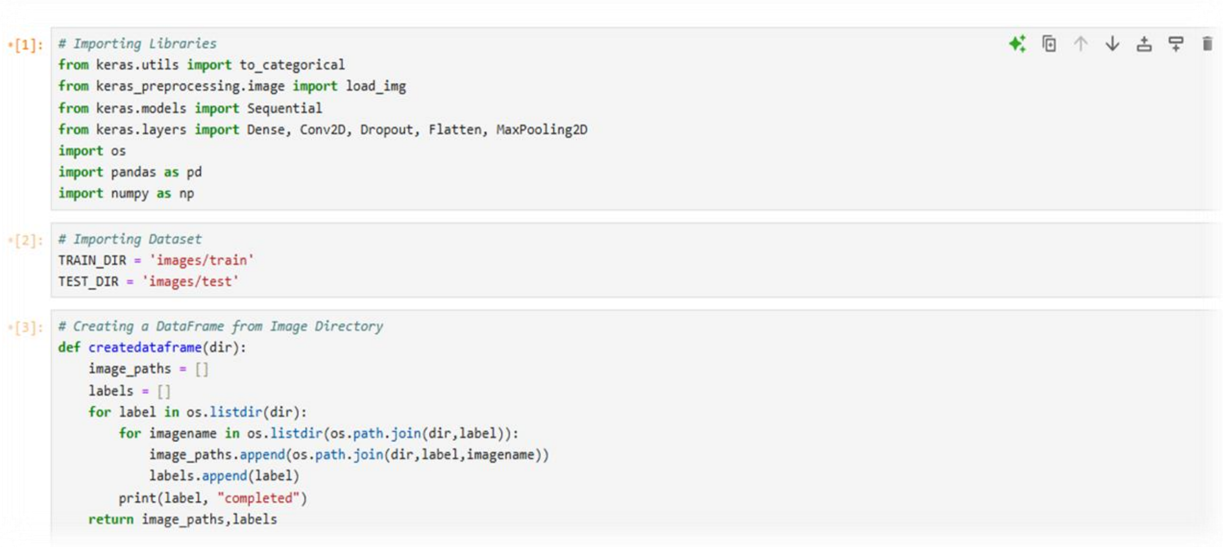
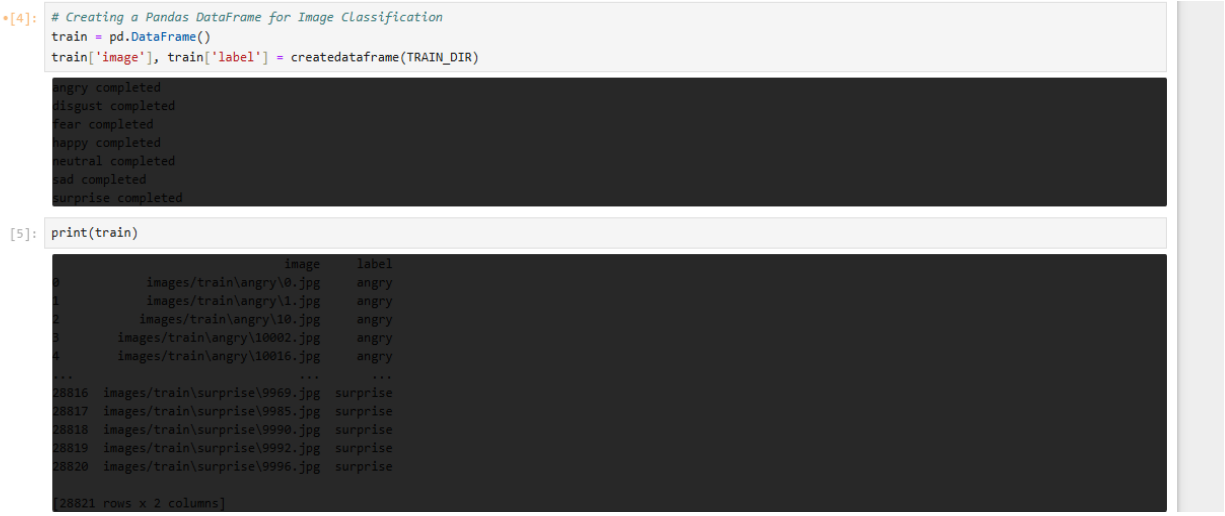


```
•[1]:  # Importing Libraries
       from keras.utils import to_categorical
       from keras_preprocessing.image import load_img
       from keras.models import Sequential
       from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
       import os
       import pandas as pd
       import numpy as np
```

```
•[2]:  # Importing Dataset
       TRAIN_DIR = 'images/train'
       TEST_DIR = 'images/test'
```

```
•[3]:  # Creating a DataFrame from Image Directory
       def createdataframe(dir):
           image_paths = []
           labels = []
           for label in os.listdir(dir):
               for imagename in os.listdir(os.path.join(dir,label)):
                   image_paths.append(os.path.join(dir,label,imagename))
                   labels.append(label)
               print(label, "completed")
           return image_paths,labels
```

