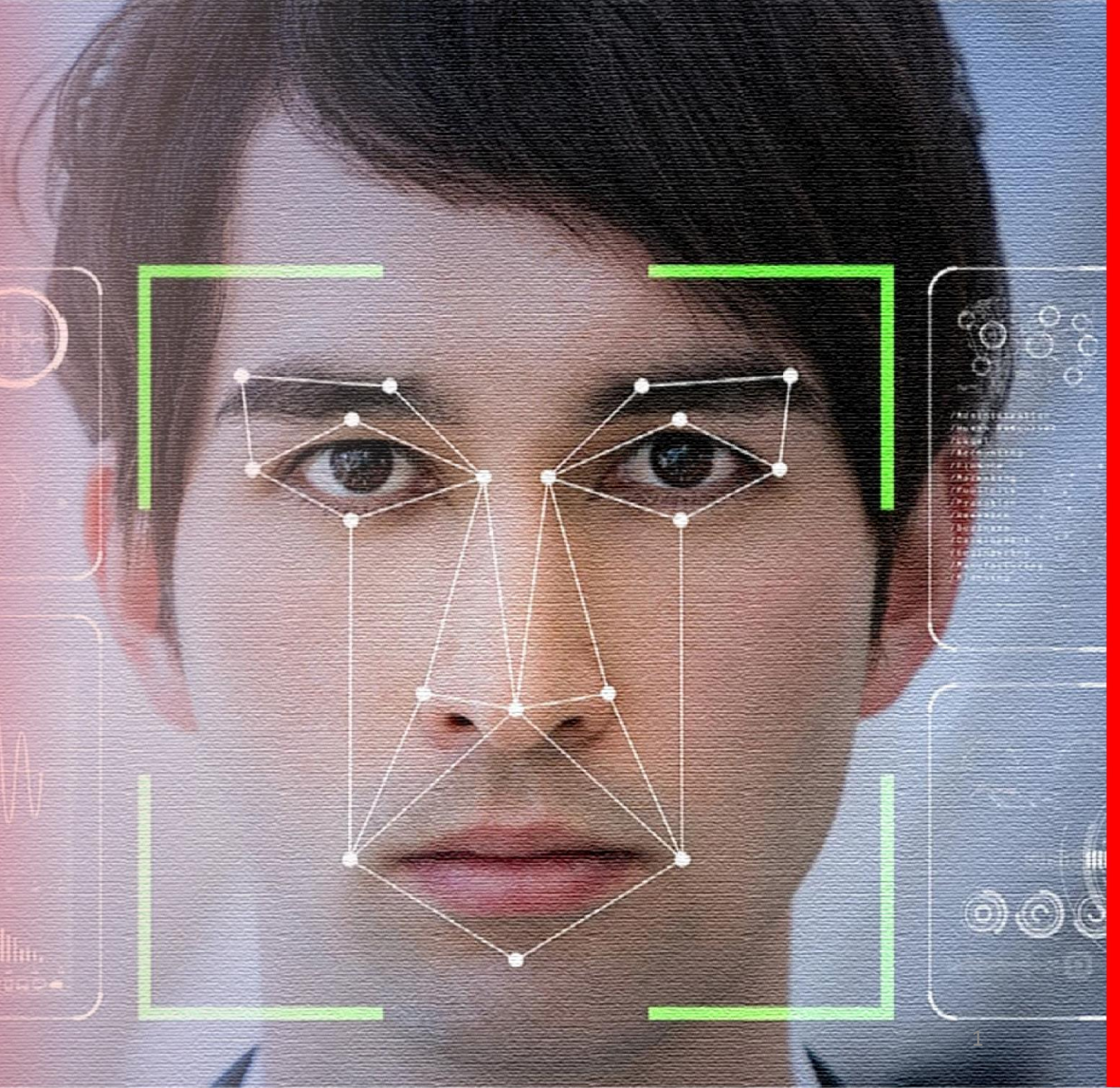
Facial Expression Recognition



Akash Sharma
10/01/2022

Outline

EXECUTIVE SUMMARY

INTRODUCTION

METHODOLOGY

RESULTS

CONCLUSION

APPENDIX

Executive Summary

- Loading Data
- Summary of methodologies
  - Data preprocessing
  - EDA with Data visualization
  - Predictive analysis
- Summary of all results
  - Data visualization result
  - Model result

Introduction

- Project background and context

  The face recognition procedure simply requires any device that has digital photographic technology to generate and obtain the images and data necessary to communicate their intentions and emotions. Facial expression can be classified into anger, disgust, fear, happy, sad, surprise and neutral.

