**Detecting AI-generated images with CNN and Interpretation using Explainable AI**

**Abstract**

Increase generation of digital media paving new ways for the malicious users to modify original content to make their own content and can make money by selling such modified digital images at cheaper price. Synthetic Fake images can be generated by modifying visual features from the real images and from human eyes it will be difficult to distinguish between real and fake images so propose paper employing deep learning algorithms which can easily detect and differentiate between real and AI Generated images.

Propose paper employing DenseNet121 pre-trained model for image processing and classification. Further GRADCAM based analysis employed for model prediction explaining. Gradient-weighted Class Activation Mapping (Grad-CAM) technique, enabling to visualize the regions within images that influenced the model's decision-making.

Author has employed another explaining model called LIME which will visually explain which features of the image contributing most for the prediction.

**Dataset Details**

To train and test above algorithm performance we are utilizing REAL and FAKE images dataset which can be downloaded from below link

<https://www.kaggle.com/datasets/gauravduttakiit/140k-real-and-fake-faces?select=train>

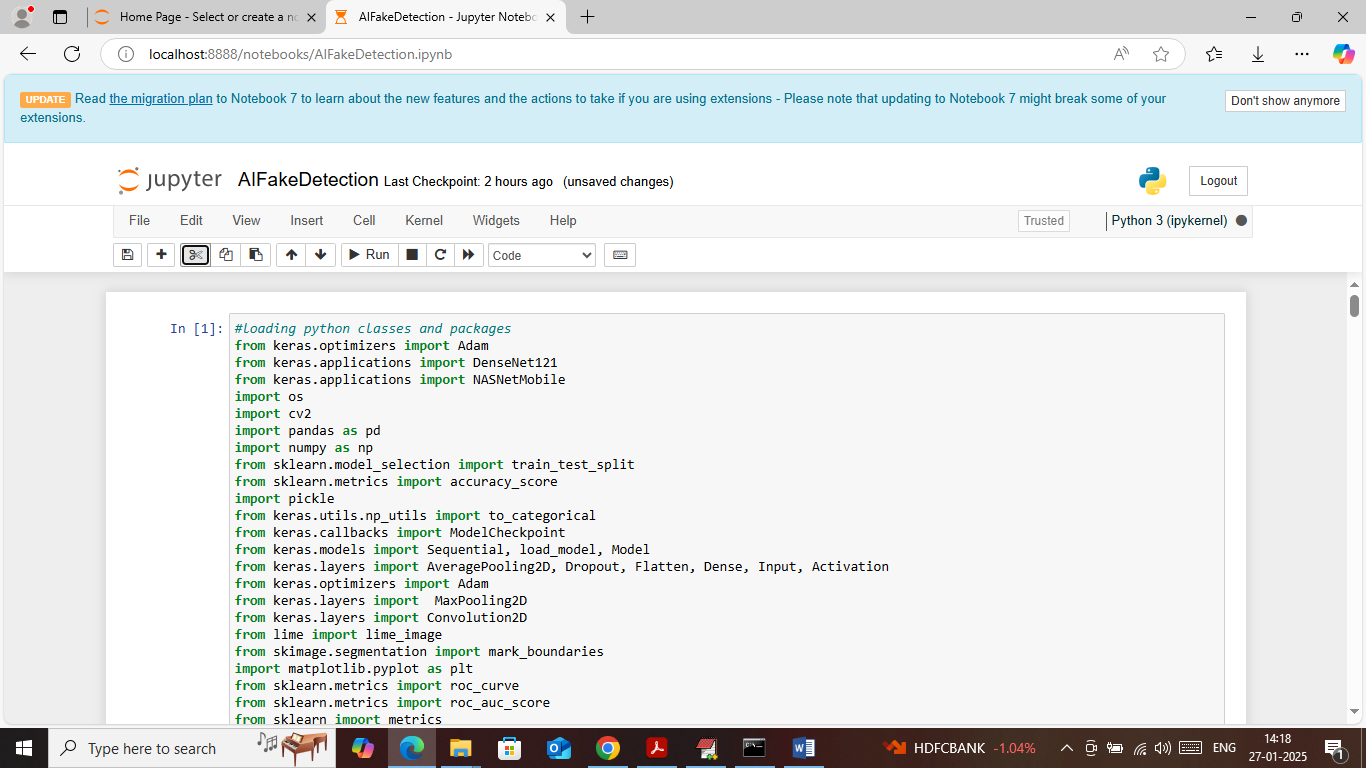
Above dataset will be processed and split into train and test where application will be using 80% dataset images for testing and 20% to calculate model prediction accuracy

**Enhancement**

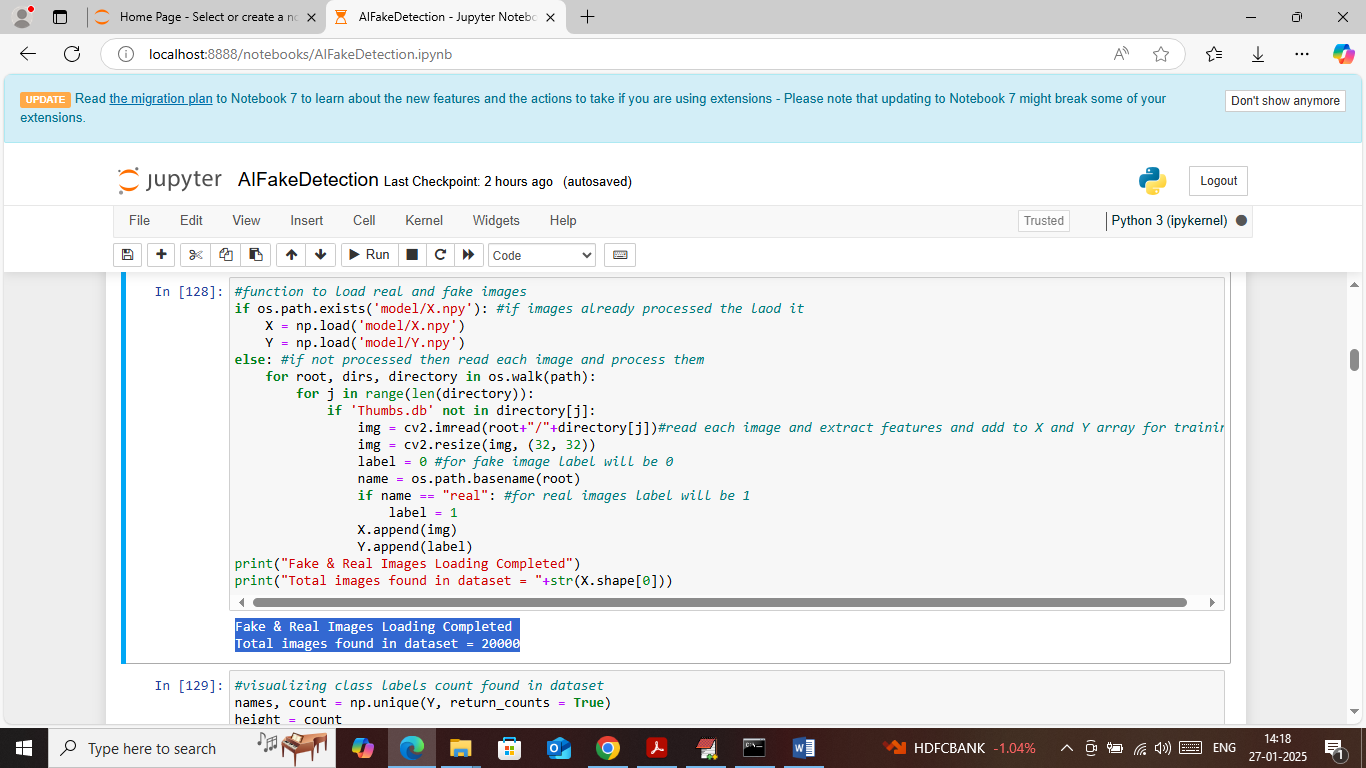
In propose work author has employed DenseNet121 algorithm whose performance can be enhance by employing other advance pre-trained model architectures such as VGG16 or 19. During implementation we have trained with experimented with various algorithms like VGG16, VGG19, XCEPTION, NASNET and many other algorithms but in all algorithms NASNET was giving high accuracy so we have used NASNET as the extension algorithm which is not used in any other previously implemented papers.

