In [1]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import glob
from PIL import Image
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import  LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
#Intialised ImageDataGenerator from tensorflow module to preprocess the image
datagen = ImageDataGenerator(
        rotation_range=15,
        shear_range=0.2,
        horizontal_flip=True,
        featurewise_center=True,
        width_shift_range=0.05,
        height_shift_range=0.05,
        zoom_range=0.1,
        fill_mode='nearest')
```

In [3]:

```
# Path to all images files of dataset - each folder - alphabet contains 100 images
Patha="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/a/*.jpg"
Pathb="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/b/*.jpg"
Pathc="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/c/*.jpg"
Pathd="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/d/*.jpg"
Pathe="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/e/*.jpg"
Pathf="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/f/*.jpg"
Pathg="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/g/*.jpg"
Pathh="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/h/*.jpg"
Pathi="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/i/*.jpg"
Pathj="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/j/*.jpg"
Pathk="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/k/*.jpg"
Pathl="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/l/*.jpg"
Pathm="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/m/*.jpg"
Pathn="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/n/*.jpg"
Patho="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/o/*.jpg"
Pathp="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/p/*.jpg"
```

```
Pathq="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/q/*.jpg"
Pathr="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/r/*.jpg"
Paths="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/s/*.jpg"
Patht="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/t/*.jpg"
Pathu="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/u/*.jpg"
Pathv="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/v/*.jpg"
Pathw="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/w/*.jpg"
Pathx="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/x/*.jpg"
Pathy="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/y/*.jpg"
Pathz="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/train/z/*.jpg"
```

In [4]:

```
# import the data from each alphabet folder and club into final dataframe
def importing_data(path):
    sample = []
    for filename in glob.glob(path):
        img = Image.open(filename,'r')
        img = img.resize((128,128))
        sample.append(img)

    return sample

data_a = importing_data(Patha)
data_b = importing_data(Pathb)
data_c = importing_data(Pathc)
data_d = importing_data(Pathd)
data_e = importing_data(Pathe)
data_f = importing_data(Pathf)
data_g = importing_data(Pathg)
data_h = importing_data(Pathh)
data_i = importing_data(Pathi)
data_j = importing_data(Pathj)
data_k = importing_data(Pathk)
data_l = importing_data(Pathl)
data_m = importing_data(Pathm)
data_n = importing_data(Pathn)
data_o = importing_data(Patho)
data_p = importing_data(Pathp)
data_q = importing_data(Pathq)
data_r = importing_data(Pathr)
data_s = importing_data(Paths)
data_t = importing_data(Patht)
data_u = importing_data(Pathu)
data_v = importing_data(Pathv)
```

```
data_w = importing_data(Pathw)
data_x = importing_data(Pathx)
data_y = importing_data(Pathy)
data_z = importing_data(Pathz)
```

In [5]:

```
def data_import(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z):
    df_data_a = pd.DataFrame({'image':a, 'label': 'a'})
    df_data_b = pd.DataFrame({'image':b, 'label': 'b'})
    df_data_c = pd.DataFrame({'image':c, 'label': 'c'})
    df_data_d = pd.DataFrame({'image':d, 'label': 'd'})
    df_data_e = pd.DataFrame({'image':e, 'label': 'e'})
    df_data_f = pd.DataFrame({'image':f, 'label': 'f'})
    df_data_g = pd.DataFrame({'image':g, 'label': 'g'})
    df_data_h = pd.DataFrame({'image':h, 'label': 'h'})
    df_data_i = pd.DataFrame({'image':i, 'label': 'i'})
    df_data_j = pd.DataFrame({'image':j, 'label': 'j'})
    df_data_k = pd.DataFrame({'image':k, 'label': 'k'})
    df_data_l = pd.DataFrame({'image':l, 'label': 'l'})
    df_data_m = pd.DataFrame({'image':m, 'label': 'm'})
    df_data_n = pd.DataFrame({'image':n, 'label': 'n'})
    df_data_o = pd.DataFrame({'image':o, 'label': 'o'})
    df_data_p = pd.DataFrame({'image':p, 'label': 'p'})
    df_data_q = pd.DataFrame({'image':q, 'label': 'q'})
    df_data_r = pd.DataFrame({'image':r, 'label': 'r'})
    df_data_s = pd.DataFrame({'image':s, 'label': 's'})
    df_data_t = pd.DataFrame({'image':t, 'label': 't'})
    df_data_u = pd.DataFrame({'image':u, 'label': 'u'})
    df_data_v = pd.DataFrame({'image':v, 'label': 'v'})
    df_data_w = pd.DataFrame({'image':w, 'label': 'w'})
    df_data_x = pd.DataFrame({'image':x, 'label': 'x'})
    df_data_y = pd.DataFrame({'image':y, 'label': 'y'})
    df_data_z = pd.DataFrame({'image':z, 'label': 'z'})



    final_data = [df_data_a, df_data_b, df_data_c, df_data_d, df_data_e, df_data_f, df_data_g, df_data_h, df_data_i, df_
    final_data = pd.concat(final_data)
    all_data = final_data['image']
    labels = final_data['label']
    all_data=np.stack(all_data,axis=0)
    labels = LabelBinarizer().fit_transform(labels)
    return all_data,labels
```

```
dataset,labels   = data_import(data_a,data_b,data_c,data_d,data_e,data_f,data_g,data_h,data_i,data_j,data_k,data_l,data_
```

In [6]:

```
dataset.shape
```

Out[6]:

```
(2600, 128, 128)
```

In [7]:

```
dataset=dataset.reshape(2600,128,128,1)
```

In [8]:

```
#augmentation method while preprocessing images
def augmentation(dataset,labels,counts):
    augs=[]
    augs_data=[]
    augs_labels=[]
    i=0
    for batch in datagen.flow(dataset,labels,
                              batch_size=260):
        i += 1
        augs.append(batch)
        if i>counts:
            break
    augs_data=[]
    for i in range(counts):
        data=augs[i][0]
        augs_data.append(data)
    x = np.vstack(augs_data)
    x = x/255.0

    for i in range(counts):
        label=augs[i][1]
        augs_labels.append(label)
    y = np.vstack(augs_labels)
    return x, y
```

In [9]:

```
#augment all the images in dataset
x,y = augmentation(dataset,labels,15)
y = np.where(y==1)[1]

from tensorflow.keras.utils import to_categorical
y_onehot = to_categorical(y)
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y_onehot,test_size=0.26,random_state=2021)
```

In [10]:

```
#preprocess and train the data with cnn model layering
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Input, MaxPooling2D, Conv2D, Dropout
from tensorflow.keras import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

model = Sequential()
model.add(Conv2D(12, (5, 5), activation='relu', padding='same', input_shape=(128, 128, 1)))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(24, (5, 5), activation='relu', padding='same',
                 kernel_regularizer =tf.keras.regularizers.l1(l=0.01)))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(36, (5, 5), activation='relu', padding='same',
                 kernel_regularizer =tf.keras.regularizers.l2(l=0.01)))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.2))


model.add(Conv2D(48, (5, 5), activation='relu',
                 kernel_regularizer =tf.keras.regularizers.l2(l=0.01)))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Flatten())

m = model.output
m = Dense(120, activation = "relu")(m)
m = Dense(100, activation = "relu")(m)
m = Dropout(0.2)(m)
m = Dense(60, activation = "relu")(m)
m = Dense(60, activation = "relu")(m)
m = Dropout(0.2)(m)
final_layer =  Dense(26, activation = "softmax")(m)


cnn_model = Model(inputs=model.input, outputs=final_layer)
```

```
cnn_model.summary()

cnn_model.compile(optimizer=Adam(learning_rate=0.0001),loss='binary_crossentropy',metrics=['accuracy'])

callback = EarlyStopping(monitor='val_loss',patience=5,
                         restore_best_weights=True)

history=cnn_model.fit(x_train, y_train, validation_split=0.2,
                  epochs=50 , batch_size=64,callbacks=[callback])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_input (InputLayer)   [(None, 128, 128, 1)]     0         
                                                                 
 conv2d (Conv2D)             (None, 128, 128, 12)      312       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 64, 64, 12)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 64, 64, 24)        7224      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 32, 32, 24)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 32, 32, 36)        21636     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 16, 16, 36)       0         
 2D)                                                             
                                                                 
 dropout (Dropout)           (None, 16, 16, 36)        0         
                                                                 
 conv2d_3 (Conv2D)           (None, 12, 12, 48)        43248     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 6, 6, 48)         0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 6, 6, 48)          0         
                                                                 
 flatten (Flatten)           (None, 1728)              0         
                                                                 
 dense (Dense)               (None, 120)               207480    
                                                                 
 dense_1 (Dense)             (None, 100)               12100     
                                                                 
 dropout_2 (Dropout)         (None, 100)               0         
                                                                 
 dense_2 (Dense)             (None, 60)                6060      
                                                                 
 dense_3 (Dense)             (None, 60)                3660      
                                                                 
 dropout_3 (Dropout)         (None, 60)                0         
 dense_4 (Dense)             (None, 26)                1586      
                                                                 
=================================================================
Total params: 303,306
Trainable params: 303,306
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
37/37 [==============================] - 17s 420ms/step - loss: 4.1134 - accuracy: 0.0373 - val_loss: 3.8452 - val_accu
racy: 0.0311
Epoch 2/50
37/37 [==============================] - 17s 466ms/step - loss: 3.5973 - accuracy: 0.0368 - val_loss: 3.2676 - val_accu
racy: 0.0311
Epoch 3/50
37/37 [==============================] - 17s 457ms/step - loss: 3.1253 - accuracy: 0.0403 - val_loss: 2.8157 - val_accu
racy: 0.0519
Epoch 4/50
37/37 [==============================] - 17s 465ms/step - loss: 2.7500 - accuracy: 0.0295 - val_loss: 2.4989 - val_accu
racy: 0.0519
Epoch 5/50
37/37 [==============================] - 18s 474ms/step - loss: 2.4381 - accuracy: 0.0399 - val_loss: 2.2179 - val_accu
racy: 0.0519
Epoch 6/50
37/37 [==============================] - 18s 488ms/step - loss: 2.1603 - accuracy: 0.0334 - val_loss: 1.9624 - val_accu
racy: 0.0519
Epoch 7/50
37/37 [==============================] - 20s 530ms/step - loss: 1.9098 - accuracy: 0.0390 - val_loss: 1.7285 - val_accu
racy: 0.0519
Epoch 8/50
37/37 [==============================] - 17s 448ms/step - loss: 1.6815 - accuracy: 0.0381 - val_loss: 1.5152 - val_accu
racy: 0.0519
Epoch 9/50
37/37 [==============================] - 16s 445ms/step - loss: 1.4773 - accuracy: 0.0429 - val_loss: 1.3240 - val_accu
racy: 0.0519
Epoch 10/50
37/37 [==============================] - 16s 443ms/step - loss: 1.2881 - accuracy: 0.0373 - val_loss: 1.1482 - val_accu
racy: 0.0519
Epoch 11/50
37/37 [==============================] - 16s 442ms/step - loss: 1.1179 - accuracy: 0.0377 - val_loss: 0.9888 - val_accu
racy: 0.0311
Epoch 12/50
37/37 [==============================] - 16s 442ms/step - loss: 0.9609 - accuracy: 0.0368 - val_loss: 0.8446 - val_accu
```

