Brugeron Thomas
Condomines Julien

4ICS

Traitement de données massives

Rapport de projet

Partie 1

Brugeron Thomas
Condomines Julien

Table des matières

1. Introduction

Dans le monde numérique actuel, la surabondance d'images rend crucial le développement de systèmes de recommandation efficaces. Ces systèmes visent à faciliter la découverte de contenu pertinent pour les utilisateurs, en prenant en compte leurs préférences individuelles. Ce projet s'inscrit dans cette optique, en se concentrant sur la recommandation d'images personnalisées à travers un processus automatisé. En utilisant des techniques d'acquisition de données, d'étiquetage, d'annotation, d'analyse et de visualisation, ce projet vise à construire un système robuste et intelligent capable de comprendre et de répondre aux besoins des utilisateurs. Sur une durée de trois séances pratiques, nous explorerons chaque étape du processus, de la collecte des données à la mise en œuvre du système de recommandation, en passant par les tests et la documentation. L'objectif ultime est de fournir aux utilisateurs une expérience enrichissante et personnalisée, en leur proposant des images pertinentes correspondant à leurs goûts et intérêts.

2. Collecte des données

2.1 Vérification de l’existence du répertoire

Dans un premier temps, il est important de vérifier la présence d’un répertoire images au préalable. Sinon il faudra créer ce répertoire.

2.2 Récupération des images

Les images sont récupérées directement depuis wikidata grâce à des requêtes.

Dans ces requêtes il est possible de choisir la catégorie souhaitée mais également le nombre limite.

Dans notre cas, nous allons récupérer 100 photos.

2.3 Téléchargement des images

Nous avons mis en place un script en Python qui permet de télécharger des images à partir d'URLs. Voici comment cela se déroule :

Tout d'abord, nous envoyons une requête HTTP à l'URL de l'image spécifiée. Si la requête aboutit avec succès (code de statut HTTP 200), nous poursuivons le processus.

Ensuite, nous créons un dossier nommé "images" dans le répertoire de travail pour stocker les images téléchargées. Si ce dossier existe déjà, aucun nouveau dossier n'est créé.

Nous extrayons ensuite l'extension du fichier à partir de l'URL fournie, ce qui nous permet de conserver le format original de l'image lors de son enregistrement.

Pour garantir que chaque image téléchargée possède un nom de fichier unique, nous générons un nom de fichier en combinant le nom du monument (extraite d'une colonne nommée "monument" dans un DataFrame) avec un numéro d'index incrémental. Cela nous assure que chaque image téléchargée à un nom différent.

Enfin, nous enregistrons l'image téléchargée dans le dossier "images" avec le nom de fichier unique que nous avons généré. Cela nous permet de stocker toutes les images téléchargées de manière organisée pour une utilisation ultérieure.

2.4 Métadonnées

Nous avons ensuite extrait les métadonnées présentes dans chaque photo. Nous les avons par la suite stockés dans un fichier Json.

Pour extraire les métadonnées nous avons utilisé la bibliothèque « PIL ».

Nous avons décidé de garder :

- file_name
- format
- mode
- height
- length
- year
- hour

3. Etiquetage et annotation

Après avoir extrait les données métadonnées des photos provenant des Exifs nous allons maintenant chercher d’autres informations à récupérer.

3.1 Couleur prédominante

Nous utilisons des algorithmes de regroupement pour analyser les images et déterminer les couleurs principales.

En plus des informations précédemment mentionnées, nous examinons les couleurs présentes dans le fichier afin d'identifier la couleur dominante dans la photo. Pour ce faire, nous avons employé l'algorithme MiniBatchKMeans fourni par la bibliothèque sklearn.

3.2 TAGS

Au début nous voulions partir sur des tags manuels en laissant entrer l’utilisateur les tags de son choix sur chaque photo.

Puis nous avons décidé de rendre cela automatique. Pour chaque photo nous avons généré deux tags. Le premier tag est généré en récupérant le premier mot commun du titre de la photo. Le deuxième tag est en généré via la couleur prédominante qu’on aura trouvé au préalable.

4. Visualisation des données

Pour la visualisation de nos données nous avons décidé de partir sur des diagrammes bâtons pour chaque métadonnées présent dans le fichier. Cela permet de bien visualiser toutes les informations importantes que nous avons pu extraites.

Nous avons utilisé la librairie matplotlib de Python qui nous a permis de créer les différentes diagrammes

Nous avons donc plusieur diagrammes en annexe :

- nombre de photo par valeur de la clé file_name
- nombre d'image par valeur de la clé format
- nombre d'image par valeur de la clé height
- nombre d'image par valeur de la clé length
- nombre d'image par valeur de la clé color1
- nombre d'image par valeur de la clé color 2
- nombre d'image par valeur de la clé color 3
- nombre d'image par valeur de la clé year
- nombre d'image par valeur de la clé hour
- nombre d'image par valeur de la clé tags

5. User

5.1 Création des users et choix des images

Ici nous traitons la sélection d'images effectuée par chaque utilisateur. Trois utilisateurs, chacun d'eux a choisi aléatoirement 10 images parmi celles disponibles. Cette sélection est enregistrée dans une liste nommée `image_selection`.

Ce processus permet de préparer les données nécessaires à l'analyse des préférences et des comportements des utilisateurs par rapport aux images sélectionnées

5.2 Choix images favorites

Ici on va simuler le processus où chaque utilisateur choisit ses images favorites parmi celles qu'ils ont sélectionnées précédemment. Ensuite, nous construisons des DataFrames à partir des données collectées précédemment. Les informations sur les images sélectionnées, telles que l'année, l'heure et les couleurs, sont organisées dans des DataFrames pandas. De plus, les étiquettes de préférence ("Favorite" ou "NotFavorite") sont stockées dans un DataFrame séparé, plus simple pour le traitement.

5.3 Arbre de décision pour chaque user

Nous visualisons les arbres de décision créés pour chaque utilisateur afin de comprendre comment le modèle prend des décisions concernant les préférences d'images. Pour chaque utilisateur, nous utilisons la bibliothèque `matplotlib` pour créer une figure et afficher l'arbre de décision correspondant. Chaque arbre de décision affiché fournit un aperçu visuel des règles utilisées par le modèle pour classer les images sélectionnées comme favorites ou non favorites pour un utilisateur donné.

