PROJECT REPORT

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 cancerrate is rapidly increasing over the last few decadesespecially 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 diseasediagnosis depends on the different characteristics like color, shape, textureetc. 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 majorlysuffered 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 essentialfor patient successand recovery.The characteristic of the skin images is diversified so thatit is a challenging job to devise an efficient and robust algorithm for automatic detection of skin diseaseand its severity. Automaticprocessing of such images for skin analysis requiresquantitative discriminator to differentiate the diseases.

2.Literature survey

2.1 Existing problem

A neglected publichealth problem Skin diseases are among the most commonhealth problems in humans. Considering their significant impact on the individual, the family, the social life of patients, and their heavy economic 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 pretrainedand skin-lesion trainedlayers,” in International Workshop on MachineLearning 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 diagnosisof Hand, Foot, and mouth disease using hybrid deep neural networks,”IEEE Access, vol. 9, pp. 143481–143494, 2021.

3. P. P. RebouçasFilho, S. A. Peixoto, R. V. Medeirosda Nobrega´ et al., “Automatichistologically-closerclassification of skin lesions,”Computerized Medical Imaging and Graphics,vol. 68, pp. 40–54, 2**018.**

2.3 Problem Statement Definition

We're tryingto find a solution to identify Skin Disease but Developed modelis under trainingbecause given an imageof skin, we can decompose, segment, and classifyin a sequential manner whichtakes to Early detection of skin cancer,psoriasis.

3. Ideation and ProposedSolution

3.1 Empathy Map Canvas

3.2 Ideation and brainstorming

3.3 Proposed Solution

Two-phase analysismodel. The originalimage 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 Stillunder 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 formprobability of a skin disease based on the body part. A Solution model used for the prevention and early psoriasis by image analyses to detect whether the personis having skin disease or not. The location of t image and improvedperformance by CNN model to focuson particular subsections of the images.

4. Requirement Analysis

4.1 Functional requirements

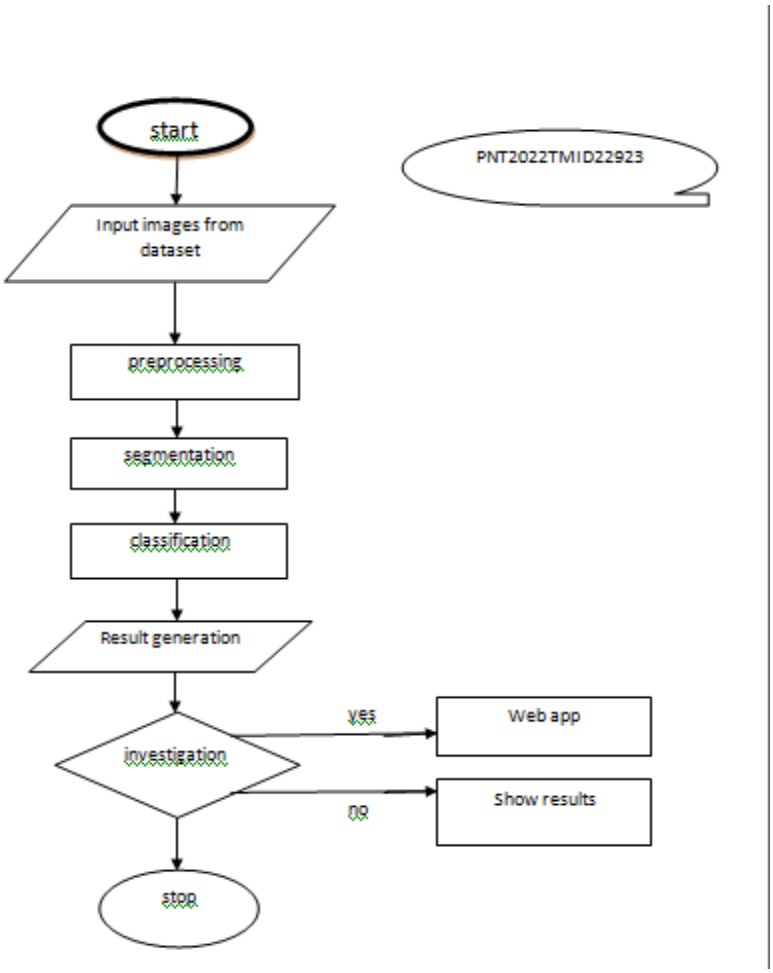
Image Acquisition, Pre-processing Steps such as Colour gradient generator on an image , Cropping and isolating region of interest and Thresholding and Clustering on image, Visual feature extraction, System Training YOLO Model for Skin disease classification with deep learning and CNN, Separate access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