```
racy: 0.0311
Epoch 13/50
37/37 [==============================] - 16s 444ms/step - loss: 0.8221 - accuracy: 0.0373 - val_loss: 0.7162 - val_accu
racy: 0.0311
Epoch 14/50
37/37 [==============================] - 16s 444ms/step - loss: 0.6984 - accuracy: 0.0390 - val_loss: 0.6036 - val_accu
racy: 0.0311
Epoch 15/50
37/37 [==============================] - 17s 449ms/step - loss: 0.5899 - accuracy: 0.0364 - val_loss: 0.5056 - val_accu
racy: 0.0311
Epoch 16/50
37/37 [==============================] - 16s 445ms/step - loss: 0.4970 - accuracy: 0.0360 - val_loss: 0.4211 - val_accu
racy: 0.0311
Epoch 17/50
37/37 [==============================] - 17s 463ms/step - loss: 0.4157 - accuracy: 0.0390 - val_loss: 0.3485 - val_accu
racy: 0.0311
Epoch 18/50
37/37 [==============================] - 17s 451ms/step - loss: 0.3502 - accuracy: 0.0368 - val_loss: 0.2895 - val_accu
racy: 0.0311
Epoch 19/50
37/37 [==============================] - 16s 446ms/step - loss: 0.2966 - accuracy: 0.0342 - val_loss: 0.2436 - val_accu
racy: 0.0311
Epoch 20/50
37/37 [==============================] - 16s 446ms/step - loss: 0.2546 - accuracy: 0.0412 - val_loss: 0.2099 - val_accu
racy: 0.0311
Epoch 21/50
37/37 [==============================] - 17s 447ms/step - loss: 0.2256 - accuracy: 0.0425 - val_loss: 0.1883 - val_accu
racy: 0.0311
Epoch 22/50
37/37 [==============================] - 17s 454ms/step - loss: 0.2080 - accuracy: 0.0355 - val_loss: 0.1782 - val_accu
racy: 0.0311
Epoch 23/50
37/37 [==============================] - 17s 448ms/step - loss: 0.2011 - accuracy: 0.0373 - val_loss: 0.1748 - val_accu
racy: 0.0311
Epoch 24/50
37/37 [==============================] - 17s 449ms/step - loss: 0.1978 - accuracy: 0.0412 - val_loss: 0.1730 - val_accu
racy: 0.0311
Epoch 25/50
37/37 [==============================] - 17s 450ms/step - loss: 0.1952 - accuracy: 0.0364 - val_loss: 0.1725 - val_accu
racy: 0.0311
Epoch 26/50
37/37 [==============================] - 17s 449ms/step - loss: 0.1932 - accuracy: 0.0377 - val_loss: 0.1705 - val_accu
racy: 0.0311
```

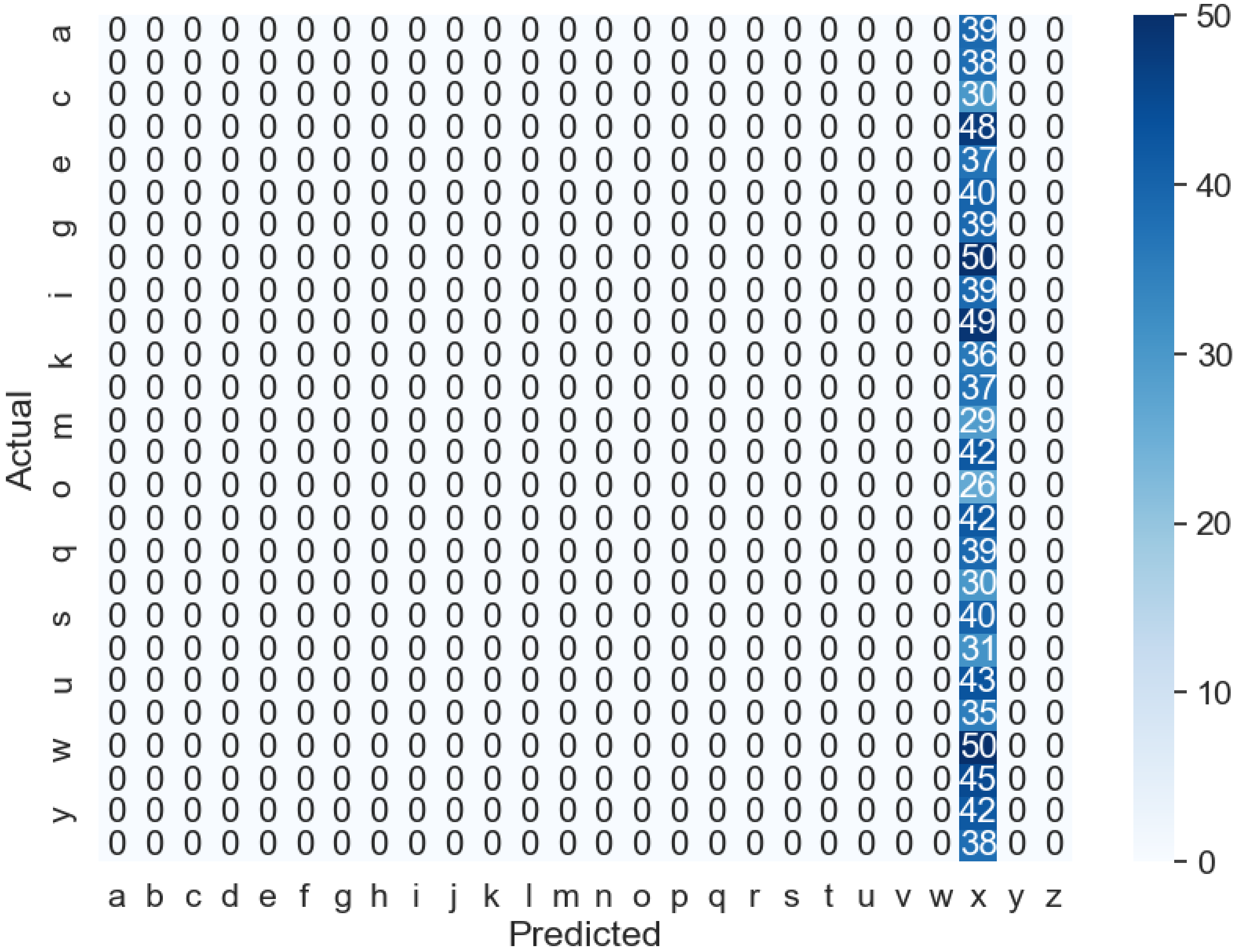
```
Epoch 27/50
37/37 [==============================] - 17s 452ms/step - loss: 0.1907 - accuracy: 0.0433 - val_loss: 0.1690 - val_accu
racy: 0.0311
Epoch 28/50
37/37 [==============================] - 17s 453ms/step - loss: 0.1884 - accuracy: 0.0412 - val_loss: 0.1684 - val_accu
racy: 0.0311
Epoch 29/50
37/37 [==============================] - 17s 457ms/step - loss: 0.1883 - accuracy: 0.0403 - val_loss: 0.1678 - val_accu
racy: 0.0311
Epoch 30/50
37/37 [==============================] - 17s 458ms/step - loss: 0.1871 - accuracy: 0.0351 - val_loss: 0.1670 - val_accu
racy: 0.0311
Epoch 31/50
37/37 [==============================] - 17s 455ms/step - loss: 0.1863 - accuracy: 0.0403 - val_loss: 0.1666 - val_accu
racy: 0.0311
Epoch 32/50
37/37 [==============================] - 17s 465ms/step - loss: 0.1862 - accuracy: 0.0351 - val_loss: 0.1663 - val_accu
racy: 0.0311
Epoch 33/50
37/37 [==============================] - 17s 457ms/step - loss: 0.1853 - accuracy: 0.0459 - val_loss: 0.1661 - val_accu
racy: 0.0311
Epoch 34/50
37/37 [==============================] - 17s 465ms/step - loss: 0.1834 - accuracy: 0.0416 - val_loss: 0.1658 - val_accu
racy: 0.0311
Epoch 35/50
37/37 [==============================] - 18s 475ms/step - loss: 0.1845 - accuracy: 0.0381 - val_loss: 0.1655 - val_accu
racy: 0.0311
Epoch 36/50
37/37 [==============================] - 17s 463ms/step - loss: 0.1836 - accuracy: 0.0407 - val_loss: 0.1654 - val_accu
racy: 0.0311
Epoch 37/50
37/37 [==============================] - 17s 456ms/step - loss: 0.1826 - accuracy: 0.0420 - val_loss: 0.1652 - val_accu
racy: 0.0311
Epoch 38/50
37/37 [==============================] - 17s 453ms/step - loss: 0.1811 - accuracy: 0.0403 - val_loss: 0.1651 - val_accu
racy: 0.0311
Epoch 39/50
37/37 [==============================] - 17s 460ms/step - loss: 0.1823 - accuracy: 0.0403 - val_loss: 0.1649 - val_accu
racy: 0.0311
Epoch 40/50
37/37 [==============================] - 17s 457ms/step - loss: 0.1813 - accuracy: 0.0394 - val_loss: 0.1654 - val_accu
racy: 0.0311
Epoch 41/50
```

```
37/37 [==============================] - 17s 455ms/step - loss: 0.1803 - accuracy: 0.0368 - val_loss: 0.1649 - val_accu
racy: 0.0311
Epoch 42/50
37/37 [==============================] - 17s 458ms/step - loss: 0.1798 - accuracy: 0.0329 - val_loss: 0.1648 - val_accu
racy: 0.0311
Epoch 43/50
37/37 [==============================] - 17s 460ms/step - loss: 0.1806 - accuracy: 0.0364 - val_loss: 0.1647 - val_accu
racy: 0.0277
Epoch 44/50
37/37 [==============================] - 17s 462ms/step - loss: 0.1808 - accuracy: 0.0338 - val_loss: 0.1647 - val_accu
racy: 0.0311
Epoch 45/50
37/37 [==============================] - 17s 459ms/step - loss: 0.1794 - accuracy: 0.0312 - val_loss: 0.1646 - val_accu
racy: 0.0311
Epoch 46/50
37/37 [==============================] - 193s 5s/step - loss: 0.1800 - accuracy: 0.0377 - val_loss: 0.1647 - val_accura
cy: 0.0311
Epoch 47/50
37/37 [==============================] - 15s 399ms/step - loss: 0.1788 - accuracy: 0.0412 - val_loss: 0.1646 - val_accu
racy: 0.0311
Epoch 48/50
37/37 [==============================] - 15s 416ms/step - loss: 0.1791 - accuracy: 0.0386 - val_loss: 0.1645 - val_accu
racy: 0.0311
Epoch 49/50
37/37 [==============================] - 16s 441ms/step - loss: 0.1790 - accuracy: 0.0347 - val_loss: 0.1646 - val_accu
racy: 0.0311
Epoch 50/50
37/37 [==============================] - 16s 439ms/step - loss: 0.1779 - accuracy: 0.0368 - val_loss: 0.1645 - val_accu
racy: 0.0311
```

In [11]:

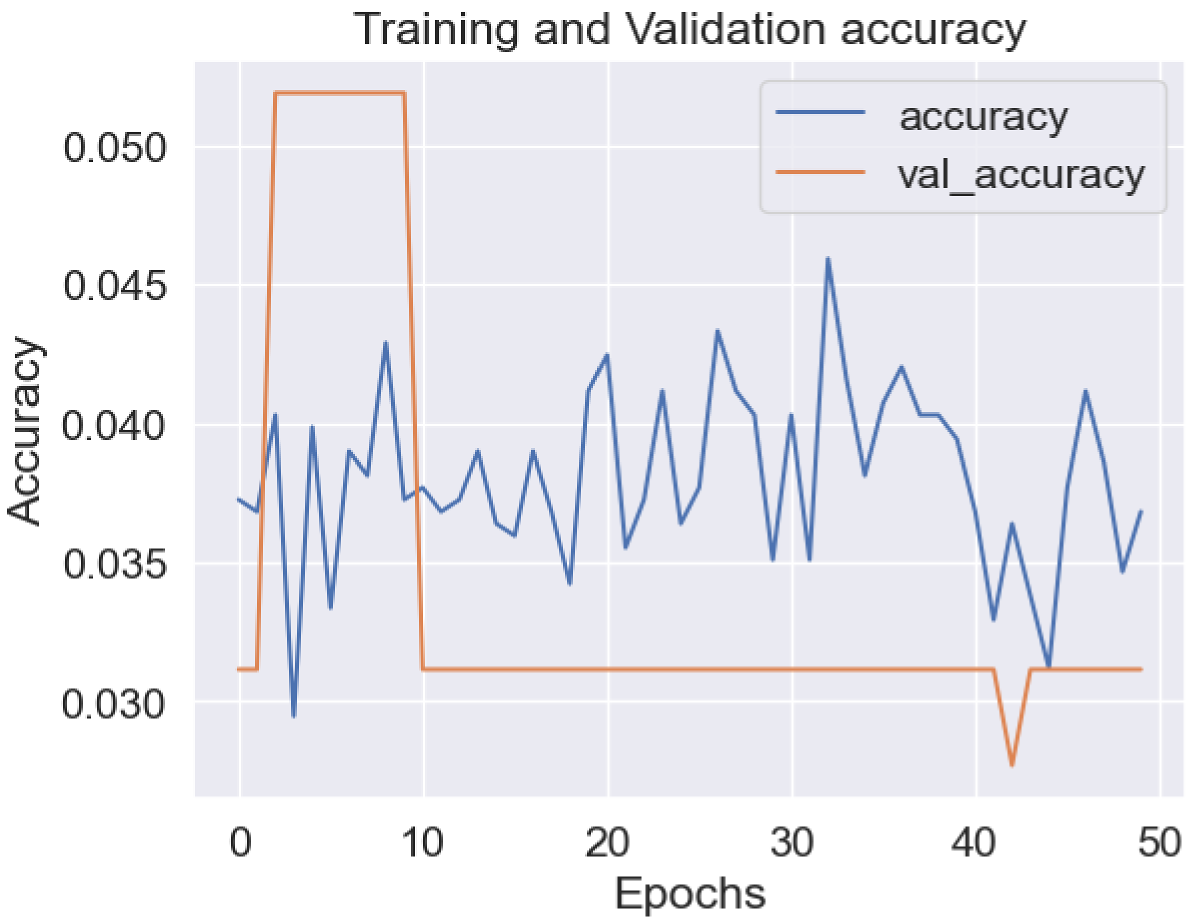
```
#get training data confusion matrix and classification report
import seaborn as sn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

predictions = cnn_model.predict(x_test)
pred_labels = np.argmax(predictions, axis = 1)
tests=np.argmax(y_test,axis=1)

