كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
Helwan University

HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Computer Science Department

# Skin Lesions Detection

A graduation project dissertation by:

Hossam Hassan Kamal Mohamed ( 20180190 )

Mostafa Hamdy Khalil Mahmoud ( 20180593 )

Mohamed Abdallah Youssef Ibrahim (20180517 )

Heba Allah Ahmed Mohamed Abdelazim(20180672)

Nada Khater Abdo Ismail ( 20180650 )

Mai Hamada Elsayed Ahmed ( 20180635 )

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

Assoc.Prof. Ensaf Hussien Mostafa

June 2022

كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
Helwan University

جامعة حلوان
كلية الحاسبات والذكاء الإصطناعي
قسم علوم الحاسب

# كشف الآفات الجلدية

رسالة مشروع تخرج مقدمة من:

حسام حسن كمال محمد (20180190)

مصطفى حمدي خليل محمود (20180593)

محمد عبدالله يوسف إبراهيم (20180517)

ندى خاطر عبده إسماعيل (20180650)

هبة الله أحمد محمد عبدالعظيم (20180672)

مي حماده السيد أحمد (20180635)

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء الإصطناعي،
بقسم علوم الحاسب، كلية الحاسبات والذكاء الإصطناعي، جامعة حلوان

تحت إشراف:

أ.م. إنصاف حسين مصطفى

يونيو / حزيران 2022

## ABSTRACT

Skin cancer is more common than any other type of cancer. It is One of the most commonly occurring cancers in humans and affects More than one million people worldwide every year. It causes more Deaths than heart disease.
A timely diagnosis of cancer increases the patient's chance of recovery, Especially in the early stages of the disease machine learning helps in That. So, we aim to make people Know whether this skin lesion is cancerous or not, so we implemented a Mobile application to detect skin cancer through a photo taken by the Mobile phone camera and user enters some personal information.
Our application helps detect the lesion and gives the patient the type of Lesion as one of following: Basal Cell Carcinoma (BCC), Squamous Cell Carcinoma (SCC), Actinic Keratosis (AK), Seborrheic Keratosis (SEK), Melanoma (MOLE), and Nevus (NEVI).
We've looked at some previous related work and learned from their Experience and trying more than one scenario to construct our machine Learning model using different datasets from different resources and got what we needed from them, our main was PAD-UFES-20.
We focused on our sixth scenario to construct our model, we removed The weighted average as specified in Scenario 4 and passed all features Of (Images + Meta) to a dense layer with "1024 neurons", activation Function "Relu", KernelRegularizerL1L2 "L1 = 1e-5, L2 = 1e-4", Dropout layer with "0.9", and another dense layer with "6 neurons", Activation function "SoftMax" Then, the model is trained for "200" Epochs using the "Adam" optimizer with learning rate of "0.0001", Epsilon "0.01", and batch size of "16". Accuracy: Images only: 63.9% Metadata only: 85.6% , Images + Metadata: 95.6% .Then we ran it on different datasets and test its efficiency. So, we added it to our Application which developed using a programming language "Flutter".
We hope that our application has contributed to maintaining the health Of communities and has increased awareness and spread of correct Treatment methods. At the Future, We'll work on our application and Model to develop them much more so that they can deal with more Different types of lesions and different regions whole wide the world.

**Contents:**

# Chapter 1: Introduction

## 1.1Overview

Skin cancer is more common than any other type of cancer. It is one of the most commonly occurring cancers in humans and affects more than one million people worldwide every year. It causes more deaths than heart disease.
A timely diagnosis of cancer increases the patient's chance of recovery, especially in the early stages of the disease. So, we aim to make people know whether this skin lesion is cancerous or not, so we implemented a mobile application to detect skin cancer through a photo taken by the mobile phone camera.
Our app helps detect the lesion and gives the patient more information about it.

## 1.2 Problem Statement

1 in 5 Americans will develop skin cancer by the age of 70.
More than 2 people die of skin cancer in the U.S. every hour.
Having 5 or more sunburns doubles your risk for melanoma.
When detected early, the 5-year survival rate for melanoma is 99 percent.
-Those mentioned stats and facts are according to *(The Skin Cancer Foundation).*
We want to develop a mobile app that helps users detect skin cancer early, identify its type and reduce its severity, and the user should consult their dermatologist for early treatment.
We want to achieve high accuracy beyond the applications that were developed before our application and achieve the principle of ease of use.

## 1.3 Objectives

- Develop an easy system that is friendly with the user.
- Give the user all the info that is available about the lesion.
- Give the user advice about the lesion that can help them until they contact the doctor.

## 1.4 Purpose

Through using our app, the patient can know if the lesion is malignant or not, and if it is, what kind of it?

## 1.5 Scope

(Skin lesion Classification Mobile Application):
- The application is aims to help Patients by taking a photo from Their smartphones detecting the skin lesion and classifying it to one of the following 6 Types:
- Basal Cell Carcinoma (BCC), Squamous Cell Carcinoma (SCC), Actinic Keratosis (AK), Seborrheic Keratosis (SEK), Melanoma (MOLE), and Nevus (NEVI).
- Ease for the user to take a picture from the smartphone from Home without wasting time.
- Correct lesion classification gives the user ease of dealing with the software system.

## 1.6 General Constraints

They are factors that can affect quality, performance and overall project Success.
- Developing time constraints:
Time allowed for analyzing, developing and testing the project which May be 5 months At least.
- Technology environment constraints:
The user must be connected to the internet when using the application.
- Software constraints:
AI Models have a limitation time to run.
- Hardware constraints:
The user must have a smartphone to deal with the application.

## 1.7 project planning

### 1.7.1 Feasibility Study

Skin cancer is the out-of-control growth of abnormal cells in the epidermis, the outermost skin layer, caused by unrepaired DNA damage that triggers mutations.
These mutations lead the skin cells to multiply rapidly and form malignant tumors.
The main types of skin cancer are basal cell carcinoma (BCC), squamous cell carcinoma (SCC), melanoma and Merkel cell carcinoma (MCC).

The two main causes of skin cancer are the sun’s harmful ultraviolet (UV) rays and the use of UV tanning beds. The good news is that if skin cancer is caught early, your dermatologist can treat it with little or no scarring and high odds of eliminating it entirely.

Often, the doctor may even detect the growth at a precancerous stage, before it has become a full-blown skin cancer or penetrated below the surface of the skin.
We hope that our mobile application will help the patient to detect skin cancer early and consult his dermatologist.

-Cost: The application has no cost because it is completely free, that make the people knows he's infected or not and the type of skin cancer and the right treatment Without paying any cost.

-Accessibility: Users will be able to access the application anytime and there will be no time constraints.

-Communication: The application provides an easy way of communication between user and the doctor.

-Time: Save user’s time instead of Spending a lot of time at the Doctor's clinic. The user can use the app from home without lost in time.

- Awareness: Spread awareness of information and correct treatment that may be unknown to users and clear the fear among users of using medical programs that help patients.

-emergency: able to serve many users in emergency situations for finding a suitable treatment or visiting the doctor in serious cases.

### 1.7.2 Estimated Cost

Hardware materials:

- personal laptop or personal computer connected to the internet

Software materials:

- Python language
- Dart
- Mobile application and library like flask

### 1.7.3 Gantt Chart

| **Task's name** | **Start (dd/mm/yyyy)** | **End (dd/mm/yyyy)** | **Duration (d)** |
|---|---|---|---|
| **Project Status** | **10/10/2021** | **8/4/2022** | **180** |
| **Pre-definition** | **10/10/2021** | **9/11/2021** | **30** |
| -Ideas brainstorming and searching | 10/10/2021 | 24/10/2021 | 15 |
| -Idea analysis | 25/10/2021 | 2/11/2021 | 9 |
| -Idea evaluation | 3/11/2021 | 8/11/2021 | 5 |
| -Choose and submit idea | 9/11/2021 | 9/11/2021 | 1 |
| **Documentation** | **10/11/2021** | **9/12/2021** | **30** |
| -Documentation ch1 | 10/11/2021 | 19/11/2021 | 10 |
| -Documentation ch2 | 20/11/2021 | 29/11/2021 | 10 |
| -Documentation ch3-ch4 | 30/11/2021 | 9/12/2021 | 10 |
| **UI Design** | **10/12/2021** | **29/12/2021** | **20** |
| -Interface | 10/12/2021 | 14/12/2021 | 5 |
| -Registration | 15/12/2021 | 19/12/2021 | 5 |
| -System UI | 20/12/2021 | 29/12/2021 | 10 |
| **First Sprint** | **30/12/2021** | **13/1/2022** | **15** |
| Interface & Layout | 30/12/2021 | 13/1/2022 | 15 |
| **Second Sprint Frontend** | **14/1/2022** | **12/2/2022** | **30** |
| **Posts Pages** | 14/1/2022 | 23/1/2022 | 10 |
| **Search pages** | 24/1/2022 | 2/2/2022 | 10 |
| **Test pages** | 3/2/2022 | 12/2/2022 | 10 |
| **Third Sprint Backend** | **13/2/2022** | **14/3/2022** | **30** |
| **Add data** | 13/2/2022 | 22/2/2022 | 10 |
| **Connect to fire base** | 23/2/2022 | 4/3/2022 | 10 |
| **Finish backend** | 5/3/2022 | 14/3/2022 | 10 |
| **Fourth Sprint** | **15/3/2022** | **8/4/2022** | **25** |
| Test Mobile App | 15/3/2022 | 29/3/2022 | 15 |
| Demo Video | 30/3/2022 | 3/4/2022 | 5 |
| Final Version Documentation ch5 | 4/4/2022 | 8/4/2022 | 5 |

## 1.8 Limitation of existing system

**- Difficulty:**

We've seen apps that did the work of identifying and grading skin cancer, but they weren't using same dataset we used in designing our model for application.

## 1.9 Need for the new system

1 in 5 Americans will develop skin cancer by the age of 70.
More than 2 people die of skin cancer in the U.S. every hour.
Having 5 or more sunburns doubles your risk for melanoma.
When detected early, the 5-year survival rate for melanoma is 99 percent.
-according to *(The Skin Cancer Foundation A 501(c)(3))*

## 1.10 Advantages of the new system

1-The mobile application makes it easier to take quick and apt decisions in emergency
Cases.
2- They can get information about important healthcare tips from time to time through This mobile application.
3-They can know which medicine has been prescribed for which disease and what its Side effects are.
4-Delivering Patient Care at Home.
5-Ease of communication and consulting a dermatologist by taking a picture from time To time to follow up on the treatment.
6-The mobile application can Help You Identify and Understand Your Medical Condition.
7-This mobile application offer convenience to the user, it also eases the burden on Their pockets reducing medical bills.

## 1.11 Risk and Risk Managements

1-Nothing beats personal advice and treatment from your dermatologist and inform
Him of all the information and medical examinations that you have had recently.
2-The data entered by the user to the application must be validated. If the patient
Enters wrong information, it may give him wrong results about his condition, and his
Condition will be incorrectly diagnosed.

# Chapter 2: Related Work

## 2.1 papers

| |
|---|
| **1- Skin Cancer Detection Using Convolutional Neural Networks:**<br><br>The dataset used for the training is a part of the 2019 ISIC Challenge, classifying dermoscopic images accounts for nine different diagnostic categories: melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, and squamous cell carcinoma, some of which are benign.<br>They have developed classifiers – a binary classifier and a multiclass classifier – on the Google Cloud Platform using Convolutional Neural Networks (CNNs).<br>The binary classifier achieved an accuracy of 79% with 220 epochs of training, and the multiclass classifier's accuracy is 72% with 200 epochs. |
| **2- Dermatologist Level Dermoscopy Skin Cancer Classification Using Different Deep Learning Convolutional Neural Networks Algorithms:**<br><br>The effectiveness and capability of convolutional neural networks have been studied in the classification of 8 skin diseases. Different pre-trained state-of-the-art architectures (Dense Net 201, ResNet 152, Inception v3, Inception ResNet v2). Were used and applied on 10135 dermoscopy skin images in total (HAM10000: 10015, PH2: 120). |

The utilized dataset includes 8 diagnostic categories - melanoma, melanocytic nevi, basal cell carcinoma, benign keratosis, actinic keratosis, and intraepithelial carcinoma, dermatofibroma, vascular lesions, and atypical nevi.
The best ROC AUC values for melanoma and basal cell carcinoma are 94.40% (ResNet 152) and 99.30% (Dense Net 201) versus 82.26% and 88.82% of dermatologists, respectively. Also, Dense Net 201 had the highest macro and micro averaged AUC values for overall classification (98.16%, 98.79%, respectively).

