orange™ Business Services

SIMPLON.CO

Projet chef-d'œuvre - Ref. E1

Détection d'anomalies sur le réseau VoiP d'Orange par un modèle de réseau de neurones

Titre professionnel "Développeur en intelligence artificielle"
de niveau 6 enregistré au RNCP sous le n°34757

Passage par la voie de la formation - parcours de 19 mois achevé le 20 octobre 2020

Tommaso Signori
Décembre 2020

INDEX

1. Introduction

Ce projet et le mémoire qui décrit ses étapes, sont issus de problèmes réels survenus lors de la réalisation d'un des projets sur lesquels j'ai travaillé pendant la période d'alternance chez Orange.

Le réseau mondial VoiP d'Orange a besoin d'une solution pour surveiller toutes ses ressources et identifier celles qui présentent un comportement anormal.

Le monitoring actuel de ces services est surtout réactif, basé sur l'ouverture de tickets d'incident qui induisent l'insatisfaction des clients finaux. Le client peut chercher un problème à l'intérieur de son périmètre pendant 30 minutes ou plus avant d'ouvrir le ticket.

Le projet a pour objectif primaire l'étude du processus de restauration, et en particulier l'utilisation des techniques d'analyse de données et d’IA dans le but d'automatiser et de consolider le processus de détection d’anomalies et de résolution de défaillances dans les réseaux.

Grâce à l'IA, nous pouvons viser une surveillance en temps réel : certaines pannes peuvent être détectées de manière proactive avant que le client n'ouvre un ticket, avec des données techniques utiles pour les équipes opérationnelles de support et pour les clients finaux.

À ce jour, le projet est en phase de test avec une solution qui repose sur une première phase de clustering et une deuxième phase de modélisation : un modèle de forêt aléatoire apprend le comportement habituel des différents clusters de ressources et nous fournit une prévision des leurs comportements futurs.

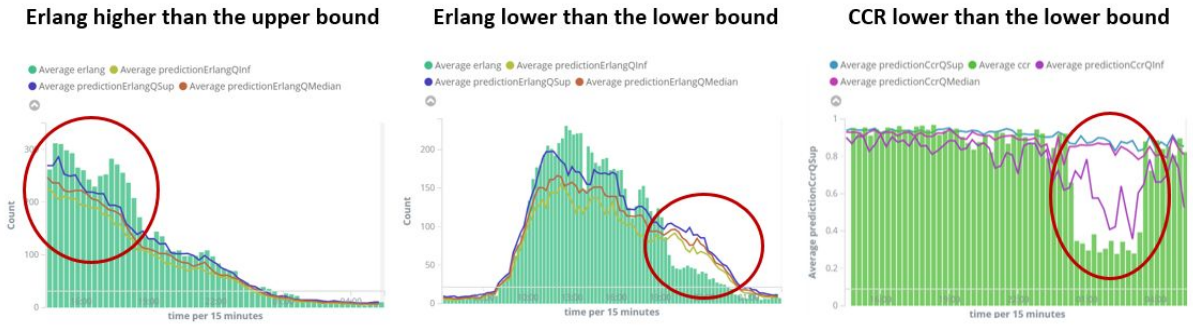
Grâce au modèle actuel nous estimons la distribution (médiane et intervalles de confiance) de deux KPIs principales :

- Erlang : permet de constater la charge de trafic
- CCR (Call Completion Ratio) : permet de constater la qualité du trafic

Disposer d'un indicateur de dispersion de la valeur d'Erlang et de CCR pour chaque cluster, nous permet de savoir si les valeurs réelles d'Erlang et de CCR sont incluses ou exclues de l'intervalle de confiance.

Nous sommes essentiellement confrontés à un problème de détection des outliers : une nouvelle observation peut donc appartenir à la même distribution que les observations existantes ou doit être considérée comme une anomalie (outlier).

Plus précisément, une ressource est détectée comme une anomalie si l'une de ces conditions est vérifiée :



Le client m'a demandé de tester une méthode alternative pour effectuer ce processus de détection des anomalies.

Une approche qui n'a pas de rapport avec l'approche actuelle Clustering + Random Forests et qui repose plutôt sur une technologie plus récente, notamment le Deep Learning.

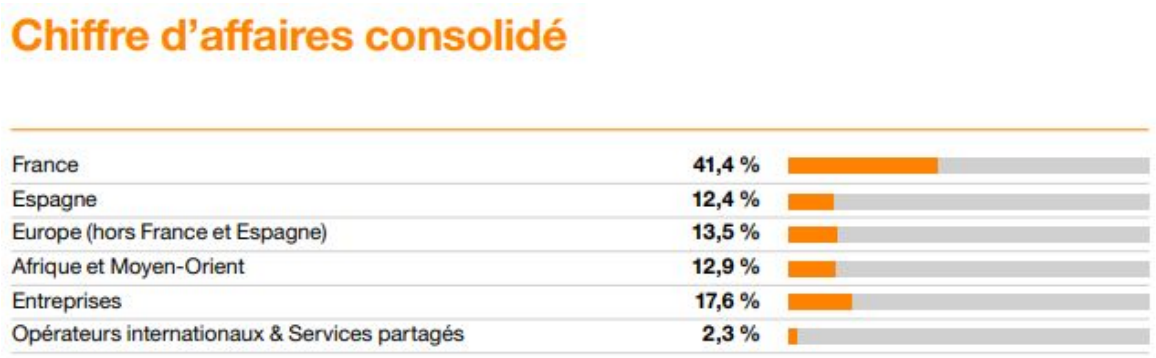
Le projet qui suit répond à ce type de demande.

2. Présentation du client

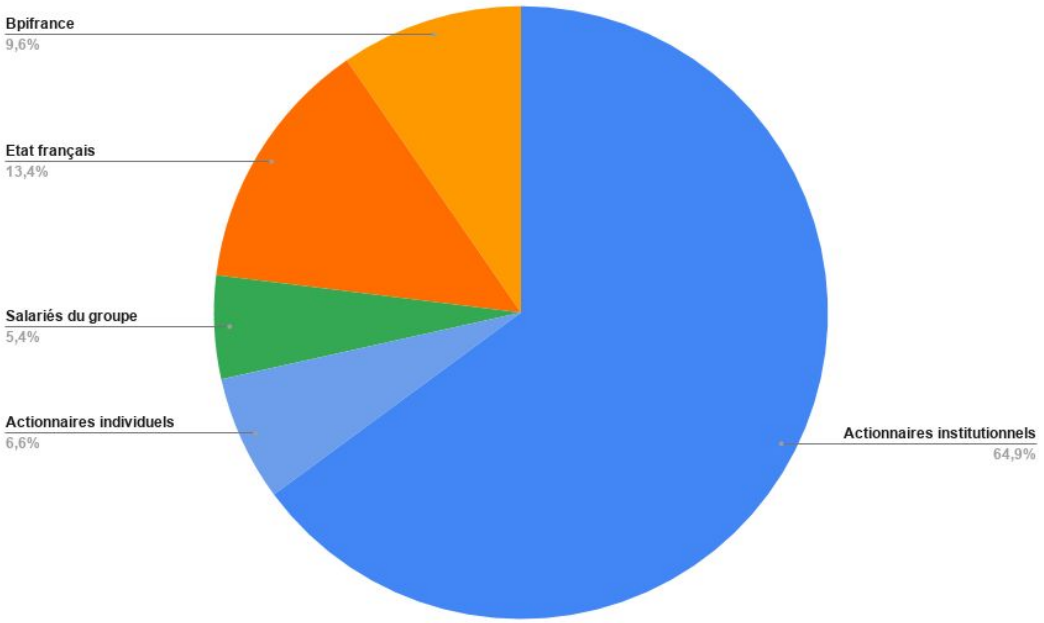
Orange est une entreprise française de télécommunication présente dans plus de 28 pays à travers le monde.

A la fin du 2019 elle comptabilise un chiffre d'affaire de 42,238 milliards d'euros[1] (+0,6% vs 2018), un résultat net de 3,006 milliards d'euros (+53,8% vs 2018) et un EBITDA des activitées télécoms de 13,019 milliards d'euros (+0,6% vs 2018; 30,8% de CA)[2].

Ci-dessous le chiffre d'affaires consolidés du groupe réparties par zone géographique :

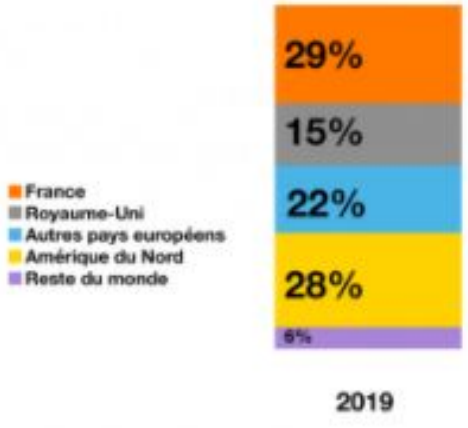


Sa capitalisation est de 25,345 milliards d'euros. Le prix actionnaire, au moment que j'écris est de 9,46€ avec un capital actionnaire actuellement répartis comme ci dessous :



[1] Rapport annuel intégré 2019 - Décembre 2019 (https://www.orange.com/fr/mentions-legales)
[2] Lettre à l'actionnaire - Février 2020 (https://club-actionnaires.orange.com)

Ci-dessous la répartition géographique des investisseurs institutionnels :



Au jour d'aujourd'hui Orange compte plus de 263 millions de clients dans le monde, 146768 salariés dont 90000 en France.

C'est également l'un des leaders mondiaux des services de télécommunications aux entreprises multinationales sous la marque Orange Business Services, notre actuel client.

Orange Business Services (OBS), est la branche B2B du groupe Orange qui fournit des services informatiques et de télécommunication aux entreprises en France et dans le monde, à travers plus de 220 pays. Créée en 2006, OBS compte à présent plus de 21 000 salariés.

3. Compréhension du besoin client

Nous avons déjà mis en place une solution qui permet le monitoring du réseau. La solution actuelle repose sur une première phase de clustering qui nous permet d'identifier les ressources qui se comportent de manière similaire selon les jours de la semaine.

Cette étape est suivie d'une deuxième phase de modélisation, grâce à laquelle un modèle de forêt aléatoire apprend le comportement habituel des différents clusters de ressources et nous fournit une prévision de leur comportement futur.

À ce jour, le projet est en phase de test et les premières réactions des utilisateurs sont plus que positives : les anomalies remontées à partir du modèle s'avèrent être de véritables anomalies et l'équipe opérationnelle utilise actuellement le dashboard Kibana mis en place pour monitorer l'ensemble du réseau.

Le client m'a demandé de trouver une méthode alternative pour effectuer ce processus de détection des anomalies, une méthode qui soit basée sur un framework de Deep Learning.

Le deuxième objectif du projet est de fournir au client une interface graphique lui permettant de visualiser les données à l'étude et les anomalies identifiées dans le jeu de données de test.

4. Etat de l’art et tendances du domaine

La problématique de la détection des anomalies intéresse chercheurs et spécialistes dans différents domaines depuis presque un siècle. En général, une anomalie est un échantillon de données qui est considérablement différent de l'ensemble des données.

Dans le secteur des réseaux téléphoniques, la définition des anomalies est directement liée à des considérations économiques et opérationnelles. Une anomalie est un échantillon de données indiquant un événement ou un composant dysfonctionnel (logiciel/matériel) qui entraîne, directement ou indirectement, une perte financière.

De nombreux systèmes de détection des anomalies ont été développés. Certains sont très spécifiques à un domaine et ne peuvent être appliqués à d'autres domaines en raison de certaines contraintes (type de données d'entrée et/ou de sortie, ressources disponibles, etc.).

D'autres systèmes sont plus flexibles et peuvent être appliqués dans de nombreux domaines différents. Les nouvelles solutions proposées, basées sur l'intelligence artificielle, sont suffisamment génériques pour être intégrées dans différents environnements.

Selon le contexte d'analyse, un système de détection d’anomalies peut avoir différents types d'entrées. Concernant les anomalies téléphoniques, les inputs peuvent être des logs produits par différents dispositifs du réseau. Il peut également s'agir de logs de communication tels que des Call Detail Record (CDR), comme dans le cas de mon projet.

Compte tenu de l'étendue de la littérature sur la détection des anomalies, j’ai regroupé les travaux autour de quelques grands axes : knowledge based, regression, classification, clustering, information theory and rule induction. Ces approches diffèrent sur de nombreux aspects tels que le champ d'application, le mode d'exécution (offline/online) et l'exigence de connaissance du domaine.

Dans le Cas Pratique 2 (Ref. E3) j’explique dans le détail les différentes techniques en énumérant leurs avantages et leurs inconvénients. Ici je laisserai juste un tableau de synthèse, avec les différentes approches qui ont été adoptées pour traiter la problématique de la détection des anomalies.

Tableau de synthèse : Comparaison des approches de détection des anomalies

Approach	Algorithm	Input	Output	Supervised Learning	Interpretable results
Regression	ARIMA	Time Series	Binary Label	No	Yes
	Neural Networks (RNN)				No
	Neural Networks (LSTM)				
Classification	Decision Trees	Numerical / Categorical	Binary / Multi-class label	Yes	Yes
	Random Forest				No
	Bayesian Network				Yes
	Neural Network				No
	SVM	Numerical			Yes
Clustering	K-means/modes	Numerical / Categorical	Unlabeled classification	No	Yes
	Hierarchical Clustering				
	EM				
	DBSCAN				
Information Theory	Entropy	Numerical / Categorical	Binary Label	No	No
Rule Induction	Association rules	Categorical	Score	Yes	Yes

Selon le contexte, la détection d'anomalies est appliquée à différentes structures de données et donne des résultats différents. D'une manière générale, il n'existe pas de configuration optimale qui convienne à tous les types de données et à tous les cas d'utilisation.

