

---

# Projet Certification Développeuse Data - IA

— Prédiction du prix d'un bien  
immobilier —

---

# Sommaire

Introduction

Partie 1 : Les bases du projet	p 2
Le besoin client	p 2
Un état de l'art	p 2 - 3
Traduction et choix techniques	p 3 - 16
Partie 2 : La mise en œuvre du projet	p 17 - 18
La gestion du projet	
Partie 3 : Bilan et axes d'améliorations du projet	
Bilan	p 19
Axes d'améliorations	p 19 - 20
Conclusion	p 20
Remerciements	p 20
Bibliographie	p 20

Annexes

- Annexe 1 : Description du dataset
- Annexe 2 : Importation des CSV
- Annexe 3 : Création des DataFrames Mutation, Cadastre et Bien
- Annexe 4 : Schéma EA
- Annexe 5 : Script SQL
- Annexe 6 : Backup automatique
- Annexe 7 : Connection Power Bi / SQL Server
- Annexe 8 : Exemples Visualisation Globale
- Annexe 9 : Map générale
- Annexe 10 : Box plot valeurs foncières
- Annexe 11 : Heatmap des NaN
- Annexe 12 : Traitement type\_local
- Annexe 13 : Résultats Algorithmes
- Annexe 14 : Script API
- Annexe 15 : DockerFile
- Annexe 16 : Azure
- Annexe 17 : Exemple Trello
- Annexe 18 : Github

# Introduction

Ce rapport finalise près de deux ans de formation auprès de Simplon. Après 7 mois de cours intensifs, je termine par ce projet mon année d'alternance en entreprise.

Genapi l'entreprise qui m'a accueillie cette année est un éditeur de logiciel spécialisé dans le notariat depuis 1988 et membre fondateur de Septeo.

Ce projet de certification n'est pas un projet d'entreprise mais un projet personnel. Le notariat ayant des données vraiment sensibles, il m'est impossible de présenter un projet issu de mon travail auprès d'eux. C'est pourquoi j'ai choisi de créer un projet fictif mais ayant pris racine auprès de mon entreprise.

## Partie 1 : Les bases du projet

### Le besoin client

Aujourd'hui, le marché de l'immobilier est tendu du fait de la baisse des taux de prêts. Beaucoup vendent et beaucoup achètent. Les deux parties veulent un prix juste. C'est pourquoi les agents immobiliers doivent estimer au mieux le prix des biens qu'ils vont devoir vendre et ainsi contenter leurs clients.

Il leur faut donc un logiciel ou une API permettant de prédire le prix du bien selon certains critères qui définiront le bien à estimer.

Le client : Agences Immobilières

Le besoin :

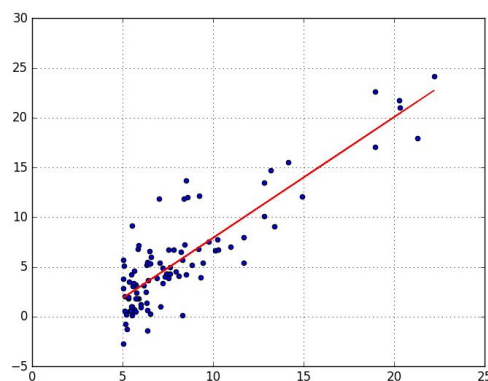
- Prédire le prix de biens immobiliers
- Avoir une API pour entrer les critères permettant la prédiction et avoir un rendu visuel du résultat.

### Un état de l'art

La prédiction d'un prix (variable dépendante) en fonction de critères ou variables prédictives (variables indépendantes) relève d'un problème de régression. Le modèle de régression linéaire multiple est l'outil statistique le plus habituellement mis en œuvre pour l'étude de données multidimensionnelles. Cas particulier du modèle linéaire, il constitue la généralisation naturelle de la régression simple.

Les algorithmes de régression linéaire modélisent la relation entre des variables prédictives et une variable cible. La relation est modélisée par une fonction mathématique de prédiction. Le cas le plus simple est la régression linéaire. Elle va trouver une fonction sous forme de droite pour estimer la relation. La régression multiple intervient quand plusieurs variables explicatives interviennent dans la fonction de prédiction. Et finalement, la régression polynomiale permet de modéliser des relations complexes qui ne sont pas forcément linéaires.

Visualisation d'une régression linéaire:



## Traduction & choix techniques

Le projet a été fait en plusieurs étapes :

### Création d'une base de données

Les données proviennent du site [data.gouv.fr](https://www.data.gouv.fr), elles sont open data c'est-à-dire qu'elles sont libre d'accès et d'exploitation.

<https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncières-geolocalisées/>

Les données sont récupérées sous format CSV pour moins de complexité lors de l'importation des données dans les différentes plateformes utilisées. Mais il existe d'autres formats comme le format XML, Excel, etc.

L'idée première était de combiner ces données avec celles issues d'un scrapping sur le site [seloger.com](https://www.seloger.com). Mais après recherches, les données de Data Gouv proviennent en partie de SeLoger. Il convient donc de ne garder que celles récupérées sur Data Gouv. Je récupère les données pour les années 2017-2018-2019 qui représentent 40 colonnes et 7 453 214 entrées. (cf Annexe 1: Description du dataset).

Je souhaite créer une base de données DVF ayant trois tables :

- Mutation
- Bien
- Cadastre

L'importation d'un si gros fichier sur SQL Server est impossible. Il faut donc préparer les données, pour cela j'utilise le langage python et un notebook qui rendront le traitement plus facile.

Après l'importation des données sur le notebook (cf Annexe 2 : Importation des CSV), je remarque que les datasets ont beaucoup de colonnes, qu'il y a des NaN (ou des entrées n'ayant pas de valeur), les colonnes '*id\_mutation*' et '*id\_parcelle*' ne sont pas uniques et il n'y a pas de colonne '*id\_bien*'. Pour arriver à mon but je vais faire du feature engineering et de la data analysis.

Pour simplifier les traitements à venir et la création des trois tables in fine, je vais concaténer les datasets en un dataframe unique.

Puis je supprime certaines colonnes qui ne vont pas me servir ou qui sont remplies de NaN. Je fais également un traitement pour la colonne '*valeur\_fonciere*' où je prends le parti de supprimer les lignes ayant des NaN. On pourrait choisir un autre traitement comme trouver des biens qui ressemblent aux biens à valeur nulle, faire une moyenne de ces biens et ainsi remplacer les NaN. Pour les id qui ne sont pas uniques je choisis de ne garder que les premiers pour '*id\_mutation*' et que les derniers pour '*id\_parcelle*'.

La dernière transformation est la création d'un '*id\_bien*'.

Je finis cette phase préparatoire par la création de trois dataframes (cf Annexe 3 : Création des DataFrames Mutation, Cadastre et Bien).

Avant de passer à la partie SQL Server je crée un modèle EA car ma base de données est de type relationnelle. Le modèle entité-association (EA) est un modèle de données ou diagramme pour des descriptions de modèles conceptuels. (cf Annexe 4 : Schéma EA)

## Importation dans SQL Server

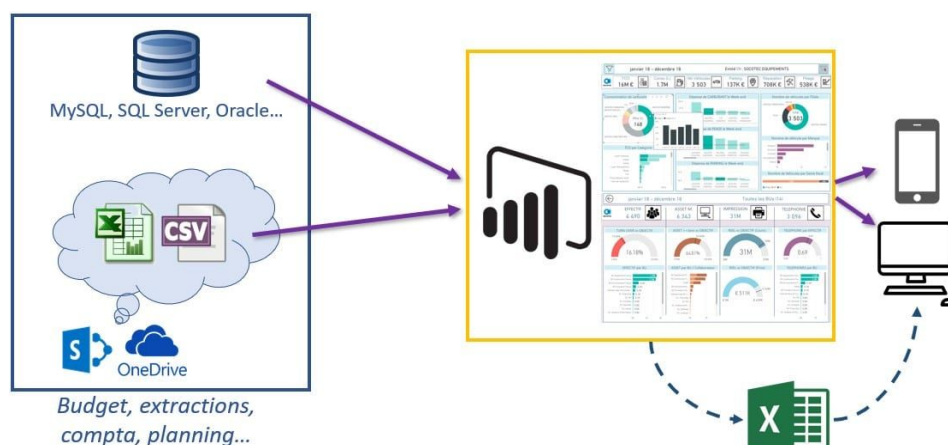
J'ai choisi SQL Server car il offre des connecteurs permettant une utilisation conjointe avec Power Bi que je vais utiliser pour mes visualisations. De plus il est facile avec SQL Server d'instancier des backups automatiques.

Tout d'abord je crée la database et les trois tables et ensuite j'importe les données dans ces tables. (cf Annexe 5 : Script SQL).

Pour finir avec SQL Server je mets en place des backups automatiques. (cf Annexe 6 : Backup automatique).

## Visualisation

Elle est faite sur Power BI avec les données brutes récupérées par une connexion avec SQL Server (cf Annexe 7 : Connection Power Bi / SQL Server).



Ce schéma nous montre un type de workflow avec Power Bi. Les données peuvent être intégrées sous différentes formes (fichiers, connecteurs avec des bases de données ou connecteurs Cloud). Une fois le dashboard créé, il est possible de le publier pour permettre son accès sur diverses plateformes (mobiles ou fixe). De plus chaque graphique permet l'export de ses données sous format Excel.

La visualisation sur Power Bi permet une forte implication du client, en effet en plus d'être interactive cela permet d'avoir de la BI en self-service, c'est-à-dire que les données sont disponibles quel que soit le niveau hiérarchique de la personne (du directeur à la secrétaire en passant par les autres employés) sans formation sur le produit. De plus il est possible d'affecter des droits selon les utilisateurs pour limiter les accès aux données.

Le dashboard est composé de deux feuilles :

- Visualisation globale (cf Annexe 8 : Exemples Visualisation Globale) donne une vision large des données suivant différents graphiques :
  - ◆ Le nombre de ventes par année avec en option une hiérarchie qui permet l'évaluation des ventes plus en détails (par trimestre, par mois et par jour)
  - ◆ Le nombre de biens par département dans un premier temps puis avec la hiérarchie par ville
  - ◆ La répartition par type de bien et par nombre de pièces
  - ◆ La répartition par type de mutation (Vente, Echange, Expropriation, etc)
- Map générale (cf Annexe 9 : Map générale) donne une dimension géographique de ces données en représentant la moyenne des valeurs foncières en fonction des régions.

La visualisation met en évidence certains points comme, la présence de différents types de mutation autre que la vente, une bonne partie des types de biens ne sont pas labellisés et une grande variation des prix de vente.

Avant de travailler sur le Machine Learning il faudra refaire du feature engineering et de la data analysis pour répondre au mieux au besoin client et faire un bon entraînement d'algorithmes.

## ***Feature engineering et Data analysis pour le Machine Learning***

Le but de cette partie est de :

- déterminer la colonne target
- traiter les valeurs NaN (suppression ou transformation)
- mettre en évidence les colonnes qui vont influencer la colonne target

### ***La colonne Target***

Tout d'abord je regarde la composition du dataframe par la commande `describe()` qui va permettre de se focaliser sur les données numériques. On remarque ainsi une grande disparité dans les valeurs foncières, les prix allant de 0,01€ à 1 750 000 000€. Il faudra donc soit faire des fourchettes de prix soit réduire les données. (cf Annexe 10 : Box plot valeurs foncières)

La grande variance de prix s'explique par le fait qu'il y ait des biens qui sont issus d'expropriations ou de donations et des biens qui font partie de grands patrimoines.

Je choisis de ne prendre que les valeurs comprises entre le 1er et le 3ème quartile c'est-à-dire les valeurs foncières entre 45 000€ et 220 000€. Il est évident que c'est une des méthodes, on pourrait choisir de faire différentes fourchettes de prix ou encore de prendre la totalité.

