PROJECT REPORT

1.Introduction:

1.1.project overview:

Skin Diseases are a common problem among young adults.There is pacity of data about it among medical students.This study aimed to find out the pattern of various types of skin diseases and to describe their perspective solution for various type of skin disorders.people suffering from Atopic Dermatitis,also called eczema is a chronic,relapsing inflammatory disease that leads to itching and risk for skin infection.
It is the most common skin disease in children about 10% to 20% of children in the United states and Western Europe have atopic dermatitis.If the skin diseases are not treated properly in an earlier stage itself,then it may lead to complication in the body including spreading of the infection from one person to another.

To overcome this skin disease caused by bacteria usually can be cured with antibiotics,though some bacteria have become resistant to the drugs and are harder to kill .Medication or presciription creams can stop most fungal infections and are several ways to treat skin disease

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:

some of the most common skin diseases include Acne,blocked skin follicles that lead to oil,bacteria and dead skin buildup in your pores.Alopecia areata,losing your hair in small patches.Atopic dermatitis,dry,itchy skin leads to swelling,cracking or scaliness.

2.2.References:

1."Skin Disease and Conditions among Students of a Medical College in South India" ' Nitin Joseph ','Ganesh S Kumar',' Maria Nelliyanil' Jan 2014" Indian Dermatology Journal"CAD- Computer Aided Diagnosis Convenient Sampling Method

2."AI Based Localization And Classification Of Skin Disease with Erythema.'" Ha Min Son', 'Wooho Jeon', 'Tai-Myoung Chung' March 5 2021" Nature Publishing Group."CAD- Computer Aided Diagnosis

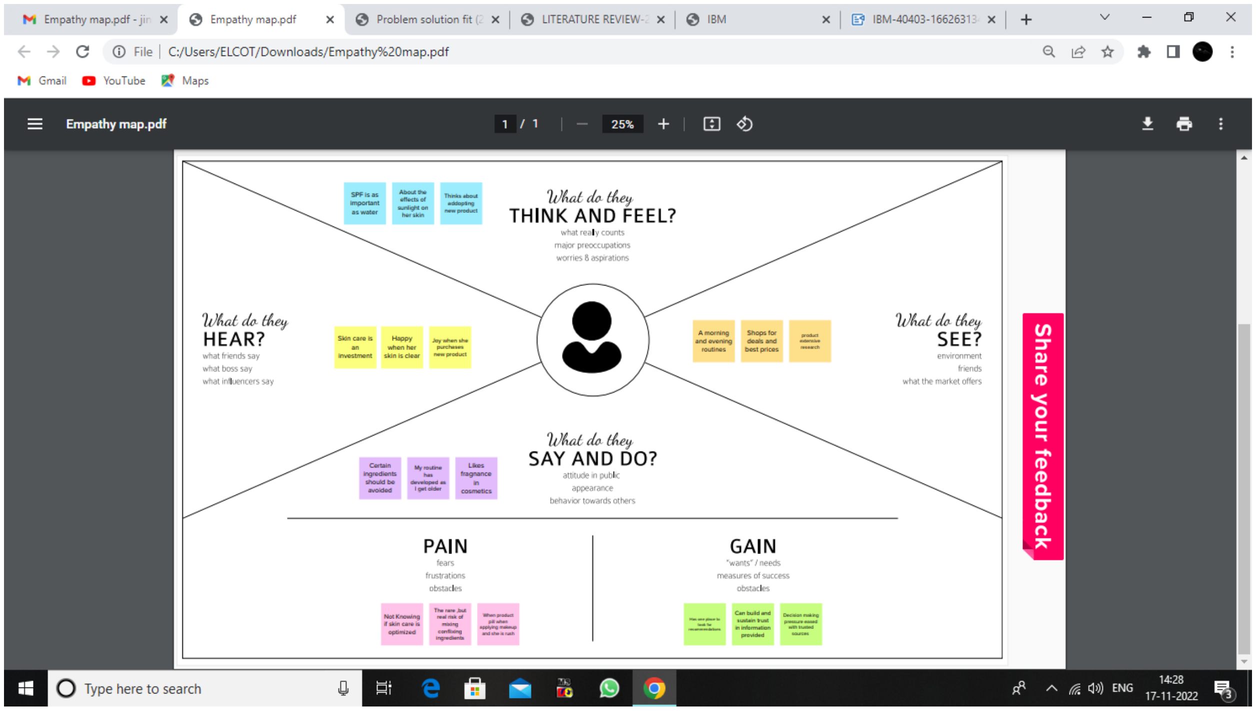
3."Global Burden Of Skin Disease Inequities and Innovations." 'Divya Seth, Ab, Khatiya Cheldize, 'Esther F.Freeman' August 2017 HHS publication Healthy Equity Task Shifting.