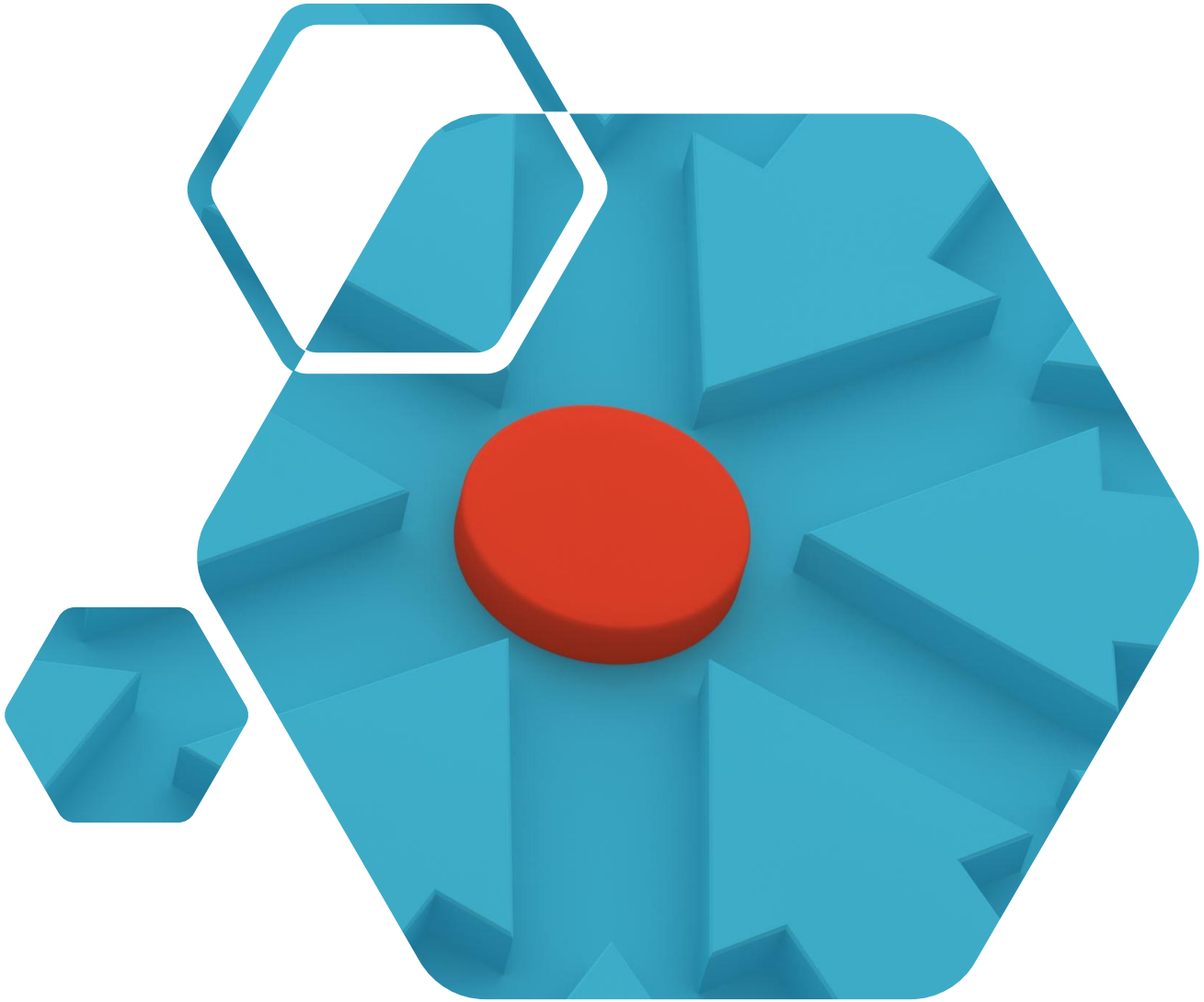
- Problems you want to find answers

  To develop a Facial Expression Recognition system which accurately recognizes facial expression in real time.

Loading Data

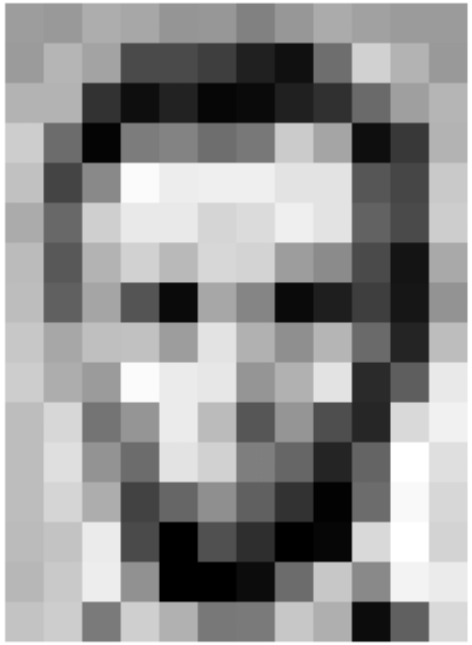
	emotion	Usage	pixels
0	0	Training	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...
1	0	Training	151 150 147 155 148 133 111 140 170 174 182 15...
2	2	Training	231 212 156 164 174 138 161 173 182 200 106 38...
3	4	Training	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...
4	6	Training	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...

Methodology



Data Wrangling

In the dataset there are pixel data of many images (48x48) in string format, so very first we need to convert the pixel data into array format.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Data Wrangling - Preprocessing

pixels_to_array converts the string pixel data into array.

image_reshape converts the 1D array into 2D array, which means the image can be regenerated using the array data.

```
def pixels_to_array(pixels):
    """
    function to convert the string pixels into list of float data
    """
    return np.array(pixels.split(), 'float64')


def image_reshape(data):
    """
    function to convert single dimension array into 2D numpy array
    """
    image_array = np.zeros(shape=(len(data), 48, 48))

    for i, pixel in enumerate(data):
        image_array[i] = pixel.reshape(48, 48)

    return image_array
```

After preprocessing the image data our data looks similar to data shown in image

```
X[:3]
```

```
array([[[ 70.,  80.,  82., ...,  52.,  43.,  41.],
        [ 65.,  61.,  58., ...,  56.,  52.,  44.],
        [ 50.,  43.,  54., ...,  49.,  56.,  47.],
        ...,
        [ 91.,  65.,  42., ...,  72.,  56.,  43.],
        [ 77.,  82.,  79., ..., 105.,  70.,  46.],
        [ 77.,  72.,  84., ..., 106., 109.,  82.]],

       [[151., 150., 147., ..., 129., 140., 120.],
        [151., 149., 149., ..., 122., 141., 137.],
        [151., 151., 156., ..., 109., 123., 146.],
        ...,
        [188., 188., 121., ..., 185., 185., 186.],
        [188., 187., 196., ..., 186., 182., 187.],
        [186., 184., 185., ..., 193., 183., 184.]],

       [[231., 212., 156., ...,  44.,  27.,  16.],
        [229., 175., 148., ...,  27.,  35.,  27.],
        [214., 156., 157., ...,  28.,  22.,  28.],
        ...,
        [241., 245., 250., ...,  57., 101., 146.],
        [246., 250., 252., ...,  78., 105., 162.],
        [250., 251., 250., ...,  88., 110., 152.]]])
```

1.

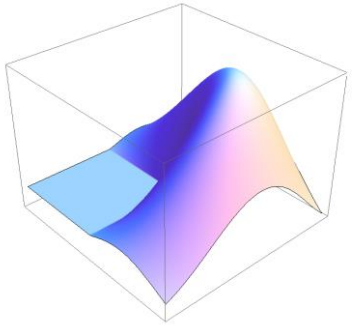
2.

3.

EDA with Data Visualization

Image Plot:

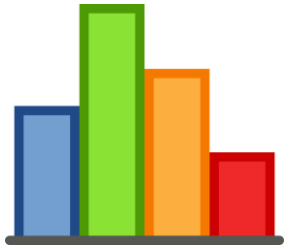
- Sample Image from dataset



Plotting array data to represent an picture using matplotlib.

Bar Graph:

- Label Proportion in split dataset



A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

Predictive Analysis (Classification)

