UNIVERSITI MALAYA

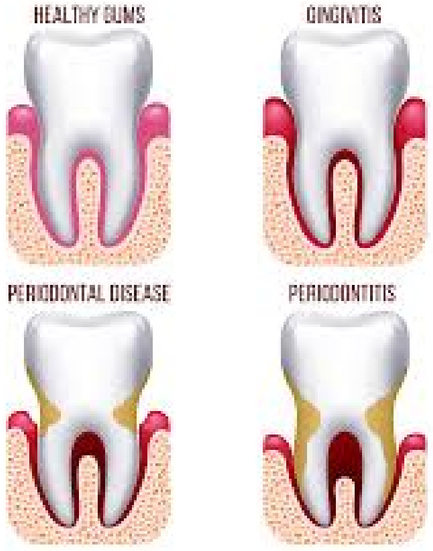
November, 22 2022

Classification of Periodontitis using Machine Learning Approach

Muhammad Asyraf Bin Mustaffa
17206591
SV: Dr Liyana

Introduction

- Periodontitis is a common oral disease that increases the chances of having it with increased age. (Periodontal Disease | Oral Health Conditions | Division of Oral Health, 2013)
- This disease is usually because of poor oral hygiene. (Periodontal Disease | Oral Health Conditions | Division of Oral Health, 2013)
- The disease is an inflammatory disease in tooth-supporting tissues and is categorized by the loss of tissue support. This will then contribute to severe cases of periodontitis, and tooth loss. (Könönen et al., 2019)
- The severe cases of Periodontitis can be measured by Clinical attachment loss (CAL), alveolar bone loss (BL), or the number of missing teeth. (Tonetti et al., 2018)

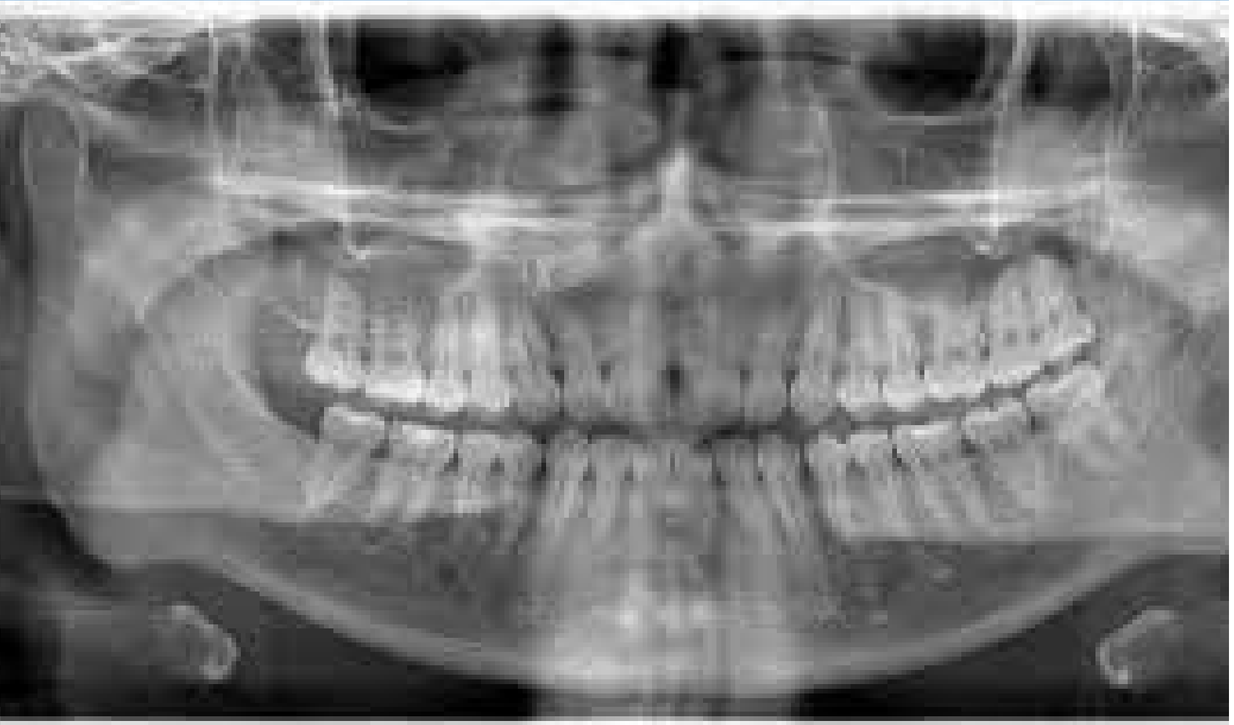


Problem Understanding

Discussion Point

- Problem Statement
- Objective

Problem Statement



01 It is hard nowadays to detect periodontitis just from panoramic x-ray images without the help of radiologist experts. (Mehta et al., 2021).

