

ABV-Indian Institute of  
Information Technology  
and Management Gwalior

**Bio-Medical Engineering**

# Emotion Recognition System

Submitted To-

Prof. Mahua Bhattacharya



## **Submitted By -**

---

2018IMG001 Aayush Singh

2018IMG008 Amit Singh

2018IMG011 Ankit Kumar

2018IMG015 Ayush Katiyar

2018IMG024 Divyansh Kumar

2018IMG028 Kamal Garg

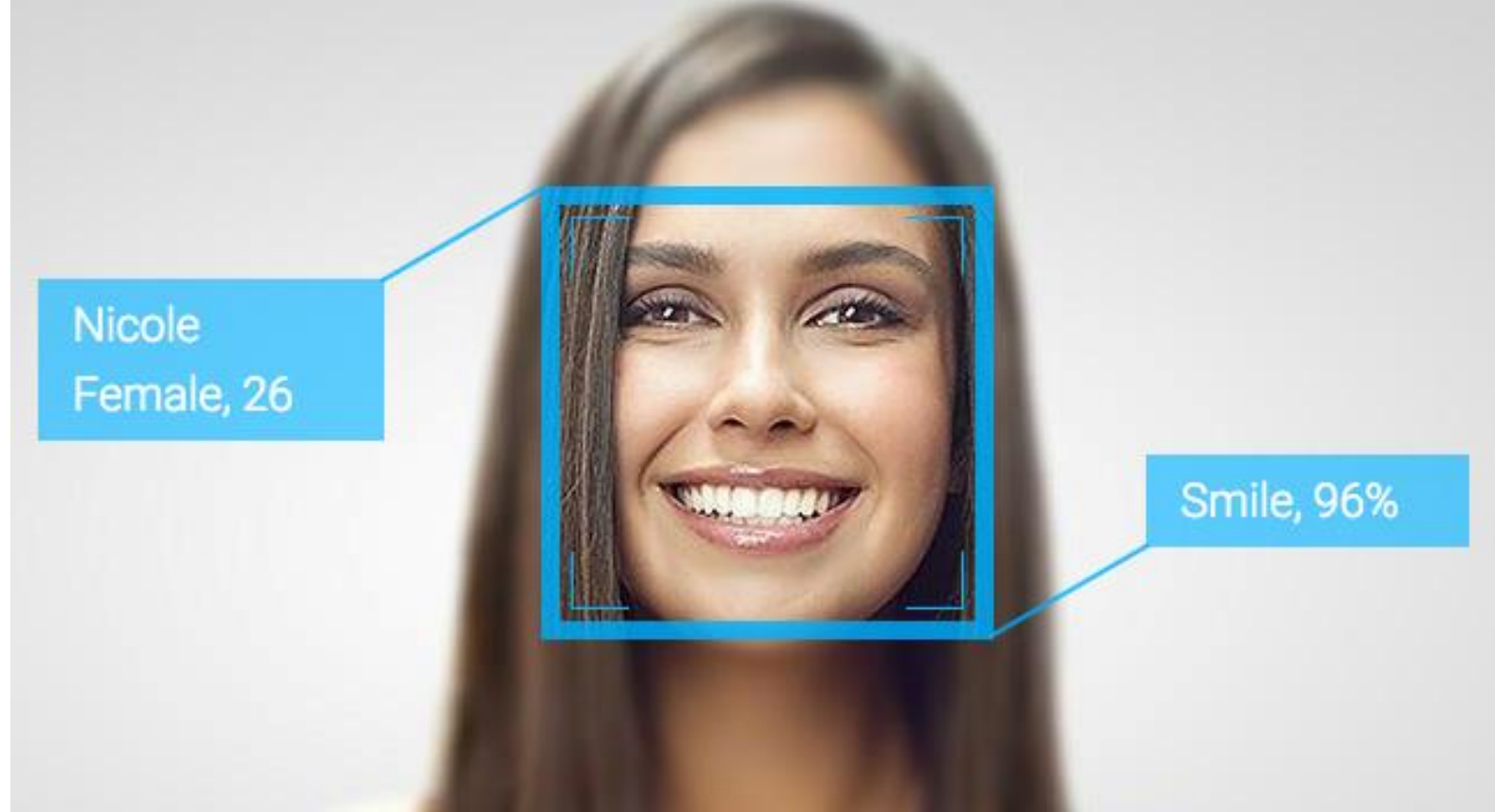
2018IMG029 Manish Dangi

2018IMG035 Prashant Dwivedi

2018IMG060 Shivam Bhasker

2018IMT108 Vishnu Kumar

Now-a-days there is a common trend for a human-computers interaction in the field of machine intelligence. Real time detection of face and interpreting different facial expressions like happy, anger, sad, fear, surprise etc. is based on facial features and their actions. The key elements of face are considered for detection of face and prediction of expressions or emotions of face. To determine the different facial expressions, the variations in each facial features are used. For detection and classification of different classes of facial expressions, machine learning algorithms are used by training of different set of images. The proposed algorithm uses open source computer vision (OpenCV) and Machine learning with python.



