AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

A REPORT ON

NMS0001: PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

IV YEAR / VII SEM

R2019

Submitted by

JHANAVARSHINI.K.L - [130719106012]
NIVETHA.S - [130719106023]
PAVITHRA.J - [130719106024]
SWETHA.K - [130719106043]

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JERUSALEM COLLEGE OF ENGINEERING
(An Autonomous Institution, Affiliated to Anna University, Chennai)
NBA & NAAC ACCREDITED INSTITUTION
Velachery Main Road, Narayanapuram, Pallikaranai, Chennai - 600100

DECEMBER 2022

Team ID	PNT2022TMID07139
Project Name	AI-based localization and classification of skin disease with erythema
Team Members	JHANAVARSHINI.K.L - [130719106012] NIVETHA.S - [130719106023] PAVITHRA.J - [130719106024] SWETHA.K - [130719106043]
Branch	Electronics and Communication Engineering
Year	Final Year
Semester	Seventh Semester
College Name	Jerusalem College of Engineering
Faculty Mentor(s) Name	SHEEJA V FRANCIS
Industry Mentor(s) Name	Sri Tulasi
Project Github Link	IBM-EPBL/IBM-Project-19593-1659701859: AI-based localization and classification of skin disease with erythema (github.com)
Source Code Link	https://codesandbox.io/s/ibm-ai-based-localization-and-classification-of-skin-disease-with-erythema-ppdbd5
Demo Video Link	https://drive.google.com/file/d/1omvDRqVauNBdqCU_vjXCJfM6wta7MSQ-/view?usp=drivesdk
Reference Links	https://www.nature.com/articles/s41598-021-84593-z https://www.researchgate.net/publication/362845514_Performance_Evaluation_of_Deep_Learning_Techniques_for_Automated_Skin_Lesion_Diagnosis https://www.researchgate.net/publication/365227490_Reliable_detection_of_eczema_areas_for_fully_automated_assessment_of_eczema_severity_from_digital_camera_images

ABSTRACT

Skin diseases are more common than other diseases. Skin diseases may be caused by fungal infection, bacteria, viruses or allergy etc. A skin disease may change texture or colour of the skin. In general, skin diseases are chronic, infectious and sometimes may develop into skin cancer. The advancement of lasers and photonics based medical technology has made it possible to diagnose the skin diseases much more quickly and accurately though the cost of such diagnosis is very expensive. So, image processing techniques help to build automated screening system for dermatology at an initial stage. The extraction of features plays a key role in helping to classify skin diseases. We proposed an image processing-based method to detect skin diseases. This method takes the digital image of disease effect skin area, then use image analysis to identify the type of disease. Our proposed approach is simple, fast and does not require expensive equipment other than a camera and a computer. Then resize the of the image to extract features using pretrained convolutional neural network. After that classified feature using Multiclass SVM. Finally, the results are shown to the user, including the type of disease, spread, and severity. This method takes the digital image of disease effect skin area then use image analysis to identify the type of disease.

TABLE OF CONTENTS

1. Introduction

1.1 Project Overview

Now a day's people are suffering from skin diseases, more than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other.
To overcome the above problem, we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.

1.2 Purpose

The diseases are not considered skin diseases, and skin tone is majorly suffered from the ultraviolet rays from the sun. However, dermatologists perform the majority of non-invasive screening tests simply with the naked eye, even though skin illness is a frequent disease for which early detection and classification are essential for patient success and recovery. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

2. Literature Survey

2.1 Existing problem

A neglected public health problem Skin diseases are among the most common health problems in humans. Considering their significant impact on the individual, the family, the social life of patients, and their heavy economical burden, the public health importance of these diseases is underappreciated.

2.2 References

[1] J. Kawahara and G. Hamarneh, "Multi-resolution-tract CNN with hybrid pretrained and skin-lesion trained layers," in International Workshop on Machine Learning in Medical Imaging, pp. 164–171, Springer, New York, NY, USA, 2016.

[2] S. Verma, M. A. Razzaque, U. Sangtongdee, C. Arpnikanondt, B. Tassaneetrithep, and A. Hossain, "Digital diagnosis of Hand, Foot, and mouth disease using hybrid deep neural networks," IEEE Access, vol. 9, pp. 143481–143494, 2021.

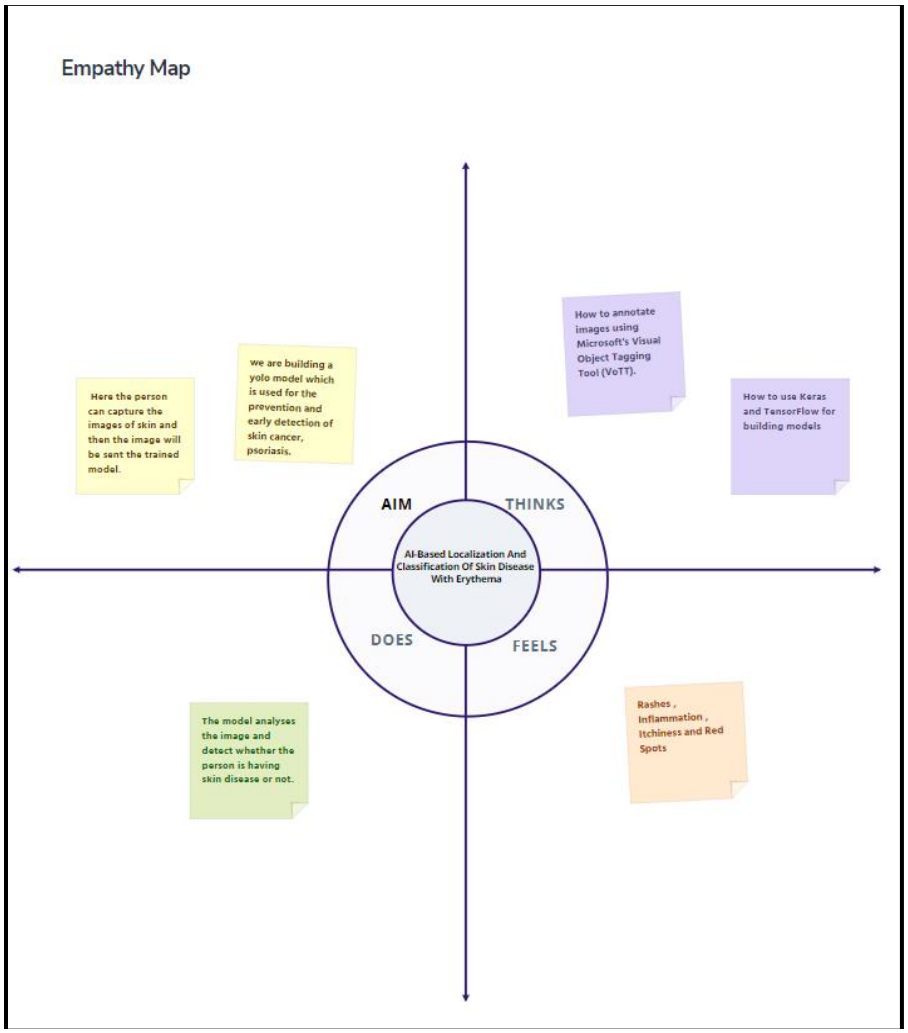
[3] P. P. Rebouças Filho, S. A. Peixoto, R. V. Medeiros da Nobrega´ et al., "Automatic histologically-closer classification of skin lesions," Computerized Medical Imaging and Graphics, vol. 68, pp. 40–54, 2018.

2.3 Problem Statement Definition

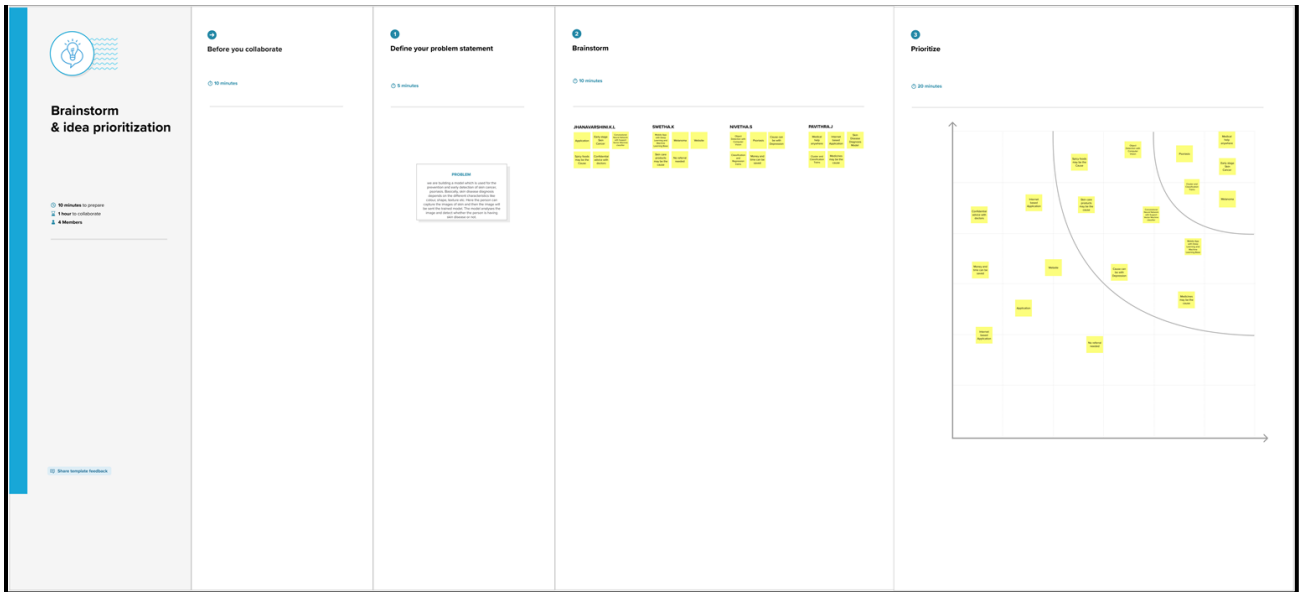
We're trying to find a solution to identify Skin Disease but Developed model is under training because given a image of skin, we can decompose, segment, and classify in a sequential manner which takes to Early detection of skin cancer, psoriasis.