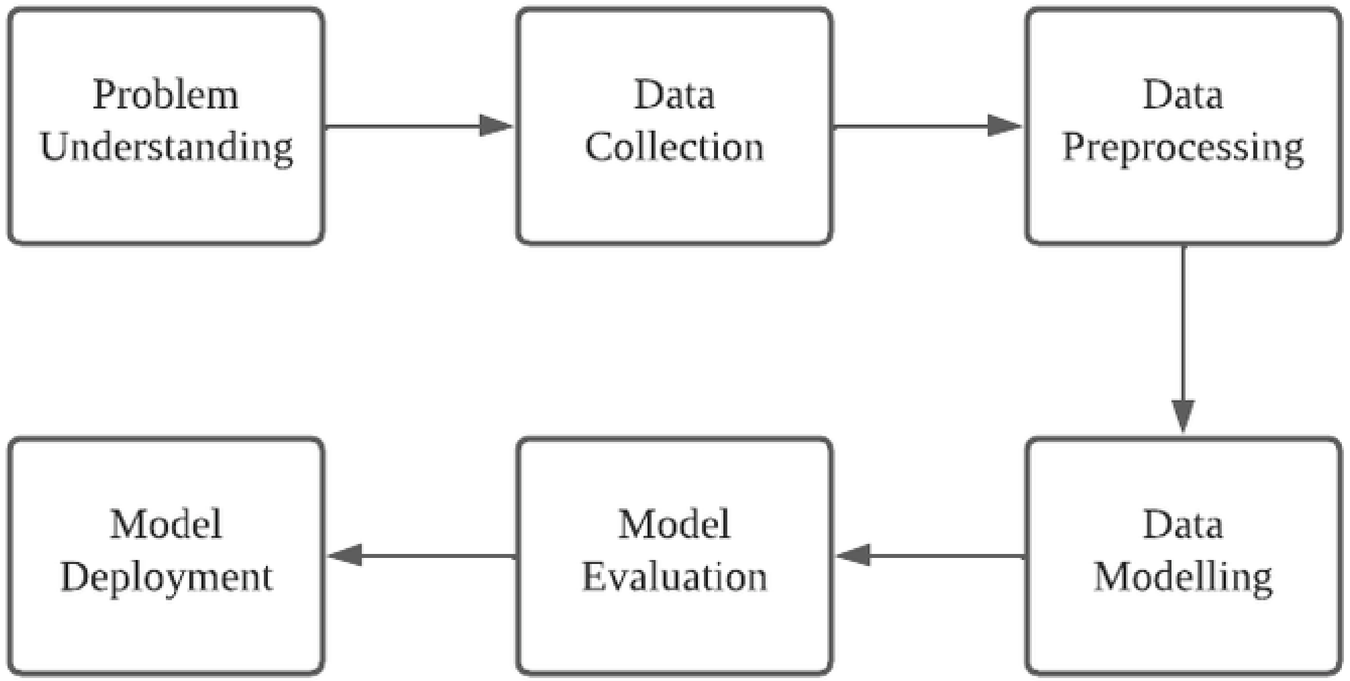
02 Some of the model isn't accurate enough to detect alveolar bone level loss(Akesson, Hakansson, & Rohlin, 1992; Hellen-Halme, Lith, & Shi, 2020; Pepelassi & Diamanti-Kipioti, 1997)

03 Adoption of the model in industry has been limited (Schwendicke, Golla, Dreher, & Krois, 2019)

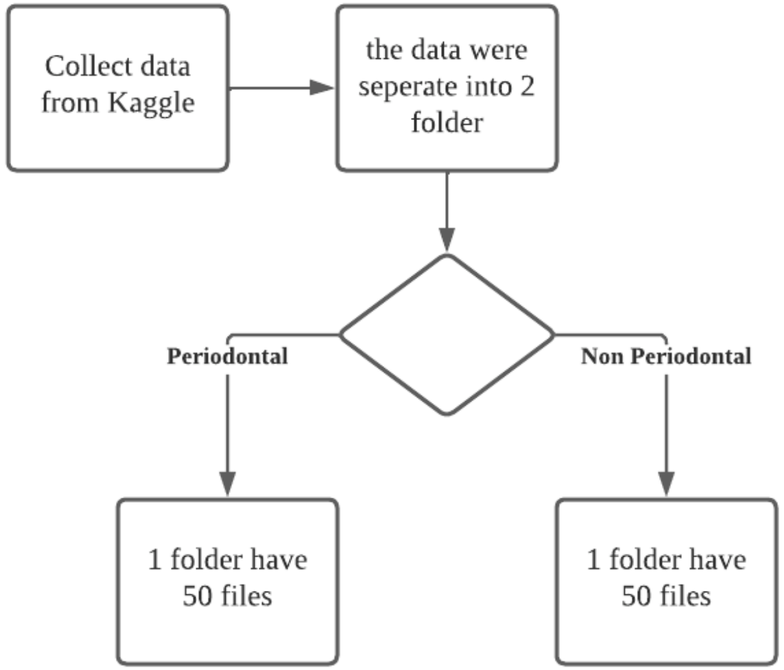
Objectives

- To develop a classification model to predict periodontitis using panoramic x-ray images with machine learning technique.
- To evaluate the proposed model on a hold-out test set of panoramic x-ray images.