Compte tenu de notre type de problème et de la demande spécifique du client de tester un modèle d'apprentissage profond, mon choix a été réduit aux 3 modèles suivants : Facebook Prophet, Michelangelo (Uber) et GluonTS (Amazon).

J'ai finalement décidé de tester Gluon Time Series (GluonTS), un toolkit Python développé par Amazon pour construire, évaluer et comparer des modèles de séries temporelles basés sur l'apprentissage profond.

GluonTS est basé sur l'interface Gluon d'Apache MXNet et contient différents types d'architecture de réseaux neuronaux (LSTM, convolution, feedforward, attention, etc.). Il fournit des composants qui permettent de construire des modèles de séries temporelles, exactement ce dont notre cas exigeait.

5. Projet : IA

Comme indiqué dans le paragraphe précédent, d'une manière générale, il n'existe pas de configuration optimale qui convienne à tous les cas d'utilisation.

Le type de données utilisées, l'objectif du projet et l'importance que la variable temporelle assume, m'ont fait tendre vers l'expérimentation d'une approche de détection des anomalies basée sur des séries temporelles.

Une série temporelle est une séquence de points de données numériques dans un ordre chronologique continu. La séquence de valeurs numériques est prise à des points successifs équidistants dans le temps.

Une série temporelle peut être établie pour toute variable qui change au fil du temps.
Elle peut également être utilisée pour examiner comment les changements associés à la variable d'étude choisie se comparent aux changements d'autres variables au cours de la même période.

L'un des domaines dans lesquels cette approche est la plus utilisée est sans aucun doute l'investissement et la finance. Dans le domaine des investissements, une série temporelle suit le mouvement des points de données choisis, comme le prix d'un titre, sur une période de temps donnée. Ce suivi peut être effectué à court terme, comme le prix d'un titre à l'heure au cours d'un jour ouvrable, ou à long terme, comme le prix d'un titre à la clôture le dernier jour de chaque mois sur une période de plusieurs années.

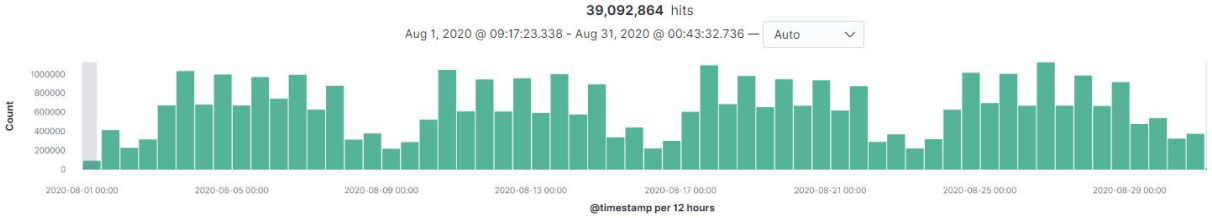
Par exemple, supposons que nous voulons analyser une série temporelle de prix de clôture quotidiens pour un titre particulier sur 5 ans. Nous obtenons une liste de tous les prix de clôture des actions pour chaque jour de cette période et nous les ordonnons par ordre chronologique. Il s'agirait d'une série temporelle de cours de clôture quotidiens du titre sur 5 ans.

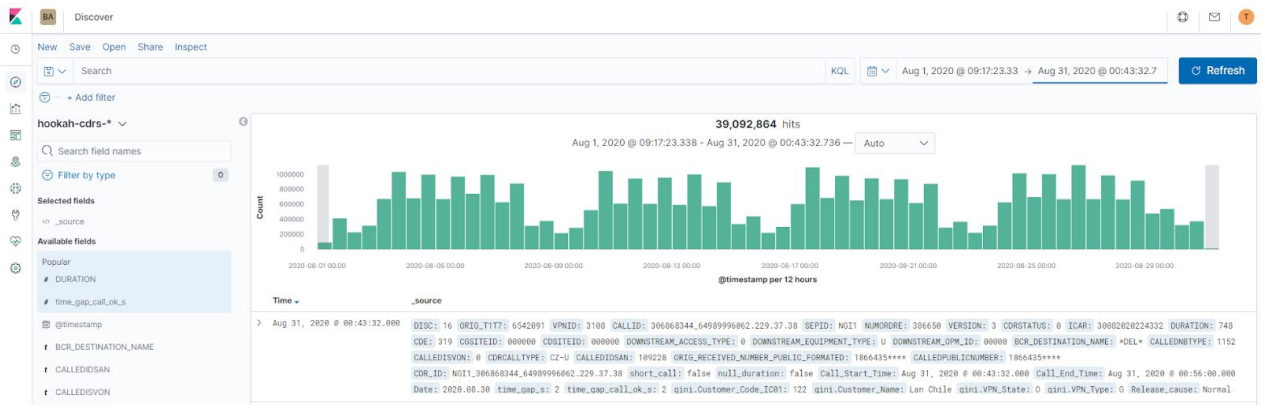
L'objectif de mon projet et le type de données utilisées (cfr. aussi le Cahier de charge) se prêtent bien à ce type d'approche. Nous utilisons des séries temporelles lorsque la variable "temps" est déterminante pour ce que nous devons prédire.

Une valeur à un instant t dépend forcément de la valeur à un instant t-1, et cette dépendance est la base des séries temporelles.

Tendance, saisonnalité, variable aléatoire (erreurs) sont des éléments qui caractérisent chaque série temporelle. Et lorsque nous analysons la distribution de nos données, ces éléments sont tous présents.

Ci-dessous le résultat de ma requête sur ELK pour le mois d'août :





Et ici le table qu'on obtient en format JSON relatif à une ressource en particulier (Calling trunk 6542091):

Table JSON


```
{
  "_index": "hookah-cdrs-t1t7_6542091",
  "_type": "_doc",
  "_id": "g5yYB3UBkEQpXa8QB0ze",
  "_version": 1,
  "_score": null,
  "_source": {
    "DISC": "16",
    "ORIG_T1T7": "6542091",
    "VPNID": "3108",
    "CALLID": "306868344_64989996@62.229.37.38",
    "SEPID": "NGI1",
    "NUMORDRE": "386650",
    "VERSION": "3",
    "CDRSTATUS": "0",
    "ICAR": "30082020224332",
    "DURATION": 748,
    "CDE": "319",
    "CGSITEID": "000000",
    "CDSITEID": "000000",
    "DOWNSTREAM_ACCESS_TYPE": "0",
    "DOWNSTREAM_EQUIPMENT_TYPE": "U",
    "DOWNSTREAM_OPM_ID": "00000",
    "BCR_DESTINATION_NAME": "*DEL*",
    "CALLEDNBTYPE": "1152",
    "CALLEDISVON": "0",
    "CDRCALLTYPE": "CZ-U",
    "CALLEDIDSAN": "109228",
    "ORIG_RECEIVED_NUMBER_PUBLIC_FORMATED": "1866435****",
    "CALLEDPUBLICNUMBER": "1866435****",
    "CDR_ID": "NGI1_306868344_64989996@62.229.37.38",
    "short_call": false,
    "null_duration": false,
    "Call_Start_Time": "2020-08-30T22:43:32Z",
    "Call_End_Time": "2020-08-30T22:56:00Z",
    "Date": "2020.08.30",
    "time_gap_s": 2,
    "time_gap_call_ok_s": 2,
    "gini.Customer_Code_IC01": "122",
    "gini.Customer_Name": "Lan Chile",
    "gini.VPN_State": "0",
    "gini.VPN_Type": "G",
    "Release_cause": "Normal Call Clearing",
    "rc_group": "HANG_UP",
    "@timestamp": "2020-08-30T22:43:32Z"
  },
  "fields": {
    "@timestamp": [
      "2020-08-30T22:43:32.000Z"
    ],
    "Call_Start_Time": [
      "2020-08-30T22:43:32.000Z"
    ],
    "Call_End_Time": [
      "2020-08-30T22:56:00.000Z"
    ]
  },
  "sort": [
    1598827412000
  ]
}
```

Le CDR (Call Detail Records) que nous récupérons sur l'ELK nous fournit des données intéressantes concernant la/les ressource.s que nous analysons, ou le trafic dans son ensemble.

L'ID de la ressource, l'ID de l'appel, la date, le moment exact du début et de la fin de l'appel et la durée de l'appel. Mais pas que, d'autres données intéressantes telles que l'ID du VPN utilisé, le préfixe du numéro appelant et du numéro appelé, la cause de la déconnexion, le type de CDR.

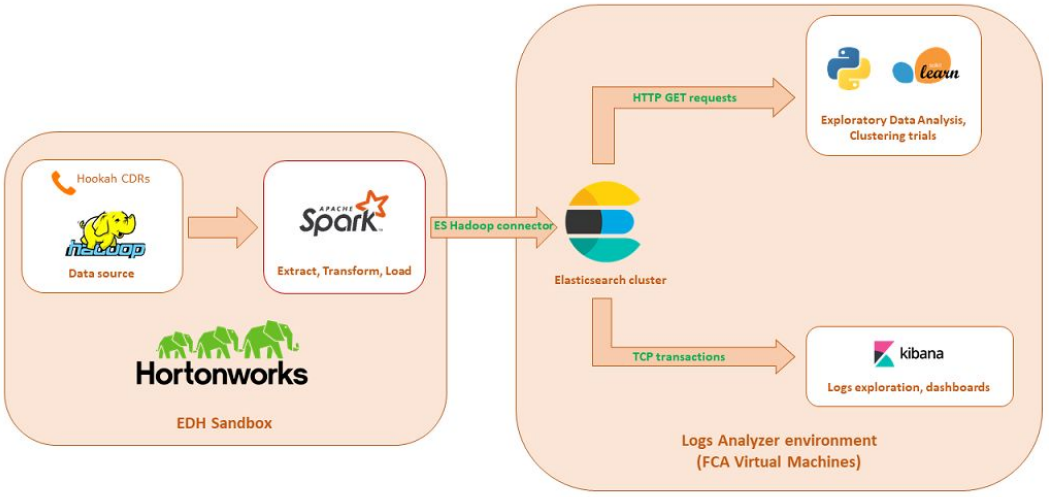
Exactement le type de données dont nous avons besoin pour poursuivre notre objectif : modéliser le comportement des ressources afin de pouvoir identifier ensuite les comportements anormaux.

Une fois qualifiées les données en vue de vérifier leur adéquation avec le projet, j'ai donc été confronté au problème de comment concevoir une base de données analytique qui pourrait servir de base d'apprentissage pour notre modèle.

L'idée était évidemment de procéder de manière à ce que notre base de données soit conçue avec l'approche orientée requêtes, en vue de l'utilisation des données pour mon IA.

Pour faire cela nous devons disposer de CDR dans une base de données prête à être utilisée.

Avant de passer à la conception de la BDD de training, je crois opportune faire d'abord une rapide aperçu de l'architecture du projet, pour voir aussi ce qui se passe en amont :



La source de données pour ce projet sont les CDR sur Hookah, un système de stockage de données d'Orange basé sur la technologie Hadoop.

En raison des mesures de sécurité, il n'est pas possible d'insérer des données dans le Sandbox de l'EDH, mais seulement d'en faire sortir des données. Pour ce faire, des composants tels que Apache Spark (utilisé ici) d'Apache Nifi sont disponibles.

Les opérations suivantes sont appliquées avec Spark avant de charger les CDR de Hookah dans Elasticsearch :

- Chargement des CDR de Hookah à partir de fichiers .ORC en tant que dataframe SQL Spark
- Filtrer les CDR sur une liste définie de ORIG_T1T7 pour réduire la quantité de données à prendre en compte. Ces ORIG_T1T7 ont été choisis "presque" aléatoirement (nous avons appliqué une contrainte pour prendre en compte plusieurs T1T7 importants).
- Ajouter des informations contextuelles en utilisant l'ID VPN (nom du client notamment)
- Traduire le code DISC en causes de rejets et en groupes de causes de rejets (conçus par l'équipe opérationnelle)
- Calculer l'heure de fin d'appel à partir de l'heure de début d'appel et la durée de l'appel
- Calculer les timegap entre :
    - tous les CDR d'un même ORIG_T1T7
    - Tous les CDR d'une durée > 0 (call_ok) du même ORIG_T1T7
- Formatter le dataframe et envoyer les données à Elasticsearch

Le principe que nous essayons d'appliquer ici est de calculer le moins de transformations possible dans Spark, et de garder les CDR aussi purs que possible dans Elasticsearch.

C'est depuis Elasticsearch que j'ai programmé l'import des données initiales nécessaires pour le training de mon IA. Après avoir importé les modules nécessaires, j'ai initié une connexion au cluster ELK de la manière suivante :

```
# Import modules
import pandas as pd
import numpy as np
from elasticsearch import Elasticsearch
import json
import seaborn as sns
sns.set_style('whitegrid')
from matplotlib import pyplot as plt
from pandas.io.json import json_normalize
from tqdm import tqdm_notebook
import plotly.express as px
import plotly.graph_objects as go
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")

# Import Elasticsearch config
dict_elastic_config = {
    "host": Données Sensibles,
    "port": Données Sensibles
    "http_auth": Données Sensibles,
    "use_ssl": True,
    "verify_certs": False
}

es_log_Analyzer = Elasticsearch([dict_elastic_config])
```

Pour préparer les données disponibles en vue de leur utilisation par les algorithmes d'intelligence artificielle, j'ai créé certains functions python utiles pour le data processing

```python
# Utils functions for data processing
def parse_agg_results(response):
    records = response['aggregations']['buckets']['buckets']
    df = json_normalize(records)
    return df

def update_body(body, after_key):
    new_body = body.copy()
    new_body['aggs']['buckets']['composite']['after'] = after_key
    return new_body

def format_df(df):
    df['timestamp'] = pd.to_datetime(df['key.timestamp'], utc=True)
    df['timestamp'] = df['timestamp'].dt.tz_localize(None)
    df.drop(columns=['key.timestamp'], inplace=True)
    return df
```

Comme on a vu dans l'image a page 10, le résultat d'une requête ELK est en format JSON. La première fonction me permet de normaliser les données JSON semi-structurées dans un tableau plat, et obtenir un dataframe pandas.

