UKS31983.docx

by

Submission date: 02-May-2023 01:33AM (UTC-0500)
Submission ID: 2081819926
File name: UKS31983.docx (672.53K)
Word count: 1922
Character count: 10865

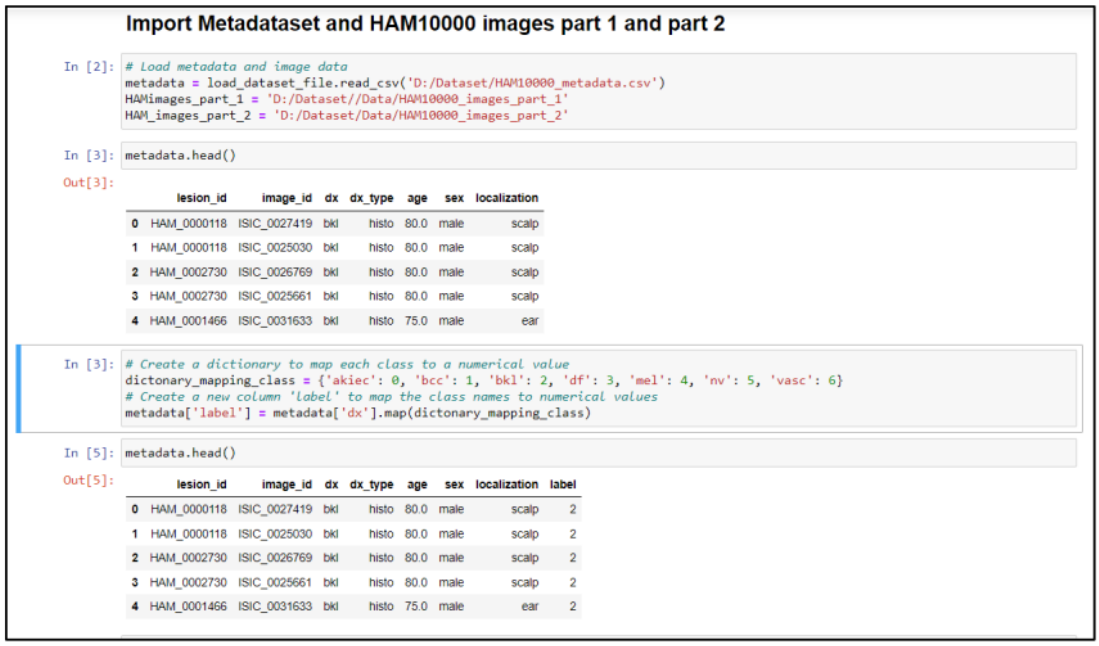
NEURAL COMPUTING AND DEEP LEARNING

Table of Contents

Introduction

Data exploration and visualization are critical phases in any kind of data science task. The study shows how to use Python tools like Pandas, Seaborn, and Keras to execute data exploration or neural network techniques on the HAM skin cancer dataset. The paper also visualizes or defines the steps of cleaning and preprocessing data, creating visualizations to acquire insights, and building and training a convolutional neural network model for skin lesion classification. The strategies and methodology described in this paper are capable of being applied to various picture datasets and can help medical practitioners construct highly precise and effective skin lesion categorization systems.

Exploratory Data Analysis and data visualization



Import Metadataset and HAM10000 images part 1 and part 2

In [2]:
```
# Load metadata and image data
metadata = load_dataset_file.read_csv('D:/Dataset/HAM10000_metadata.csv')
HAMimages_part_1 = 'D:/Dataset//Data/HAM10000_images_part_1'
HAM_images_part_2 = 'D:/Dataset/Data/HAM10000_images_part_2'
```

In [3]:
```
metadata.head()
```

Out[3]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [3]:
```
# Create a dictionary to map each class to a numerical value
dictonary_mapping_class = {'akiec': 0, 'bcc': 1, 'bkl': 2, 'df': 3, 'mel': 4, 'nv': 5, 'vasc': 6}
# Create a new column 'label' to map the class names to numerical values
metadata['label'] = metadata['dx'].map(dictonary_mapping_class)
```