Annexes : - arbres de décision user 1

- arbres de décision user 2

- arbres de décision user 3

6. Recommandation Users

6.1 Initialisation des données pour users avec tests

Pour cette partie nous souhaitons commencer le processus de recommandations d'images pour les users, pour ce faire nous allons tenter d'entrainés notre modèle. Nous commençons par charger les données de métadonnées à partir du fichier `metadata.json`, qui contient des informations sur les images.

Ensuite, nous initialisons une liste `reco` pour stocker les indices des images recommandées pour chaque utilisateur.

Pour chaque utilisateur, nous parcourons toutes les images disponibles et préparons les données nécessaires pour la prédiction en utilisant les mêmes transformations que celles effectuées lors de l'entraînement du modèle. Nous utilisons ensuite le modèle d'arbre de décision correspondant à chaque utilisateur pour prédire si l'image est considérée comme favorite ou non.

Parfois, certaines des prédictions des photos ne passent pas, cela se produit lorsque le modèle rencontre des valeurs dans les données de test qui n'ont pas été observées lors de l'entraînement du modèle.

6.2 Tentative de proposition

A l'aide du modèle entrainés précédemment, nous allons tenter de fournir des recommandations aux users en se basant sur leurs préférences préalablement identifiées. Pour chaque utilisateur, une image est sélectionnée aléatoirement parmi les recommandations, puis affichée après normalisation du nom de fichier pour éviter les caractères spéciaux. Ensuite, les images favorites de l'utilisateur sont affichées si elles sont disponibles. Ce processus permet de fournir une expérience personnalisée à chaque utilisateur en leur présentant des images qui correspondent à leurs goûts.

7. Tests

Pour vérifier le bon fonctionnement des différentes fonctions de notre projet et évaluer l'efficacité du système de recommandation, nous avons mis en œuvre plusieurs méthodes de test. Voici comment nous avons procédé :

1. **Tests unitaires des fonctions individuelles** : Des tests unitaires ont été écrits pour chaque fonction afin de vérifier qu'elles retournent les résultats attendus pour différentes entrées. Par exemple, nous avons vérifié que la fonction `load_metadata` charge correctement les données depuis un fichier JSON, que la fonction `get_image_count` calcule le nombre d'images par valeur de chaque clé dans les métadonnées, et que la fonction `clean_and_normalize_filename` normalise correctement les noms de fichiers.
2. **Tests de bout en bout** : Le code a été exécuté dans son ensemble pour vérifier que toutes les étapes se déroulent comme prévu. Nous avons utilisé des données de test représentatives pour chaque fonctionnalité, en nous assurant que les données étaient variées et couvraient différents cas d'utilisation possibles.
3. **Validation visuelle des résultats** : Les résultats produits par le système de recommandation ont été examinés visuellement. Par exemple, nous avons vérifié que les diagrammes en barres affichent correctement le nombre d'images par valeur de chaque clé dans les métadonnées. De plus, nous avons vérifié que les images recommandées pour chaque utilisateur semblaient pertinentes par rapport à leurs préférences préalablement identifiées.
4. **Comparaison avec des données de référence** : Si des données de référence étaient disponibles, nous les avons utilisées pour comparer les résultats produits par notre système de recommandation. Par exemple, nous avons comparé les images recommandées avec les images favorites réelles des utilisateurs pour voir si elles correspondaient.