2.3.Problem Statement Definition

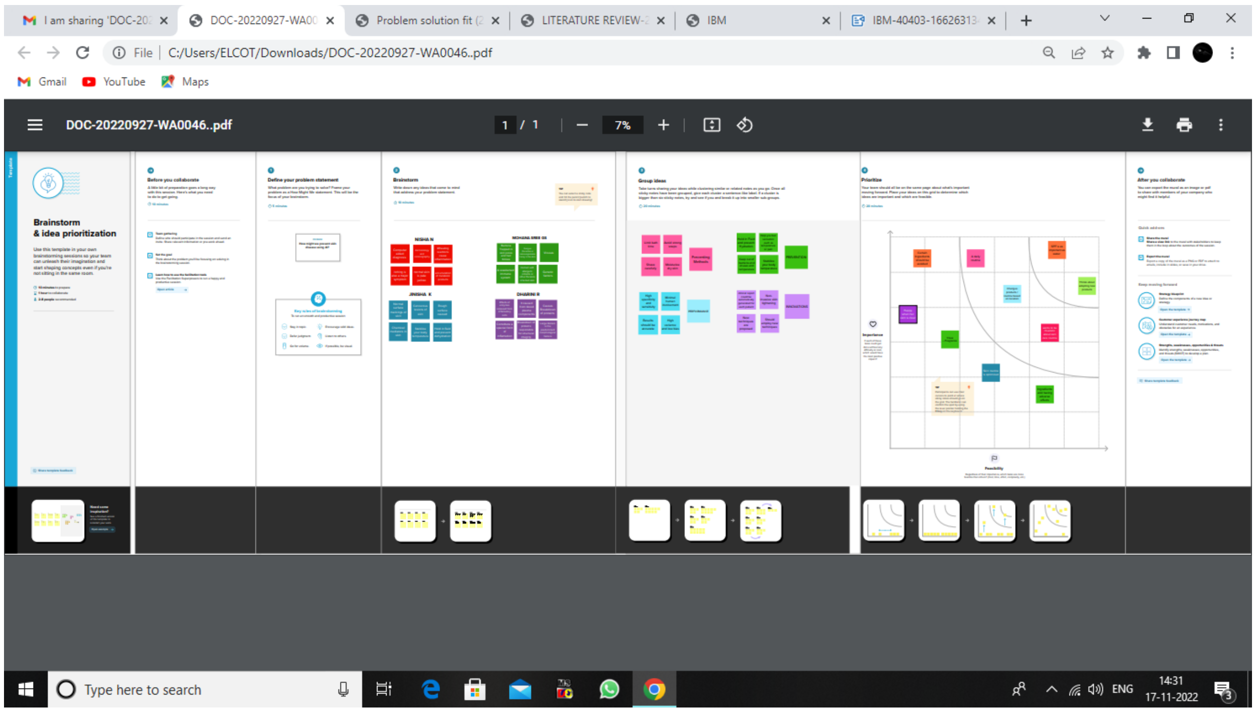
We are trying to find a solution to identify Skin Disease but Developed model is under training because given an image of skin, we can decompose, segment, and classify in a sequential manner which takes to Early detection of skin cancer, psoriasis Ecezma.

3.Ideation and proposed solution:

3.1.Empathy Map Canvas:


Empathy map.pdf
C:/Users/ELCOT/Downloads/Empathy%20map.pdf
1 / 1
25%
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
SPF is as important as water
About the effects of sunlight on her skin
Thinks about addopting new product
What do they HEAR?
what friends say
what boss say
what influencers say
Skin care is an investment
Happy when her skin is clear
Joy when she purchases new product
What do they SEE?
environment
friends
what the market offers
A morning and evening routines
Shops for deals and best prices
product extensive research
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Certain ingredients should be avoided
My routine has developed as I get older
Likes fragnance in cosmetics
PAIN
fears
frustrations
obstacles
Not Knowing if skin care is optimized
The rare ,but real risk of mixing conflixing ingredients
When product pill when applying makeup and she is rush
GAIN
"wants" / needs
measures of success
obstacles
Has one place to look for recommendations
Can build and sustain trust in information provided
Decision making pressure eased with trusted sources
Share your feedback
Type here to search
ENG
14:28
17-11-2022

3.2.Ideation and BrainStroming:



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 present in an image and improved performance by CNN model to focus on particular subsections of the images.

4. Requirement Analysis

4.1 Functional requirements

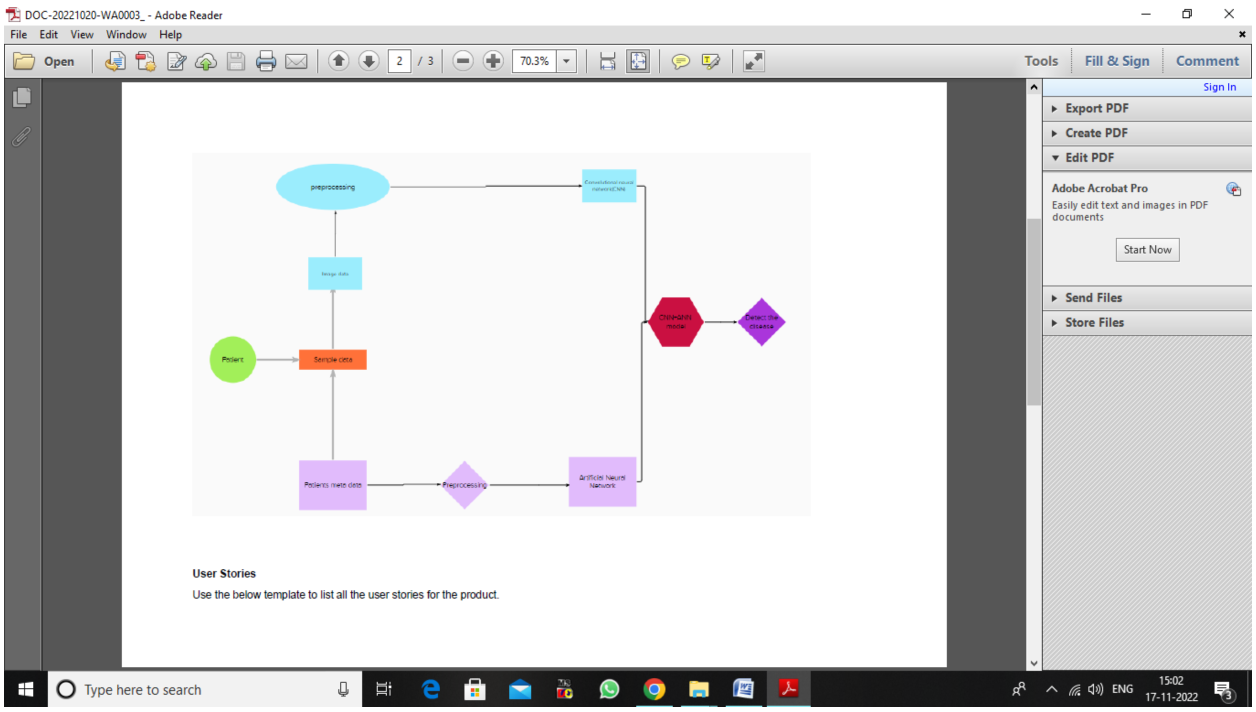
Image Acquisition, Pre-processing Steps such as Colour gradient generator on an image , Cropping and isolating region of interest and Thresholding and Clustering on image, Visual feature extraction, System Training YOLO Model for Skin disease classification with deep learning and CNN, Separate access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

4.2 Non-Functional requirements

Software Quality Attributes, Prediction, Accuracy,usability,security,Reliability,Performance,Availability,Scalability.

