COLOR CLASSIFICATION

WHY?

COLORS ARE A FUNDAMENTAL ASPECT OF OUR VISUAL PERCEPTION, AND PLAY AN ESSENTIAL ROLE IN OUR DAILY LIVES. APPLICATIONS IN AREAS SUCH AS IMAGE RECOGNITION, COLOR-BASED SEARCH, AND COLOR-BASED RECOMMENDATIONS. FURTHERMORE.

MAIN PROBLEM FACED

THE TRAINING DATA WE INITAILY USED WAS NOT CLEAN AT ALL.

IT HAD LOTS OF INCONSISTENCIES, AND WAS DESTINED TO OUTCOME A BIASED OUTPUT.

SOLUTION?

IF YOU CANT FIX IT..... JUST THROW IT AWAY!!

WE CHANGED ALL OF THE DATASET INTO A NEW CLEAN ONE

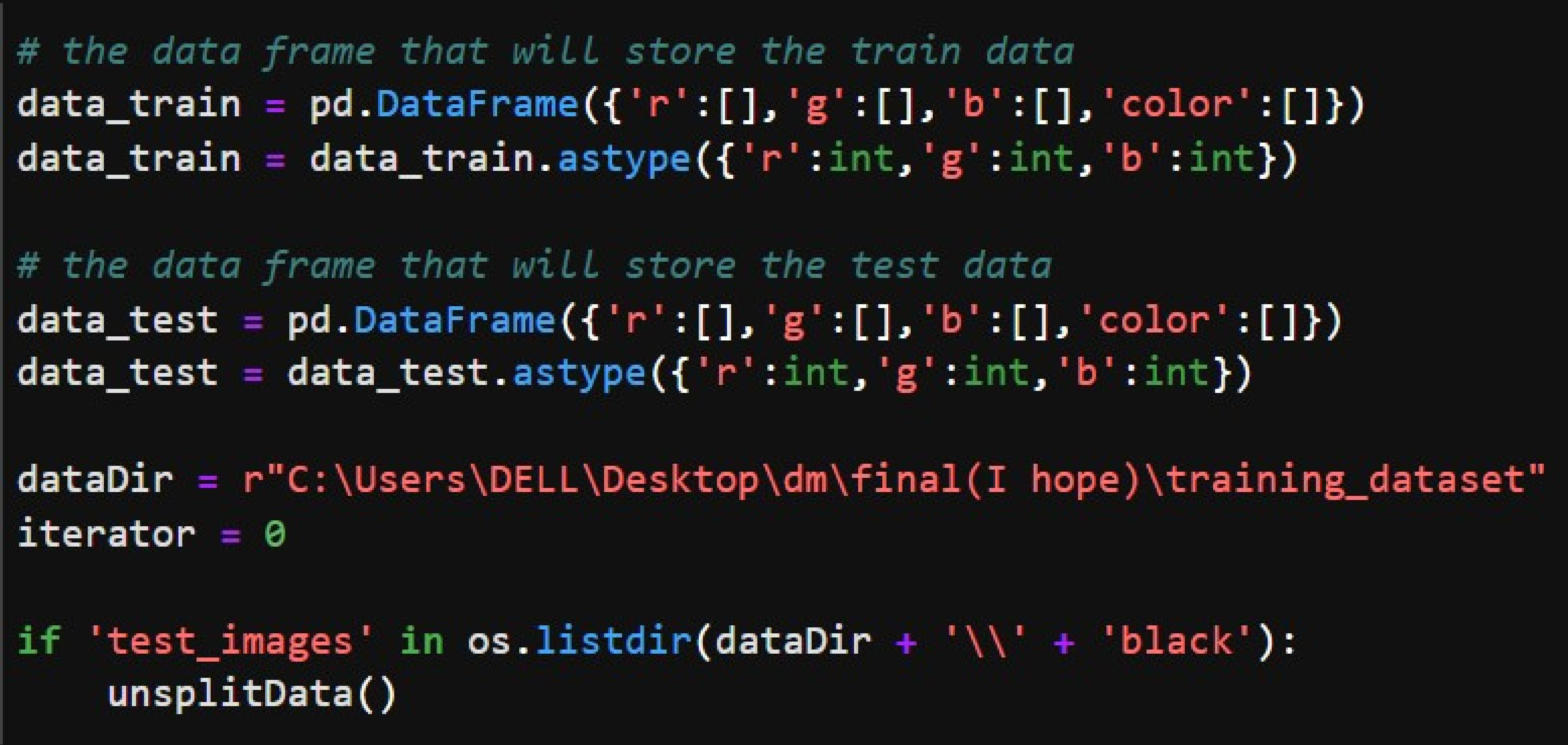
[DATASET]

DATA PROCESSING:

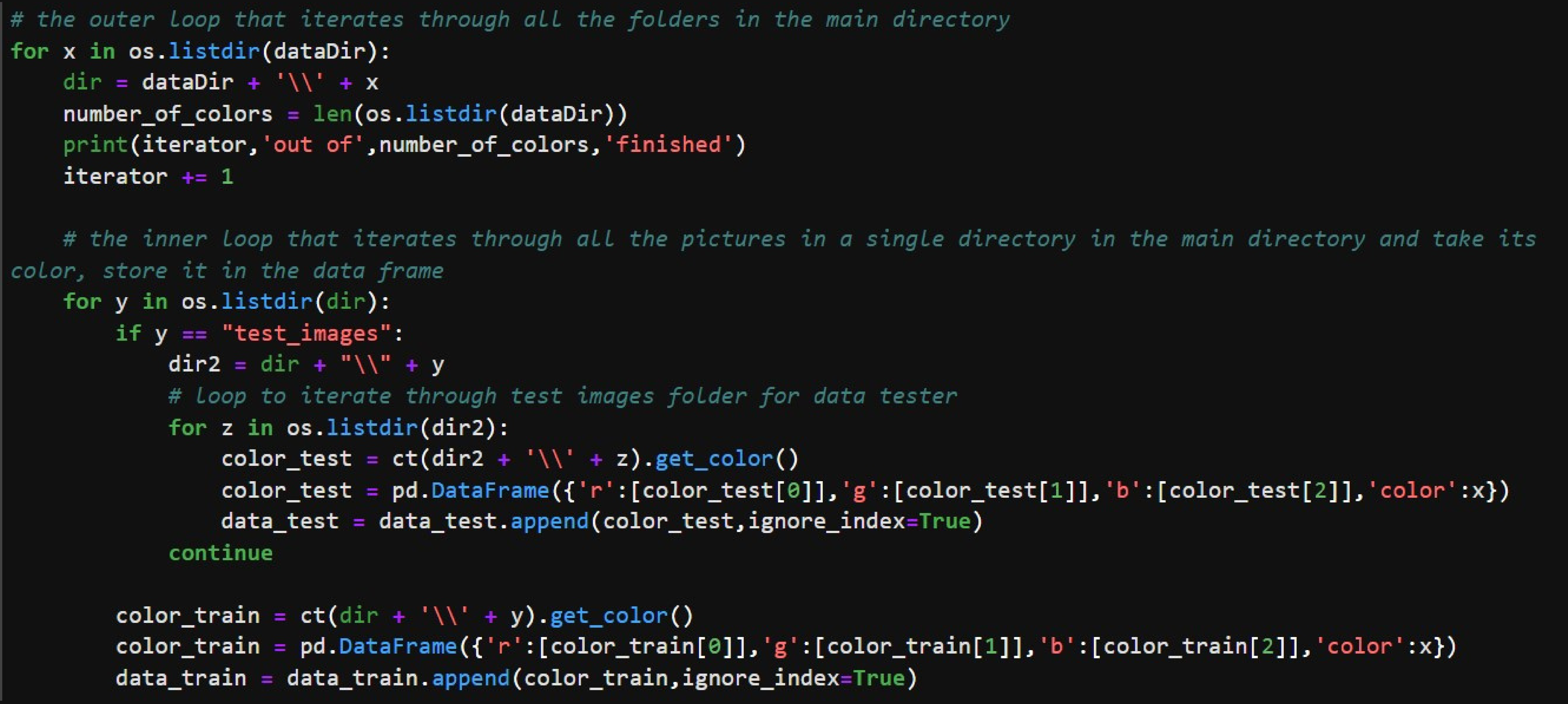
WE START BY LOADING THE DATASET FROM A DIRECTORY USING THE OS LIBRARY.

