
ĐÈN GIAO THÔNG

Báo Cáo: AiThing

NGƯỜI THỰC HIỆN:

VŨ VĂN HUY

NGUYỄN VĂN TUẤN

HƯỚNG DẪN:

TS. NGUYỄN PHONG

LAB. COMPUTER VISION

HÀ NỘI, 11/2023

MỤC LỤC

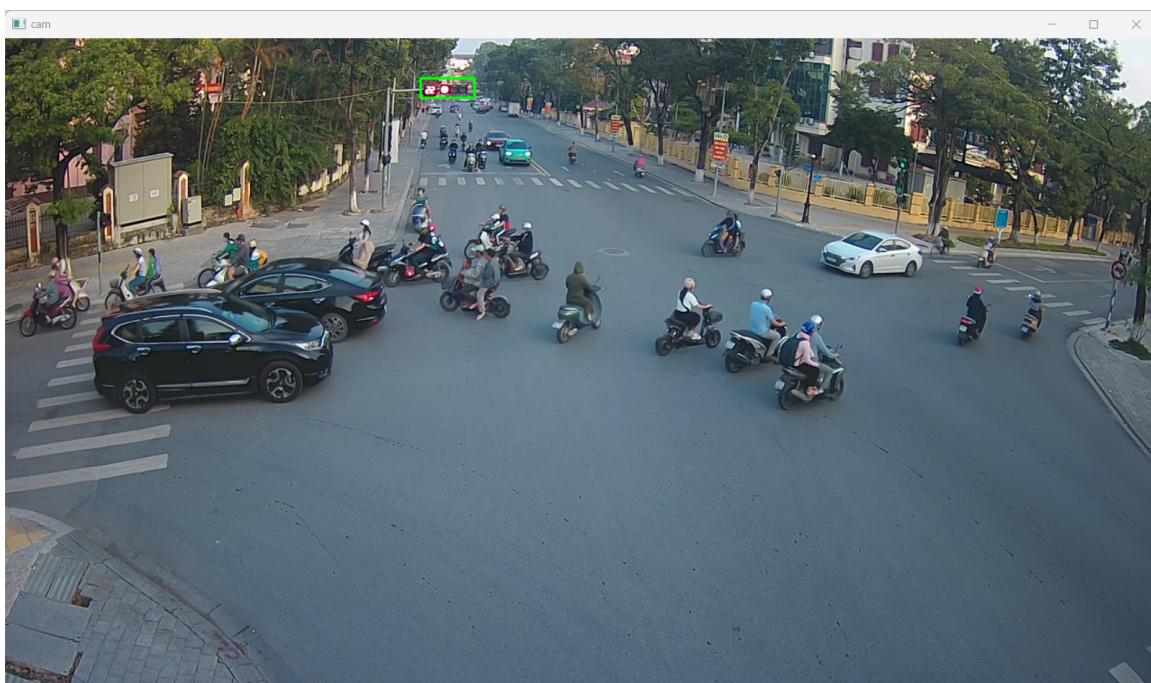
1	Bài toán	4
1.1	Đèn giao thông	4
1.2	Các mẫu dữ liệu thu thập được	4
2	Trích xuất các đặc trưng	5
2.1	Quá trình thu thập dữ liệu	5
2.2	Màu sắc	5
3	Kết quả thực nghiệm test các loại model	6
3.1	Cài đặt thư viện	6
3.2	1 vài đoạn code thân chương trình	6
3.3	Desion Tree	9
3.4	SVM	12
3.5	Gradient Boosting	15
3.6	Adaboost	17
4	Nhận xét	19
5	Kết luận	21
5.1	Cải thiện	21
5.2	Khó khăn	21
Tài liệu tham khảo		22

Abstract

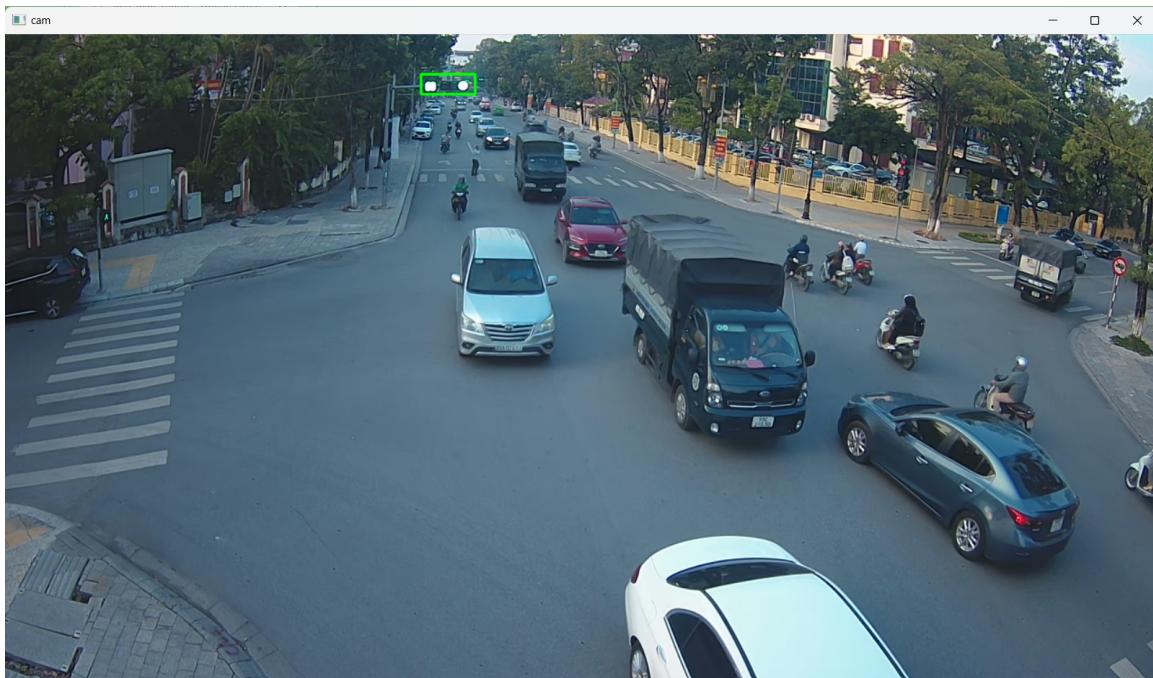
Sử dụng Machine Learning trong việc phân biệt màu sắc của đèn báo giao thông

Trong cuộc sống hàng ngày, đèn báo giao thông đóng vai trò quan trọng trong việc duy trì an toàn và thông suốt của giao thông đường bộ. Trong nỗ lực tối ưu hóa hệ thống này, chúng ta sử dụng sự sáng tạo bằng cách áp dụng các thuật toán cơ bản như Decision Tree, SVM, AdaBoost và Gradient Boost để phân loại màu sắc của các đèn báo.

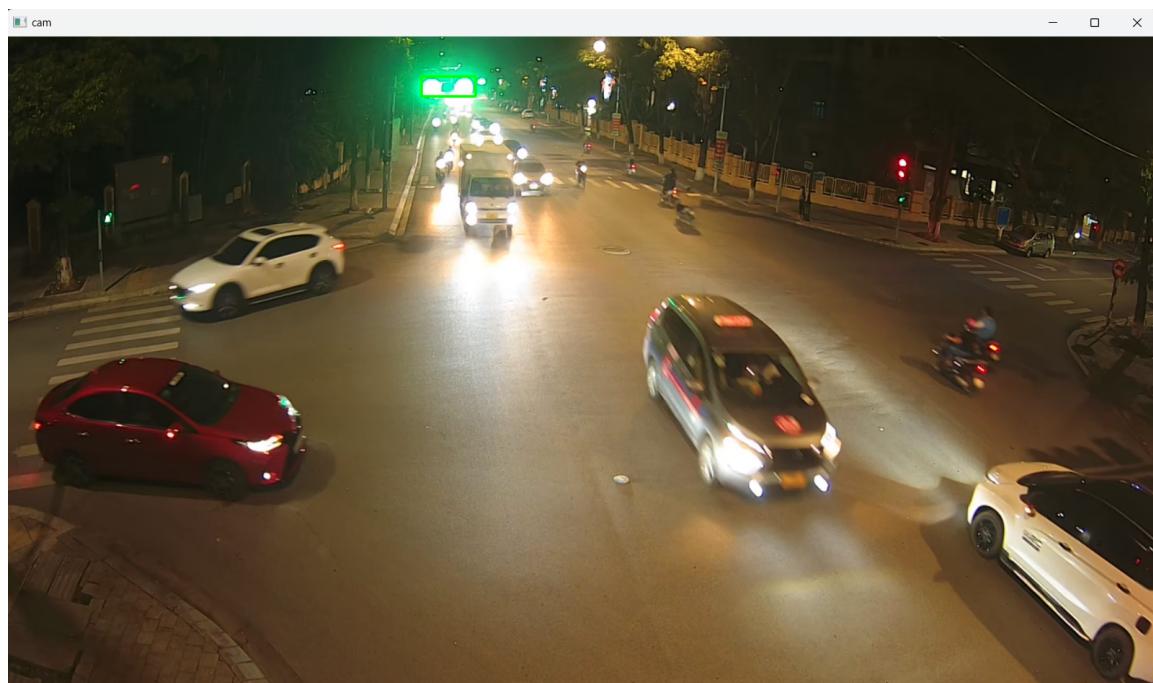
Bằng việc tận dụng xử lý ảnh để trích xuất thông tin từ hình ảnh của đèn báo giao thông, và sau đó sử dụng các thuật toán này để phân loại màu sắc một cách đáng tin cậy. Kết quả của nghiên cứu này không chỉ mang lại khả năng cải thiện quản lý giao thông mà còn góp phần đảm bảo tính liên tục của luồng giao thông trên đường phố, mà còn thể hiện sự sáng tạo của con người thông qua việc ứng dụng các phương pháp cơ bản vào cuộc sống hàng ngày.



Hình 1: ảnh sáng



Hình 2: ảnh chiều



Hình 3: ảnh tối

1 Bài toán

1.1 Đèn giao thông

Bài toán của dự án là phân biệt màu sắc của đèn giao thông, bao gồm màu vàng, xanh và đỏ. Tọa độ của đèn giao thông là cố định, và ảnh của đèn được cắt trực tiếp từ một camera với tọa độ cố định. Dữ liệu đầu vào bao gồm 1150 ảnh của đèn giao thông, được thu thập vào nhiều thời điểm trong ngày thông qua thực nghiệm. Mục tiêu của bài toán là dự đoán màu sắc của đèn dựa trên ảnh đó

Mục tiêu của dự án là biến một ảnh của đèn giao thông thành một dự đoán màu sắc đáng tin cậy. Dự án này không chỉ đòi hỏi kiến thức về máy học và xử lý ảnh mà còn đặt ra một câu hỏi quan trọng: Làm thế nào chúng ta có thể dự đoán màu sắc của một đèn từ một khung ảnh, mà đèn này đã tồn tại với màu sắc đặc trưng của nó?

1.2 Các mẫu dữ liệu thu thập được

Ảnh tối	Ảnh chiều	Ảnh sáng

Bảng 1: Các trạng thái của đèn giao thông trong ngày

2 Trích xuất các đặc trưng

2.1 Quá trình thu thập dữ liệu

Dữ liệu về hình ảnh đèn giao thông được thu thập trực tiếp qua camera giao thông.

Quy trình:

- Cắt ảnh từ camera giao thông.
- Lọc riêng các ảnh xanh, đỏ, vàng.
- Đổi tên các ảnh thu được lần lượt theo thứ tự.
- Gộp các ảnh vào 1 file và gắn nhãn hàng loạt.
 - * 1: xanh
 - * 0: vàng
 - * -1: đỏ

2.2 Màu sắc

Từ những bức ảnh ban đầu, chúng được chuyển đổi thành một ma trận kích thước 10×10 với ba kênh màu (RGB). Mỗi phần tử trong ma trận đại diện cho một pixel và chứa các giá trị màu (R, G, B). Vì tọa độ đèn giao thông không thay đổi, mỗi phần tử trong ảnh được xem xét như một thuộc tính riêng biệt, dẫn đến tổng cộng 100 thuộc tính.

Để phân biệt các màu sắc, chúng ta tiếp tục quan sát các phân phối màu thông qua giá trị RGB. Ví dụ, đèn đỏ thường có giá trị cao cho cả kênh R và G, đèn vàng có giá trị cao cho kênh G và B, và đèn xanh có giá trị cao cho kênh B. Dựa trên những quan sát này, chúng ta trích xuất 3 thuộc tính: tổng giá trị R, tổng giá trị G và tổng giá trị B trong mỗi bức ảnh.

3 Kết quả thực nghiệm test các loại model

3.1 Cài đặt thư viện

Cài đặt thư viện

```
1 py -m pip install requirement.txt
```

Với **requirement.txt** là file chứa tên các thư viện cần. **Hoặc**

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import tensorflow as tf
4 import webcolors
5 import pandas as pd
```

3.2 1 vài đoạn code thân chương trình

Đổi kích cỡ thành 10x10 và 3 kênh màu.

Giải mã chuỗi byte thành một tensor ảnh.

```
1 def preprocess(file_path):
2     byte_img = tf.io.read_file(file_path)
3     img = tf.io.decode_jpeg(byte_img)
4     img = tf.image.resize(img, (10, 10))
5     img = img / 255.0
6
7     return img
```

Hàm lấy màu:

- Lấy tất cả các màu có trong ảnh
- Lấy kích thước ảnh
- Chuyển đổi Tensor thành NumPy array
- Khởi tạo một để lưu trữ các giá trị màu
- Duyệt qua từng điểm ảnh và thu thập các giá trị màu

Trả về 103 thuộc tính 100 thuộc tính đầu là tổng giá trị rgb tại 1 pixel , 3 thuộc tính còn lại tổng từng r,g,b mỗi ảnh

```
1 def get_all_colors(image):
2     r=g=b=0
3
4     height, width, _ = image.shape
5
6     image_np = np.array(image)
7
8     all_colors = []
9
10    for y in range(height):
11        for x in range(width):
12            pixel_color = tuple(image_np[y, x])
13            all_colors.append(pixel_color[0]+pixel_color[1]+pixel_color
14                [2])
15            r=pixel_color[0]+r
16            g=pixel_color[1]+g
17            b=pixel_color[2]+b
18
19            all_colors.append(r)
20            all_colors.append(g)
21            all_colors.append(b)
22
23    return all_colors
```

Gán nhān:

Lấy vào đường dẫn

```

1 import os
2 import numpy
3 img_list=os.listdir("..\Image")
4 sorted_img_list = sorted(img_list)
5
6 print(sorted_img_list)
7 print(len(sorted_img_list))

```

Thực hiện tiền xử lý tất cả ảnh, lấy ma trận điểm ảnh trong thư mục.

```

1 it=os.listdir("..\Image")
2 s = sorted(it)
3
4 dt = []
5 nt=[]
6 for img in s:
7     if(img != '.DS_Store'):
8         image = preprocess("../Image/" + img)
9         col=get_all_colors(image)
10        nt.append(img)
11        dt.append(col)
12 dff = pd.DataFrame(dt)
13 print(dff)
14 print(nt)
15 df=dff
16 X=dff

```

3.3 Design Tree

```

1 clf1=tree.DecisionTreeClassifier(random_state=0)
2 clf1.fit(X_train,Y_train)
3 from sklearn.metrics import accuracy_score,confusion_matrix
4 y_pred1 = clf.predict(X_test)

# Tính recall, precision và F1-score
5 recall1 = recall_score(Y_test, y_pred1,average='macro')
6 precision1 = precision_score(Y_test, y_pred1,average='macro')
7 f11 = f1_score(Y_test, y_pred1,average='macro')

8 accuracy01 = clf1.score(X_test, Y_test)
9 accuracy011 = clf1.score(X_train, Y_train)
10 print("Accuracy decision tree train: {:.2f}".format(accuracy011))
11 print("Accuracy : {:.2f}".format(accuracy01))
12 print("Recall: {:.2f}".format(recall1))
13 print("Precision: {:.2f}".format(precision1))
14 print("F1-score: {:.2f}".format(f11))

Accuracy decision tree train: 1.00
Accuracy : 0.65
Recall: 0.54
Precision: 0.76
F1-score: 0.57

```

```

1 clf1=tree.DecisionTreeClassifier(random_state=0)
2 clf1.fit(X_train,Y_train)
3 from sklearn.metrics import accuracy_score,confusion_matrix
4 y_pred1 = clf.predict(X_test)

# Tính recall, precision và F1-score
5 recall1 = recall_score(Y_test, y_pred1,average='macro')
6 precision1 = precision_score(Y_test, y_pred1,average='macro')
7 f11 = f1_score(Y_test, y_pred1,average='macro')

8 accuracy01 = clf1.score(X_test, Y_test)
9 accuracy011 = clf1.score(X_train, Y_train)
10 print("Accuracy decision tree train: {:.2f}".format(accuracy011))
11 print("Accuracy : {:.2f}".format(accuracy01))
12 print("Recall: {:.2f}".format(recall1))
13 print("Precision: {:.2f}".format(precision1))
14 print("F1-score: {:.2f}".format(f11))

Accuracy decision tree train: 1.00
Accuracy : 0.79
Recall: 0.82
Precision: 0.79
F1-score: 0.80

```

Hình 4: Code chưa tối ưu với dữ liệu 300 Hình 5: Code chưa tối ưu với dữ liệu 1150 ảnh

Thuật toán:

```

1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.metrics import accuracy_score, recall_score,
3     precision_score, f1_score
4 from sklearn.model_selection import GridSearchCV
5 import matplotlib.pyplot as plt
6
7 # Dinh nghia danh sach cac gia tri max_depth va min_samples_split de thu
8 param_grid = {
9     'max_depth': [None, 10, 20],
10    'min_samples_split': [2, 5, 10]
11 }
12 # Tao du lieu mau voi 500 mau va 25 dac trung
13 X, Y = make_classification(n_samples=500, n_features=25, random_state=1)
14
15 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
16 random_state=1)
17
18 # Tao mot doi tuong Grid Search
19 grid_search = GridSearchCV(estimator=DecisionTreeClassifier(random_state
20 =0), param_grid=param_grid, scoring='accuracy', cv=3)
21

```

```

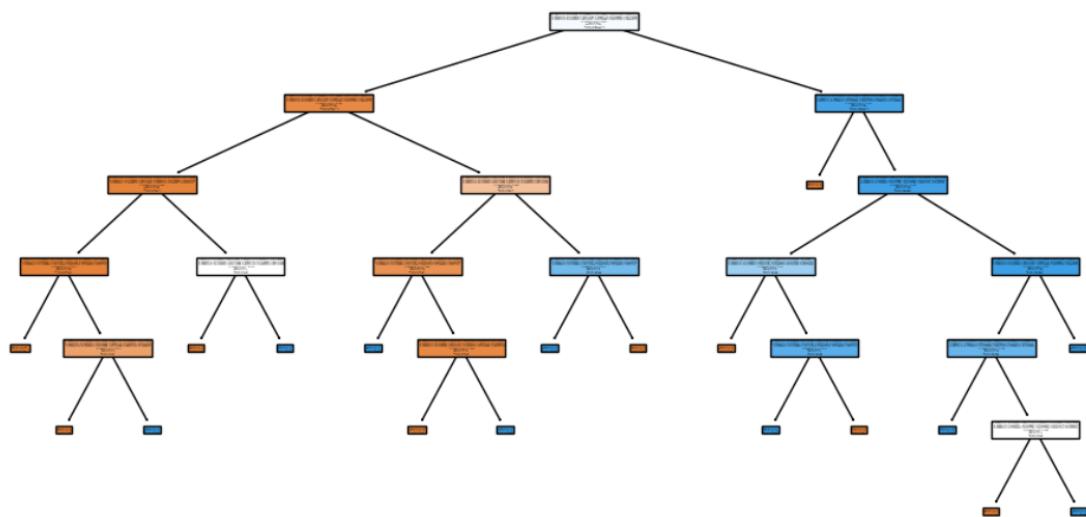
22 # In ra gia tri tot nhat cua sieu tham so
23 best_params = grid_search.best_params_
24 print("Best Parameters:", best_params)
25
26 # Su dung mo hinh voi sieu tham so tot nhat de danh gia
27 best_clf = grid_search.best_estimator_
28 y_pred1 = best_clf.predict(X_test)
29
30 # Tinh cac chi so danh gia
31 accuracy_train = best_clf.score(X_train, Y_train)
32 accuracy_test = best_clf.score(X_test, Y_test)
33 recall = recall_score(Y_test, y_pred1, average='macro')
34 precision = precision_score(Y_test, y_pred1, average='macro')
35 f1 = f1_score(Y_test, y_pred1, average='macro')
36
37 print("Accuracy (train) decision tree: {:.2f}".format(accuracy_train))
38 print("Accuracy (test): {:.2f}".format(accuracy_test))
39 print("Recall: {:.2f}".format(recall))
40 print("Precision: {:.2f}".format(precision))
41 print("F1-score: {:.2f}".format(f1))
42
43 plt.figure(figsize=(12, 6))
44 from sklearn.tree import plot_tree
45 plot_tree(best_clf, filled=True, feature_names=list(X_train),
        class_names=['red', 'green', 'yellow'])
46 plt.show()

```

Kết quả:

❖ Tốt nhất với accuracy: 98%

```
Best Parameters: {'max_depth': None, 'min_samples_split': 2}
Accuracy (train) decision tree: 1.00
Accuracy (test): 0.98
Recall: 0.98
Precision: 0.98
F1-score: 0.98
```



Hình 6: Code đã tối ưu với dữ liệu 1150 ảnh

3.4 SVM

```

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn import svm
from sklearn.metrics import accuracy_score,confusion_matrix

clf = svm.SVC(kernel = 'linear')
clf.fit(X_train,Y_train)
from sklearn.metrics import precision_score, recall_score, f1_score

# Đoán nhãn dựa trên dữ liệu kiểm tra
y_pred = clf.predict(X_test)

# Tính recall, precision và F1-score
recall = recall_score(Y_test, y_pred,average='macro')
precision = precision_score(Y_test, y_pred,average='macro')
f1 = f1_score(Y_test, y_pred,average='macro')

accuracy = clf.score(X_test, Y_test)
accuracy1 = clf.score(X_train, Y_train)
print("Accuracy svm train: {:.2f}".format(accuracy1))
print("Accuracy : {:.2f}".format(accuracy))
print("Recall: {:.2f}".format(recall))
print("Precision: {:.2f}".format(precision))
print("F1-score: {:.2f}".format(f1))

```

Accuracy svm train: 0.74
 Accuracy : 0.65
 Recall: 0.54
 Precision: 0.76
 F1-score: 0.57

```

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn import svm
from sklearn.metrics import accuracy_score,confusion_matrix

clf = svm.SVC(kernel = 'linear')
clf.fit(X_train,Y_train)
from sklearn.metrics import precision_score, recall_score, f1_score

# Đoán nhãn dựa trên dữ liệu kiểm tra
y_pred = clf.predict(X_test)

# Tính recall, precision và F1-score
recall = recall_score(Y_test, y_pred,average='macro')
precision = precision_score(Y_test, y_pred,average='macro')
f1 = f1_score(Y_test, y_pred,average='macro')

accuracy = clf.score(X_test, Y_test)
accuracy1 = clf.score(X_train, Y_train)
print("Accuracy svm train: {:.2f}".format(accuracy1))
print("Accuracy : {:.2f}".format(accuracy))
print("Recall: {:.2f}".format(recall))
print("Precision: {:.2f}".format(precision))
print("F1-score: {:.2f}".format(f1))

```

Accuracy svm train: 0.81
 Accuracy : 0.81
 Recall: 0.82
 Precision: 0.79
 F1-score: 0.80

Hình 7: Code chưa tối ưu với dữ liệu 1150 ảnh

Hình 8: Code chưa tối ưu với dữ liệu 1150 ảnh

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.svm import SVC
3 from sklearn.metrics import accuracy_score, confusion_matrix,
   precision_score, recall_score, f1_score
4 from sklearn.datasets import make_classification
5 from sklearn.model_selection import GridSearchCV
6
7 # Tao du lieu mau voi 500 mau va 25 dac trung
8 X, Y = make_classification(n_samples=500, n_features=25, random_state=1)
9
10 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
   random_state=1)
11
12 # Dinh nghia danh sach cac gia tri C va kernel de thu
13 param_grid = {
14     'C': [0.1, 1, 10],
15     'kernel': ['linear', 'rbf', 'poly']
16 }
17
18 # Tao mot doi tuong Grid Search
19 grid_search = GridSearchCV(estimator=SVC(), param_grid=param_grid,
   scoring='accuracy', cv=3)
20

```

```

21 # Tien hanh tim sieu tham so tot nhat
22 grid_search.fit(X_train, Y_train)
23
24 # In ra gia tri tot nhat cua sieu tham so
25 best_params = grid_search.best_params_
26 print("Best Parameters:", best_params)
27
28 # Su dung mo hinh voi sieu tham so tot nhat de danh gia
29 best_clf = grid_search.best_estimator_
30 Y_pred = best_clf.predict(X_test)
31
32 # Tinh cac chi so danh gia
33 accuracy_train = best_clf.score(X_train, Y_train)
34 accuracy_test = best_clf.score(X_test, Y_test)
35 recall = recall_score(Y_test, Y_pred, average='macro')
36 precision = precision_score(Y_test, Y_pred, average='macro')
37 f1 = f1_score(Y_test, Y_pred, average='macro')
38
39 print("Accuracy (train) svm: {:.2f}".format(accuracy_train))
40 print("Accuracy (test): {:.2f}".format(accuracy_test))
41 print("Recall: {:.2f}".format(recall))
42 print("Precision: {:.2f}".format(precision))
43 print("F1-score: {:.2f}".format(f1))
44
45 import matplotlib.pyplot as plt
46 from sklearn.decomposition import PCA
47
48 # Su dung PCA de giam so chieu du lieu xuong con 2 chieu de bieu dien de
49 # dang hon
50 pca = PCA(n_components=2)
51 X_reduced = pca.fit_transform(X)
52
53 # Khoi tao mo hinh SVM voi sieu tham so tot nhat da tim duoc
54 clf = SVC(C=best_params['C'], kernel=best_params['kernel'])
55 clf.fit(X_reduced, Y)
56
57 # Tao luoi du doan
58 h = .02
59 x_min, x_max = X_reduced[:, 0].min() - 1, X_reduced[:, 0].max() + 1
60 y_min, y_max = X_reduced[:, 1].min() - 1, X_reduced[:, 1].max() + 1
61 xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))

```

```

62 # Du doan cho tung diem tren luoi
63 Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
64 Z = Z.reshape(xx.shape)
65
66 # Bieu dien do thi ket qua
67 plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
68 plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=Y, cmap=plt.cm.coolwarm,
69               s=20)
70 plt.xlabel('Principal Component 1')
71 plt.ylabel('Principal Component 2')
72 plt.title('SVM Decision Boundary')
73 plt.show()

```

Kết quả:

```

Best Parameters: {'C': 1, 'kernel': 'rbf'}
Accuracy (train) svm: 0.99
Accuracy (test): 0.95
Recall: 0.95
Precision: 0.95
F1-score: 0.95

```

Hình 9: Code đã tối ưu với dữ liệu 1150 ảnh

3.5 Gradient Boosting

```

from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier(learning_rate=0.01,random_state=1)
model.fit(X_train, Y_train)
y_pred2=model.predict(X_test)
recall2 = recall_score(Y_test, y_pred2,average='macro')
precision2 = precision_score(Y_test, y_pred2,average='macro')
f2 = f1_score(Y_test, y_pred2,average='macro')

accuracy02 = model.score(X_test, Y_test)
accuracy2 = model.score(X_train, Y_train)
print("Accuracy gradient boost train: {:.2f}".format(accuracy2))
print("Accuracy : {:.2f}".format(accuracy02))
print("Recall: {:.2f}".format(recall2))
print("Precision: {:.2f}".format(precision2))
print("F1-score: {:.2f}".format(f2))

Accuracy gradient boost train: 0.85
Accuracy : 0.63
Recall: 0.52
Precision: 0.66
F1-score: 0.54

```



```

from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier(learning_rate=0.01,random_state=1)
model.fit(X_train, Y_train)
y_pred2=model.predict(X_test)
recall2 = recall_score(Y_test, y_pred2,average='macro')
precision2 = precision_score(Y_test, y_pred2,average='macro')
f2 = f1_score(Y_test, y_pred2,average='macro')

accuracy02 = model.score(X_test, Y_test)
accuracy2 = model.score(X_train, Y_train)
print("Accuracy gradient boost train: {:.2f}".format(accuracy2))
print("Accuracy : {:.2f}".format(accuracy02))
print("Recall: {:.2f}".format(recall2))
print("Precision: {:.2f}".format(precision2))
print("F1-score: {:.2f}".format(f2))

Accuracy gradient boost train: 0.81
Accuracy : 0.69
Recall: 0.57
Precision: 0.80
F1-score: 0.60

```

Hình 10: Code chưa tối ưu với dữ liệu 300 Hình 11: Code chưa tối ưu với dữ liệu 1150 ảnh

```

1 from sklearn.ensemble import GradientBoostingClassifier
2 from sklearn.model_selection import GridSearchCV
3
4 # Tao mot mo hinh Gradient Boosting
5 model = GradientBoostingClassifier(random_state=1)
6 # Tao du lieu mau voi 500 mau va 25 dac trung
7 X, Y = make_classification(n_samples=500, n_features=25, random_state=1)
8
9 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
10 random_state=1)
11
12 # Dinh nghia danh sach cac gia tri sieu tham so can thu
13 param_grid = {
14     'n_estimators': [50, 100, 200],
15     'max_depth': [3, 4, 5],
16     'min_samples_split': [2, 3, 4],
17     'min_samples_leaf': [1, 2, 3]
18 }
19
20 # Tao mot doi tuong Grid Search
21 grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
22 scoring='accuracy', cv=3)
23
24 # Tien hanh tim sieu tham so tot nhat
grid_search.fit(X_train, Y_train)

```

```

25 # In ra gia tri tot nhat cua cac sieu tham so
26 print("Best Parameters:", grid_search.best_params_)
27
28 # Su dung mo hinh voi sieu tham so tot nhat de danh gia
29 best_model = grid_search.best_estimator_
30 y_pred2 = best_model.predict(X_test)
31 recall2 = recall_score(Y_test, y_pred2, average='macro')
32 precision2 = precision_score(Y_test, y_pred2, average='macro')
33 f2 = f1_score(Y_test, y_pred2, average='macro')
34 accuracy02 = best_model.score(X_test, Y_test)
35 accuracy2 = best_model.score(X_train, Y_train)
36
37 print("Accuracy gradient boost train: {:.2f}".format(accuracy2))
38 print("Accuracy : {:.2f}".format(accuracy02))
39 print("Recall: {:.2f}".format(recall2))
40 print("Precision: {:.2f}".format(precision2))
41 print("F1-score: {:.2f}".format(f2))

```

Kết quả:

```

Best Parameters: {'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 200}
Accuracy gradient boost train: 1.00
Accuracy : 0.97
Recall: 0.97
Precision: 0.97
F1-score: 0.97

```

Hình 12: Code đã tối ưu với dữ liệu 1150 ảnh

3.6 Adaboost

```

1 from sklearn.ensemble import AdaBoostClassifier
2
3 model1 = AdaBoostClassifier(random_state=1)
4 model1.fit(X_train, Y_train)
5 y_pred3=model1.predict(X_test)
6 recall3 = recall_score(Y_test, y_pred3,average='macro')
7 precision3 = precision_score(Y_test, y_pred3,average='macro')
8 f3 = f1_score(Y_test, y_pred3,average='macro')
9 accuracy3 = model1.score(X_test, Y_test)
10 accuracy03 = model1.score(X_train, Y_train)
11 print("Accuracy adaboost train: {:.2f}".format(accuracy03))
12 print("Accuracy : {:.2f}".format(accuracy3))
13 print("Recall: {:.2f}".format(recall3))
14 print("Precision: {:.2f}".format(precision3))
15 print("F1-score: {:.2f}".format(f3))

Accuracy adaboost train: 0.73
Accuracy : 0.65
Recall: 0.55
Precision: 0.74
F1-score: 0.57

```

Hình 13: Code chưa tối ưu với dữ liệu 300 ảnh

```

1 from sklearn.ensemble import AdaBoostClassifier
2
3 model1 = AdaBoostClassifier(random_state=1)
4 model1.fit(X_train, Y_train)
5 y_pred3=model1.predict(X_test)
6 recall3 = recall_score(Y_test, y_pred3,average='macro')
7 precision3 = precision_score(Y_test, y_pred3,average='macro')
8 f3 = f1_score(Y_test, y_pred3,average='macro')
9 accuracy3 = model1.score(X_test, Y_test)
10 accuracy03 = model1.score(X_train, Y_train)
11 print("Accuracy adaboost train: {:.2f}".format(accuracy03))
12 print("Accuracy : {:.2f}".format(accuracy3))
13 print("Recall: {:.2f}".format(recall3))
14 print("Precision: {:.2f}".format(precision3))
15 print("F1-score: {:.2f}".format(f3))

Accuracy adaboost train: 0.71
Accuracy : 0.59
Recall: 0.61
Precision: 0.62
F1-score: 0.59

```

Hình 14: Code chưa tối ưu với dữ liệu 1150 ảnh

```

1 from sklearn.ensemble import AdaBoostClassifier
2 from sklearn.model_selection import GridSearchCV
3
4 # Tao mot mo hinh AdaBoost
5 model1 = AdaBoostClassifier(random_state=1)
6
7 # Tao du lieu mau voi 500 mau va 25 dac trung
8 X, Y = make_classification(n_samples=500, n_features=25, random_state=1)
9
10 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
11 random_state=1)
12
13 # Dinh nghia danh sach cac gia tri sieu tham so can thu
14 param_grid = {
15     'n_estimators': [50, 100, 200],
16     'learning_rate': [0.01, 0.1, 1.0]
17 }
18
19 # Tao mot doi tuong Grid Search
20 grid_search = GridSearchCV(estimator=model1, param_grid=param_grid,
21 scoring='accuracy', cv=3)
22
23 # Tien hanh tim sieu tham so tot nhat
24 grid_search.fit(X_train, Y_train)

```

```

24 # In ra gia tri tot nhat cua cac sieu tham so
25 print("Best Parameters:", grid_search.best_params_)
26
27 # Su dung mo hinh voi sieu tham so tot nhat de danh gia
28 best_model1 = grid_search.best_estimator_
29 y_pred3 = best_model1.predict(X_test)
30 recall3 = recall_score(Y_test, y_pred3, average='macro')
31 precision3 = precision_score(Y_test, y_pred3, average='macro')
32 f3 = f1_score(Y_test, y_pred3, average='macro')
33 accuracy3 = best_model1.score(X_test, Y_test)
34 accuracy03 = best_model1.score(X_train, Y_train)
35
36 print("Accuracy adaboost train: {:.2f}".format(accuracy03))
37 print("Accuracy : {:.2f}".format(accuracy3))
38 print("Recall: {:.2f}".format(recall3))
39 print("Precision: {:.2f}".format(precision3))
40 print("F1-score: {:.2f}".format(f3))

```

Kết quả:

```

Best Parameters: {'learning_rate': 1.0, 'n_estimators': 100}
Accuracy adaboost train: 1.00
Accuracy : 0.95
Recall: 0.95
Precision: 0.95
F1-score: 0.95

```

Hình 15: Code đã tối ưu với dữ liệu 1150 ảnh

4 Nhận xét

Khi ta không sử dụng Grid Search để tìm siêu tham số tốt nhất cho các mô hình SVM, AdaBoost, Decision Tree, Gradient Boost, ta sẽ sử dụng các tham số mặc định của mô hình, hoặc tự chọn các giá trị tham số. Điều này có thể dẫn đến việc mô hình không tìm được các giá trị tham số tối ưu cho dữ liệu, dẫn đến độ chính xác thấp và hiệu suất không tốt.

Trong khi đó, khi sử dụng Grid Search, ta sẽ tìm kiếm các giá trị tham số tối ưu cho mô hình. Grid Search sẽ lặp lại quá trình đào tạo và đánh giá với các giá trị tham số khác nhau để tìm ra giá trị tham số tối ưu cho các mô hình. Kết quả là chúng ta sẽ tìm được giá trị tham số tối ưu tốt nhất cho mô hình, dẫn đến độ chính xác cao và hiệu suất tốt hơn.

Vì vậy, sử dụng Grid Search để tìm siêu tham số tối ưu cho các mô hình SVM, AdaBoost, Decision Tree, Gradient Boost có thể giúp tăng độ chính xác và hiệu suất của mô hình.

Ưu điểm:

- Đơn giản và dễ hiểu, không cần phải có kiến thức chuyên sâu về mô hình học máy.
- Đảm bảo tìm được kết quả tốt nhất trong không gian tham số đã cho.
- Có thể song song hóa quá trình tìm kiếm để tăng tốc độ.

Nhược điểm:

- Tốn nhiều thời gian và tài nguyên tính toán, đặc biệt khi không gian tham số lớn và phức tạp.
- Không thể xử lý các tham số liên tục hoặc có phụ thuộc lẫn nhau.
- Không thể mở rộng cho các mô hình học máy phức tạp hơn, như mạng nơ-ron.

Việc tạo dữ liệu mẫu với 500 mẫu và 25 đặc trưng có thể làm cho mô hình tối ưu hơn vì:

- Nó giảm thiểu kích thước của dữ liệu, làm cho việc xử lý và lưu trữ dữ liệu nhanh hơn và tiết kiệm tài nguyên.

- Nó giúp tránh hiện tượng quá khớp (overfitting), khi mô hình học quá nhiều chi tiết và nhiễu từ dữ liệu huấn luyện, làm giảm khả năng khái quát hóa cho dữ liệu mới.
- Nó cho phép sử dụng các phương pháp lựa chọn siêu tham số (hyperparameter tuning) như Grid Search, để tìm ra bộ tham số tối ưu nhất cho mô hình, cải thiện độ chính xác và hiệu năng của mô hình.

Tuy nhiên, việc tạo dữ liệu mẫu cũng có một số nhược điểm, như:

- Nó có thể làm mất đi một số thông tin quan trọng từ dữ liệu gốc, làm giảm độ đại diện của mẫu.
- Nó có thể dẫn đến sai lệch thống kê (bias) nếu phương pháp chọn mẫu không ngẫu nhiên hoặc không phản ánh đúng phân bố của dữ liệu gốc.
- Nó có thể không phù hợp với một số loại dữ liệu hoặc mô hình, ví dụ như dữ liệu chuỗi thời gian (time series) hoặc mô hình học sâu (deep learning).

5 Kết luận

5.1 Cải thiện

Cải thiện hơn so với kết quả lần trước:

1. Kết quả tốt nhất tăng từ 81% lên 98%.
2. Tăng lượng sample từ 300 lên hơn 1150.
3. Tối ưu code hơn chút.

5.2 Khó khăn

1. Khó có thể phân biệt màu sắc trong một vài trường hợp.



Hình 16: Đây là màu gì 😊?

2. Chất lượng hình ảnh thu được từ camera còn kém, mờ, độ nét thấp.
3. Ảnh bị lặp cho delay khi chụp từ laptop.
4. Lọc ảnh lâu, dễ bị lặp ảnh.
5. Dữ liệu thu thập bị chêch lệch giữa các màu ảnh hướng đến kết quả của quá trình học.
6. Đặc biệt là tỷ lệ màu vàng so với các màu khác.

Tham khảo source tại đây:

File image, lable tại đây:

Tài liệu

Scikit-learn: Supervised learning