## 3- B-Seg Net: Branched-Seg Mentor Network for Skin Lesion Segmentation:

They propose a novel end-to-end (CNN) for a precise and robust skin lesion localization and segmentation. The proposed network has
3 sub-encoders branching out from the main encoder.
The 3 sub-encoders are inspired by Coordinate Convolution, Hourglass n, Hourglass, and Octave Convolutional blocks.
They trained their segmentation model just on the ISIC 2018 dataset.
To demonstrate the generalizability of our model, they evaluated their model on the ISIC 2018 and unseen datasets including ISIC 2017 and PH2.
Their approach showed an average 5% improvement in performance over different datasets.

## 4- Automatically Detection of Skin Cancer by Classification of Neural Network:

The diagnosing methodology uses Digital Image Processing Techniques and Artificial Neural Networks for the classification of Malignant Melanoma from other skin diseases.

Dermoscopic images were collected, and they are processed by various Image processing techniques.
The cancerous region is separated from healthy skin by the method of segmentation. The unique features of the segmented images were extracted using 2-D Wavelet Transform. Based on the features, the images were classified as Cancerous and Non-cancerous wo neural networks are used as classifier, Back-propagation neural network (BNN) and Auto-associative neural network (AANN). Recognition accuracy of the 3- layers back-propagation neural network classifier is 91% and auto-associative neural network is 82.6% in the image database that include dermoscopy photo and digital photo. The analysis of work based on MATLAB.

## **5- Vision-Based Classification of Skin Cancer using Deep Learning:**

This study proposes the use of deep learning algorithms to detect the presence of skin cancer, specifically melanoma, from images of skin lesions taken by a standard camera.
Aims to produce an inexpensive and fast computer-vision based machine learning tool that can be used by doctors and patients to track and classify suspicious skin lesions as benign or malignant with adequate accuracy using only a cell phone camera.
The data set was trained on 3 separate learning models with increasingly improved classification accuracy.
The 3 models included logistic regression, a deep neural network, and a fine-tuned, pre-trained, VGG-16 Convolutional Neural Network (CNN). Preliminary results show the developed algorithm’s ability to segment moles from images with 70% accuracy and classify skin lesions as melanoma with 78% balanced accuracy using a fine-tuned VGG-16 CNN.

## 6- Performance analysis of Convolutional Neural Network (CNN) based Cancerous Skin Lesion Detection System:

In 2019, G.S. Jayalakshmi and V. Sathiesh Kumar developed convolutional neural network model to diagnose and detect skin cancer from lesion images and identify the type of Skin lesion whether it is benign or malignant.
They used the ISIC dataset and improved the classification accuracy by using a Batch Normalized Convolutional Neural Network (BN-CNN). They obtained an accuracy of 89.30%.

## 7- A Deep CNN Model for Skin Cancer Detection and Classification:

In 2021, Masum Shah Junayed et al. Proposed a deep learning-based model to detect and classify skin cancer using the concept of deep Convolution Neural Network (CNN).
They used the Dhaka Medical College (DMC) dataset which consists of four types of skin cancer and used augmentation techniques to increase the dataset. They used some regularization methods like batch normalization to avoid overfitting and used convolution, max pooling, dropout, and fully connected layers. An accuracy of 95.98% achieved with the CNN classifier. In future, a light architecture can be designed without compromising the accuracy to detect skin cancer by minimizing computational complexity.

## 8- Skin Lesion Classification Using Ensembles of Multi-Resolution EfficientNets with Meta Data:

In 2019, Nils Gessert et al. addressed the challenging problem with a simple, data driven approach by including external data with skin lesion types that are not present in the training set. They used HAM10000, BCN 20000, MSK 7-point and in-house datasets. By using Image and Meta Data Pre-processing techniques, they deployed a CNN model that achieved an accuracy of 74.20%.

## 9- Automatic Lesion Detection System (ALDS) for Skin Cancer Classification Using SVM and Neural Classifiers:

In 2016, Muhammad Ali Farooq et al. developed an improvement of ALDS framework based on probabilistic approach that initially utilizes active contours and watershed merged mask for segmenting out the mole and later SVM and Neural Classifier are applied for the classification of the segmented mole.
They used DermIS, DermQuest and PH2 datasets. They deployed SVM and ANN model which detect and classify framework that can illustrate the non-pigmented skin malignancies along with the pigmented skin growth. They obtained an accuracy of 80.00%.
In future this system can be made to learn the evolution of the cancerous mole before time based on probabilistic and forecasting measures.

### 10- Skin Lesion Classification using Machine Learning Algorithms:

In 2017, Ilker Ali OZKAN and Murat KOKLU. Pre-classified the skin lesions in three groups as normal, abnormal and melanoma by machine learning methods and to develop a decision support

system that should make the decision easier for a doctor.
(objective) skin lesions based on dermoscopic images using 4 different machine learning methods namely: ANN, SVM, KNN and Decision Tree. They used PH2 dataset.
ANN has more successful classified the PH2 data set than SVM, KNN and DT. An accuracy of 92.50% achieved with the ANN classifier reveals that this classifier is a medical decision support system which could help dermatologists to diagnose the skin lesions.
This study maybe further progressed by using the different preliminary data processing techniques and hybrid classification algorithms. In addition, it can be combined with the related image processing techniques also to be able to make autonomous decisions in several medical issues.

## **11- A Smartphone based Application for Skin Cancer Classification Using Deep Learning with Clinical Images and Lesion Information:**

In 2021, a research team from Brazil wrote this paper. This paper aims to build mobile app to detect skin cancer type using clinical images and patients' demographics The team used PADUFES-20 dataset.
this dataset contains images for lesion and patient information like his (Age, lesion localization, if lesion hurts or not …... etc) and used Restnet50 model using 5- Fold cross validation and training phase for 100 epochs using Stochastic Gradient Descent (SGD) optimizer with a learning rate equal to 0.01 that decreases by a factor of 1/2 or 1/5 every 20 epochs and applied a standard data augmentation.
the team used t-SNE algorithm for feature reducing and when they combined between images features and patient information feature, they found that the best factor is 70% images feature to 30% patient information feature and finally the result was

accuracy 85% and recall 96% the result increased by 1.4% for accuracy and 1.1% for recall.

## 12- A Deep Learning Approach to Skin Cancer Detection in Dermoscopy Images:

In 2020, Ali Ameri built deep learning model to classify skin cancer if Melanoma or Benign using Alexnet model he used HAM10000 dataset he trained model with only lesion images not patient information and he get accuracy of 84% this model training time = 16 minute in Nvidia GTX 1050 GPU.

## 13- Computer Aided Melanoma Skin Cancer Detiction Using Artificial Neural Network Classifier:

In 2016, Singaporean Journal of Scientific Research published this paper, this paper Using image filters to detect skin cancer or not cancer this paper used background subtraction, edge detection, creation binary mask, skin lesion segmentation, feature extraction then classification accuracy about 96% and dataset not available.

## 14- Skin Lesion Classification Using Convolutional Neural Network with Novel Regularizer:

In 2019, Marwan Ali Albahar from Saudi Arabia searched for skin cancer and published techniques to build deep neural network to detect skin cancer from images and finally he wrote a helpful paper to help us to build models like that this paper used ISIC dataset with 2 CNN layers with regularizer and its result was accuracy 97%.

# Chapter 3: Background

## 3.1 Mobile-Net

As the name applied, the Mobile Net model is designed to be used in mobile applications, and it is Tensor Flow’s first mobile computer vision model.
Mobile-Net uses depth wise separable convolutions. It significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.
A depth wise separable convolution is made from two operations :-
1-Depthwise convolution.
2-Pointwise convolution.
Mobile-Net is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.

## The Architecture of MobileNet

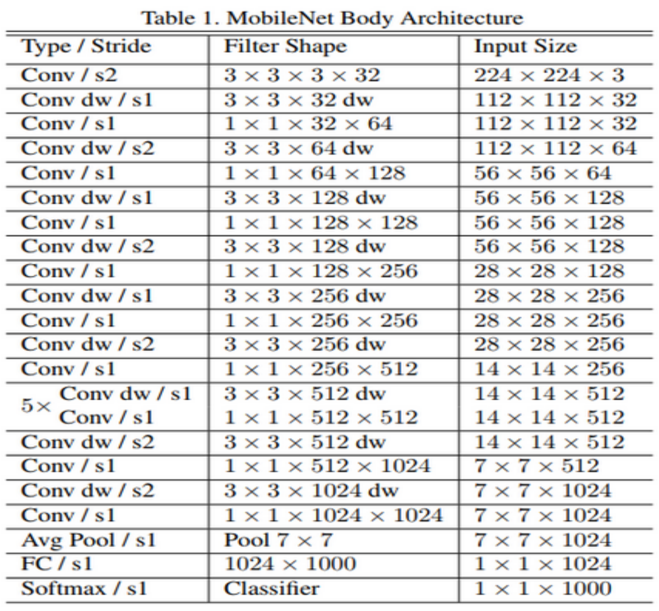


Table 1. MobileNet Body Architecture

| Type / Stride | Filter Shape | Input Size |
|---|---|---|
| Conv / s2 | $3 \times 3 \times 3 \times 32$ | $224 \times 224 \times 3$ |
| Conv dw / s1 | $3 \times 3 \times 32$ dw | $112 \times 112 \times 32$ |
| Conv / s1 | $1 \times 1 \times 32 \times 64$ | $112 \times 112 \times 32$ |
| Conv dw / s2 | $3 \times 3 \times 64$ dw | $112 \times 112 \times 64$ |
| Conv / s1 | $1 \times 1 \times 64 \times 128$ | $56 \times 56 \times 64$ |
| Conv dw / s1 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 128$ | $56 \times 56 \times 128$ |
| Conv dw / s2 | $3 \times 3 \times 128$ dw | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 256$ | $28 \times 28 \times 128$ |
| Conv dw / s1 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 256$ | $28 \times 28 \times 256$ |
| Conv dw / s2 | $3 \times 3 \times 256$ dw | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| $5\times$ Conv dw / s1 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| $5\times$ Conv / s1 | $1 \times 1 \times 512 \times 512$ | $14 \times 14 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 512$ dw | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 1024$ | $7 \times 7 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 1024$ dw | $7 \times 7 \times 1024$ |
| Conv / s1 | $1 \times 1 \times 1024 \times 1024$ | $7 \times 7 \times 1024$ |
| Avg Pool / s1 | Pool $7 \times 7$ | $7 \times 7 \times 1024$ |
| FC / s1 | $1024 \times 1000$ | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

Mobilenet Layers

The main difference between the Mobile-Net architecture and a traditional CNN instead of a single 3x3 convolution layer followed by the batch norm and ReLU. Mobile Nets split the convolution into a 3x3 depth-wise conv and a 1x1 pointwise conv, as shown in the figure.

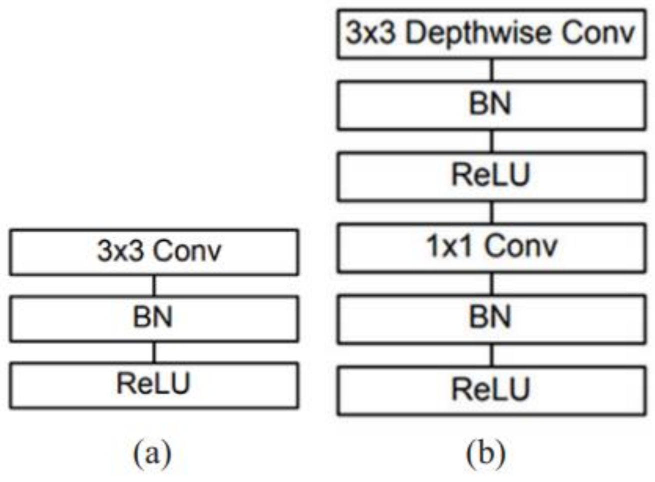
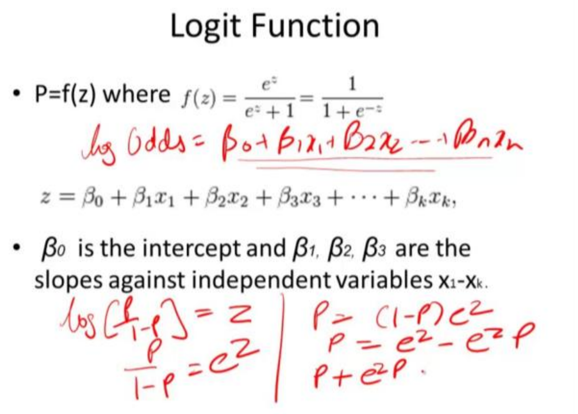


Fig. (a) Standard convolutional layer with batch normalization and ReLU. (b) Depth-wise separable convolution with depth-wise and pointwise layers followed by batch normalization and ReLU.

