Machine Learning and Data Mining II

Water Bodies Classification

Final report

Group Members:
Le Viet Anh
Vu Tuan Phuong Nam
Do Thanh Dat
Bui Quang Ha
Tran Thi Huong Giang
Le Duc Manh
Nguyen Huu Dung

I. Introduction

A challenging problem in geo-based applications is the mechanization of data extraction from topographic maps. This involves the identification of maps to discover several geographical features and their locations.

Unfortunately, solving this problem has never been straightforward as the given information is often insufficient and junk to fully recognize.

One solution to this bottleneck problem can be granted by using machine learning.

To execute machine learning on cracking this difficult problem, there is a must of generous data.

Traditionally, the data itself is often available only on paper maps, which requires progress to convert it to digital format. Luckily, due to the boom of technology, we can easily obtain raw data by utilizing available online sources of maps.

Our project aims to identify water bodies from maps, such as lakes, rivers, etc. so we took abundant screenshots from Google Map and separated them into different categories as we could implement them into our machine learning models.

The system was able to provide a pretty accurate prediction and even better results given the increase in the input data size.

II. Contribution Table

Name	**ID**	**Role**
Le Viet Anh	BI9-035	Main coder
Vu Tuan Phuong Nam	BI9-022	Resources finding
Le Duc Manh		Resources refining
Nguyen Huu Dung	BI9-072	Presentation, Resources finding
Tran Thi Huong Giang	BI9-085	Slide making Resource finding
Do Thanh Dat	BI9-065	Report Making

III. General Tech Stacks

Programming Language: Python
Frameworks

- Pandas
- Sklearn

IV. Overall Ideas

We want to extract water bodies from a satellite map and classify them, in order to better understand the geography of a region.

Our objective is to create a model that could tell various water bodies apart by their colors.

We believe that lakes, rivers and seas, if their locations are not included, will likely have the colors in RGB values lie within a certain range.

V. Project Overview

Regarding the similar research topics related to this project, we have found that most of the available resources focused on Deep Learning, or Convolutional Neural Network (CNN).

The popular method is applying image classification using CNN, a deep learning method that recognizes features and patterns on an image and classifies them.

As far as we know, there is hardly any work mentioning classifying water images based on colors, so we decided to take the initiative.

We believe that this could be a very potential solution since by eye only, the difference between color shades of various water sources are already quite clear.