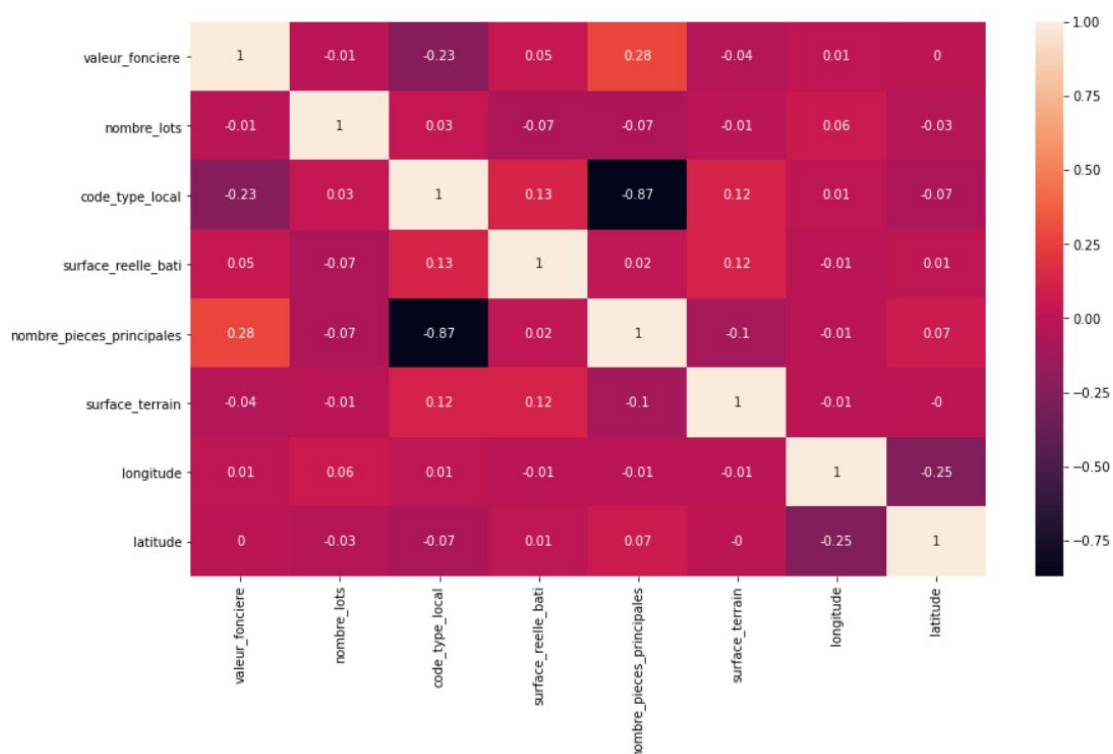
## Traitement des NaN

Ensuite je regarde où sont les valeurs NaN. La visualisation par heatmap, ou carte de chaleur, donne aux données un aspect visuel qui est moins complexe à saisir qu'un tableau (cf Annexe 11 : Heatmap des NaN). Un traitement s'avère nécessaire pour réduire le bruit qui conduirait à des résultats faussés.

En premier lieu pour répondre au mieux au besoin du client je ne récupère que les données de ventes immobilières. Ensuite la colonne 'type\_local', a une grande quantité de valeurs sont labellisées 'None'. À l'aide d'un plot, j'essaie de comparer le comportement de ces 'None' avec les autres types pour trouver celui qui leur ressemble le plus et les changer par ce type de local. (cf Annexe 12 : Traitement type\_local). Dans notre cas les 'None' vont avoir un comportement proche du type 'Local industriel'. Je fais donc le remplacement.

## La corrélation

Je passe maintenant à l'étude de la corrélation entre colonnes. La corrélation met en évidence les relations entre variables du jeu de données. Il est important de quantifier l'intensité de dépendance des données les une par rapport aux autres. Une heatmap permet de bien visualiser la matrice de corrélation.



La matrice montre une forte corrélation négative entre le nombre de pièces et le code type de local. La feature selection pour le dataframe final sera :

- 'valeur\_fonciere' (valeur target)
- 'code\_departement' (qui seront regroupés en régions)
- 'type\_local'
- 'nombre\_pieces\_principales'

La création de la colonne 'regions' se fait par la création de 6 listes de régions, 'NordOuest', 'NordEst', 'SudOuest', 'SudOuest', 'RegionParis' et 'DomTom'. Ces listes sont composées des différents 'code\_departement'. La colonne 'regions' est alors remplie selon la correspondance des listes et de la colonne 'code\_departement'.

## Machine Learning

Pour répondre au besoin du client qui est une prédiction de prix, il faut résoudre un problème de régression comme expliqué dans la partie '**Un état de l'art**'.

### Phase 1 : Première approche des algorithmes

#### Dummies

La transformation des colonnes 'regions' et 'type\_local' en dummies car les algorithmes ne prennent pas du texte. En effet la création de dummies consiste en la conversion des variables catégoriques en dummy ou en variable indicative. Chaque donnée devient une colonne et les entrées de ces colonnes sont soit 0 qui correspond au fait que la variable n'est pas présente soit 1 quand la variable est présente.

#### Scaling

Le Scaling ou mise à l'échelle est une méthode utilisée pour normaliser la plage de variables ou de fonctionnalités indépendantes des données. Dans le traitement des données, elle est également connue sous le nom de normalisation des données et est généralement effectuée lors de l'étape de prétraitement des données. Ici on va tester le MinMaxScaler() et le StandardScaler(), de la librairie Sklearn, sur le nombre de pièces.

Le **MinMaxScaler** également connu sous le nom de mise à l'échelle min-max ou normalisation min-max. C'est la méthode la moins complexe et elle consiste à redimensionner la plage de données en [0, 1] ou [-1, 1]. La sélection de la plage cible dépend de la nature des données. La formule générale est  $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$ .

Le **StandardScaler** suppose que les données sont normalement distribuées dans chaque entité et les met à l'échelle de telle sorte que la distribution est centrée autour de 0 avec un écart-type de 1. La moyenne et l'écart type sont calculés pour l'entité, puis l'entité est mise à l'échelle en fonction de  $x' = \frac{x - \text{mean}(x)}{\text{stdev}(x)}$

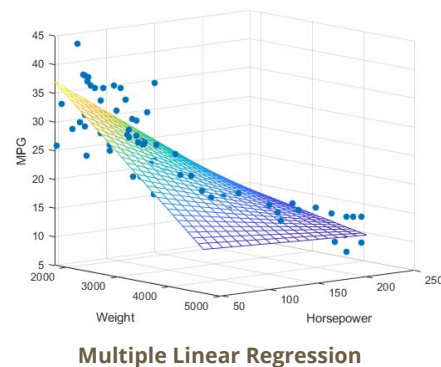
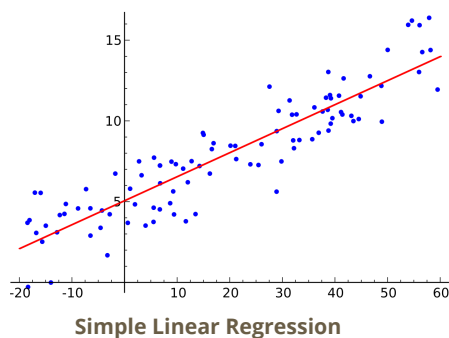


## Test d'algorithmes

On peut maintenant attaquer la partie test des algorithmes avec Sklearn. Scikit-Learn est une librairie python permettant l'accès à de nombreux algorithmes et de méthodes de preprocessing. Cette librairie est fournie avec une documentation riche qui décomplexifie les algorithmes en donnant des exemples d'exécution.

### → Linear Regression :

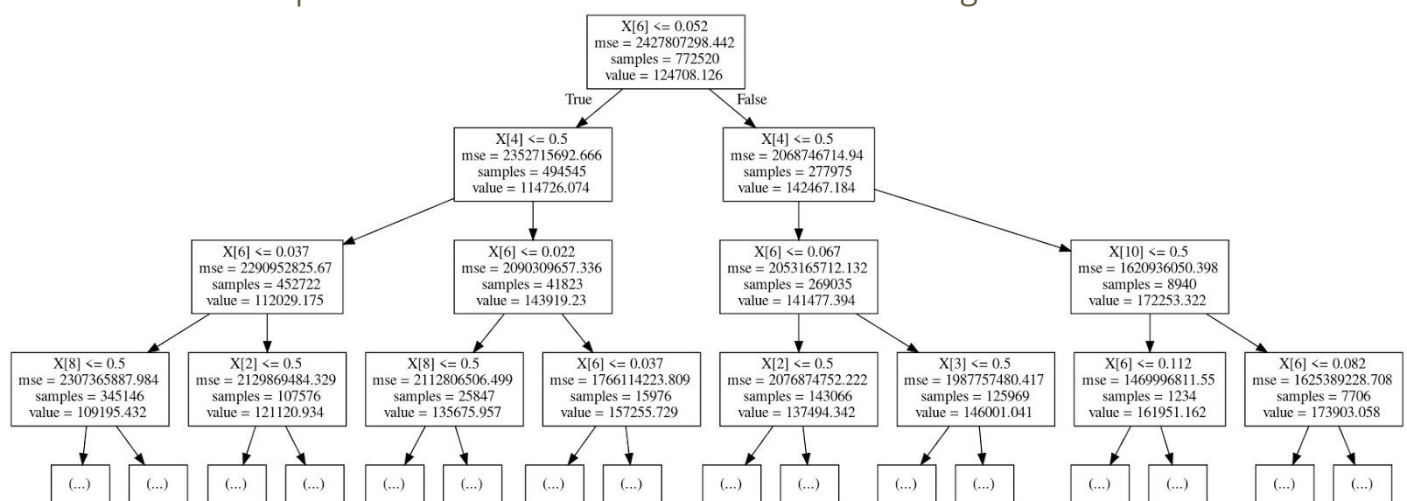
La régression linéaire effectue la tâche de prédire une valeur de variable dépendante (y) basée sur une variable indépendante donnée (x). Ainsi, cette technique de régression découvre une relation linéaire entre x (entrée) et y (sortie). Par conséquent, le nom est la régression linéaire. Si nous traçons la variable indépendante (x) sur l'axe des x et la variable dépendante (y) sur l'axe des y, la régression linéaire nous donne une ligne droite qui correspond le mieux aux points de données.



### → Decision Tree :

Outils de prise de décision qui utilise une structure arborescente de type organigramme ou est un modèle de décisions et de tous leurs résultats possibles.

Cet algorithme appartient à la catégorie des algorithmes d'apprentissage supervisé. Il fonctionne à la fois pour les variables de sortie continues et catégorielles.



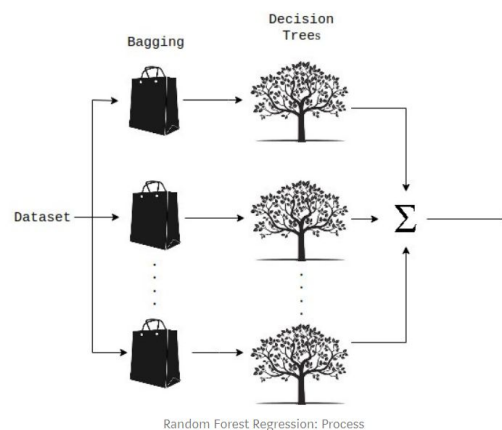
Exemple d'arbre de décision

### → Ridge :

Version régularisée de l'algorithme Linear Regression. On ajoute un terme régularisé qui oblige l'algorithme d'apprentissage à s'adapter aux données et aide à maintenir les poids aussi bas que possible.

### → Random Forest : ou forêt aléatoire

Technique d'ensemble capable d'exécuter à la fois des tâches de régression et de classification à l'aide d'arbre de décision et d'une technique Bagging. Le bagging implique la formation de chaque arbre de décision sur un échantillon de données différent où l'échantillonnage est effectué avec remplacement. L'idée est de combiner plusieurs arbres pour déterminer la sortie.



### → XGBoost :

XGBoost est un algorithme très populaire en Machine Learning et qui domine les compétitions Kaggle.

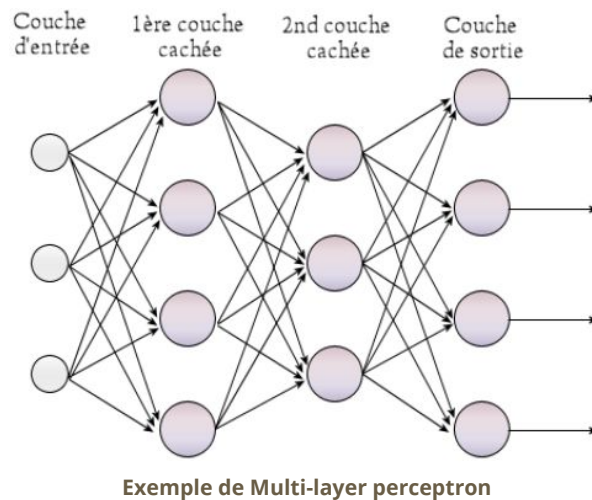
Implémentation open source optimisée de l'algorithme de boosting de gradient, dont le principe est de combiner les résultats d'un ensemble de modèles plus simples et plus faibles afin de fournir une meilleure prédiction. Il travaille de manière séquentielle ce qui le rend plus lent mais ce qui permet de capitaliser sur les exécutions précédentes.

### → MLPRegressor :

Multi-Layer Perceptron (MLP) est un type de réseau neuronal artificiel (ANN) organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau à propagation directe (feedforward).

Un MLP se compose d'au moins trois couches : une couche d'entrée, une couche cachée et une couche de sortie. À l'exception des nœuds d'entrée, chaque nœud est un neurone qui utilise une fonction d'activation non linéaire.

