**Weather Image Recognition**

**CS - 5661 Data Science**

**Project Report**

**Team Members –**

1. Venkateswara Rohit Roy Amarthaluru
2. Rohith Surya Podugu
3. Ganesh Suhas Shinde
4. Shubham Deepak Pawar
5. Tanmay Sure
6. Yaswanth Madanapalli
7. Sagar Addala Ram
8. Keerthana Veena

**Responsibilities -**

1. Rohith Surya Podugu:

* Implemented and researched about VGG 16 and helped tune hyper parameters.
* Implemented the confusion matrix and ROC curve implementation One vs Rest for VGG 16 Algorithm.
* Added documentation for ReLU and softmax function in project report.

2. Venkateswara Rohit Roy Amarthaluru:

* Researched on ResNet152V2.
* Contributed to report.

3. Ganesh Suhas Shinde:

* Researched & Implemented ResNet50(weather-image-classification-resnet50.ipynb) & MobileNet(weather-image-mobilenet.ipynb) algorithm.
* Completely implemented the weather classification from the accuracy till giving Input the model & classifying the class of the input image.
* Contributed in the Report, PowerPoint publication & presentation of the project.

4. Shubham Deepak Pawar:

* Implemented and researched on EfficientNetB0 (efficientnetb0.ipynb) and VGG19 (vgg19.ipynb).
* Contributed to the presentation and report publication.

5. Tanmay Sure:

* Implemented XceptionNet Algorithm.

6. Yaswanth Madanapalli:

* Researched on FastAI and LeNet.
* Contributed to the Report and PowerPoint presentation.

7. Sagar Addala Ram:

* Researched on VGG16 model.
* Contributed to the PowerPoint Presentation and report.

8. Keerthana Veena:

* Implemented and researched VGG16.
* Added documentation for Project Description and Project Goals in the report apart from VGG16 report and PowerPoint Publication.

**Abstract:**

In this project we applied machine learning algorithms to the real-world problem of recognizing weather from an image. This will help us in better identification of the weather just from the given image.

**Dataset:**

We have taken the dataset from Kaggle.

[Dataset\_Link](https://www.kaggle.com/datasets/jehanbhathena/weather-dataset)

The dataset contains 11 classes. These classes represent different weather patterns.

The 11 classes are as follow:

1. Dew
2. Fog smog
3. Frost
4. Glaze
5. Hail
6. Lightning
7. Rain
8. Rainbow
9. Rime
10. Sandstorm
11. Snow

The Dataset consists of a total of 6862 images.

The Data is distributed amongst 11 classes as mentioned below:

|  |  |
| --- | --- |
| **Class** | **Image Count** |
| Hail | 591 |
| Rainbow | 232 |
| Frost | 475 |
| Rime | 1160 |
| Fogsmog | 851 |
| Snow | 621 |
| Rain | 526 |
| Glaze | 639 |
| Lightning | 377 |
| Sandstorm | 692 |
| Dew | 698 |



**Project Description:**

Weather Image Recognition is a project that uses machine learning techniques to recognize and predict weather conditions from images. The project can be used to classify an image into different weather conditions, such as sunny, cloudy, rainy, snowy, etc. Given an image of a particular weather condition, the model can predict what type of weather it is.

In precise, Weather Image Recognition project is an exciting and useful application of computer vision and deep learning techniques to predict the weather from images.

**Project Goals:**

Develop an accurate image recognition model: The primary goal of the project is to build an accurate image recognition model that can predict the weather condition of a given image with high accuracy. This requires collecting and preparing a high-quality dataset, selecting an appropriate deep learning framework and architecture, and optimizing the model to achieve the highest possible accuracy.

Overall, the goal of Weather Image Recognition is to leverage the power of deep learning and computer vision to predict weather conditions from images and provide valuable insights for weather forecasting and analysis. The project has the potential to make a significant impact on many industries and applications, and it is an exciting area of research and development for the future.

**Objectives:**

To predict the given image with the most accurate class amongst the 11 classes.

**Technologies and Libraries:**

* Python
* Machine Learning
* Numpy, Pandas, Sklearn, Tensorflow, Keras

**Activation Functions Used**

**SoftMax Activation Function:**

SoftMax is an activation function that scales numbers into probabilities. The output of a SoftMax is a vector (say v) with possibilities of each possible outcome—the probabilities in vector v sum to one for all possible outcomes or classes.

Softmax is generally used at the last layer for classification problems, where it's used to identify which class the neural network predicts the Input to be. For example, we are expecting the breed of the ten dogs; the output values of softmax tell which probability the neural network predicts is the breed.

**ReLU Activation Function:**

ReLU stands for the Rectified linear unit. It is a linear function that outputs the Input directly if it is positive or else it outputs zero.

a = max(0,z)



**Implemented Algorithms:**

1. Xception-Net
2. EfficientNetB0
3. VGG19
4. VGG16
5. MobileNet
6. ResNet50
7. **Xception-Net :**

Xception is a deep convolutional neural network architecture that was proposed by François Chollet in 2016. It is inspired by the Inception architecture, but instead of using standard convolutions, it uses depthwise separable convolutions.

Depthwise separable convolutions are a type of convolution that factorizes a standard convolution into two simpler operations: a depthwise convolution and a pointwise convolution. This results in a much more efficient and parameter-efficient architecture than standard convolutions, making it faster to train and easier to optimize.

Xception has been shown to achieve state-of-the-art performance on several image classification benchmarks, such as the ImageNet dataset. It has also been used as a feature extractor for a variety of computer vision tasks, including object detection, segmentation, and transfer learning.

Overall, Xception is a powerful and efficient deep learning architecture that has proven to be highly effective for a wide range of computer vision applications.

1. **EfficientNetB0:**

* EfficientNet-b0 is a convolutional neural network that is trained on more than a million images from the ImageNet database.
* It can be imported as - tf.keras.applications.efficientnet.EfficientNetB0
* The suitable image resolution for EfficientNetB0 is 224 X 224
* We need to manually set – { include\_top : false }.

1. **VGG19:**

* VGG-19 is a convolutional neural network that is 19 layers deep
* It can be imported as - tensorflow.keras.applications.vgg19
* We have kept the default resolution for image – 224 X 224
* Preprocessing is being done using tensorflow.keras.applications.vgg19.preprocess\_input

1. **VGG16:**



**Fig: Architecture of VGG 16**

VGG16 is a deep convolutional neural network architecture that was first introduced by researchers at the Visual Geometry Group (VGG) at the University of Oxford in 2014. It is one of the most popular and widely used CNN architectures for image classification tasks and has achieved outstanding performance on various benchmark datasets.

The VGG16 architecture is composed of 13 convolutional layers and 3 fully connected layers. The convolutional layers are designed to extract hierarchical features from the input image, while the fully connected layers act as a classifier that produces the final output. One of the distinctive features of VGG16 is its use of 3x3 convolutional filters throughout the network, which allows it to learn more intricate and fine-grained features.

VGG16 has been pre-trained on the ImageNet dataset, which consists of millions of images across thousands of categories. This pre-training has enabled the model to learn a wide range of features that are useful for image classification tasks. VGG16 can be used as a feature extractor or fine-tuned for specific tasks such as object detection, segmentation, and transfer learning.

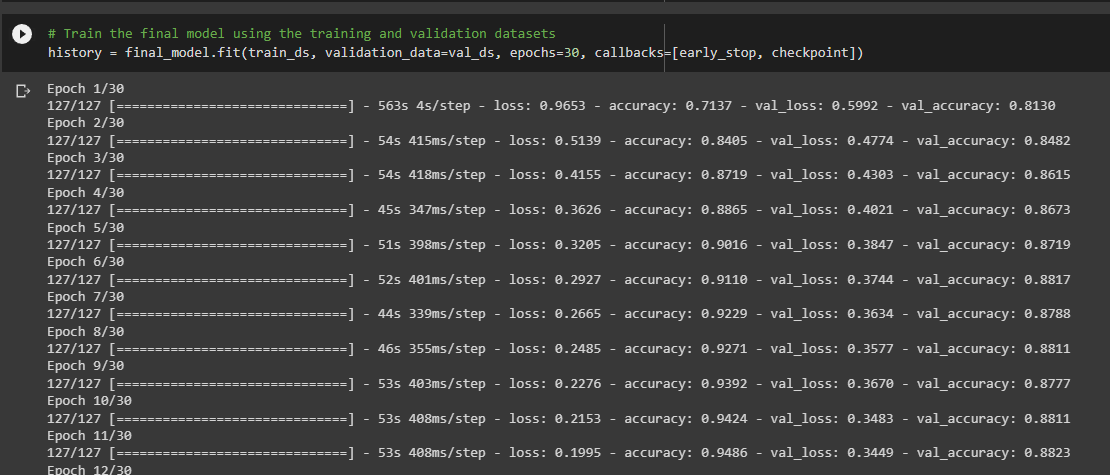
**Implementation Details and Code**

1. **Xception-Net:**

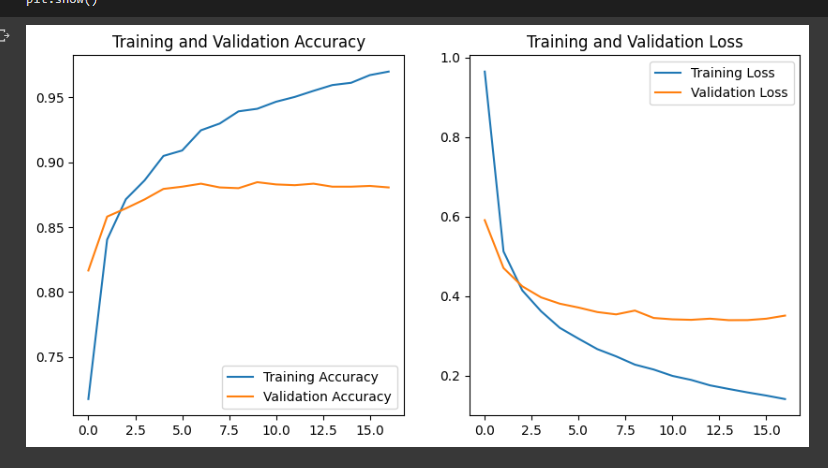
The training process was carried out in two stages. In the first stage, the upper layers of the Xception model were trained while keeping the lower layers fixed. This approach is known as transfer learning and is often used when working with deep neural networks. The lower layers of the model are already trained to detect low-level features, such as edges and corners, and can be reused for other tasks.

In the second stage, the entire Xception model was fine-tuned by training all the layers of the model. The fine-tuning process involved updating the weights of the model using the backpropagation algorithm.

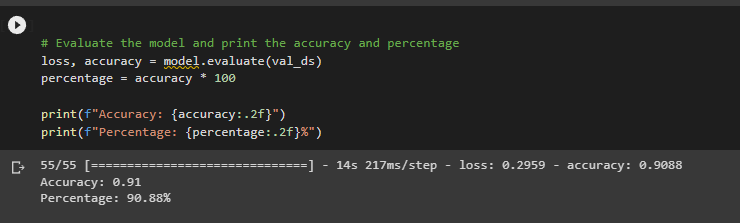
**Fitting the model:**

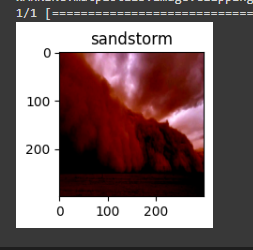
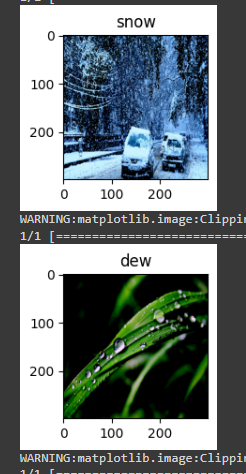


**Training And Validation Accuracy/Loss vs Epochs**

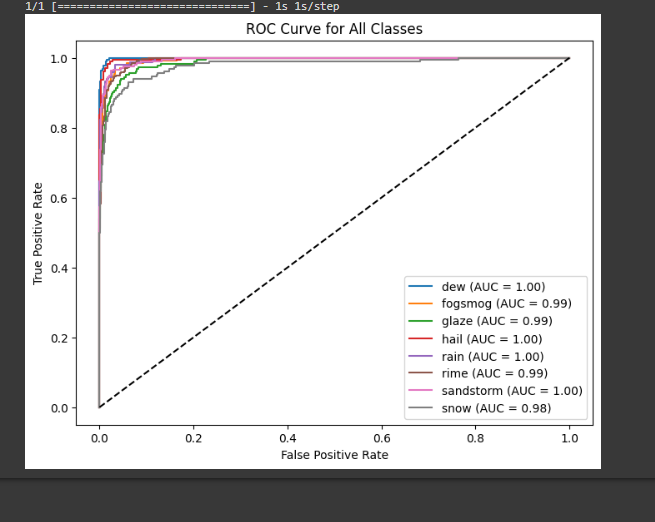


**Accuracy and Prediction:**  
The Xception model achieved an accuracy of 90.88% on the validation set of the weather recognition dataset. The training and validation accuracy/loss curves show that the model was able to achieve a high level of accuracy with only a few epochs of training. The final accuracy of the model indicates that it can correctly recognize different weather conditions w**ith a high degree of accuracy.**

