NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Artificial Intelligence and Expert Systems

Project Report

Group Members

Muhammad Shazil Khan	CT-22045
Syed Faiq Hussain Naqvi	CT-22029
Hamza Ahmed Bham	CT-22026

CSIT-A

Batch-2022

Contents

Skin Lesion (Melanoma) Classification

Problem Description:

The goal is a binary classification: given an image of a skin lesion, predict **benign** vs **malignant (melanoma)**. In practice, dermatologists examine dermoscopy images to identify melanomas, which can look visually similar to benign moles. Machine learning models can assist by learning subtle patterns. As one review summarizes, “computer-aided diagnosis utilizing machine learning can be used to differentiate malignant and benign skin lesions”pmc.ncbi.nlm.nih.gov. This project aims to train a CNN-based model to make that distinction automatically from images.

Domain: Healthcare Imaging

The project falls under medical imaging and healthcare AI. It focuses on **dermoscopic skin images**, a key domain where deep learning has shown great potential. Convolutional neural networks (**CNNs**) are widely used in this area; for example, recent research notes significant research into the use of CNNs to classify skin lesions from dermoscopic images”. Skin cancer (particularly *melanoma*) is a serious health issue, and automated image analysis can aid dermatologists by screening suspicious lesions.

Real world Impact:

Accurate automated classification can improve

- **early detection** of melanoma.
- Early-detected melanoma has a **very high survival rate** – about 99% 5-year survival when caught early – but drops sharply if diagnosed late. Thus, timely screening is crucial.
- AI tools can make screening **more accessible**, especially where dermatologists are scarce (e.g., rural areas or as a preliminary mobile app).

By flagging potential melanomas early, the model could help patients seek treatment sooner, potentially saving lives. In summary,

- this project has high impact for early cancer detection and improving diagnostic access.

Deep Learning Models:

This image classification task is well-suited to **CNNs** with transfer learning. Modern architectures pretrained on ImageNet can be fine-tuned on skin lesion data. In literature, models like **ResNet50** and **DenseNet201** have been successfully applied to melanoma classification Likewise, recent work shows **EfficientNet (B0)** achieving state-of-the-art accuracy on ISIC challenge data (e.g. EfficientNet-B7 reached 84.4% accuracy)

Model Used:

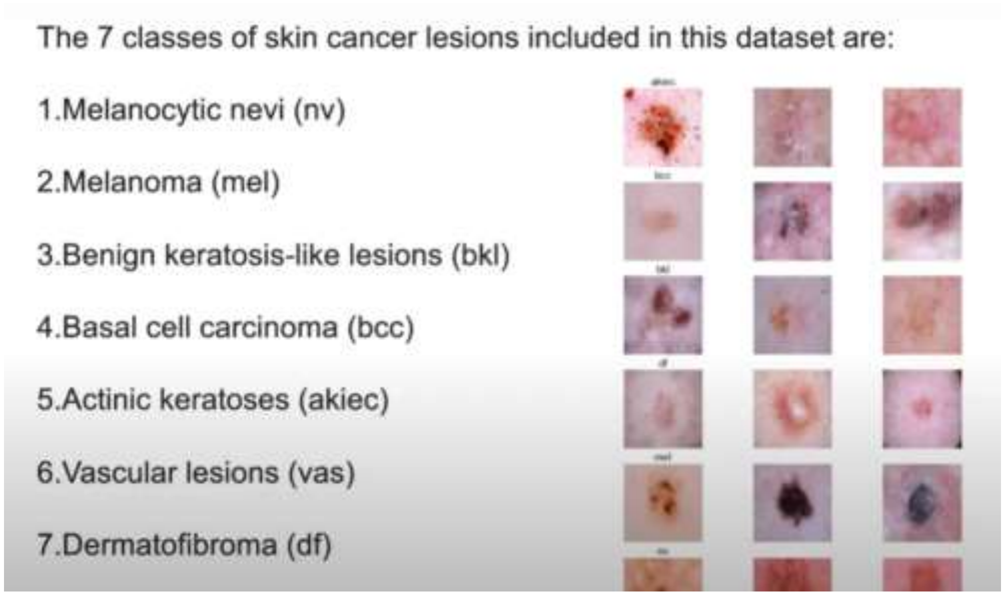
EfficientNet (B0–B7): Scalable CNNs optimized for accuracy/size; top-ranked on ImageNet and ISIC benchmarks

The Model can be used **with transfer learning**: load pretrained weights (e.g. from ImageNet) and then retrain final layers on the lesion dataset. Transfer learning is essential since medical image datasets are relatively small

Dataset Used:

Skin Cancer MNIST (HAM10000): A public collection of 10,015 dermatoscopic images covering common lesions. Available on Kaggle here. (The HAM10000 dataset is also on ISIC; it contains 10,015 images in seven classes including melanoma)

Brief Explanation of The Dataset:


The 7 classes of skin cancer lesions included in this dataset are:

1.Melanocytic nevi (nv)

2.Melanoma (mel)

3.Benign keratosis-like lesions (bkl)

4.Basal cell carcinoma (bcc)

5.Actinic keratoses (akiec)

6.Vascular lesions (vas)

7.Dermatofibroma (df)

Out of these Seven Classes some are malignant and some are benign

Abbreviation	Full Name	Malignancy
akiec	Actinic Keratoses and Intraepithelial Carcinoma	✓ Malignant
bcc	Basal Cell Carcinoma	✓ Malignant
bkl	Benign Keratosis-like Lesions	✗ Benign
df	Dermatofibroma	✗ Benign
mel	Melanoma	✓ Malignant
nv	Melanocytic Nevi	✗ Benign
vasc	Vascular Lesions	✗ Benign

