

## **ABSTRACT**

Recent Studies indicate that autism is a brain growth related disorder. Due to the structural and functional connectivity issues, the persons with autism has impact in making relationship and communication with others. Small and repeated conduct patterns are also seen in the disorder. Deep neural network efficiency is strong in many applications. In this project, we use KNN algorithm to detect the autism children in earlier life cycle stage.

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 DOMAIN OF PROJECT**

#### **MACHINE LEARNING:**

In the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance. Machine learning is a relatively new discipline within Computer Science that provides a collection of data analysis techniques.

Machine learning and AI-enabled devices can access and store more data than humans, including statistics from mind-blogging. Machines can detect patterns and use this information to generate solutions to any environmental problem.

### **1.2 PROBLEM STATEMENT**

- The problem statement is to find the child is autistic or non-autistic by using image dataset of children.
- In existing system you cannot find the child is autistic or non-autistic by visually.

### **1.3 OBJECTIVE**

(i)To make the image clear, we are using pre-processing method

(ii)Finally we are predicting whether the child is autistic or non-autistic by using CNN algorithm.

### **1.4 SOFTWARE REQUIREMENT**

**Software** : Jupyter Notebook

**Language** : Python

**Packages** : Opencv, Tensorflow

**Dataset** : Image Dataset Of Children

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 PREPROCESSING**

Preprocessing in Machine Learning refers to the technique of preparing cleaning and organizing the raw data to make it suitable for a building and training Machine Learning models. The aim of pre-processing is to improve the quality of the image so that we can analyse it in a better way. By preprocessing we can suppress undesired distortions and enhance some features which are necessary for the particular application we are working for

##### **2.1.1 RGB TO GREY SCALE IMAGE CONVERSION**

In preprocessing Moneeb Ahmed[1] proposed the color image to gray scale image conversion and the gray scale image algorithm experiment that algorithm has preserved the salient of the color image contrast, sharpness and shadow improve. The captured image in RGB image so it necessary to convert from RGB to gray scale image for image processing conversion. The values of three primary color RGB and encodes this linear intensity values convert in gray scale images using probability, The transformation equation using in existing algorithm.

$$\sum I_y = 0.296MR + 0.514MG + 0.1243MB$$

#### **2.2 MODEL TRAIN**

During training phase, KNN arranges the data (sort of indexing process) in order to find the closest neighbors efficiently during the inference phase. Otherwise, it would have to compare each new case during inference with the whole dataset making it quite inefficient

#### **2.3 CLASSIFICATION**

Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes

##### **2.2.1 K-NEAREST NEIGHBOUR ALGORITHM:**

Osman alay and Mustafa ulas [2] proposed a system, With the help of KNN algorithm it is used for the classification. There is no operation on the data except to convert it to numerical values. Precision, Recall, F1 Score and support values are calculated

## **LIMITATIONS OF EXISTING SYSTEM**

For the above existing system the accuracy is given less using child image

## CHAPTER 3

### SYSTEM DESIGN AND ARCHITECTURE

#### 3.1 ARCHITECTURE DIAGRAM

(i) Images are pre-processed into RGB to grey and noise removal and image enhancements are done under this process.

(ii) The model is trained using knn classifier.

(iii) In classification, using knn algorithm to predict the child is normal or abnormal child.

