pour les étudiants *Azure for Students* ne permet pas d'allouer une machine virtuelle pour un endpoint. Comme indiqué sur le commentaire de l'interface, l'allocation de la VM échoue lors du déploiement. On essaye alors avec toutes les autres VMs plus grosses :

Virtual machine *

i 'Standard_DS1_v2' and 'Standard_F2s_v2' may be too small for bigger models and may lead to container termination due to insufficient memory, not enough space on the disk, or probe failure as it takes too long to initialize the container. [Learn more](#)

Standard_DS1_v2 1 Core, 3.5 GB (RAM), 7 GB (Disk), \$0.09/hr

Not enough quota available for Standard_DS1_v2. Current usage/limit: 0/1. Additional requested: 3 (20% more quota than expected may be needed for redundancy purposes)

Virtual machine * ⓘ

Standard_F4s_v2 4 Cores, 8 GB (RAM), 32 GB (Disk), \$0.20/hr

Not enough quota available for Standard_F4s_v2. Current usage/limit: 0/8. Additional requested: 12 (20% more quota than expected may be needed for redundancy purposes)

Virtual machine * ⓘ

Standard_F8s_v2 8 Cores, 16 GB (RAM), 64 GB (Disk), \$0.40/hr

Not enough quota available for Standard_F8s_v2. Current usage/limit: 0/8. Additional requested: 24 (20% more quota than expected may be needed for redundancy purposes)

Ainsi, on se propose de simuler notre propre *endpoint* en créant un serveur virtuel et un endpoint factice à l'aide de la classe `DummyServer` et du *scoring script* `score.py`.

Le *scoring script* récupère le contenu d'une requête au format JSON, prétraite les données et les fait passer par le modèle. Une fois cela fait, il sérialise à l'aide d'MLFlow les métriques et les artifacts qui intéressent le développeur. Pour nous, il s'agit de la donnée d'entrée, la probabilité de chaque classe, la prédiction et la probabilité de la prédiction si elle obtient une valeur anormalement haute ou basse.

Ceci fait, on peut donc créer nos données de productions à partir du dataset MNIST classique et des datasets orientaux.

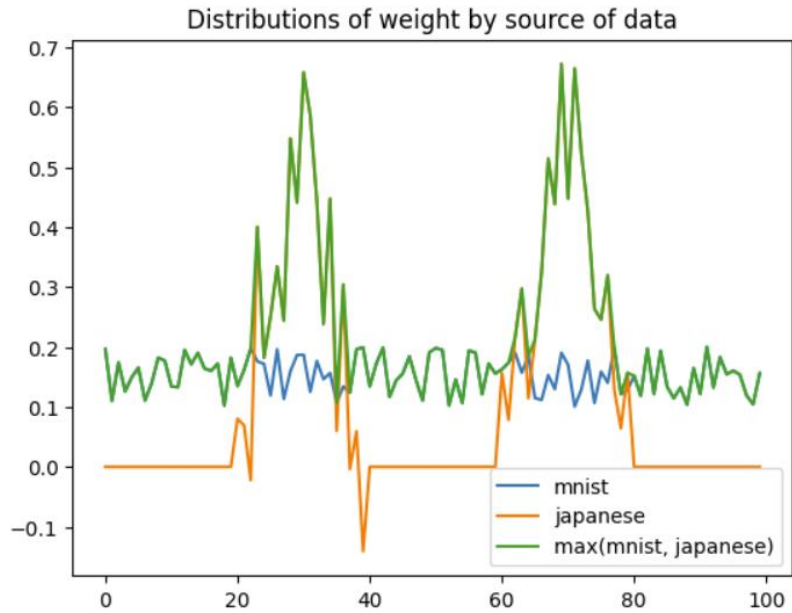
On constitue un dataset de 100 images, en imaginant que 1 par jour peut être envoyée à notre serveur, et que l'utilisateur ayant le poids le plus important est servi.

On suppose que les utilisateurs du dataset MNIST constituent les utilisateurs habituels et que leur poids fluctue relativement peu. Leur poids s'apparente à $\frac{1}{10} + \varepsilon$.

On se propose de créer une simulation de la production dans un scénario de changement saisonnier. On suppose que les utilisateurs du dataset japonais constituent des utilisateurs saisonnier et donc que leur poids est nul l'essentiel du temps, sauf 2 fois par an lors desquelles il devient majoritaire. Leur poids est une fonction continue par morceaux qui s'apparente à $\sin(x) + \varepsilon / x \in [0, \pi]$ entre 20 et 40 et entre 60 et 80, et $0 + \varepsilon$ le reste du temps.