La deuxième fonction me permet de mettre à jour le body de ma requête en me basant sur la date de la dernière réponse obtenue (after_key).

La troisième fonction me permet de convertir le timestamp (en string sur ELK) dans un format time vraiment exploitable en utilisant la methode pd.to_datetime.

Ces trois fonctions sont ensuite utilisées dans la fonction suivante :

```python
def get_couples_time_aggregation_results(es_client, body, index_pattern):

    # Initialize request
    response = es_client.search(body=body, index=index_pattern,size=0)
    after_key =response['aggregations']['buckets']['after_key']
    df = parse_agg_results(response)
    df = format_df(df)
    boolean = True
    retries = 0

    # Iterate until pagination is over
    while boolean and retries < 5:
        new_body = update_body(body, after_key)
        response = es_client.search(body=new_body, index=index_pattern,size=0)
        if len(response['aggregations']['buckets']['buckets']) > 0:
            new_after_key = response['aggregations']['buckets']['after_key']
            boolean = after_key != new_after_key
            if boolean:
                after_key = new_after_key
                retries = 0
                df_new = parse_agg_results(response)
                df_new = format_df(df_new)
                df = pd.concat([df,df_new])
            else:
                boolean = True
                retries += 1
        else:
            boolean=True
            retries += 1
    return df
```

Cette fonction me permet de requêter la base ELK et de générer mon dataframe dans sa forme finale. Les arguments qu'il faut passer à cette fonction sont les suivants :

1. Cluster ELK
2. Requête
3. Index Pattern

Ci-dessous le détail de la requête :

```
request_body = {
  "size": 0,
  "query": {
    "bool": {
      "must": [
        {
          "range": {
            "@timestamp": {
              "gte": "2020-08-01T00:00:00Z",
              "lte": "2020-10-01T00:00:00Z"
            }
          }
        }
      ]
    }
  },
  "aggs": {
    "buckets": {
      "composite": {
        "size": 10000,
        "sources": [
          {
            "timestamp": {
              "date_histogram": {
                "field": "@timestamp",
                "fixed_interval": "15m",
                "format": "yyyy-MM-dd HH:mm:SSZ"
              }
            }
          },
          {
            "null_duration": {
              "terms": {
                "field": "null_duration"
              }
            }
          }
        ]
      }
    }
  }
}
```

À ce stade, il nous suffit de créer notre dataframe en utilisant la fonction créée précédemment, et en lui passant les arguments nécessaires.

```
df_call_drops = get_couples_time_aggregation_results(es_log_Analyzer, request_body, 'hookah-cdrs-t1t7_*')
```

```
1 df_call_drops
```

	doc_count	key.null_duration	timestamp
0	2488	False	2020-08-01 00:00:00
1	213	True	2020-08-01 00:00:00
2	3323	False	2020-08-01 00:15:00
3	194	True	2020-08-01 00:15:00
4	4198	False	2020-08-01 00:30:00

Afin qu'il puisse servir notre objectif de détection des anomalies, le dataframe obtenu nécessite encore quelques modification :

```
1 df_call_drops = get_couples_time_aggregation_results(es_log_Analyzer, request_body, 'hookah-cdrs-t1t7_*')
2 df_call_drops.set_index('timestamp',inplace=True)
3 df_call_drops = df_call_drops.astype(dtype={'key.null_duration': str})
4 df_call_drops = df_call_drops.pivot_table(index=['timestamp'], columns='key.null_duration', values='doc_count')
5 df_call_drops['total'] = df_call_drops['True'] + df_call_drops['False']
6 # df.reset_index(inplace=True)
```

```
1 freq='15min'
2 df_call_drops = df_call_drops.groupby(by=pd.Grouper(freq=freq)).agg({'total': np.sum})
```

```
1 df_call_drops.head()
```

	total
timestamp	
2020-08-01 00:15:00	3517
2020-08-01 00:30:00	4369
2020-08-01 00:45:00	4728
2020-08-01 01:00:00	6742
2020-08-01 01:15:00	5698

J'ai finalement obtenu le dataframe sous la forme que je voulais : le nombre total de CDR finaux (d'où le nom df_call_drops) regroupés par paquets de 15 minutes, avec le timestamp défini comme index.

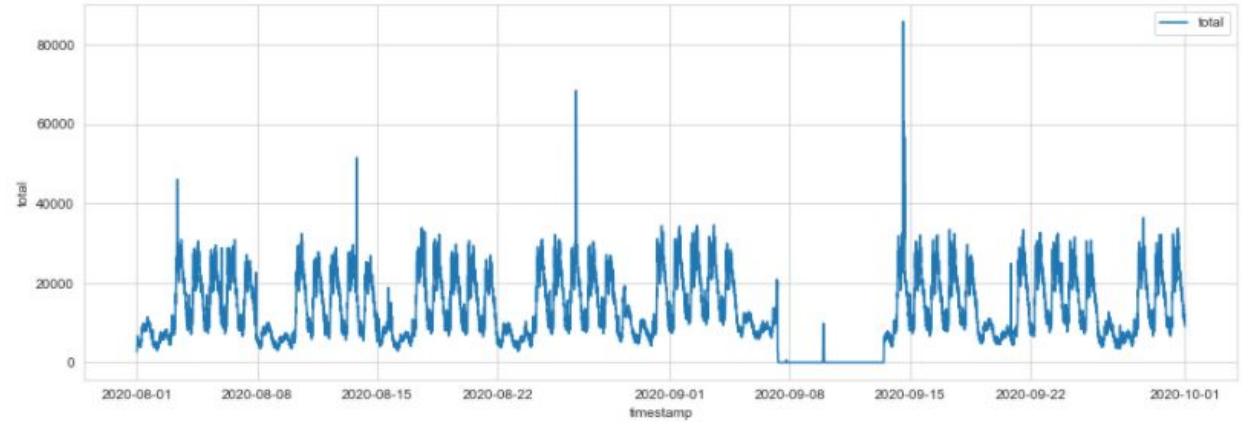
J’ai donc créé une fonction pour pouvoir afficher mon dataframe rapidement, en pouvant choisir un intervalle de temps à choix :

```python
# Utils functions for data visualization
def plot_metric(values, df, start_date, end_date, freq, aggfunc='sum', yscale='linear'):
    # prepare df before plotting
    df_viz = df.copy()
    df_viz = df_viz.reset_index()
    df_viz.set_index('timestamp', inplace=True)

    # Create figure container
    plt.figure(figsize=(15,5))

    # Plot values columns
    for column in values:
        df_temp = df_viz.copy()
        df_temp = df_viz.groupby(by=pd.Grouper(freq=freq)).agg({column: aggfunc})
        df_temp = df_temp[(df_temp.index >= start_date) & (df_temp.index <= end_date)]
        df_temp.reset_index(inplace=True)
        sns.lineplot(x='timestamp',y=column, data=df_temp)

    # Show figure
    plt.legend(values)
    plt.yscale(yscale)
    plt.show()
```

Cette fonction me permet d'afficher ma série temporelle en lui passant les suivants arguments :

1. colonne à afficher
2. dataframe
3. date de début
4. date de fin
5. fréquence

```python
# Plot global CDRs counts
start_date = '2020-08-01 00:00:00'
end_date = '2020-10-01 00:00:00'
freq = '15min'
cols = ['total']
plot_metric(cols, df_call_drops, start_date, end_date, freq)
```



Lorsque nous visualisons pour la première fois notre série temporelle, deux choses nous sautent aux yeux :

1) Il y a des outliers dans l'intervalle de temps choisi. Ce sont des quart d'heures avec un pic considérable de CDR.

2) Le dataset contient des valeurs manquantes, au cours de la deuxième semaine de septembre.

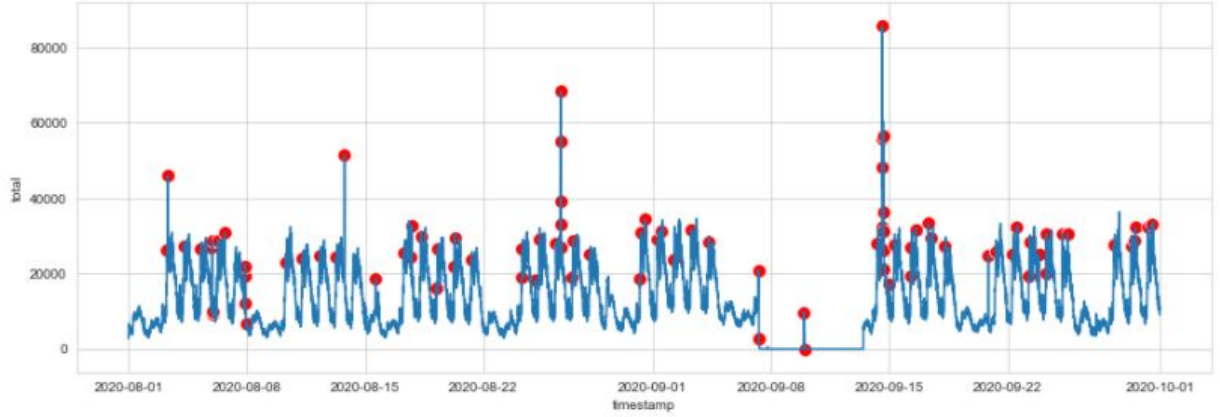
Une phase de cleaning est donc nécessaire avant de commencer le training de notre modèle.

En ce qui concerne le premier point, j'ai examiné d'autres intervalles de temps pour déterminer si les outliers identifiés étaient des valeurs aberrantes (une valeur qui est manifestement fausse) ou plutôt une valeur atypique (une valeur extrême, mais pas forcément fausse).

J'ai donc décidé de procéder à une sorte de lissage des pics pour permettre à mon modèle d'effectuer le training sans que celui-ci ne soit affecté par les mêmes pics.

La première étape a consisté à créer une fonction qui m'a permis d'identifier ces pics dans ma série temporelle. La fonction est construite en tenant compte de l'écart absolu de chaque point par rapport à la médiane mobile. Si cet écart dépasse le seuil de 7000 (résultant de plusieurs tests), la fonction identifie le point comme un outlier et ensuite il le signale par un cercle rouge.

```
# Remove spikes
threshold = 7000
df_call_drops['rolling_median'] = df_call_drops['total'].rolling(5).median()

relative_difference = np.abs(df_call_drops['total'] - df_call_drops['rolling_median'])#/df_call_drops['rolling_median']
df_call_drops['is_outlier'] = relative_difference > threshold


plt.figure(figsize=(15,5))
test = df_call_drops.copy().reset_index()
start_date = '2020-08-01 00:00:00'
end_date = '2020-10-01 00:00:00'
test = test[(start_date <=test['timestamp']) & (test['timestamp']<= end_date)]
sns.lineplot(y='total',x='timestamp', data=test)
sns.scatterplot(y='total',x='timestamp',data=test[test['is_outlier']],color='red',s=100)
plt.show()
```



Le fait d'avoir créé une colonne permettant d'identifier les pics (boolean), nous permet maintenant de les remplacer par la méthode d`interpolation :

```
df = df_call_drops[['total','is_outlier']]
```

```
df['cleaned'] = df['total'].where(df['is_outlier'] == False)
df['cleaned'] = df['cleaned'].interpolate(method='linear')
df
```

timestamp	total	is_outlier	cleaned
2020-08-01 00:00:00	2701.0	False	2701.0
2020-08-01 00:15:00	3517.0	False	3517.0
2020-08-01 00:30:00	4369.0	False	4369.0
2020-08-01 00:45:00	4728.0	False	4728.0
2020-08-01 01:00:00	6742.0	False	6742.0

En particulier, la méthode d'interpolation linéaire que j'ai choisi, nous permet de remplacer les outliers par des valeurs équidistantes des valeurs les plus proches :

timestamp	total	is_outlier	cleaned
2020-08-03 06:15:00	14621.0	False	14621.0
2020-08-03 06:30:00	17761.0	False	17761.0
2020-08-03 06:45:00	21014.0	False	21014.0
2020-08-03 07:00:00	26387.0	True	21940.0
2020-08-03 07:15:00	22866.0	False	22866.0

Nous avons donc résolu notre premier problème en identifiant les valeurs aberrantes et en les remplaçant par la méthode d'interpolation linéaire.

Maintenant il nous reste à résoudre le second problème précédemment identifié : le dataframe contient des valeurs manquantes, au cours de la deuxième semaine de septembre.

