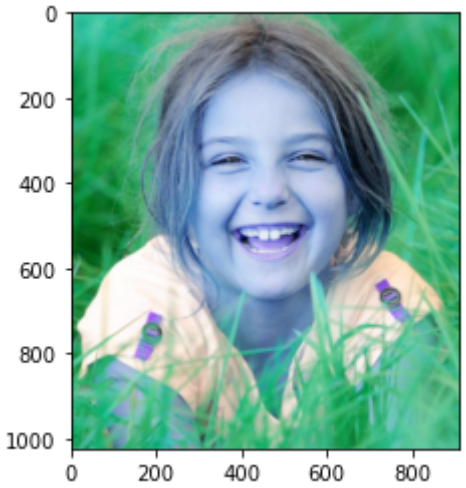
In [1]:
```
import cv2
from deepface import DeepFace
```

In [2]:
```
img = cv2.imread('girl.jpg')
```

In [3]:
```
import matplotlib.pyplot as plt
```

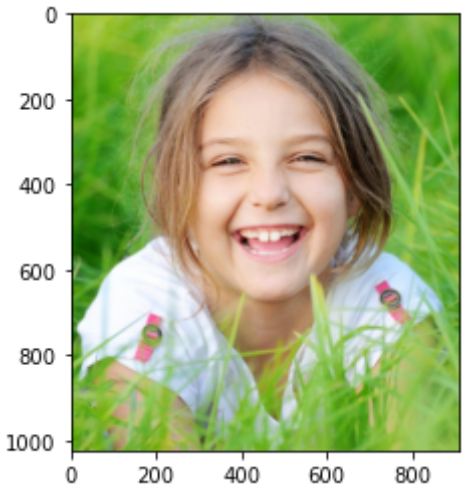
In [4]:
```
plt.imshow(img)
```

Out[4]: <matplotlib.image.AxesImage at 0x7fefd7eeaa00>



In [5]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[5]: <matplotlib.image.AxesImage at 0x7fefd81eddf0>



In [6]:
```
 predictions= DeepFace.analyze(img)
```

```
Action: race: 100%|██████████| 4/4 [00:04<00:00,  1.10s/it]
```

In [7]:
```
predictions
```

Out[7]:
```
{'emotion': {'angry': 1.888663696290904e-10,
  'disgust': 5.334442781523806e-21,
  'fear': 2.0174264108575915e-12,
  'happy': 99.90104438054033,
  'sad': 3.6566709186924285e-08,
  'surprise': 9.867774184941574e-05,
  'neutral': 0.09885077583333542},
 'dominant_emotion': 'happy',
 'age': 32.890034054852265,
 'gender': 'Woman',
 'race': {'asian': 0.0002970414016090217,
  'indian': 0.0002181492618547054,
  'black': 1.265066362776679e-06,
  'white': 98.71610999107361,
  'middle eastern': 0.42871590703725815,
  'latino hispanic': 0.8546503260731697},
 'dominant_race': 'white'}
```