VI. Steps We Took

Procedure Diagram

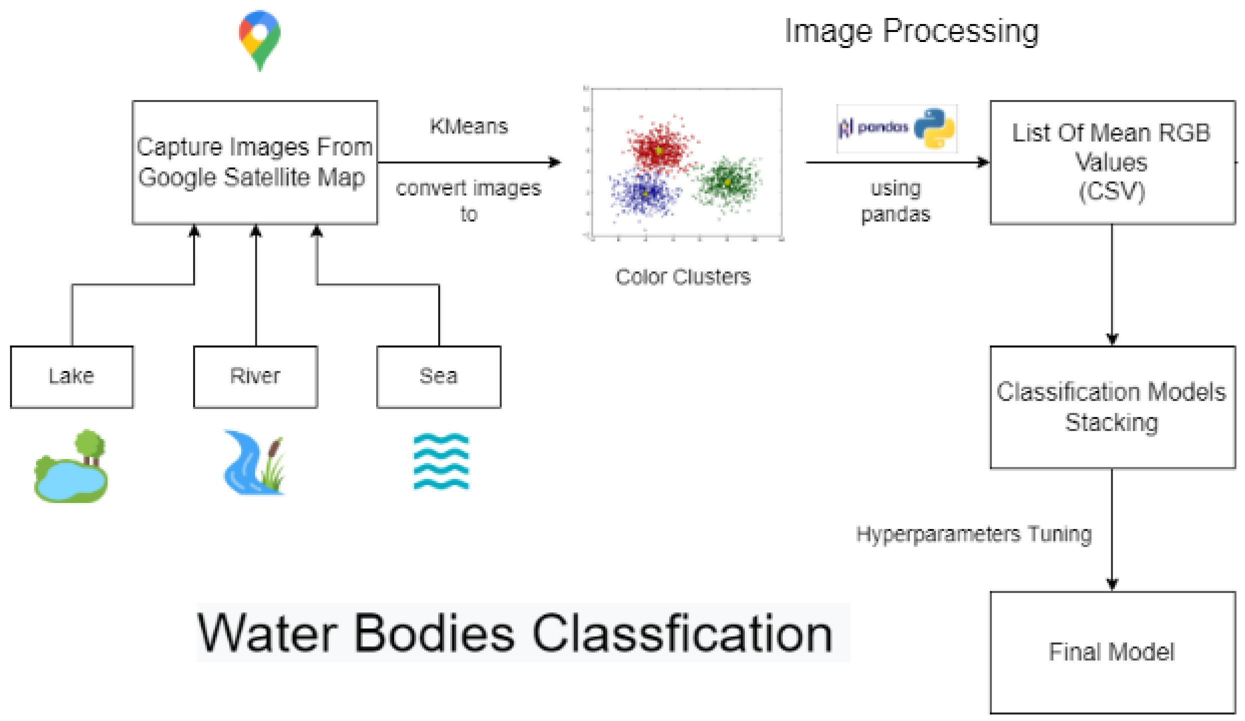


Image Processing
Capture Images From Google Satellite Map
KMeans
convert images to
Color Clusters
pandas
using pandas
List Of Mean RGB Values (CSV)
Lake
River
Sea
Classification Models Stacking
Hyperparameters Tuning
Water Bodies Classfication
Final Model

Diagram Of Water Bodies Classification Process

1st Attempt

Capture Images From Google Satellite Map

Firstly, we take screenshots of different areas that contains lakes, rivers and seas, under the following conditions for each category:

1. For lakes

The screenshot must cover all of its surface. The frame of the screenshot must fit to the shape of the lake as much as possible.

Akataiyo Sushi
(赤太陽 レストラン)

Example Of A Lake Screenshot

2. For rivers

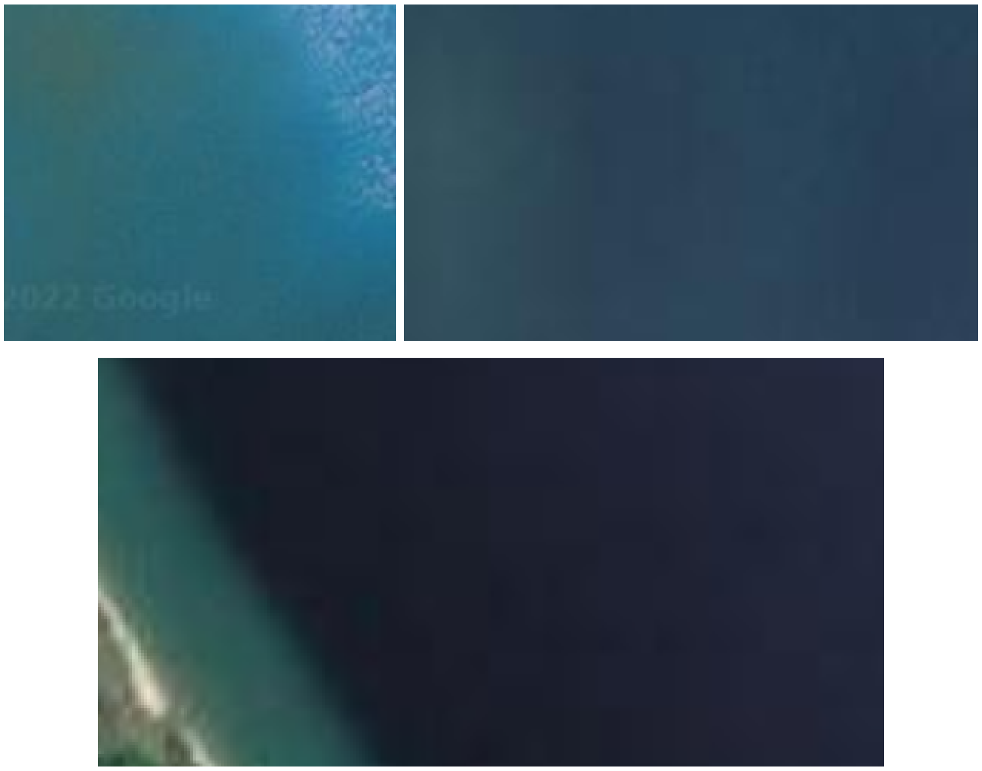
Rivers are usually long and they flow from one place to another, and the water can vary in colors depending on the soil and other factors.

Therefore, we take captures of different parts of a river that have seemingly different colors.

