# Facial emotion Recognition

Team Members
Sowmya P (u4cse17314)
Sindhuja D (u4cse17503
Niranjan V (u4cse17505

# Background

Facial expression plays an important role in indicating feelings of a person. People can immediately identify emotional state of a person by identifying his/her facial expression.  As a consequence many automatic systems use information regarding facial expressions to identify emotion. The aim of this project is to recognize seven basic emotional states like neutral, joy, surprise, anger, sadness, fear and disgust based on facial expressions.

# Problem Statement

Automatic recognition of facial expressions is an important component in human machine interface. It needs to perform detection, facial feature extraction and then classification. Our main task is to classify a given set of facial image into one of the classes (Angry, disgust, fear, happy, sad, surprise, and natural) that best depict the emotion of the facial expression in the image.

It is a challenging task because emotions may vary depending on face recognition and appearance leading to ambiguous data.

# Datasets

- Facial Emotion Recognition 2013(FER-2013)
- Extended Cohn-Kanade Dataset (CK+)
- Karolinska Directed Emotional Faces (KDEF)

# FER-2013

- FER-2013 is a dataset from Kaggle.
- The training set consists of 28709 examples and public test set consists of 3589 examples which were categorized into 7 classes namely happy, neutral, sad, fear, angry, surprise, disgust .
- Link to dataset: https://www.kaggle.com/msambare/fer2013
- I have used only the validation part of the whole dataset which contains 3589 images.
- The seven expressions in the dataset were labelled as
  0 - neutral 1 - angry 2 - disgust 3 - fear 4 - happy
  5 - sad 6 - surprise

# CK+

- Link: https://www.kaggle.com/ankur133047/modified-ck-facial-expression-dataset
- Ck+ or Extended Cohn Kanade Dataset is widely used for facial expression recognition
- It contains in total of 593 images
- Seven types of emotions can be classified, with the help of this dataset.
- Emotions:
  - 0 - neutral 1 - angry 2 - disgust 3 - fear 4 - happy
  - 5 - sad 6 - surprise
- There wasn't equal distribution of images in this dataset. So did random sampling
- People in the pictures don't have beards and moustaches.
- Characters in the pictures are around 20-30 years old.

# Kdef

- Link: https://www.kaggle.com/ankur133047/modified-kdef-facial-expression-dataset
- Contains 490 images of 70 individual displaying 7 different emotional states.
- 70 individuals contain 35 Males and 35 Females and age ranging from 20 and 30 years of age.
- Pictures does not contain any beards or mustaches.
- There are seven expressions in the dataset labelled as:
  0 - neutral 1 - angry 2 - disgust 3 - fear 4 - happy
  5 - sad 6 - surprise
- Each emotion contains 50-60 facial images.

# Preprocessing

**Dlib:**
It's a landmark's facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person's face like image below.

**LBP:**

**Local Binary Pattern** (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

# Algorithm Implementation And Results

# Support Vector Machine

- Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression problems
- Works well even with high dimensional, unstructured data, images, text.
- For our problem, we divided the dataset into 3 different parts - Training, validation and test.
- There are 3 types of kernels (linear, poly and rbf) and we used linear kernel. (Poly and Linear were giving almost same accuracy, where as rbf was giving little less accuracy). We used Sklearn for implementation.

