In [ ]:
```
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
```

In [ ]:
```
TRAIN_DIR = 'images/train'
TEST_DIR = 'images/test'
```

In [ ]:
```
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
```

In [ ]:
```
train = pd.DataFrame()
train['image'], train['label'] = createdataframe(TRAIN_DIR)
```

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

In [ ]:
```
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]
```

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

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

In [ ]:
```
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
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
```

In [ ]:
```
from tqdm.notebook import tqdm
```

In [ ]:
```
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
```

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

```
  0%|          | 0/28821 [00:00<?, ?it/s]
```

```
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\keras_preprocessing
\image\utils.py:107: UserWarning: grayscale is deprecated. Please use color_mode
= "grayscale"
  warnings.warn('grayscale is deprecated. Please use '
```

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

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

In [ ]:
```
x_train = train_features/255.0
x_test = test_features/255.0
```

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

In [ ]:
```
le = LabelEncoder()
le.fit(train['label'])
```

Out[ ]:
```
▾ LabelEncoder
LabelEncoder()
```

In [ ]:
```
y_train = le.transform(train['label'])
y_test = le.transform(test['label'])
```

In [ ]:
```
# Trying basic ML model like Linear and Logistic Regression
```

In [ ]:
```
x_tr = np.vstack([x_train[i].flatten() for i in range(x_train.shape[0])])
x_te = np.vstack([x_test[i].flatten() for i in range(x_test.shape[0])])
y_tr = y_train
y_te = y_test
len(x_tr[0])
```

Out[ ]:
```
2304
```

In [ ]:
```
X = np.concatenate((x_tr, x_te))
y = np.concatenate((y_tr, y_te))
```

In [ ]:
```
dftr = pd.DataFrame(x_tr)
dftr["result"] = y_tr
dftr
```

Out[ ]:

	0	1	2	3	4	5	6	7
0	0.282353	0.305882	0.317647	0.294118	0.231373	0.211765	0.247059	0.239216
1	0.596078	0.584314	0.576471	0.615686	0.572549	0.521569	0.447059	0.541176
2	0.113725	0.098039	0.082353	0.090196	0.101961	0.094118	0.192157	0.262745
3	0.125490	0.090196	0.078431	0.219608	0.168627	0.133333	0.149020	0.180392
4	0.870588	0.854902	0.792157	0.741176	0.780392	0.815686	0.756863	0.525490
...	...	...	...	...	...	...	...	...
28816	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
28817	0.509804	0.525490	0.517647	0.517647	0.447059	0.407843	0.368627	0.349020
28818	0.952941	0.862745	0.811765	0.796078	0.792157	0.749020	0.811765	0.737255
28819	0.992157	0.988235	0.996078	0.980392	1.000000	0.909804	0.435294	0.352941
28820	0.835294	0.854902	0.839216	0.854902	0.882353	0.850980	0.843137	0.811765

28821 rows × 2305 columns

In [ ]:
```python
dfte = pd.DataFrame(x_te)
dfte["result"] = y_te
dfte
```

Out[ ]:

	0	1	2	3	4	5	6	7	
0	0.227451	0.258824	0.274510	0.301961	0.458824	0.603922	0.537255	0.423529	0
1	0.090196	0.101961	0.082353	0.035294	0.023529	0.074510	0.129412	0.043137	0
2	0.788235	0.713725	0.713725	0.721569	0.803922	0.800000	0.796078	0.862745	0
3	0.364706	0.337255	0.305882	0.305882	0.313725	0.360784	0.427451	0.388235	0
4	0.043137	0.023529	0.003922	0.000000	0.000000	0.003922	0.000000	0.000000	0
...	...	...	...	...	...	...	...	...	
7061	1.000000	0.992157	1.000000	1.000000	0.992157	1.000000	1.000000	0.980392	0
7062	0.329412	0.278431	0.274510	0.266667	0.235294	0.184314	0.207843	0.176471	0
7063	0.980392	0.992157	0.992157	0.988235	0.988235	0.988235	0.992157	0.984314	0
7064	0.894118	0.878431	0.890196	0.792157	0.176471	0.031373	0.023529	0.031373	0
7065	0.305882	0.454902	0.384314	0.407843	0.431373	0.443137	0.501961	0.352941	0

7066 rows × 2305 columns

In [ ]:
```python
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
```

```
lr_clf.fit(x_tr,y_tr)
lr_clf.score(x_te, y_te)
```

Out[ ]:
```
-0.010575024068948169
```

In [ ]:
```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

cross_val_score(LinearRegression(), X, y, cv=cv)
```

