PROJECT REPORT

1.INTRODUCTION

1.1 Project Overview

Now a day's people are suffering from skin diseases, morethan 125 million peoples suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melonoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage,then it will 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 send to the trained model. The model analyses the image and detect whether the person is having skin disease ornot.

1.2 Purpose

The diseases are not considered skin diseases and skin tone is majorly suffered from the ultraviolet rays from the sun.However, dermatologisits performs the majority of non-invasive screening testes simply with the naked eye,even the skin illness is the frequent disease for which early detection and classification are essential for patient success and recovery. The characteristics if the skin images is diversified so that it is the challenging task to device an efficient and robust algorithm for automatic detection of skin disease and iths severity. Automatic processing of such images for skin analysis requires quantitative discriminators 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,social life of patients and their heavy economic burden, the public health importance of these disases is under appeciated.

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,Newyork, 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 nural network,"IEEE Access,vol 9,pp. 143481-143494, 2021.

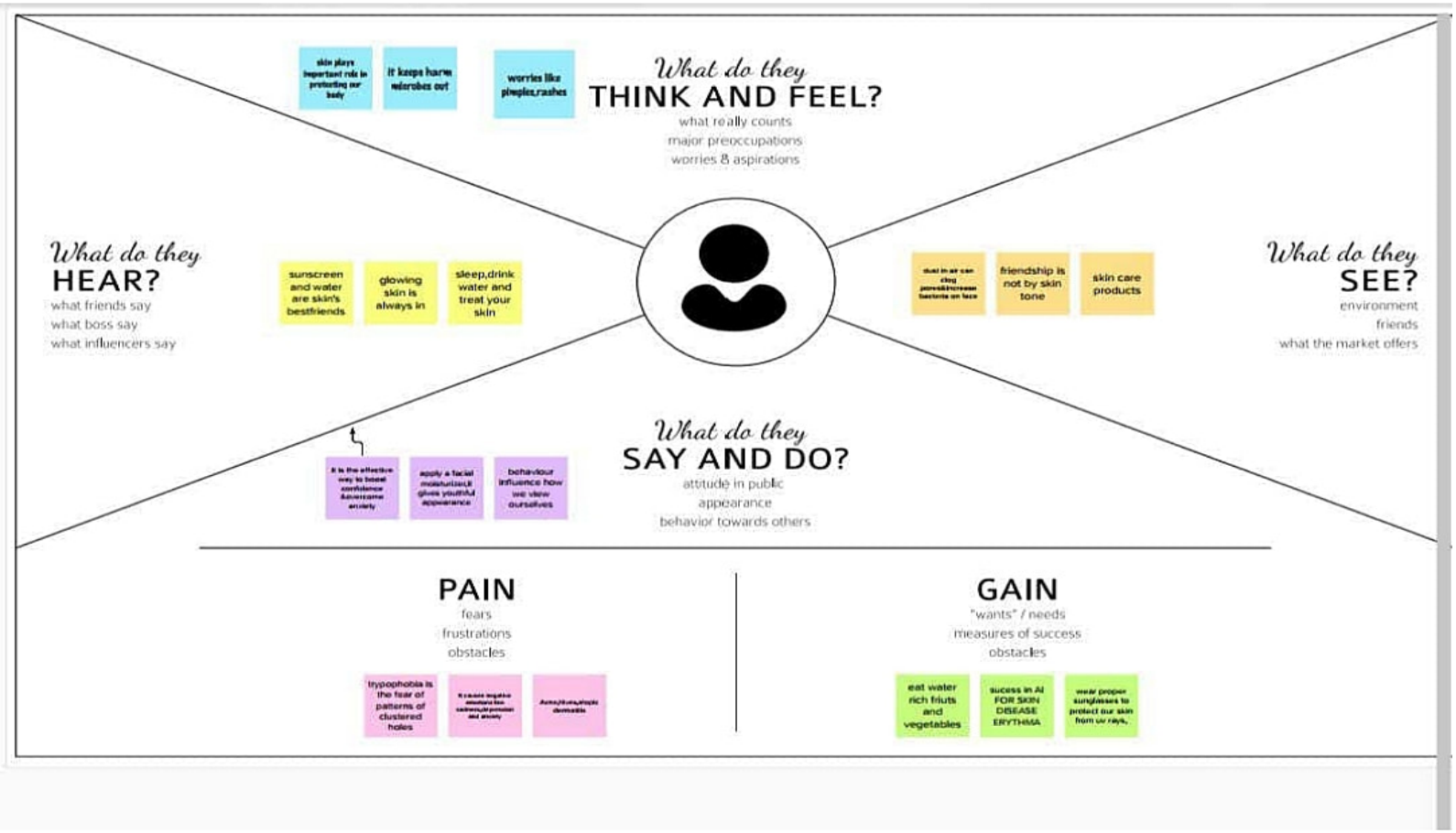
[3] P. P. Repoucas Filho, S. A. Peixoto, R. V. Mederos da Nobrega et al.,"Automatic Histologically-closer classification of lesions, Computarized Medical Imaging and Graphics, vol.68,pp. 40-54, 2018.