WE ITERATE OVER EVERY DIRECTORY IN THE DATASET, EACH OF WHICH CORRESPONDS TO A DIFFERENT COLOR.

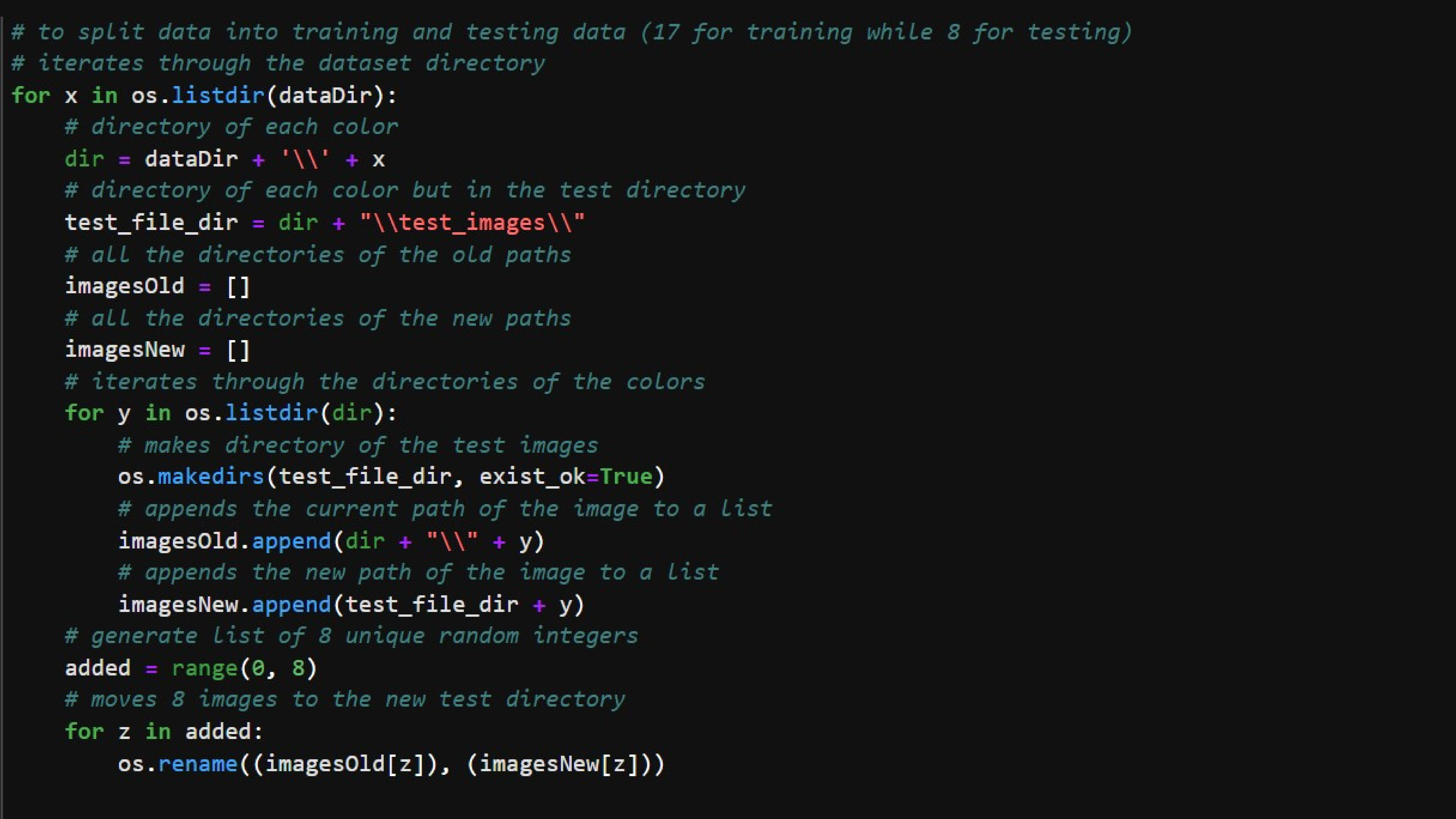
FOR EACH IMAGE IN THE DIRECTORY, WE USE THE COLORTHIEF LIBRARY TO EXTRACT ITS DOMINANT COLOR IN R, G ,B VALUES (0- 255) AND STORE IT ALONG WITH THE IMAGE'S LABEL (I.E., THE COLOR DIRECTORY) IN A PANDAS DATAFRAME.
WE THEN SPLIT THE DATA INTO TRAINING AND TESTING SETS AND PREPROCESS IT BY ONE-HOT ENCODING THE LABELS.



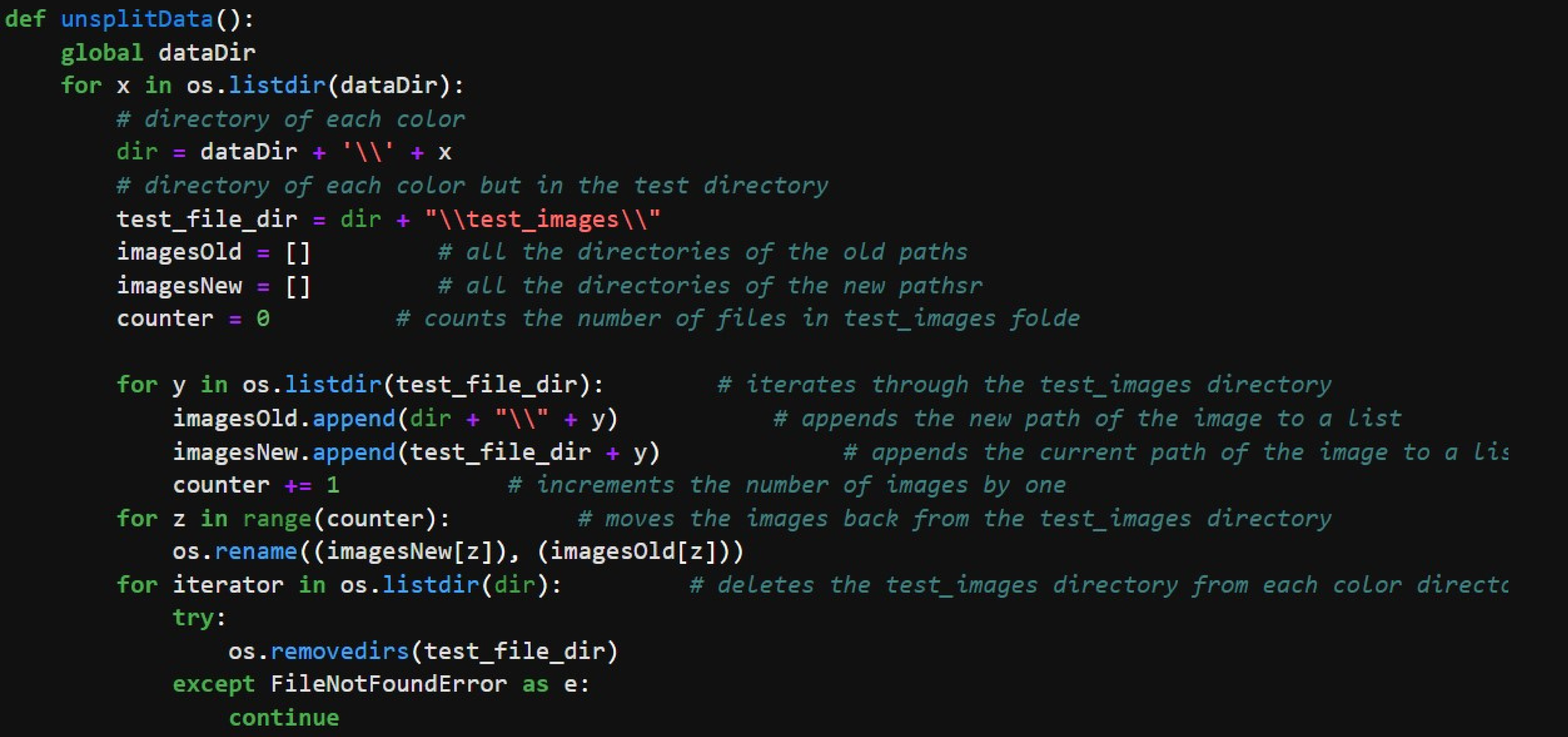
```python
# the data frame that will store the train data
data_train = pd.DataFrame({'r':[],'g':[],'b':[],'color':[]})
data_train = data_train.astype({'r':int,'g':int,'b':int})

# the data frame that will store the test data
data_test = pd.DataFrame({'r':[],'g':[],'b':[],'color':[]})
data_test = data_test.astype({'r':int,'g':int,'b':int})

dataDir = r"C:\Users\DELL\Desktop\dm\final(I hope)\training_dataset"
iterator = 0

if 'test_images' in os.listdir(dataDir + '\\' + 'black'):
    unsplitData()
```