Mobile-Nets are a family of mobile-first computer vision models for
Tensorflow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application.
Mobile-Nets are small, low-latency, low-power models parametrized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embeddings, and segmentation.

## 3.2 Logistic Regression

This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:
- Logit(pi) = 1/(1+ exp(-pi))
- ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k



Logit Function

- P=f(z) where $f(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$

$\log \text{Odds} = \underline{\beta_0 + \beta_1 x_1 + \beta_2 x_2 \cdots + \beta_n x_n}$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k,$$

- $\beta_0$ is the intercept and $\beta_1, \beta_2, \beta_3$ are the slopes against independent variables $x_1$-$x_k$.

$\log \left[\frac{P}{1-P}\right] = z$ | $P = (1-P)e^z$

$\frac{P}{1-P} = e^z$ | $P = e^z - e^z P$

$P + e^z P$.

There are three types of logistic regression models, which are defined based on categorical response.
1-Binary logistic regression
2-Multinomial logistic regression
3-Ordinal logistic regression
We used $2^{nd}$ type Multinomial logistic regression: In this type of logistic regression model, the dependent variable has three or more possible outcomes; however, these values have no specified order.

## 3.3 Random Forest

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.
A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.
The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the

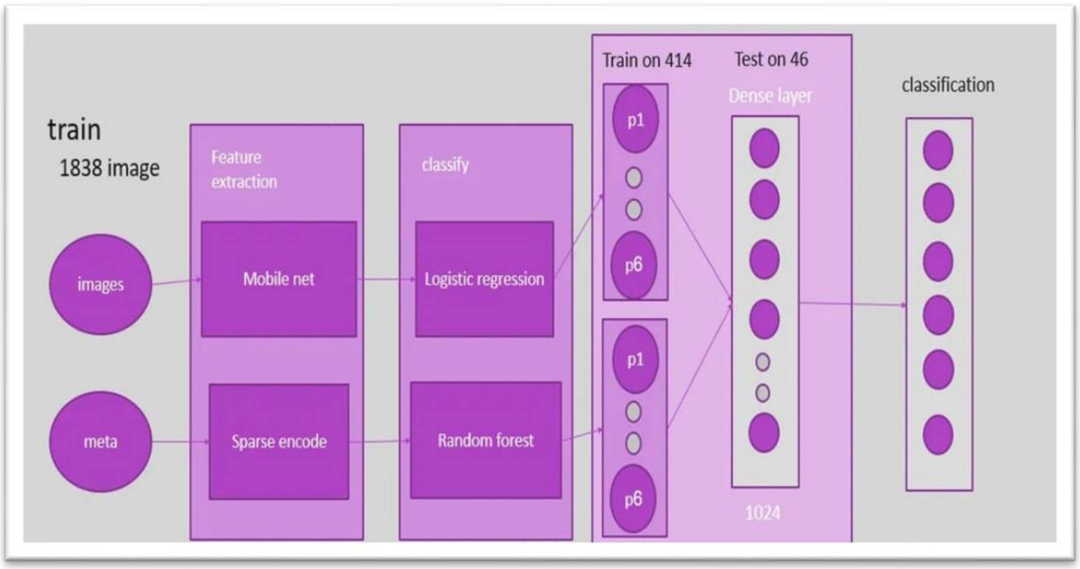
average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.
A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like scikit-learn).

## Classification in random forests:

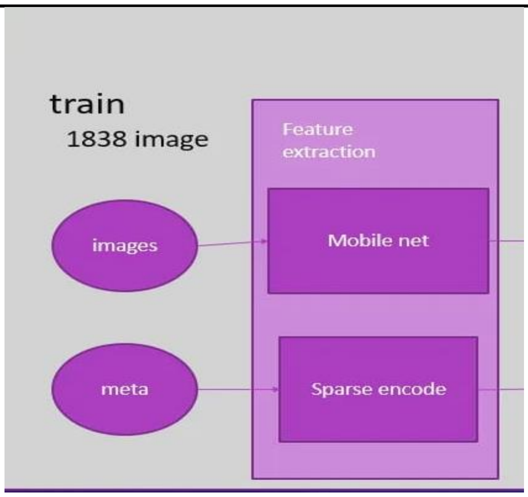
Classification in random forests employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes.
A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system. The diagram below shows a simple random forest classifier.

# Chapter 4: Architecture

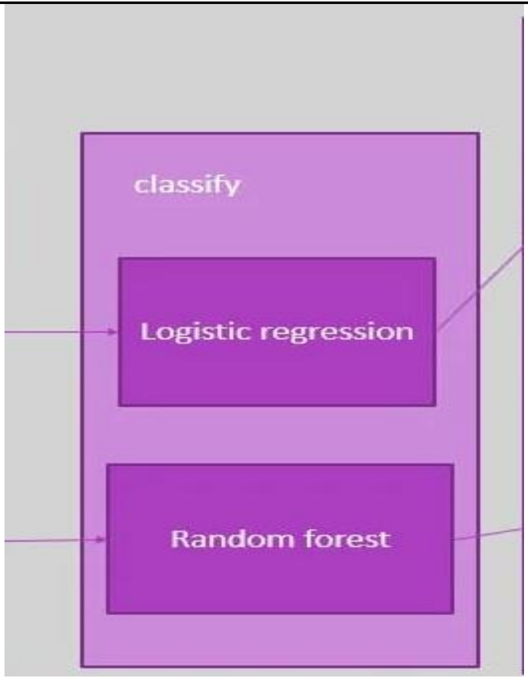
## 4.1 Model Architecture



### 4.1.1 Feature Extraction & Preprocessing:

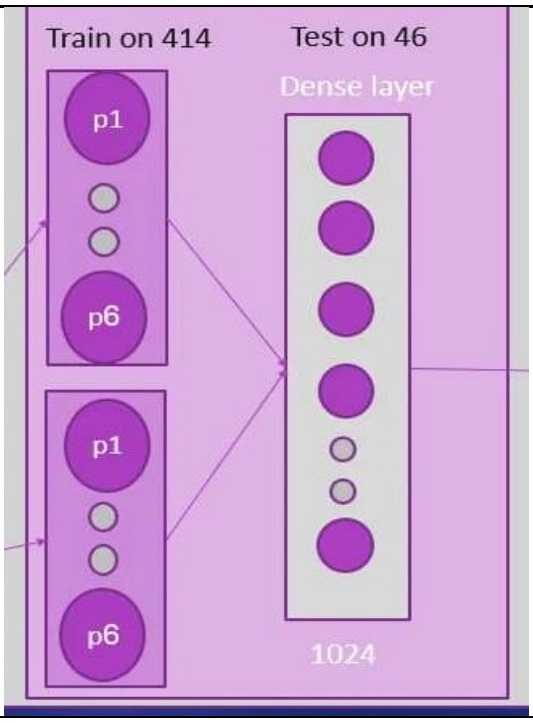
| | |
|---|---|
|   train<br>1838 image<br>images<br>meta<br>Feature extraction<br>Mobile net<br>Sparse encode | In the first part of our model, we have a feature extraction block that consists of a mobile net to extract the features (lesions) from the images and a sparse encode for metadata preprocessing. For example, if there is any string in our metadata, convert it to integers. |

## 4.1.2 Image & Metadata Classification:



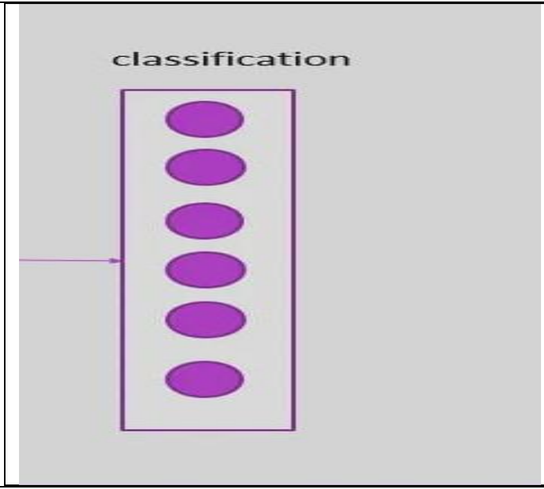
The second block, which is called classify, consists of a logistic regression classifier, which takes as an input the output of the mobile net feature extractor to classify the lesions and outputs 6 probabilities for each lesion class, and a random forest, which takes as an input the output of spare encode of the metadata, also classifies it and outputs another 6 probabilities for the same 6 lesion classes.

## 4.1.3 Training & Test the Model



The third block: First, we split the remaining 20% of the data (460 records) into 80% train, 10% test and 10% validation. Test = 46 records, Val=46 records, train = 368 records, then we pass train and Val to the logistic model and random forest model to get 6 predictions per model. Finally, we concatenate all those predictions and pass it to one dense layer with 1024 neurons.

## 4.1.4 Final Prediction

| | |
|---|---|
|  | The final block, which takes as input the dense layer's output and outputs the class that assumes the lesion belongs to it. |

# Chapter 5: Experimental Setup, Implementation, Results & Discussion

## 5.1 Model Explanation

### 5.1.1 Intro

Skin Lesion Detection is a deep hybrid model to detect skin lesions from mobile images.
We tried different experimental scenarios. The models were implemented using Python 3.7.13, and experiments were carried out on a Google Collab notebook using the GPU run-time type. Experiments were conducted on the original dataset.

### 5.1.2 Experimental scenarios

**Scenario 1:**
Applying 10 pre-trained models to the PAD-UFES-20 dataset using one dense layer contains "1024 neurons", activation function "Relu", and another dense layer with "6" neurons, activation function "SoftMax" Then, the model is trained for "20" epochs using the "Adam" optimizer and a batch size of "32". EfficientNetB4 achieves the highest accuracy with 79.1%.

| Model | Images Only | Images + Metadata | Metadata only |
|---|---|---|---|
| EfficientNetB4 | 67.3 | 79.1 | 73.4 |
| MobileNetV2 | 44.3 | 75.6 | 73.4 |
| ResNet50 | 62.6 | 73.9 | 73.4 |
| NASNetLarge | 50.8 | 73 | 73.4 |
| Xception | 47.8 | 72.6 | 73.4 |
| DenseNet121 | 51.3 | 71.7 | 73.4 |
| EfficientNetV2L | 60 | 71 | 73.4 |
| Vgg19 | 58.2 | 70.8 | 73.4 |
| Vgg16 | 58.2 | 70.4 | 73.4 |
| InceptionResNetV | 38.6 | 63.9 | 73.4 |

**<u>Scenario 2:</u>**

Adding another two dense layers with "512 neurons", activation function "Relu",
dense layer with "256 neurons", activation function "Relu", dropout layer with "0.5", and finally a dense layer with "6 neurons", activation function "SoftMax"
Then, the model is trained for "20" epochs using the "Adam" optimizer and batch size"32".
EfficientNetB4 achieves the highest accuracy with 82.1%.

| Model | Images Only | Images + Metadata | Metadata only |
|---|---|---|---|
| EfficientNetB4 | 67.8 | 82.1 | 73.4 |
| MobileNetV2 | 50 | 77.3 | 73.4 |

**<u>Scenario 3:</u>**

Deep Hybrid Model (MobileNetv2+ML Classifier):
We extracted image features from a pretrained deep learning model and applied a machine learning classifier.
We tried 10 models. The best one was MobileNet. Then we applied 7 machine learning classifiers. The best classifier for images was logistic regression, and the best classifier for meta was random forest. We calculate accuracy by getting the average predication between logistic regression and random forest.
Accuracy:
• Images only: 63.9%
• Meta only: 86.5%
• Images + Meta: 85.6%

**<u>Scenario 4:</u>**

Deep Hybrid Model (MobileNetv2+ML Classifier +Weighted Avg):
We noticed that the meta-accuracy was better than image accuracy, so we decided to make the final model depend on meta more than image, so we tried a weighted average between logistic and random instead of a normal average. The best weighted average was 0.7 of meta and 0.3 of image.