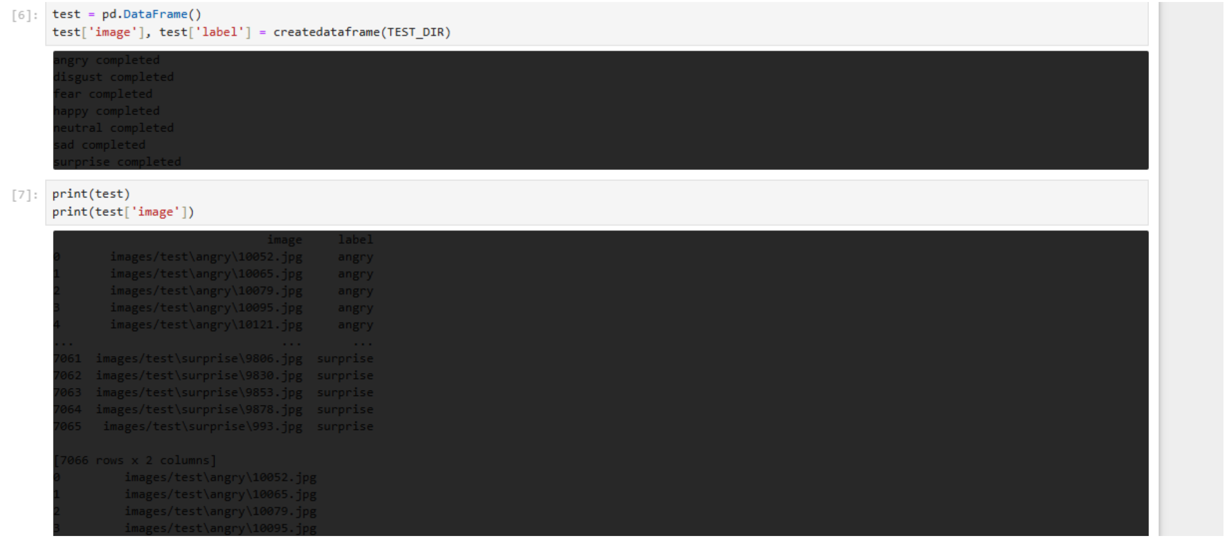


```
•[4]:  # Creating a Pandas DataFrame for Image Classification
       train = pd.DataFrame()
       train['image'], train['label'] = createdataframe(TRAIN_DIR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

```
[5]:  print(train)
```

```
                              image     label
0             images/train\angry\0.jpg     angry
1             images/train\angry\1.jpg     angry
2            images/train\angry\10.jpg     angry
3         images/train\angry\10002.jpg     angry
4         images/train\angry\10016.jpg     angry
...                                ...       ...
28816  images/train\surprise\9969.jpg  surprise
28817  images/train\surprise\9985.jpg  surprise
28818  images/train\surprise\9990.jpg  surprise
28819  images/train\surprise\9992.jpg  surprise
28820  images/train\surprise\9996.jpg  surprise