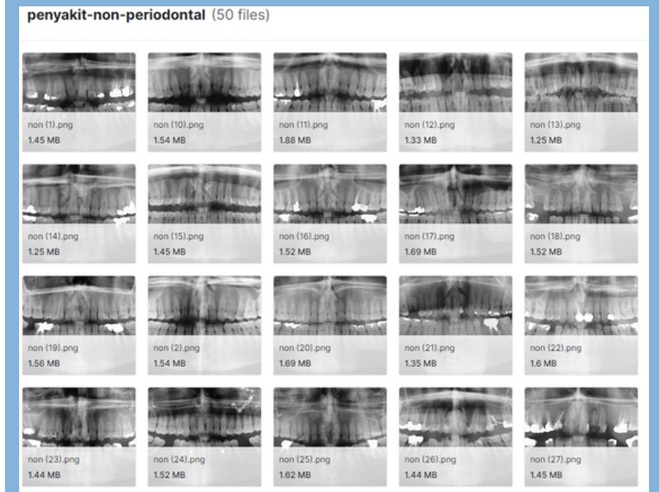
Data Science Methodology

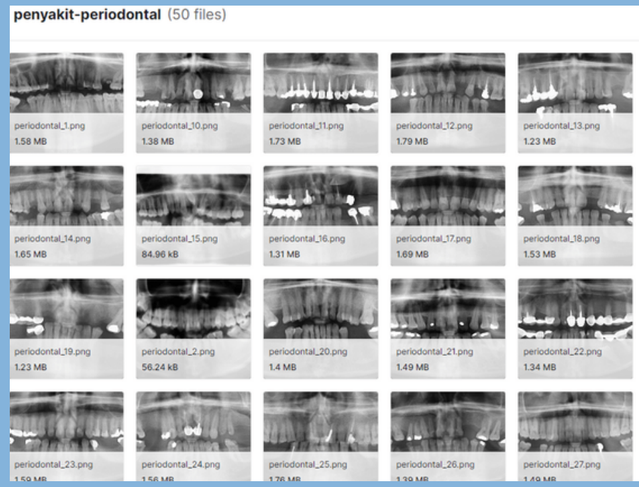


Data Collection

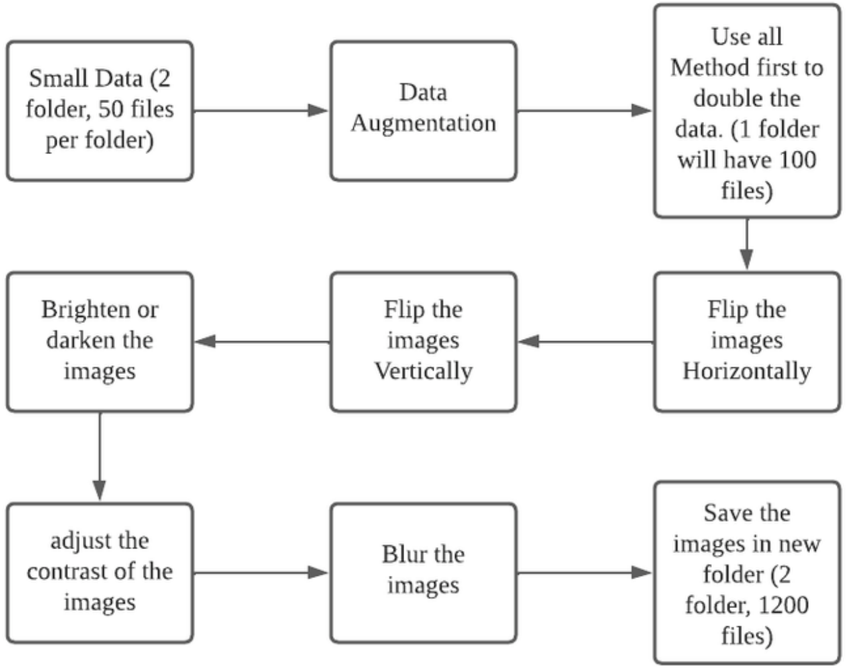


sample data



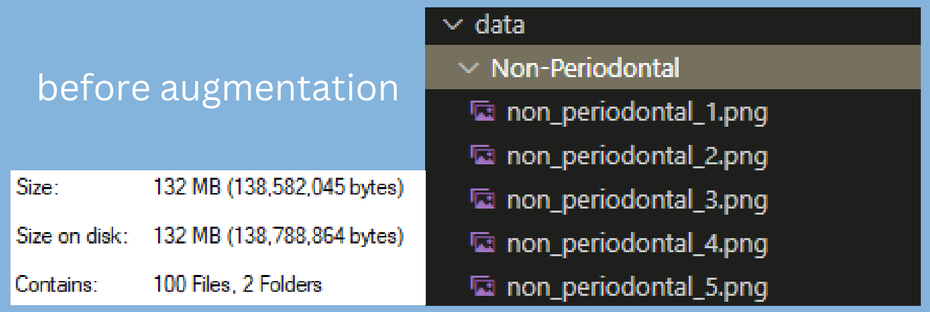


Data Preparation

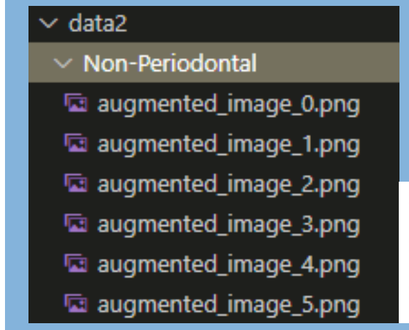


sample data

before augmentation



Size:	132 MB (138,582,045 bytes)
Size on disk:	132 MB (138,788,864 bytes)
Contains:	100 Files, 2 Folders