In [15]:
```
type(predictions)
predictions['dominant_emotion']
```

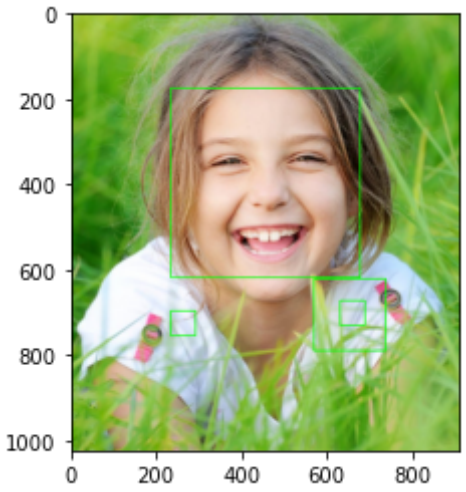
Out[15]:
```
'happy'
```

In [16]:
```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_fron
```

In [17]:
```
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces= faceCascade.detectMultiScale(gray,1.1,4)
for(x,y,w,h)in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

In [18]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

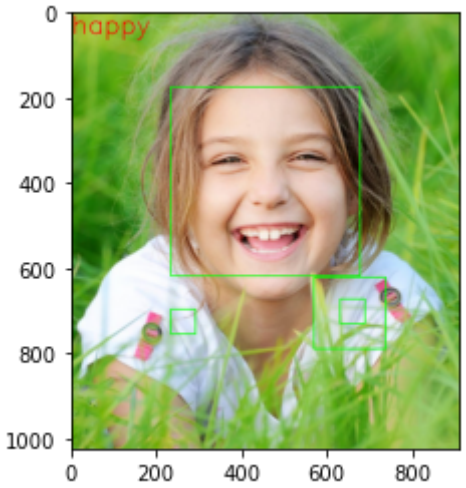
Out[18]:
```
<matplotlib.image.AxesImage at 0x7f9ee63846d0>
```



In [19]:
```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,
            predictions['dominant_emotion'],
             (0,50),
             font,2,
             (0,0,255),
             2,
             cv2.LINE_4);
```

In [20]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[20]: <matplotlib.image.AxesImage at 0x7f9ee4dc3a60>



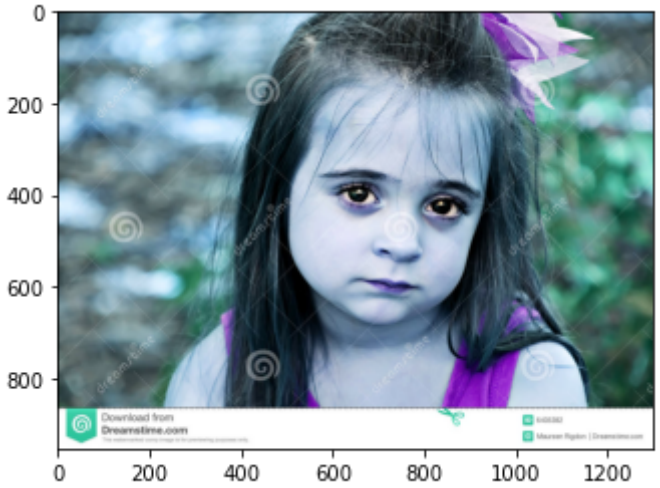
In [15]:
```
import cv2
from deepface import DeepFace
```

In [16]:
```
img = cv2.imread('kid2.jpg')
```

In [17]:
```
import matplotlib.pyplot as plt
```

In [18]:
```
plt.imshow(img)
```

Out[18]: <matplotlib.image.AxesImage at 0x7feb388a7a00>



In [19]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[19]:
```
<matplotlib.image.AxesImage at 0x7feb399eb040>
```

In [20]:
```
predictions= DeepFace.analyze(img)
```

```
Action: emotion:   0%|          | 0/4 [00:00<?, ?it/s]

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_pred
ict_function.<locals>.predict_function at 0x7feb387eeaf0> triggered tf.fu
nction retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) pa
ssing tensors with different shapes, (3) passing Python objects instead o
f tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: age:  25%|██        | 1/4 [00:00<00:02,  1.15it/s]

WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_pred
ict_function.<locals>.predict_function at 0x7fea7b537820> triggered tf.fu
nction retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) pa
ssing tensors with different shapes, (3) passing Python objects instead o
f tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: gender:  50%|█████     | 2/4 [00:02<00:02,  1.16s/it]

WARNING:tensorflow:7 out of the last 7 calls to <function Model.make_pred
ict_function.<locals>.predict_function at 0x7feb387ee430> triggered tf.fu
nction retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) pa
ssing tensors with different shapes, (3) passing Python objects instead o
f tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: race:  75%|███████   | 3/4 [00:03<00:01,  1.14s/it]

WARNING:tensorflow:8 out of the last 8 calls to <function Model.make_pred
ict_function.<locals>.predict_function at 0x7fea5a50ec10> triggered tf.fu
nction retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) pa
ssing tensors with different shapes, (3) passing Python objects instead o
f tensors. For (1), please define your @tf.function outside of the loop.
```