[28821 rows x 2 columns]
```

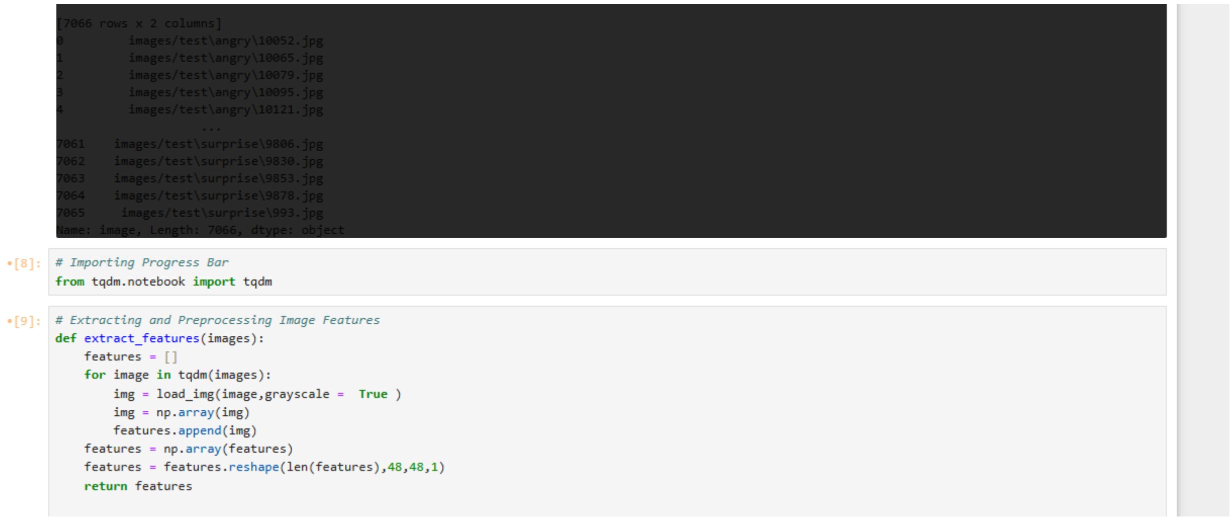


```
[6]:  test = pd.DataFrame()
      test['image'], test['label'] = createdataframe(TEST_DIR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

```
[7]:  print(test)
      print(test['image'])
```