after augmentation

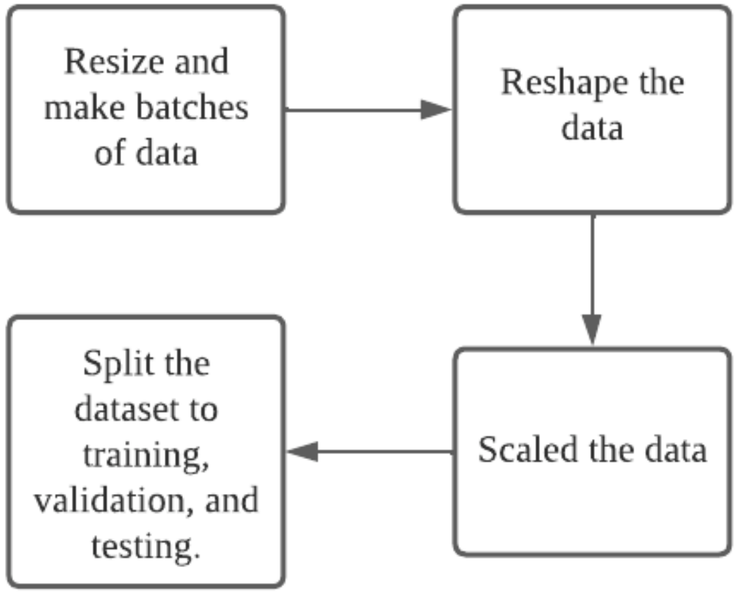
Size:	3.47 GB (3,729,031,613 bytes)
Size on disk:	3.47 GB (3,731,529,728 bytes)
Contains:	1,200 Files, 2 Folders

Data Augmentation code samples

```python
#augment image
augmentation_images = augmentation(images = images)

# Save the augmented image
for i, augmented_image in enumerate(augmentation_images):
  # Save each augmented image to a separate file
  cv2.imwrite("data/Non-Periodontal/augmented_image_{}.png".format(i), augmented_image)


#augment image
augmentation_images = augmentationhr(images = images)

# Save the augmented image
for i, augmented_image in enumerate(augmentation_images):
  # Save each augmented image to a separate file
  cv2.imwrite("data/Non-Periodontal/augmented_imagehr_{}.png".format(i), augmented_image)


#augment image
augmentation_images = augmentationud(images = images)

# Save the augmented image
for i, augmented_image in enumerate(augmentation_images):
  # Save each augmented image to a separate file
  cv2.imwrite("data/Non-Periodontal/augmented_imageud_{}.png".format(i), augmented_image)


#augment image
augmentation_images = augmentationml(images = images)

# Save the augmented image
for i, augmented_image in enumerate(augmentation_images):
  # Save each augmented image to a separate file
  cv2.imwrite("data/Non-Periodontal/augmented_imageml_{}.png".format(i), augmented_image)
```

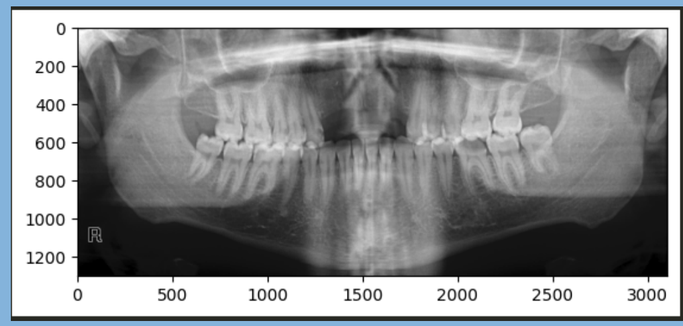
```python
import imgaug.augmenters as iaa
import cv2
import glob

#load dataset
images_path = glob.glob('data/Non-Periodontal/*.png')

images = []

for img_path in images_path:
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    images.append(img)

#image augmentation
augmentation = iaa.Sequential([
    iaa.Fliplr(0.5),#flip the data horizontally
    iaa.Flipud(0.5),#flip the data vertically
    iaa.Sometimes(0.5,iaa.Multiply((0.8, 1.2))),# multiply pixels in the image with 0.5
probability.
    iaa.Sometimes(0.5,iaa.LinearContrast((0.6,1.4))),# make contrast to the image with 0.5
probability.
    iaa.Sometimes(0.5,iaa.GaussianBlur((0.0, 3.0)))# blur the image with 0.5 probability
using gaussian kernels.
])

augmentationhr = iaa.Sequential([
    iaa.Fliplr(1.0)])
augmentationud = iaa.Sequential([
    iaa.Flipud(1.0)])
augmentationml = iaa.Sequential([
    iaa.Multiply((0.8, 1.2))])
augmentationln = iaa.Sequential([
    iaa.LinearContrast((0.6, 1.4))])
augmentationbl = iaa.Sequential([
    iaa.GaussianBlur((0.0, 3.0))])
```

Data Preparation cont.