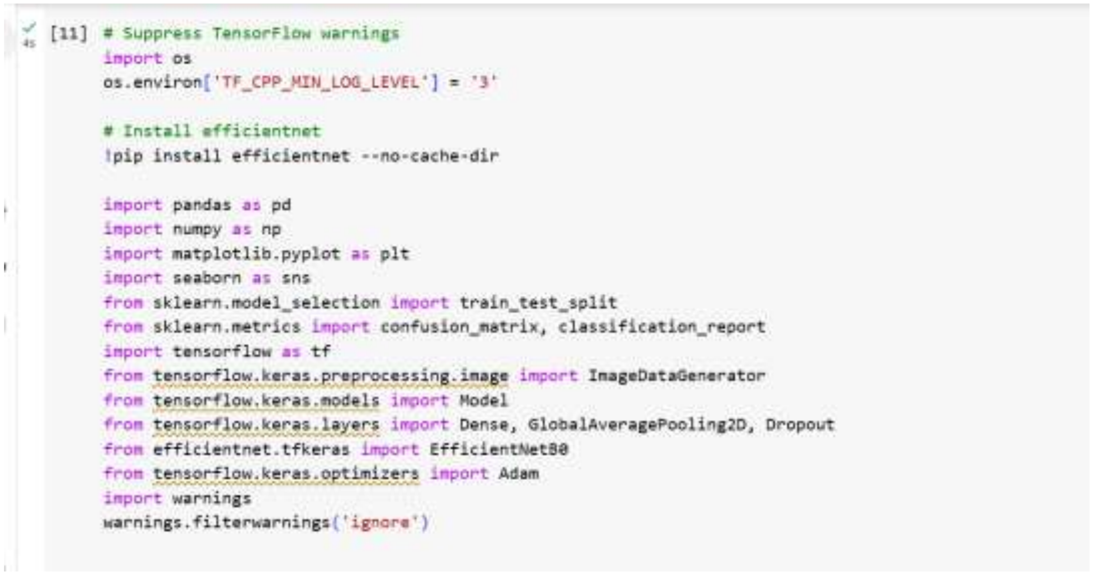
Q: What is Malignant and Benign ?

- **Benign** is like a growth that stays in one place and doesn't spread to other parts of the body. It's generally not harmful. Think of it as a neighbor who stays in their own yard.
- **Malignant** is like a harmful growth that can invade nearby tissues and spread to other parts of the body. It's cancerous and can be life-threatening.

Think of it as an unwanted guest who not only comes into your house but also tries to take over the whole neighborhood.

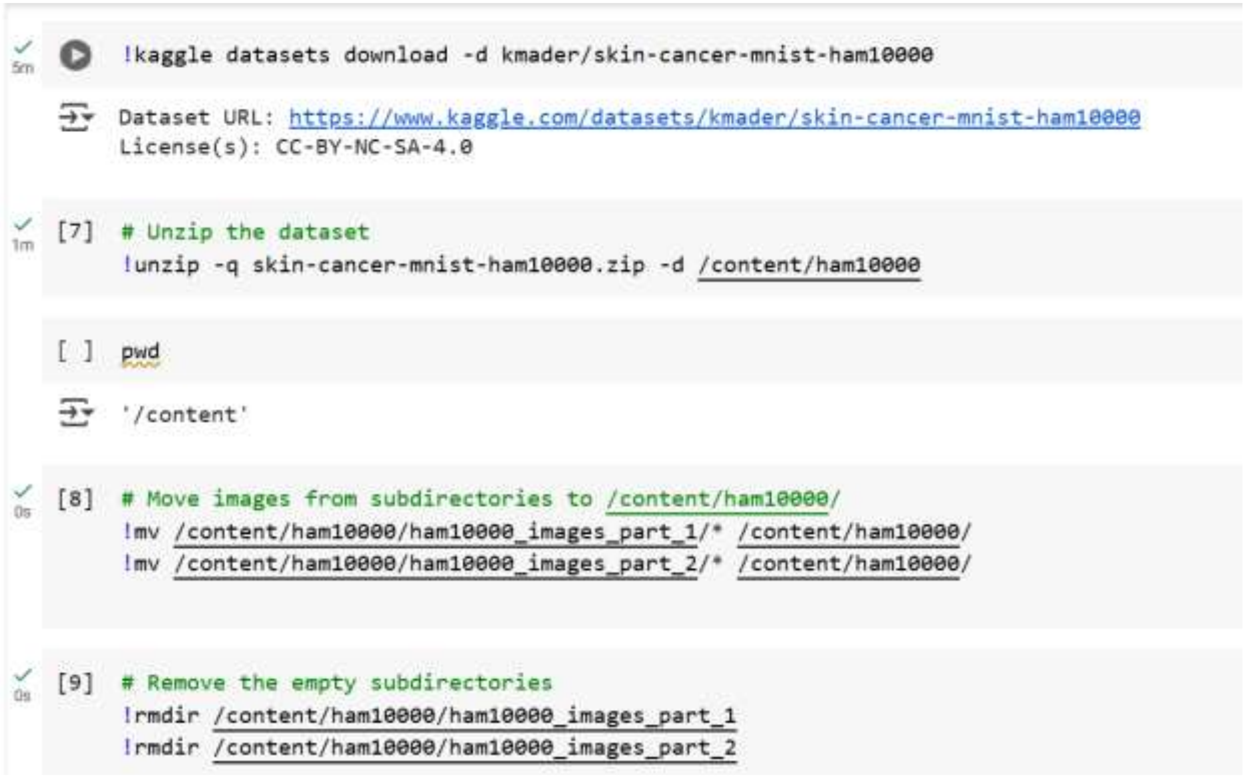
JUPYTER NOTEBOOK EXPLANATION:

Importing Libararies:



```
[11] # Suppress TensorFlow warnings
     import os
     os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

     # Install efficientnet
     !pip install efficientnet --no-cache-dir

     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix, classification_report
     import tensorflow as tf
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.models import Model
     from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
     from efficientnet.tfkeras import EfficientNetB0
     from tensorflow.keras.optimizers import Adam
     import warnings
     warnings.filterwarnings('ignore')
```

Load Dataset From Kaggle and Unzip in Colab Content Dir.



```
!kaggle datasets download -d kmader/skin-cancer-mnist-ham10000
```

```
Dataset URL: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000
License(s): CC-BY-NC-SA-4.0
```

```
[7]  # Unzip the dataset
     !unzip -q skin-cancer-mnist-ham10000.zip -d /content/ham10000
```