Different Captures Of Rivers

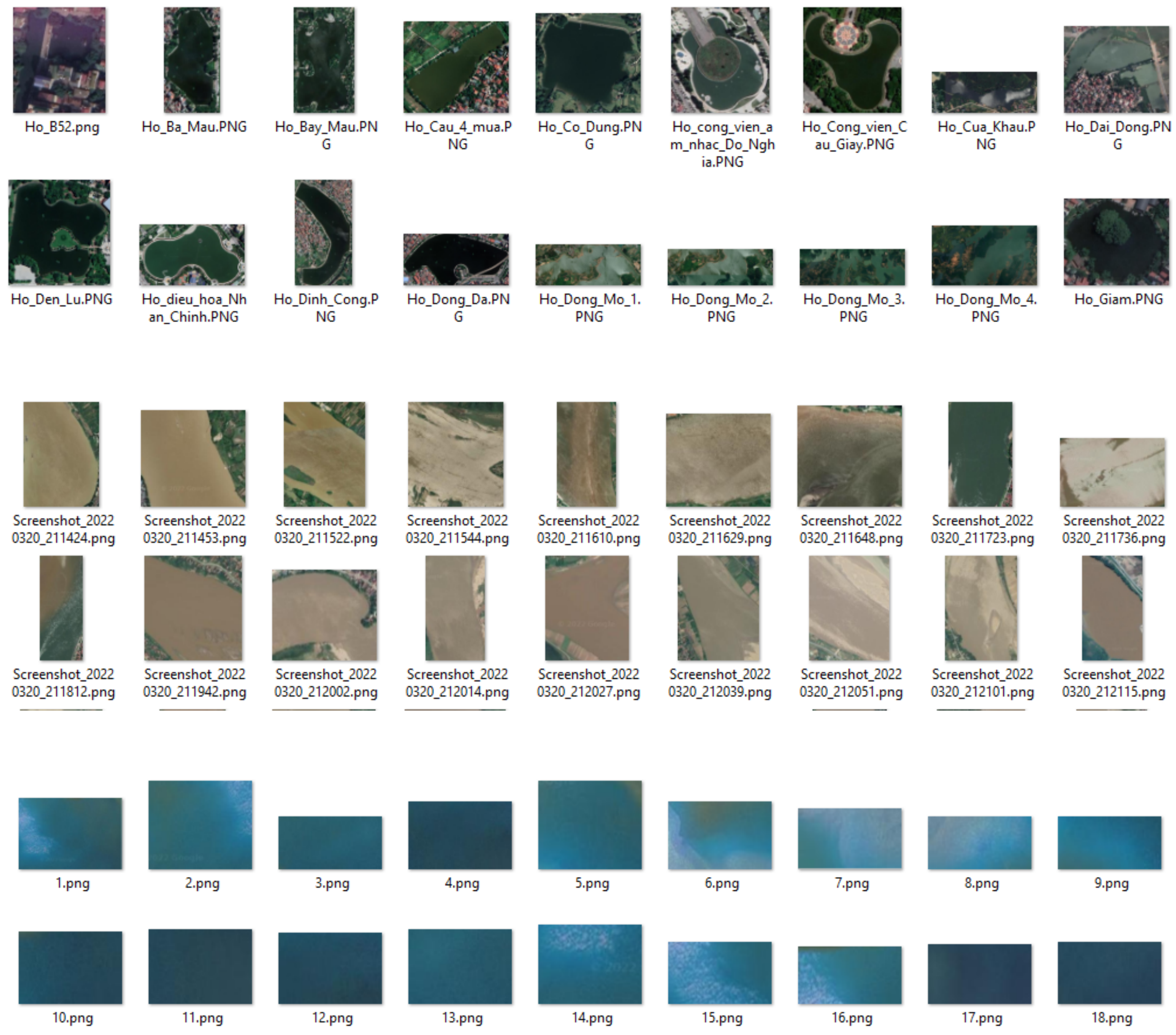
3. For seas

We did the same with the seas what we did with the rivers, taking captures of different water areas that have different color shades.


Captures Of Seas

4. Results

In the end we obtained a total of 30 sea, 123 river and 77 lake images for the first attempt.



Ho_B52.png
Ho_Ba_Mau.PNG
Ho_Bay_Mau.PNG
Ho_Cau_4_mua.PNG
Ho_Co_Dung.PNG
Ho_cong_vien_am_nhac_Do_Nghia.PNG
Ho_Cong_vien_Cau_Giay.PNG
Ho_Cua_Khau.PNG
Ho_Dai_Dong.PNG
Ho_Den_Lu.PNG
Ho_dieu_hoa_Nhan_Chinh.PNG
Ho_Dinh_Cong.PNG
Ho_Dong_Da.PNG
Ho_Dong_Mo_1.PNG
Ho_Dong_Mo_2.PNG
Ho_Dong_Mo_3.PNG
Ho_Dong_Mo_4.PNG
Ho_Giam.PNG
Screenshot_20220320_211424.png
Screenshot_20220320_211453.png
Screenshot_20220320_211522.png
Screenshot_20220320_211544.png
Screenshot_20220320_211610.png
Screenshot_20220320_211629.png
Screenshot_20220320_211648.png
Screenshot_20220320_211723.png
Screenshot_20220320_211736.png
Screenshot_20220320_211812.png
Screenshot_20220320_211942.png
Screenshot_20220320_212002.png
Screenshot_20220320_212014.png
Screenshot_20220320_212027.png
Screenshot_20220320_212039.png
Screenshot_20220320_212051.png
Screenshot_20220320_212101.png
Screenshot_20220320_212115.png
1.png
2.png
3.png
4.png
5.png
6.png
7.png
8.png
9.png
10.png
11.png
12.png
13.png
14.png
15.png
16.png
17.png
18.png