2.3 Problem Statement Definition

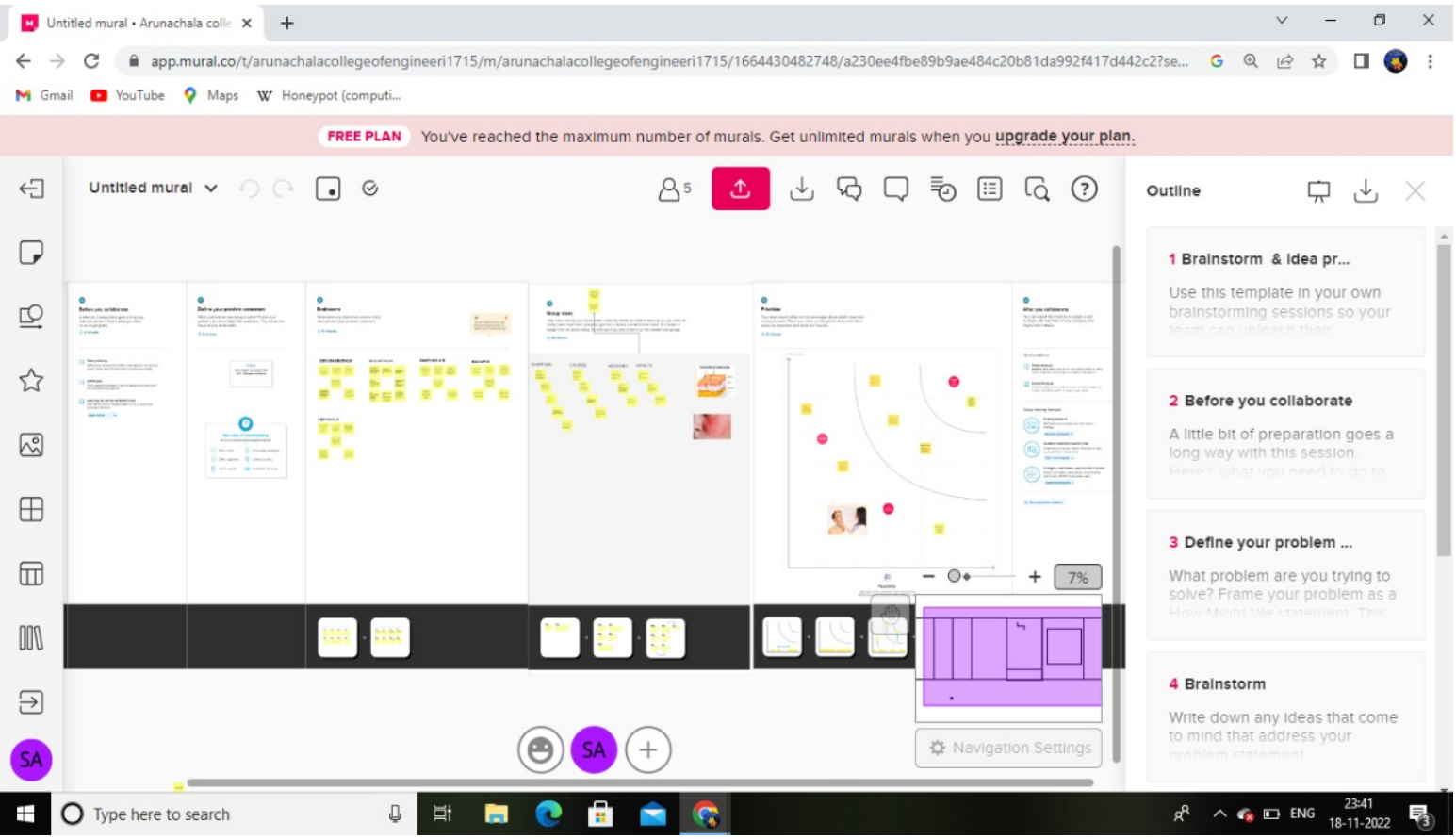
We are trying to find a solution to identify skin disease but developed model is under training because given an image of skin, we can decompose, segment and classify in a sequential manner which takes to Earlr detection of skin cancer, Psoriasis.

