Python Code Scripts-

Oral Disease Classification Report

Final Achievement Scores:

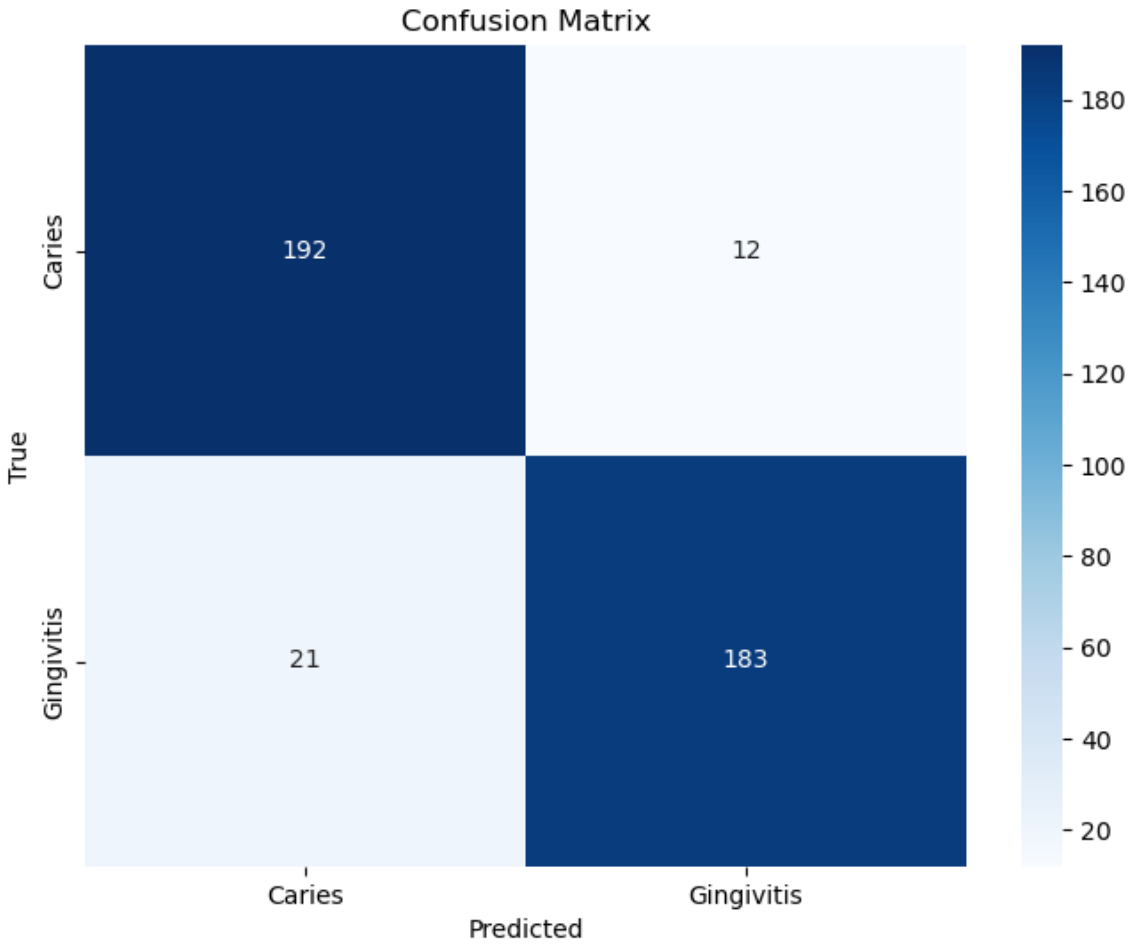
Accuracy: 0.92

Precision: 0.92

Recall: 0.92

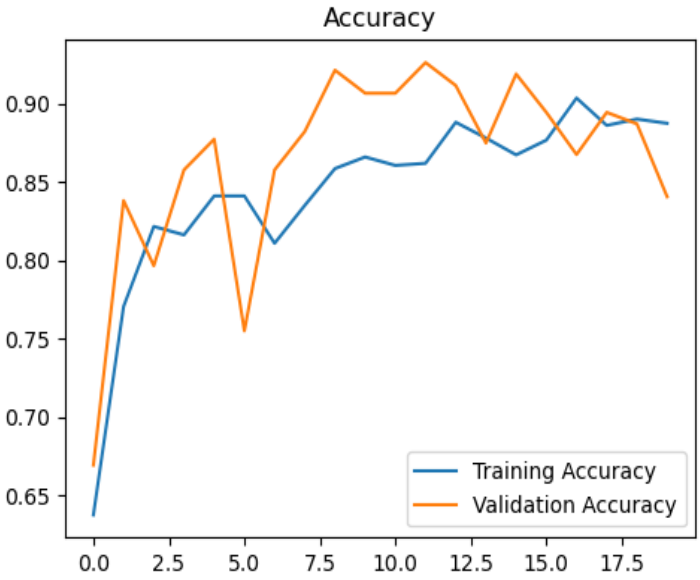
F1-Score: 0.92

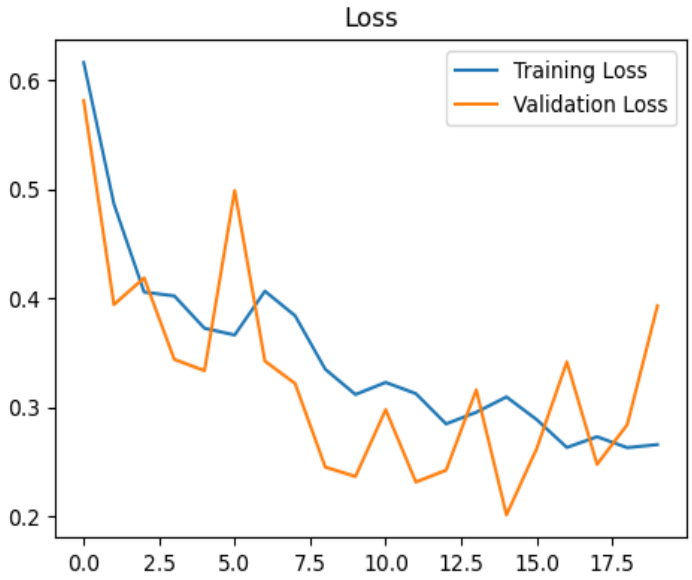
Confusion Matrix Analysis:



Python Code Scripts-

Training Accuracy and Validation Loss





Insights and Model Limitations

- The model performs well on the dataset, as evidenced by high accuracy and F1-scores.
- **Limitations**:
    - Dataset size might be a limitation if the data is not diverse enough.
    - Images with poor lighting or different angles may lead to misclassifications.
- **Improvements**:
    - Increase training data to include diverse samples.
    - Experiment with more advanced architectures like ResNet or EfficientNet.
    - Fine-tune a pre-trained model for improved performance on specific diseases.

Python Code Scripts-

oral_detection_model.ipynb

D: > oral_detection_model.ipynb > import os

+ Code + Markdown ···

Select Kernel

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
```

[2] Python

```python
train_dir = "OA/TRAIN"
test_dir = "OA/TEST"
```

[3] Python

```python
img_width, img_height = 128, 128
batch_size = 32

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
```

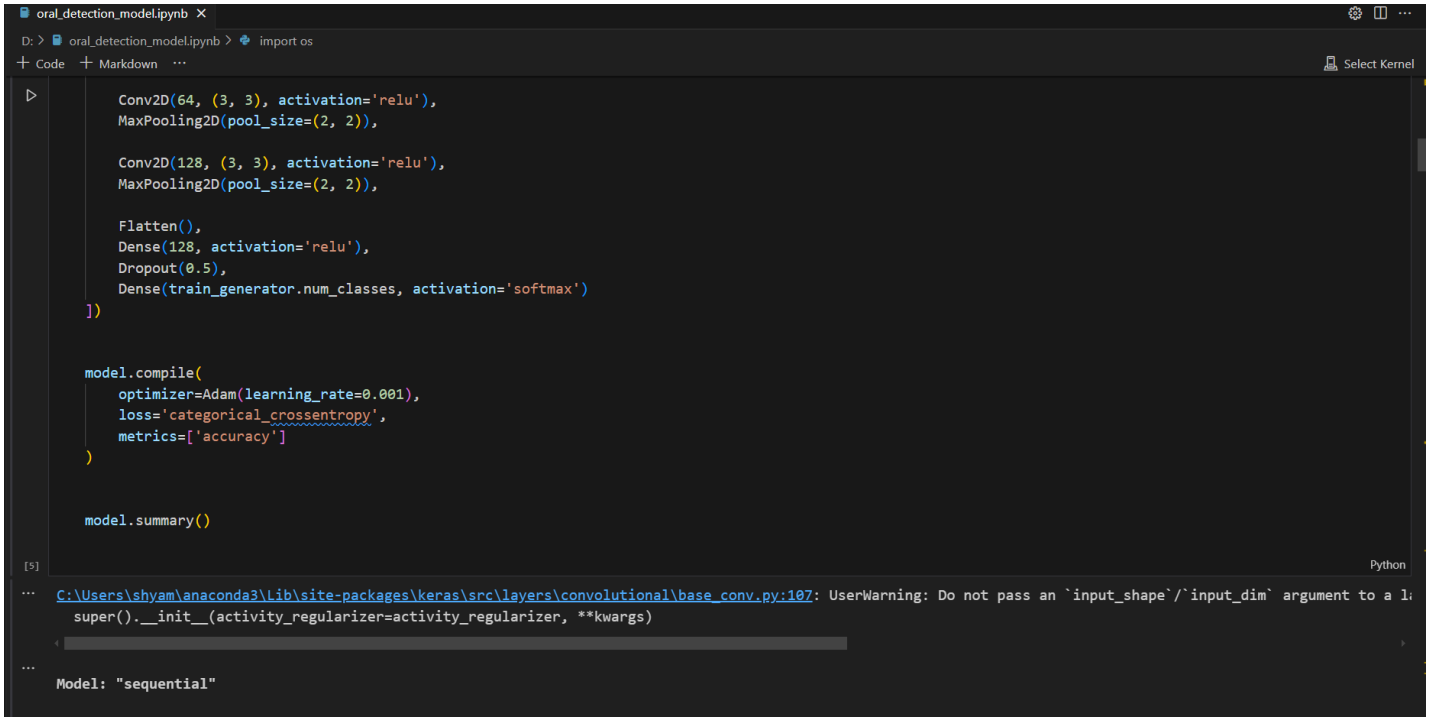
[4] Python