3. Ideation and Proposed Solution

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming



3.3 Proposed Solution

Two-phase analysis model. The original image primarily enters a pre?processing stage, where normalization and decomposition occur. Afterwards, the first step is segmentation, where cluster of abnormal skin are segmented and cropped. The second step is classification, where each cluster is classified into its corresponding class. Developed Model is Still under training.

3.4 Problem Solution fit

Skin disease can appear in virtually any part of body and there is a lack of data required to form an association between the probability of a skin disease based on the body part. A Solution model used for the prevention and early detection of skin cancer and psoriasis by image analyses to detect whether the person is having skin disease or not. The location of the disease that is prsent in an image and improved performance by CNN model to focus on particular subsections of the images.

4. Requirement Analysis

4.1 Functional requirements

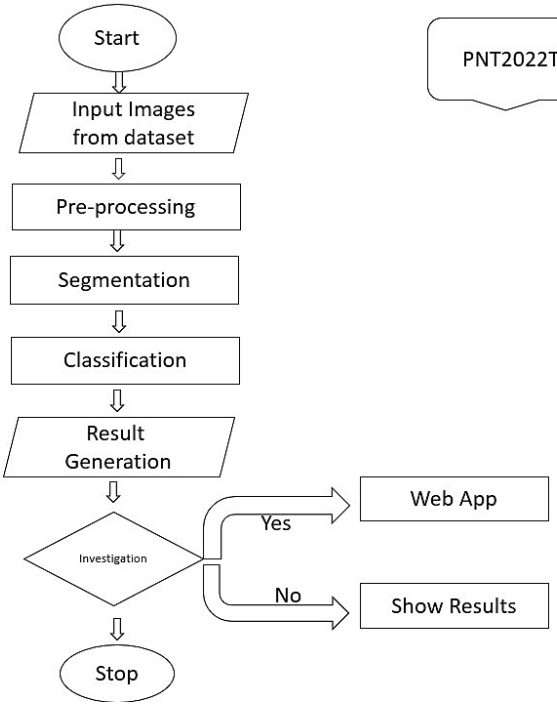
Image Acquisition, Preprocessing Steps such as Color gradient generator on an image , Cropping and isolating region of interest and Thresholding and Clustering on image, Visual feature extraction, System Training YOLO Model for Skin disease classification with deep learning and CNN, Separate access of application for admin, Diagnosis of Skin disease and Data retrieval and Data manipulation.

4.2 Non-Functional requirements

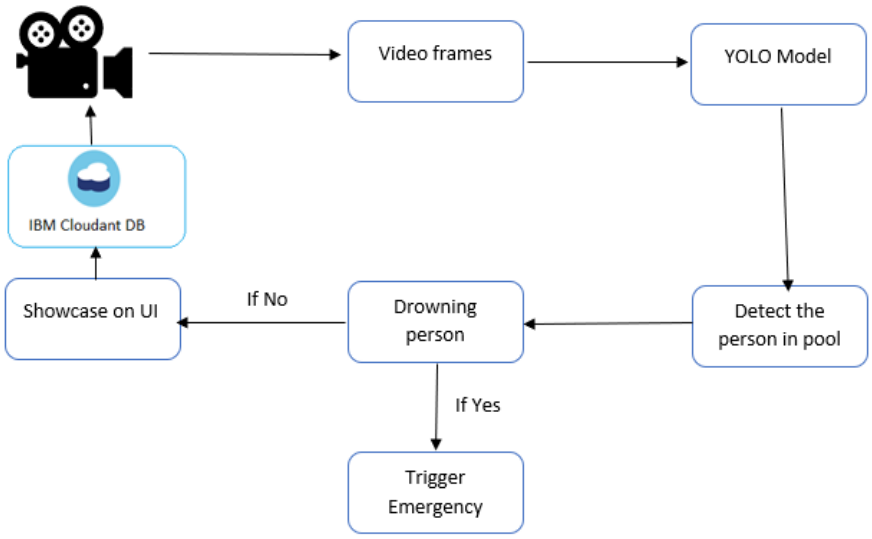
Software Quality Attributes, Prediction, Accuracy.

5. Project Design

5.1 Data Flow Diagram



5.2 Solution and Technical Architecture



5.3 User Stories

Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Prerequisites	USN-1	Install Python IDE, Python packages, Microsoft Visual Object Tagging Tool, Yolo Structure	3	High
Data Collection	USN-2	Dataset should be collected from google or using a Chrome extension such as Fatkun Batch Downloader	3	High
Annotate Images	USN-3	Create A Project in VOTT (Microsoft's Visual Object Tagging Tool)	2	Medium
Training YOLO	USN-4	train our model using YOLO weights	2	Medium
	USN-5	To Download and Convert Pre-Trained Weights	3	High
	USN-6	To Train YOLOv3 Detector	3	High
Cloudant DB	USN-7	Register & Login to IBM Cloud	3	High
	USN-8	Create Service Instant and Credentials	2	Medium
	USN-9	Launch DB and Create database	3	High
Development Phase	USN-10	To build a web application	3	High
	USN-11	Building HTML pages with python code	2	Medium
	USN-12	To run the application	3	High
Testing Phase	USN-13	As a user login to dashboard	2	Medium
	USN-14	As a user import the images with skin diseases to the software application	2	Medium
	USN-15	YOLO processes the image and give the necessary details	3	High

6. Project Planning and Scheduling

6.1 Sprint Planning and Estimation

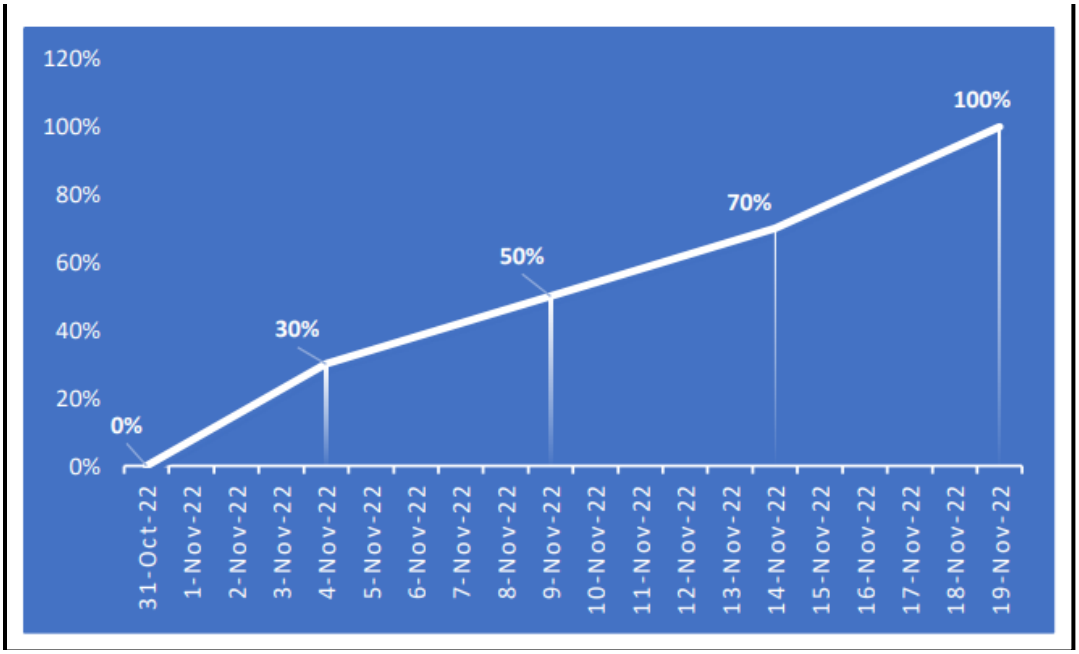
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Prerequisites	USN-1	Install Python IDE, Python packages, Microsoft Visual Object Tagging Tool, Yolo Structure	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-1	Data Collection	USN-2	Dataset should be collected from google or using a Chrome extension such as Fatkun Batch Downloader	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-1	Annotate Images	USN-3	Create A Project in VOTT (Microsoft's Visual Object Tagging Tool)	2	Medium	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-2	Training YOLO	USN-4	train our model using YOLO weights	2	Medium	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-2		USN-5	To Download and Convert Pre-Trained Weights	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-2		USN-6	To Train YOLOv3 Detector	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-3	Cloudant DB	USN-7	Register & Login to IBM Cloud	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-3		USN-8	Create Service Instant and Credentials	2	Medium	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-3		USN-9	Launch DB and Create database	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S

Sprint-3	Development Phase	USN-10	To build a web application	3	High	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-3		USN-11	Building HTML pages with python code	2	Medium	Jhanavarshini.K.L Swetha.K Pavithra.J Nivetha.S
Sprint-3		USN-12	To run the application	3	High	Jhanavarshini.K.L Swetha.K
Sprint-4	Testing Phase	USN-13	As a user login to dashboard	2	Medium	Jhanavarshini.K.L Nivetha.S
Sprint-4		USN-14	As a user import the images with skin diseases to the software application	2	Medium	Jhanavarshini.K.L Pavithra.J
Sprint-4		USN-15	YOLO processes the image and give the necessary details	3	High	Jhanavarshini.K.L

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022

6.3 Reports from Jira



7. Coding and Solutioning

7.1 Web App Dvelopment

7.1.1 Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">


    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
```

```
    <script src="https://kit.fontawesome.com/8b9cdc2059.js" crossorigin="anonymous"></script>
    <link href="https://fonts.googleapis.com/css2?family=Akronim&family=Times New
Roman&display=swap" rel="stylesheet">

    <title>SKINBOSS</title>
</head>
<body>
<style>
.icon-bar {
  width: 90px; /* Set a specific width */
  background-color: #555; /* Dark-grey background */
}

.icon-bar a {
  display: block; /* Make the links appear below each other instead of side-by-side */
  text-align: center; /* Center-align text */
  padding: 16px; /* Add some padding */
  transition: all 0.3s ease; /* Add transition for hover effects */
  color: white; /* White text color */
  font-size: 36px; /* Increased font-size */
}

.icon-bar a:hover {
  background-color:black; /* Add a hover color */
}

.active {
  background-color:grey; /* Add an active/current color */
}
.nav--items {
  overflow: hidden;
  background-color: #f1f1f1;
}

/* Style the buttons that are used to open the tab content */
.nav--items a {
  color:black;
  background-color: inherit;
  float: right;
  border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
}

/* Change background color of buttons on hover */
.nav--items a:hover {
  background-color: #ddd;
```

