* DATASET
* LFW:

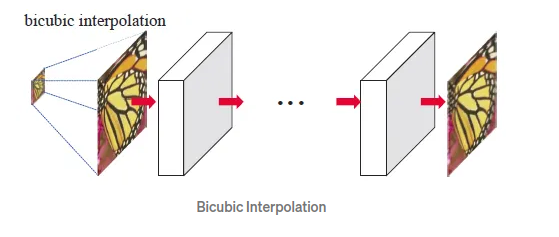
The LFW dataset, also known as the Labeled Faces in the Wild dataset, is a popular benchmark dataset for face recognition tasks. It consists of more than 13,000 images of faces collected from the internet, with significant variations in pose, lighting, expression, and background. The LFW dataset has images of over 5,000 individuals, with each represented by an average of 2.5.

* MODELS
* Super Resolution Models:

Super resolution models are deep learning models designed to enhance the resolution and quality of low-resolution images. They achieve this by learning to map a low-resolution input image to a high-resolution output image, using a combination of convolutional and deconvolutional layers.

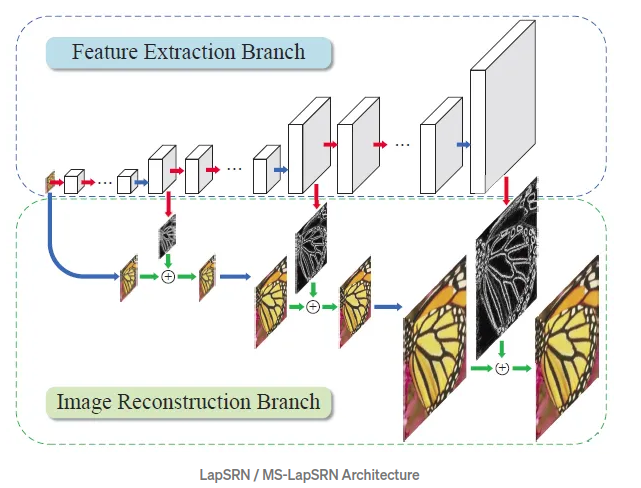
1. BICUBIC:

Bicubic super resolution is a simple and widely used method for image upscaling, which can be considered as a basic super resolution model. It is a mathematical interpolation method that uses a cubic function to interpolate the pixel values of a low-resolution image to create a higher-resolution version. The post-processing step can be implemented using various techniques, such as denoising, sharpening, or deep learning-based methods. Deep learning-based methods can be used to learn the residual image between the bicubic up sampled image and the ground truth high-resolution image, and then refine the image using this residual information. Despite its simplicity, bicubic super resolution can produce satisfactory results for moderate scaling factors, such as 2x or 3x. However, it can struggle with more extreme scaling factors, such as 4x or higher, and can produce images with blurred or jagged edges.



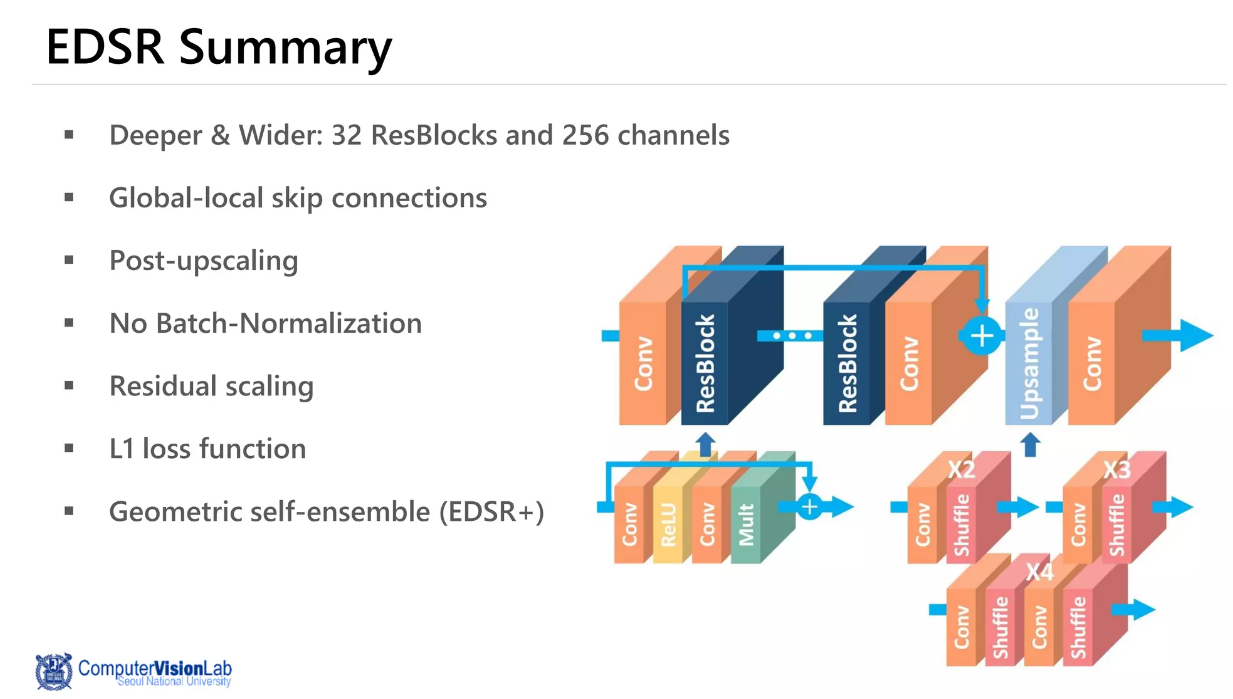
1. LAPSRN:

LAPSRN stands for Laplacian Pyramid Super-Resolution Network, is a deep learning-based super resolution model. It is designed to enhance the resolution and quality of low-resolution images using a combination of a Laplacian pyramid and a deep neural network. The Laplacian pyramid is a multi-scale image decomposition technique that decomposes an image into a set of band-pass filtered sub-images, each being a different frequency band. The idea behind LAPSRN is to use the Laplacian pyramid to divide the low-resolution input image into multiple sub-images, and then use a deep neural network to learn a mapping from each sub-image to a corresponding high-resolution sub-image. LAPSRN consists of three main components: a feature extraction network, a Laplacian pyramid decomposition module, and a reconstruction network. The feature extraction network extracts low-level and high-level features from the input image, which are then fed into the Laplacian pyramid decomposition module. The Laplacian pyramid decomposition module decomposes the input image into multiple sub-images and applies the reconstruction network to each sub-image to generate a corresponding high-resolution sub-image. The high-resolution sub-images are then combined to generate the final high-resolution output image.



1. EDSRN:

EDSRN, which stands for Enhanced Deep Super-Resolution Network, is a deep learning-based super resolution model designed to enhance the resolution and quality of low-resolution images using a deep neural network with enhanced features. The EDSRN model consists of a series of residual blocks, composed of convolutional layers, batch normalization layers, and activation functions. The residual blocks allow the model to learn residual features, which are the differences between the low-resolution input image and the corresponding high-resolution ground truth image. The model also uses skip connections, which enable the network to preserve low-level details in the image. EDSRN also uses a "multi-scale approach," which means that the model learns to generate super resolved images at different scales, allowing it to better handle images with various levels of detail.



* Face Detection Model:

Face recognition is a computer vision task that involves finding and verifying the identity of individuals in digital images or video frames. The goal of face recognition is to match the detected face with a known identity, either from a database or from earlier observations. There are several approaches to face recognition, including traditional computer vision techniques and deep learning-based methods. Traditional computer vision techniques typically involve extracting facial features, such as the distance between the eyes, the shape of the nose, and the curve of the lips, and then using these features to compare and match faces. Deep learning-based methods, on the other hand, involve training a deep neural network on a large dataset of images with labeled identities, and then using this trained network to find and verify faces in new images or video frames. These methods can also use features extracted from different layers of the neural network to achieve better accuracy and robustness.

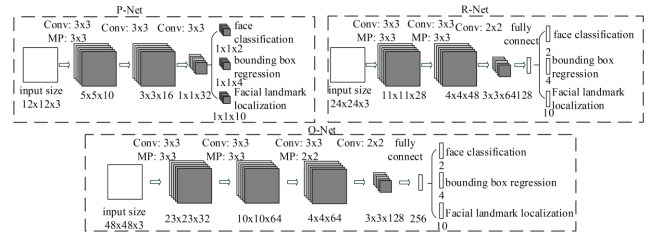
1. MTCNN:

MTCNN, which stands for Multi-Task Cascaded Convolutional Neural Network, is a deep learning-based face detection model. The MTCNN model is designed to detect faces in images with varying poses, orientations, and lighting conditions.