```
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.
Action: race: 100%|██████████| 4/4 [00:04<00:00,  1.15s/it]
```

In [21]:
```
predictions
```

Out[21]:
```
{'emotion': {'angry': 5.115531012415886,
  'disgust': 2.043011038121989e-08,
  'fear': 5.649503692984581,
  'happy': 4.907451334190682e-07,
  'sad': 59.22046899795532,
  'surprise': 7.55081316583528e-05,
  'neutral': 30.014419555664062},
 'dominant_emotion': 'sad',
 'age': 28.06506405501705,
 'gender': 'Woman',
 'race': {'asian': 0.023524889494844645,
  'indian': 0.033082113594054496,
  'black': 0.00034987021802325636,
  'white': 93.72318349618813,
  'middle eastern': 4.125730566723977,
  'latino hispanic': 2.094126617558397},
 'dominant_race': 'white'}
```

In [22]:
```
type(predictions)
predictions['dominant_emotion']
```

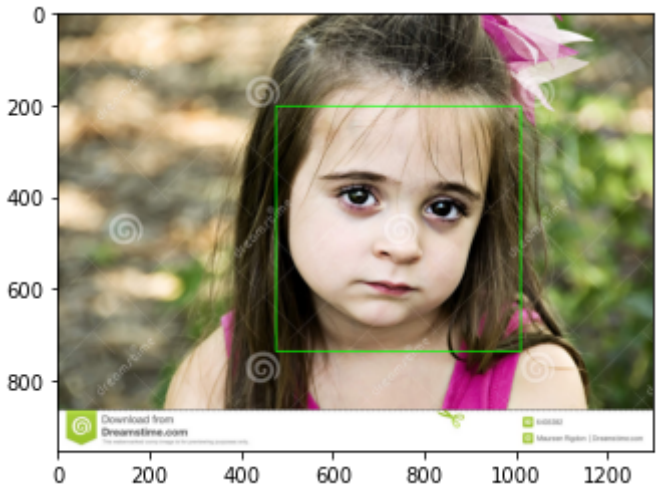
Out[22]:
```
'sad'
```

In [23]:
```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_fron
```

In [24]:
```
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces= faceCascade.detectMultiScale(gray,1.1,4)
for(x,y,w,h)in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

In [25]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

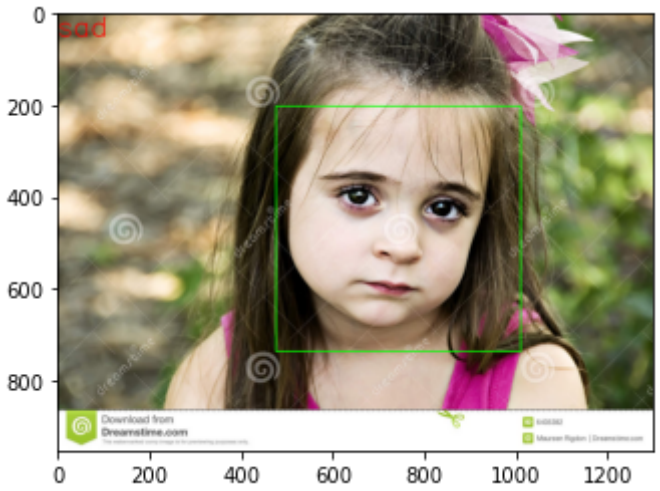
Out[25]: <matplotlib.image.AxesImage at 0x7fea5a5014f0>



In [26]:
```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,
            predictions['dominant_emotion'],
             (0,50),
             font,2,
             (0,0,255),
             2,
             cv2.LINE_4);
```

In [27]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[27]:
```
<matplotlib.figure.AxesImage at 0x7feb5752f9d0>
```



In [28]:
```
import cv2
from deepface import DeepFace
```

In [29]:
```
img = cv2.imread('kid.jpg')
```

In [30]:
```
import matplotlib.pyplot as plt
```

In [31]:
```
plt.imshow(img)
```

Out[31]:
```
<matplotlib.figure.AxesImage at 0x7fea58470550>
```

In [32]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[32]:
```
<matplotlib.figure.AxesImage at 0x7fea5a4c33d0>
```

In [33]:
```
predictions= DeepFace.analyze(img)
```

```
Action: emotion:   0%|          | 0/4 [00:00<?, ?it/s]

WARNING:tensorflow:9 out of the last 9 calls to <function Model.make_pred
ict_function.<locals>.predict_function at 0x7fe9b84bb0d0> triggered tf.fu
nction retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) pa
ssing tensors with different shapes, (3) passing Python objects instead o
f tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: age:  25%|██        | 1/4 [00:00<00:00,  3.19it/s]

WARNING:tensorflow:10 out of the last 10 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe9b84bbee0> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: gender:  50%|████      | 2/4 [00:01<00:01,  1.64it/s]

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe9985e9040> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: race:  75%|██████    | 3/4 [00:02<00:00,  1.28it/s]

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe99849f0d0> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
```

```
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: race: 100%|██████████| 4/4 [00:03<00:00,  1.17it/s]
```