4.2 Non-Functionalrequirements

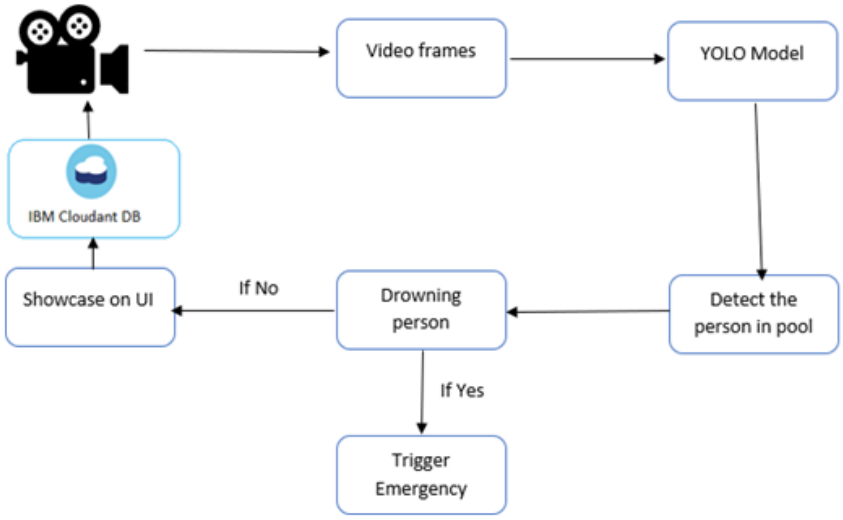
Software Quality Attributes, Prediction, Accuracy.

5. Project Design

5.1 Data Flow Diagram



5.2 Solution and technical architecture


Video frames
YOLO Model
IBM Cloudant DB
Showcase on UI
If No
Drowning person
Detect the person in pool
If Yes
Trigger Emergency

5.3 Userstories

FunctionalRequirement(Epic)	User StoryNumber	User Story/ Task	Story Points	Priority
Prerequisites	USN-1	Install Python IDE, Python packages,Microsoft VisualObject Tagging Tool,YoloStructure	3	High
Data Collection	USN-2	Dataset should be collected fromgoogle or using a Chromeextension suchasFatkun Batch Downloader	3	High
AnnotateImages	USN-3	Create A Projectin VOTT (Microsoft'sVisualObject Tagging Tool)	2	Medium
TrainingYOLO	USN-4	train our modelusing YOLOweights	2	Medium
	USN-5	To Download and Convert Pre-TrainedWeights	3	High
	USN-6	To Train YOLOv3 Detector	3	High
Cloudant DB	USN-7	Register & Loginto IBM Cloud	3	High
	USN-8	Create Service Instant and Credentials	2	Medium
	USN-9	Launch DB and Createdatabase	3	High
DevelopmentPhase	USN-10	To builda web application	3	High
	USN-11	Building HTMLpages with pythoncode	2	Medium
	USN-12	To run the application	3	High
Testing Phase	USN-13	As a user loginto dashboard	2	Medium
	USN-14	As a user importthe images with skindiseases to the software application	2	Medium

	USN-15	YOLO processes the image and give thenecessary 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, YoloStructure	3	High	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-1	Data Collection	USN-2	Dataset should be collected from google or using a Chrome extension such as FatkunBatch Downloader	3	High	DEVISHREE HARINI EZHILR AJBALAJI
Sprint-1	Annotate Images	USN-3	Create A ProjectinVOTT (Microsoft'sVisual Object TaggingTool)	2	Medium	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-2	Training YOLO	USN-4	train our modelusingYOLO weights	2	Medium	DEVISHREE HARINI EZHILRAJ BALAJI

Sprint-2		USN-5	To Download and Convert Pre-Trained Weights	3	High	DEVISHREE HARINI EZHILRAJ BALAJI

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2		USN-6	To Train YOLOv3 Detector	3	High	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-3	Cloudant DB	USN-7	Register &Login to IBM Cloud	3	High	DEVISHREE HARINI EZHILR AJBALAJI
Sprint-3		USN-8	Create Service Instant and Credentials	2	Medium	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-3		USN-9	Launch DB and Createdatabase	3	High	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-3	Development Phase	USN-10	To build a web application	3	High	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-3		USN-11	Building HTML pages with python code	2	Medium	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-3		USN-12	To run the application	3	High	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-	Testing Phase	USN-13	As a user log into dashboard	2	Medium	DEVISHREE HARINI EZHILRAJ