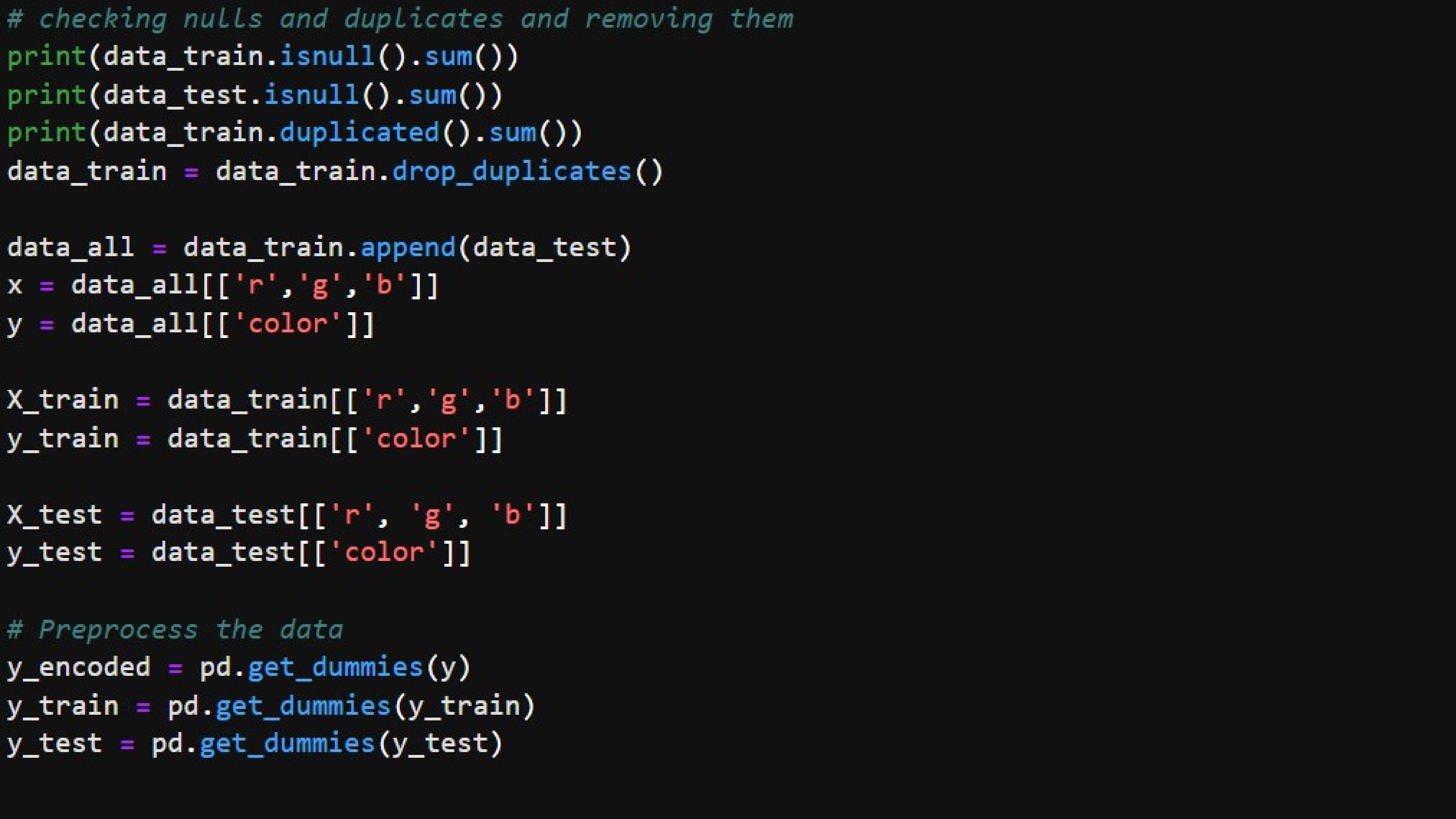
```python
# the outer loop that iterates through all the folders in the main directory
for x in os.listdir(dataDir):
    dir = dataDir + '\\' + x
    number_of_colors = len(os.listdir(dataDir))
    print(iterator,'out of',number_of_colors,'finished')
    iterator += 1

    # the inner loop that iterates through all the pictures in a single directory in the main directory and take its color, store it in the data frame
    for y in os.listdir(dir):
        if y == "test_images":
            dir2 = dir + "\\" + y
            # loop to iterate through test images folder for data tester
            for z in os.listdir(dir2):
                color_test = ct(dir2 + '\\' + z).get_color()
                color_test = pd.DataFrame({'r':[color_test[0]],'g':[color_test[1]],'b':[color_test[2]],'color':x})
                data_test = data_test.append(color_test,ignore_index=True)
            continue

        color_train = ct(dir + '\\' + y).get_color()
        color_train = pd.DataFrame({'r':[color_train[0]],'g':[color_train[1]],'b':[color_train[2]],'color':x})
        data_train = data_train.append(color_train,ignore_index=True)
```



```python
# to split data into training and testing data (17 for training while 8 for testing)
# iterates through the dataset directory
for x in os.listdir(dataDir):
    # directory of each color
    dir = dataDir + '\\' + x
    # directory of each color but in the test directory
    test_file_dir = dir + "\\test_images\\"
    # all the directories of the old paths
    imagesOld = []
    # all the directories of the new paths
    imagesNew = []
    # iterates through the directories of the colors
    for y in os.listdir(dir):
        # makes directory of the test images
        os.makedirs(test_file_dir, exist_ok=True)
        # appends the current path of the image to a list
        imagesOld.append(dir + "\\" + y)
        # appends the new path of the image to a list
        imagesNew.append(test_file_dir + y)
    # generate list of 8 unique random integers
    added = range(0, 8)
    # moves 8 images to the new test directory
    for z in added:
        os.rename((imagesOld[z]), (imagesNew[z]))
```



```python
def unsplitData():
    global dataDir
    for x in os.listdir(dataDir):
        # directory of each color
        dir = dataDir + '\\' + x
        # directory of each color but in the test directory
        test_file_dir = dir + "\\test_images\\"
        imagesOld = []         # all the directories of the old paths
        imagesNew = []         # all the directories of the new pathsr
        counter = 0         # counts the number of files in test_images folde

        for y in os.listdir(test_file_dir):        # iterates through the test_images directory
            imagesOld.append(dir + "\\" + y)           # appends the new path of the image to a list
            imagesNew.append(test_file_dir + y)             # appends the current path of the image to a lis
            counter += 1            # increments the number of images by one
        for z in range(counter):         # moves the images back from the test_images directory
            os.rename((imagesNew[z]), (imagesOld[z]))
        for iterator in os.listdir(dir):         # deletes the test_images directory from each color directo
            try:
                os.removedirs(test_file_dir)
            except FileNotFoundError as e:
                continue
```



```
# checking nulls and duplicates and removing them
print(data_train.isnull().sum())
print(data_test.isnull().sum())
print(data_train.duplicated().sum())
data_train = data_train.drop_duplicates()

data_all = data_train.append(data_test)
x = data_all[['r','g','b']]
y = data_all[['color']]

X_train = data_train[['r','g','b']]
y_train = data_train[['color']]

X_test = data_test[['r', 'g', 'b']]
y_test = data_test[['color']]

# Preprocess the data
y_encoded = pd.get_dummies(y)
y_train = pd.get_dummies(y_train)
y_test = pd.get_dummies(y_test)
```

NEURAL NETWORK MODEL

NEXT, WE TRAIN NEURAL NETWORK MODEL ON THE DATA USING KERAS' SEQUENTIAL CLASS.