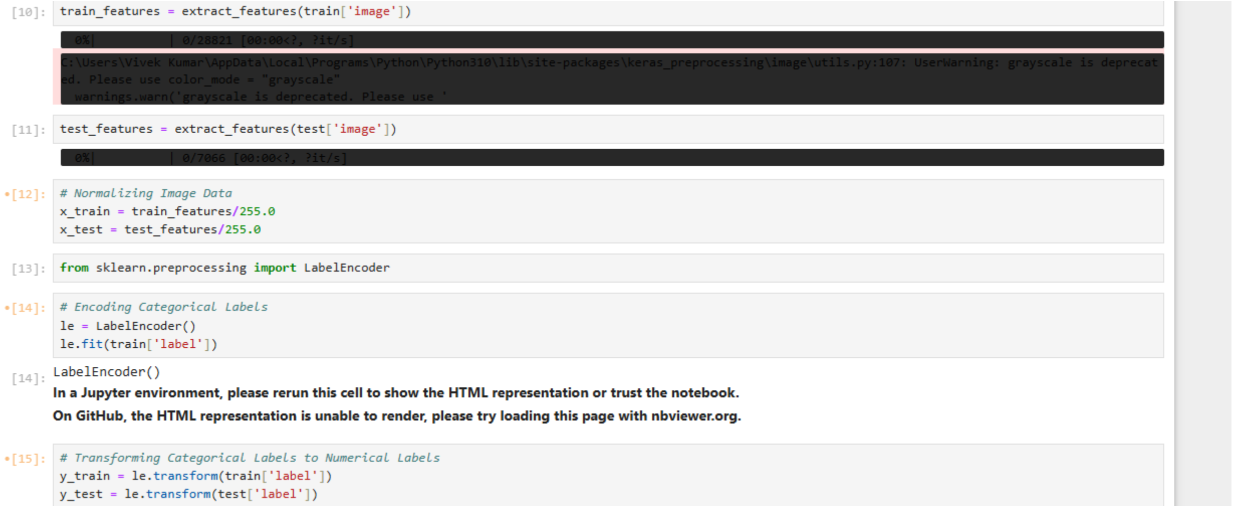
```
                              image     label
0        images/test\angry\10052.jpg     angry
1        images/test\angry\10065.jpg     angry
2        images/test\angry\10079.jpg     angry
3        images/test\angry\10095.jpg     angry
4        images/test\angry\10121.jpg     angry
...                              ...       ...
7061  images/test\surprise\9806.jpg  surprise
7062  images/test\surprise\9830.jpg  surprise
7063  images/test\surprise\9853.jpg  surprise
7064  images/test\surprise\9878.jpg  surprise
7065   images/test\surprise\993.jpg  surprise

[7066 rows x 2 columns]
0          images/test\angry\10052.jpg
1          images/test\angry\10065.jpg
2          images/test\angry\10079.jpg
3          images/test\angry\10095.jpg
```



```
[7066 rows x 2 columns]
0         images/test\angry\10052.jpg
1         images/test\angry\10065.jpg
2         images/test\angry\10079.jpg
3         images/test\angry\10095.jpg
4         images/test\angry\10121.jpg
                     ...
7061    images/test\surprise\9806.jpg
7062    images/test\surprise\9830.jpg
7063    images/test\surprise\9853.jpg
7064    images/test\surprise\9878.jpg
7065     images/test\surprise\993.jpg
Name: image, Length: 7066, dtype: object
```

```
•[8]: # Importing Progress Bar
      from tqdm.notebook import tqdm
```

```
•[9]: # Extracting and Preprocessing Image Features
      def extract_features(images):
          features = []
          for image in tqdm(images):
              img = load_img(image,grayscale =  True )
              img = np.array(img)
              features.append(img)
          features = np.array(features)
          features = features.reshape(len(features),48,48,1)
          return features
```



```
[10]: train_features = extract_features(train['image'])
```

```
  0%|          | 0/28821 [00:00<?, ?it/s]
C:\Users\Vivek Kumar\AppData\Local\Programs\Python\Python310\lib\site-packages\keras_preprocessing\image\utils.py:107: UserWarning: grayscale is deprecat
ed. Please use color_mode = "grayscale"
  warnings.warn('grayscale is deprecated. Please use '
```

```
[11]: test_features = extract_features(test['image'])
```

```
  0%|          | 0/7066 [00:00<?, ?it/s]
```

```
•[12]: # Normalizing Image Data
       x_train = train_features/255.0
       x_test = test_features/255.0
```

```
[13]: from sklearn.preprocessing import LabelEncoder
```