Model Building

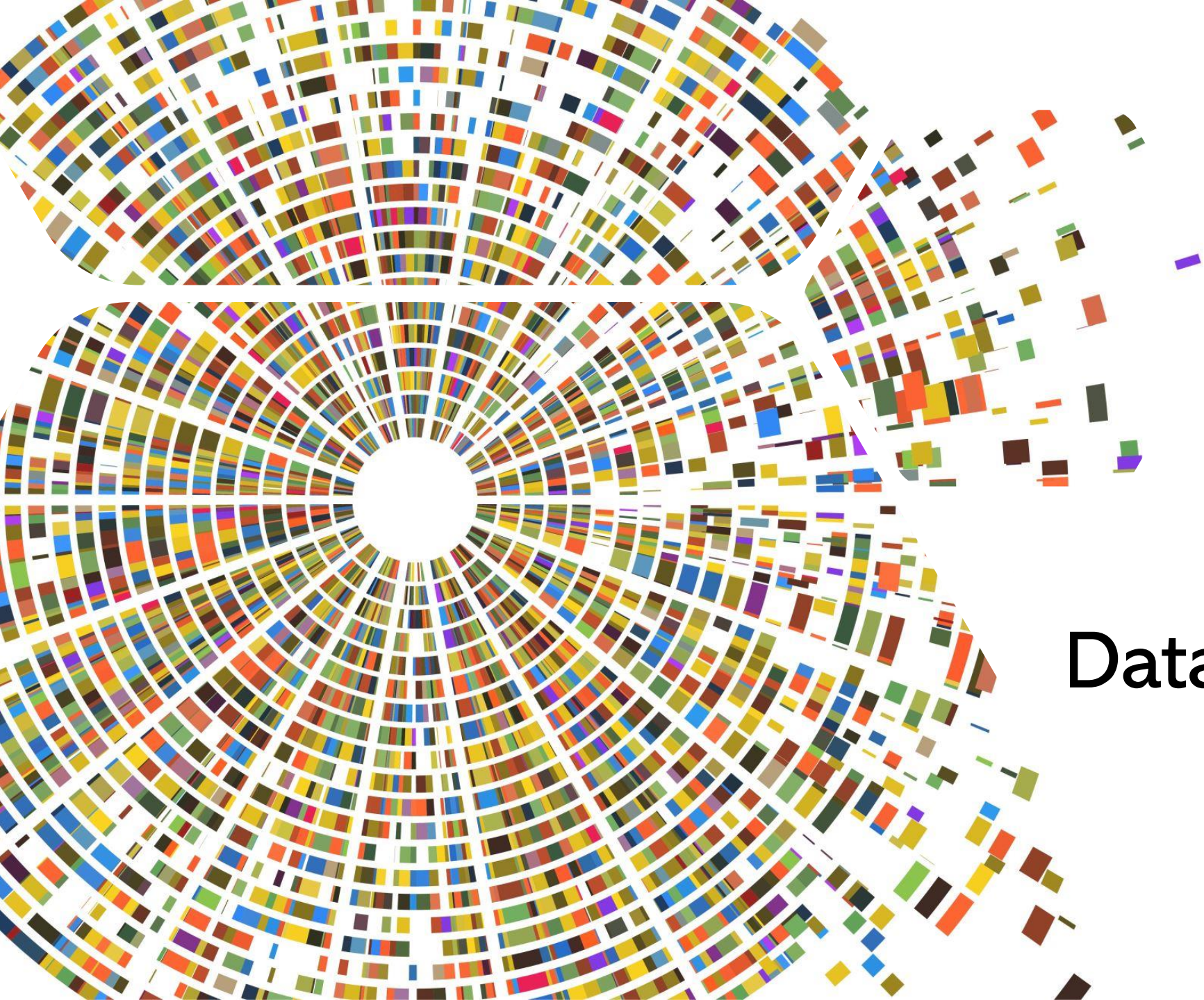
- Pandas to Load and manipulate data
- Numpy to create an array
- Standardize the data
- Splitting data for training and testing model
- Initializing CNN model

Model Evaluation

- Best parameter
- Calculating accuracy of model

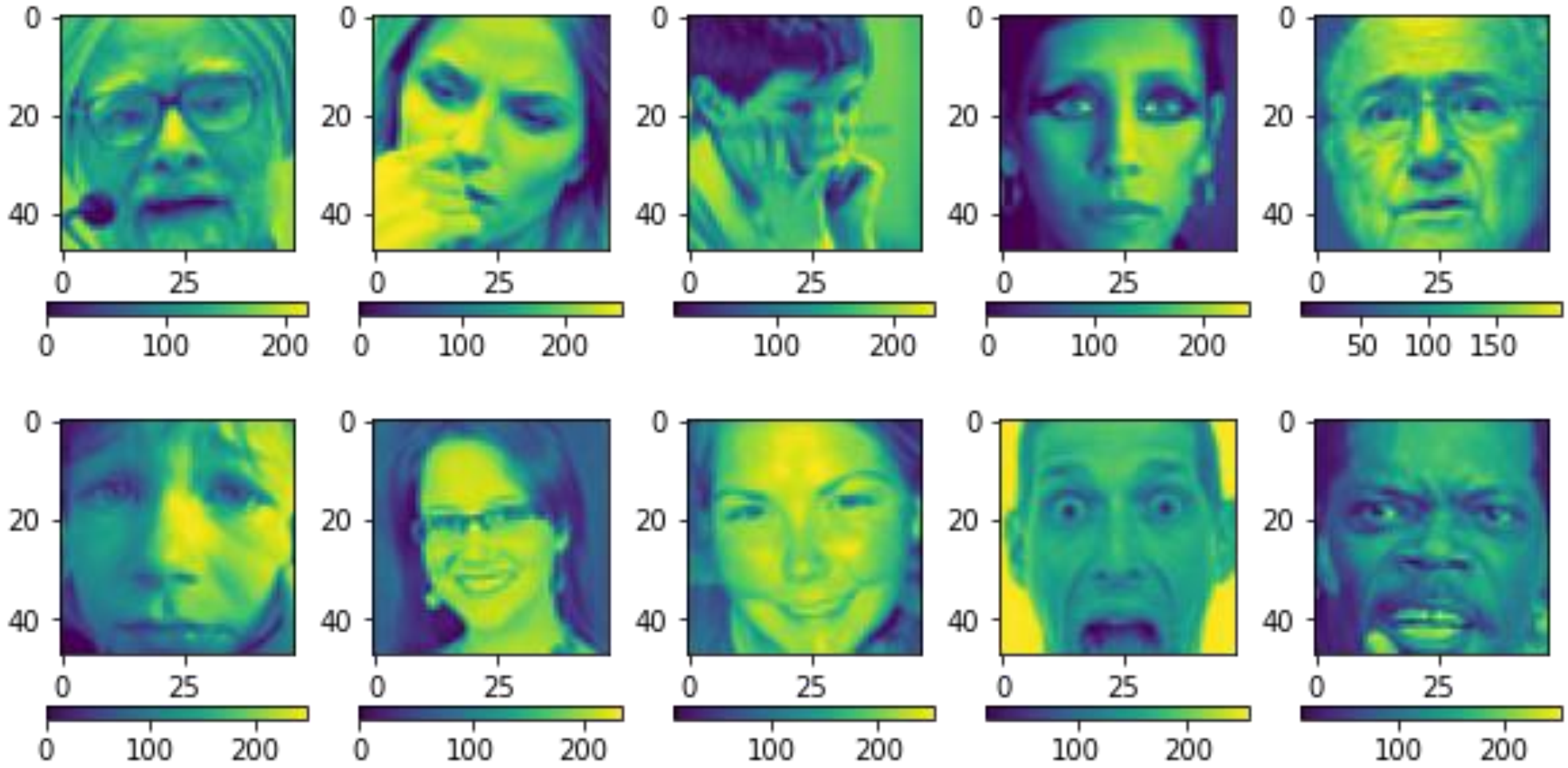
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Data Visualization