5. Project Design

5.1 Data Flow Diagram



User Stories

Use the below template to list all the user stories for the product.

5.2 Solution and Technical Architecture

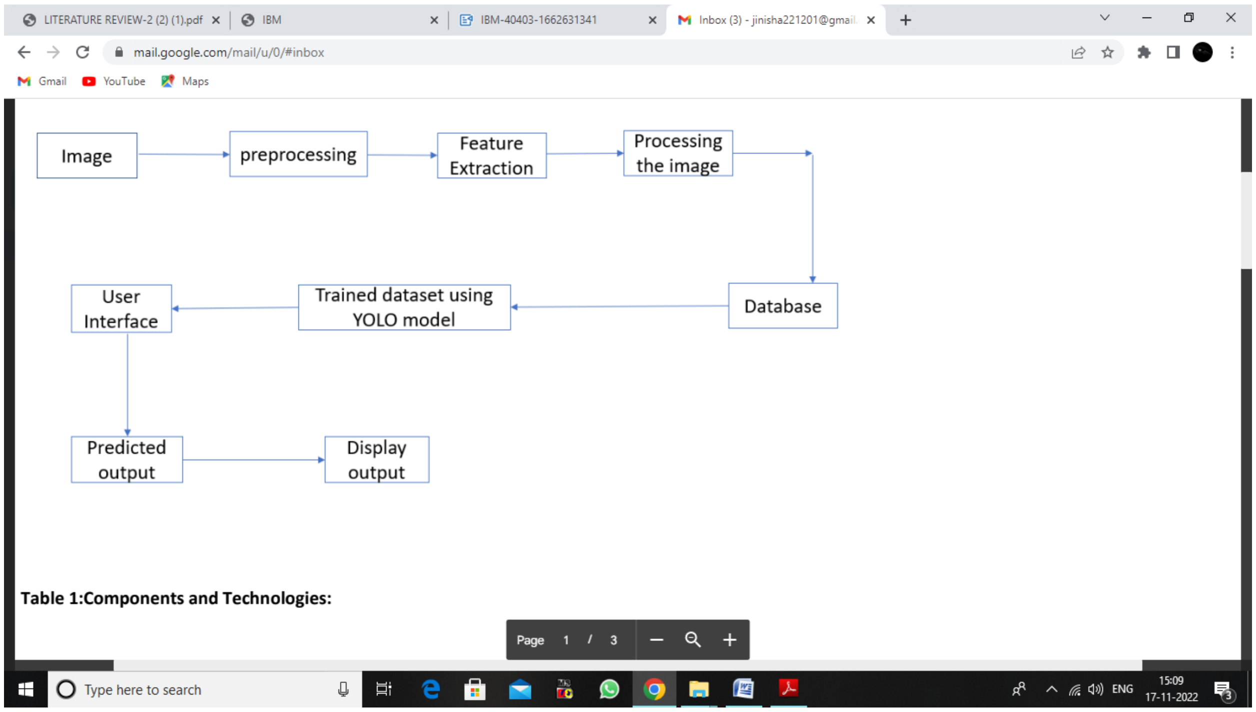
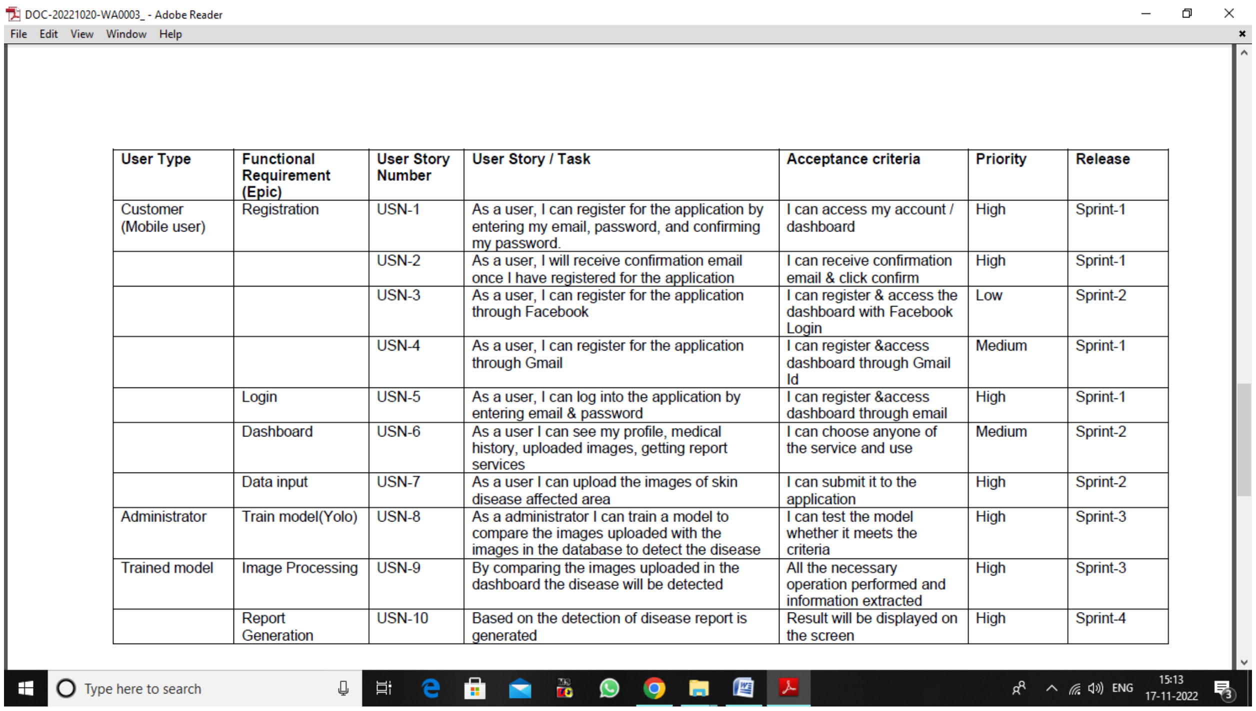