La classe MLPRegressor implémente un MLP qui s'entraîne en utilisant la rétropropagation (backpropagation) sans fonction d'activation dans la couche de sortie, qui peut également être considérée comme utilisant la fonction d'identité comme fonction d'activation. Par conséquent, il utilise l'erreur carrée comme fonction de perte et la sortie est un ensemble de valeurs continues.



## Conclusion de la phase 1

Les résultats sont compilés dans un tableau pour faciliter la prise de décision sur l'algorithme à choisir ([cf Annexe 13 : Résultats Algorithmes - Phase 1](#)). Les résultats ne sont pas bons, les  $r^2$  score sont bas (avec un max à 0.12), il faudrait reprendre toute la partie préparation pour voir les manquements commis. Pour essayer d'arranger les résultats on peut également essayer de tester d'autres paramètres.

## Phase 2: Une autre idée de preprocessing

Cette phase fait suite à la première qui avait mis en évidence un potentiel problème de Feature Engineering. Pour cela je suis repartie de la base de donnée brute et j'ai ensuite modifié mon process.

### Feature Engineering

#### → Valeurs manquantes

##### ◆ Variables catégoriques

Les valeurs catégoriques ayant des valeurs manquantes sont mises en évidence : 'adresse\_nom\_voie' et 'type\_local'.

Une fonction va remplacer les valeurs manquantes au niveau de ces variables par une nouvelle catégorie 'Missing'.

##### ◆ Variables numériques

Les variables numériques avec des valeurs manquantes sont mises en évidence : 'code\_type\_local', 'surface\_reelle\_bati', 'nombre\_pieces\_principales', 'surface\_terrain', 'longitude', 'latitude'.

Une fonction va y remplacer ces valeurs manquantes par 0.

## → Traitement des variables

### ◆ Variables numériques

Les valeurs numériques et plus précisément celles contenant les valeurs continues vont être transformées grâce à une fonction logarithmique afin d'obtenir une meilleure distribution. Cette étape aide les modèles d'apprentissage linéaire.

### ◆ Variables catégoriques

Les catégories des variables présentes dans moins de 1 % des observations vont être supprimées.

Ensuite je vais transformer le texte par des chiffres car les algorithmes ne prennent en charge que le numérique. Ici les colonnes transformées sont : *'nature\_mutation'*, *'type\_local'* et *'code\_departement'*.

## → Scaling

La mise à l'échelle est faite en dernier par la méthode de scaling de sklearn `MinMaxScaler()`.

## → Feature Selection

Scikit Learn propose plusieurs méthodes pour la sélection des colonnes.

Je choisis de faire avec la méthode `SelectFromModel()` qui est un méta-transformateur. Il peut être utilisé avec tout estimateur ayant un attribut *"coef\_"* ou *"feature\_importances\_"* après ajustement. Mon choix d'estimateur se porte sur l'estimateur Lasso. Les entités sont considérées comme non importantes et supprimées, si les valeurs *coef\_* ou *feature\_importances\_* correspondantes sont inférieures au paramètre de seuil fourni.

Dans notre cas, les colonnes ayant le plus d'importance sont : *'nature\_mutation'*, *'code\_departement'*, *'code\_type\_local'*, *'type\_local'*, *'surface\_reelle\_bati'*, *'surface\_terrain'*.

## ***Test des algorithmes***

Cette étape est la même que la phase 1, je reprends les mêmes algorithmes et je les entraîne et les teste sur cette sélection. Les résultats sont regroupés dans un tableau ([cf Annexe 13 : Résultats Algorithmes - Phase 2](#)).

## ***Conclusion de la phase 2***

Malheureusement les résultats ne sont pas bons les R2 Score sont négatifs ou proches de 0 et les RMSE dépassent le maximum des prix de vente.

Cette phase a permis de tester une autre approche de feature engineering qui certes ne fonctionne pas mais qui ouvre des perspectives d'amélioration. La méthode de la phase 1 a de meilleurs résultats donc je vais repartir du feature engineering effectué pour cette phase pour créer la phase 3 avec une autre librairie.

### ***Phase 3 : En automatique avec Pycaret***

Cette partie est faite sur un notebook de Google Colab et PyCaret.

PyCaret est une bibliothèque de machine learning open source en python. Elle permet de passer de la préparation des données au déploiement du modèle rapidement en utilisant un environnement préalablement choisi. Toutes les étapes, dont le preprocessing, sont automatiquement enregistrées dans une pipeline qui pourra être déployée en production, il faudra pour cela les initialiser dans le setup du modèle.

Le module de régression de PyCaret est un module d'apprentissage automatique supervisé qui est utilisé pour estimer les relations entre une variable dépendante (souvent appelée «variable de résultat» ou «cible») et une ou plusieurs variables indépendantes (souvent appelées «caractéristiques», «prédicteurs», ou «covariables»). L'objectif de la régression est de prédire des valeurs continues telles que la prévision du montant des ventes, la prévision de la quantité, la prévision de la température, etc. Ce module fournit plusieurs fonctionnalités de prétraitement qui préparent les données pour la modélisation via la fonction de configuration. Il dispose de plus de 25 algorithmes prêts à l'emploi et de plusieurs graphiques pour analyser les performances des modèles formés.

#### ***Le preprocessing***

Pour les fonctions de preprocessing je choisis le module Scale and Transform de PyCaret. Ce module va faire de la mise à l'échelle 'Scaling' par de la standardisation qui est une technique souvent appliquée dans le cadre de la préparation de données pour le machine learning. Son objectif est de mettre à l'échelle les valeurs des colonnes numériques, pour notre dataset la colonne *nombre\_pieces\_principales*, sans fausser les différences dans les plages de valeurs ni perdre des informations. De plus je choisis d'appliquer l'option 'Target Transformation' qui va modifier la distribution de la colonne target 'valeur\_fonciere' pour en faire une distribution normale.

Pour ce qui est de la préparation des données catégoriques, PyCaret fait automatiquement du 'One Hot Encoding'. Les algorithmes ne peuvent pas travailler directement avec des données catégorielles. Elles doivent être transformées en valeurs numériques avant d'entraîner le modèle. Le type d'encodage le plus courant est le One Hot Encoding qui va changer les catégories en entité distincte, ou colonne, dont les données seront des valeurs binaires, 1 ou 0.

#### ***Le modèle***

Pour choisir l'algorithme que je vais utiliser la fonction `compare_model()`. Cette fonction forme et compare les métriques d'évaluation courantes à l'aide de la validation croisée k-fold pour tous les modèles disponibles dans la bibliothèque du module que vous avez importé. Les paramètres d'évaluation utilisés sont les suivants : MAE, MSE, RMSE, R2, RMSLE, MAPE.

- MAE ou Mean Absolute Error: une erreur est la différence absolue entre les vraies valeurs et les valeurs prédites. MAE est la moyenne de cette erreur.
- MSE ou Mean Squared Error est la moyenne du carré de l'erreur.

- RMSE ou Root Mean Square Error est l'écart type des erreurs qui se produisent. C'est la même chose que le MSE mais ici c'est la racine carrée de la valeur qui est prise en compte.
- R2 ou R Squared indique à quel point la droite de régression est proche des valeurs de données réelles. La valeur de R2 se situe entre 0 et 1 où 0 indique que ce modèle ne correspond pas aux données et 1 que le modèle correspond parfaitement.
- RMSLE ou Root Mean Squared Logarithmic Error variante du RMSE qui utilise les logarithmes des valeurs
- MAPE ou Mean Absolute Percentage Error

La sortie de la fonction est un tableau montrant le score moyen de tous les modèles. (cf Annexe 13 : Résultats Algorithmes - Phase 3).

Parmi les algorithmes testés il y a :

- Linear Regression
- Decision Tree
- Random Forest
- XGBoost

Après avoir choisi le modèle, le XGBoost qui fait parti des 3 meilleurs algorithmes testés, j'utilise une fonction pour le créer `create_model()` va donner un modèle entraîné mais avec ses hyperparamètres par défaut et une fonction pour ajuster ces derniers, `tune_model()`. L'optimisation des hyperparamètres passe par le test de différentes combinaisons. La combinaison ayant les meilleurs résultats pour la prédiction sera sauvegardée.