SCREEN SHOTS

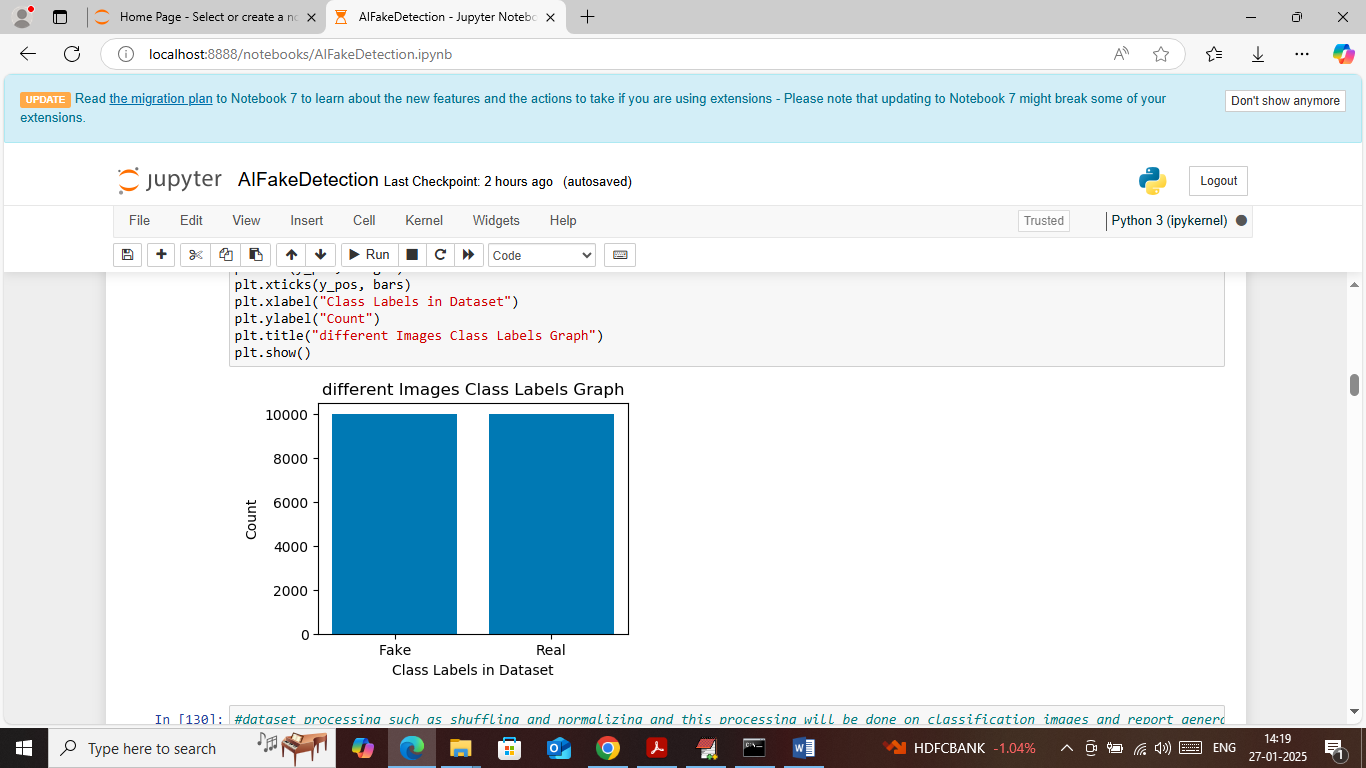
For training, processing, testing and prediction we have used JUPYTER notebook and below are the code and output screens with blue colour comments



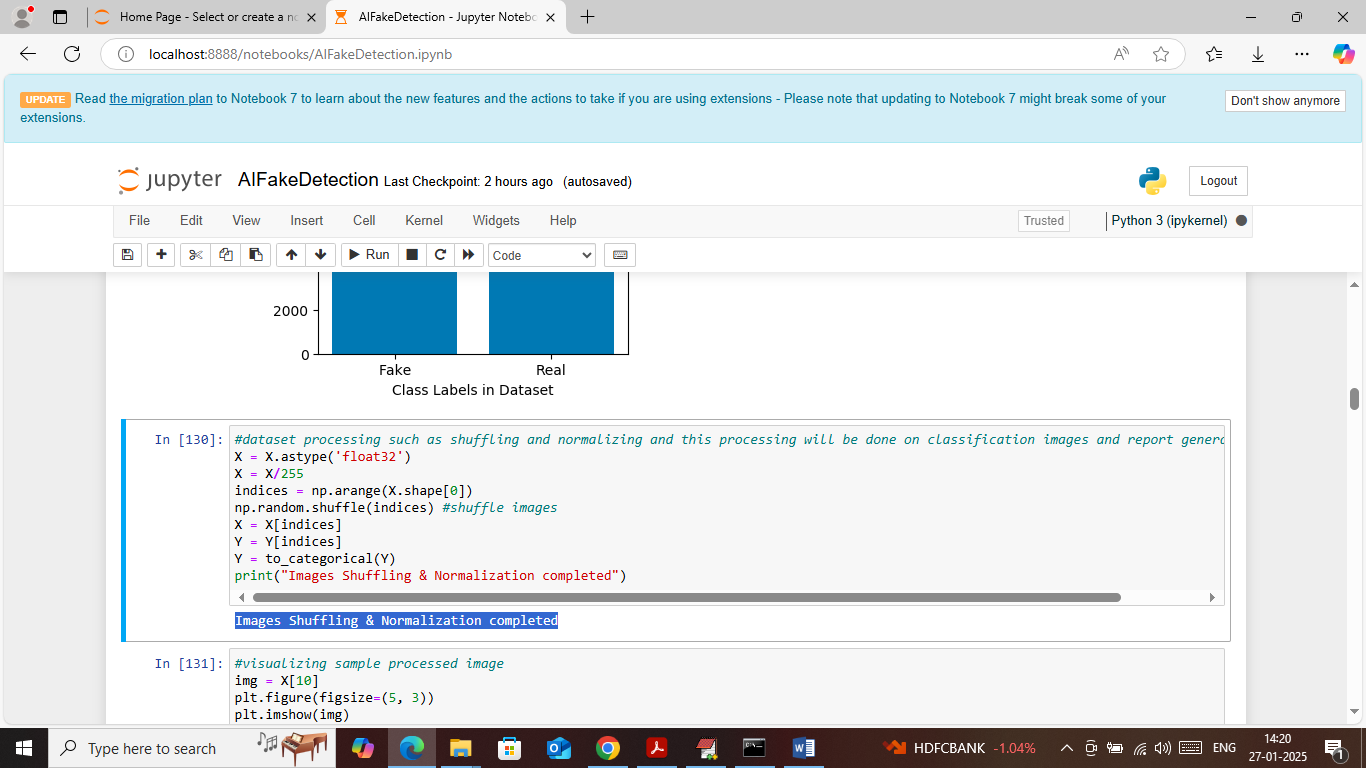
In above screen importing required python classes and packages



