

PROJET CLASSIFICATION D'IMAGE SATELLITE

PYTHON FOR DATA ANALYSIS

CLÉMENT BERLE
SUZANNE BERTRAND

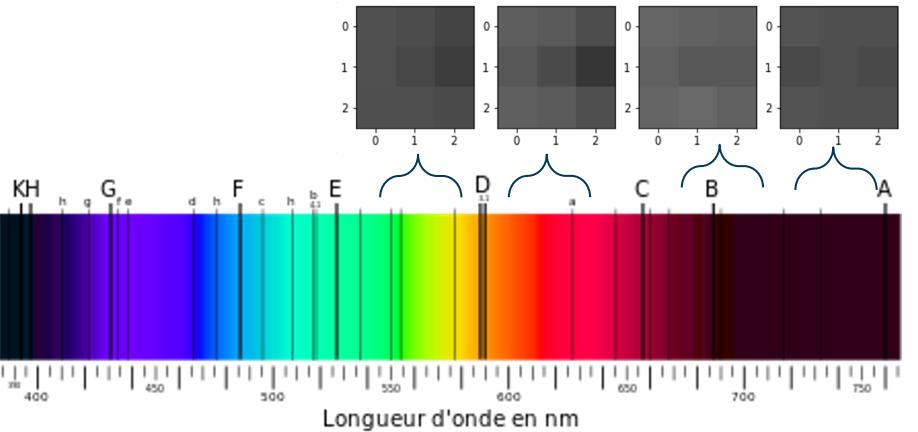


1. PRÉSENTATION DU DATASET

GENERAL

La base de données est faite de valeurs multi-spectrales de taille 3x3 pixels d'images prises d'un satellite

Notre dataset ne contient **que des pixels de niveaux de gris** qui représentent les **valeurs de réflectance** des surfaces au sol pour un intervalle de longueurs d'ondes donné



1. PRÉSENTATION DU DATASET

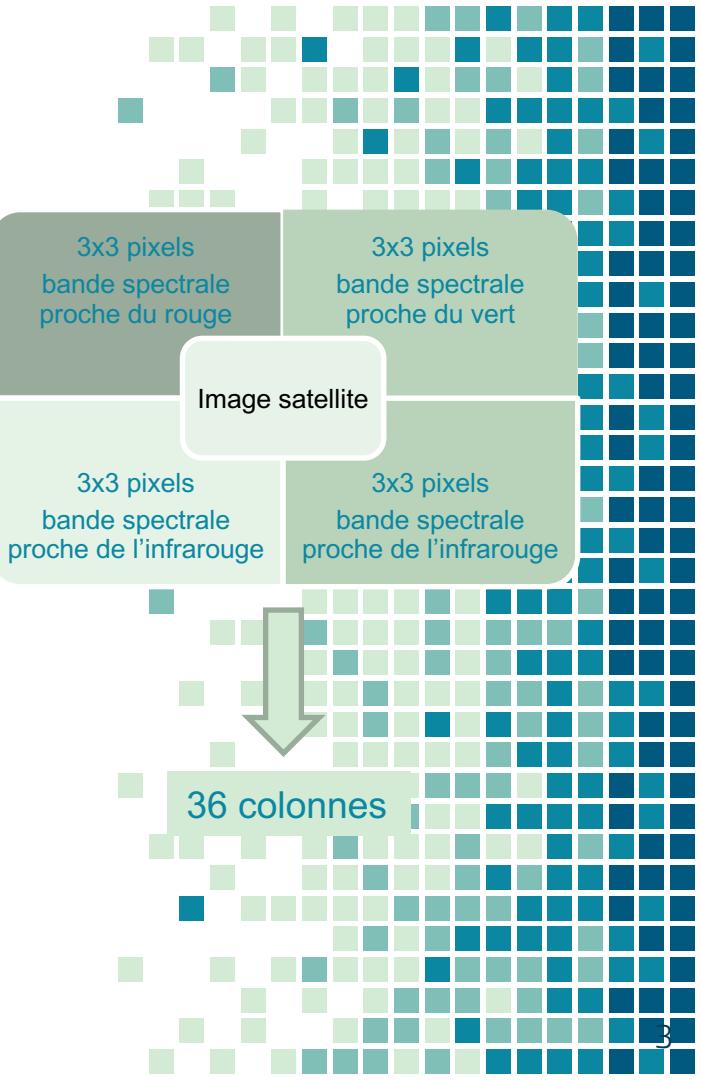
L'UNIQUE VARIABLE PIXEL

Une image correspondant à une ligne est constituée de quatre images numériques de la même scène dans différentes bandes spectrales