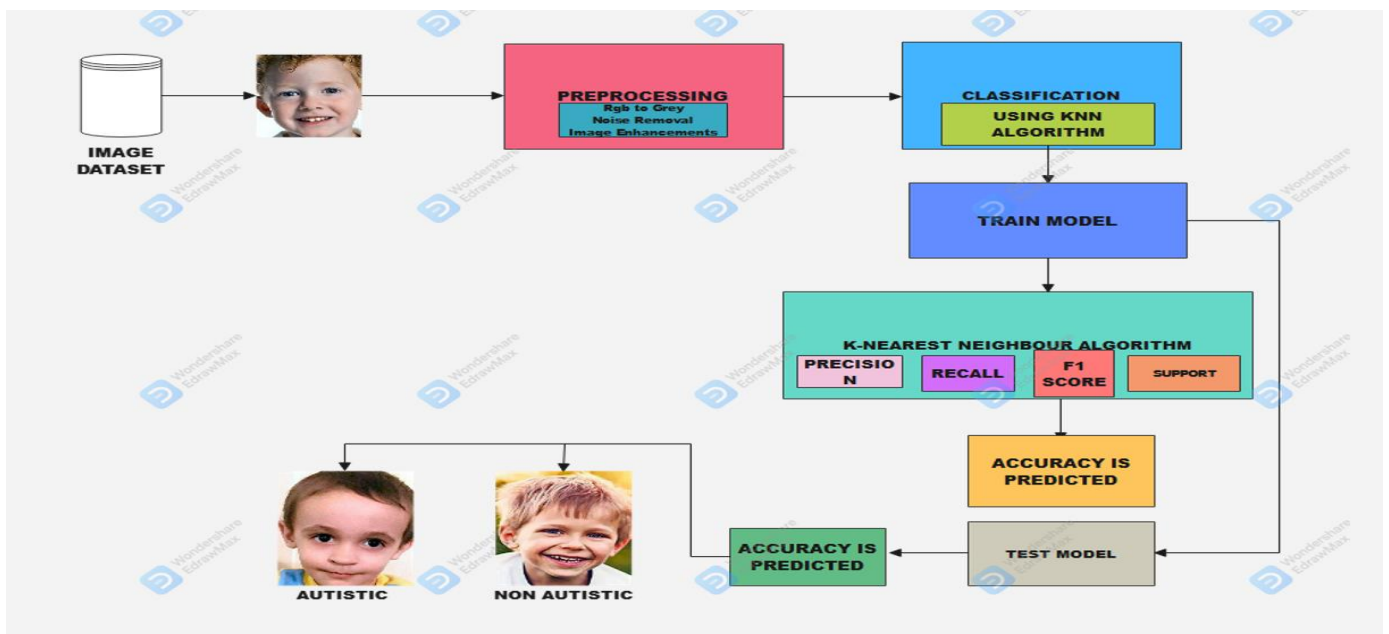


Figure 3.1: System Architecture

### 3.3 MODULES DESCRIPTION

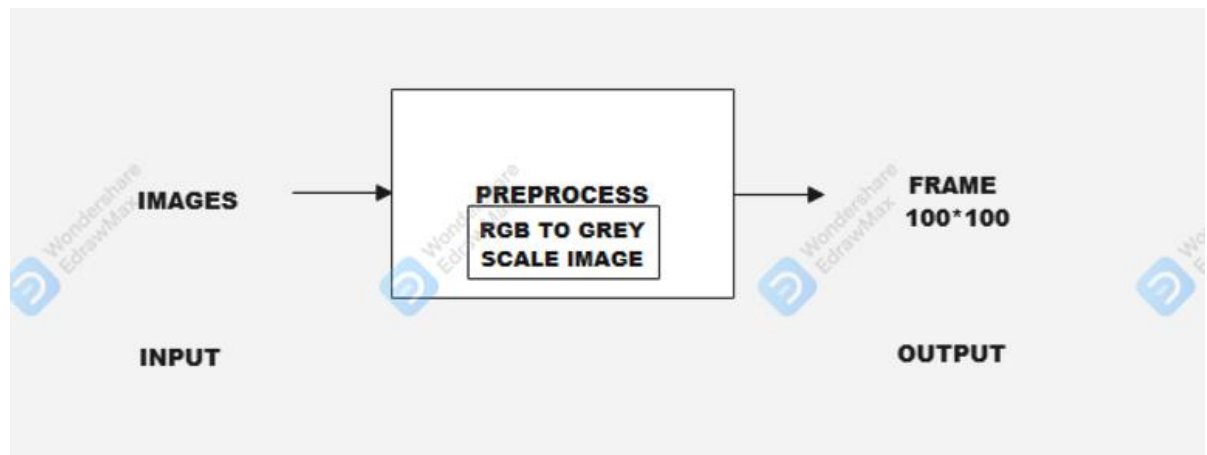
There are 3 components in the system. They are

- [1] Pre-processing
- [2] Model Training
- [3] Classification

#### 3.3.1 PRE-PROCESSING

In this module the given input image is converted into grey format. We need to check any external noise is present. If its present that noise has to be removed by median filter. Then the Image is enhanced and sent for segmentation.