Brugeron Thomas
Condomines Julien

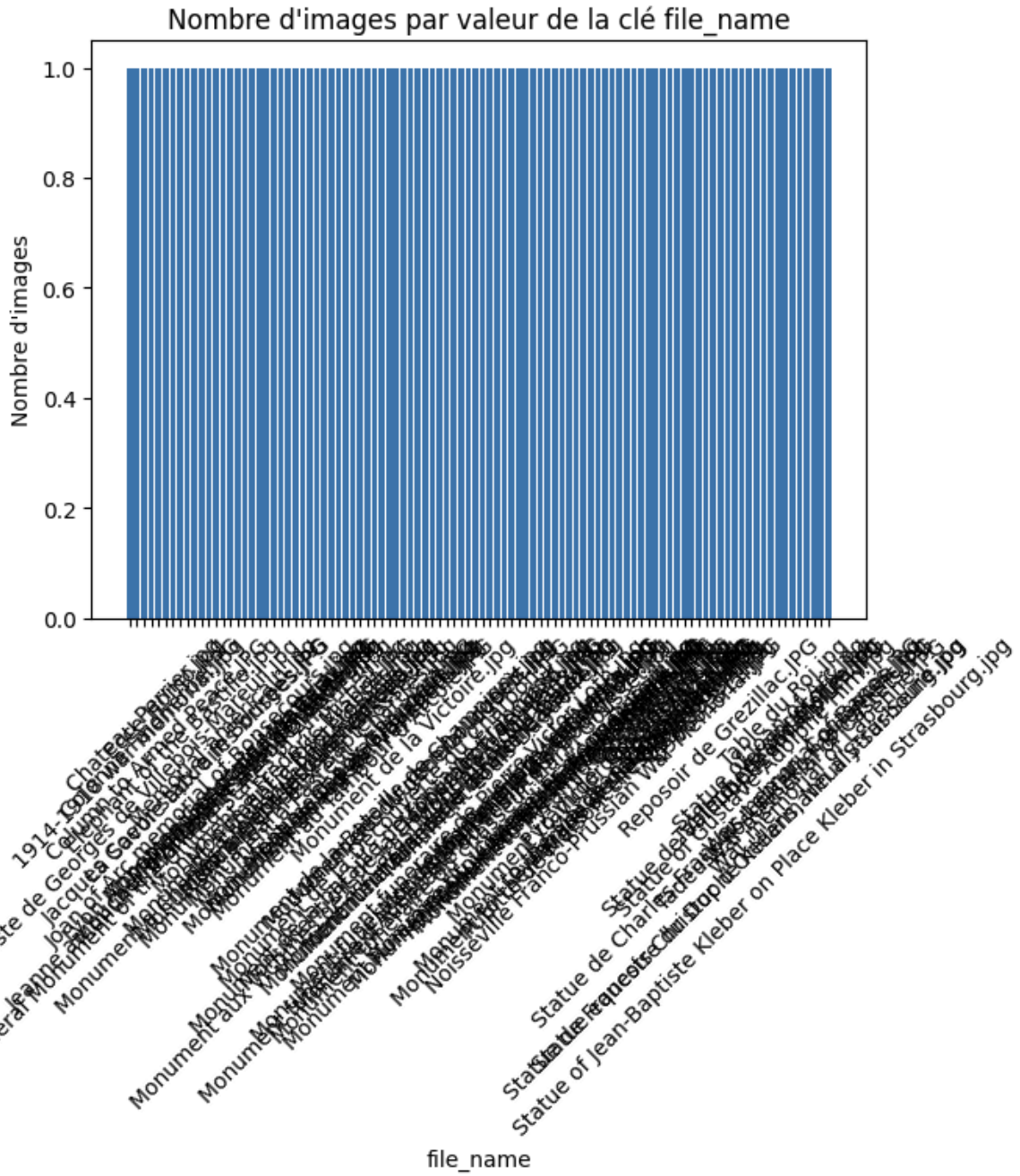
Auto-Evaluation

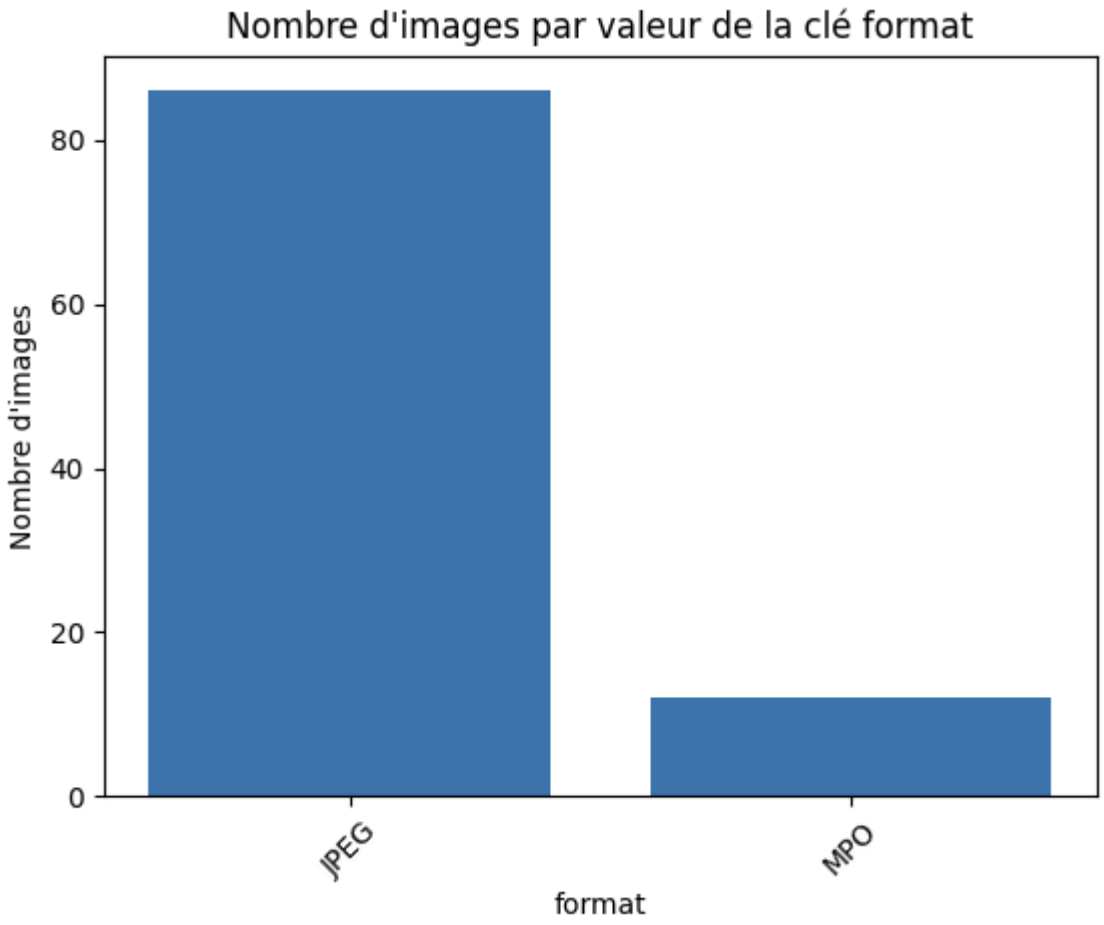
Nous avons essayé d’automatiser le plus de chose possible tout en respectant l'entièreté du cahier des charges