Accuracy:
• Images only: 63.9%
• Meta only: 86.5%
• Images + Meta: 86.9%

**Scenario 5:**

Concatenate:
We tried to pass all features (images + meta) to one of the machine learning classifiers.
So, we add a concatenate layer to the output of MobileNet and add meta to it.

**Roc-Auc-Score:**

| Model | Accuracy |
|---|---|
| Logistic Regression | 74.7 |
| Random Forest | 67.9 |
| Gaussian NB | 66.2 |
| Ada Boost | 66.8 |
| KNN | 66.9 |
| LDA | 66.6 |
| SVM | 50 |

**Scenario 6:**

Hyper + 2 Dense (The approved scenario)
In the final version, we removed the weighted average as specified in Scenario 4 and passed all features of (Images + Meta) to a dense layer with "1024 neurons", activation function "Relu", KernelRegularizerL1L2 "L1 = 1e-5, L2 = 1e-4", dropout layer with "0.9", and another dense layer with "6 neurons", activation function "SoftMax" Then, the model is trained for "200" epochs using the "Adam" optimizer with learning rate of "0.0001", epsilon "0.01", and batch size of "16".
Accuracy:
• Images only: 63.9%
• Meta only: 85.6% • Images + Meta: 95.6%

### 5.1.3 Final model architecture

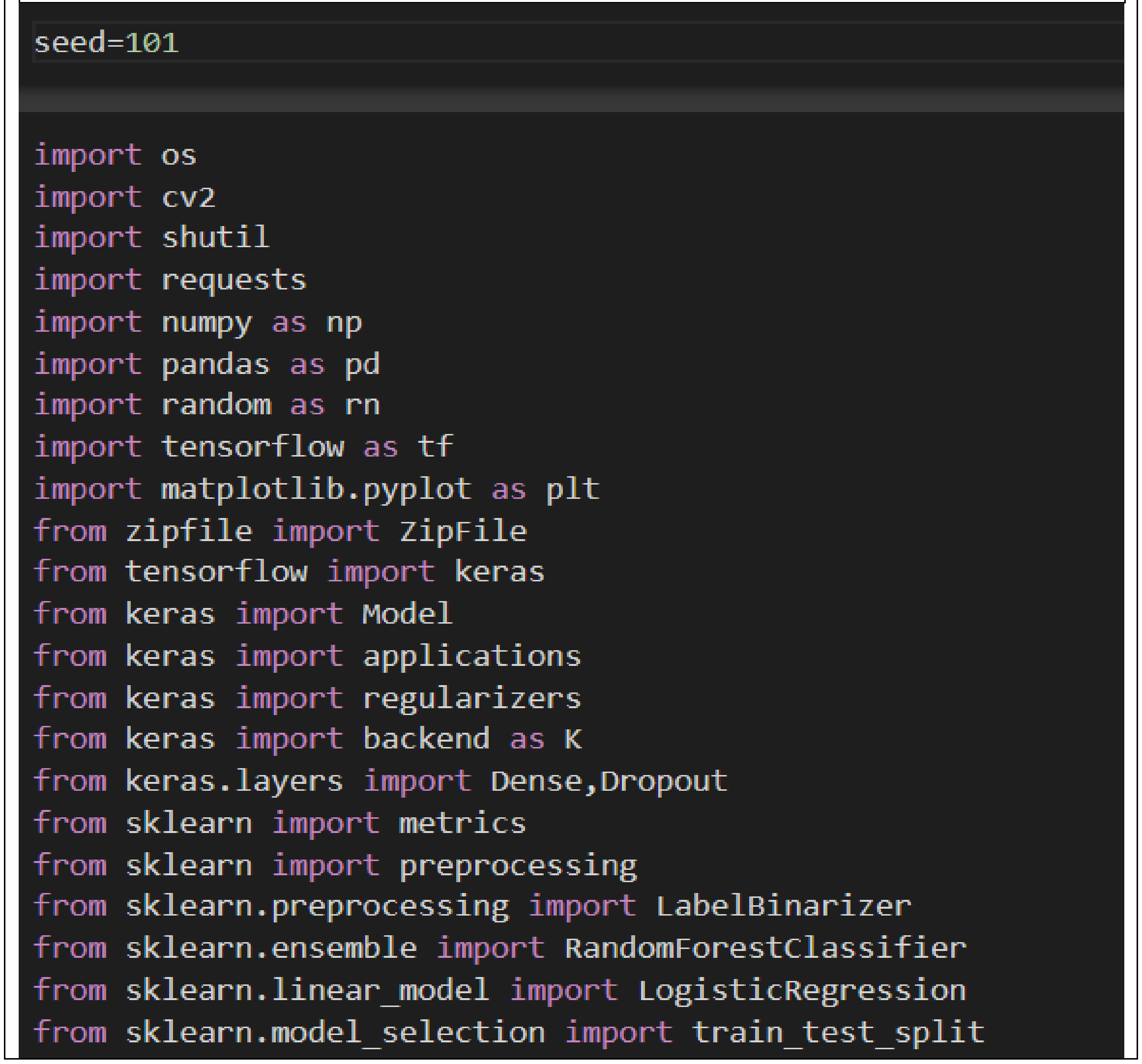
```
1. Set Seed = 101.
2. Import libraries.
3.Download the dataset and extract it.
4. Move all the images in the dataset into one folder.
5. Read meta by using Pandas's library.
6. Categorize meta and deal with Null values.
7. Encode string in meta.
8. Build a MobileNetV2 pre-trained model with the input size
(380,380,3).
9. Loop in meta.
10. Get image names.
11.Read image with size (380,380).
12.Convert the colour from BGR to RGB.
13.Normalize images by dividing them into 255.
14.Get features from images using model.predict.
15.Use train_test_split to split the data into 80% train and 20%
test.
16.Split test data into test and validation [test = 50% & validation
= 50% and validation = test = 10% of all data].
17.Train images With Logistic Regression and training meta with
the Random Forest classifier from Sklearn.
18.Get probabilities from test and validation for both images and
meta.
19.Create a DataFrame which contains probabilities of images,
meta, and the true label (Target Value) for them.
20.Split the data frame into 80% train, 10% validation, and 10%
test.
21.Initialize model_input as an empty list.
22. Create a Keras instance.Input for all classes in the dataset.
23.Appending all instances in model_input
```

24.Merging images and meta and passing them to specific layers [Dense layer (1024) with Relu–Dropout layer (0.9)–Dense layer (6) with SoftMax].
Finally, the model was trained for "200" epochs using the "Adam" optimizer with learning rate of "0.0001", epsilon "0.01", and batch size of "16".

## 5.2 Model implementation

### 5.2.1 Python libraries

These are all the libraries that we used.



```
seed=101
```

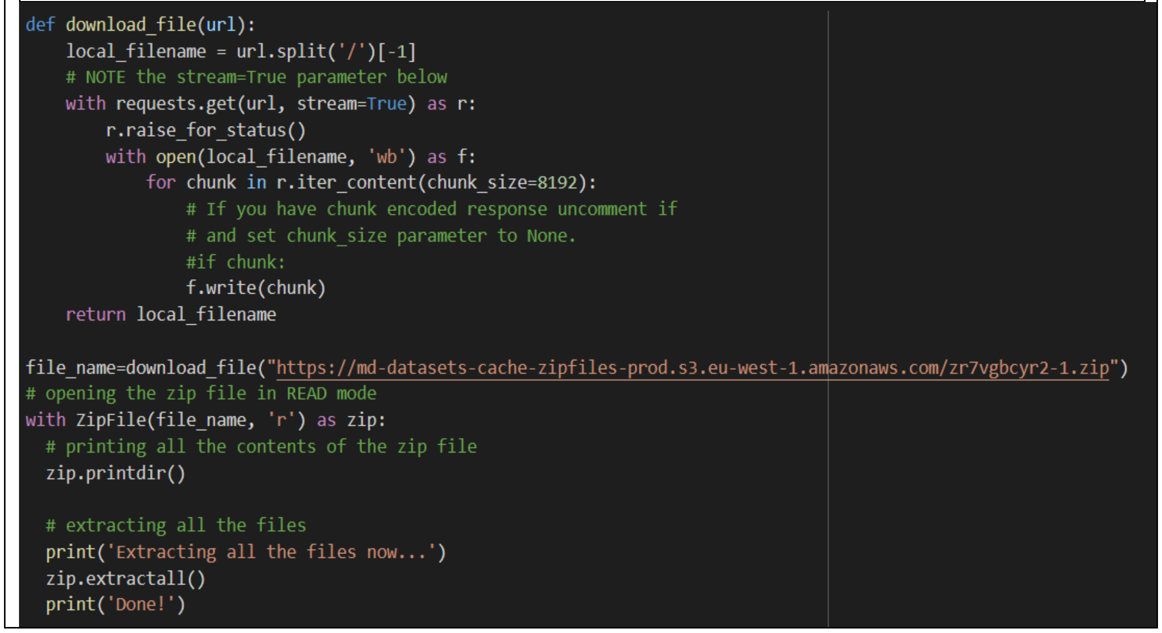
```
import os
import cv2
import shutil
import requests
import numpy as np
import pandas as pd
import random as rn
import tensorflow as tf
import matplotlib.pyplot as plt
from zipfile import ZipFile
from tensorflow import keras
from keras import Model
from keras import applications
from keras import regularizers
from keras import backend as K
from keras.layers import Dense,Dropout
from sklearn import metrics
from sklearn import preprocessing
from sklearn.preprocessing import LabelBinarizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

## 5.2.2 Dataset

We used data sets from different resources and got what we needed from them. We will attach the various data sets, as well as the resource links, in the final report.

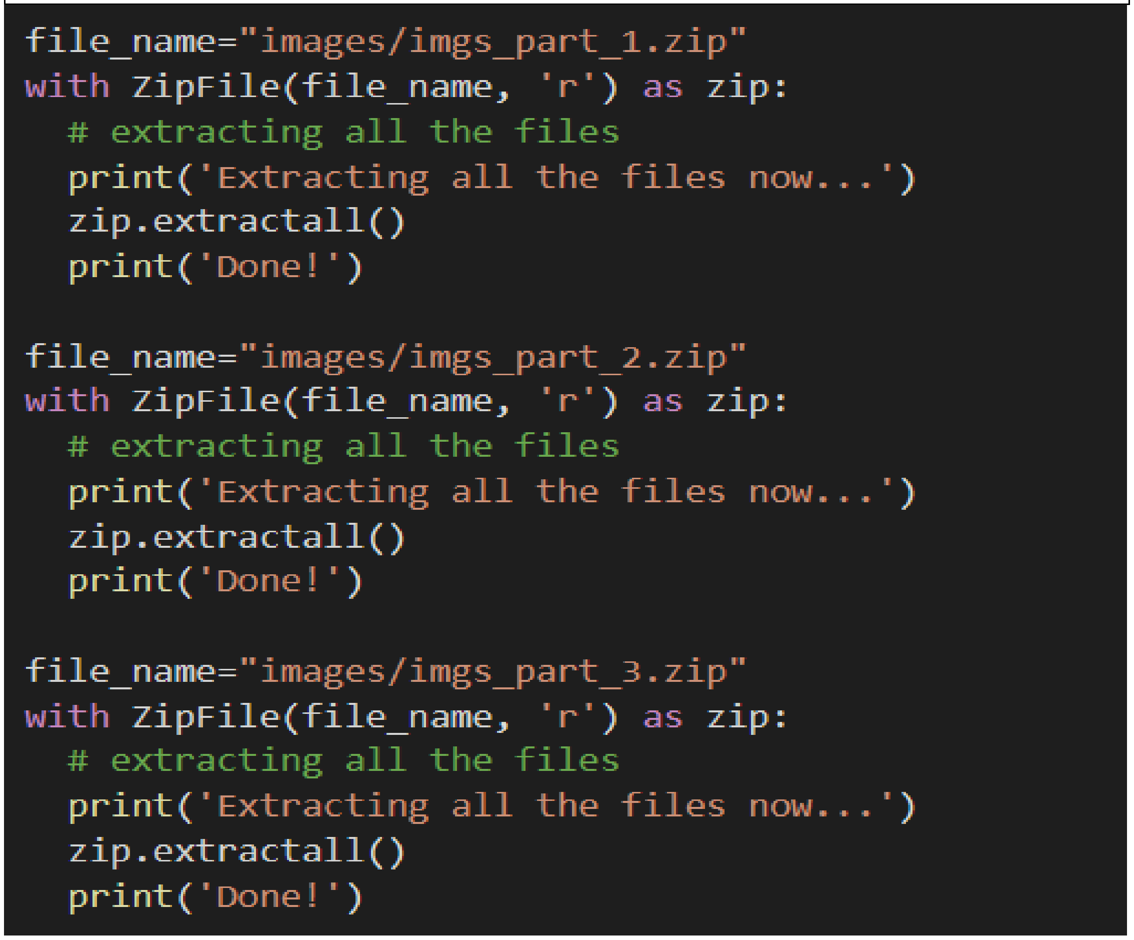
PAD-UFES-20 (the main dataset): A skin lesion dataset composed of patient data and clinical images collected from smartphones.
A patient may have one or more skin lesions and a skin lesion may
have one or more images. In total, there are 1373 patients, 1641 skin lesions, and 2298 images present in the dataset.
The images in this dataset are collected using different smartphone devices they present different resolutions, sizes, and lighting conditions.
The metadata associated with each skin lesion is composed of 26 attributes: 21 patient clinical features, four identifying features (patient ID, lesion ID, Image ID, and if the sample is biopsy-proven), and a diagnostic label.

The image below shows the download of the dataset.



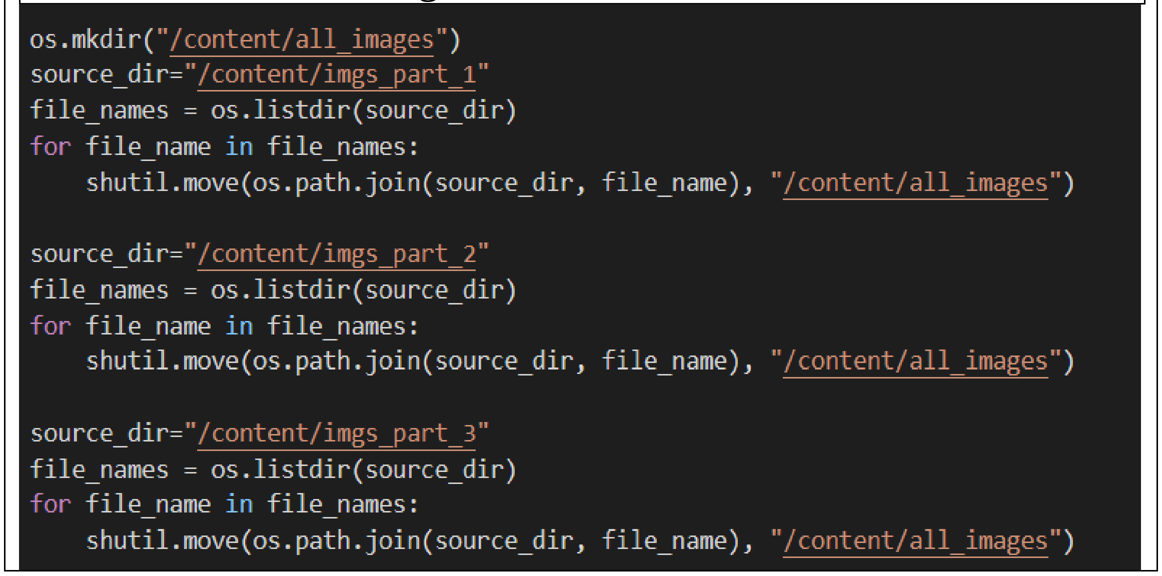
```
def download_file(url):
    local_filename = url.split('/')[-1]
    # NOTE the stream=True parameter below
    with requests.get(url, stream=True) as r:
        r.raise_for_status()
        with open(local_filename, 'wb') as f:
            for chunk in r.iter_content(chunk_size=8192):
                # If you have chunk encoded response uncomment if
                # and set chunk_size parameter to None.
                #if chunk:
                f.write(chunk)
    return local_filename