3.IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas



3.2 Ideation and Brainstroming



3.3 Proposed Solution

Two Phase analysis model. The original image primarily enters a preprocessing 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 in to its corresponding class. Developed Model is still under training.

3.4 Problem Solution Fit

Skin disease can appear in virtually any part of the 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 Requiements

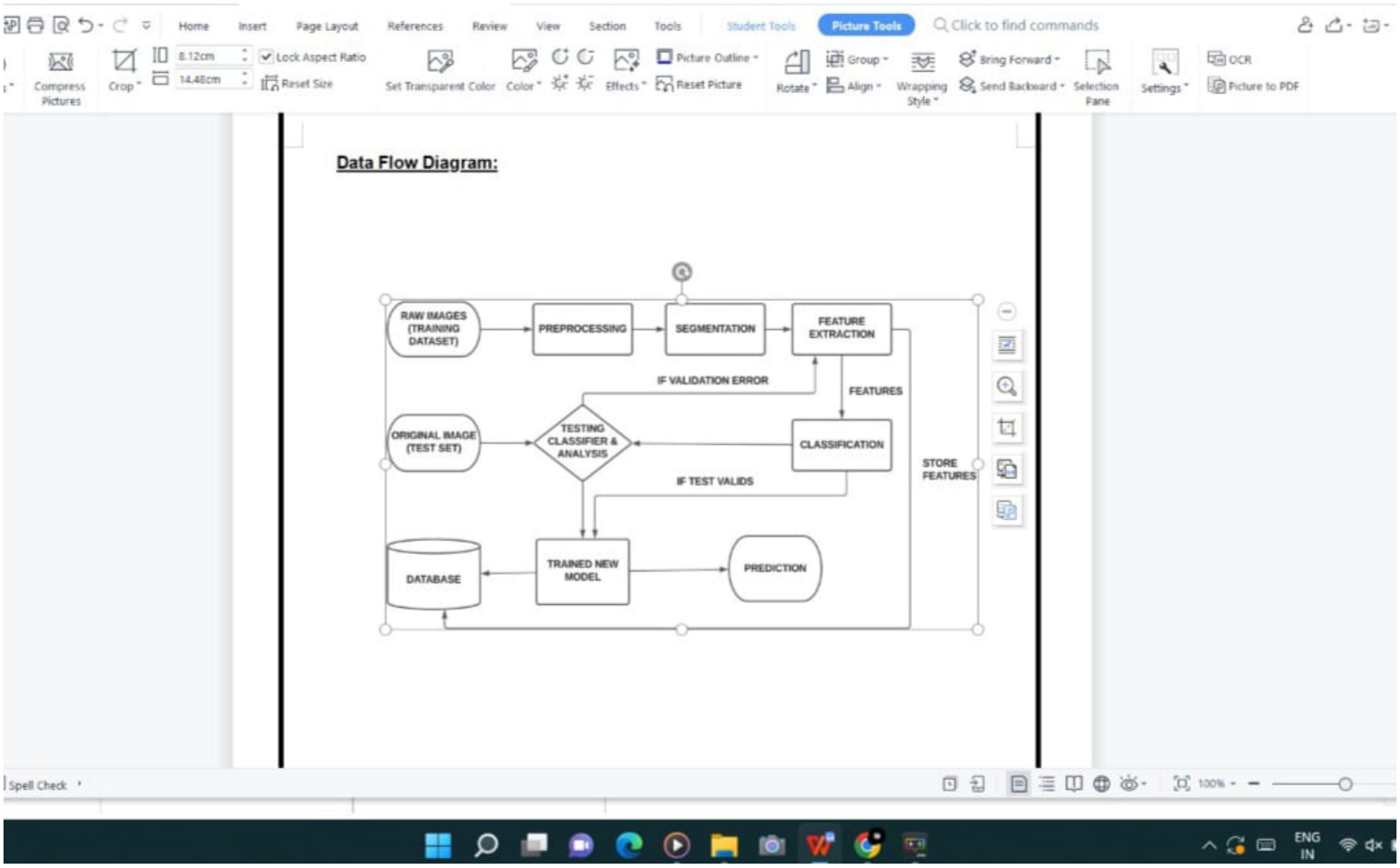
Image Acquistion, Preprocessing steps such as colour gradient generator on an image, Cropping and isolation region of interest and Thersholding and Clustering on image, Visual feature extraction, System Traning YOLO Model for skin disease classification with deep learning and CNN, seperate acces of application for admin, diagnosis of skin disease and data retrival and data manipulation.

