Classification of the emotional state of a human by extracting facial features

18th November 2018

OVERVIEW

Our project aims to classify the emotional state of a person, given his/her image, in the following 7 standard categories: anger, disgust, joy, neutral, sadness, surprise, and fear. The emotional state of the person is purely on the basis of the facial expressions portrayed in the image. This finds a lot of application in numerous domains like Transportation (to avoid rage driving), help machines understand the emotional state of the person to provide appropriate recommendations (soothing music when one is sad), Marketing (impact of ads on the person), etc.

GOALS

1. To gather knowledge on the process of picking datasets best suited for our problem.
2. To learn the art of organizing the dataset.
3. To get hands on experience on extracting features from an image, given the raw dataset.
4. Utilize techniques of EDA to make sense of the features extracted.
5. Establish methods for tuning hyperparameters for a model.
6. Learn to express our model through an appropriate evaluation metric.

APPROACH

Collection of Dataset

After exploring several online datasets, we went ahead with FacesDB, an open dataset consisting of six universal expressions (happiness, sadness, surprise, anger, disgust, and fear) and neutral between human races. The dataset had a rich variety of images of people in terms of gender, age-group, etc. to make sure there's not any kind of bias while collecting the data. The one disadvantage we expect the dataset to have is the existence of lab-like conditions (ex: static illumination) and extreme animated emotions while collecting the data.

Google images however, have a lot of noise and thus not suitable for extraction in the given timeframe.

Data Organization

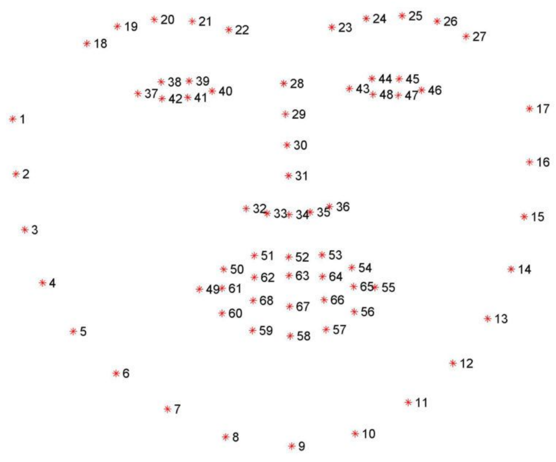
We organised the raw dataset in a way to make our job easier while performing feature engineering. Writing scripts to automate our tasks proved to be very efficient and time saving.

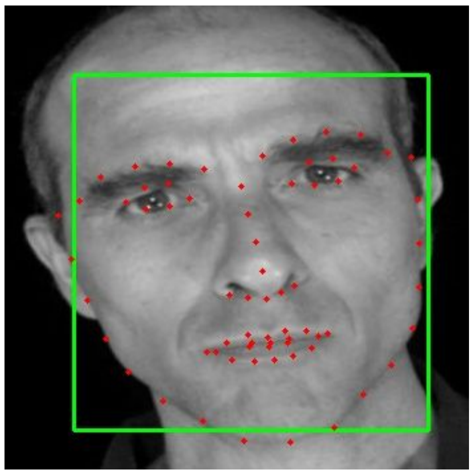
Theoretical Basis

Considering this as one of the most critical part, we spent quite a few hours reading research papers to answer questions like how faces are detected, what kind of features can be used for detecting faces, what kind of features can be used to express emotions, etc. Considering the time constraint, we glanced through most of the contents of the paper and decided to use pre-trained models for face detection, as that is not our primary goal. The publications we read can be found in the “References” section.

Face detection and extracting crucial facial landmarks

We experimented with multiple pre-trained Haar Cascade Classifiers and Histogram of Oriented Gradients (HOG) Classifiers to get the bounding box representing the face of the person. This bounding box is important for Dlib, a C++ library popularly used for extracting 68 facial landmarks on the face of the person. With this we were finally set to do some feature engineering.