In [5]:
```
metadata.head()
```

Out[5]:

	lesion_id	image_id	dx	dx_type	age	sex	localization	label
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	2
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	2
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	2
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	2
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	2

Figure 1: HAM metadata, and image datasets are imported here

(Source: Self-Created)

The included codes are utilized for importing the metadata dataset as well as image dataset from the user system to the jupyter notebook platform. Besides, the head function enables to display the first five data from the dataset. This code assists in loading the image or metadata data from a dataset for additional processing or analysis (Fraiwan and Faouri, 2022). The picture data includes

photographs of skin lesions, but the metadata file also includes information on the patient's age, gender, and lesion kind. The code provided lets users explore and analyze the dataset utilizing different data analysis or neural networks approaches by importing both metadata or images.

The programme is building a dictionary which helps to assign a number to each category of skin lesion. Strings, such as 'akiec', 'bcc', 'bkl', etc., are employed to indicate the many kinds of skin lesions. Every one of these strings receives an integer number by the dictionary_mapping_class variable. The Map' function of the 'dx' column of the metadata' dataframe, which implements the dictionary_mapping_class to every component of the 'dx' column, serves to generate the new column (Fatima, 2023). As a consequence, a new column is generated with the relevant numerical value for each class of skin lesion.

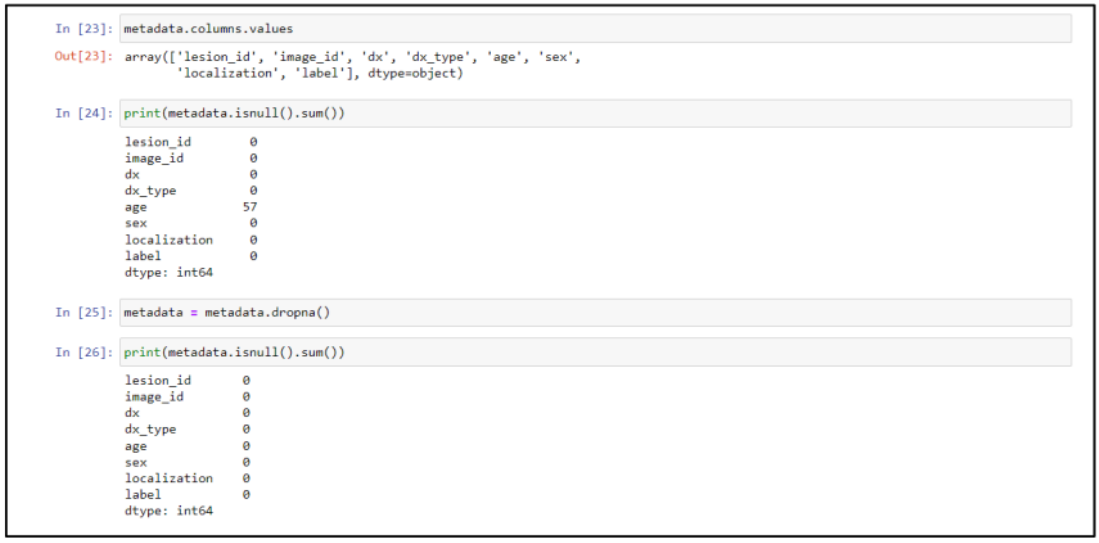


Figure 2: Removing null values from the HAM dataset

(Source: Self-Created)

Here, the metadata file has been cleaned by utilizing the *"dropna ()"* function that helps to get accurate results after performing the neural network methods corresponding to the task.

