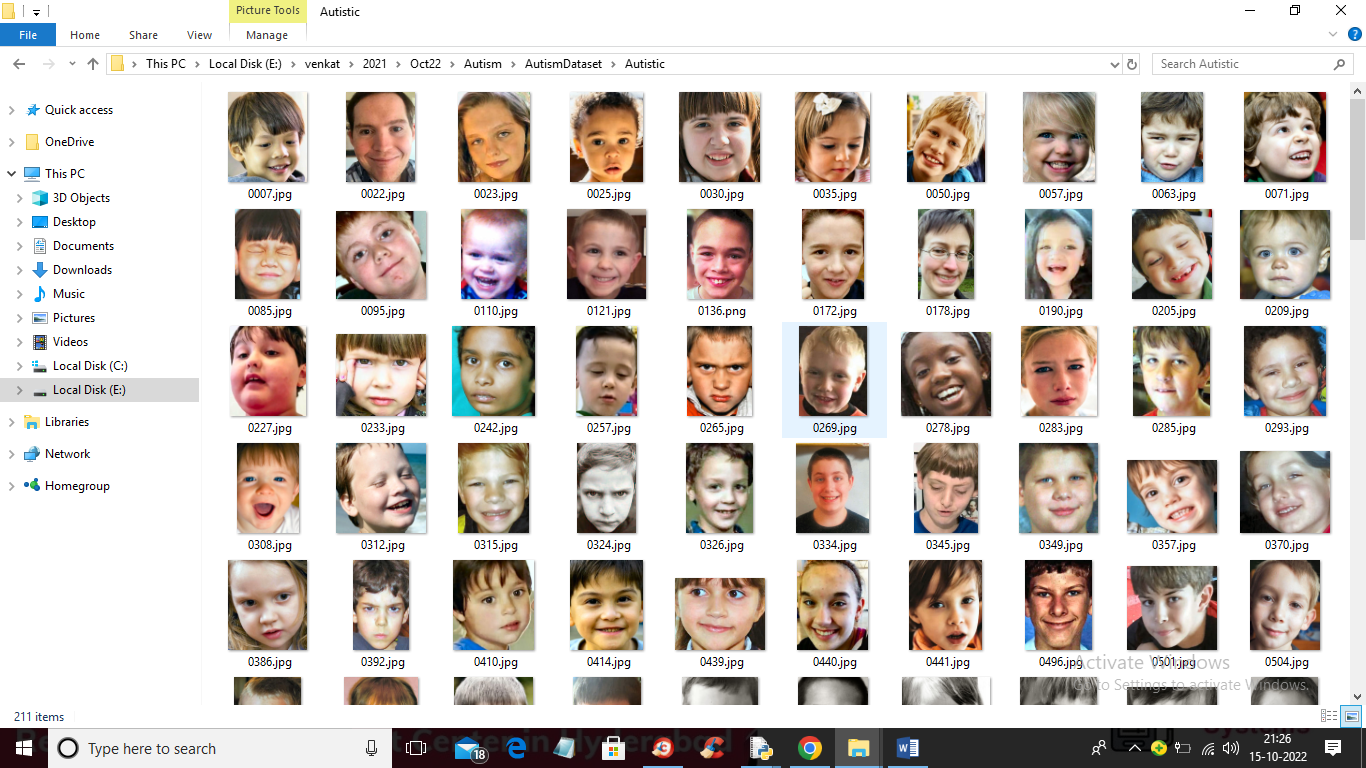
Autism Detection using Resnet50 & Xception Transfer Learning

In this project we are using Resnet50 and Xception algorithm with transfer learning technique to train Autism detection model. To train both algorithms we have used same dataset given by you. This dataset consists of two different classes such as ‘Autistic’ and ‘Non-Autistic’ and below screen showing images from dataset folder



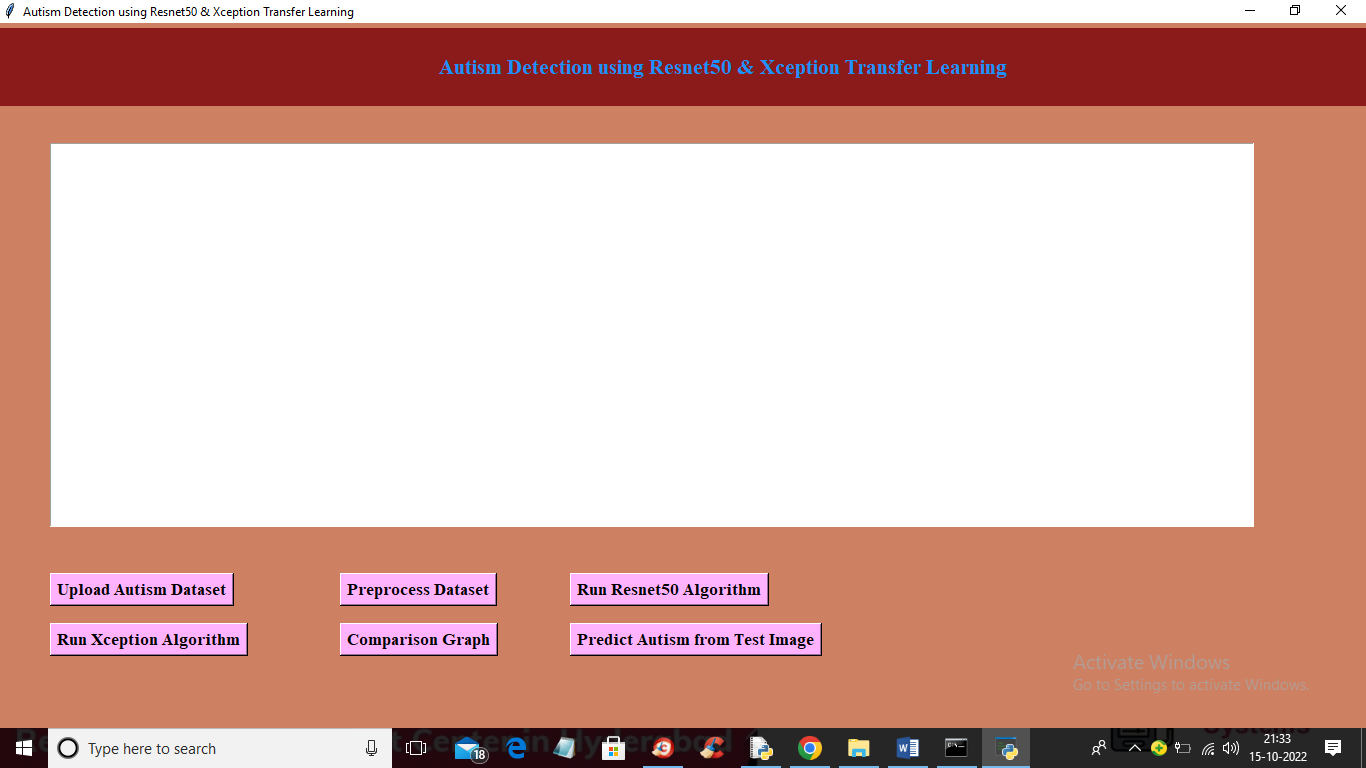
So by using above images we will train both algorithms.