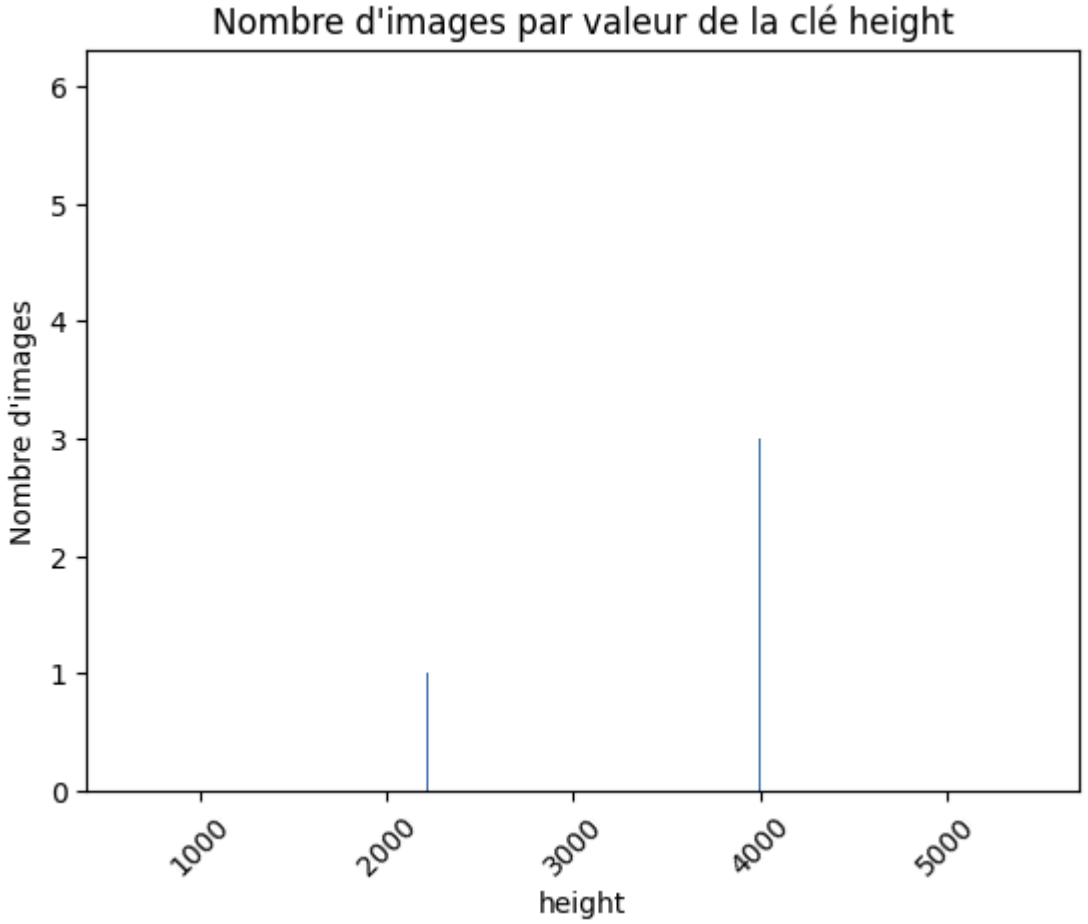
Remarques concernant les séances pratiques, les exercices et les possibilités d'amélioration.

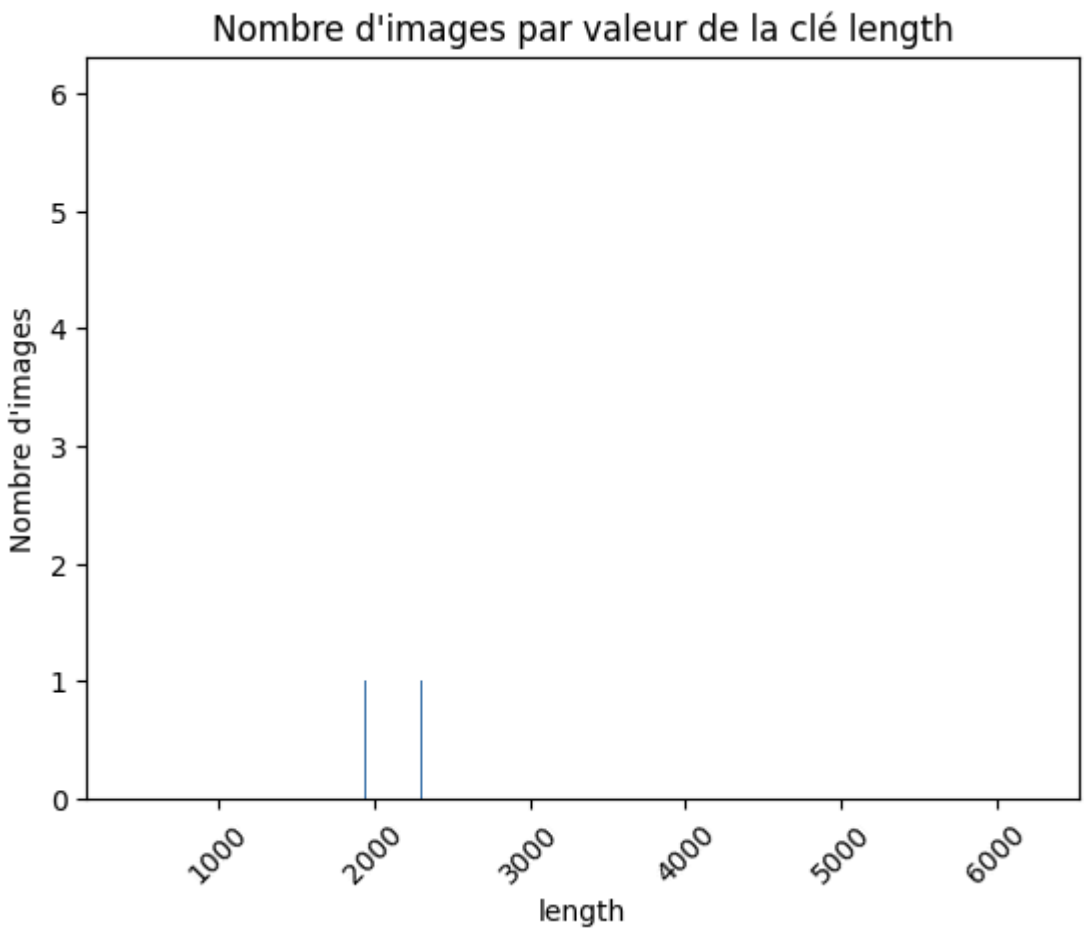
Les TPs et les exemples qui nous sont fournies permettent une meilleure compréhension et offrent une base sur laquelle s’appuyert pour commencer le développement.