On se propose également de créer une simulation de la production dans un scénario de changement dans les conditions externes. On suppose que les utilisateurs du dataset indiens et népalais constituent des utilisateurs de plus en plus importants au cours du temps. Leur poids est une fonction continue par morceaux qui s'apparente à $-x^2 + 1.68 - \frac{1}{2} / x \in [0, 1]$.

IV. Analyse des résultats obtenus



Après avoir lancé la simulation du scénario de changement saisonnier dans le notebook `production.ipynb`, on peut regarder les résultats de l'inférence dans MLFlow :

inference61636 [Provide Feedback](#) [Share](#)

Experiment ID: 602687763296459017 Artifact Location: file:///C:/Users/boiss/workspaces/Conda/MLOps/production/mlruns/602687763296459017

> Description [Edit](#)

Table view | Chart view | | Sort: Created | Refresh

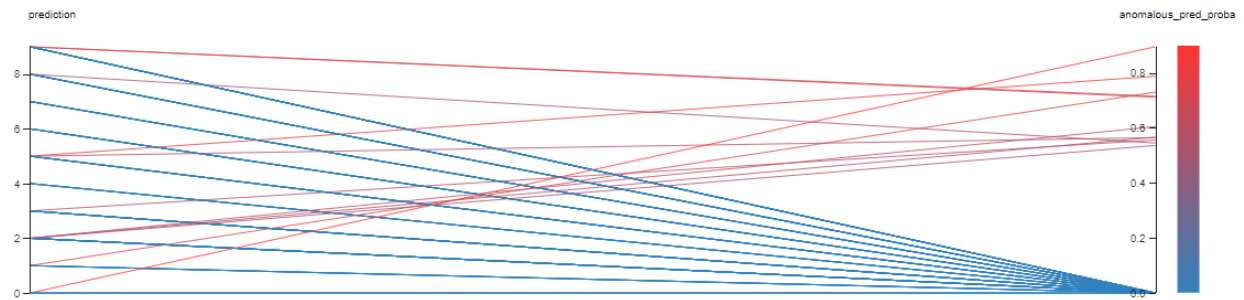
Columns | Time created: All time | State: Active

				Metrics					
<input type="checkbox"/>	<input type="checkbox"/>	Run Name	Created	Duration	anomalous_prec	prediction	probability_class_0	probability_class_1	probability
<input type="checkbox"/>	<input type="checkbox"/>	victorious-lamb-142	1 minute ago	127ms	-	9	7.321e-9	6.450e-7	3.220e-1
<input type="checkbox"/>	<input type="checkbox"/>	capricious-gull-410	1 minute ago	126ms	0.556	3	2.110e-4	0.013	0.272
<input type="checkbox"/>	<input type="checkbox"/>	classy-crane-126	1 minute ago	128ms	-	2	3.471e-6	7.955e-4	0.993
<input type="checkbox"/>	<input type="checkbox"/>	nosy-owl-643	1 minute ago	129ms	-	8	2.778e-8	4.305e-10	6.685e-1
<input type="checkbox"/>	<input type="checkbox"/>	inquisitive-fly-452	1 minute ago	122ms	-	6	7.575e-10	1.765e-11	1.628e-1
<input type="checkbox"/>	<input type="checkbox"/>	traveling-hen-759	1 minute ago	118ms	-	2	2.700e-7	7.200e-7	0.998
<input type="checkbox"/>	<input type="checkbox"/>	powerful-zebra-649	1 minute ago	124ms	-	9	3.934e-10	3.217e-11	5.260e-1
<input type="checkbox"/>	<input type="checkbox"/>	flawless-moth-107	1 minute ago	136ms	0.545	8	4.418e-8	4.222e-5	1.291e-1