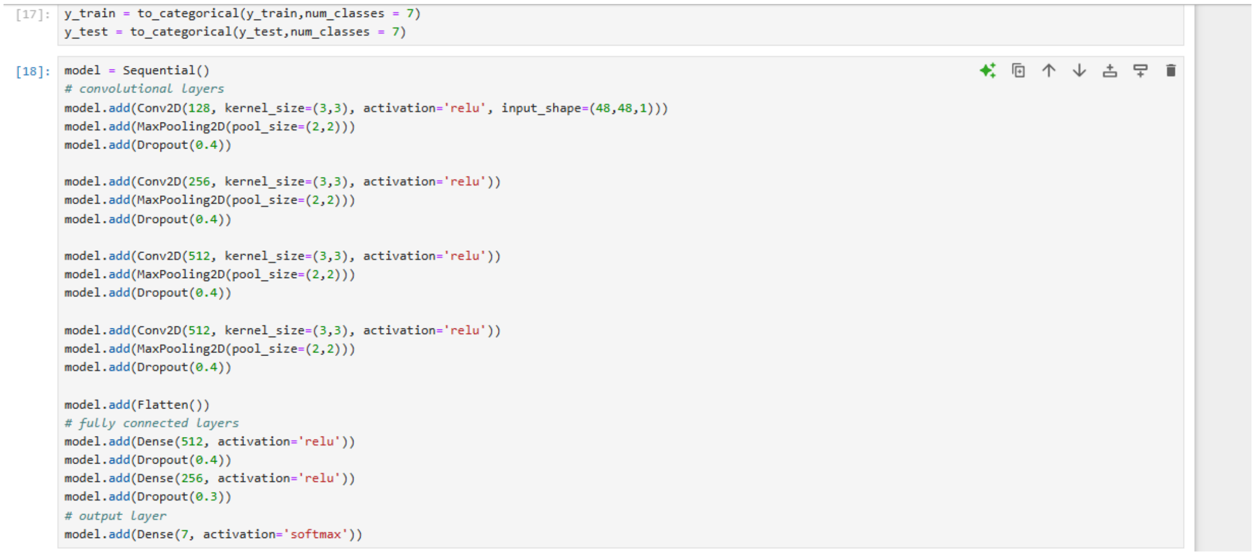
```
•[14]: # Encoding Categorical Labels
       le = LabelEncoder()
       le.fit(train['label'])
```

```
[14]: LabelEncoder()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

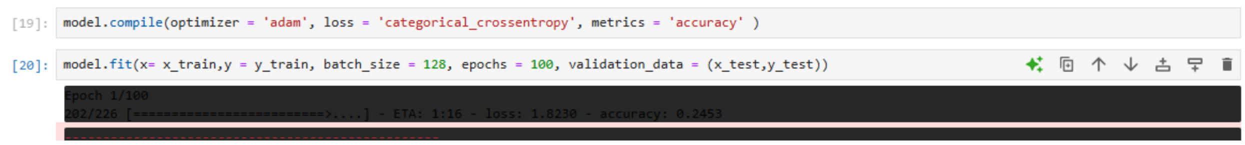
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
•[15]: # Transforming Categorical Labels to Numerical Labels
       y_train = le.transform(train['label'])
       y_test = le.transform(test['label'])
```



```
[17]: y_train = to_categorical(y_train,num_classes = 7)
      y_test = to_categorical(y_test,num_classes = 7)
```

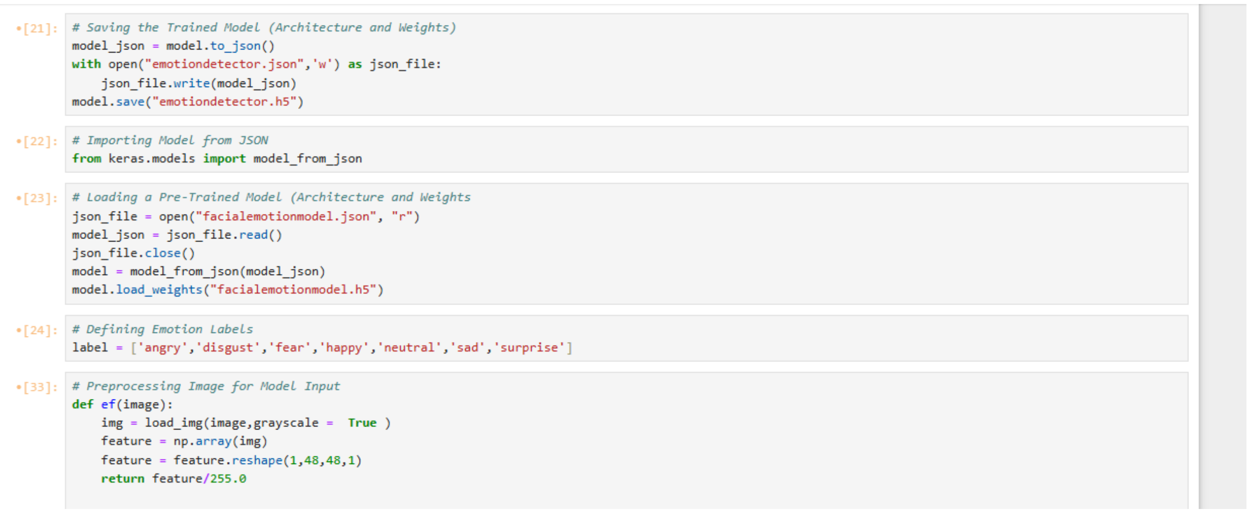
```
[18]: model = Sequential()
      # convolutional layers
      model.add(Conv2D(128, kernel_size=(3,3), activation='relu', input_shape=(48,48,1)))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Dropout(0.4))

      model.add(Conv2D(256, kernel_size=(3,3), activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Dropout(0.4))

      model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Dropout(0.4))

      model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
      model.add(MaxPooling2D(pool_size=(2,2)))
      model.add(Dropout(0.4))

      model.add(Flatten())
      # fully connected layers
      model.add(Dense(512, activation='relu'))
      model.add(Dropout(0.4))
      model.add(Dense(256, activation='relu'))
      model.add(Dropout(0.3))
      # output layer
      model.add(Dense(7, activation='softmax'))
```



```
[19]: model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = 'accuracy' )
```

```
[20]: model.fit(x= x_train,y = y_train, batch_size = 128, epochs = 100, validation_data = (x_test,y_test))
```

```
Epoch 1/100
202/226 [==========================>....] - ETA: 1:16 - loss: 1.8230 - accuracy: 0.2453
---------------------------------------------------
```