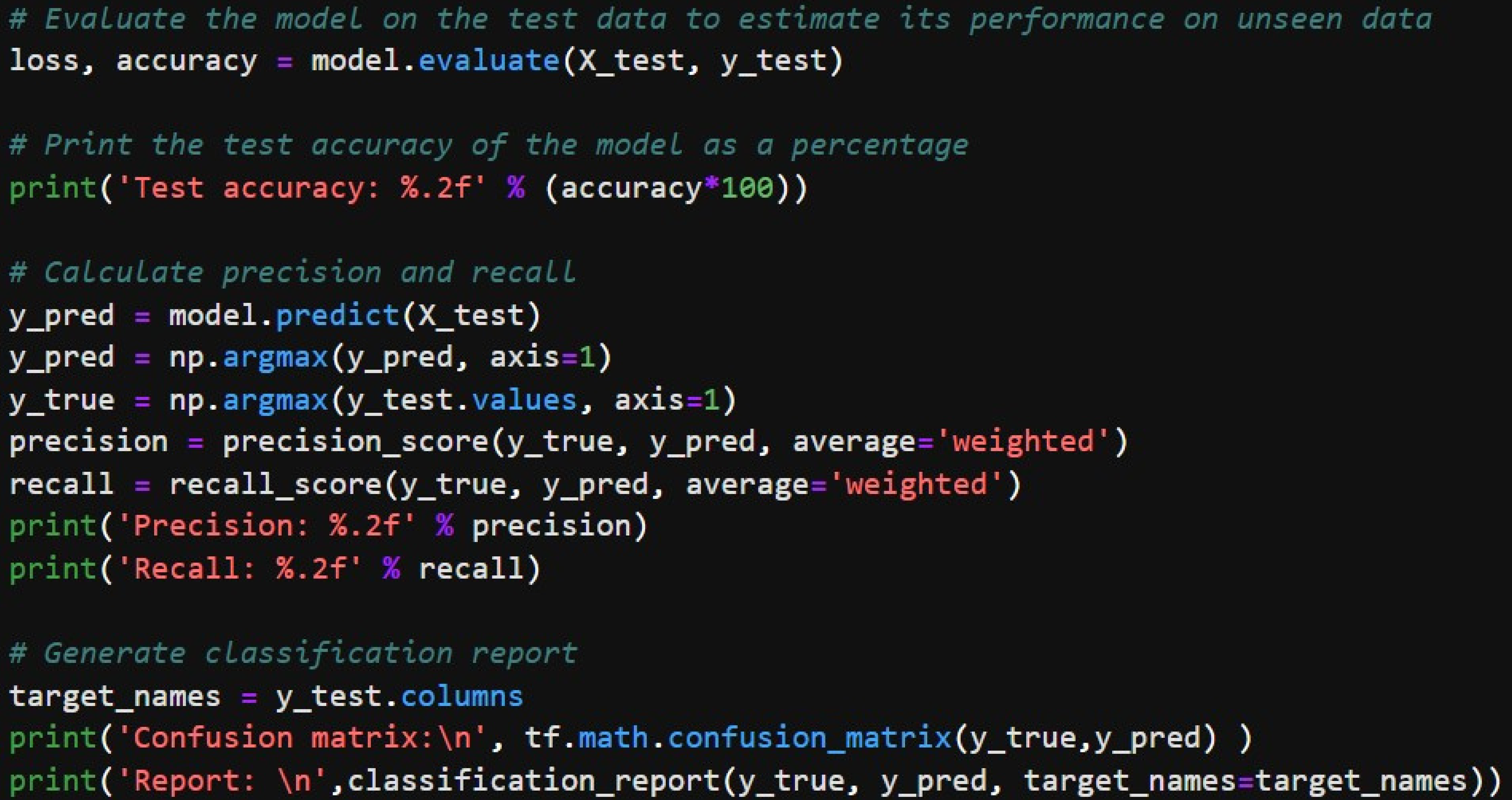
THE ANN MODEL CONSISTS OF TWO FULLY CONNECTED LAYERS WITH RELU AND SOFTMAX ACTIVATION FUNCTIONS, RESPECTIVELY.

WE TRAIN THE MODEL ON THE TRAINING SET FOR 50 EPOCHS WITH A BATCH SIZE OF 32 AND USE VALIDATION DATA TO MONITOR PERFORMANCE.

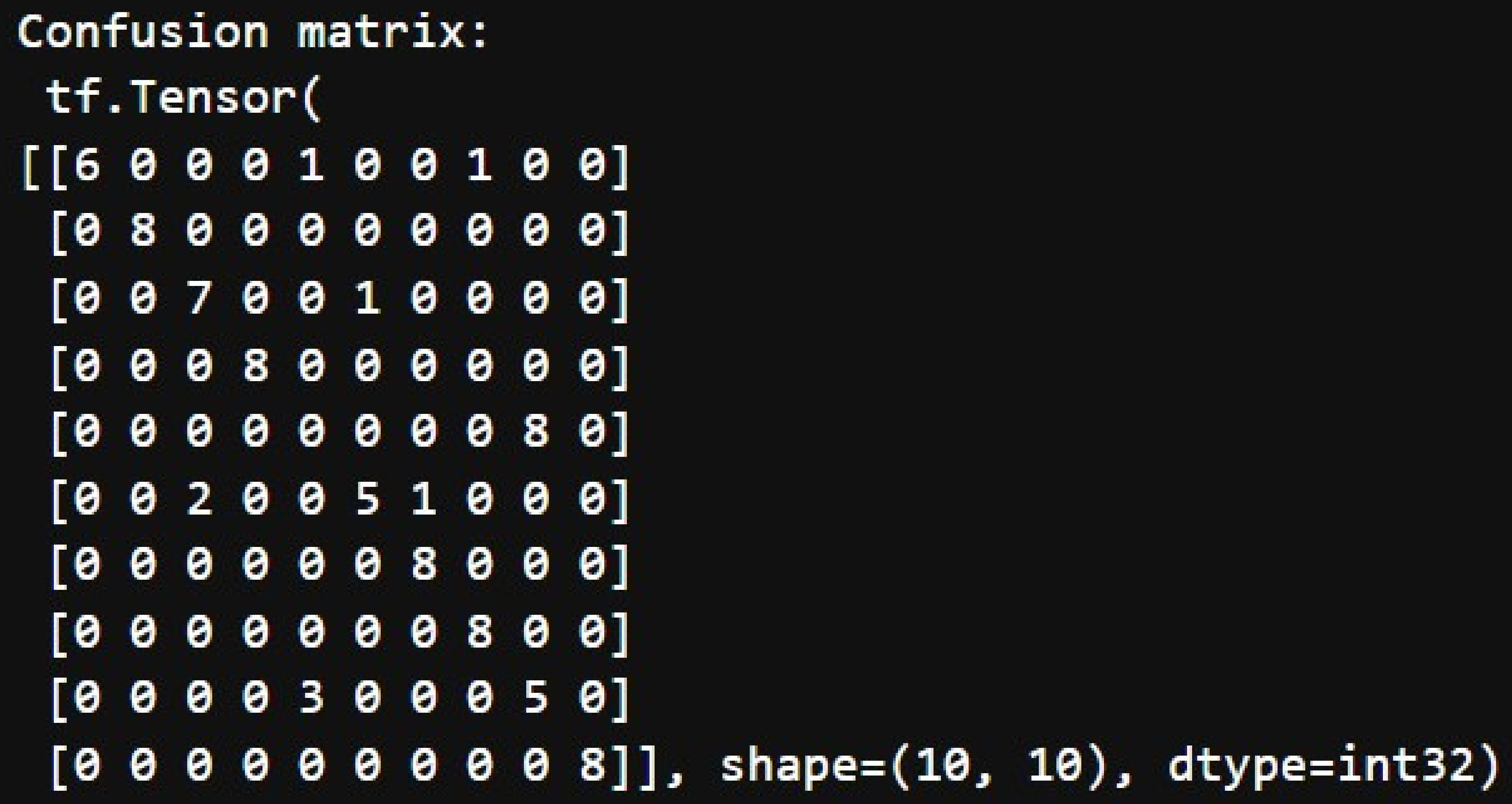
WE EVALUATE THE MODEL'S PERFORMANCE ON THE TEST SET USING ACCURACY, PRECISION, RECALL, AND A CLASSIFICATION REPORT.