conf_mx = confusion_matrix(tests, pred_labels)
conf_mx

heat_cm = pd.DataFrame(conf_mx, columns=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t
                                  "w","x","y","z"), index =("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o"
                                                                       ,"p","q","r","s","t","u","v","w","x","y","z"))
heat_cm.index.name = 'Actual'
heat_cm.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)
sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')
plt.show()


print(classification_report(tests, pred_labels))

history_df = pd.DataFrame(history.history)

plt.plot(history_df.loc[:, ['accuracy']], label='accuracy')
plt.plot(history_df.loc[:, ['val_accuracy']], label='val_accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(history_df.loc[:, ['loss']], label='loss')
plt.plot(history_df.loc[:, ['val_loss']], label='val_loss')

plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

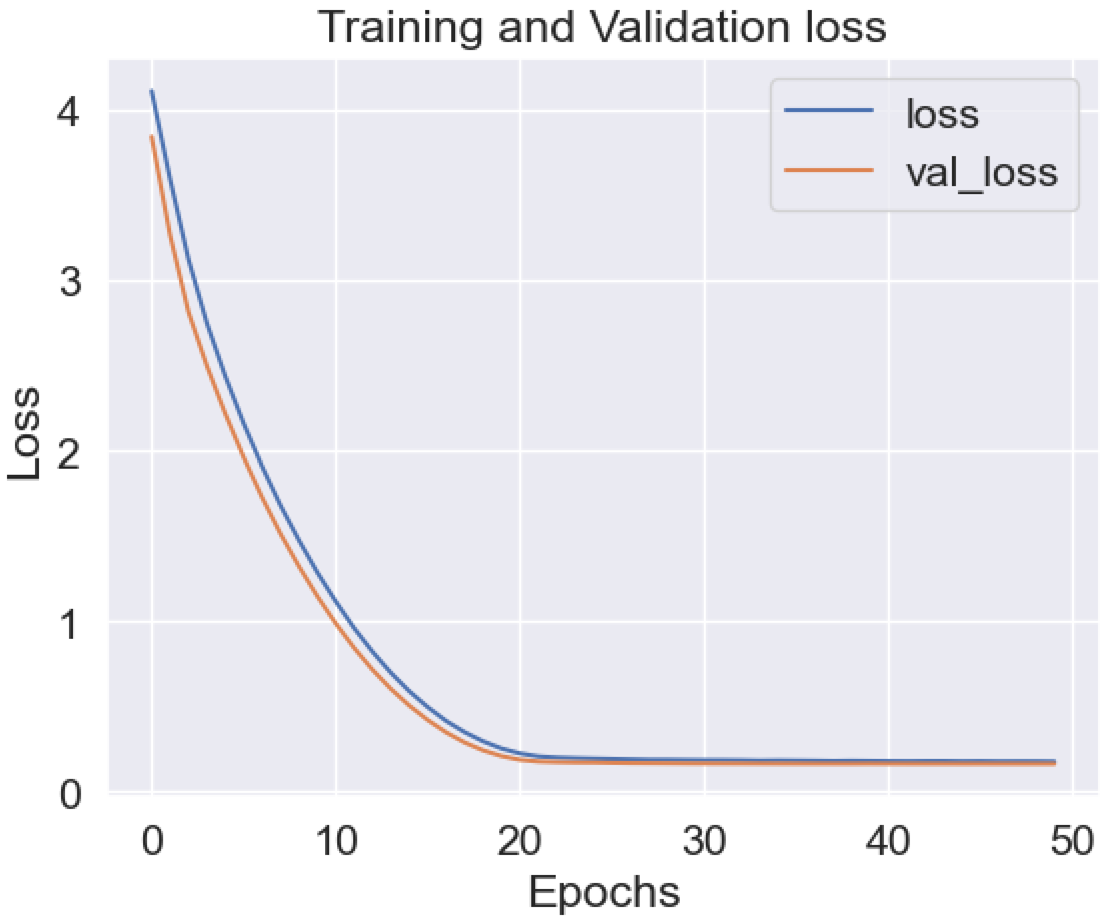
```
32/32 [==============================] - 2s 53ms/step
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0
g	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49	0	0
k	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        39
           1       0.00      0.00      0.00        38
           2       0.00      0.00      0.00        30
           3       0.00      0.00      0.00        48
           4       0.00      0.00      0.00        37
           5       0.00      0.00      0.00        40
           6       0.00      0.00      0.00        39
           7       0.00      0.00      0.00        50
           8       0.00      0.00      0.00        39
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        36
          11       0.00      0.00      0.00        37
          12       0.00      0.00      0.00        29
          13       0.00      0.00      0.00        42
          14       0.00      0.00      0.00        26
          15       0.00      0.00      0.00        42
          16       0.00      0.00      0.00        39
          17       0.00      0.00      0.00        30
          18       0.00      0.00      0.00        40
          19       0.00      0.00      0.00        31
          20       0.00      0.00      0.00        43
          21       0.00      0.00      0.00        35
          22       0.00      0.00      0.00        50
          23       0.04      1.00      0.08        45
          24       0.00      0.00      0.00        42
          25       0.00      0.00      0.00        38

    accuracy                           0.04      1014
   macro avg       0.00      0.04      0.00      1014
weighted avg       0.00      0.04      0.00      1014
```

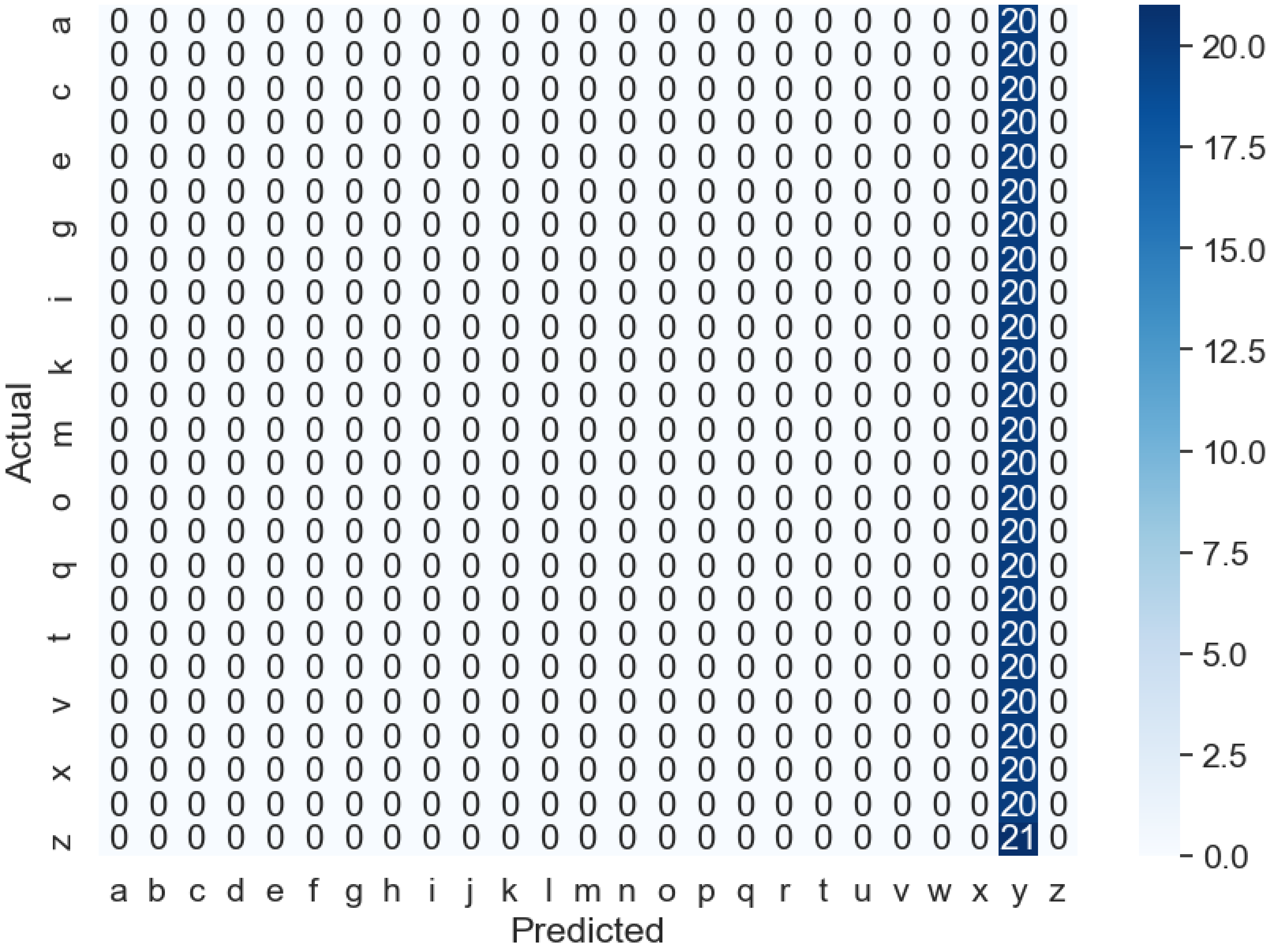

Training and Validation accuracy
accuracy
val_accuracy
Accuracy
0.050
0.045
0.040
0.035
0.030
0
10
20
30
40
50
Epochs



In [12]:
```
# all the test data is loaded in test folder, were imported and clubbed into final df
externala="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/a/*.jpg"
externalb="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/b/*.jpg"
externalc="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/c/*.jpg"
externald="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/d/*.jpg"
externale="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/e/*.jpg"
externalf="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/f/*.jpg"
externalg="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/g/*.jpg"
externalh="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/h/*.jpg"
externali="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/i/*.jpg"
externalj="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/j/*.jpg"
externalk="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/k/*.jpg"
externall="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/l/*.jpg"
externalm="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/m/*.jpg"
```

```
externaln="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/n/*.jpg"
externalo="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/o/*.jpg"
externalp="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/p/*.jpg"
externalq="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/q/*.jpg"
externalr="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/r/*.jpg"
externals="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/s/*.jpg"
externalt="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/t/*.jpg"
externalu="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/u/*.jpg"
externalv="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/v/*.jpg"
externalw="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/w/*.jpg"
externalx="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/x/*.jpg"
externaly="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/y/*.jpg"
externalz="C:/Users/thipp/Fall2022/CS5710_13469/Sign Language for Alphabets/test/z/*.jpg"

external_test_a = importing_data(externala)
external_test_b = importing_data(externalb)
external_test_c = importing_data(externalc)
external_test_d= importing_data(externald)
external_test_e= importing_data(externale)
external_test_f= importing_data(externalf)
external_test_g= importing_data(externalg)
external_test_h= importing_data(externalh)
external_test_i= importing_data(externali)
external_test_j= importing_data(externalj)
external_test_k= importing_data(externalk)
external_test_l= importing_data(externall)
external_test_m= importing_data(externalm)
external_test_n= importing_data(externaln)
external_test_o= importing_data(externalo)
external_test_p= importing_data(externalp)
external_test_q= importing_data(externalq)
external_test_r= importing_data(externalr)
external_test_s= importing_data(externals)
external_test_t= importing_data(externalt)
external_test_u= importing_data(externalu)
external_test_v= importing_data(externalv)
external_test_w= importing_data(externalw)
external_test_x= importing_data(externalx)
external_test_y= importing_data(externaly)
external_test_z= importing_data(externalz)


external_test_data,external_test_label   = data_import(external_test_a,external_test_b,external_test_c,external_test_d,
                                                       external_test_e,external_test_f,external_test_g,external_test_h,
                                                       external_test_i,external_test_j,external_test_k,external_test_l,
                                                       external_test_m,external_test_n,external_test_o,external_test_p,
                                                       external_test_q,external_test_r,external_test_s,external_test_t,
                                                       external_test_u,external_test_v,external_test_w,external_test_x,
                                                      external_test_y,external_test_z)

print(external_test_data.shape)
external_test_data=external_test_data.reshape(501,128,128,1)
external_test_data = external_test_data /255.0

external_test_pred = cnn_model.predict(external_test_data)
external_pred_labels = np.argmax(external_test_pred, axis = 1)
external_tests=np.argmax(external_test_label,axis=1)


external_conf_mx = confusion_matrix(external_tests, external_pred_labels)
heat_cm = pd.DataFrame(external_conf_mx,
                       columns=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","t","u","v","w"
                       ,index =("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","t","u","v","w"
heat_cm.index.name = 'Actual'
heat_cm.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)
sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')
plt.show()
```

```
(501, 128, 128)
16/16 [==============================] - 1s 44ms/step
```



In [13]:

```
x_for_ml = model.predict(x_train)
x_test_ml = model.predict(x_test)
y_train_ml = np.where(y_train==1)[1]
y_test_ml = np.where(y_test==1)[1]
```

```
external_data_ml = model.predict(external_test_data)
external_label_ml = np.where(external_test_label==1)[1]
```

```
91/91 [==============================] - 5s 51ms/step
32/32 [==============================] - 2s 54ms/step
16/16 [==============================] - 1s 60ms/step
```

