*Face Analyzer  
Predicting Emotions using FER and Emoji*

Table of Contents

[Overview 2](#_Toc80108622)

[Scope 2](#_Toc80108623)

[Pre-requisites 2](#_Toc80108624)

[Setup 3](#_Toc80108625)

[Conclusion 8](#_Toc80108626)

Overview

Images are powerful tools for conveying rich semantics and inducing various emotions to viewers. Through facial expressions people can express their feelings and communicate with other people. Such affective information can be used by search engines and social networks for better understanding the user’s preferences.

Scope

The target of this tutorial is to develop a lightweight command line based utility, through Python based modules to predict the personalized emotional state of a person and that by decoding his facial expressions mainly the most important ones listed hereafter: (Happy, Sad, Fear, Anger, Surprise, Neutral and Disgust).  
  
If this tutorial intrigues you, then grab its code from the following GitHub repository: “<https://github.com/bassemmarji/FaceAnalyzer/> ”.

# Pre-requisites

The following components come into play:

* **FER**: developed by Justing Shenk this library recognizes Facial Expressions using a deep neural network while leveraging Tensorflow and Keras libraries. It depends on OpenCV 3.2+, Tensorflow 1.7+ and Python 3.6+.
* **Emoji**: This library makes it easy to integrate emojis with Python programs.
* **OpenCV**: is an open-source library for computer vision, machine learning and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java and it is used for all sorts of image and video analysis like facial detection and recognition, photo editing, optical character recognition and a whole heap more.
* **Dlib**: is principally a C++ library however you can use a number of its functions from Python applications.
* **filetype**: is a small and dependency free Python package to infer file and MIME types.
* **imutils**: encompasses a series of functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges.
* **Pillow**: This library (Python Imaging Library) is best suited for image archival and batch processing applications. You can do almost anything on digital images using pillow module.

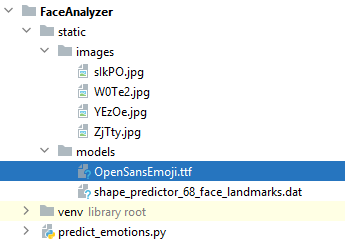
# Setup

To setup the environment, you need python3 installed on your system. It is highly recommended to setup a virtual environment which will host the needed libraries.

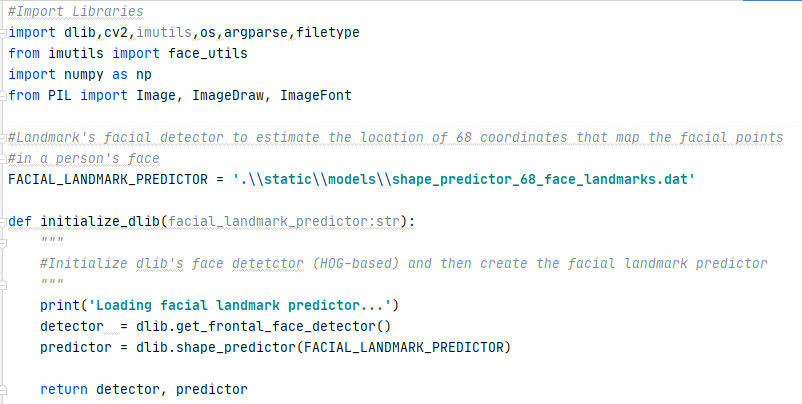
1. Create a virtual environment and activate it.
2. Create a file named requirements.txt and add the following lines to it.

|  |
| --- |
| requirements.txt |
| fer==21.0.3 emoji==1.4.2  filetype==1.0.7 Pillow==8.0.1 |

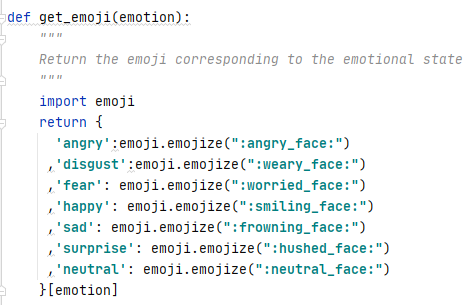
1. Now, let’s install the required libraries to the project.  
   pip install –r requirements.txt
2. Create a folder for our project called “FaceAnalyzer”.

At the end, our folder structure will look like the following:  
  
  
  
**NB:**   
  
For the purpose of this article we will use the “shape\_predictor\_68\_face\_landmarks.dat”; A dlib pre-trained model to estimate the location of 68 coordinates (x,y) that map the facial points on a person’s face. The model details are available in the below link   
<https://github.com/davisking/dlib-models>.