THE ANN MODEL ACHIEVES HIGH ACCURACY, PRECISION, AND RECALL SCORES ON THE TEST SET, OUTPERFORMING THE LOGISTIC REGRESSION MODEL.

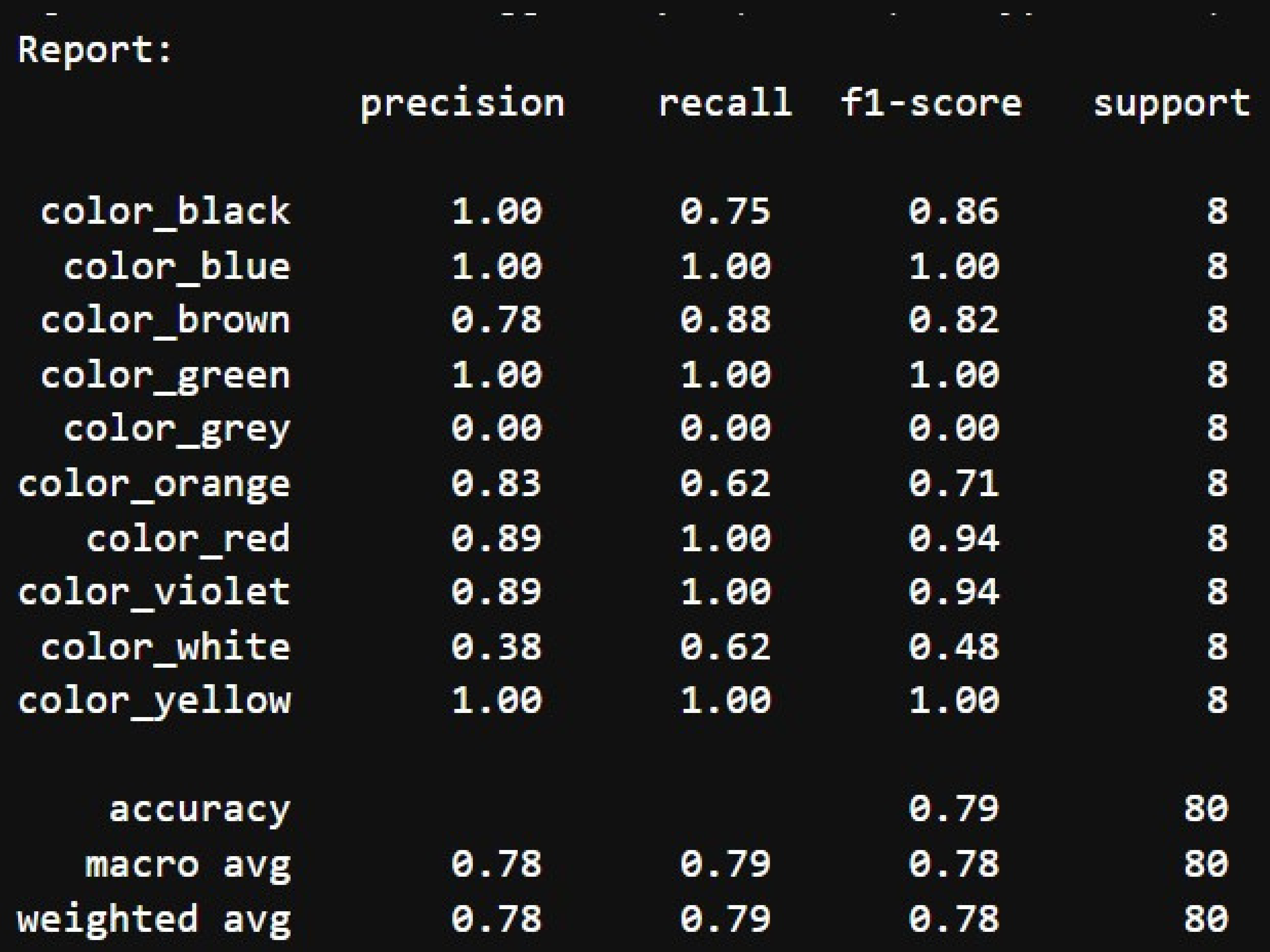
```python
print('first model','-'*50)


# Create a new instance of a Sequential model
model = Sequential()

# Add a fully connected layer to the model with 64 neurons and a ReLU activation function
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))

# Add another fully connected layer to the model with a number of neurons equal to the number of unique classes in y_encoded, and a softmax activation function
model.add(Dense(y_encoded.shape[1], activation='softmax'))

# Compile the model using categorical cross-entropy loss, Adam optimizer, and accuracy metric
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model on the training data for 50 epochs with a batch size of 32, and use validation data to monitor performance
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
```



```python
# Evaluate the model on the test data to estimate its performance on unseen data
loss, accuracy = model.evaluate(X_test, y_test)

# Print the test accuracy of the model as a percentage
print('Test accuracy: %.2f' % (accuracy*100))

# Calculate precision and recall
y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test.values, axis=1)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
print('Precision: %.2f' % precision)
print('Recall: %.2f' % recall)

# Generate classification report
target_names = y_test.columns
print('Confusion matrix:\n', tf.math.confusion_matrix(y_true,y_pred) )
print('Report: \n',classification_report(y_true, y_pred, target_names=target_names))
```



```
Confusion matrix:
 tf.Tensor(
[[6 0 0 0 1 0 0 1 0 0]
 [0 8 0 0 0 0 0 0 0 0]
 [0 0 7 0 0 1 0 0 0 0]
 [0 0 0 8 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 8 0]
 [0 0 2 0 0 5 1 0 0 0]
 [0 0 0 0 0 0 8 0 0 0]
 [0 0 0 0 0 0 0 8 0 0]
 [0 0 0 0 3 0 0 0 5 0]
 [0 0 0 0 0 0 0 0 0 8]], shape=(10, 10), dtype=int32)
```