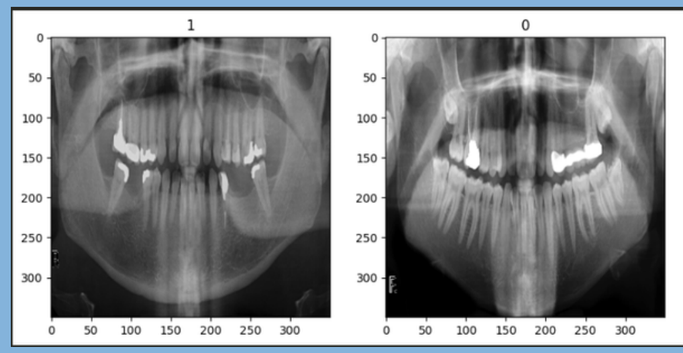


sample data

before scaled and reshape



after scale and reshape



Resize, Scaled, and split dataset code samples

Resize and batch size codes

```
data = tf.keras.utils.image_dataset_from_directory('data2', image_size=(350,350),
batch_size=32)
```

Scale codes

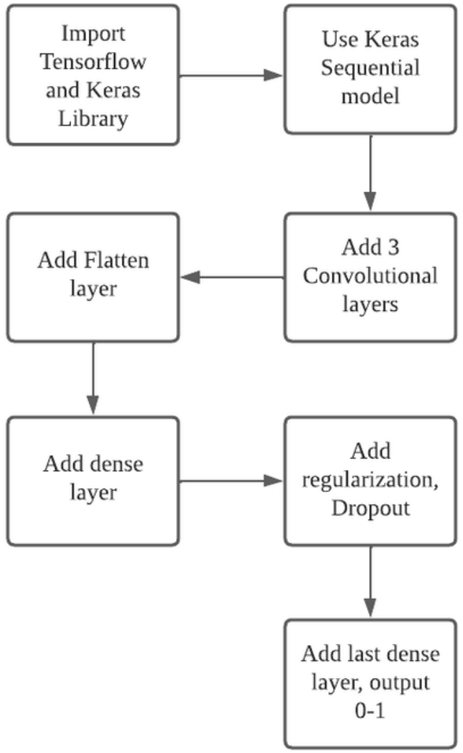
```
scaled_data = data.map(lambda x, y: (x/255, y))
```

Split code

```
train = scaled_data.take(train_size)
val = scaled_data.skip(train_size).take(val_size)
test = scaled_data.skip(train_size).skip(val_size).take(test_size)
```

```
#train 70%, validation 20%, test 10%
train_size = int(len(scaled_data)*.7)
val_size = int(len(scaled_data)*.2)+1
test_size = int(len(scaled_data)*.1)+1
train_size + val_size + test_size
```