4. 2.Non Functional Requirements

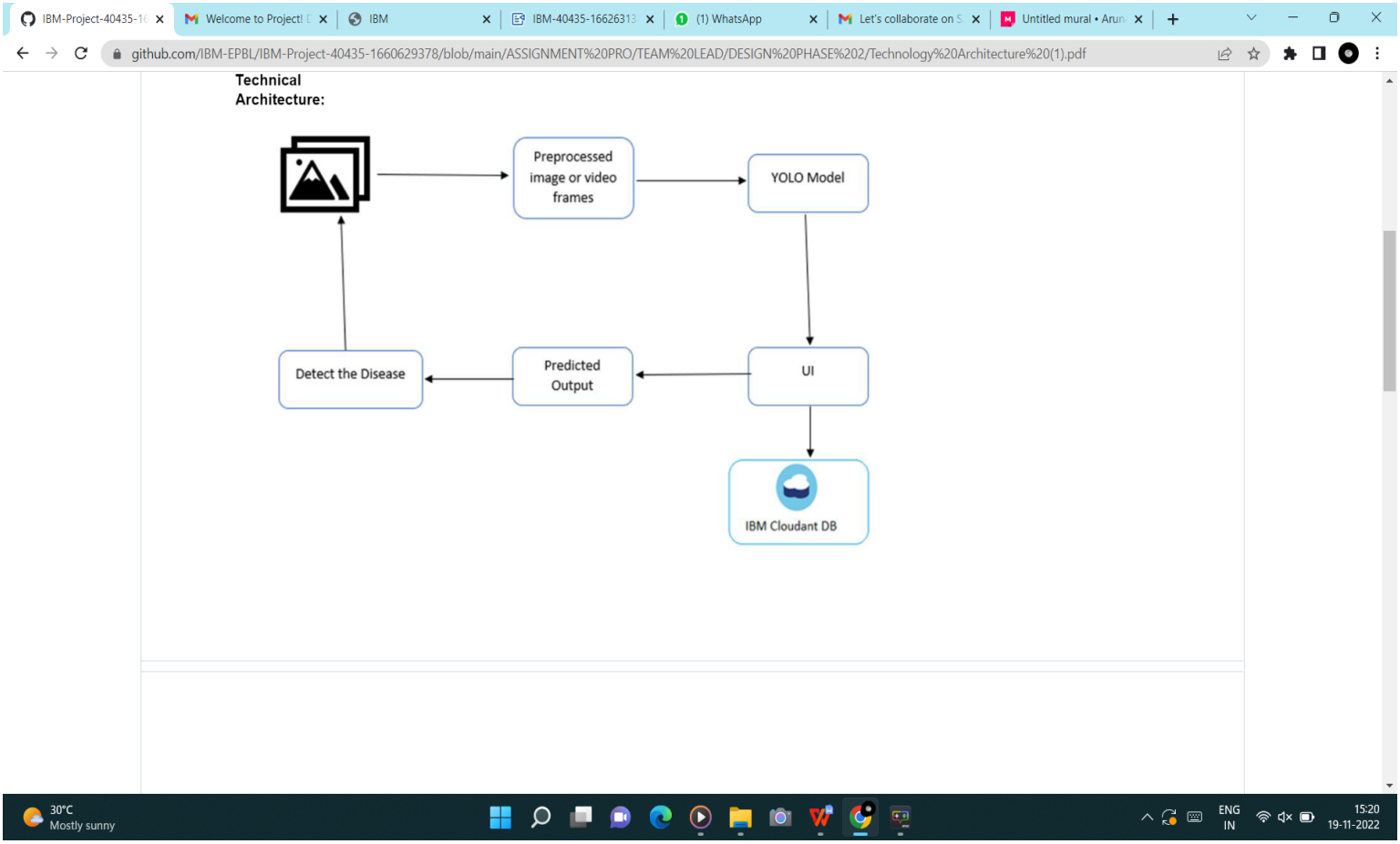
Software Quality attributes, Prediction, Accuracy.