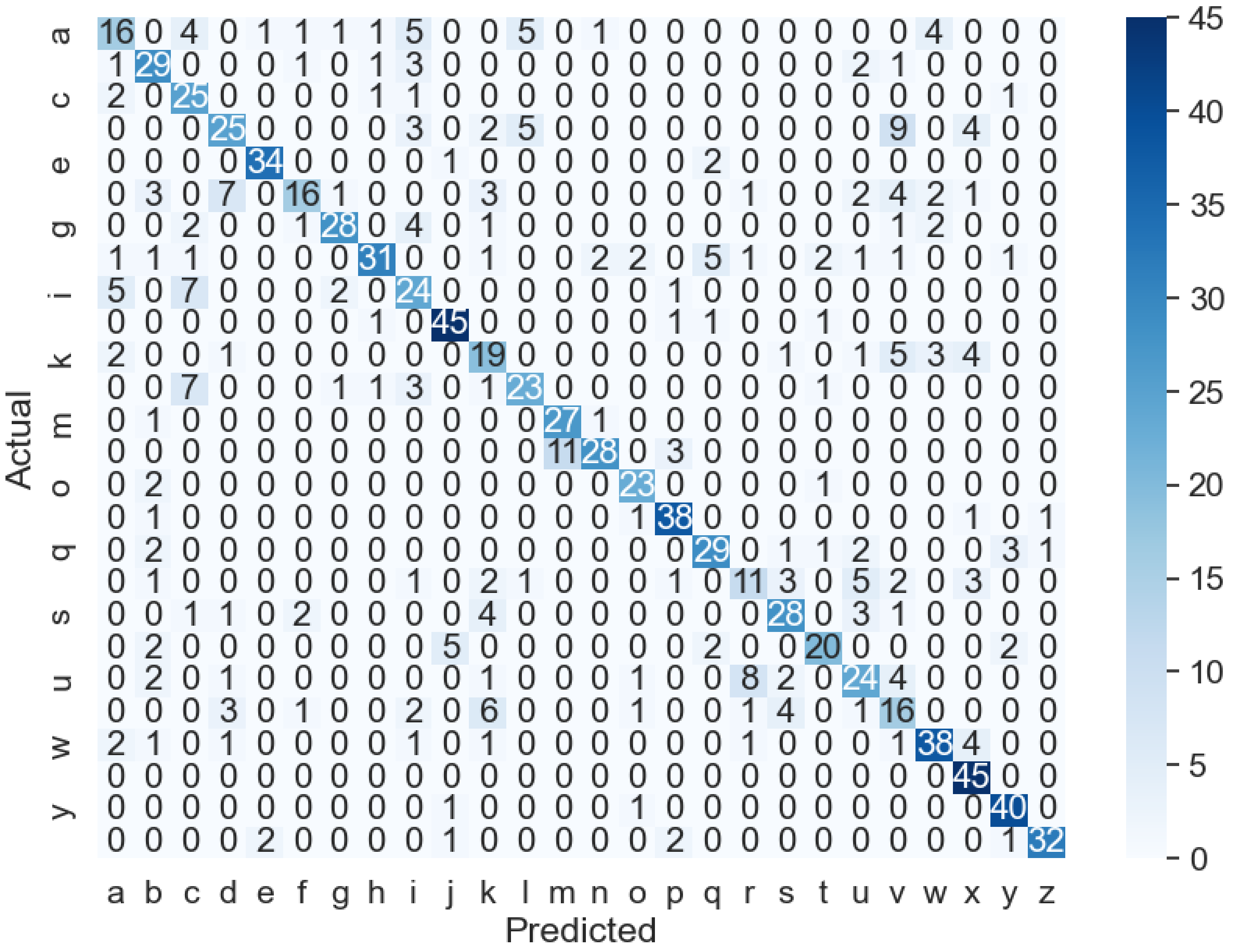
In [14]:
```
#common method to execute various machine learning algorithms and their performances
def machine_learning(algorithm):
    algorithm.fit(x_for_ml, y_train_ml)

    prediction_algo = algorithm.predict(x_test_ml)
    cm_algo = confusion_matrix(y_test_ml, prediction_algo)
    heat_cm = pd.DataFrame(cm_algo, columns=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p",
                                             "q","r","s","t","u","v","w","x","y","z"),
                                             index=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p"
                                                    ,"q","r","s","t","u","v","w","x","y","z"))
    heat_cm.index.name = 'Actual'
    heat_cm.columns.name = 'Predicted'
    plt.figure(figsize = (10,7))
    sn.set(font_scale=1.4)
    sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')
    cm_algo=plt.show()

    report = print(classification_report(y_test_ml, prediction_algo))
    algo_accuracy =  accuracy_score(y_test_ml, prediction_algo)

    prediction_self_algo = algorithm.predict(external_data_ml)
    cm_self_algo = confusion_matrix(external_label_ml, prediction_self_algo)
    algo_test_accuracy = accuracy_score(external_label_ml, prediction_self_algo)
    heat_cm = pd.DataFrame(cm_self_algo, columns=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","
                           index =("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u",
    heat_cm.index.name = 'Actual'
    heat_cm.columns.name = 'Predicted'
    plt.figure(figsize = (10,7))
    sn.set(font_scale=1.4)
    sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')
    cm_self_algo=plt.show()
    return cm_algo, report, cm_self_algo, algo_accuracy, algo_test_accuracy
```

In [15]:
```
def machine_learning1(algorithm):
    #fit the training data into ML algorithm
    algorithm.fit(x_for_ml, y_train_ml)
    #Predict
    prediction_algo = algorithm.predict(x_test_ml)
    #confustion matrix
    cm_algo = confusion_matrix(y_test_ml, prediction_algo)
    #plotting the confusion matrix
    heat_cm = pd.DataFrame(cm_algo, columns=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p",
                                             "q","r","s","t","u","v","w","x","y","z"),
                                             index=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p"
                                                    ,"q","r","s","t","u","v","w","x","y","z"))
    heat_cm.index.name = 'Actual'
    heat_cm.columns.name = 'Predicted'
    plt.figure(figsize = (10,7))
    sn.set(font_scale=1.4)
    sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')
    cm_algo=plt.show()
    #classification report of training data
    report = print(classification_report(y_test_ml, prediction_algo))
    #accuracy of training data
    algo_accuracy =  accuracy_score(y_test_ml, prediction_algo)

    #predicting the test dataset
    prediction_self_algo = algorithm.predict(external_data_ml)
    #test dataset confusion matrix
    cm_self_algo = confusion_matrix(external_label_ml, prediction_self_algo)
    #test dataset accuracy
    algo_test_accuracy = accuracy_score(external_label_ml, prediction_self_algo)
    heat_cm = pd.DataFrame(cm_self_algo, columns=("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","
                           index =("a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","t","u","v",
    heat_cm.index.name = 'Actual'
    heat_cm.columns.name = 'Predicted'
    plt.figure(figsize = (10,7))
    sn.set(font_scale=1.4)
    sn.heatmap(heat_cm, cmap="Blues", annot=True, annot_kws={"size": 16},fmt='g')
    cm_self_algo=plt.show()
    return cm_algo, report, cm_self_algo, algo_accuracy, algo_test_accuracy
```

In [16]:

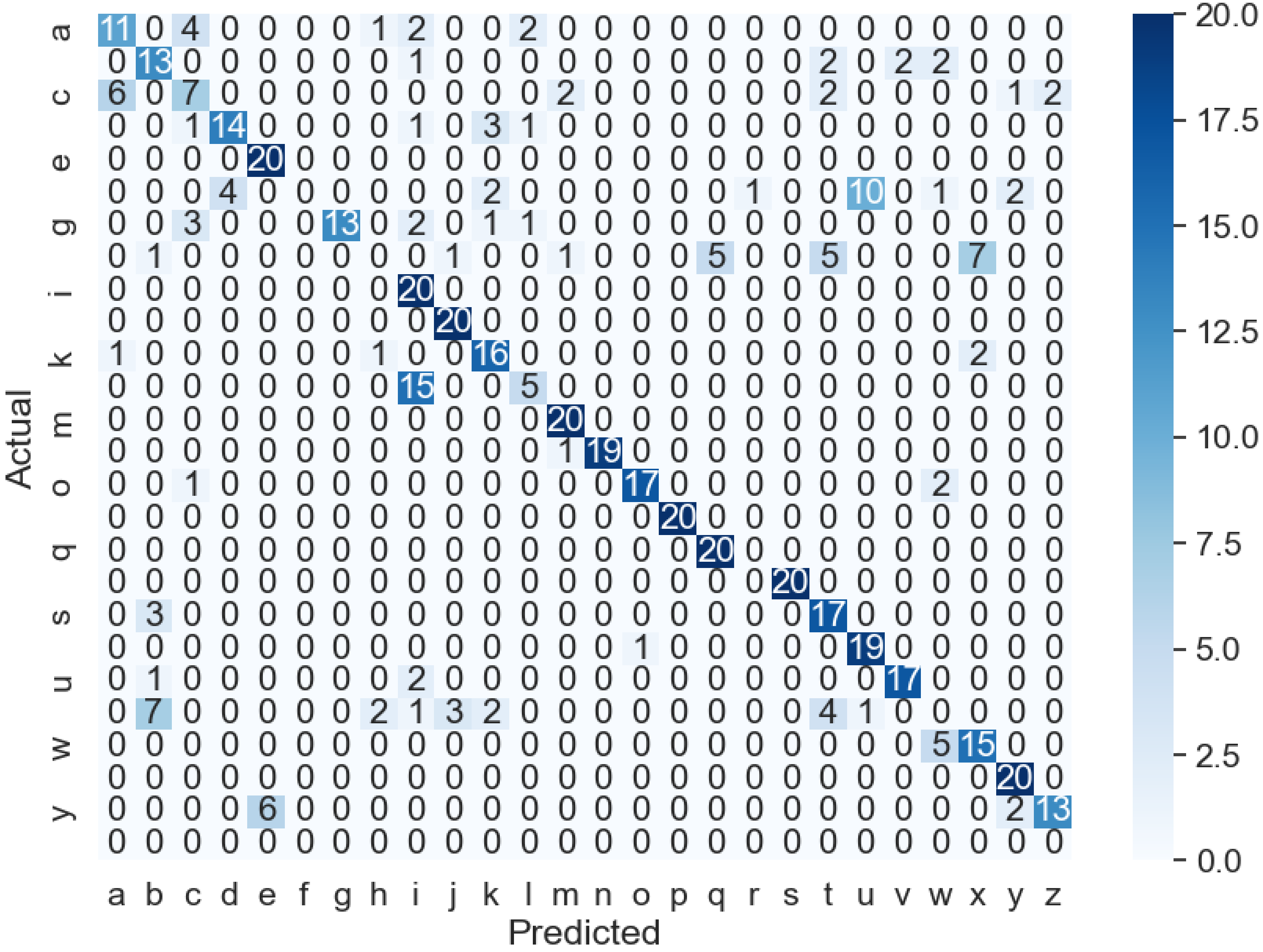
```
#XgBoost Algorithm
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
xgb = XGBClassifier()

xgboost = XGBClassifier(n_estimators=50,learning_rate=0.1,
                        max_depth=2,reg_lambda=0.1)
```

```
xgb_cm, xgb_report, xgb_external_cm, xgb_acc, xgb_ext_acc = machine_learning(xgboost)
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	16	0	4	0	1	1	1	1	5	0	0	5	0	1	0	0	0	0	0	0	0	0	4	0	0	0
	1	29	0	0	0	1	0	1	3	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0
c	2	0	25	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	0	0	0	25	0	0	0	0	3	0	2	5	0	0	0	0	0	0	0	0	0	9	0	4	0	0
e	0	0	0	0	34	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
	0	3	0	7	0	16	1	0	0	0	3	0	0	0	0	0	0	1	0	0	2	4	2	1	0	0
g	0	0	2	0	0	1	28	0	4	0	1	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0
	1	1	1	0	0	0	0	31	0	0	1	0	0	2	2	0	5	1	0	2	1	1	0	0	1	0
i	5	0	7	0	0	0	2	0	24	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	45	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0
k	2	0	0	1	0	0	0	0	0	0	19	0	0	0	0	0	0	0	1	0	1	5	3	4	0	0
	0	0	7	0	0	0	1	1	3	0	1	23	0	0	0	0	0	0	0	1	0	0	0	0	0	0
m	0	1	0	0	0	0	0	0	0	0	0	0	27	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	11	28	0	3	0	0	0	0	0	0	0	0	0	0
o	0	2	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	1	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	38	0	0	0	0	0	0	0	1	0	1
q	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	1	1	2	0	0	0	3	1
	0	1	0	0	0	0	0	0	1	0	2	1	0	0	0	1	0	11	3	0	5	2	0	3	0	0
s	0	0	1	1	0	2	0	0	0	0	4	0	0	0	0	0	0	0	28	0	3	1	0	0	0	0
	0	2	0	0	0	0	0	0	0	5	0	0	0	0	0	0	2	0	0	20	0	0	0	0	2	0
u	0	2	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	8	2	0	24	4	0	0	0	0
	0	0	0	3	0	1	0	0	2	0	6	0	0	0	1	0	0	1	4	0	1	16	0	0	0	0
w	2	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	38	4	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	0
y	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	40	0
	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	32

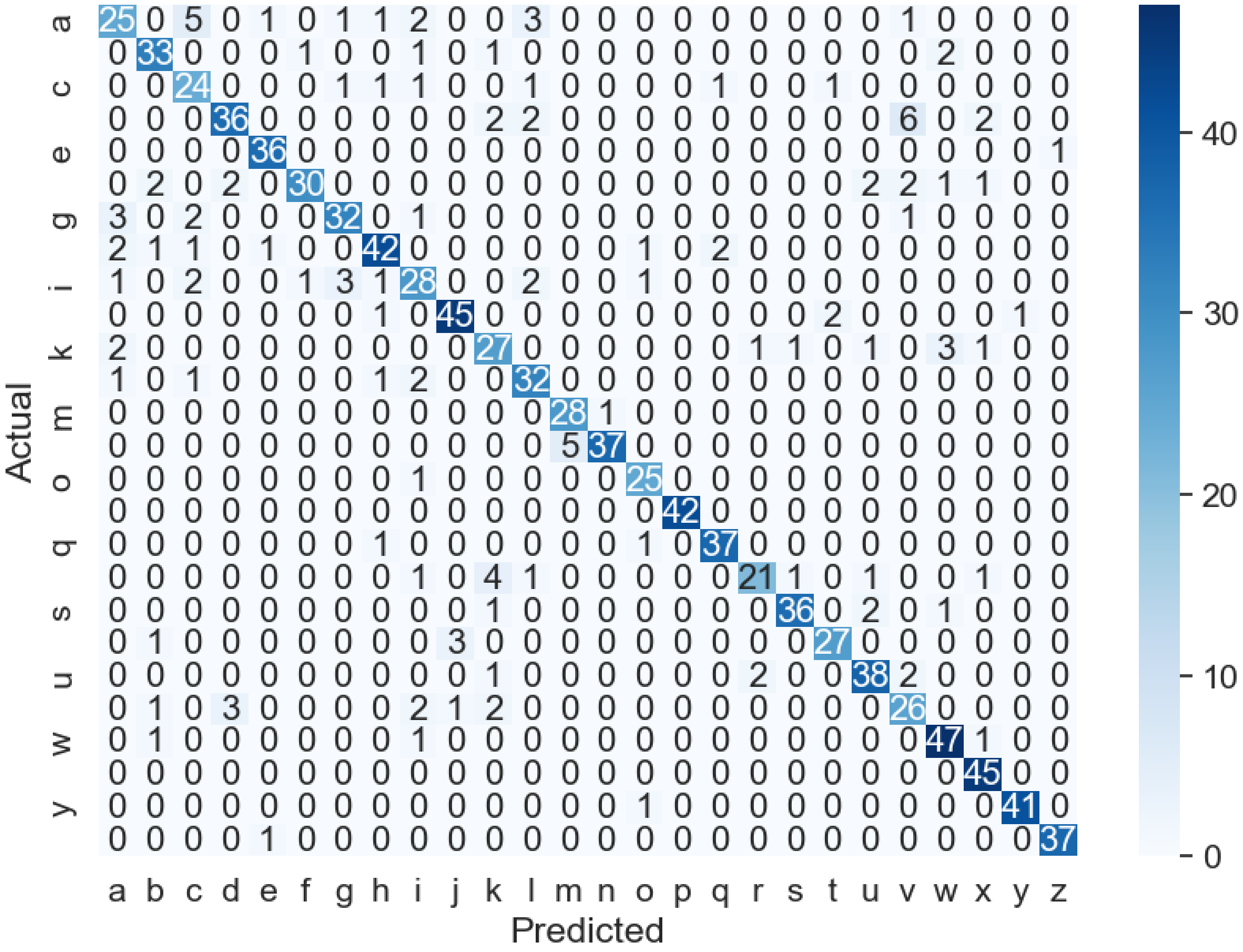
```
              precision    recall  f1-score   support