Data Modelling

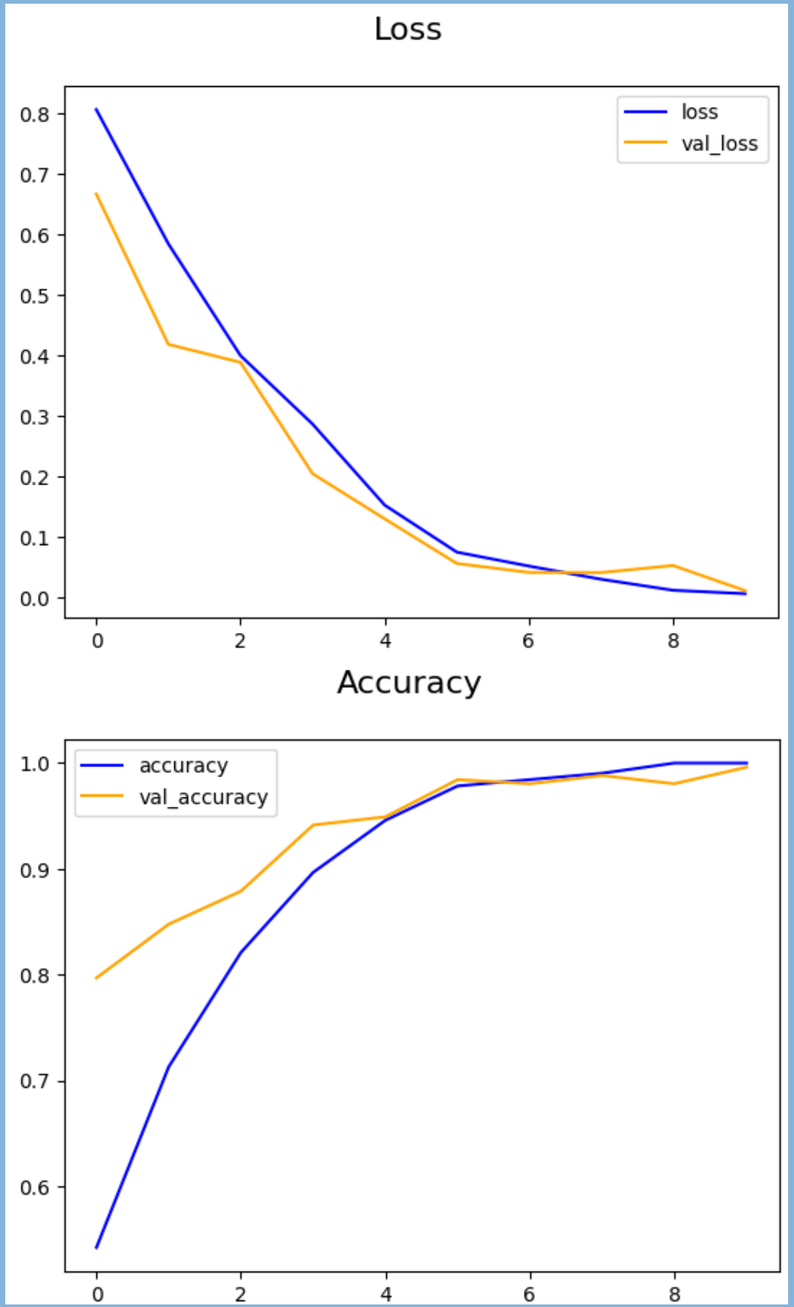


summary of the model

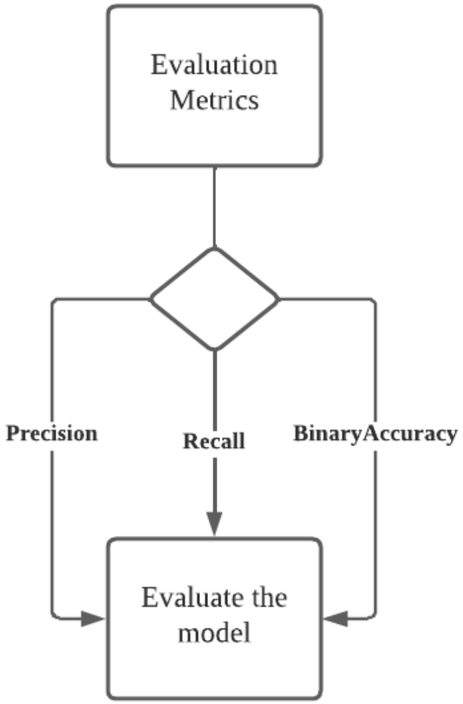
```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 348, 348, 16)      448

 max_pooling2d (MaxPooling2D  (None, 174, 174, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 172, 172, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 86, 86, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 84, 84, 16)        4624

 max_pooling2d_2 (MaxPooling  (None, 42, 42, 16)       0
 2D)

 flatten (Flatten)           (None, 28224)             0

 dense (Dense)               (None, 256)               7225600

 dropout (Dropout)           (None, 256)               0

...
Total params: 7,235,569
Trainable params: 7,235,569
Non-trainable params: 0
_________________________________________________________________
```

Data Modelling code samples and visualization of loss and accuracy graph.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
model = Sequential()
model.add(Conv2D(16,(3,3), 1, activation = 'relu', input_shape = (350,350,3)))
model.add(MaxPooling2D())# take maximum value after relu activation then return that value

model.add(Conv2D(32,(3,3),1, activation = 'relu'))
model.add(MaxPooling2D())

model.add(Conv2D(16,(3,3), 1, activation= 'relu'))
model.add(MaxPooling2D())

model.add(Flatten())# flattening the data down from multi dimension to 1 dim

model.add(Dense(256, activation = 'relu'))# fully connected layers
model.add(Dropout(0.5))
model.add(Dense(1, activation = 'sigmoid'))# get single output with 0-1 range
```



Model Evaluation



sample output

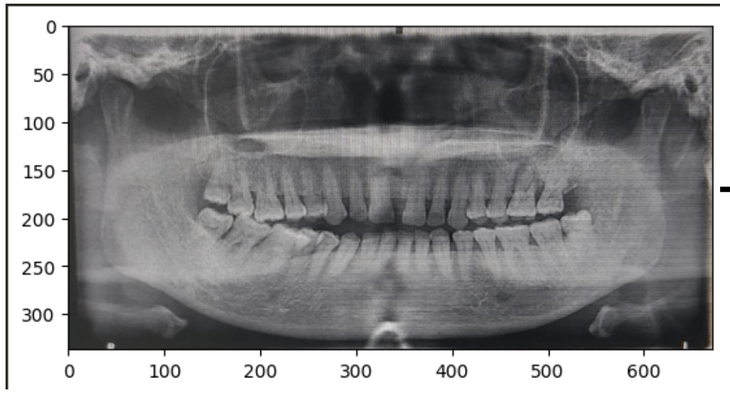
```
Precision:1.0, Recall:1.0, BAccuracy:1.0
```

Model evaluation code samples

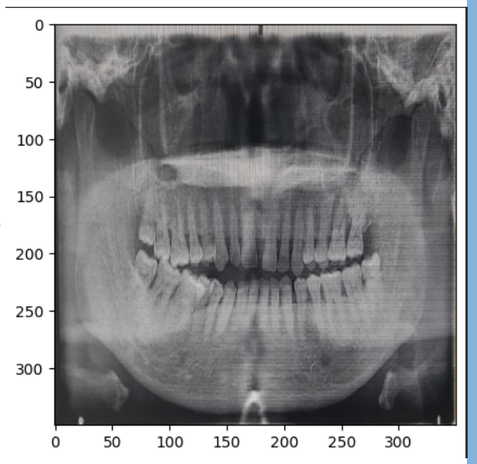
```python
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy

pre = Precision()
re = Recall()
Bacc = BinaryAccuracy()

for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    Bacc.update_state(y, yhat)

print(f'Precision:{pre.result().numpy()}, Recall:{re.result().numpy()}, BAccuracy:
{Bacc.result().numpy()}')
```

Model Evaluation cont.

- Test model using new images
- save the model in .h5 format



→



sample output