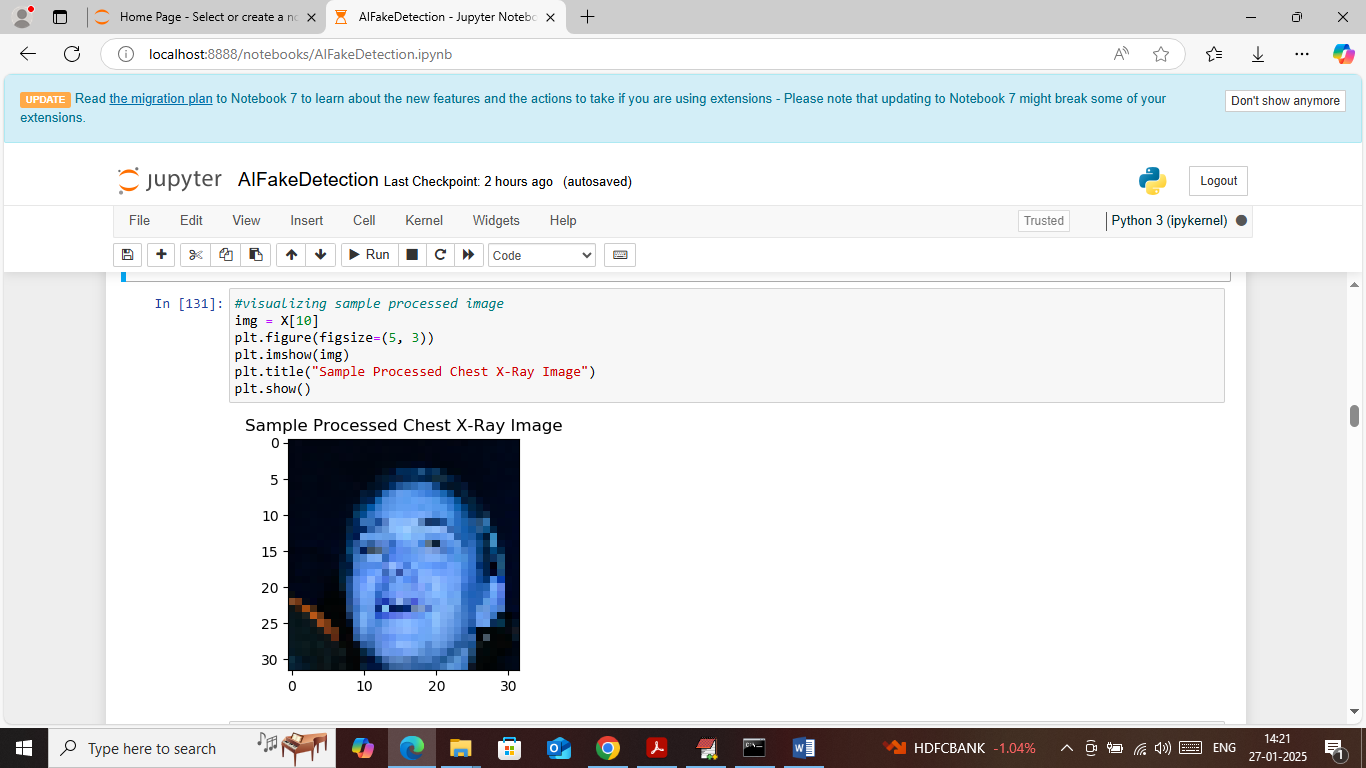
In above screen loading all images from dataset by looping each image and then extracting features and then adding all those extracted features to training X and Y array. In blue text can see total 20000 images loaded from dataset



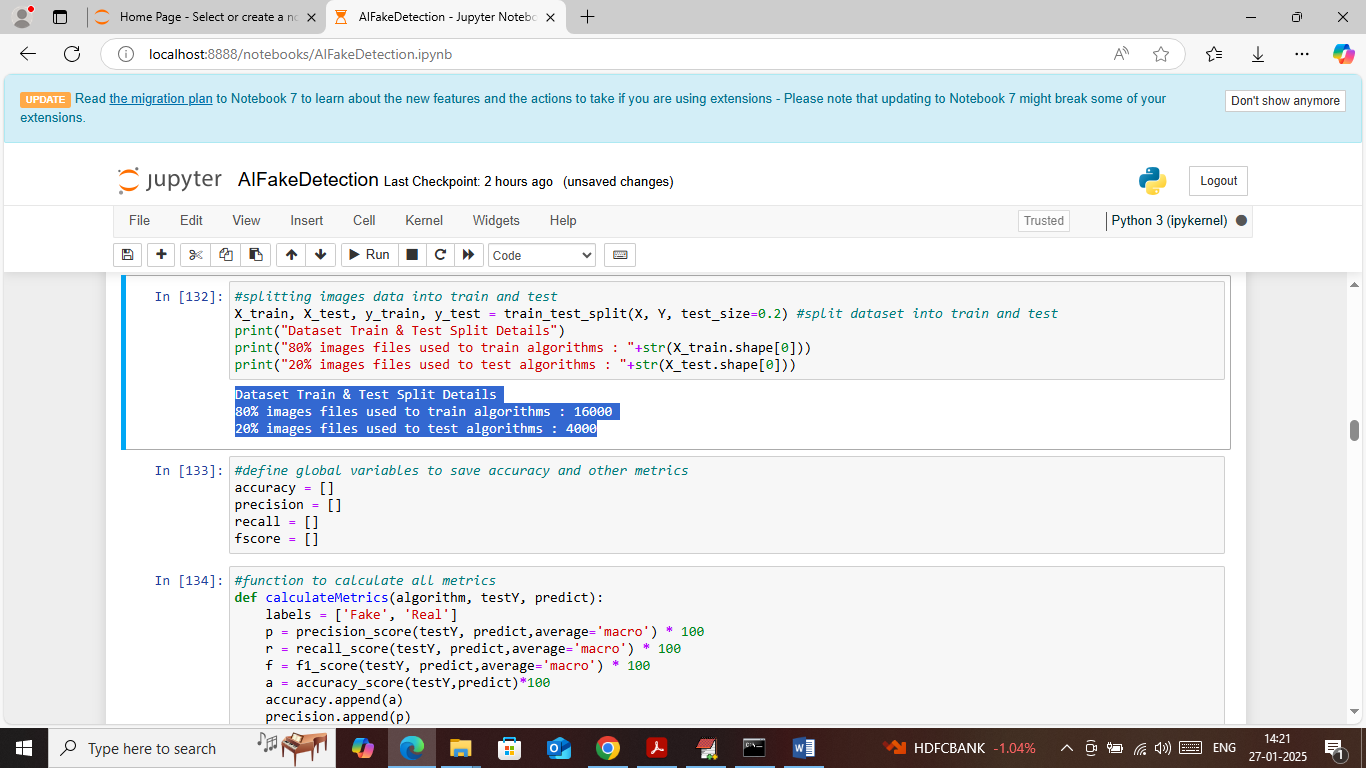
In above screen visualizing different class labels graph where x-axis represents type of images and y-axis represents number of images available under that class label