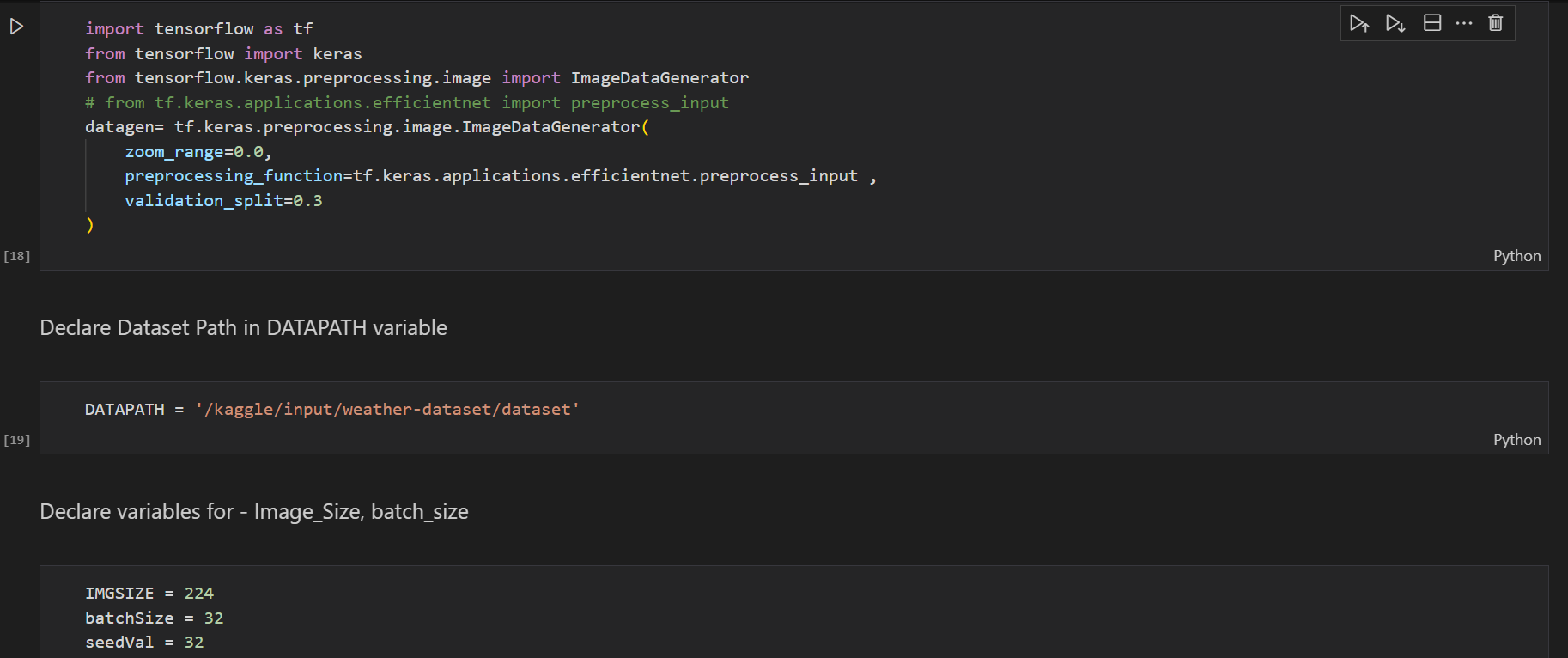




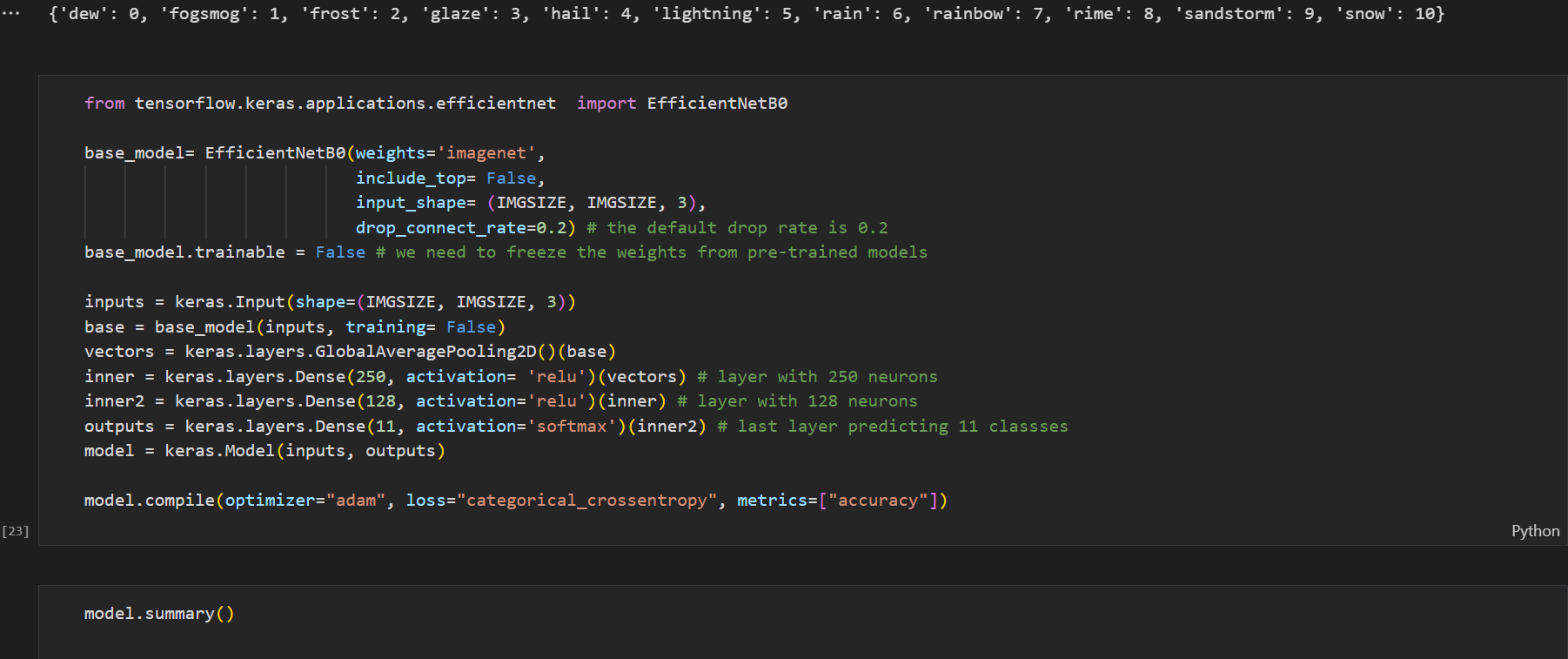
**ROC Curve:**

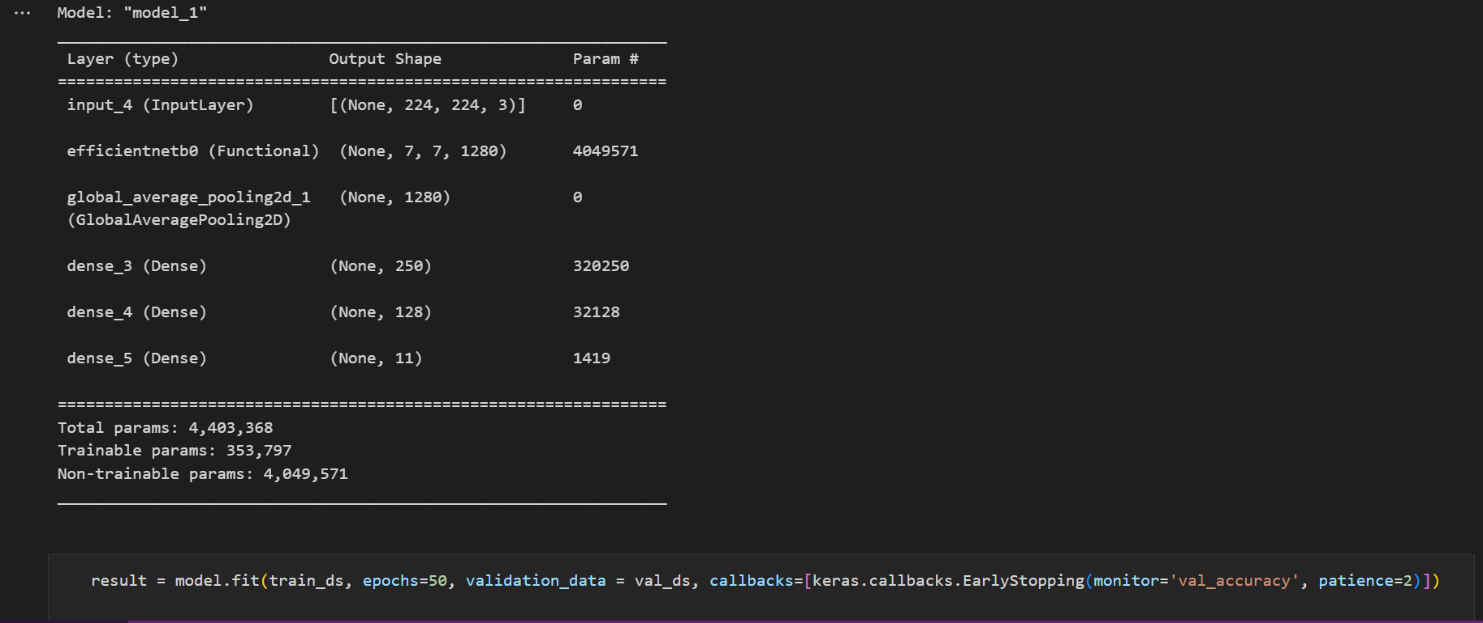


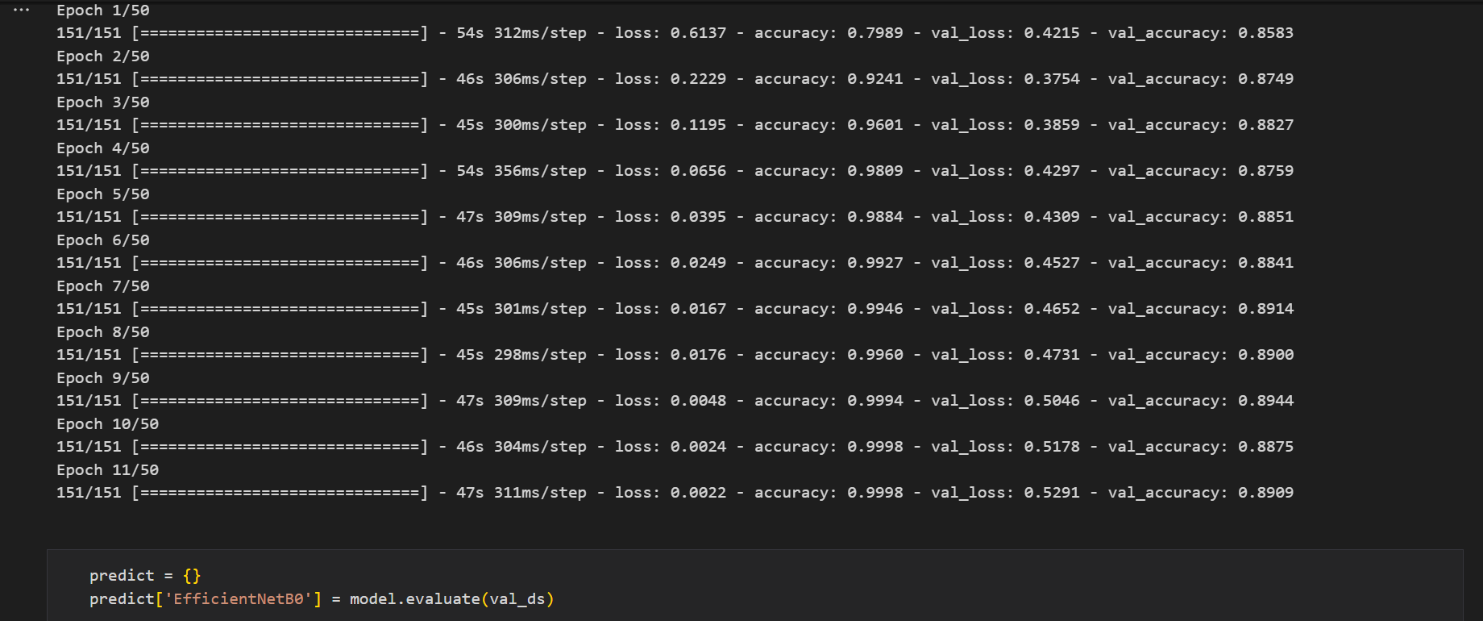
1. **EfficientNetB0:**



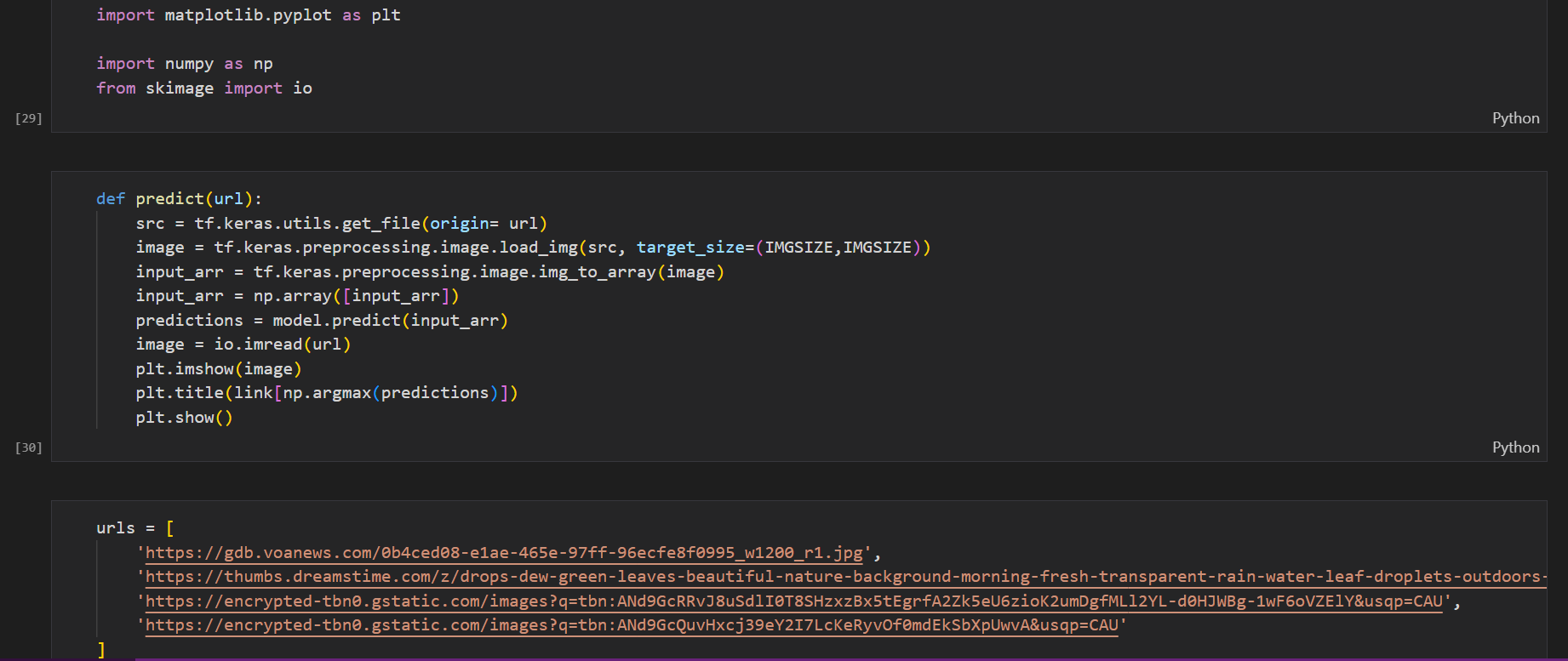


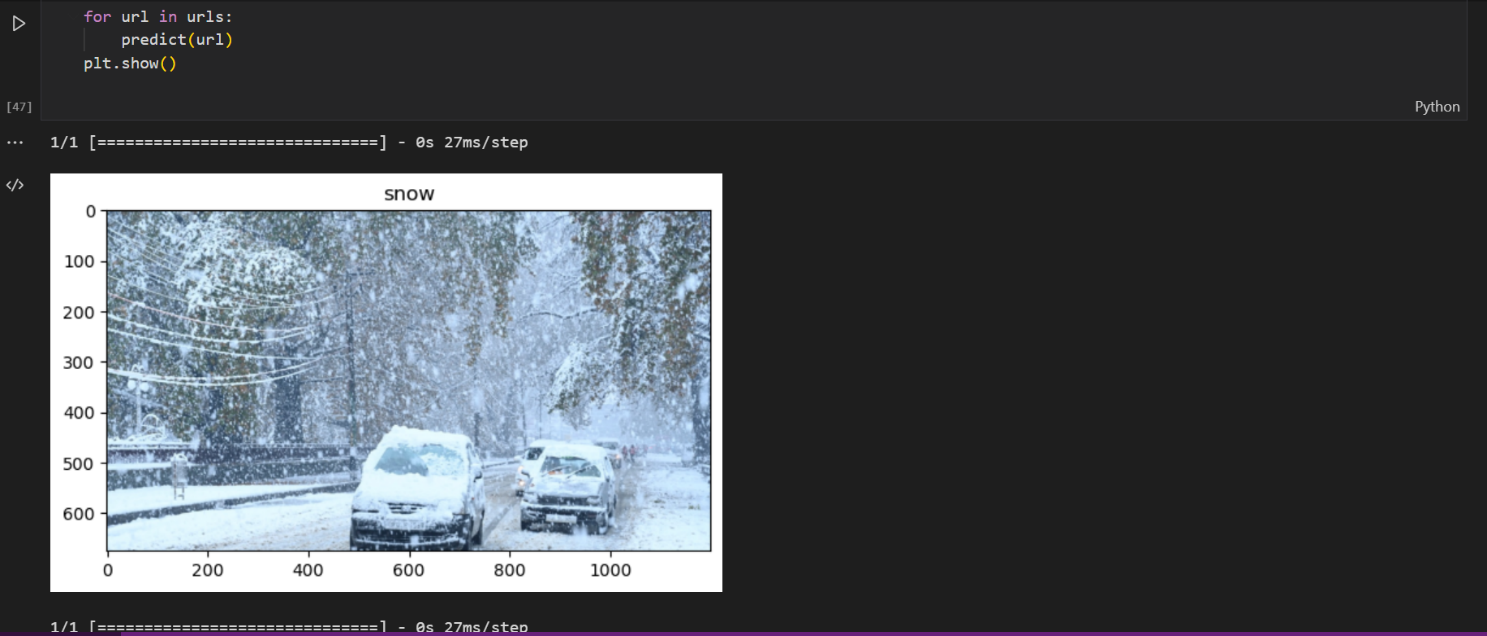


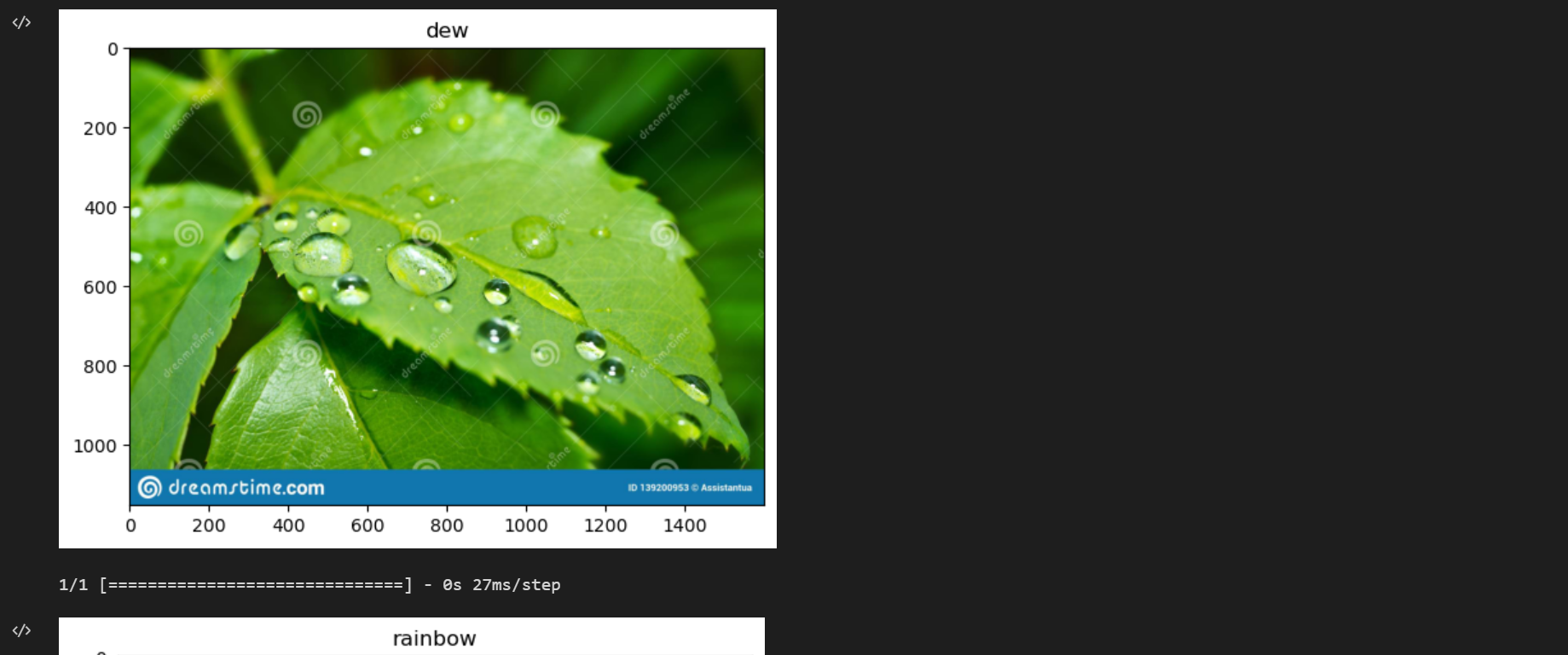


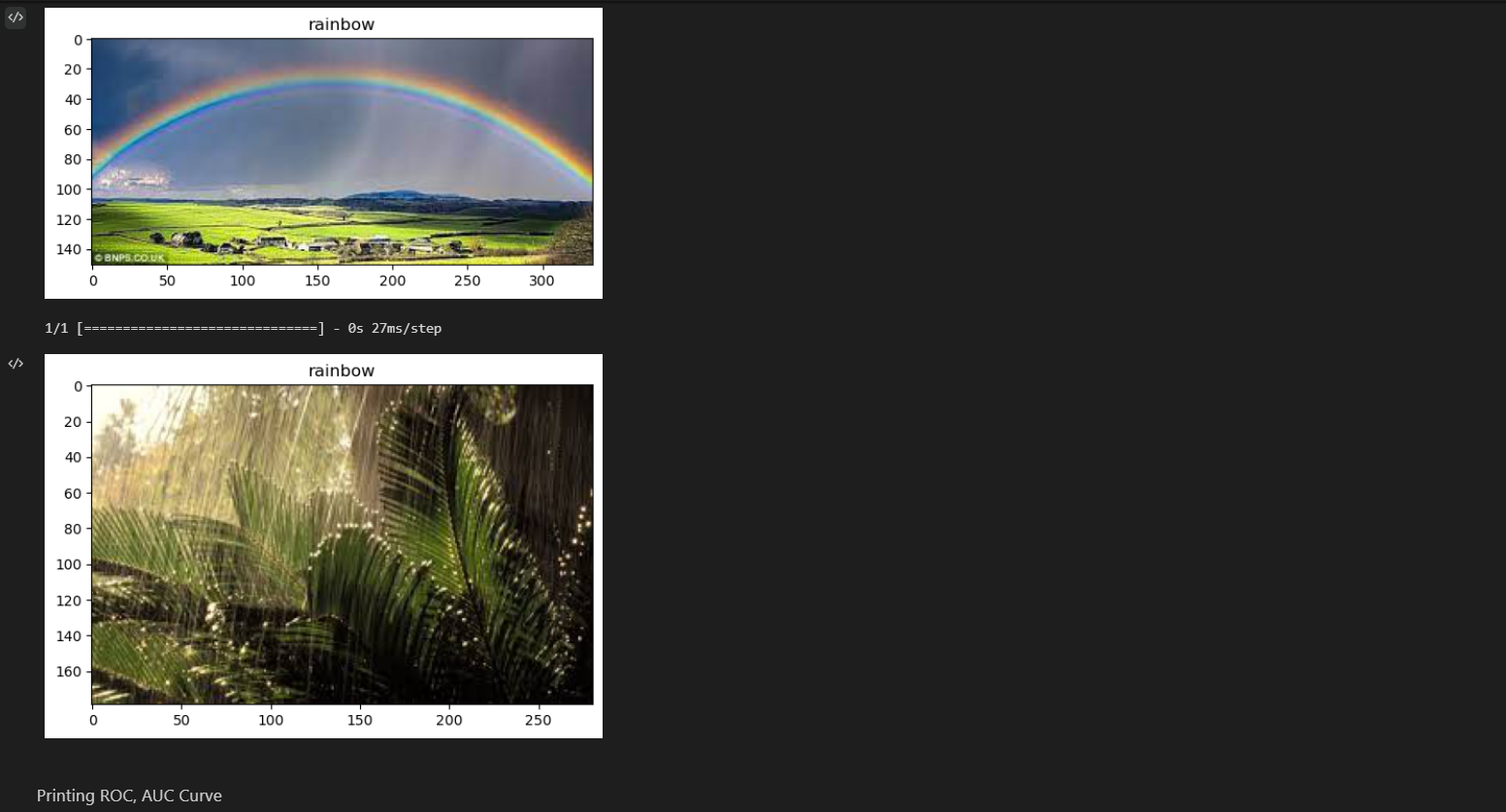