100 matching runs

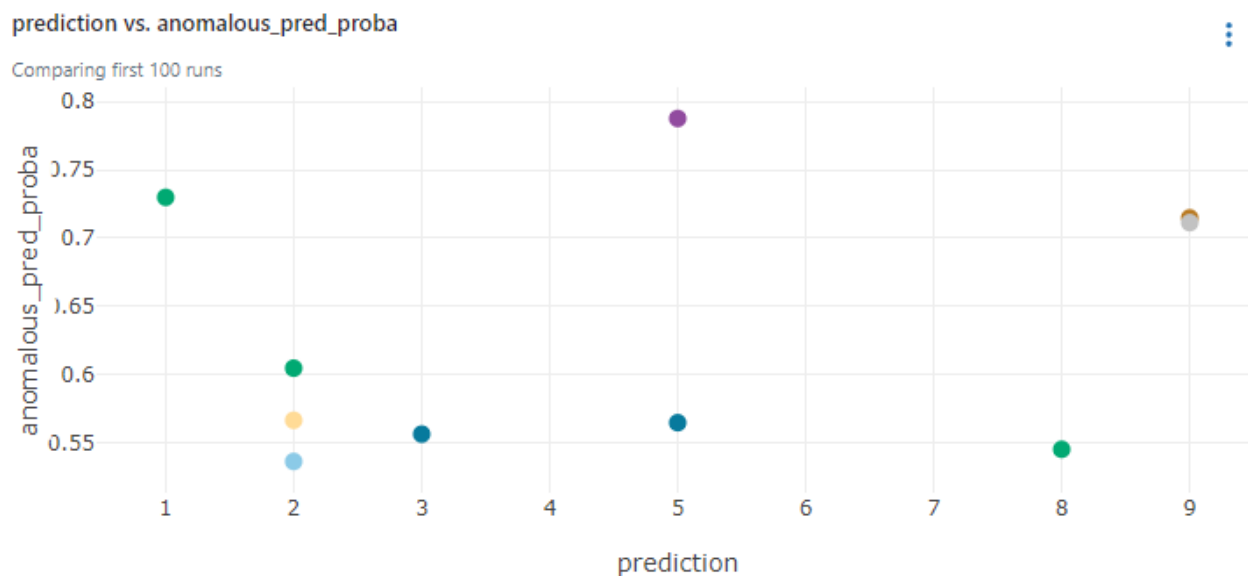
Vue tabulaire de l'interface MLFlow

On peut ensuite créer des graphiques sur ces résultats :



Coordonnées parallèles entre les prédictions et les prédictions anormales (proba < 0.9)

Les lignes bleues indiquent tous les runs d'inférence qui ont une valeur None de probabilité de prédiction anormale. En regardant les lignes rouges, on voit que certaines classes prédites ont plus de prédictions anormales.



Probabilité de prédictions anormales en fonction de la classe prédite

Ici, on peut plus clairement identifier quelles classes ont posé problème et à quel point. Plus la probabilité est faible, moins le modèle était sûr de lui.

On peut alors sélectionner un point tel que (2, 0.536074) et observer son artifact :

inference61636 >

clumsy-yak-128

Run ID: ce1e1e8a081f459780dcc354d8b08fde Date: 2023-04-02 22:20:04 Source: C:\Users\boiss\anaconda3\envs\mlops-model-env\lib\site-packages\ipykernel_launcher.py User: boiss

