RAPPORT PROJET E1

Détection d'îlots de chaleur urbain

Aissa Mabroumi - Apprenant Simplon Grenoble

Remerciements

Je tiens à exprimer toute ma gratitude à ma tutrice de stage, Sonia Pelloux, cheffe de projet en management de la donnée et Open Data à la Métropole de Grenoble. Cette alternance a été une expérience extrêmement enrichissante, qui m'a permis de développer un grand nombre de compétences en travaillant sur des projets passionnants. Un exemple de ces projets sera exposé dans ce document.

Mon appréciation s'étend également à l'équipe de Simplon. Leur contribution a été décisive dans la réalisation de cette formation. Je tiens à remercier tout particulièrement mes formateurs, Pierre-Loïc Bayart et Romain Clément, pour leur assistance précieuse, leur bienveillance et leur approche pédagogique durant cette période d'apprentissage. Ma reconnaissance va aussi à Christine Cavarretta, notre responsable de formation, pour son soutien indéfectible et son accompagnement bienveillant tout au long de ces 18 mois.

Enfin, je désire exprimer ma gratitude à la Métropole de Grenoble qui m'a accordé sa confiance et m'a offert l'opportunité d'intégrer ses rangs. Un remerciement particulier à l'équipe Open Data de la Métropole, avec qui j'ai eu le plaisir de collaborer fréquemment. Leur contribution a non seulement rendu mes journées de travail plus agréables, mais a aussi enrichi mon expérience et aiguisé mon intérêt pour le domaine.

Sommaire

1 Introduction 4
- 1.1 Contexte et objectifs du projet 4
- 1.2 État de l'art 5

2 Développement du modèle de prédiction basé sur les données numériques. 6
- 2.1 Description des données 6
- 2.2 Analyse des données 6
- 2.3 Entraînement et évaluation du modèle 9

3 Développement du modèle de prédiction basé sur les données images 14
- 3.1 Description des données 15
- 3.2 Entraînement et évaluation du modèle 17

4 Développement de l'application Web 22
- 4.1 Application de prédiction 23
- 4.2 les éléments de conception technique 25
- 4.3 BDD 27
- 4.4 Tests Unitaires 28
- 4.5 Suivi des modèles et de l'application 28
- 4.6 Organisation 29

5 Conclusion 30

Références 32

1 Introduction

L'objectif de ce document est de fournir une description détaillée des travaux entrepris pour la création du projet E1.

Le projet que vous êtes sur le point de découvrir a été conçu et développé lors de mon alternance à la Métropole de Grenoble et chez Simplon. Il englobe la majorité des techniques et connaissances acquises durant cette période de formation et d'apprentissage.

Ce projet a été réalisé en suivant une approche d'Open Data, un mouvement qui promeut l'ouverture et la mise à disposition des données produites et collectées par les services publics.

Ce projet m'a permis de découvrir de nombreux aspects du travail avec les données, tels que la collecte des sources, le nettoyage et l'analyse des données, l'architecture de la base de données, ainsi que l'utilisation de différents modèles de machine learning et de deep learning. J'ai également pu approfondir mes compétences en visualisation de données pour rendre l'information facilement compréhensible.

1.1 Contexte et objectifs du projet

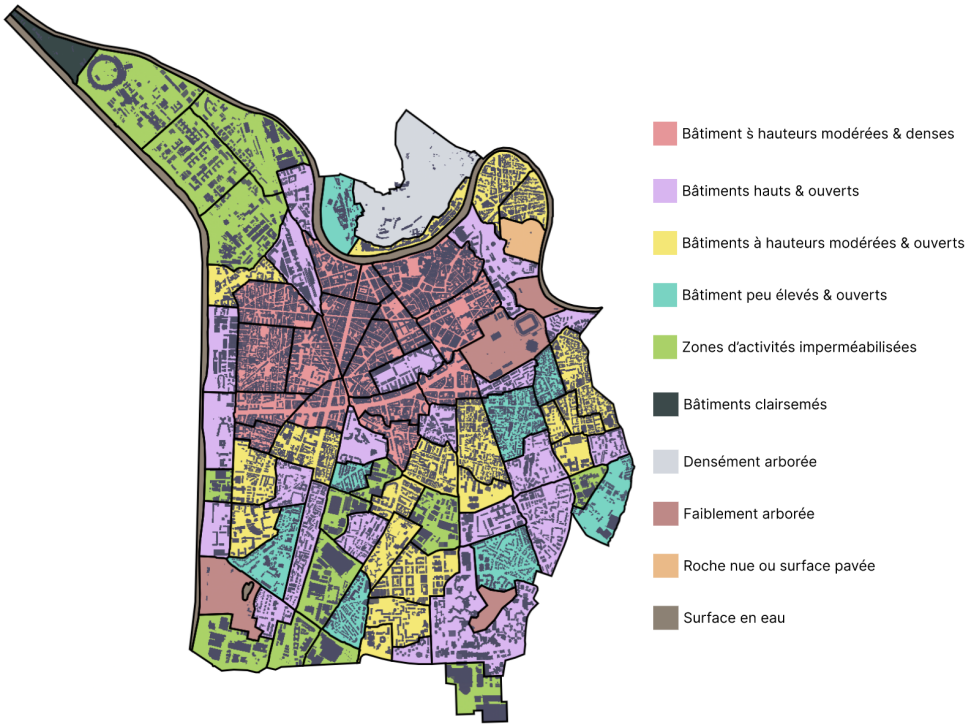
Les vagues de chaleur plus nombreuses et plus intenses exposent les villes à des températures extrêmes, en particulier la nuit où la température reste plus élevée qu'à la campagne. Cette différence de température nocturne ville/campagne correspond à l'îlot de chaleur urbain (ICU).

L'objectif principal de cette application développée dans le cadre de ce projet est de sensibiliser et d'informer les citoyens de la Métropole de Grenoble au phénomène d' îlots de chaleur urbains et leur impact sur l'environnement et la santé. D'autre part, l'interface développée vise à à promouvoir la démarche d'Open Data en démontrant l'utilité potentielle de la mise à disposition des données. Pour ce faire, nous avons utilisé des graphiques interactifs, des cartes et d'autres éléments visuels attrayants afin de rendre l'application à la fois divertissante et facile à comprendre pour un utilisateur .

Cette application vise également à tester la faisabilité de la prédiction de l'intensité des îlots de chaleur urbains en fonction de caractéristiques décrivant le territoire (surface végétalisée, surface imperméabilisé, …) afin de généraliser l'identification des îlots de chaleur sur d'autres territoires . Cela pourrait fournir une piste à explorer ultérieurement avec un volume de données plus important. L'application permettrait ainsi d'identifier les zones subissant un phénomène d' îlot de chaleur urbain élevé et de sensibiliser les autorités.
En utilisant la stratégie du "Data Storytelling", nous avons l'intention de décrire précisément et de manière compréhensible les conséquences des îlots de chaleur urbains sur la population. Nous visons à identifier leurs causes et à proposer des solutions adaptées pour améliorer les situations où ces phénomènes sont préjudiciables pour les citoyens.

Carte affichant les Zones de Classification Locale (LCZ) qui constituent un système de classification international utilisé pour analyser le climat urbain. Ces zones, d'environ 10 hectares, sont classées en fonction de leur morphologie urbaine, réalisée grâce à Mapbox :

Carte LCZ



Pour répondre à ces problématiques, plusieurs solutions ont été apportées :

- Un premier algorithme pour la prédiction des ICU, basé sur des données numériques.
- Un deuxième algorithme pour la prédiction des ICU, cette fois-ci basé sur des images.
- Une application web qui intègre ces deux algorithmes.
- Un autre site axé sur le "data storytelling", qui communique les données de manière claire

1.2 État de l'art

Pour la réalisation de ce projet, il a été nécessaire d'entreprendre une phase de recherche et de réflexion sur les technologies à employer. Cependant, lors de cette exploration, j'ai constaté un manque de données sur le sujet. Malgré cela, j'ai découvert une étude intitulée "Using deep-learning to forecast the magnitude and characteristics of urban heat island in Seoul Korea". Dans cette recherche, les auteurs ont utilisé un réseau neuronal profond (DNN - Deep Neural Network) pour prédire les îlots de chaleur urbains à Séoul, en Corée, en s'appuyant sur des données satellites Landsat et des données de capteurs de température sur une période de 20 ans.

Cependant, compte tenu de la complexité de leur processus, du volume de données utilisées et des exigences en termes de puissance de calcul, j'ai cherché à simplifier leur méthode. J'ai ainsi tenté d'appliquer une approche de classification à partir des données déjà fournies par l'UGA (Université de Grenoble-Alpes), tout en utilisant des images satellites des Zones de Classification Locale du climat (LCZ - Local Climate Zones).

2 Développement du modèle de prédiction basé sur les données numériques.

2.1 Description des données

Les données utilisées pour ce projet proviennent de l'Université Grenoble Alpes (UGA), de Grenoble Alpes Métropole et d'une équipe interne de scientifiques de la ville de Grenoble. Parmi les fichiers qui m'ont été fournis, nous disposons des éléments suivants :

- Les coordonnées géographiques des capteurs de température situés à Grenoble, disponibles dans le format GeoJSON.

- La position et la configuration des bâtiments, également disponibles dans le format GeoJSON.

- Les contours des Zones de Classification Locale (LCZ) au format GeoJSON.