Out[ ]:
```
array([ 0.01350426,  0.009681  ,  0.00026708, -0.00556879, -0.00540845])
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

cross_val_score(LogisticRegression(), X, y, cv=cv)
```

```
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_mode
l\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_mode
l\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_mode
l\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_mode
l\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_mode
l\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[ ]:
```
array([0.37238785, 0.3670939 , 0.37322374, 0.37252717, 0.36653664])
```

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

In [ ]:
```
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(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'))
```

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

In [ ]:
```
model.fit(x= x_train,y = y_train, batch_size = 128, epochs = 100, validation_dat
```

In [ ]:
```
model_json = model.to_json()
with open("emotiondetector.json",'w') as json_file:
    json_file.write(model_json)
model.save("emotiondetector.h5")
```

In [ ]:
```
from keras.models import model_from_json
```

In [ ]:
```
json_file = open("emotiondetector.json", "r")
model_json = json_file.read()
json_file.close()
model = model_from_json(model_json)
model.load_weights("emotiondetector.h5")
```

In [ ]:
```
label = ['angry','disgust','fear','happy','neutral','sad','surprise']
```

In [ ]:
```
def ef(image):
    img = load_img(image,grayscale =  True )
    feature = np.array(img)
    feature = feature.reshape(1,48,48,1)
    return feature/255.0
```

In [ ]:
```
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)
```

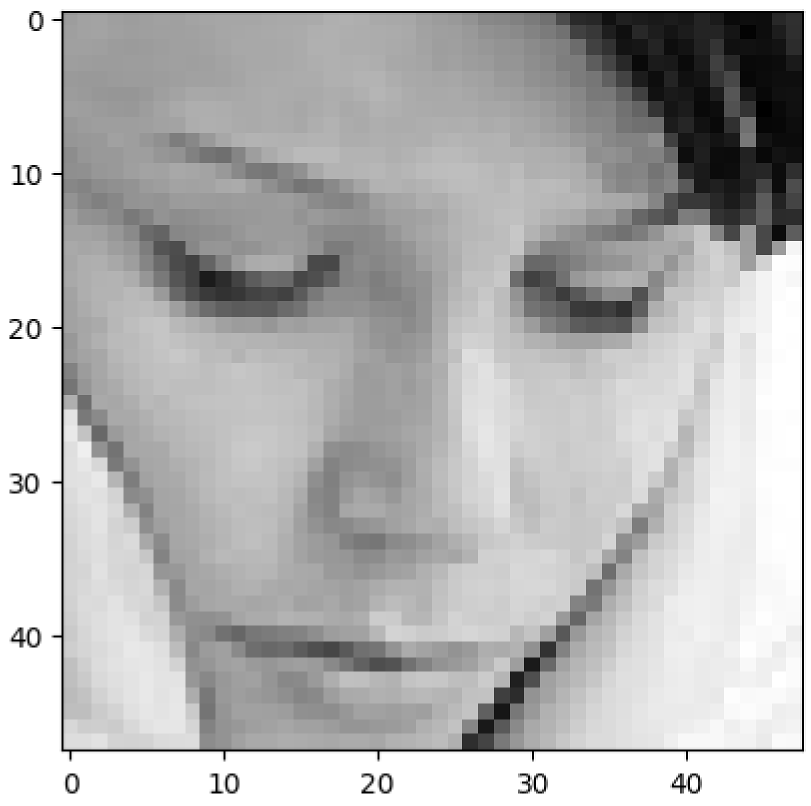
```
original image is of sad
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\keras_preprocessing
\image\utils.py:107: UserWarning: grayscale is deprecated. Please use color_mode
= "grayscale"
  warnings.warn('grayscale is deprecated. Please use '
1/1 [==============================] - 1s 720ms/step
1/1 [==============================] - 1s 720ms/step
model prediction is  sad
```

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