file_name=download_file("https://md-datasets-cache-zipfiles-prod.s3.eu-west-1.amazonaws.com/zr7vgbcyr2-1.zip")
# opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
  # printing all the contents of the zip file
  zip.printdir()

  # extracting all the files
  print('Extracting all the files now...')
  zip.extractall()
  print('Done!')
```

After that, extract all files from the dataset.



```python
file_name="images/imgs_part_1.zip"
with ZipFile(file_name, 'r') as zip:
  # extracting all the files
  print('Extracting all the files now...')
  zip.extractall()
  print('Done!')

file_name="images/imgs_part_2.zip"
with ZipFile(file_name, 'r') as zip:
  # extracting all the files
  print('Extracting all the files now...')
  zip.extractall()
  print('Done!')

file_name="images/imgs_part_3.zip"
with ZipFile(file_name, 'r') as zip:
  # extracting all the files
  print('Extracting all the files now...')
  zip.extractall()
  print('Done!')
```

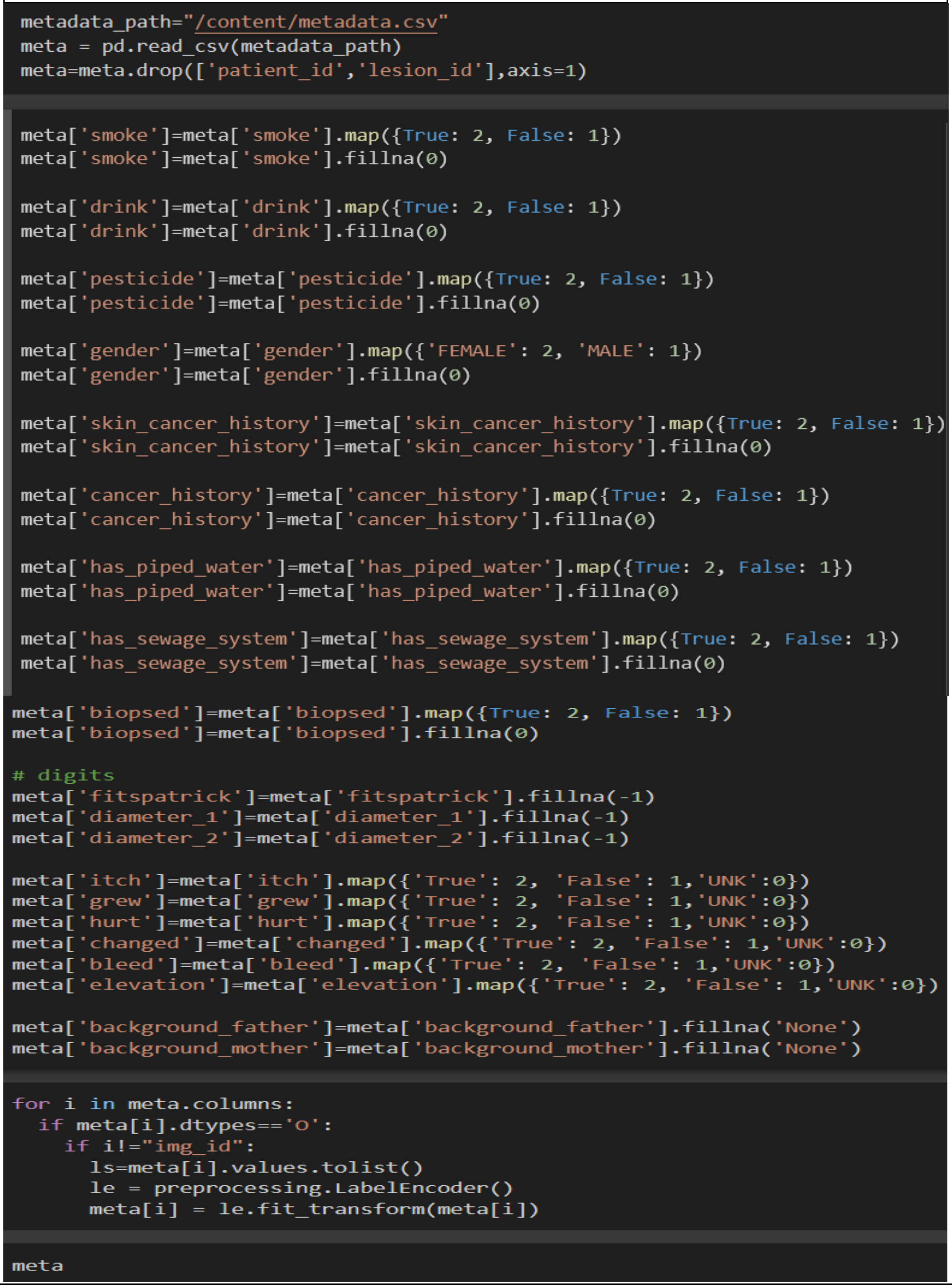
Then move all the images to one folder.



```python
os.mkdir("/content/all_images")
source_dir="/content/imgs_part_1"
file_names = os.listdir(source_dir)
for file_name in file_names:
    shutil.move(os.path.join(source_dir, file_name), "/content/all_images")

source_dir="/content/imgs_part_2"
file_names = os.listdir(source_dir)
for file_name in file_names:
    shutil.move(os.path.join(source_dir, file_name), "/content/all_images")

source_dir="/content/imgs_part_3"
file_names = os.listdir(source_dir)
for file_name in file_names:
    shutil.move(os.path.join(source_dir, file_name), "/content/all_images")
```

## 5.2.3 Metadata & its pre-processing

Here we read the metadata file, encode string in meta and deal with null values.



```
metadata_path="/content/metadata.csv"
meta = pd.read_csv(metadata_path)
meta=meta.drop(['patient_id','lesion_id'],axis=1)
```

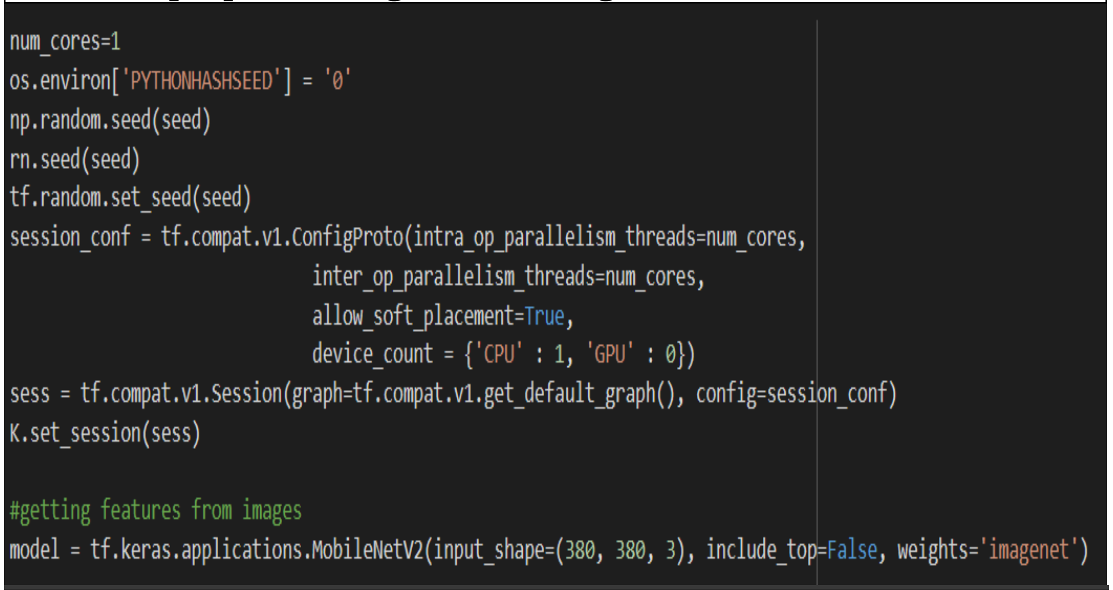
```
meta['smoke']=meta['smoke'].map({True: 2, False: 1})
meta['smoke']=meta['smoke'].fillna(0)

meta['drink']=meta['drink'].map({True: 2, False: 1})
meta['drink']=meta['drink'].fillna(0)

meta['pesticide']=meta['pesticide'].map({True: 2, False: 1})
meta['pesticide']=meta['pesticide'].fillna(0)

meta['gender']=meta['gender'].map({'FEMALE': 2, 'MALE': 1})
meta['gender']=meta['gender'].fillna(0)

meta['skin_cancer_history']=meta['skin_cancer_history'].map({True: 2, False: 1})
meta['skin_cancer_history']=meta['skin_cancer_history'].fillna(0)

meta['cancer_history']=meta['cancer_history'].map({True: 2, False: 1})
meta['cancer_history']=meta['cancer_history'].fillna(0)

meta['has_piped_water']=meta['has_piped_water'].map({True: 2, False: 1})
meta['has_piped_water']=meta['has_piped_water'].fillna(0)

meta['has_sewage_system']=meta['has_sewage_system'].map({True: 2, False: 1})
meta['has_sewage_system']=meta['has_sewage_system'].fillna(0)
meta['biopsed']=meta['biopsed'].map({True: 2, False: 1})
meta['biopsed']=meta['biopsed'].fillna(0)