Table 1:Components and Technologies:

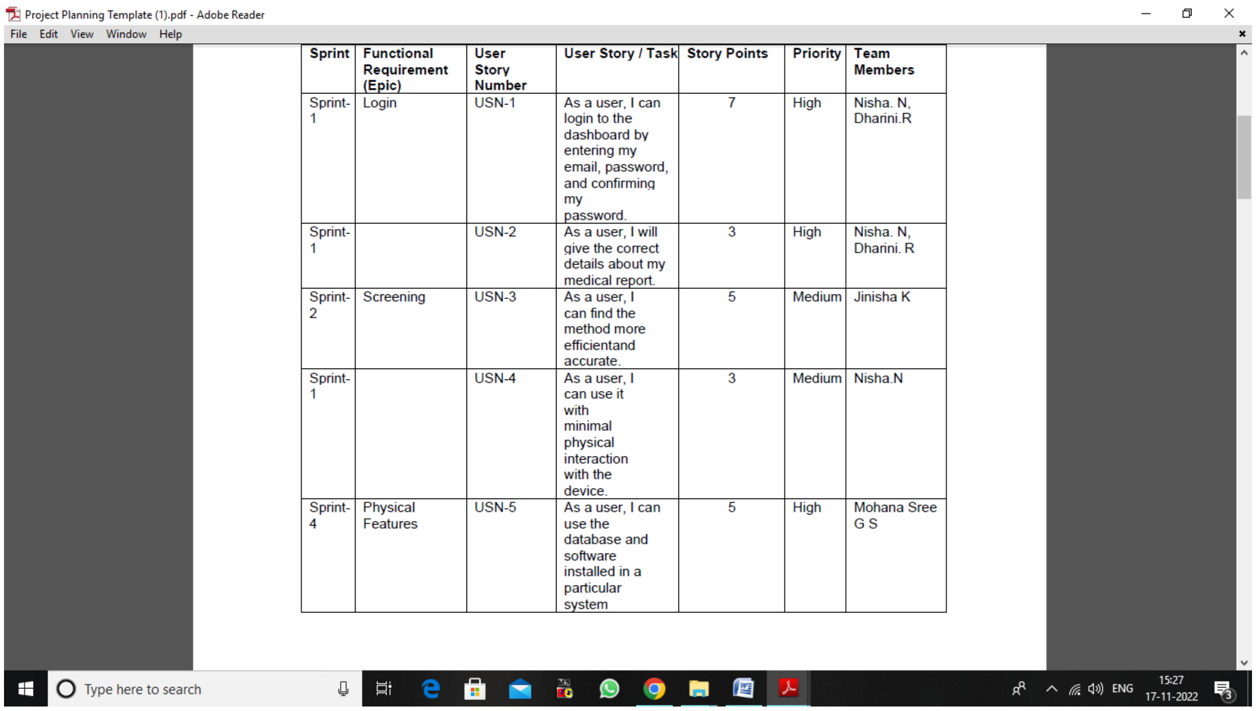
5.3.User Stories:



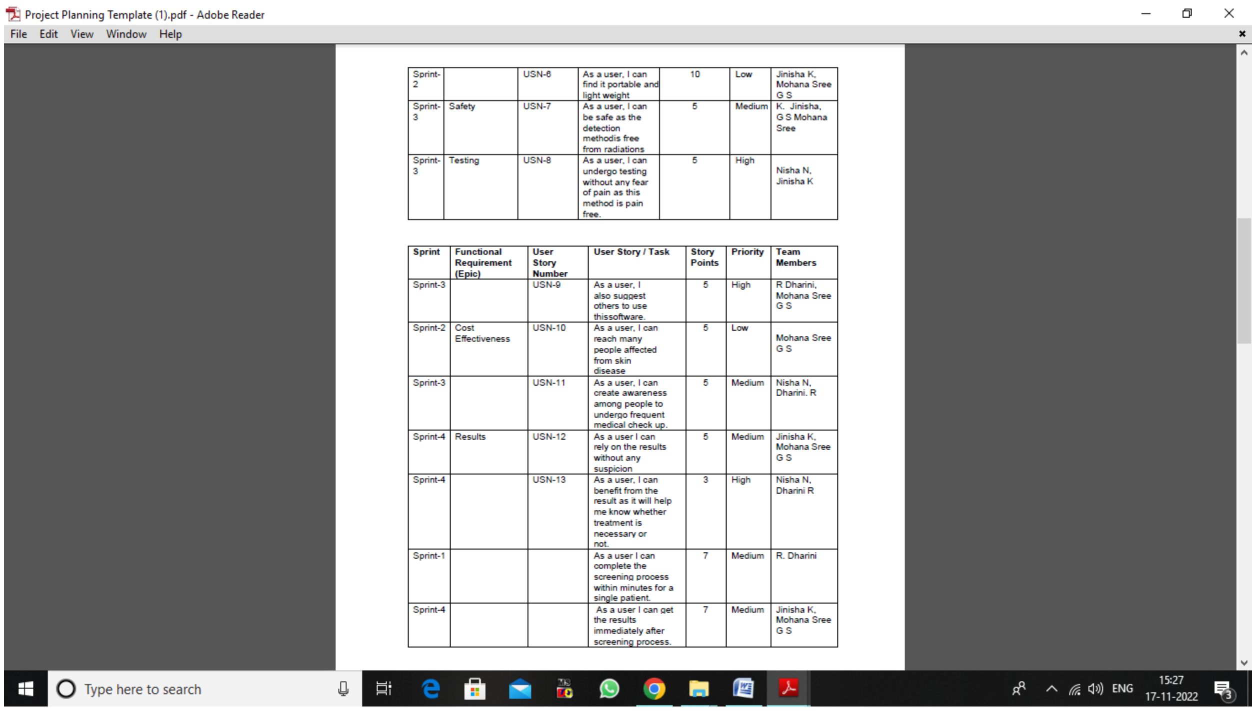
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register &access dashboard through Gmail Id	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register &access dashboard through email	High	Sprint-1
	Dashboard	USN-6	As a user I can see my profile, medical history, uploaded images, getting report services	I can choose anyone of the service and use	Medium	Sprint-2
	Data input	USN-7	As a user I can upload the images of skin disease affected area	I can submit it to the application	High	Sprint-2
Administrator	Train model(Yolo)	USN-8	As a administrator I can train a model to compare the images uploaded with the images in the database to detect the disease	I can test the model whether it meets the criteria	High	Sprint-3
Trained model	Image Processing	USN-9	By comparing the images uploaded in the dashboard the disease will be detected	All the necessary operation performed and information extracted	High	Sprint-3
	Report Generation	USN-10	Based on the detection of disease report is generated	Result will be displayed on the screen	High	Sprint-4