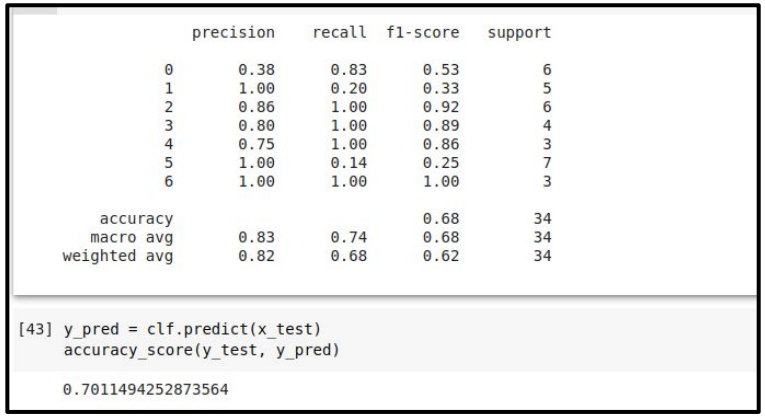
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.83 | 0.83 | 6 |
| 1 | 0.83 | 1.00 | 0.91 | 5 |
| 2 | 1.00 | 0.83 | 0.91 | 6 |
| 3 | 1.00 | 1.00 | 1.00 | 4 |
| 4 | 1.00 | 1.00 | 1.00 | 3 |
| 5 | 1.00 | 1.00 | 1.00 | 7 |
| 6 | 1.00 | 1.00 | 1.00 | 3 |
| accuracy | | | 0.94 | 34 |
| macro avg | 0.95 | 0.95 | 0.95 | 34 |
| weighted avg | 0.95 | 0.94 | 0.94 | 34 |

```
[ ] y_pred = clf.predict(x_test)
    accuracy_score(y_test, y_pred)

    0.9310344827586207
```

# Random Forest

- Random forest is supervised machine learning algorithm
- The forest it builds is an ensemble of decision trees, usually trained with bagging method.
- The general idea of the bagging method is that a combination of learning models increases the overall result.
- It can be used for both classification and regression tasks.
- This algorithm doesn’t use hyperparameters but produces a good results.
- We used random forest classifier which was imported from sklearn library.



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.38 | 0.83 | 0.53 | 6 |
| 1 | 1.00 | 0.20 | 0.33 | 5 |
| 2 | 0.86 | 1.00 | 0.92 | 6 |
| 3 | 0.80 | 1.00 | 0.89 | 4 |
| 4 | 0.75 | 1.00 | 0.86 | 3 |
| 5 | 1.00 | 0.14 | 0.25 | 7 |
| 6 | 1.00 | 1.00 | 1.00 | 3 |
| accuracy | | | 0.68 | 34 |
| macro avg | 0.83 | 0.74 | 0.68 | 34 |
| weighted avg | 0.82 | 0.68 | 0.62 | 34 |

```
[43] y_pred = clf.predict(x_test)
     accuracy_score(y_test, y_pred)

     0.7011494252873564
```

# K Nearest Neighbours

- K Nearest Neighbours is a supervised machine learning algorithm which classifies new data based on the similarity with old data.
- In our implementation we used K value 10, which implies that whenever model sees a new data point it is labelled belonging to a particular class which gets maximum votes by K (10 in this case) of its neighbours.
- Here, we have implemented KNN from scratch and used euclidean distance to compute neighbours.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.60 | 0.50 | 0.55 | 6 |
| 1 | 0.40 | 0.40 | 0.40 | 5 |
| 2 | 0.80 | 0.67 | 0.73 | 6 |
| 3 | 1.00 | 1.00 | 1.00 | 4 |
| 4 | 1.00 | 1.00 | 1.00 | 3 |
| 5 | 0.78 | 1.00 | 0.88 | 7 |
| 6 | 1.00 | 1.00 | 1.00 | 3 |
| accuracy | | | 0.76 | 34 |
| macro avg | 0.80 | 0.80 | 0.79 | 34 |
| weighted avg | 0.76 | 0.76 | 0.76 | 34 |

```
[40] y_pred = clf.predict(x_test)
     accuracy_score(y_test, y_pred)

     0.8735632183908046
```

# Mini Batch K Means Clustering

- Mini batch K means is a variant of K means clustering where data is split into mini baches and k means is applied.
- Each iteration a new random sample from dataset is obtained and used to update clusters until  convergence.
- In our implementation we used K =  7 i.e we used 7 clusters.
- Here, each new data point belongs to the cluster with nearest mean

# Results

- Preprocessing using Dlib library that helps in recognising facial landmarks gave better results when compared to Local Binary Pattern (LBP)

- Among the different algorithms used (i.e K Nearest Neighbours, Support Vector Machine, Random forest, K means clustering) best result was obtained in Support Vector Machine followed by K Nearest Neighbours, Random Forest and K Means Clustering respectively.