```
array([[0.9793197]], dtype=float32)
```

```
Predicted class is Periodontitis
```

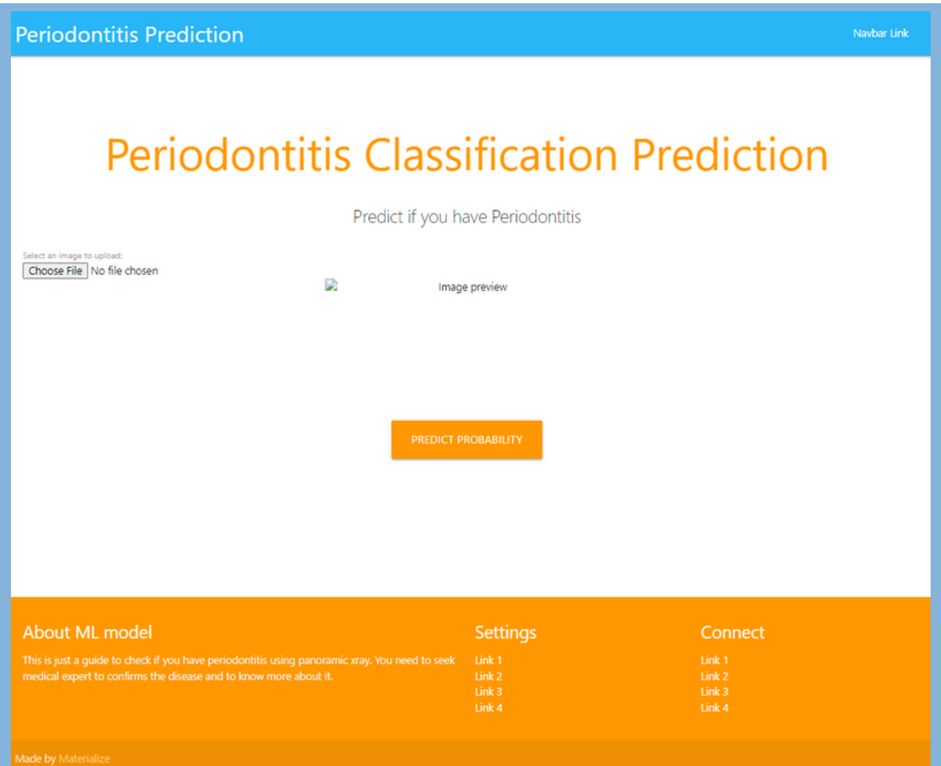
Model evaluation code samples cont.

```
img = cv2.imread('peritest2.png')
imgC = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(imgC)
plt.show()
resize = tf.image.resize(imgC, (350,350))
plt.imshow(resize.numpy().astype(int))
plt.show()
yhat = model.predict(np.expand_dims(resize/255,0))
print(yhat)
if yhat > 0.5:
    print("Predicted class is Periodontitis")
else:
    print("Predicted class is NON Periodontitis")

model.save(os.path.join('models', 'ClassificationPeriodontitis10epd2.h5'))
```

Deployment

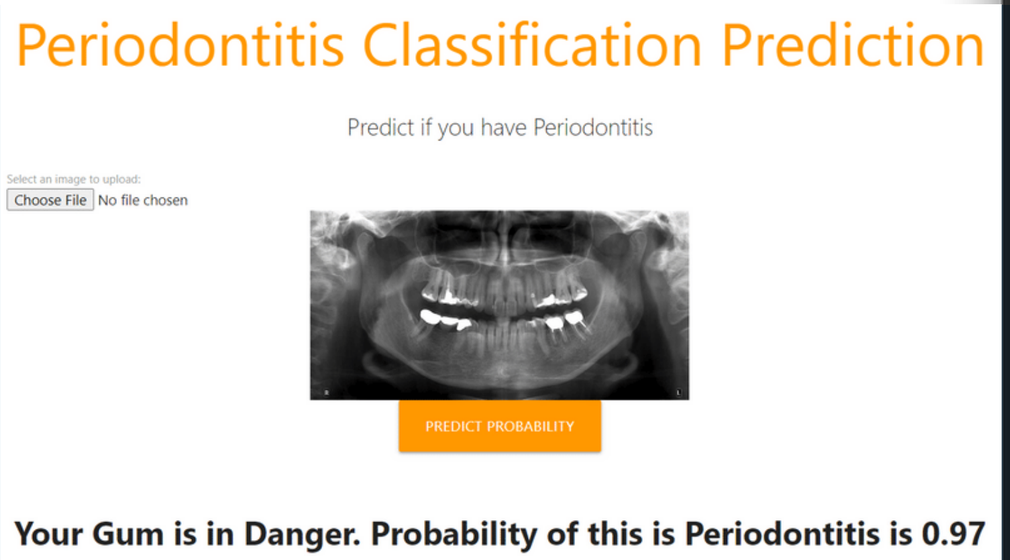
Building a Web Application using Flask and HTML.
The web is called Periodontitis Classification Prediction.

sample interface using html

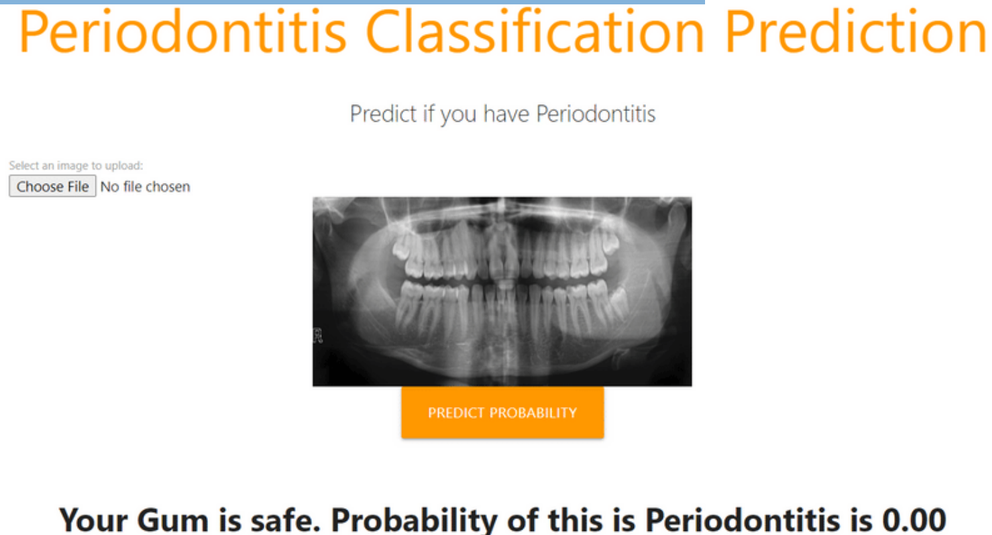


Model deployment code samples and output

Positive result



Negative result



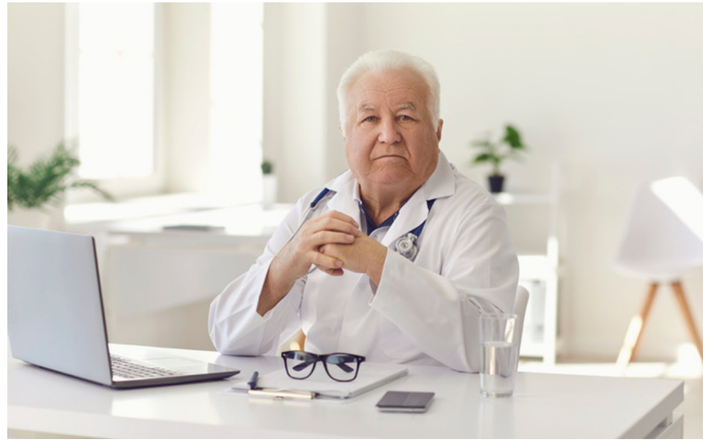