#### BLOCK DIAGRAM



#### 3.2 MODEL TRAINING

In this step, we have to split the dataset into the Training set, on which the model will be trained and the Test set, on which the trained model will be applied to classify the results. In this the `test_size=0.25` denotes that 25% of the data will be kept as the Test set and the remaining 75% will be used for training as the Training set. The class KNN neighbour class is imported and is assigned to the variable "model". The `model.fit` function is fitted with `x_train` and `y_train` on which the model will be trained.

### 3.3 CLASSIFICATION

In this module the image is been classified using K-Nearest Neighbour algorithm. This algorithm will help in finding out whether the child is non autistic child or autistic child.

#### 3.3.1 PREDICTING THE TEST DATA

In this step, the `model.predict()` function is used to predict the values for the Test set and the values are stored to the variable `y_predict`

#### 3.3.2 CONFUSION MATRIX AND ACCURACY

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

This is a step that is mostly used in classification techniques. In this, we see the Accuracy of the trained model and plot the confusion matrix. The confusion matrix is a table that is used to show the number of correct and incorrect predictions on a classification problem when the real values of the Test Set are known. It is of the format, using confusion matrix we can calculate the precision, recall, f1score and support values are calculated.

<b>True Positive</b>	<b>False Positive</b>
<b>False Negative</b>	<b>True Negative</b>

## **PRECISION**

The precision can be calculated using this formula

$$\text{Precision} = \text{TruePositives} / (\text{TruePositives} + \text{FalsePositives})$$

## **RECALL**

The recall can be calculated using this formula

$$\text{Recall} = \text{TruePositives} / (\text{TruePositives} + \text{FalseNegatives})$$

## **F1- SCORE**

The f1score can be calculated using this formula

$$\text{F1-score} = (2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$



## CHAPTER 4

### IMPLEMENTATION AND ALGORITHM

#### 4.1 PRE-PROCESSING

In pre-processing the images are converted from rgb to grey scale image format.

**PROCESS:** In pre-process the original image is converted from RGB to grayscale image.

**INPUT:** Images are taken from the dataset

- 
- 1: Start
  - 2: Read the images non autistic as zero and autistic as one
  - 3: Divide the images into categories
  - 4: Convert the images from rgb to grey scale image using the formula  
$$\text{imgGray} = 0.2989 * R + 0.5870 * G + 0.1140 * B$$
  - 5: plot the image as grey scale format

**Algorithm 4.1 Pre-processing**

---

**OUTPUT:** Images are displayed in a grey scale format

#### 4.2 MODEL TRAINING:

In model training the images are trained using KNN classifier

**PROCESS:** In model training, the trained model splited into train data and test data

**INPUT:** Image dataset is given as input

**OUTPUT:** The model is trained

- 
- 1: Start
  - 2: Read the images as autistic as 0's and 1's as non autistic using label encoder
  - 3: The dataset is splited into 75% as train dataset and 25% as test data
  - 4: Splited dataset is given using this code  
`(trainX, testX, trainY, testY ) = train_test_split(data, labels, test_size= 0.25, random_state=38)`
  - 5: The model is trained and fitted using this code  
`model = KNeighborsClassifier(n_neighbors=2, n_jobs=-1)`  
`model.fit(trainX, trainY)`

---

**Algorithm 4.2 model training**

### 4.3 CLASSIFICATION

Here we use the concept of K-Nearest Neighbour algorithm.

**PROCESS:** The image is classified using autistic or non autistic child using KNN algorithm

**INPUT:** children image is given as input

**OUTPUT:** Final output is displayed as autistic child or non autistic child using accuracy.

---

#### Algorithm:

1: start

2: using KNN algorithm, testing dataset of children image is taken and accuracy is calculated and we are predicting the child is autistic or non-autistic child.