## Introduction

**Emotion recognition** is the process of identifying human **emotion**, most typically from facial expressions as well as from verbal expressions.



# Dilbag Singh

*Prof. at Manipal University Jaipur*

This paper discusses the application of feature extraction of facial expressions with combination of neural network for the recognition of different facial emotions (happy, sad, angry, fear, surprised, neutral etc..). Humans are capable of producing thousands of facial actions during communication that vary in complexity, intensity, and meaning. This paper analyses the limitations with existing system Emotion recognition using brain activity.

## Department of Computer Science & Engineering, Beijing University of Aeronautics & Astronautics

*Faizan Ahmad, Aaima Najam and Zeeshan Ahmed*

This paper focuses on implementing face detection technologies in public places and takes accuracy and speed of identification as the main issue.

The goal of this paper is to evaluate various face detection and recognition methods, provide complete solution for image based face detection and recognition with higher accuracy, better response rate as an initial step for video surveillance.



## Research Review

Researching about previous developments is a vital part of any project to learn from previous mistakes and improve the quality of work



# Research Gaps

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas porttitor congue massa



Surveillance  
Angle

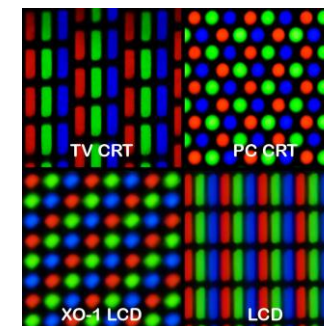


Image Size &  
Quality



Processing &  
Storing

# Design Details



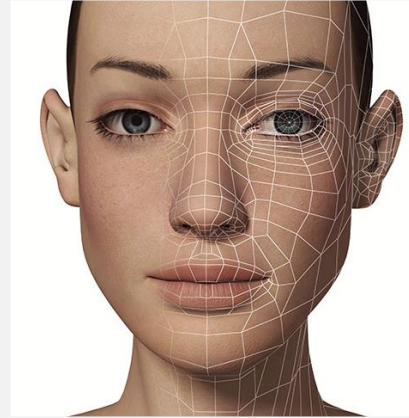
## Image Capture

In the first step we have taken the input image using webcam



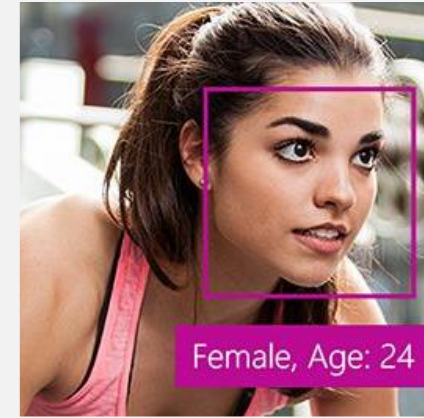
## Face Detection

Detect the face using OpenCV in python



## Feature Extraction

Try to get the features from the obtained face image using CNN concepts



## Classification

The extracted features are given to the classifier like Logistic Regression, SVM etc.



## Output

Classifier predicts the recognized expression as output



# Implementation

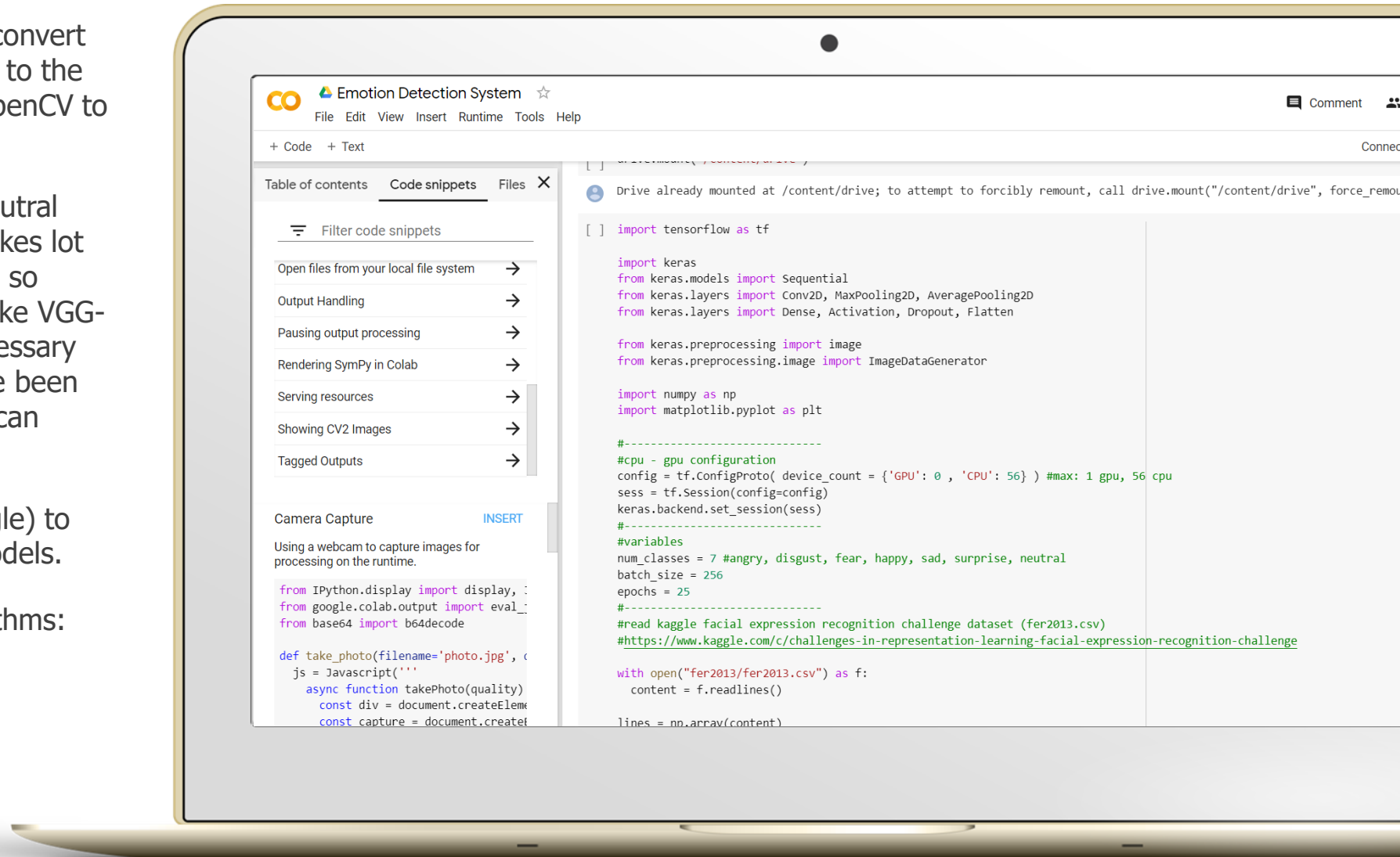
We need to find the face on each image, convert it to grayscale, crop it and save the image to the dataset. We can use a HAAR filter from OpenCV to automate face finding.