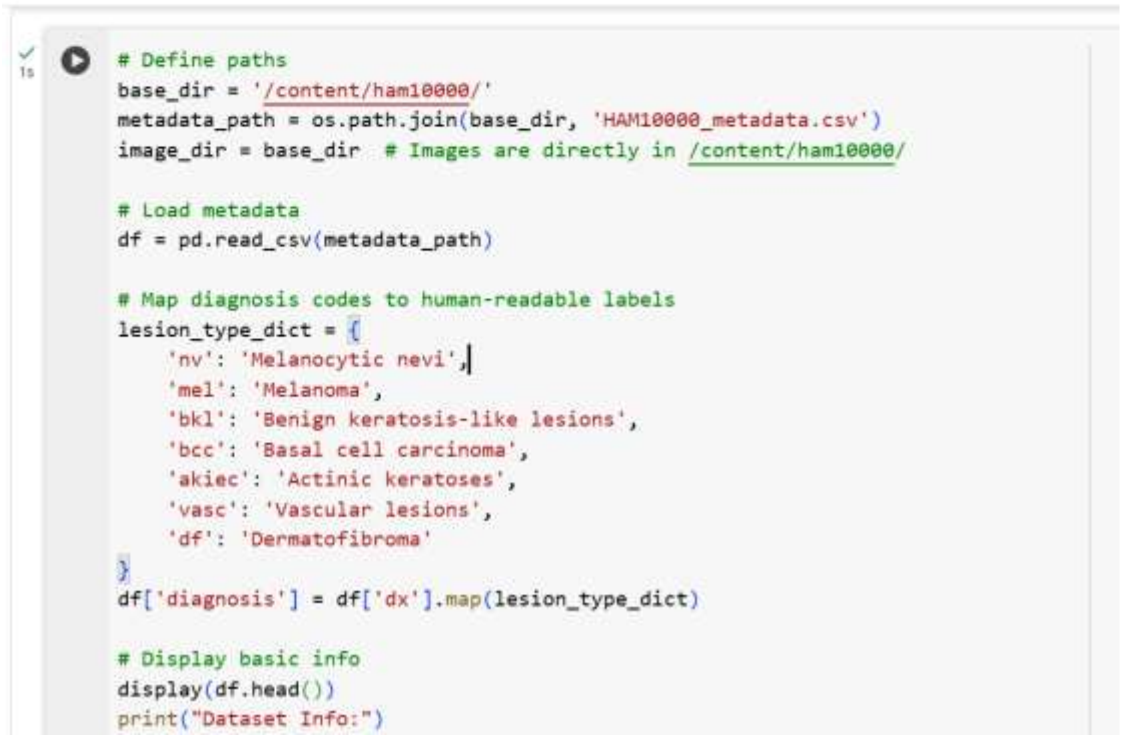
```
[ ]  pwd
```

```
'/content'
```

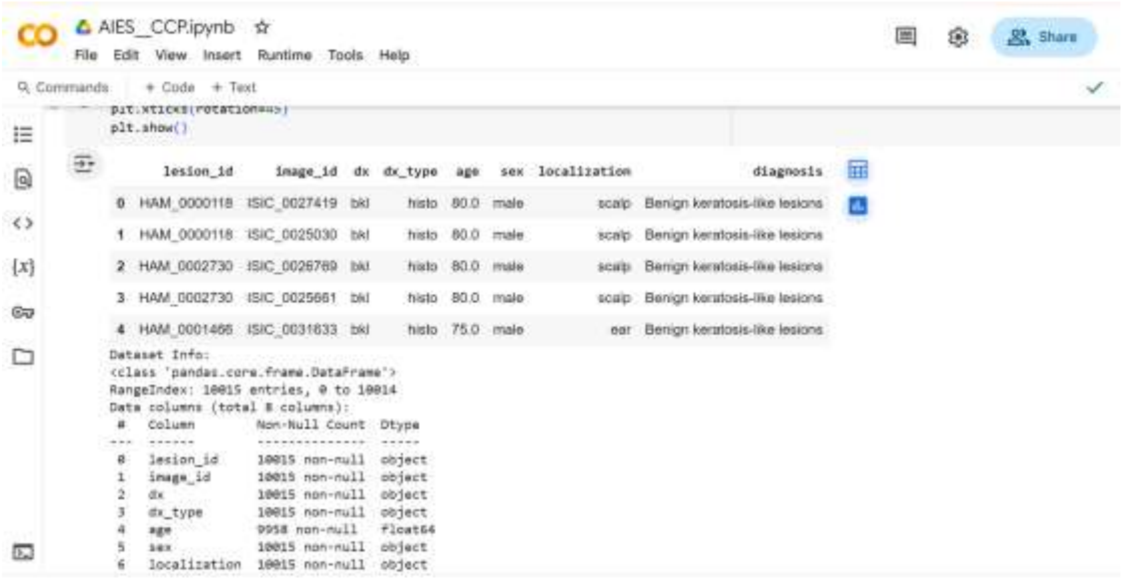
```
[8]  # Move images from subdirectories to /content/ham10000/
     !mv /content/ham10000/ham10000_images_part_1/* /content/ham10000/
     !mv /content/ham10000/ham10000_images_part_2/* /content/ham10000/
```

```
[9]  # Remove the empty subdirectories
     !rmdir /content/ham10000/ham10000_images_part_1
     !rmdir /content/ham10000/ham10000_images_part_2
```

Dataset Meta Description and Visualization Distribution:



```python
# Define paths
base_dir = '/content/ham10000/'
metadata_path = os.path.join(base_dir, 'HAM10000_metadata.csv')
image_dir = base_dir  # Images are directly in /content/ham10000/

# Load metadata
df = pd.read_csv(metadata_path)

# Map diagnosis codes to human-readable labels
lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}
df['diagnosis'] = df['dx'].map(lesion_type_dict)

# Display basic info
display(df.head())
print("Dataset Info:")
```