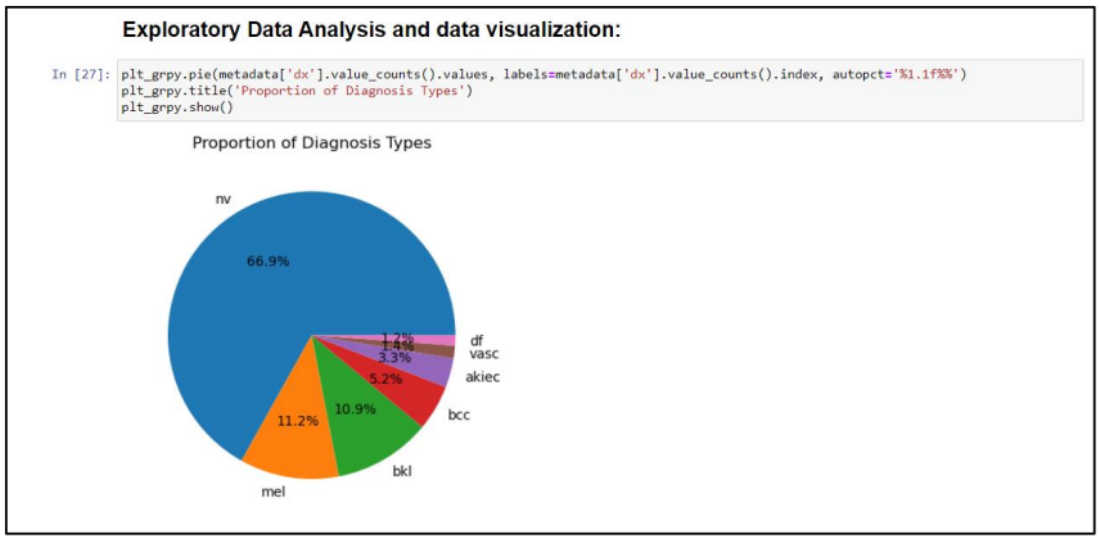


Figure 3: Pie Chart for proportion of Diagnosis types

(Source: Self-Created)

This code creates a pie chart that displays the percentage for every diagnosis category within the dataset. As a result, this provides an overall perspective of the distribution of multiple kinds of skin lesions within the dataset, this may be helpful when performing determining the types of lesion types (Purnama *et al*. 2019). The efficacy of machine learning algorithms based on the data set might be impacted by the dataset's imbalance or inequalities, and can be estimated utilizing this data.

```
In [29]: ty = load_dataset_file.crosstab(metadata['dx'], metadata['sex'])
         ty.plot(kind='bar', stacked=True)
         plt_grpy.title('Number of Images for Each Diagnosis Type by Sex')
         plt_grpy.xlabel('Diagnosis Type')
         plt_grpy.ylabel('Count')
         plt_grpy.show()
```

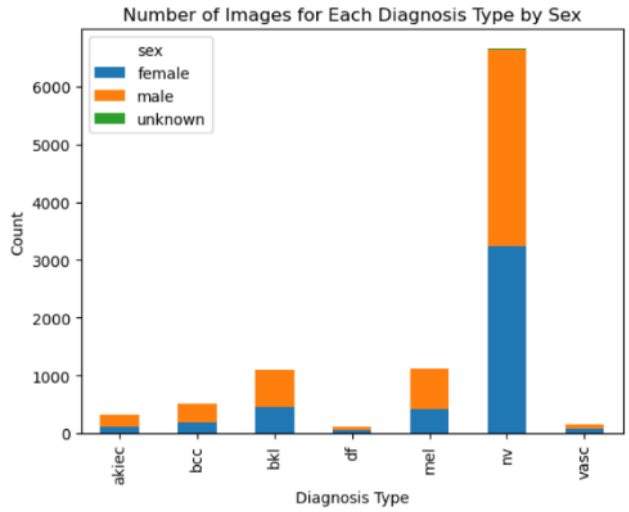


Figure 4: Bar charts between the numbers of images for each diagnosis type by sex

(Source: Self-Created)

The above offered code appears to import a dataset file and construct a crosstabulation of the counts of photos for each diagnosis category by sex using the Python module pandas. The generated crosstab data is then used to produce a stacked bar chart. As a graphical representation of the distribution of images among diagnosis kinds as well as sex, the resulting chart might prove helpful in classifying skin lesions (Pranav *et al*. 2021). Since, it will become easier to identify any common patterns or abnormalities within information which might have to be corrected to produce accurate and objective categorization findings.

```
In [30]: sns_grpy.kdeplot(data=metadata, x='age', hue='sex')
         plt_grpy.title('Distribution of Ages by Sex')
         plt_grpy.xlabel('Age')
         plt_grpy.ylabel('Density')
         plt_grpy.show()
```