For a normal computer, training a large neural network with millions of images or data takes lot of computing power and is very expensive so better idea is to use a pre-trained model like VGG-16 without having to master the skills necessary to tune and train those models which have been already trained with lots of images as we can directly use the weights.

We have used online software Colab(Google) to run our python programs and train the models.

For this Model we use the following algorithms:

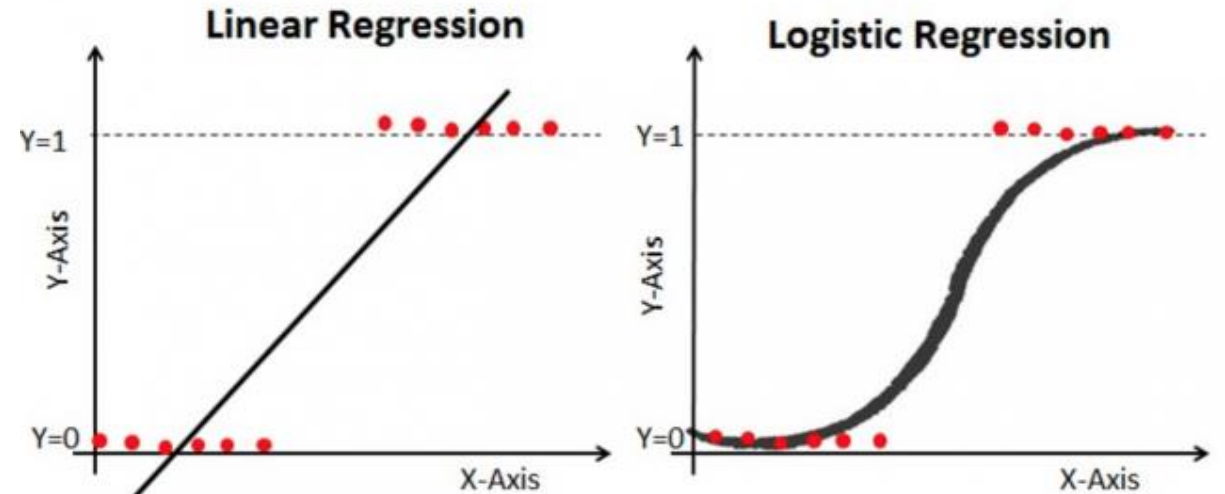
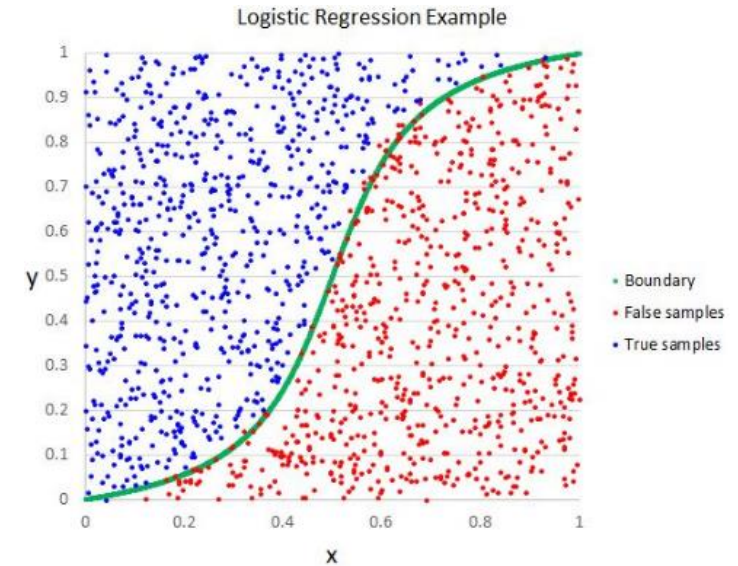
- ☐ Logistic Regression
- ☐ KNN
- ☐ Support Vector Machines (SVM)



# Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a logistic model (a form of binary regression).

Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

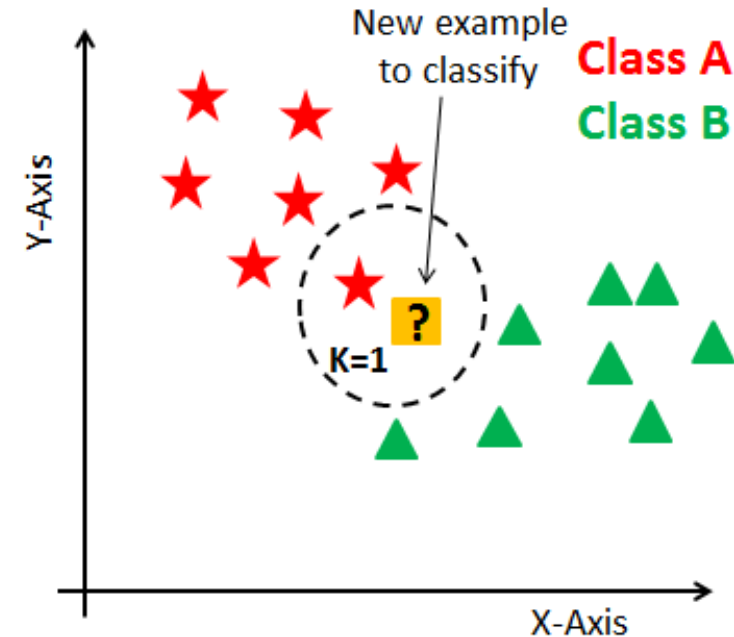




# ***k*-nearest neighbors algorithm**

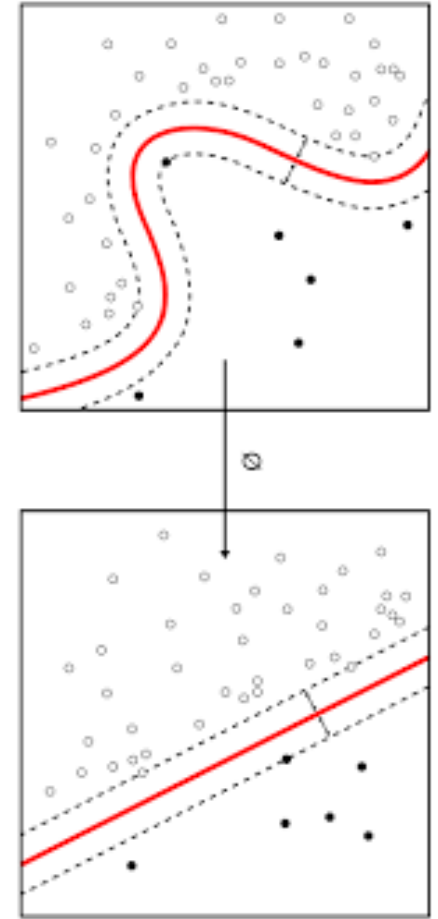
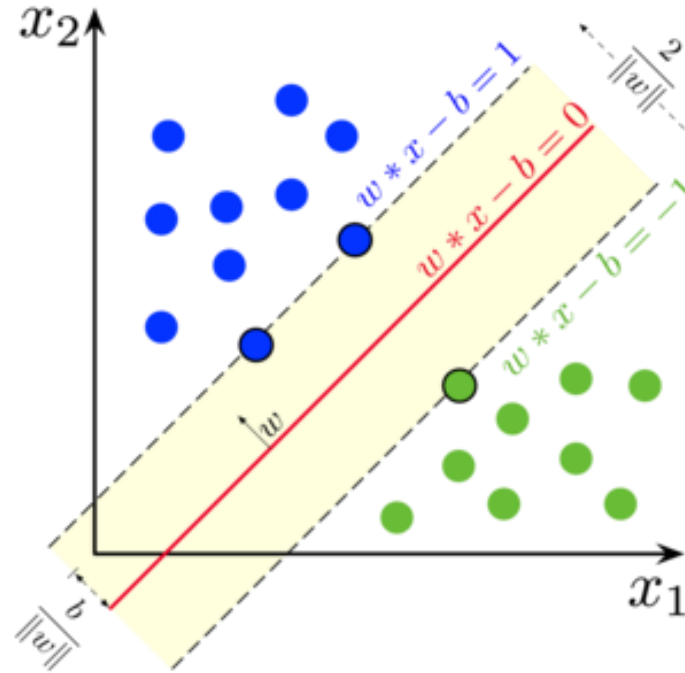
In pattern recognition, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric method used for classification and regression. In both cases, the input consists of the  $k$  closest training examples in the feature space. The output depends on whether  $k$ -NN is used for classification or regression:

- In *k*-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its  $k$  nearest neighbors ( $k$  is a positive integer, typically small). If  $k = 1$ , then the object is simply assigned to the class of that single nearest neighbor.
- In *k*-NN regression, the output is the property value for the object. This value is the average of the values of  $k$  nearest neighbors.



# Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.



# Creation Of Project

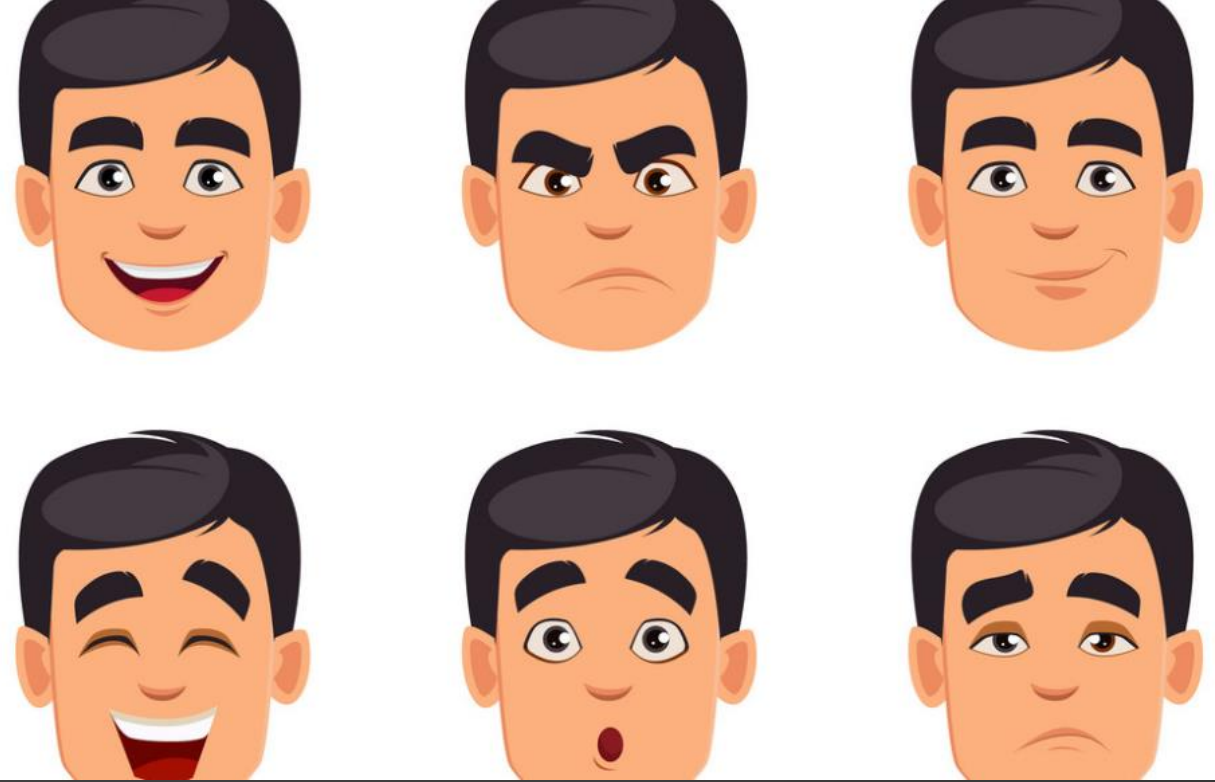
---

The different steps through which we gave shape to this final project...