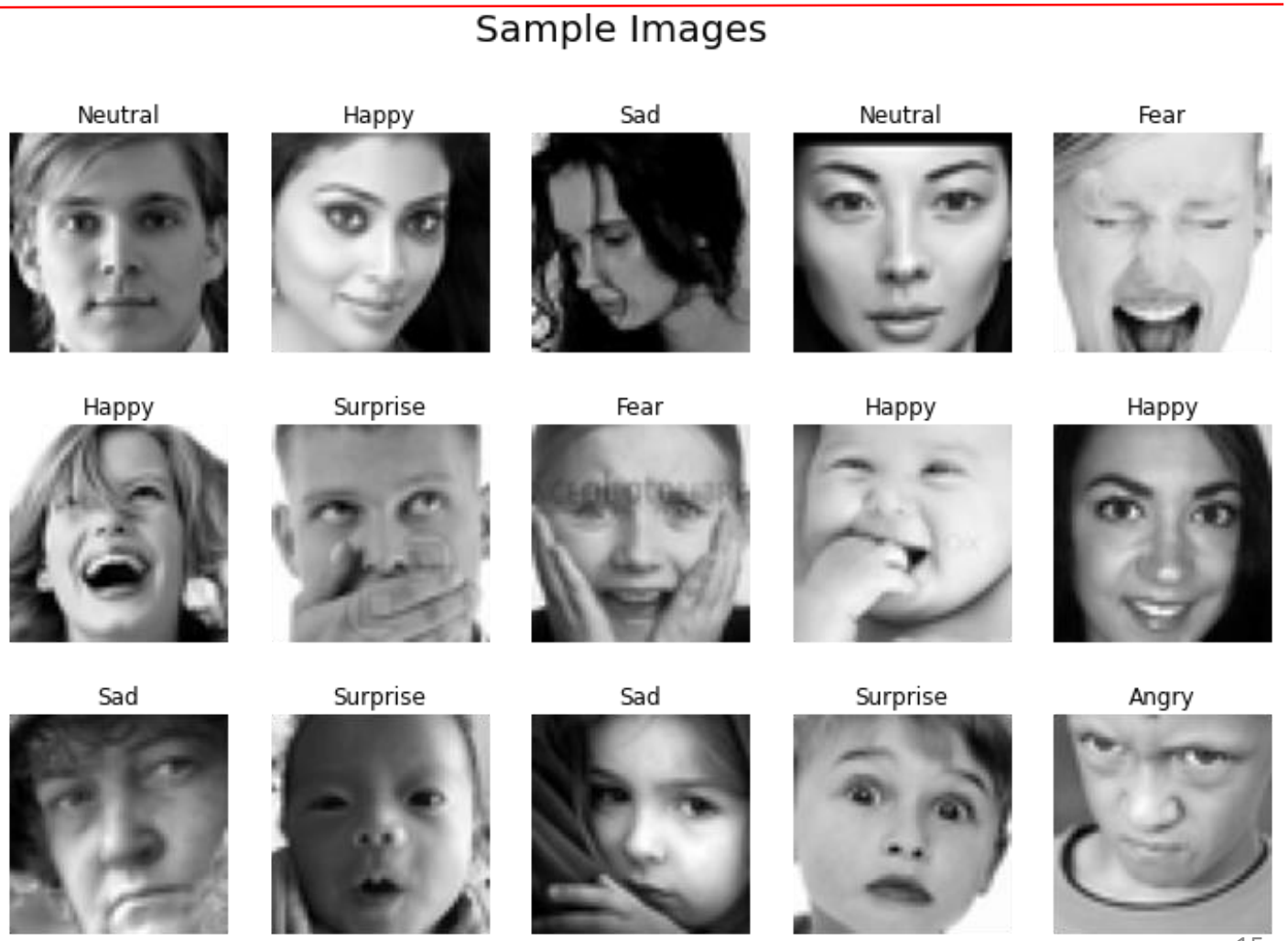
Sample Image Data



Sample Image Data

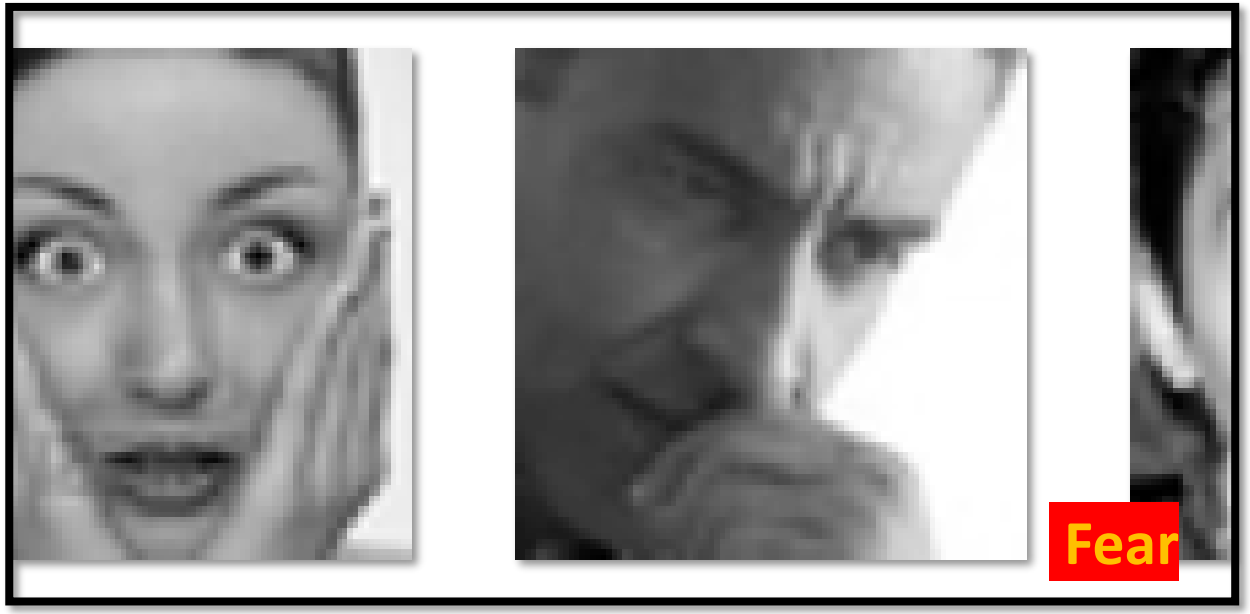
Here we have listed out 15 images,

- Each image are of size 48x48
- Every image contains only face part
- There are 7 different emotions.