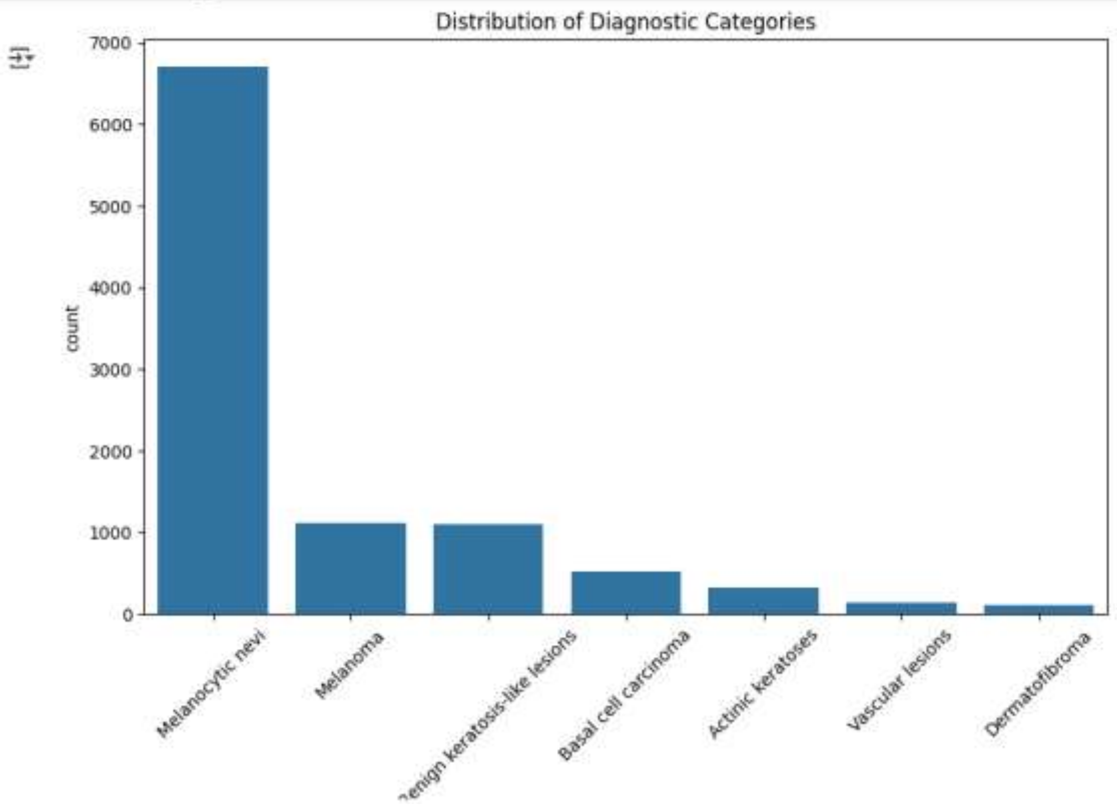
```python
plt.xticks(rotation=45)
plt.show()
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	diagnosis
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	Benign keratosis-like lesions
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	Benign keratosis-like lesions
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	Benign keratosis-like lesions
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	Benign keratosis-like lesions
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	Benign keratosis-like lesions

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10015 entries, 0 to 10014
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   lesion_id     10015 non-null  object
 1   image_id      10015 non-null  object
 2   dx            10015 non-null  object
 3   dx_type       10015 non-null  object
 4   age           9958 non-null   float64
 5   sex           10015 non-null  object
 6   localization  10015 non-null  object
```

- **lesion_id:** Unique identifier for the lesion.
- **image_id**: Unique identifier for the image (e.g., ISIC_0024306).
- **dx**: Diagnosis label (one of the seven categories: mel, nv, bcc, akiec, bkl, df, vasc).
- **dx_type**: Confirmation method (histopathology, follow-up, consensus, confocal).

- **age**: Patient age (approximate).
- **sex**: Patient sex.
- **localization**: Body part where the lesion is located (e.g., back, face, arm).

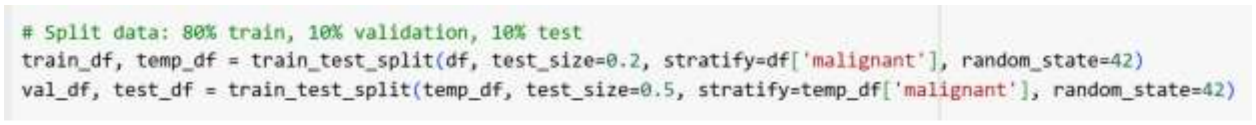


Create Binary Labels:

```
# Add binary label for malignant vs. benign
malignant_classes = ['akiec', 'bcc', 'mel']
df['malignant'] = df['dx'].apply(lambda x: 1 if x in malignant_classes else 0)
```

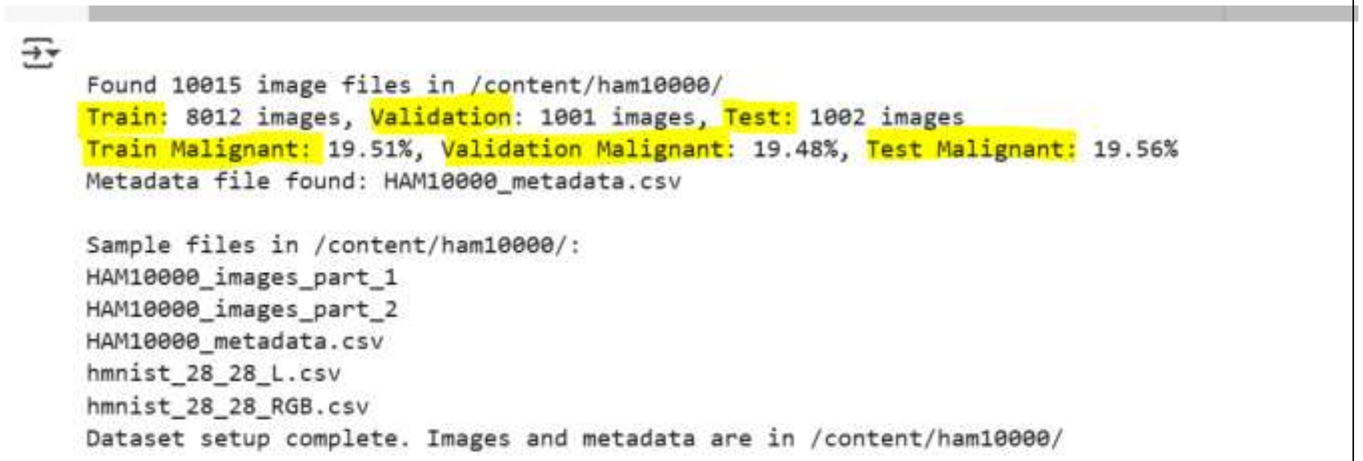
Creating a column in the data frame with name “malignant” containing values as 0 or 1 by conditionally checking from the **malignant_classes** (A list of diagnoses that are malignant (cancerous)

Splitting Data into Training, Validation, Evaluation:



```
# Split data: 80% train, 10% validation, 10% test
train_df, temp_df = train_test_split(df, test_size=0.2, stratify=df['malignant'], random_state=42)
val_df, test_df = train_test_split(temp_df, test_size=0.5, stratify=temp_df['malignant'], random_state=42)
```