```
In [ ]: 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 44ms/step
model prediction is  sad
C:\Users\91787\AppData\Roaming\Python\Python311\site-packages\keras_preprocessing
\image\utils.py:107: UserWarning: grayscale is deprecated. Please use color_mode
= "grayscale"
  warnings.warn('grayscale is deprecated. Please use '
```

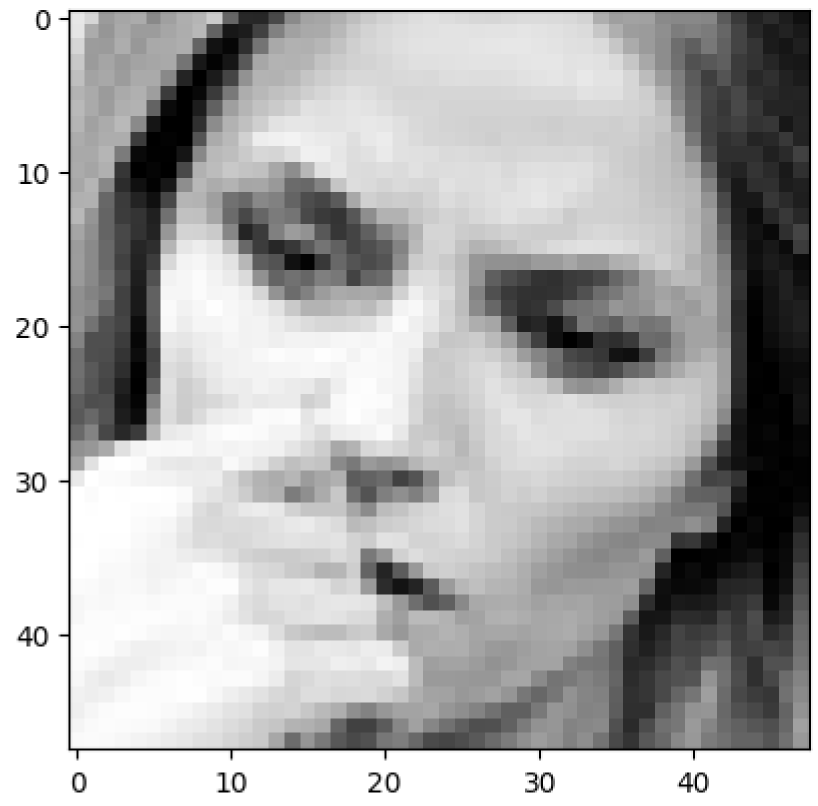
```
Out[ ]: <matplotlib.image.AxesImage at 0x238d15c1a90>
```



```
In [ ]: 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 39ms/step
model prediction is  sad
```

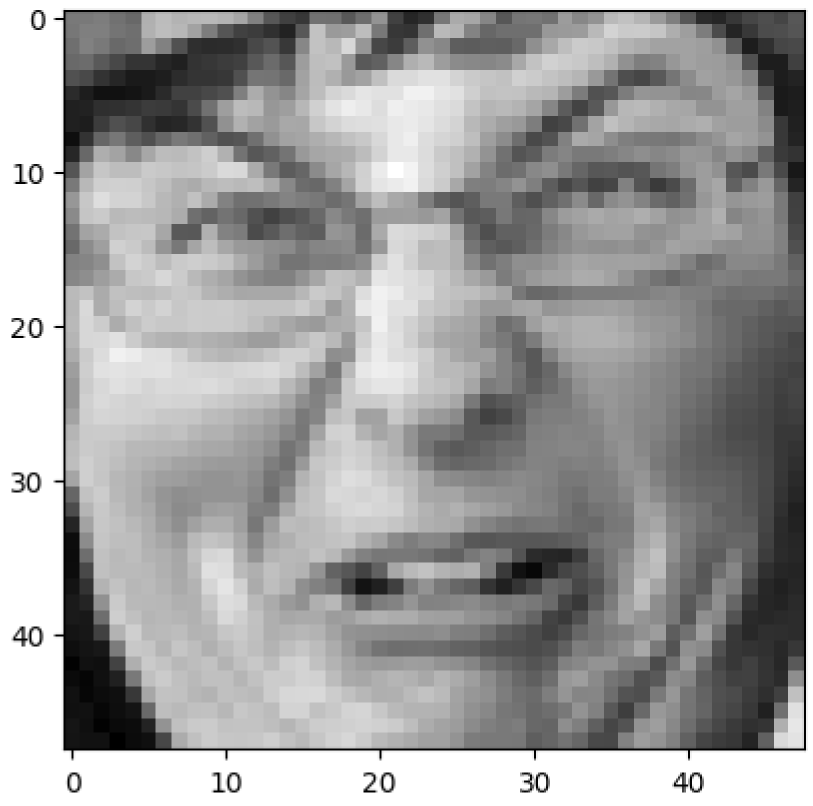
```
Out[ ]: <matplotlib.image.AxesImage at 0x238d28f3390>
```