Status: FINISHED Lifecycle Stage: active

- > Description [Edit](#)
- > Parameters
- > Metrics (13)
- > Tags
- > Artifacts

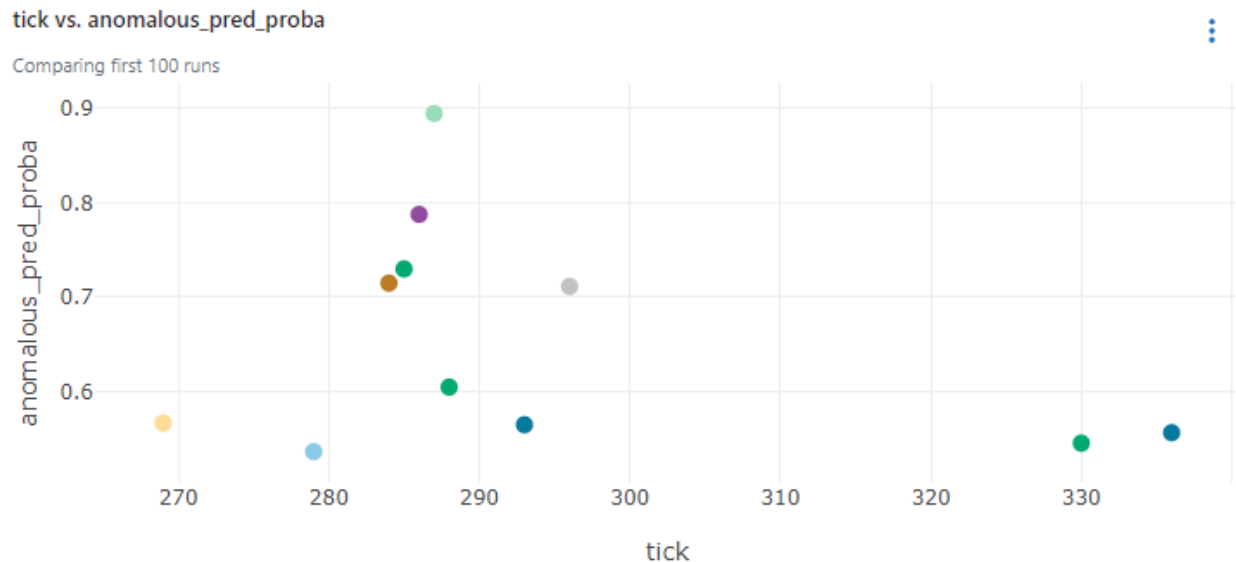
best_pred.txt
 data.png

Full Path: file:///C:/Users/boiss/workspaces/Conda/MLops/production/mlruns/602687763296459017/ce1e1e8a081f459780dcc354d8b08fde/artifacts/data.png

 Size: 166B

Donnée d'entrée de l'inférence ayant eu une prédiction anormale

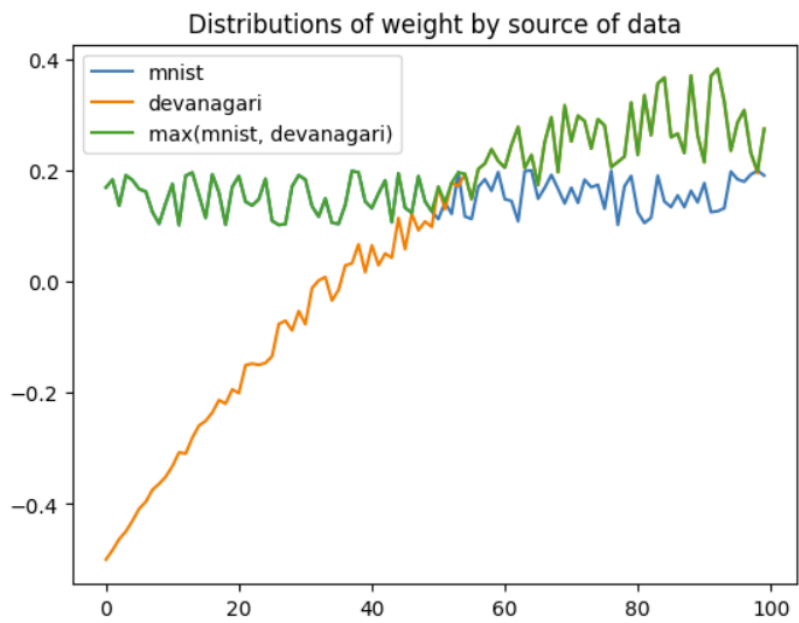
On constate alors que le nombre n'est clairement pas issu de MNIST et est sans doute plutôt indien ou népalais.



Probabilité de prédiction anormale en fonction du temps

En regardant les probabilités de prédictions anormales en fonction du temps, on peut alors comprendre que les zones qui concentrent le plus de prédictions anormales sont celles où nous avons eu les changements saisonniers.

Après avoir lancé la simulation du scénario de changement saisonnier dans le notebook production.ipynb, on peut regarder les résultats de l’inférence dans MLFlow :



Après avoir lancé la simulation du scénario de changement dans les conditions externes dans le notebook production.ipynb, on peut regarder les résultats de l’inférence dans MLFlow :

inference16421

Provide Feedback

Share

Experiment ID: 942678829923063288

Artifact Location: file:///C:/Users/boiss/workspaces/Conda/MLOps/production/mlruns/942678829923063288

> Description

Edit

Table view

Chart view

metrics.rmse < 1 and params.model = "tree"