# digits
meta['fitspatrick']=meta['fitspatrick'].fillna(-1)
meta['diameter_1']=meta['diameter_1'].fillna(-1)
meta['diameter_2']=meta['diameter_2'].fillna(-1)

meta['itch']=meta['itch'].map({'True': 2, 'False': 1,'UNK':0})
meta['grew']=meta['grew'].map({'True': 2, 'False': 1,'UNK':0})
meta['hurt']=meta['hurt'].map({'True': 2, 'False': 1,'UNK':0})
meta['changed']=meta['changed'].map({'True': 2, 'False': 1,'UNK':0})
meta['bleed']=meta['bleed'].map({'True': 2, 'False': 1,'UNK':0})
meta['elevation']=meta['elevation'].map({'True': 2, 'False': 1,'UNK':0})

meta['background_father']=meta['background_father'].fillna('None')
meta['background_mother']=meta['background_mother'].fillna('None')
```

```
for i in meta.columns:
  if meta[i].dtypes=='O':
    if i!="img_id":
      ls=meta[i].values.tolist()
      le = preprocessing.LabelEncoder()
      meta[i] = le.fit_transform(meta[i])
```

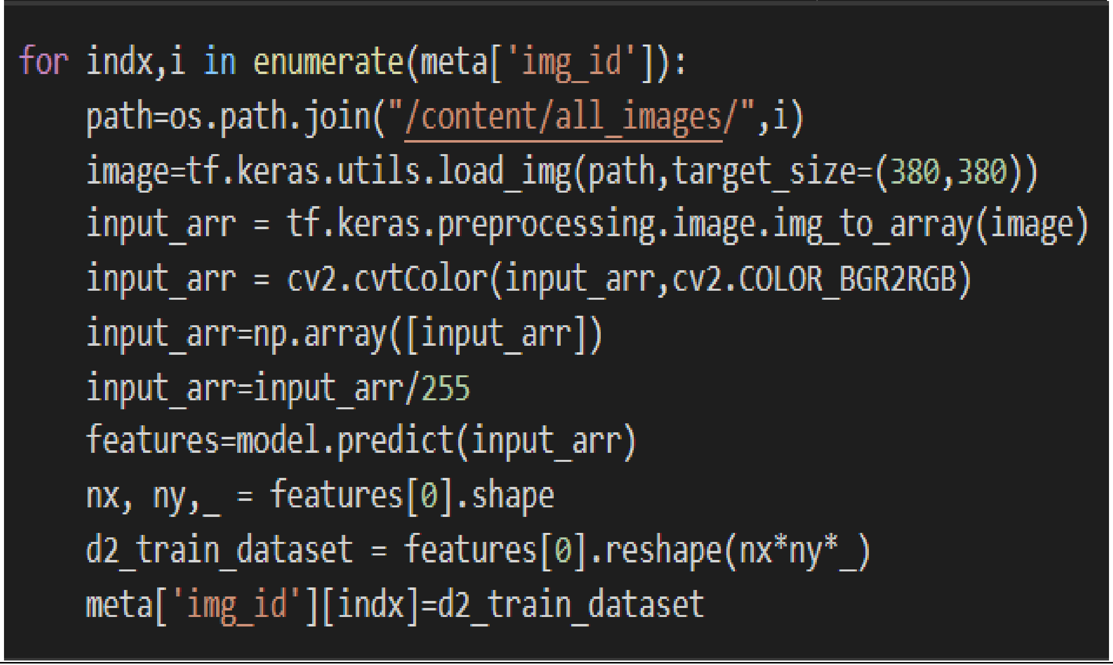
```
meta
```

## 5.2.4 Pre-processing

In this part, we built a MobileNetV2 pre-trained model and then did preprocessing on the images.



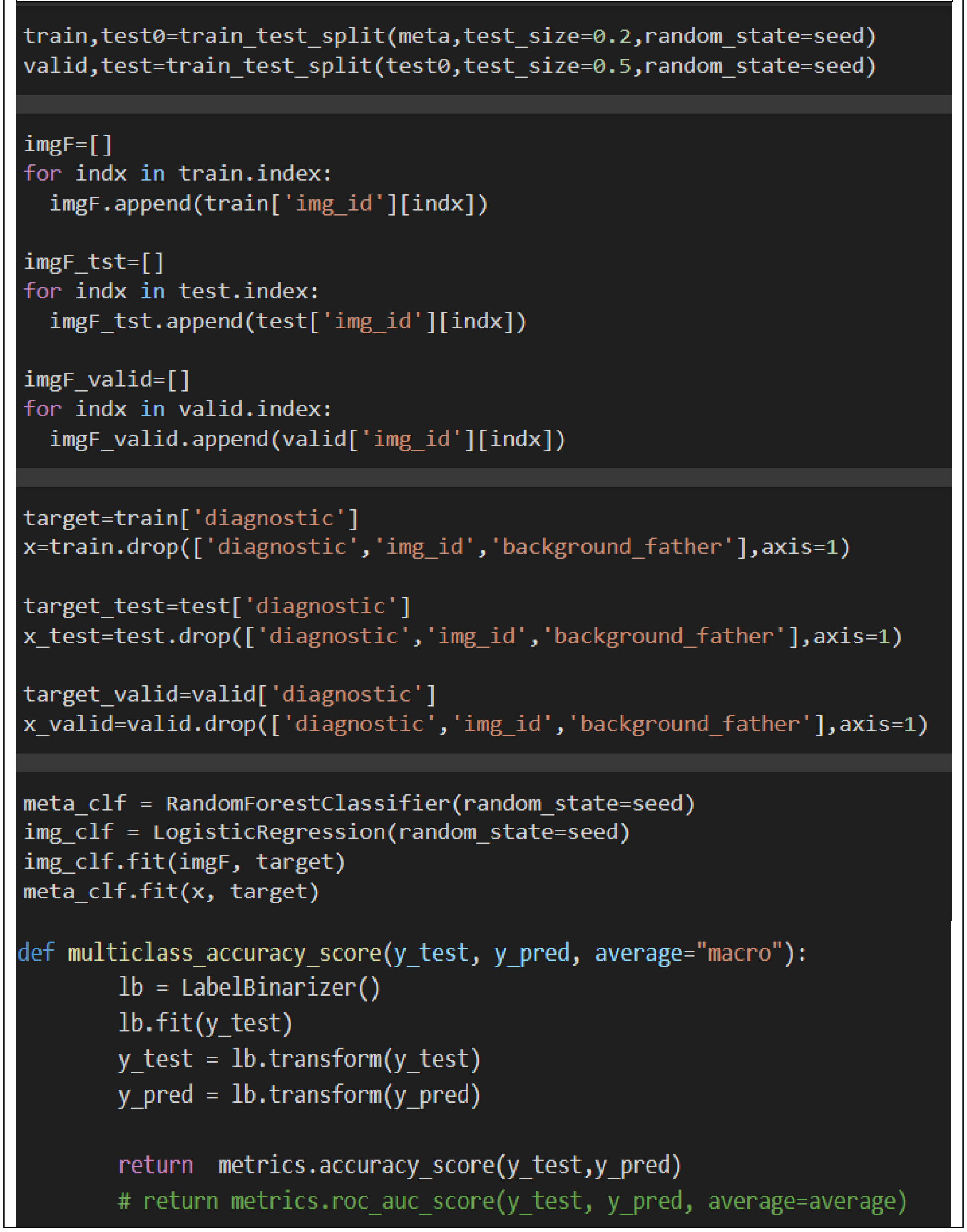
```
num_cores=1
os.environ['PYTHONHASHSEED'] = '0'
np.random.seed(seed)
rn.seed(seed)
tf.random.set_seed(seed)
session_conf = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=num_cores,
                                        inter_op_parallelism_threads=num_cores,
                                        allow_soft_placement=True,
                                        device_count = {'CPU' : 1, 'GPU' : 0})
sess = tf.compat.v1.Session(graph=tf.compat.v1.get_default_graph(), config=session_conf)
K.set_session(sess)

#getting features from images
model = tf.keras.applications.MobileNetV2(input_shape=(380, 380, 3), include_top=False, weights='imagenet')
```



```
for indx,i in enumerate(meta['img_id']):
    path=os.path.join("/content/all_images/",i)
    image=tf.keras.utils.load_img(path,target_size=(380,380))
    input_arr = tf.keras.preprocessing.image.img_to_array(image)
    input_arr = cv2.cvtColor(input_arr,cv2.COLOR_BGR2RGB)
    input_arr=np.array([input_arr])
    input_arr=input_arr/255
    features=model.predict(input_arr)
    nx, ny,_ = features[0].shape
    d2_train_dataset = features[0].reshape(nx*ny*_)
    meta['img_id'][indx]=d2_train_dataset
```

## 5.2.5 Split the dataset

Here we split the data into 80% train and 20% test.
Then split test data into test and validation [test = 50% & validation = 50% and validation = test = 10% of all data].



```
train,test0=train_test_split(meta,test_size=0.2,random_state=seed)
valid,test=train_test_split(test0,test_size=0.5,random_state=seed)
```

```
imgF=[]
for indx in train.index:
  imgF.append(train['img_id'][indx])

imgF_tst=[]
for indx in test.index:
  imgF_tst.append(test['img_id'][indx])

imgF_valid=[]
for indx in valid.index:
  imgF_valid.append(valid['img_id'][indx])
```

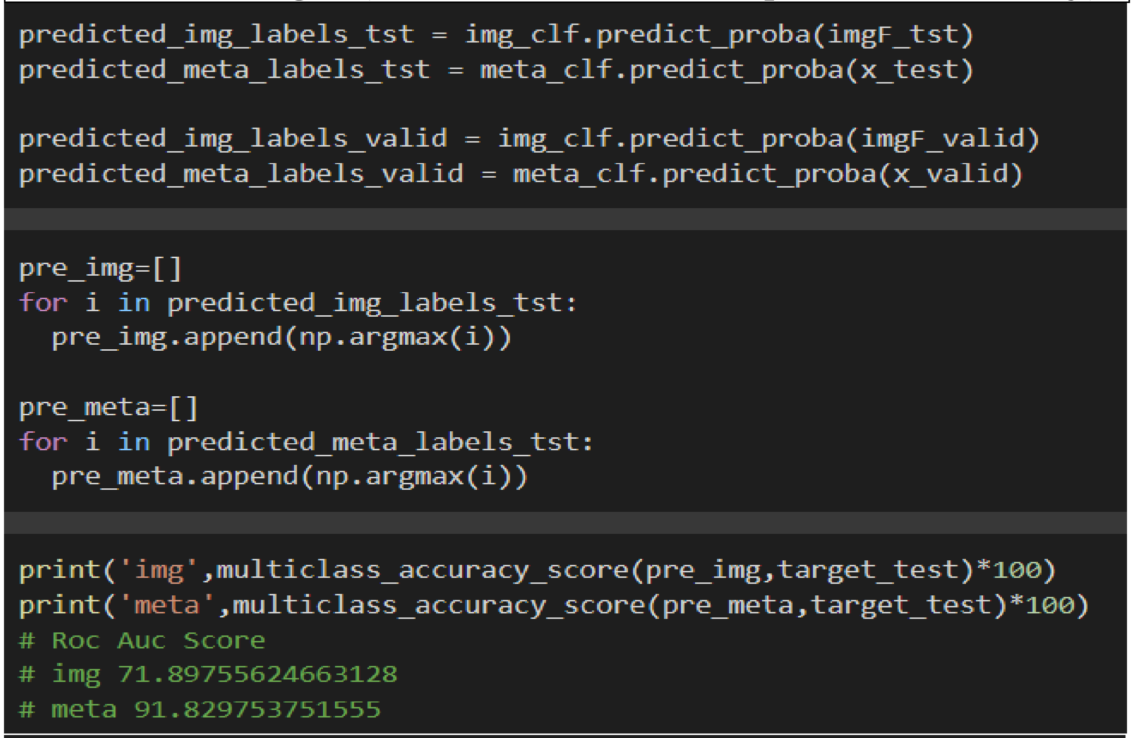
```
target=train['diagnostic']
x=train.drop(['diagnostic','img_id','background_father'],axis=1)

target_test=test['diagnostic']
x_test=test.drop(['diagnostic','img_id','background_father'],axis=1)

target_valid=valid['diagnostic']
x_valid=valid.drop(['diagnostic','img_id','background_father'],axis=1)
```

