- # Grayscale:

- Model 1:

1 – Load data from github link using !git clone

2- Store the data in my notebook and divide it randomly into two parts, Train and test, at a ratio of 80:20.

3- use openCV library in python to read the images in both train and test path, and use parameter IMG = CV2.IMREAD(FULL_FILE_PATH, CV2.IMREAD_GRAYSCALE) in library to Specifies that images be read in grayscale mode

4- Create the labels from name of folders [0,1,2,3,4,……,9]

5- Convert the dictionary that return from openCV into array of arrays to be ready to get into the model

6- Preprocess data (train & test)

7- Create evolution function that calculates different measurements (confussion matrix & Recall & Precision & f1-score).

8- Build the model and assume k-fold =5 for cross validation

Architecture of model is:

4 conv2D

1 Dropout (0.1)

1 MaxPooling

1 AvgPooling

1 BatchNormalization

1 flatten

4 Dense Layers

9- Calculate the average score and average loss for all k-folds

10- Save model as .h5 file

11- Call evaluation function

12- Test the model by calculate y_predict and compare it with y_test

13- Plot 16 images with (y_predict VS y_actual)

- Results:

  Accuracy on Testing data: 96.82%
  F-score: 97%
  Recall: 97%
  Precision: 97%

- Model 2:

- Same steps as Model 1

 # Architecture of model is:

5 conv2D

2 Dropout(0.1)

2 MaxPooling

1 BatchNormalization

1 flatten

5 Dense Layers

➢ Results:

Accuracy on Testing data: 99.51%
F-score: 100%
Recall: 100%
Precision: 100%

- # RGB:

- CNN Model:

1 – Load data from github link using !git clone

2- Store the data in my notebook and divide it randomly into two parts, Train and test, at a ratio of 80:20.

3- **To Avoid Underfitting:** Apply Data Augmentation: The image augmentation technique is a great way to expand the size of your dataset. You can come up with new transformed images from your original dataset.

Use Imagedatagenerator object from keras library to apply augmentation and use this parameters:

```
ImageDataGenerator
(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

And create 3 new images from each image in my training data

4- use openCV library in python to read the images in both train and test path, and use parameter `IMG = CV2.IMREAD(FULL_FILE_PATH, CV2.IMREAD_BGR2RGB)` **In the library to Specifies that images be read in RGB mode**

5- Create the labels from name of folders [0,1,2,3,4,……,9]

6- Convert the dictionary that return from openCV into array of arrays to be ready to get into the model

7- Preprocess data (train & test)

8- Create evolution function that calculates different measurements (confussion matrix & Recall & Precision & f1-score).

9- Build the model and assume k-fold =3 for cross validation

- Results:
  - Accuracy on Testing data: 96.33%
  - F-score: 96%
  - Recall: 96%
  - Precision: 96%

- Logistic Regression Model:

1- Use x_train , y_train , x_test, y_test from previous step

2- Flattened X_train & X_test to fit the model

3- Reshape y_train & y_test

4- Use cross_val_score(logisticRegr, flattened_training, y_train, cv=5) function to apply Cross Validation and make k (cv)= 5

5- Call prediction method on (test_flatten) to get y_pred

6- Print confusion matrix

- Results:

Accuracy : 0.3422982885085575

Accuracy from k-fold = 5: [0.23339541, 0.21166977, 0.21042831, 0.22470515, 0.23229814]

Confusion matrix:

```
[[26  1  0  0  0  1  5  4  3  1]
 [ 0 21  6  4  0  0  0  2  6  2]
 [ 4  4  8  8  3  0  3  9  1  1]
 [ 0  4  0 18  1  4  1  2  7  4]
 [ 0  3  2  3 10  6  3  7  6  1]
 [ 1  0  1  5  1 14  2  4  6  7]
 [ 2  0  3  4  8  2 11  4  5  2]
 [ 1  2  5  2  8  1  5  9  3  5]
 [ 2  2  2  2  5  2  4  5 10  7]
 [ 0  5  1  2  2  6  1  4  6 13]]
```