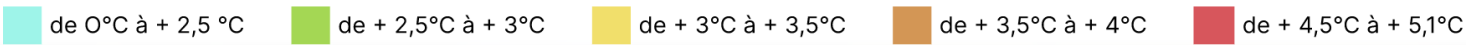
- J'ai utilisé les valeurs des îlots de chaleur urbains de 79 LCZ (Local Climate Zones), également au format GeoJSON, pour entraîner l'algorithme "numérique".
Cet algorithme prend en compte plusieurs « features » d'une LCZ telles que :

- Le taux de surface bâtie
- Le taux de surface imperméabilisée
- Le taux de surface végétalisée végétation totale
- Le taux de végétation haute La haute végétation (telle que les arbres)
- Le taux de végétation basse La basse végétation (comme les parcs, jardins, etc.)
- Le taux de surface en eau (rivières, lacs, etc.)
- le sky view factor qui est « L'effet canyon » (bâtiments hauts et rapprochés qui piègent le rayonnement), décrit par le degré d'ouverture du ciel depuis le sol
- La hauteur moyenne des bâtiments d'une LCZ

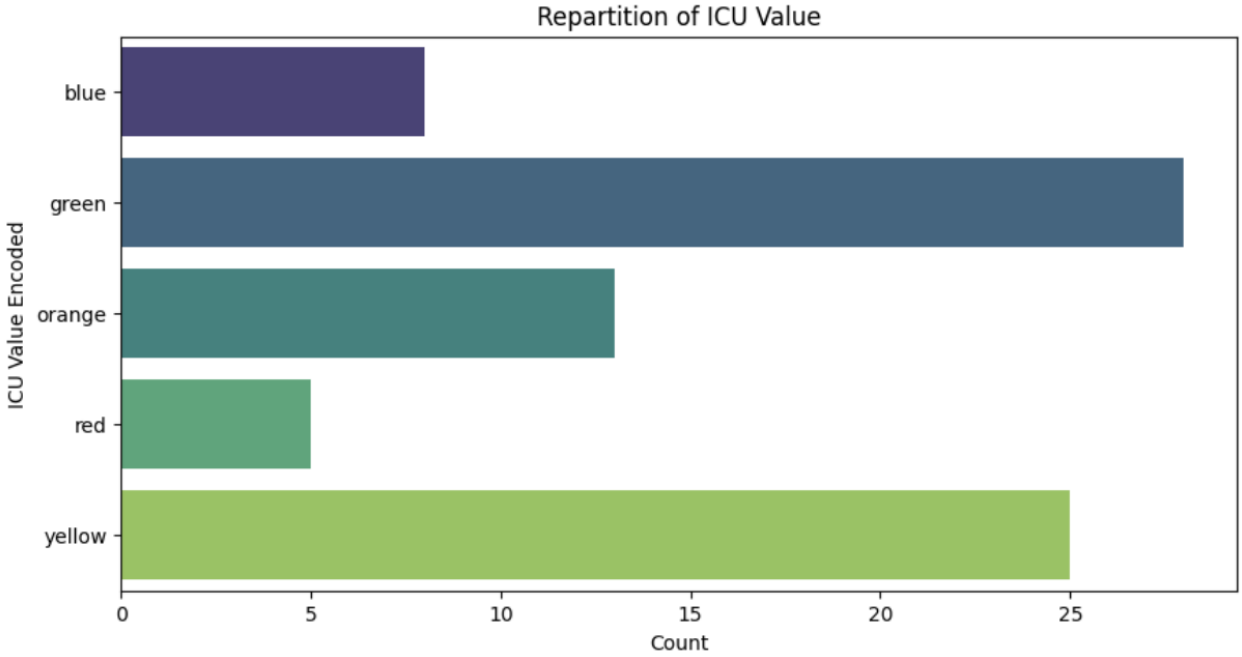
2.2 Analyse des données

Cette image nous montre les différentes classes représentant les écarts de température entre Grenoble et une ville de référence très peu urbanisée (Versoud), ainsi que les classes

correspondantes choisies par les scientifiques de la ville. Nous essayons de prédire ces classes, ce qui fait de ce problème un problème de classification en intelligence artificielle.

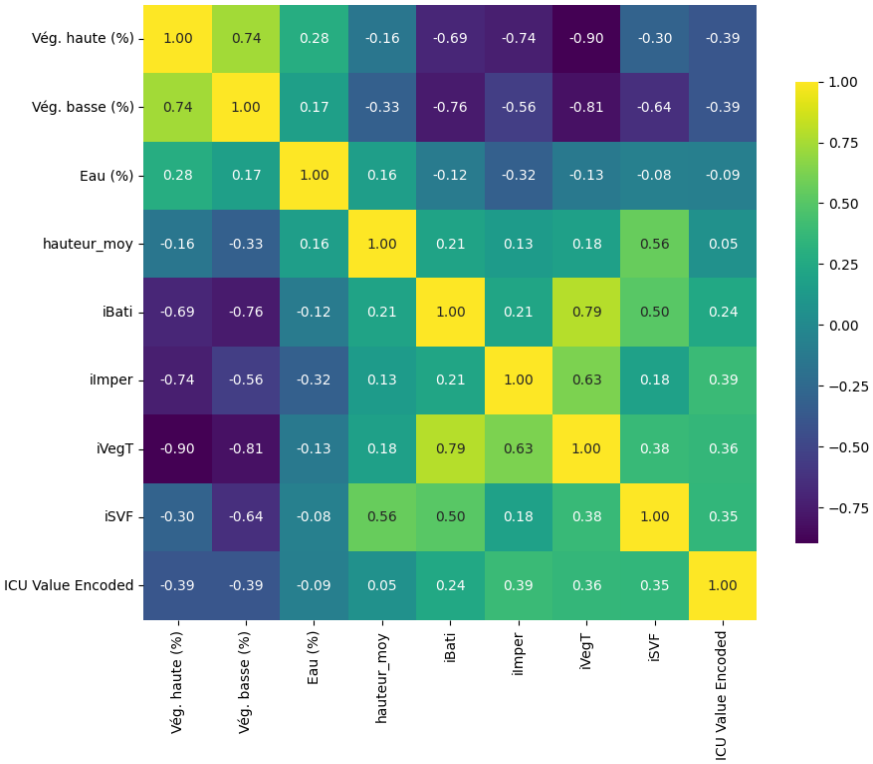


J'ai mené une phase d'analyse de données, au cours de laquelle j'ai exploré un tableau de corrélation pour identifier les catégories ayant le plus d'influence sur la valeur des îlots de chaleur urbains (ICU). J'ai également étudié la distribution des différentes classes et vérifié l'absence de valeurs manquantes dans l'ensemble de données.



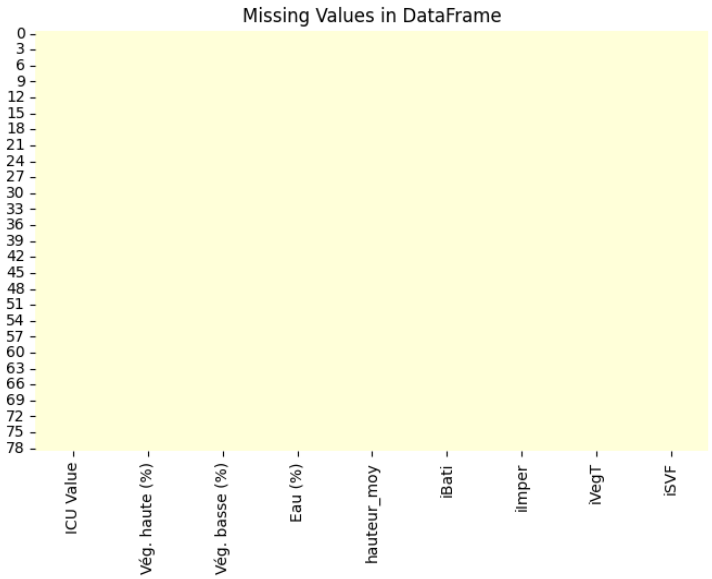
Dans ce graphique, nous observons un déséquilibre significatif entre les classes à prédire, une problématique que nous allons traiter plus en détail ultérieurement dans ce rapport.

Matrice de corrélation:



Dans cette matrice de corrélation, on peut observer qu'il y a des classes qui influent sur la valeur de l'îlot de chaleur urbain, telles que le taux de végétation, le taux de surface imperméable, etc. Cela pourrait signifier qu'il est possible d'utiliser un modèle de prédiction. Cependant, il est important de noter que la corrélation ne signifie pas forcément une causalité, mais cela ouvre néanmoins cette possibilité. D'autre part, on observe également que certaines classes semblent avoir moins d'impact sur la valeur de l'îlot de chaleur urbain, comme la hauteur moyenne des bâtiments et le taux d'eau dans une zone de confort climatique (LCZ).

On peut observer que dans les données présentées, on ne constate pas de manque de données :



2.3 Entraînement et évaluation du modèle

Après avoir converti mon jeu de données initial reçu de l'UGA en dataframe, j'ai procédé à une première analyse en utilisant un "Lazy Predict" qui est une bibliothèque Python qui permet d'automatiser le processus de construction et d'évaluation de plusieurs modèles de machine learning. C'est un outil très utile pour l'exploration initiale des données, car il permet de rapidement estimer quel type de modèle pourrait être le plus efficace pour un problème donné..

Cette approche m'a permis d'identifier les algorithmes les plus prometteurs et d'obtenir une évaluation préliminaire qui correspond approximativement à mes attentes.

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score
ExtraTreesClassifier	0.75	0.75	None	0.75
RandomForestClassifier	0.75	0.75	None	0.75
LinearSVC	0.69	0.71	None	0.67
BaggingClassifier	0.69	0.71	None	0.72
LinearDiscriminantAnalysis	0.62	0.71	None	0.67
LGBMClassifier	0.69	0.71	None	0.66
DecisionTreeClassifier	0.62	0.71	None	0.66
ExtraTreeClassifier	0.62	0.67	None	0.63
LogisticRegression	0.62	0.67	None	0.61
KNeighborsClassifier	0.62	0.67	None	0.66
LabelPropagation	0.56	0.63	None	0.58
LabelSpreading	0.56	0.63	None	0.58
SGDClassifier	0.56	0.62	None	0.63
Perceptron	0.44	0.54	None	0.38
NearestCentroid	0.38	0.54	None	0.41
CalibratedClassifierCV	0.62	0.46	None	0.59
GaussianNB	0.31	0.46	None	0.34
RidgeClassifier	0.56	0.42	None	0.53
RidgeClassifierCV	0.56	0.42	None	0.53
SVC	0.56	0.42	None	0.51
BernoulliNB	0.44	0.38	None	0.40
AdaBoostClassifier	0.50	0.33	None	0.43
QuadraticDiscriminantAnalysis	0.38	0.29	None	0.27