```python
import tensorflow as tf
from keras.models import load_model
from flask import Flask,request, render_template
import cv2
import numpy as np
import base64

app = Flask(__name__)

model=load_model('app1\ClassificationPeriodontitis10epd2.h5')

@app.route('/')
def hello_world():
    return render_template("Perio.html")

@app.route('/',methods=["GET","POST"])
def predict():

    file= request.files.get('image')
    if file is None:
        return 'No file input found in the request'

    file_bytes = file.read()
    if file_bytes is None:
        return 'Unable to read file data'

    try:
        image = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
    except cv2.error:
        # Try saving the file to disk and reading it back using cv2.imread
        with open('temp.jpg', 'wb') as f:
            f.write(file_bytes)
        image = cv2.imread('temp.jpg')
        if image is None:
            return 'Unable to decode file as an image'

    imgC = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    resize = tf.image.resize(imgC, (350,350))

    prediction=model.predict(np.expand_dims(resize/255,0))

    output='{0:.{1}f}'.format(prediction[0][0], 2)

    base64_string = base64.b64encode(file_bytes).decode('utf-8')

    if output>str(0.5):
        return render_template('Perio.html',pred='Your Gum is in Danger.\nProbability of
this is Periodontitis is {}'.format(output), base64image = base64_string)

    else:
        return render_template('Perio.html',pred='Your Gum is safe.\n Probability of this
is Periodontitis is {}'.format(output), base64image = base64_string)


if __name__ == '__main__':
    app.run()
```

Tools and coding

- Python
- Tensorflow
- Keras
- Flask
- Materialize
- HTML

Potential Stakeholder

- Clinicians
- Public
- Researchers



References

- Periodontitis - Symptoms and causes. (2020, February 14). Mayo Clinic. Retrieved November 18, 2022, from https://www.mayoclinic.org/diseases-conditions/periodontitis/symptoms-causes/syc-20354473
- Könönen E, Gursoy M, Gursoy UK. Periodontitis: A Multifaceted Disease of Tooth-Supporting Tissues. J Clin Med. 2019 Jul 31;8(8):1135. doi: 10.3390/jcm8081135. PMID: 31370168; PMCID: PMC6723779.
- Tonetti, MS, Greenwell, H, Kornman, KS. Staging and grading of periodontitis: Framework and proposal of a new classification and case definition. J Periodontol. 2018; 89(Suppl 1): S159– S172. https://doi.org/10.1002/JPER.18-0006
- Åkesson, L., Håkansson, J., & Rohlin, M. (1992). Comparison of panoramic and intraoral radiography and pocket probing for the measurement of the marginal bone level. Journal of clinical periodontology, 19(5), 326-332.
- Hellén-Halme, K., Lith, A., & Shi, X. Q. (2020). Reliability of marginal bone level measurements on digital panoramic and digital intraoral radiographs. Oral radiology, 36(2), 135-140.
- Schwendicke, F., Golla, T., Dreher, M., & Krois, J. (2019). Convolutional neural networks for dental image diagnostics: A scoping review. Journal of dentistry, 91, 103226.
- https://towardsdatascience.com/data-science-methodology-101-ce9f0d660336
- Periodontal Disease | Oral Health Conditions | Division of Oral Health. (2013). CDC. Retrieved January 9, 2023, from https://www.cdc.gov/oralhealth/conditions/periodontal-disease.html