3. The confusion matrix and accuracy is calculated using precision, recall, f1 score, support

4. The precision is calculated using this formula

$$\text{Precision} = \frac{\text{TruePositives}}{(\text{TruePositives} + \text{FalsePositives})}$$

5. The recall is calculated using this formula

$$\text{Recall} = \frac{\text{TruePositives}}{(\text{TruePositives} + \text{FalseNegatives})}$$

6. The f1 score is calculated using this formula

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

7. The final output is given in classification format

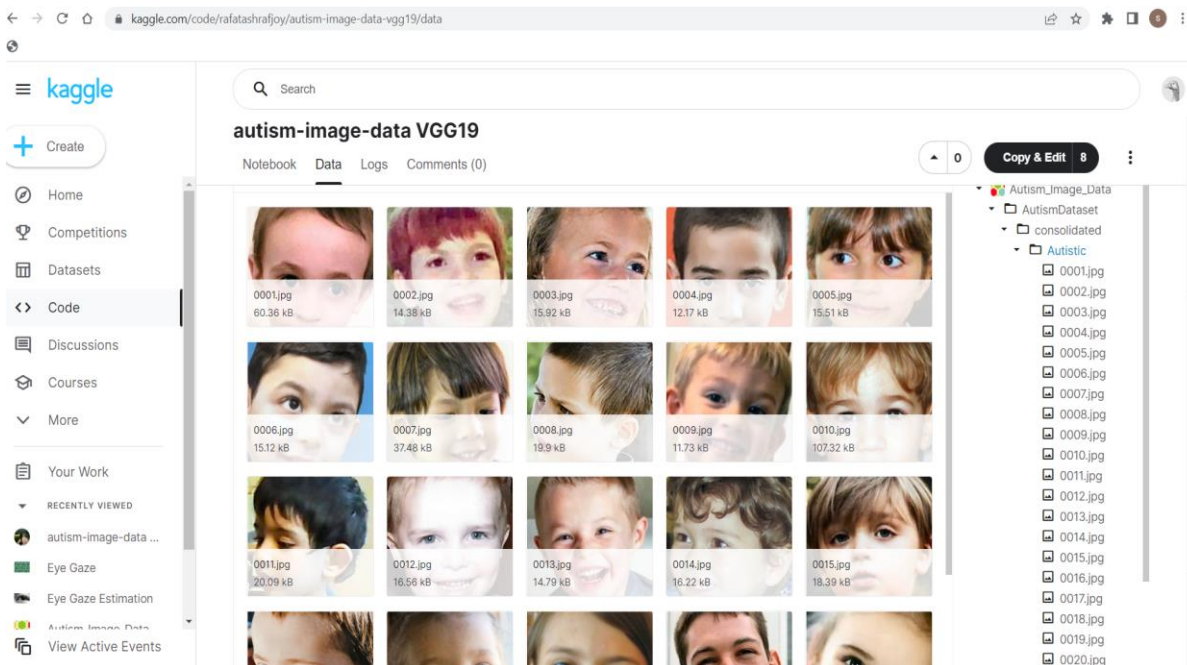
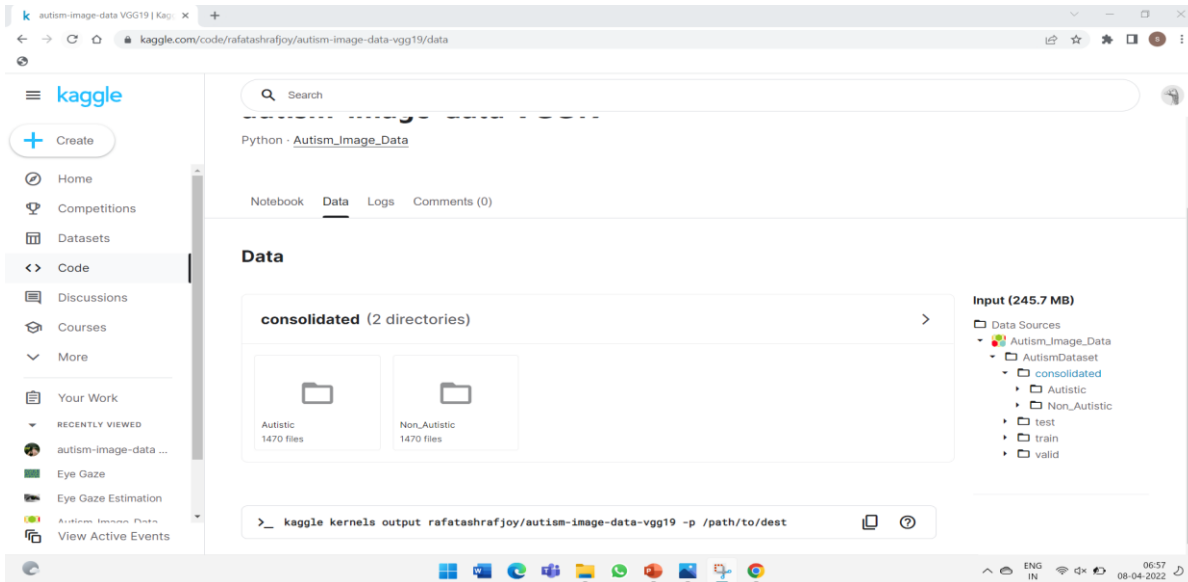
---