6.Project planning and Scheduling:

6.1.sprint planning and estimation:



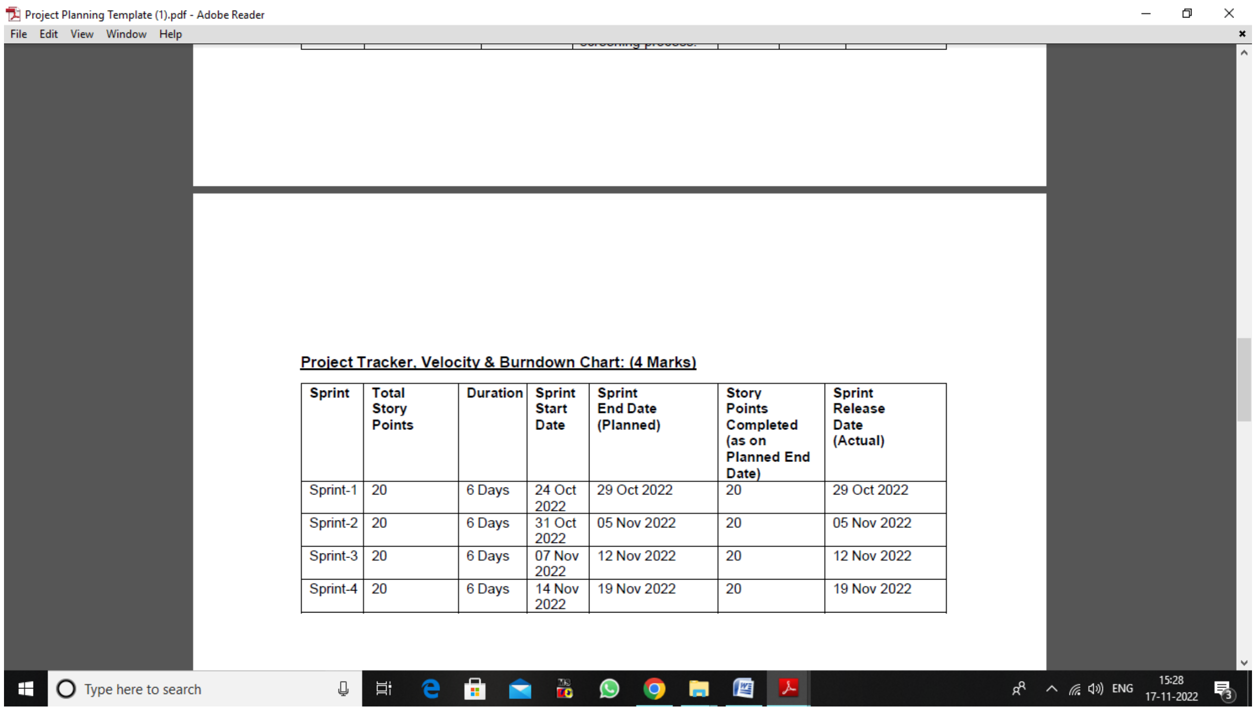
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Login	USN-1	As a user, I can login to the dashboard by entering my email, password, and confirming my password.	7	High	Nisha. N, Dharini.R
Sprint-1		USN-2	As a user, I will give the correct details about my medical report.	3	High	Nisha. N, Dharini. R
Sprint-2	Screening	USN-3	As a user, I can find the method more efficientand accurate.	5	Medium	Jinisha K
Sprint-1		USN-4	As a user, I can use it with minimal physical interaction with the device.	3	Medium	Nisha.N
Sprint-4	Physical Features	USN-5	As a user, I can use the database and software installed in a particular system	5	High	Mohana Sree G S



Sprint-2		USN-6	As a user, I can find it portable and light weight	10	Low	Jinisha K, Mohana Sree G S
Sprint-3	Safety	USN-7	As a user, I can be safe as the detection methodis free from radiations	5	Medium	K. Jinisha, G S Mohana Sree
Sprint-3	Testing	USN-8	As a user, I can undergo testing without any fear of pain as this method is pain free.	5	High	Nisha N, Jinisha K

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		USN-9	As a user, I also suggest others to use thissoftware.	5	High	R Dharini, Mohana Sree G S
Sprint-2	Cost Effectiveness	USN-10	As a user, I can reach many people affected from skin disease	5	Low	Mohana Sree G S
Sprint-3		USN-11	As a user, I can create awareness among people to undergo frequent medical check up.	5	Medium	Nisha N, Dharini. R
Sprint-4	Results	USN-12	As a user I can rely on the results without any suspicion	5	Medium	Jinisha K, Mohana Sree G S
Sprint-4		USN-13	As a user, I can benefit from the result as it will help me know whether treatment is necessary or not.	3	High	Nisha N, Dharini R
Sprint-1			As a user I can complete the screening process within minutes for a single patient.	7	Medium	R. Dharini
Sprint-4			As a user I can get the results immediately after screening process.	7	Medium	Jinisha K, Mohana Sree G S

6.2.Sprint delievery schedule:


Project Planning Template (1).pdf - Adobe Reader
File Edit View Window Help
Type here to search
ENG
15:28
17-11-2022

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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

7. Coding and Solutioning

```
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 = [&quot;benign&quot;, &quot;malignant&quot;]
```