           0       0.55      0.41      0.47        39
           1       0.64      0.76      0.70        38
           2       0.53      0.83      0.65        30
           3       0.64      0.52      0.57        48
           4       0.92      0.92      0.92        37
           5       0.73      0.40      0.52        40
           6       0.85      0.72      0.78        39
           7       0.86      0.62      0.72        50
           8       0.51      0.62      0.56        39
           9       0.85      0.92      0.88        49
          10       0.46      0.53      0.49        36
          11       0.68      0.62      0.65        37
          12       0.71      0.93      0.81        29
          13       0.88      0.67      0.76        42
          14       0.79      0.88      0.84        26
          15       0.83      0.90      0.86        42
          16       0.74      0.74      0.74        39
          17       0.48      0.37      0.42        30
          18       0.72      0.70      0.71        40
          19       0.77      0.65      0.70        31
          20       0.59      0.56      0.57        43
          21       0.36      0.46      0.40        35
          22       0.78      0.76      0.77        50
          23       0.73      1.00      0.84        45
          24       0.83      0.95      0.89        42
          25       0.94      0.84      0.89        38

    accuracy                           0.70      1014
   macro avg       0.71      0.70      0.70      1014
weighted avg       0.71      0.70      0.70      1014
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	11	0	4	0	0	0	0	1	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	2	2	0	0	0
c	6	0	7	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	1	2
	0	0	1	14	0	0	0	0	1	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	4	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	10	0	1	0	2	0
g	0	0	3	0	0	0	13	0	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	5	0	0	5	0	0	0	7	0	0
i	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	1	0	0	0	0	0	0	1	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
	0	0	0	0	0	0	0	0	15	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	19	0	0	0	0	0	0	0	0	0	0	0	0
o	0	0	1	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	2	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0
s	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	19	0	0	0	0	0
u	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0
	0	7	0	0	0	0	0	2	1	3	2	0	0	0	0	0	0	0	0	4	1	0	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	15	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0
y	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	13
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

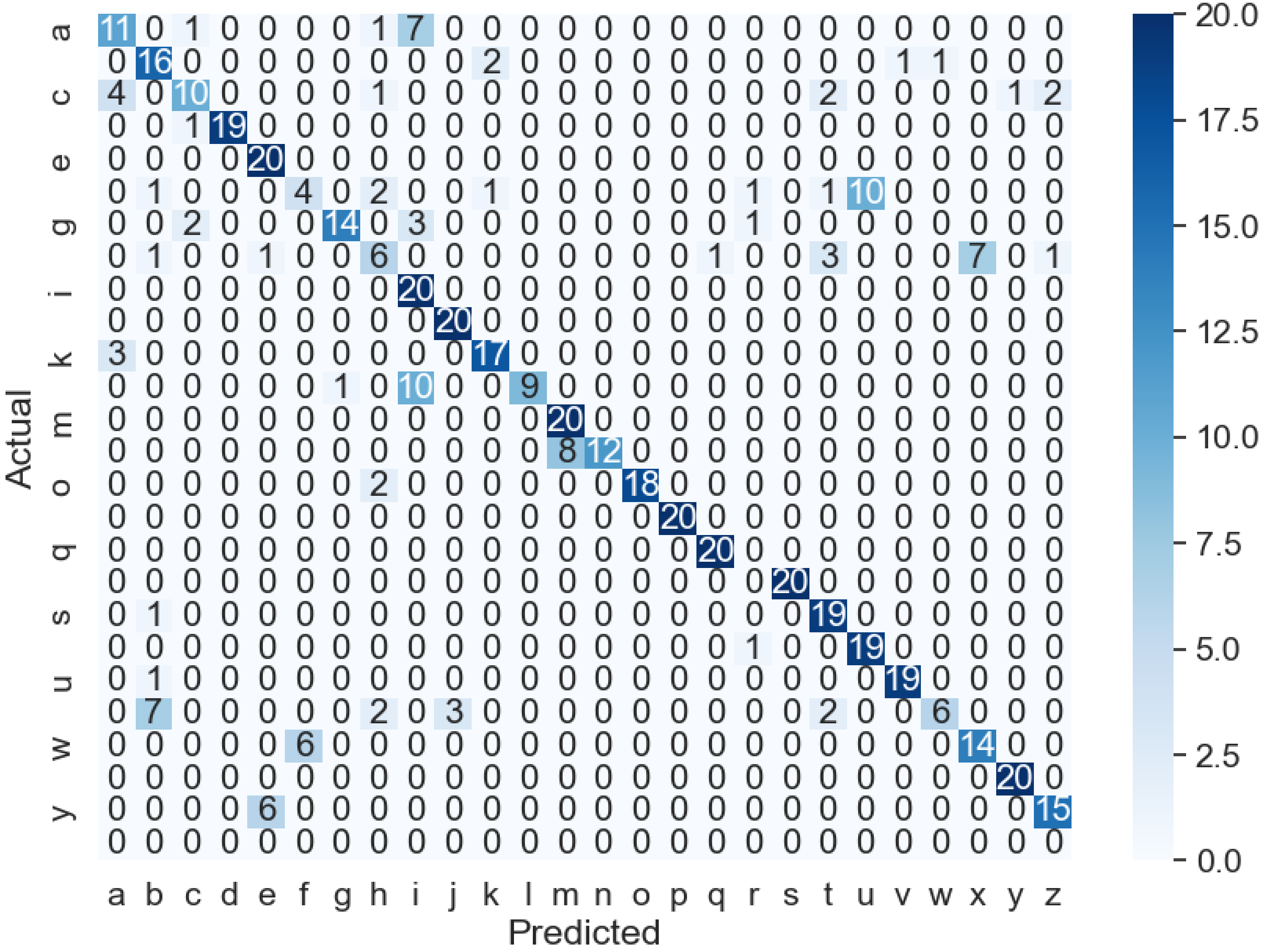
In [17]:

```
#%% LGBM
from lightgbm import LGBMClassifier
lgbm = LGBMClassifier(n_estimators=500,random_state=2021)

lgbm_cm, lgbm_report,lgbm_cm_external,lgb_acc, lgb_ext_acc = machine_learning(lgbm)
```