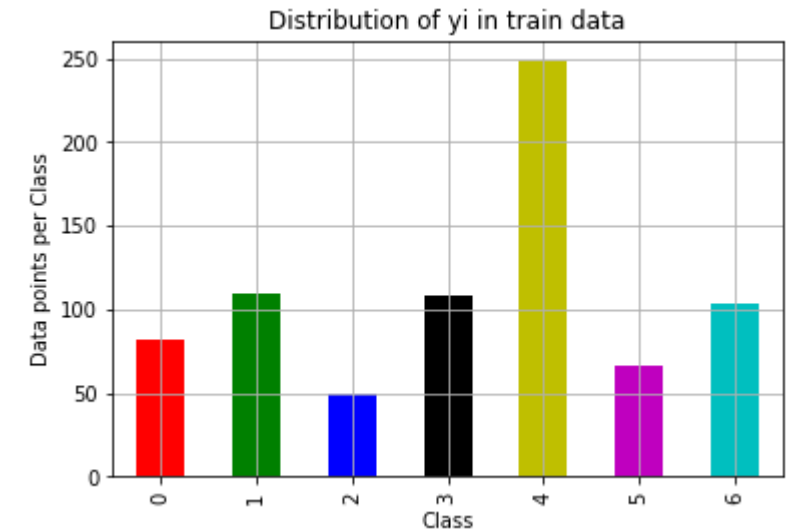
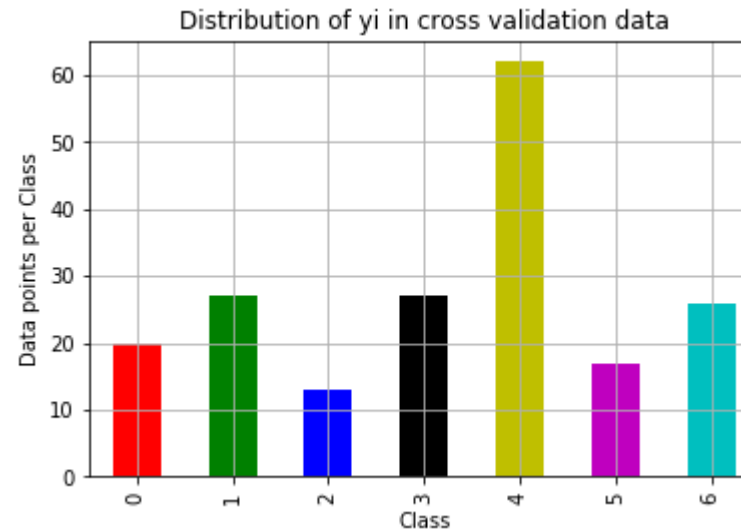
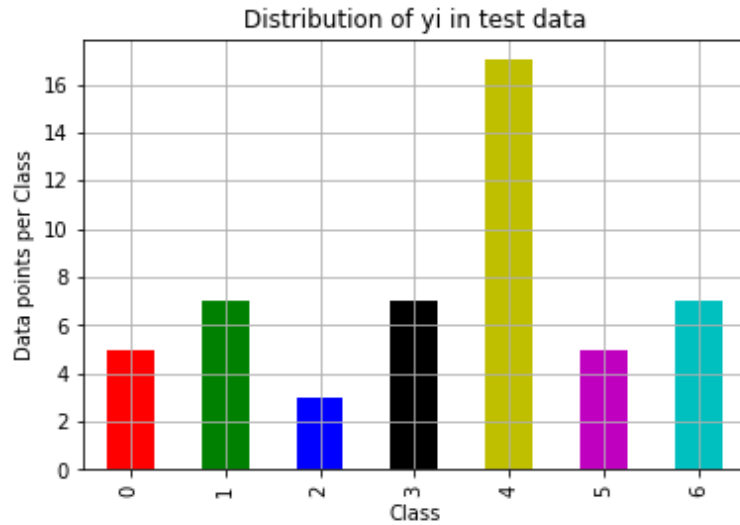
- We use the training set to train(or teach) the classifier to recognize the to-be-predicted classes(expressions) and use the testing set to estimate the classifier performance. **But then what is the use of cross-validation** — It is mostly used in problems where the objective is a prediction, and one wants to calculate how accurately a predictive model will perform in real practice.
- The aim of cross-validation is to define a dataset to “test” the model in the training phase (i.e., the *validation set*), in order to limit problems like **overfitting** and **underfitting**, and give an insight on how the model will generalize to an unknown dataset, for instance from a real problem



## Training and Testing

After getting the features from the images, now the next important task to build a classification model is to split the whole dataset into training, cross-validation and testing sets.

So, we randomly split 64% of our dataset in training data, 16% in cross-validation set and 20% in the testing set using the `train_test_split` method of `sklearn` in python.



Here, it can be easily seen that we encoded different expression(or class labels) as numerical values from 0 to 6 as each number representing different class ("anger": 0, "disgust": 1, "fear": 2, "happy": 3, "neutral": 4, "sad": 5, "surprise": 6 )

# Outcome of Project



/Administration  
/Human Resources  
/Legal  
/Accounting  
/Finance  
/Marketing  
/Publicity  
/Promotion  
/Research  
/Business  
/Development  
/Engineering  
/Manufacturing  
/Planning



# Results

We have three different algorithms and these different algorithms give different results having different levels of accuracy and cross validation log-

## Using Logistic Regression -

```
The train log loss is: 0.768237309949  
The cross validation log loss is: 1.05564794889  
The test log loss is: 0.971378253018
```