5.PROJECT DESIGN

5.1.Data Flow Diagram



5.2.Solution and Technical Architecture



5.3.User Stories

Functional Requirements (EPIC)	USER Story Number	User Story/Task	Story Points	Priority
Prerequisites	USN-11	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	2	Medium

		Object Tagging Tool)		
Training YOLO	USN-4	Train our model using YOLO weights	2	Medium
	USN-5	To download and convert Pretrained Weights	3	High
	USN-6	To train YOLOv3 Detector	3	
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 images and given the neccessary details.	3	High

6.PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirements (EPIC)	User story number	User Story/Task	Story points	Priority	Team Members
Sprint -1	Prerequisites	USN-1	Install Python IDE,Microsoft Visual Object Tagging tool,YOLO Structure	3	High	J.S.Keerthana K.B.Pavithra
Sprint -1	Data Collection	USN-2	Data set should be collected from googleor using chrome extension such as Fatkun Batch Downloader	3	High	S.P.Bala Ajitha S.P.Bala Abitha

Sprint-1	Annotate Images	USN-3	Create A project in VOTT	2	Medium	M.Devi Bharathi
Sprint-2	Training YOLO	USN-4	Train our model using YOLO Weights	2	Medium	J.S.Keerthana
Sprint-2		USN-5	To Download and Convert PreTrained Weights	3	High	S.P.Bala Ajitha
Sprint-2		USN-6	To Train YOLO V3 Detector	3	High	S.P.Bala Abitha
Sprint-3	Cloudant DB	USN-7	Register and Login to IBM Cloud	3	High	K.B.Pavithra
Sprint-3		USN-8	Create Service Instant and credentials	2	Medium	M.Devi Bharathi
Sprint-3		USN-9	Launch DB and Create database	3	High	J.S.Keerthana
Sprint-3	Development Phase	USN-10	To build a web application	3	Medium	K.B.Pavithra
Sprint-3		USN-11	Building HTML pages with python code	2	Medium	S.P.Bala Ajitha
Sprint-3		USN-12	To run the application	3	High	M.DeviBharathi
Sprint-4	Testing Phase	USN-13	As a user login to dashboard	2	Medium	S.P.Bala Abitha
Sprint-4		USN-14	As a user import the images with skin diseases to the software application	2	Medium	M.DeviBharathi
Sprint-4		USN-15	YOLO processes the image and give the necessary details	3	High	K.B.Pavithra

6.2.Sprint Delivery Schedule

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 tensor flow_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.speed(7)
random.seed(7)
```

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

Preparing the Dataset

```
    def download_and_extract_dataset():
        #datasetfrom https://github.com/udacity/dermatologist-ai
        #5.3GB
        train url="https://s3-us-westt-1.amazonaws.com/udacity-dlnfd/datasets/skincancer/train.zip"
       #824.5MB
       valid-url="https://s3-us-west-1.amazonaws.com/udacity-dflnd/datasets/skincancer/valid.zip"
       #5.1GB    test-url  =  "https://s3-us-west-1.amazonaws.com/udacity-dlfnd/datasets/skin-cancer/test.zip"  for
l,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 downoaded 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)    #generate
CSVfile df  =  pd.DataFrame(coloumns=["file","label,"*"));
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,labelling nevus and seborrheic keratosis
#as 0(beningn), and melonoma 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,"melaanoma":1})
#loading data
train_metadata_filename=&amp;quot;train.csv&amp;quot; valid_metadata_filename=&amp;quot;valid.csv&amp;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(&amp;quot;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 unit8 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("Lable:", lable.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,cache="train-catched-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].numpy].title())
plt.axis('off')
show_batch(batch)
```

Output:

#building the model

#InceptionV3 model & pretrained weights
module_url = "https://tfhub.dev/google/f2-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, metrices=[*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"benings-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.callsbacks.ModelCheckpoint(modelname + "_{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,
callsbacks=[tensorboard,modelcheckpoint])

Output

Train for 31 steps , validate for 2 steps
Epoch 1/100
30/31[==============================>.] - ETA: 9s - loss: 0.4409 - accuracy: 0.7760

Epoch 00001: val_loss improved from inf to 0.49703, saving model to benign-vsmalignant_64_rmprop_0.497.h5
31/31[==========] - 282s 9s/step - loss: 0.4646 - accuracy: 0.7722 - val_loss: 0.4970 - val_accuracy: 0.8125
<..SNIPPED..>
Epoch 27/100
30/31 [==========] - ETA:0s -loss: 0.2982 -accuracy:0.8708Epoch 00027:val_loss improved from 0.40253 to 0.38991,saving model to beningn-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
<..SNIPPED..>
Epoach 41/100
30/31[==========] - ETA:0s - loss:0.2800 - accuracy:0.8802
Epoach 00041:val_loss did not improve from 0.38991
31/31[==========] - 21s 690ms/step - loss:loss:0.2829 - accuracy:0.8790 - val_loss: 0.3948 - val_accuracy :0.8281
Epoach 42/100
30/31[==========>] - ETA:0s -loss:0.2680 - accuracy:0.8859
Epoach 00042: val_loss did not improve from 0.38991

Model Evaluation:

```
# evaluation #
load testing set
test_metadata_filename = "test.csv"
df_test = pd.read_csv(test_metatest_filename)
n_testing_samples = len(df_test)print("Number of
testuing 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 isintance(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_filenname = "test.csv" df_test =
pd.read_csv(test_metadata_filename)
n_testing_samples = len(dftest)
print("Numberof testing samples:",n_testing_samples)test_ds =
tf.data.Data set.from_tensor_slices((df_test[filename],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.csche()
   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-catched-data")


#load the weights with the least loss
m.load_weights("beningn-vs-malignant_64_rmsprop_0.392.h5")
print("Evaluating the model...")
loss,accuracy = m.evaluate(X_test,y,verbose=0)
print("Loss:",   Accuracy:", accuracy)
```

Output

Evaluating the model...

Loss: 0.4476394319534302   Accuracy: 0.8

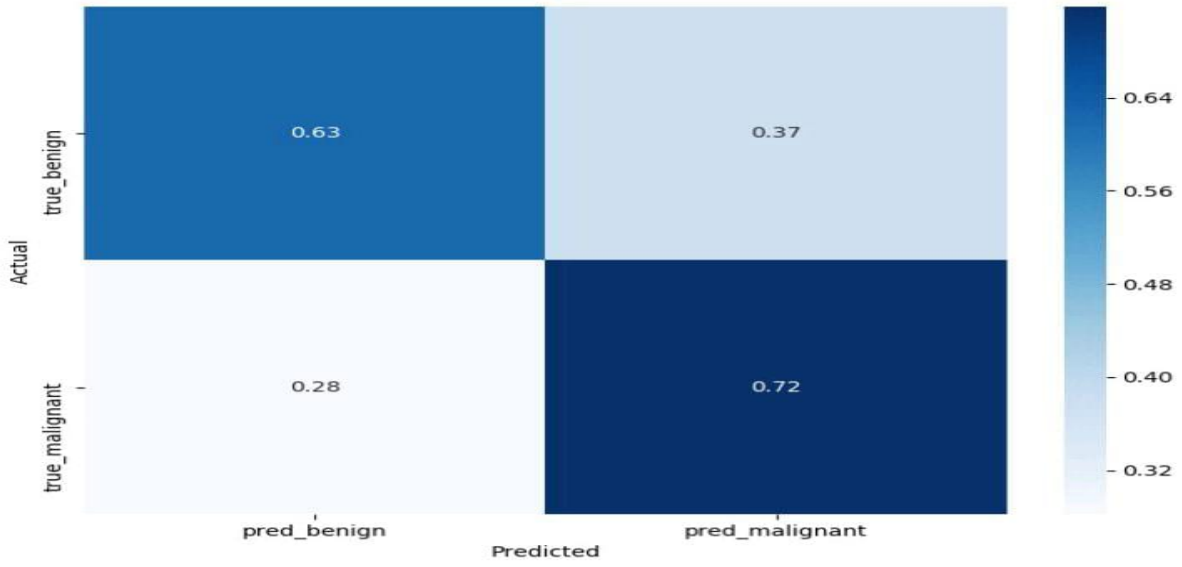
```
def get_predictions(threshold=None):
  """
  Returns predictions for binary classification given 'threshold'
  For instance, if threshold is 0.3, then it'll output 1 (malignant)for that sample if
the probability of 1 is 30% or more (instead of 50%)
   """