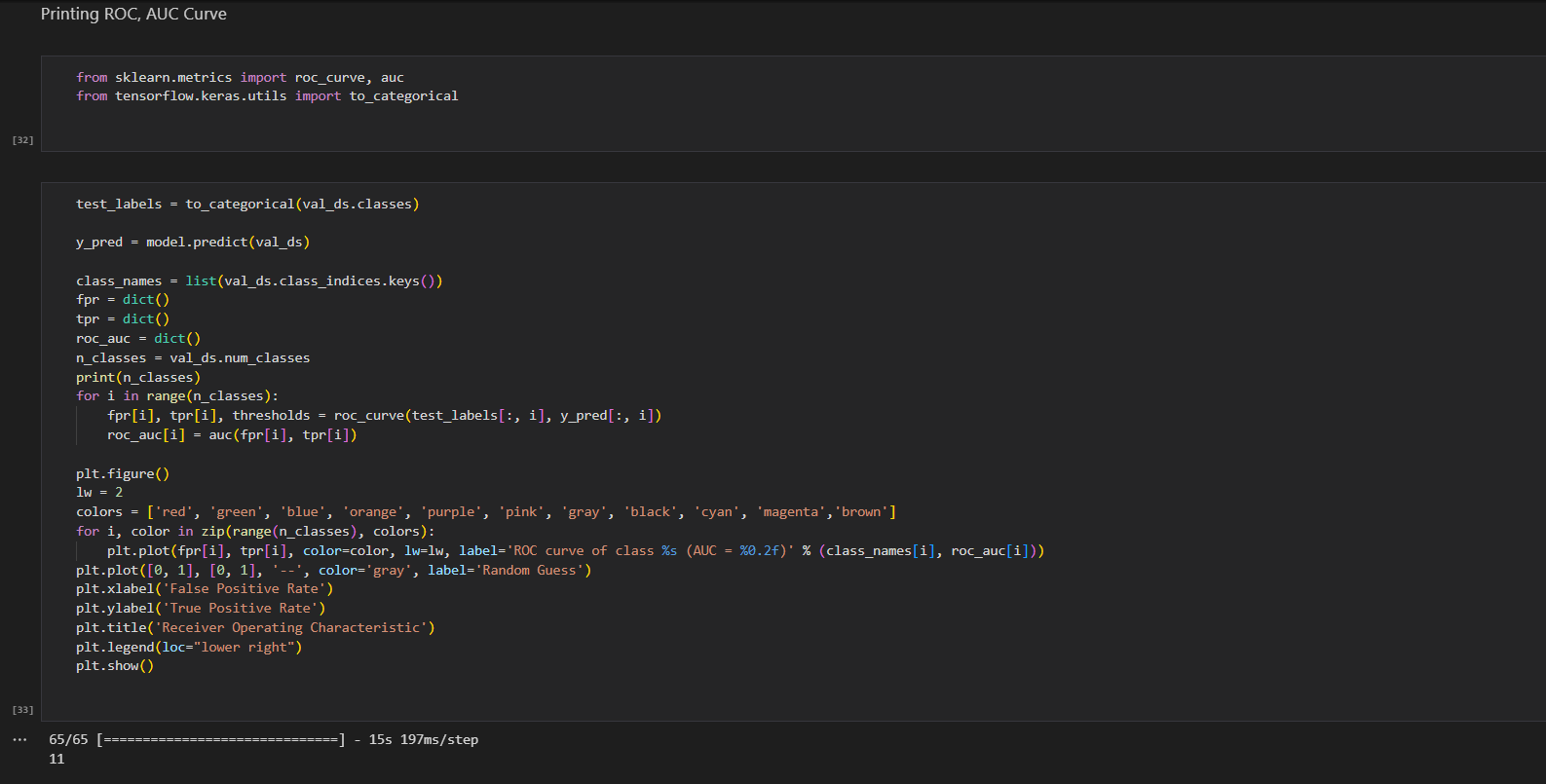


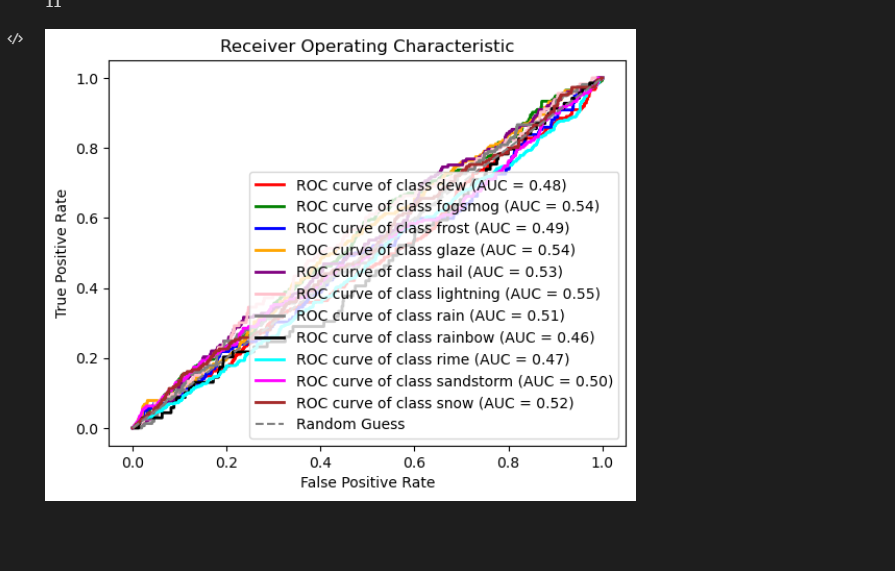






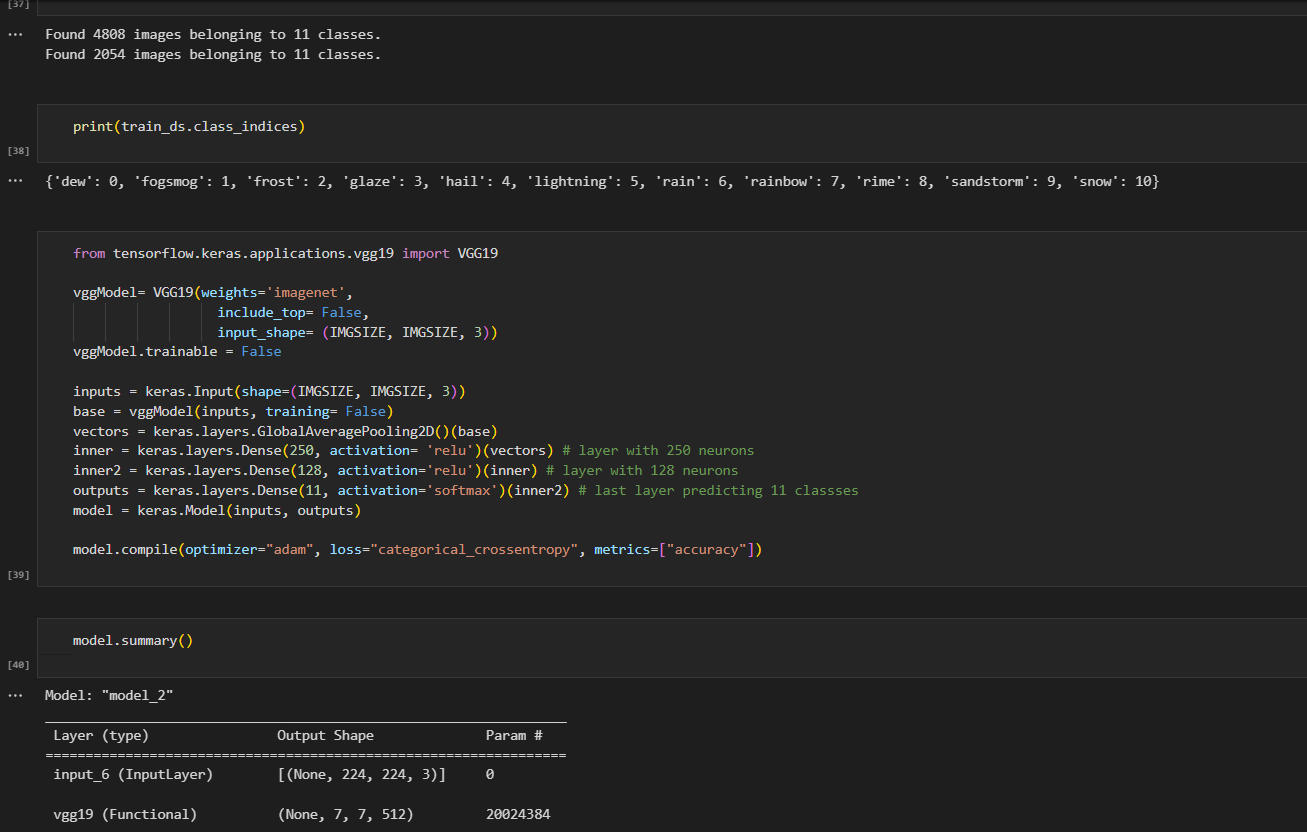


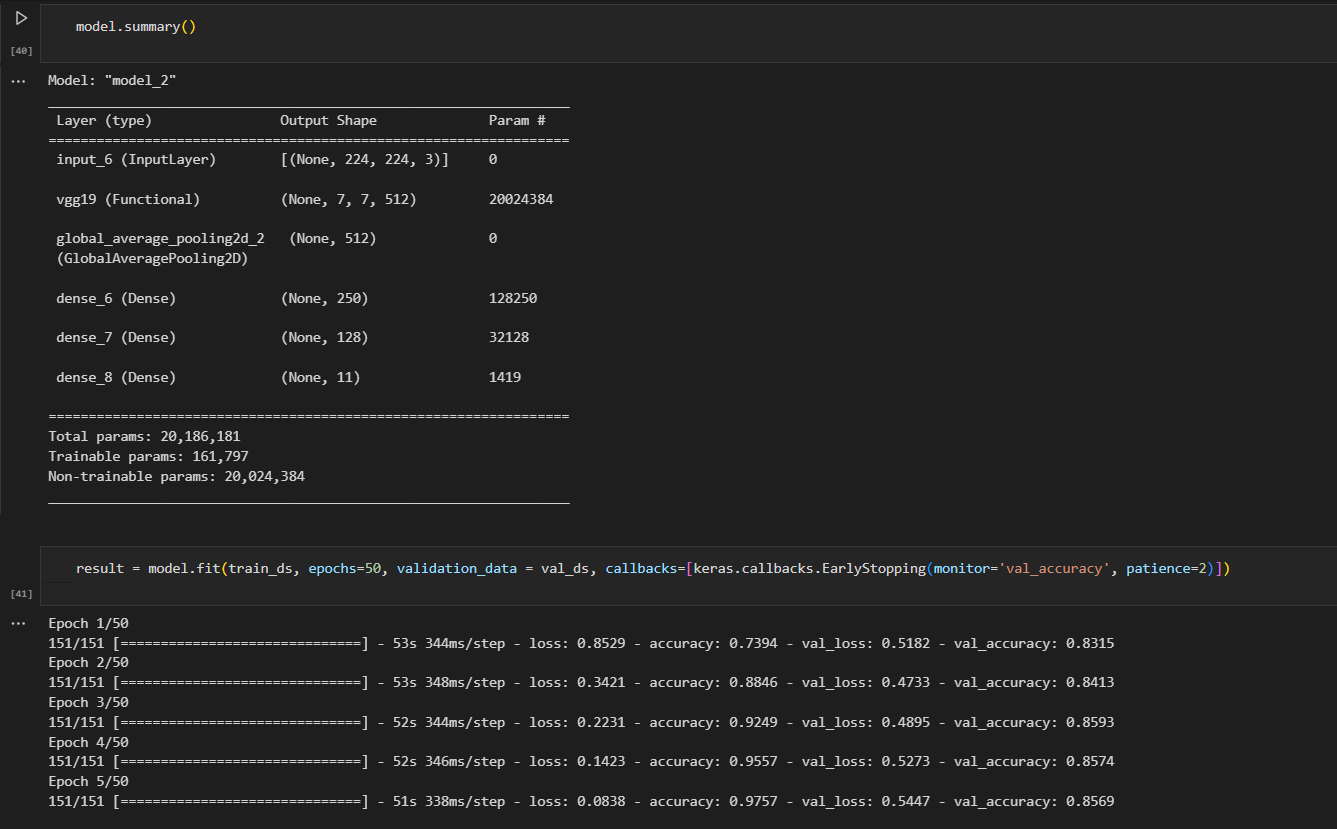


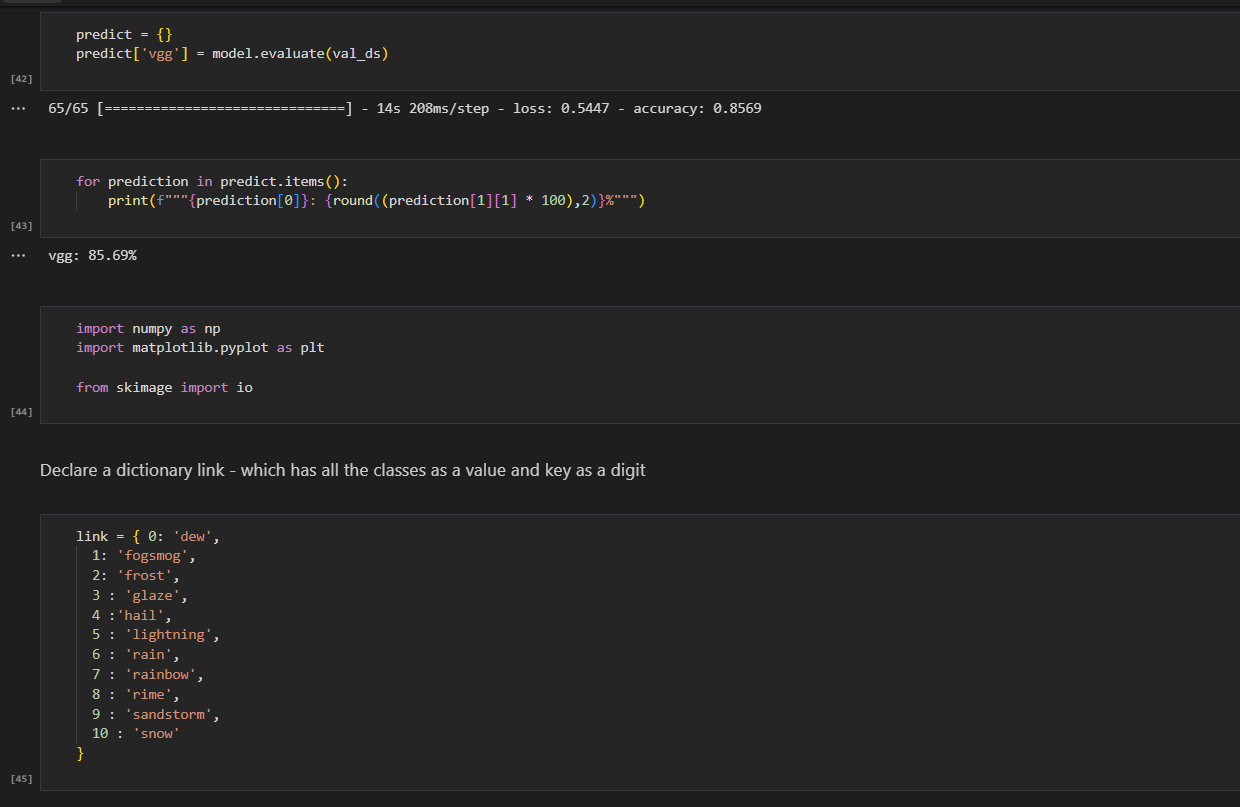


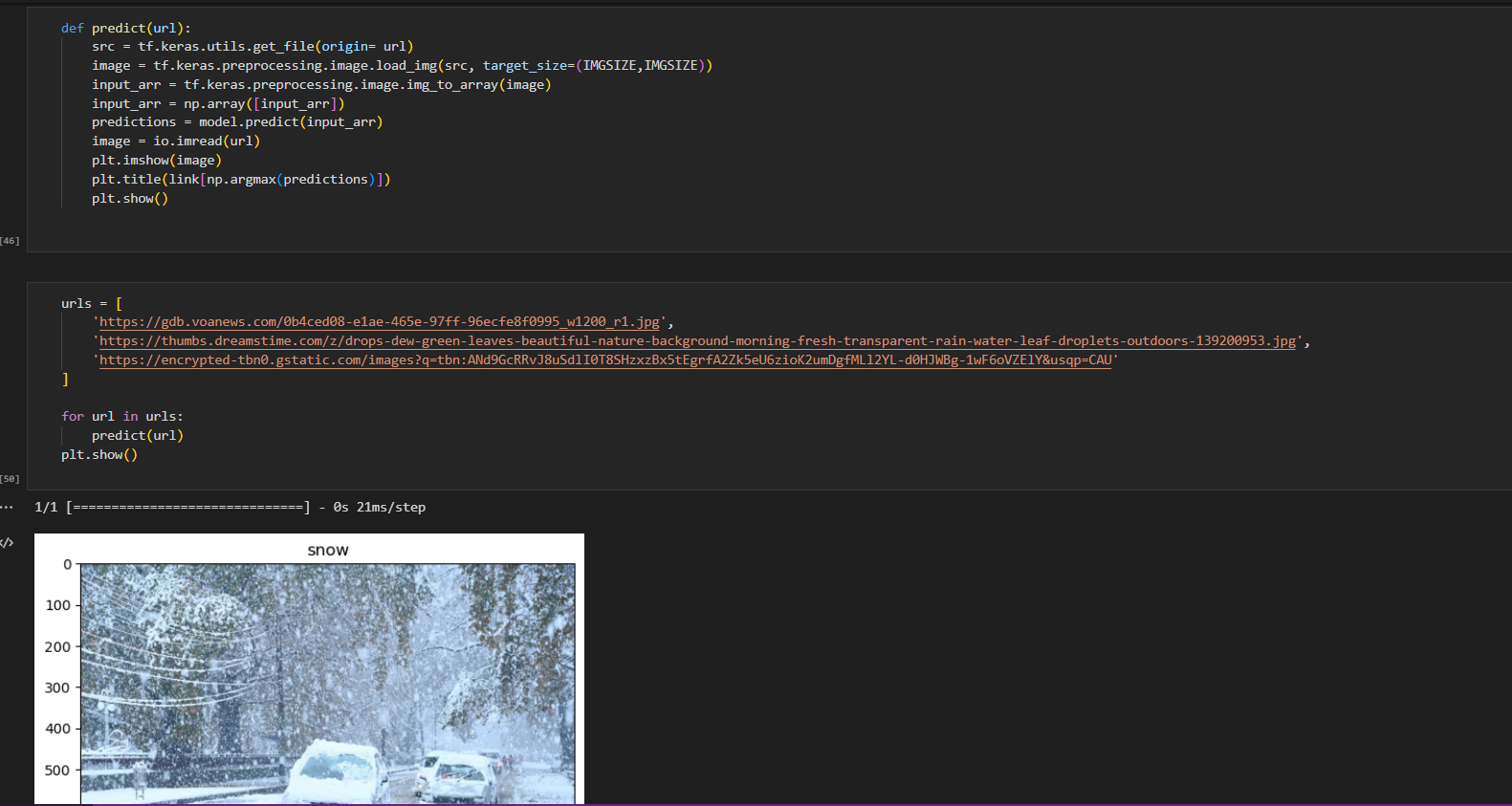
1. **VGG19:**

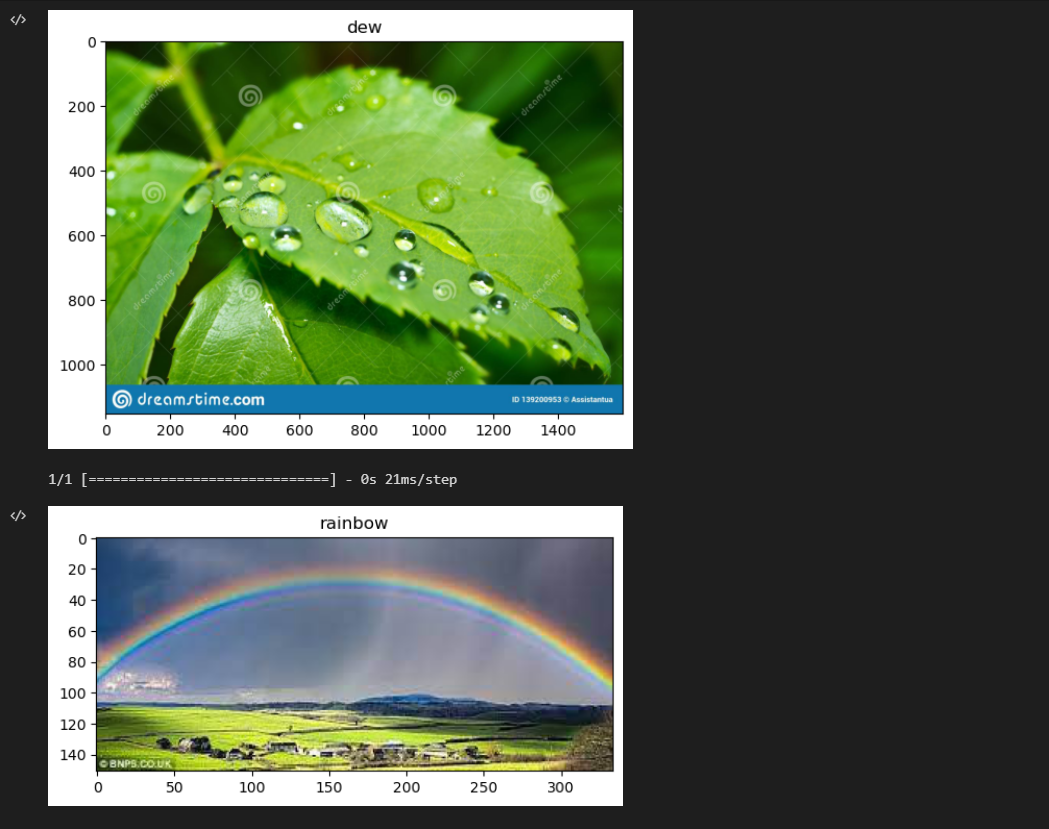


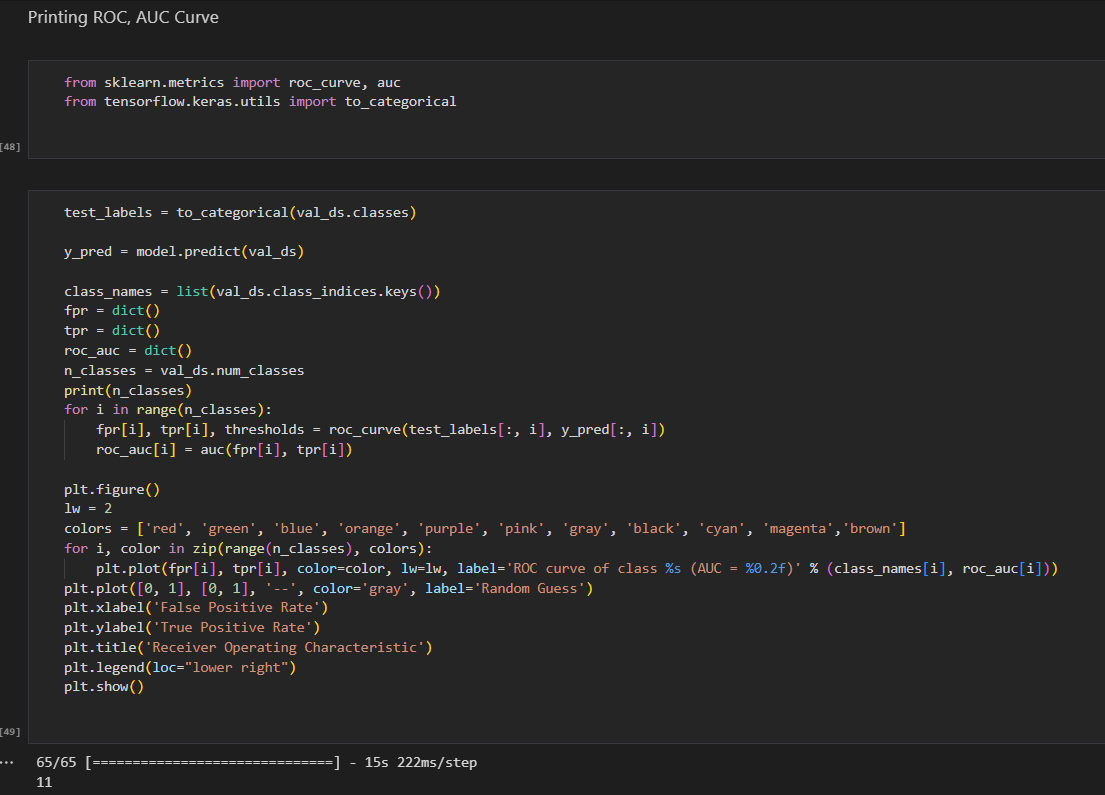


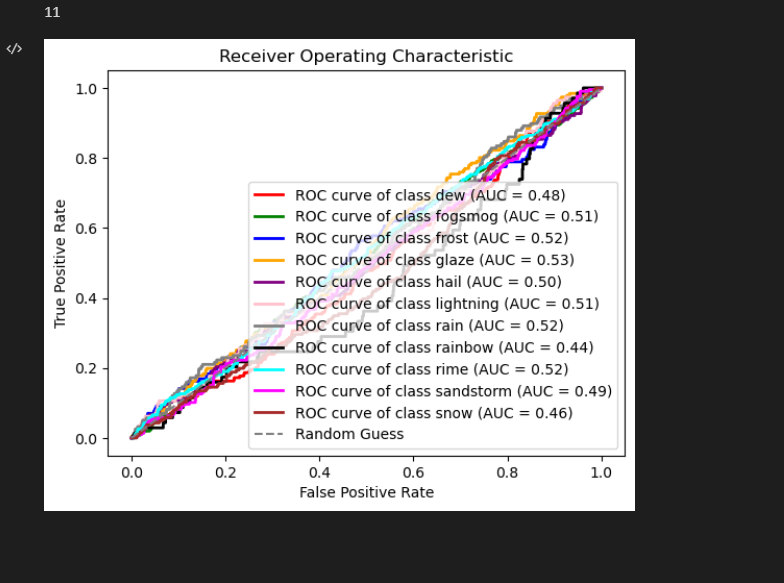