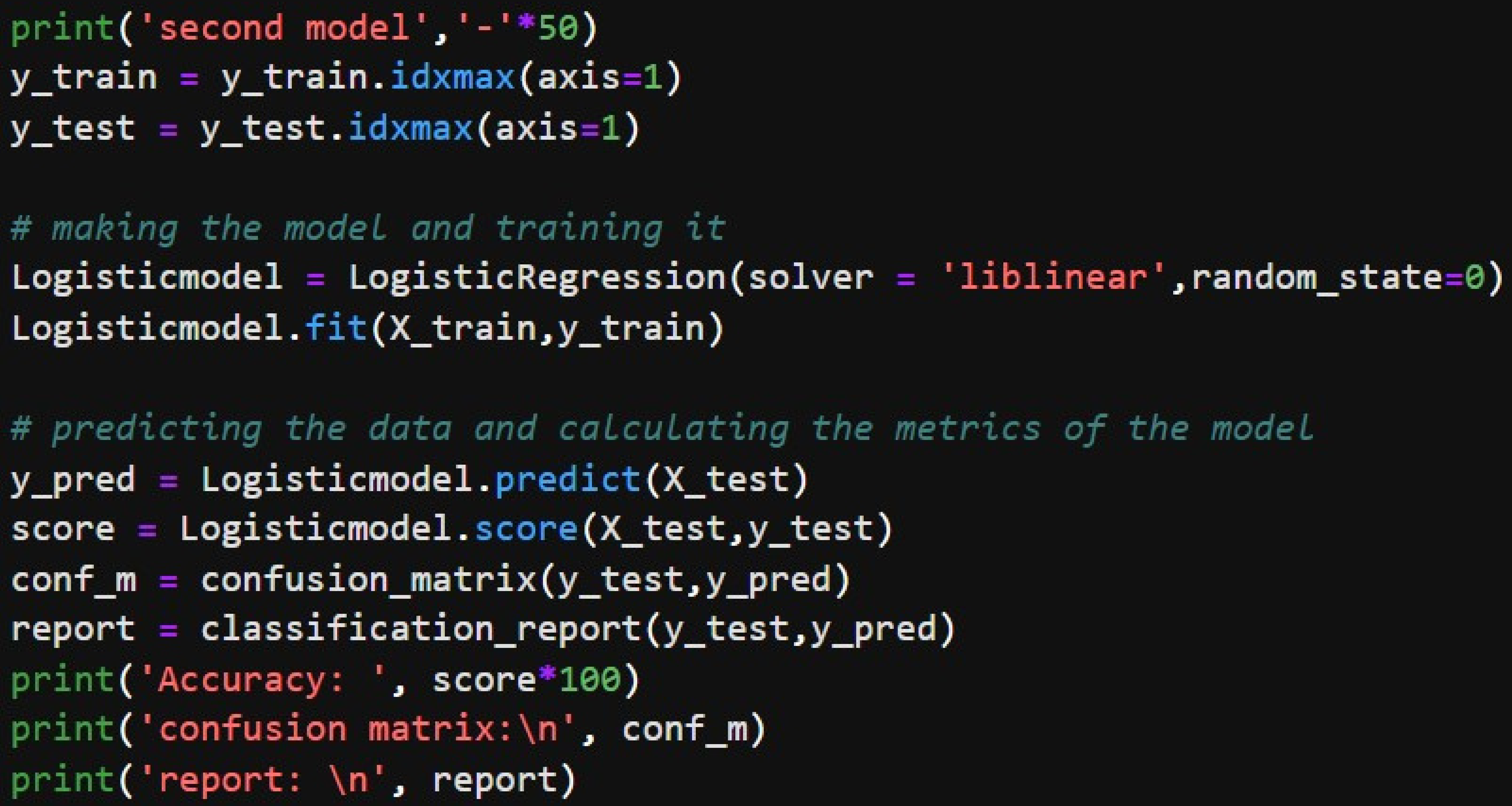
```
Report:
               precision    recall  f1-score   support

 color_black       1.00      0.75      0.86         8
  color_blue       1.00      1.00      1.00         8
 color_brown       0.78      0.88      0.82         8
 color_green       1.00      1.00      1.00         8
  color_grey       0.00      0.00      0.00         8
color_orange       0.83      0.62      0.71         8
   color_red       0.89      1.00      0.94         8
color_violet       0.89      1.00      0.94         8
 color_white       0.38      0.62      0.48         8
color_yellow       1.00      1.00      1.00         8

    accuracy                           0.79        80
   macro avg       0.78      0.79      0.78        80
weighted avg       0.78      0.79      0.78        80
```

LOGISTIC REGRESSION MODEL

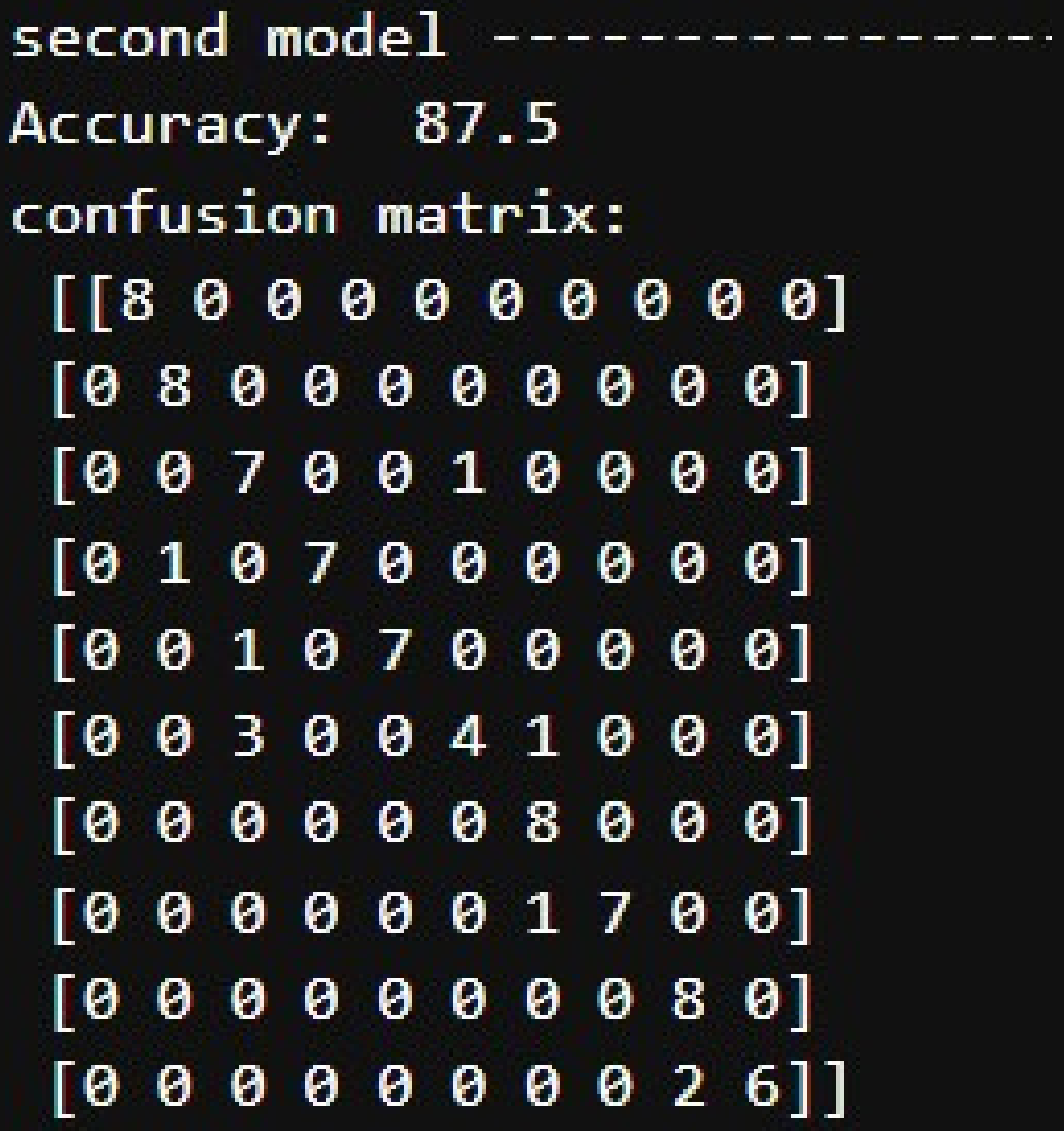
WE FIRST TRAIN A LOGISTIC REGRESSION MODEL ON THE DATA USING SCIKIT-LEARN'S LOGISTICREGRESSION CLASS.

