* **Problem Statement** **:**

Given an image, detect objects in the frame & predict their category class.

(Using Image Processing Method ) and also use Some Libraries.

* **Objective** **:**
  + 1. Defining the classes of images from the data set .
    2. Create a frame and determine the object name from the class .
    3. Performance analysis to predict the objects using accuracy and error metrics .

# **Classifying Images in Python:**

From choosing profile photos on Facebook and Instagram to categorising

clothing images in shopping apps like Myntra, Amazon, and Flipkart,

image classification occurs everywhere

on social media. Any e-commerce platform's classification has become an

essential component.

Image classification in machine learning is an example of lassifying clothing

photos, which implies classifying the images into their proper category

classes. We'll utilise the fashion mnist dataset that comes with TensorFlow

to get clothes images. Clothing photos from ten different categories are

included in this dataset. It's a replacement for the MNIST dataset for

beginners, which is made up of handwritten digits. As time goes on, we'll

learn more about it.

* **Solution :**

1. **Image read:-**

Image Processing (IP) is a type of computer technology that allows us to

process, analyse, and extract information from images.

When it comes to image processing, Python has a lot of strong features. Let's look at how to process photos using several libraries such as ImageIO, OpenCV, Matplotlib, PIL, and others.

**Example:- (Sample)**

1. Using OpenCV: OpenCV (Open Source Computer Vision) is a computer vision library that includes a number of methods for manipulating images and videos. It was created by Intel and later maintained by Willow Garage . This library is cross-platform, meaning it may be used with a variety of programming languages, including Python, C++, and others.

# Python program to read image using OpenCV

# importing OpenCV(cv2) module

**import** cv2

# Save image in set directory

# Read RGB image

# R=Red,B=Blue,G=Green

img **=** cv2.imread('myimage.png')

# Output img with window name as 'image'

cv2.imshow('image', img)

# Maintain output window utill

# user presses a key

cv2.waitKey(0)

# Destroying present windows on screen

cv2.destroyAllWindows()

1. **Feature extraction:**

#Here we extract the requires libraries based on our project

# importing the necessary libraries

**import** tensorflow as tf

**import** numpy as np

**import** matplotlib.pyplot as plt

### // ****Loading and exploring the data****

### 

# we load the fashion\_mnist dataset and examine the training and

testing data shapes.

# storing the dataset path

clothing\_fashion\_mnist = tf.keras.datasets.fashion\_mnist

# loading the dataset from tensorflow

(x\_train, y\_train),

(x\_test, y\_test) = clothing\_fashion\_mnist.load\_data()

# displaying the shapes of training and testing dataset

print('Shape of training cloth images: ',x\_train.shape)

print('Shape of training label: ',y\_train.shape)

print('Shape of test cloth images: ',x\_test.shape)

print('Shape of test labels: ',y\_test.shape)

# We store the real class names in a variable to use them later

for data visualisation because the class names are not added to

the fashion mnist dataset.

# storing the class names as it is

# not provided in the dataset

label\_class\_names = ['T-shirt/top', 'Trouser',

'Pullover', 'Dress', 'Coat',

'Sandal', 'Shirt', 'Sneaker',

'Bag', 'Ankle boot']

# display the first images

plt.imshow(x\_train[0])

plt.colorbar() # to display the colourbar

plt.show()

### //  ****Preprocessing the data****

### 

### #Here we preprocessing the data

### 

### # The data is in pixel values and its range is 0 to 255.

### #To scale the value between 0 and 1,

x\_train = x\_train / 255.0 # normalizing the training data

### 

### # we must divide each by 255.

x\_test = x\_test / 255.0 # normalizing the testing data

### 

### // ****Data Visualization`****

### 

### # We plotted x train with colormap as binary

### # inserted the class names from the label class

### names array we had previously saved.

plt.figure(figsize=(15, 5))  # figure size

i = 0

while i < 20:

     plt.subplot(2, 10, i+1)

     # showing each image with colourmap as binary

     plt.imshow(x\_train[i], cmap=plt.cm.binary)

     # giving class labels

     plt.xlabel(label\_class\_names[y\_train[i]])

     i = i+1

plt.show()  # plotting the final output figure

**3-ML model:-**

### # Building the model

### #Flatten() takes a two-dimensional array of pictures

### #Turns them to a one-dimensional array, which is then

### passed to tf.keras.layers.

model **=** tf.keras.Sequential([

     tf.keras.layers.Flatten(input\_shape**=**(28, 28)),

     tf.keras.layers.Dense(128, activation**=**'relu'),

     tf.keras.layers.Dense(10)

])

# compiling the model

#**SparseCategoricalCrossentropy as the loss function**

cloth\_model.compile(optimizer**=**'adam',

                    loss**=**tf.keras.losses.SparseCategoricalCrossentropy(

                        from\_logits**=**True),

                    metrics**=**['accuracy'])

**4-Train test split for model building :-**

#we will feed the **x\_train, y\_train**

**#**The **model.fit()** method helps in fitting the training data into our

model.