Sort: Created

Refresh

Columns

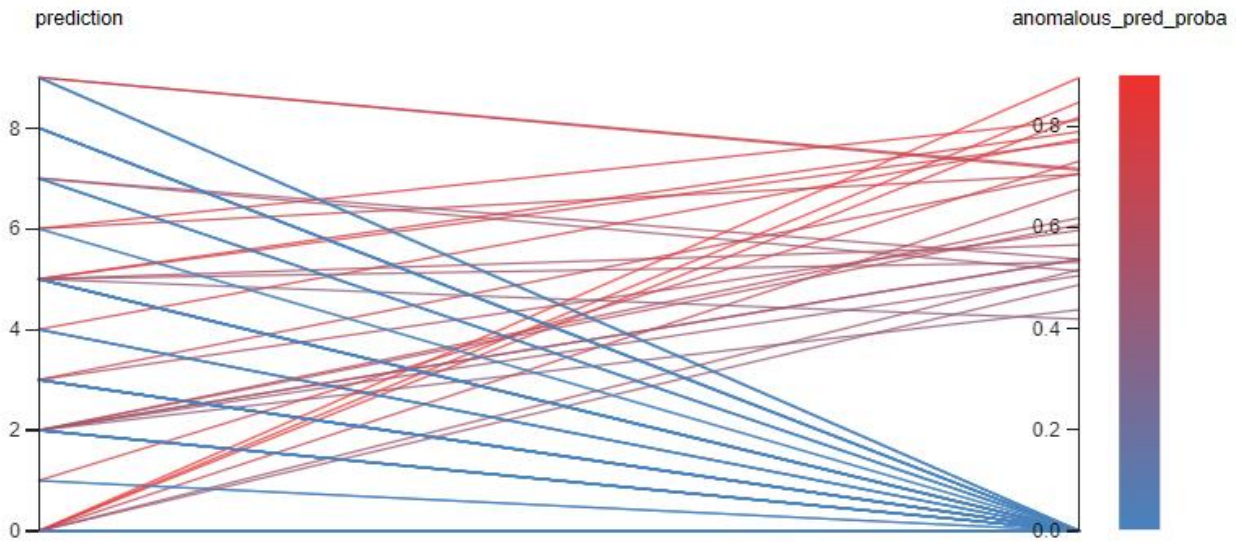
Time created: All time

State: Active

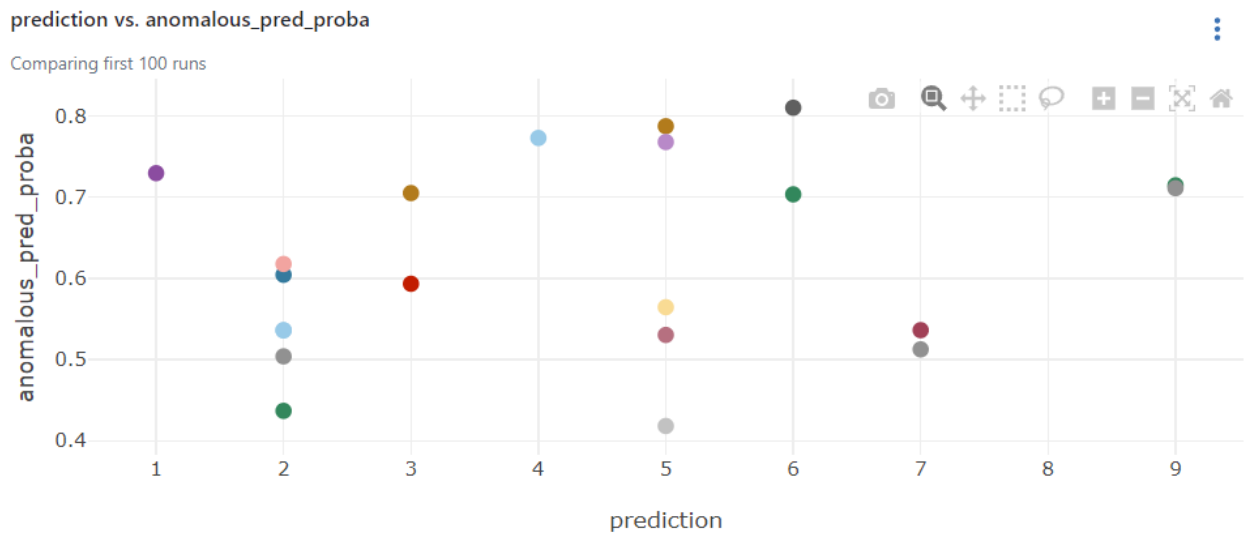
Metrics									
Run Name	Created	Duration	prediction	probability_class_0	probability_class_1	probability_class_2	probability_class_3	probability_class_4	probability_class_5
funny-snipe-200	8 seconds ago	159ms	7	1.344e-7	2.910e-6	0.002	6.804e-4		
skillful-hare-747	8 seconds ago	154ms	2	4.011e-4	2.692e-6	0.618	0.109		
peaceful-hound-826	8 seconds ago	156ms	5	1.295e-11	2.832e-10	1.198e-15	2.408e-5		
efficient-kit-357	9 seconds ago	153ms	0	1	5.165e-7	2.996e-6	2.942e-6		
charming-dog-932	9 seconds ago	154ms	3	7.195e-7	1.086e-10	2.054e-6	1		
clumsy-moose-323	9 seconds ago	168ms	5	1.778e-4	1.821e-6	3.415e-5	6.154e-4		
resilient-cow-458	9 seconds ago	168ms	0	0.817	4.707e-4	2.753e-6	1.961e-6		
magnificent-hare-172	9 seconds ago	167ms	2	4.099e-13	2.956e-11	1	8.165e-6		