To implement this project we have designed following modules

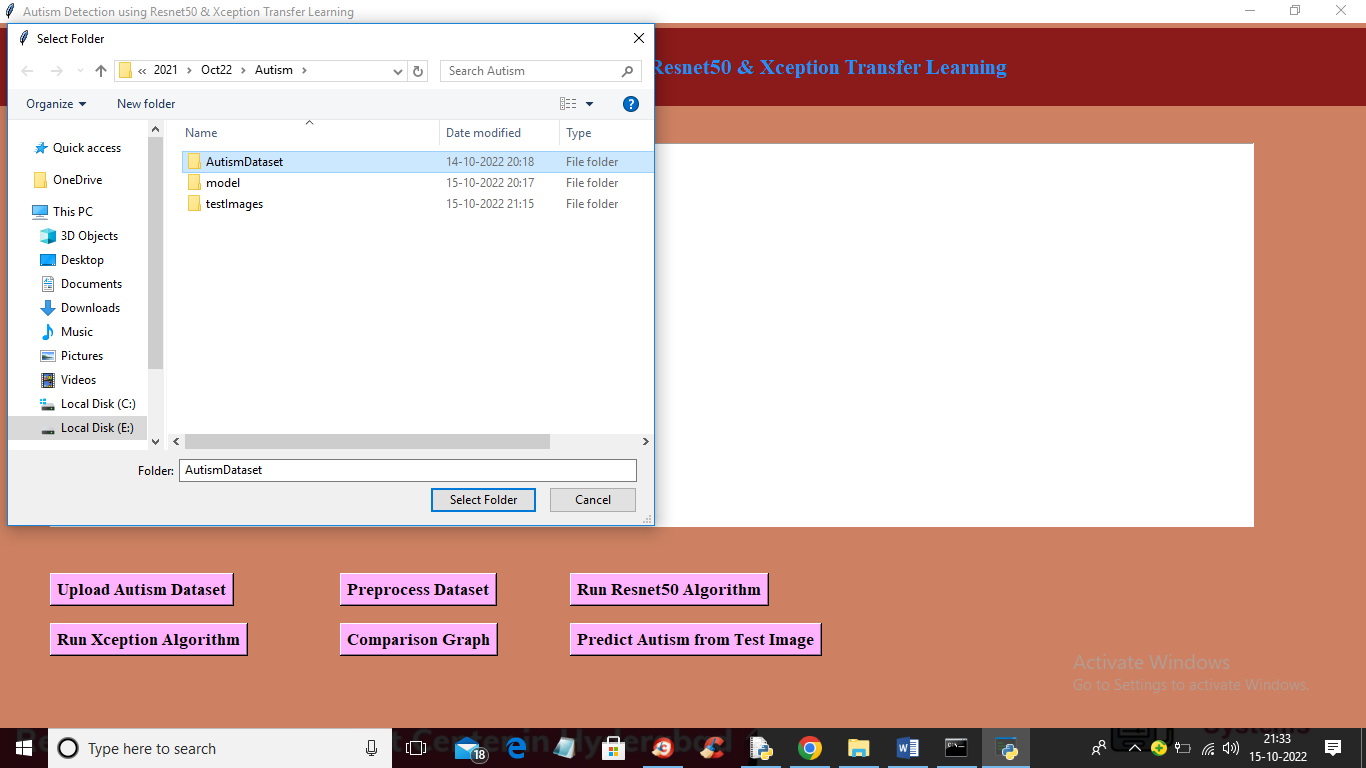
1. Upload Autism Dataset: using this module we will upload dataset images to application
2. Preprocess Dataset: using this module we will read each images and then resize all images to equal sizes and then normalize pixel values. Split dataset into train and test where application used 80% images for training and 20% for testing
3. Run Resnet50 Algorithm: processed train images will be input to Resnet50 transfer learning algorithm to train Autism prediction model. This model will be applied on test images to calculate prediction accuracy
4. Run Xception Algorithm: processed train images will be input to Xception transfer learning algorithm to train Autism prediction model. This model will be applied on test images to calculate prediction accuracy
5. Comparison Graph: using this module we will plot comparison graph between both algorithms
6. Predict Autism from Test Image: using this module we will upload test image and then algorithm will predict weather image is ‘Autistic’ or ‘Non- Autistic’.