Annexe

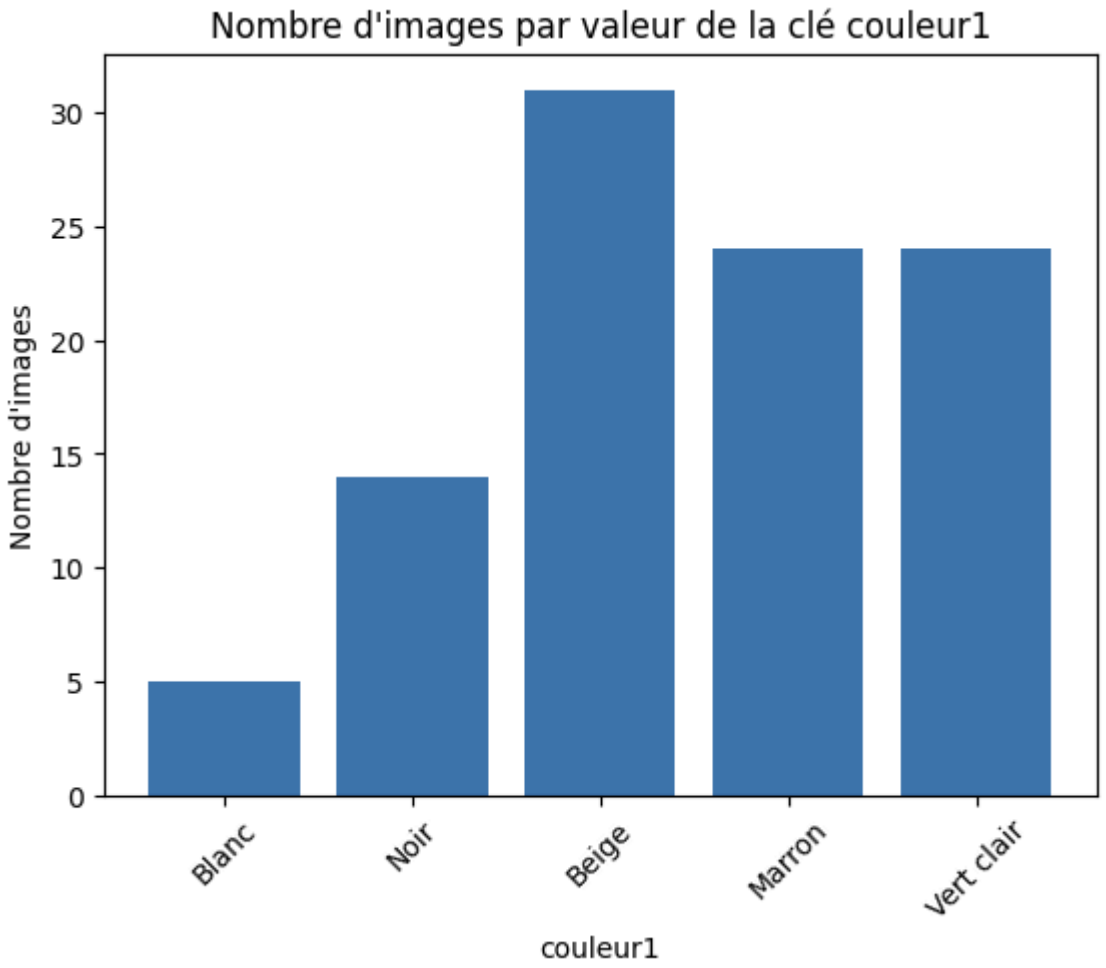

Nombre d'images par valeur de la clé file_name
Nombre d'images
1.0
0.8
0.6
0.4
0.2
0.0
file_name


Nombre d'images par valeur de la clé format
Nombre d'images
0
20
40
60
80
JPEG
MPO
format

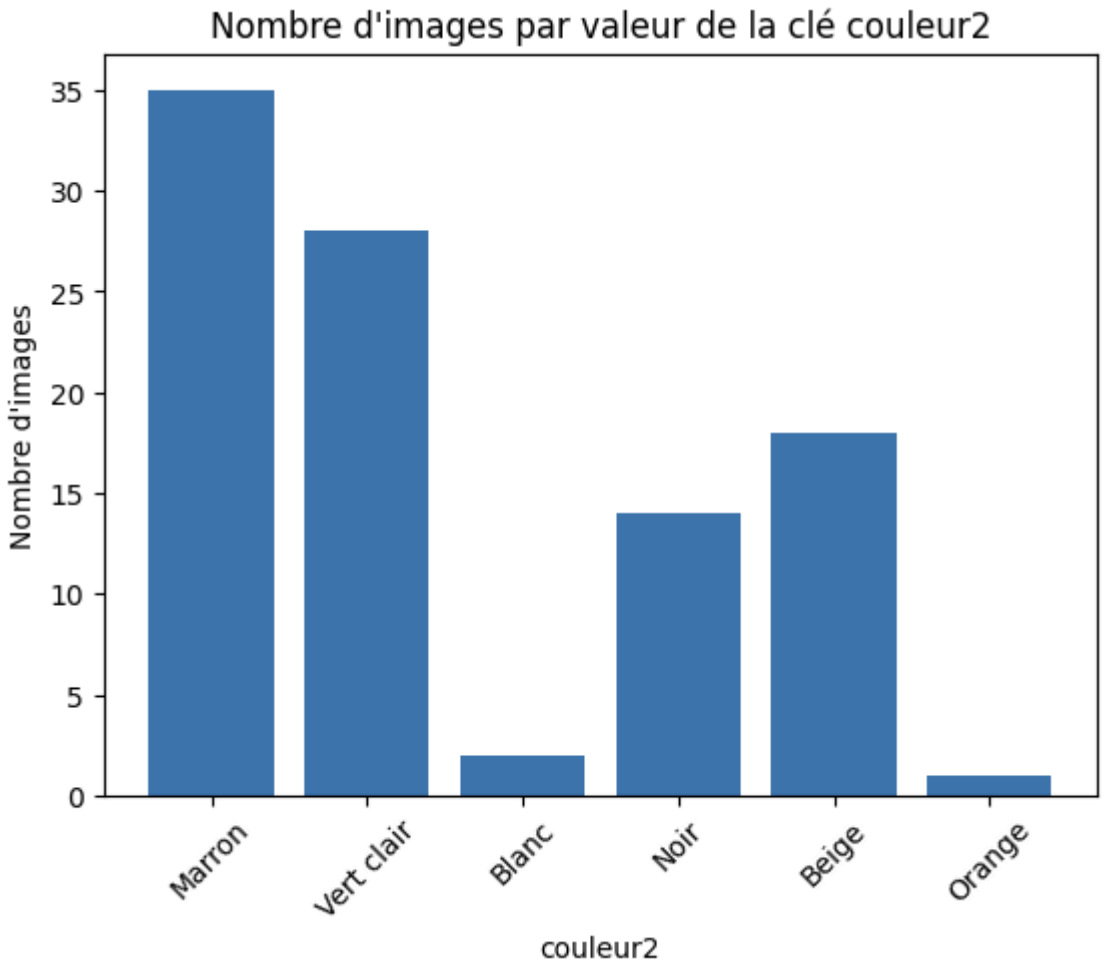

Nombre d'images par valeur de la clé height
Nombre d'images
0
1
2
3
4
5
6
1000
2000
3000
4000
5000
height