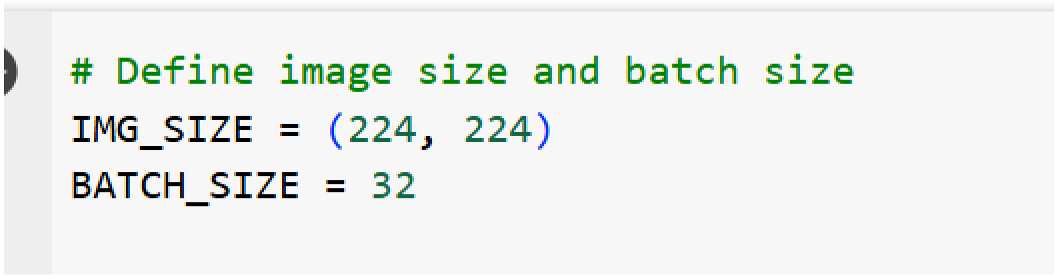
stratify=df['malignant']: Ensures the splits have the same proportion of Malignant and Benign cases as the original dataset (e.g., ~27% Malignant in all splits).



```
Found 10015 image files in /content/ham10000/
Train: 8012 images, Validation: 1001 images, Test: 1002 images
Train Malignant: 19.51%, Validation Malignant: 19.48%, Test Malignant: 19.56%
Metadata file found: HAM10000_metadata.csv

Sample files in /content/ham10000/:
HAM10000_images_part_1
HAM10000_images_part_2
HAM10000_metadata.csv
hmnist_28_28_L.csv
hmnist_28_28_RGB.csv
Dataset setup complete. Images and metadata are in /content/ham10000/
```

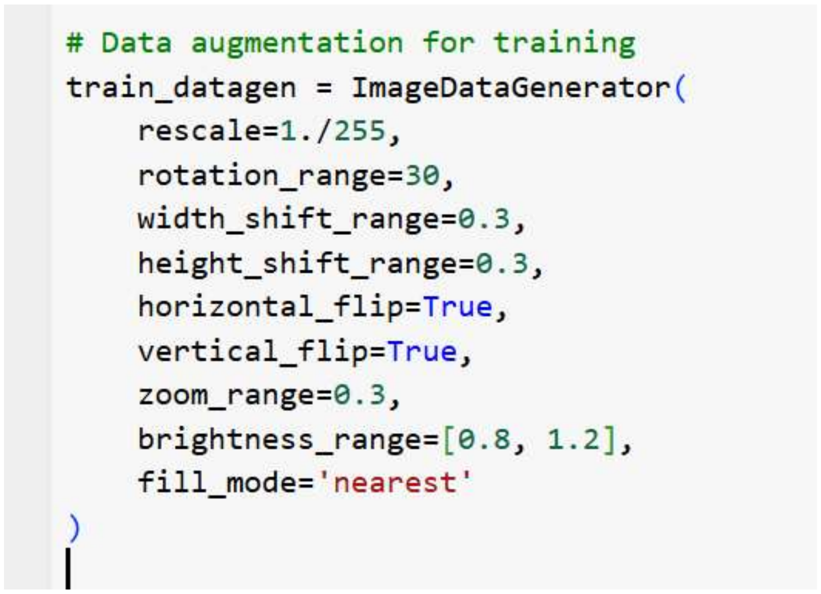
Setting Up the Data Generators:

Prepares the images for training by resizing them, applying transformations (augmentation), and organizing them into batches for the model to process



```
# Define image size and batch size
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
```

224 x 224 pixels images processing 32 in a batch (number of images the model process in once)



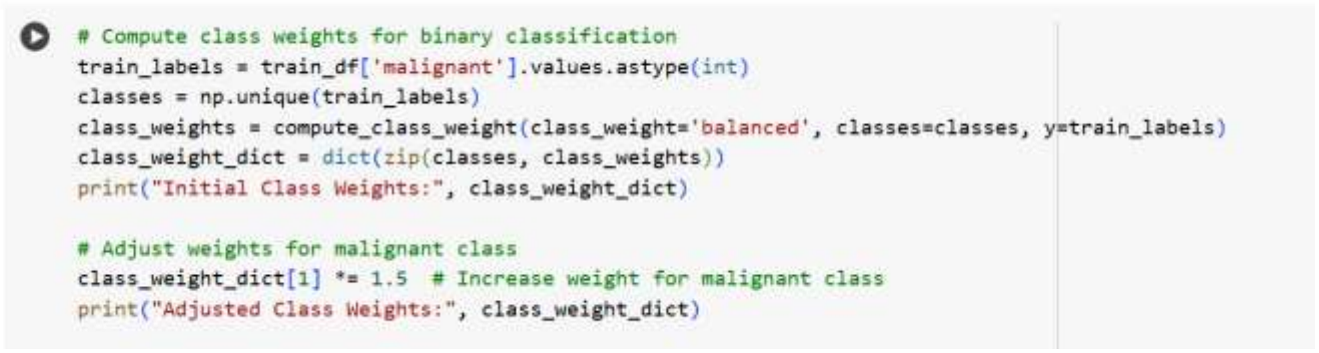
```
# Data augmentation for training
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.3,
    height_shift_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=0.3,
    brightness_range=[0.8, 1.2],
    fill_mode='nearest'
)
```

- **rescale**=1./255: Converts pixel values from 0–255 to 0–1 (e.g., 255 becomes 1.0), which the model prefers.
- **rotation_range**=30: Rotates images up to 30 degrees.
- **width_shift_range**=0.3, height_shift_range=0.3: Shifts images left/right or up/down by up to 30%.
- **horizontal_flip**=True,
- **vertical_flip**=True: Flips images horizontally or vertically (like mirroring).
- **zoom_range**=0.3: Zooms in or out by up to 30%.

- **brightness_range**=[0.8, 1.2]: Changes brightness (makes images 80% darker to 120% brighter).
- **fill_mode**='nearest': Fills empty spaces (after rotation/shift) with nearby pixel values.

Defining Model and Computing Class Weights:

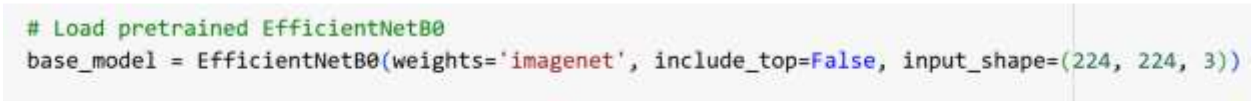
Builds the machine learning model using EfficientNetB0 and adjusts it to pay more attention to Malignant cases.



```
# Compute class weights for binary classification
train_labels = train_df['malignant'].values.astype(int)
classes = np.unique(train_labels)
class_weights = compute_class_weight(class_weight='balanced', classes=classes, y=train_labels)
class_weight_dict = dict(zip(classes, class_weights))
print("Initial Class Weights:", class_weight_dict)

# Adjust weights for malignant class
class_weight_dict[1] *= 1.5  # Increase weight for malignant class
print("Adjusted Class Weights:", class_weight_dict)
```

Increases the weight for Malignant (1) by 1.5x (e.g., from 1.85 to 2.78) to make the model focus even more on detecting Malignant cases, since missing a cancer is more dangerous than missing a benign lesion.