```
}

.nav--items a.active {
    color:black;
  background-color: #ccc;
}
.heading{
text-align:center;
color:white;
background-color:#153532;
}
.top{
background-color:#3a736d;
}
p{
color:black;
}
.head{
font-family:timesnewroman;
color:black;
text-align:center;
}

</style>
    <header id="head" class="header">

   <section id="navbar">
          <h1 class="heading">SKIN BOSS</h1>
       <div class="nav--items">

  <a href="{{url_for('login')}}">Log In</a>
  <a href="{{url_for('signup')}}">Sign Up</a>
             <a href="{{url_for('logout')}}">Log Out</a>
              <a href="{{url_for('prediction')}}">Prediction</a>
       </div>
   </section>

   <div class="top">
    <h2 class="title text-muted">
        <p style="font-family:timesnewromain">A PERFECT LIFE WITH PERFECT SKIN</p>
    </h2>

</div>
   <section id="slider">
    <div id="carouselExampleIndicators" class="carousel" data-ride="carousel">
        <ol class="carousel-indicators ">
            <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active "></li>
            <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
            <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
        </ol>
```

```
      <div class="carousel-item active">
        <img class="d-block w-100" src="https://regencyhealthcare.in/wp-
content/uploads/2021/04/Artboard-5-1.png" alt="First slide" style="height:500px">
      </div>
      <div class="carousel-item">
        <img alt="Second slide" class="d-block w-100"
src="https://img.graphicsurf.com/2020/10/skin-disease-vector-design-concept.jpg"
style="height:500px">
      </div>

      <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-
slide="prev">
        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
      </a>
      <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-
slide="next">
        <span class="carousel-control-next-icon" aria-hidden="true"></span>
        <span class="sr-only">Next</span>
      </a>
    </div>
      <div class="Problem">
        <h1 class="head">Problem Statement</h1>
        <p style="font-family:tahoma;">
          Now a day’s people are suffering from skin diseases, More than 125 million people
suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades
especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier
stage, then it may lead to complications in the body including spreading of the infection from one
individual to the other. The skin diseases can be prevented by investigating the infected region at
an early stage. The characteristic of the skin images is diversified so that it is a challenging job to
devise an efficient and robust algorithm for automatic detection of skin disease and its severity.
Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness
of skin are visually different. Automatic processing of such images for skin analysis requires
quantitative discriminator to differentiate the diseases.
To overcome the above problem we are building a model which is used for the prevention and
early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the
different characteristics like colour, shape, texture etc. Here the person can capture the images
of skin and then the image will be sent the trained model. The model analyses the image and
detect whether the person is having skin disease or not.</p>
        <h1 class="head">Proposed Solution</h1>
        <p style="font-family:tahoma;">
          Different skin disorders can be detected by just
          submitting photographs, and this approach is
          quite effective at helping people in the
          community identify ailments earlier.
          Our return on investment will be the creation
          and distribution of a proprietary product that
          will be used as a solution.
          This system is more scalable because it accepts
          any picture type, regardless of resolution, and
          offers good performance in any situation.
```

```
        </p></div>

    </header>

</body>
</html>
```

7.1.2 Login.html

```
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title> SKIN DISEASE </title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="static/css/style.css">
<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>


<style >
.header {
      top:0;
      margin:0px;
      left: 0px;
      right: 0px;
      position: fixed;
      background-color: #28272c;
      color: white;
      box-shadow: 0px 8px 4px grey;
      overflow: hidden;
      padding-left:20px;
      font-family: 'Josefin Sans';
      font-size: 2vw;
      width: 100%;
      height:8%;
      text-align: center;
    }
.topnav {
  overflow: hidden;
  background-color: #333;
}
```

```
.topnav-right a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
  background-color: #ddd;
  color: black;
}

.topnav-right a.active {
  background-color: #767d90;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
}

.SignIn{
margin-top:-70px;
}
body {

  background-color:#cb81a3fb;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  background-image: url('https://ak9.picdn.net/shutterstock/videos/29682919/thumb/1.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: cover;
  }
.SignIn{
  margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

input[type=text], input[type=email],input[type=number],input[type=password] {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}
```

```
button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 100%;
  font-weight:bold;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}

img.avatar {
  width: 30%;
  border-radius: 50%;
}

.container {
  padding: 16px;
}

span.psw {
  float: right;
  padding-top: 16px;

}


@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
```

```
}

</style>
</head>

<body style="font-family:Montserrat;">

<div class="header">
 <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-top:1%">SKIN
DISEASE</div>
  <div class="topnav-right" style="padding-top:0.5%;">

    <a  href="index.html">Home</a>
    <a class="active" href="login.html">Sign In</a>
    <a  href="register.html">Sign Up</a>

  </div>
</div>

<div id="SignIn" class="SignIn">


  <form action="prediction.html" method="post">
    <div class="imgcontainer">
      <h1>SIGN IN</h1>
      <h3 class="information-text">Enter your registered email and your password.</h3>
    </div>

    <div class="container">

      <input type="email" placeholder="Enter registered email ID" name="_id" required><br>

      <input type="password" placeholder="Enter Password" name="psw" required><br>

      <a href="forgot.html">Forgot password?</a><br><br>

      <button type="submit">Sign In</button><br>

    </div>
  </form>

</div>
</div>

</body>
</html>
```

7.1.3 Register.html

```
<!DOCTYPE html>
```

```
<html >

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title> SKIN DISEASE </title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="static/css/style.css">
<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>


<style >
.header {
      top:0;
      margin:0px;
      left: 0px;
      right: 0px;
      position: fixed;
      background-color: #28272c;
      color: white;
      box-shadow: 0px 8px 4px grey;
      overflow: hidden;
      padding-left:20px;
      font-family: 'Josefin Sans';
      font-size: 2vw;
      width: 100%;
      height:8%;
      text-align: center;
    }
.topnav {
  overflow: hidden;
  background-color: #333;
}

.topnav-right a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
  background-color: #ddd;
```

```
  color: black;
}

.topnav-right a.active {
  background-color: #565961;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
}

.SignIn{
margin-top:-70px;
}
body {

  background-color:#97dcf1fb;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  background-image: url('https://ak9.picdn.net/shutterstock/videos/29682919/thumb/1.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: cover;
  }
.SignIn{
  margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

input[type=text], input[type=email],input[type=number],input[type=password] {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 100%;
  font-weight:bold;
}
```

```
button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}

img.avatar {
  width: 30%;
  border-radius: 50%;
}

.container {
  padding: 16px;
}

span.psw {
  float: right;
  padding-top: 16px;

}


@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}

</style>
</head>

<body style="font-family:Montserrat;">

<div class="header">
 <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-top:1%">SKIN
DISEASE</div>
  <div class="topnav-right" style="padding-top:0.5%;">
```

```
    <a  href="index.html">Home</a>
    <a class="active" href="login.html">Sign In</a>
    <a  href="register.html">Sign Up</a>

  </div>
</div>

<div id="SignIn" class="SignIn">


  <form action="prediction.html" method="post">
    <div class="imgcontainer">
      <h1>SIGN IN</h1>
      <h3 class="information-text">Enter your registered email and your password.</h3>
    </div>

    <div class="container">

      <input type="email" placeholder="Enter registered email ID" name="_id" required><br>