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.9, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=110, min_child_weight=2, missing=None, n_estimators=200,
             n_jobs=-1, nthread=None, objective='reg:linear', random_state=42,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=0.7, verbosity=0)
```

## ***Finalisation du modèle et pipeline***

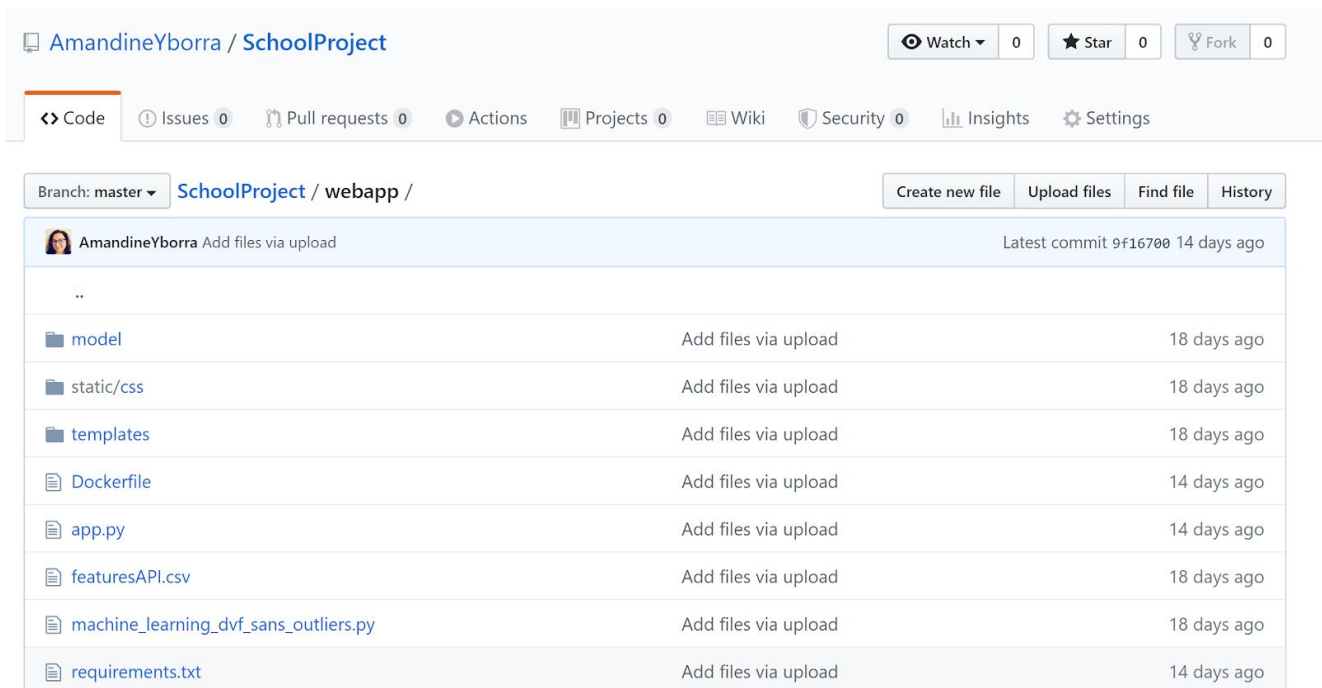
La finalisation du modèle passe par la fonction `finalize_model()` de PyCaret qui va fixer la pipeline et tester sur le reste des données le modèle. Cette étape est la dernière étape de test du modèle et de la pipeline.

La pipeline comme expliqué précédemment comportant le preprocessing et le modèle avec les ajustements des hyperparamètres se charge au fur et à mesure des étapes. Pour la finalisation du modèle, ce dernier a bien créé la pipeline et est prêt à l'emploi.

Pour sauvegarder le tout la fonction `save_model()`. La fonction prend le modèle formé et enregistre la pipeline de transformation complet et l'objet modèle formé en tant que fichier pickle transférable pour une utilisation ultérieure.

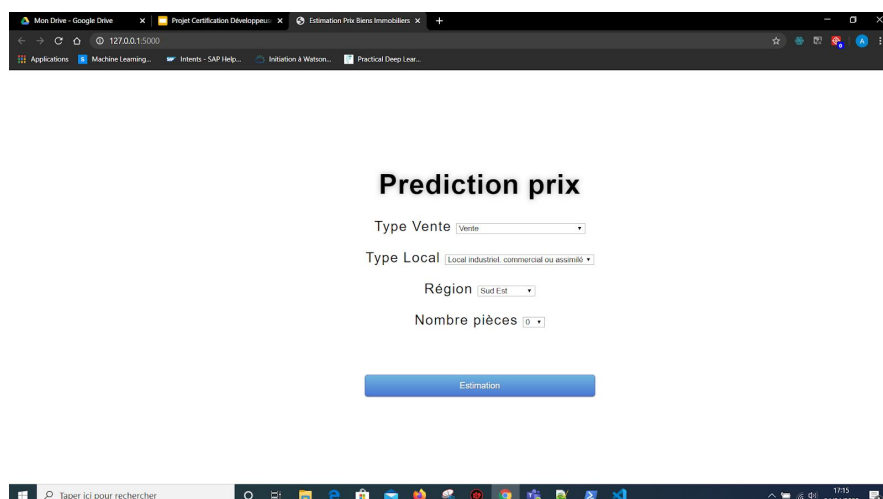
## Création d'une API

L'API va être créée avec Flask, framework léger qui permet d'éditer des API en python. De plus il est associé à des templates HTML et CSS pour la mise en forme. Le dossier pour la création de l'API devra être composé comme suit.



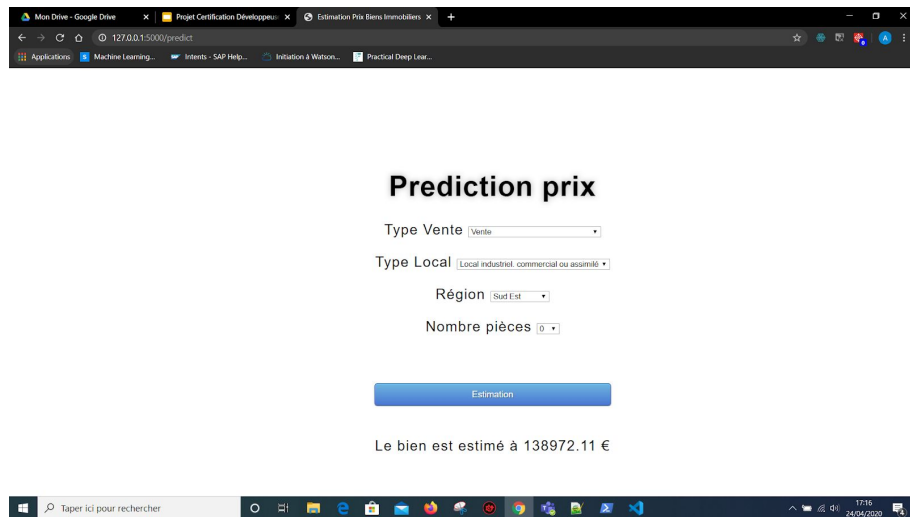
Avec en racine le script de l'API, le fichier *requirements.txt*, regroupant les librairies nécessaires au bon fonctionnement du modèle et de l'API. Un fichier csv avec seulement les colonnes pour récupérer les données du formulaire qui seront les inputs du modèle prédictif. Le dossier static contient le CSS, le dossier templates a le fichier HTML qui sert de page d'accueil et de formulaire. Et enfin le dossier model contient le modèle sauvegarder (pipeline de preprocessing avec l'algorithme entraîné).

Le script *app.py* (cf Annexe 14 : Script API), va à l'aide du fichier *index.html*, charger une page d'accueil qui sera un formulaire.





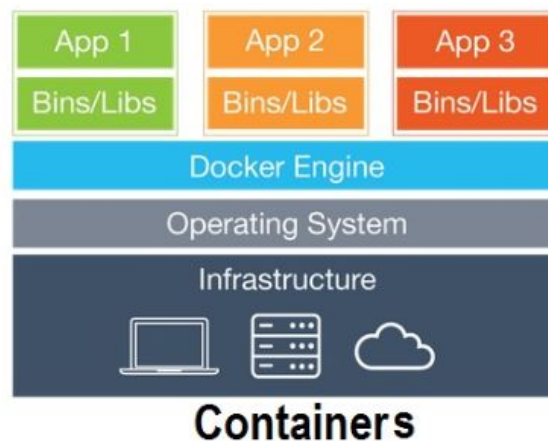
Une fois les données validées en appuyant sur le bouton “Estimation” elle sont récupérées et ajoutées au fichier CSV. Ensuite la prédiction est faite par le modèle chargé avec le csv en input. L’API charge ensuite la page avec la prédiction.



The screenshot shows a web browser window with the URL `127.0.0.1:5000/predict`. The page has a title "Prediction prix". Below the title are four input fields: "Type Vente" (dropdown menu), "Type Local" (dropdown menu), "Région" (dropdown menu), and "Nombre pièces" (text input). A blue button labeled "Estimation" is positioned below these fields. Under the button, a message states "Le bien est estimé à 138972.11 €". The browser's taskbar at the bottom shows various application icons and the system clock indicating 17:16 on 24/04/2020.

## Mise en place d'un conteneur

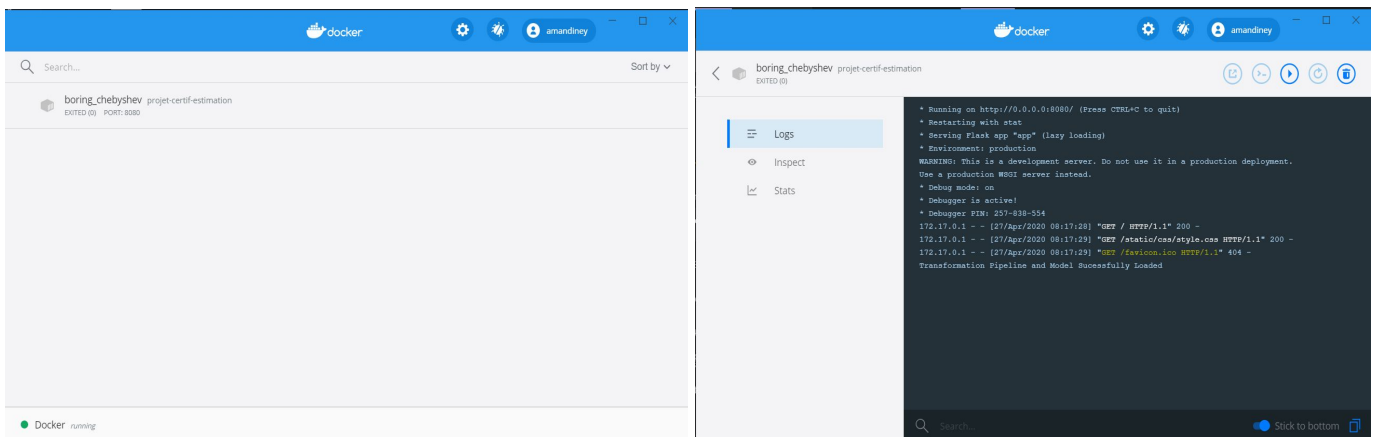
Une fois l'API Flask fonctionnelle je la mets dans un conteneur pour pouvoir par la suite la déployer plus facilement. Pour cela j'utilise Docker avec lequel on peut avoir rapidement des conteneurs embarquant tous les environnements des API facilitant le partage et le déploiement.



Pour mettre en place un conteneur il faut créer une image (cf Annexe 15 : DockerFile) qui doit être mise à la racine du dossier de mon application. Puis faire un build de cette image.



Résultat sur Docker Dashboard :



On peut tester que tout fonctionne bien en suivant le lien dans les logs du dashboard.

## Déploiement

Il se fait via le portail d'Azure pour que l'API soit accessible à tous et à tout moment. Azure favorise les déploiements rapides avec une interface intuitive et documentée. Son lien avec Docker permet de pusher directement l'image dans Azure Containers Registry.

Les étapes pour déployer mon application dans Azure (cf Annexe 16 : Azure) :

- Paramétrage d'un environnement Azure
  - ◆ Création d'un compte
  - ◆ Activation des crédits gratuits
  - ◆ Création d'un groupe de ressources
- Push de l'image Docker dans le Azure Container Registry
- Création d'une WebApp Azure et exécution

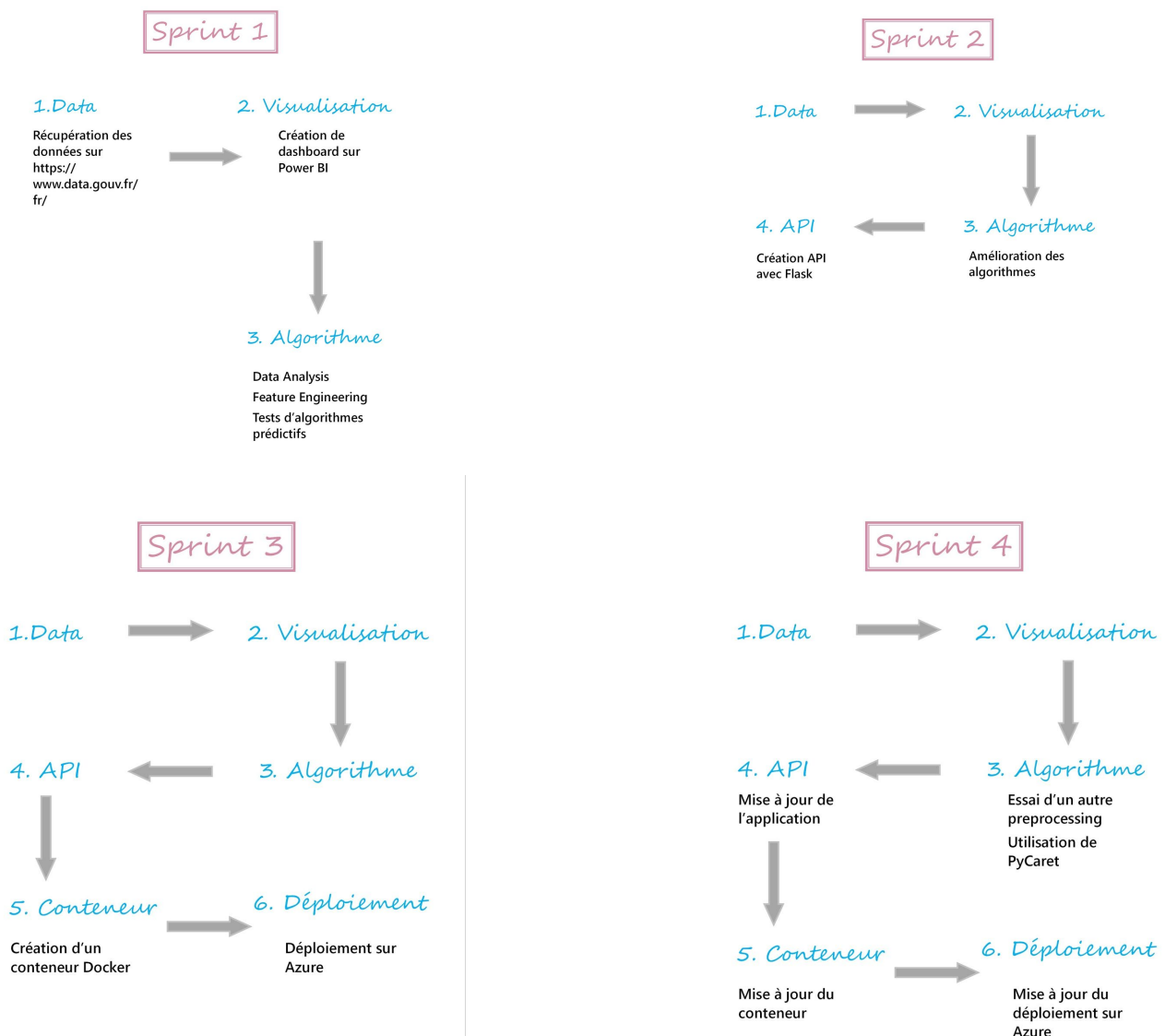
# Partie 2 : La mise en œuvre du projet

## La gestion du projet

La gestion c'est faite sur 3 axes de travail. Cela me permet ainsi d'avoir une vue d'ensemble sur mon projet, de me donner des objectifs à atteindre et avoir un meilleur suivi de mon travail. De plus ce travail de gestion de projet permet le suivi pour mon formateur et mon manager tout au long de celui-ci ainsi si j'ai des questions ou un blocage il leur est tout à fait possible de m'aider rapidement.

## Mise en place de sprints

Créer des sprints donne la vision des grandes étapes à effectuer pour mener à bien le projet. J'ai choisi de répartir ces étapes en trois sprints en base de travail et un quatrième pour des améliorations :



## **Mise en place d'un Trello**

Pour la mise en place du Trello j'ai essayé d'adopter une méthodologie agile en découpant les étapes en différentes tâches qui auront chacune une valeur d'effort. L'estimation des efforts a été faite avec le Scrum Master de ma tribu. Les tâches et les estimations sont dans des cartes que sont déplacées suivant les tâches accomplies. (cf Annexe 17 : Exemples Trello)

Lien Trello:

<https://trello.com/invite/b/BxzsRuO0/c3ade7ba1f472aa884df16455ec624fc/projet-%C3%A9cole-amandine>

## **Mise en place d'un backup automatique via github**

Le projet a eu plusieurs *update* sur le github au fur et à mesure de l'avancement et des modifications. Cela permet un suivi des formateurs et me garantit des sauvegardes en cas de problèmes. (cf Annexe 18 : Github)

Lien Github :

<https://github.com/AmandineYborra/SchoolProject>

## Partie 3 : Bilan et axes d'améliorations du projet

### Bilan du projet

Le projet a été très stimulant car il m'a permis de rassembler les connaissances acquises tout au long de cette formation. Même si les résultats obtenus ne sont pas ceux attendus, ce projet m'a aidé à me confronter aux différentes étapes de construction d'une application et je peux maintenant échanger plus aisément avec les différents acteurs : Data Analysts, Data Scientists et DevOps.

#### *Points négatifs*

Les scores des algorithmes ne sont pas bons (cf Annexe 13 : Résultats Algorithmes). On peut donc supposer un problème dû au traitement des données. Mais surtout l'algorithme utilisé pour la prédiction n'est pas efficient.

Les algorithmes utilisés sont lourds et mettent donc du temps à tourner. Le fait de faire la partie machine learning en local (sur ma machine) a été fastidieux et souvent mon kernel a été interrompu.

#### *Points positifs*

Les parties Data et Visualisation ont été les parties les plus passionnantes pour moi et j'aurai aimé prendre plus de temps pour améliorer cette partie.

Faire la partie API, Docker et WebApp avec Azure a été une découverte pour moi. La création de la WebApp a été la tâche la plus abordable pour moi que ce soit en compréhension ou en application. Le reste de cette partie a été plus difficile à appréhender mais à présent j'arrive à avoir un peu de recul sur les problèmes que je peux rencontrer et arriver à découper ce problème en plusieurs tâches plus accessibles.

Faire le travail de machine learning en plusieurs phases me conforte dans mon raisonnement initial. Cela m'a permis aussi de pousser les questionnements et le traitement de données.

### Axes d'améliorations

#### *Partie Data*

- ✓ Avoir des retours réguliers vers le métier ou le client
- ✓ Reprendre les données avec le client pour obtenir une base cohérente et propre
- ✓ Élargir les sources de données
- ✓ Effectuer d'autres traitements sur les valeurs aberrantes
- ✓ Garder la colonne '`code_departement`' au lieu de faire une colonne '`regions`'

## ***Partie Machine Learning***

- ✓ Essayer les algorithmes avec traitement des valeurs aberrantes
- ✓ Tester d'autres algorithmes de Deep Learning
- ✓ Faire tourner les algorithmes dans le cloud

## ***Partie API***

- ✓ Création d'une page prédiction
- ✓ Création d'une page d'accueil
- ✓ Création d'un système de login/password

## **Conclusion**

Je conclus cette année avec la présentation de ce projet. Ces deux années ont été riches tant professionnellement que personnellement. Étant de base néophyte dans le domaine de l'informatique, j'en ressors grandie et surtout avec l'envie d'en apprendre plus dans les domaines qui me passionnent et que j'ai découvert : Data Analysis et Business Intelligence.

Ce projet tout en étant une fin marque le début d'une activité professionnelle auprès de Genapi.

## **Remerciements**

Aux différents formateurs, Driss Benchakroune, Hatem Kallal, David Azria et Benjamin Dalard qui m'ont suivie et soutenue tout au long de ces deux ans ainsi qu'Élodie Boyer pour le suivi.

À mon équipe rencontrée lors de la formation, Serge Adomayakpo, Nassim Laouiti, Qaïs Amini et Haïfa Ben Khalaf qui m'apportent un plein d'énergie.

À ma tribu Serenity et l'équipe dirigeante de Genapi pour leur confiance, leur soutien et pour me donner envie de toujours m'améliorer.

Et merci à vous Jury, qui prenez du temps pour nous !

## **Bibliographie**

Documentation générale : <https://fr.wikipedia.org/>

Documentation SQL Server :

<https://docs.microsoft.com/fr-fr/sql/sql-server/?view=sql-server-ver15>

Documentation Power Bi : <https://docs.microsoft.com/fr-fr/power-bi/>

Documentation Scklearn : <https://scikit-learn.org/stable/>

Documentation PyCaret : <https://pycaret.org/>

Documentation Flask : <https://flask.palletsprojects.com/en/1.1.x/>

Documentation Docker : <https://docs.docker.com/>

Documentation Azure: <https://docs.microsoft.com/fr-fr/azure/?product=featured>

# Annexes

## Annexe 1 : Description du dataset

### Description du dataset

- `id_mutation` : Identifiant de mutation (non stable, sert à grouper les lignes)
- `date_mutation` : Date de la mutation au format ISO-8601 (YYYY-MM-DD)
- `numero_disposition` : Numéro de disposition
- `valeur_fonciere` : Valeur foncière (séparateur décimal = point)
- `adresse_numero` : Numéro de l'adresse
- `adresse_suffixe` : Suffixe du numéro de l'adresse (B, T, Q)
- `adresse_code_voie` : Code FANTOIR de la voie (4 caractères)
- `adresse_nom_voie` : Nom de la voie de l'adresse
- `code_postal` : Code postal (5 caractères)
- `code_commune` : Code commune INSEE (5 caractères)
- `nom_commune` : Nom de la commune (accentué)
- `ancien_code_commune` : Ancien code commune INSEE (si différent lors de la mutation)
- `ancien_nom_commune` : Ancien nom de la commune (si différent lors de la mutation)
- `code_departement` : Code département INSEE (2 ou 3 caractères)
- `id_parcelle` : Identifiant de parcelle (14 caractères)
- `ancien_id_parcelle` : Ancien identifiant de parcelle (si différent lors de la mutation)
- `numero_volume` : Numéro de volume
- `lot_1_numero` : Numéro du lot 1
- `lot_1_surface_carrez` : Surface Carrez du lot 1
- `lot_2_numero` : Numéro du lot 2
- `lot_2_surface_carrez` : Surface Carrez du lot 2
- `lot_3_numero` : Numéro du lot 3
- `lot_3_surface_carrez` : Surface Carrez du lot 3
- `lot_4_numero` : Numéro du lot 4
- `lot_4_surface_carrez` : Surface Carrez du lot 4
- `lot_5_numero` : Numéro du lot 5
- `lot_5_surface_carrez` : Surface Carrez du lot 5
- `nombre_lots` : Nombre de lots
- `code_type_local` : Code de type de local
- `type_local` : Libellé du type de local
- `surface_reelle_bati` : Surface réelle du bâti
- `nombre_pieces_principales` : Nombre de pièces principales
- `code_nature_culture` : Code de nature de culture
- `nature_culture` : Libellé de nature de culture
- `code_nature_culture_speciale` : Code de nature de culture spéciale
- `nature_culture_speciale` : Libellé de nature de culture spéciale
- `surface_terrain` : Surface du terrain
- `longitude` : Longitude du centre de la parcelle concernée (WGS-84)
- `latitude` : Latitude du centre de la parcelle concernée (WGS-84)

## Annexe 2 : Importation des CSV

### Préparation données pour SQL Server

#### Import des librairies