4						BALAJI
Sprint-4		USN-14	As a user import the images with skin diseases tothe software application	2	Medium	DEVISHREE HARINI EZHILRAJ BALAJI
Sprint-4		USN-15	YOLO processes theimageand give the necseeary details	3	High	DEVISHREE HARINI EZHILRAJ BALAJI

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint EndDate (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 Nov2022	12 Nov2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022

7.Coding and Solutioning

pip3 install tensorflow tensorflow_hub matplotlib seaborn numpypandas sklearn imblearn

```
import tensorflow as tf
importtensorflow_hub as hub
import matplotlib.pyplot asplt
import numpy as np
import pandas as pd
import seabornassns
from tensorflow.keras.utils import get_file
```

from sklearn.metrics import roc_curve, auc, confusion_matrix

from imblearn.metrics importsensitivity_score,

specificity_score

import os

import glob import zipfile import random

to get consistent results after

multiplerunstf.random.set_seed(7)

np.random.seed(7)

random.seed(7)

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

Preparingthe Dataset

```
def download_and_extract_dataset():
  # dataset from
  https://github.com/udacity/der
  matologist-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)

    # commentthe below line if you
    alreadydownloadedthe
    datasetdownload_and_extract_dataset
    ()
    # preparingdata
    # generateCSV metadatafile to read img
    pathsandlabelsfrom itdef
    generate_csv(folder, label2int):
        folder_name = os.path.basename(folder)labels= list(label2int)
        # generateCSV filedf =
        pd.DataFrame(columns=["fil
        epath","label"])i = 0
        for labelin labels:
            print("Reading
                ",
                os.path.join(fo
                lder, label,
                "*"))for
                filepath in
                glob.glob(os.p
                ath.join(folder
                ,label,
                "*")):df.loc[i]
                = [filepath,
                label2int[label
                ]]
                i += 1
        output_file =
        f"{folder_name}.cs
        v"print("Saving",
        output_file)
        df.to_csv(output_fi
        le)

    # generateCSV files for all data portions, labelingnevus 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_fil
ename)df_valid =
pd.read_csv(valid_metadata_fil
ename)n_training_samples =
len(df_train)
n_validation_samples =
len(df_valid) print("Number of
training samples:",
n_training_samples) print("Number
of validationsamples:",
n_validation_samples)
train_ds =
tf.data.Dataset.from_tensor_slices((df_train["filepat
h"],df_train["label"])) valid_ds =
tf.data.Dataset.from_tensor_slices((df_valid["filepat
h"], df_valid["label"]))
```

Output:

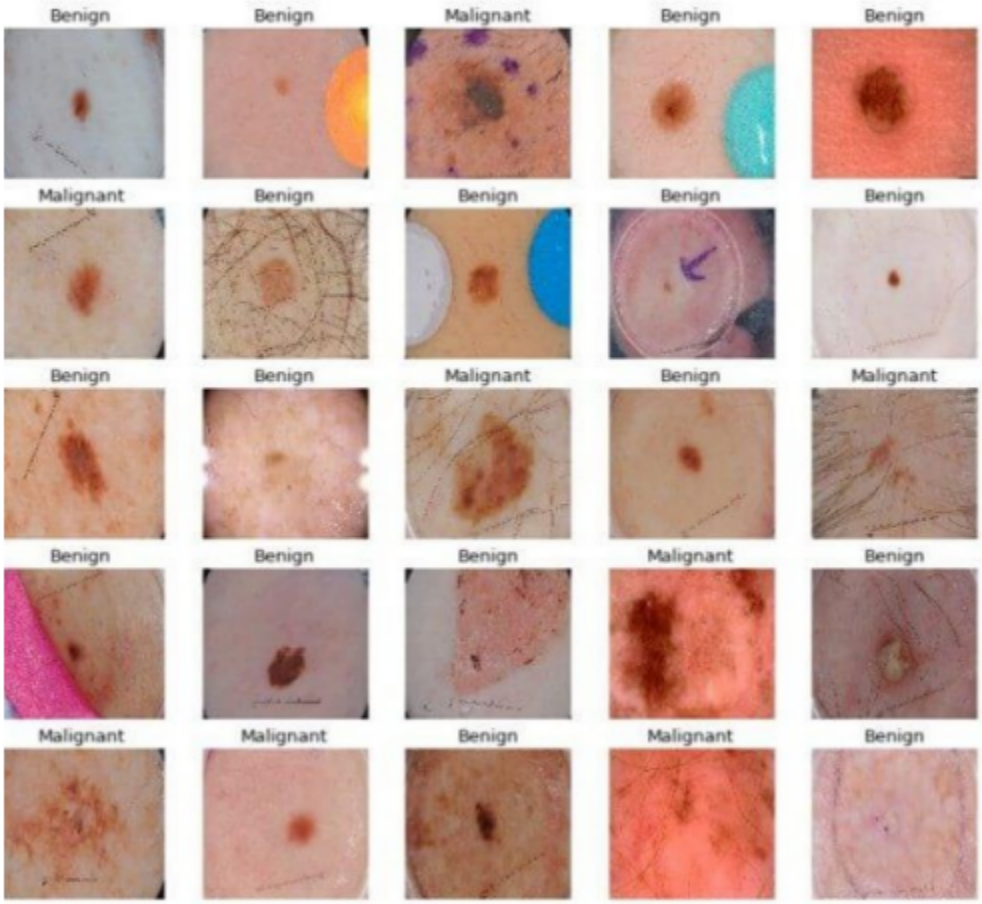
```
Number of training
samples: 2000
Number of validation
samples:150
```

```
#
preproce ss  data def decode_img(img):

# convert the compressed
 stringto a 3D uint8
 tensorimg =
 tf.image.decode_jpeg(img,
 channels=3)
 #  Use  `convert_image_dtype`  to
 convert to floatsin the [0,1] range.img =
 tf.image.convert_image_dtype(img,
 tf.float32)
 # resize the
 image to the
 desired
 size.returntf
 .image.resiz
 e(img, [299,
 299])


def
 process_path(filep
 ath, label):# load
 the raw data from
 the fileas a
 stringimg =
 tf.io.read_file(file
 path)
 img = decode_img(img)return img, label


valid_ds                    =
valid_ds.map(process_ph)train_ds                    =
train_ds.map
(process_pat
h)# test_ds =
test_ds
for image, label in train_ds.take(1):
 print("Imageshape:", image.shape)
print("Label:",
 label.numpy())
 Image shape: (299,
```

```
299, 3)
Label: 0#
trainingp arametersbatch_size
= 64
optimizer
=
"rmsprop"


def prepare_for_training(ds, cache=True,
 batch_size=64,
 shuffle_buffer_size=1000):ifcache:
  if isinstance(cache, str):
   ds = ds.cache(cache)else:
   ds = ds.cache()
 # shufflethe datasetds =
 ds.shuffle(buffer_size=shuffle_buffer
 _size)#
 Repeat
 forev
 erds =
 ds.repeat () #
 split to batches
 ds = ds.batch(batch_size)
 # `prefetch` lets the datasetfetch batches in the background
 while themodel#is training.
 ds =
 ds.prefetch(buffer_size=tf.data.experimen
 tal.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
 sho
 w_
 bat
```

```
ch
(b
atc
h):
plt.
fig
ure
(fig
si
ze=
(12
,12
))f
or
n
in
ran
ge(
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:




```
        # buildingthe 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 wheneverwe 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 improvedfrom 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_lossimproved 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:

0.8281Epoch
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_metadat
a_filename
="test.csv"

df_test =
pd.read_csv(test_metadata_fil
ename)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_met
adata_filenam
e)n_testing_samples
= len(df_test)
print("Numberof 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):ifcache:
   if isinstance(c ache, str):ds=

   ds.cache(cache) else:
ds = ds.cache()
```

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


  # load the weightswith
  the least loss
  m.load_weights("beni
  gn-vs-
  malignant_64_rmspro
  p_0.390.h5")
  print("Evaluating the
  model...")
loss, accuracy = m.evaluate(X_test, y_test,verbose=0)print("Loss:", loss, "Accuracy:", accuracy)
```