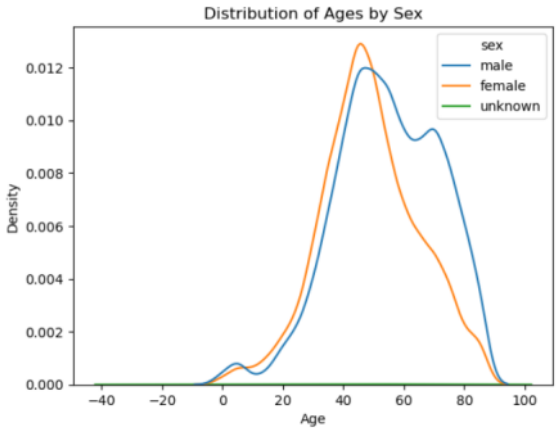


Figure 5: Line plot for the distribution of ages by sex

(Source: Self-Created)

The code "*sns_grpy.kdeplot(data=metadata, x='age, hue='sex')*" employs the Seaborn library in order to produce a kernel density estimate (KDE) plot of the ages within the dataset, segregated by sex. The above code could be employed for skin lesion classification to investigate the age distribution in the dataset as well as to determine whether there are any age disparities among male and female patients. This data might be applied to help create a skin lesion classification system that requires both sexes into account.

Implementation

Data Pre-processing

Data Preprocessing

In [6]:
```
# Function to read images and resize them to 224x224 pixels
def img_visualization_funct(filepath):
    imges = cv2.imread(filepath)
    imges = cv2.resize(imges, (224,224))
    return imges
```

In [7]:
```
# load the HAM images as well as store them within a numpy array
HAMimages = []
for HAMimage_name in metadata['image_id']:
    if os.path.isfile(os.path.join(HAMimages_part_1, HAMimage_name + '.jpg')):
        imges = img_visualization_funct(os.path.join(HAMimages_part_1, HAMimage_name + '.jpg'))
        HAMimages.append(imges)
    else:
        imges = img_visualization_funct(os.path.join(HAM_images_part_2, HAMimage_name + '.jpg'))
        HAMimages.append(imges)
HAMimages = math_cal.array(HAMimages)
```

In [8]:
```
# Convert the images and labels to numpy arrays
HAMimages = math_cal.array(HAMimages)
HAMlabels = math_cal.array(metadata['label'])
```

In [9]:
```
# HAM image and CSV dataset  has been splited between trainning and testing format
Xtrainset, Xtestset, ytrainset, ytestset = splittingprocess(HAMimages, HAMlabels, test_size=0.2, random_state=42)
```

Figure 6: Performing the data preprocessing Steps

(Source: Self-Created)

The code portion provided creates an img_visualization_funct Python function that accepts an image file path as input as well as returns a resized picture. This function just adjusts the picture into 224x224 pixels in size. After that, the code creates an empty list called "HAMimages" before it loops through each picture ID in the "metadata" dictionary. It verifies to determine if the picture file appears in one of two folders ("HAMimages_part_1" or "HAMimages_part_2") before loading it with the "img_visualization_funct" function. The picture is added to the "HAMimages" list. The "HAMimages" list has been converted to a numpy list utilizing the "math_cal.array" function after iterating through all of the images.

This code is useful for users that require the ability to load as well as manipulate photos in a machine learning or machine vision programmed while interacting with the HAM dataset (Li *et al*. 2021). Users may quickly conduct computations on the image data, such as scaling, motion, or trimming, as well as train or test machine learning algorithms by importing the pictures into a numpy array. For this project, splitting codes has been employed to separate HAM pictures and their accompanying labels into testing and training sets.