```
In [1]: import pandas as pd
import numpy as np
```

#### Import des données

```
In [2]: data2017=pd.read_csv("C:/Users/amand/Desktop/ProjetEcole/Data/DVF2017.csv", low_memory=False)
data2017.head()
```

```
Out[2]:
```

	id_mutation	date_mutation	numero_disposition	nature_mutation	valeur_fonciere	adresse_numero	adresse_suffixe	adresse_nom_voie	adresse_code_voie
0	2017-1	2017-01-02	1	Vente	27000.0	83.0	NaN	RUE CHARLES ROBIN	0820
1	2017-2	2017-01-05	1	Vente	115000.0	NaN	NaN	LES VAVRES	B032
2	2017-3	2017-01-08	1	Vente	1.0	NaN	NaN	LA POIPE	B080
3	2017-3	2017-01-08	1	Vente	1.0	NaN	NaN	LA POIPE	B080
4	2017-3	2017-01-08	1	Vente	1.0	NaN	NaN	LA POIPE	B080

5 rows × 10 columns

```
In [3]: data2018=pd.read_csv("C:/Users/amand/Desktop/ProjetEcole/Data/DVF2018.csv", low_memory=False)
data2018.head()
```

```
Out[3]:
```

	id_mutation	date_mutation	numero_disposition	nature_mutation	valeur_fonciere	adresse_numero	adresse_suffixe	adresse_nom_voie	adresse_code_voie
0	2018-1	2018-01-03	1	Vente	109000.0	13.0	NaN	RUE GEN LOGEROT	1880
1	2018-1	2018-01-03	1	Vente	109000.0	13.0	NaN	RUE GEN LOGEROT	1880
2	2018-2	2018-01-04	1	Vente	239300.0	4.0	NaN	RUE DE LA BARMETTE	0025
3	2018-2	2018-01-04	1	Vente	239300.0	4.0	NaN	RUE DE LA BARMETTE	0025
4	2018-2	2018-01-04	1	Vente	239300.0	4.0	NaN	RUE DE LA BARMETTE	0025

5 rows × 10 columns

```
In [4]: data2019=pd.read_csv("C:/Users/amand/Desktop/ProjetEcole/Data/DVF2019.csv", low_memory=False)
data2019.head()
```

```
Out[4]:
```

	id_mutation	date_mutation	numero_disposition	nature_mutation	valeur_fonciere	adresse_numero	adresse_suffixe	adresse_nom_voie	adresse_code_voie
0	2019-1	2019-01-11	1	Vente	84000.0	552.0	NaN	AV DE LYON	0280
1	2019-1	2019-01-11	1	Vente	84000.0	552.0	NaN	AV DE LYON	0280
2	2019-2	2019-02-08	1	Vente	210000.0	5189.0	NaN	LE METRILLOT	B041
3	2019-2	2019-02-08	1	Vente	210000.0	5189.0	NaN	LE METRILLOT	B041
4	2019-3	2019-04-04	1	Vente	38000.0	40.0	NaN	PL DE LA FONTAINE	0090

5 rows × 10 columns



### Annexe 3 : Création des DataFrames Mutation, Cadastre et Bien

Entrée [14]:

```
dvfMutation=dvf.drop(['adresse_nom_voie',
                      'nom_commune',
                      'code_departement',
                      'nombre_lots',
                      'code_type_local',
                      'type_local',
                      'surface_reelle_bati',
                      'nombre_pieces_principales',
                      'surface_terrain',
                      'longitude',
                      'latitude'], axis=1)

dvfMutation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2224142 entries, 1 to 1017132
Data columns (total 6 columns):
#   Column              Dtype
---  -----
0   id_mutation         object
1   date_mutation       object
2   nature_mutation     object
3   valeur_fonciere     float64
4   id_parcelle         object
5   id_bien              object
dtypes: float64(1), object(5)
memory usage: 118.8+ MB
```

Entrée [15]:

```
dvfCadastre=dvf.drop(['id_mutation',
                      'date_mutation',
                      'nature_mutation',
                      'valeur_fonciere',
                      'code_type_local',
                      'type_local',
                      'surface_reelle_bati',
                      'nombre_pieces_principales',
                      'surface_terrain',
                      'id_bien'], axis=1)

dvfCadastre.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2224142 entries, 1 to 1017132
Data columns (total 7 columns):
#   Column              Dtype
---  -----
0   adresse_nom_voie   object
1   nom_commune        object
2   code_departement   object
3   id_parcelle        object
4   nombre_lots        int64
5   longitude          float64
6   latitude           float64
dtypes: float64(2), int64(1), object(4)
memory usage: 135.8+ MB
```

Entrée [16]:

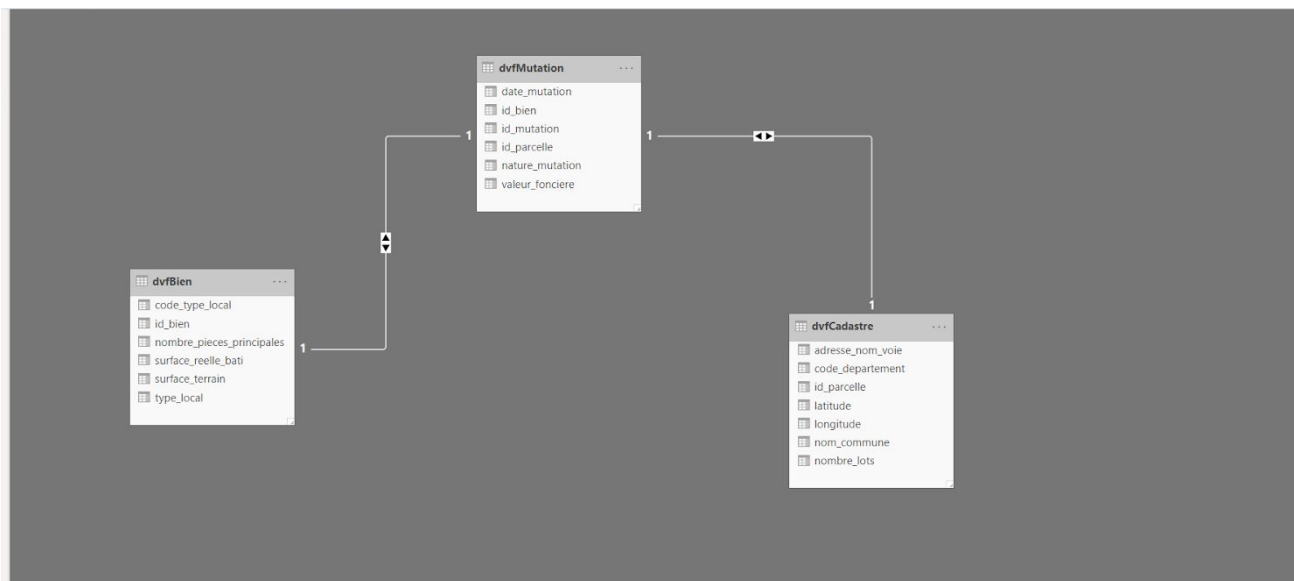
```
dvfBien=dvf.drop(['id_parcelle',
                  'adresse_nom_voie',
                  'nom_commune',
                  'code_departement',
                  'nombre_lots',
                  'longitude',
                  'latitude',
                  'id_mutation',
                  'date_mutation',
                  'nature_mutation',
                  'valeur_fonciere'], axis=1)