```
In [ ]: 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 41ms/step
model prediction is  disgust
```

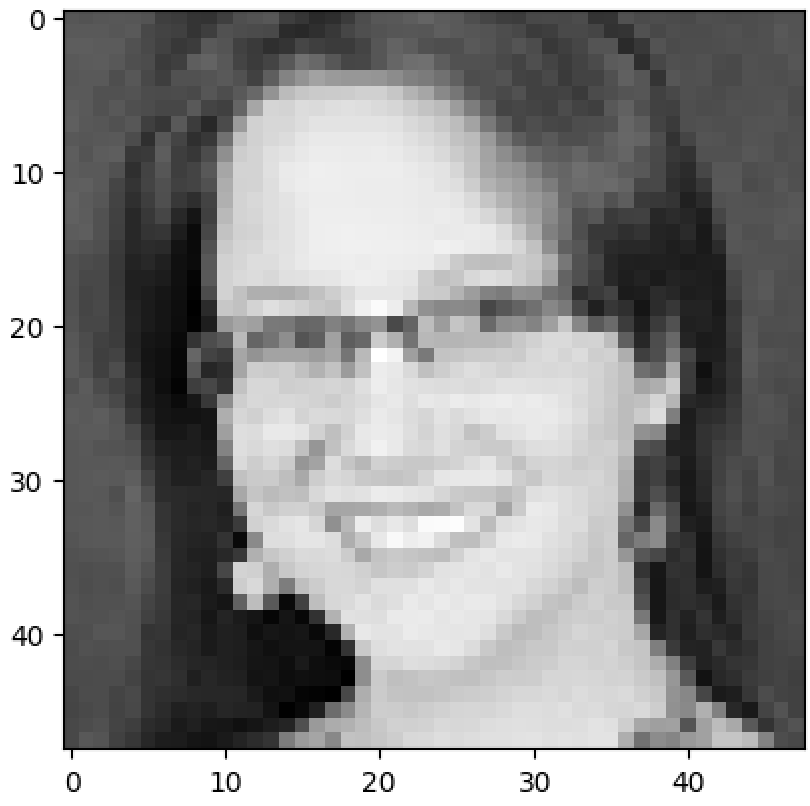
```
Out[ ]: <matplotlib.image.AxesImage at 0x238d2934390>
```



```
In [ ]: 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 38ms/step
model prediction is  happy
```

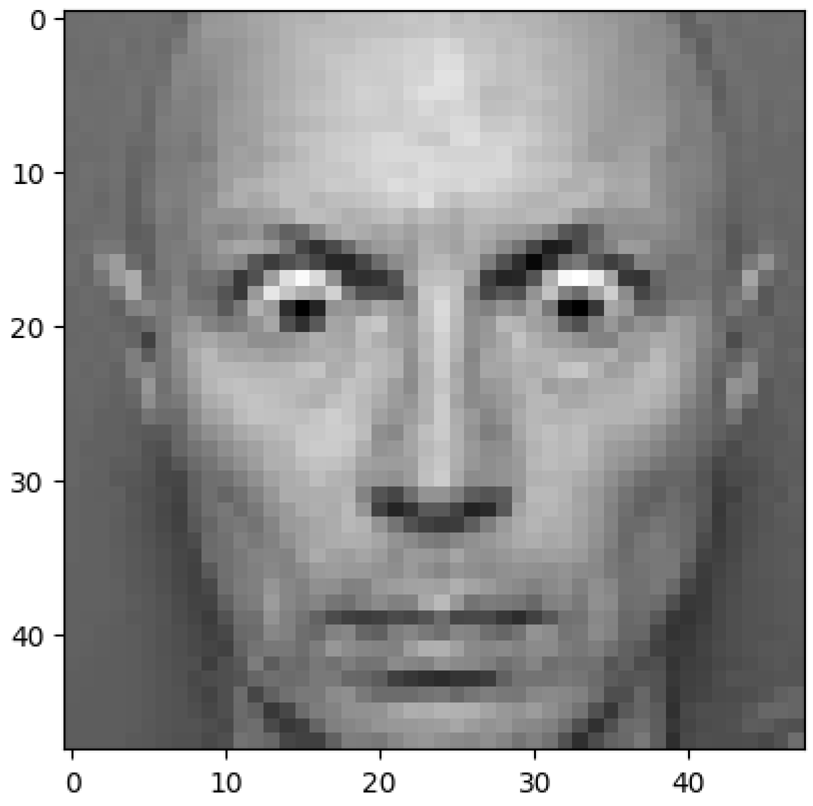
```
Out[ ]: <matplotlib.image.AxesImage at 0x238d2a31590>
```



In [ ]:
```
image = 'images/train/surprise/15.jpg'
print("original image is of surprise")
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 surprise
1/1 [==============================] - 0s 31ms/step
model prediction is  surprise
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x238d2a59790>
```


0
10
20
30
40
0
10
20
30
40