Neural network algorithms

Neural Network Algorithms

In [10]:
```
# Execute codes for CNN model
cnnmodel = mdl.Sequential([
    lyr.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    lyr.MaxPooling2D((2, 2)),
    lyr.Conv2D(64, (3, 3), activation='relu'),
    lyr.MaxPooling2D((2, 2)),
    lyr.Conv2D(128, (3, 3), activation='relu'),
    lyr.MaxPooling2D((2, 2)),
    lyr.Flatten(),
    lyr.Dense(128, activation='relu'),
    lyr.Dense(7, activation='softmax')
])
```

In [11]:
```
cnnmodel.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 flatten (Flatten)           (None, 86528)             0
```

Figure 7: Build the CNN neural network algorithm

(Source: Self-Created)

For picture classification, this code creates a Convolutional Neural Network (CNN) model utilizing the Keras Sequential API. The algorithm's architecture comprises numerous convolutional as well as pooling layers, then follows fully connected layers, and predicts its chance distribution across the seven classes utilizing the function of SoftMax activation of the resultant layer. Users can profit from this programmed by employing it to create a CNN model on an individual image dataset for a particular sorting purpose (Younis, Bhatti and Azeem, 2019). To improve effectiveness on their dataset, they can easily modify the design of the model by modifying the number of filtration systems, kernel size, activation function, and various other hyperparameters. The model that has been trained is able to be utilized to make predictions about fresh images.

```
In [12]: cnnmodel.compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])

In [14]: # Here CNN model has been trained with respect to the test dataset
         cnnmodel_model_hist = cnnmodel.fit(Xtrainset, ytrainset, epochs=10, validation_data=(Xtestset, ytestset))

         Epoch 1/10
         251/251 [==============================] - 292s 1s/step - loss: 9.9481 - accuracy: 0.6516 - val_loss: 1.0236 - val_accuracy: 0.
         6695
         Epoch 2/10
         251/251 [==============================] - 297s 1s/step - loss: 0.9297 - accuracy: 0.6754 - val_loss: 0.9296 - val_accuracy: 0.
         6720
         Epoch 3/10
         251/251 [==============================] - 292s 1s/step - loss: 0.8978 - accuracy: 0.6820 - val_loss: 0.9117 - val_accuracy: 0.
         6770
         Epoch 4/10
         251/251 [==============================] - 294s 1s/step - loss: 0.8536 - accuracy: 0.6970 - val_loss: 0.9445 - val_accuracy: 0.
         6795
         Epoch 5/10
         251/251 [==============================] - 288s 1s/step - loss: 0.8209 - accuracy: 0.7068 - val_loss: 0.9733 - val_accuracy: 0.
         6700
         Epoch 6/10
         251/251 [==============================] - 277s 1s/step - loss: 0.8082 - accuracy: 0.7234 - val_loss: 0.9547 - val_accuracy: 0.
         6735
         Epoch 7/10
         251/251 [==============================] - 278s 1s/step - loss: 0.8134 - accuracy: 0.7244 - val_loss: 0.9929 - val_accuracy: 0.
         6790
         Epoch 8/10
         251/251 [==============================] - 282s 1s/step - loss: 0.7359 - accuracy: 0.7410 - val_loss: 1.0822 - val_accuracy: 0.
         6410
         Epoch 9/10
         251/251 [==============================] - 280s 1s/step - loss: 0.6542 - accuracy: 0.7664 - val_loss: 1.0841 - val_accuracy: 0.
         6830
         Epoch 10/10
         251/251 [==============================] - 275s 1s/step - loss: 0.6025 - accuracy: 0.7857 - val_loss: 1.1921 - val_accuracy: 0.
         6725
```

Figure 8: Fitting the neural network model

(Source: Self-Created)

The code snippet creates a CNN model utilizing the 'adam' optimizer,'sparse_categorical_crossentropy' loss function, with 'accuracy' metric. The model is subsequently trained on a training dataset (Xtrainset and ytrainset) for 10 epochs before being validated using the test dataset (Xtestset and ytestset). This is critical to train a CNN model with proper hyperparameters, optimizer, and loss function in order to achieve high levels of precision in tasks such as image classification. In multi-class classification tasks, the'sparse_categorical_crossentropy' loss function is typically utilised, and the 'accuracy' measure is employed to assess the model's efficiency. Apart from that, building a model on bigger data sets that includes additional epochs may improve accuracy, however it can also result in overfitting (Purnama *et al*. 2019). The validation_data parameter prevents over-fitting through enabling the model to be validated and tested on a dataset that has not been employed during the training phase. In general, this code snip is useful for users since it allows them to develop a CNN model for identifying images tasks, assess its efficacy, as well as minimize overfitting.