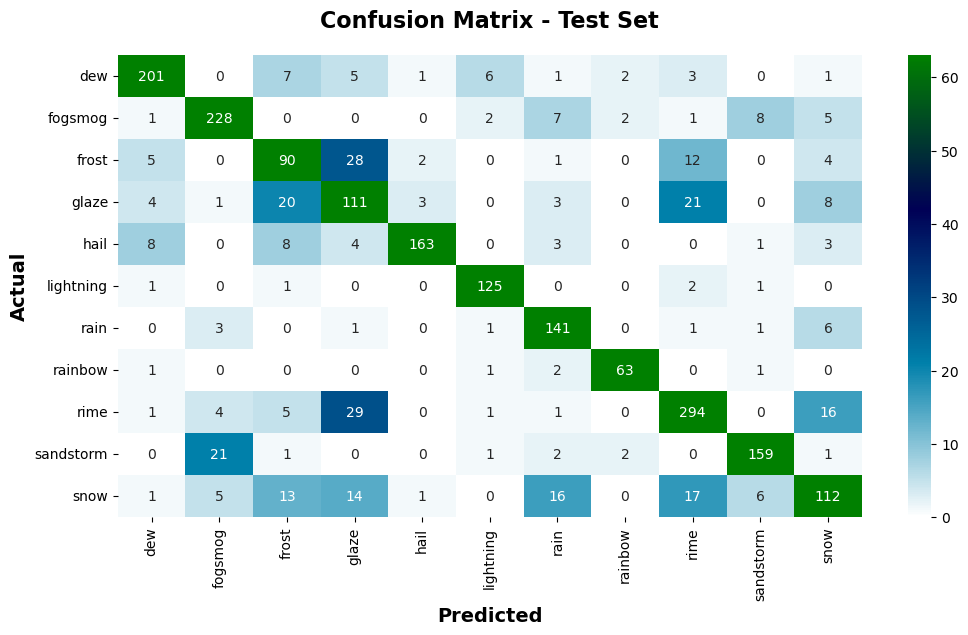




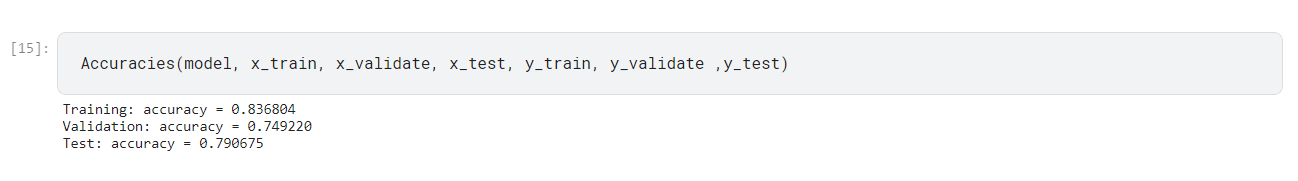
1. **VGG16:**

* Of the whole data, 70% of the data is used for Training and Validation, and the rest 30% data is used for testing.
* For the data allocated for training and testing, 80% is used for training and 20% for validation.
* The whole dataset is normalized by using x = (x-x\_mean)/x\_std
* We used a 120 x 120 input size instead of the default 224x224.
* Used the Adam optimizer and Reduce Learning Rate as optimizers and hyperparameter tuners for better gradient descent and convergence.
* An image Data generator is used to augment new data by zooming and rotating the images, therefore reducing overfitting of data.
* A batch size of 256 with 50 epochs is used for training.

