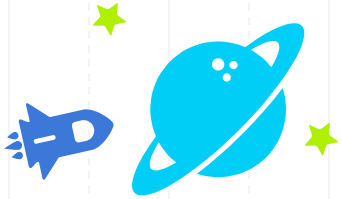
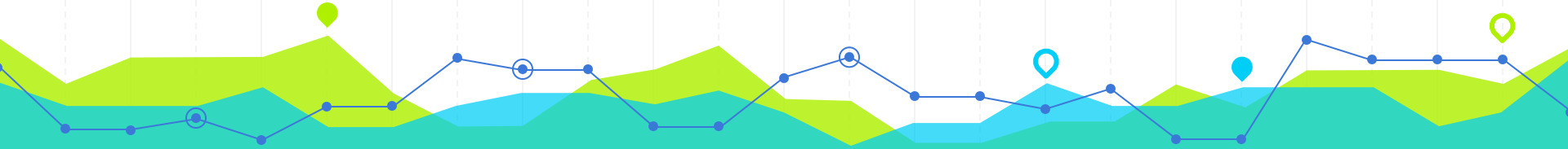
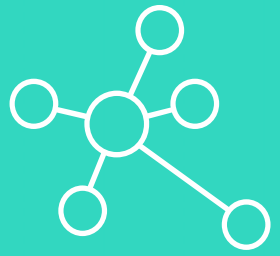






Projet Cloud Computing

M IOT - H3Hitema

Présentation du projet

- Vous êtes un ML Engineer Junior pour un client (fictif ou non) et vous devez présenter une mini pipeline data, de l'élaboration d'une problématique pertinente, la collecte de données, le développement d'une API, le paquetage Docker la tester et rédiger une documentation claire et explicite.

Livrables

- Un rapport PDF (minimum 4 pages) présentant l'élaboration technique du projet ainsi que les différentes étapes.
- Un repo git (public) avec un README.md clair et précis qui peut servir de documentation du projet.
- Un lien vers votre application hébergé sur votre VM ou sur le cloud de votre choix (Heroku, aws…)
- De remplir le gsheet mis à disposition par le délégué avec le lien vers votre projet.

Contraintes

- Data avec **minimum 15 colonnes** avec **au moins une colonne représentant une date et 3 colonnes catégorielles**
- Exposer **concrètement votre problématique / question à définir** : quel est le but de votre modèle et à quelle situation métier peut il correspondre ?
- Une documentation impeccable du code, avec des **commentaires** et de la **docstring**

- **BONUS(+1,5) SI VOUS CHOISISSEZ DES DATA NON TRAITEES EN COURS**

Le workflow

Collect

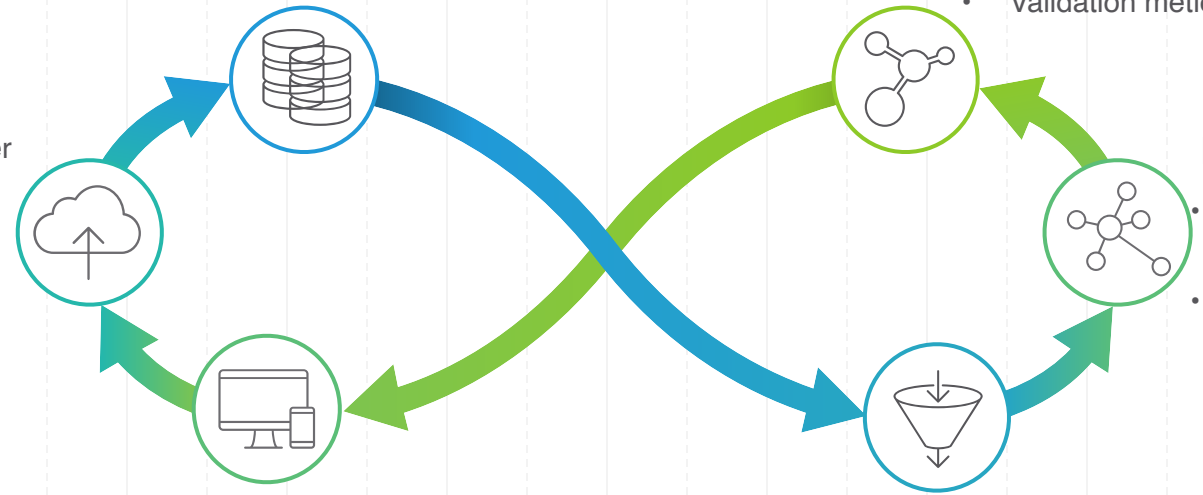
- Centralization of data sources
- Automating data processing (script/pipeline)