SCREEN SHOTS

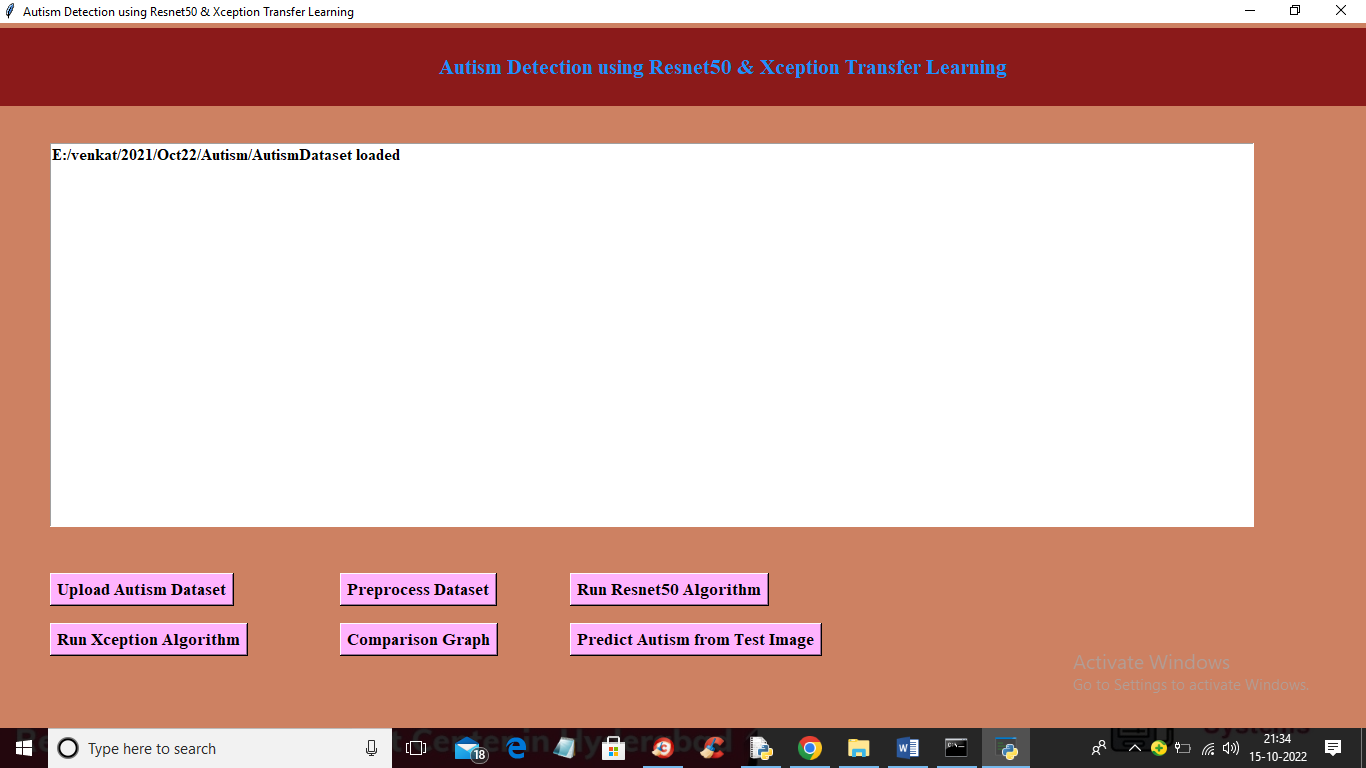
To run project double click on ‘run.bat’ file to get below screen



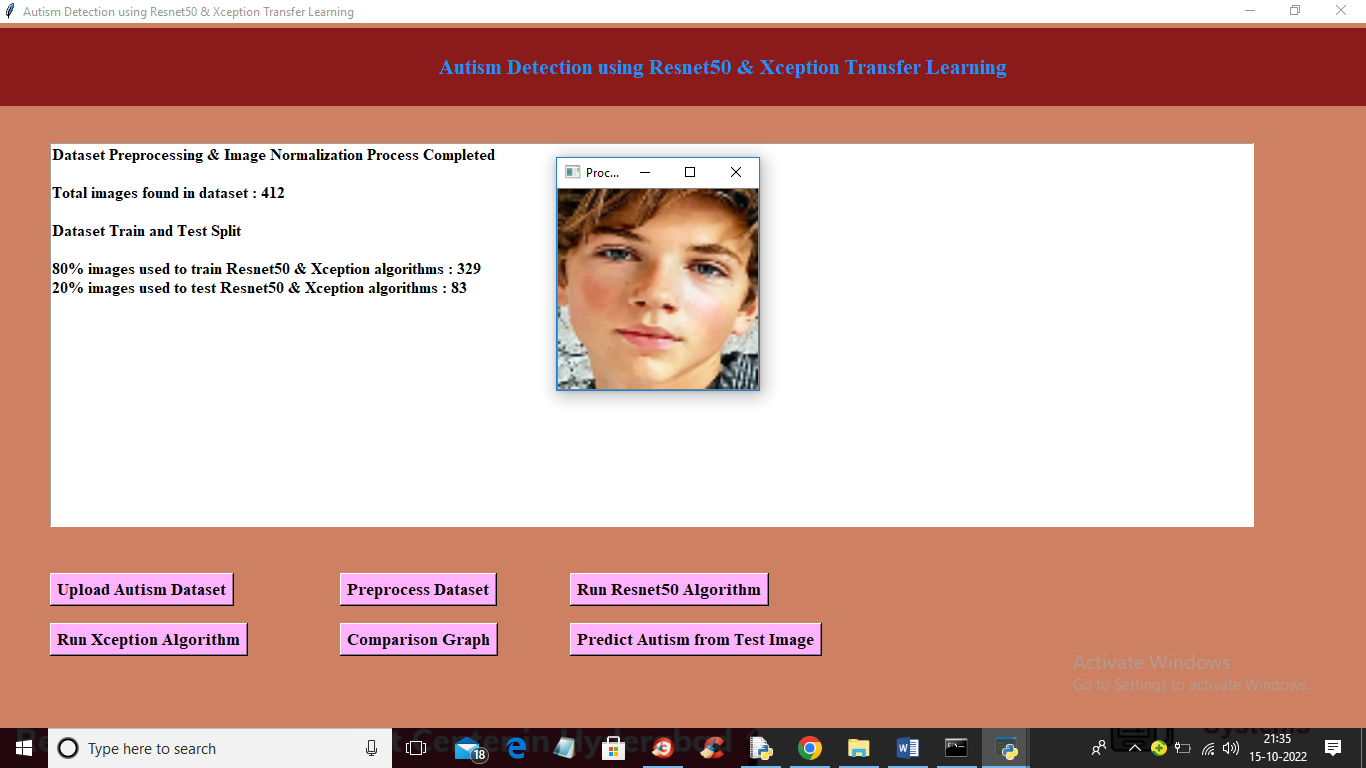
In above screen click on ‘Upload Autism Dataset’ button to upload dataset and get below output