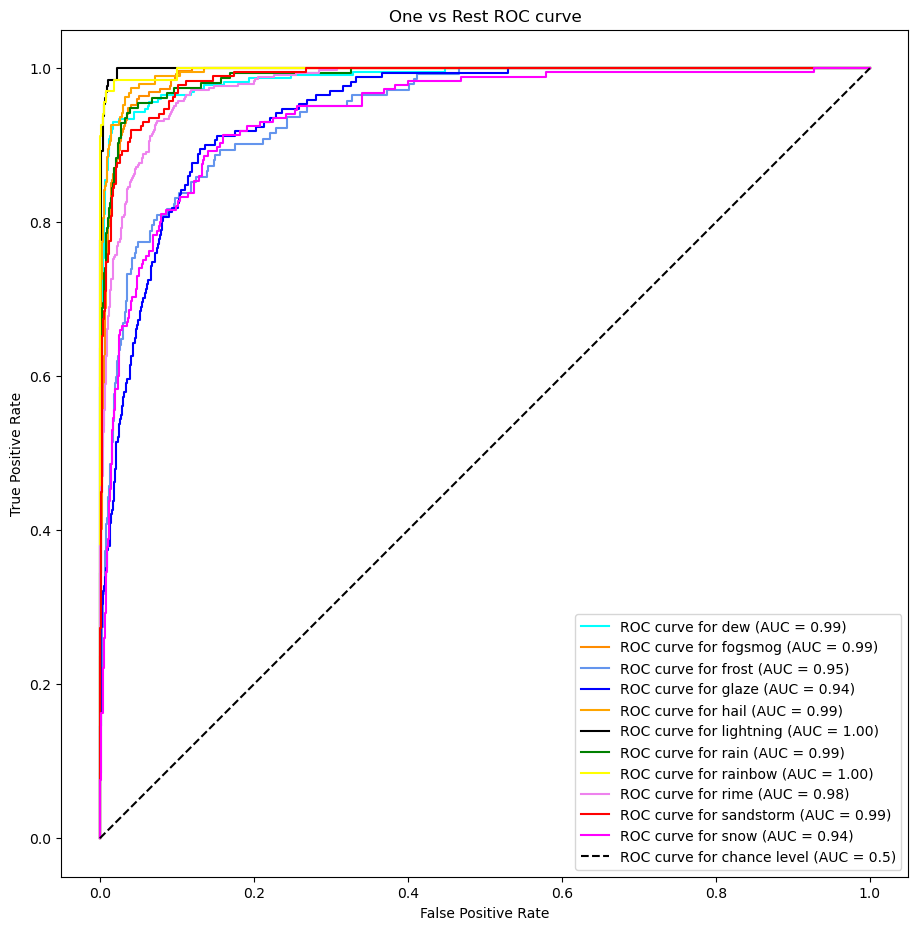




**Confusion Matrix for VGG 16 Test Data**



**Accuracy predicted for the training, validation, and test data**

**One vs Rest Classes ROC Curve**

****

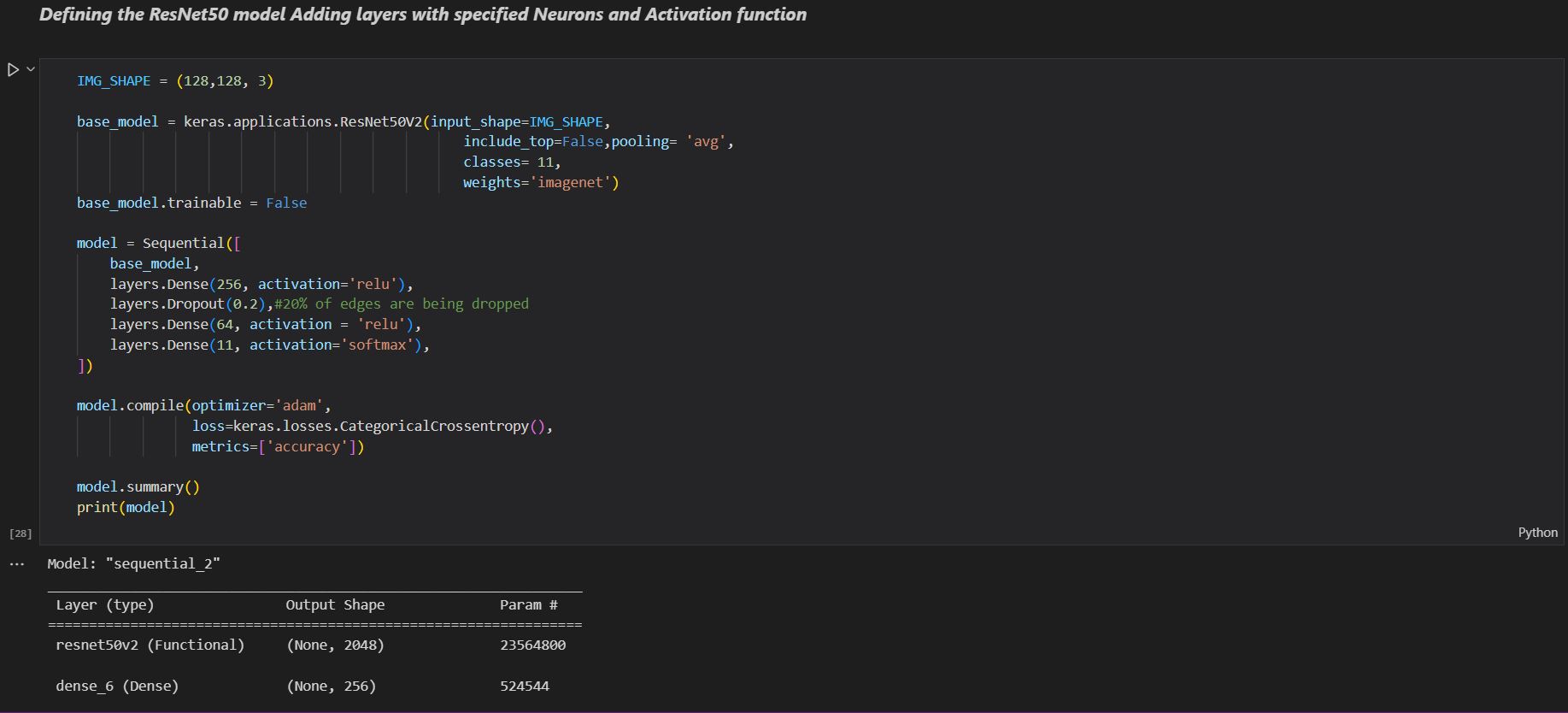
1. **ResNet50:**

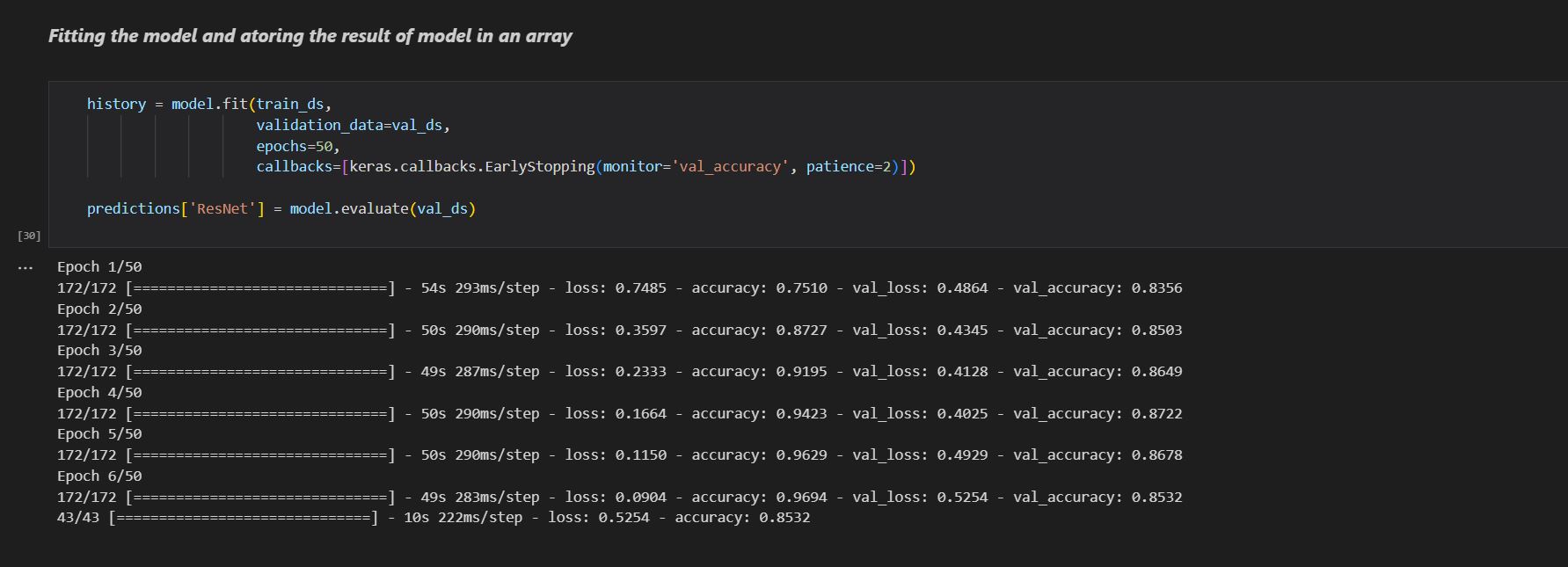
Keras resnet50 is nothing but a residual neural network that is a classic neural network that was used as the backbone of multiple computer tasks.

ResNet-50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). Residual neural networks are a type of artificial neural network (ANN) that forms networks by stacking residual blocks.