dvfBien.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2224142 entries, 1 to 1017132
Data columns (total 6 columns):
#   Column              Dtype
---  -----
0   code_type_local     float64
1   type_local          object
2   surface_reelle_bati float64
3   nombre_pieces_principales float64
4   surface_terrain     float64
5   id_bien             object
dtypes: float64(4), object(2)
memory usage: 118.8+ MB
```



## Annexe 4 : Schéma EA



## Annexe 5 : Script SQL

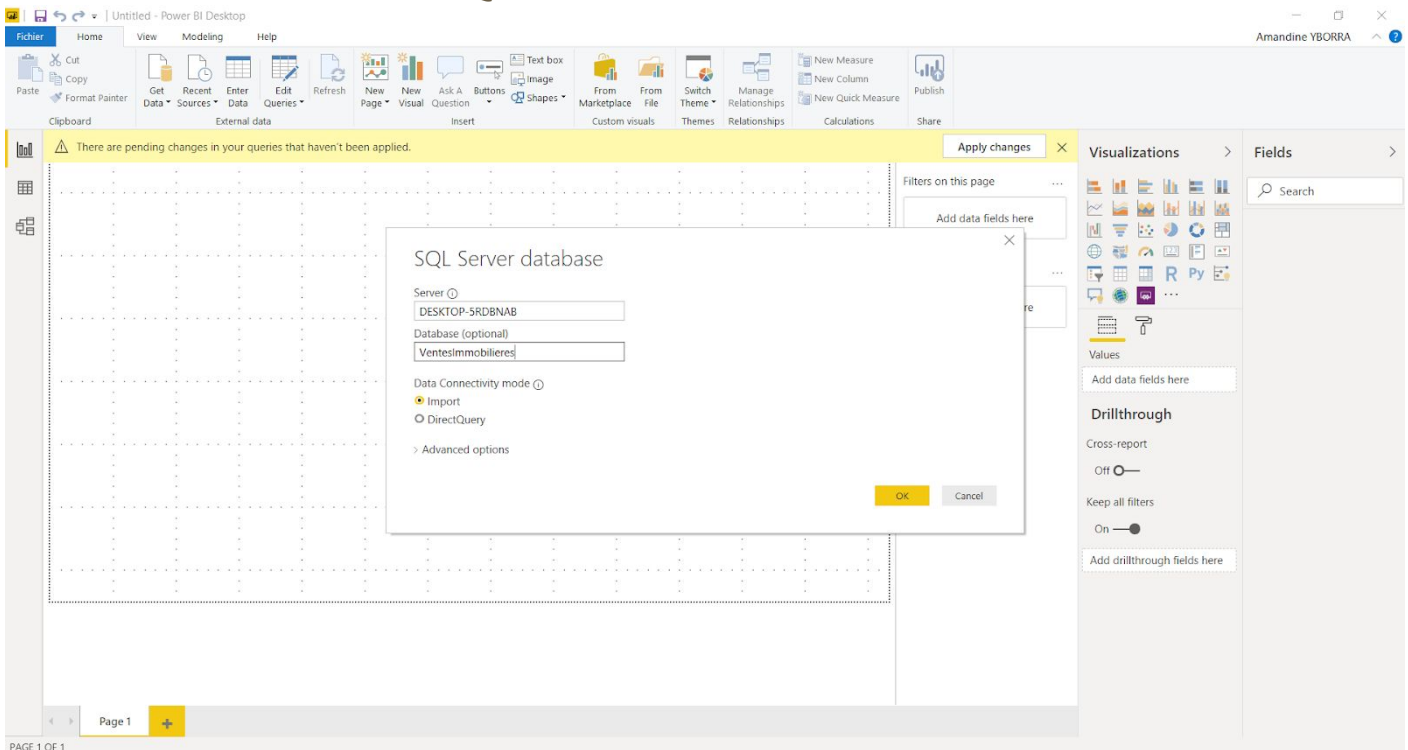
```