	1	2	3	4	5	6	7	8	9	10	11	12
0	84	102	106	79	84	102	102	83	80	102	102	79
1	84	102	102	83	80	102	102	79	84	94	102	79
2	80	102	102	79	84	94	102	79	80	94	98	76
3	84	94	102	79	80	94	98	76	80	102	102	79
4	80	94	98	76	80	102	102	79	76	102	102	79
...
4429	56	64	108	96	64	71	108	96	68	75	108	96
4430	64	71	108	96	68	75	108	96	71	87	108	88
4431	68	75	108	96	71	87	108	88	71	91	100	81
4432	71	87	108	88	71	91	100	81	76	95	108	88
4433	71	91	100	81	76	95	108	88	80	95	104	85

4434 rows x 37 columns

	25	26	27	28	29	30	31	32	33	34	35	36	Y
0	88	121	128	100	84	107	113	87	84	99	104	79	3
1	84	107	113	87	84	99	104	79	84	99	104	79	3
2	84	99	104	79	84	99	104	79	84	103	104	79	3
3	84	99	104	79	84	103	104	79	79	107	109	87	3
4	84	103	104	79	79	107	109	87	79	107	109	87	3
...
4429	63	79	108	92	66	83	108	96	66	87	104	89	5
4430	66	83	108	96	66	87	104	89	63	87	104	89	5
4431	66	87	104	89	63	87	104	89	70	100	104	85	4
4432	63	87	104	89	70	100	104	85	70	91	104	85	4
4433	70	100	104	85	70	91	104	85	63	91	100	81	4



1. PRÉSENTATION DU DATASET

CLASSIFICATION

La dernière colonne Y indique l'étiquette de classification du pixel central

La classification permet de déterminer le type de sol

Ce numéro compris entre 1 et 7 est un code pour les classes suivantes :



