**Compte Rendu**

**Introduction:**

Le principal objectif de ce projet était de créer un système de recommandations en fonction des préférences des utilisateurs attribuées aléatoirement (Likes, Tags…) sur des images de chiens et de chats, que nous avions précédemment sélectionnées. En collectant les données de nos images et de nos utilisateurs, nous avons obtenu les couleurs ainsi que les animaux de compagnie préférés par chaque utilisateur.

**Données des images utilisées:**

- Sources des images et de leur licence:

Nous avons utilisé 500 images durant notre projet (250 chats + 250 chiens).

Toutes ces images ont été trouvées sur Kaggle. Il s’agissait d’images au format PNG, libres de droit, avec une licence CC0: Public Domain ce qui signifie qu’on avait le droit d’utiliser et de modifier ces images comme on le souhaitait gratuitement.

- Informations que nous avons décidé de stocker pour chaque image.

Dans **data.json,** nous avons décidé de stocker, dans un premier temps, la classe de chaque image (chat, chien).

Nous avons également stocké les informations relatives au format, à la taille (toutes nos images avaient une même taille de 1000\*1000 pixels) et le mode d’encodage des couleurs de nos différentes images (ex: RGBA…). Nous avons aussi récupéré l’ID de nos différentes images, même si nos images étaient initialement numérotées de 0 à 499, nous avons choisi de numéroter nos IDs de 0 à 249 pour tous les animaux, il y a donc 2 fois le même ID stocké, mais pour des animaux différents (il y a deux ID 0, un pour le premier chat, l’autre pour un chien, etc…). Ainsi, nous avons un ID unique pour une classe d’animal (0 à 249), mais aussi un identifiant pour la totalité de nos images, qui sont les clefs (de 0 à 499) de notre dictionnaire.

Enfin, afin d’avoir des données sur les couleurs des images, nous avons décidé de calculer les deux couleurs prédominantes de celles-ci. Pour ce faire, nous avons utilisé l’algorithme de regroupement **MiniBatchKMeans.** Nous avons préféré utiliser celui-ci plutôt que le KMeans, pour avoir un gain de rapidité d'exécution. Nous stockons donc le nom des couleurs. Pour pouvoir déterminer les noms exacts des couleurs étudiées, nous avons dû utiliser une fonction **convert\_rgb\_to\_names(rg\_tuple)** pour obtenir les noms exacts associés au code RGB des couleurs prédominantes. De même, nous nous sommes servis de la fonction **hex\_to\_rgb(value)** permettant de faire une conversion de l’hexadécimal au RGB pour pouvoir utiliser la fonction précédente.

En éditant le fichier **data.json**, on se retrouve donc avec toutes nos données souhaitées.

Dans **LikeImages.json,** on a à nouveau stocké l’indice des animaux pour les repérer, cette fois-ci de 0 à 499, ainsi que le nombre total de Likes donnés à chaque image par les utilisateurs.

**Analyse des données utilisateur & recommandation:**

- Informations concernant les préférences de l'utilisateur

Pour pouvoir travailler sur nos utilisateurs, nous avons commencé par créer un nombre aléatoire d’utilisateurs fictifs (le nombre d’utilisateurs est compris entre 100 et 1000).

Nous avons donné à chaque utilisateur la possibilité de Liker ou de taguer chaque image (l’utilisateur pouvait taguer chat, chien ou des couleurs sur chaque image). L’étiquetage de chaque image par un utilisateur se faisait avec une certaine probabilité de succès, les Tags et les Likes étaient ainsi attribués automatiquement par les utilisateurs.

L’utilisateur n’avait la possibilité de taguer qu’uniquement l’une des deux couleurs prédominantes d’une image (l’utilisateur tague l’une des deux couleurs ou les deux selon une certaine probabilité). De plus, l’utilisateur ne pourra mettre le Tag “chien” uniquement sur des images de chiens, il en va de même pour les chats.

Les Likes, ainsi que les Tags de chaque utilisateur sont ensuite stockés dans le fichier **LikeUsers.json**. Pour les Likes ainsi que pour les Tags de chiens et de chats, on a juste stocké l’ID de l’image étiquetée par l’utilisateur. Pour les Tags de couleur, on conserve un dictionnaire où la clé est l’ID de l’image taguée et la valeur associée contient le nom des couleurs taguées sur l’image.

- Les modèles d'exploration de données et/ou d'apprentissage machine que nous avons utilisés avec les métriques obtenues.

Pour analyser les données précédemment obtenues, nous avons mis en relation l’ID des images précédemment Likées avec les caractéristiques des images correspondantes (couleurs, chat, chien). En s'intéressant ainsi aux caractéristiques de toutes les images Likées par chaque utilisateur, on a pu déterminer les couleurs et les animaux favoris de chaque utilisateur, en comparant par exemple le nombre de chats likés vis-à-vis du nombre de chiens qui ont été likés par un utilisateur.