```
Found 1486 images belonging to 2 classes.
Found 408 images belonging to 2 classes.
```

```python
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_width, img_height, 3)),
    MaxPooling2D(pool_size=(2, 2)),
```

[5]

Python Code Scripts-



oral_detection_model.ipynb

D: > oral_detection_model.ipynb > import os

+ Code + Markdown ···

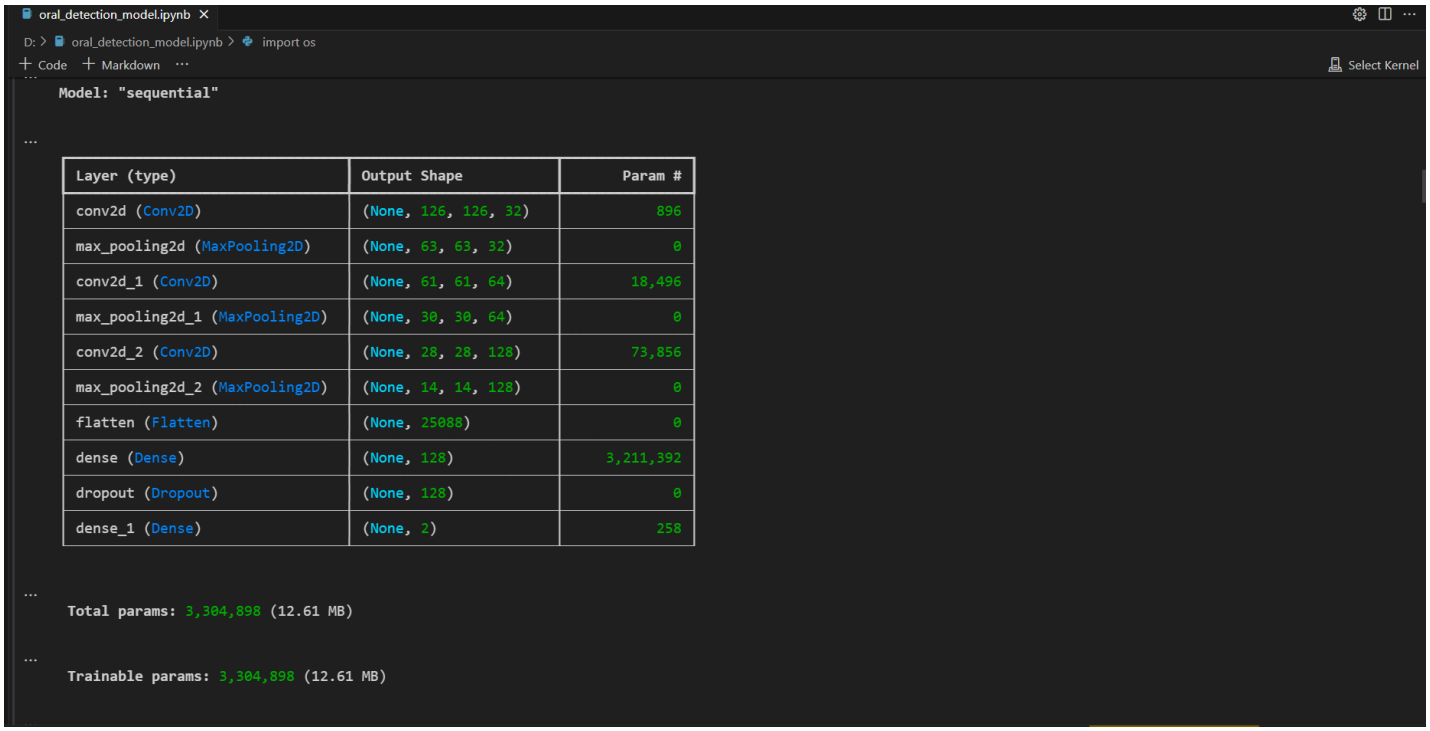
Select Kernel