      <input type="password" placeholder="Enter Password" name="psw" required><br>

      <a href="forgot.html">Forgot password?</a><br><br>

      <button type="submit">Sign In</button><br>

    </div>
  </form>

</div>
</div>

</body>
</html>
```

7.1.4 Predicton.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!--Bootstrap -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
```

```
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>


    <script src="https://kit.fontawesome.com/8b9cdc2059.js" crossorigin="anonymous"></script>
    <link href="https://fonts.googleapis.com/css2?family=Akronim&family=Roboto&display=swap"
rel="stylesheet">
    <style>
        :root{
    --main-bg-color: #fff;
    --text-color:#ced4da;
    --bs-font-sans-serif: Poppins, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica
Neue", Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI
Symbol", "Noto Color Emoji";
    --navbar-bg:#333;
    --hover-color:#228B22;
    --yellow:#FFD700;
    --box-shadow:rgba(100, 100, 111, 0.2) 0px 7px 29px 0px

}

/* reset */
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body{
    background: var(--main-bg-color);
    font-family: var(--bs-font-sans-serif);
    color: #333;
    line-height: 1.6;
}
ul{
    list-style:none;
}
a{
    text-decoration: none;
    color: var(--text-color);
}

h1,h2{
```

```
    font-weight: 360;
    line-height: 1.2;
}
p{
 margin: 10px 0px;

}

.m2{
    margin-right: 10px;
}

/* utility */
.title{
    margin-top: 10px;
    text-align: center;
}
html {
    scroll-behavior: smooth;
  }

/* Header */
#head #navbar{
    height: 100px;
    width: auto;
    background-color: var(--navbar-bg);
    color: #fff;
    padding: 10px;
}
#navbar{
    display: flex;
    justify-content: space-between;
    align-items: center;
}
#navbar .nav--items ul{
    display: flex;
    align-items: center;
}

#navbar .nav--items ul li a{
margin: 10px;
text-decoration: none;
}
#navbar .nav--items ul li a:hover{
    color:var(--hover-color) ;
}

/* header carousel */
#head #slider .carousel-item img{
    display: block;
    width:100%;
```

```
    height: 50vh;
}

.font{
    font-size: 50px;
    font-weight: bold;
    color: #fff;
}

/* About */
#about .top{
    margin-top: 20px;
}

 .line{
    background-color: var(--yellow);
    width: 200px;
    height: 2px;
    margin: auto;
    margin-top: 10px;
}
#about .body{
    margin-top: 20px;
    display: grid;
    grid-template-columns: 1fr 1fr;
    text-align: center;
}

#about .body .right,#about .body .left
{
    box-shadow: rgba(0, 0, 0, 0.15) 0px 3px 3px 0px;
margin: 0.5rem;
}

#about .body .right p{
    justify-self: center;
    margin-top: 50px;
}
/* Services */
#services .testimonials{
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-column-gap: 10px;
    grid-row-gap: 20px;
    margin: 40px;
   justify-items: center;

}
#services .testimonials .card{
 box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
 text-align: center;
```

```
}

#services .testimonials .card h5{
text-transform: uppercase;
}

/* Contcat form */
#contact .contact-container{
    display: grid;
    grid-template-columns: repeat(2,1fr);
    justify-items: center;
    margin: 3rem;
}
#contact .contact-container .conatct-left .items h3{
    display: inline;
    margin-left: 10px;
}

#contact .contact-container .conatct-left .items{
    margin: 10px;
    margin-bottom: 30px;

}

#contact .contact-container .contact-right form input,
#contact .contact-container .contact-right form button
 {
    display: block;
    margin: 20px
    }

 /* footer */
 #footer {
     width: auto;
     height: 80px;
        background-color: var(--navbar-bg);
        color: #fff;
        display: flex;
        align-items: center;
        justify-content: space-around;
 }
 #footer .social a{
    margin-left: 20px;
    text-decoration: none;
 }
 #footer .social a:hover{
     color: var(--hover-color);
 }
/* prediction.html */

#prediction .prediction-input{
```

```
    display: flex;
    align-items: center;
    justify-content: center;
    margin-top: 1.5rem;
}
#prediction .prediction-input form{
    margin-left: 1.2rem;
}
#prediction .circle {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    margin-bottom: 5px;
    box-shadow:var(--box-shadow);
    transition:all ease-in 1s;
}

.output{
    width: 200px;
    margin: 10rem 1.5rem;
    padding: 6px;
    text-align: center;
    box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
}
.output-container{
    display: grid;
    row-gap: 10px;
    grid-template-areas: 'img1 img2 img3 img4 img5 img6';
}

/* Hidden class */
.hidden{
    visibility: hidden;
}
.hide{
    visibility: hidden;
}
    </style>

    <script defer src="C:\Users\K L J Varshini\OneDrive\Desktop\IBM\IBM-Project-19593-
1659701859\yolo_structure\Static\JS\JScript.js.txt"></script>
    <title>Prediction</title>
</head>
<body>
    <header id="head" class="header">
        <section  id="navbar">
                <h1 class="nav-heading"></i>Skin Disease Detection</h1>
            <div class="nav--items">
                <ul>
                    <li><a  href="{{url_for('home')}}">Home</a></li>
    <li><a href="{{url_for('logout')}}">Logout</a></li>
```

```
            </ul>
          </div>
        </section>
    </header>
    <!-- dataset/Training/metal/metal326.jpg -->
        </br>
    <section id="prediction">
    <h2 class="title text-muted">SKIN ANALYTICS- AI-based localization and classification of skin disease with erythema</h1>
    <div class="line" style="width: 1000px;"></div>
         </section>
         </br>
    <section id="about">

<div class="body">
<div class="left">
    <p>
      Nowadays people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.
    </p>
</div>
<div class="left">

    <div class="prediction-input">
        <img class="d-block" src="https://img.graphicsurf.com/2020/06/Woman-in-a-protective-mask-against-the-virus-concept.jpg" alt="Second slide" style="width:80%!important; align:center; padding-left:100px">
        </br>
              <form id="form" action="/result" method="post"  enctype="multipart/form-data">
                 <input type="file" />
                 <input type="submit" class="submitbtn" value="Submit">
               </form>
          </div>

</div>
</div>
</section>
</body>
</html>
```

7.1.5 Logout.html

```
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>SKIN BOSS</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>


<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>

<style>
.header {
      top:0;
      margin:0px;
      left: 0px;
      right: 0px;
      position: fixed;
      background-color: #28272c;
      color: white;
      box-shadow: 0px 8px 4px grey;
      overflow: hidden;
      padding-left:20px;
      font-family: 'Josefin Sans';
      font-size: 2vw;
      width: 100%;
      height:8%;
      text-align: center;
    }
.topnav {
  overflow: hidden;
  background-color: #333;
}

.topnav-right a {
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
```

```
}

.topnav-right a:hover {
  background-color: #ddd;
  color: black;
}

.topnav-right a.active {
  background-color: #565961;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
}

.login{
margin-top:-70px;
}
body {

  background-color:#ffffff;
  background-repeat: no-repeat;
  background-size:cover;
  background-position: 0px 0px;
  }
.main{
  margin-top:100px;
  text-align:center;
}
form { margin-left:400px;margin-right:400px;}

input[type=text], input[type=email],input[type=number],input[type=password] {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
  cursor: pointer;
  width: 20%;
}
```

```
button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}

img.avatar {
  width: 30%;
  border-radius: 50%;
}

.container {
  padding: 16px;
}

span.psw {
  float: right;
  padding-top: 16px;

}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}

</style>
</head>

<body style="font-family:Montserrat;">

<div class="header">
 <div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-top:1%">SKIN
BOSS</div>
  <div class="topnav-right" style="padding-top:0.5%;">
```

```
    <a  href="{{url_for('home')}}">Home</a>
    <a href="{{url_for('login')}}">Log In</a>
    <a href="{{url_for('signup')}}">Sign Up</a>
  </div>
</div>
<div class="main">
<h1>Successfully Logged Out!</h1>
<h3 style="color:#4CAF50">Login for more information<h3>

  <a href="{{url_for('login')}}"><button type="submit">Login</button></a>
</form>
</div>

</body>
</html>
```

7.1.6 Python App.py

```
import re
import numpy as np
import os
from flask import Flask, app,request,render_template
import sys
from flask import Flask, request, render_template, redirect, url_for
import argparse
from tensorflow import keras
from PIL import Image
from timeit import default_timer as timer
import test
from pyngrok import ngrok
import pandas as pd
import numpy as np
import random

def get_parent_dir(n=1):
    """" returns the n-th parent dicrectory of the current
    working directory """"
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


src_path=r'C:\Users\K L J Varshini\OneDrive\Desktop\IBM\IBM-Project-19593-
1659701859\yolo_structure\2_Training\src'
print(src_path)
utils_path=r'C:\Users\K L J Varshini\OneDrive\Desktop\IBM\IBM-Project-19593-
1659701859\yolo_structure\Utils'
print(utils_path)
```

```
sys.path.append(src_path)
sys.path.append(utils_path)

import argparse
from keras_yolo3.yolo import YOLO, detect_video
from PIL import Image
from timeit import default_timer as timer
from utils import load_extractor_model, load_features, parse_input, detect_object
import test
import utils
import pandas as pd
import numpy as np
from Get_File_Paths import GetFileList
import random

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"

# Set up folder names for default values
data_folder = os.path.join(get_parent_dir(n=1), "yolo_structure", "Data")

image_folder = os.path.join(data_folder, "Source_Images")

image_test_folder = os.path.join(image_folder, "Test_Images")

detection_results_folder = os.path.join(image_folder, "Test_Image_Detection_Results")
detection_results_file = os.path.join(detection_results_folder, "Detection_Results.csv")

model_folder = os.path.join(data_folder, "Model_Weights")

model_weights = os.path.join(model_folder, "trained_weights_final.h5")
model_classes = os.path.join(model_folder, "data_classes.txt")

anchors_path = os.path.join(src_path, "keras_yolo3", "model_data", "yolo_anchors.txt")

FLAGS = None


from cloudant.client import Cloudant

# Authenticate using an IAM API key
client = Cloudant.iam('ce111064-57da-41da-8228-097576124e14-
bluemix','BH5oRD7XlEvhfUstaYQZP7RkGxmF-k1NndCB9IUlXMwD', connect=True)


# Create a database using an initialized client
my_database = client.create_database('my_database')


app=Flask(__name__)
port_no=5000
```

```
ngrok.set_auth_token("41bc80b6918b46beb7f2435a77b6345d_NVwpRyWjjsIuVoM3m8WYVraI
")
public_url =  ngrok.connect(port_no).public_url
print(f"To acces the Gloable link please click {public_url}")

#default home page or route
@app.route('/')
def index():
    return render_template('index.html')



@app.route('/index.html')
def home():
    return render_template("index.html")


#registration page
@app.route('/register')
def register():
    return render_template('register.html')

@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    '_id': x[1], # Setting _id is optional
    'name': x[0],
    'psw':x[2]
    }
    print(data)

    query = {'_id': {'$eq': data['_id']}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))

    if(len(docs.all())==0):
        url = my_database.create_document(data)
        #response = requests.get(url)
        return render_template('register.html', pred="Registration Successful, please login using
your details")
    else:
        return render_template('register.html', pred="You are already a member, please login using
your details")

#login page
@app.route('/login')
```

```
def login():
    return render_template('login.html')

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user = request.form['_id']
    passw = request.form['psw']
    print(user,passw)

    query = {'_id': {'$eq': user}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))

    if(len(docs.all())==0):
        return render_template('login.html', pred="The username is not found.")
    else:
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')


@app.route('/logout')
def logout():
    return render_template('logout.html')

@app.route('/prediction')
def prediction():
    return render_template('prediction.html',path="../static/img/6623.jpg",)


@app.route('/result',methods=["GET","POST"])
def res():
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """
    f = request.files['file']
    f.save("C:\Users\K L J Varshini\OneDrive\Desktop\IBM\IBM-Project-19593-
1659701859"+f.filename)




    parser.add_argument(
        "--input_path",
        type=str,
```

```
        default=image_test_folder,
        help="Path to image/video directory. All subdirectories will be included. Default is "
        + image_test_folder,
    )

    parser.add_argument(
        "--output",
        type=str,
        default=detection_results_folder,
        help="Output path for detection results. Default is "
        + detection_results_folder,
    )

    parser.add_argument(
        "--no_save_img",
        default=False,
        action="store_true",
        help="Only save bounding box coordinates but do not save output images with annotated
boxes. Default is False.",
    )

    parser.add_argument(
        "--file_types",
        "--names-list",
        nargs="*",
        default=[],
        help="Specify list of file types to include. Default is --file_types .jpg .jpeg .png .mp4",
    )

    parser.add_argument(
        "--yolo_model",
        type=str,
        dest="model_path",
        default=model_weights,
        help="Path to pre-trained weight files. Default is " + model_weights,
    )

    parser.add_argument(
        "--anchors",
        type=str,
        dest="anchors_path",
        default=anchors_path,
        help="Path to YOLO anchors. Default is " + anchors_path,
    )

    parser.add_argument(
        "--classes",
        type=str,
        dest="classes_path",
        default=model_classes,
        help="Path to YOLO class specifications. Default is " + model_classes,
```

```
    )

    parser.add_argument(
        "--gpu_num", type=int, default=1, help="Number of GPU to use. Default is 1"
    )

    parser.add_argument(
        "--confidence",
        type=float,
        dest="score",
        default=0.25,
        help="Threshold for YOLO object confidence score to show predictions. Default is 0.25.",
    )

    parser.add_argument(
        "--box_file",
        type=str,
        dest="box",
        default=detection_results_file,
        help="File to save bounding box results to. Default is "
        + detection_results_file,
    )

    parser.add_argument(
        "--postfix",
        type=str,
        dest="postfix",
        default="_disease",
        help='Specify the postfix for images with bounding boxes. Default is "_disease"',
    )




    yolo = YOLO(
        **{
            "model_path": FLAGS.model_path,
            "anchors_path": FLAGS.anchors_path,
            "classes_path": FLAGS.classes_path,
            "score": FLAGS.score,
            "gpu_num": FLAGS.gpu_num,
            "model_image_size": (416, 416),
        }
    )


    img_path="C:\Users\K L J Varshini\OneDrive\Desktop\IBM\IBM-Project-19593-
1659701859\Static"+f.filename
    prediction, image,lat,lon= detect_object(
            yolo,
            img_path,
```

```
        save_img=save_img,
        save_img_path=FLAGS.output,
        postfix=FLAGS.postfix,
    )

    yolo.close_session()
    return
render_template('prediction.html',prediction=str(prediction),path="../static/img/"+f.filename)

""" Running our application """
if __name__ == "__main__":
    app.run(port=port_no)
```