Tests

- Unitaire, ;odulaire & intégration
- Validation métier

Continuous Improvement& Quality

- Use of new customer data to feed the model
- CI / CD Bitbucket



Models

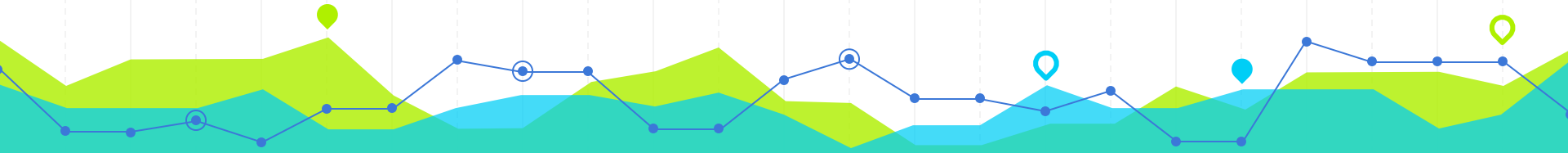
- Supervised model : tree methods
- POO Modeling

Deployment & production

- Docker containerization with FastAPI
- Run on GCP VM

Analysis & refining

- Cleaning
- Feature selection



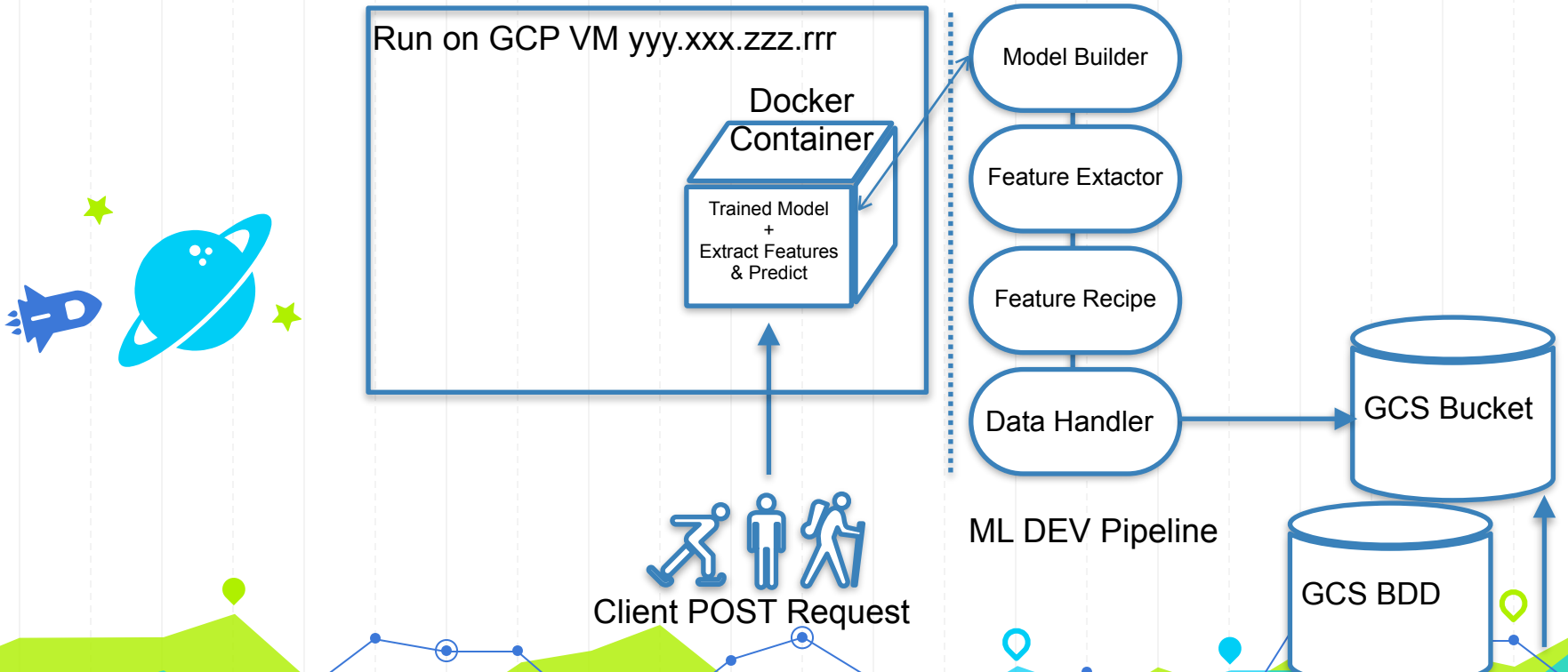
Les différentes étapes du projet

ARCHITECTURE DU PROJET

```
.
ml_template_api
|   |
|   ├── main.py
|   ├── requirements.txt
|   ├── Dockerfile
|   |   └── ml
|   |       ├── __init__.py
|   |       ├── model.joblib
|   |       ├── EDA.ipynb
|   |       ├── algo_pipeline_demo.ipynb
|   |       └── model.py
|   |           └── utils
|   |               ├── __init__.py
|   |               └── utils.py
|
```

ARCHITECTURE GCP

ML Batch Processing pipeline & predict on the fly



Partie 1 : Analyse graphique des données (EDA)

- **A EFFECTUER DANS LE NOTEBOOK EDA.ipynb**
- Diagrammes de répartition des données (type gaussienne sur les données)
- Vérification du nombre de données, si plusieurs données sont peu représentés (<3%) alors regrouper dans une seule et même catégorie, 1 pie chart avant/apres
- Nettoyage des données manquantes, encodage (OneHot, dictionnaire ou Sklearn Encoder)
- Boites à moustache si données extrêmes avec commentaire (que prévoyez vous pour ces individus)
- Heatmap + observations sur les corrélations si pertinent

Partie 2 : Model Building

- Expliquer l'algorithme choisi (quelque ligne suffisent) avec ces différents paramètres (ex pour les CART: max_depth, n_estimators,….)
- Affichage des coefficients/ accuracy + commentaires
- Le model est-il en overfitting/underfitting/OK ?
- Que proposez vous comme pistes d'améliorations ?