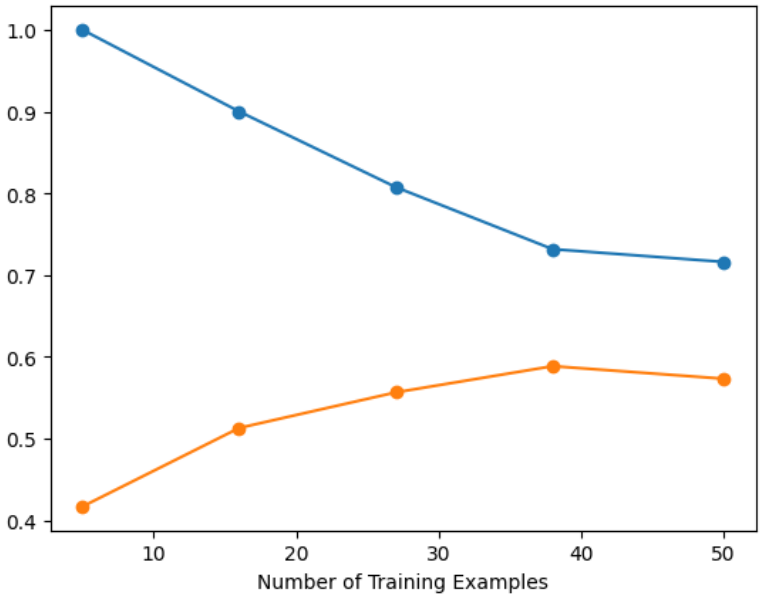
Selon les métriques fournies par Lazy Predict, trois algorithmes semblent particulièrement prometteurs pour cette tâche : l'ExtraTreesClassifier, le RandomForestClassifier et le LinearSVC.
Pour explorer davantage cette piste d'analyse, j'ai choisi de tester ces trois algorithmes.

- **LinearSVC:**

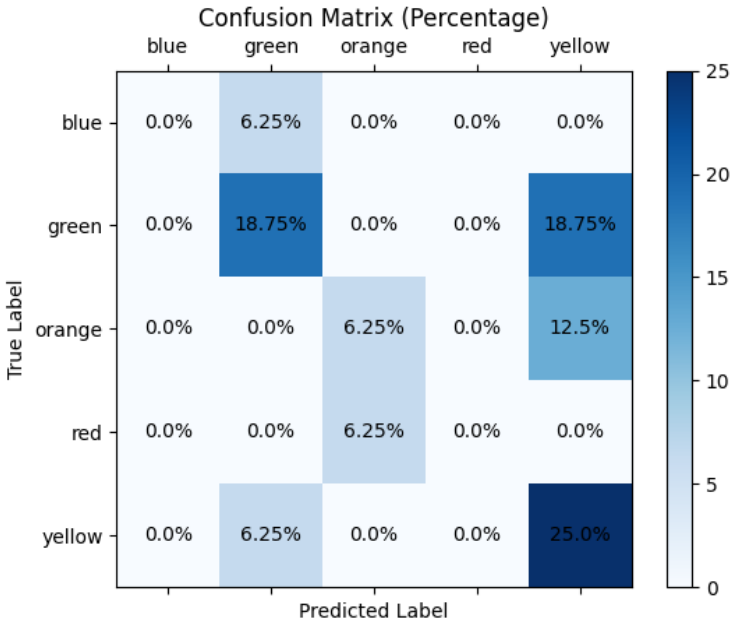
Classification Report:

	precision	recall	f1-score	support
blue	0.00	0.00	0.00	1
green	0.60	0.50	0.55	6
orange	0.50	0.33	0.40	3
red	0.00	0.00	0.00	1
yellow	0.44	0.80	0.57	5
accuracy			0.50	16
macro avg	0.31	0.33	0.30	16
weighted avg	0.46	0.50	0.46	16

Courbe d’apprentissage:



Matrice de confusion:

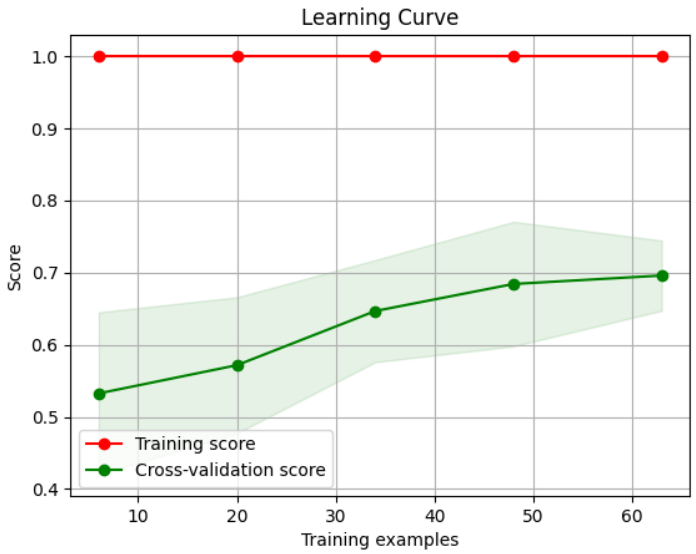


- **Random Forest:**

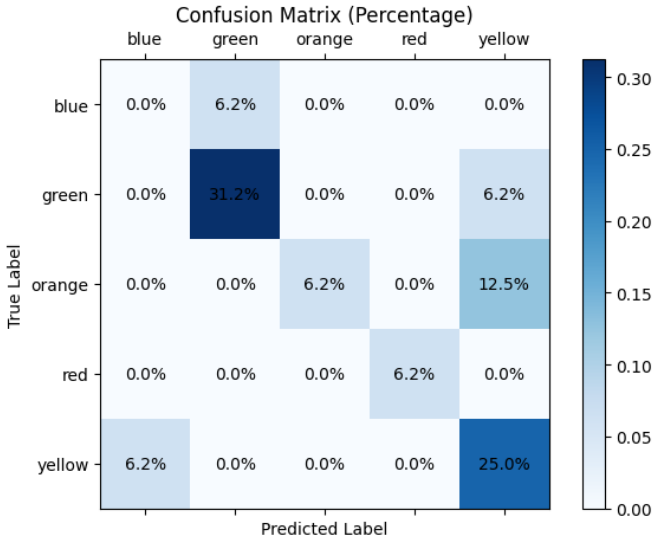
Classification report:

	precision	recall	f1-score	support
blue	0.00	0.00	0.00	1
green	0.83	0.83	0.83	6
orange	1.00	0.33	0.50	3
red	1.00	1.00	1.00	1
yellow	0.57	0.80	0.67	5
accuracy			0.69	16
macro avg	0.68	0.59	0.60	16
weighted avg	0.74	0.69	0.68	16

Courbe d'apprentissage:



Matrice de confusion:

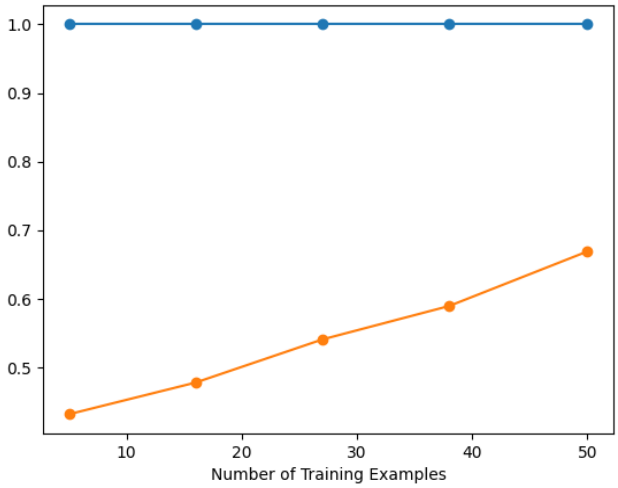


- **Extra tree classifier:**

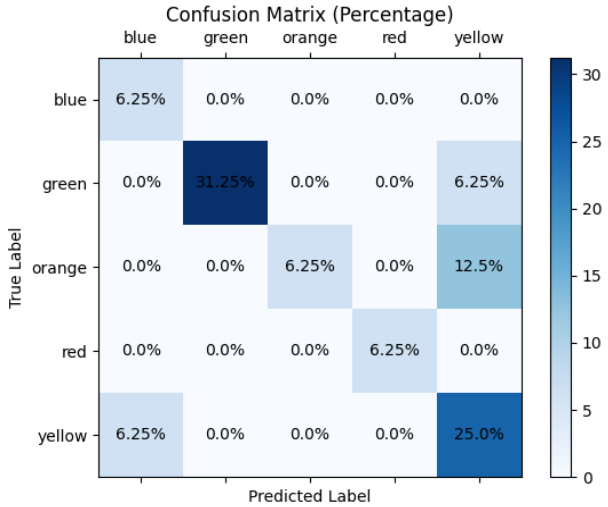
Classification report:

	precision	recall	f1-score	support
blue	0.50	1.00	0.67	1
green	1.00	0.83	0.91	6
orange	1.00	0.33	0.50	3
red	1.00	1.00	1.00	1
yellow	0.57	0.80	0.67	5
accuracy			0.75	16
macro avg	0.81	0.79	0.75	16
weighted avg	0.83	0.75	0.75	16

Courbe d'apprentissage:



Matrice de confusion:



Avant d'analyser les métriques des différents modèles, j'ai réalisé une étude pour déterminer sur quelle métrique me concentrer afin d'évaluer correctement les différents modèles par rapport à ma problématique. J'ai donc choisi la moyenne macro du F1 score, car je souhaite que chaque classe à prédire soit traitée de manière équitable, étant donné que nous ne connaissons pas la distribution des classes à prédire des autres territoires. je ne sais pas quel type de zones mon algorithme devra évaluer, ni si elles respecteront la proportion du jeu de données initial.

“Le score F1 moyen macro (ou score F1 macro) est calculé en utilisant la moyenne arithmétique (aussi connue sous le nom de moyenne non pondérée) de tous les scores F1 par classe. Cette méthode traite toutes les classes de manière égale, indépendamment de leurs valeurs de support.”source

Label	Per-Class F1 Score	Macro-Averaged F1 Score
 **Airplane**	0.67	$\frac{0.67 + 0.40 + 0.67}{3}$ **= 0.58**
 **Boat**	0.40	
 **Car**	0.67	

Suite à cette phase, je peux confirmer que mes algorithmes présentent une efficacité relative si je les compare à un algorithme naïf hypothétique. Étant donné la distribution inégale de mon jeu de données, cet algorithme naïf pourrait se contenter de prédire systématiquement la classe majoritaire, ici "vert", ce qui me donnerait un macro F1 score de 11%.