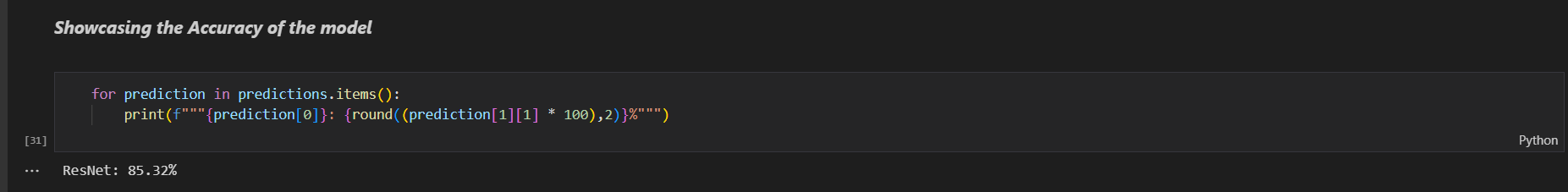
ResNet-50 is a pretrained Deep Learning model for image classification of the Convolutional Neural Network(CNN, or ConvNet), which is a class of deep neural networks

**Model Definition:**

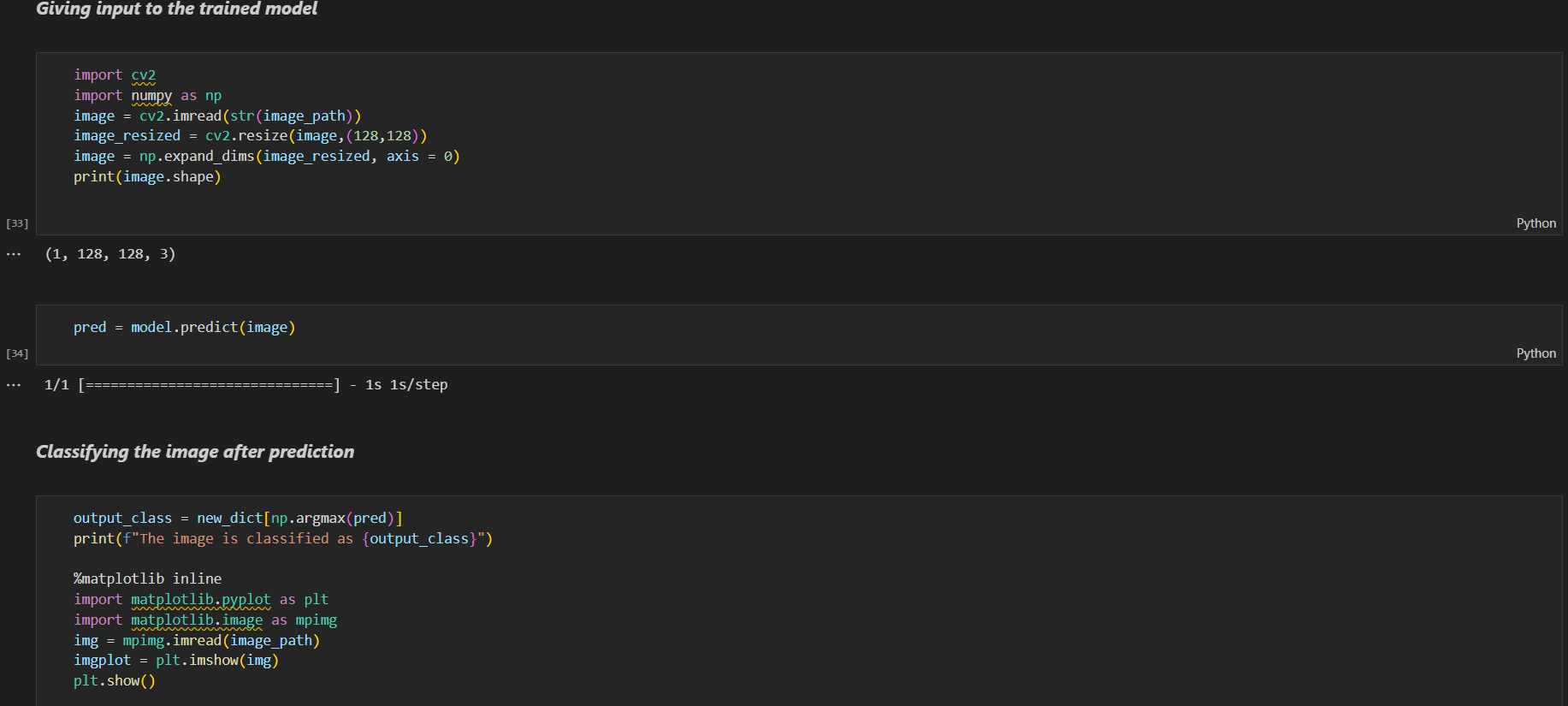


**Fitting the model:**

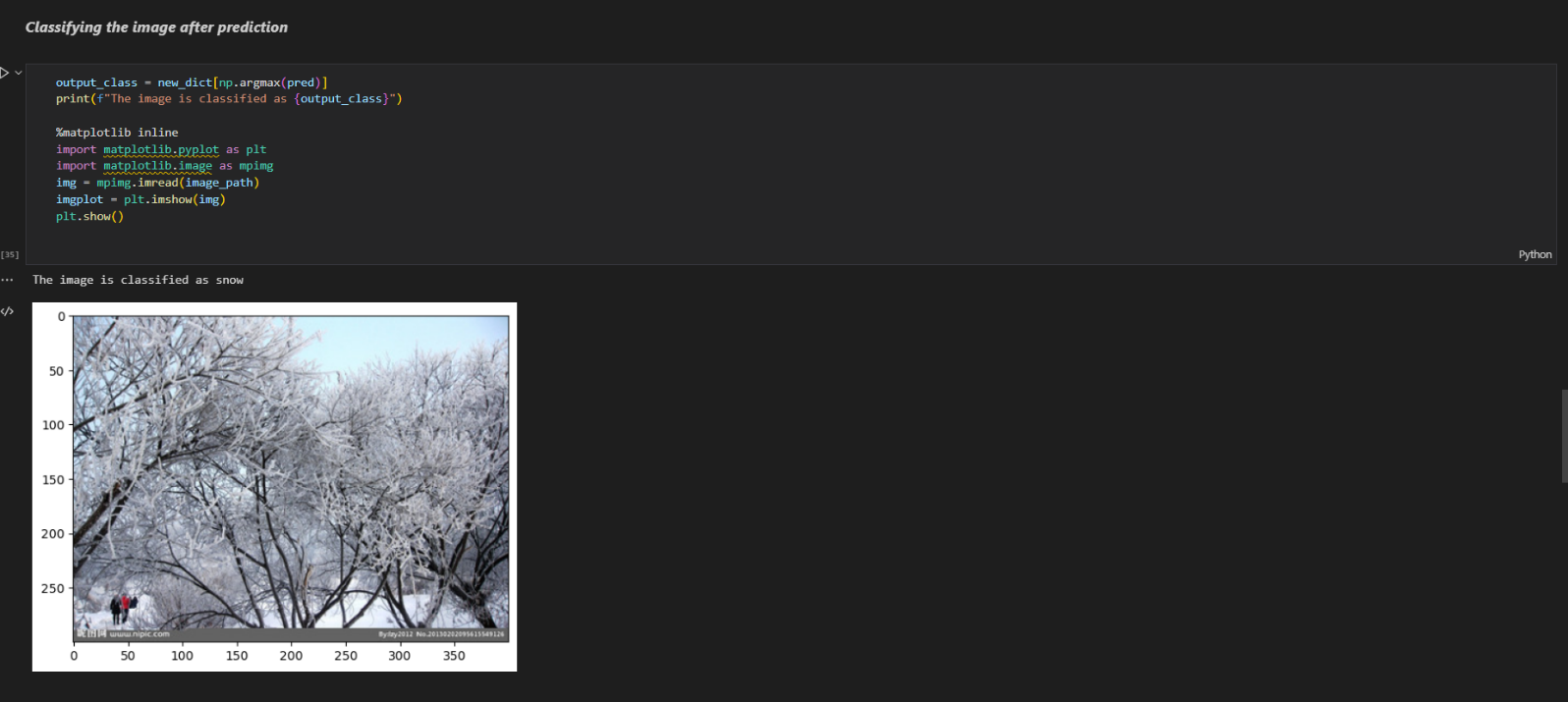
**Accuracy of the model:**



**Giving Input to the trained model:**



**the image:**



**ROC for the ResNet50:**



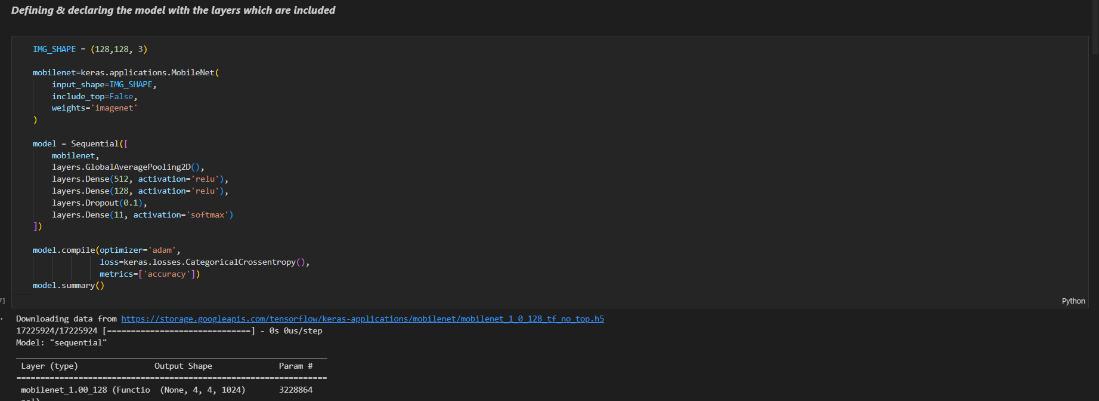
1. **MobileNet:**

MobileNet is a computer vision model open-sourced by Google and designed for training classifiers.