```
meta_clf = RandomForestClassifier(random_state=seed)
img_clf = LogisticRegression(random_state=seed)
img_clf.fit(imgF, target)
meta_clf.fit(x, target)
```

```
def multiclass_accuracy_score(y_test, y_pred, average="macro"):
        lb = LabelBinarizer()
        lb.fit(y_test)
        y_test = lb.transform(y_test)
        y_pred = lb.transform(y_pred)

        return  metrics.accuracy_score(y_test,y_pred)
        # return metrics.roc_auc_score(y_test, y_pred, average=average)
```

## 5.2.6 The model

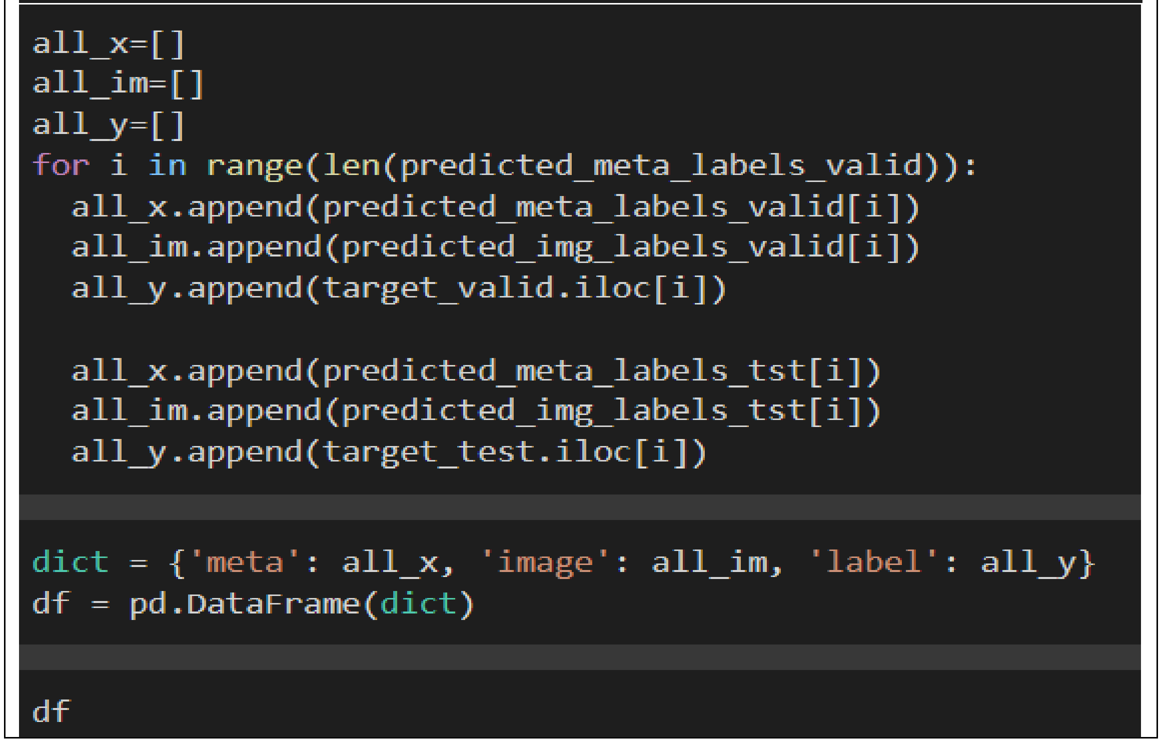
Metadata + Images (Train The Model & Improve Its Accuracy)



```
predicted_img_labels_tst = img_clf.predict_proba(imgF_tst)
predicted_meta_labels_tst = meta_clf.predict_proba(x_test)

predicted_img_labels_valid = img_clf.predict_proba(imgF_valid)
predicted_meta_labels_valid = meta_clf.predict_proba(x_valid)
```

```
pre_img=[]
for i in predicted_img_labels_tst:
  pre_img.append(np.argmax(i))

pre_meta=[]
for i in predicted_meta_labels_tst:
  pre_meta.append(np.argmax(i))
```

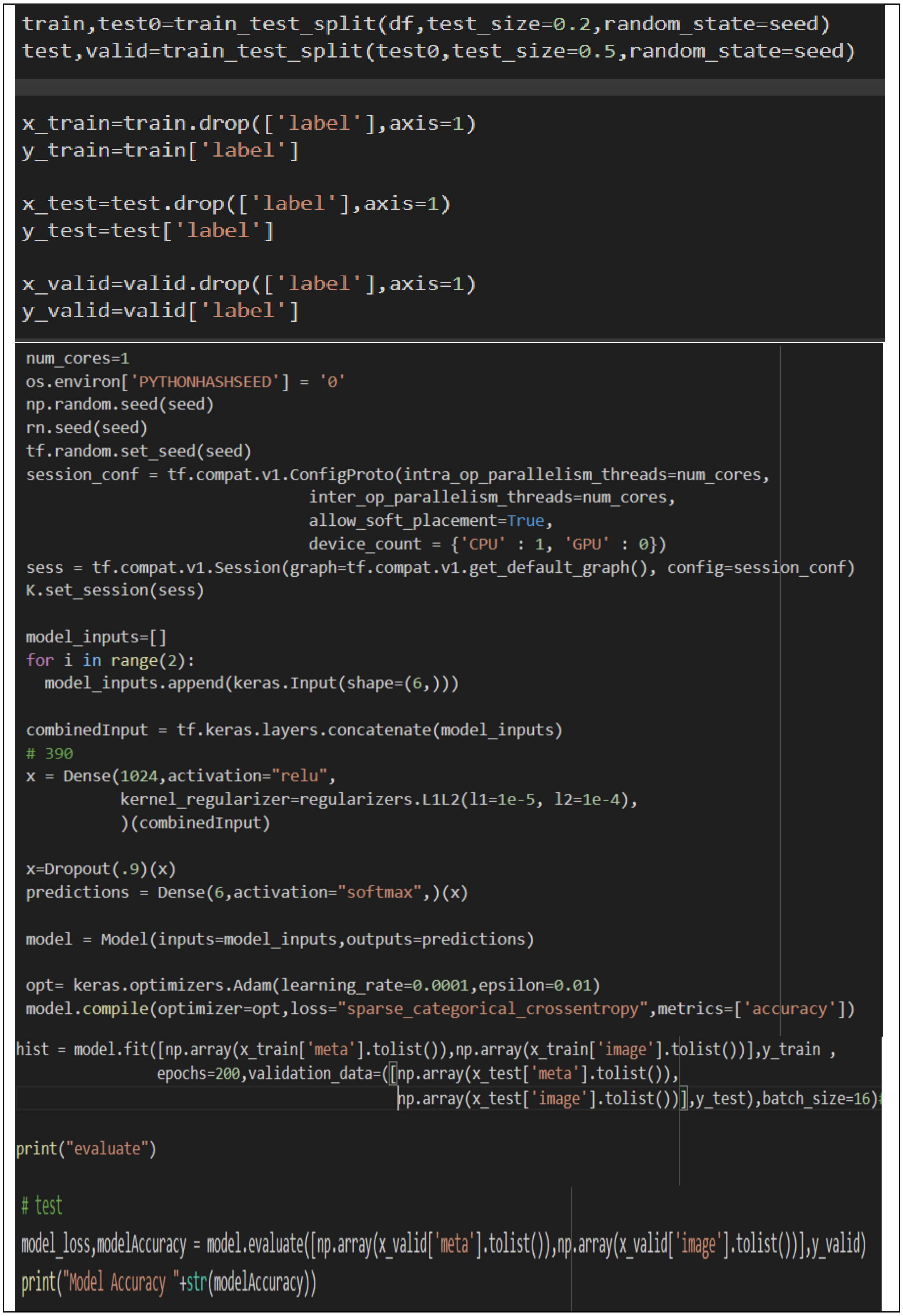
```
print('img',multiclass_accuracy_score(pre_img,target_test)*100)
print('meta',multiclass_accuracy_score(pre_meta,target_test)*100)
# Roc Auc Score
# img 71.89755624663128
# meta 91.829753751555
```



```
all_x=[]
all_im=[]
all_y=[]
for i in range(len(predicted_meta_labels_valid)):
  all_x.append(predicted_meta_labels_valid[i])
  all_im.append(predicted_img_labels_valid[i])
  all_y.append(target_valid.iloc[i])

  all_x.append(predicted_meta_labels_tst[i])
  all_im.append(predicted_img_labels_tst[i])
  all_y.append(target_test.iloc[i])
```

```
dict = {'meta': all_x, 'image': all_im, 'label': all_y}
df = pd.DataFrame(dict)
```

```
df
```



```
train,test0=train_test_split(df,test_size=0.2,random_state=seed)
test,valid=train_test_split(test0,test_size=0.5,random_state=seed)

x_train=train.drop(['label'],axis=1)
y_train=train['label']

x_test=test.drop(['label'],axis=1)
y_test=test['label']

x_valid=valid.drop(['label'],axis=1)
y_valid=valid['label']
```

```
num_cores=1
os.environ['PYTHONHASHSEED'] = '0'
np.random.seed(seed)
rn.seed(seed)
tf.random.set_seed(seed)
session_conf = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=num_cores,
                              inter_op_parallelism_threads=num_cores,
                              allow_soft_placement=True,
                              device_count = {'CPU' : 1, 'GPU' : 0})
sess = tf.compat.v1.Session(graph=tf.compat.v1.get_default_graph(), config=session_conf)
K.set_session(sess)

model_inputs=[]
for i in range(2):
  model_inputs.append(keras.Input(shape=(6,)))

combinedInput = tf.keras.layers.concatenate(model_inputs)
# 390
x = Dense(1024,activation="relu",
          kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
          )(combinedInput)

x=Dropout(.9)(x)
predictions = Dense(6,activation="softmax",)(x)

model = Model(inputs=model_inputs,outputs=predictions)

opt= keras.optimizers.Adam(learning_rate=0.0001,epsilon=0.01)
model.compile(optimizer=opt,loss="sparse_categorical_crossentropy",metrics=['accuracy'])
```

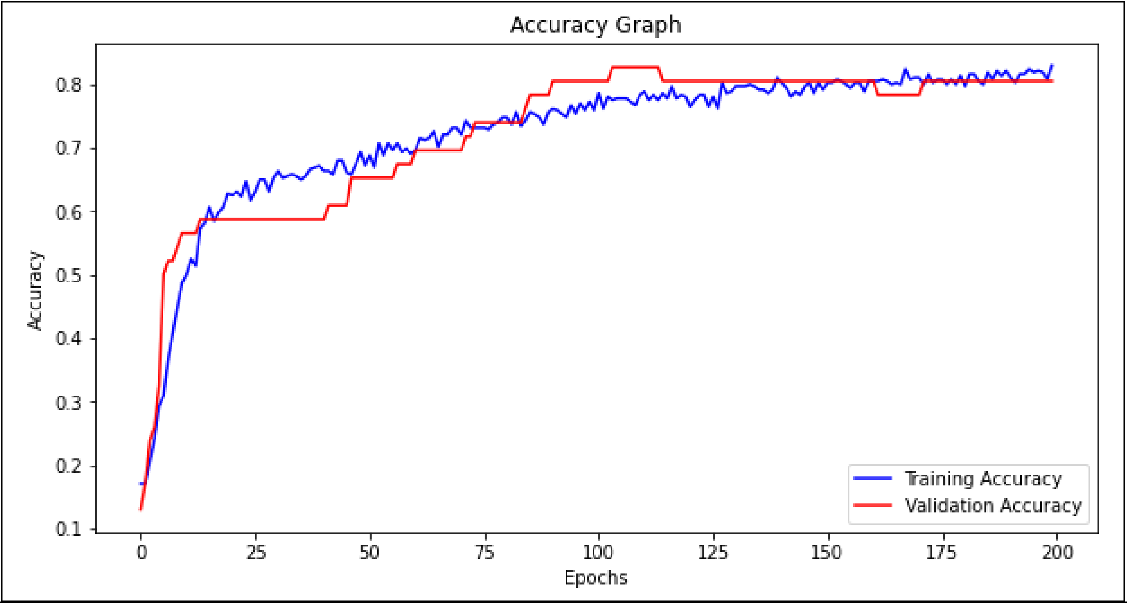
```
hist = model.fit([np.array(x_train['meta'].tolist()),np.array(x_train['image'].tolist())],y_train ,
                 epochs=200,validation_data=([np.array(x_test['meta'].tolist()),
                                              np.array(x_test['image'].tolist())],y_test),batch_size=16)