The MTCNN model consists of three stages, each of which performs a specific task in the face detection process:

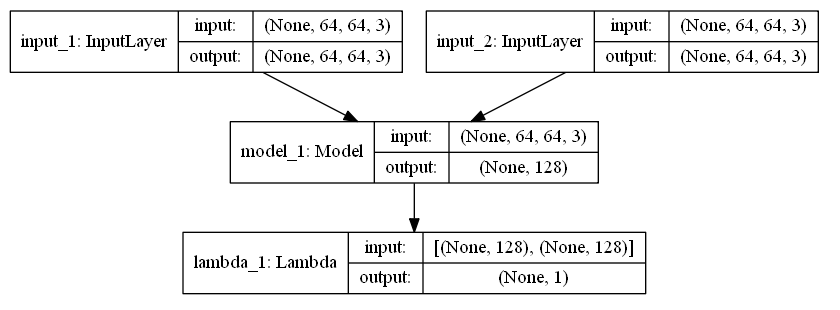
1. The first stage uses a convolutional neural network (CNN) to generate candidate face regions in the image. This stage is designed to be extremely fast, so it can generate many candidate regions in a short amount of time.
2. The second stage uses another CNN to refine the candidate face regions generated in the first stage, by adjusting their position and size to better fit the face. This stage also drops any candidate regions that are not likely to have a face.
3. The third stage uses CNN to find the facial landmarks, such as the position of the eyes, nose, and mouth, in the remaining candidate regions. This stage also classifies the candidate regions as either a face or a non-face.

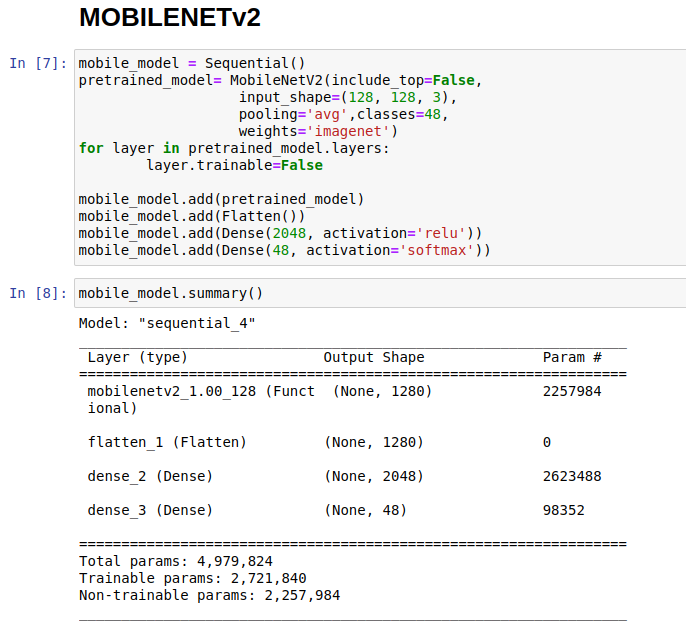
By using a cascaded architecture, where each stage's output is used as input to the next stage, the MTCNN model can progressively refine the candidate face regions and improve the accuracy of the face detection.



1. MoblieNetV2:

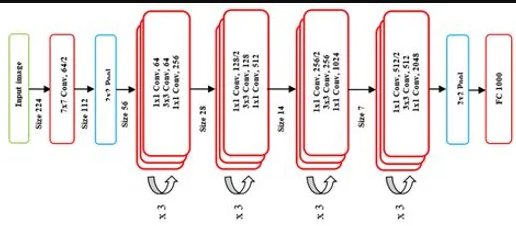
MobileNetv2 is a neural network architecture designed for mobile and embedded devices that require low computational resources and memory footprint. MobileNetv2 is built using a combination of depth wise separable convolutions and linear bottleneck layers. The linear bottleneck layers are used to reduce the dimensionality of the features, which helps to reduce the computational cost of the model. They consist of a 1x1 convolutional layer followed by a 3x3 depth wise separable convolutional layer and another 1x1 convolutional layer. MobileNetv2 also introduces a novel mechanism called inverted residuals, which involves increasing the number of filters in the bottleneck layers and decreasing them in the next 1x1 convolutional layers. This helps to keep a balance between the number of parameters and the computational efficiency of the model. MobileNetv2 has been shown to achieve ultramodern accuracy on various image classification and object detection benchmarks while requiring fewer parameters and less computational resources than other popular neural network architectures.