1. Terre rouge



2. Culture du coton



3. Sol gris



4. Sol gris humide



5. Sol avec des chaumes de végétation



6. Classe de mélange (tous types présents)

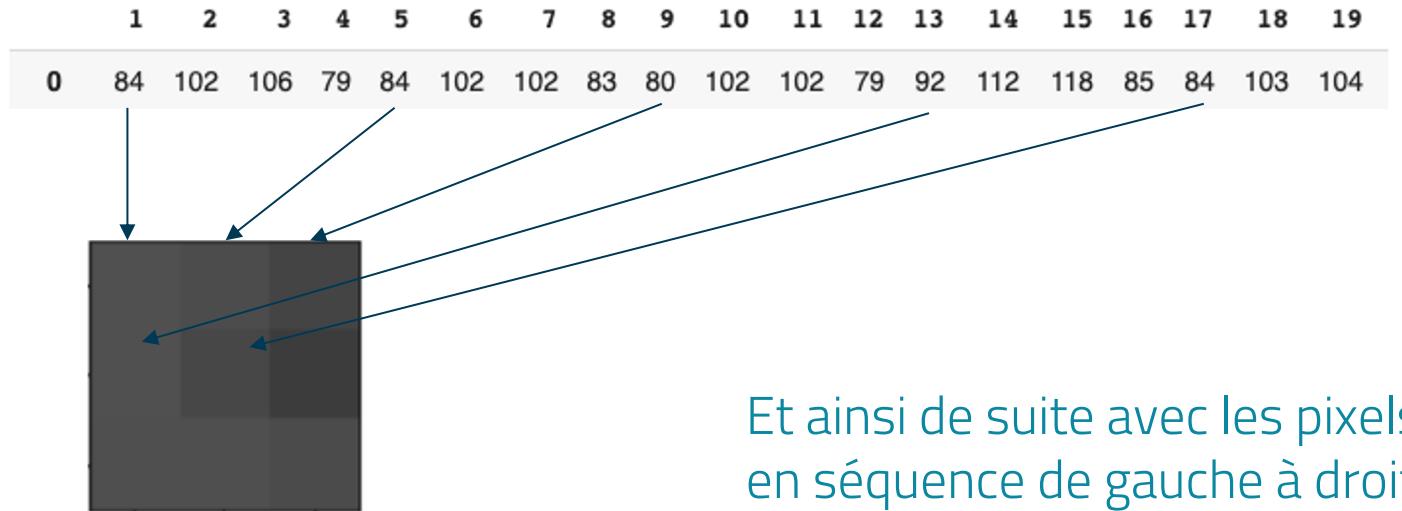


7. Sol gris très humide

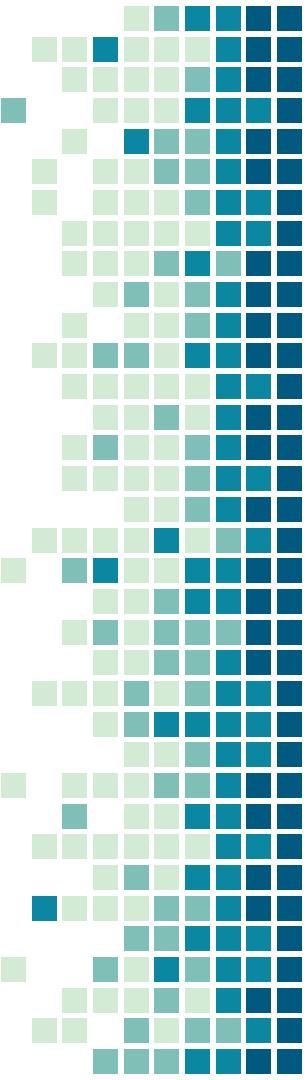
NB : Il n'y a pas d'exemples avec la classe 6 dans cette base de données

1. PRÉSENTATION DU DATASET

Les pixels d'une image correspondant à une bande spectrale sont rangées dans un ordre particulier :



Et ainsi de suite avec les pixels lus en séquence de gauche à droite et de haut en bas ...



2. COMPRÉHENSION DES DONNÉES

La **valeur minimum** d'un pixel sur la totalité des colonnes est de **27** et la **valeur maximum** est de **157**

Les pixels pouvant varier de 0 (noir) à 255 (blanc), on comprend alors que les images satellites représentent les valeurs de réflectance des surfaces **visant** le sol.

Ainsi, pas d'image dans le vide ni d'image trop proche pouvant donner des extrêmes.

	1	2	3
count	4434.000000	4434.000000	4434.000000
mean	69.468877	83.848218	99.318223
std	13.646980	22.722826	16.671154
min	40.000000	27.000000	56.000000
25%	60.000000	71.000000	85.000000
50%	68.000000	87.000000	101.000000
75%	80.000000	103.000000	113.000000
max	104.000000	137.000000	140.000000

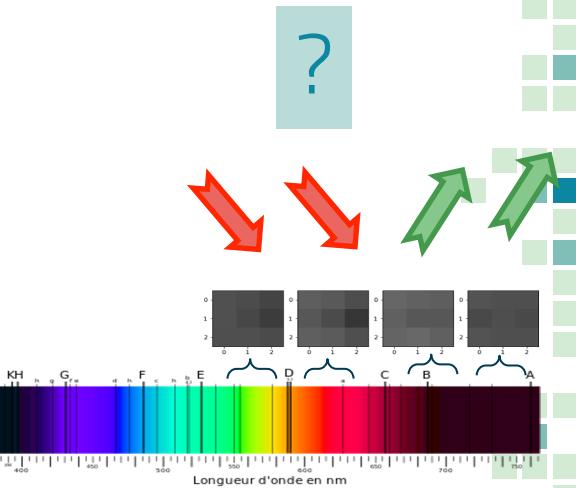
3. VÉRIFICATION DE LA COHÉRENCE DU DATASET

Afin de vérifier la cohérence des données, on se base sur un article scientifique qui stipule :

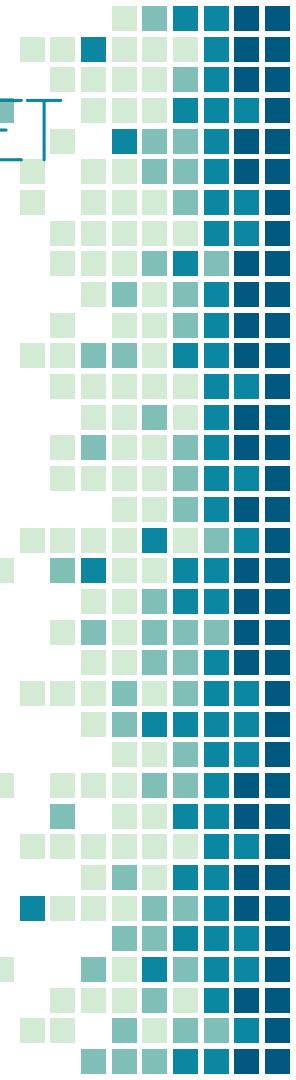
considéré. Dans l'intervalle de longueur d'onde correspondant aux couleurs vertes et du rouges, la végétation a des valeurs de réflectance basses, alors qu'elle a des valeurs élevée dans l'intervalle des longueurs d'onde de l'infrarouge. L'eau a des valeurs