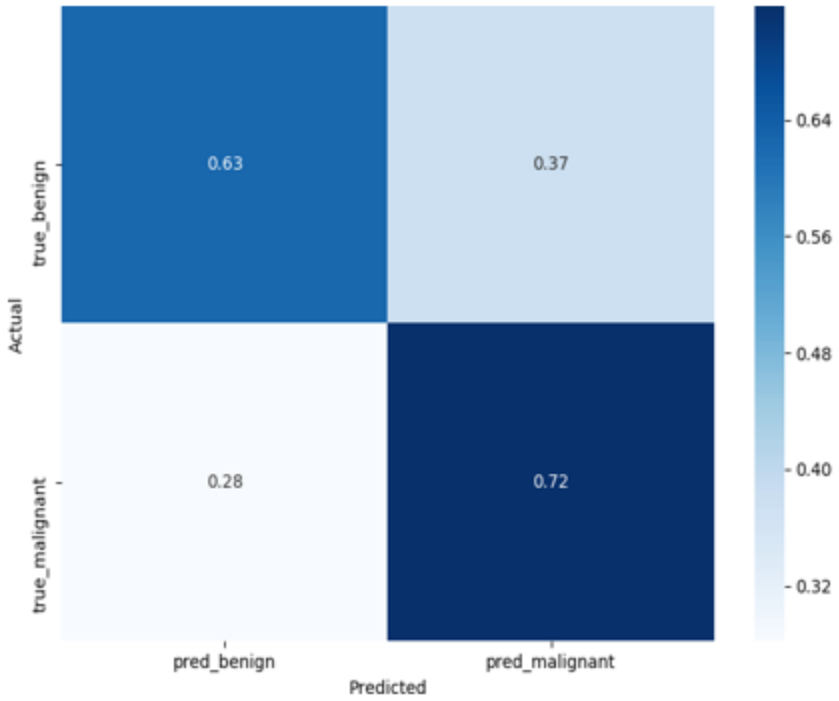
Output

```
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) forthat sample ifthe
  probability of 1 is 30% or more (insteadof 50%)

  """
  y_pred = m.predict(X_te st)if not threshold: threshold= 0.5 result =
  np.zeros((n_testing_samples
  ,))for i in
  range(n_test
  ing_samples
  ):
```

```
    # test melanomaprobabilityif 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 ismalignant,# it's assignedas
 malignant, otherwiseit's benign
 y_pred = get_predictions(threshold)
 def plot_confusion_matrix(y_test, y_pred):
  cmn =
  confusion_
  matrix(y_tes
  t,y_pred)#
  Normalise
  cmn = cmn.astype('float') /
  cmn.sum(axis=1)[:,np.newaxis]#
  print it
  print(cmn)fig, ax =
  plt.subplots(figsi
  ze=(10,10))
  sns.heatmap(cm
  n, 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('Predic

  ted')
  # plot the resultingconfusion matrixplt.show()

plot_confusion_matrix(y_test, y_pred
```

Output


0.63
0.37
0.28
0.72
true_benign
true_malignant
Actual
pred_benign
pred_malignant
Predicted
0.64
0.56
0.48
0.40
0.32

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

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

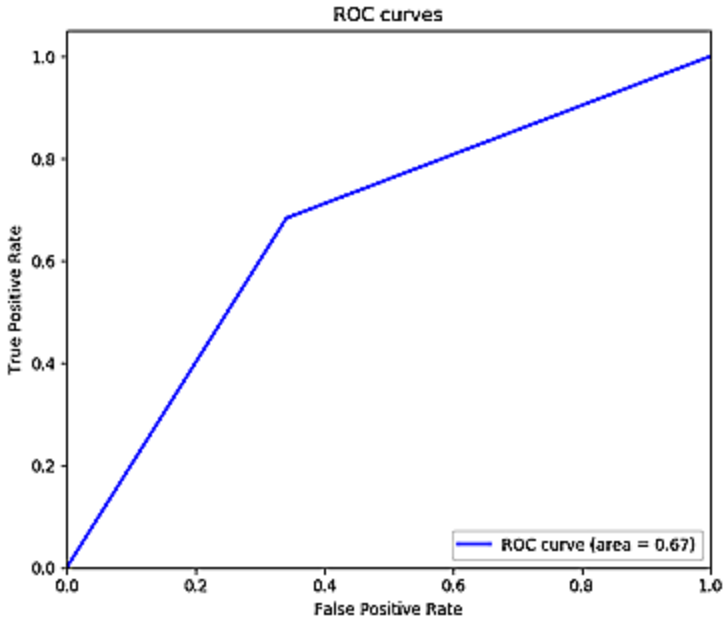
Output

```
Melanoma Sensitivity: 0.717948717948718
Melanoma Specificity: 0.6252587991718427
```

```
    def plot_roc_auc(y_true, y_pred):
        """
        This functionplots the ROC curves and



providesthescores."""
        #
        preparefor figure plt.figu re()
        fpr, tpr, _ = roc_curve(y_true, y_pred)
```

```
# obtainROC AUC
roc_auc = auc(fpr, tpr)# printscore
print(f"ROC AUC:
{roc_
auc:.3
f}")#
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
PositiveRate')plt.ylabel('True Positive Rate')
plt.title('ROC curves')plt.legend(loc="lo wer
right") plt.show()


plot_roc_auc(y_test, y_pred)
```