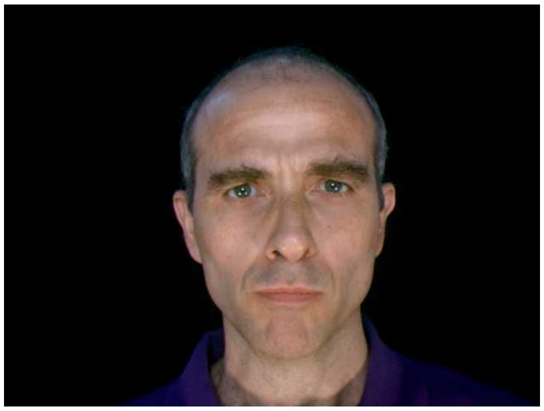


Data Preprocessing and Data Augmentation

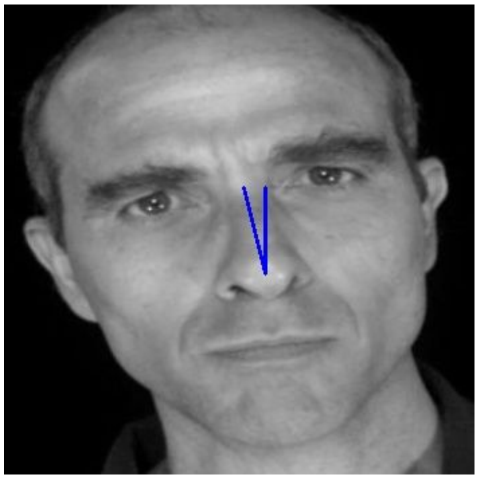
As the background is useless for the problem we're solving, we use face detection to get our region of interest (ROI) and crop the image with only the ROI. With computational cost at the back of our mind, we rescaled the images and converted them into single channels (grayscale).

Although FacesDB had vast collection of categorical attributes of emotions, the size of the dataset was very small to train a robust model: 7 * 38 (The number of emotions that we had worked on was 7, and there were 38 subjects in total).
The better choice was to increase the number of datapoints (images in this case) by performing 'Data Augmentation' through techniques like flipping and rotating of the existing images. After performing data augmentation, we had about 1200 images in our dataset.

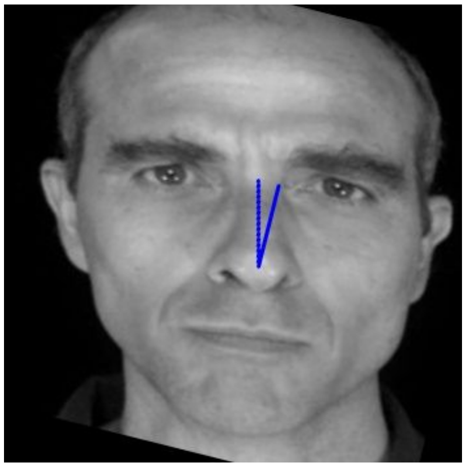
Original:



Before rotating (after cropping, grayscaling, augmenting):



After rotating:



Note: The angle to rotate is the angle between those line segments potrayed.

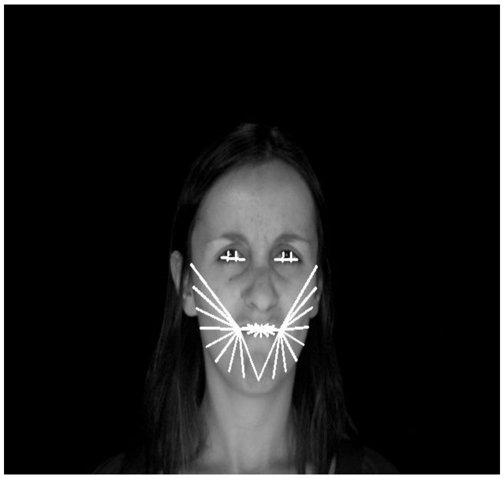
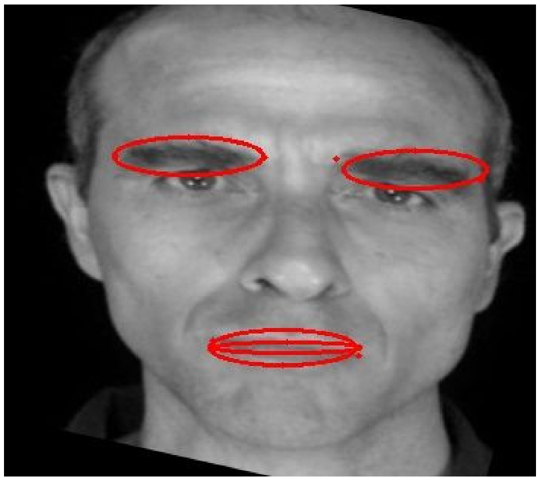
Feature Engineering

After gathering a bit of domain knowledge (blogs and the mentioned research papers), we decided to extract geometrical features as: Linear Features, Trigonometric Features and Eccentricity Features.