Loading Pretrained Model



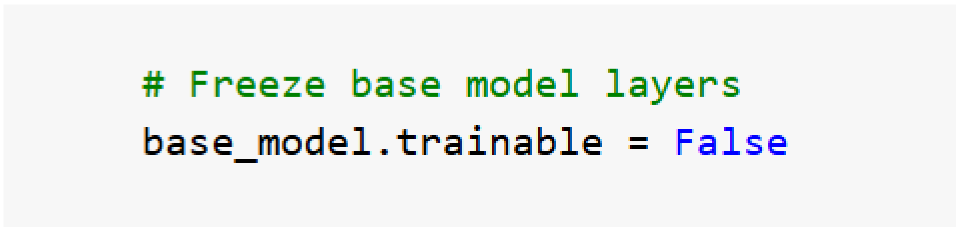
```
# Load pretrained EfficientNetB0
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

A pre-trained model (like a pre-built robot) that has already learned to recognize patterns in images (trained on a huge dataset called ImageNet).

Using the pretrained weights from Imagenet

include_top=False: Removes the final layer of EfficientNetB0 because we’ll add our own for binary classification.

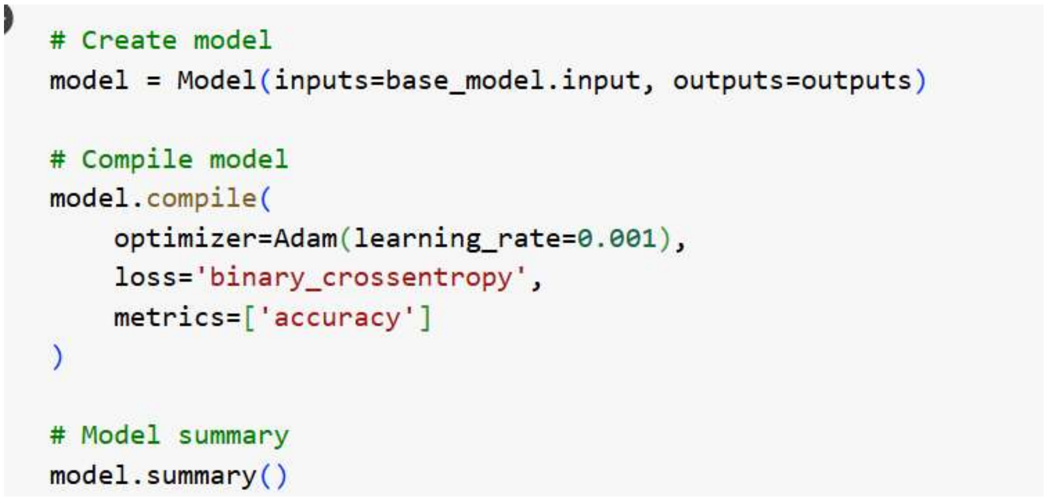
We have freeze the pretrained layers so that we they do not change during the training



```
# Freeze base model layers
base_model.trainable = False
```

Combining the Model:

Combines the pre-trained base model and our new layers into a single model.



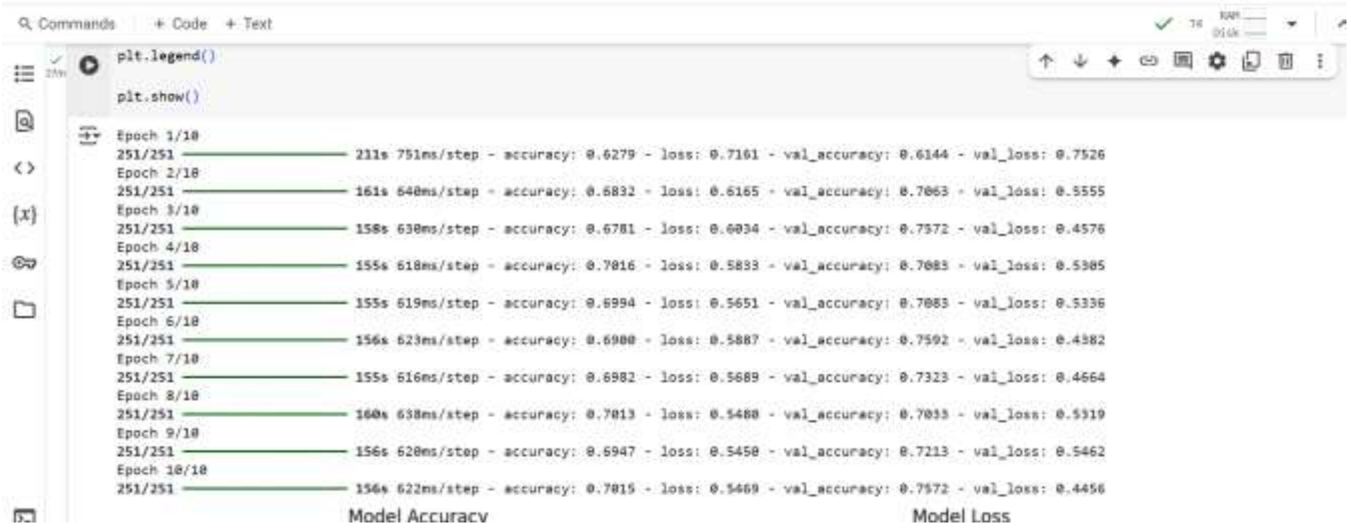
```
# Create model
model = Model(inputs=base_model.input, outputs=outputs)

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

# Model summary
model.summary()
```

- **optimizer**=Adam: Uses the Adam optimization method to adjust the model's weights during training (like a smart teacher guiding the learning process).
- **learning_rate**=0.001: Controls how fast the model learns (small steps to avoid mistakes).
- **loss**='binary_crossentropy': The error function for binary classification (measures how far the model's predictions are from the true labels).
- **metrics**=['accuracy']: Tracks accuracy during training (percentage of correct predictions).

Training the Model:

```python
# Train the model
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator,
    class_weight=class_weight_dict
)