WE FIT THE MODEL ON THE RGB VALUES OF THE IMAGES AND EVALUATE ITS PERFORMANCE ON THE TEST SET USING ACCURACY, PRECISION, RECALL, AND A CLASSIFICATION REPORT.

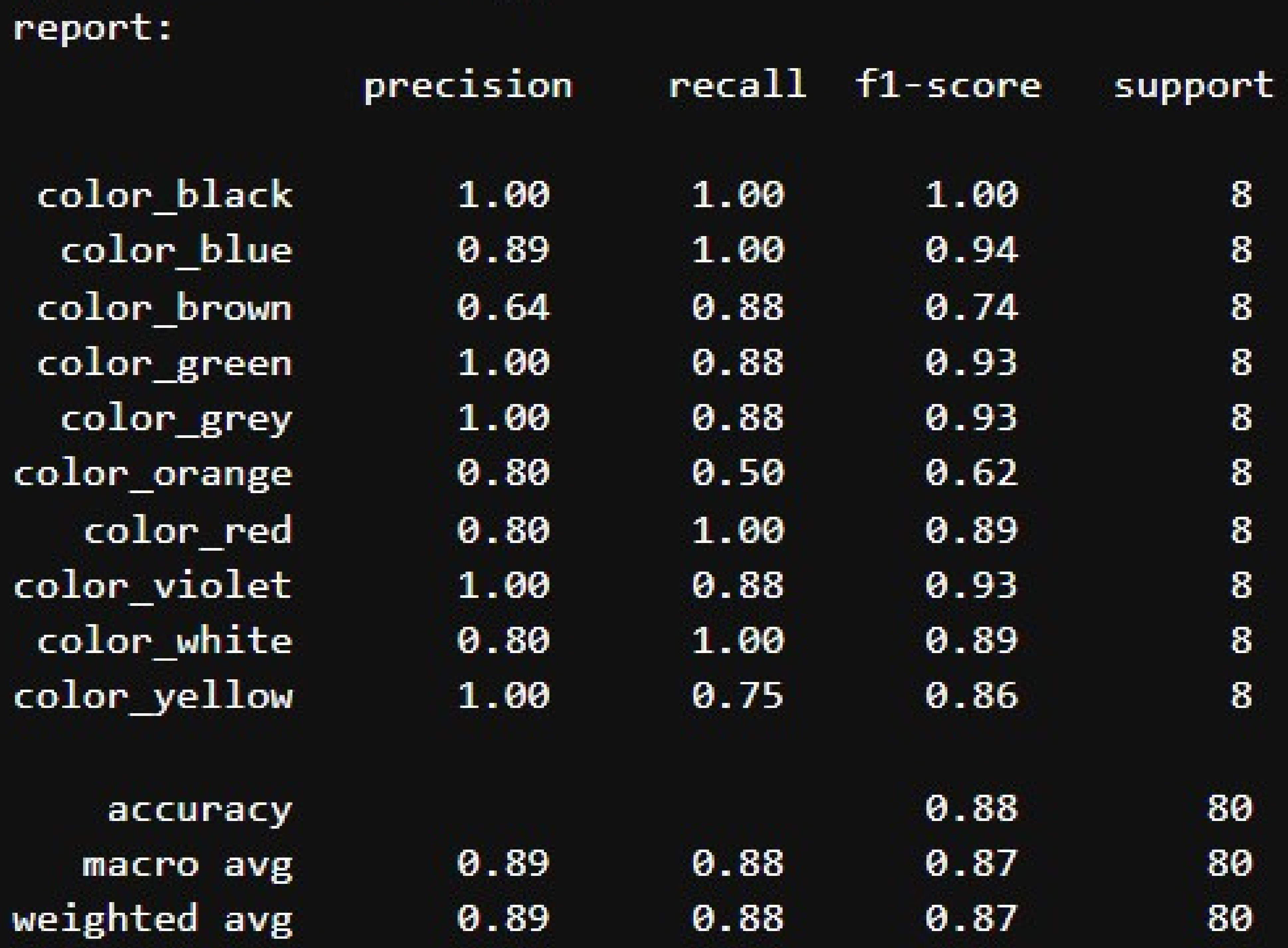
THE LOGISTIC REGRESSION MODEL ACHIEVES HIGH ACCURACY ON THE TEST SET, ALTHOUGH ITS PRECISION AND RECALL SCORES VARY ACROSS DIFFERENT COLORS.



```
print('second model','-'*50)
y_train = y_train.idxmax(axis=1)
y_test = y_test.idxmax(axis=1)

# making the model and training it
Logisticmodel = LogisticRegression(solver = 'liblinear',random_state=0)
Logisticmodel.fit(X_train,y_train)

# predicting the data and calculating the metrics of the model
y_pred = Logisticmodel.predict(X_test)
score = Logisticmodel.score(X_test,y_test)
conf_m = confusion_matrix(y_test,y_pred)
report = classification_report(y_test,y_pred)
print('Accuracy: ', score*100)
print('confusion matrix:\n', conf_m)
print('report: \n', report)
```



```
second model ------------------
Accuracy:  87.5
confusion matrix:
 [[8 0 0 0 0 0 0 0 0 0]
 [0 8 0 0 0 0 0 0 0 0]
 [0 0 7 0 0 1 0 0 0 0]
 [0 1 0 7 0 0 0 0 0 0]
 [0 0 1 0 7 0 0 0 0 0]
 [0 0 3 0 0 4 1 0 0 0]
 [0 0 0 0 0 0 8 0 0 0]
 [0 0 0 0 0 0 1 7 0 0]
 [0 0 0 0 0 0 0 0 8 0]
 [0 0 0 0 0 0 0 0 2 6]]
```