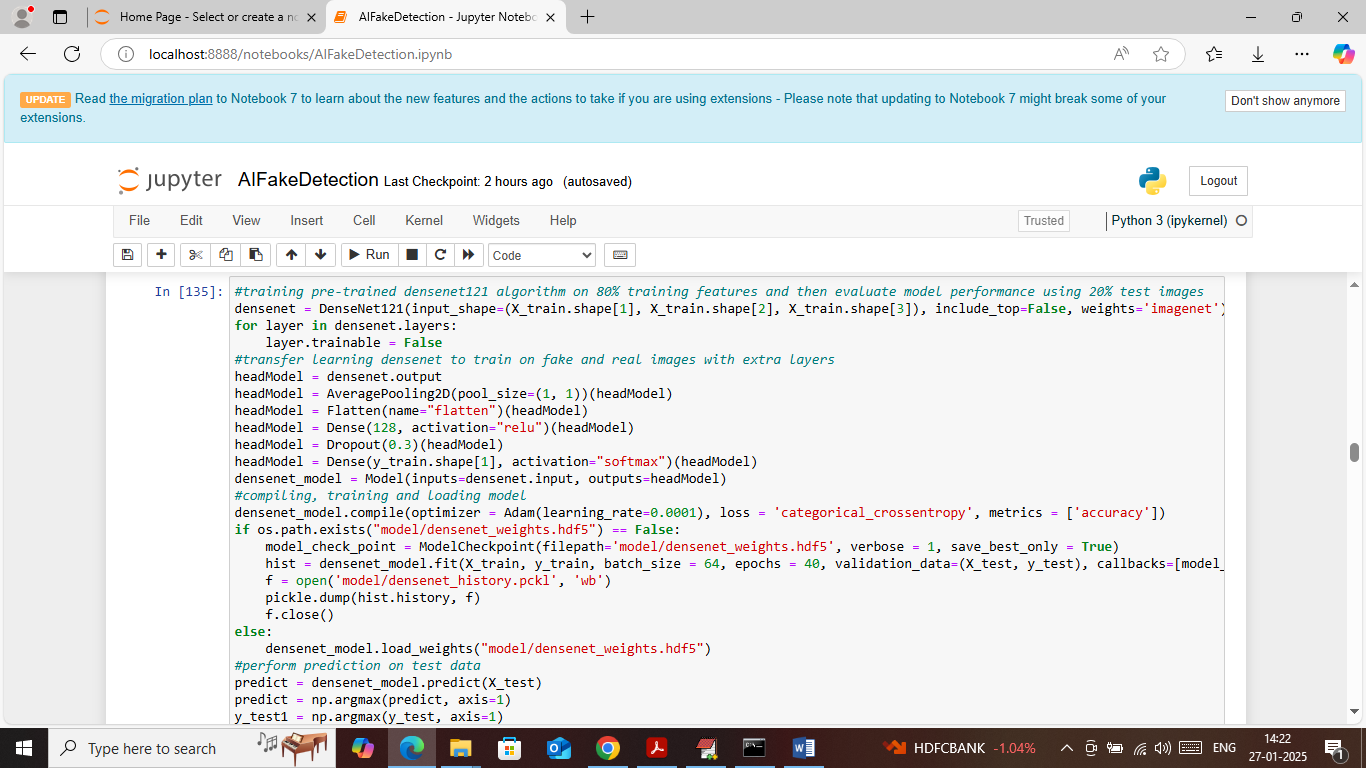
In above screen applying image processing techniques such as shuffling and normalization



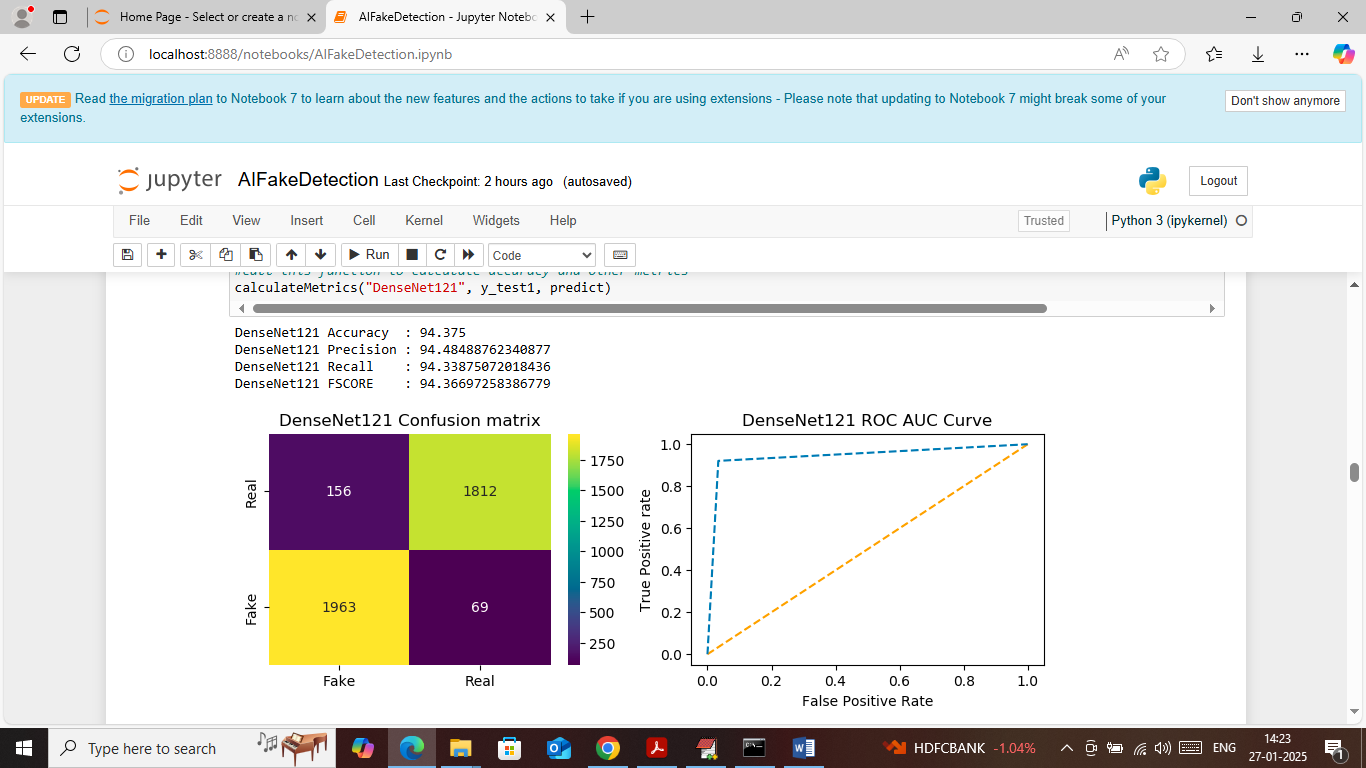
In above screen visualizing sample processed image to check weather images are loaded properly or not