Results

```
In [16]: # Evaluate metrics
modelaccuracy = accuracy_score(ytestset, modelpredictval)
modelf1val = f1_score(ytestset, modelpredictval, average='weighted')
modelprec = precision_score(ytestset, modelpredictval, average='weighted')
modelrecallval = recall_score(ytestset, modelpredictval, average='weighted')
confusmatrix = confusion_matrix(ytestset, modelpredictval)

In [17]: # Print evaluation metrics
print('CNN Model Accuracy is Shown as:', modelaccuracy)
print('CNN Model F1 Score is shown as :', modelf1val)
print('CNN Model Precision is  Shown as :', modelprec)
print('CNN model Recall value is shown as :', modelrecallval)

CNN Model Accuracy is Shown as: 0.672491263105342
CNN Model F1 Score is shown as : 0.630867025073616 5
CNN Model Precision is  Shown as : 0.6115257282086123
CNN model Recall value is shown as : 0.672491263105342
```

Figure 9: Model Accuracy, f1-score, precision and confusion matrix value

(Source: Self-Created)

The code computes several evaluation measures for a machine learning model's forecasts on a test set, such as accuracy, F1 score, precision, recall, and confusion matrix. These metrics aid in determining how effectively the model works on unknown data and offer insight into places in which the model might require to be improved. The accuracy score is the proportion of accurately predicted cases among all instances. The F1 score combines precision and recall to provide a comprehensive assessment of the model's performance. Apart from that, the accuracy score of the built model is shown as 67%, f1 score is 63% as well as precision score is 61%.

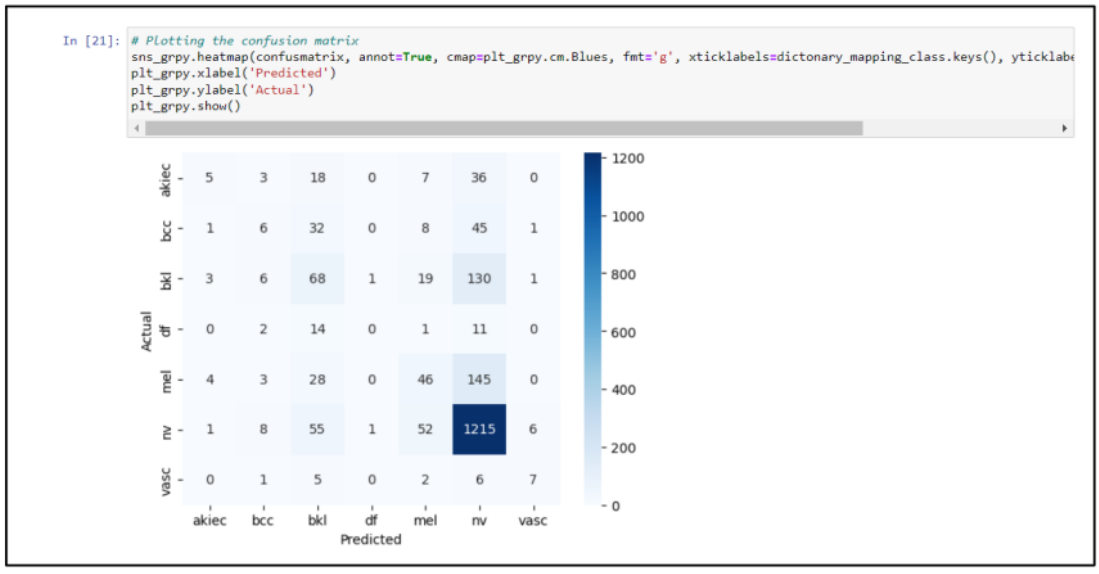


Figure 10: Plotting Confusion matrix

(Source: Self-Created)