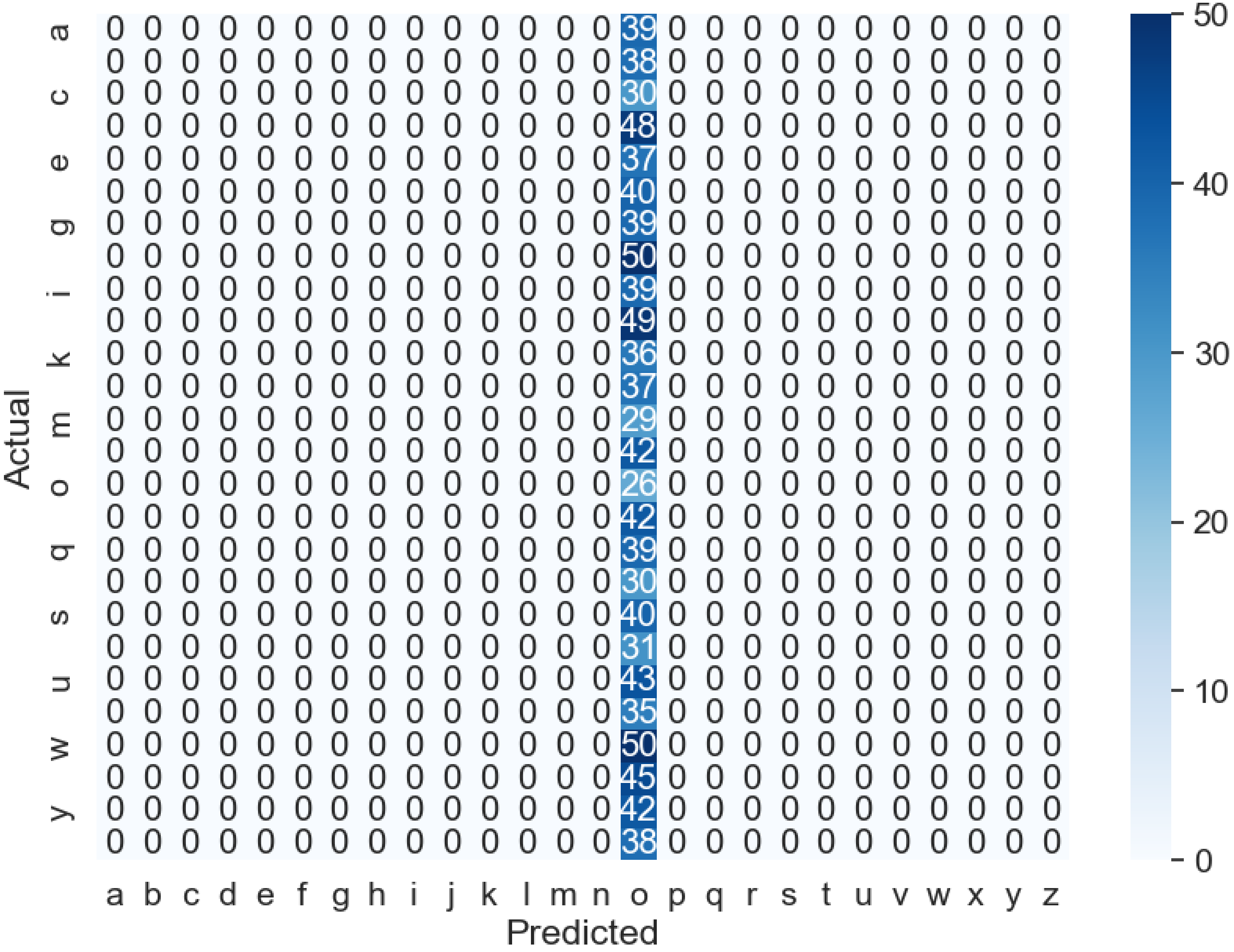
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	25	0	5	0	1	0	1	1	2	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	0	33	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
c	0	0	24	0	0	0	1	1	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0
	0	0	0	36	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	6	0	2	0	0
e	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0	2	0	2	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	1	1	0	0
g	3	0	2	0	0	0	32	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	2	1	1	0	1	0	0	42	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0
i	1	0	2	0	0	1	3	1	28	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	45	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0
k	2	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	1	1	0	1	0	3	1	0	0
	1	0	1	0	0	0	0	1	2	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	28	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	5	37	0	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	1	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	37	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	4	1	0	0	0	0	0	21	1	0	1	0	0	1	0	0
s	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	36	0	2	0	1	0	0	0
	0	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	38	2	0	0	0	0
	0	1	0	3	0	0	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0
w	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	47	1	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	41	0
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37

```
              precision    recall  f1-score   support

           0       0.74      0.64      0.68        39
           1       0.85      0.87      0.86        38
           2       0.69      0.80      0.74        30
           3       0.88      0.75      0.81        48
           4       0.92      0.97      0.95        37
           5       0.94      0.75      0.83        40
           6       0.86      0.82      0.84        39
           7       0.88      0.84      0.86        50
           8       0.70      0.72      0.71        39
           9       0.92      0.92      0.92        49
          10       0.71      0.75      0.73        36
          11       0.78      0.86      0.82        37
          12       0.85      0.97      0.90        29
          13       0.97      0.88      0.93        42
          14       0.86      0.96      0.91        26
          15       1.00      1.00      1.00        42
          16       0.93      0.95      0.94        39
          17       0.88      0.70      0.78        30
          18       0.95      0.90      0.92        40
          19       0.90      0.87      0.89        31
          20       0.86      0.88      0.87        43
          21       0.68      0.74      0.71        35
          22       0.87      0.94      0.90        50
          23       0.88      1.00      0.94        45
          24       0.98      0.98      0.98        42
          25       0.97      0.97      0.97        38

    accuracy                           0.86      1014
   macro avg       0.86      0.86      0.86      1014
weighted avg       0.87      0.86      0.86      1014
```



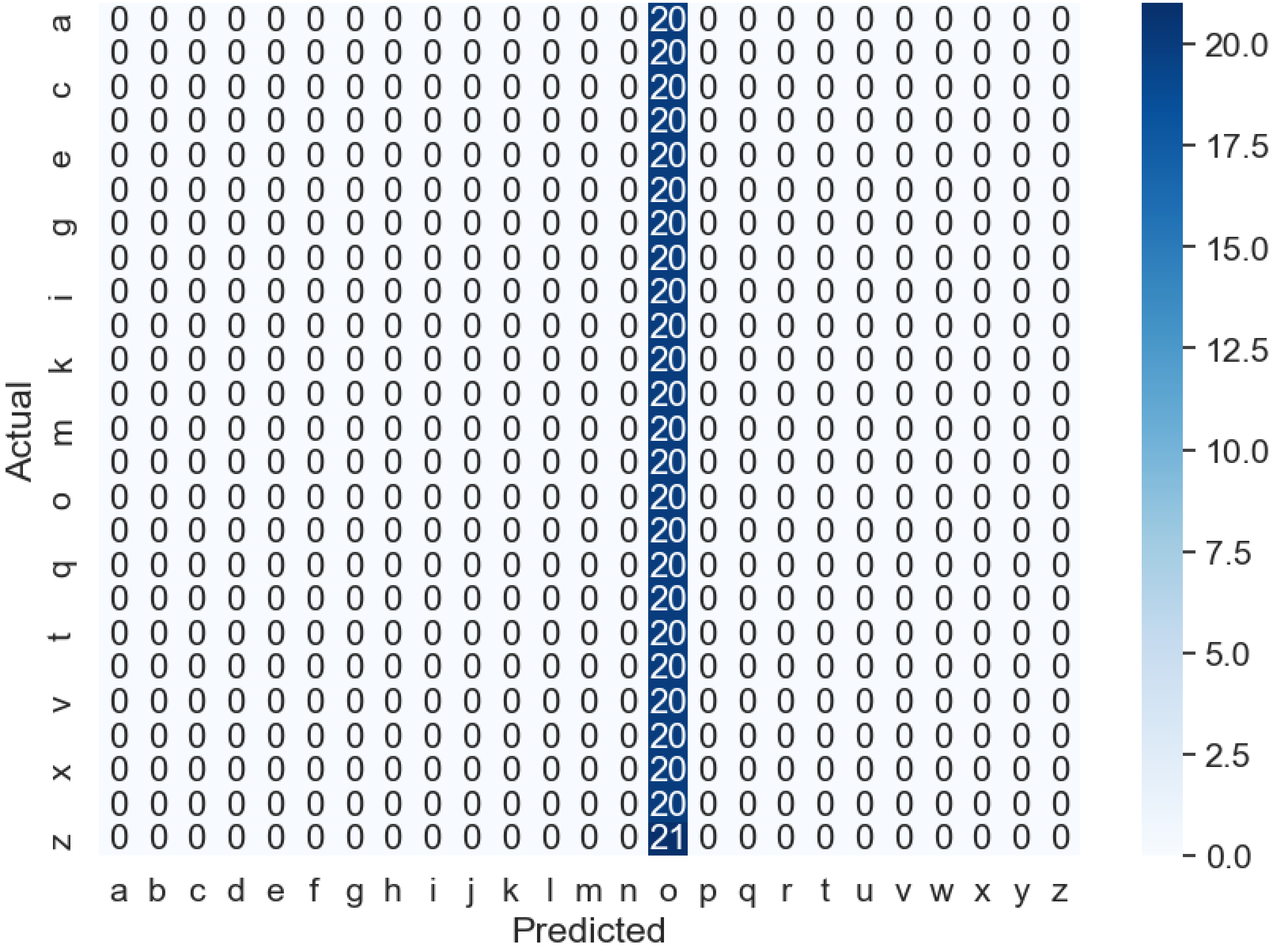
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	11	0	1	0	0	0	0	1	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	16	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
c	4	0	10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	2
	0	0	1	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	4	0	2	0	0	1	0	0	0	0	0	0	1	0	1	10	0	0	0	0	0
g	0	0	2	0	0	0	14	0	3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0	1	0	0	1	0	0	6	0	0	0	0	0	0	0	0	1	0	0	3	0	0	0	7	0	1
i	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	3	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0	10	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	8	12	0	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	2	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0
s	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	19	0	0	0	0	0
u	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0
	0	7	0	0	0	0	0	2	0	3	0	0	0	0	0	0	0	0	0	2	0	0	6	0	0	0
w	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0
y	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
In [18]: #Support vector machine
         from sklearn.svm import SVC
         SVCClf = SVC(kernel = 'linear',gamma = 'scale', shrinking = False,)

         SVCClf_cm, SVCClf_report,SVCClf_cm_external, svc_acc, svc_ext_acc = machine_learning1(SVCClf)
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0

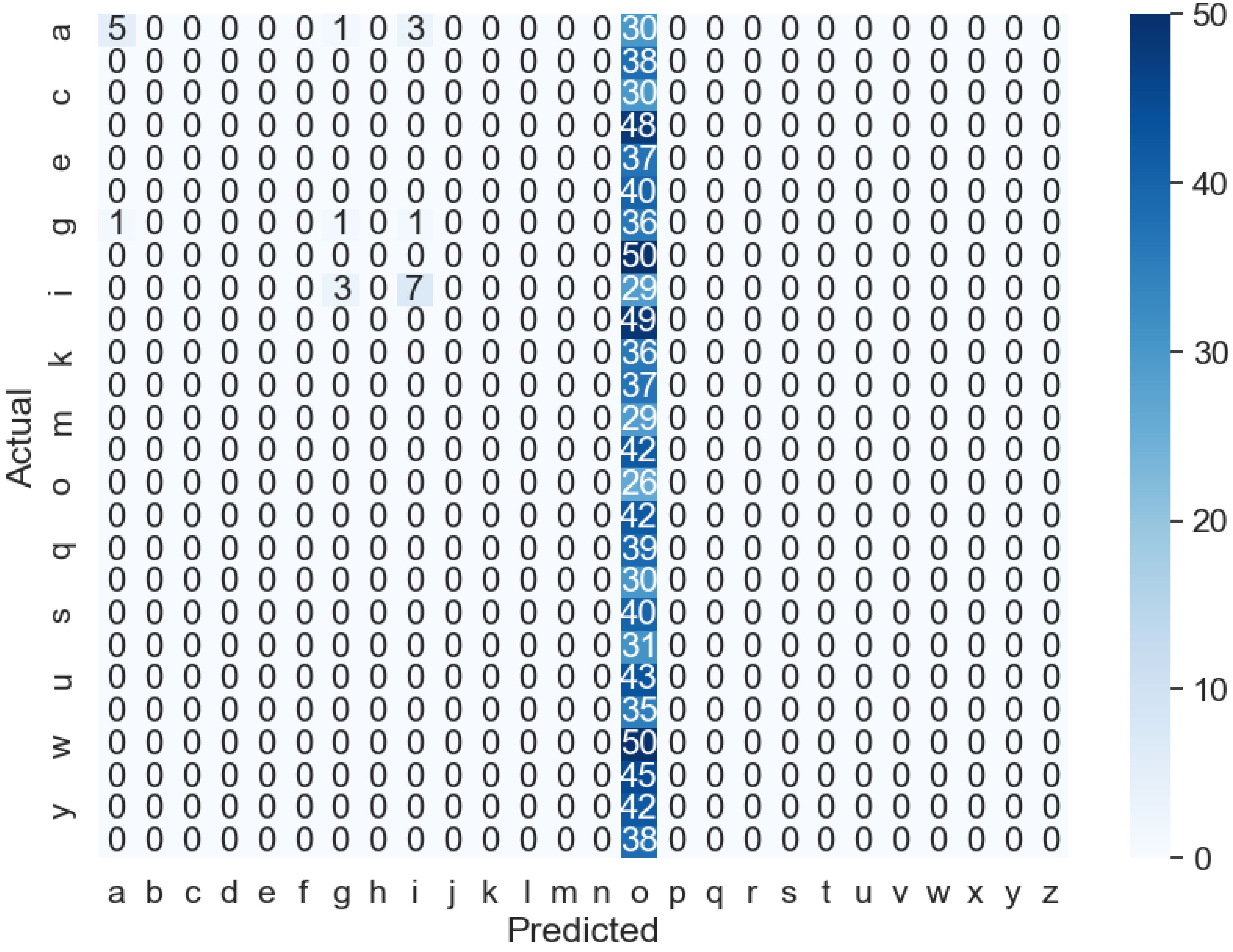
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        39
           1       0.00      0.00      0.00        38
           2       0.00      0.00      0.00        30
           3       0.00      0.00      0.00        48
           4       0.00      0.00      0.00        37
           5       0.00      0.00      0.00        40
           6       0.00      0.00      0.00        39
           7       0.00      0.00      0.00        50
           8       0.00      0.00      0.00        39
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        36
          11       0.00      0.00      0.00        37
          12       0.00      0.00      0.00        29
          13       0.00      0.00      0.00        42
          14       0.03      1.00      0.05        26
          15       0.00      0.00      0.00        42
          16       0.00      0.00      0.00        39
          17       0.00      0.00      0.00        30
          18       0.00      0.00      0.00        40
          19       0.00      0.00      0.00        31
          20       0.00      0.00      0.00        43
          21       0.00      0.00      0.00        35
          22       0.00      0.00      0.00        50
          23       0.00      0.00      0.00        45
          24       0.00      0.00      0.00        42
          25       0.00      0.00      0.00        38