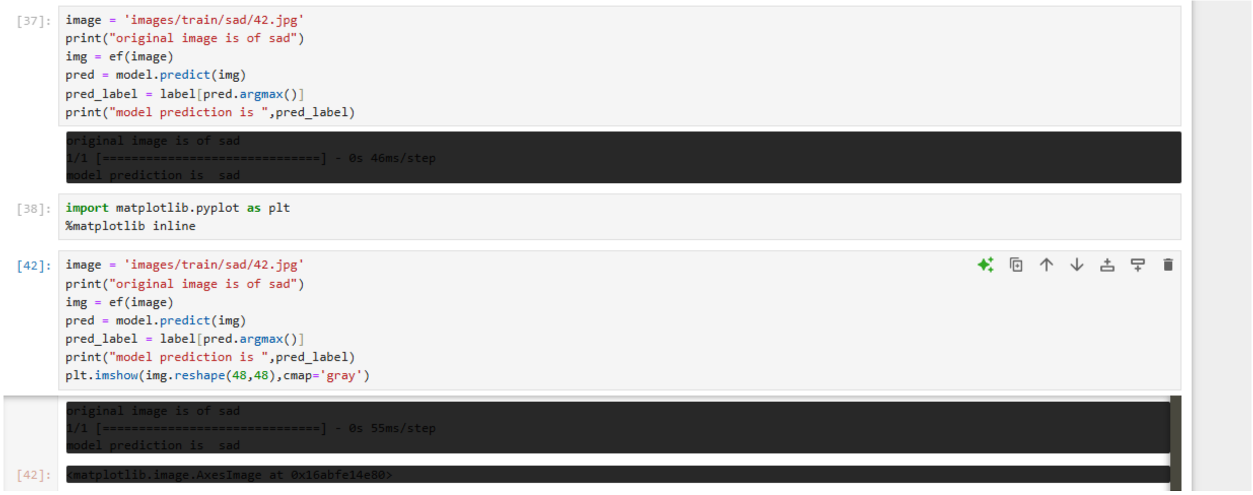
```
[21]: # Saving the Trained Model (Architecture and Weights)
model_json = model.to_json()
with open("emotiondetector.json",'w') as json_file:
    json_file.write(model_json)
model.save("emotiondetector.h5")
```

```
[22]: # Importing Model from JSON
from keras.models import model_from_json
```

```
[23]: # Loading a Pre-Trained Model (Architecture and Weights
json_file = open("facialemotionmodel.json", "r")
model_json = json_file.read()
json_file.close()
model = model_from_json(model_json)
model.load_weights("facialemotionmodel.h5")
```

```
[24]: # Defining Emotion Labels
label = ['angry','disgust','fear','happy','neutral','sad','surprise']
```

```
[33]: # Preprocessing Image for Model Input
def ef(image):
    img = load_img(image,grayscale =  True )
    feature = np.array(img)
    feature = feature.reshape(1,48,48,1)
    return feature/255.0
```



```
[37]: image = 'images/train/sad/42.jpg'
print("original image is of sad")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
```