Employing the Seaborn library, the code generates a confusion matrix. It accepts a confusion matrix as its input as well as a heatmap containing labeled values. The cmap argument specifies the color scheme, the fmt parameter specifies the number format, as well as xticklabels and yticklabels are keys in a dictionary that translates class names to labels. It can assist users in understanding how effectively their model is functioning and identifying areas for improvements. The addition of a heatmap or annotated values for displaying the confusion matrix allows for less difficult for users to understand the outcomes while recognizing trends in the data.

```
In [22]: # Print classification report
         print(classification_report(ytestset, modelpredictval, target_names=dictonary_mapping_class.keys()))

                      precision    recall  f1-score   support

               akiec       0.36      0.07      0.12        69
                 bcc       0.21      0.06      0.10        93
                 bkl       0.31      0.30      0.30       228
                  df       0.00      0.00      0.00        28
                 mel       0.34      0.20      0.25       226
                  nv       0.77      0.91      0.83      1338
                vasc       0.47      0.33      0.39        21

            accuracy                           0.67      2003
           macro avg       0.35      0.27      0.29      2003
        weighted avg       0.61      0.67      0.63      2003


In [ ]:
```

Figure 11: Generate Classification report based on the CNN model values

(Source: Self-Created)

The code generates a report on a machine learning model's classification performance using the scikit-learn library's classification_report function. For every category in the dataset, the report contains metrics such as precision, recall, and F1-score. The summary of the report is displayed on the console using the print function. The ytestset argument holds the test dataset's real labels, modelpredictval provides the model's forecasted labels, while dictionary_mapping_class.keys () provides the class labels according to the numerical labels (Pranav *et al*. 2021). This code is useful for users since it gives them a thorough picture of how well their machine learning model performs on each class. This data can be employed to refine the algorithm or to select an alternative model or technique that works higher on particular groups.

Conclusion

It can be concluded that the study presents the use of the HAM10000 dataset to develop a Convolutional Neural Network (CNN) model for skin lesion categorization. The paper describes in detail the code implementation for importing metadata and image datasets, as well as pre-

processing processes and neural network methods. The article uses several visualizations and approaches, such as pie charts, bar charts, and line graphs, to investigate the dataset and provide insights regarding the distribution of photos for each diagnosis type by sex and age distribution. Overall, the study presents a thorough approach to the development of a CNN model for skin lesion classification and is a valuable resource for researchers and practitioners working in this field.

Reference Lists

Fatima, A., 2023. The Skin Lesion Detection and Classification Using Deep Learning. *International Journal for Electronic Crime Investigation*, *7*(1), pp.39-48.

Fraiwan, M. and Faouri, E., 2022. On the Automatic Detection and Classification of Skin Cancer Using Deep Transfer Learning. *Sensors*, *22*(13), p.4963.

Li, Y., Wang, D., Xu, Z. and Zhao, Z., 2021, October. Intelligent Skin Cancer Detection System Based on Convolutional Neural Networks. In *Proceedings of the 2nd International Symposium on Artificial Intelligence for Medicine Sciences* (pp. 188-198).

Pranav, M.V., Koushik, C., AV, S.M. and Ganapathy, S., 2021, November. Analyzing the Diagnostic Efficacy of Deep Vision Networks for Malignant Skin Lesion Recognition. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)* (Vol. 1, pp. 194-199). IEEE.

Purnama, I.K.E., Hernanda, A.K., Ratna, A.A.P., Nurtanio, I., Hidayati, A.N., Purnomo, M.H., Nugroho, S.M.S. and Rachmadi, R.F., 2019, November. Disease classification based on dermoscopic skin images using convolutional neural network in teledermatology system. In *2019 International conference on computer engineering, network, and intelligent multimedia (CENIM)* (pp. 1-5). IEEE.

Younis, H., Bhatti, M.H. and Azeem, M., 2019, December. Classification of skin cancer dermoscopy images using transfer learning. In *2019 15th International Conference on Emerging Technologies (ICET)* (pp. 1-4). IEEE.

UKS31983.docx

ORIGINALITY REPORT

2%	2%	%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	1%
2	www.wjgnet.com Internet Source	1%
3	www.hindawi.com Internet Source	<1%
4	www.mdpi.com Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		