Angry


Happy


Fear


Disgust

Sad

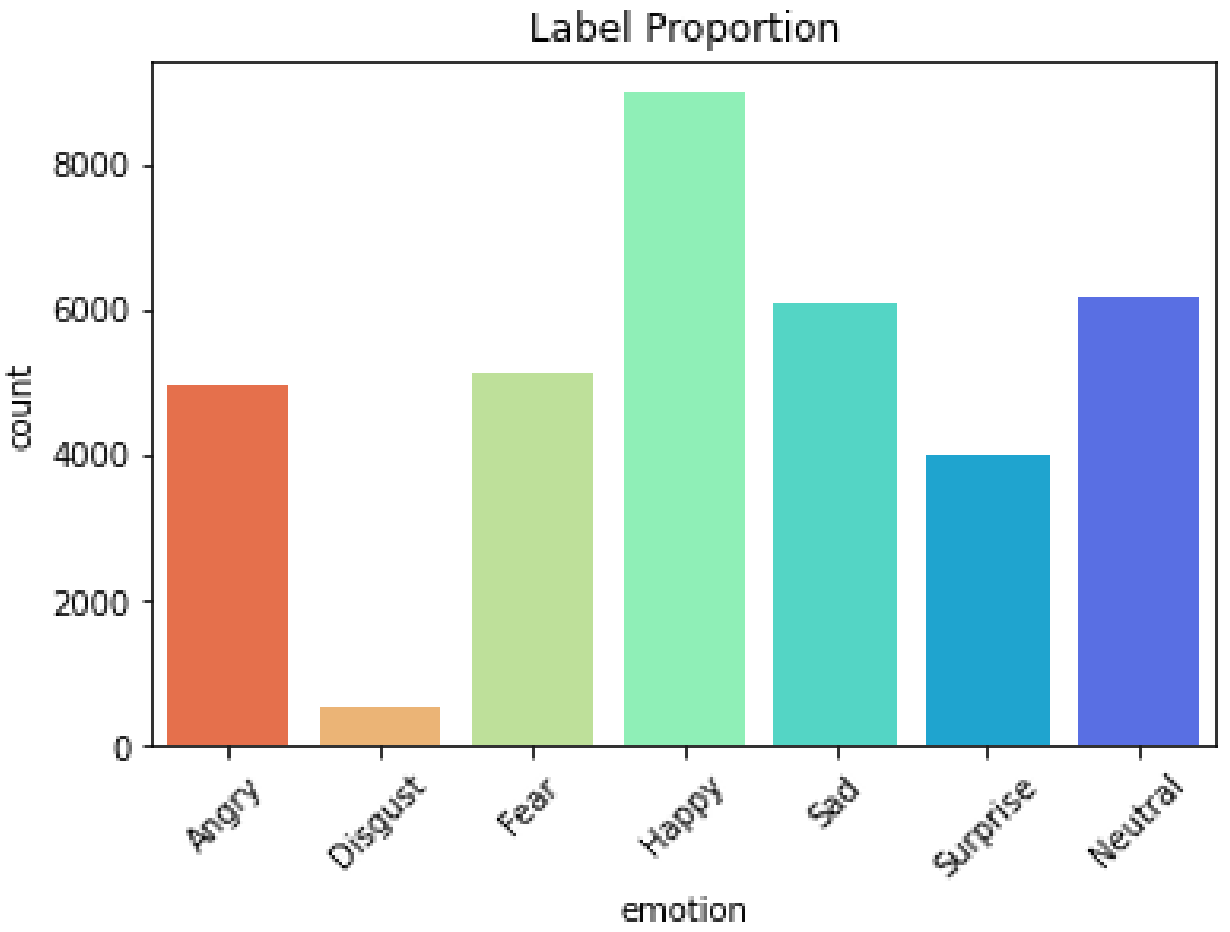
Surprise

Neutral

Label Proportion

Highest availability of image with Happy emotion.

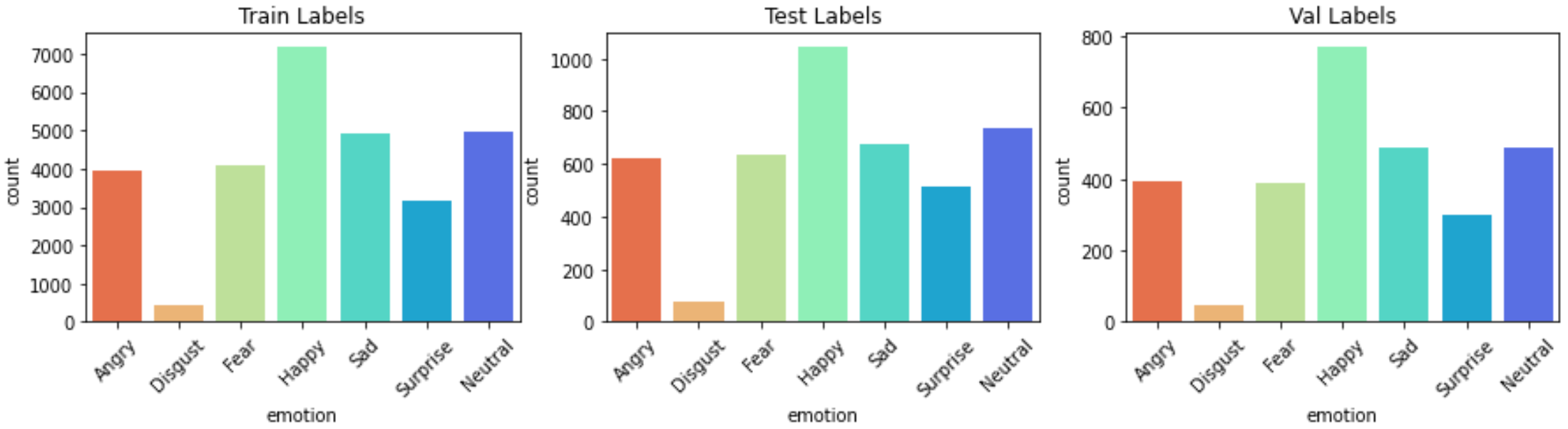
Lowest availability of Disgust emotion.