Classification report de l’algorithme naïf :

	precision	recall	f1-score	support
blue	0.00	0.00	0.00	1
green	0.38	1.00	0.55	6
orange	0.00	0.00	0.00	3
red	0.00	0.00	0.00	1
yellow	0.00	0.00	0.00	5
accuracy			0.38	16
macro avg	0.07	0.20	0.11	16
weighted avg	0.14	0.38	0.20	16

	LinearSVC	Random Forest	Extra Tree Classifier
Score F1 Macro	30%	60%	75%

Sur la base de la métrique sélectionnée, ainsi que des autres métriques disponibles, l'Extra Tree Classifier semble être l'algorithme le plus adapté à notre problématique.
Il est à noter que le volume de données (support) utilisé pour chaque classe est relativement faible. Cela pourrait suggérer que ces métriques sont biaisées et qu'elles pourraient ne pas refléter le véritable comportement de l'algorithme lorsqu'il sera déployé en production avec de nouvelles données. Pour pallier ce problème, on pourrait envisager de mettre en œuvre une validation croisée ou d'augmenter le pourcentage de données utilisées pour le test (par exemple, passer de 20% à 30%). Cependant, comme l'objectif principal de cet algorithme - démontrer la faisabilité de cette prédiction avec notre jeu de données - a été atteint, j'ai décidé de passer à l'étape suivante et d'explorer une autre approche.

3 Développement du modèle de prédiction basé sur les données images

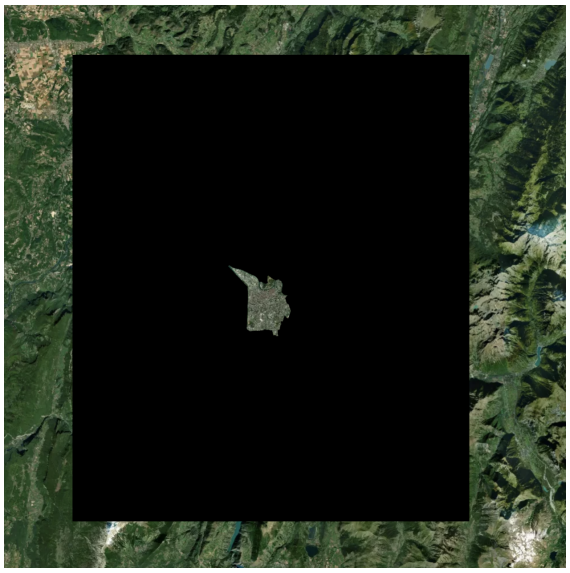
Suite à une consultation avec mes formateurs de Simplon, au cours de laquelle j'ai présenté mon projet et les algorithmes que j'ai utilisés, ils m'ont suggéré une approche pour résoudre ma problématique. L'idée serait d'essayer de prédire la valeur des îlots de chaleur urbains en utilisant des images satellites de la zone sélectionnée. Adopter cette stratégie signifierait que je pourrais augmenter les données sur les images et tenter une approche de Deep Learning, qui pourrait donner des métriques plus prometteuses. De plus, cela me permettrait d'utiliser des méthodes plus sophistiquées, telles que les réseaux neuronaux profonds (Deep Neural Networks) ou le Transfer Learning.

Bien que prometteuse, cette approche modifie significativement la manière dont l'utilisateur interagit avec l'application. Au lieu de fournir les statistiques de sa zone résidentielle, l'utilisateur devrait nous transmettre une image de son quartier ou de ses environs pour obtenir une prédiction. Ceci permettrait également l'utilisation de mon application web même en l'absence de données numériques disponibles.

Un autre aspect qui est modifié concerne la récupération des données. En effet, ces dernières ne seraient plus fournies par la Métropole de Grenoble mais directement fournies par l’utilisateur.

3.1 Description des données

Pour récupérer ces images satellites de la ville de Grenoble, j'ai initialement utilisé QGIS. C'est un logiciel libre de système d'information géographique (SIG) qui permet de visualiser, éditer et analyser des données géospatiales. Cela m'a permis de créer un fichier GeoJSON couvrant une large zone rectangulaire et d'ajuster l'emplacement de la ville de Grenoble (que j'ai par la suite obscurci, à l'exception de la zone nécessaire, avec l'API de Mapbox).
Cette démarche permet d'exclure les zones situées hors des limites de la ville lors de la capture de l'image avec l'API.



Après cette étape, j'ai ajouté une autre couche sur Mapbox comportant les contours de toutes les Zones de Classification Locale (LCZ) de Grenoble. J'ai rempli tous les quartiers en noir afin d'assombrir l'ensemble de la carte.

Pour obtenir des images satellites claires, j'ai utilisé l'API de Mapbox. Celle-ci m'a permis de faire une boucle sur chacune de mes LCZ, de la rendre transparente et de capturer une image haute définition de cette dernière :

Après cette phase de collecte de données, j'ai constitué un DataFrame Pandas contenant 79 images. Ce dernier contient le chemin d'accès de l'image ainsi que son étiquette (label).

image path	icu value
path/to/image	valeur icu
path/to/image	valeur icu

3.2 Entraînement et évaluation du modèle

Dans une première étape, j'ai décidé de réaliser un Lazy Predict sur les images converties en tableaux numpy. Cela, dans le but de vérifier si, même sans augmentation des données, nous pourrions obtenir des résultats assez satisfaisants. Cependant, il était fort probable que les métriques ne seraient pas prometteuses en raison de la quantité limitée de données. Par conséquent, l'approche du Deep Learning ne me semblait pas viable étant donné le nombre restreint d'images dans le jeu de données.

Lazy predict:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score
KNeighborsClassifier	0.44	0.48	None	0.44
DecisionTreeClassifier	0.44	0.29	None	0.45
AdaBoostClassifier	0.44	0.25	None	0.38
LinearDiscriminantAnalysis	0.38	0.22	None	0.31
CalibratedClassifierCV	0.38	0.21	None	0.30
GaussianNB	0.31	0.21	None	0.30
DummyClassifier	0.38	0.20	None	0.20
ExtraTreeClassifier	0.31	0.20	None	0.31
SVC	0.38	0.20	None	0.20
QuadraticDiscriminantAnalysis	0.38	0.20	None	0.24
LabelPropagation	0.06	0.20	None	0.01
LabelSpreading	0.06	0.20	None	0.01
NearestCentroid	0.31	0.18	None	0.28
LGBMClassifier	0.31	0.18	None	0.27
BernoulliNB	0.31	0.18	None	0.27
BaggingClassifier	0.31	0.17	None	0.24
SGDClassifier	0.25	0.17	None	0.32
LogisticRegression	0.31	0.17	None	0.19
ExtraTreesClassifier	0.25	0.15	None	0.20
RandomForestClassifier	0.25	0.14	None	0.20
LinearSVC	0.19	0.13	None	0.20
PassiveAggressiveClassifier	0.19	0.13	None	0.21
Perceptron	0.19	0.13	None	0.21

Après avoir observé des résultats peu satisfaisants, j'ai procédé à une augmentation d'images sur mon échantillon d'entraînement (80% du dataset) en utilisant des techniques qui ne modifient pas la couleur des images du jeu de données, étant donné que celles-ci sont importantes pour la prédiction.

J'ai utilisé la bibliothèque Python "albumentations" pour effectuer des augmentations d'images. Voici une explication de chaque augmentation :

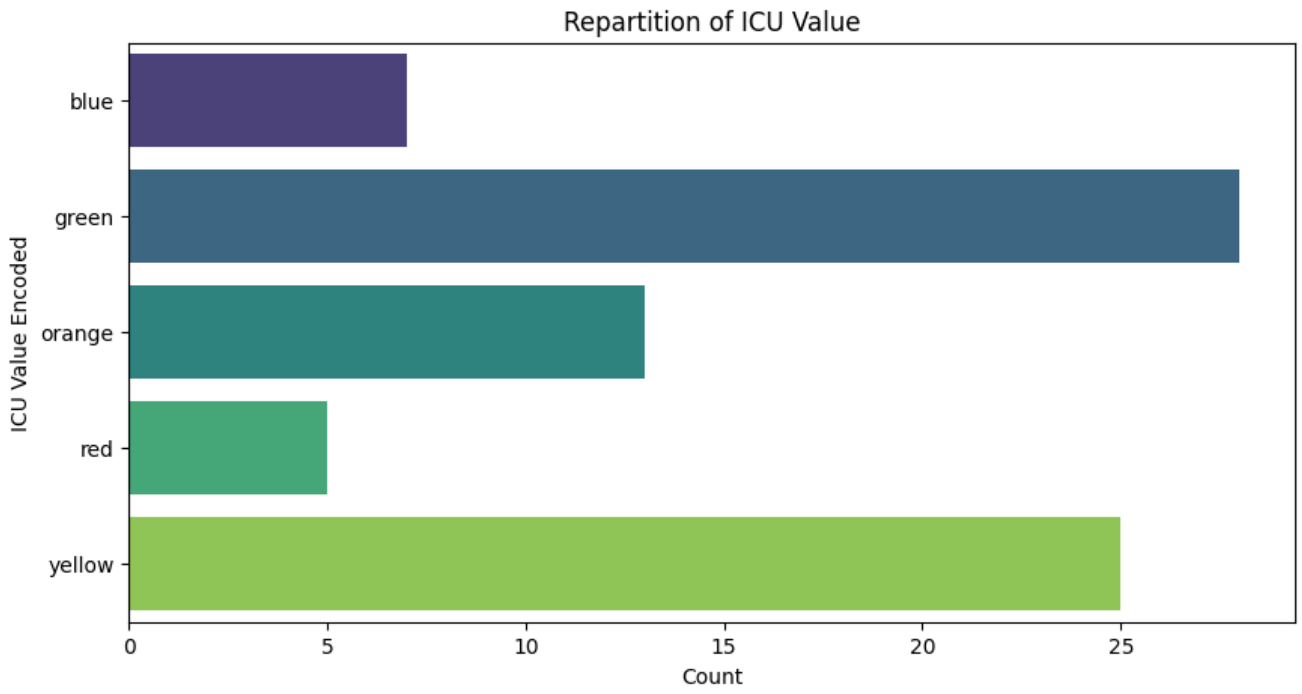
1. HorizontalFlip: Cette augmentation effectue une symétrie horizontale sur l'image, ce qui signifie que l'image peut être retournée horizontalement.