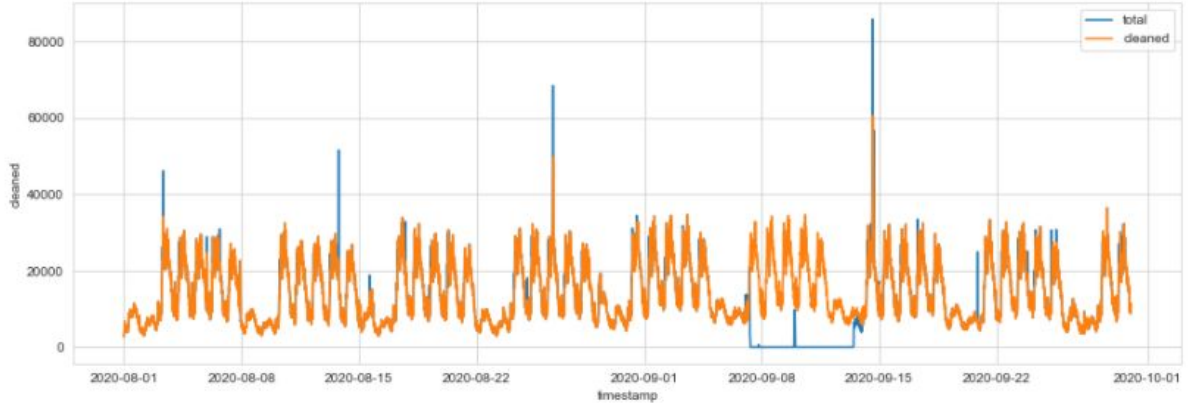
Étant donné le type de dataset, ignorer les données manquantes n'était en aucun cas une option possible. Travailler avec un jeu de données qui contient un "trou" d'une semaine n'était pas une solution acceptable.

En observant la distribution de la variable, j'ai donc décidé de combler le trou par les valeurs de la semaine précédente, en utilisant cette fonction :

```
# Rimpiazzare la settimana mancante con la settimana precedente
mask = (df.index >= '2020-09-07 00:00:00') & (df.index <='2020-09-14 00:00:00')
df['last_week'] = df['cleaned'].shift(672)
df['cleaned'] = np.where(mask, df['last_week'], df['cleaned'])
```

```
# Plot global CDRs counts
start_date = '2020-08-01 00:00:00'
end_date = '2020-09-30 00:00:00'
freq = '15min'
cols = ['total','cleaned']
plot_metric(cols, df, start_date, end_date, freq)
```

Nous sommes enfin en mesure de représenter la série temporelle nettoyée, sur laquelle nous pouvons effectuer le training de notre modèle. Ci-dessous les deux séries temporelles (originale et post-cleaning) affiché sur le même graphique :



J'ai démarré le processus de modélisation avec les modèles les plus utilisés pour les séries temporelles. Avec le modèle ARIMA d'abord, qui n'a pas donné de résultats particulièrement satisfaisants. Bien que le modèle SARIMAX m'ait donné la possibilité de considérer la saisonnalité et les facteurs exogènes, j'ai décidé de tester un autre type de modèle basé sur les réseaux des neurones.

Après une analyse de l'état de l'art (cfr. Cas Pratique 2, Ref. E3), mon choix a été réduit aux 3 modèles suivants : Facebook Prophet, Michelangelo (Uber) et GluonTS (Amazon). J'ai décidé de tester Gluon Time Series (GluonTS), un toolkit Python développé par Amazon pour construire, évaluer et comparer des modèles de séries temporelles basés sur l'apprentissage profond.

GluonTS est basé sur l'interface Gluon d'Apache MXNet et fournit des composants qui permettent de construire des modèles de séries temporelles.

```
import mxnet as mx
from mxnet import gluon
from gluonts.model.simple_feedforward import SimpleFeedForwardEstimator
from gluonts.model.seasonal_naive import SeasonalNaiveEstimator
from gluonts.model.deepar import DeepAREstimator
from gluonts.model.prophet import ProphetPredictor
from gluonts.evaluation import Evaluator
from gluonts.trainer import Trainer
import numpy as np
```

Après avoir importé les modules nécessaires, examinons comment utiliser l'un des modèles de séries temporelles inclus dans GluonTS pour faire des prévisions sur un dataset réel.

Étant donné une ou plusieurs séries temporelles, le modèle est conçu pour prédire les valeurs de prédiction de prediction_length , compte tenu des valeurs du contexte précédentes. Au lieu de prédire les meilleures valeurs individuelles pour chaque position de la plage de prédiction, le modèle paramètre une distribution de probabilité paramétrique pour chaque “output position”.

Pour encapsuler les modèles et les artefacts de modèle formés, GluonTS utilise une paire d'abstractions Estimateur/Prédicteur. Un Estimateur représente un modèle qui peut être entraîné sur un jeu de données pour obtenir un Prédicteur, qui peut ensuite être utilisé pour faire des prédictions sur des données inconnues.

A l’aide aussi de Grid Search, une méthode d’optimisation qui permet de tester une série de paramètres et de comparer les performances pour en déduire le meilleur paramétrage, j’ai décidé d’utiliser les suivants hyperparamètres :

```
1 N = 1 # number of time series
2 T = 2880  # number of timesteps
3 prediction_length = 672
4 freq = '15min'
```

Dans notre cas nous avons une seule série temporelle avec l'index dans un format *pandas.Timestamp* La fréquence des séries temporelles (pour cet exemple, j'utilise 15 minutes, donc freq="15min").
Concernant la prediction_length, disons que nous voulons prédire le comportement de nos ressources sur une semaine. La durée de la prédiction sera donc de 672 points dans le temps, (nombre des quarts d’heures présents dans une semaine).

GluonTS n'exige pas un format spécifique pour un jeu de données. La seule exigence pour un jeu de données est d'avoir un champ "target" et un champ "start".

Maintenant, nous pouvons découper notre jeu de données et le présenter dans un format GluonTS approprié. J'ai décidé de diviser mon jeu de données comme dans l'image suivante, en utilisant le 85% des données pour le training et les restants 15% pour la phase de test.

```
from gluonts.dataset.common import ListDataset
from gluonts.evaluation.backtest import make_evaluation_predictions

start_train_date = '2020-08-01 00:00:00'
end_train_date = '2020-09-21 00:00:00'
start_test_date = '2020-09-21 00:00:00'
end_test_date = '2020-09-28 00:00:00'
col = 'cleaned'

# train dataset: cut the last window of length "prediction_length", add "target" and "start" fields
train_ds = ListDataset([{"start": df.index[0], "target": df[col][:end_date]}],
                       freq=freq)
# test dataset: use the whole dataset, add "target" and "start" fields
test_ds = ListDataset([{"start": df.index[0], "target": df[col][:end_test_date]}],
                      freq=freq)
```

Le réseau neuronal feedforward intégré de GluonTS (SimpleFeedForwardEstimator) accepte une fenêtre d'entrée de longueur definie et prédit la distribution des valeurs de prediction_length suivantes.

Le modèle de réseau neuronal feedforward (à action anticipée) est un exemple d'estimateur. Dans GluonTS, les objets Estimator représentent un modèle de prévision ainsi que des détails tels que ses coefficients, ses poids, etc.

En général, chaque estimateur (pré-construit ou personnalisé) est configuré par un certain nombre d'hyper paramètres qui peuvent être soit communs (mais non contraignants) à tous les estimateurs (par exemple, la longueur de la prédiction), soit spécifiques à l'estimateur particulier (par exemple, le nombre de couches pour un réseau neuronal).

Enfin, chaque estimateur est configuré par un trainer, qui définit la manière dont le modèle sera entraîné (nombre d'époques, learning rate, etc.)

```
sff_estimator = SimpleFeedForwardEstimator(
    num_hidden_dimensions=[10],
    prediction_length=prediction_length,
    freq=freq,
    trainer=Trainer(
        ctx="cpu",
        epochs=10
    )
)
```

Après avoir spécifié notre estimateur avec tous les hyperparamètres nécessaires, nous pouvons l'entraîner en utilisant notre jeu de données de train “dataset.train” en invoquant la méthode de train de l'estimateur.

L'algorithme d'apprentissage renvoie un fitted modèle (ou un Predictor en langage GluonTS) qui peut être utilisé pour construire des prévisions.

```
sff_predictor = sff_estimator.train(train_ds)
```

```
100%|| 50/50 [00:04<00:00, 11.13it/s, epoch=1/10, avg_epoch_loss=10.7]
100%|| 50/50 [00:04<00:00, 12.07it/s, epoch=2/10, avg_epoch_loss=10]
100%|| 50/50 [00:04<00:00, 12.04it/s, epoch=3/10, avg_epoch_loss=9.82]
100%|| 50/50 [00:04<00:00, 12.11it/s, epoch=4/10, avg_epoch_loss=9.7]
100%|| 50/50 [00:04<00:00, 12.29it/s, epoch=5/10, avg_epoch_loss=9.64]
100%|| 50/50 [00:04<00:00, 12.15it/s, epoch=6/10, avg_epoch_loss=9.53]
100%|| 50/50 [00:04<00:00, 11.89it/s, epoch=7/10, avg_epoch_loss=9.46]
100%|| 50/50 [00:04<00:00, 12.05it/s, epoch=8/10, avg_epoch_loss=9.46]
100%|| 50/50 [00:04<00:00, 11.99it/s, epoch=9/10, avg_epoch_loss=9.42]
100%|| 50/50 [00:04<00:00, 12.13it/s, epoch=10/10, avg_epoch_loss=9.4]
Running evaluation: 100%|| 1/1 [00:00<00:00, 111.40it/s]
```

Une fois qu'on a notre predictor, nous pouvons maintenant prédire la dernière fenêtre de notre jeu de données-test et évaluer les performances de notre modèle.

GluonTS est livré avec la fonction make_evaluation_predictions qui automatise le processus de prédiction et d'évaluation du modèle. Cette fonction effectue les étapes suivantes :

- Supprime la dernière fenêtre de longueur prediction_length du dataset.test que nous voulons prédire
- L'estimateur utilise les données restantes pour prédire (sous forme de chemins d'échantillon) la fenêtre "future" qui vient d'être supprimée
- Le module produit les trajectoires d'échantillons de prévision et le jeu de données test

```python
def compute_forecast(predictor, test_ds):
    forecast_it, ts_it = make_evaluation_predictions(
        dataset=test_ds,  # test dataset
        predictor=predictor,  # predictor
        num_samples=100,  # number of sample paths we want for evaluation
    )

    forecasts = list(forecast_it)
    tss = list(ts_it)

    # first entry of the forecast list
    forecast_entry = forecasts[0]

    # first entry of the time series list
    ts_entry = tss[0]

    return ts_entry, forecast_entry
```

Si nous examinons la documentation, GluonTS propose également une méthode graphique qui permet de représenter les trajectoires de prévision sous forme de moyenne et d'intervalles de confiance.

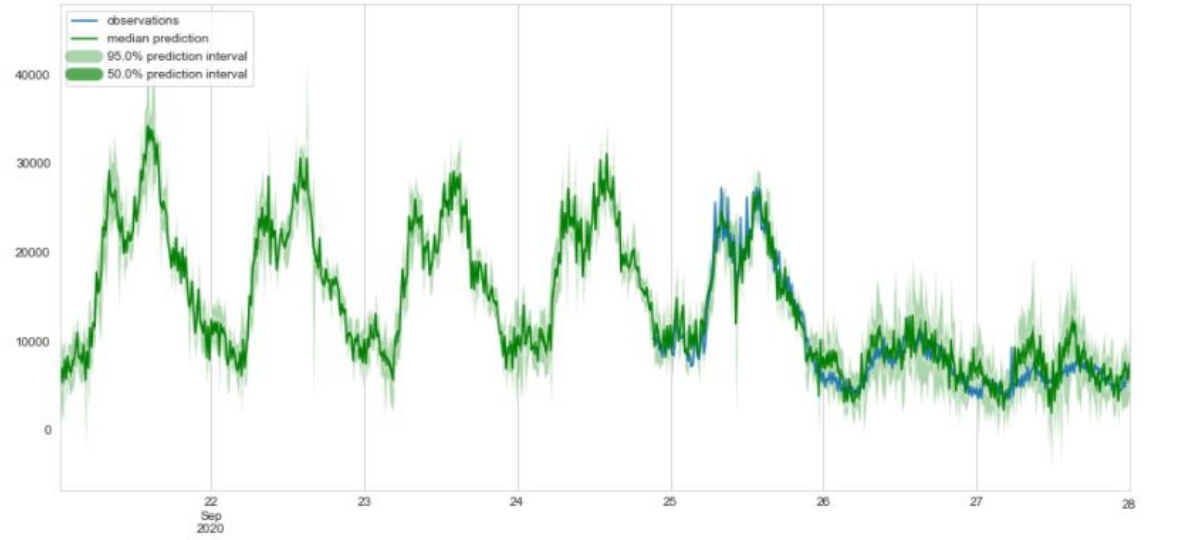
```python
def plot_prob_forecasts(ts_entry, forecast_entry, plot_length=300):
    prediction_intervals = (50.0, 95.0)
    legend = ["observations", "median prediction"] + [f"{k}% prediction interval" for k in prediction_intervals][::-1]

    fig, ax = plt.subplots(1, 1, figsize=(15, 7))
    ts_entry[-plot_length:].plot(ax=ax)  # plot the time series
    forecast_entry.plot(prediction_intervals=prediction_intervals, color='g')
    plt.grid(which="both")
    plt.legend(legend, loc="upper left")
    plt.show()
```

Il nous reste donc à fixer les intervalles de confiance qu'on veut et appeler les le deux fonctions natives de GluonTS :

Using only GluonTS build-in functions

```
1 # Compute forecasts
2 ts_entry, forecast_entry= compute_forecast(sff_predictor, test_ds)
```

```
1 plot_prob_forecasts(ts_entry, forecast_entry)
```



Comme nous l'avons vu, GluonTS nous permet d'obtenir la distribution prévue pour la semaine de test. En particulier, grâce à un training sur les données du mois précédent (effectuée en quelques minutes), il est en mesure de nous fournir la distribution de la médiane prédite et les intervalles de confiance souhaités, ainsi que les observations réelles (ligne bleu).

Cependant, j'ai décidé d'aller un peu plus loin et d'apporter des modifications afin que le résultat obtenu réponde encore mieux aux besoins du client.