    accuracy                           0.03      1014
   macro avg       0.00      0.04      0.00      1014
weighted avg       0.00      0.03      0.00      1014
```



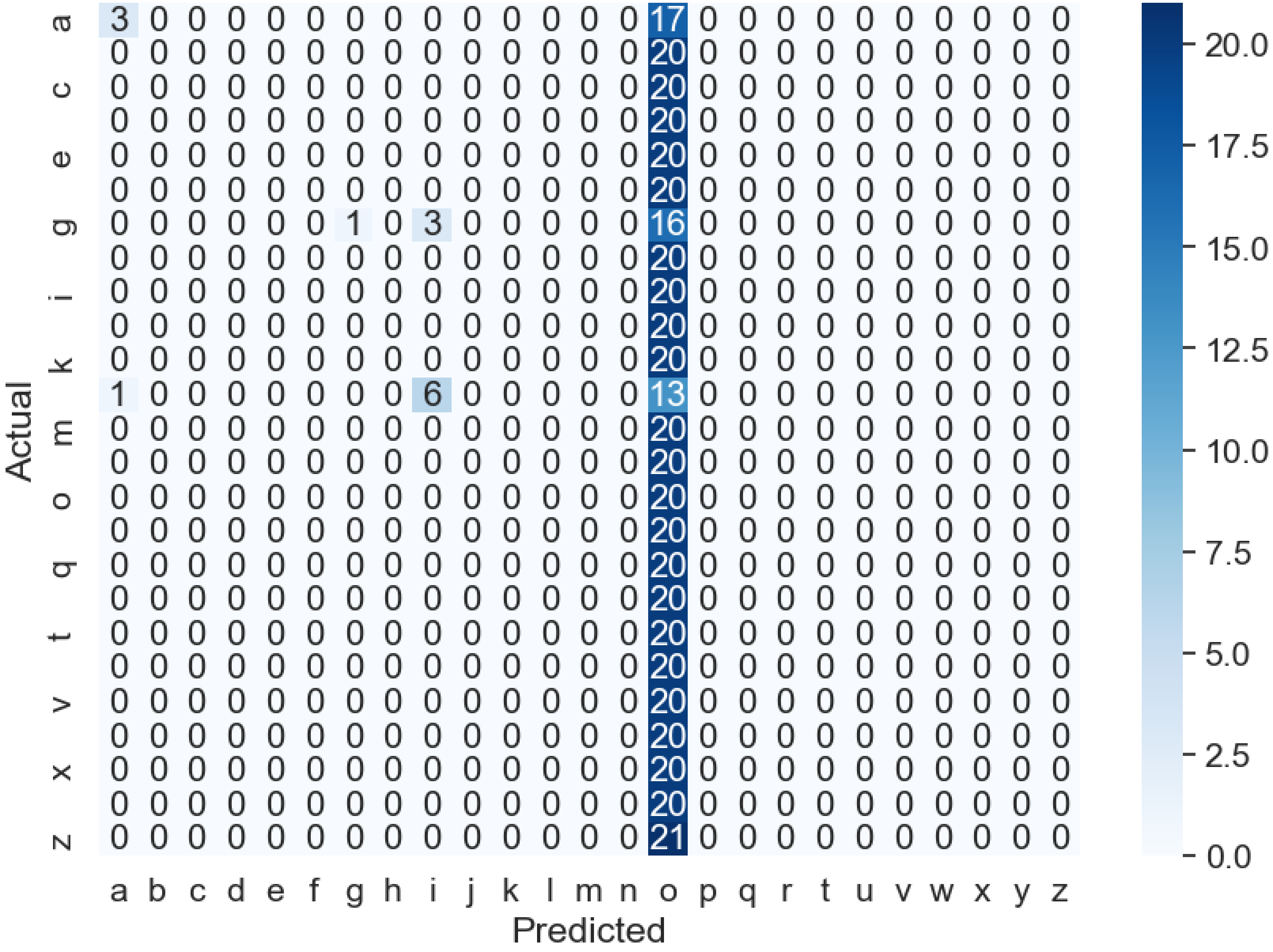
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	t	u	v	w	x	y	z
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0

In [19]:

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion="gini", random_state=42,max_depth=3, min_samples_leaf=5)
clf_cm, clf_report,clf_cm_external, clf_acc, clf_ext_acc = machine_learning1(clf)
```


Actual
a
c
e
g
i
k
m
o
q
s
u
w
y
5 0 0 0 0 0 1 0 3 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 1 0 0 0 0 0 36 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 50 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 3 0 7 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 36 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 43 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 50 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0
a b c d e f g h i j k l m n o p q r s t u v w x y z
Predicted
50
40
30
20
10
0

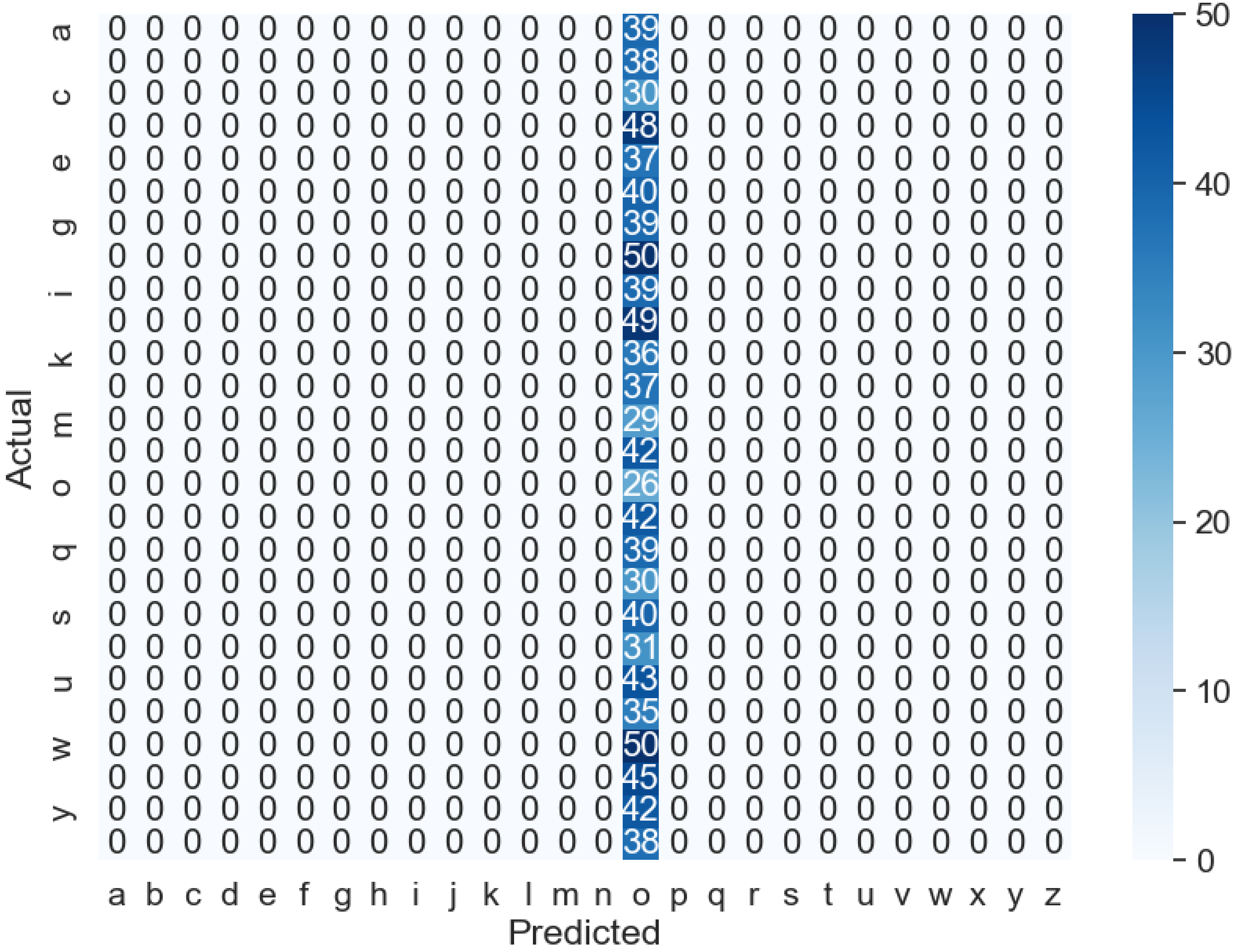
```
              precision    recall  f1-score   support

           0       0.83      0.13      0.22        39
           1       0.00      0.00      0.00        38
           2       0.00      0.00      0.00        30
           3       0.00      0.00      0.00        48
           4       0.00      0.00      0.00        37
           5       0.00      0.00      0.00        40
           6       0.20      0.03      0.05        39
           7       0.00      0.00      0.00        50
           8       0.64      0.18      0.28        39
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        36
          11       0.00      0.00      0.00        37
          12       0.00      0.00      0.00        29
          13       0.00      0.00      0.00        42
          14       0.03      1.00      0.05        26
          15       0.00      0.00      0.00        42
          16       0.00      0.00      0.00        39
          17       0.00      0.00      0.00        30
          18       0.00      0.00      0.00        40
          19       0.00      0.00      0.00        31
          20       0.00      0.00      0.00        43
          21       0.00      0.00      0.00        35
          22       0.00      0.00      0.00        50
          23       0.00      0.00      0.00        45
          24       0.00      0.00      0.00        42
          25       0.00      0.00      0.00        38

    accuracy                           0.04      1014
   macro avg       0.07      0.05      0.02      1014
weighted avg       0.06      0.04      0.02      1014
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	t	u	v	w	x	y	z
a	3	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	1	0	3	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	6	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0

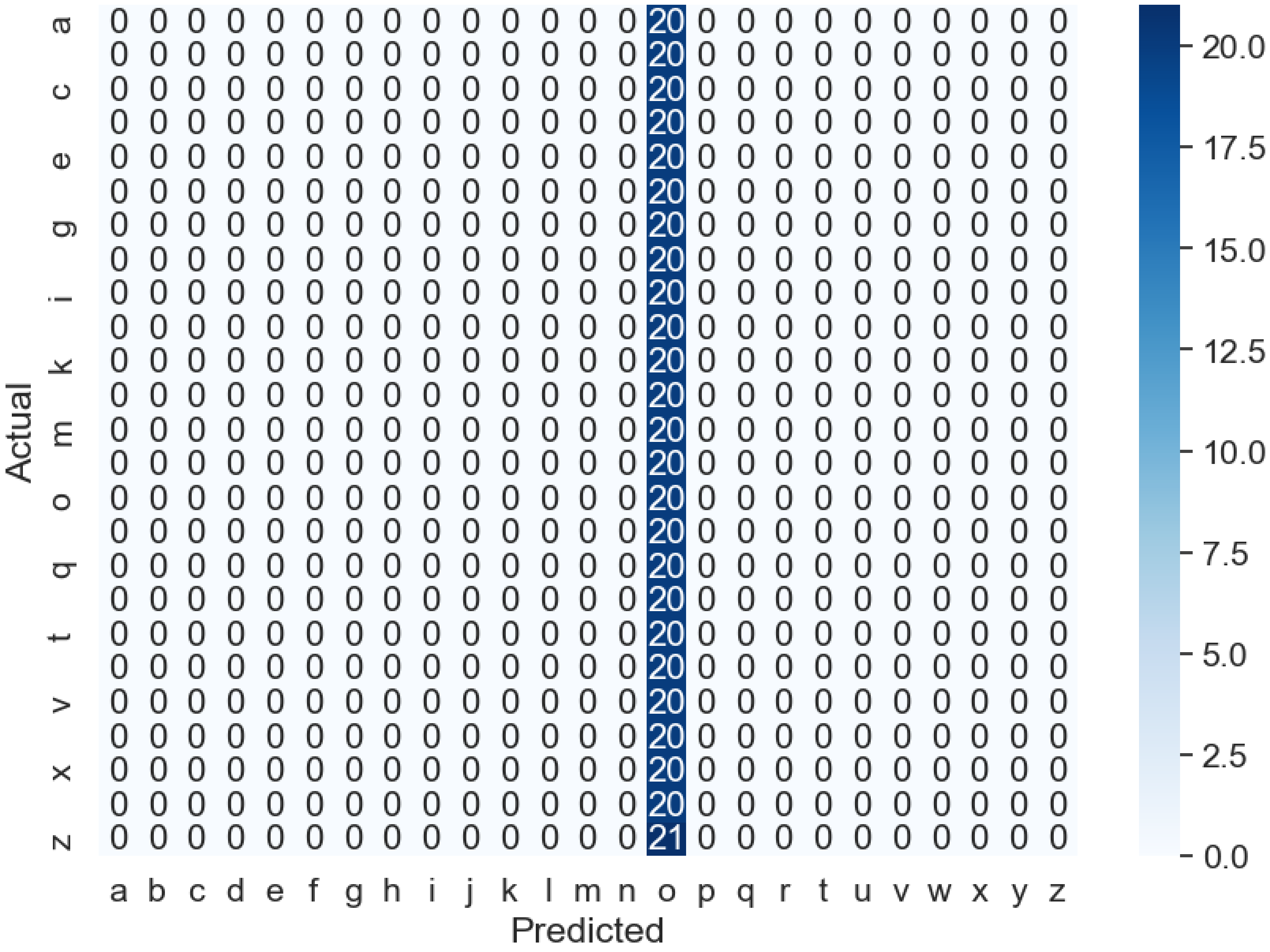
In [20]:

```python
#LogisticRegression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(random_state = 0)
logreg_cm, logreg_report,logreg_cm_external, logreg_acc, logreg_ext_acc = machine_learning1(logreg)
```



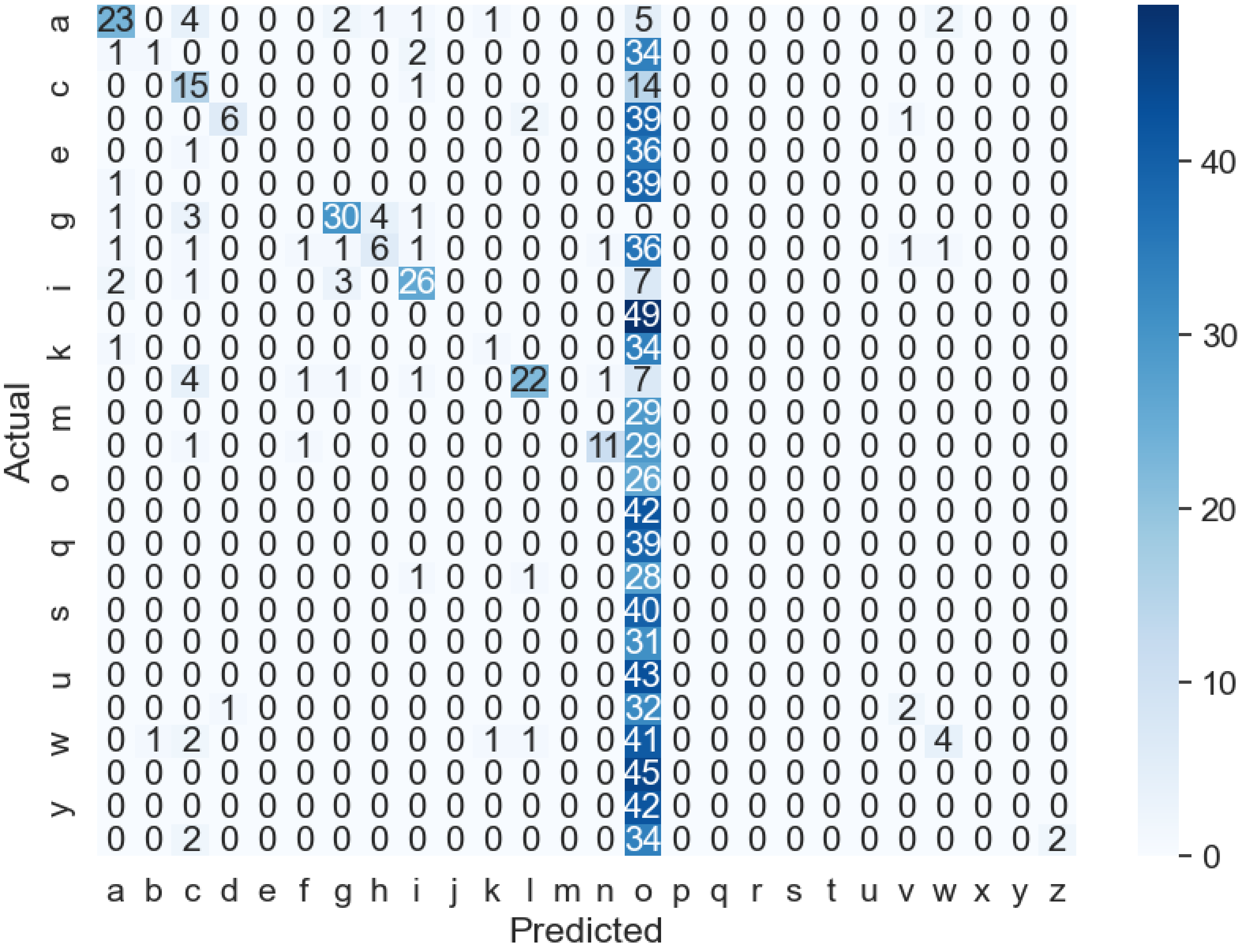
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        39
           1       0.00      0.00      0.00        38
           2       0.00      0.00      0.00        30
           3       0.00      0.00      0.00        48
           4       0.00      0.00      0.00        37
           5       0.00      0.00      0.00        40
           6       0.00      0.00      0.00        39
           7       0.00      0.00      0.00        50
           8       0.00      0.00      0.00        39
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        36
          11       0.00      0.00      0.00        37
          12       0.00      0.00      0.00        29
          13       0.00      0.00      0.00        42
          14       0.03      1.00      0.05        26
          15       0.00      0.00      0.00        42
          16       0.00      0.00      0.00        39
          17       0.00      0.00      0.00        30
          18       0.00      0.00      0.00        40
          19       0.00      0.00      0.00        31
          20       0.00      0.00      0.00        43
          21       0.00      0.00      0.00        35
          22       0.00      0.00      0.00        50
          23       0.00      0.00      0.00        45
          24       0.00      0.00      0.00        42
          25       0.00      0.00      0.00        38

    accuracy                           0.03      1014
   macro avg       0.00      0.04      0.00      1014
weighted avg       0.00      0.03      0.00      1014
```



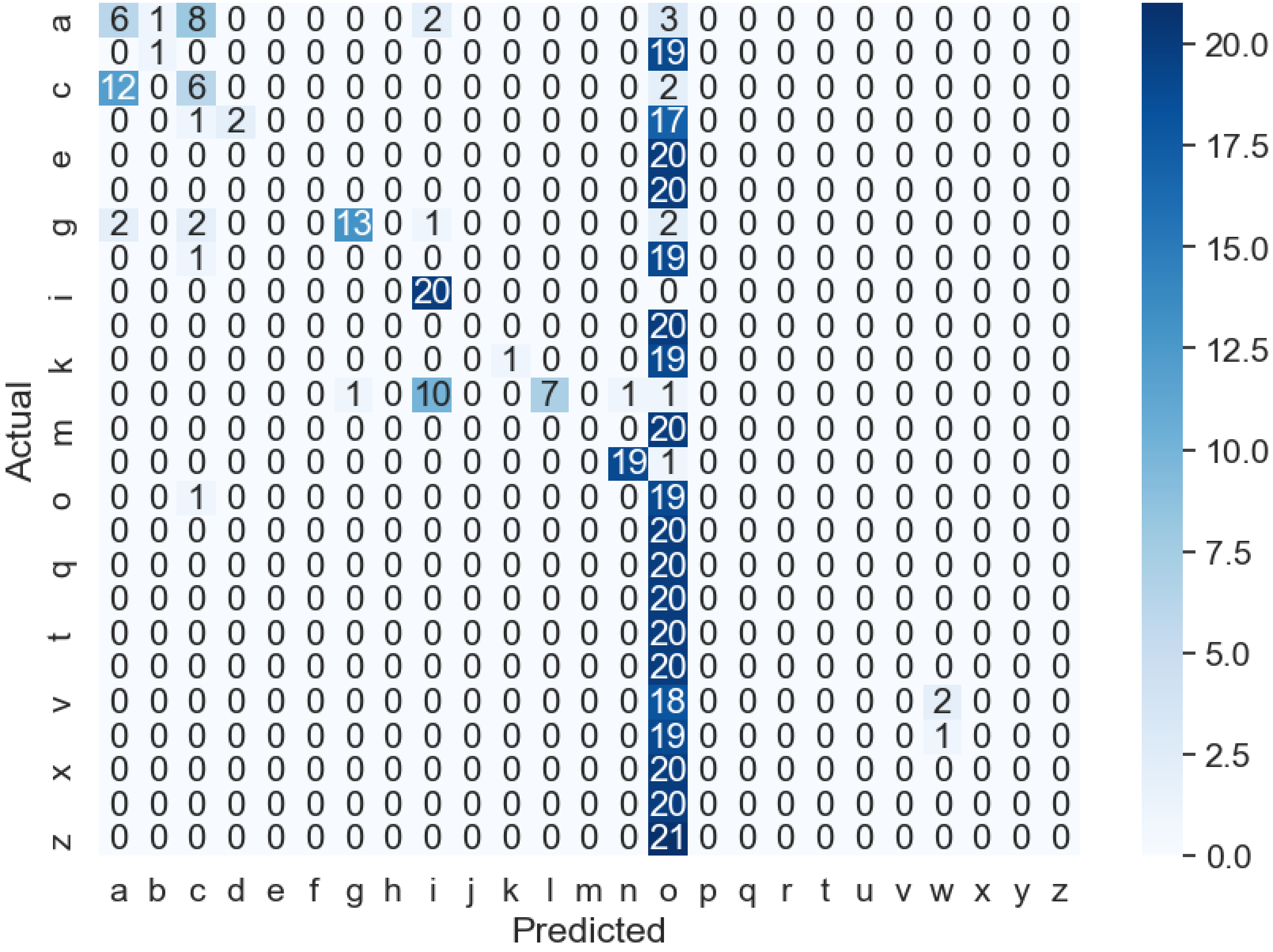
In [21]:

```
#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(n_estimators = 100)
rf_clf_cm, rf_clf_report,rf_clf_cm_external, rf_acc, rf_ext_acc = machine_learning1(rf_clf)
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	23	0	4	0	0	0	2	1	1	0	1	0	0	0	5	0	0	0	0	0	0	0	2	0	0	0
	1	1	0	0	0	0	0	0	2	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0
c	0	0	15	0	0	0	0	0	1	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	6	0	0	0	0	0	0	0	2	0	0	39	0	0	0	0	0	0	1	0	0	0	0
e	0	0	1	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
g	1	0	3	0	0	0	30	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	1	0	0	1	1	6	1	0	0	0	0	1	36	0	0	0	0	0	0	1	1	0	0	0
i	2	0	1	0	0	0	3	0	26	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49	0	0	0	0	0	0	0	0	0	0	0
k	1	0	0	0	0	0	0	0	0	0	1	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0
	0	0	4	0	0	1	1	0	1	0	0	22	0	1	7	0	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	1	0	0	0	0	0	0	0	11	29	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	0	1	0	0	28	0	0	0	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	2	0	0	0	0
w	0	1	2	0	0	0	0	0	0	0	1	1	0	0	41	0	0	0	0	0	0	0	4	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0
	0	0	2	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	2

```
              precision    recall  f1-score   support

           0       0.77      0.59      0.67        39
           1       0.50      0.03      0.05        38
           2       0.44      0.50      0.47        30
           3       0.86      0.12      0.22        48
           4       0.00      0.00      0.00        37
           5       0.00      0.00      0.00        40
           6       0.81      0.77      0.79        39
           7       0.55      0.12      0.20        50
           8       0.76      0.67      0.71        39
           9       0.00      0.00      0.00        49
          10       0.33      0.03      0.05        36
          11       0.85      0.59      0.70        37
          12       0.00      0.00      0.00        29
          13       0.85      0.26      0.40        42
          14       0.03      1.00      0.06        26
          15       0.00      0.00      0.00        42
          16       0.00      0.00      0.00        39
          17       0.00      0.00      0.00        30
          18       0.00      0.00      0.00        40
          19       0.00      0.00      0.00        31
          20       0.00      0.00      0.00        43
          21       0.50      0.06      0.10        35
          22       0.57      0.08      0.14        50
          23       0.00      0.00      0.00        45
          24       0.00      0.00      0.00        42
          25       1.00      0.05      0.10        38

    accuracy                           0.17      1014
   macro avg       0.34      0.19      0.18      1014
weighted avg       0.35      0.17      0.18      1014
```



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	t	u	v	w	x	y	z
a	6	1	8	0	0	0	0	0	2	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0
c	12	0	6	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
	0	0	1	2	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
g	2	0	2	0	0	0	13	0	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	0	0	0	0	1	0	0	0	19	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0	10	0	0	7	0	1	1	0	0	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	19	1	0	0	0	0	0	0	0	0	0	0
o	0	0	1	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	2	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	1	0	0	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0

In [22]:

```
#Accuary of each alogirthm displaying as table
res_table = pd.DataFrame({
    'Model': ['XGBoost','LGBM','SVC','Decision Tree','Logistic Regression','Random Forest'],
    'Train Score': [xgb_acc,lgb_acc,svc_acc,clf_acc,logreg_acc,rf_acc],
    'Test Score' : [xgb_ext_acc,lgb_ext_acc,svc_ext_acc,clf_ext_acc,logreg_ext_acc,rf_ext_acc]})
res_table.sort_values(by='Train Score', ascending=False)
```

Out[22]:

	Model	Train Score	Test Score
1	LGBM	0.864892	0.510978
0	XGBoost	0.704142	0.483034
5	Random Forest	0.172584	0.189621
3	Decision Tree	0.038462	0.047904
2	SVC	0.025641	0.039920
4	Logistic Regression	0.025641	0.039920