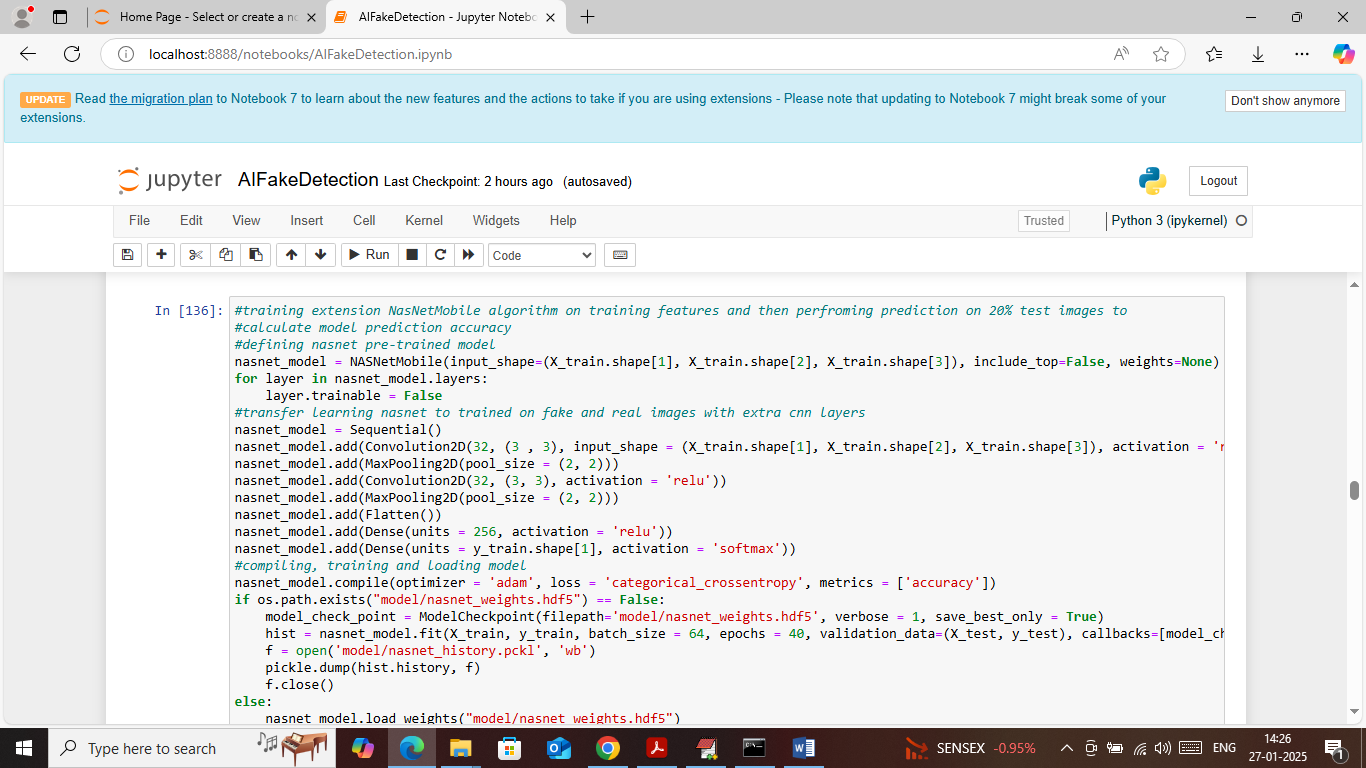
In above screen defining function to split dataset into train and test where application using 80% images for training and 20% for testing and then in blue colour text can see train and test data size. In next block defining function to calculate accuracy and other metrics



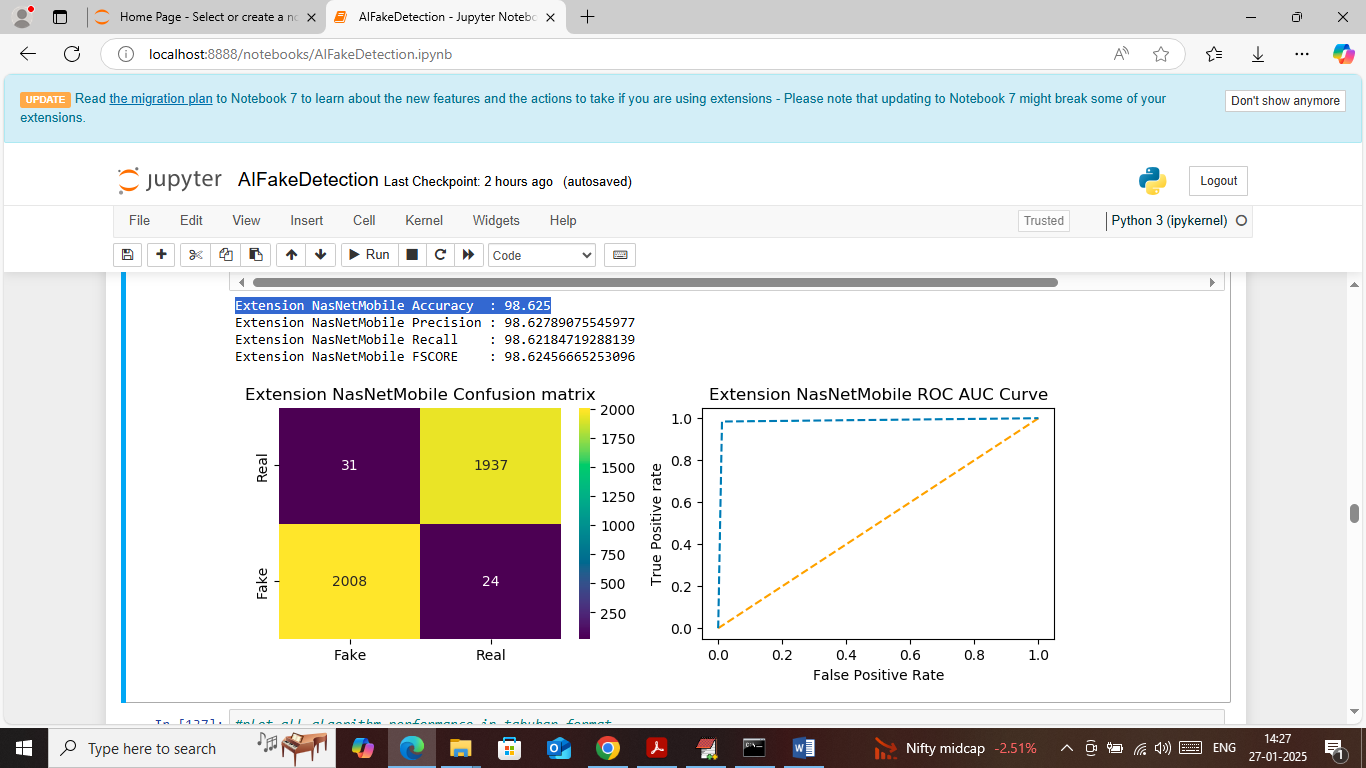
In above screen defining and training DenseNet121 algorithm and after executing above block will get below output



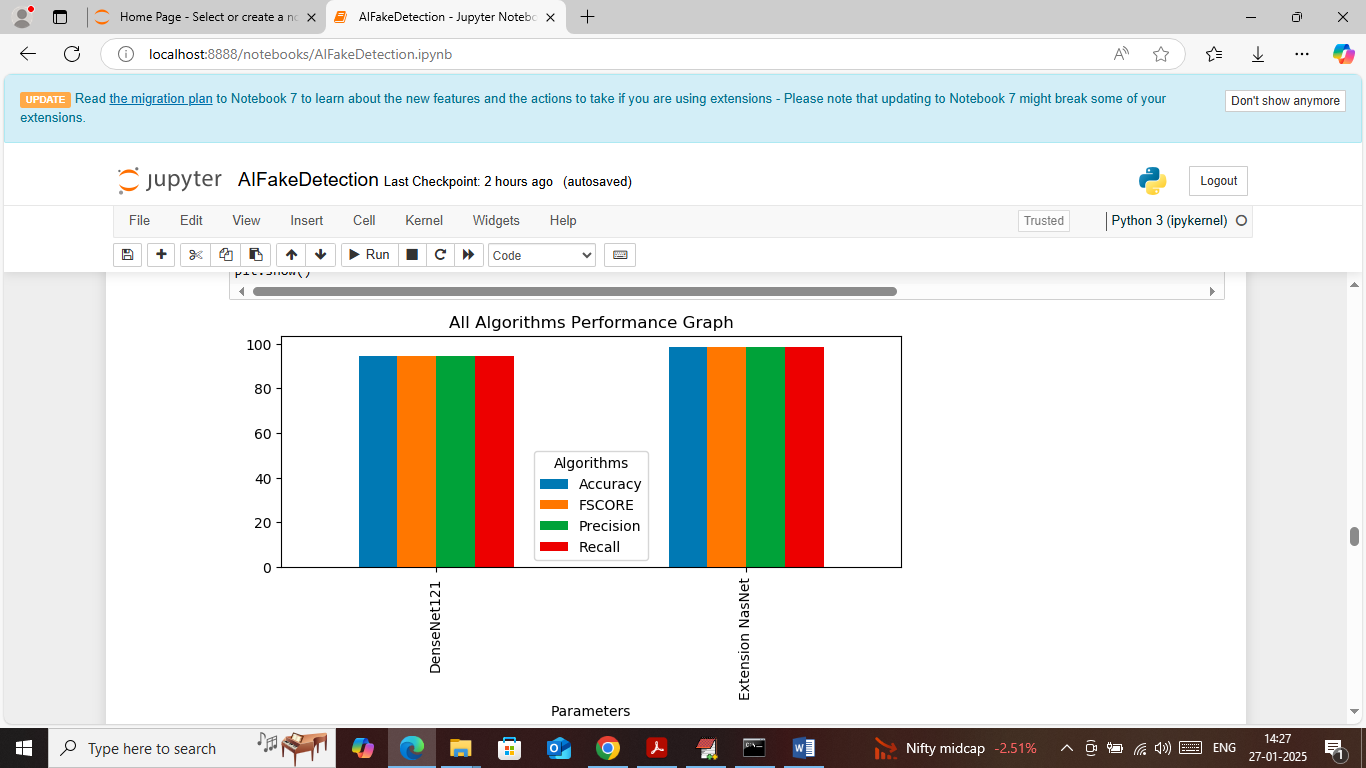
In above screen DenseNet121 got 94% accuracy on test images and then can see other metrics like precision, recall and FSCORE. In confusion matrix graph x-axis represents ‘Predicted Labels’ and y-axis represents True Labels and then yellow and light green boxes in diagonal represents correct prediction count and remaining blue boxes represents incorrect prediction count which are very few. In ROC graph x-axis represents ‘False Positive Rate’ and y-axis represents ‘True Positive Rate’ and if blue line comes on top of orange line then all predictions are correct and if goes below orange line then all predictions are false.