Le graphique ci-dessus nous donne tous les éléments pour identifier les comportements anormaux : il suffit de vérifier quand les observations réelles sortent de l'intervalle de confiance.

Il est également vrai que l'identification des anomalies n'est pas aussi immédiate à l'œil nu. C'est pour cette raison que j'ai décidé d'essayer d'améliorer le type de visualisation afin que l'identification des anomalies soit encore plus rapide pour un être humain.

J'ai également pensé qu'il était approprié de donner au client la possibilité de vérifier les performances du modèle en direct, afin de savoir comment il se comporte par rapport à certains KPIs de performance tels que le RMSE par exemple.

Pour ce faire, j'ai créé deux fonctions. La première analyze_result (qui récupère de toute façon la fonction compute_forecast) me permet de calculer les écarts entre la prédiction de mon modèle et les données réelles.

Il calcule les principaux KPIs d'évaluation des performances (RMSE, MASE, Absolute Error) et sauvegarde ces données sur plusieurs dataframe.

```
def analyze_results(predictor, test_ds, df, distance_threshold=0.0):

    # Compute forecasts
    ts_entry, forecast_entry= compute_forecast(predictor, test_ds)

    # Compute intelligible columns for previous forecasts
    mask = (df.index > start_test_date) & (df.index <= end_test_date)
    df_results = df[['cleaned','total']][mask]
    df_results['yhat'] = forecast_entry.median
    df_results['yhat_upper'] = forecast_entry.quantile(1.0)
    df_results['yhat_lower'] = 2*df_results['yhat'] - df_results['yhat_upper']
    df_results['gap_sup'] = np.abs(df_results['total'] - df_results['yhat_upper'])
    df_results['gap_inf'] = np.abs(df_results['total'] - df_results['yhat_lower'])
    df_results['distance'] = df_results.apply(return_min_distance, axis=1)
    df_results['rel_distance'] = df_results['distance'] / df_results['yhat']*100
    df_results['anomalous'] =  ((df_results['total'] > df_results['yhat_upper']) | (df_results['total'] < df_results['yhat_l

    df_anos = df_results[df_results['anomalous']]

    # Evaluate predictor performance on test dataset
    evaluator = Evaluator(seasonality=672, quantiles=[0.5,0.95])
    agg_metrics, item_metrics = evaluator([ts_entry], [forecast_entry], num_series=len(test_ds))
    return df_results, df_anos, agg_metrics
```

La deuxième fonction, dans laquelle j'utilise Plotly, me permet de créer un graphique personnalisé de mes séries temporelles.

Plotly me permet évidemment de tracer la médiane et les intervalles de confiance, mais il me permet surtout de mettre en évidence le moment où les données relai sortent des intervalles de confiance (grâce au mode "markers").

```python
def plot_anomalies_detection(df_results, df_anos):
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=df_results.index, y=df_results['total'],
        fill=None,
        mode='lines',
        line_color='blue',
        name = 'y'
        ))
    fig.add_trace(go.Scatter(x=df_results.index, y=df_results['yhat'],
        fill=None,
        mode='lines',
        line_color='green',
        name='yhat'
        ))
    fig.add_trace(go.Scatter(
        x=df_results.index,
        y=df_results['yhat_upper'],
        line_color = 'rgba(26,150,65,0.1)',
        mode='lines',
        name='yhat_upper'))

    fig.add_trace(go.Scatter(
        x=df_results.index,
        y=df_results['yhat_lower'],
        fill='tonexty', # fill area between trace0 and trace1,
        fillcolor='rgba(26,150,65,0.3)',
        line_color = 'rgba(26,150,65,0.2)',
        mode='lines',
        name='yhat_lower'))
    fig.add_trace(go.Scatter(x=df_anos.index, y=df_anos['total'],
        fill=None,
        mode='markers',
        line_color='red',
        name='anomaly',
        marker=dict(size=0.5*df_anos['rel_distance'])
        ))
    fig.show()
```

Mais ce n'est pas tout. Ayant calculé dans la fonction précédente la distance entre les points anormaux et leur prédiction, Plotly me permet de faire ressortir également cette dernière grandeur.

Le diamètre du cercle qui met en évidence l'anomalie nous donne également une idée de la criticité de cette dernière.

Comme nous le voyons dans le dernier graphique, l'identification des anomalies est désormais beaucoup plus efficace et rapide, en répondant plus précisément à la demande de notre client.

La détection des anomalies se fait de manière beaucoup plus intuitive et elle permet aussi de prioriser l'alerting (et l'intervention) selon la criticité de l'anomalie.

6. Projet : Web App

L'objectif primaire du projet a été réalisé : le client m'a demandé s'il y avait une méthode alternative pour effectuer ce processus d'identification des anomalies, et j'ai testé avec succès un nouveau type d'approche à cette problématique.

Cette nouvelle approche est basée sur des séries temporelles. Nous utilisons également un modèle d'apprentissage profond pour l'identification des anomalies et nous sommes finalement en mesure d'identifier un comportement anormal dans l'ensemble du réseau.

Le deuxième objectif du projet est de fournir au client une interface graphique. Il faudra donc fournir au client une dashboard lui permettant de visualiser les séries temporelles à l'étude et les anomalies identifiées dans le jeu de données de test.

Une application qui, dans le futur, puisse être utilisée pour vérifier le comportement d'autres types de ressources et d'autres KPIs que le "Count of CDR per quarter", comme par exemple l'Erlang, le CCR, ou le nombre de Call KO.

Ci-dessous une résumé des fonctionnalités prévues pour l'utilisateur, qui devra être capable de :

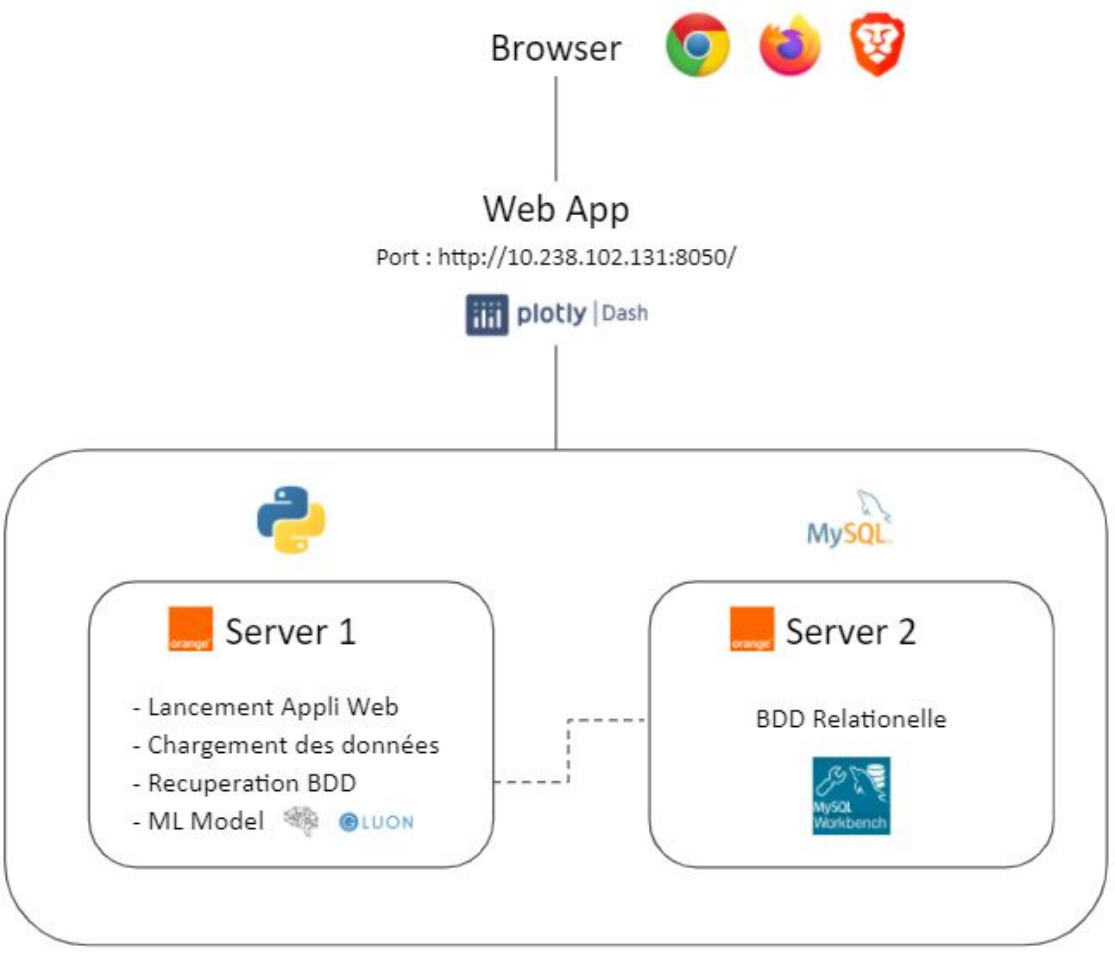
1	Avoir une alternative au modèle actuel pour la détection des anomalies
2	Avoir accès aux détails du fonctionnement du nouveau modèle (pour des éventuels modifications futures et/ou paramétrage)
3	Avoir une première interface graphique (MVP) accessible via un URL
4	Se connecter à l'application
5	Avoir accès à une visualisation du jeu de donnés étudié
6	Avoir accès à une visualisation des anomalies identifiés par le modèle
7	Vérifier les performances du modèle
Next 1	Si satisfaite du premier MVP fourni, pouvoir répéter cette approche avec d'autres type de ressources
Next 2	Si satisfaite du premier MVP fourni, pouvoir répéter cette approche en étudiant d'autres KPIs que le "Count of CDR per quarter" (Ex. Erlang, CCR, Call KO)

Concernant les choix techniques, vu le but de mon application et le fait que je voulais utiliser Plotly pour la visualisation, j'ai décidé d'utiliser Dash, un framework Python pour la construction d'applications web analytiques.

Dash est un produit de l'entreprise canadienne Plotly et il est construit sur Flask, un framework web gratuit écrit aussi en Python.

Concernant la base de données mon choix a été MySQL, un système de gestion de bases de données relationnelles (SGBDR). C'est un logiciel de gestion de base de données open source parmi les plus utilisés au monde. Les bases de données sont accessibles en utilisant plusieurs langages informatiques, y compris Python.

Ci-dessous un schéma qui résume l'architecture de mon application :

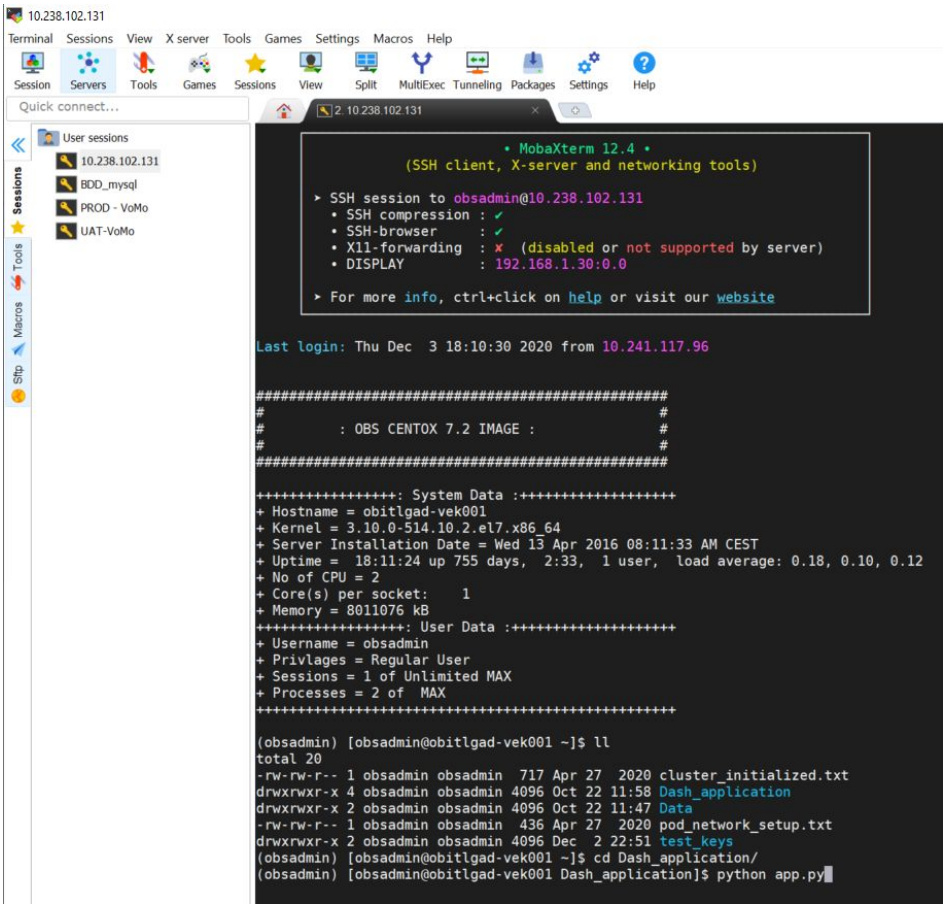


Mon application est finalement hébergée sur un serveur Orange auquel j'ai accès par MobaXtern.

MobaXtern est un logiciel Internet et réseau gratuit développé par Mobatek. Le logiciel fonctionne comme un client Xserver et SSH pour les ordinateurs.

Il fournit une interface utilisateur intelligente permettant d'accéder à des serveurs distants en entrant dans des réseaux ou des systèmes variés sur une seule plateforme.

Ci-dessous une aperçu de l'interface fournie par MobaXtern avec les commandes qui me permettent de lancer l'application :



Mais voyons un peu plus en détail comment j'ai développé mon application.