Nombre d'images par valeur de la clé length
Nombre d'images
0
1
2
3
4
5
6
1000
2000
3000
4000
5000
6000
length

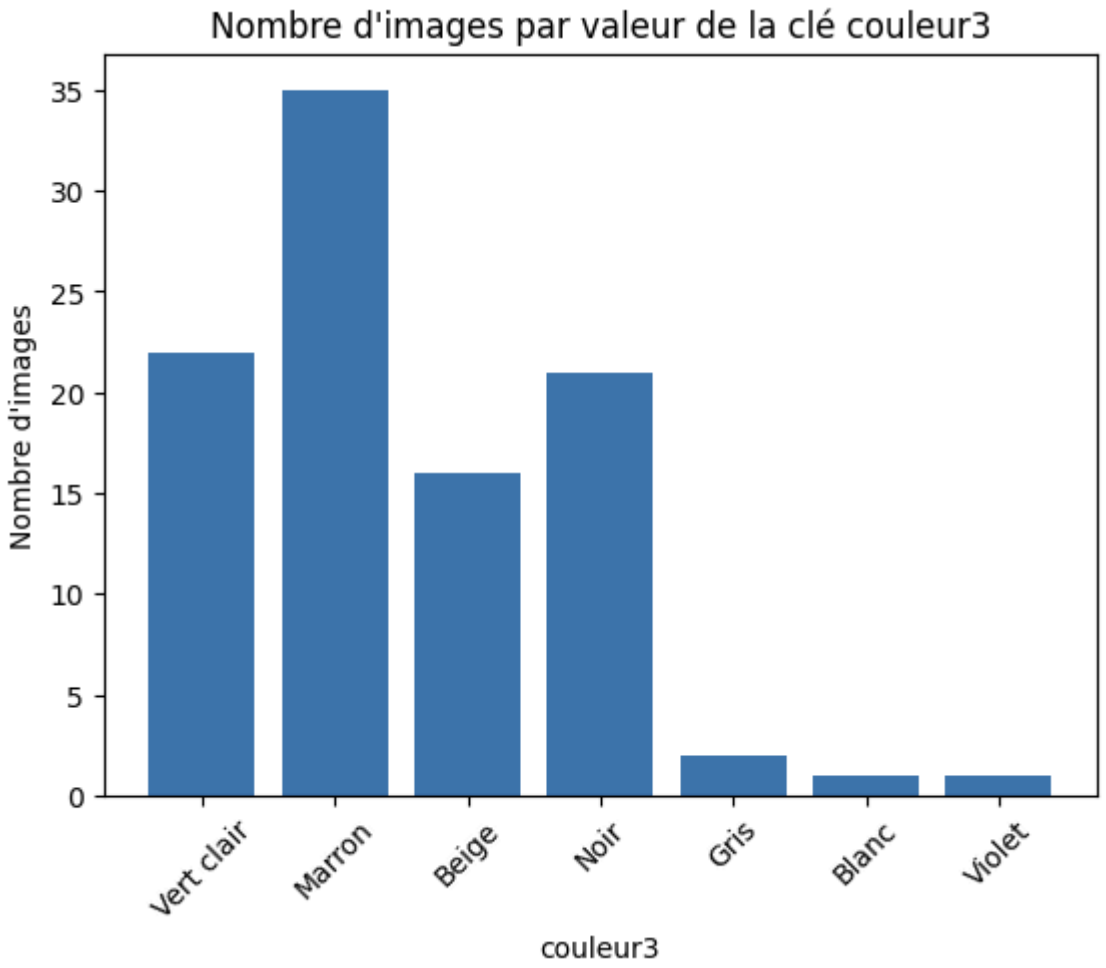
Brugeron Thomas
Condomines Julien



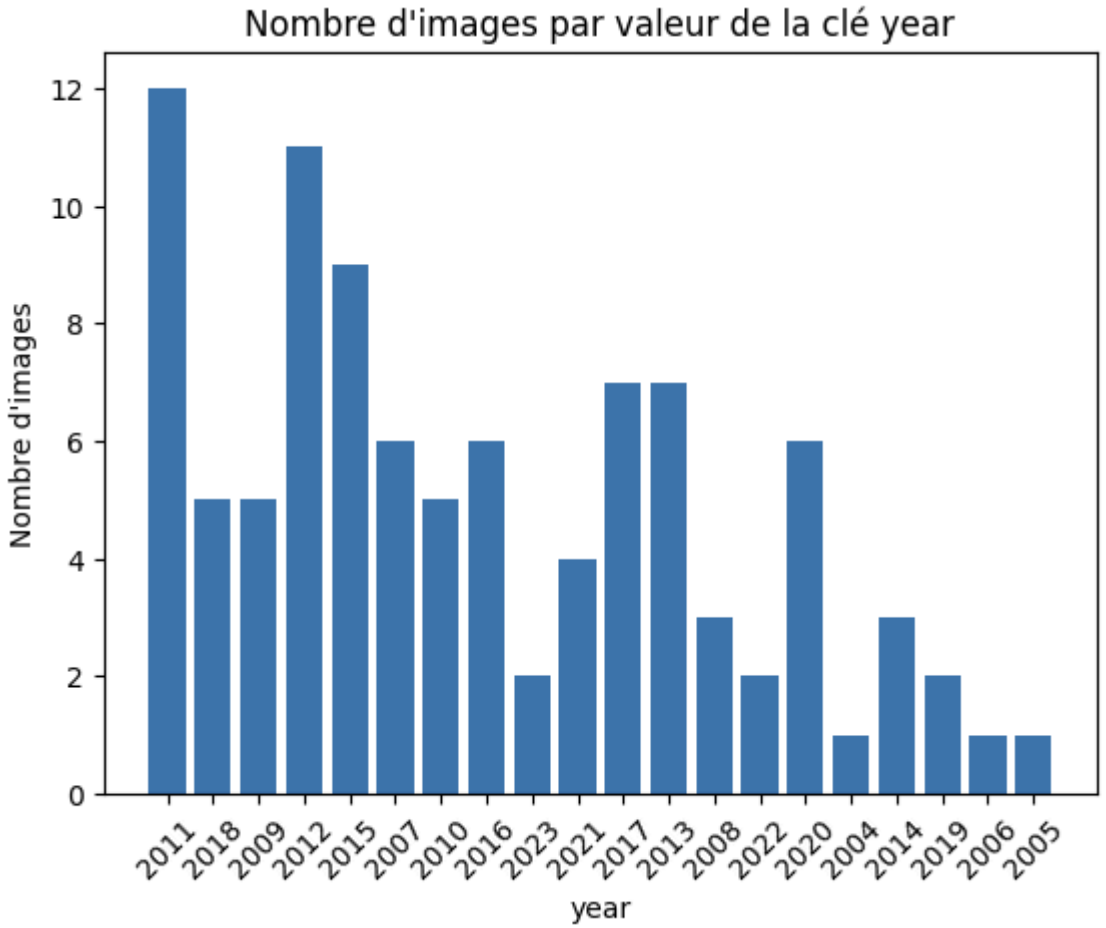