In [34]:
```
predictions
```

Out[34]:
```
{'emotion': {'angry': 83.57749581336975,
  'disgust': 8.146710389622669e-09,
  'fear': 0.3148074960336089,
  'happy': 0.004117118805879727,
  'sad': 6.1553314328193665,
  'surprise': 0.0003165270300087286,
  'neutral': 9.94793176651001},
 'dominant_emotion': 'angry',
 'age': 23.020086226224556,
 'gender': 'Man',
 'race': {'asian': 11.645475775003433,
  'indian': 5.2439771592617035,
  'black': 1.4845226891338825,
  'white': 37.31638789176941,
  'middle eastern': 16.68788194656372,
  'latino hispanic': 27.621757984161377},
 'dominant_race': 'white'}
```

In [35]:
```
type(predictions)
predictions['dominant_emotion']
```

Out[35]:
```
'angry'
```

In [36]:
```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_fron
```

In [37]:
```
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces= faceCascade.detectMultiScale(gray,1.1,4)
for(x,y,w,h)in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

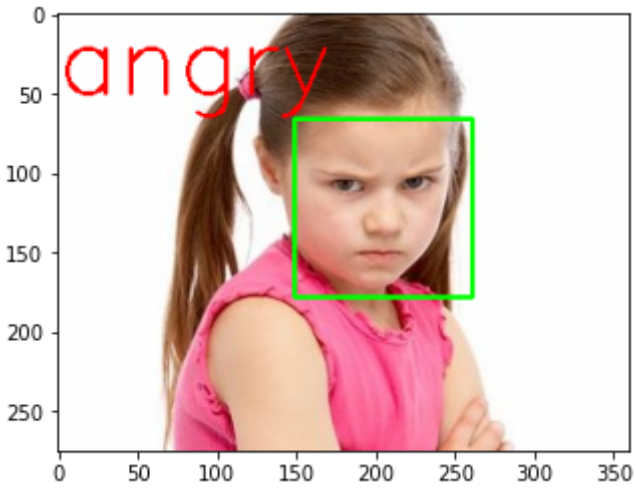
In [38]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[38]:
```
<matplotlib.image.AxesImage at 0x7fe99875d040>
```

In [39]:
```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,
            predictions['dominant_emotion'],
             (0,50),
             font,2,
             (0,0,255),
             2,
             cv2.LINE_4);
```

In [40]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[40]:
```
<matplotlib.image.AxesImage at 0x7fe99a02b460>
```



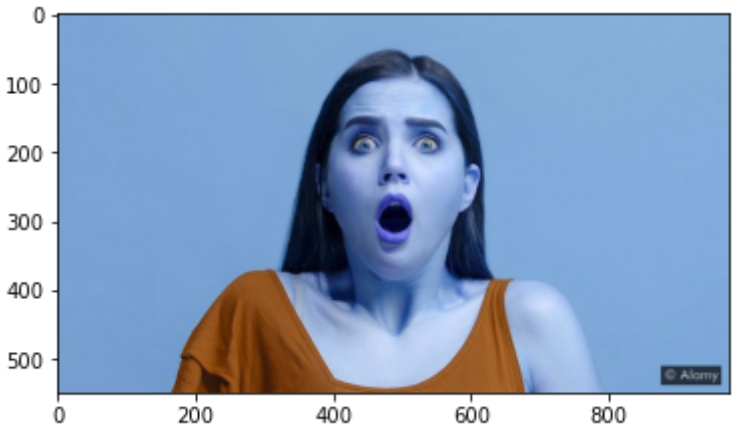
In [41]:
```
import cv2
from deepface import DeepFace
```

In [42]:
```
img = cv2.imread('women1.jpg')
```

In [43]:
```
import matplotlib.pyplot as plt
```

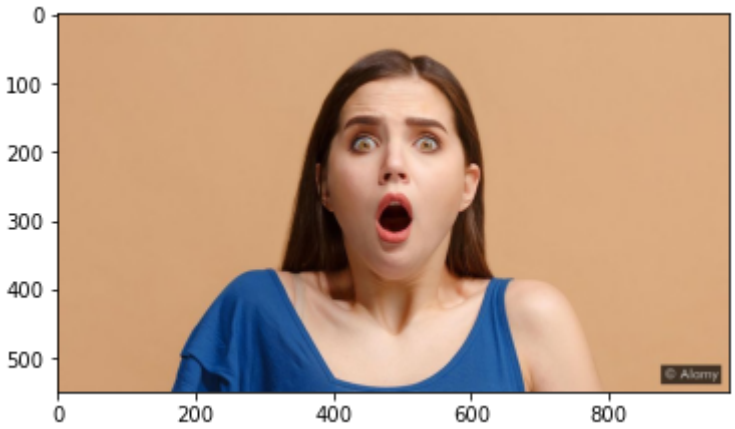
In [44]:
```
plt.imshow(img)
```

Out[44]:
```
<matplotlib.image.AxesImage at 0x7fea59988460>
```



In [45]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[45]:
```
<matplotlib.image.AxesImage at 0x7fe99888b580>
```



In [46]:
```
predictions= DeepFace.analyze(img)
```