Label Proportion

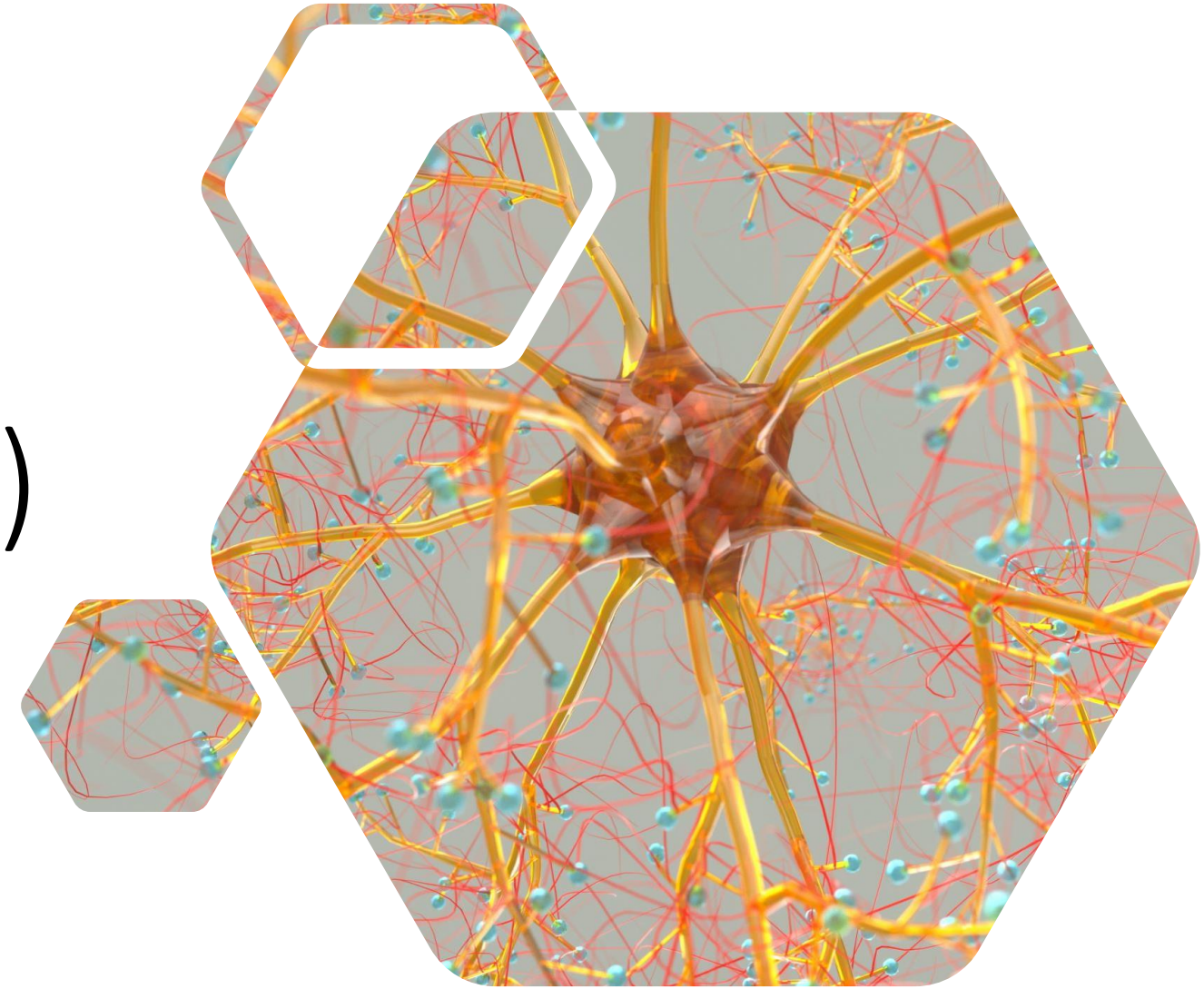
Train Test Split

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=121)
x_test, x_val, y_test, y_val = train_test_split(x_test, y_test, test_size=0.4, random_state=121)
```



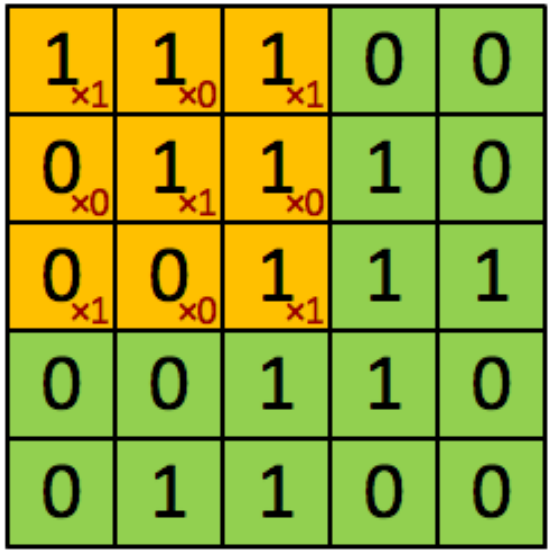
Train, test and validation set are likely having the same proportion of emotions data.

Predictive Analysis (Classification)

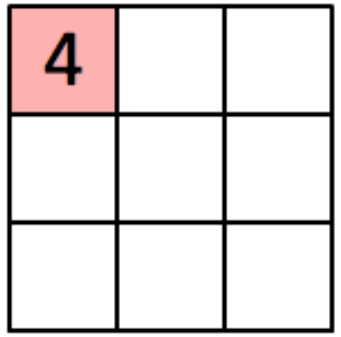


CNN Model

CNNs use image recognition and classification in order to detect objects, recognize faces, etc. They are made up of neurons with learnable weights and biases. Each specific neuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output.



Image

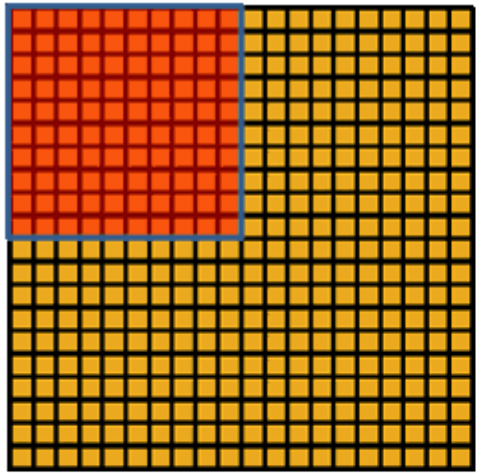


Convolved Feature

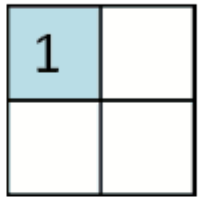
CNN Model

A common CNN model architecture is to have a number of convolution and pooling layers stacked one after the other.

- Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.
- The pooling layer summarises the features present in a region of the feature map generated by a convolution layer.



Convolved feature



Pooled feature

CNN - Defining Sequential Model

Step 1: Adding an Input Layer

```
tf.keras.layers.experimental.preprocessing.Rescaling(scale=1./255, input_shape=(48,48,1)),
tf.keras.layers.experimental.preprocessing.RandomContrast(factor = 0.2),
tf.keras.layers.experimental.preprocessing.RandomFlip(mode='horizontal'),
```

Step 2: Adding 2D convolution layer

```
tf.keras.layers.Conv2D(16,3,activation='relu',padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Conv2D(16,5,activation='relu',padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Conv2D(16,3,activation='relu',padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.MaxPooling2D(2),

tf.keras.layers.Conv2D(16,3,activation='relu',padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Conv2D(16,3,activation='relu',padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.2),
```

Step 3: reshapes the tensor to have a shape that is equal to the number of elements contained in the tensor

```
tf.keras.layers.Flatten(),
tf.keras.layers.Dropout(0.4),
```

Step 4: Classification Output