Demo Of All Captures

Using KMeans to group colors from an image

KMeans is an unsupervised learning method that tries to group data points into groups based on their distance to each other. First the algorithm takes any k number of neighbors we want, then randomly choose k points to be the centroids.

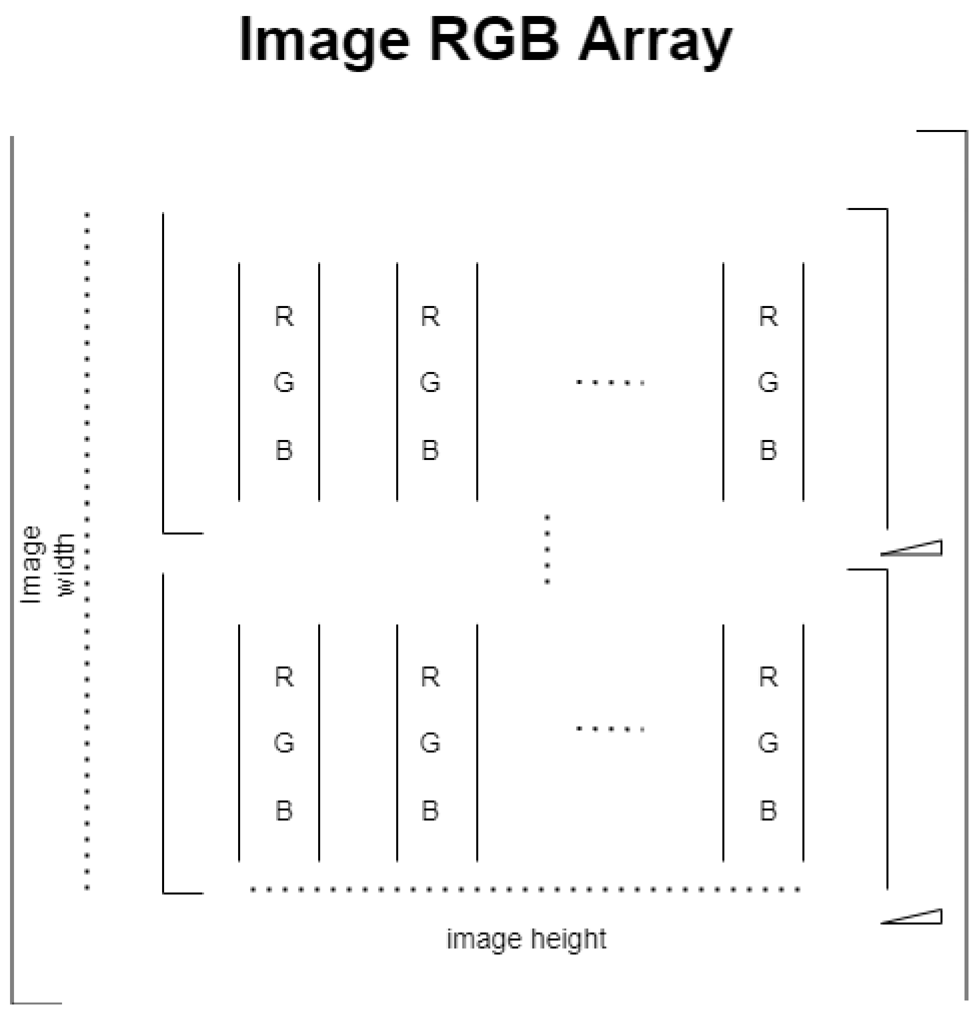
We then continuously repeat the process to compute and find the new centroids based on the distance among data points until the centroids do not change their positions or no new clusters are formed.

