Rapport de Projet Machine Learning / Data Mining

AUTHOR: Mathieu ERIAU, Florent MONNET

EDITION DATE: 25/03/2022

VERSION: 1.0

REFERENCE: 0

Sommaire

[1 Objectif du Projet 4](#_Toc100006667)

[2 Récupération des images 5](#_Toc100006668)

[3 Etiquetages / Annotation des données 6](#_Toc100006669)

[4 Data Analyses 7](#_Toc100006670)

[5 Création du profil pour un utilisateur 8](#_Toc100006671)

[6 Apprentissage et prédictions 9](#_Toc100006672)

[7 Auto-évaluation 10](#_Toc100006673)

# Objectif du Projet

L’objectif de ce Projet est de recommander des images en fonction des préférences d’un utilisateur.

On va commencer par obtenir des images, puis étiqueter ces dernières et enfin les analyser afin de pouvoir conseiller une image à notre utilisateur qui correspond à son profil.

# Récupération des images

On souhaite avoir un set d’images que l’on pourra utiliser pour réaliser notre Projet.

Pour ce faire nous avons utilisé le site suivant : <https://www.kaggle.com/>.

Nous avons décidé de prendre le dataset suivant : <https://www.kaggle.com/datasets/nielspace/pexels-mountain-images>.

Le code ci-dessous nous permet de récupérer les images de cet espace et des les stocker dans un dossier images :

Une image contenant texte

Description générée automatiquement

Figure : Récupération des images

Ainsi nous avons 101 images de montagnes pour faire nos tests.

# Etiquetages / Annotation des données

Une fois les données obtenues on a besoin d’avoir des données en lien avec ces images.

Dans notre cas avec des images de montagne on se rend déjà compte que le choix n’est pas optimal puisqu’il est plus complexe d’imaginer une variable intéressante à exploiter dans le cadre de notre Projet. Avec des photos de chat et chien par exemple on aurait pu ajouter un tag sur l’image et travailler sur ce tag par la suite.

Nous avons donc décidé de travailler avec les métadonnées suivantes :

* Les couleurs prédominantes d’une image (via KMEANS)
* L’orientation d’une image (*portrait, landscape…)*
* La taille d’une image
* La date de création d’une image
* Son extension

Et le code ci-dessous remplit automatiquement un fichier JSON qui contiendra donc toutes les métadonnées de nos images :



Figure : Dump des métadonnées de chaque image dans un JSON

La variable ‘*metadata’* est remplit par un autre code trop large pour être renseigné dans ce rapport.

On a aussi ajouté des tags sur chaque image, avec un nombre de like allant de 0 à 100 de manière aléatoire et un hastag ‘moutain’.

# Data Analyses

Une fois les données étiquetés et annotés, nous souhaitons analyser les data et pour cela, une manière efficace de le faire est d’afficher des graphs. Ces graphs peuvent être de différents types (bâton, camembert, etc.).

Ces diagrammes peuvent nous aider à remarquer les valeurs extrêmes de nos dataset et donc voir si nos données sont propres.

Pour cela, on a eu le temps de faire qu’un graph, ce dernier permet de voir l’orientation préféré par un utilisateur (cela dépend du profil utilisé). Dans l’idée, il faudrait montrer bien plus de graph comme une courbe avec la moyenne des bleus, et bien d’autres graphs.

Ci-dessous le code permettant d’afficher le graph des orientations pour chaque profil utilisateur.

Une image contenant texte

Description générée automatiquement

# Création du profil pour un utilisateur

Une fois les données étiquetés et annotés, nous souhaitons maintenant avoir une liste d’image qu’un utilisateur a aimé.

Pour faire cela nous avons créé deux profils :

* Un qui aime les images de manière aléatoire une fois sur cinq. En théorie ça devrait être impossible d’avoir un modèle qui recommande des images pertinentes à cet utilisateur.
* Un qui aime les images qui sont orientés en *landscape*. En théorie, notre modèle devrait pouvoir prédire à presque 100% les bonnes images à recommander à cet utilisateur.

Ci-dessous le code permettant de faire cela :

Une image contenant texte

Description générée automatiquement

Figure : Code permettant la création de profil utilisateur

# Apprentissage et prédictions

Nous voulons avoir un modèle qui recommande une image à un utilisateur en fonction de son profil.

Pour faire cela, nous avons utilisé la méthodologie suivante :

* Split du dataset en deux. Les deux premiers tiers pour l’apprentissage, le dernier tier pour vérifier la véracité des prédictions.
* Création d’un dataframe d’apprentissage avec les deux premiers tiers, et d’un dataframe avec les images animées par l’utilisateur.
* Création du modèle avec les deux dataframes du point précédent.
* Test sur le dernier tier avec le modèle obtenu.

Pour le modèle, on utilise la classe **tree** de sklearn, pour utiliser l’attribut **DecisionTreeClassifier** de ladite classe.

On ne fournit aucun paramètre à la fonction car dans notre cas ou souhaite que le programme trouve **le** critère prédominant**.** Nous n’avons pas fait de combinaison de critères.

Ainsi, dans le cas de nos profils d’utilisateurs on a les résultats suivants :

Une image contenant texte

Description générée automatiquement

Figure : Résultat tree.DecisionTreeClassifier

C’est un résultat cohérent, comme le nombre d’images pour tester est assez faible, dans le cas aléatoire on aura toujours une accuracy proche des 50%, et dans le cas de notre utilisateur qui a aimé toutes les images orientées en *landscape*, comme il n’y a pas de variation dans ses choix on obtient une accuracy de 100%.

On voit de plus que dans le cas rand, le modèle ne sait pas quel critère est prédominant, alors que dans le cas orientation on a un critère qui domine, qui est bien sûr celui qui représente l’orientation de l’image.

# Auto-évaluation

On a donc obtenu un modèle cohérent, mais nous ne sommes pas allez bien loin.

On aurait pu imaginer faire les choses suivantes :

* Prendre un dataset plus large, avec plus de données
* Faire des ‘pièges’ dans les profils, et observer le comportement du modèle
* Faire des modèles qui cherchent des combinaisons de critères
* Utiliser d’autres modèles que la classe tree de sklearn
* …

TODO