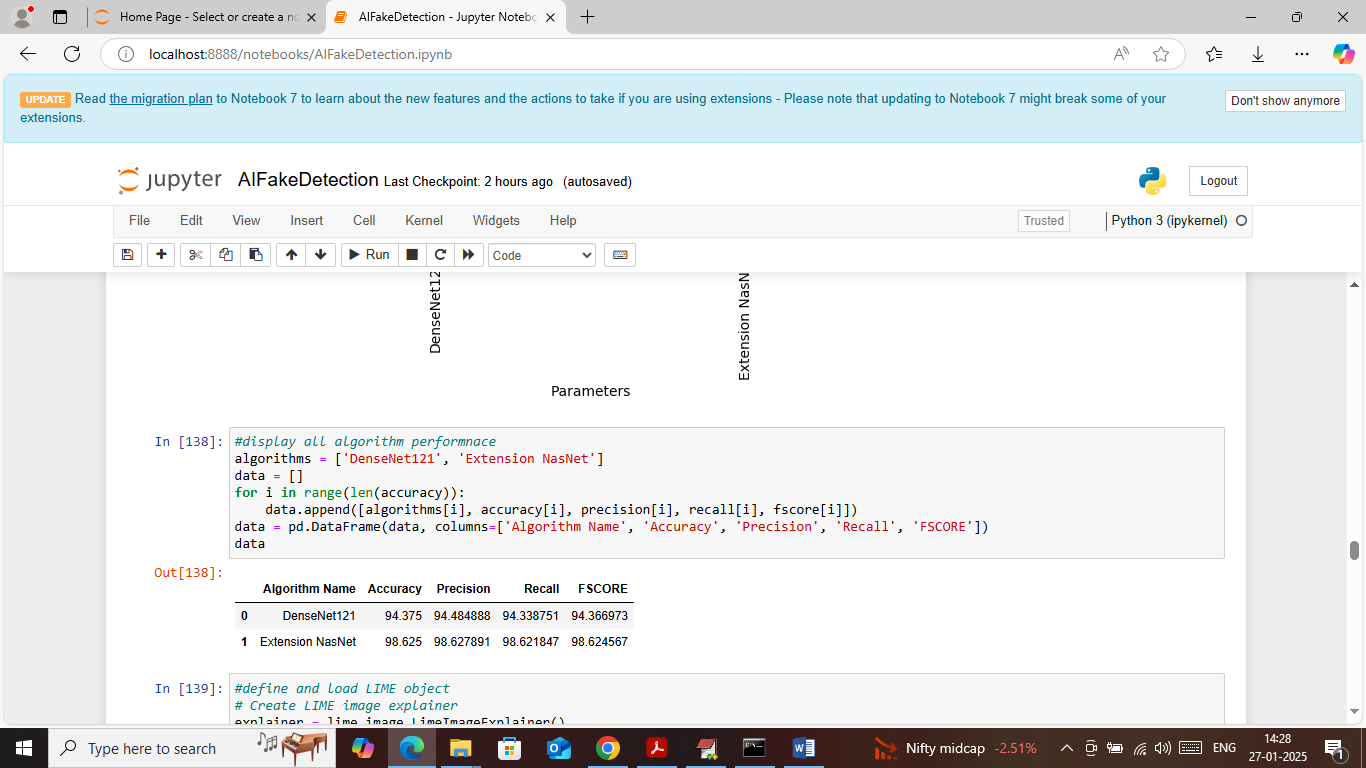
In above screen defining and training NASNET algorithm and after executing above block will get below output



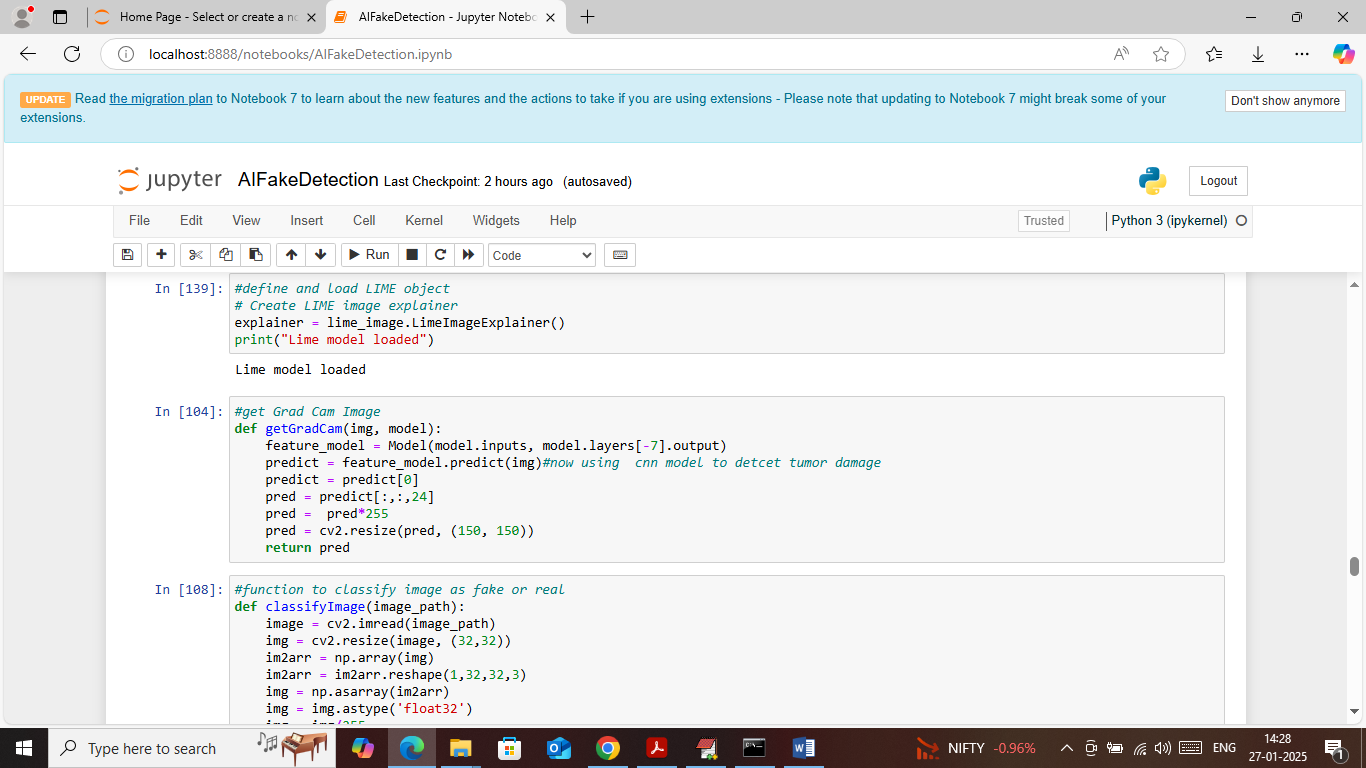
In above screen NASNET got 98% accuracy and can see other metrics also