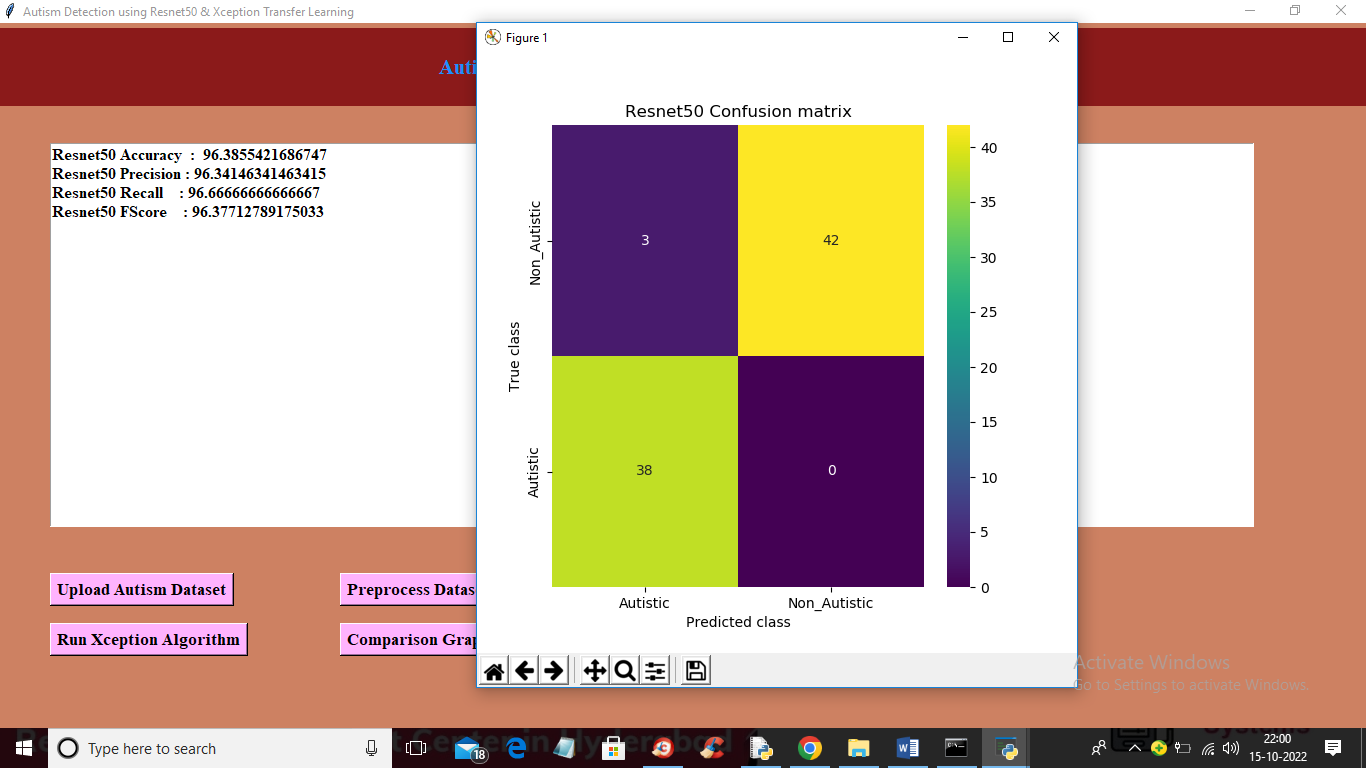
In above screen selecting and uploading ‘Autism Dataset’ folder and then click on ‘Select Folder’ button to load dataset and get below output



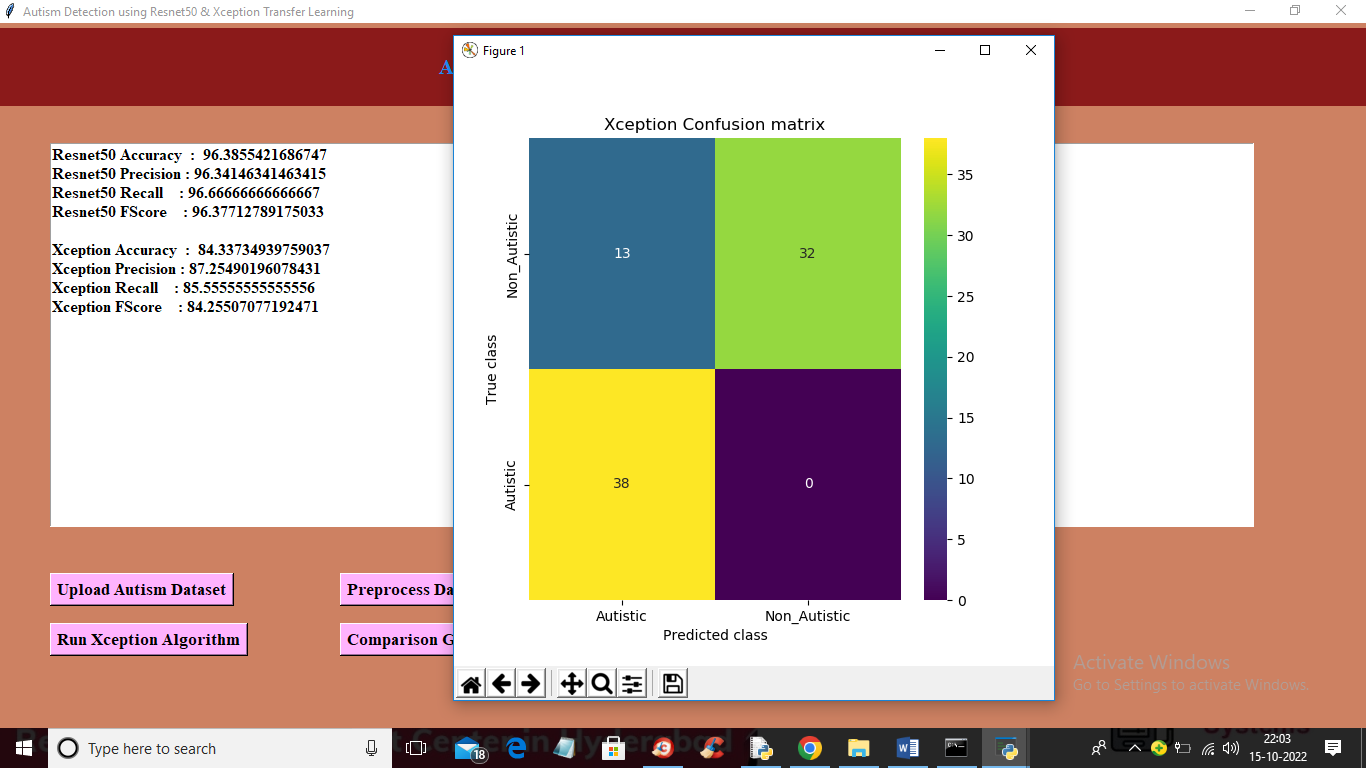
In above screen dataset loaded and now click on ‘Preprocess Dataset’ button to read and process all images and get below output