```
tf.keras.layers.Dense(256, activation='relu'),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(len(emotions), activation='softmax'),
```

Step 5: Optimizing Model

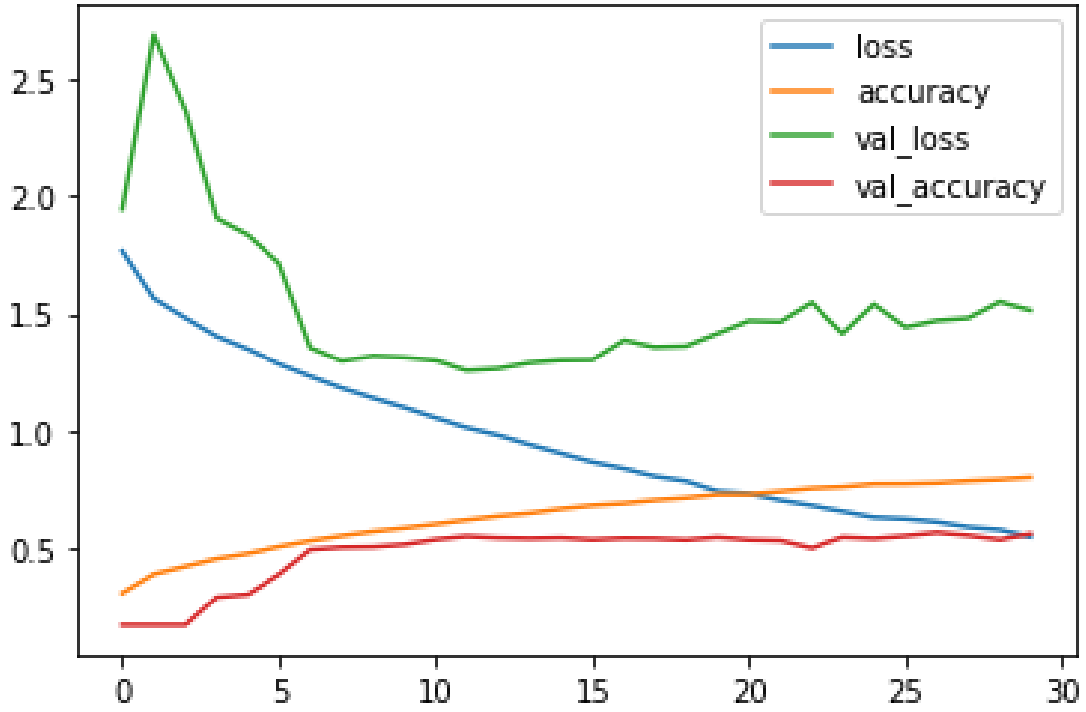
```
cnn.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
            loss='categorical_crossentropy',
            metrics=['accuracy'], run_eagerly=True)
earlystop = tf.keras.callbacks.EarlyStopping(patience=10, min_delta=1e-4,
                                             restore_best_weights=True)
lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', patience=3,
                                          verbose=1, factor=0.5, min_lr=1e-7)
```

Link to Notebook (click here)

Classification Accuracy

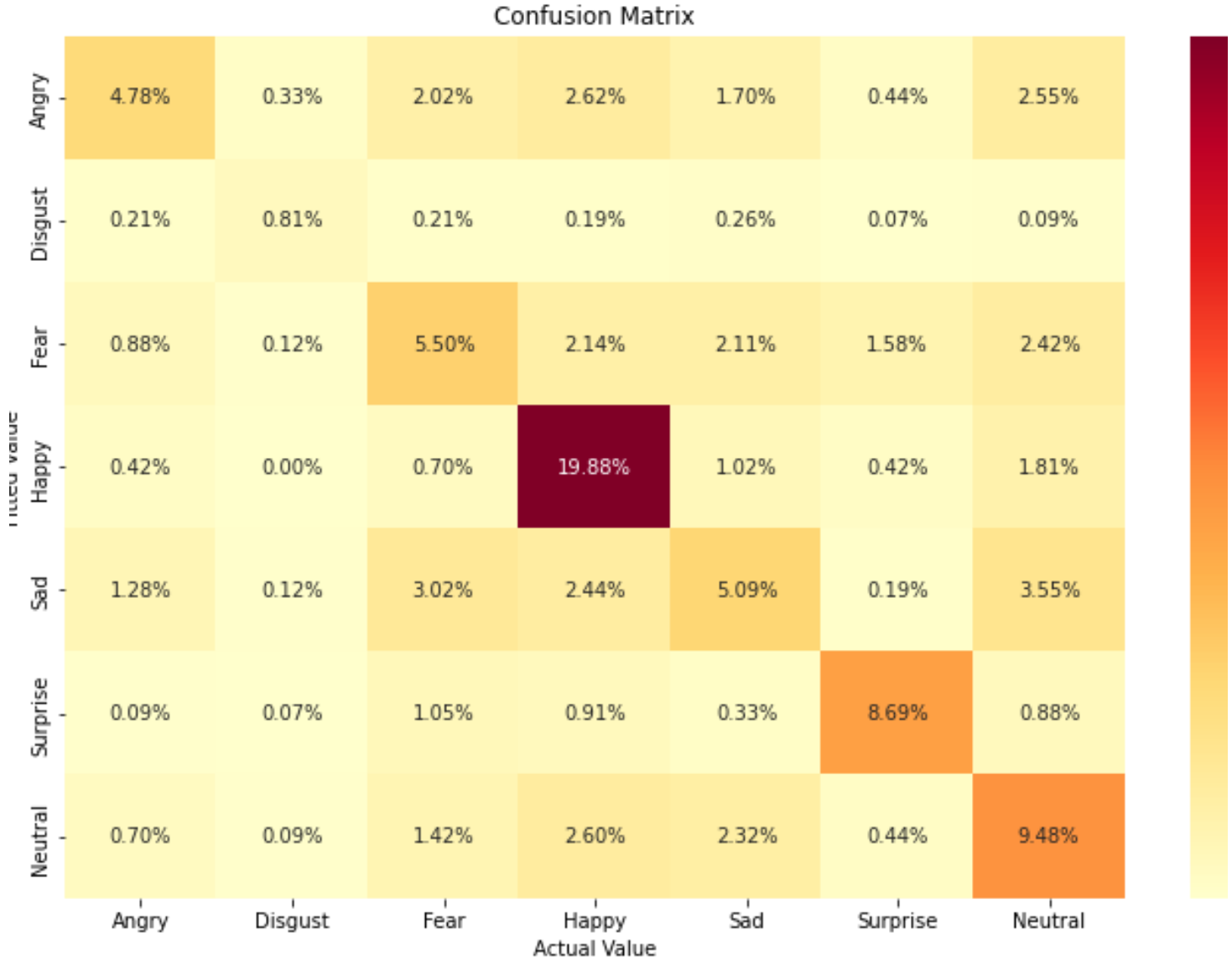
Decrease in loss has been recorded with the learning rate of 0.001