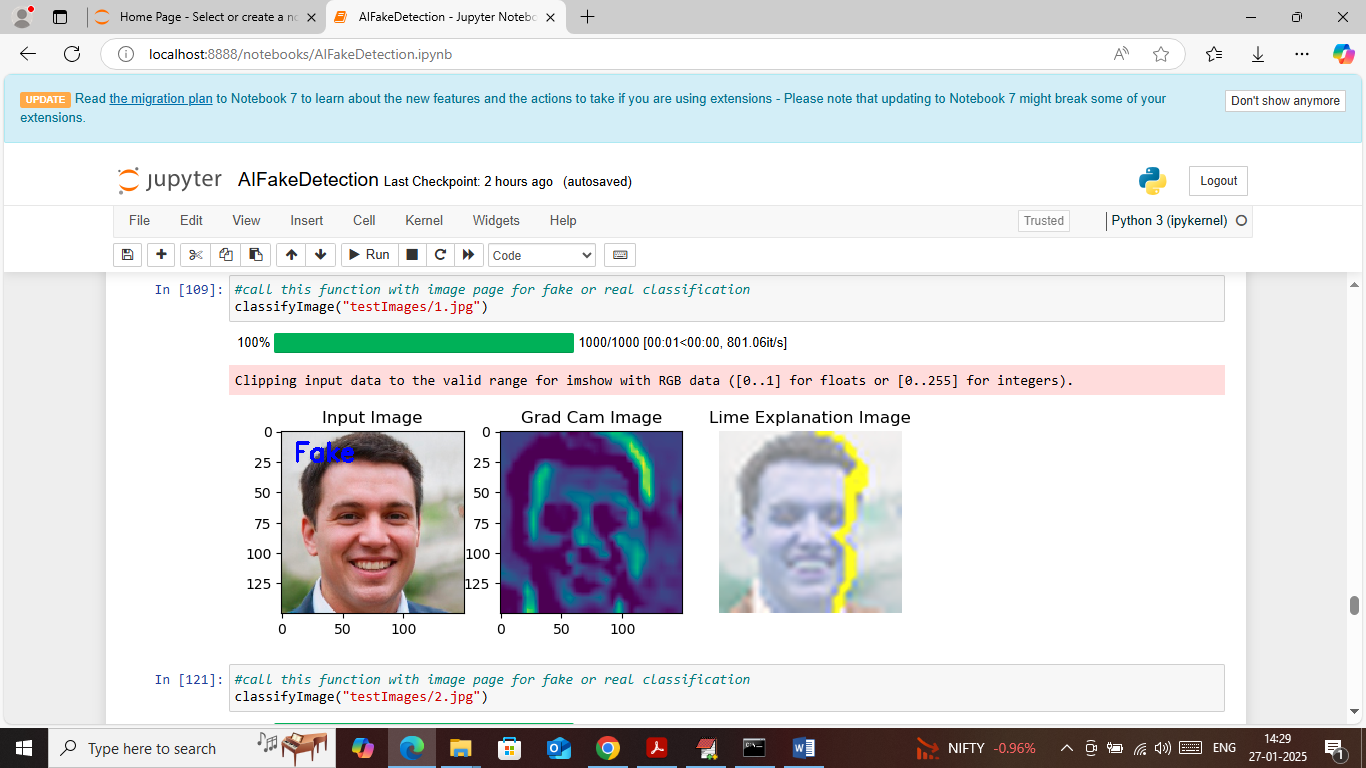
In above screen visualizing both algorithms performance where x-axis represents algorithm names and y-axis represent accuracy and other metrics in different colour bars and in both algorithms NASNET got high accuracy



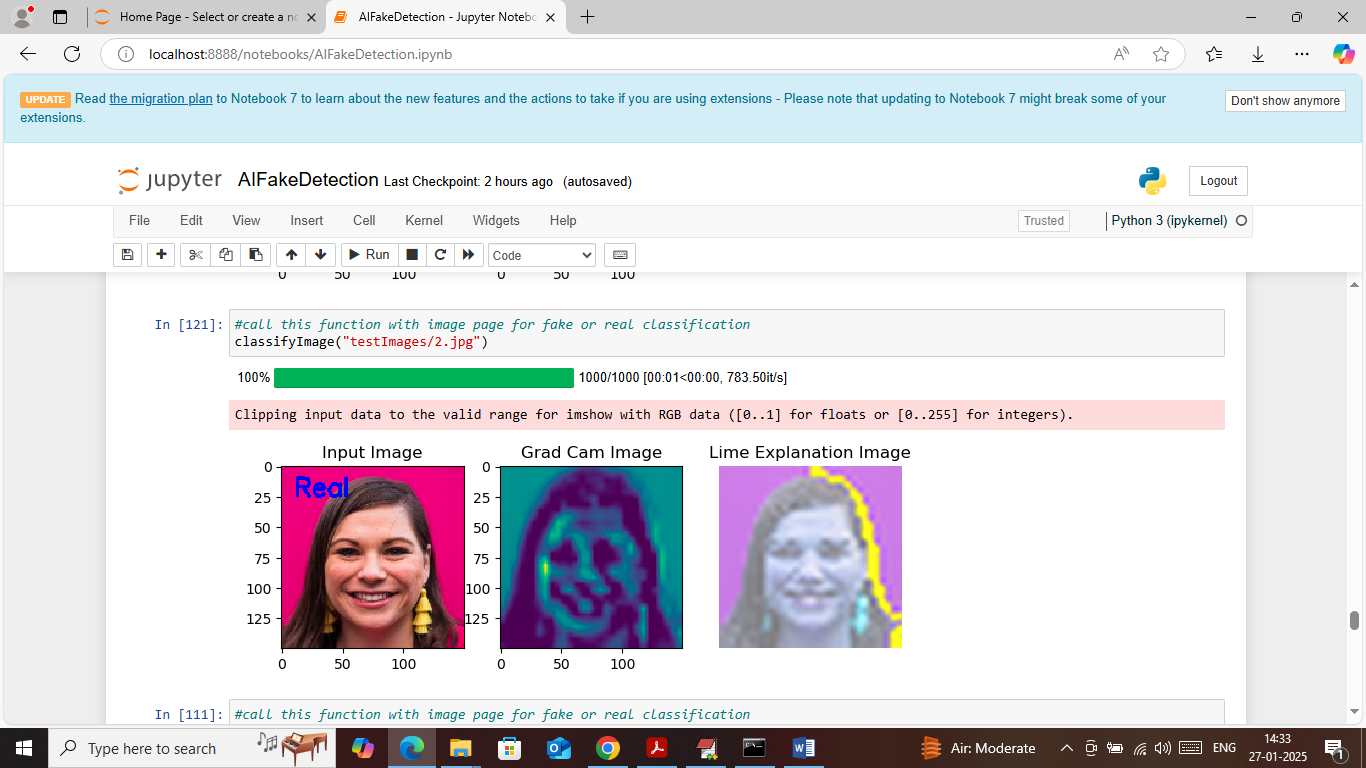
In above screen displaying both algorithm performance in tabular format