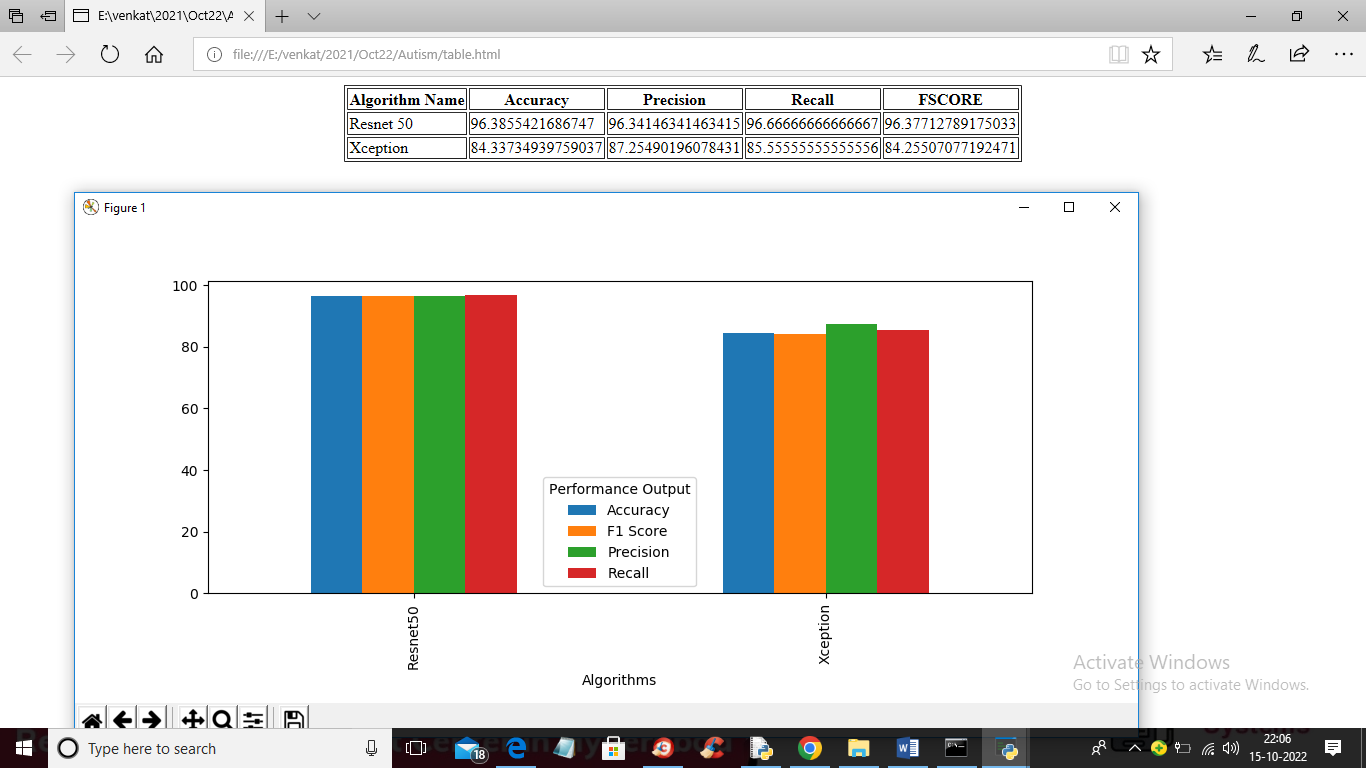
In above screen dataset processed and to check weather images processed properly I am showing sample image and now close above image and we can see dataset contains 412 images where application using 329 images for training and 83 for testing. Now click on ‘Run Resnet50 Algorithm’ button to train Resnet50 and get below output