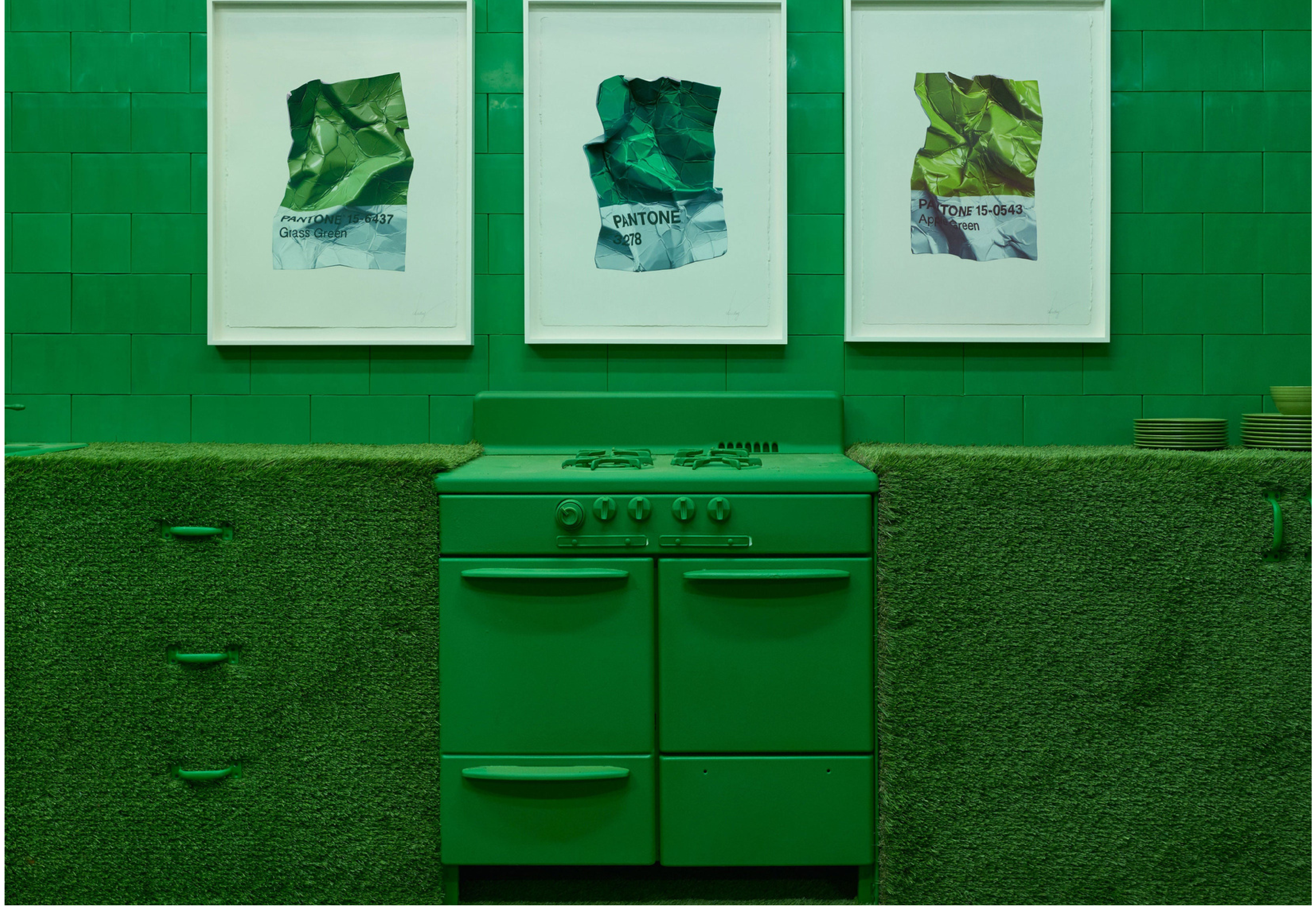
```
report:
              precision    recall  f1-score   support

 color_black       1.00      1.00      1.00         8
  color_blue       0.89      1.00      0.94         8
 color_brown       0.64      0.88      0.74         8
 color_green       1.00      0.88      0.93         8
  color_grey       1.00      0.88      0.93         8
color_orange       0.80      0.50      0.62         8
   color_red       0.80      1.00      0.89         8
color_violet       1.00      0.88      0.93         8
 color_white       0.80      1.00      0.89         8
color_yellow       1.00      0.75      0.86         8

    accuracy                           0.88        80
   macro avg       0.89      0.88      0.87        80
weighted avg       0.89      0.88      0.87        80
```

TESTING ON A NEW IMAGE

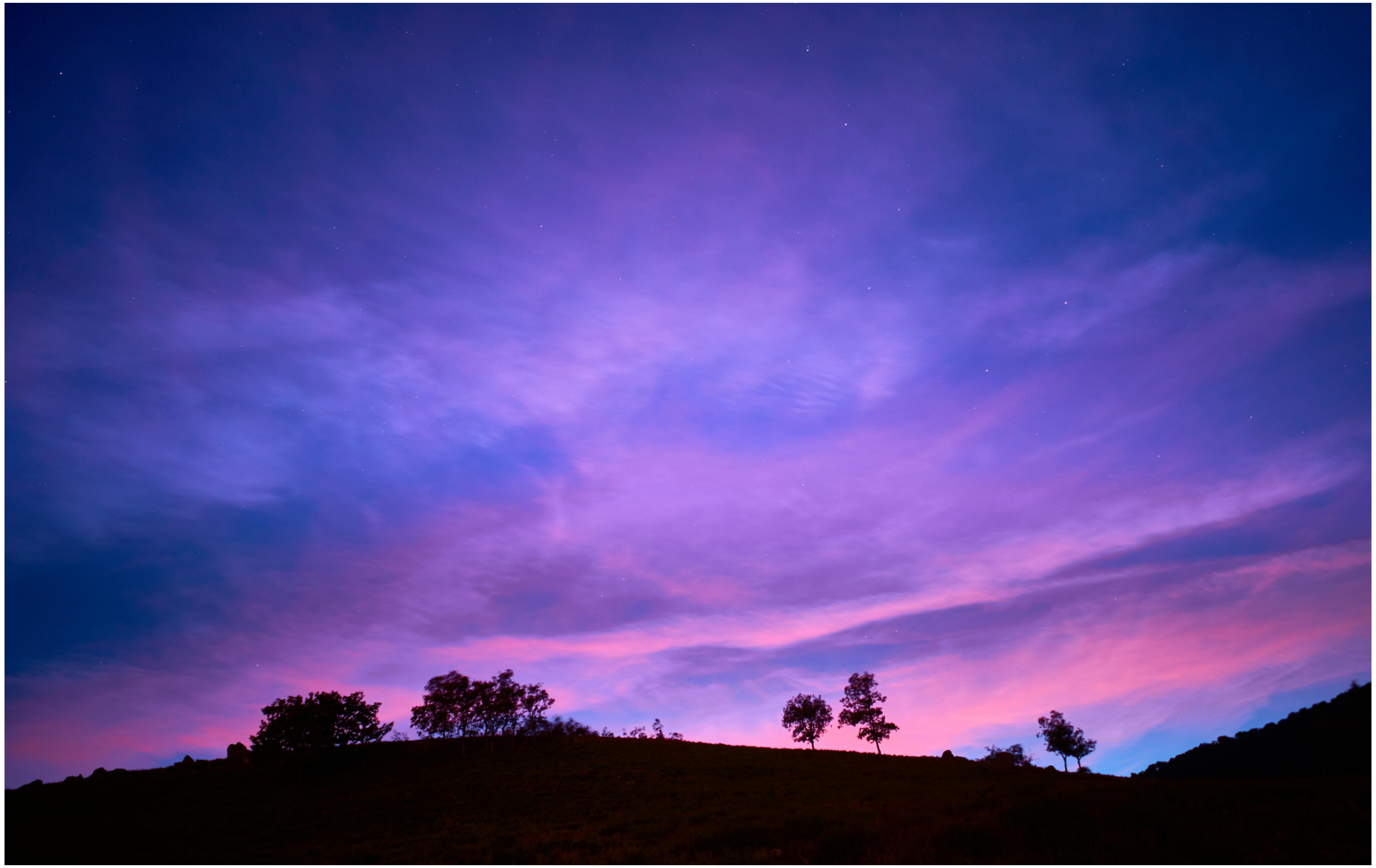
FINALLY, WE ALLOW THE USER TO TEST THE MODELS ON AN IMAGE OF THEIR CHOICE.

THE USER CAN INPUT THE PATH OF THE IMAGE TO TEST, AND THE SCRIPT WILL PREDICT THE COLOR LABEL USING BOTH THE LOGISTIC REGRESSION AND NN MODELS.

TEST RUNS



```
logistic output: color_green
1/1 [==============================] - 0s 33ms/step
ANN output: color_green
```



```
logistic output: color_blue
1/1 [==============================] - 0s 33ms/step
ANN output: color_blue
```

Conclusion

WE EXPLORED TWO MACHINE LEARNING APPROACHES TO COLOR CLASSIFICATION:
LOGISTIC REGRESSION AND ARTIFICIAL NEURAL NETWORKS.

WE TRAINED AND EVALUATED THE MODELS ON A DATASET OF IMAGES AND THEIR DOMINANT COLORS USING SCIKIT-LEARN AND KERAS LIBRARIES.

THE ANN MODEL OUTPERFORMED THE LOGISTIC REGRESSION MODEL IN TERMS OF ACCURACY, PRECISION, AND RECALL ON THE TEST SET. THE SCRIPT ALSO ALLOWS THE USER TO TEST THE MODELS ON A NEW IMAGE, DEMONSTRATING THE PRACTICAL APPLICATIONS OF COLOR CLASSIFICATION IN COMPUTER VISION.

THANK YOU