The details of KMeans algorithm could be found in the following article
https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

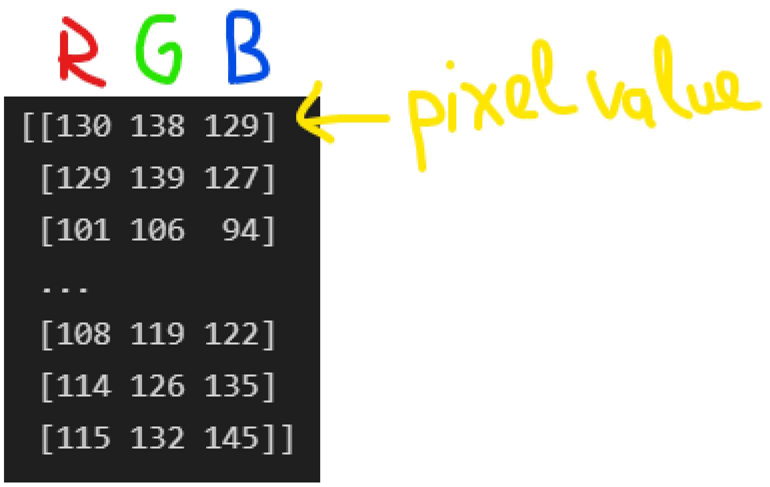
In this project, we used KMeans to group pixel RGB values into groups, or color clusters. We chose the maximum number of clusters to be 10 to reduce the computational time.

In order for KMeans to work on the images, we must first preprocess them in 4 steps:

1. Read the image and obtain a 3 dimensional arrays of RGB values



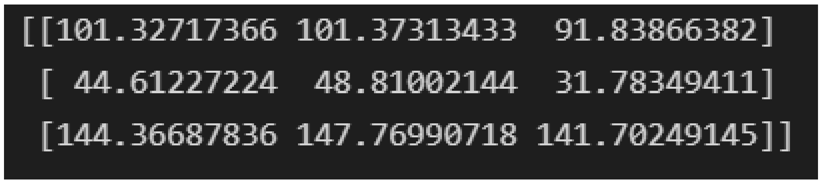
2. Resize the image to 100x100 to reduce computational time
3. Reshape the image to a 2 dimensional array to fit into the KMeans model. The result shape of of RGB array is (image_width * image_height, 3), meaning that we will get an array of RGB with image_width * image_height elements.



```
[[130 138 129]
 [129 139 127]
 [101 106  94]
 ...
 [108 119 122]
 [114 126 135]
 [115 132 145]]
```

Image Input To KMeans

4. Passing the K=3 as a parameter to KMeans. We then obtain the cluster centers, or a list of RGB arrays of every centroid. In the image below is an example of a list that contains RGB arrays:



```
[[101.32717366 101.37313433  91.83866382]
 [ 44.61227224  48.81002144  31.78349411]
 [144.36687836 147.76990718 141.70249145]]
```

Cluster Center With K=3

Get a list of the images RGB values

Lastly, we take the mean of the cluster center and obtain a mean RGB array for an image. Do this for every image in the dataset and we obtain a table of mean RGB values for every image.

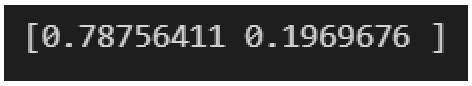
Scaling the dataset

To minimize any potential errors, we used StandardScaler() built-in function in sklearn.

The standard scaler standardizes the data, in other words, the data will have the mean at 0 and the standard deviation equal to 1.

PCA - Principal Component Analysis

To implement the classification models like KNN and SVC, we applied PCA on the dataset to reduce 3 features (R, G, B) down to 2.



```
[0.78756411 0.1969676 ]
```

Explained Variance Ratio

The sum of the explained variance ratio for 2 components PC1 and PC2 is more than 98%, which suggests only 2 components are capable of capturing more than 98% of the original information.

Evaluation

We tried many available classification models, including:

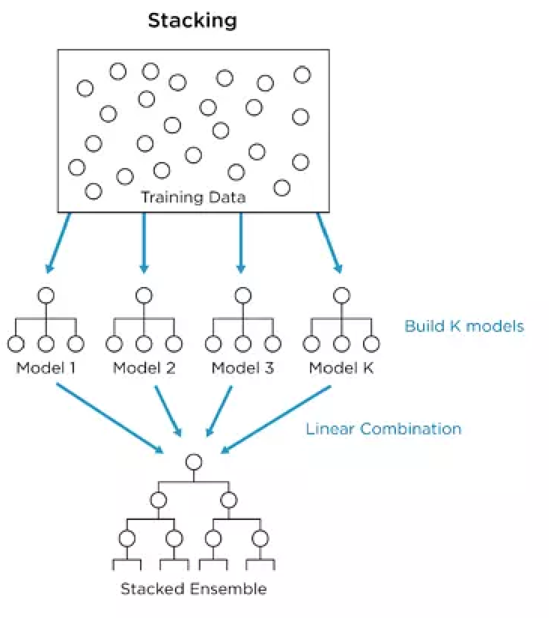
- KNN - K nearest neighbor
- SVC - Support Vector Classifier
- Decision Tree
- Naives Bayes (Gaussian, Multinomial, Categorical)

After applying Grid Search with 10 fold cross validation and hyperparameters tuning, we obtained the following results:

Model	**KNN**	**SVC**	**Decision Tree**	**GNB**	**MNB**	**CNB**
Accuracy Score	0.80	0.89	0.89	0.67	0.78	0.56
n_neighbors	5	-	-	-	-	-
kernel	-	rbf	-	-	-	-
C	-	10000	-	-	-	-
gamma	-	0.1	-	-	-	-
max_depth	-	-	5	-	-	-
min_samples_leaf	-	-	5	-	-	-

We can see that SVC and Decision Tree have the best accuracy. Therefore, we chose these two to be in the stacked model.