2. VerticalFlip: Cette augmentation effectue une symétrie verticale sur l'image, ce qui signifie que l'image peut être retournée verticalement.

3. ElasticTransform: Cette augmentation applique une transformation élastique à l'image.

4. ShiftScaleRotate: Cette augmentation effectue une combinaison de translation, de mise à l'échelle et de rotation sur l'image.

Répartition initiale des données :



Répartition des données après l'augmentation :

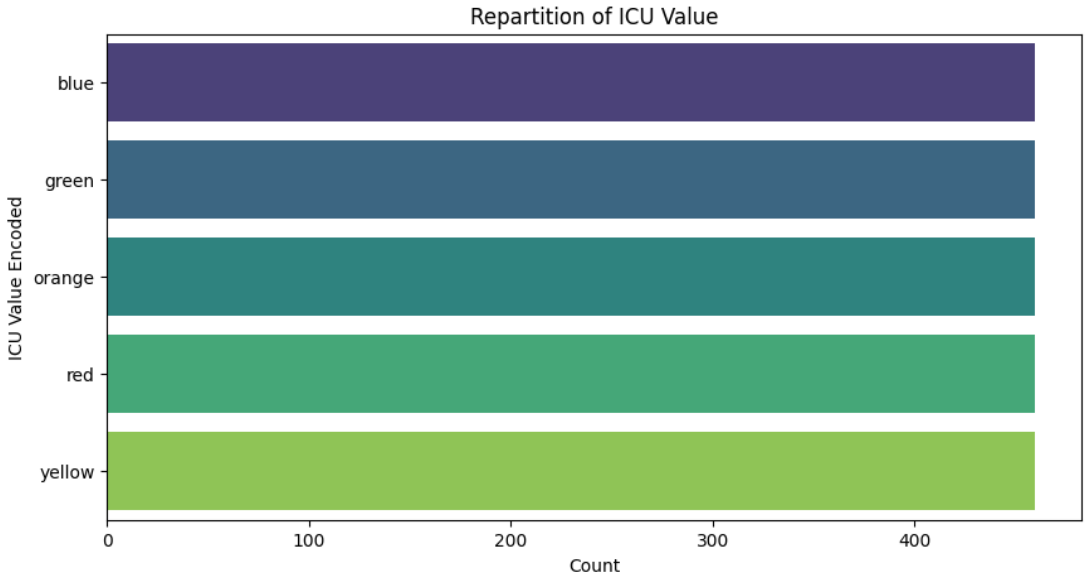
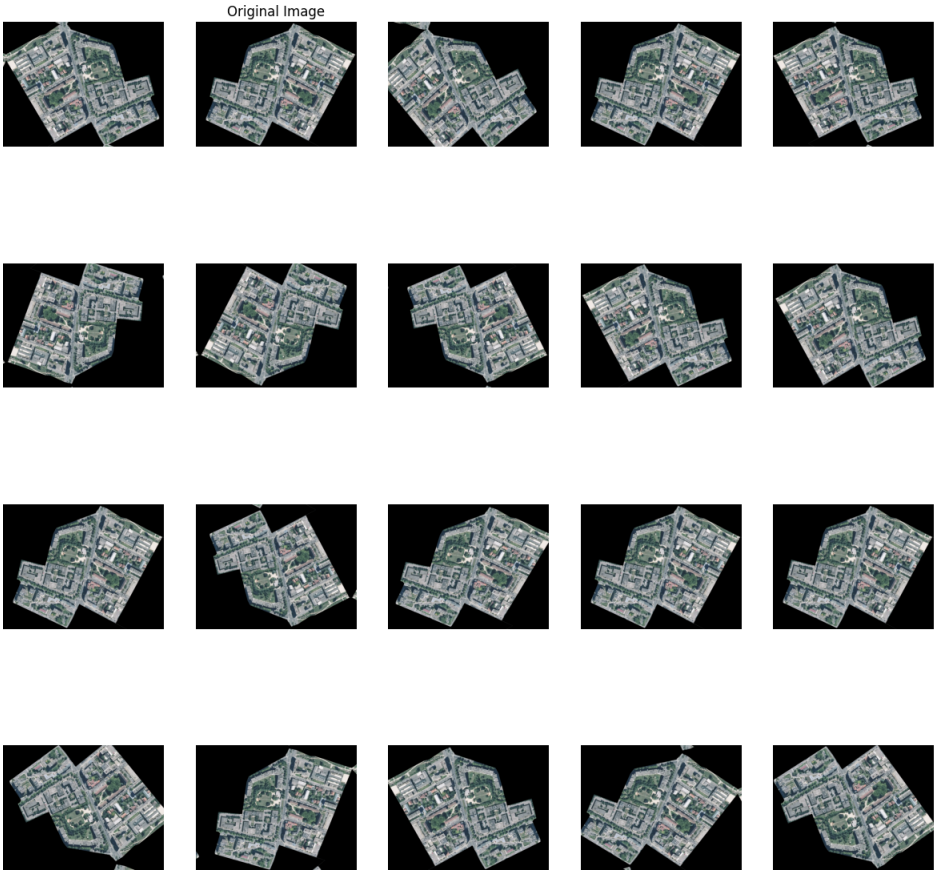


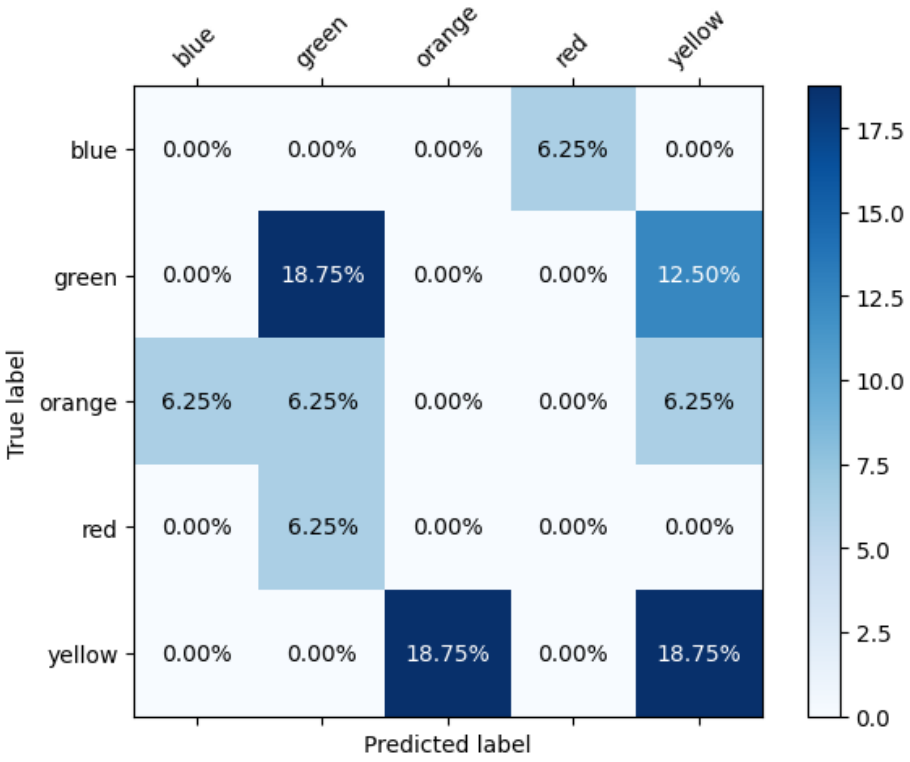
illustration du résultat de l’augmentation de données :



Après l'augmentation d'images, j'ai exploré plusieurs approches, dont le transfer learning qui “consiste à compléter l'apprentissage d'un modèle de machine learning, préalablement entraîné à résoudre une tâche donnée, en vue de lui permettre de résoudre une tâche similaire, généralement plus précise” en utilisant l'algorithme MobileNet que j'ai choisi car il est largement utilisé dans l'état de l'art de la vision par ordinateur.
MobileNet est un modèle pré-entraîné sur le vaste ensemble de données ImageNet, ce qui lui confère une connaissance approfondie des caractéristiques visuelles et des motifs dans les images. Il est réputé pour sa capacité à extraire des caractéristiques significatives à partir d'images, ce qui en fait un choix populaire pour le transfer learning

MobileNet:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.60	0.60	0.60	5
2	0.00	0.00	0.00	3
3	0.00	0.00	0.00	1
4	0.50	0.50	0.50	6
accuracy			0.38	16
macro avg	0.22	0.22	0.22	16
weighted avg	0.38	0.38	0.38	16

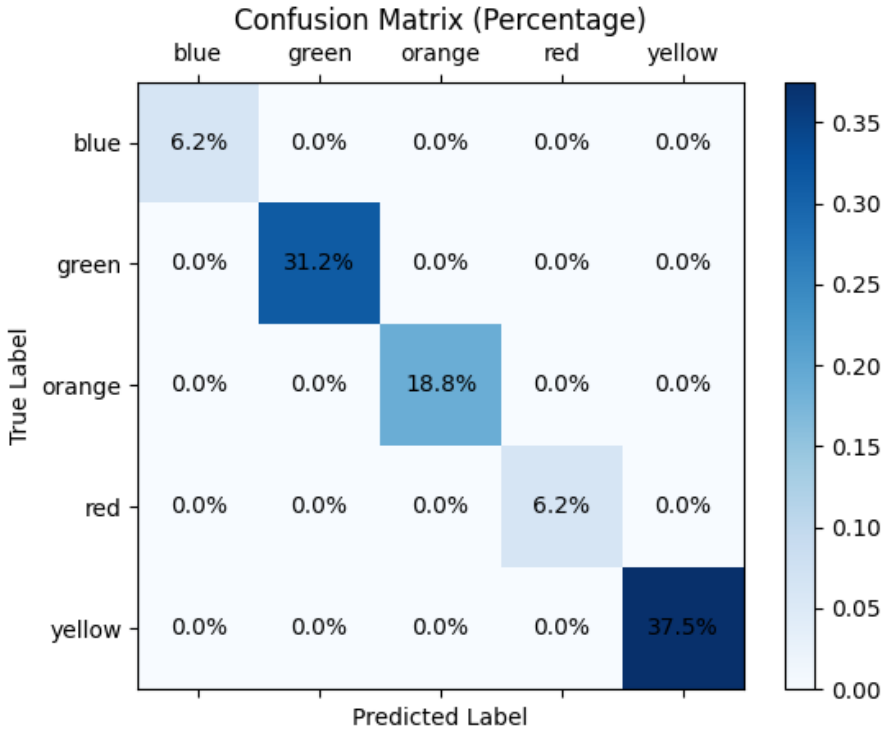


Vu que je n'ai pas obtenu de résultats satisfaisants, j'ai décidé de suivre une autre approche.

