

Data Mining: Système de recommandation d'images

But du projet

Le but de notre projet est de concevoir et mettre en place un **système de recommandation** d'images en fonction des préférences de notre utilisateur. Nous avons essayé de respecter au mieux les étapes principales du cycle de vie des données vues en cours, consistant en la:

- Collecte de données
- Étiquetage et annotation
- Analyses de données
- Visualisation des données
- Système de recommandation

Collecte de données: Source des données

Pour faire au mieux ce projet, nous avons eu recours à **kaggle**, une plateforme qui donne accès à un vaste référentiel de données et de code publiés par la communauté.

Notre dataset comportant des images de chats, chiens et singes ainsi que des informations associées dans un fichier csv, le traitement qui a suivi n'était pas très compliqué.

Le dataset contient 2 dossiers d'images:

- Train ⇒ c'est lui qu'on a utilisé, il contenait 1309 éléments.
- Test.

Nettoyage des données

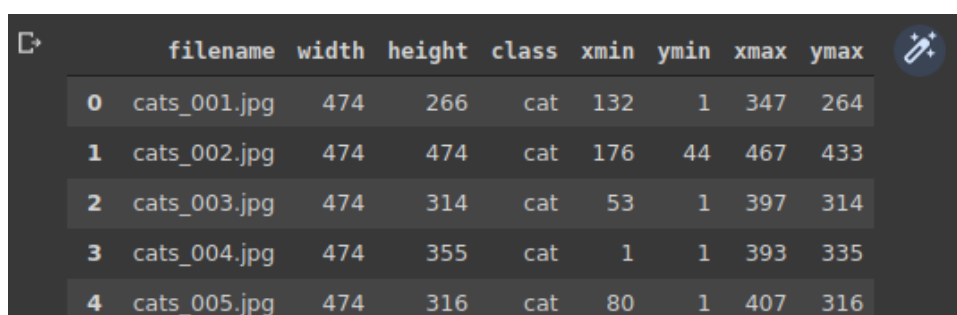
Avant de pouvoir utiliser le dataset téléchargé, nous avons d'abord filtré les informations utiles, nous avons notamment relevé les points suivants:

- Le dataset contient des fichiers ".xml" (un fichier associé à chaque image): nous avons fait attention dans nos boucles "for" de n'inclure que les fichiers "*.jpg".
- Le dataset contient des doublons: nous avons donc appliqué la méthode **.drop_duplicates()** au fichier .csv avant de le convertir en fichier json et créer le dataframe.
- Après nettoyage, nous nous sommes retrouvés avec **469 images à traiter** dans le dossier train.

Étiquetage et annotation Informations qu'on a décidé de stocker

Après avoir récupéré notre dataset de Kaggle, nous avons étudié nos images et nous sommes passées par 2 étapes principales: La transformation, ainsi que l'étiquetage et l'annotation.

Notre fichier .csv contenait les informations suivantes:



| | filename | width | height | class | xmin | ymin | xmax | ymax |
|---|--------------|-------|--------|-------|------|------|------|------|
| 0 | cats_001.jpg | 474 | 266 | cat | 132 | 1 | 347 | 264 |
| 1 | cats_002.jpg | 474 | 474 | cat | 176 | 44 | 467 | 433 |
| 2 | cats_003.jpg | 474 | 314 | cat | 53 | 1 | 397 | 314 |
| 3 | cats_004.jpg | 474 | 355 | cat | 1 | 1 | 393 | 335 |
| 4 | cats_005.jpg | 474 | 316 | cat | 80 | 1 | 407 | 316 |

Nous avons donc 8 informations clés:

- **filename**: le nom de l'image, qui servait l'identifiant.
- **width**: la largeur de notre image.
- **height**: la longueur de notre image.
- **class**: décrit le contenu de l'image, peut prendre la valeur "cats", "dogs", "monkeys".
- **xmin, ymin, xmax, ymax**: les coordonnées du bounding box.