With the stacking technique, which combines the prediction of many different models and passes them as features into the final learning model (Logistic Regression in this case). The final model is also known as the meta learner or level 1 model.



Stacking Visualized

After stacking SVC and Decision Tree, unfortunately the accuracy did not improve, going down to 0.86 or 86%.

To further better the accuracy of our classification model, we moved toward the 2nd attempt.

2nd Attempt

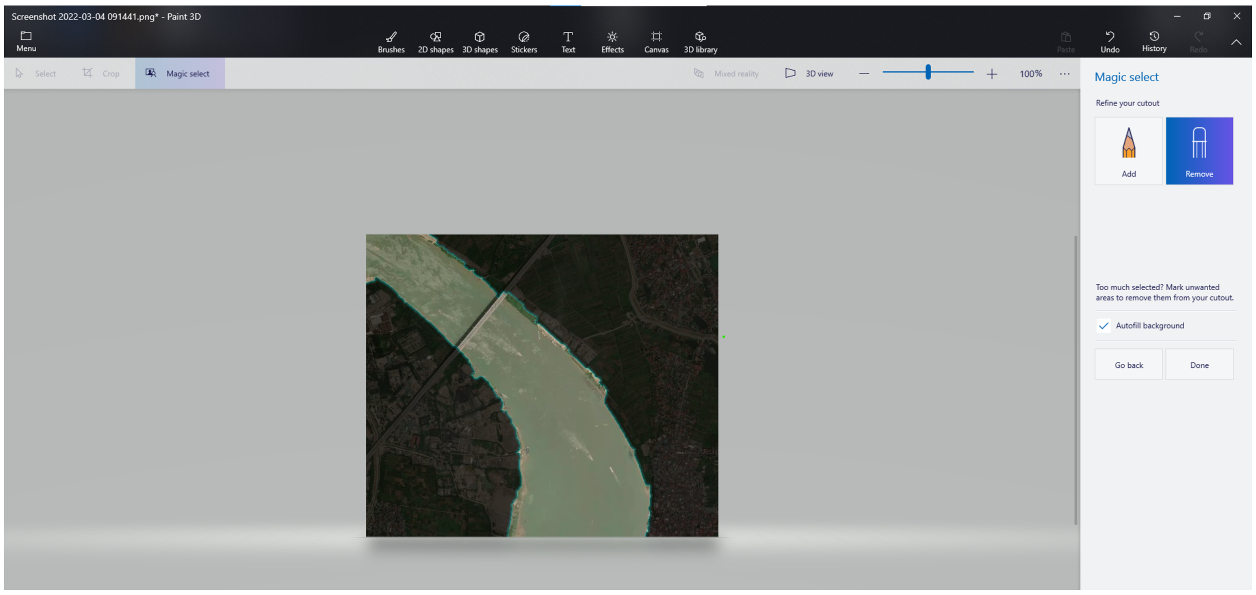
Improve input images

In the second attempt we got a total of 75 lakes, 124 rivers and 76 sea images.

After using the dataset from the first attempt, we came up with a classification model at best 89% accuracy after hyperparameters tuning.

In order to generate a better result, we required more images and furthermore, we extracted only the water, leaving out the rest of the details.