#### Algorithm 4. 3Classification

# CHAPTER 5

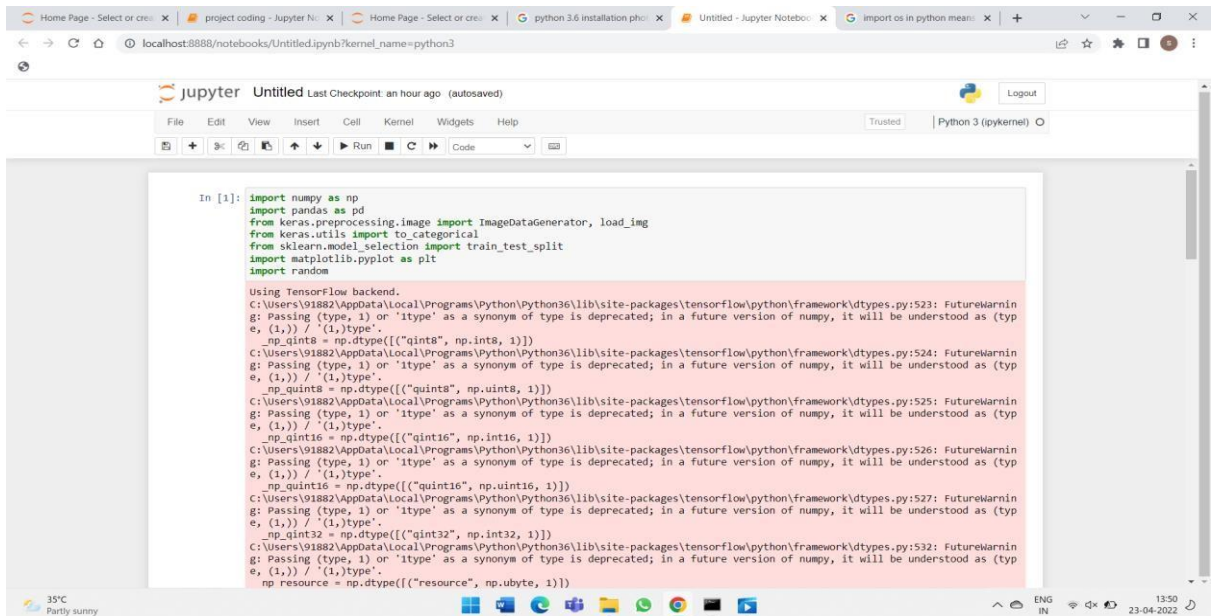
## OUTPUT AND SCREENSHOTS

### 5.1 DATASET



## 5.2 PRE-PROCESSING

### PACKAGE INSTALLATION



The screenshot shows a Jupyter Notebook with the following code in cell [1]:

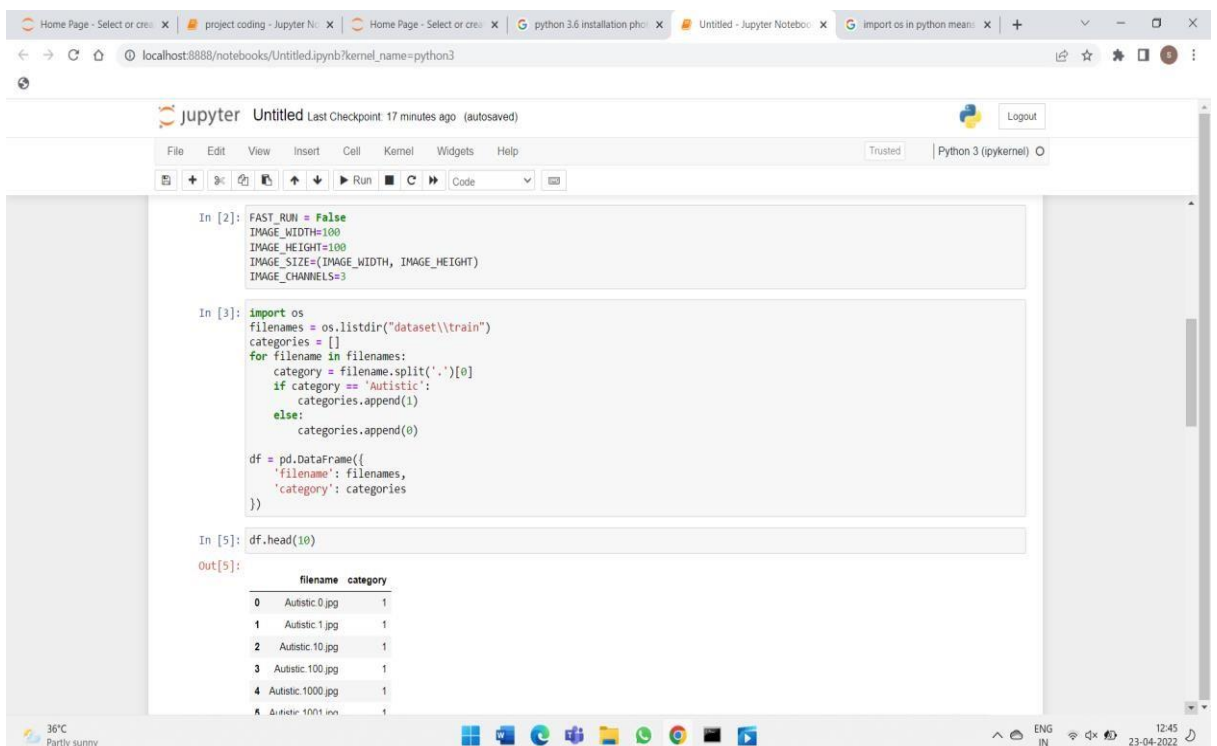
```
In [1]: import numpy as np
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random

Using TensorFlow backend.
```

The output of the code shows several FutureWarning messages from TensorFlow regarding deprecated type aliases in NumPy. The warnings are as follows:

- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
- g: Passing (type, 1) or 'dtype' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.

### INSERTING DATASET



The screenshot shows a Jupyter Notebook with the following code in cells [2], [3], and [5]:

```
In [2]: FAST_RUN = False
IMAGE_WIDTH=100
IMAGE_HEIGHT=100
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3

In [3]: import os
filenames = os.listdir("dataset\\train")
categories = []
for filename in filenames:
    category = filename.split('.')[0]
    if category == 'Autistic':
        categories.append(1)
    else:
        categories.append(0)

df = pd.DataFrame({
    'filename': filenames,
    'category': categories
})

In [5]: df.head(10)
```

The output of the code in cell [5] is a DataFrame with 10 rows and 2 columns: filename and category. The data is as follows:

	filename	category
0	Autistic 0.jpg	1
1	Autistic 1.jpg	1
2	Autistic 10.jpg	1
3	Autistic 100.jpg	1
4	Autistic 1000.jpg	1
5	Autistic 1001.jpg	1



The screenshot shows a Jupyter Notebook titled "Knn image classifier1" in the JupyterLab interface. The notebook is running on a local host (localhost:8888). The first cell contains the following code:

```

110 [44]: data=[]
        for image in imagePath:
            img = cv2.imread(image)
            img = cv2.resize(img, (32, 32), interpolation = cv2.INTER_AREA)
            data.append(img)

In [22]: data = np.array(data)
        data