/***** Object: Table [dbo].[dvfBien]    Script Date: 16/04/2020 14:19:24 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[dvfBien](
    [code_type_local] [nvarchar](50) NULL,
    [type_local] [nvarchar](50) NULL,
    [surface_reelle_bati] [nvarchar](50) NULL,
    [nombre_pieces_principales] [nvarchar](50) NULL,
    [surface_terrain] [nvarchar](50) NULL,
    [id_bien] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_dvfBien] PRIMARY KEY CLUSTERED
)
(
    [id_bien] ASC
)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
/***** Object: Table [dbo].[dvfCadaastre]    Script Date: 16/04/2020 14:19:24 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[dvfCadaastre](
    [adresse_nom_voie] [nvarchar](50) NULL,
    [nom_commune] [nvarchar](50) NULL,
    [code_departement] [nvarchar](50) NULL,
    [id_parcelle] [nvarchar](50) NOT NULL,
    [nombre_lots] [nvarchar](50) NULL,
    [longitude] [nvarchar](50) NULL,
    [latitude] [nvarchar](50) NULL,
    CONSTRAINT [PK_dvfCadaastre] PRIMARY KEY CLUSTERED
)
(
    [id_parcelle] ASC
)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
/***** Object: Table [dbo].[dvfMutation]    Script Date: 16/04/2020 14:19:24 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[dvfMutation](
    [id_mutation] [nvarchar](50) NOT NULL,
    [date_mutation] [datetime2](7) NOT NULL,
    [nature_mutation] [nvarchar](50) NOT NULL,
    [valeur_foncieres] [nvarchar](50) NOT NULL,
    [id_parcelle] [nvarchar](50) NOT NULL,
    [id_bien] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_dvfMutation] PRIMARY KEY CLUSTERED
)
(
    [id_mutation] ASC
)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
ALTER TABLE [dbo].[dvfMutation] WITH CHECK ADD CONSTRAINT [FK_dvfMutation_dvfBien] FOREIGN KEY([id_bien])
REFERENCES [dbo].[dvfBien] ([id_bien])
GO
ALTER TABLE [dbo].[dvfMutation] CHECK CONSTRAINT [FK_dvfMutation_dvfBien]
GO
ALTER TABLE [dbo].[dvfMutation] WITH CHECK ADD CONSTRAINT [FK_dvfMutation_dvfCadaastre] FOREIGN KEY([id_parcelle])
REFERENCES [dbo].[dvfCadaastre] ([id_parcelle])
GO
ALTER TABLE [dbo].[dvfMutation] CHECK CONSTRAINT [FK_dvfMutation_dvfCadaastre]
GO
USE [master]
GO
ALTER DATABASE [VentesImmobilieres] SET READ_WRITE
GO
  
```

## Annexe 6 : Backup automatique



## Annexe 7 : Connection Power Bi / SQL Server

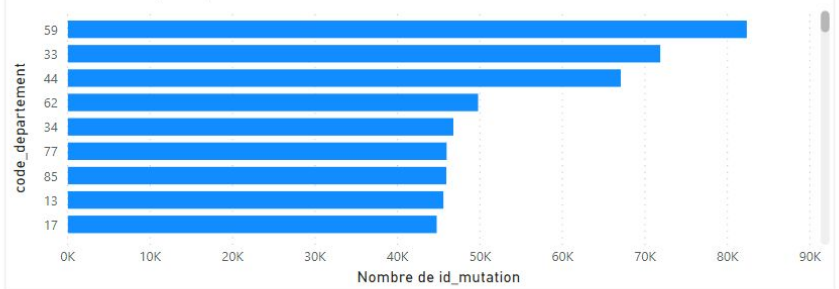


## Annexe 8 : Exemples Visualisation Globale

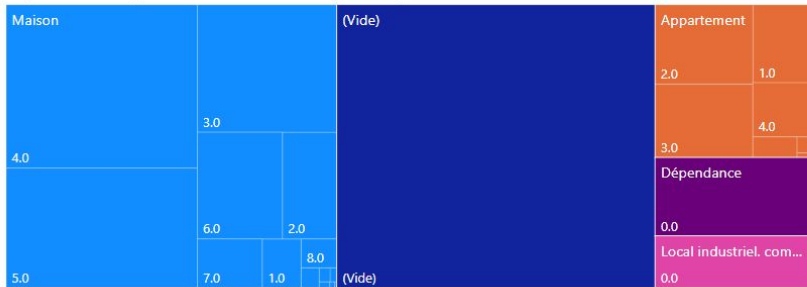
Nombre de Ventes par Années



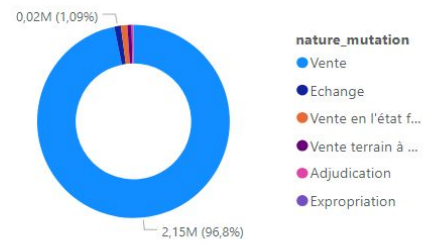
Nombre de biens par Départements et Communes



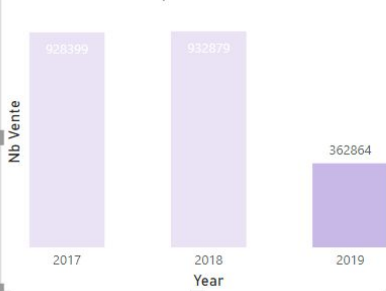
Répartition des types de locaux et du nombre de pièces



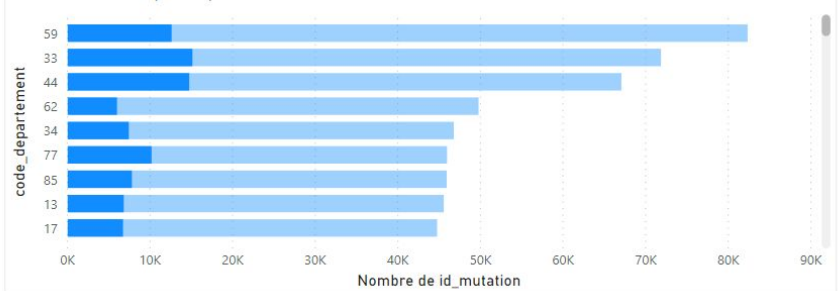
Répartition des types de Mutation



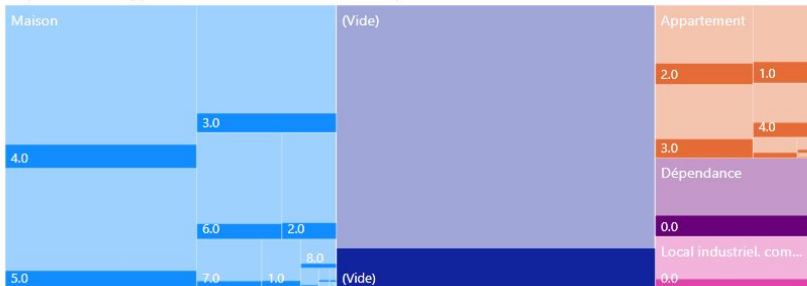
Nombre de Ventes par Années



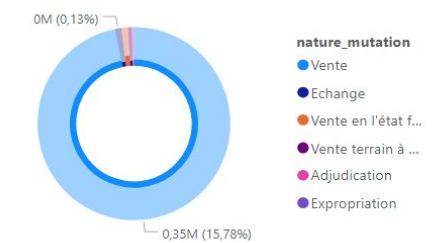
Nombre de biens par Départements et Communes



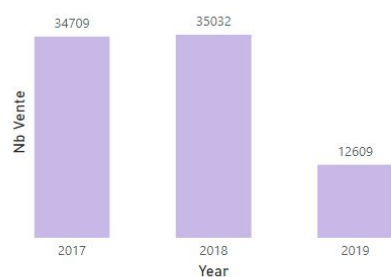
Répartition des types de locaux et du nombre de pièces



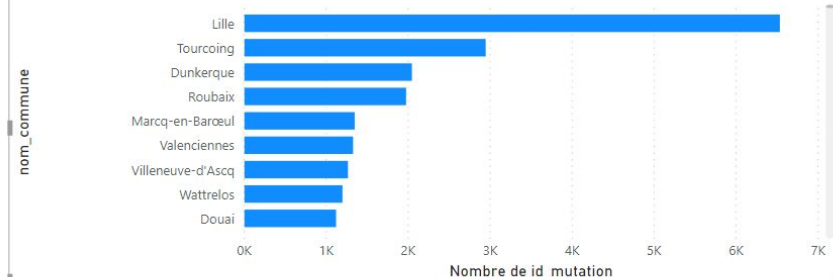
Répartition des types de Mutation



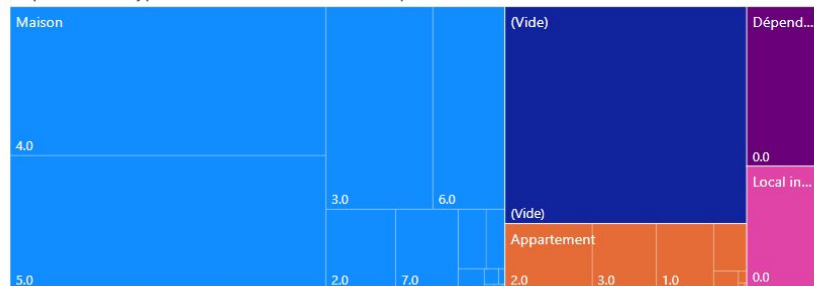
Nombre de Ventes par Années



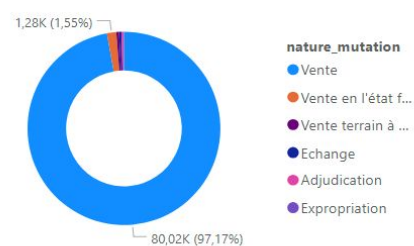
Nombre de biens par Départements et Communes



Répartition des types de locaux et du nombre de pièces



Répartition des types de Mutation

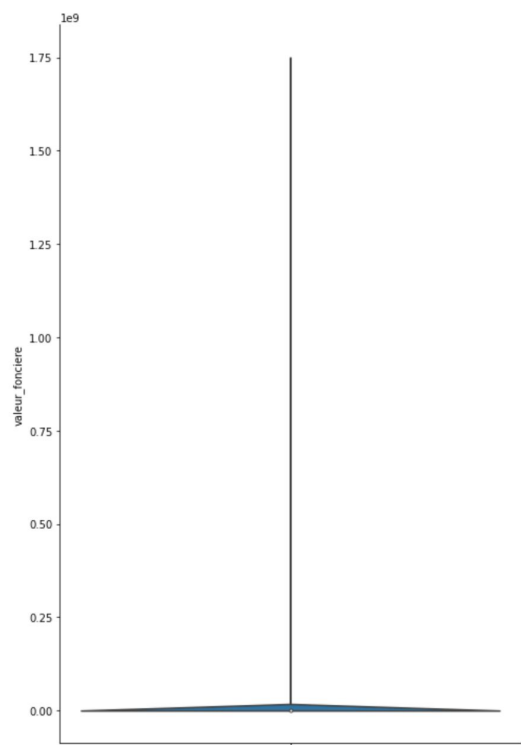


## Annexe 9 : Map générale

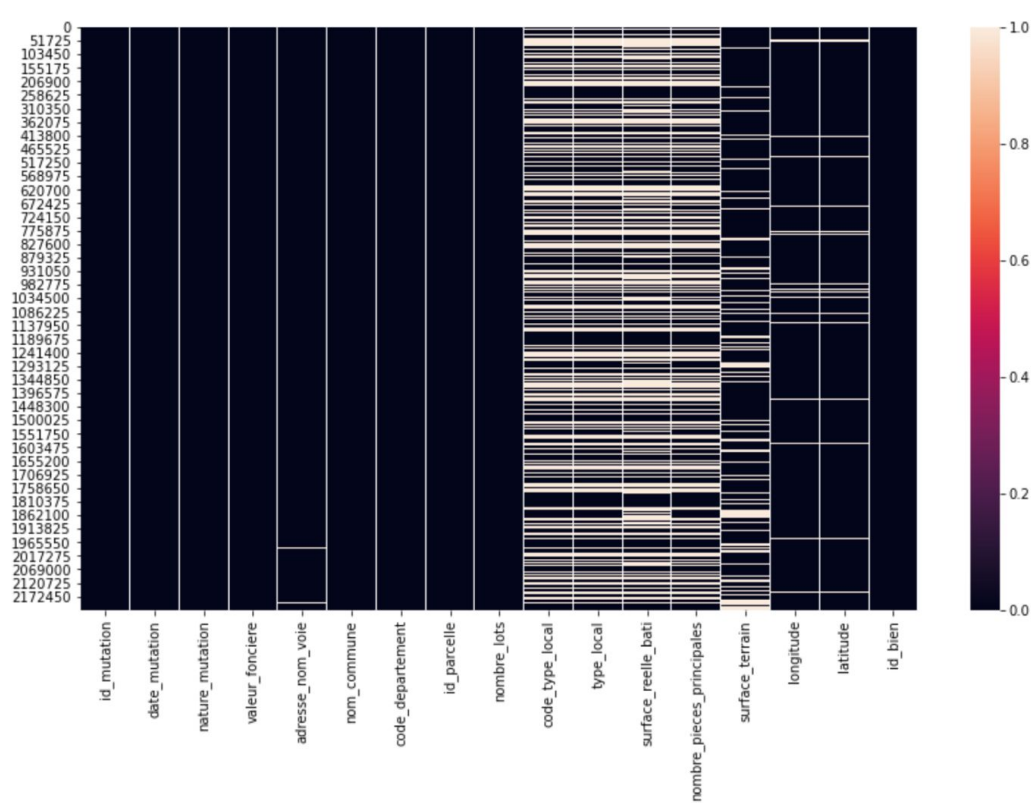
Moyenne de prix par régions



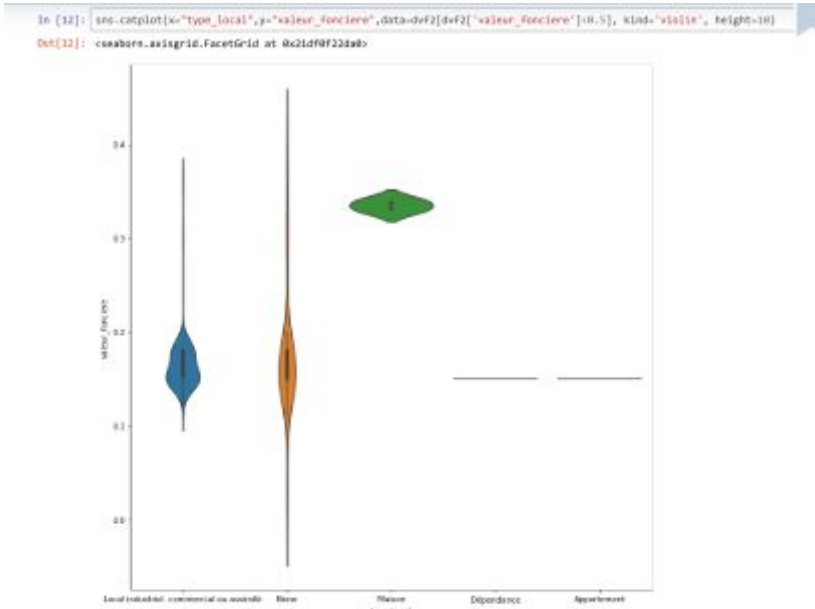
Annexe 10 : Box plot valeurs foncières



Annexe 11 : Heatmap des NaN



## Annexe 12 : Traitement type\_local



## Remarques

On peut également se poser la question des valeurs extrêmes de `valeur_foncieres`.

```

In [35]: dw42['type_local'] = dw42['type_local'].replace('Mise', 'local industriel, commercial ou assésité')

In [34]: dw42['type_local'].unique()

Out[34]: array(['local industriel, commercial ou assésité', 'Appartement',
               'Maison', 'Dépendance'], dtype=object)

```

## Annexe 13 : Résultats Algorithmes

## Phase 1 :

MinMaxScaler				StandardScaler		
Algorithmes	RMSE	r2 score		Algorithmes	RMSE	r2 score
LinearRegression	46 480.23	0.11		LinearRegression	46 480.11	0.11
DecisionTreeRegressor	46 146.10	0.12		DecisionTreeRegressor	46 146.10	0.12
Ridge	46 479.93	0.11		Ridge	46 480.07	0.11
RandomForestRegressor	46 144.55	0.12		RandomForestRegressor	46 144.47	0.12
XGBoost	46 145.35	0.12		XGBoost	46 145.35	0.12
MLPRegressor	46 766.67	0.12		MLPRegressor	46 372.62	0.11
			Meilleur résultat			
			Pire résultat			



## Phase 2 :

Algorithmes	RMSE	r2 score	
<b>LinearRegression</b>	6 648 012.44	-4.33	
<b>DecisionTreeRegressor</b>	3585490.32	-0.55	
<b>Ridge</b>	6300884.44	-3.79	
<b>XGBoost</b>	2845288.14	0.02	
			Meilleur résultat
			Pire résultat

## Phase 3 :

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Gradient Boosting Regressor	38241.4836	2.105499e+09	45882.7351	0.1346	0.4045	0.3741
1	<b>Extreme Gradient Boosting</b>	<b>38246.6595</b>	<b>2.105490e+09</b>	<b>45882.6792</b>	<b>0.1346</b>	<b>0.4045</b>	<b>0.3742</b>
2	Random Forest	38213.1700	2.107628e+09	45906.0334	0.1337	0.4046	0.3737
3	Extra Trees Regressor	38219.5760	2.108554e+09	45916.0991	0.1333	0.4047	0.3737
4	Decision Tree	38223.9136	2.109690e+09	45928.4130	0.1329	0.4048	0.3738
5	Ridge Regression	38523.0725	2.128837e+09	46136.2947	0.1250	0.4064	0.3769
6	Least Angle Regression	38522.6509	2.128846e+09	46136.3981	0.1250	0.4064	0.3769
7	Bayesian Ridge	38524.3415	2.128813e+09	46136.0402	0.1250	0.4064	0.3770
8	Linear Regression	38519.6092	2.129235e+09	46140.5951	0.1248	0.4064	0.3768
9	Random Sample Consensus	38319.7262	2.145184e+09	46312.5759	0.1183	0.4075	0.3756
10	Huber Regressor	38329.2948	2.145462e+09	46315.7911	0.1182	0.4063	0.3728
11	Support Vector Machine	37937.0006	2.146186e+09	46323.6649	0.1179	0.4064	0.3679
12	AdaBoost Regressor	39227.1570	2.160830e+09	46481.9652	0.1119	0.4109	0.3858
13	TheilSen Regressor	39161.3798	2.200469e+09	46906.4149	0.0955	0.4107	0.3796
14	Orthogonal Matching Pursuit	40024.4213	2.258934e+09	47526.1121	0.0715	0.4187	0.3932
15	Lasso Regression	42329.5910	2.454641e+09	49542.8151	-0.0090	0.4376	0.4210
16	Elastic Net	42329.5910	2.454641e+09	49542.8151	-0.0090	0.4376	0.4210
17	Lasso Least Angle Regression	42329.5910	2.454641e+09	49542.8151	-0.0090	0.4376	0.4210
18	K Neighbors Regressor	41538.8783	2.578297e+09	50766.8067	-0.0600	0.4475	0.4070
19	Passive Aggressive Regressor	50607.9098	3.968342e+09	62686.4931	-0.6312	0.5210	0.4871

## Annexe 14 : Script API

```

File Edit Selection View Go Run Terminal Help
app.py - ProjectCode - Visual Studio Code

EXPLORER
  OPEN EDITORS
  PROJECTCODE
    > Data
    > Phase1
    > Phase2
    > Phase3
    > PowerBI
    > ProjectEnv
    > ScreenShotProjectCode
    > Support
    > webapp
      > .pycache_
      > model
      > static
      > templates
      > index.html
      app.py
      Dockerfile
      featuresAPI.csv
      machine_learning_dv_sans_outliers...
      requirements.txt
      requirements.txt