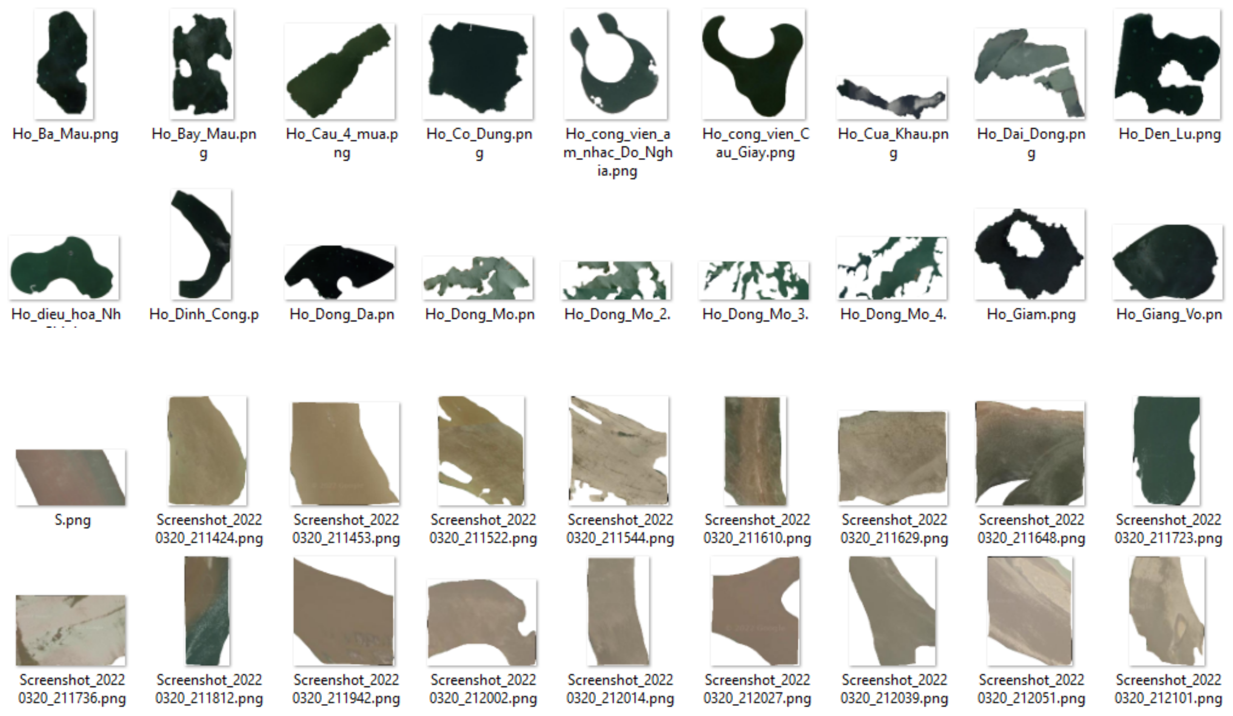
To extract only the water body from an image, we used the magic select feature in Paint3D.



Too much selected? Mark unwanted areas to remove them from your cutout.

Paint3D Magic Select

Our new input images are as follows:


Ho_Ba_Mau.png
Ho_Bay_Mau.png
Ho_Cau_4_mua.png
Ho_Co_Dung.png
Ho_cong_vien_am_nhac_Do_Nghia.png
Ho_cong_vien_Cau_Giay.png
Ho_Cua_Khau.png
Ho_Dai_Dong.png
Ho_Den_Lu.png
Ho_dieu_hoa_Nh
Ho_Dinh_Cong.p
Ho_Dong_Da.pn
Ho_Dong_Mo.pn
Ho_Dong_Mo_2.
Ho_Dong_Mo_3.
Ho_Dong_Mo_4.
Ho_Giam.png
Ho_Giang_Vo.pn
S.png
Screenshot_20220320_211424.png
Screenshot_20220320_211453.png
Screenshot_20220320_211522.png
Screenshot_20220320_211544.png
Screenshot_20220320_211610.png
Screenshot_20220320_211629.png
Screenshot_20220320_211648.png
Screenshot_20220320_211723.png
Screenshot_20220320_211736.png
Screenshot_20220320_211812.png
Screenshot_20220320_211942.png
Screenshot_20220320_212002.png
Screenshot_20220320_212014.png
Screenshot_20220320_212027.png
Screenshot_20220320_212039.png
Screenshot_20220320_212051.png
Screenshot_20220320_212101.png

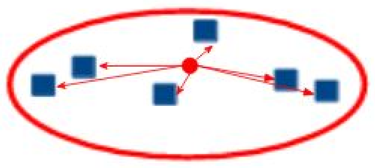
Extracted Water Features From Images

For the sea images specifically, there was no need for water features extraction since they were already water only images.

Choosing the best K number of clusters in KMeans

Select the best K (number of clusters) by choosing the one with the min inertia, or the sum of squares of centroids with their corresponding clusters.

In other words, the best K is obtained by minimizing the distance between the centroids to other data points inside the clusters



Intra cluster distance

Distance of a centroid to other data points in a cluster

Now each image will generate the best K number of clusters (K with the lowest inertia), we proceed to retrying to fit the new dataset to the learning models.

Final Evaluation

We tried stacking all the models including SVC, Decision Tree, KNN, Random Forest, XGBoost, Gaussian Naives Bayes to the new dataset with their default parameters and obtain a much better accuracy at 0.96 or 96%.

Thus, we knew that improving the input images and tuning KMeans yielded fruits.