```
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),

        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(train_generator.num_classes, activation='softmax')
    ])


    model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )


    model.summary()
```

[5] Python

```
C:\Users\shyam\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a la
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Model: "sequential"
```



oral_detection_model.ipynb

D: > oral_detection_model.ipynb > import os

+ Code + Markdown ···

Select Kernel

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3,211,392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 2)	258

```
Total params: 3,304,898 (12.61 MB)
```

```
Trainable params: 3,304,898 (12.61 MB)
```

Python Code Scripts-

oral_detection_model.ipynb

D: > oral_detection_model.ipynb > import os

+ Code + Markdown ··· Select Kernel

```
Non-trainable params: 0 (0.00 B)
```

```
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_generator,
    epochs=20,
    validation_data=test_generator,
    callbacks=[early_stopping]
)
```

[6] Python

```
C:\Users\shyam\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__
  self._warn_if_super_not_called()
Epoch 1/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.5608 - loss: 0.7067
C:\Users\shyam\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__
  self._warn_if_super_not_called()
47/47 ━━━━━━━━━━━━━━━━━━━━ 95s 2s/step - accuracy: 0.5624 - loss: 0.7048 - val_accuracy: 0.6691 - val_loss: 0.5814
Epoch 2/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 44s 938ms/step - accuracy: 0.7547 - loss: 0.5163 - val_accuracy: 0.8382 - val_loss: 0.3943
Epoch 3/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 51s 1s/step - accuracy: 0.8118 - loss: 0.4099 - val_accuracy: 0.7966 - val_loss: 0.4188
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 50s 1s/step - accuracy: 0.8209 - loss: 0.3885 - val_accuracy: 0.8578 - val_loss: 0.3440
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 50s 1s/step - accuracy: 0.8375 - loss: 0.3760 - val_accuracy: 0.8775 - val_loss: 0.3337
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 56s 1s/step - accuracy: 0.8510 - loss: 0.3512 - val_accuracy: 0.7549 - val_loss: 0.4989
```

oral_detection_model.ipynb

D: > oral_detection_model.ipynb > import os

+ Code + Markdown ··· Select Kernel

```
47/47 ━━━━━━━━━━━━━━━━━━━━ 50s 1s/step - accuracy: 0.8375 - loss: 0.3760 - val_accuracy: 0.8775 - val_loss: 0.3337
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 56s 1s/step - accuracy: 0.8510 - loss: 0.3512 - val_accuracy: 0.7549 - val_loss: 0.4989
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 47s 1s/step - accuracy: 0.8251 - loss: 0.3925 - val_accuracy: 0.8578 - val_loss: 0.3425
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 48s 1s/step - accuracy: 0.8546 - loss: 0.3693 - val_accuracy: 0.8824 - val_loss: 0.3219
Epoch 9/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 54s 1s/step - accuracy: 0.8709 - loss: 0.3272 - val_accuracy: 0.9216 - val_loss: 0.2452
Epoch 10/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 46s 975ms/step - accuracy: 0.8596 - loss: 0.3111 - val_accuracy: 0.9069 - val_loss: 0.2366
Epoch 11/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 52s 1s/step - accuracy: 0.8707 - loss: 0.3256 - val_accuracy: 0.9069 - val_loss: 0.2979
Epoch 12/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 55s 1s/step - accuracy: 0.8429 - loss: 0.3374 - val_accuracy: 0.9265 - val_loss: 0.2315
Epoch 13/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 55s 1s/step - accuracy: 0.8940 - loss: 0.2744 - val_accuracy: 0.9118 - val_loss: 0.2424
...
Epoch 19/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.8912 - loss: 0.2680 - val_accuracy: 0.8873 - val_loss: 0.2842
Epoch 20/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 50s 999ms/step - accuracy: 0.8837 - loss: 0.2661 - val_accuracy: 0.8407 - val_loss: 0.3931
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
plt.figure(figsize=(12, 4))
```

[7]

Python Code Scripts-

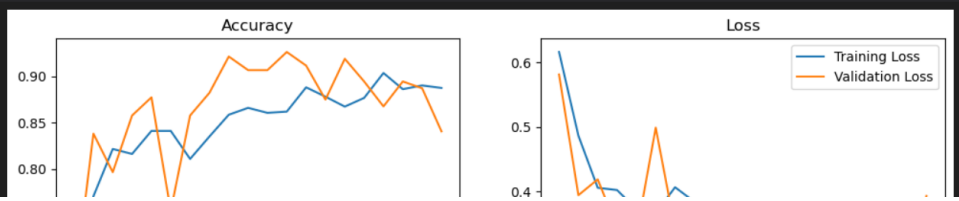
oral_detection_model.ipynb

D: > oral_detection_model.ipynb > import os

+ Code + Markdown ··· Select Kernel