Après réflexion, nous avons décidé de garder uniquement les informations qui nous intéressent:

- **width**
- **height**
- **class**

Et nous avons décidé d'effectuer des traitements et transformations pour récupérer 2 autres informations clés:

- Transformation: La taille de l'objet (l'animal) à l'intérieur de l'image.
 - Pour ce faire, nous avons utilisé les bounding box coordonnées présentes dans notre fichier .csv de base, nous avons donc:
 - **tailleX** = xmax - xmin.
 - **tailleY** = ymax - ymin.
 - Ce traitement nous a donné énormément de valeurs distinctes, et on s'est rendu compte que celles ci seraient compliquées à gérer avec le label encoder, nous avons donc trouvé ça judicieux de diviser les tailles X et Y en 3 catégories :
 - taille de l'objet inférieure à 30% de la taille de l'image: small ⇒ valeur 0 affectée.
 - taille de l'objet entre 30% et 70% de la taille de l'image: médium ⇒ valeur 1 affectée.
 - taille de l'objet supérieure à 70% de la taille de l'image: large ⇒ valeur 2 affectée.

| xmin | ymin | xmax | ymax | | tailleX | tailleY |
|------|------|------|------|--|---------|---------|
| 132 | 1 | 347 | 264 | | 1 | 1 |
| 176 | 44 | 467 | 433 | | 1 | 1 |
| 53 | 1 | 397 | 314 | | 2 | 1 |
| 1 | 1 | 393 | 335 | | 2 | 1 |
| 80 | 1 | 407 | 316 | | 1 | 1 |
| ... | ... | ... | ... | | ... | ... |
| 52 | 132 | 107 | 297 | | 0 | 1 |
| 1 | 234 | 104 | 322 | | 0 | 0 |
| 46 | 18 | 130 | 201 | | 0 | 1 |
| 23 | 109 | 141 | 296 | | 0 | 1 |
| 18 | 9 | 302 | 266 | | 1 | 1 |

- Traitement: La couleur dominante.
 - Pour effectuer ce traitement, nous avons utilisé l'algorithme de clustering **Mini Batch K-means**, avec 3 clusters.
 - Nous avons importé des fonctions de conversions from webcolors, notamment `css3_hex_to_names` et `hex_to_rgb` pour rendre les couleurs obtenues lisibles.
 - Nous avons donc récupéré dans un tableau la **couleur dominante** de chaque image appartenant au training dataset.

```
couleur1
dimgray
silver
dimgray
dimgray
lightgray
...
gainsboro
sienna
silver
darkslategray
silver
```

Après avoir tout traité, nous avons récupéré toutes les informations qui nous intéressent, et nous avons mis ça dans un dataframe que nous allons utiliser pour les étapes qui suivent.

Avant de continuer notre analyse, nous avons annoté nos données à l'aide du **LabelEncoder** importé depuis **sklearn.preprocessing**.

Le **dataframe** contenait donc au final:

| width | height | class | tailleX | tailleY | couleur1 |
|-------|--------|-------|---------|---------|----------|
|-------|--------|-------|---------|---------|----------|

Analyses de données: informations concernant les préférences de l'utilisateur

Pour simuler un utilisateur ayant choisi ses images préférées, nous avons généré une colonne « likes » contenant soit “favorite” ou “not favorite”, qu'on a par la suite insérée dans notre **resultframe**. Cette colonne a été créée en générant aléatoirement un entier 0 ou 1, si l'entier est égal à 1 alors on rajoute « favorite » dans la colonne et vice-versa.