J'ai utilisé l'algorithme MobileNet pour l'extraction de caractéristiques, puis j'ai utilisé un modèle d'apprentissage automatique tel que le XGBClassifier pour prédire les valeurs des îlots de chaleur d'une LCZ à partir des caractéristiques extraites.

XGBClassifier:

	precision	recall	f1-score	support
blue	1.00	1.00	1.00	1
green	1.00	1.00	1.00	5
orange	1.00	1.00	1.00	3
red	1.00	1.00	1.00	1
yellow	1.00	1.00	1.00	6
accuracy			1.00	16
macro avg	1.00	1.00	1.00	16
weighted avg	1.00	1.00	1.00	16



Comparé au macro F1 score d'un algorithme aléatoire (12%), je peux affirmer que mon algorithme est nettement plus performant. De plus, il est intéressant de noter que cet algorithme, basé sur des images et utilisant des techniques d'augmentation de données, obtient un macro F1 score supérieur à ceux des algorithmes numériques testés jusqu'à présent. Ainsi, nous pouvons affirmer que cette approche de prédiction des îlots de chaleur urbains semble très prometteuse, malgré le faible volume de données initialement utilisé et les données de support employées pour évaluer le modèle.

Algorithme Aléatoire :

Random Classifier Classification Report:

	precision	recall	f1-score	support
blue	0.00	0.00	0.00	1
green	0.25	0.20	0.22	5
orange	0.00	0.00	0.00	3
red	0.00	0.00	0.00	1
yellow	0.50	0.33	0.40	6
accuracy			0.19	16
macro avg	0.15	0.11	0.12	16
weighted avg	0.27	0.19	0.22	16

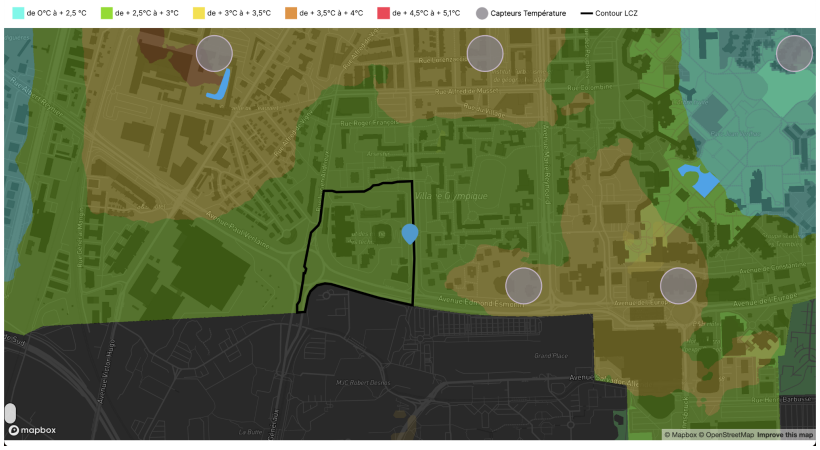
4 Développement de l’application Web

Sur le front du développement web, j'ai créé deux sites. Le premier est destiné aux habitants de la métropole, où nous présentons la problématique des îlots de chaleur urbains sur la ville de Grenoble. Le second est un site où, après connexion, sont intégrés les deux algorithmes d'IA choisis pour une prédiction basée sur des images et une autre basée sur des données numériques. L'objectif final est de les fusionner en un seul site.

En ce qui concerne le site de Data Storytelling, j'ai choisi d'utiliser temporairement le framework web Flask, car il est facile à utiliser. Mon objectif final est de migrer vers le framework utilisé dans l'application web dans laquelle sont utilisés les algorithmes de prédiction, à savoir Django.

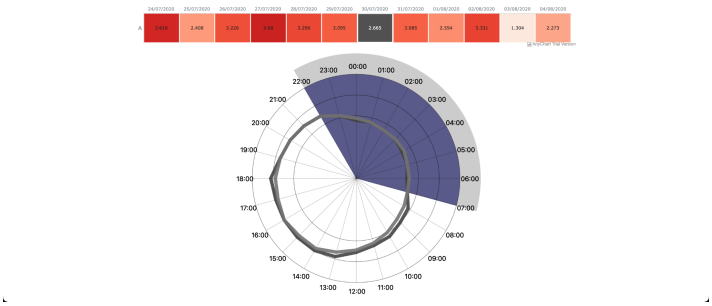
J'ai conçu ce site de Data Storytelling avec Figma, une application de design collaboratif gratuite basée sur le Cloud, qui permet aux designers UX/UI de réaliser des prototypes. Figma est couramment utilisé par les UI designers. Les graphiques utilisés pour faciliter la compréhension de ce phénomène ont été réalisés en JavaScript, avec des bibliothèques telles que D3.js, AnyCharts, Mapbox, etc.

Carte Créée Avec MapBox



Heatmap créée avec AnyChart et radar chart créé avec D3.js.

EXPLOREZ LA COURBE DE TEMPÉRATURE DE VOTRE QUARTIER ET COMPAREZ LA À LA STATION DE RÉFÉRENCE SITUÉE AU VERSOUD

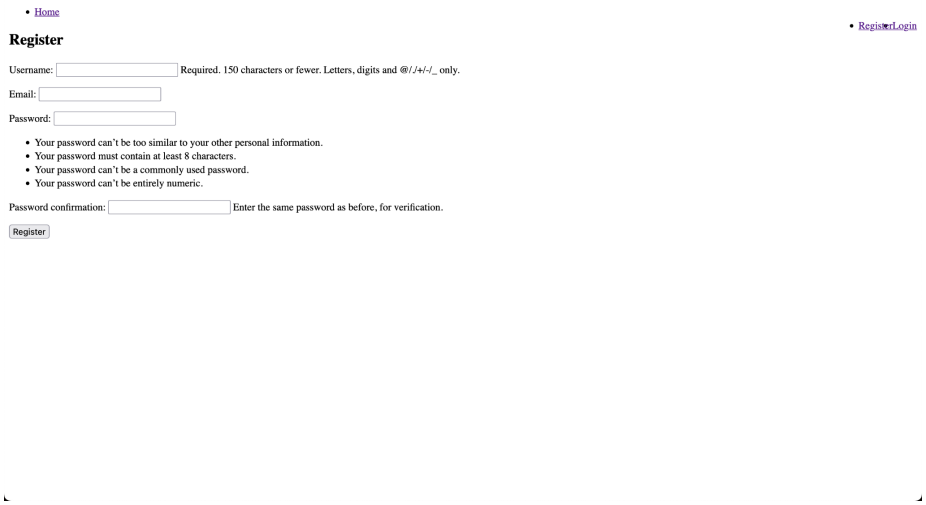


4.1 Application de prédiction

Pour l'application web, j'ai créé une application en utilisant le framework Python Django. Cette application comporte un espace utilisateur qui permet d'utiliser les deux types d'algorithmes sélectionnés pour faire les prédictions sur les îlots de chaleur.

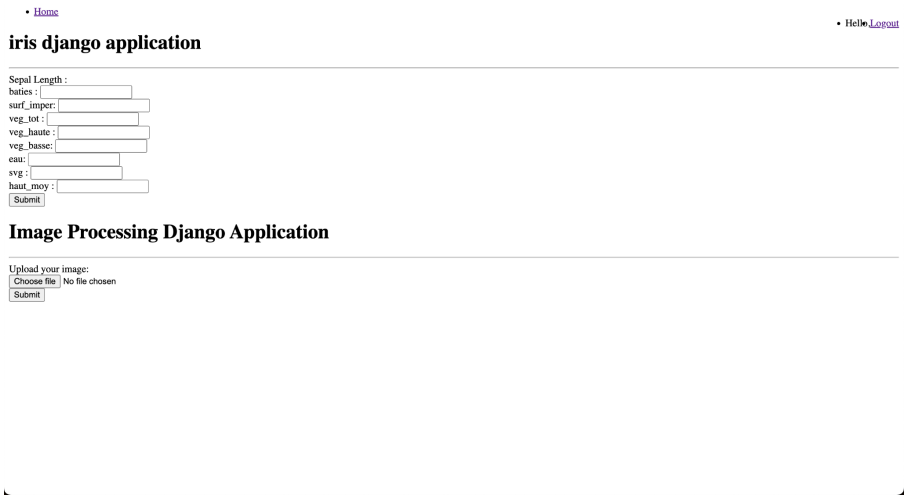
La fonction de prédiction est protégée par une authentification. L'utilisateur doit d'abord créer un compte, puis se connecter avec ses identifiants.
Pour ce faire, j'ai utilisé le système d'authentification de Django et j'ai restreint l'accès à la fonction de prédiction uniquement aux utilisateurs qui se sont inscrits.