Source : CMS - GéoBretagne

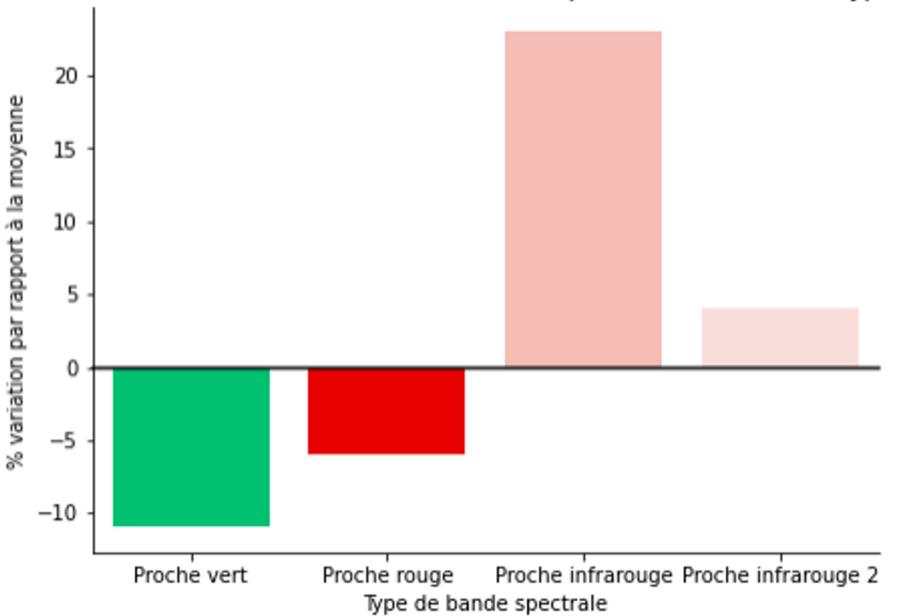
Nous allons donc regarder s'il existe une corrélation entre la classe 5 : Sol avec des chaumes de végétation et les niveaux de réflectance associés



3. VÉRIFICATION DE LA COHÉRENCE DU DATASET



Variation de la reflectance en fonction des bandes spectrales sur des sols type végétation



Il y a bien une cohérence entre le type de sol photographié et sa réflectance

MODÈLES DE PRÉDICTIONS



1. PRÉPARATION DES DONNÉES

Notre dataset était déjà séparé en train_set et test_set

```
print("Nombre de lignes dans le train set : " + str(satellites_images_train.shape[0]))  
print("Nombre de lignes dans le test set : " + str(satellites_images_test.shape[0]))
```

```
Nombre de lignes dans le train set : 4434  
Nombre de lignes dans le test set : 1999
```

On sépare nos valeurs d'entrée et de sortie :

Entrée du modèle (X) : 35 valeurs de pixels

Sortie du modèle (Y) : Type de sol

Nous avons ici un modèle de classification multi classe

2. LES ALGORITHMES DE CLASSIFICATION

Pour un problème de classification multi classe, nous avons choisi d'utiliser les algorithmes de classification suivants :

- Nearest Neighbor
- Keras Tensorflow
- Support Vector Machines
- Naive Bayes Classifier

2. LES ALGORITHMES DE CLASSIFICATION

NEAREST NEIGHBOR (KNN)

Recherche des meilleurs paramètres

```
#Affichage du best score avec les meilleurs paramètres :  
print("score max : " + str(score_max) + " avec n_neighbors = " + str(i))  
  
score max : 0.903951975987994 avec n_neighbors = 35
```

Nous avons 90,4% d'accuracy avec le paramètre n_neighbors

2. LES ALGORITHMES DE CLASSIFICATION

NEAREST NEIGHBOR (KNN)

```
# Determination du pourcentage de certitude
row = 12
class_names = ['red soil', 'cotton crop', 'grey soil', 'damp grey soil', 'soil with \
               very damp grey soil','truc']

predict_proba = neigh.predict_proba(x_test[row-1:row])
print("Pourcentage de certitude pour la ligne " + str(row) + " :")
for i in range(6) :
    print("    Pour " + class_names[i] + " : " + str(predict_proba[0,i]*100) + "%")
```

Pourcentage de certitude pour la ligne 12 :

Pour red soil : 0.0%
Pour cotton crop : 0.0%
Pour grey soil : 0.0%
Pour damp grey soil : 22.857142857142858%
Pour soil with vegetation stubble : 0.0%
Pour very damp grey soil : 77.14285714285715%