Ici le fichier app.py où, apres avoir importé importé les librairies nécessaires, j'initialise l'app Dash avec le code suivant :

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_auth
from dash.dependencies import Input, Output
```

```
# logger.info('starting dash application')
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
```

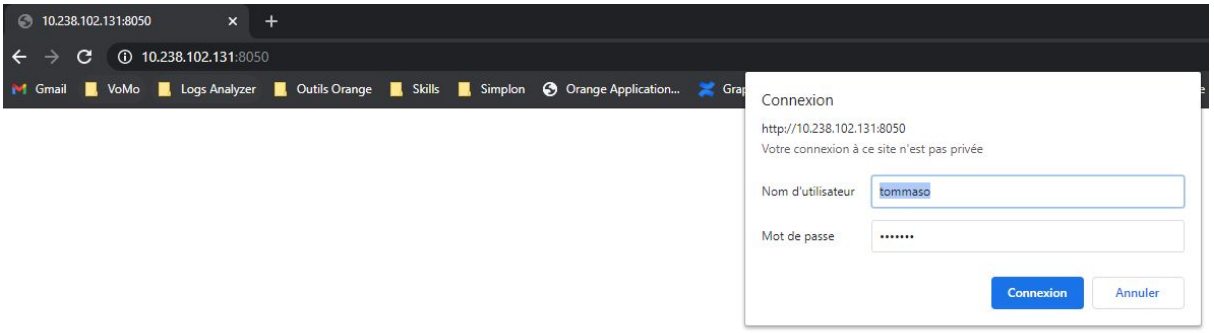
Ci-dessous un morceau du code qui montre comment j'ai créé le front end de ma page principale :

```
app.layout = html.Div(children=[
    html.H1(children='Anomalies detection using neural networks based models'),
    html.H4('Time series dataset'),
    html.Div([
        dcc.Graph(
            id='dataset_exploration',
            figure=ts_fig
        )
    ]),
    html.H4('Model plot'),
    html.Div([
        dcc.RadioItems(
            id='model_selection',
            options=[
                {'label': 'Simple Feed Forward', 'value': 'sff'},
                {'label': 'Deep AR', 'value': 'deepar'}
            ],
            value='sff',
            labelStyle={'display': 'inline-block'}
        )
    ],
    style={'width': '100%', 'display': 'inline-block'}
    ),
```

Concernant la partie back, un morceau du code que j'ai utilisé pour afficher le dataset :

```
import plotly.graph_objs as go
import pandas as pd
from plotly.subplots import make_subplots
import json

def plot_timeseries(df):
    fig = go.Figure()
    df_outliers = df[df['is_outlier']]
    fig.add_trace(
        go.Scatter(x=df['timestamp'], y=df['total'],
            fill=None,
            mode='lines',
            line_color='blue',
            name = 'Original ts'
        )
    ),
    fig.add_trace(
        go.Scatter(x=df['timestamp'], y=df['cleaned'],
            fill=None,
            mode='lines',
            line_color='orange',
            name = 'Cleaned ts'
        )
    ),
    fig.add_trace(
        go.Scatter(x=df_outliers['timestamp'], y=df_outliers['total'],
            fill=None,
            mode='markers',
            line_color='red',
            name='Outlier',
            marker=dict(size=10)
        )
    )
    return fig
```

Une fois connecté au serveur par Mobaxterm et lancé l'application avec la commande python app.py, on arrive sur un formulaire pour la connexion. Ci-dessous la page de login :



Au moment où l'utilisateur saisit username et mot de passe, on interroge la base de données Mysql où est présente la table "Credentials" qui contient la liste des utilisateurs (usernames et mots de passe) qu'ont le droit d'accéder à l'application.

Au début de mon projet username et mot de passe étaient stockés en dur, mais vu la nécessité de créer une deuxième base de données, en plus de la base de données de training, j'ai modifié aussi la façon avec laquelle les credentials d'accès étaient conçus initialement.

Ci-dessous le code créé pour gérer les authentication pour l'application :

```
# Set up authentication
auth = dash_auth.BasicAuth(
    app,
    dict_valid_creds
)
```

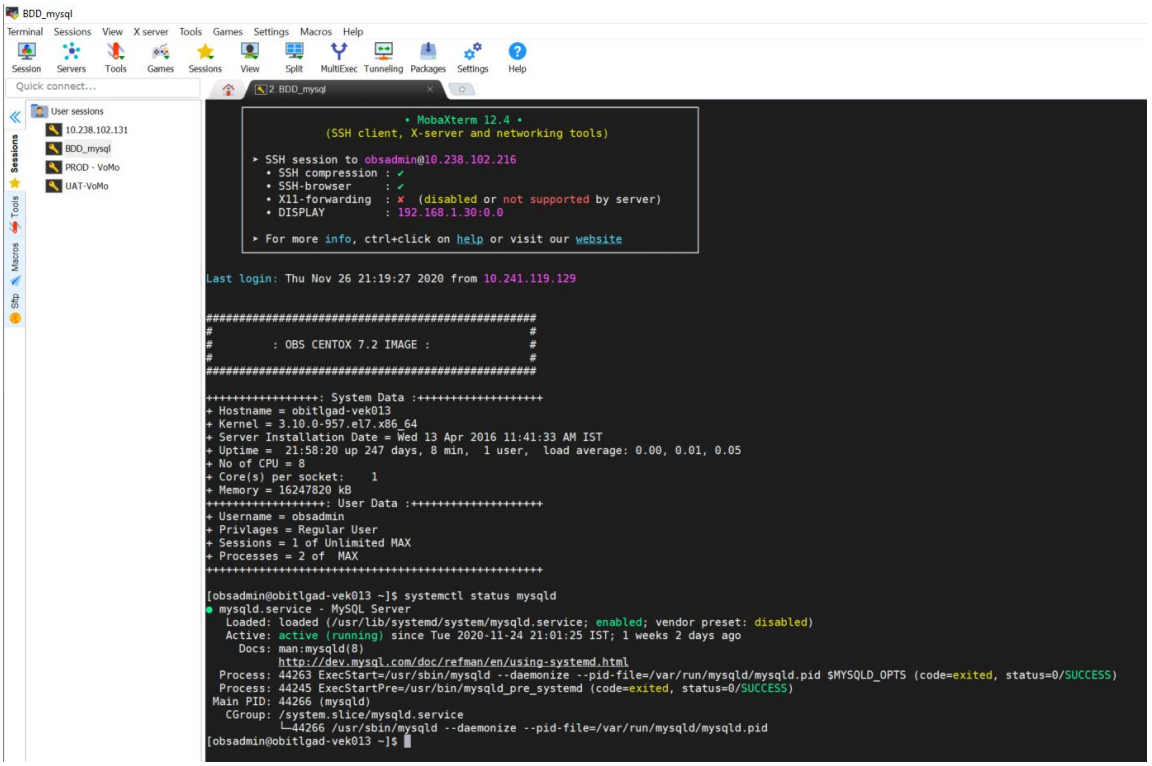
```
dict_valid_creds = fetch_passwords_from_mysql()
```

L'application récupère le tableau "Credentials" depuis ma BDD relationnelle précédemment créé, grâce à mysql.connector, un module Python autonome pour communiquer avec les serveurs MySQL.

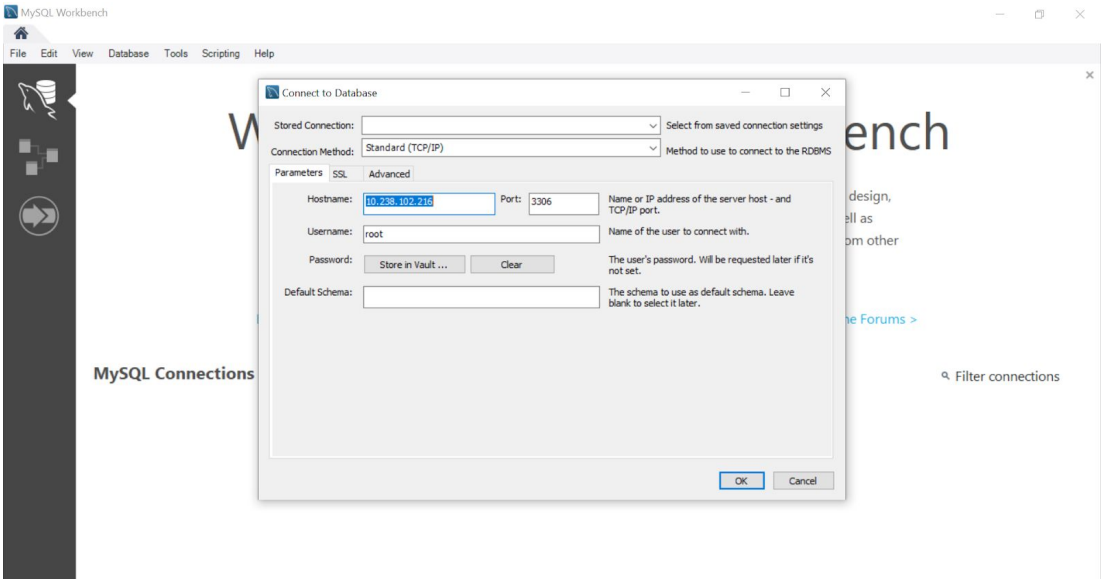
Ci-dessous le script python qui me permet de récupérer les credentials (username & password) depuis la table "credentials" précédemment créée :

```python
import mysql.connector
import pandas as pd

def fetch_passwords_from_mysql():
    mydb = mysql.connector.connect(
      host="10.238.102.216",
      user="root",
      password="Orange000!"
    )
    df_creds = pd.read_sql('SELECT * FROM dash_app.credentials', con=mydb)

    dict_creds = {}
    for cred in df_creds.to_dict(orient='records'):
        dict_creds[cred['user']] = cred['password']