   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%sureor more that is mallignant,
#it's assigned as malignant,otherwise it's begin
y_pred = gey_predictions(threshold)def
plot_confusion-matrix(y_test,y_pred):  cmn=
confusion_matrix(y_test,Y_tpred):
     #Normalize
     cmn  = cmn.astype('float') /cmn.sum(axis=1)[:,npnewaxis]
     #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:

Melaloma 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)
```

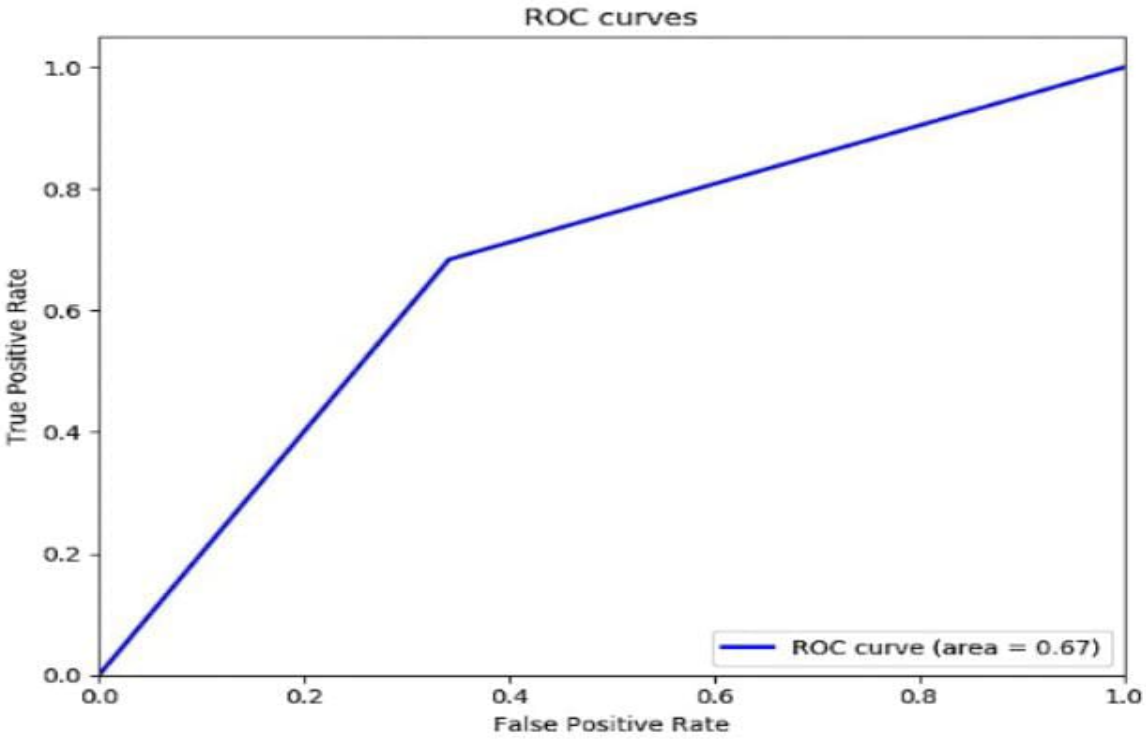
#obtain ROC AUC