For example, in linear features, we computed distances from specific points which we found appropriate in classifying different emotions, say, distance between the eyes and the eyebrows, distance between the upper lip and the lower lip, etc.

In trigonometric features, we computed cosines of specific vectors connecting two landmarks.

In eccentricity features, we computed the length of the semi-major axis, length of the semi-minor axis and the eccentricity of the ellipse surrounding regions like the eyes (left and right), eyebrows (left and right) and the mouth region (upper and lower).

Visualizations and EDA:

To avoid cramming, we suggest you to refer the Jupyter Notebook titled “EDA”.

Training:

First, we started training our models on the non-augmented dataset (approx. 250 images). Series of models chosen were:

1. Logistic Regression
2. Support Vector Machines (with ‘linear’ and ‘rbf’ kernel)
3. Gaussian Naive Bayes

4. Decision Tree
5. Bagging Classifier
6. Random Forest
7. XGBoost
8. Gradient Boosting
9. AdaBoost
10. Voting Classifier (Logistic, Decision Tree and Gaussian Naive Bayes)

The accuracies we attained from this dataset came out to be too low, as seen in the screenshots stored in the directory **“non_aug_model_acc”.**

So, we then went through the same process but now with the augmented dataset (approx. 1150 images). The accuracies obtained can be found in the directory titled **“aug_model_acc”**.

Hyperparameter Tuning:

Picking Logistic Regression as one of the best performing model even with the default parameters, we, then, planned to extend it’s performance by tuning the hyperparameters. Making use of GridSearchCv and StratifiedKFold, we were able to realise this, giving us the best hyperparameters as:

“C“ = **3**

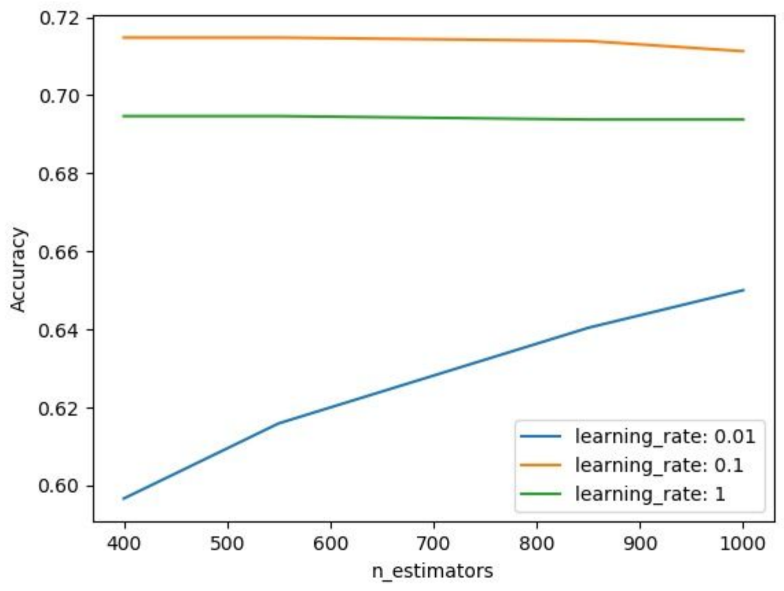
Note: Increasing C from ‘1’ suggests that my previous model was underfitting.