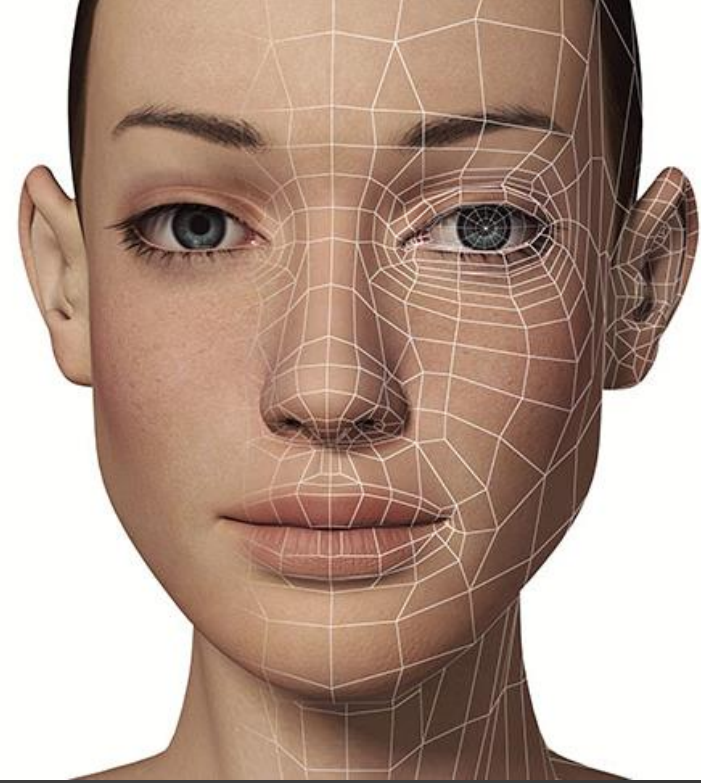
## Using KNN -

```
The train log loss is: 1.13836352362  
The cross validation log loss is: 1.23858168969  
The test log loss is: 1.19100240199
```

## Using SVM -

```
The train log loss is: 0.7118609702  
The cross validation log loss is: 1.05977434469  
The test log loss is: 0.921972157437
```

- Improvement in Image quality and size.
- Development of Better Sensors
- Better Scanning Algorithms
- Faster Training Modules
- Much detailed Datasets
- Implementation of scanning technologies in real world problems.



## Scope of Improvement

Every technology has a scope of future improvement because nothing is perfect and new technologies are coming out everyday...



# **Real Time Application**

Table of contents Code snippets Files

Upload Refresh Mount Drive

- drive
- fer2013
- sample\_data
  - capture.jpg
  - haarcascade\_frontalface\_alt.xml
  - images 5.jpg
  - images.2.jpg
  - images01.jpg
  - images3.jpg
  - images4.jpg
  - model25.h5
  - photo.jpg

Disk 23.51 GB available


```
plt.gray()
plt.imshow(true_image)
plt.show()
```

/usr/local/lib/python3.6/dist-packages/keras\_preprocessing/image/utils.py:104: UserWarning: grayscale is deprecated. Please use color\_mode='rgb' instead. warnings.warn('grayscale is deprecated. Please use ')

emotion



| emotion  | percentage |
|----------|------------|
| angry    | 0.0        |
| disgust  | 0.0        |
| fear     | 0.0        |
| happy    | 1.0        |
| sad      | 0.0        |
| surprise | 0.0        |
| neutral  | 0.0        |





## Welcome To Colaboratory

File Edit View Insert Runtime Tools Help Unsaved changes since 2:51 AM

Share



+ Code + Text Copy to Drive

RAM  
Disk

Editing

Table of contents Code snippets Files

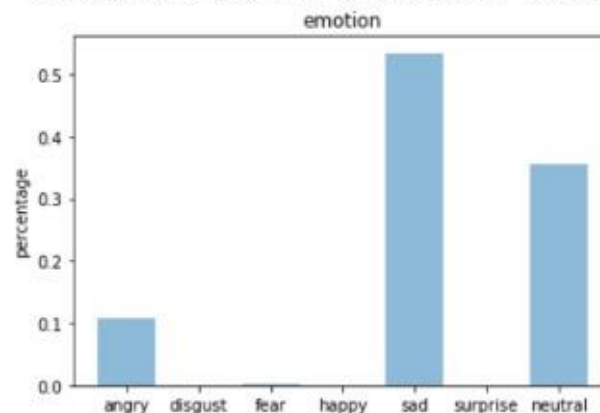
Upload Refresh Mount Drive

- ..
- drive
- fer2013
- sample\_data
  - capture.jpg
  - haarcascade\_frontalface\_alt.xml
  - images 5.jpg
  - images.2.jpg
  - images01.jpg
  - images3.jpg
  - images4.jpg
  - model25.h5
  - photo.jpg

```
x = np.array(x, 'float32')  
x = x.reshape([48, 48]);
```

```
plt.gray()  
plt.imshow(true_image)  
plt.show()
```

/usr/local/lib/python3.6/dist-packages/keras\_preprocessing/image/utils.py:104: UserWarning: grayscale is deprecated. Please use color\_mode='rgb' instead.  
warnings.warn('grayscale is deprecated. Please use ')



Disk 23.51 GB available



Table of contents Code snippets Files X

Upload Refresh Mount Drive

- ..
- drive
- fer2013
- sample\_data
  - capture.jpg
  - haarcascade\_frontalface\_alt.xml
  - images 5.jpg
  - images.2.jpg
  - images01.jpg
  - images3.jpg
  - images4.jpg
  - model25.h5
  - photo.jpg

Disk 23.51 GB available


```
x = x.reshape([48, 48]);  
  
plt.gray()  
plt.imshow(true_image)  
plt.show()
```

/usr/local/lib/python3.6/dist-packages/keras\_preprocessing/image/utils.py:104: UserWarning: grayscale is deprecated. Please use color\_mode='rgb' instead.