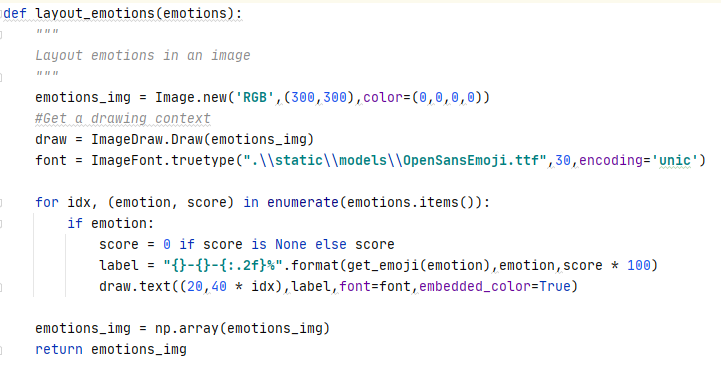
Let’s move into coding:  
 *#predict\_emotions.py*



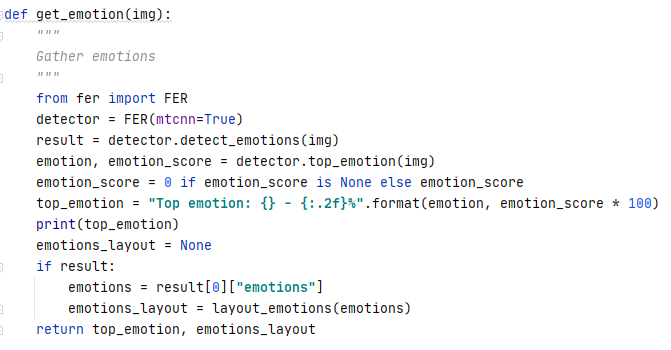
* This function initializes the dlib using the pre-trained model and returns:
  + detector: used for detecting the face in an image.
  + predictor: shape or landmark predictor used to predict the coordinates of a given shape. The facial landmark predictor is used to localize individual facial structures.



* This function returns an emoji corresponding to the emotional state reflected by the facial expressions.

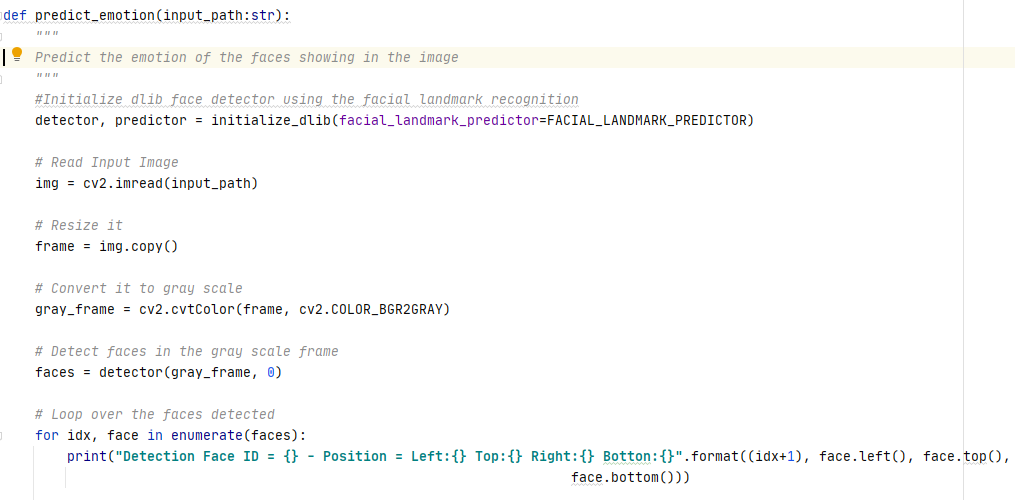


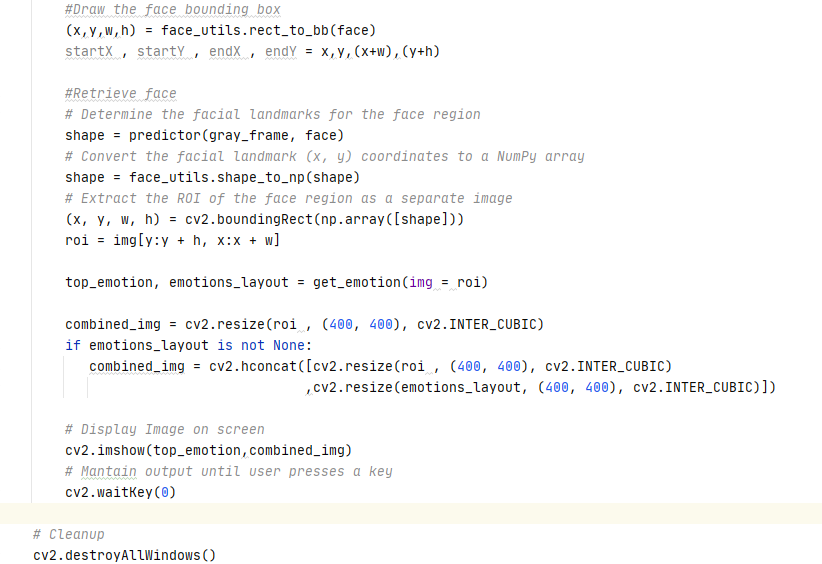
* This function builds an image showing an emoji representing each of the 7 emotional states previously laid out and a score respective for each emotional state.