```python
plt.figure(figsize=(12, 4))


plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()


plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()

plt.show()
```

[7] Python



oral_detection_model.ipynb

D: > oral_detection_model.ipynb > plt.figure(figsize=(12, 4))

+ Code + Markdown ··· Select Kernel

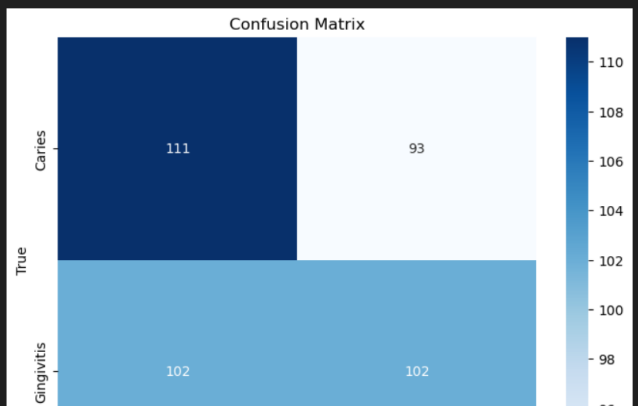
```python
loss, accuracy = model.evaluate(test_generator)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")


y_pred = model.predict(test_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_generator.classes


class_labels = list(test_generator.class_indices.keys())
print("\nClassification Report:\n")
print(classification_report(y_true, y_pred_classes, target_names=class_labels))


conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

[16] Python

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 168ms/step - accuracy: 0.9080 - loss: 0.2025
Test Loss: 0.20140476524829865
Test Accuracy: 0.9191176295280457
13/13 ━━━━━━━━━━━━━━━━━━━━ 2s 168ms/step
```

Python Code Scripts-

```
Classification Report:

              precision    recall  f1-score   support

      Caries       0.52      0.54      0.53       204
  Gingivitis       0.52      0.50      0.51       204

    accuracy                           0.52       408
   macro avg       0.52      0.52      0.52       408
weighted avg       0.52      0.52      0.52       408
```




```python
model = load_model("oral_disease_classifier.keras")


test_dir = "OA/TEST"
class_labels = sorted(os.listdir(test_dir))


Tabnine | Edit | Test | Explain | Document | Ask
def predict_image(image_path, model, class_labels):
    # Load and preprocess the image
    img = load_img(image_path, target_size=(128, 128))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)


    predictions = model.predict(img_array)
    predicted_class = class_labels[np.argmax(predictions)]
    confidence = np.max(predictions)

    return predicted_class, confidence, img
```

Python

```
C:\Users\shyam\anaconda3\Lib\site-packages\keras\src\saving\saving_lib.py:757: UserWarning: Skipping variable loading for optimizer 'rmsprop', because it has 12
  saveable.load_own_variables(weights_store.get(inner_path))
```

Python Code Scripts-

oral_detection_model.ipynb

D: > oral_detection_model.ipynb > plt.figure(figsize=(12, 4))

+ Code + Markdown ··· Select Kernel

```
test_images = []
for class_folder in class_labels:
    class_folder_path = os.path.join(test_dir, class_folder)
    test_images.extend([os.path.join(class_folder_path, img) for img in os.listdir(class_folder_path) if img.endswith(('.jpg', '.png', '.jpeg'))])


plt.figure(figsize=(12, 8))
for i, image_path in enumerate(test_images[:6]):
    predicted_class, confidence, img = predict_image(image_path, model, class_labels)


    plt.subplot(2, 3, i + 1)
    plt.imshow(img)
    plt.title(f"Predicted: {predicted_class}\nConfidence: {confidence:.2f}")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

[3] Python

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 592ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 164ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step
```