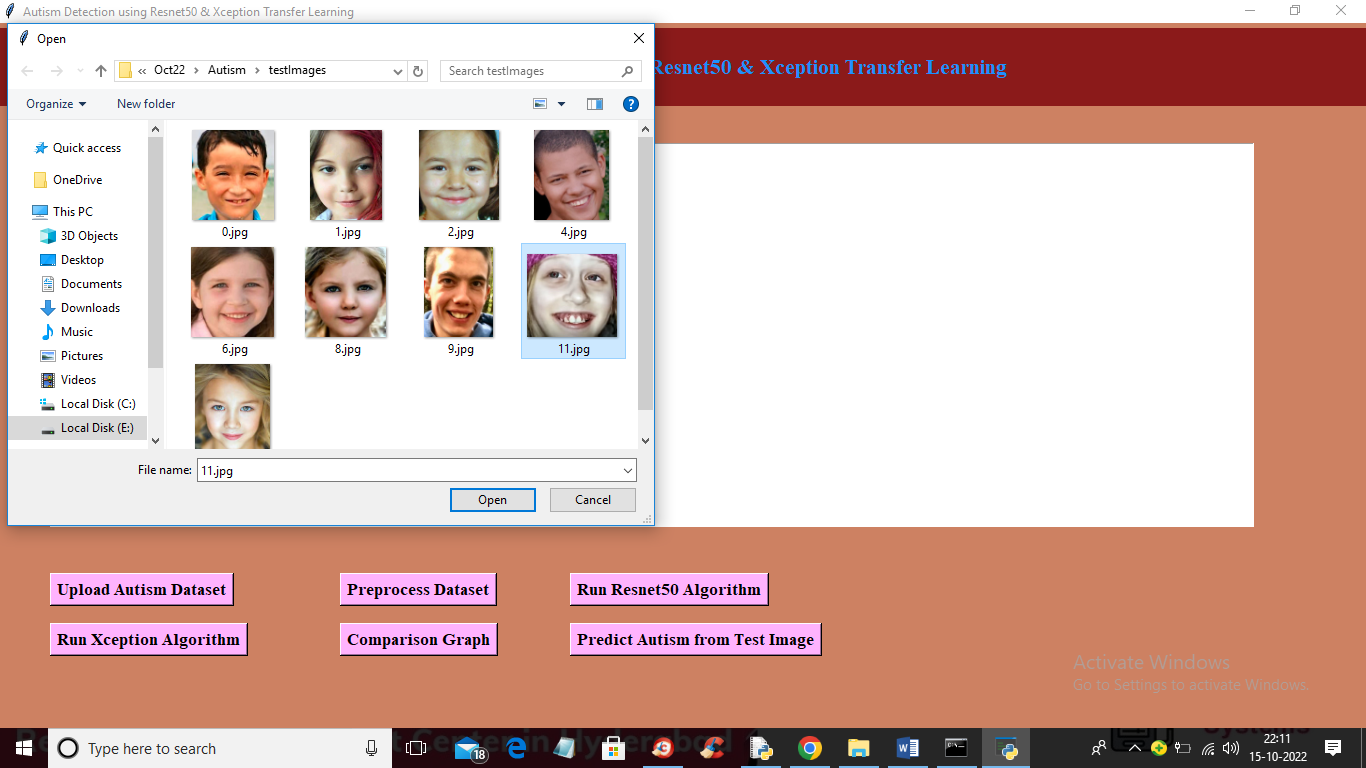
In above screen Resnet50 training completed and we got accuracy as 96% and in confusion matrix graph x-axis represents PREDICTED classes and y-axis represents TRUE CLASSES. In above graph same colour boxes represents INCORRECT prediction count and different colour boxes represents CORRECT prediction count and Resnet50 predict only 3 records as incorrectly. Now close above graph and the click on ‘Run Xception Algorithm’ button to train Xception and get below output



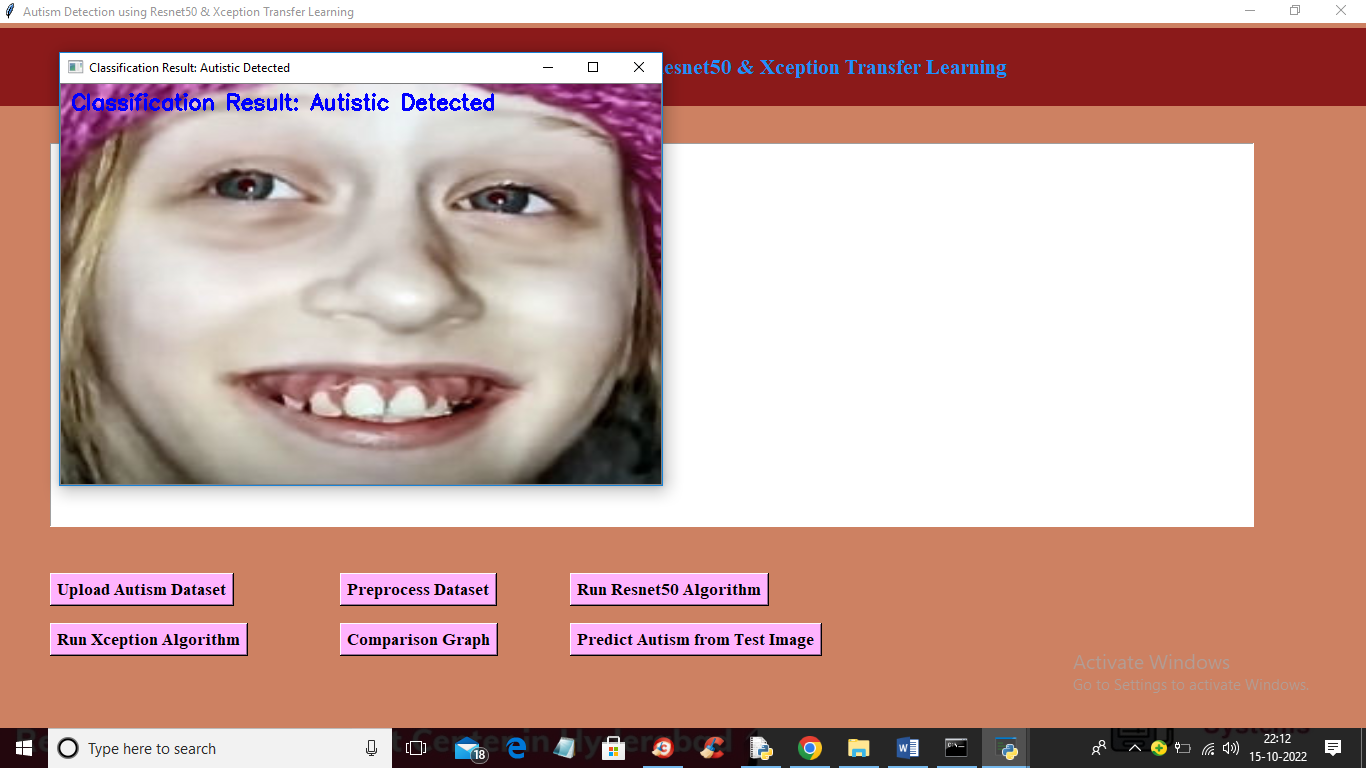
In above screen Xception training completed and with Xception we got 84% accuracy and in confusion matrix graph we can see Xception predict 13 records incorrectly. So from both algorithms Resnet50 got high accuracy. Now click on ‘Comparison Graph’ button to get below graph