print("evaluate")
```

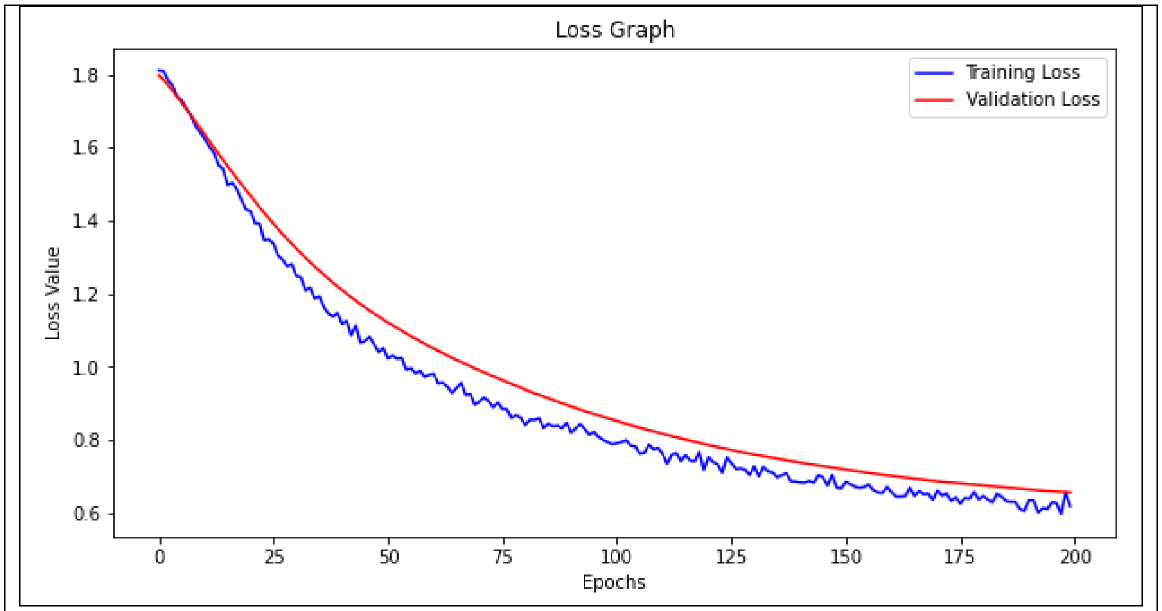
```
# test
model_loss,modelAccuracy = model.evaluate([np.array(x_valid['meta'].tolist()),np.array(x_valid['image'].tolist())],y_valid)
print("Model Accuracy "+str(modelAccuracy))
```

## 5.2.7 Visualization

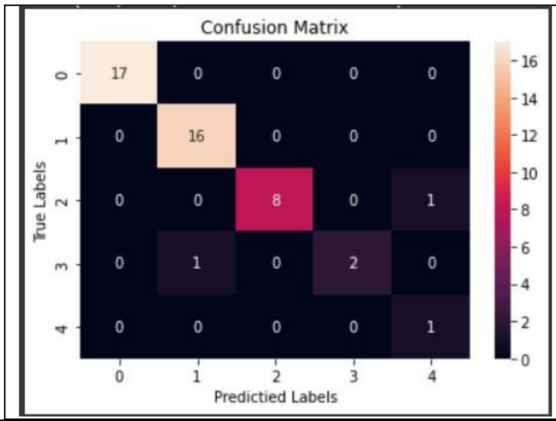
Visualize the accuracy graph and loss graph.

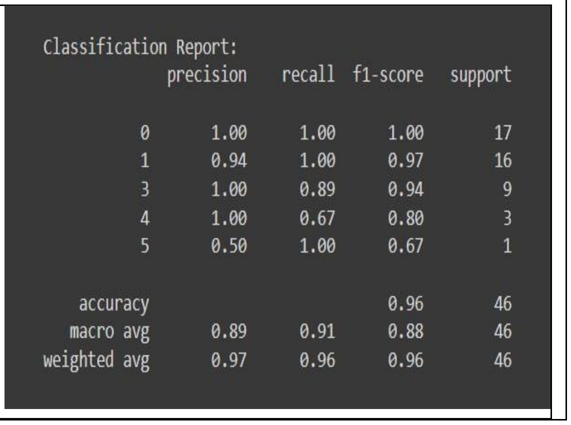
```
plt.figure(figsize=(10,5))
plt.plot(hist.history['accuracy'],color='b',label='Training Accuracy')
plt.plot(hist.history['val_accuracy'],color='r',label='Validation Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Accuracy Graph")
plt.legend(loc='lower right')

plt.figure(figsize=(10,5))
plt.plot(hist.history['loss'],color='b',label='Training Loss')
plt.plot(hist.history['val_loss'],color='r',label='Validation Loss')
plt.xlabel("Epochs")
plt.ylabel("Loss Value")
plt.title("Loss Graph")
plt.legend(loc='upper right')
```





In this model the dataset contains 52 of mel class so when we split the data first time to train the logistic model and random forest model and split it again to train and test the final model (dense layer), the mel class does not appear on test. This is the confusion matrix and its report that explain what we are talking about.





```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.94      1.00      0.97        16
           3       1.00      0.89      0.94         9
           4       1.00      0.67      0.80         3
           5       0.50      1.00      0.67         1

    accuracy                           0.96        46
   macro avg       0.89      0.91      0.88        46
weighted avg       0.97      0.96      0.96        46
```

## 5.3 Final model testing

To make sure that Scenario 6 was generic, we applied the same model to different
datasets, so we worked only with 10,000 images from each dataset, and the outcomes are as follows:

| Model | Images Only | Metadata only | Images + Metadata |
|---|---|---|---|
| HAM10000 | 49.1 | 87 | 90 |
| ISIC 2019 | 57.5 | 75.8 | 78 |
| HAM10000 (5000 Image only) | 72.8 | 86.2 | 88 |

## 5.4 Discussion

By comparing the expected results with our actual results,
We were able to get the same expected result for a higher accuracy of our designed machine learning model used to build our app.
We were able to get the same expected result of developing a mobile app that works easily with the user.

# Chapter 6: Conclusion & Future work

## 6.1 Conclusion

Through the use of technology, we can reduce the risk of diseases and their spread in societies, whether it is through developed applications or official health institution websites, and see first-hand the statistics and facts through these official sources and not listen to rumors, because people’s health is all they rely on and it is the fuel of their lives.

We hope that our application has contributed to maintaining the health of communities and has increased awareness and spread of correct treatment methods.

Finally, we hope that everyone realizes how "a few clicks" can affect society and make a huge difference to people’s life, and starting to pay more attention to technology and trying to use it to solve our daily big and small problems.

## 6.2 Future Work

1- This model depends on the data set of the skin being white. We want to use a data set of different skin colors in the future. In the future, we will collect metadata about people who used certain treatment methods and became free from this disease so that it is in the patient’s treatment plan and some instructions

2- We want to do an enhanced segmentation on the images to improve accuracy and accurately detect the lesion and separate it from the rest of the surroundings.

3- In the future, we want to add a new feature to the app which is easy contact with the doctors to help the users more.

# Bibliography

1- https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000?select=HAM10000_metadata.csv

2- https://www.researchgate.net/publication/321937436_Skin_Lesion_Classification_using_Machine_Learning_Algorithms

3- https://arxiv.org/pdf/1910.03910.pdf?fbclid=IwAR38hT9hRnRQWZvwSmxR61E-KgemflvqrsXsCeBY27xEKNSF8lDQiFXFVJ8

4- https://www.researchgate.net/publication/336415055_Performance_analysis_of_Convolutional_Neural_Network_CNN_based_Cancerous_Skin_Lesion_Detection_System

5- http://wscg.zcu.cz/WSCG2021/FULL/I02.pdf?fbclid=IwAR1tBmD8IPy0NrqNN2dnMzJdyj5VhCP9K9fK1E9bfIrRMSk9RG5OdHTsnik

6- https://arxiv.org/ftp/arxiv/papers/2003/2003.06276.pdf?fbclid=IwAR0aWhr3Pqq8HGgTHysROm8NFbXxvVrjZ5RX5AkTIagXoZXY1x3WxWG3Nls

7- https://www.kaggle.com/datasets/andrewmvd/isic-2019

8- https://www.nature.com/articles/nature21056

9- http://journal.waocp.org/article_32496_d2c3d8de9a47f9a9ca5d08a1ea9d1f04.pdf

10-
https://www.sciencedirect.com/science/article/pii/S2352914819302047

11-
https://dl.acm.org/doi/epdf/10.1145/3450439.3451873

12-
https://jamanetwork.com/journals/jama/article-abstract/2536642

13-
https://academic.oup.com/ptj/article/82/12/1232/2857765

14-
https://scholar.google.com/scholar?hl=en&q=Swati+Srivastava+Deepti+Sharma.+2016.+Automatically+Detection+of+Skin+Cancer+by+Classification+of+Neural+Network.+International+Journal+of+Engineering+and+Technical+Research+4%2C+1+%282016%29%2C+15%2D%2D18.#d=gs_qabs&u=%23p%3D5ri0mryK89IJ

15-
https://www.aafp.org/afp/2000/0715/p357.html?searchwebme

16-
https://dl.acm.org/doi/epdf/10.5555/3447307.3447320

17-
https://www.researchgate.net/profile/Satya-Singh-16/publication/339804392_Deep_Learning_Solutions_for_Skin_Cancer_Detection_and_Diagnosis/links/5ea16572299bf14389406872/Deep-Learning-Solutions-for-Skin-Cancer-Detection-and-Diagnosis.pdf

18-
https://otik.uk.zcu.cz/handle/11025/45011

19-
https://d1wqtxts1xzle7.cloudfront.net/61702792/714-Article_Text-2388-2-10-20190520200107-32906-51yql9-with-cover-page-v2.pdf?Expires=1633455587&Signature=A6cWeutN~LIm83y7vi3P-lOrbNpuopYfQpY-dzlG40Z2CMtLx6oDbicKuzJlGruJkr2Ur4SJFYUtY-KX5E6JdJFGR~6N6ThW8XMPORe4qv2gU7vnO4-

~TDMNSyaqHagIyqCtIW-WwSZ6QORaAzky-6r9WUTXu7ZJnLjurA8mTxdBpErE2PnEEUFklsTzQ9jbj5VAF2KP4WDRxcxBs9MkKWbvFM7EuyfH9~eh~CJIl~wVHg22wxQRPUGmCJfe2TsNYDEQV5OERtET-UuWtssUJ4TvHYznAOJu852bXBrcL537WhUl57wo3V6ln0mcxnsdOXsJt7MpWkB6wWobp6HpFQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

20- https://arxiv.org/ftp/arxiv/papers/2003/2003.06276.pdf

21- https://arxiv.org/pdf/1910.03910v1.pdf

22- https://www.sciencedirect.com/science/article/pii/S235234092031115X

23- https://arxiv.org/pdf/2104.14353.pdf

24- https://www.mdpi.com/1660-4601/18/10/5479/pdf

25- https://sjsronline.com/Papers/Papers/sjsrvol8no22016-5.pdf

26- https://arxiv.org/ftp/arxiv/papers/1810/1810.10348.pdf

27- https://web.stanford.edu/~kalouche/docs/Vision_Based_Classification_of_Skin_Cancer_using_Deep_Learning_(Kalouche).pdf