```
roc_auc = auc(fpr,tpr)
 #print score
```

```
 print(f"ROC AUC :{roc_auc:.3f}")
 #plot ROC curve
 plt.plott(fpr,tpr,colour="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)
```

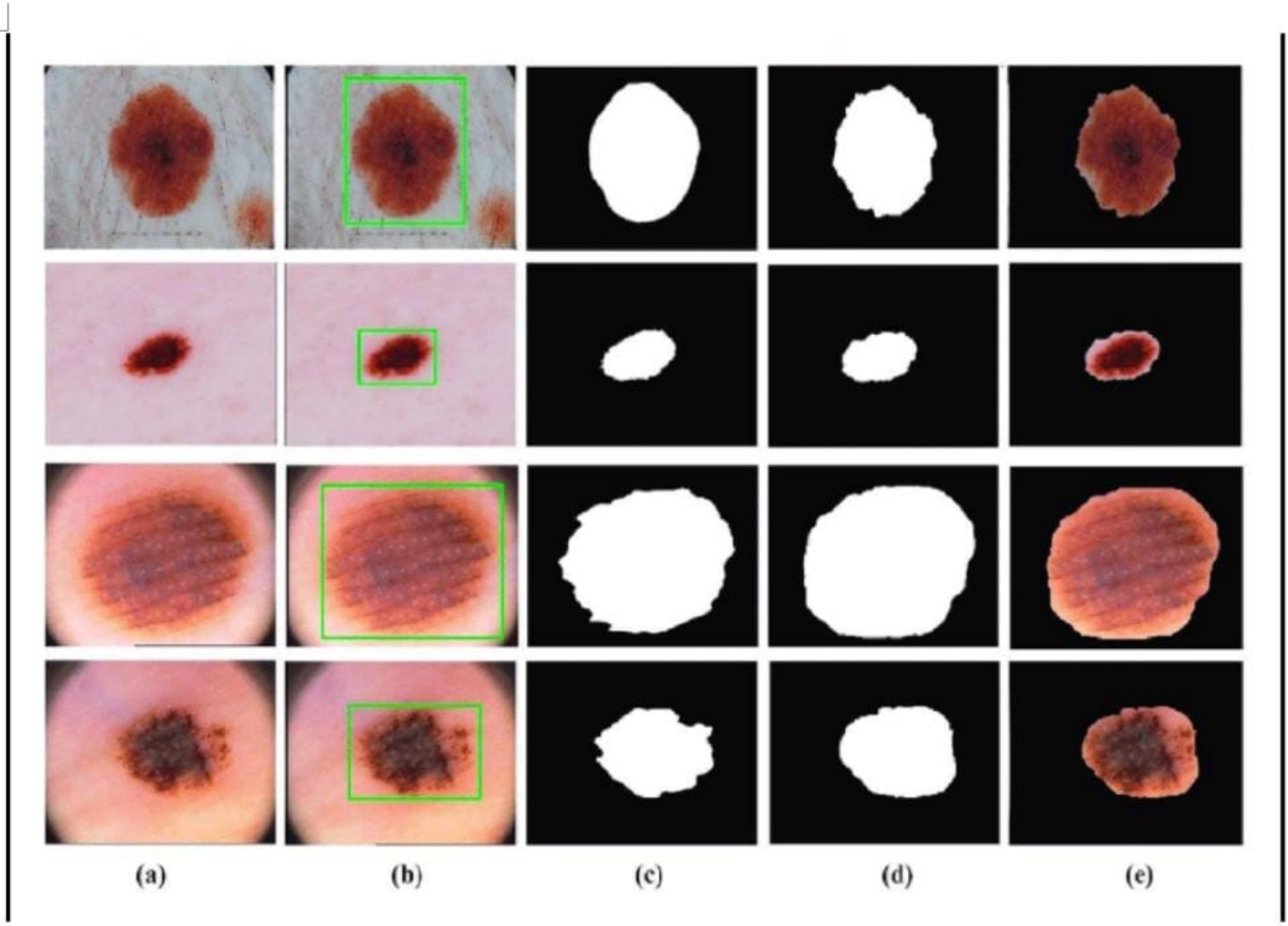
Output



ROC AUC: 0.671

8. RESULTS

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



9.ADVANTAGES AND DISADVANTAGES

9.1 Advantages

Instant Response , improved prediction ofskin disease ,no reference needed ,Saves money and Time and confidential

advice.

9.2 Disadvantages

Network connectivity and accuracy

10.CONCLUSION

We have shown that even without a large dataset and high quality images,it is possible to achieve sufficient accuracy rate. In addition,we have shown that current state-of-the-art CNN model can outperform model created by previous research,through proper data preprocessing, self supervised learning , transfer learning and special CNN architecture techniques . Furthermore, with accurate segmentation we can gain knowledge of the location of the disease , which is useful in the preprocessing of data used in classification,as it allows the CNN model to focus on the area of interests. Lastly, unlike previous studies our method provide 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

This implementation of the structural co occurrence matrices for feature extration 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 disease are considered as the risk factor in all over the world.Our proposed approach provides 97% of the classification of the accuracy results while another existing model such as FFT+SCM gives 80% SVM+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.

12.APPENDIX

Github Link:
https://github.com/IBM-EPBL/IBM-Project-40435-1660629378
Team Id:PNT2022TMID34120