

Le tableau contient les liens vers les images choisis puis j'ai parcourus ces images grâce a la bibliothèque b4s avec la méthode « `urllib.request.urlretrieve(url,full_path)` » et j'ai télécharger toute les images dans ces liens dans un fichier images qui est générer lui aussi automatiquement grâce a la bibliothèque OS ,

Et enfin j'ai enregistrer les donnes de chaque image grâce a la méthode `exif` qui donne les information sur l'image , mais ensuite il s'est avérer que le tas d'information que j'enregistrer n'ai pas nécessairement important pour l'entraînement ni pour l'étiquetage des photos vu que chaque photo avec des information complètement différente et avec des noms différents comme par exemple la hauteur de l'image avait pour nom `exifHeight` et pour d'autre `Height` etc.. et d'autre information inutile comme la date de la prise de l'image ou bien le type de camera j'ai préférer utiliser d'autre méthode pour l'étiquetage de mes images qu'on verra par la suite .

Étiquetage et annotation

Dans cette taches on devait étiqueter, annoter et enregistrer des informations sur chaque image pour ensuite l'analyser en utilisant des algorithmes de regroupement pour trouver les couleurs dominantes ,

Pour l'étiquetage j'ai choisie 4 information a mettre en valeur, et qui sont la taille de l'image, la forme de l'image (portrait ou landscape), la couleur dominante et enfin un tag pour chaque photo

- Tout d'abord pour la taille et la forme de l'image j'ai trouver que le plus simple serai d'utiliser la bibliothèque PIL et de n'utiliser que deux données qui seront la hauteur et la largeur de l'image

Puis je déduisais si la largeur est plus grande que la hauteur alors l'image est en mode Landscape si non elle est en mode portrait puis pour la taille je prenais la surface de l'image qui sera égale à la largeur x la hauteur

```
image=Image.open(path)
x,y=image.size
if y>x:
    s.append('portrait')
else:
    s.append('landscape')
s.append(x*y)
```

- Ensuite pour la couleur dominante j'ai utiliser l'algorithme vue dans le tp 2 qui nous permet de trouver les couleurs dans une image grâce a la méthode des Kmeans qui donnait en sortie un tuple contenant les valeur rgb de la couleur dominante puis je transformais ces valeurs en une chaine de caractère avec le nom de la couleur la plus proche aux valeur rgb obtenue avec une fonction que j'avais nommé convert_rgb_to_names(rgb_tuple) cette fonction utilise un dictionnaire appelé css3_db contenant les valeurs et les noms correspondant
- Enfin pour les tags je voulais au départ utiliser une méthode automatisée alors j'avais opter pour une API nommée IMAGGA qui me permettait de recevoir un texte en format JSON contenant tout les éléments de mon image et avec un pourcentage de précision par exemple pour ma premiere photo je recevais le texte suivant comme tag :

```
{'result': {'tags': [{'confidence': 59.2787246704102, 'tag': {'en': 'cat'}}, {'confidence': 45.0267524719238, 'tag': {'en': 'pet'}}, {'confidence': 43.8887825012207, 'tag': {'en': 'kitten'}.....
}
```

Donc on voit très bien que cette API a pu distinguer que mon image contenait un chat mais le problème était la taille du texte reçu :



```
print(str)
print(len(str))

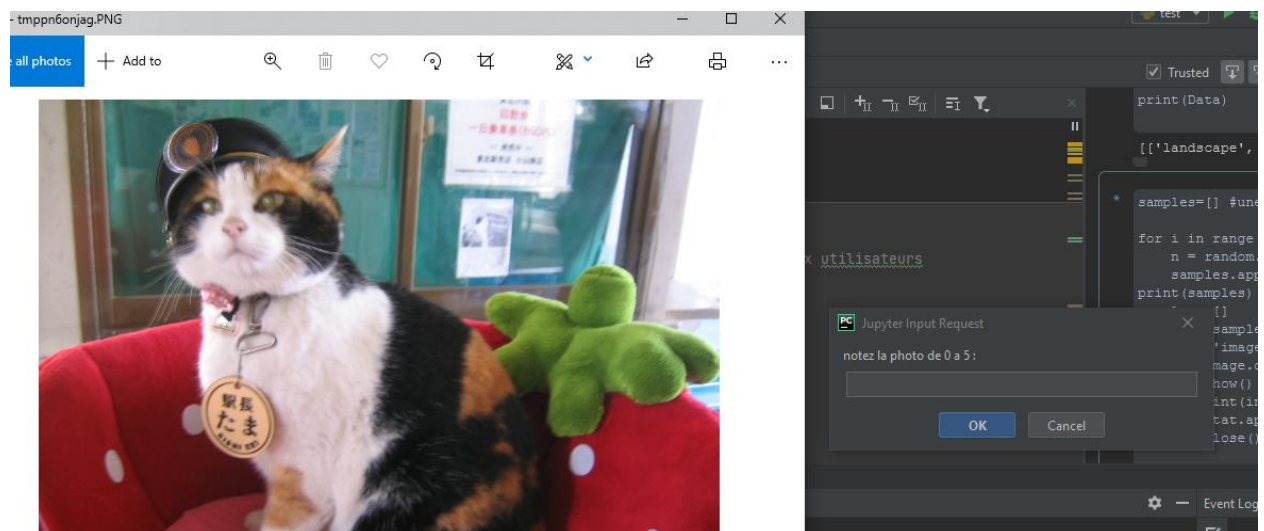
Run: tags
C:\Users\lenovo\PycharmProjects\untitled1\venv\Scripts\python.exe C:/Users/lenovo/PycharmProjects/untitled1/tags.py
{'tags': [{'confidence': 59.2787246704102, 'tag': {'en': 'cat'}}, {'confidence': 45.0267524719238, 'tag': {'en': 'pet'}}, {'confidence': 4353
```