# Plot training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

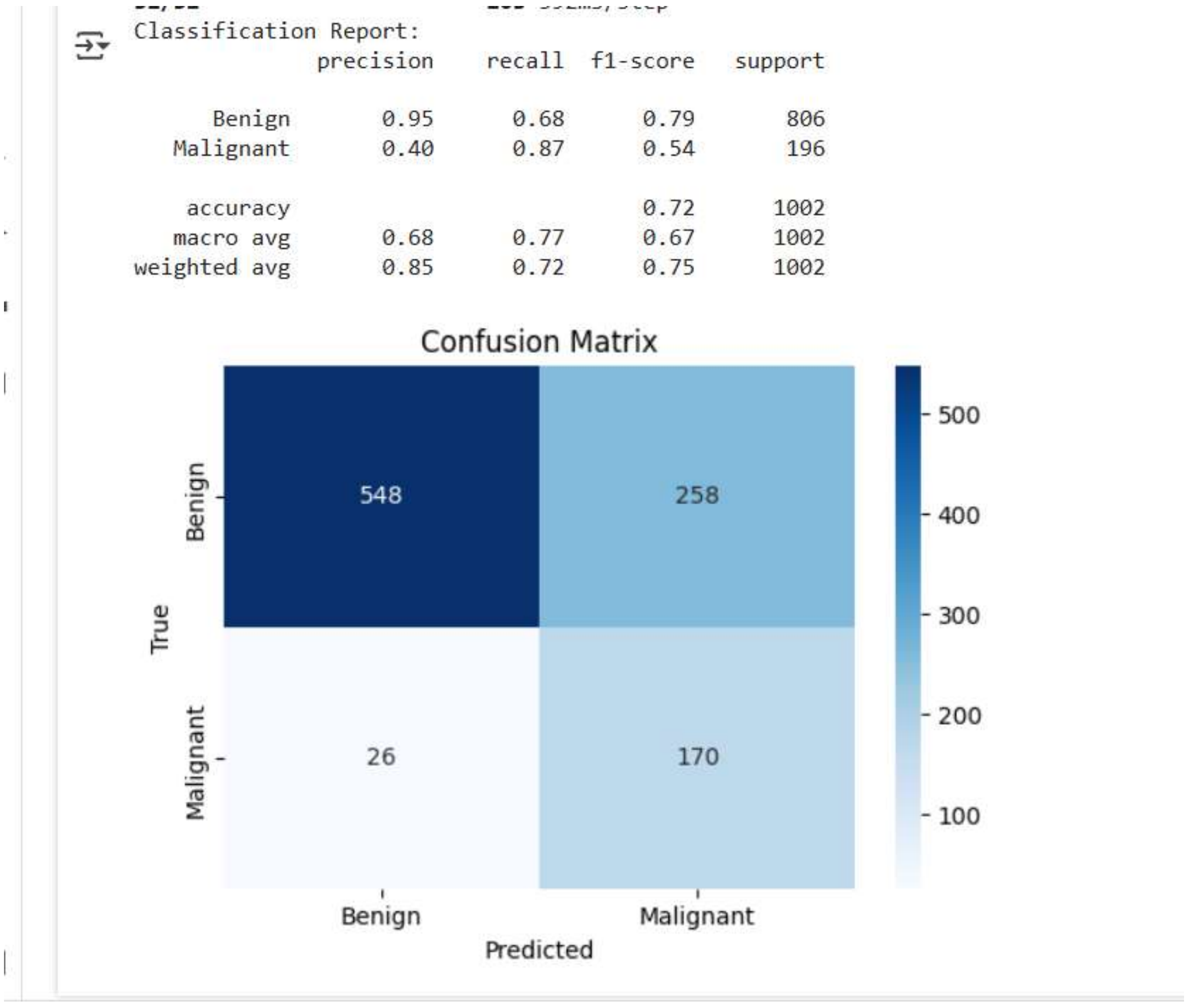
epochs=10: The model will go through the entire training data 10 times (like studying the same book 10 times to learn better).



```
plt.legend()

plt.show()
```

```
Epoch 1/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 211s 751ms/step - accuracy: 0.6279 - loss: 0.7161 - val_accuracy: 0.6144 - val_loss: 0.7526
Epoch 2/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 161s 640ms/step - accuracy: 0.6832 - loss: 0.6165 - val_accuracy: 0.7063 - val_loss: 0.5555
Epoch 3/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 158s 630ms/step - accuracy: 0.6781 - loss: 0.6034 - val_accuracy: 0.7572 - val_loss: 0.4576
Epoch 4/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 155s 618ms/step - accuracy: 0.7016 - loss: 0.5833 - val_accuracy: 0.7083 - val_loss: 0.5305
Epoch 5/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 155s 619ms/step - accuracy: 0.6994 - loss: 0.5651 - val_accuracy: 0.7083 - val_loss: 0.5336
Epoch 6/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 156s 623ms/step - accuracy: 0.6900 - loss: 0.5887 - val_accuracy: 0.7592 - val_loss: 0.4382
Epoch 7/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 155s 616ms/step - accuracy: 0.6982 - loss: 0.5689 - val_accuracy: 0.7323 - val_loss: 0.4664
Epoch 8/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 160s 638ms/step - accuracy: 0.7013 - loss: 0.5480 - val_accuracy: 0.7033 - val_loss: 0.5319
Epoch 9/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 156s 620ms/step - accuracy: 0.6947 - loss: 0.5450 - val_accuracy: 0.7213 - val_loss: 0.5462
Epoch 10/10
251/251 ━━━━━━━━━━━━━━━━━━━━ 156s 622ms/step - accuracy: 0.7015 - loss: 0.5469 - val_accuracy: 0.7572 - val_loss: 0.4456
```

Evaluating the Model on the Test Data:

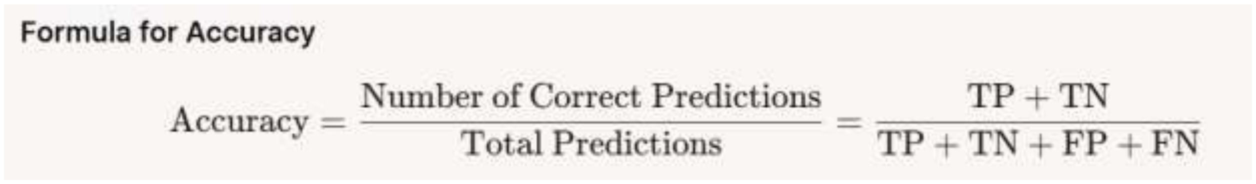


```
Classification Report:
              precision    recall  f1-score   support

      Benign       0.95      0.68      0.79       806
   Malignant       0.40      0.87      0.54       196

    accuracy                           0.72      1002
   macro avg       0.68      0.77      0.67      1002
weighted avg       0.85      0.72      0.75      1002
```

- **True Benign, Predicted Benign (548):**
  548 images that are actually Benign were correctly predicted as Benign (True Negatives, TN).
- **True Benign, Predicted Malignant (258):**
  258 images that are actually Benign were incorrectly predicted as Malignant (False Positives, FP).
- **True Malignant, Predicted Benign (26):**
  26 images that are actually Malignant were incorrectly predicted as Benign (False Negatives, FN).
- **True Malignant, Predicted Malignant (170)**:
  170 images that are actually Malignant were correctly predicted as Malignant (True Positives, TP).