```

The output of the first cell is displayed below the code:

```

Out[22]: array([[[[241, 243, 246],
                  [241, 244, 247],
                  [242, 246, 245],
                  ...,
                  [245, 252, 251],
                  [249, 255, 248],
                  [254, 255, 249]],
                [[241, 244, 247],
                  [242, 244, 245],
                  [243, 245, 246],
                  ...,
                  [243, 252, 251],
                  [248, 254, 249],
                  [253, 255, 249]],
                [[241, 244, 248],
                  [241, 244, 247],
                  [243, 245, 247],
                  ...,
                  [243, 252, 251],
                  [248, 254, 249],
                  [253, 255, 249]]],
                ...,
                [[241, 244, 248],
                  [241, 244, 247],
                  [243, 245, 247],
                  ...,
                  [243, 252, 251],
                  [248, 254, 249],
                  [253, 255, 249]]],
                ...,
                [[241, 244, 248],
                  [241, 244, 247],
                  [243, 245, 247],
                  ...,
                  [243, 252, 251],
                  [248, 254, 249],
                  [253, 255, 249]]]])

```

The second cell contains the following code:

```

In [23]: le=LabelEncoder()
        labels=le.fit_transform(imagename)
        print(labels)

[0 0 0 ... 1 1 1]

```

The screenshot shows a Jupyter Notebook environment with the following content:

```
[[[[135, 155, 150, ..., 71, 59, 53],
...,
[ 17, 29, 52, ..., 62, 76, 107],
[ 65, 73, 104, ..., 61, 88, 130],
[139, 139, 139, ..., 184, 184, 184]], dtype=uint8)

In [26]: (trainX, testX, trainY, testY) = train_test_split(data, labels, test_size= 0.25, random_state=38)

In [27]: 

Out[27]: KNeighborsClassifier(n_jobs=-1, n_neighbors=2)

In [28]: print(classification_report(testY, model.predict(testX), target_names=le.classes_))
```

	precision	recall	f1-score	support
Artistic	0.64	0.66	0.65	297
Non_Artistic	0.69	0.68	0.69	338
accuracy			0.67	635
macro avg	0.67	0.67	0.67	635
weighted avg	0.67	0.67	0.67	635

```
In [29]: imagePathtest = getListOfFiles(r"D:\Autism_image_classification\test')

In [30]: imagePathtest

Out[30]: ['D:\\Autism_image_classification\\test\\Artistic.0.jpg',
'D:\\Autism_image_classification\\test\\Artistic.1.jpg',
'D:\\Autism_image_classification\\test\\Artistic.10.jpg',
'D:\\Autism_image_classification\\test\\Artistic.100.jpg',
```

Home Page - Select or create a notebook | Knn image classifier1 - Jupyter | IEEE Xplore Full-Text PDF: | +

localhost:8888/notebooks/Knn%20image%20classifier1.ipynb

jupyter Knn image classifier1 Last Checkpoint: Last Friday at 06:40 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

[ 0, 100, 120, ..., 29, 31, 100]], dtype=uint8)

```
In [44]: test_output_names = []
for i in test_output:
    if i==0:
        test_output_names.append('Autistic')
    else:
        test_output_names.append('Non_Autistic')
```

```
In [45]: print(classification_report(imagenametest, test_output_names))
```

	precision	recall	f1-score	support
Autistic	0.74	0.71	0.72	150
Non_Autistic	0.72	0.75	0.73	150
accuracy			0.73	300
macro avg	0.73	0.73	0.73	300
weighted avg	0.73	0.73	0.73	300

```
In [46]: confusion_matrix(imagenametest, test_output_names)
```

```
Out[46]: array([[106, 44],
               [ 38, 112]], dtype=int64)
```

In [ ]:

In [ ]:

In [ ]:

36°C Partly sunny

ENG IN 11:25 02-06-2022

## **CHAPTER 6**

### **REFERENCES**

- [1] Moneeb ahmed “Pliable Algorithm RGB image Convert in Gray image Using Transformation Equation” International journal of computer science and software engineering(IJCSSE) on 12 december 2019.
- [2] Osman alay and Mustafa ulas Prediction of the Autism Spectrum Disorder Diagnosis with Linear Discriminant Analysis Classifier and K-Nearest Neighbor in Children in IEEE XPLORE on 1<sup>st</sup> September 2020
- [3] Zhong Zhaol , Xiabin Zhang Applying Machine Learning to Identify Autism With Restricted Kinematic Features in IEEE EXPLORE on 24<sup>th</sup> September 2021.