Comme on peut l'apercevoir l'api envoie un texte avec plus de 4000 caractères pour chaque image et donc pour 100 images ça fait un texte énorme avec des centaines de milliers de lignes. Et donc au final j'ai tagué mes photos moi-même avec une boucle for puisque mes photos étaient bien rangées d'une manière à avoir x photos de chat au début puis y photos de sandwich et enfin z photos d'îles, alors j'ai opté finalement pour une liste de tags que j'ai appelée tags.

Analyses de données

Dans cette partie il fallait demander à l'utilisateur de choisir des images et d'ajouter des balises de la manière la plus automatisée possible,

Pour ceci j'ai préféré proposer à l'utilisateur 15 images qui sont choisies arbitrairement du dossier images, ce nombre fait à peu près 20% des images totales du fichier, donc une bonne base de données pour déterminer le goût de l'utilisateur. Ensuite l'utilisateur note chaque image entre 0 et 5, et ceci me facilitera la numérisation des goûts de l'utilisateur et puis prédire une note pour toutes les images dans mon dossier, et toutes les images qui obtiendront une note supérieure à 3 (la moyenne entre 0 et 5) pourront être considérées comme des images favorites ou bien-aimées par l'utilisateur et pourront lui être proposées par la suite :



Les images proposées sont stockées dans la liste samples, puis les notes sont stockées dans la liste samplesData. Ces images et leurs notes seront le modèle d'entraînement de notre système de recommandation de photos.

Système de recommandation

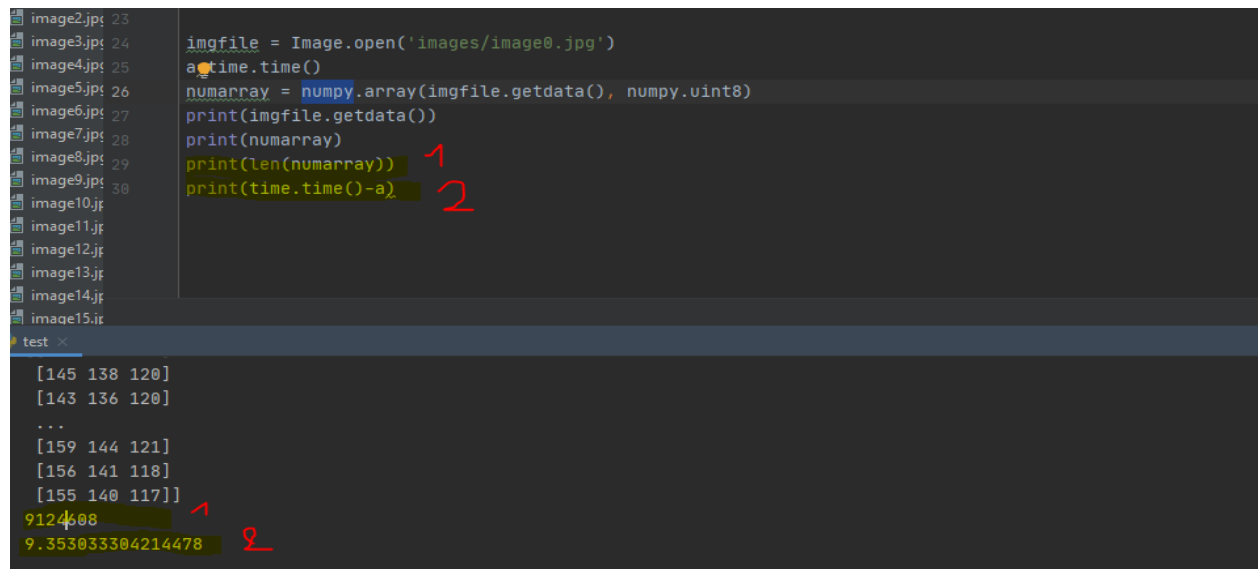
Dans cette partie il a fallut recommander des images a l'utilisateurs grâce a l'echantillon de photo qu'on lui a proposer et ses notes , pour ceci on a utiliser un algorithme vu dans le 3eme tp qui l'algorithme de l'arbre de décision puis on survole toute les images du fichier images et on prédit une note pour chacune