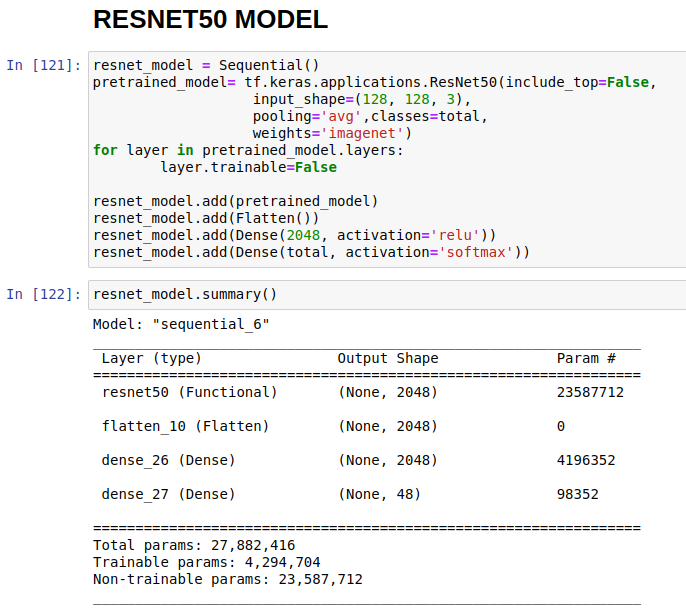




1. ResNet50:

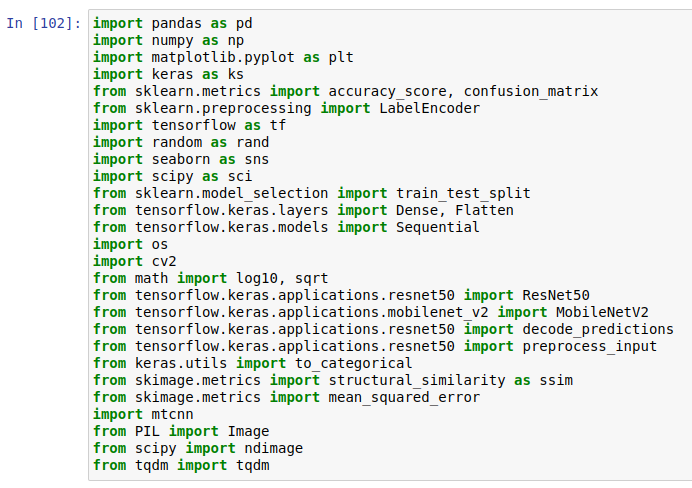
ResNet50 is part of the ResNet family of models, which are designed to address the problem of vanishing gradients in deep neural networks. ResNet50 has 50 layers and uses residual connections to improve the performance of the model. A residual connection is a shortcut connection that skips over one or more layers of the network, allowing the model to bypass certain computations and keep valuable information from earlier layers. The architecture of ResNet50 consists of several blocks of layers, each of which holds a combination of convolutional layers, batch normalization layers, activation functions, and residual connections. The blocks are designed to progressively reduce the spatial dimensions of the features while increasing the number of filters, allowing the model to capture increasingly complex and abstract features.





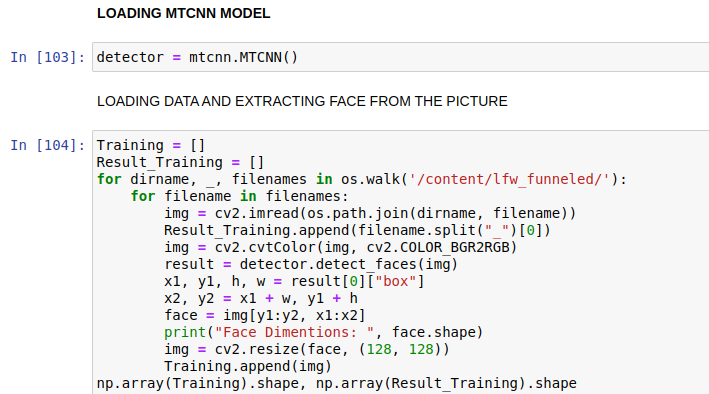
* CODE EXPAINATION
* Libraries:

Importing important libraries some are simple like pandas, NumPy and TensorFlow. While others are for just the little purpose. These libraries will have our models and will help us in completing the model, i.e., evaluation. All the libraries are shown in figure:



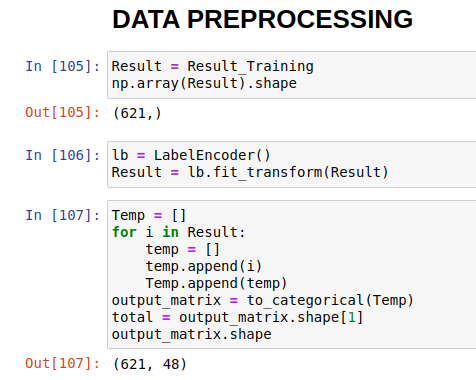
* DATASET:

Each single file is read by extracting the face by MTCNN model which just reads the file one by one, and model extracts the facial region and is then the cropped image of the human face is saved, and it will be used to process data.



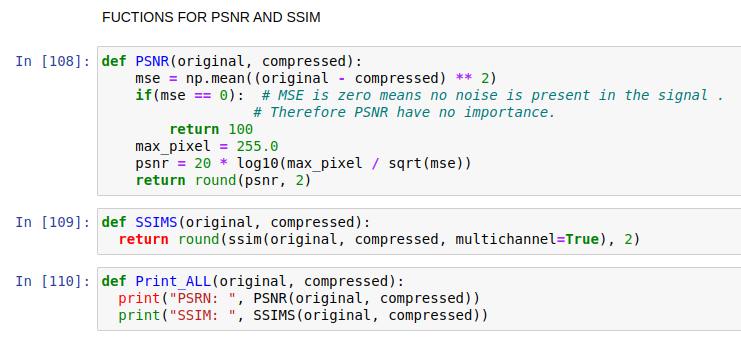
* Preprocessing:

The next step is simple. The labels are encoded from name to an integer and then they are one-hot-encoded so that we can easily process them and can train our model.

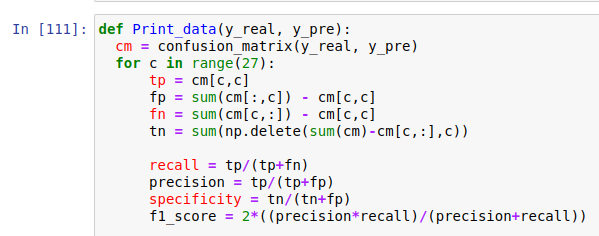


* Evaluation Functions:

SSIM and PSNR are useful metrics for measuring the quality of compressed or processed images or videos, but they should be used in conjunction with other metrics and subjective evaluation to supply a more complete picture of the quality of the output.

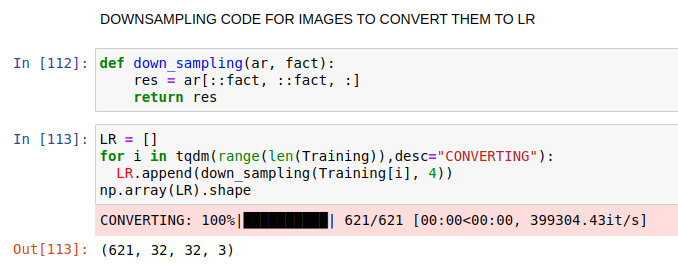


Recall, precision, and F1 score are all important metrics for evaluating the performance of a classification model. However, the relative importance of each metric may depend on the specific application and the costs associated with diverse types of classification errors.



* Down sampling:

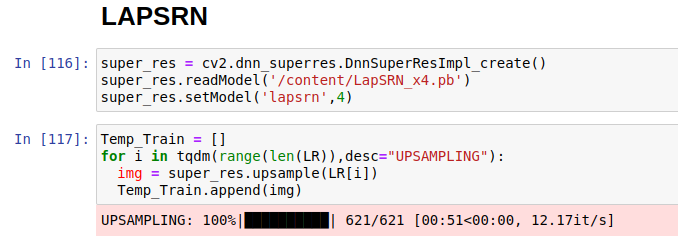
Down sampling, also known as down sampling or decimation, is the process of reducing the resolution or size of an image, signal, or dataset by removing some of the data points. Down sampling is often used to reduce the computational complexity of an algorithm or to make data easier to handle, store or transmit.