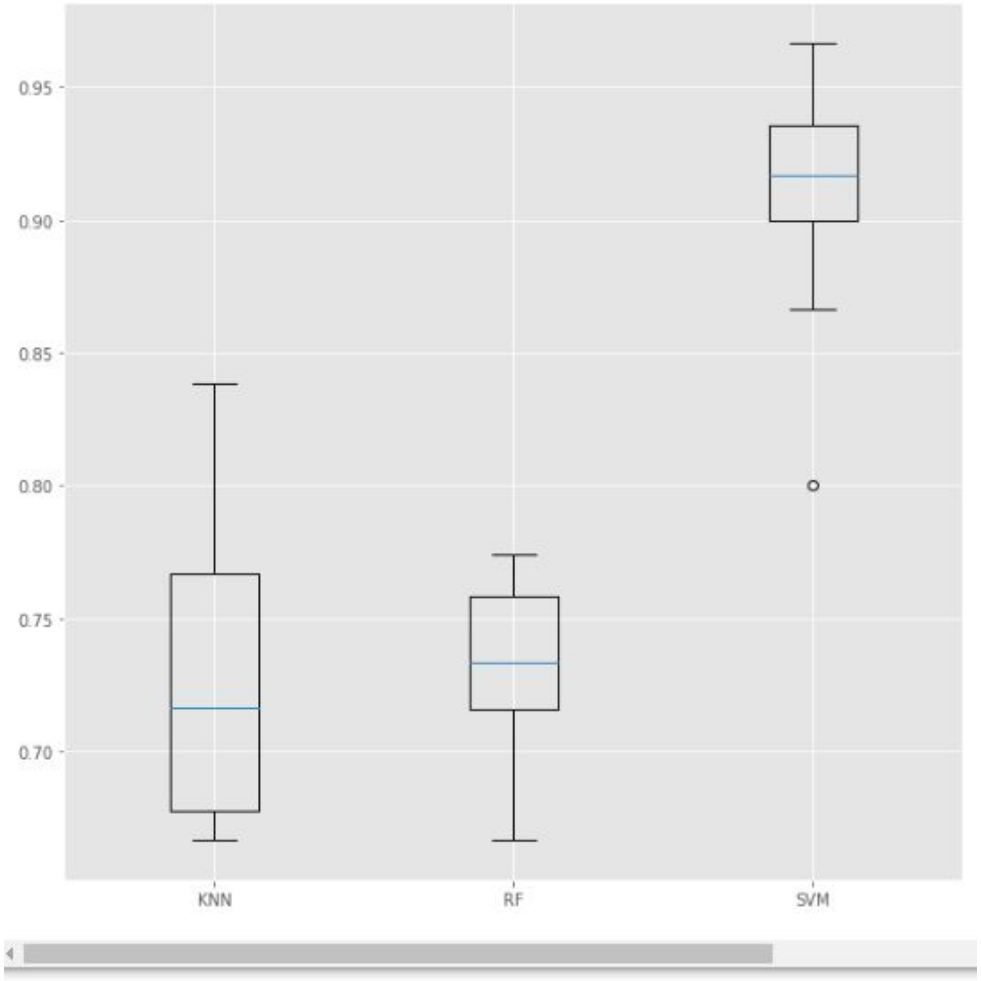
- We implemented K Nearest Neighbours from scratch. Accuracy is similar to that of the inbuilt K nearest neighbours function from Sklearn.

# Results

- **FER**
  KNN accuracy: 71.04
  SVM accuracy: 71.15
  RF accuracy: 63.23

- **Kdef**
  Knn accuracy :  72.30
  SVM accuracy :  76.53
  Rf accuracy :  71.38

- **CK+**
  KNN accuracy: 87.35
  SVM accuracy: 93.10
  RF accuracy: 70.11


0.95
0.90
0.85
0.80
0.75
0.70
KNN
RF
SVM

# Thank You