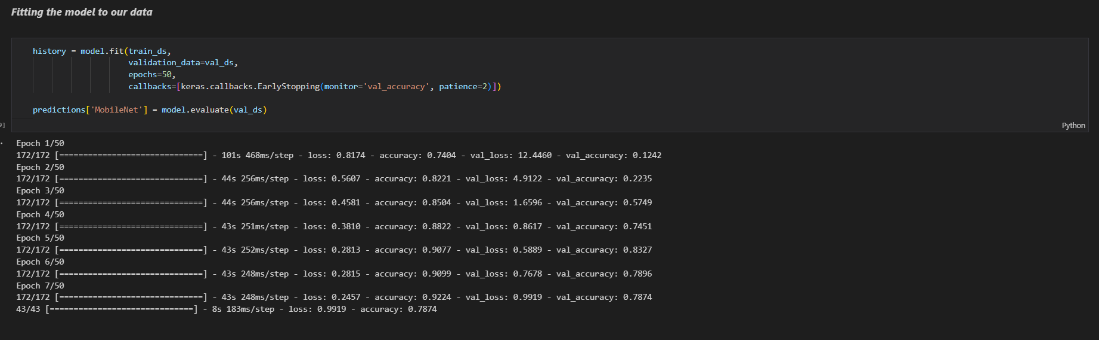
It uses depthwise convolutions to significantly reduce the number of parameters compared to other networks, resulting in a lightweight deep neural network.

MobileNet is Tensorflow's first mobile computer vision model.

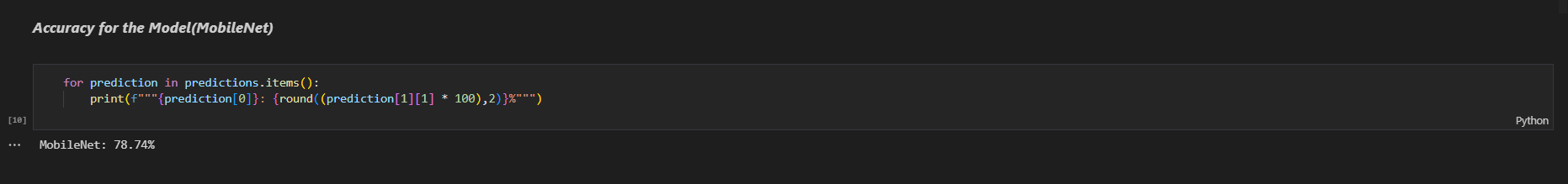
**Model Definition:**



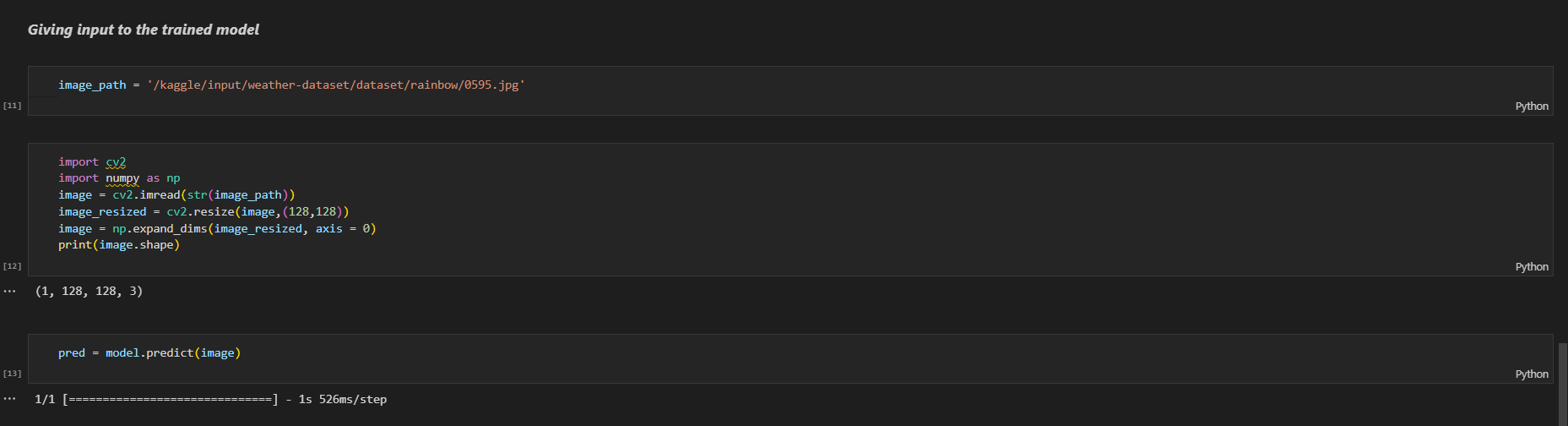
**Fitting the model:**



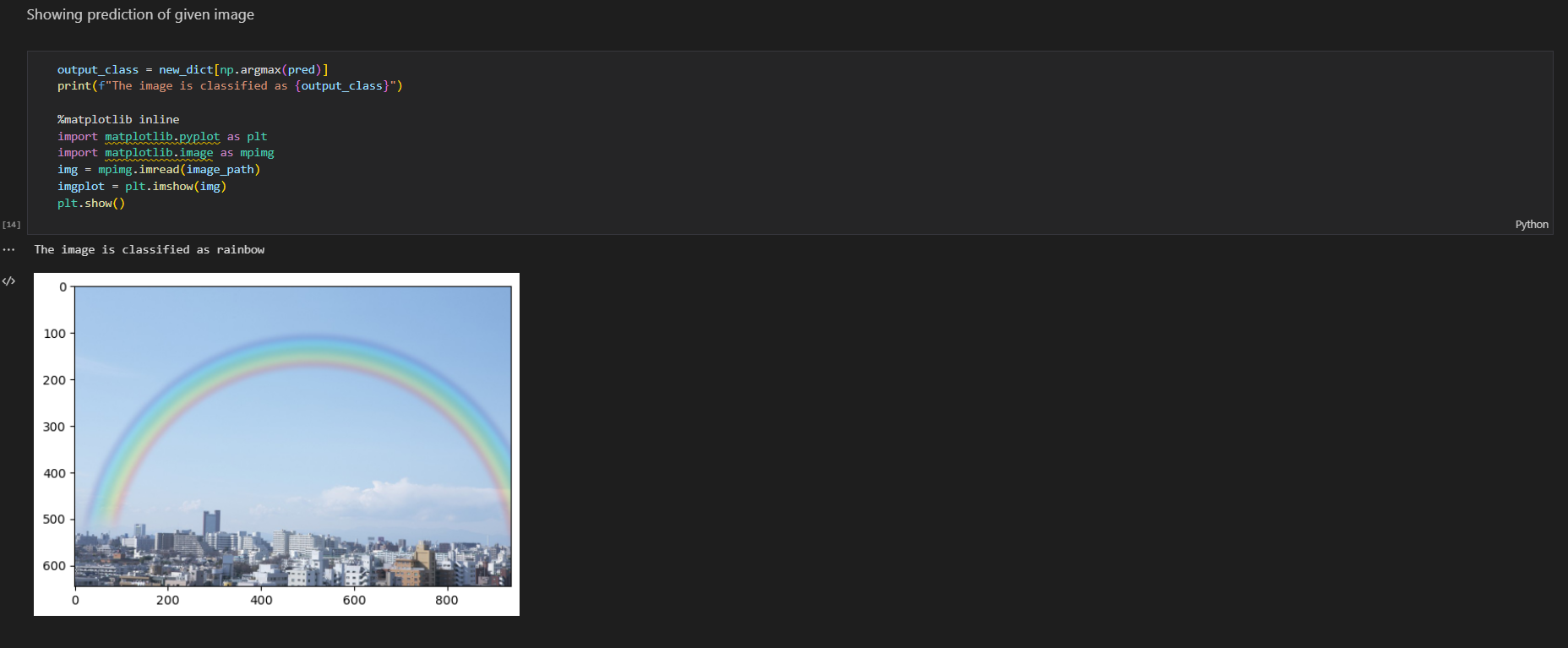
**Accuracy of the model:**



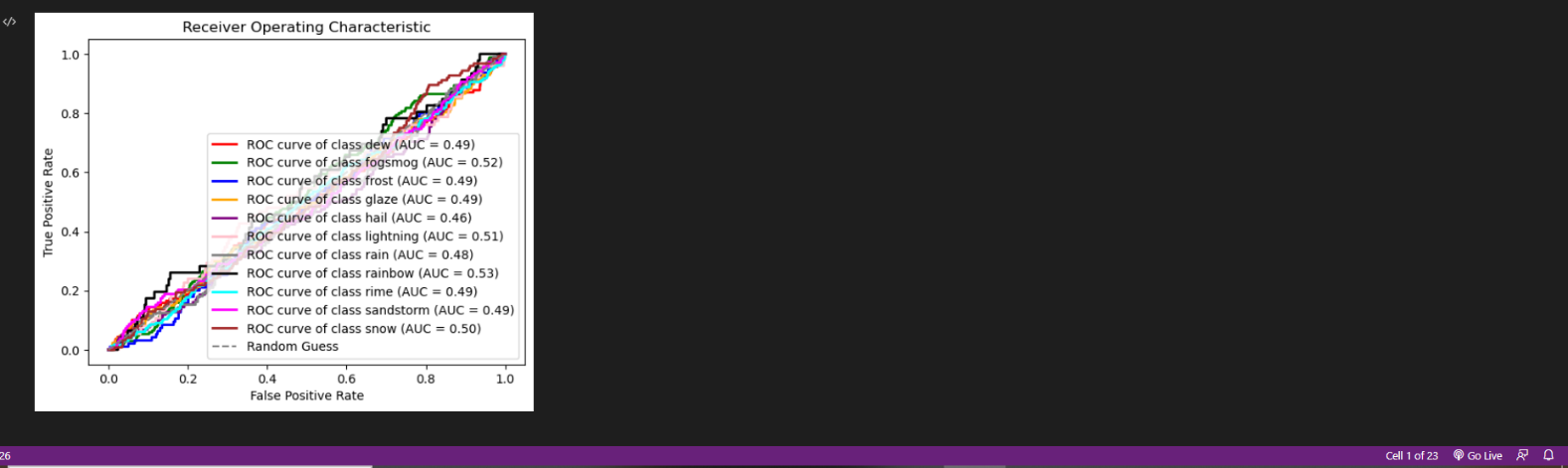
**Giving Input to the trained model:**



**Classifying the image:**



**ROC for the MobileNet:**



Result:

The Algorithms we used to solve the weather predictions are Xception Net, VGG 16, VGG 19, MobileNet, ResNet50, EfficientNetB0

The algorithm with the highest accuracy is XceptionNet - with 90.83%.