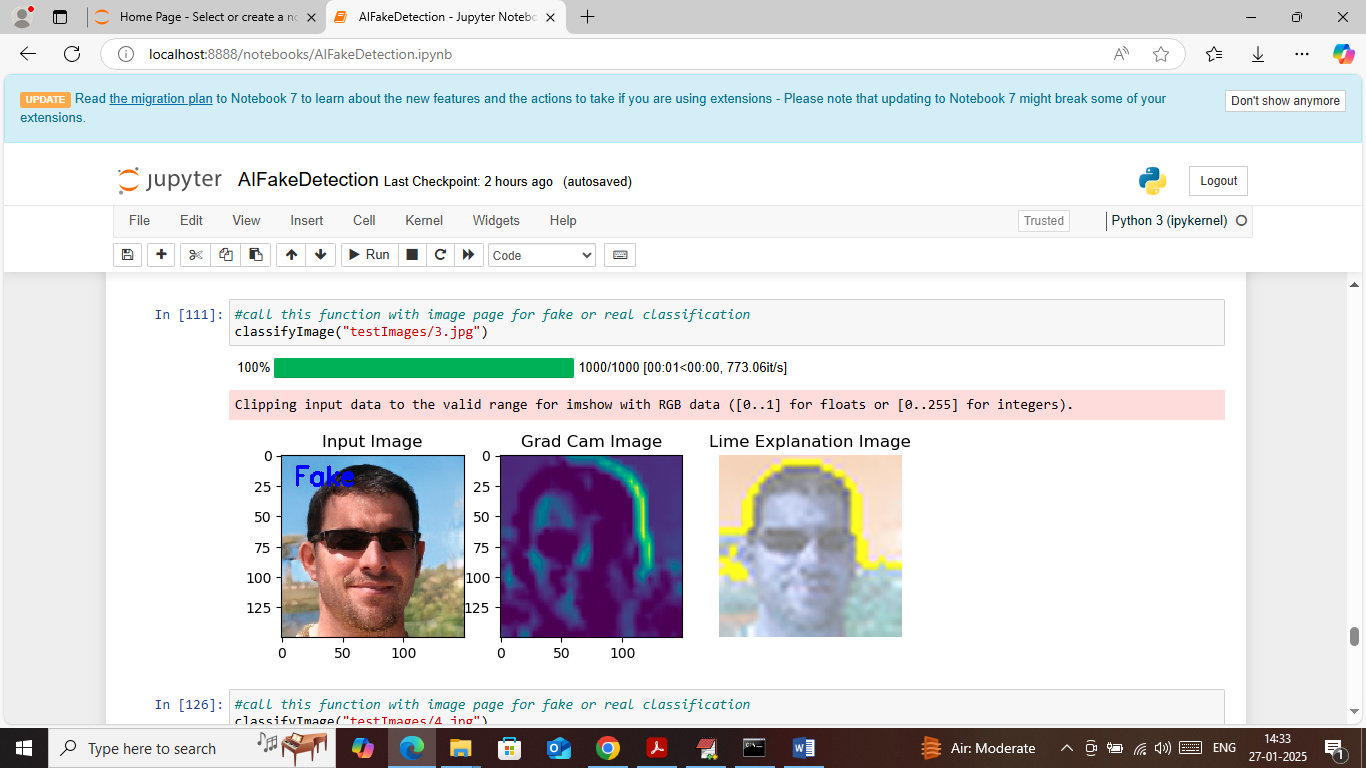
In above screen in first block loading LIME object and in second block defining function to get ‘GRAD CAM’ features mapping and in 3rd block defining function to classify image and ‘Fake or Real’.



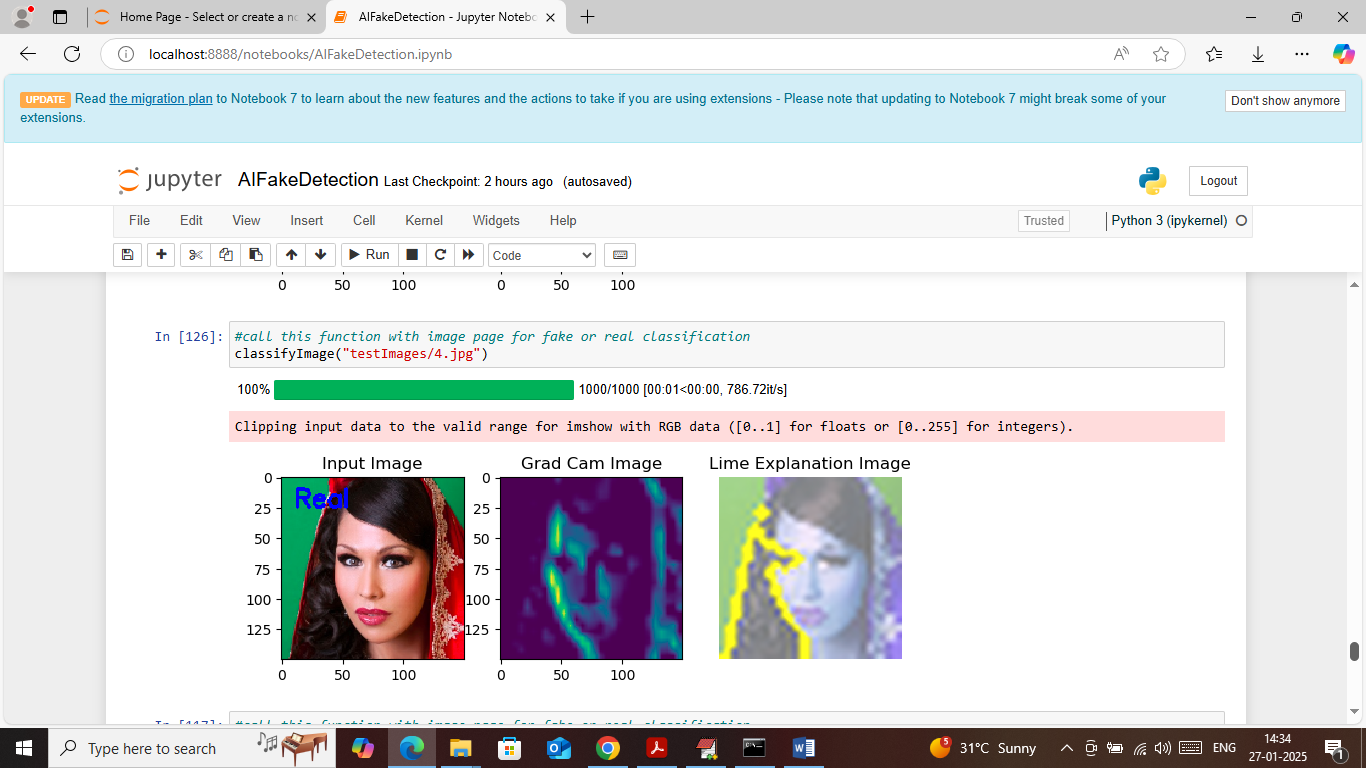
In above screen calling ‘Classify Image’ function along with test image path and then function will return 3 images where fist image is the INPUT image which is marked with predicted labels as ‘Fake or Real’ in blue text and in above screen input image is predicted as Fake. In 2nd image showing GRAD CAM features mapping image where the regions with dark colour are the features contributing most for prediction. In 3rd image showing LIME explain features in yellow colour which says those are the features contributing most for prediction. In above screen can see both GRAD-CAM and LIME showing same regions features which are contributing most for prediction.



In above screen testing another images which is predicted as REAL



Above image predicted as Fake and showing along with GRAD-CAM and LIME explanation



Above image predicted as Real.

Similarly change image path in calling function to classify any image