Preparing the Dataset

```
def download_and_extract_dataset():
# dataset from https://github.com/udacity/dermatologist-ai
# 5.3GB
train_url = &quot;https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-
cancer/train.zip&quot;
# 824.5MB
valid_url = &quot;https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-
cancer/valid.zip&quot;
# 5.1GB test_url = &quot;https://s3-us-west-1.amazonaws.com/udacity-
dlnfd/datasets/skin-
cancer/test.zip&quot; for i, download_link in enumerate([valid_url, train_url, test_url]):
temp_file =
f&quot;temp{i}.zip&quot;
 data_dir = get_file(origin=download_link, fname=os.path.join(os.getcwd(), temp_file))
print(&quot;Extracting&quot;, download_link) with zipfile.ZipFile(data_dir, &quot;r&quot;)
as z:
z.extractall(&quot;data&quot;) # 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=[&quot;filepath&quot;, &quot;label&quot;]) i = 0 for
label in labels: print(&quot;Reading&quot;, os.path.join(folder, label, &quot;*&quot;))
for filepath in glob.glob(os.path.join(folder, label, &quot;*&quot;)):
                df.loc[i] = [filepath, label2int[label]]
i += 1
     output_file = f&quot;{folder_name}.csv&quot; print(&quot;Saving&quot;,
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 &quot;data&quot; path to your extracted dataset path # don&#39;t
replace if you
used download_and_extract_dataset() function generate_csv(&quot;data/train&quot;,
{&quot;nevus&quot;:
0, &quot;seborrheic_keratosis&quot;: 0, &quot;melanoma&quot;: 1})
generate_csv(&quot;data/valid&quot;, {&quot;nevus&quot;: 0,
&quot;seborrheic_keratosis&quot;: 0, &quot;melanoma&quot;: 1})
generate_csv(&quot;data/test&quot;, {&quot;nevus&quot;: 0,
&quot;seborrheic_keratosis&quot;: 0, &quot;melanoma&quot;: 1})
# loading data
train_metadata_filename = &quot;train.csv&quot; valid_metadata_filename =
&quot;valid.csv&quot; # 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(&quot;Number of training samples:&quot;,
n_training_samples) print(&quot;Number of validation samples:&quot;,
n_validation_samples)
train_ds = tf.data.Dataset.from_tensor_slices((df_train[&quot;filepath&quot;],
df_train[&quot;label&quot;]))
valid_ds = tf.data.Dataset.from_tensor_slices((df_valid[&quot;filepath&quot;],
df_valid[&quot;label&quot;]))
```

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])
```

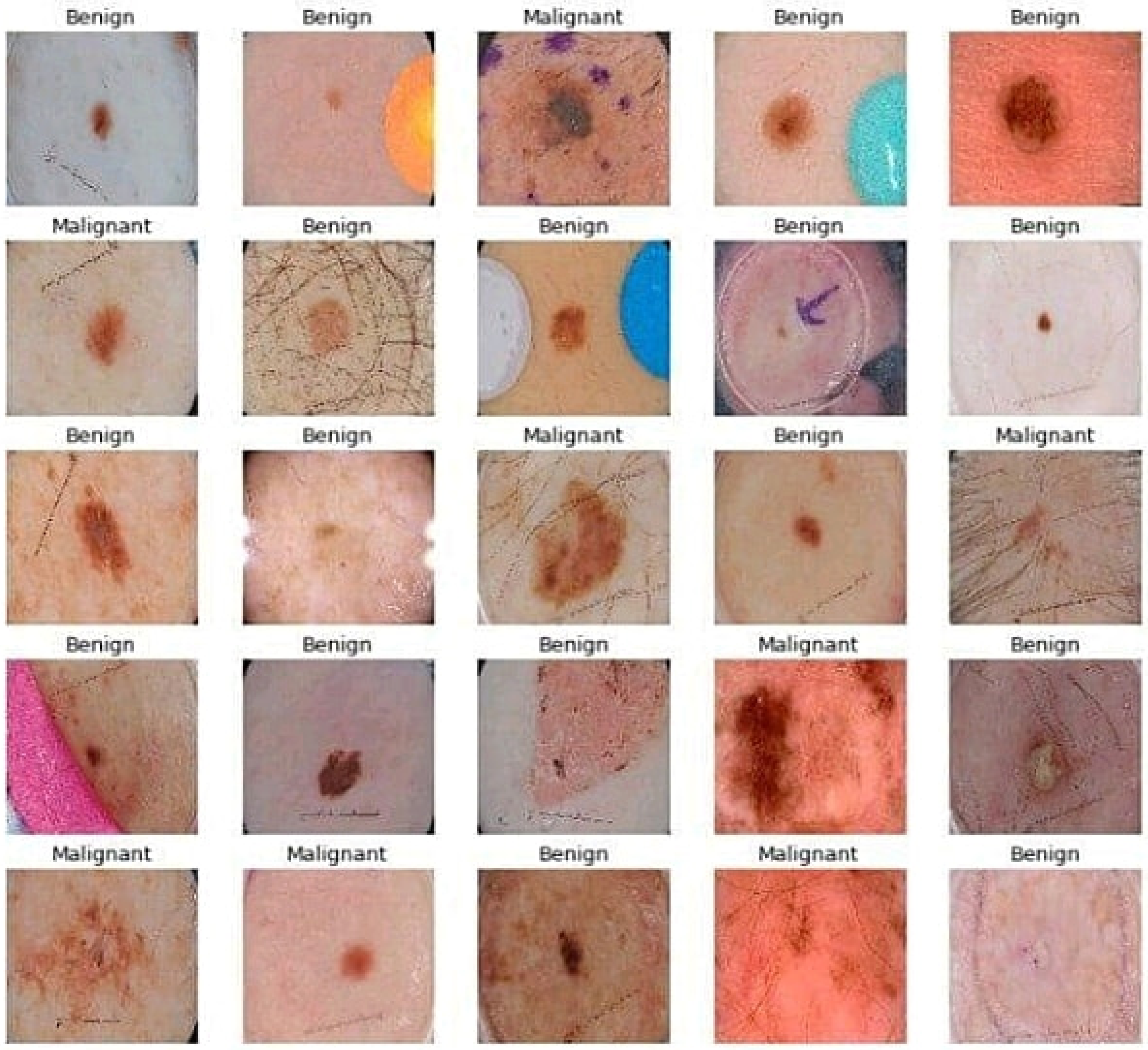
```
def process_path(filepath, label): # load the
raw data from the file as a string img =
tf.io.read_file(filepath) img = decode_img(img)
return img, label