7.2 A Separate Coding Part to understand the Skin Cancer Detection using TensorFlow in Python

```
pip3 install tensorflow tensorflow_hub matplotlib seaborn numpy pandas sklearn imblearn
import tensorflow as tf
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from tensorflow.keras.utils import get_file
from sklearn.metrics import roc_curve, auc, confusion_matrix
from imblearn.metrics import sensitivity_score, specificity_score

import os
import glob
import zipfile
import random

# to get consistent results after multiple runs
tf.random.set_seed(7)
np.random.seed(7)
random.seed(7)

# 0 for benign, 1 for malignant
class_names = ["benign", "malignant"]
```

Preparing the Dataset

```
def download_and_extract_dataset():
  # dataset from https://github.com/udacity/dermatologist-ai
  # 5.3GB
  train_url = "https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-cancer/train.zip"
  # 824.5MB
  valid_url = "https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-cancer/valid.zip"
  # 5.1GB
  test_url  = "https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-cancer/test.zip"
  for i, download_link in enumerate([valid_url, train_url, test_url]):
    temp_file = f"temp{i}.zip"
    data_dir = get_file(origin=download_link, fname=os.path.join(os.getcwd(), temp_file))
```

```
    print("Extracting", download_link)
    with zipfile.ZipFile(data_dir, "r") as z:
      z.extractall("data")
    # remove the temp file
    os.remove(temp_file)

# comment the below line if you already downloaded the dataset
download_and_extract_dataset()
# preparing data
# generate CSV metadata file to read img paths and labels from it
def generate_csv(folder, label2int):
    folder_name = os.path.basename(folder)
    labels = list(label2int)
    # generate CSV file
    df = pd.DataFrame(columns=["filepath", "label"])
    i = 0
    for label in labels:
        print("Reading", os.path.join(folder, label, "*"))
        for filepath in glob.glob(os.path.join(folder, label, "*")):
            df.loc[i] = [filepath, label2int[label]]
            i += 1
    output_file = f"{folder_name}.csv"
    print("Saving", output_file)
    df.to_csv(output_file)

# generate CSV files for all data portions, labeling nevus and seborrheic keratosis
# as 0 (benign), and melanoma as 1 (malignant)
# you should replace "data" path to your extracted dataset path
# don't replace if you used download_and_extract_dataset() function
generate_csv("data/train", {"nevus": 0, "seborrheic_keratosis": 0, "melanoma": 1})
generate_csv("data/valid", {"nevus": 0, "seborrheic_keratosis": 0, "melanoma": 1})
generate_csv("data/test", {"nevus": 0, "seborrheic_keratosis": 0, "melanoma": 1})
# loading data
train_metadata_filename = "train.csv"
valid_metadata_filename = "valid.csv"
# load CSV files as DataFrames
df_train = pd.read_csv(train_metadata_filename)
df_valid = pd.read_csv(valid_metadata_filename)
n_training_samples = len(df_train)
n_validation_samples = len(df_valid)
print("Number of training samples:", n_training_samples)
print("Number of validation samples:", n_validation_samples)
train_ds = tf.data.Dataset.from_tensor_slices((df_train["filepath"], df_train["label"]))
valid_ds = tf.data.Dataset.from_tensor_slices((df_valid["filepath"], df_valid["label"]))
```

Output:

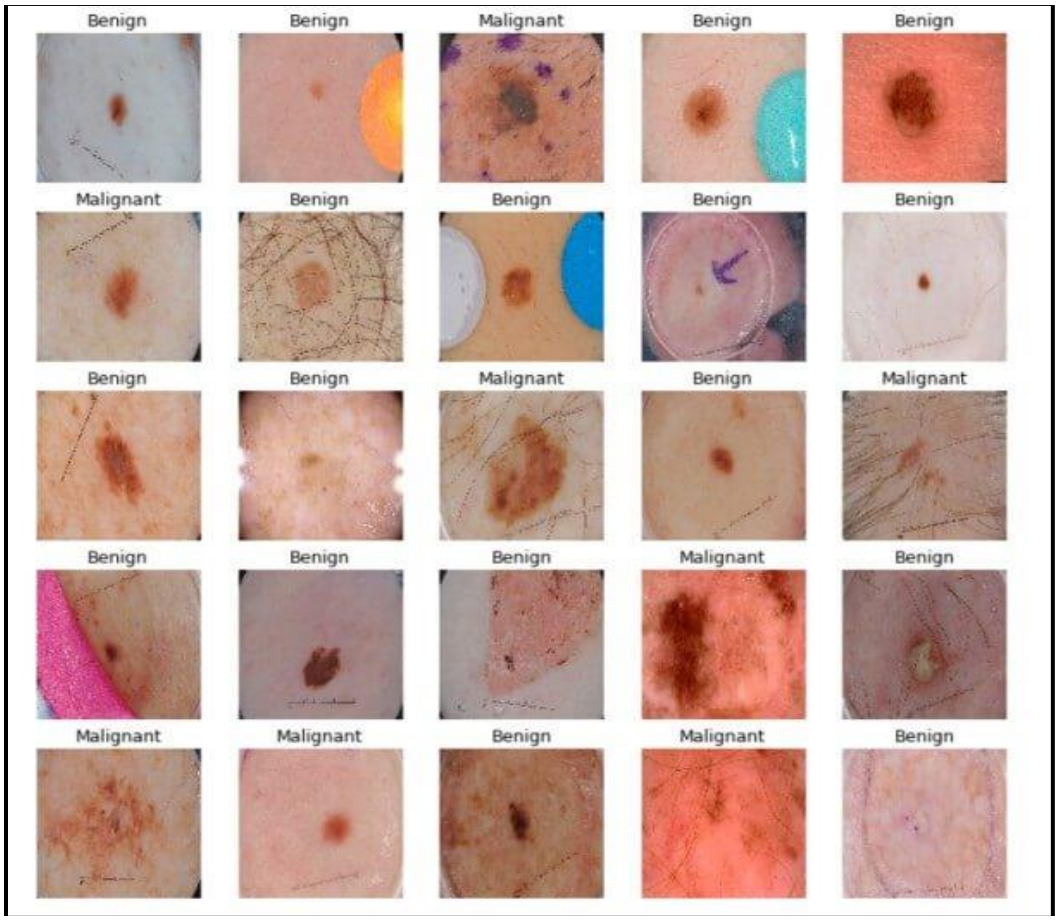
Number of training samples: 2000
Number of validation samples: 150

```
# preprocess data
def decode_img(img):
```

```
# convert the compressed string to a 3D uint8 tensor
  img = tf.image.decode_jpeg(img, channels=3)
  # Use `convert_image_dtype` to convert to floats in the [0,1] range.
  img = tf.image.convert_image_dtype(img, tf.float32)
  # resize the image to the desired size.
  return tf.image.resize(img, [299, 299])


def process_path(filepath, label):
  # load the raw data from the file as a string
  img = tf.io.read_file(filepath)
  img = decode_img(img)
  return img, label


valid_ds = valid_ds.map(process_path)
train_ds = train_ds.map(process_path)
# test_ds = test_ds
for image, label in train_ds.take(1):
    print("Image shape:", image.shape)
    print("Label:", label.numpy())
Image shape: (299, 299, 3)
Label: 0
# training parameters
batch_size = 64
optimizer = "rmsprop"

def prepare_for_training(ds, cache=True, batch_size=64, shuffle_buffer_size=1000):
  if cache:
    if isinstance(cache, str):
      ds = ds.cache(cache)
    else:
      ds = ds.cache()
  # shuffle the dataset
  ds = ds.shuffle(buffer_size=shuffle_buffer_size)
  # Repeat forever
  ds = ds.repeat()
  # split to batches
  ds = ds.batch(batch_size)
  # `prefetch` lets the dataset fetch batches in the background while the model
  # is training.
  ds = ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
  return ds

valid_ds = prepare_for_training(valid_ds, batch_size=batch_size, cache="valid-cached-data")
train_ds = prepare_for_training(train_ds, batch_size=batch_size, cache="train-cached-data")
batch = next(iter(valid_ds))

def show_batch(batch):
  plt.figure(figsize=(12,12))
  for n in range(25):
      ax = plt.subplot(5,5,n+1)
      plt.imshow(batch[0][n])
      plt.title(class_names[batch[1][n].numpy()].title())
      plt.axis('off')
```

show_batch(batch)

Output:




```
# building the model
# InceptionV3 model & pre-trained weights
module_url = "https://tfhub.dev/google/tf2-preview/inception_v3/feature_vector/4"

m = tf.keras.Sequential([
    hub.KerasLayer(module_url, output_shape=[2048], trainable=False),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

m.build([None, 299, 299, 3])
m.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accuracy"])
m.summary()
```

Output:

Model: "sequential"

_____

Layer (type)	**Output Shape**	**Param #**
keras_layer (KerasLayer)	**multiple**	**21802784**

```
_________________________________________________________________
dense (Dense)                multiple                  2049
=================================================================
Total params: 21,804,833
Trainable params: 2,049
Non-trainable params: 21,802,784
_________________________________________________________________
```