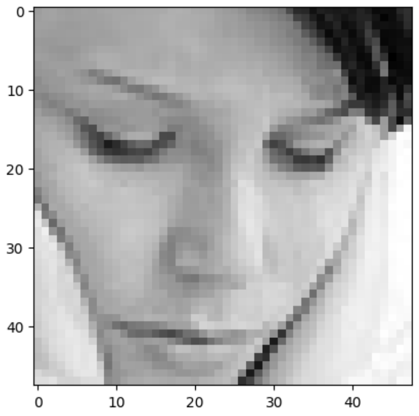
```
original image is of sad
1/1 [==============================] - 0s 46ms/step
model prediction is  sad
```

```
[38]: import matplotlib.pyplot as plt
%matplotlib inline
```

```
[42]: image = 'images/train/sad/42.jpg'
print("original image is of sad")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of sad
1/1 [==============================] - 0s 55ms/step
model prediction is  sad
```

```
[42]: <matplotlib.image.AxesImage at 0x16abfe14e80>
```



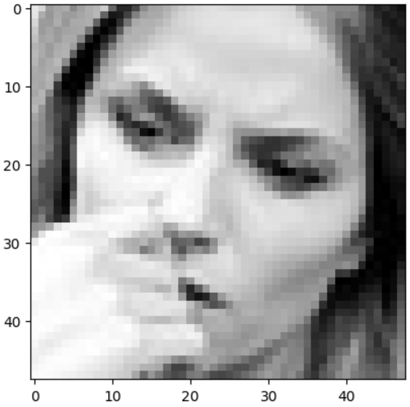
[43]:

```python
image = 'images/train/fear/2.jpg'
print("original image is of fear")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of fear
1/1 [==============================] - 0s 31ms/step
model prediction is  sad
```

[43]:

```
<matplotlib.image.AxesImage at 0x16abfe99060>
```



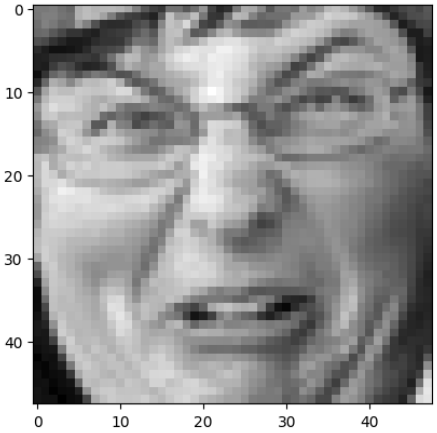
[44]:

```python
image = 'images/train/disgust/299.jpg'
print("original image is of disgust")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of disgust
1/1 [==============================] - 0s 57ms/step
model prediction is  disgust
```

[44]:

```
<matplotlib.image.AxesImage at 0x16abfef4d90>
```



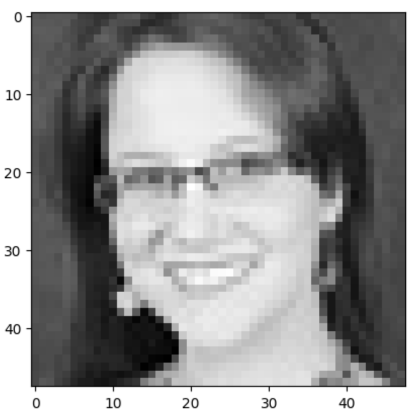
[45]:

```python
image = 'images/train/happy/7.jpg'
print("original image is of happy")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of happy
1/1 [==============================] - 0s 42ms/step
model prediction is  happy
```

[45]:

```
<matplotlib.image.AxesImage at 0x16ac00a8970>
```


0
10
20
30
40
0
10
20
30
40