```
for key in h1.history.keys():
    plt.plot(h1.history[key], label=key)
plt.legend()
```



Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	TN TRUE NEGATIVE	FP FALSE POSITIVE
Actual 1	FN FALSE NEGATIVE	TP TRUE POSITIVE



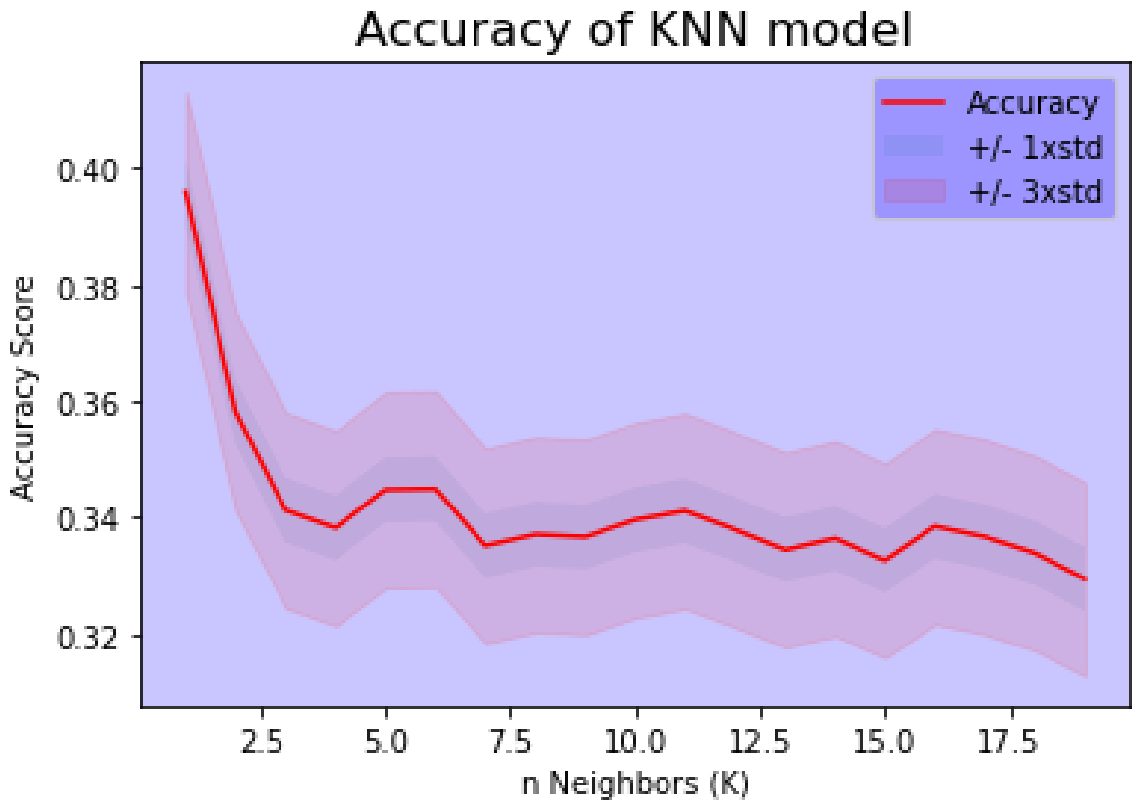
Conclusion

Face detection and emotion recognition are very challenging problems and we've successfully developed a Deep Learning model implemented with computer vision which is now capable of predicting the emotion in real time.

Appendix

- Tensorflow Docs
- Keras Docs
- CNN (geeks for geeks)
- Max Pooling Layer
- Open CV Docs

Extras -Accuracy with KNN Model



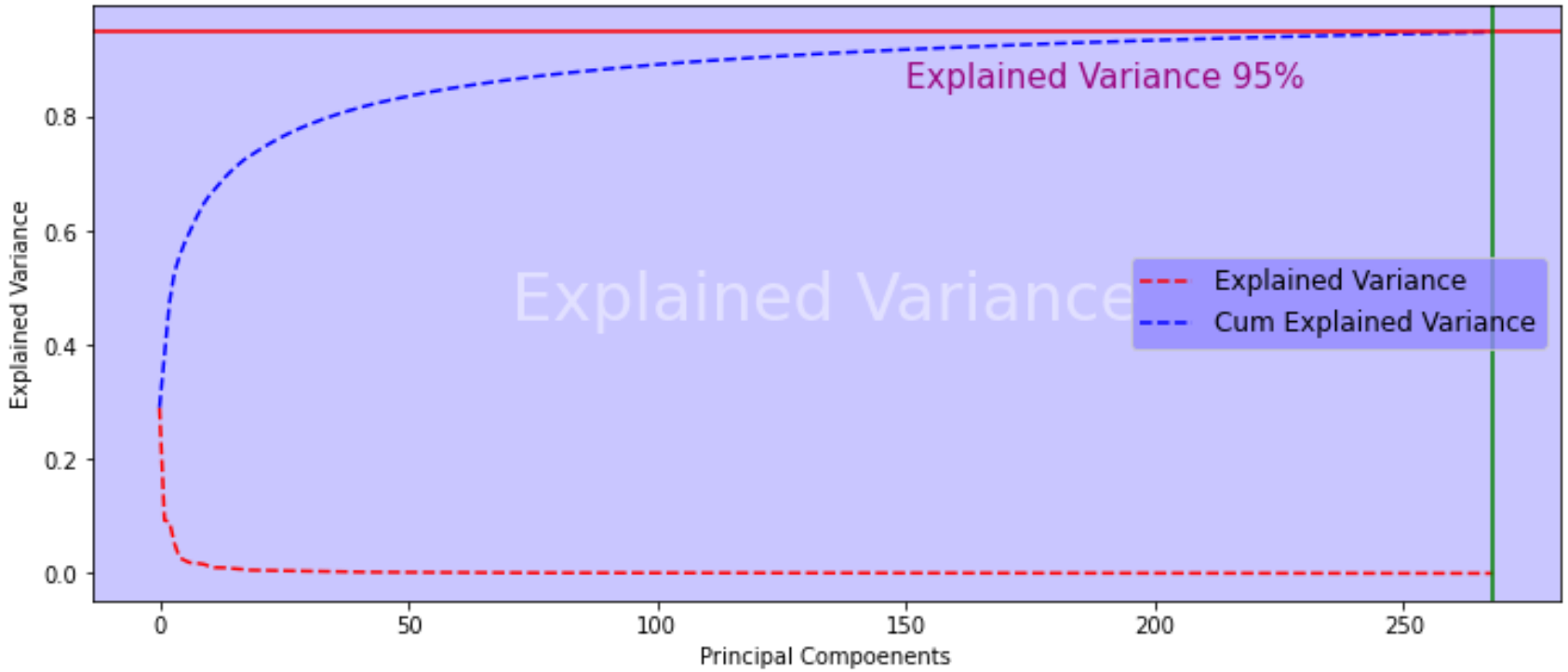

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn = KNeighborsClassifier(n_neighbors=169, n_jobs=-1)
knn.fit(x_train,y_train)
y_knn_pred = knn.predict(x_test)
```

```
accuracy_score(y_test, y_knn_pred)
```

```
0.3007801616049039
```

30 %

[Link to Notebook (click here)]



Extras – Optimizing Model

After PCA we have found that 250 features are enough for good classification.

[Link to Notebook (click here)]

Setting up the application on the local system

Source Code: [Find Project on Github (click here)]

1

Download

- Find the facial-exp.py file in Github repo.

2

Download

- Also download the facial-exp.h5 model here)

3

Place

- Place both the file in same directory

4

Run

- Open terminal and run the command python facial-exp.py

Thank You!