* This function runs a detector which returns a list containing an ordered dictionary of bounding box notations where a face is detected and all the 7 emotions in decimal values from 0 to 1.  
  MTCNN stands for multi cascade convolutional network, it is an advanced technique for detecting faces.  
  The detect\_emotions function is called to classify the emotion into (happy, sad, disgust, anger, fear, neutral) with corresponding values for each.

Once executed, this function displays on the console the emotion having the highest score using the Top\_emotion function.

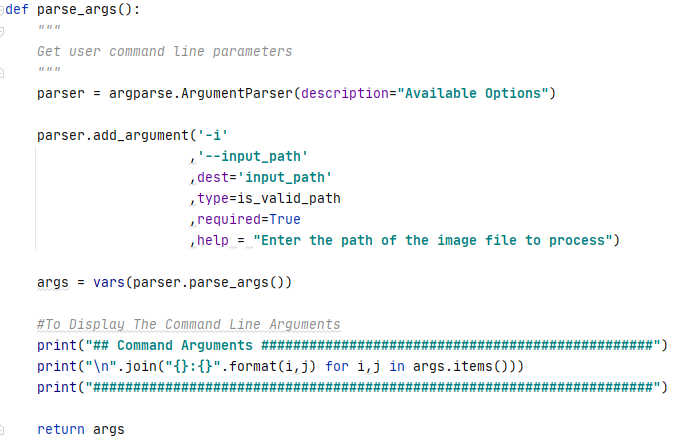




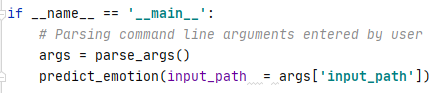
* This function constitutes the core of our program and performs the following:
  + Initialize the dlib face detector.
  + Reads the input image.
  + Converts the image into gray scale.
  + Detects faces in the grayed image.
  + Iterates throughout the faces detected and displays the positions of the faces detected.
  + Discerns the top emotions and builds an emotions layout.
  + Displays an image including the original image and the emotions scores.



* This function validates a path inputted as a parameter and ensures that it is a file path also it ascertains that the type of the file chosen is an image.



* This function defines and sets the appropriate constraints for the command line argument to be specified by the user when running this utility.
  + input\_path: A required parameter containing the path of the image file to process associated with the predefined function “is\_valid\_path”.

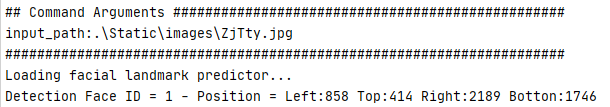


* The above represents the main function of our program.

Let’s test our program:  
  
Kindly proceed as per the following steps:

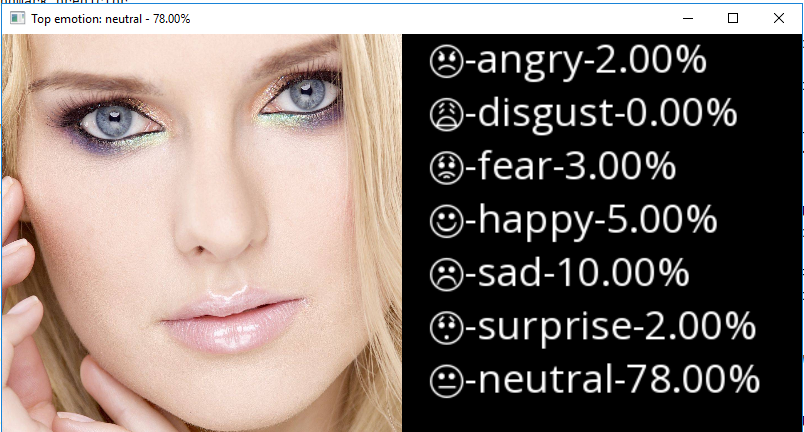
1. Open up a terminal window and type the following in it:

**predict\_emotions –i ".\static\images\ZjTty.jpg"**

The following summary will be displayed in the terminal:  
  
  
.

.





Conclusion  
  
The human face has 43 muscles, which can stretch, lift and contort it into dozens of expressions. This vast range of movements makes extremely difficult to interpret facial expressions.  
Hope you enjoyed this article.

|  |  |
| --- | --- |
|  | ***Bassem Marji*** *is a project implementation manager at BLOM Bank with a proven track record of success.  He managed the implementation of over 50 projects and propelled the digital transformation of mission critical applications. He spends his free time discovering the latest technology trends in the IT field.* |