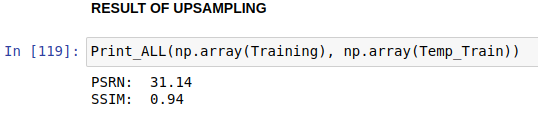


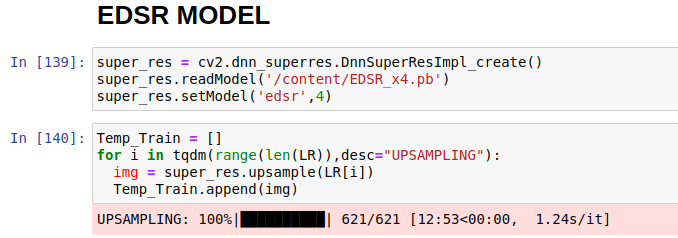
* Up sampling:

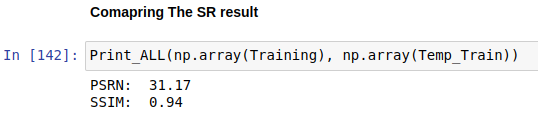
Up sampling, also known as interpolation, is the process of increasing the resolution or size of an image, signal, or dataset by adding more data points. Up sampling is often used to recover lost information or to improve the quality of the data. In image processing, up sampling refers to increasing the number of pixels in an image. This can be done by either increasing the resolution of the image or by interpolation, which involves estimating new pixel values based on the existing ones. For example, a 512x512 image can be up sampled to a 1024x1024 image by inserting new pixels between the existing ones and interpolating their values.

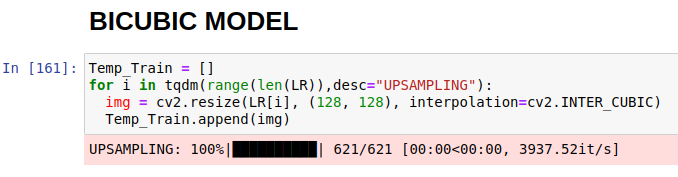
Models that were used are:

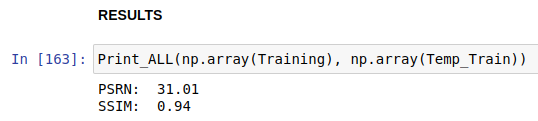






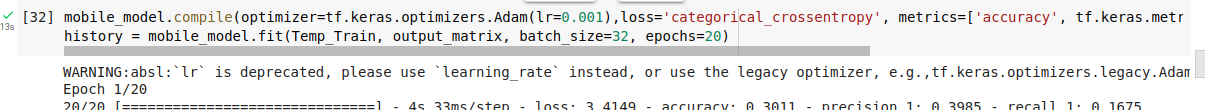






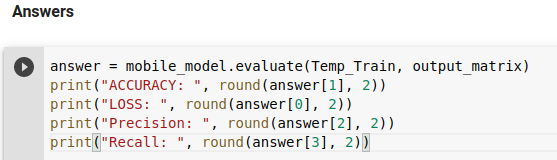
* Training:

These models are compiled using the **Adam** optimizer with a learning rate of 0.001, and the loss function is set to **categorical\_crossentropy**, which is commonly used for multi-class classification problems. In addition to accuracy, the **compile** method specifies two additional metrics to evaluate the model during training: **Precision** and **Recall**. These metrics are useful for evaluating classification models, as they provide additional insights into the model's performance beyond simple accuracy. The **fit** method is used to train the model using the **Temp\_Train** data and **output\_matrix** labels, with a batch size of 32 and for a total of 20 epochs. Overall, this code is training a deep learning model for a multi-class classification problem, using a common optimizer and loss function, and specifying additional metrics for evaluation during training.



* Evaluation:

These models are then evaluated by performance on the same data used for training. The **evaluate** method takes the input data (**Temp\_Train**) and the corresponding labels (**output\_matrix**) as input and returns a list of metrics calculated by the model, including the loss function and any specified metrics during compilation. The code then prints the accuracy, loss, precision, and recall metrics calculated by the **evaluate** method.



* Results:

The results are already shared in an excel file where each combination is applied and a result of 10 iterations for each combination is given.