emotion



| emotion  | percentage |
|----------|------------|
| angry    | 0.7        |
| disgust  | 0.0        |
| fear     | 0.0        |
| happy    | 0.0        |
| sad      | 0.0        |
| surprise | 0.0        |
| neutral  | 0.3        |





## Welcome To Colaboratory

File Edit View Insert Runtime Tools Help Unsaved changes since 2:51 AM

Share

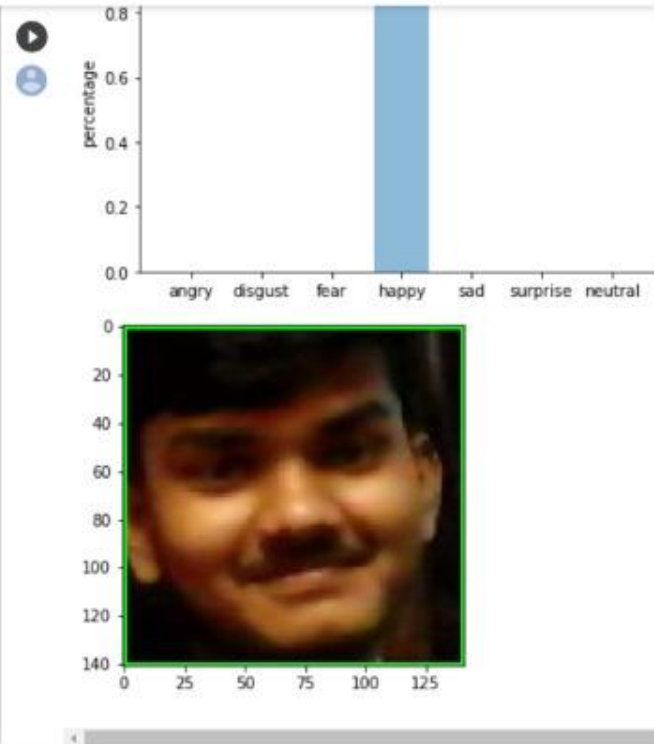
+ Code + Text Copy to Drive

✓ RAM Disk Editing

Table of contents Code snippets **Files** X

Upload Refresh Mount Drive

- ..
- drive
- fer2013
- sample\_data
  - capture.jpg
  - haarcascade\_frontalface\_alt.xml
  - images 5.jpg
  - images.2.jpg
  - images01.jpg
  - images3.jpg
  - images4.jpg
  - model25.h5
  - photo.jpg



```
[131] import cv2

def facecrop(image):
    facedata = "haarcascade_frontalface_alt.xml"
    cascade = cv2.CascadeClassifier(facedata)
```

Disk 23.51 GB available



## Welcome To Colaboratory

File Edit View Insert Runtime Tools Help Unsaved changes since 7:28 PM

Share



+ Code + Text Copy to Drive

RAM  
Disk

Editing

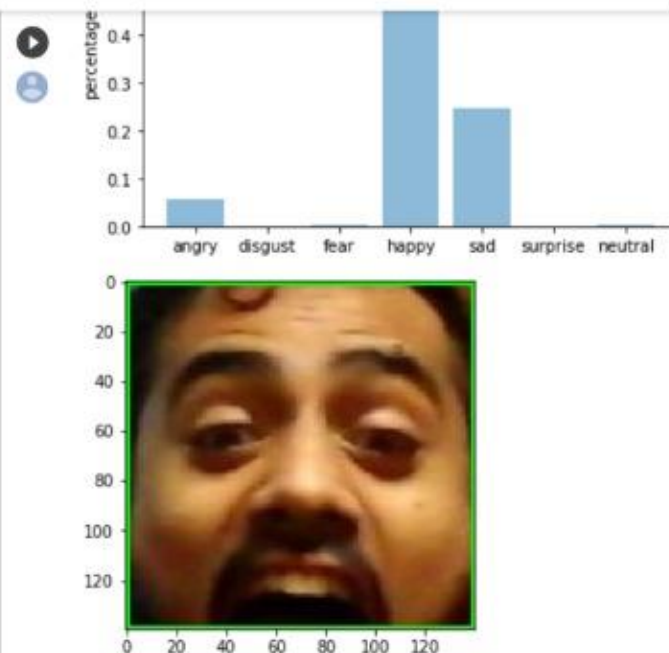
Table of contents

Code snippets

Files

Upload Refresh Mount Drive

- drive
- fer2013
- sample\_data
  - capture.jpg
  - haarcascade\_frontalface\_alt.xml
  - images 5.jpg
  - images.2.jpg
  - images01.jpg
  - images3.jpg
  - images4.jpg
  - model25.h5
  - photo.jpg



```
[113] import cv2

def facecrop(image):
    facedata = "haarcascade_frontalface_alt.xml"
    cascade = cv2.CascadeClassifier(facedata)

    img = cv2.imread(image)
```

Disk 23.51 GB available

# References

- <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>
- <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
- <https://medium.com/@hinasharma19se/facial-expressions-recognition-b022318d842a>



**Thank You**