Nous avons par la suite mis en évidence les préférences d’un utilisateur choisi au hasard en affichant un diagramme camembert représentant ses couleurs préférées (chaque couleur affichée sur le diagramme correspondant au code RGB de chaque couleur représentée) ainsi qu’un second diagramme affichant sa préférence entre les chiens et les chats.

Nous avons aussi utilisé les données du fichier **LikeImages.json,** pour afficher le graphique du nombre total de Likes reçus par image. Sur ce graphe, nous pouvons observer qu’il y a une légère dispersion dans nos valeurs du nombre de Likes attribués par image, ce qui est tout à fait normal, l’attribution d’un Like par un utilisateur étant parfaitement aléatoire.

Le diagramme et le graphique ont été affichés via l’utilisation de **matplotlib**.

En ce qui concerne le système de recommandations, nous allons utiliser un arbre de décisions comme classificateur. Nous avons donc commencé par classer nos images entre les images Likées par l’utilisateur d’un côté, auxquelles nous avons rajouté le Tag ‘Favoris’ et les images non Likées qu’on a tagué par ‘NonFavoris’, pour ainsi réaliser l’apprentissage de notre algorithme. Nous avons gardé les images ‘NonFavoris’ de côté pour pouvoir les utiliser dans un système de prédiction qui utilise les données des images déjà Likées (classe, couleur des animaux) par l’utilisateur pour prédire quelles images parmi les images non Likées par l’utilisateur seraient susceptibles de lui plaire. Ensuite, parmi les images non Likées mais recommandées à l’utilisateur, on en affiche une au hasard avec sa clef dans le dictionnaire **data.json**.

Enfin, nous avons codé un test permettant, pour un utilisateur donné, de vérifier la concordance entre les couleurs prédominantes de l’image proposée par le système de recommandations avec les couleurs préférées par l’utilisateur. Pour cela, on a affiché, les données liées à l’utilisateur et celles liées à l’image, ainsi qu'un graphique détaillant les couleurs préférées par l’utilisateur. On peut ainsi constater que notre système de recommandations fonctionne bien comme prévu.

**Potentielles améliorations & Conclusion:**

- l'auto-évaluation de notre travail:

Nous n’avons pas eu le temps de totalement finaliser notre travail, avec un peu plus de temps, nous aurions sans doute essayer de diversifier un peu les images utilisées, parce qu’elles avaient toutes le même format et la même taille, ce qui ne nous permettait pas d’analyser l’influence de la taille de l’image sur les préférences des utilisateurs.

Ensuite, nous aurions pu mettre plus de tests dans notre code, pour vérifier qu’on obtienne bien le résultat attendu, mais nous avons estimé que l’ajout systématique de captures d’écran de résultats souhaités permettait de visualiser clairement le résultat attendu pour chaque code.

Tel que souhaité au début du projet, nous sommes bien parvenus à mettre en place un système de recommandations qui va permettre de prévoir les images que l’utilisateur va aimer en s’appuyant sur celles qu’il a Likées précédemment.

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

En ce qui concerne les séances pratiques, nous pensons que vous pourriez complexifier le TP1 parce qu’il peut être fait exclusivement par copiers-collers. Vous pourriez certainement le rendre plus difficile en demandant aux étudiants de rajouter un peu de codes aux différents programmes déjà implémentés.

En ce qui concerne le projet, nous pensons que vous devriez expliquer clairement dès le début du module le travail que nous devons réaliser dans le détail, ainsi, on pourrait commencer le projet dès la fin du TP2, ce qui nous permettrait d’aller plus loin dans le projet et de rendre un travail de haute qualité.

**Conclusion:**

Pour conclure, nous pouvons dire que ce projet nous a permis de découvrir les outils nécessaires pour faire du Data Mining. Ce projet nous a également sensibilisé à l’utilisation de librairies majeures de Python, telles que pandas, matplotlib ou encore numpy.

Lors de ce projet, nous avons travaillé sur les principales étapes de la création d’un système de recommandations: Tout d’abord, nous avons collecté et stocké les données de nos images, puis nous avons créé des utilisateurs pour qu’ils annotent nos images (Likes, Tags) pour ensuite pouvoir analyser les préférences de nos utilisateurs pour qu’on puisse leur faire des recommandations en recherchant les images qui correspondent le mieux à leurs préférences. Nous avons aussi tracé des diagrammes et des graphiques permettant de visualiser plus facilement les données rassemblées.

**Webographie:**

- Source des images:

<https://www.kaggle.com/datasets/akrsnv/catdog>

- Cours et TP

- Pour le problèmes des couleurs:

<https://medium.com/codex/rgb-to-color-names-in-python-the-robust-way-ec4a9d97a01f>

<https://stackoverflow.com/questions/65225262/python-webcolors-no-defined-color-name>

- Conversion hex to rgb:

<https://stackoverflow.com/questions/29643352/converting-hex-to-rgb-value-in-python>