Picking XGBoost as the other best performing model, we tuned it’s hyperparameters too:

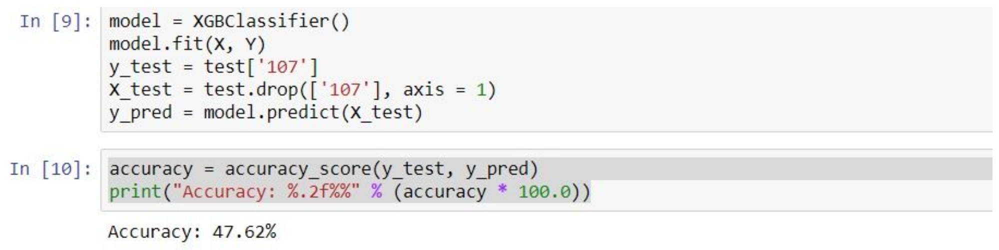
“Learning_rate” = **0.1**

“N_estimators” = **400**

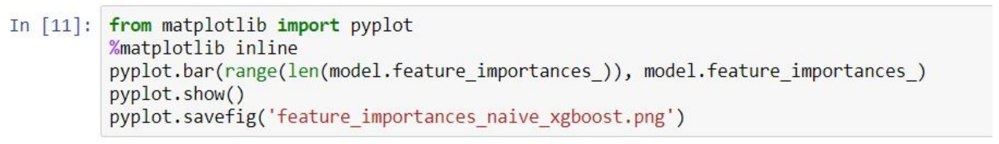
Note: Due to time constraints and the immense need of computational power, we did not tune other hyperparameters.



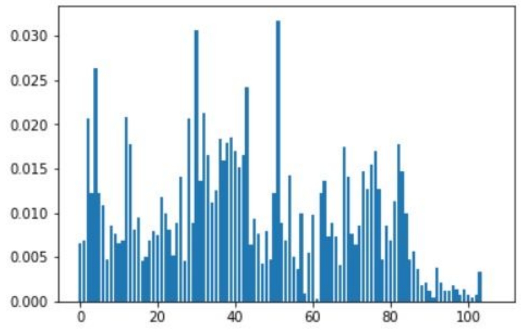
This graph is on the training data.

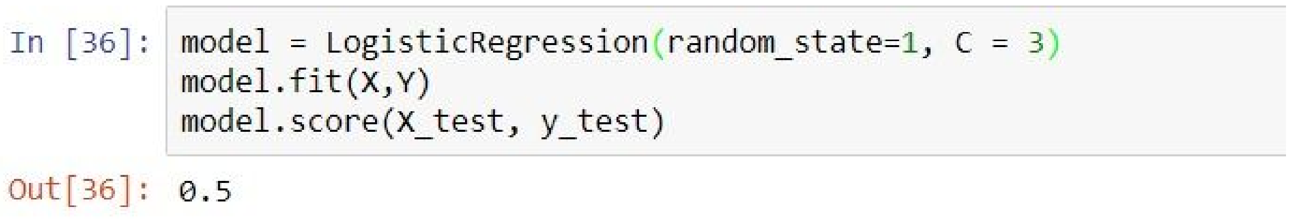


```
In [9]:  model = XGBClassifier()
         model.fit(X, Y)
         y_test = test['107']
         X_test = test.drop(['107'], axis = 1)
         y_pred = model.predict(X_test)

In [10]: accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy: %.2f%%" % (accuracy * 100.0))

         Accuracy: 47.62%
```



```
In [11]: from matplotlib import pyplot
         %matplotlib inline
         pyplot.bar(range(len(model.feature_importances_)), model.feature_importances_)
         pyplot.show()
         pyplot.savefig('feature_importances_naive_xgboost.png')
```





```
In [36]:  model = LogisticRegression(random_state=1, C = 3)
          model.fit(X,Y)
          model.score(X_test, y_test)

Out[36]:  0.5
```

Note: As seen above, results of our final models are displayed. Both models' pickled version has been provided.

Future Work:

1. Extract more features and use feature selection techniques to get the best ones.
2. Get a more diverse dataset (people from different nationalities/races/sex, images from different angles, etc.)
3. Perform deeper hyperparameter tuning.
4. Use appropriate evaluation metrics to better evaluate the trained model.

References

Dataset:

http://app.visgraf.impa.br/database/faces

Research Papers:

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

https://www.semanticscholar.org/paper/One-millisecond-face-alignment-with-an-ensemble-of-Kazemi-Sullivan/1824b1ccace464ba275ccc86619feaa89018c0ad

https://ieeexplore.ieee.org/document/7294832

Haar Cascades Classifiers:

https://github.com/opencv/opencv/tree/master/data/haarcascades

Libraries:

http://dlib.net/

https://scikit-learn.org/stable/documentation.html

https://keras.io/preprocessing/image/