Training the Model

```
model_name = f"benign-vs-malignant_{batch_size}_{optimizer}"
tensorboard = tf.keras.callbacks.TensorBoard(log_dir=os.path.join("logs", model_name))
# saves model checkpoint whenever we reach better weights
modelcheckpoint = tf.keras.callbacks.ModelCheckpoint(model_name + "_{val_loss:.3f}.h5",
save_best_only=True, verbose=1)

history = m.fit(train_ds, validation_data=valid_ds,
          steps_per_epoch=n_training_samples // batch_size,
          validation_steps=n_validation_samples // batch_size, verbose=1, epochs=100,
          callbacks=[tensorboard, modelcheckpoint])
```

Output:

```
Train for 31 steps, validate for 2 steps
Epoch 1/100
30/31 [============================>.] - ETA: 9s - loss: 0.4609 - accuracy: 0.7760
Epoch 00001: val_loss improved from inf to 0.49703, saving model to benign-vs-
malignant_64_rmsprop_0.497.h5
31/31 [==============================] - 282s 9s/step - loss: 0.4646 - accuracy: 0.7722 -
val_loss: 0.4970 - val_accuracy: 0.8125
<..SNIPED..>
Epoch 27/100
30/31 [============================>.] - ETA: 0s - loss: 0.2982 - accuracy: 0.8708
Epoch 00027: val_loss improved from 0.40253 to 0.38991, saving model to benign-vs-
malignant_64_rmsprop_0.390.h5
31/31 [==============================] - 21s 691ms/step - loss: 0.3025 - accuracy: 0.8684 -
val_loss: 0.3899 - val_accuracy: 0.8359
<..SNIPED..>
Epoch 41/100
30/31 [============================>.] - ETA: 0s - loss: 0.2800 - accuracy: 0.8802
Epoch 00041: val_loss did not improve from 0.38991
31/31 [==============================] - 21s 690ms/step - loss: 0.2829 - accuracy: 0.8790 -
val_loss: 0.3948 - val_accuracy: 0.8281
Epoch 42/100
30/31 [============================>.] - ETA: 0s - loss: 0.2680 - accuracy: 0.8859
Epoch 00042: val_loss did not improve from 0.38991
31/31 [==============================] - 21s 693ms/step - loss: 0.2722 - accuracy: 0.8831 -
val_loss: 0.4572 - val_accuracy: 0.8047
```

Model Evaluation:

```
# evaluation

# load testing set
```

```
test_metadata_filename = "test.csv"
df_test = pd.read_csv(test_metadata_filename)
n_testing_samples = len(df_test)
print("Number of testing samples:", n_testing_samples)
test_ds = tf.data.Dataset.from_tensor_slices((df_test["filepath"], df_test["label"]))
def prepare_for_testing(ds, cache=True, shuffle_buffer_size=1000):
  if cache:
    if isinstance(cache, str):
      ds = ds.cache(cache)
    else:
      ds = ds.cache()
  ds = ds.shuffle(buffer_size=shuffle_buffer_size)
  return ds
test_ds = test_ds.map(process_path)
test_ds = prepare_for_testing(test_ds, cache="test-cached-data")
```

```
Number of testing samples: 600
# evaluation
# load testing set
test_metadata_filename = "test.csv"
df_test = pd.read_csv(test_metadata_filename)
n_testing_samples = len(df_test)
print("Number of testing samples:", n_testing_samples)
test_ds = tf.data.Dataset.from_tensor_slices((df_test["filepath"], df_test["label"]))


def prepare_for_testing(ds, cache=True, shuffle_buffer_size=1000):
  if cache:
    if isinstance(cache, str):
      ds = ds.cache(cache)
    else:
      ds = ds.cache()
```

```
  ds = ds.shuffle(buffer_size=shuffle_buffer_size)
  return ds
test_ds = test_ds.map(process_path)
test_ds = prepare_for_testing(test_ds, cache="test-cached-data")


# load the weights with the least loss
m.load_weights("benign-vs-malignant_64_rmsprop_0.390.h5")
print("Evaluating the model...")
loss, accuracy = m.evaluate(X_test, y_test, verbose=0)
print("Loss:", loss, "  Accuracy:", accuracy)
```

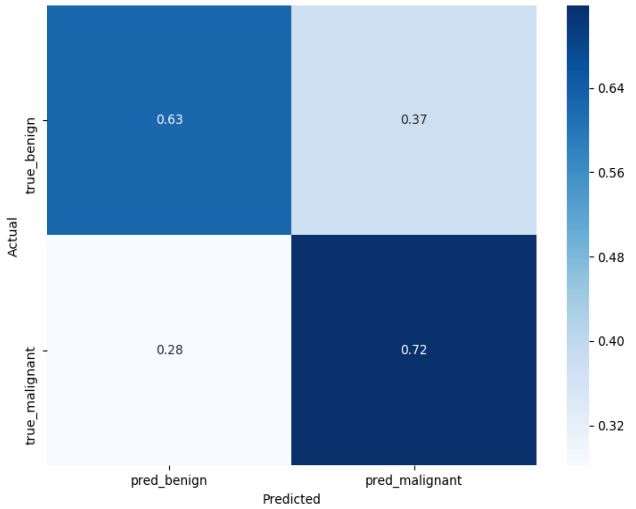
Output:

Evaluating the model...
Loss: 0.4476394319534302 Accuracy: 0.8

```
def get_predictions(threshold=None):
  """
  Returns predictions for binary classification given `threshold`
  For instance, if threshold is 0.3, then it'll output 1 (malignant) for that sample if
  the probability of 1 is 30% or more (instead of 50%)
  """
  y_pred = m.predict(X_test)
  if not threshold:
    threshold = 0.5
  result = np.zeros((n_testing_samples,))
  for i in range(n_testing_samples):
    # test melanoma probability
    if y_pred[i][0] >= threshold:
      result[i] = 1
    # else, it's 0 (benign)
  return result

threshold = 0.23
# get predictions with 23% threshold
# which means if the model is 23% sure or more that is malignant,
# it's assigned as malignant, otherwise it's benign
y_pred = get_predictions(threshold)
def plot_confusion_matrix(y_test, y_pred):
  cmn = confusion_matrix(y_test, y_pred)
  # Normalise
  cmn = cmn.astype('float') / cmn.sum(axis=1)[:, np.newaxis]
  # print it
  print(cmn)
  fig, ax = plt.subplots(figsize=(10,10))
  sns.heatmap(cmn, annot=True, fmt='.2f',
              xticklabels=[f"pred_{c}" for c in class_names],
              yticklabels=[f"true_{c}" for c in class_names],
              cmap="Blues"
              )
```

```
  plt.ylabel('Actual')
  plt.xlabel('Predicted')
  # plot the resulting confusion matrix
  plt.show()

plot_confusion_matrix(y_test, y_pred)
```

Output:




```
sensitivity = sensitivity_score(y_test, y_pred)
specificity = specificity_score(y_test, y_pred)

print("Melanoma Sensitivity:", sensitivity)
print("Melanoma Specificity:", specificity)
```

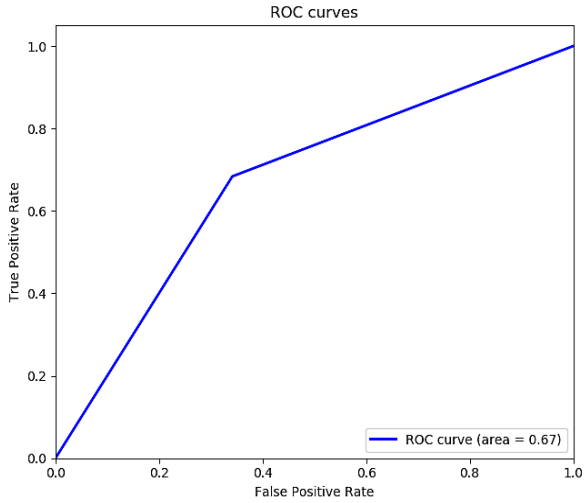
Output:

Melanoma Sensitivity: 0.717948717948718
Melanoma Specificity: 0.6252587991718427

```
def plot_roc_auc(y_true, y_pred):
    """
    This function plots the ROC curves and provides the scores.
    """
    # prepare for figure
    plt.figure()
    fpr, tpr, _ = roc_curve(y_true, y_pred)
    # obtain ROC AUC
    roc_auc = auc(fpr, tpr)
    # print score
    print(f"ROC AUC: {roc_auc:.3f}")
    # plot ROC curve
```

```
    plt.plot(fpr, tpr, color="blue", lw=2,
             label='ROC curve (area = {f:.2f})'.format(d=1, f=roc_auc))
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC curves')
    plt.legend(loc="lower right")
    plt.show()

plot_roc_auc(y_test, y_pred)
```

Output:



ROC AUC: 0.671

8. Testing

8.1 Test Cases

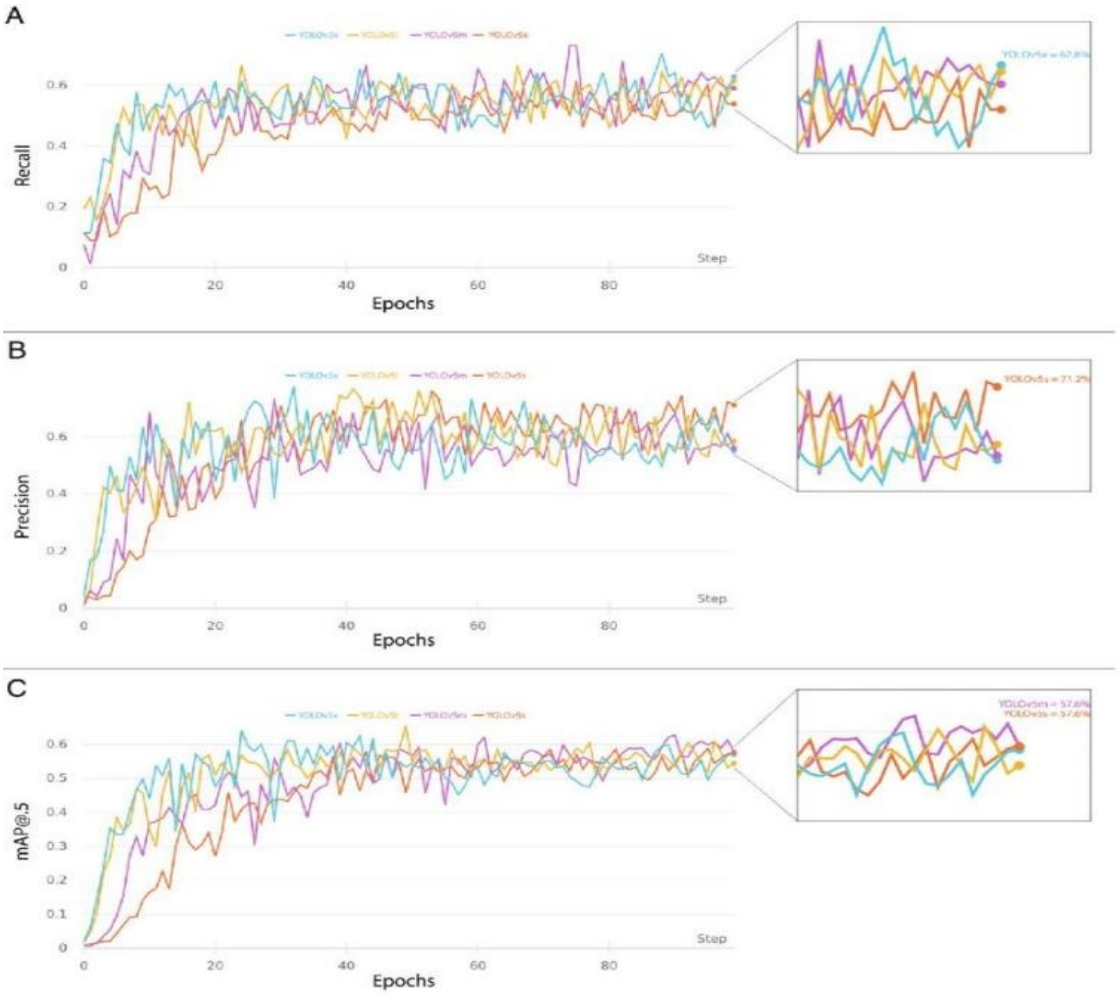
				Date	03-Nov-22							
				Team ID	PNT2022TMID07139							
				Project Name	Project - AI-based localization and classification of skin disease with erythema							
				Maximum Marks	4 marks							
Test case ID	**Feature Type**	**Component**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Comments**	**TC for Automation(Y/N)**	**BUG ID**	**Executed By**
HomePage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	Python Code(VS Code) , Chrome or Microsoft Edge	1.Enter URL and click go 2.User must be able to see the Home Page	From VS Code run the codewith edge or chrome to see the home page.	Home Page should be displayed	Working as expected	Pass	Nil	N	Nil	Jhanavarshini.K.L
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Home Page	Python Code(VS Code) , Chrome or Microsoft Edge	1.Enter URL and click go 2.Verify UI elements in home page	Enter URL and click go	Application should show below UI elements: a.email text box b.password text box c.Login button d.If New then Register e.Forgot password? Recovery password link	Working as expected	Pass	Nil	N	Nil	Swetha.K
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	Python Code(VS Code) , Chrome or Microsoft Edge	1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	User will be able to log into Home page	User should navigate to user account homepage	Working as expected	Pass	Nil	N	Nil	Pavithra.J
PredictPage_TC_OO4	Functional	Prediction page	Verify user is able to redirected to Prediction Page	Python Code(VS Code) , Chrome or Microsoft Edge	1.Enter URL and click go 2.Enter UI Elements in Predict Page	User must be able to see the prediction page	User should navigate to prediction page	Working as expected	Pass	Nil	N	Nil	Nivetha.S
PredictPage_TC_OO5	Functional	Prediction page	Verify user able to see the dropdown value or not	Python Code(VS Code) , Chrome or Microsoft Edge, Dataset (Images)	1.Enter URL and click go 2.Verify whether user can be redirected to prediction page 3.Verify whether user is able to select the dropdown value	Skin Diseases	Application should show skin diseases option in dropdown list	Working as expected	Pass	Nil	N	Nil	Pavithra.J
PredictPage_TC_OO6	Functional	Prediction page	Verify user is able to upload the image or not	Python Code(VS Code) , Chrome or Microsoft Edge, Dataset (Images)	1.Enter URL and click go 2.Verify whether user can be redirected to prediction page 3.Verify whether user is able to select the dropdown value 4.Verify whether the user can upload the images or not	Upload Images	Application should show the uploaded image	Working as expected	Pass	Nil	N	Nil	Nivetha.S
PredictPage_TC_OO7	Functional	Prediction page	Verify whether the image is predicted or not	Python Code(VS Code) , Chrome or Microsoft Edge, Dataset (Images)	1.Enter URL and click go 2.Verify whether user can be redirected to prediction page 3.Verify whether user is able to select the dropdown value 4.Verify whether the user can upload the images or not 5.Verify whether the image is predicted correctly or not	Click the Submit Button	Application should show the predicted output	Working as expected	Pass	Nil	N	Nil	Jhanavarshini.K.L

8.2 User Acceptance Testing

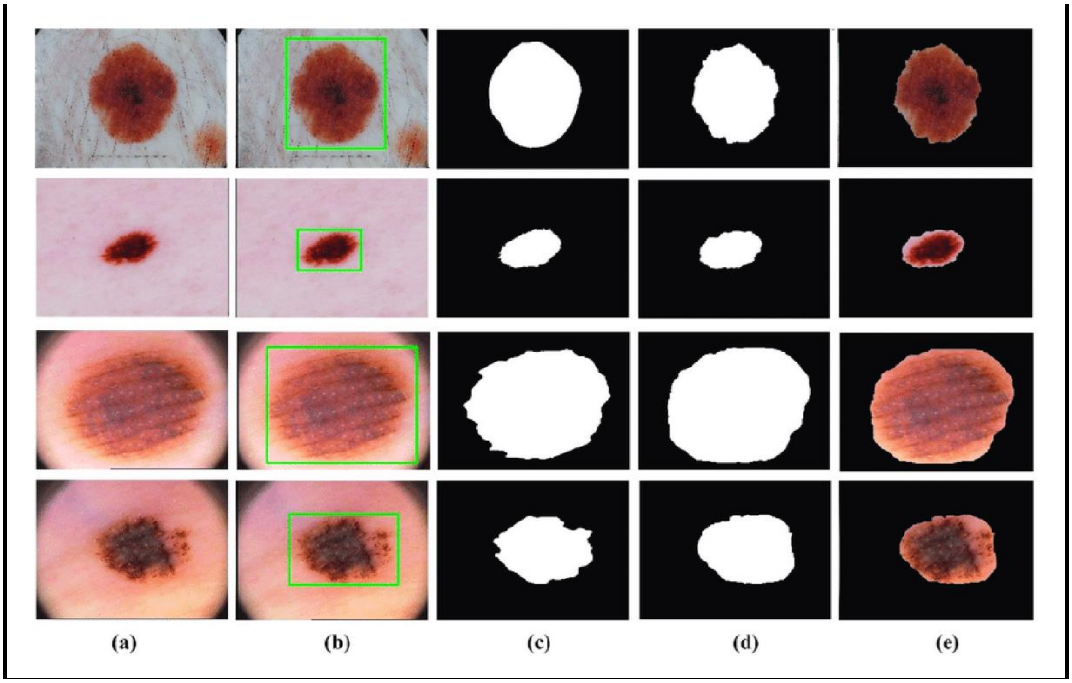
Section	Total Cases	Not Tested	Fail	Pass
Header Section - Website	3	0	0	3
Footer Section - Website	2	0	0	2
Log in	3	0	1	2
Register	3	0	0	3
Upload Image	5	1	2	2
Forgot Password	3	0	1	2
Button	1	0	0	1
Final Report Output	20	1	4	15