However, that was just a test. We dug deeper to see which model actually did the best.

Therefore, after hyperparameters tuning with 10 fold cross validation, we obtained the following results

Model	**Decision Tree**	**SVC**	**KNN**	**GNB**	**CNB**	**MNB**
Accuracy Score	0.92	0.96	0.94	0.91	0.74	0.87
C	-	10	-	-	-	-
gamma	-	1	-	-	-	-

In conclusion, SVC with C=10 and gamma=1 performed best, with an accuracy score after cross validation at 96%.

VII. Summary

Most accurate model

SVC or Support Vector Classifier after 2 attempts remained the most accurate classification model we had.

Application

As far as we can see, detecting water bodies and classifying them based on the colors is a promising approach that can be applied in geographical analysis and geomarketing.

With this approach, we will be able to generate a water map from the satellite images, as well as assessing the quality of such water.

In general, real estate properties tend to have higher prices when they are near a water source. The atmosphere is more pleasant when there is a lake nearby, and agricultural activities could be better facilitated.

Methodology

By nature, lakes, rivers and the sea have different water sources, the fact that makes them easily discernible by eye.

However, when we want to work on a large dataset, for example a satellite map of a not yet urbanized area, not only we need to tell various water bodies apart, but also extract them en masse.

We do not need to resort to Deep Learning, which gives you little control over how the mechanism works behind the scene to generate the result.

Classifying water bodies based on colors enables us to get a better feel for the problem we are trying to solve.

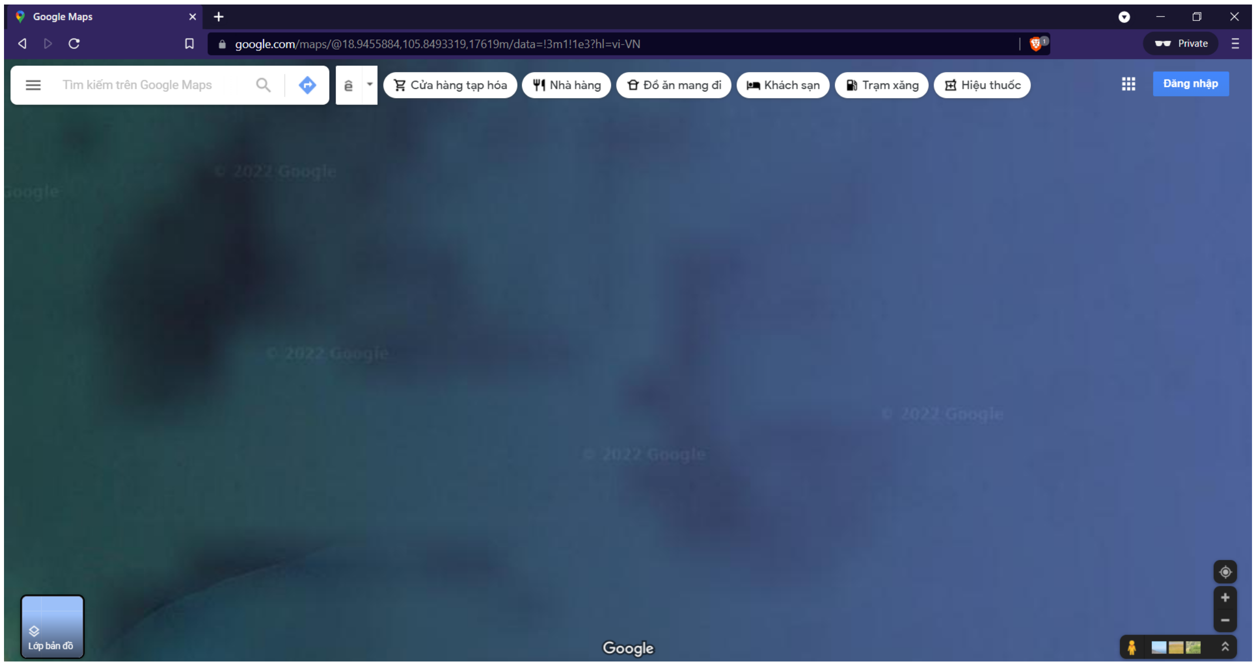
Future Improvements

The only feature we are missing in the dataset is the location of the water source. Rivers in Northern Vietnam will have different color shades compared to the Ganga River in India since they have different sources.

Interesting Findings

It seemed to us that Google Map satellite images were unreliable in certain regions, due to geopolitical reasons.

We theorize that In some water regions there are conflicts or sensitive information, thus the images were of strange colors and patterns.



High Sea Captures With Purple-ish Color

VIII. Source Code

https://github.com/vietanh2000april/water_bodies_classification