Page d'enregistrement:



Login Page

Page de prédiction

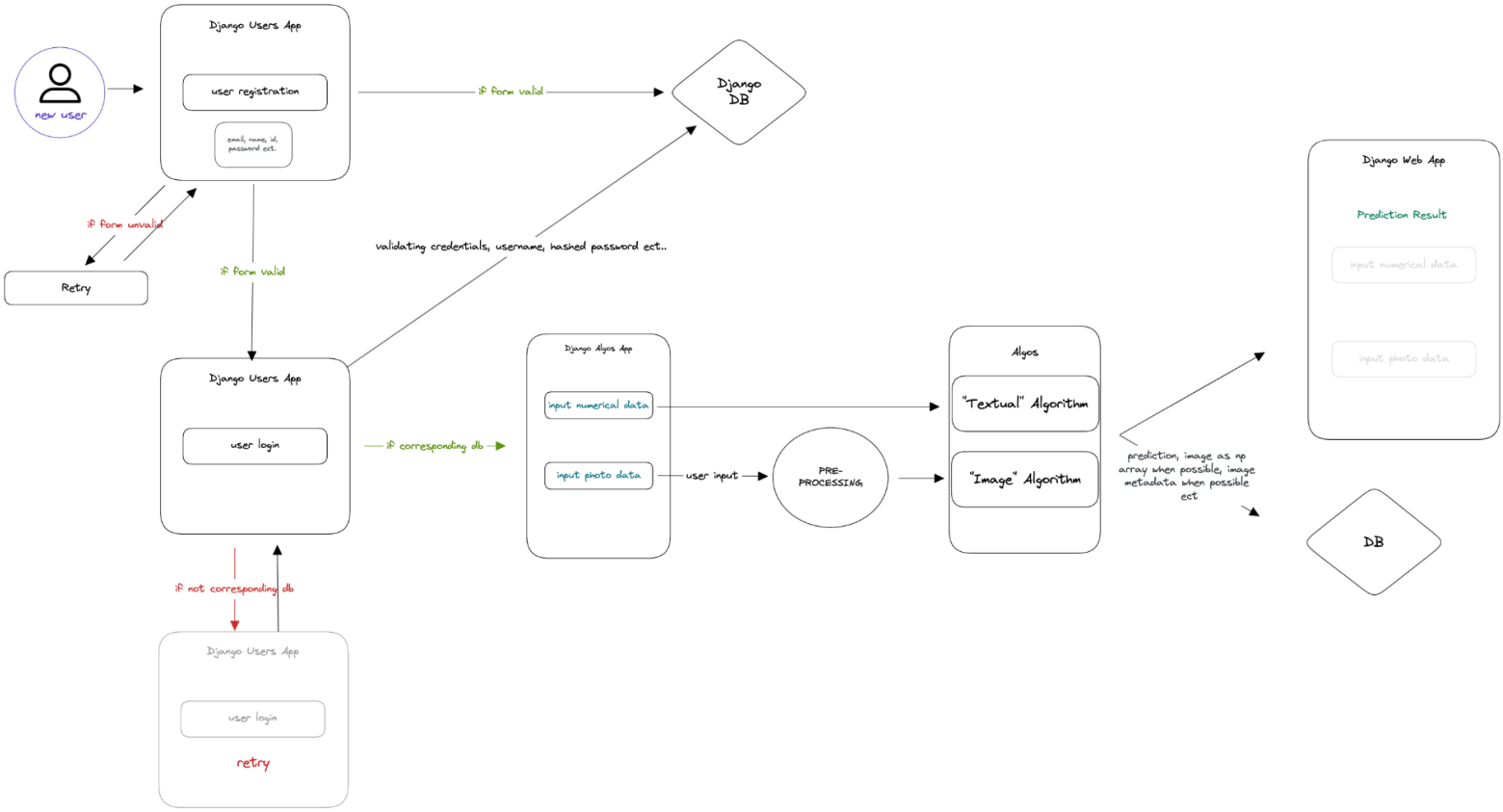


J'ai également créé une maquette d'interface utilisateur sur Figma pour rendre l'application plus facile à comprendre et à utiliser, mais je n'ai pas eu le temps de l'implémenter.

Voici la maquette Figma de la page dédiée aux algorithmes que j'ai réalisée :

4.2 les éléments de conception technique

Schema Technique

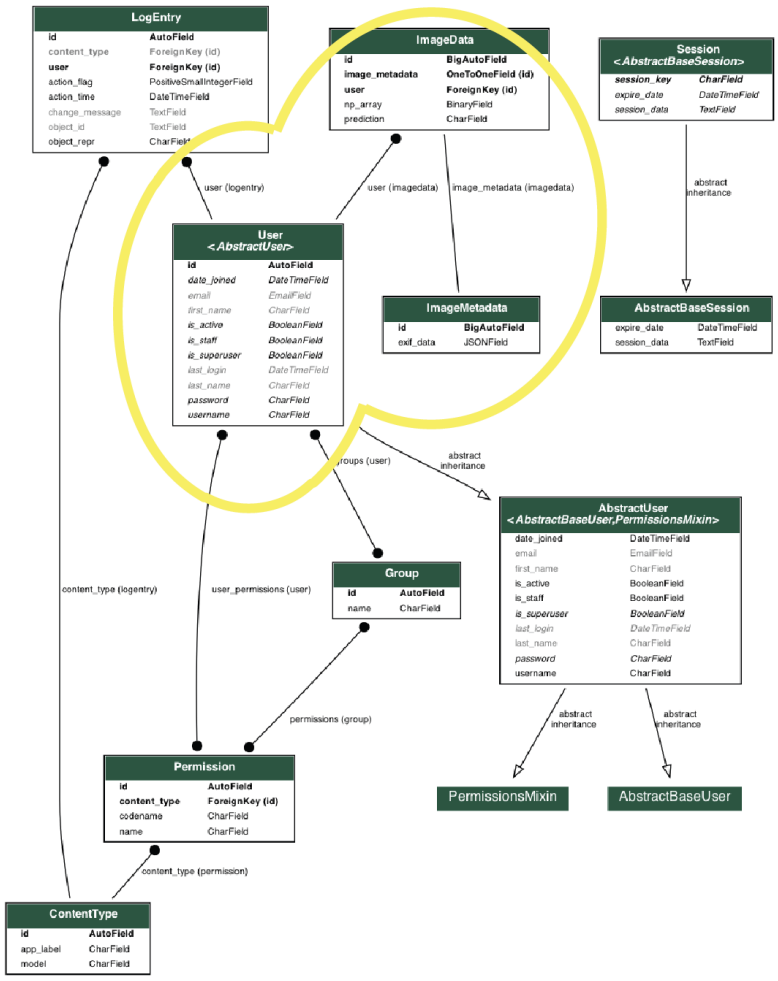


Voici le schéma technique élaboré pour le volet "communes de la métropole". Ce schéma a été réalisé avant la création du site et met en évidence les algorithmes disponibles, la base de données utilisée, entre autres éléments.

Voici les étapes principales :

1. Dans un premier temps, un nouvel utilisateur doit s'inscrire sur la plateforme en fournissant plusieurs informations telles que son adresse e-mail, son nom d'utilisateur, son mot de passe, etc.
2. Après inscription, cet utilisateur peut se connecter à la plateforme, sous réserve que ses informations soient validées dans la base de données gérée par Django.
3. Une fois connecté, l'utilisateur a accès à deux algorithmes
4. Après cette étape, ces données sont envoyées aux algorithmes, sauf quand il s'agit de données d'images qui nécessitent un prétraitement.
5. Par la suite, plusieurs éléments sont enregistrés, comme le résultat de la prédiction, le tableau Numpy de l'image, ses métadonnées, etc. dans la base de données.
6. Enfin, la prédiction est affichée en fonction de l'algorithme utilisé.

4.3 BDD



J'ai mis en place une table nommée 'User' grâce à Django, ce qui facilite grandement la gestion des utilisateurs dans ma base de données relationnelle (sqlite3). Suite à cela, j'ai créé deux autres tables : 'ImageData' et 'ImageMetadata'. La table 'ImageData' est liée à la table 'User' selon une relation One-to-Many, que j'exploite exclusivement lorsque l'algorithme d'image est utilisé. Cette liaison est établie grâce à l'ID de l'utilisateur, qui sert de clé étrangère. De son côté, 'ImageMetadata' est connectée à la table 'ImageData' selon une relation One-to-One, puisqu'une image ne peut posséder qu'un unique ensemble de métadonnées. Dans la table 'ImageData', je stocke ensuite la prédiction ainsi que le tableau numpy associé aux pixels de l'image, mais seulement dans les cas où l'algorithme d'image est employé.
Dans 'ImageMetadata', je stocke les métadonnées d'une image et j'utilise la bibliothèque Pillow pour récupérer ces métadonnées.

4.4 Tests Unitaires

Voici Mes Tests Unitaires :

1. **PredictorViewTestCase** : Teste la vue `predictor`. Il met en place une usine de requêtes et un utilisateur de test, puis envoie une requête POST avec certaines données à l'URL du prédicteur. Il vérifie si le code de statut de la réponse est 200, ce qui signifie que la requête a réussi.
2. **ImageUploadFormTestCase** : Teste le formulaire `ImageUploadForm`. Il crée un fichier d'image fictif et vérifie si le formulaire est valide avec ce fichier d'image en entrée. Si le formulaire n'est pas valide, il imprime les erreurs du formulaire. Le test affirme que le formulaire doit être valide.
3. **URLTests** : Teste plusieurs URL de votre application :
   - predict : Un utilisateur connecté envoie une requête GET à l'URL de prédiction. Le test vérifie si le code de statut de la réponse est 200.

   - process_image : Un utilisateur connecté envoie une requête GET à l'URL de process_image. Le test vérifie si le code de statut de la réponse est 200.

   - metrics : Un utilisateur connecté envoie une requête GET à l'URL de metrics. Le test vérifie si le code de statut de la réponse est 200.

   - login : Envoie une requête GET à l'URL de login. Le test vérifie si le code de statut de la réponse est 200.