    return dict_creds
```

Ci-dessous une image qui montre l'ouverture d'une nouvelle session "BDD_MySql" su Mobaxterm et la commande qui me permet de vérifier que MySQL il est actuellement en mode "run":



Connection à la base de données avec MySQL Workbench :



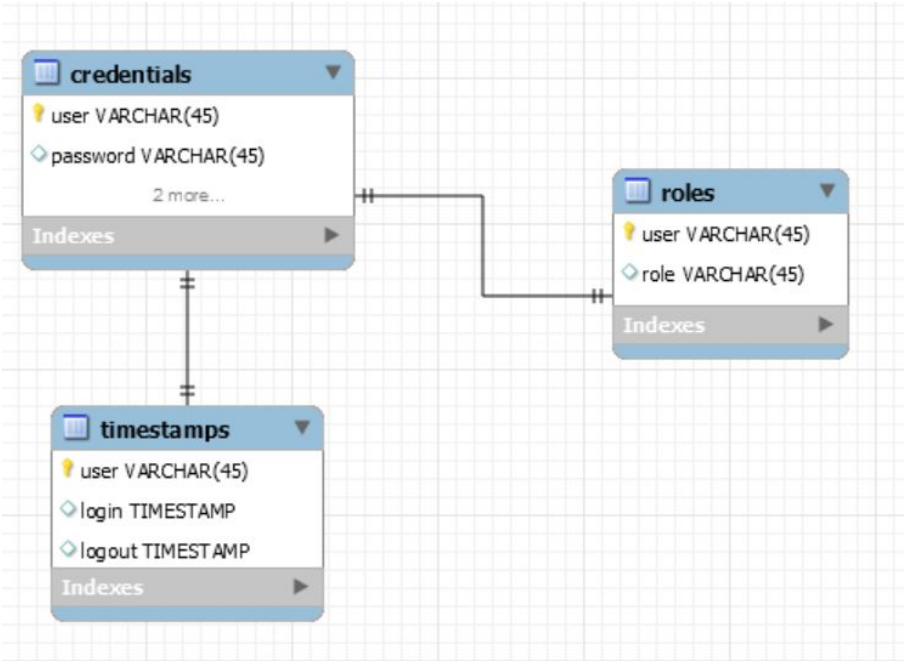
Pour rentrer un peu plus dans le détail la base de données relationnelle, j'ai d'abord créé une table “Credentials” où sont stockés le nom d'utilisateur et le mot de passe pour accéder à l'application.

A cette table est liée la table “Timestamp”, où sont stockées toutes les données relatives au login des utilisateurs et qui me permet de monitorer l’activité sur mon application.

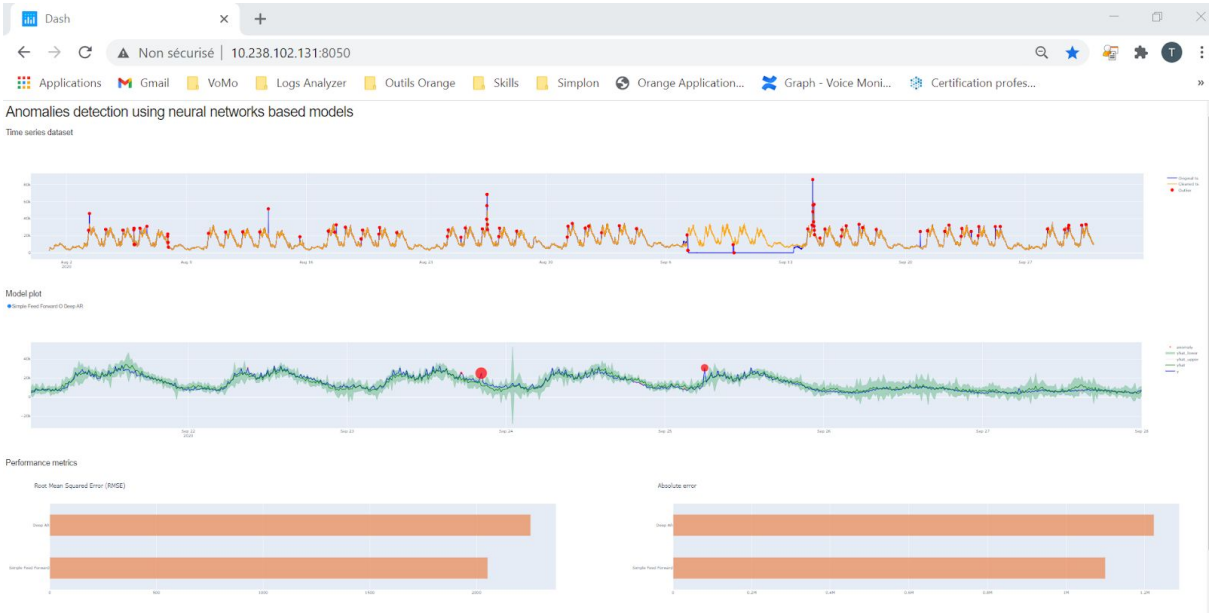
Pour des raisons de sécurité, j'ai ensuite créé une autre table "Rôles", avec les rôles et le nom d'utilisateur (la Primary Key qui relie les deux tables).

Ainsi, au cas où dans un second temps je devrais modifier les rôles, ou donner accès à un tiers à la table Roles, l'utilisateur n'aurait pas un accès direct aussi aux mots de passe, qui resteraient accessibles seulement sur la table “Credentials”.

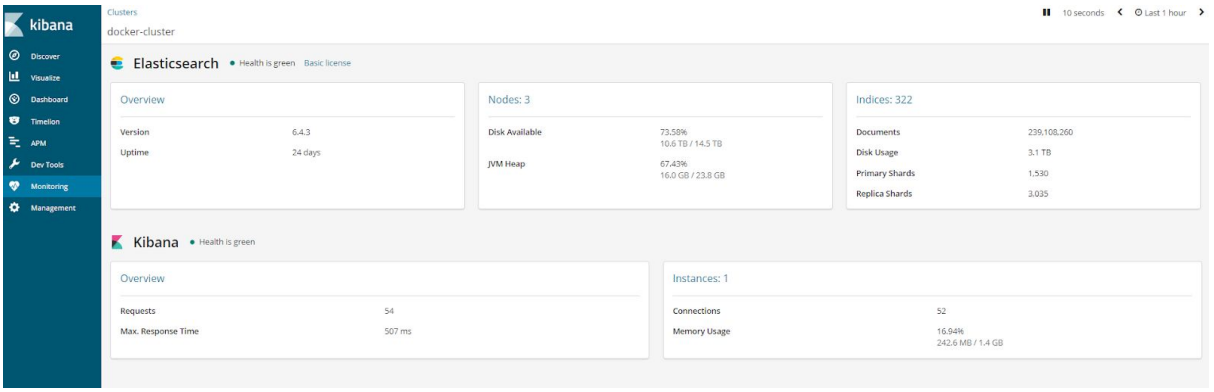
Ci-dessous le schéma de la base de données :



Une fois que nos credentials sont vérifiés, on a accès à la Main Page :



Concernant la phase de monitoring, une première phase de monitoring est mise en place directement dans la suite ELK (Logstash), d'où proviennent les données que j'ai utilisées pour créer la base de training.



Concernant plus en particulier mon application, afin de détecter et corriger les éventuels dysfonctionnements, le monitoring est assuré par un certain nombre d'actions à mener en fonction des risques estimés en utilisant cette matrice :

Risque	Criticité (Probabilité * Gravité)	KPI Monitoring	Action à envisager
Manipulation non securisée des données utilisateur	12	Injection de données dans la BDD incorrectes	Utiliser une fonction permettant de saisir les données utilisateur de maniere sécurisée (ex. limite minimum/maximum)
Mots de passe non securisé	6	Un mot de passe à faible securité est detecté (Sequence de chiffre 0-9 ou 9-0, Suite de chiffre similaires, Format date de naissance)	Preciser les criteres pour un mot de passe securisé. Empecher la sauvegarde d'un mot de passe qui ne respect pas les criteres definies.
Execution d'un contenu pas attendu	9	Insertion d'un fichier different que celui en format time series	Ne pas authoriser l'insertion de contenu externes

Concernant la phase de testing, la façon recommandée de tester une application Dash, comme indiqué dans la documentation, est d'instancier l'application au sein d'une fonction de test.

Ci-dessous une aperçu du code utilisé pour le test :

```python
# imports for monitoring
import dash
import dash_html_components as html


# give each testcase a tcid, and pass the fixture
# as a function argument, less boilerplate

def monitoring_test(dash_br.server_url = "http://10.238.102.130:8050/"):

    # define app inside the test function
    app = dash.Dash(__name__)
    app.layout = html.Div(id="nully-wrapper", children=0)

    # host the app locally in a thread, all dash server configs could be
    # passed after the first app argument
    dash_duo.start_server(app)

    # use wait_for_* if your target element is the result of a callback,
    # keep in mind even the initial rendering can trigger callbacks
    dash_duo.wait_for_text_to_equal("#nully-wrapper", "0", timeout=4)

    # use this form if its present is expected at the action point
    assert dash_duo.find_element("#nully-wrapper").text == "0"

    # to make the checkpoint more readable, you can describe the
    # acceptance criterion as an assert message after the comma.
    assert dash_duo.get_logs() == [], "browser console should contain no error"

    # visual testing with percy snapshot
    dash_duo.percy_snapshot("layout")
```

Pour les tests unitaires j'ai utilisé aussi pytest, un framework permettant de faire des tests et de vérifier si les différentes conditions sont correctes. Pytest m'as permis d'effectuer les tests des applications et composants Dash :

```python
import pytest
from pytest_dash.errors import DashAppLoadingError, NoAppFoundError
from pytest_dash.application_runners import import_app