2. LES ALGORITHMES DE CLASSIFICATION

TensorFlow

Utilisation de Tensorflow

```
#On entraîne notre modèle tensorflow
model.fit(x_train_normalized, y_train, epochs=100,batch_size=500)

Epoch 100/100
9/9 [=====] - 0s 1ms/step - loss: 0.4696 - accuracy: 0.8307
```

```
# Affichage de l'accuracy de notre modèle avec notre jeu de test
print('Accuracy de notre modèle sur notre jeu de test:', test_acc)
```

Accuracy de notre modèle sur notre jeu de test: 0.8024011850357056

On obtient une accuracy de 80,2%

On observe un léger overfitting entre notre modèle d'entraînement et notre test (83% et 80,2%)

2. LES ALGORITHMES DE CLASSIFICATION

Support Vector Machine

Algorithme de Support Vector Machine

```
SVN = svm.SVC(decision_function_shape='ovo')
SVN_model = SVN.fit(x_train,y_train)
SVN_score = SVN_model.score(x_test,y_test)
print("Score du classifier Support Vector Machine : ", SVN_score)

Score du classifier Support Vector Machine : 0.8859429714857429
```

Nous avons 88,6% d'accuracy avec cet algorithme

2. LES ALGORITHMES DE CLASSIFICATION

Naive Bayes Classifier

Algorithme de Naive Bayes Classifier

```
algo = GaussianNB()
modele = algo.fit(x_train, y_train)
score = modele.score(x_test, y_test)
print("Accuracy de notre model GaussianNB sur notre jeu de test : " + str(score))
```

```
Accuracy de notre model GaussianNB sur notre jeu de test : 0.7963981990995498
```

Nous avons 79,6% d'accuracy avec cet algorithme

3. CONCLUSION SUR LES MODELES

Nearest Neighbor : 90,4% d'accarucy

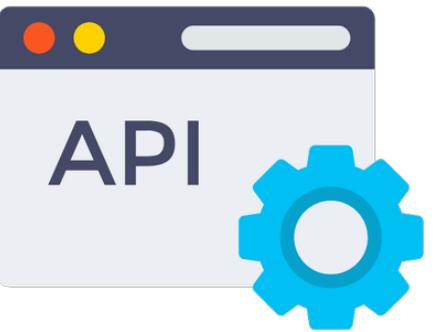
Keras Tensorflow : 80,2% d'accarucy

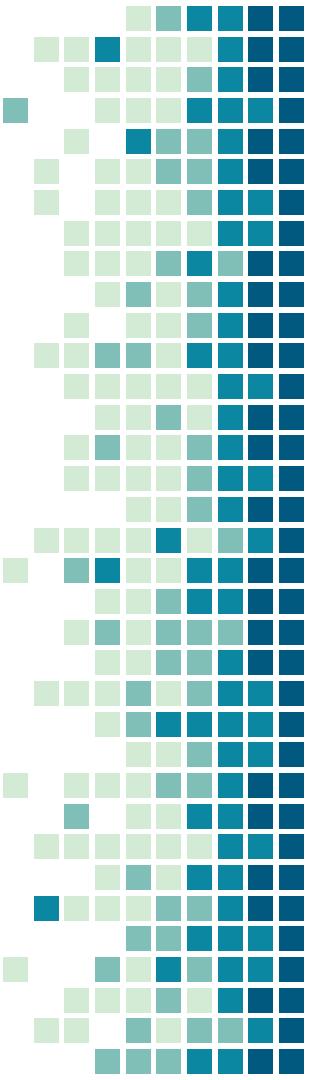
Support Vector Machines : 88,6% d'accarucy

Naive Bayes Classifier : 79,6% d'accarucy

Chacun des algorihtmes possède un bon taux d'accuracy, mais celui qui se démarque le plus est l'algorithme KNN

4. TRANSFORMATION DU MODELE EN API





4. TRANSFORMATION DU MODELE EN API

Lancement de l'environnement Flask :

```
(base) C:\Users\cleme\Desktop\Projet Python>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 223-450-933
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

4. TRANSFORMATION DU MODELE EN API

Envoi de la requête POST :

```
import requests
import json

url = 'http://localhost:5000/api/'

data = [[80, 102, 102, 79, 76, 102, 102, 79, 76, 102, 106, 83, 76, 99, 108, 85, 76, 103,
j_data = json.dumps(data)
headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}
r = requests.post(url, data=j_data, headers=headers)
print(r, r.text)

<Response [200]> "[3]"
```

On a bien une réponse 200 à notre requête, avec comme contenu de la réponse, la classe de notre image.

THANKS!