9. Results

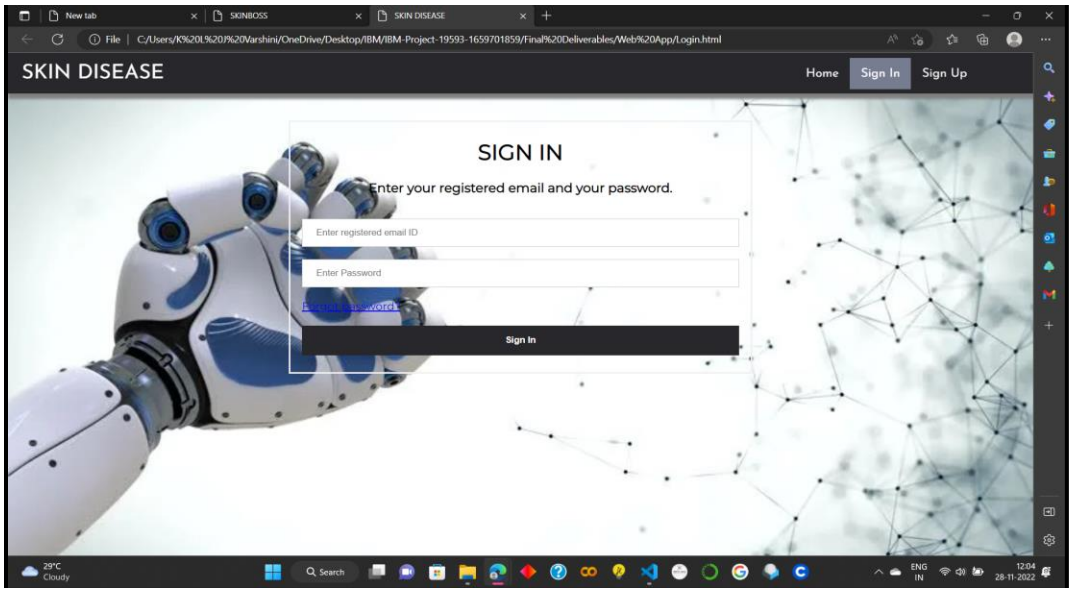
9.1 Performance Metrics

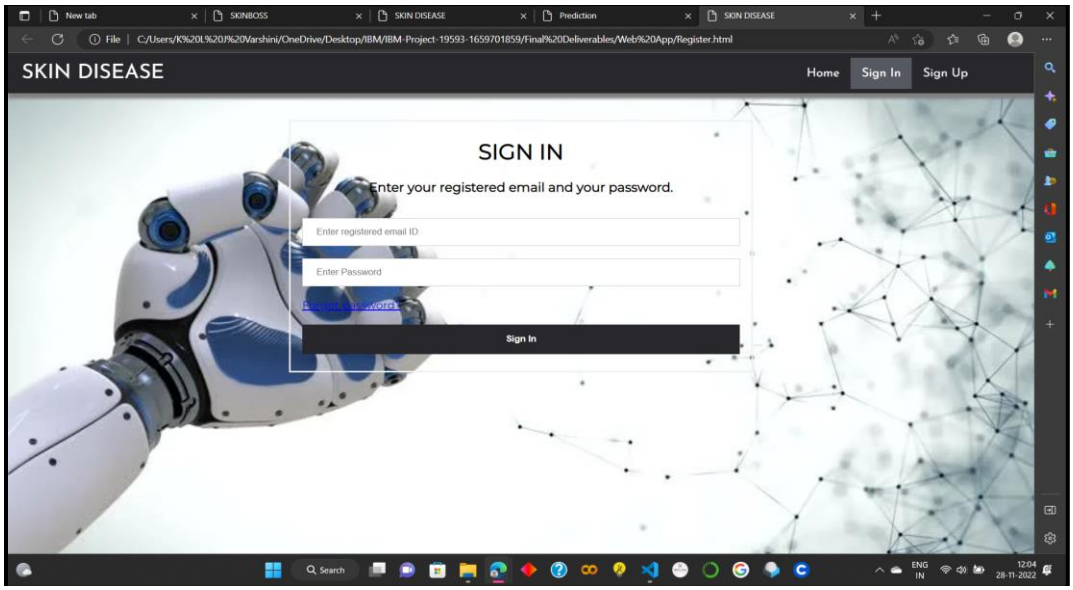


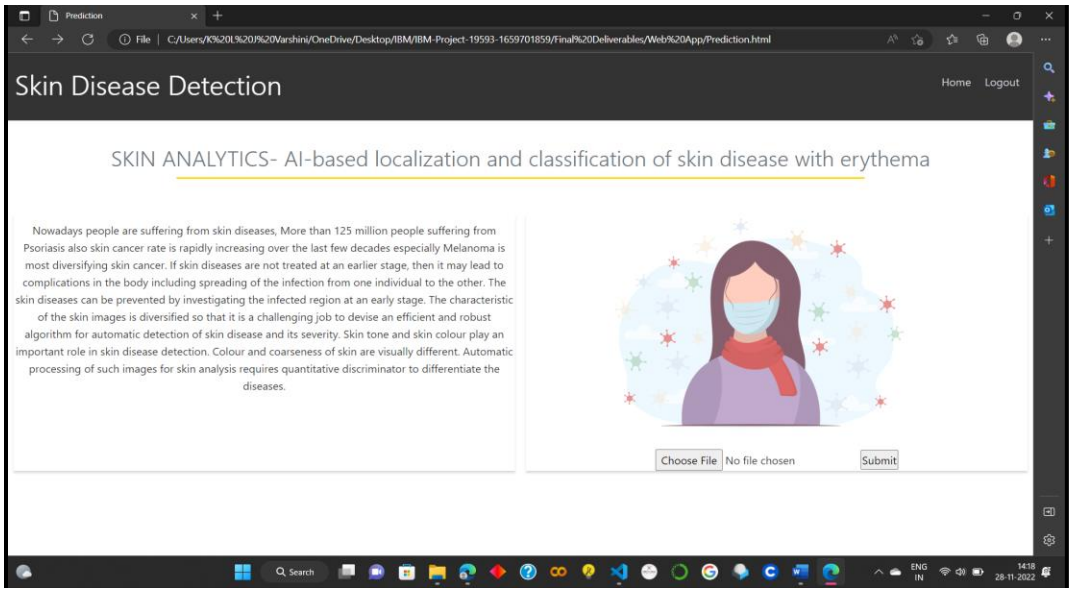
The final results are based on the accuracy results in the form of the melanoma and the non-melanoma skin diseases classifications. For a better understanding please have a look at the section 7.2

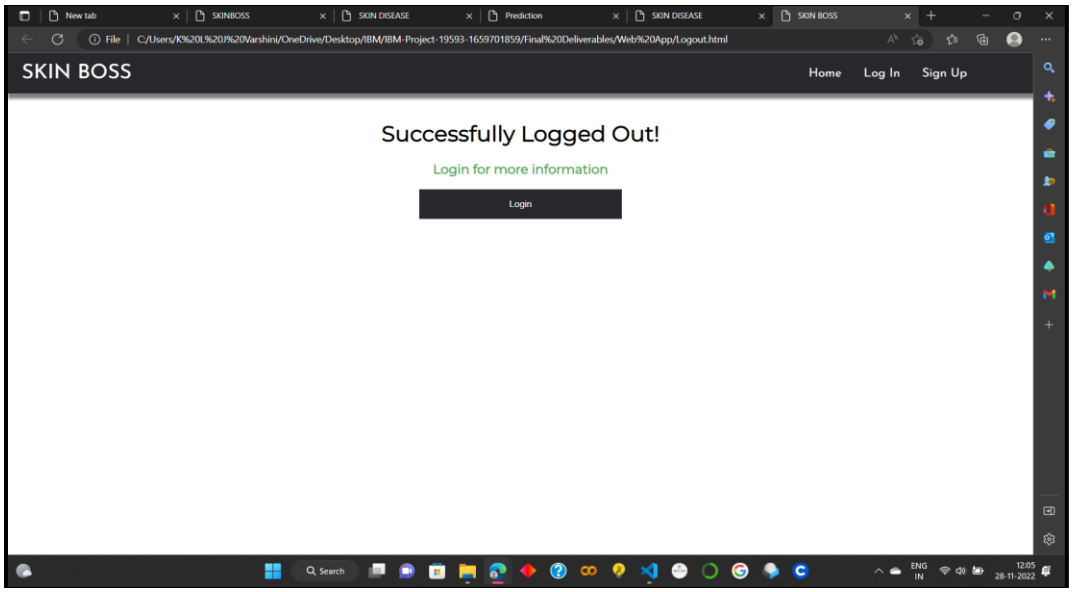


9.2 Web App Screenshots


New tab
SKINBOSS
SKIN DISEASE
File | C:/Users/K%20L%20/P%20Varshini/OneDrive/Desktop/IBM/IBM-Project-19593-1659701859/Final%20Deliverables/Web%20App/Login.html
SKIN DISEASE
Home
Sign In
Sign Up
SIGN IN
Enter your registered email and your password.
Enter registered email ID
Enter Password
Sign In
29°C
Cloudy
Search
ENG
IN
12:04
28-11-2022


New tab
SKINBOSS
SKIN DISEASE
Prediction
SKIN DISEASE
File | C:/Users/K%20L%20/P%20Varshini/OneDrive/Desktop/IBM/IBM-Project-19593-1659701859/Final%20Deliverables/Web%20App/Register.html
SKIN DISEASE
Home
Sign In
Sign Up
SIGN IN
Enter your registered email and your password.
Enter registered email ID
Enter Password
Sign In
Search
ENG
IN
12:04
28-11-2022





10. Advantages and Disadvantages

Advantages

- In dermatology, although skin disease is a common disease, one in which early detection and classification is crucial for the successful treatment and recovery of patients, dermatologists perform most noninvasive screening tests only with the naked eye.
- Our Project improves prediction of Skin Disease
- Results in avoidable diagnostic inaccuracies as a result of human error, as the detection of the disease can be easily overlooked.

- Therefore, it would be beneficial to exploit the strengths of CAD using artificial intelligence techniques, in order to improve the accuracy of dermatology diagnosis.

- Implementation of the Structural Co-Occurrence matrices for feature extraction in the skin diseases classification and the preprocessing techniques are handled by using the Median filter, this filter helps to remove the salt and pepper noise in the image processing; thus, it enhances the quality of the images

Disadvantages

- An inherent disadvantage of clustering a skin disease is its lack of robustness against noise. Centroid that can generalize a cluster of data can significantly degrade the performance of these algorithms.

- The degradation problem that occurs when CNN models become too large and complex.

- Hence we implement skip-connections in both segmentation and classification models

- A dependent model must be developed for the increased support vector machine's accuracy in classifying skin illnesses, and SCM is used to manage the feature extraction technique.

11. Conclusion

Our Project AI-Based Localization of Skin Disease With Erythema is used to find whether the person is having erythema or not. And our project helps lots of people to find whether their skin disease is erythema or not. Our website shows the accurate result so it helps the user to check their skin Disease. Clinical procedures to detect skin diseases are very expensive and time-consuming. Image processing techniques help to build automated screening system for dermatology at an initial stage.We have shown understanding with a TensorFlow in python code for skin cancer detection gives results even without a large dataset and high-quality images, it is possible to achieve sufficient accuracy rates. In addition, we have shown that current state-of-the-art CNN models can outperform models created by previous research, through proper data pre-processing, self-supervised learning, transfer learning, and special CNN architecture techniques. Furthermore, with accurate segmentation, we gain knowledge of the location of the disease, which is useful in the pre-processing of data used in classification, as it allows the CNN model to focus on the area of interest. Lastly, unlike previous studies, our method provides a solution to classify multiple diseases within a single image. With higher quality and a larger quantity of data, it will be viable to use state-of-the-art models to enable the use of CAD in the field of dermatology.

12. Future Scope

Our Project AI - Based Localization Of Skin Disease With Erythema is to try new algorithms and improve the accuracy of the result. And also developing a Android application is our scope of the project. The TensorFlow python code implementation of the Structural Co-Occurrence matrices for feature extraction in the skin diseases classification and the pre-processing techniques are handled by using the Median filter, this filter helps to remove the salt and pepper noise in the image processing; thus, it enhances the quality of the images, and normally, the skin diseases are considered as the risk factor in all over the world. Our proposed approach provides 97% of the classification of the accuracy results while other existing model such as FFT + SCM gives 80%, SVM + SCM gives 83%, KNN + SCM gives 85%, and SCM + CNN gives 82%. Future work is dependent on the increased support vector machine's accuracy in classifying skin illnesses, and SCM is used to manage the feature extraction technique

13. Appendix

13.1 Source Code Link:

https://codesandbox.io/s/ibm-ai-based-localization-and-classification-of-skin-disease-with-erythema-ppdbd5

13.2 GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-19593-1659701859

13.3 Project Demo Link:

IBM_AI DEMO VIDEO - Made with FlexClip.mp4 - Google Drive