def test_no_app_found():
    with pytest.raises(NoAppFoundError):
        import_app('app.invalid_app')
```

Pour le test fonctionnel j'ai créé un cahier de test en format excel. Le but de ces tests est de vérifier que le logiciel respecte ses spécifications et qu'il réalise les fonctions attendues correctement.

Ci-dessous une aperçu du test fonctionnel effectué pour la connexion à l'application :

Code Test	Test	Resultat Attendu	Status	Remarques	Ticket Ouvert	NB Ticket	Responsable
A	Connexion à l'App - Formulaire de connexion						
A1	Je lance l'application en cliquant sur le URL	La fenetre de connexion apparait avec un formulaire pour le login qui s'affiche	OK	-	Non	-	-
A2	Je saisi les identifiant (username+mdp) et je clique sur le bouton "connexion". Les identifiants sont presentes en BDD.	On arrive sur la page principale de l'application	OK	-	Non	-	-
A3	Je saisi les identifiant (username+mdp) et je clique sur le bouton "connexion". Les identifiants ne sont pas presentes en BDD.	La page de connexion se recharge avec un message d'erreur qui apparait : "Votre connexion a echoué"	OK	-	Non	-	-

7. Gestion de projet

Cela peut sembler banal mais, concernant les aspects liés à la gestion de projet, la première chose que j'ai faite c'a été de lire le référentiel en détail.

La complexité du projet m'a conduit dès le début à essayer de le décomposer en plusieurs parties afin de trouver la meilleure approche possible pour sa réalisation.

Une première décomposition a été celle suggérée par le référentiel lui-même en 4 macro-étapes :

1) Développer la gestion des données analytiques d'un projet d'application
2) Développer un programme d'intelligence artificielle
3) Développer une application d'intelligence artificielle
4) Gérer un projet de développement d'application

Cette première subdivision me donnait déjà une idée macro de ce que j'étais censé faire, mais ce n'est que lorsque j'ai procédé à la lecture des compétences sous-jacentes à chacune de ces macro-catégories que je suis passé à la deuxième étape : la rédaction d'une première version du cahier de charge.

Le fait que j'ai commencé quasiment de suite à rédiger le cahier de charge (annexe 1) m'a permis de commencer immédiatement à définir les besoins du client et la manière dont ces besoins pourraient être traduits en objectifs d'abord, et en aspects fonctionnels ensuite.

Le fait d'avoir défini les objectifs du projet dès le début m'a sans doute beaucoup aidé, car de l'autre côté je n'avais pas trop de liberté à cause d'un référentiel défini en amont important et très précis.

Au début du projet, pendant le période d'alternance, les mois ont été caractérisés par un rythme de travail très simple qu'on peut résumer avec le tableau suivant :

Période	Type de projet	Gestion de projet	Durée Sprint	Outil
3 semaines en entreprise	Projets OBS	SCRUM	3 Semaines	JIRA
1 semaine en formation	Projet Simplon	SCRUM	1 Semaines	Trello

Je suis plutôt convaincu qu'une méthode agile, si elle est bien mise en œuvre, offre plusieurs avantages par rapport à une méthode classique (ex. cycle en V).

Je crois qu'elle peut vraiment favoriser la communication, l'apprentissage et les changements. Le fait de concentrer les efforts vers une livraison rapide d'un produit ou d'une fonctionnalité permet une vraie réactivité de l'équipe en face au client.

Et surtout le fait d'impliquer le client dans le projet du début, à mon avis, permet de mieux comprendre ses problèmes et répondre à ses besoins.

Cette méthode a aussi d'autres avantages moins évidents comme la responsabilisation des membres de l'équipe et la rapidité d'exécution (le Scrum plus que le Kanban à cause du concept de sprint).

Donc, si j'étais fermement convaincu qu'une méthode comme celle-ci était le meilleur choix pour mon projet, la complexité du référentiel et le changement lié à ma vie professionnelle m'ont obligé à repenser l'approche liée à la gestion de projet.

En effet, tout au long du projet j'ai continué à travailler pour OBS, même pendant cette dernière période. Lorsque j'ai signé mon contrat, je me suis également rendu compte que je n'aurais pas pu utiliser les mois de novembre et décembre pour travailler sur ce projet.

En sachant que je n'aurais pu profiter pleinement des deux derniers mois avant la deadline, j'ai dû changer complètement d'approche. La première chose ç'a été de repenser totalement mes semaines de travail en le structurant de la manière suivante :

- LUNDI - VENDREDI
  9h -18h : Travail pour OBS
  18h30 - 21h : Projet Simplon (3 jours par semaine)

- SAMEDI - DIMANCHE
  6h par jour sur le Projet Simplon (Créneau libre)

Le temps de travail a donc augmenté de manière exponentielle. La charge de travail et le fait de devoir suivre plusieurs projets en même temps m'ont amené à repenser aussi ma méthode globale de gestion de projet.

Si pour les projets OBS je n'avais pas la main dessus et j'ai continué à travailler en SCRUM avec des sprint de 3 semaines, pour le projet Simplon j'ai finalement opté pour une méthode de gestion de projet hybride.

Une méthode qui pour certains aspect était totalement en ligne avec la méthode agile : une méthode itérative et incrémentale, centrée sur la réalisation d'un produit fonctionnel et sur la communication constante avec le client, dans l'objectif de récolter le maximum de feedback pour pouvoir avancer dans la bonne direction.

De l'autre côté, j'ai aussi décidé de mettre au centre de la gestion de projet le référentiel, comme s'il s'agissait d'une spécification du rendu final à livrer.

J'ai ensuite créé un outil de gestion de projet sur mesure : un fichier excel qui m'a beaucoup aidé pendant la réalisation du projet et la rédaction de cette thèse. Il s'agit d'un outil basé sur le référentiel, mais qui prend en compte aussi le meilleur des aspects de la gestion agile, comme par exemple l'estimation de la charge de travail de chaque tâche.

Ci dessous plusieurs images qui présentent le projet à différents stades de sa réalisation :

Image 1 : Le projet au moment du mois de juillet, quand l'objectif à court termes etait de preparer l'oral blanc (j'étais encore sur Trello)

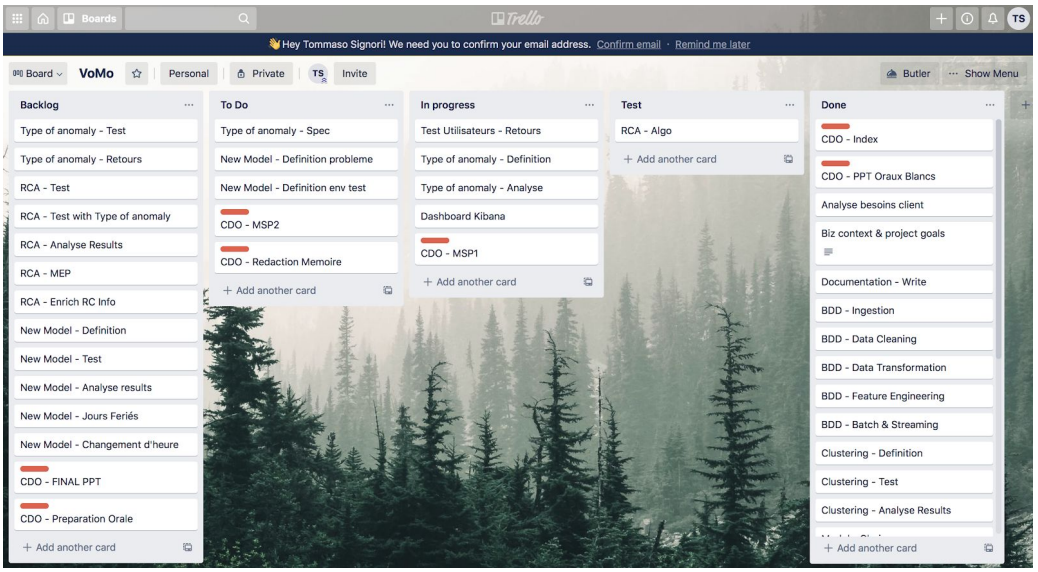


Image 2 : Le projet au moment où j'avais réalisé (et rédigé) la partie IA

Activités visées	Compétences attestées	Evaluation	Estimation Charge travail	Status Project	Status Redaction
1 Développer la gestion des données analytiques d'un projet d'application	C1. **Qualifier les données** grâce à des outils d'analyse et de visualisation de données en vue de vérifier leur adéquation avec le projet.	E1	8	DONE	DONE
	C2. **Concevoir une base de données analytique** avec l'approche orientée requêtes en vue de la mise à disposition des données pour un traitement analytique ou d'intelligence artificielle.	E1	13	DONE	DONE
	C3. **Programmer l'import de données** initiales nécessaires au projet en base de données, afin de les rendre exploitables par un tiers, dans un langage de programmation adapté et à partir de la stratégie de nettoyage des données préalablement définie.	E1	5	DONE	DONE
2 Développer un programme d'intelligence artificielle	C4. **Préparer les données disponibles depuis la base de données analytique** en vue de leur utilisation par les algorithmes d'intelligence artificielle.	E1	8	DONE	DONE
	C5. **Concevoir le programme d'intelligence artificielle** adapté aux données disponibles afin de répondre aux objectifs fonctionnels du projet, à l'aide des algorithmes, outils et méthodes standards, notamment de machine learning et de deep learning.	E1	20	DONE	DONE
	C6. **Développer le programme d'intelligence artificielle** selon les données du projet et les éléments de conception définis, en exploitant les algorithmes et les outils standards couramment utilisés dans le domaine.	E1	20	DONE	DONE
	C7. **Développer l'interaction entre les fonctionnalités de l'application et l'intelligence artificielle** dans le respect des objectifs visés et des bonnes pratiques du domaine.	E1	20	Thinking	In progress

Image 3 : Le projet au moment où j'avais terminé de travailler sur l'application web mais je n'avais pas encore rédigé cette partie.

Activités visées	Compétences attestées	My Project	Evaluation	Estimation Charge travail	Status Project	Status Redaction
3 Développer une application d'intelligence artificielle	C10. **Concevoir une base de données relationnelle** à l'aide de méthodes standards de modélisation de données.	Conception BDD 2 (app) Choix BDD 2 (justifie) ANNEXE 3 Schema BDD relationelle (enregistrement des login de l'app)	E1	13	DONE	In progress
	C11. **Développer les requêtes et les composants d'accès aux données** dans un langage adapté afin de persister et mettre à jour les données issues de l'application en base de données.	Fournir les requêtes	E1	8	DONE	In progress
	C12. **Développer le back-end de l'application d'intelligence artificielle** dans le respect des spécifications fonctionnelles et des bonnes pratiques du domaine.	TEST Test almeno una function principale (importante mostrare che hai identificato the main function e l'hai testata, ricorda creare dossier test). Di base nel dossier test dovresti avere un macro test (ex. test login) e vari sub-test sotto (ex. url ok, test formulaire ok, test envoi mail ok, test acces new page, login completed) Dovrai avere un 20-30 funzionalità Si fanno i test unitari all'inizio (piu test unitari ok > test functionel ok) test unitaires (coding) test d'integration (non necessario per simplon) test fonctionnelle (gestion projet, fai un excel con lista funzionalità, OK/KO) test de non regression (da fare per il caso pratico, quando aggiunigi l'IA ho cassé qlc o no?) Test - ToDo: Una 10na di test unitari Un test funzionale completo di una funzione CRITICA (puoi mettere anche una note tra 1-10 sulla criticità della funzione testata) Crea cahier de test avec les test ok/ko AUTHENTIFICATION LOGS & MONITORING	E1	100	DONE	In progress

Image 4 : Le projet au moment où il ne me restait que la partie gestion de projet à rédiger.

Activités visées	Compétences attestées	My Project	Evaluation	Estimation Charge travail	Status Project	Status Redaction
	C15. **Maintenir l'application d'intelligence artificielle** à l'aide des techniques de monitorage afin de détecter et corriger les éventuels dysfonctionnements.	LOGS & MONITORING	E1	20	DONE	DONE
4 Gérer un projet de développement d'application	C16. **Planifier les actions du projet** à l'aide d'un outil adapté afin de prévoir la complétion du projet dans les temps impartis.	SCRUM 3W sprint for OBS --> JIRA 1W sprint for Simplon --> Trello, Excel Signature du contrat > Changement de rythme + Methode Hybride	E1	3	DONE	In Progress
	C17. **Concevoir un système de veille technologique** permettant de collecter, classifier et analyser l'information afin d'améliorer la prise de décisions techniques.	Detection anomalies reseaux telecom 1. Problematique IA 2. Etat de l'art 3. Analyse sources > CAS PRATIQUE 2 - REF. E3	E3	8	DONE	DONE
	C18. **Communiquer avec les parties prenantes** afin de rendre compte de l'avancement du projet en mettant en oeuvre les canaux de communication nécessaires.	Communication avec Profs & Simplon Communication reguliere avec le client Respect des Livrables et des Deadlines	E1	2	In Progress	In Progress

Image 5 : La Burndown Chart mise en place pour le dernier mois de travail

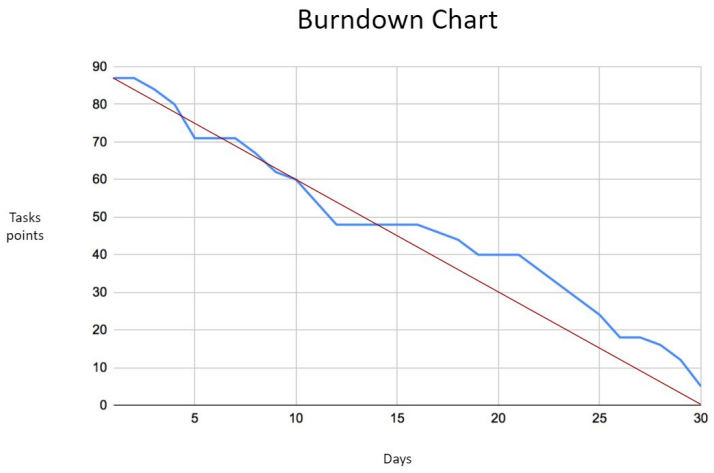


Image 6 : Un tableau de synthèse de la roadmap projet dans son ensemble

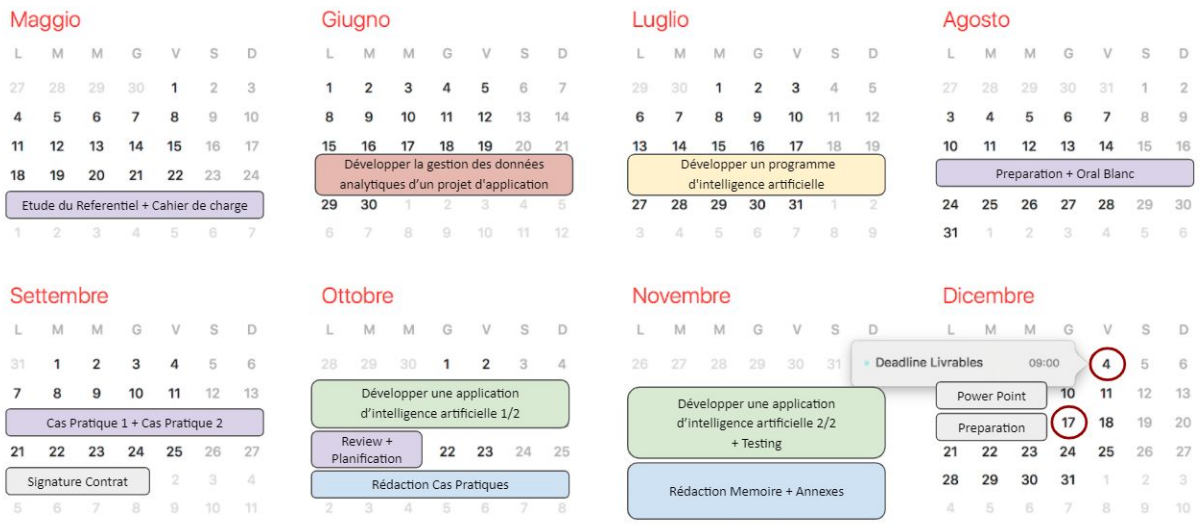
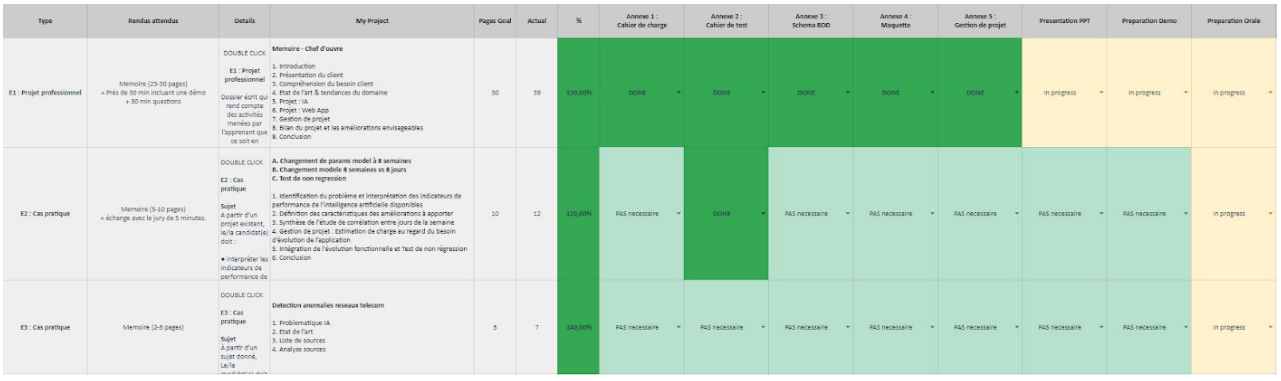


Image 7 : L'onglet de mon outil de gestion de projet sur excel qui me permet de vérifier l'envoi de toutes les livrables attendues



8. Bilan du projet

En toute clarté, la réalisation de ce projet n’a pas été facile.

Dès la première lecture du référentiel, j'ai réalisé la complexité du projet. Mais le fait que ce référentiel ait été modifié à plusieurs reprises a rendu ce projet encore plus complexe et stressant.

Je crois sincèrement que cela n'arrivera plus à l'avenir. Les autres promotions pourront bénéficier de la disponibilité du référentiel du projet dès le début, et donc d’un programme cohérent avec ce dernier.

À cela, il faut ajouter qu'à la fin de mon contrat, j'ai été engagé par OBS. Je ne peux certainement pas me plaindre de cela, mais cela a certainement eu un impact sur l'esprit dans lequel j'ai conclu ce projet et avec lequel j'écris maintenant ces lignes.

Comme j’ai dit dans le chapitre précédent sur la gestion de projet, le fait de n’avoir pas pu compter sur les mois qui ont suivi la fin de la formation m'a obligé à un impressionnant tour de force.

J'espère sincèrement que Simplon prendra en compte l'éventualité que d’autres personnes pourraient se retrouver dans la même situation lors des prochaines promotions.

Il y a sans aucun doute des choses positives que j'ai apprises grâce à ce projet.

Je pense avoir développé un niveau d'autonomie important. Ma capacité de problem solving a certainement augmenté ainsi que ma capacité à maintenir une charge de travail importante sur le long terme.

Je suis aussi très fier de ma croissance personnelle et de la montée en compétence technique que j'ai eu cette année.

C'était la principale motivation qui m'a poussé à entreprendre cette formation, et sans doute quelque chose qui me donne beaucoup de confiance pour l'avenir.

9. Conclusion

Au-delà des problèmes qui se sont posés au cours de cette période, le projet a été très enrichissant pour moi car il m'a permis de découvrir plus en détail le domaine de la data, de l'IA, du développement web, de la gestion de bases de données et de la gestion de projet.

Avoir travaillé en autonomie sur un sujet réel de détection d'anomalies téléphoniques m'a permis aussi de monter en compétence sur le domaine du réseaux de télécommunications.

Concernant plus spécifiquement le projet, le client m'a demandé de trouver une méthode alternative pour effectuer ce processus de détection des anomalies, une méthode qui soit basée sur une technologie plus récente.

J'ai mis en place un modèle de deep learning pour la détection, qui tourne actuellement sur une application web interactive. Les KPIs de référence que j'ai étudié comme le RMSE, le MASE et le Absolute Error montrent que le modèle a une précision comparable à celle requise.

Ces résultats me font réfléchir à ce que pourraient être les évolutions possibles du projet.

Tout d'abord, la possibilité d'étendre ce type d'analyse à d'autres types de ressources, dans l'idée de pouvoir un jour surveiller l'ensemble du réseau Orange grâce à cette approche.

De plus, je pense qu'il serait certainement intéressant d'essayer d'étendre cette méthode à d'autres KPIs metier comme l'Erlang et le CCR, pour avoir une vision encore plus complète de l'état de santé du réseau.

Une dernière étape possible pourrait être de tester d'autres modèles de Deep Learning en plus du Gluon TS, notamment Michelangelo (Uber) ou Facebook Prophet, et vérifier quel est le modèle le plus performant pour ce type de cas.

Je crois fortement que l'IA puisse automatiser et consolider le processus de détection d'anomalies et de résolution de défaillances dans les réseaux.

Nous pouvons vraiment viser à une surveillance en temps réel encore plus efficace, avec d'énormes bénéfices pour les équipes opérationnelles de support et pour les clients finaux.