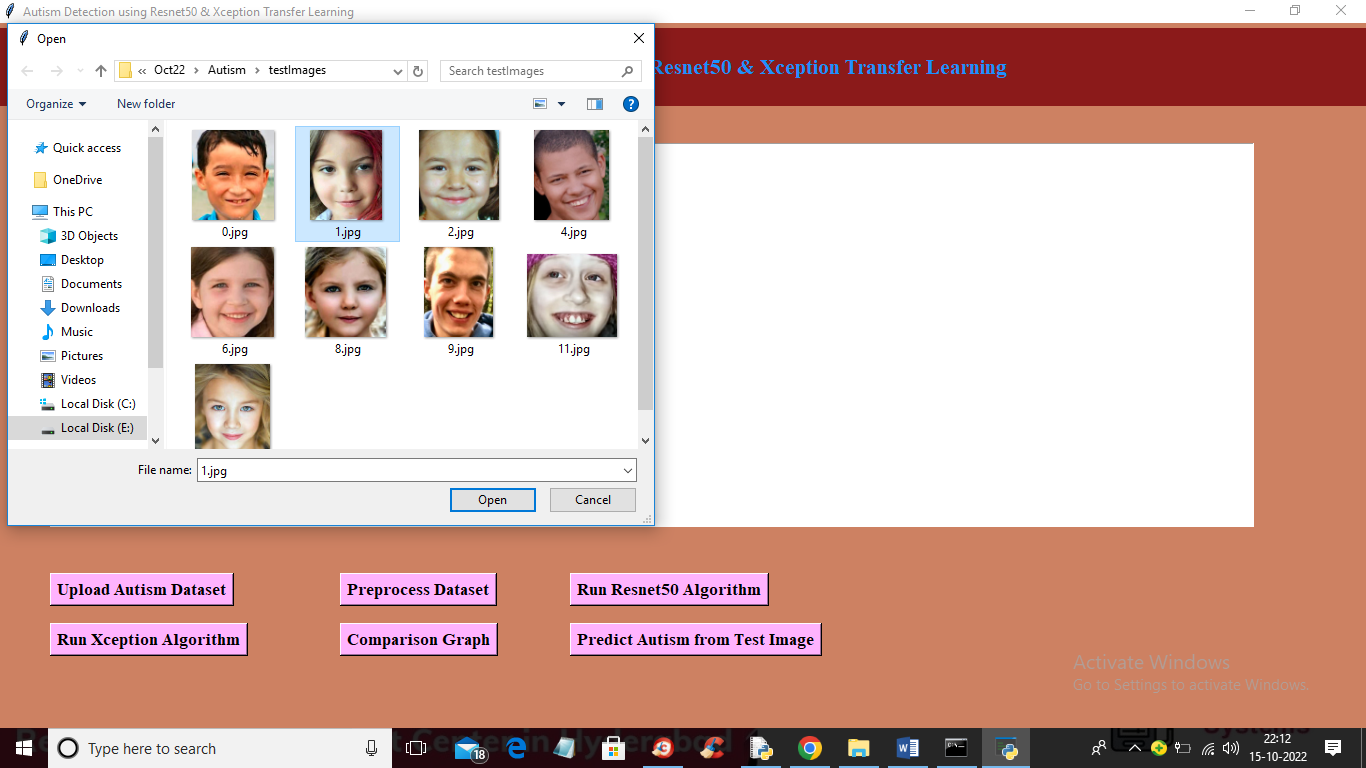
In above graph x-axis represents algorithm names and y-axis represents accuracy, precision, recall and F1SCORE in different colour bars. In above graph we can see Resnet50 got high performance. Now close above graph and then click on ‘Predict Autism from Test Image’ button to upload test image and get below output