```
Action: emotion:   0%|          | 0/4 [00:00<?, ?it/s]

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe8fb51f1f0> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: age:  25%|██        | 1/4 [00:00<00:01,  2.82it/s]

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe8fb51fe50> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: gender:  50%|████      | 2/4 [00:01<00:01,  1.80it/s]

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe8d913c310> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: race:  75%|██████    | 3/4 [00:02<00:00,  1.42it/s]

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_pr
edict_function.<locals>.predict_function at 0x7fe8d84711f0> triggered tf.
function retracing. Tracing is expensive and the excessive number of trac
ings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead
of tensors. For (1), please define your @tf.function outside of the loop.
```

```
For (2), @tf.function has experimental_relax_shapes=True option that rela
xes argument shapes that can avoid unnecessary retracing. For (3), please
refer to https://www.tensorflow.org/tutorials/customization/performance#p
ython_or_tensor_args (https://www.tensorflow.org/tutorials/customization/
performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/funct
ion) for  more details.

Action: race: 100%|██████████| 4/4 [00:03<00:00,  1.27it/s]
```

In [47]:
```
predictions
```

Out[47]:
```
{'emotion': {'angry': 7.22112439288658e-09,
  'disgust': 8.15013020002507e-20,
  'fear': 2.8499687232397264e-05,
  'happy': 3.577984852842775e-08,
  'sad': 1.2788484379404113e-13,
  'surprise': 99.99997615814209,
  'neutral': 5.966272812585993e-18},
 'dominant_emotion': 'surprise',
 'age': 32.31139190871506,
 'gender': 'Woman',
 'race': {'asian': 0.3982672467827797,
  'indian': 0.7600479759275913,
  'black': 0.058300443924963474,
  'white': 67.213374376297,
  'middle eastern': 12.365676462650299,
  'latino hispanic': 19.204336404800415},
 'dominant_race': 'white'}
```

In [48]:
```
type(predictions)
predictions['dominant_emotion']
```

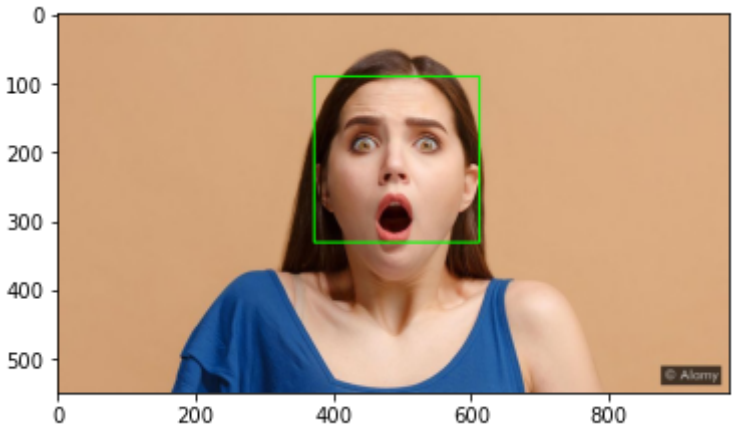
Out[48]:
```
'surprise'
```

In [49]:
```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_fron
```

In [50]:
```
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces= faceCascade.detectMultiScale(gray,1.1,4)
for(x,y,w,h)in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

In [51]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[51]:
```
<matplotlib.image.AxesImage at 0x7fe8d85021f0>
```



In [52]:
```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,
            predictions['dominant_emotion'],
             (0,50),
             font,2,
             (0,0,255),
             2,
             cv2.LINE_4);
```

In [53]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Out[53]:
```
<matplotlib.image.AxesImage at 0x7fe999fc9280>
```

In [*]:
```
import cv2
from deepface import DeepFace
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_fron
cap = cv2.VideoCapture(1)
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raiseIOError("Cannot open webcam")
while True:
    ret,frame=cap.read()
    result=DeepFace.analyze(frame,actions=['emotion'])
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces=faceCascade.detectMultiScale(gray,1.1,4)
    for(x,y,w,h)in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(frame,
           result['dominant_emotion'],
            (0,50),
            font,3,
            (0,0,255),

            2,
            cv2.LINE_4);
    cv2.imshow('Demo video',frame)
    if cv2.waitKey(2) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

In [ ]: