TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA TOÁN – CƠ – TIN HỌC

BÁO CÁO CUỐI KÌ

ĐỀ TÀI SỐ 01

Môn học**: Học Máy**

Giảng viên: **TS. Cao Văn Chung**

Nhóm thực hiện:

Ngô Văn Minh

Nguyễn Trung Hiếu

Hoàng Minh Tuấn

Học Máy, 6/2022

MỤC LỤC

I, Giới thiệu tổng quan

1.1 Tổng quan về dữ liệu

Tất cả dữ liệu được lưu trữ trong thư mục data trong đó:

+ Thư mục training chứ các ảnh traning của từng con vật

+ Thư mục validation chứa các ảnh test của của từng con vật

Trong từng thư mục training và test chứa 3 thư mục đại diện cho 3 lớp con vật: chó (dogs), mèo (cats), sóc (scoiattolo)

Dữ liệu đầu vào bao gồm 4500 ảnh (3000 ảnh train, 1500 ảnh test)

Dữ liệu training bao gồm 1000 ảnh chó, 1000 ảnh mèo, 1000 ảnh sóc

Dữ liệu test: 500 ảnh chó, 500 ảnh mèo, 500 ảnh sóc

Kiểu dữ liệu: file ảnh (.jpg)

Nguồn dữ liệu:

https://www.kaggle.com/datasets/alessiocorrado99/animals10

Các thư viện sử dụng bao gồm:

+ numy

+ os

+ tensorflow

+ pandas

+ sklearn

Thư viện cần bổ sung là tensorflow:

Hướng dẫn cài đặt tensorflow: https://www.tensorflow.org/install

1.2 Đọc và làm sạch dữ liệu

Đầu tiên về đường dẫn chúng ta gán một biến lưu đường dẫn

```
base_dir = './/data'
```

Liệt kê toàn bộ các nội dung trong thư mục data bằng đường dẫn

```
print("Contents of base directory:")
print(os.listdir(base_dir))

Contents of base directory: ['train', 'validation']
```

Kết quả thu được bao gồm 2 thư mục train và validation

Tiếp tục ta liệt kê toàn bộ các lớp trong thư mục train

```
print("\nContents of train directory:")
print(os.listdir(f'{base_dir}\\train'))

Contents of train directory: ['cats', 'dogs', 'scoiattolo']
```

Ta thu được 3 lớp lần lượt là ‘cats’, ‘dogs’, ‘scoattolo’

Tương tự vậy ta cũng thu được kết quả tương tự với thư mục validation

```
print("\nContents of validation directory:")
print(os.listdir(f'{base_dir}\\validation'))
Contents of validation directory: ['cats', 'dogs', 'scoiattolo']
```

Sau khi thu được các thông tin về cấu trúc thư mục ta sẽ tạo ra các đường dẫn đi đến từng thư mục của từng con vật khác nhau

```
#Tạo các đường dẫn tới từng thư mục của từng con vật
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

# Directory with training cat/dog pictures
train_cats_dir = os.path.join(train_dir, 'cats')
train_dogs_dir = os.path.join(train_dir, 'dogs')
train_scoiattolo_dir = os.path.join(train_dir, 'scoiattolo')

# Directory with validation cat/dog pictures
validation_cats_dir = os.path.join(validation_dir, 'cats')
validation_dogs_dir = os.path.join(validation_dir, 'dogs')
validation_scoiattolo_dir = os.path.join(validation_dir, 'scoiattolo')
```

```
print("\nContents of train directory:")
print(os.listdir(f'{base_dir}\\train'))

print("\nContents of validation directory:")
print(os.listdir(f'{base_dir}\\validation'))


Contents of train directory: ['cats', 'dogs', 'scoiattolo']
Contents of validation directory: ['cats', 'dogs', 'scoiattolo']
```

Đọc dữ liệu thư mục Train

Tận dụng phần đọc dữ liệu của phần CNN ta đọc dữ liệu hình ảnh bằng thư viện tensorflow.keras.preprocessing.image để đọc và chuẩn hóa ảnh thành các ma trận

Ở đây ta để size ảnh là 40 x 40 để tiến hành kiểm thử.

```
from tensorflow.keras.preprocessing.image import img_to_array, load_img

X_train =[]
y_train =[]

# đọc train cats
c = os.listdir(train_cats_dir) # ds tên ảnh cat
for name_image in c:
    img = load_img(train_cats_dir+'/'+name_image,target_size=(40,40))
#40,40  #20,20  #32,32
    x = img_to_array(img)
    X_train.append(x)
    y_train.append(0)

# đọc train dogs
d = os.listdir(train_dogs_dir) # ds tên ảnh dog
for name_image in d:
    img = load_img(train_dogs_dir+"//"+name_image,target_size=(40,40))
    x = img_to_array(img)
    X_train.append(x)
    y_train.append(1)

# đọc train scoiattolo
s = os.listdir(train_scoiattolo_dir) # ds tên ảnh scoiattolo
for name_image in s:
    img = load_img(train_scoiattolo_dir+"//"+name_image,target_size=(40,40))
    x = img_to_array(img)
    X_train.append(x)
    y_train.append(2)
```

Kiểm tra kích thước và kích thước của X_train và y_train sau khi đọc dữ liệu

```
X_train = np.asarray(X_train)

print("Training data shape "+str(X_train.shape))

y_train = np.asarray(y_train)

print("Training label shape "+ str(y_train.shape))

Training data shape (3000, 40, 40, 3)

Training label shape (3000,)
```

Sau khi chạy ta thu được kích thước của X_train gồm 3000 file ảnh với 3000 hàng 1 cột mỗi hàng tương ứng với một ma trận 40 hàng và 40 cột trong mỗi ma trận 40x40 lại có một ma trận 1x3 chứa các chỉ số màu RGB.

Kích thước của y_train ma trận 1 hàng và 3000 cột và chứa nhãn tương ứng với từng hàng trong dữ liệu của X_train, các nhãn là 0:cat, 1 :dog, 2 : scoiattolo.

Đọc dữ liệu thư mục Validation

Tương tự phần với đọc dữ liệu của thư mục Train

```
X_test =[]
y_test =[]

# đọc test cats
c = os.listdir(validation_cats_dir) # ds tên ảnh cat
for name_image in c:
    img = load_img(validation_cats_dir+'/'+name_image,target_size=(40,40))
    x = img_to_array(img)
    X_test.append(x)
    y_test.append(0)

# đọc test dogs
d = os.listdir(validation_dogs_dir) # ds tên ảnh dog
for name_image in d:
    img = load_img(validation_dogs_dir+"//"+name_image,target_size=(40,40))
    x = img_to_array(img)
    X_test.append(x)
    y_test.append(1)

# đọc test scoiattolo
s = os.listdir(validation_scoiattolo_dir) # ds tên ảnh scoiattolo
```

```
for name_image in s:
    img = load_img(validation_scoiattolo_dir+"//"+name_image,target_size=(40,40))
    x = img_to_array(img)
    X_test.append(x)
    y_test.append(2)
```

```
X_test = np.asarray(X_test)
print("Testing data shape:"+ str(X_test.shape))
y_test = np.asarray(y_test)
print("Testing label shape:"+ str(y_test.shape))

Testing data shape:(1500, 40, 40, 3)
Testing label shape:(1500,)
```

Sau khi chạy ta thu được kích thước của X_test gồm 1500 file ảnh với 1500 hàng 1 cột mỗi hàng tương ứng với một ma trận 40 hàng và 40 cột trong mỗi ma trận 40x40 lại có một ma trận 1x3 chứa các chỉ số màu RGB của từng điểm ảnh.

y_test là một ma trận một chiều 1500 hàng tương với 1500 nhãn của ảnh với các nhãn 0:cat, 1: dog, 2: scoiattolo

In thử hình ảnh

Sau khi đã đọc và chuẩn hóa dữ liệu ta sẽ thực hiện in ngẫu nhiên các hình ảnh của từng lớp con vật

```
import matplotlib.pyplot as plt
classes = ['cats' , 'dogs' , 'scoiattolo']
num_classes = len(classes)
samples_per_class = 3
for y, cls in enumerate(classes):
    idxs = np.flatnonzero(y_train == y)
    idxs = np.random.choice(idxs, samples_per_class, replace=False)
    for i, idx in enumerate(idxs):
        plt_idx = i * num_classes + y + 1
        plt.subplot(samples_per_class, num_classes, plt_idx)
        plt.imshow(X_train[idx].astype('uint8'))
        plt.axis('off')
        if i == 0:
            plt.title(cls)
plt.show()
```

II, Thực hiện việc rút gọn số chiều dữ liệu, sau đó tham khảo để hiển thị trực quan các phân lớp dữ liệu dạng 3D hoặc 2D.

Định hình dữ liệu thành các hàng

```
X_train = X_train.reshape(X_train.shape[0],-1)
X_test = X_test.reshape(X_test.shape[0],-1)
X = np.concatenate ((X_train, X_test), axis = 0)
y = np.concatenate ((y_train, y_test), axis = 0)
```

Ta sẽ chuẩn hóa toàn bộ dữ liệu của X_train và X_test định hình lại các ma trận thành các hàng để dễ xử lý

Sau đó ta ghép toàn bộ X_train, X_test thành X và y_train, y_test thành biến y

B1: Ta chuẩn hóa ma trận X để thu được X_norm:

X_norm = (X - X.min())/(X.max() - X.min())

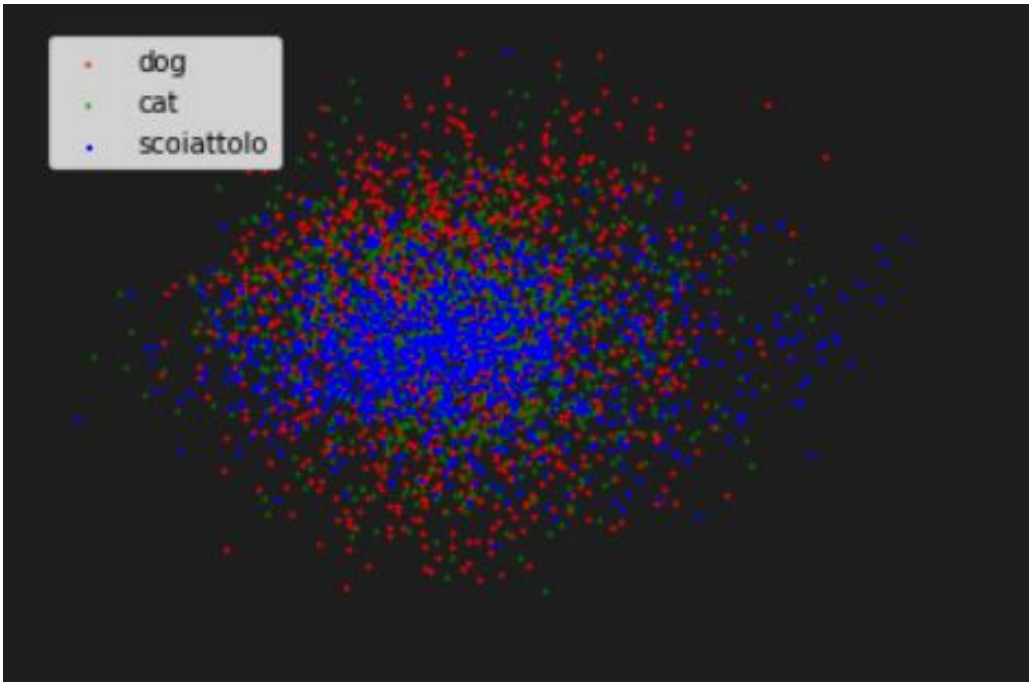
B2: Sử dụng thư viện sklearn để phân tích dữ liệu của ma trận X_norm thành 2 thành phần chính

```
import pandas as pd
from sklearn.decomposition import PCA as sklearnPCA

# Normalize data
X_norm = (X - X.min())/(X.max() - X.min())
```

```
pca = sklearnPCA(n_components=2) #2-dimensional PCA
transformed = pd.DataFrame(pca.fit_transform(X_norm))
plt.axis("off")
plt.scatter(transformed[y==0][0], transformed[y==0][1], s=1, label='dog',
c='red')
plt.scatter(transformed[y==1][0], transformed[y==1][1], s=1, label='cat',
c='green', marker="^")
plt.scatter(transformed[y==2][0], transformed[y==2][1], s=1, label='scoiattolo',
c='blue', marker="s")
plt.legend()
plt.show()
```

B4: Từ đó chúng ta đi vẽ biểu đồ hiển thị phân lớp dữ liệu dưới dạng 2D



Kết luận: Hình ảnh thu được thể hiện khoảng cách giữa các điểm cùng và khác lớp.

III, Xây dựng chương trình sử dụng mô hình Multinomial Logistic Regression (Softmax)

B1: Sử dụng thư viện sklearn gọi đến mô hình LogisticRegression để giải quyết bài toán dựa trên Stochastic Average Gradient

B2: Huấn luyện mô hình dựa vào X_train và y_train

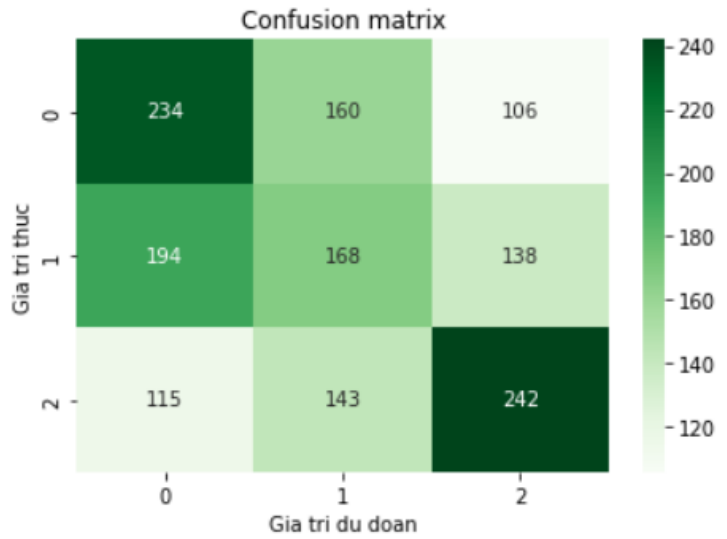
B3: Sau đó đưa ra y_pred là dữ liệu dự đoán dựa trên dữ liệu X_Test

B4: Để kiểm tra mô hình hoạt động có hiệu quả không, ta so sánh dữ liệu thu được ở y_pred với y_test

```
#from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
# Call to Logistic Regression Model - SAG: solving is based on Stochastic Average
Gradient
lorg=LogisticRegression(multi_class='multinomial',solver='sag',max_iter=5000)
# and train model by Training Dataset
lorg.fit(X_train,y_train)
# Then Predict the Test data
y_pred=lorg.predict(X_test)
print(y_pred)
```

B5: Thực hiện đánh tìm accuracy, confusion matrix, recall và precision

```
Accuracy:  0.42933333333333334
```



	precision	recall	f1-score	support
0	0.43	0.47	0.45	500
1	0.36	0.34	0.35	500
2	0.50	0.48	0.49	500
accuracy			0.43	1500
macro avg	0.43	0.43	0.43	1500
weighted avg	0.43	0.43	0.43	1500

Đánh giá :

+ Accuray ~ 43% cho thấy độ chính xác của mô hình khá thấp

+ Cofussion matrix có đường chéo chính không đậm màu, đặc biệt ở vị trí lớp 1x1 màu nhạt và dự đoán sai rất nhiều

+ Precision của từng lớp:

Lớp 0: 0.43 thấp

Lớp 1: 0.36 rất thấp

Lớp 2: 0.5 trung bình

Precsion của từng lớp thấp cho thấy độ chính xác phân loại của từng lớp cũng thấp từ đó đi đến độ chính xác của mô hình cũng thấp.

+Recall của từng lớp:

Lớp 0: 0.47 khá thấp

Lớp 1: 0.34 rất thấp

Lớp 2: 0.48 thấp

Recall của từng lớp thấp cho thấy tỉ lệ bỏ sót các hình ảnh là cao.

IV, Xây dựng chương trình sử dụng mô hình Kernel-Classes Support Vector Machine

Ý tưởng cơ bản của Kernel SVM và các phương pháp kernel là tìm một phép biến đổi sao cho dữ liệu ban đầu là *không phân biệt tuyến tính* được biến sang không gian mới. Ở không gian mới này, dữ liệu trở nên *phân biệt tuyến tính*.
Hàm Kernel thường tạo ra dữ liệu mới có số chiều cao hơn số chiều của
dữ liệu ban đầu.

Có 1 số loại hàm Kernel:

a) Linear:

Đây là trường hợp đơn giản với kernel chính tích vô hướng của hai vector:

$k(x,z) = x^T z$

Sử dụng sklearn.svm.SVC, kernel này được chọn bằng cách đặt kernel = 'linear'.

b) Polynomial:

$k(x,z) = (r + \gamma x^T z)^d$

Với dd là một số dương để chỉ bậc của đa thức. Polynomial kernel có thể dùng để mô tả hầu hết các đa thức có bậc không vượt quá d nếu d là một số tự nhiên.

Sử dụng sklearn.svm.SVC, kerrnel này được chọn bằng cách đặt kernel = 'poly'

c) Radial Basic Function:

RBF được sử dụng nhiều nhất trong thực tế

k(x,z)=exp(−γ||x−z||22), γ>0

Trong sklearn, kernel = 'rbf'

d) Sigmoid:

Sử dụng sklearn.svm.SVC, kerrnel này được chọn bằng cách đặt kernel = 'sigmoid'

Tóm tắt các kernel thông dụng

Tên	Công thức	kernel	Thiết lập hệ số
linear	$\mathbf{x}^T\mathbf{z}$	'linear'	không có hệ số
polynomial	$(r + \gamma\mathbf{x}^T\mathbf{z})^d$	'poly'	d: degree, γ: gamma, r: coef0
sigmoid	$\tanh(\gamma\mathbf{x}^T\mathbf{z} + r)$	'sigmoid'	γ: gamma, r: coef0
rbf	$\exp(-\gamma\|\|\mathbf{x} - \mathbf{z}\|\|_2^2)$	'rbf'	$\gamma > 0$: gamma

Sử dụng mô hình Kernel Support Vector Machine để phân loại hình ảnh

4.1 Sử dụng Kernel Poly

B1: Sử dụng thư viện sklearn gọi đến hàm svm.SVC, sau đó tạo một đối tượng và huấn luyện mô hình dựa vào X_train và y_train

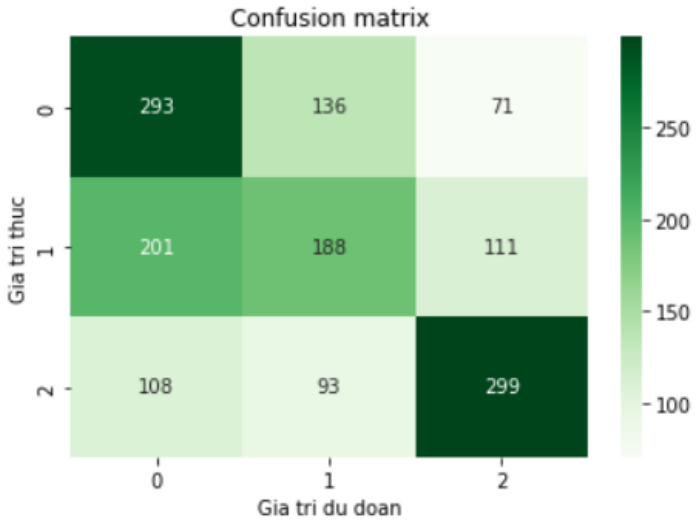
B2: Sau đó đưa ra y_pred là dữ liệu dự đoán dựa trên dữ liệu X_test

B3: Để kiểm tra mô hình hoạt động có hiệu quả không, ta so sánh dữ liệu thu được ở y_pred với y_test

```
from sklearn import svm
M_svm = svm.SVC(kernel='poly', degree=3, C=1).fit(X_train, y_train)
y_pred = M_svm.predict(X_test)
print(y_pred)
```

B4: Thực hiện đánh tìm accuracy, confusion matrix, recall và precision

```
Accuracy:  0.52
```



	precision	recall	f1-score	support
0	0.49	0.59	0.53	500
1	0.45	0.38	0.41	500
2	0.62	0.60	0.61	500
accuracy			0.52	1500
macro avg	0.52	0.52	0.52	1500
weighted avg	0.52	0.52	0.52	1500

Đánh giá :

+ Accuray ~ 52% cho thấy độ chính xác của mô hình ở mức trung bình

+ Cofussion matrix có đường chéo chính không đậm màu, đặc biệt ở vị trí lớp 1x1 màu nhạt và dự đoán sai rất nhiều

+ Precision của từng lớp:

Lớp 0: 0.49 trung bình

Lớp 1: 0.45 thấp

Lớp 2: 0.62 trung bình khá

Precsion của từng lớp trung bình cho thấy độ chính xác phân loại của từng lớp cũng trung bình từ đó đi đến độ chính xác của mô hình cũng trung bình.

+Recall của từng lớp:

Lớp 0: 0.59 trung bình

Lớp 1: 0.38 rất thấp

Lớp 2: 0.6 trung bình khá

Recall của từng lớp trung bình cho thấy tỉ lệ bỏ sót các hình ảnh là trung bình.

4.2, Sử dụng Kernel RBF

B1: Sử dụng thư viện sklearn gọi đến hàm svm.SVC, sau đó tạo một đối tượng và huấn luyện mô hình dựa vào X_train và y_train

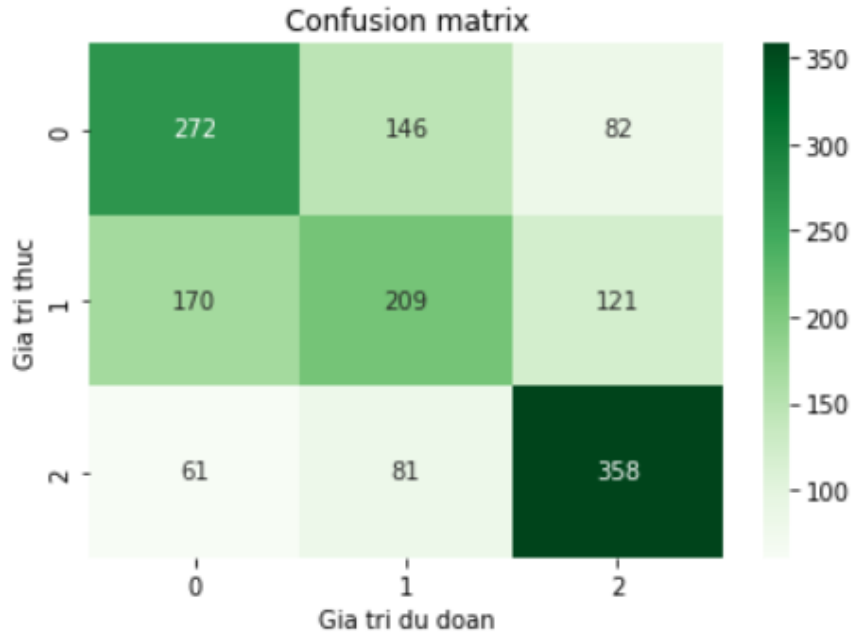
B2: Sau đó đưa ra y_pred là dữ liệu dự đoán dựa trên dữ liệu X_test

B3: Để kiểm tra mô hình hoạt động có hiệu quả không, ta so sánh dữ liệu thu được ở y_pred với y_test

```
from sklearn import svm
M_svm = svm.SVC(kernel='rbf', degree=3, C=1).fit(X_train, y_train)
y_pred = M_svm.predict(X_test)
```

B4: Thực hiện đánh tìm accuracy, confusion matrix, recall và precision

```
Accuracy:  0.5593333333333333
```



	precision	recall	f1-score	support
0	0.54	0.54	0.54	500
1	0.48	0.42	0.45	500
2	0.64	0.72	0.67	500
accuracy			0.56	1500
macro avg	0.55	0.56	0.55	1500
weighted avg	0.55	0.56	0.55	1500

Đánh giá :

+ Accuray ~ 56% cho thấy độ chính xác của mô hình ở mức trung bình

+ Cofussion matrix có đường chéo chính không đậm màu, đặc biệt ở vị trí lớp 1x1 màu nhạt và dự đoán sai rất nhiều

+ Precision của từng lớp:

Lớp 0: 0.54 trung bình

Lớp 1: 0.48 trung bình

Lớp 2: 0.64 trung bình khá

Precsion của từng lớp trung bình cho thấy độ chính xác phân loại của từng lớp cũng trung bình từ đó đi đến độ chính xác của mô hình cũng trung bình.

+Recall của từng lớp:

Lớp 0: 0.54 trung bình

Lớp 1: 0. 42 thấp

Lớp 2: 0.72 trung bình khá

Recall của từng lớp trung bình cho thấy tỉ lệ bỏ sót các hình ảnh là trung bình.

V, So sánh độ chính xác sử dụng accuracy, confusion matrix, recall và precision

Accuracy:

Mô hình sử dụng Multinomial Logistic Regression có độ chính xác ~ 43%

Mô hình sử dụng Kernel Support Vector Machine có độ chính xác ~52-56%

Nhận xét: Với bộ dữ liệu sử dụng thì mô hình sử dụng Kernel SVM cho độ chính xác cao hơn

Confusion matrix:

Mô hình sử dụng Multinomial Logistic Regression có phần đường chéo chính ở vị trí lớp 1x1 nhạt hơn với tỉ lệ đúng chỉ 168/500

Mô hình sử dụng Kernel Support Vector Machine có đường chéo chính ở vị trí đậm hơn ở vị trí ô 1x1 khi sử dụng poly kernel Poly cho tỉ lệ chính xác là 188/500 còn sử dụng kernel RBF cho tỉ lệ chính xác cao hơn là 209/500 ở vị trí lớp 1x1, ở vị lớp 2x2 màu rất đậm độ chính xác là 358/500

Nhận xét: Mô hình sử dụng đường Kernel SVM có đường chéo chính đậm hơn và cho tỉ lệ đúng cao hơn so với Multinomial Logistic Regression.

Precsion:

Mô hình sử dụng Multinomial Logistic Regression có precsion của từng lớp thấp đến trung bình (0.36 - 0.5)

Mô hình Kernel SVM có độ chính xác đối với từng lớp:

+Kernel Poly có precsion của các lớp từ thấp tới trung bình khá (0.45 – 0.62)

+ Kernel RBF có precsion của các lớp từ trung bình tới trung bình khá (0.54 –0.64)

Nhận xét: Mô hình Multinomial Logistic Regression có độ precsion thấp của từng lớp thấp hơn so với Kernel SVM kéo theo độ chính xác của mô hình cũng thấp hơn so với SVM

Recall:

Mô hình sử dụng Multinomial Logistic Regression có recall của từng lớp thấp đến trung bình (0.34 - 0.48)

Mô hình Kernel SVM có độ chính xác đối với từng lớp:

+Kernel Poly có recall của các lớp từ thấp tới trung bình khá (0.38 – 0.6)

+ Kernel RBF có recall của các lớp từ trung bình tới trung bình khá (0.42 –0.72)

Nhận xét: Mô hình sử dụng Kernel SVM có recall cao hơn Multinomial Logistic Regression cho thấy độ bỏ sót ít hơn cho thấy mô hình Kernel SVM tốt hơn

Kết Luận: Qua các thông số trên ta thấy với bộ dữ liệu hiện có Kernel SVM được đánh giá tốt hơn độ chính xác phân loại.