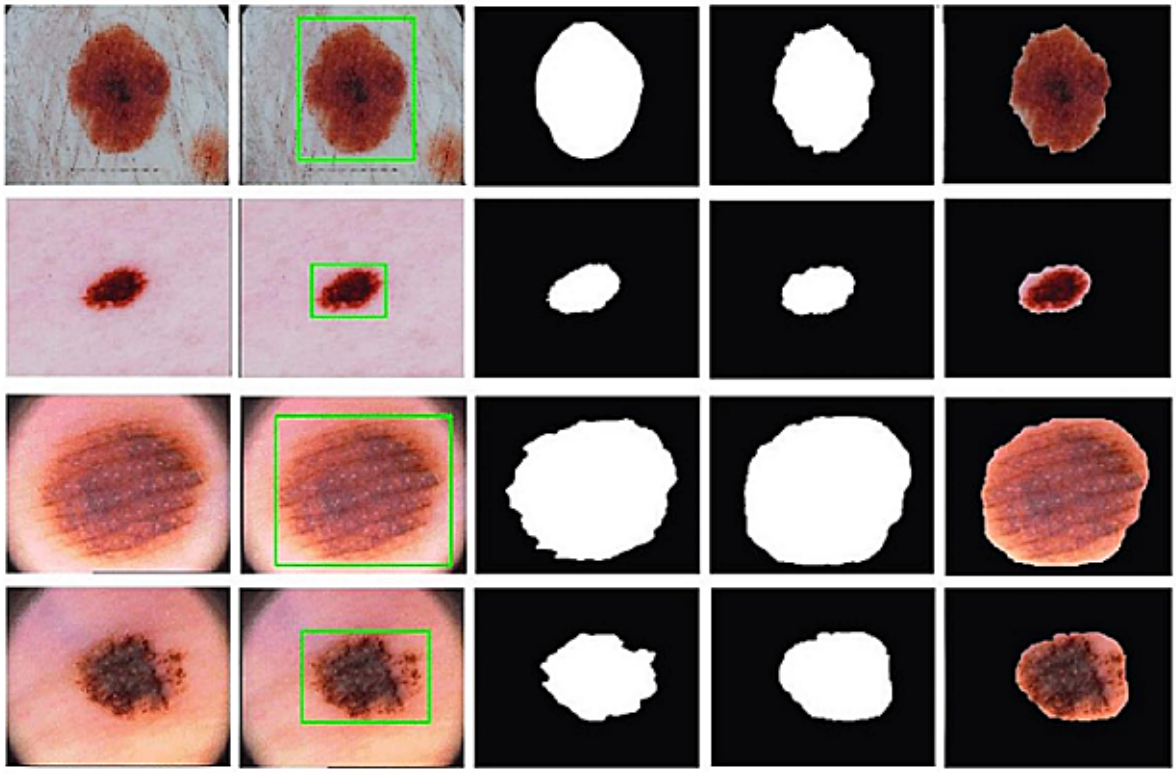
Output



ROC AUC: 0.671

8.Results

The finalresults are based on the accuracyresultsin the form of the melanoma and the non- melanomaskin diseases classifications.



9.Advantages and Disadvantages

9.1Advantages

Instant Response, improvesprediction of Skin Disease, no referral 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 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 enablethe use of CAD in the fieldof dermatology.

11.Future Scope

This implementation of the Structural Co-Occurrence matrices for featureextraction in the skin diseasesclassification and the pre-processing techniques are handled by using the Median filter, this filter helps toremove the salt and pepper noise in the image processing; thus, it enhances the qualityof the images, and normally, the skin diseasesare considered as the risk factorin all over the world.Our proposed approach provides97% of the classification of the accuracyresults while anotherexisting model such asFFT + SCM gives 80%, SVM + SCM gives 83%, KNN + SCM gives 85%, and SCM + CNN gives 82%. Future work is dependent on the increasedsupport vector machine'saccuracy in classifying skin illnesses, and SCM is used to manage the featureextraction technique.

12.Appendix

Github link:

https://github.com/IBM-EPBL/IBM-Project-36112-1660292850