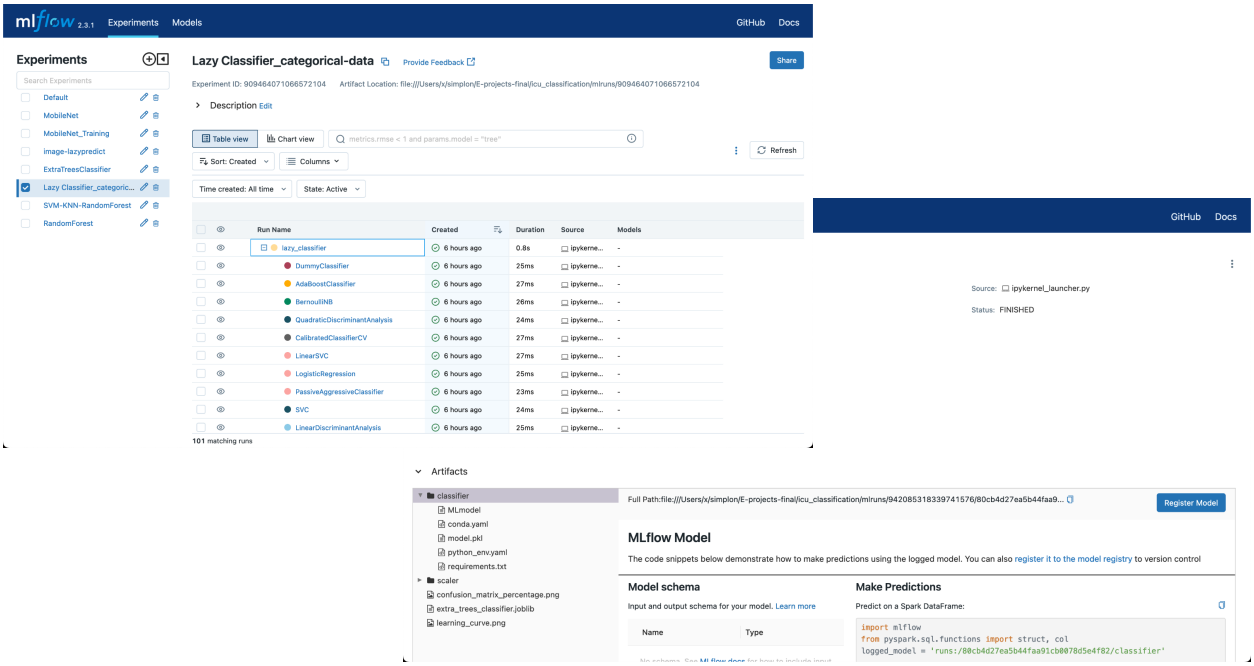
   - logout : Un utilisateur connecté envoie une requête GET à l'URL de logout. Le test vérifie si le code de statut de la réponse est 302, ce qui signifie que la requête a entraîné une redirection.

   - index : Envoie une requête GET à l'URL d'index. Le test vérifie si le code de statut de la réponse est 302, ce qui signifie également une redirection.

Ces tests aident à garantir que l'application fonctionne comme prévu en vérifiant les codes de statut de réponse et la validité des formulaires. Si l'un de ces tests échoue, cela signifie qu'il y a un bug ou un comportement inattendu dans l'application.

4.5 Suivi des modèles et de l'application

Pour suivre chaque algorithme testé, j'ai utilisé MLflow. Cet outil me permet d'enregistrer les métriques, les paramètres de chaque algorithme, de conserver leurs graphiques (tels que les matrices de confusion et les courbes d'apprentissage) et de sauvegarder l'algorithme sous les formats joblib et pkl. De plus, il me donne la possibilité de changer l'algorithme de production et de mise en scène sur l'application web.

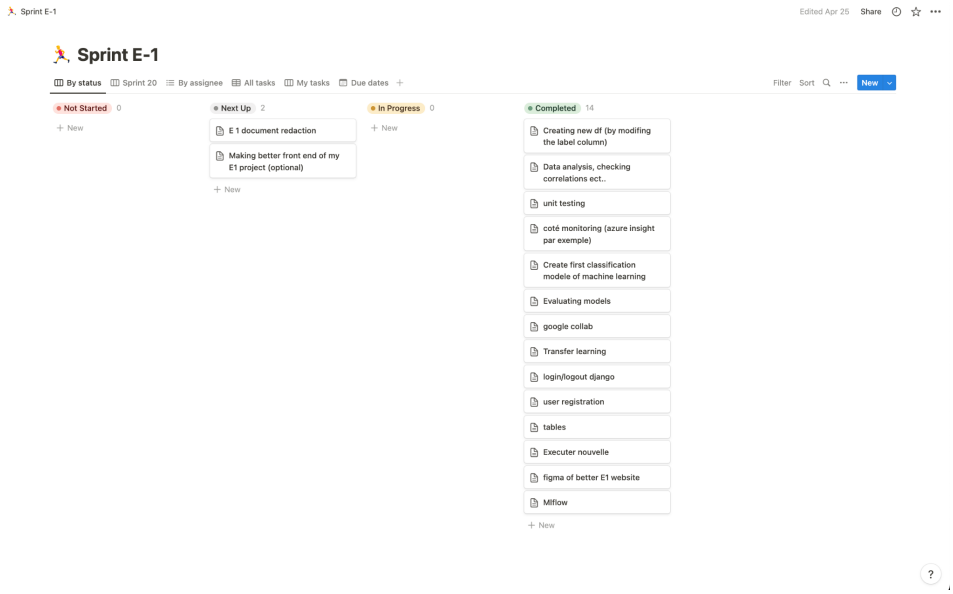


Dans la partie surveillance, j'ai fait appel à Prometheus, une plateforme de monitoring open source. Elle permet de suivre diverses métriques, comme le temps de prédiction d'un algorithme, le nombre de fois où il a été utilisé, le pourcentage d'utilisation du CPU, etc.
J'ai associé Prometheus à Grafana pour disposer d'un tableau de bord facile à comprendre. J'y ai affiché la somme des temps de réponse par algorithme au sein de mon application ainsi que la fréquence d'utilisation de chaque algorithme.

En outre, j'ai mis en place un système d'alerte par e-mail à l'aide de la bibliothèque Python RedMail. Ce système m'alerte sur mon adresse e-mail personnelle lorsqu'un algorithme utilise plus de 10% de mon CPU. L'e-mail d'alerte contient des informations telles que le pourcentage d'utilisation exacte du CPU par l'algorithme concerné et le nom de cet algorithme.

4.6 Organisation

Pour l'organisation du projet, j'ai utilisé Notion, qui me permet de créer des tâches et d'indiquer leur statut.



Sur cette partie Notion, on peut voir comment j'ai séparé les différentes tâches en catégories qui les catégorisent.

Voici les 4 catégories utilisées :

- Not Started : pour les tâches non commencées
- Next Up : pour les tâches à faire très prochainement
- In Progress : pour les tâches sur lesquelles je travaillais
- Completed : pour les tâches terminées

Je n'ai pas ressenti le besoin d'utiliser des sprints ou d'autres organisations en groupes, car j'ai travaillé de manière autonome sur ce projet sans l'aide d'une personne spécialisée en IA. Cependant, j'ai régulièrement fait des bilans à ma cheffe, Sonia Pelloux, pour la tenir au courant des pistes explorées et de l'avancement du projet.

5 Conclusion

Nous avions initialement plusieurs objectifs. Le premier était d'informer la population sur la problématique des îlots de chaleur urbains. Nous avons atteint cet objectif en créant une page web dotée de graphiques interactifs, élaborés avec des outils comme D3.js et AnyChart.js, pour expliquer le concept de manière claire et accessible à tous. Ainsi, les citoyens de Grenoble peuvent comprendre l'impact des îlots de chaleur urbains sur eux.

Un autre objectif était d'expérimenter l'utilisation de l'intelligence artificielle pour la prédiction des îlots de chaleur urbains, afin de déterminer si cette approche est viable pour l'avenir. Bien que cela nécessite plus de données et une expertise approfondie, nous avons réussi à démontrer la faisabilité de cette approche avec des résultats plutôt satisfaisants. Ces résultats suggèrent

qu'avec plus de temps, de ressources et de connaissances, il serait possible d'obtenir des résultats fiables pour une mise en production. Cela simplifierait le travail des chercheurs de la Métropole de Grenoble en facilitant la détection des îlots de chaleur urbains particulièrement élevés.

L'utilisation de l'intelligence artificielle est concrétisée par la mise en place de deux algorithmes de prédiction des îlots de chaleur urbains intégrés dans une application pour répondre aux différentes requêtes des utilisateurs, un pour l'analyse d'images et un autre pour le traitement de données numériques.

Nous avons donc la possibilité d'utiliser deux types d'algorithmes. Le premier, basé sur les images, peut permettre aux villes avec des ressources limitées d'identifier les zones présentant un risque d'îlot de chaleur urbain fort simplement à partir de photos satellites de leurs quartiers, ou, lorsque la fonctionnalité sera implémentée, d'utiliser des fichiers GeoJson. Le second algorithme, numérique, est destiné aux collectivités qui possèdent les données nécessaires. Cependant, à l'heure actuelle, il est préférable d'utiliser l'algorithme basé sur les images.

Ce projet m'a permis d'apprendre et d'approfondir de nombreux sujets abordés en formation. Il m'a fait réaliser la difficulté d'aborder des problématiques non conventionnelles et m'a fait prendre conscience de l'importance des données dans les problématiques d'IA.

J'ai également dû réaliser ce travail en intelligence artificielle de manière autonome, sans assistance de la part de quelqu'un de la Métropole de Grenoble, puisqu'il n'y avait pas de personnes spécialisées dans l'intelligence artificielle. Ce facteur m'a permis d'explorer de nombreuses pistes, de gagner en autonomie et d'apprendre énormément. Cependant, cela signifie également que je n'ai pas eu la possibilité d'avoir quelqu'un pour m'accompagner sur le volet de l'intelligence artificielle, qui aurait pu m'aider et m'orienter lorsque je faisais face à de grandes incertitudes et problématiques.

Pour améliorer ce projet, je pense qu'irl serait possible de collecter plus de données, d'obtenir des données satellitaires plus complètes et de considérer des approches que je n'ai pas pu analyser faute de données et de temps.

Références

Using deep-learning to forecast the magnitude and characteristics of urban heat island in Seoul Korea : https://www.nature.com/articles/s41598-020-60632-z

Décodage F1 : https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

MobileNet : https://arxiv.org/abs/1704.04861

Open Data : https://www.cnil.fr/fr/definition/open-data

Django : https://www.djangoproject.com/

Mlflow : https://mlflow.org/

Figma : https://www.figma.com/

D3.js : https://d3js.org/

Scikit Learn : https://scikit-learn.org/stable/

MapBox : https://www.mapbox.com/

Lazy Predict : https://lazypredict.readthedocs.io/en/latest/

Flask : https://flask.palletsprojects.com/en/2.3.x/

Prometheus : https://prometheus.io/