# Fitting the model to the training data

cloth\_model.fit(x\_train, y\_train, epochs**=**10)

# calculating loss and accuracy score

# we can see that the accuracy score on the testing data is less than that of

training data

test\_loss, test\_acc **=** cloth\_model.evaluate(x\_test,

                                           y\_test,

                                           verbose**=**2)

print('\nTest loss:', test\_loss)

print('\nTest accuracy:', test\_acc)

**5- Performance analysis :-**

### 

### //  ****Making predictions on trained model with test data****

# We used predictions[0] to try to forecast the first test image, x test[0],

# using Softmax() function to convert

# linear output logits to probability

prediction\_model **=** tf.keras.Sequential(

     [cloth\_model, tf.keras.layers.Softmax()])

# feeding the testing data to the probability

# prediction model

prediction **=** prediction\_model.predict(x\_test)

# predicted class label

**print**('Predicted test label:', np.argmax(prediction[0]))

# predicted class label name

**print**(label\_class\_names[np.argmax(prediction[0])])

# actual class label

print('Actual test label:', y\_test[0])

### // ****Data Visualization of predicted vs actual test labels****

### **# Finally, we'll compare the projected vs. real class labels for our Given**

### **Image .**

### **# It determine how accurate our model is.**

# assigning the figure size

plt.figure(figsize**=**(15, 6))

i **=** 0

# plotting total 24 images by iterating through it

**while** i < 24:

      image, actual\_label **=** x\_test[i], y\_test[i]

     predicted\_label **=** np.argmax(prediction[i])

     plt.subplot(3, 8, i**+**1)

     plt.tight\_layout()

     plt.xticks([])

     plt.yticks([])

     # display plot

     plt.imshow(image)

     # if else condition to distinguish right and

     # wrong

     color, label **=** ('green', 'Correct Prediction')

**if** predicted\_label **==** actual\_label **else** (

         'red', 'Incorrect Prediction')

     # plotting labels and giving color to it

     # according to its correctness

    plt.title(label, color**=**color)

     # labelling the images in x-axis to see

     # the correct and incorrect results

     plt.xlabel(" {} ~ {} ".format(

         label\_class\_names[actual\_label],

       label\_class\_names[predicted\_label]))

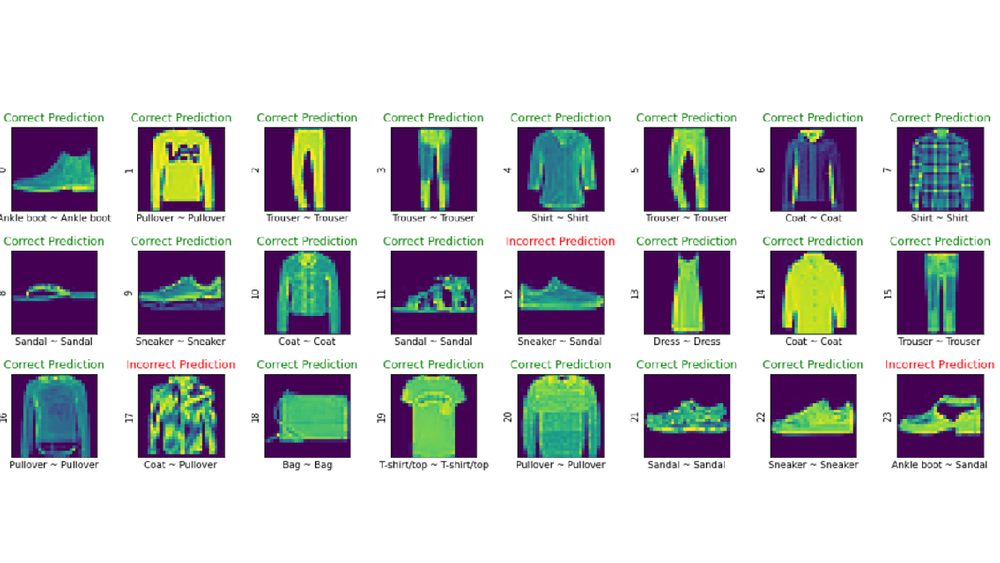
     # labelling the images orderwise in y-axis

     plt.ylabel(i)

     # incrementing counter variable

     i **+=** 1

* **OUTPUT:-**



#we get the good accuracy in this prediction

The Given Image category is “Fashion Shirt Women” and As can be seen, the 12th, 17th, and 23rd predictions are incorrectly categorised, but the remainder of the predictions are right. We designed a good model because no classification model can be 100 percent accurate in reality.So, Performance analysis to predict the objects using accuracy and error metrics is Successfully Completed.