OUTLINE
TIMELINE

c:\Users\amand\Desktop> webapp > app.py
1 import numpy as numpy
2 from pyspark.regression import *
3 import pandas as pd
4 from flask import Flask, request, jsonify, render_template
5 import pickle
6
7 app = Flask(__name__)
8 model = load_model("model/FinalXGBoost")
9
10 @app.route("/")
11 def home():
12     return render_template("index.html")
13
14 @app.route("/predict", methods=['POST'])
15 def predict():
16     feats = pd.read_csv("featuresAPI.csv")
17
18     features = [f for f in request.form.values()]
19
20     feats['regions'] = features[2]
21     feats['type_local'] = features[1]
22     feats['nature_mutation'] = features[0]
23     feats['nombre_pieces_principales'] = features[3]
24
25     prediction = predict_model(model, data=feats)
26     print(prediction)
27     output = round(prediction['label'][0], 2)
28
29     return render_template("index.html", prediction_text = 'Le bien est estimé à {} €'.format(output))
30
31 if __name__ == "__main__":
32     app.run(debug=True, host='0.0.0.0', port=80)

```

## Annexe 15 : DockerFile

```
app.py Dockerfile # style.css index.html
Dockerfile > ...
1 FROM ubuntu:18.04
2
3 RUN apt-get update -y && \
4 apt-get install -y python3-pip python3-dev
5
6 # We copy just the requirements.txt first to leverage Docker cache
7 COPY ./requirements.txt /app/requirements.txt
8
9 WORKDIR /app
10
11 RUN pip3 install -r requirements.txt
12
13 COPY . /app
14
15 EXPOSE 8080
16
17 ENTRYPOINT [ "python3" ]
18
19 CMD [ "app.py" ]
```

## Annexe 16 : Azure

### Paramétrage Azure

The screenshot displays the Microsoft Azure portal interface for a container registry named 'projetcertif'. The left sidebar shows navigation options like 'Vue d'ensemble', 'Journal d'activité', and 'Paramètres'. The main content area shows the registry's configuration details, including its resource group ('API'), location ('France-Centre'), and subscription ('Free Trial'). Below this, there are sections for 'Utilisation' (Usage) showing storage metrics, 'ACR Tasks' for container management, and 'Intégrations de sécurité pour les conteneurs' (Container security integrations) featuring 'Aqua Security' and 'Twistlock'.

Paramètre	Valeur
Groupe de ressources...	: API
Emplacement	: France-Centre
Abonnement (modifier)	: Free Trial
ID d'abonnement	: f1773a08-df55-4409-8911-c51629c2a8bf
Serveur de connexion	: projetcertif.azurecr.io
Date de création	: 27/04/2020 à 10:43 UTC+2
SKU	: Standard
État de l'approvisionnement...	: Opération réussie

**Utilisation**

Metric	Value
Indus dans la réf...	100 GiB
Utilisé	0.00 GiB
Stockage supplém...	0.00 GiB

**ACR Tasks**

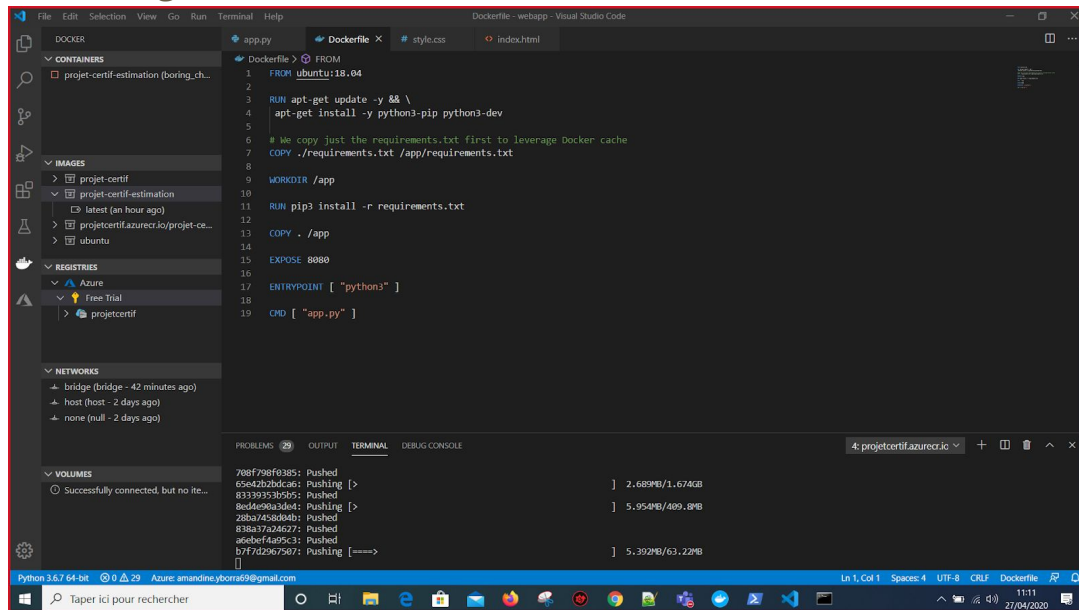
Générer, exécuter, pousser et corriger des conteneurs dans Azure avec ACR Tasks. Les tâches prennent en charge Windows, Linux et ARM avec QEMU.

**Intégrations de sécurité pour les conteneurs**

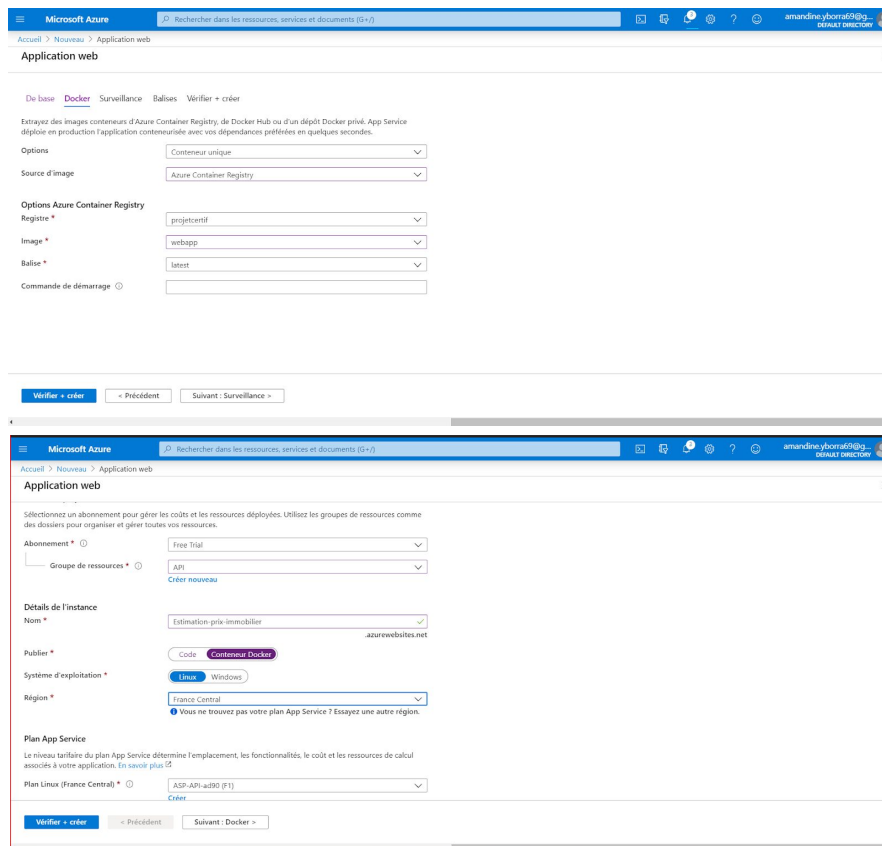
- Aqua Security**: Aqua fournit des outils de contrôle du cycle de vie pour sécuriser les applications en conteneur du développement à la production.
- Twistlock**: Solution de gestion des vulnérabilités et de protection du runtime dans tous vos environnements.



## Push de l'image Docker dans Azure



## Création d'une WebAPP



# Déploiement

Microsoft Azure

Rechercher dans les ressources, services et documents (G+/J)

amandine.yborra69@g...  
DEFAULT DIRECTORY

Accueil > Microsoft.Web-WebApp-Portal-e90acaf1-bcb6 | Vue d'ensemble

Microsoft.Web-WebApp-Portal-e90acaf1-bcb6 | Vue d'ensemble

Rechercher (Ctrl+J)

Supprimer Annuler Redéployer Actualiser

**Vue d'ensemble**

Entrées

Sorties

Modèle

**●●● Votre déploiement est en cours**

Nom du déploiement : Microsoft.Web-WebApp-Portal-e90acaf1-b...  
Abonnement : Free Trial  
Groupe de ressources : API

Heure de début : 27/04/2020 à 20:35:41  
ID de corrélation : d530235a-29d0-4139-9615-7202fd7697ec

^ Détails du déploiement (Télécharger)

Ressource	Type	Statut	Détails de l'opération
Aucun résultat.			

**Centre de sécurité**  
Sécuriser vos applications et votre infrastructure  
[Accéder à Azure Security Center >](#)

**Tutoriels Microsoft gratuits**  
[Commencer l'apprentissage aujourd'hui >](#)

**Travailler avec un expert**  
Les experts Azure sont des partenaires fournisseurs de services qui peuvent vous aider à gérer vos ressources sur Azure et constituer votre première ligne de support.  
[Trouver un expert Azure >](#)

## Annexe 17 : Exemple Trello

Bonjour Amandine Yborra ! Nous devons confirmer votre adresse e-mail. [Confirmer l'adresse e-mail](#) · [Me le rappeler plus tard](#)

Projet École Amandine

Équipe privée Visible par les membres d'une équipe Inviter Afficher le menu

**Backlog**

- (13) Conteneurisation docker de la BDD, de l'API, des Algorithmes et la Visualisation
- (13) Déployer le conteneur
- + Ajouter une autre carte

**A faire**

- (8) Créer un API
- + Ajouter une autre carte

**En cours**

- (1) MAJ du Git
- (5) Améliorer les algorithmes
- (3) Création de 5 Slides
- (2) Ajouter des commentaires dans les notebooks
- (3) Créer la pipeline pour le modèle
- + Ajouter une autre carte

**Terminé**

- (5) Récupération des données
- (3) Création base de données
- (5) Mise en place de backup automatique
- (3) Nettoyer les données récupérées
- (3) Dashboard de visualisation des données
- (3) Trouver des algorithmes de prédictions
- (3) Trouver des algorithmes de prédictions V2
- + Ajouter une autre carte

+ Ajoutez une autre liste

Tableaux

Bonjour Amandine Yborra ! Nous devons confirmer votre adresse e-mail. [Confirmer l'adresse e-mail](#) · [Me le rappeler plus tard](#)

Projet École Amandine

Équipe privée Visible par les membres d'une équipe Inviter Butler Afficher le menu

Backlog

+ Ajouter une carte

A faire

(13) Déployer le conteneur

+ Ajouter une autre carte

En cours

(1) MAJ du Git

(5) Faire les Slides de présentation

(5) Faire le rapport

(13) Conteneurisation docker de la BDD, de l'API, des Algorithmes et la Visualisation

+ Ajouter une autre carte

Terminé

(5) Récupération des données

(3) Création base de données

(5) Mise en place de backup automatique

(3) Nettoyer les données récupérées

(3) Dashboard de visualisation des données

(3) Trouver des algorithmes de prédictions

(3) Trouver des algorithmes de prédictions V2

(3) Création de 5 Slides

+ Ajouter une autre carte

+ Ajoutez une autre liste

## Annexe 18: Github

AmandineYborra / SchoolProject

Watch 1

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security 0

Insights

Settings

No description, website, or topics provided.

Edit

Manage topics

47 commits

1 branch

0 packages

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

AmandineYborra

Delete Machine\_Learning\_DVF\_sans\_outliers.ipynb

Latest commit 117a751 yesterday

Phase 1	Add files via upload	yesterday
Phase 2	Add files via upload	yesterday
Phase 3	Add files via upload	yesterday
PowerBIScreens	Add files via upload	2 months ago
Support	Add files via upload	11 days ago
webapp	Add files via upload	last month
README.md	Update README.md	2 months ago
Rapport Certification.docx	Add files via upload	last month
requirements.txt	Add files via upload	last month
script.sql	Add files via upload	last month

README.md

SchoolProject

Projet Chef d'Oeuvre pour ma certification auprès de l'École Microsoft IA powered by Simplon

Les Données

Les données de base se trouve sur le site de Data Gouv :  
<https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncières-geolocalisées/>  
Pour ce sujet j'ai pris les données des années 2017, 2018 et 2019.

Le dashboard

Le dashboard étant trop lourd je laisse à disposition des screens et un lien pour le retrouver ailleurs:  
<https://drive.google.com/open?id=1mOZ95yHLnAW2QLnY00QZX5LNUwKFmGO8>  
Le dashboard est sous format .pbix et ne peut être visionné que sur PowerBI, voici le lien de téléchargement pour le PowerBI :  
<https://powerbi.microsoft.com/fr-fr/downloads/>