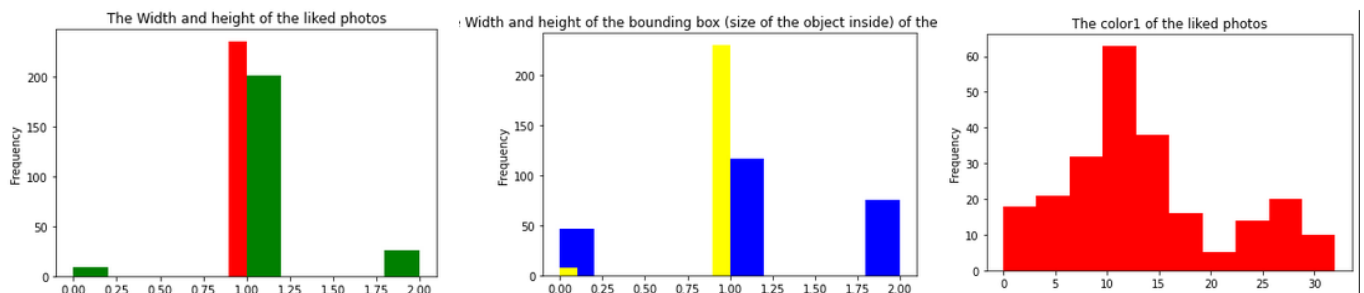
```
likes
0    favorite
1    favorite
2  not favorite
3    favorite
4    favorite
...
464 not favorite
465 not favorite
466 not favorite
467  favorite
468 not favorite

[469 rows x 1 columns]
```

Visualisation de données

Nous avons essayé de visualiser les informations les plus pertinentes; pour ce faire, nous avons mis en place:

- Un plot.bar pour visualiser les 2 propriétés “width” et “height”:
- Un plot.bar pour visualiser les 2 propriétés “tailleX” et “tailleY”:
- Un plot.bar pour visualiser la couleur dominante qui revient le plus souvent.



- Dans cet exemple, l'utilisateur a tendance à liker les images de width **moyen** et de height **moyen**.
- Il a tendance à liker les objets **moyennement larges** (tailleX) ainsi que les objets **soit moyens soit trop grands en hauteur** (tailleY).
- L'utilisateur aime bien les images avec la couleur dominante de code 12.

Système de recommandation

Cette dernière étape consiste à satisfaire l'objectif initial de ce projet, qui est de pouvoir recommander à des utilisateurs des images qu'ils seront en mesure d'apprécier et ça en se basant sur le traitement déjà effectué et surtout sur leurs préférences. Pour ce faire, nous avons décidé de travailler avec les arbres de décisions.

En effet, les arbres de décision sont une méthode d'apprentissage supervisé non paramétrique utilisée pour la classification et la régression. L'objectif est de créer un modèle qui prédit la valeur d'une variable cible en apprenant des règles de décision simples déduites des caractéristiques des données.

Dans notre cas, les arbres de décision apprennent à partir des données pour approximer l'ensemble des données à proposer à un utilisateur. Plus l'arbre est profond, plus les règles de décision sont complexes et plus le modèle est adapté.

Pourquoi on a choisi de travailler avec cette méthode ?

- Simple à comprendre et à interpréter. Les arbres peuvent être visualisés.
- Nécessite peu de préparation des données. D'autres techniques nécessitent souvent une normalisation des données, des variables fictives doivent être créées et des valeurs vides doivent être supprimées.
- Capable de gérer les problèmes multi-sorties.

Mise en oeuvre de cette méthode au sein du projet

On a commencé tout d'abord par importer DecisionTreeClassifier avec ces deux lignes :

```
from sklearn import tree
```

```
Xxx = tree.DecisionTreeClassifier()
```

On a ensuite appliqué ce classificateur sur nos données avec `xxxx = xxx.fit()`

L'auto-évaluation du travail :

Nous n'avons malheureusement pas eu assez de temps pour bien élaborer la partie test, en effet on voulait tester notre système sur un nouveau dataset avec de nouvelles images mais le traitement allait prendre beaucoup de temps.

Cependant notre travail a des limites et des contraintes, et peut s'améliorer, on propose notamment:

- Ajouter plus de caractéristiques à prendre en compte.
- Tester et simuler le système pour plus d'utilisateurs.
- Évaluer notre système avec une matrice de confusion et calculer sa précision.

Conclusion:

Ce projet nous a permis de bien comprendre et d'appliquer les tâches clés du data mining. Nous sommes désormais capables de faire la différence entre chaque étape et de faire le lien entre la théorie vue en cours et la pratique.

Nous sommes aussi en mesure d'identifier les limites de notre projet et de proposer des solutions pour l'optimiser.

Ce projet nous a aussi permis de comprendre la réalité derrière les systèmes de recommandations utilisés surtout dans les réseaux sociaux.