Overall Accuracy:

Overall Accuracy is the proportion of correct predictions (both True Positives and True Negatives) out of the total number of predictions.



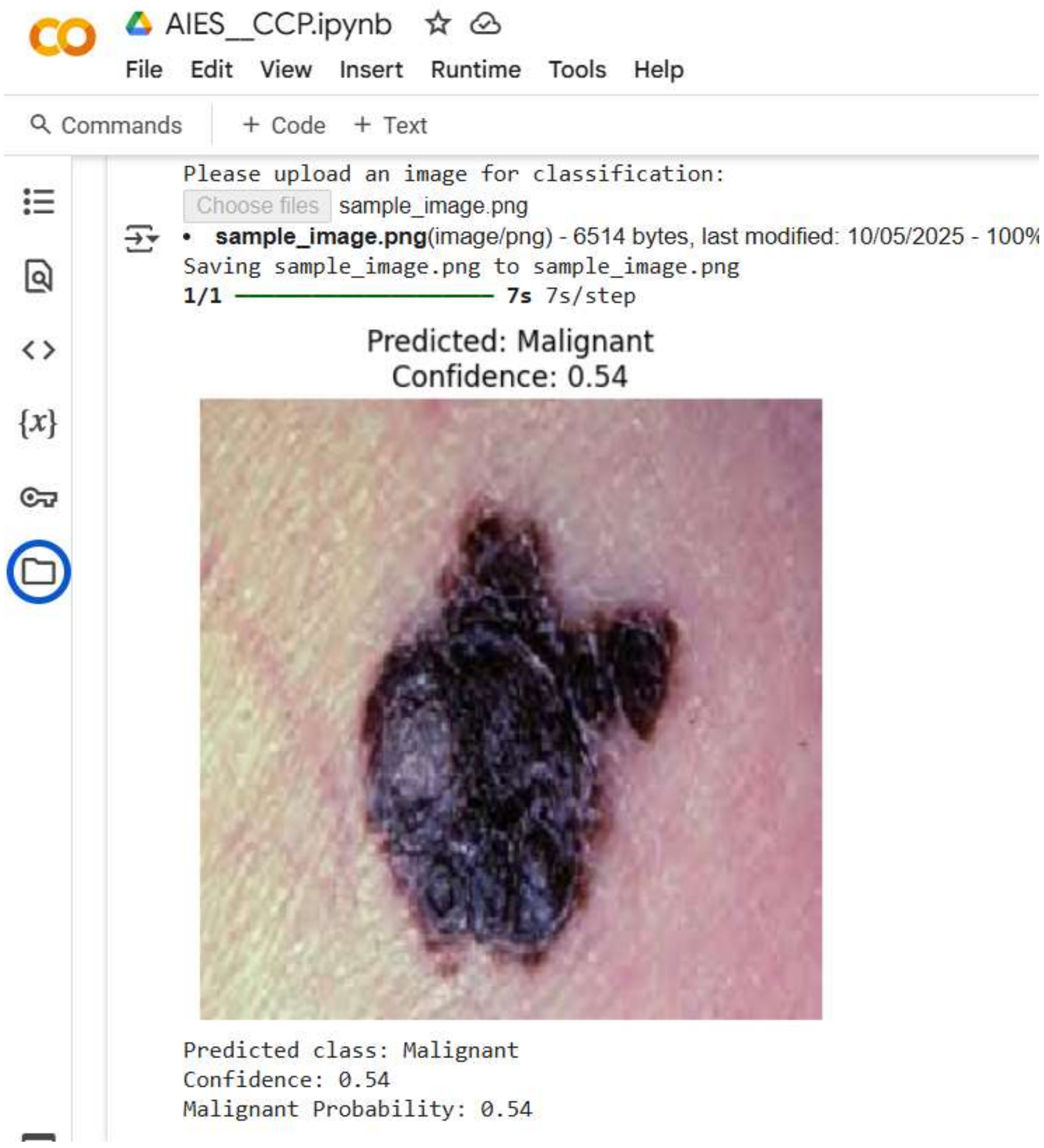
Formula for Accuracy

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **True Negatives (TN):** 548 (Benign correctly predicted as Benign).
- **False Positives (FP):** 258 (Benign incorrectly predicted as Malignant).
- **False Negatives (FN):** 26 (Malignant incorrectly predicted as Benign).
- **True Positives (TP):** 170 (Malignant correctly predicted as Malignant).

Accuracy = 718 / 1002 ≈ 0.7166 ≈ 71.66 %

INPUT IMAGE CLASSIFICATION:



```
Please upload an image for classification:
Choose files sample_image.png
• sample_image.png(image/png) - 6514 bytes, last modified: 10/05/2025 - 100%
Saving sample_image.png to sample_image.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 7s 7s/step
```

Predicted: Malignant
Confidence: 0.54

```
Predicted class: Malignant
Confidence: 0.54
Malignant Probability: 0.54
```

FUTURE MODEL IMPROVEMENTS:

Reduce False Positives (Improve Malignant Precision):

- **Adjust Class Weights**: Decrease the weight for Malignant (e.g., from 3.844 to 2.5) to balance the focus, reducing over-prediction of Malignant cases.

Reduce False Negatives (Improve Malignant Recall Further):

- **Fine-Tune the Model**: unfreeze more layers of EfficientNetB0 and train with a lower learning rate (e.g., 0.0001) for 5–10 more epochs.
- **Oversampling**: Increase the number of Malignant samples in the training set using techniques like SMOTE or duplicating Malignant images with augmentation.

Address Class Imbalance:

- The imbalance (806 Benign vs. 196 Malignant) affects performance. We can Consider using more advanced techniques like weighted loss functions or resampling in the data generator.

Evaluate with Medical Metrics:

- Focus on sensitivity (recall for Malignant) and specificity (recall for Benign) to align with medical priorities. Current sensitivity is 0.87, and specificity is 0.68—aim to improve both.

CONCLUSION:

This project helps us to understand application of Deep Learning models and to use them in real world problems, However the accuracy can be increased by working on the balanced datasets and fine tune certain parameters.