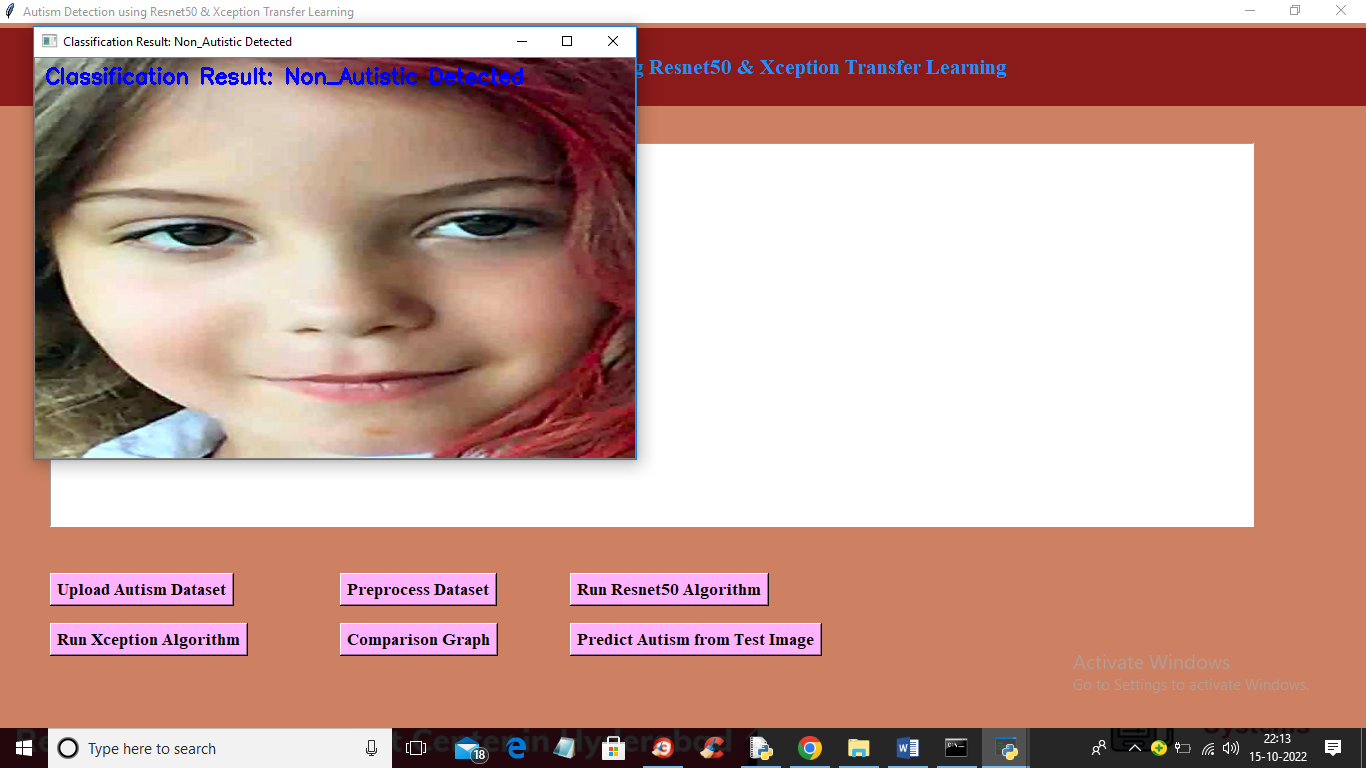
In above screen selecting and uploading ‘11.jpg’ and then click on ‘Open’ button to get below prediction output



In above screen image is classified as ‘Autistic Detected’ and now upload other image and get output



In above screen selecting and uploading ‘1.jpg’ and then click on ‘Open’ button to upload image and get below output



In above screen image is classified as ‘Non Autistic’. Similarly you can upload and test other images

Certainly! Here's a breakdown of the types of testing used in the code and what they are performing:

1. \*\*Unit Testing:\*\*

- \*\*Functionality:\*\* The code uses functions like `uploadDataset`, `preprocess`, `runResnet`, `runXception`, and `predict`. These functions handle specific tasks and can be considered as units. For example:

- `uploadDataset`: Loads the dataset.

- `preprocess`: Processes and prepares the dataset for training.

- `runResnet` and `runXception`: Train and evaluate ResNet50 and Xception models.

- `predict`: Makes predictions on test images.

2. \*\*Functional Testing:\*\*

- \*\*Functionality:\*\* Each function performs a distinct task related to dataset handling, model training, evaluation, and prediction. For instance:

- `uploadDataset`: Loads the dataset directory.

- `preprocess`: Processes the images, shuffles, and splits the dataset.

- `runResnet` and `runXception`: Define and train the ResNet50 and Xception models.

- `predict`: Makes predictions on individual test images.

3. \*\*Integration Testing:\*\*

- \*\*Functionality Integration:\*\* The code integrates different components, including:

- \*\*Model Training:\*\* Utilizes libraries like Keras to define and train models.

- \*\*Data Handling:\*\* Processes images using OpenCV and NumPy.

- \*\*GUI Integration:\*\* Integrates GUI elements using Tkinter.

- \*\*Metrics and Visualization:\*\* Utilizes libraries like Matplotlib, Pandas, Seaborn for visualizations and metric calculation.

4. \*\*Manual Testing:\*\*

- \*\*GUI Interaction:\*\* Allows users to interact with the GUI (buttons) to trigger specific functionalities like uploading data, preprocessing, model training, and prediction.

- \*\*Image Output:\*\* Displays images and classification results through the GUI using OpenCV.

5. \*\*Model Evaluation Testing:\*\*

- \*\*Metric Calculation:\*\* Calculates and displays performance metrics such as accuracy, precision, recall, and F1-score.

- \*\*Confusion Matrix:\*\* Generates confusion matrices to visualize model performance.

6. \*\*Data Testing:\*\*

- \*\*Dataset Splitting:\*\* Splits the dataset into training and testing subsets (`train\_test\_split`) to validate model performance on unseen data.

These types of testing collectively ensure that each function/module behaves as expected, different components integrate seamlessly, the GUI functions properly, and the models perform well on the given dataset.

Determining whether an image contains signs related to autism or not involves a classification process. In the context of the code you've provided, this classification is performed using deep learning techniques after the features are extracted from the images by models like ResNet50 and Xception.

The classification process involves these general steps:

1. \*\*Data Preparation:\*\*

- Images are preprocessed, resized, and converted into a format suitable for the model input.

2. \*\*Feature Extraction:\*\*

- Pre-trained models like ResNet50 or Xception extract features from the images. These models are used as a starting point to capture relevant patterns and features from the input images.

3. \*\*Model Training:\*\*

- Custom layers are added on top of the pre-trained models to create a new model. This combined model is then trained using a labeled dataset. The model learns to associate specific patterns extracted from the images with their corresponding labels (autistic or non-autistic).

4. \*\*Model Evaluation:\*\*

- The trained model is evaluated using a separate test dataset to assess its performance. Metrics like accuracy, precision, recall, and F1-score are calculated to measure how well the model classifies images into autism or non-autism categories.

5. \*\*Prediction:\*\*

- Once trained and evaluated, the model can be used to predict whether a new image contains signs related to autism or not. The model takes the image as input, performs feature extraction, and then classifies it as autistic or non-autistic based on the learned patterns.

In the code you've shared, after the model is trained and evaluated, the `predict()` function allows you to input a test image, preprocess it, and obtain predictions from the trained model.

Ultimately, the model makes a prediction based on the features it has learned during training. It identifies patterns in the image that align with the characteristics associated with autism, as determined by the labeled training data.