Puis au final j'ai choisit de proposer les images avec une note supérieur à 2.5 pour chaque utilisateurs puisque le 2.5 est la moyenne des notes possibles .

Les limites de mes choix et les difficultés retrouvé:

- **Etiquetage des couleurs :**

Pendant l'étiquetage des couleurs j'ai retrouvé un très grand problème concernant la taille de mes images , quelques une dépassaient les 2MO et donc la machine prenait beaucoup de temp pour analyser ces images et déterminer leur couleur par exemple pour ma premier image qui est d'une taille de 4.69 MB rien que la création d'un tableau avec toute les couleurs des pixels me prenait 10sec et le tableau en sortie avait une taille de 9 124 608 tuplet



```
image2.jpg: 23
image3.jpg: 24
image4.jpg: 25
image5.jpg: 26
image6.jpg: 27
image7.jpg: 28
image8.jpg: 29
image9.jpg: 30
image10.jpg:
image11.jpg:
image12.jpg:
image13.jpg:
image14.jpg:
image15.jpg:

imgfile = Image.open('images/image0.jpg')
a = time.time()
numarray = numpy.array(imgfile.getdata(), numpy.uint8)
print(imgfile.getdata())
print(numarray)
print(len(numarray))
print(time.time()-a)
```

test

```
[145 138 120]
[143 136 120]
...
[159 144 121]
[156 141 118]
[155 140 117]]
9124608
9.353033304214478
```

Et donc des fois quand ca prend trop de temp le compilateur affiche un message d'erreur et ne continue pas le traitement , pour éviter cela j'ai opter pour l'utilisation des try and except et donc j'essaye de trouver la couleur si non j'affiche 'NONE ' ceci me permet d'éviter les messages d'erreurs mais n'aide pas a diminuer le temps d'analyse des images de grande taille . Et au final le temp que prenais ma machine pour analyser la couleurs de toutes les images de mon fichier était de 17 minutes !!

- **la proposition d'échantillon**

pour cette partie j'avais opter pour une proposition aléatoire d'image mais cette méthode a des limites puisque la proposition aléatoire peut ne pas donner une base de donnée solide pour l'entraînement de ma machine , des fois elle ne proposera pas toute les couleurs possibles que j'ai dans le fichier images et d'autre fois elle pourra ne proposer que des photos de mode landscape et non pas des photo en mode portrait et donc la machine considèrera que ces données n'ont pas été traiter et enverra un message d'erreur , pour éviter cela j'ai choisi d'utiliser une autre fois la méthode du try and except et donner la note 0 a toute les images aux quel mon code n'a pas pu prédire de note .

- **le choix des images recommandées**

pour cette partie j'avais choisis de recommander les images ayant obtenu une note de prédiction supérieur a 2.5 au départ puis je me suis rendu compte que les gens ne donne pas des notes de la même manière , des uns sont plus sévère que les autres lors de la notation et donc j'ai changé la valeur de 2.5 à la moyenne des notes attribué par l'utilisateur c'est-à-dire que j'ai proposer les images ayant obtenue une note de prédiction supérieur ou égale à $\text{mean}(\text{samplesData})$, et si la personne n'aime aucune photo alors je choisi de ne rien lui affiché

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

Vous êtes le seul professeur à nous proposer la corrections des tp et ça m'a énormément aider !
merci

Conclusion:

Je trouve que ce projet m'a permis d'avoir une vision global et d'utiliser tout les tp et cours de la matière datamining et de bien comprendre le fonctionnement du machine Learning qui était une ambiguïté pour moi au début et je suis satisfait du résultat final de mon projet .