- Coder une class pour sauvegarder votre model

Partie 3 (optional) : Feature Importance

- Affichage sous forme de barplot la feature importance de votre algorithme à l'aide de la fonction feature_importance des sklearn.estimator (les objet algorithmes de sklearn)
- Autre forme d'affichage si vous avez le temps

Partie 4 (optional) : Model Réexécution avec les features sélectionnés

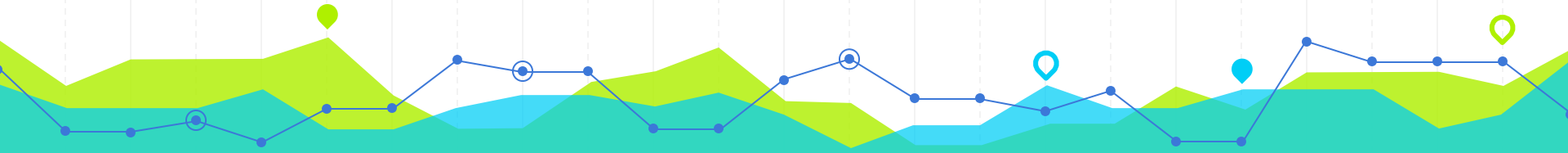
- Utiliser une cross validation pour vérifier le « predictive power » de vos features. Expliquer en quelque phrases à quoi sert la cross validation.
- Affichage des metrics standard et commentaire sur la pertinence des features selectionnées

Partie 5: API, Conteneurisation et déploiement GCP

- Construction d'une api avec une des librairies suivantes :
  - Fastapi
  - Flask
  - Streamlit
- Livraison sur git avec documentation et test (un fichier bash test.sh qui effectue une requête de test suffit ou la doc interactive fastapi)
- Déployer votre api sur votre vm google (cf gsheet)

Partie 6: Pacquaging, POO & refactoring

- Implémenter votre projet suivant l'architecture définit au préalable
- Les classes de votre pipeline DEV seront implémenter dans le fichier ./ml_template_api/ml/utils/utils.py
- Les ports à utiliser sont le 8888 pour jupyter et le 5000 ou 8000 pour votre api
- Vous pouvez utiliser des data en local
- Le notebook algo_pipeline_demo.ipynb sera votre notebook « fil rouge » dans lequel vous exposerez toute les classes nécessaire a l'exécution de votre pipeline
- BONUS (+1) : utiliser les argparse python afin de lancer votre script model.py de la façon suivante `*python model.py --model=linearRegression —split=0.1*`



« Be the measure of quality.
Many people are not used to an environment or what is expected is excellence. »

S.Jobs