100 matching runs

Il suffit alors d’appliquer les mêmes graphiques, tout simplement :



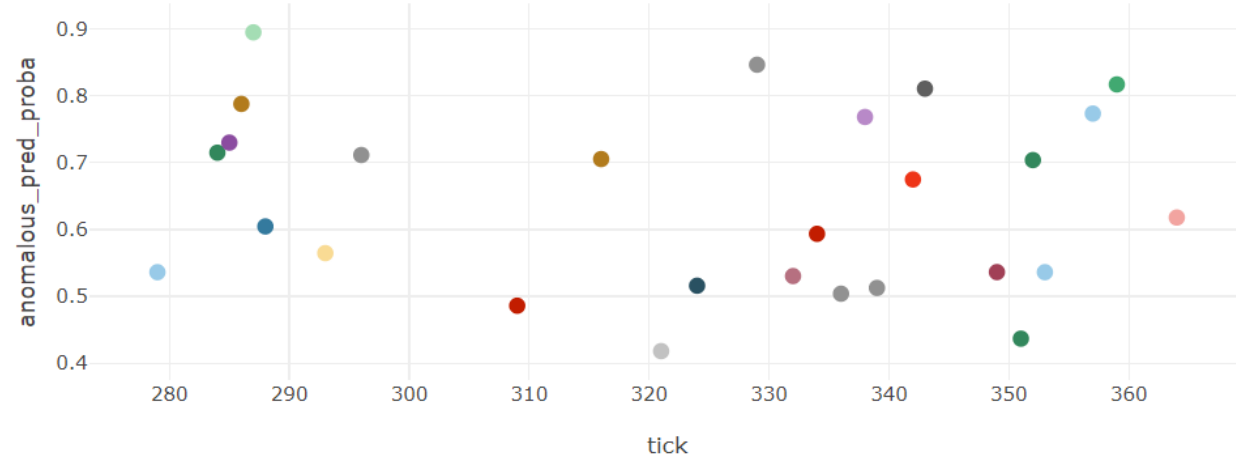
Coordonnées parallèles entre les prédictions et les prédictions anormales (proba < 0.9)



Probabilité de prédictions anormales en fonction de la classe prédite

tick vs. anomalous_pred_proba

Comparing first 100 runs



Probabilité de prédiction anormale en fonction du temps

C'est alors en fonction du temps que l'on comprend qu'on est face à un changement dans les conditions externes.

RESSOURCES UTILES

- Guide d'MLOps (ci-joint)
- Dépôt GitHub: <https://github.com/theo-boi/mlops-demo>
- Notebook original de Data Science : <https://www.kaggle.com/code/poonaml/deep-neural-network-keras-way/>
- MNIST dataset : <https://www.kaggle.com/datasets/hojjatk/mnist-dataset>
- Hindi/Devanagari MNIST-like dataset : <https://www.kaggle.com/datasets/anurags397/hindi-mnist-data>
- Japanese Handwritten Digits (JHD) dataset : <https://www.kaggle.com/datasets/anwarzalek/japanese-handwritten-digits>
- Azure infrastructure for Machine Learning : <https://www.youtube.com/watch?v=BJxMHYz9hiQ>
- Azure model deployment using Machine Learning Studio : <https://www.youtube.com/watch?v=JvZtiKKL7Go>