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(&quot;Image
shape:&quot;, image.shape) print(&quot;Label:&quot;,
label.numpy())
Image shape: (299, 299, 3)
Label: 0
# training parameters
batch_size = 64 optimizer =
&quot;rmsprop&quot; 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=&quot;valid-
cached-data&quot;) train_ds =
prepare_for_training(train_ds, batch_size=batch_size, cache=&quot;train-cached-
data&quot;) 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(&#39;off&#39;)
show_batch(batch)
```

Output:



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-vsmalignant_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-vsmalignant_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_tes
ting_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"]))defprepare_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"]))
defprepare_for_testing(ds, cache=True,
shuffle_buffer_size=1000):  if cache:
ifisinstance(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
```

```
defget_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)
defplot_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:

```
# 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()
```

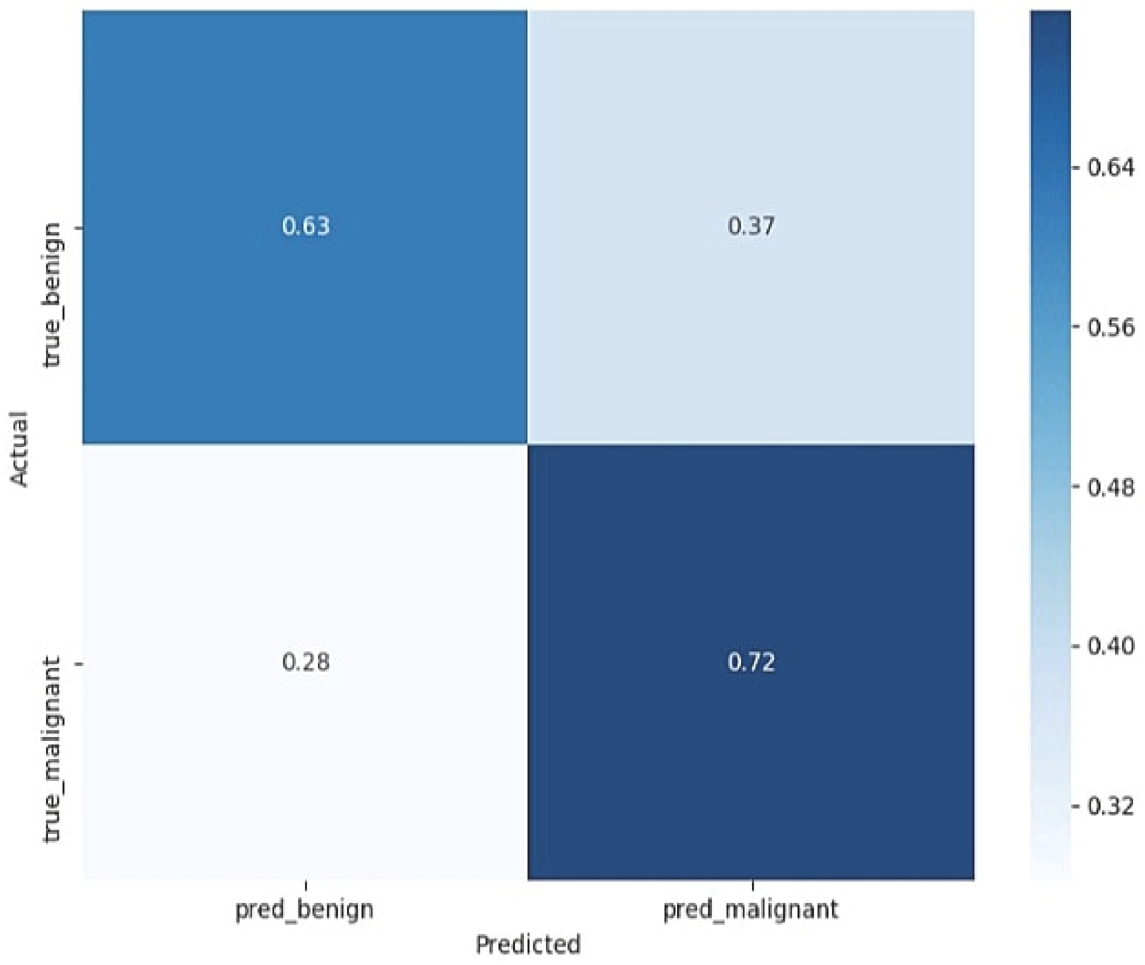
Output:

```
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])
```



```
 sensitivity = sensitivity_score(y_test, y_pred)
specificity = specificity_score(y_test, y_pred)
print("Melanoma Sensitivity:", sensitivity)
print("Melanoma Specificity:", specificity)
```

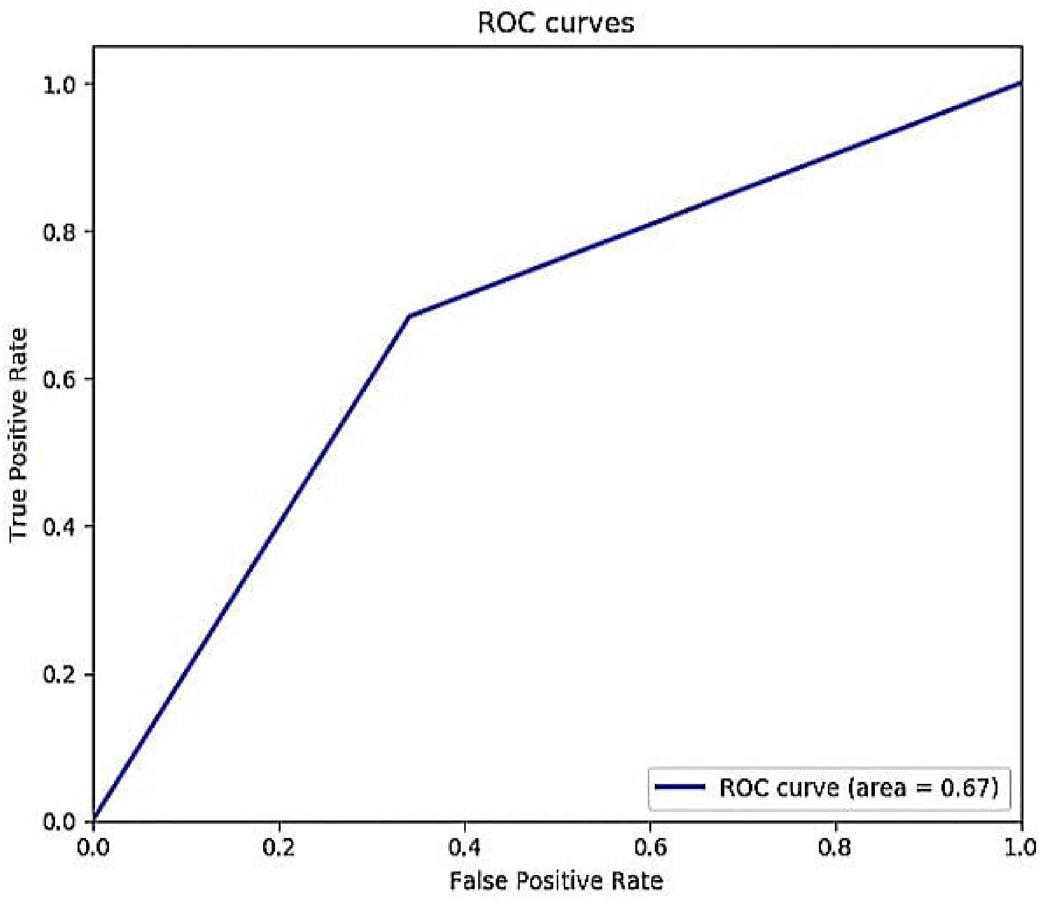
Output:

Melanoma Sensitivity: 0.717948717948718
Melanoma Specificity: 0.6252587991718427

```
defplot_roc_auc(y_true, y_pred):
    """
    This function plots the ROC curves and provides the scores.
    """
    # prepare for figureplt.figure()
fpr, tpr, _ = roc_curve(y_true, y_pred)
    # obtain ROC AUCroc_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)
```

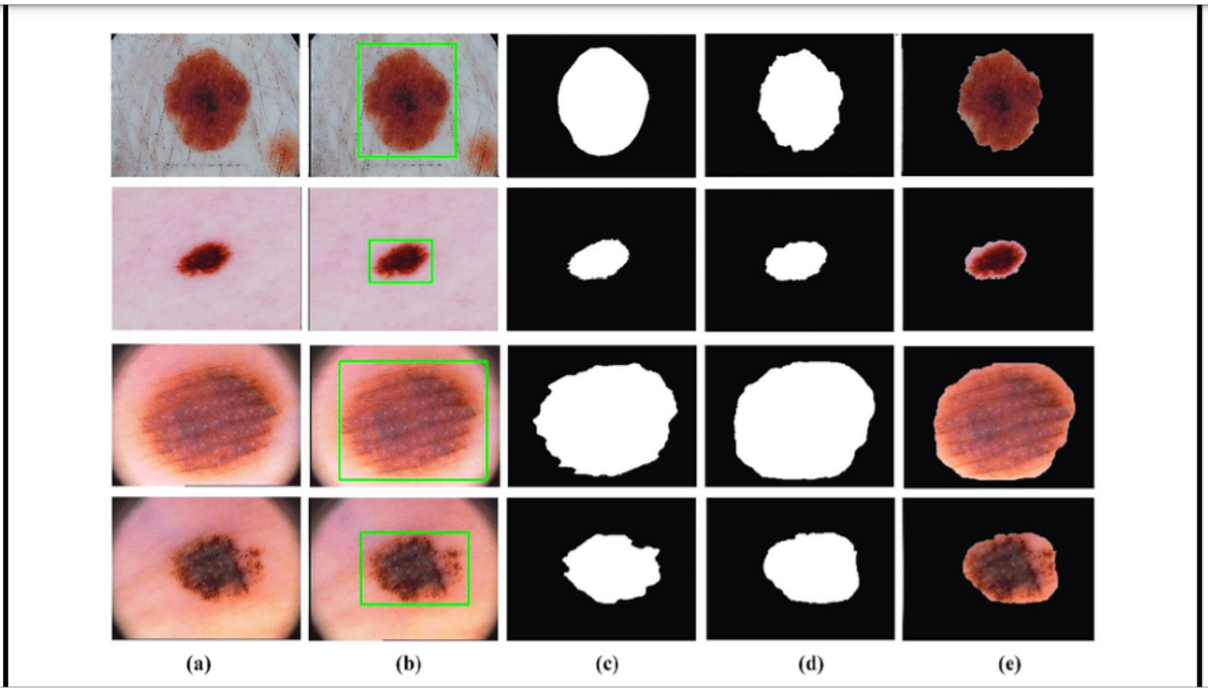
Output:



ROC AUC:0.671

8. Result:

The final results are based on the accuracy results in the form of melanoma and the non-melanoma skin diseases classification.



9.Advantages and Disadvantages:

9.1.Advantages:

High Enhancement,No therapeutic limitation,Easily incorporate depot,Delivery not majorly affected by diseased state of skin.

9.2.Disadvantages:

High cost,Security concern

10.Conclusion:

Skin diseases are a bit like the common cold.They vary enormously from mild conditions which may affect only the appearance of the skin to severe disease which are totally incapacitating.The degree of treatment required or even sought varies accordinglyIn 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.

11.Future Scope:

In future, this machine learning model may bind with a variouswebsite which can provide real-time data for skin disease prediction. Also, we may add large historical data on skin diseasewhich can help to improve theaccuracy of the machine learning model. We can build an android app as a userinterface for interactingwith the user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates, and train it on clusters of data ratherthan the whole dataset.

12.Appendix:

Github link:

IBM-Project-40403-1660629006

Team ID:PNT2022TMID34161