Brugeron Thomas
Condomines Julien



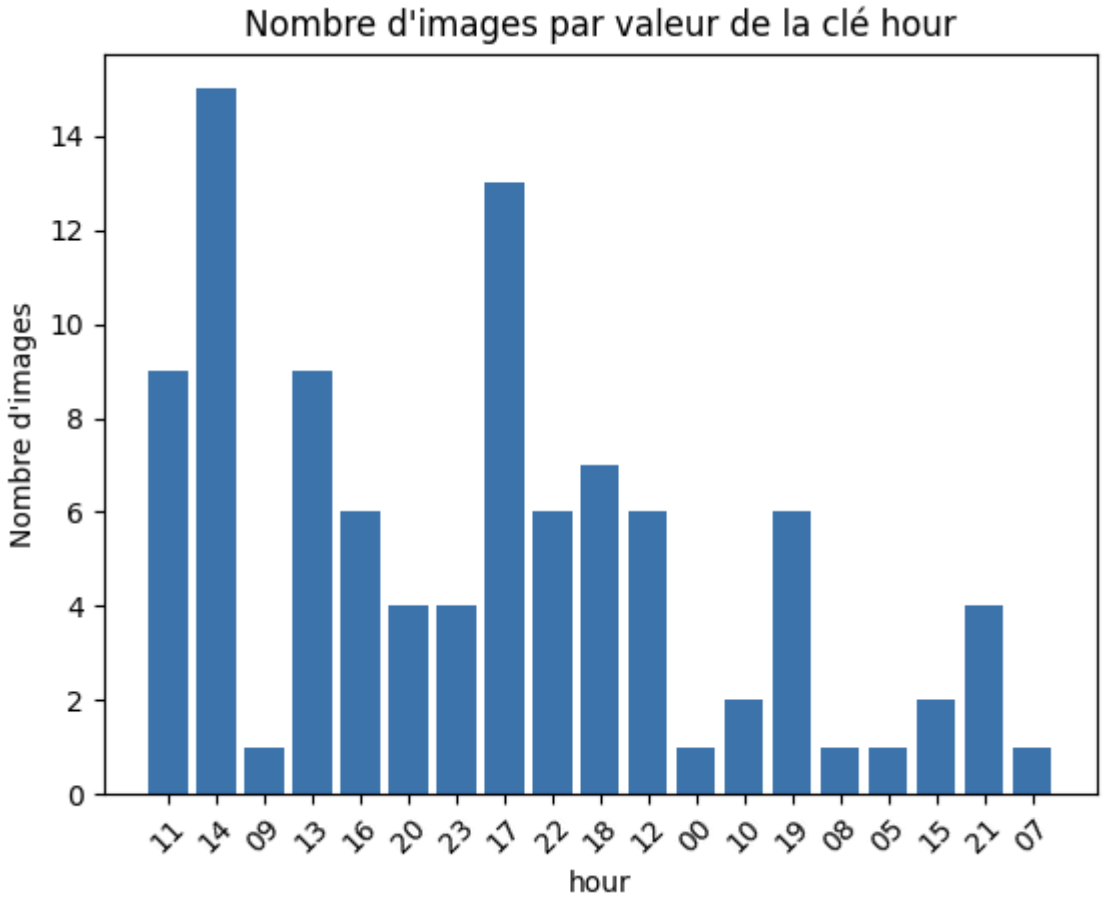
Brugeron Thomas
Condomines Julien

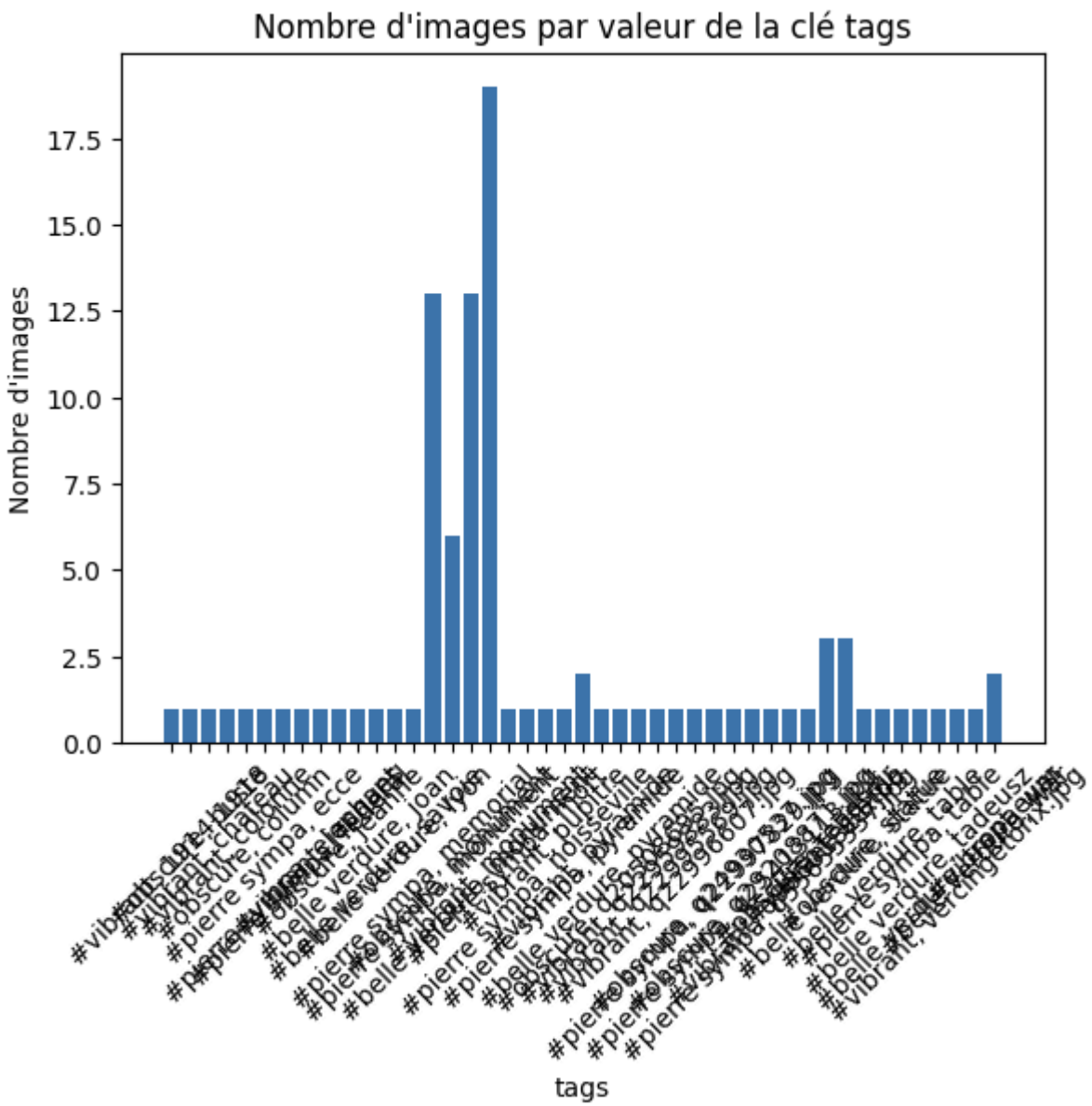


Brugeron Thomas
Condomines Julien

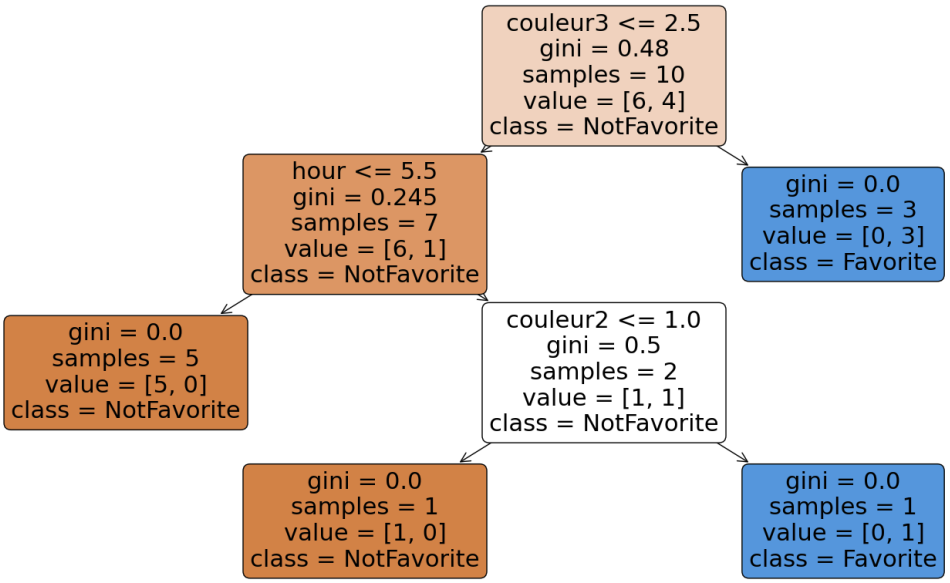


Brugeron Thomas
Condomines Julien



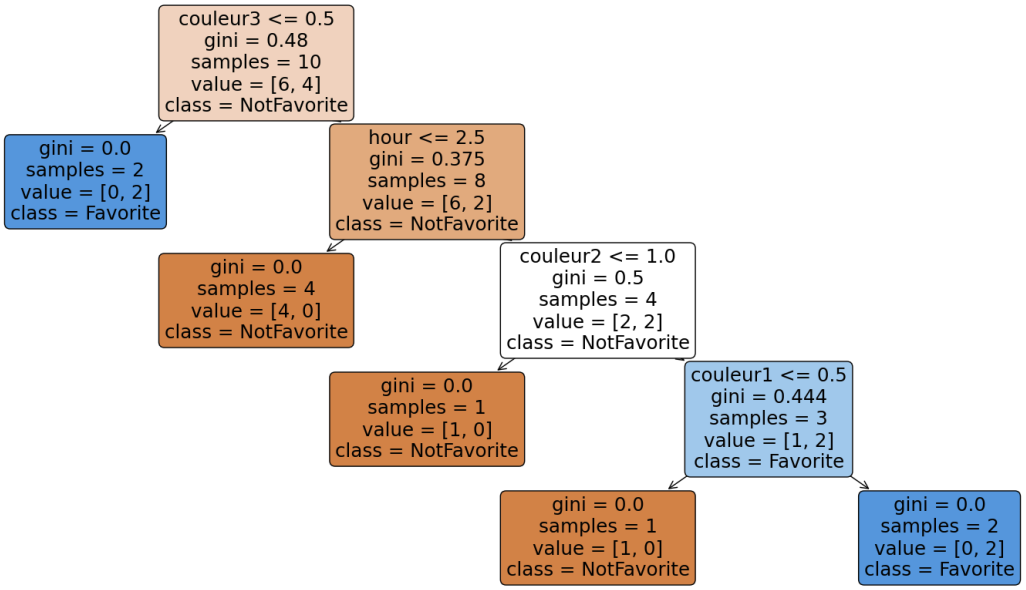


Arbre de décision user 1:



Brugeron Thomas
Condomines Julien 4ICS

Arbre de décision user 2 :



Arbre de décision user 3 :