MSAI-349, Fall 2022
Final Project: Preliminary Results

Group 2

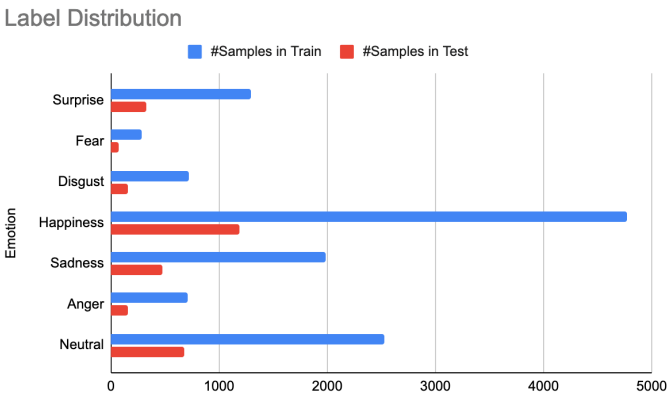
1. Overview

We present a machine learning system with the ability to detect human emotion through facial expressions. The task we are performing is to recognize and classify human emotions such as Happiness, Sadness, Surprise, Fear, Disgust and Anger from images from the RAF Database[1].

2. Data

The source of our data is from the RAF (Real-world Affective Faces) Database[1]. This database has around 15,000 diverse, large scale facial expression images. Each image has been labeled with 5 manual annotations and 37 automatic annotations, besides demographic tags. The subjects are from a diverse background, meaning with all kinds of ethnicity, age and gender. The image conditions vary as well with different lighting, head angles, occlusions and special filters and effects.

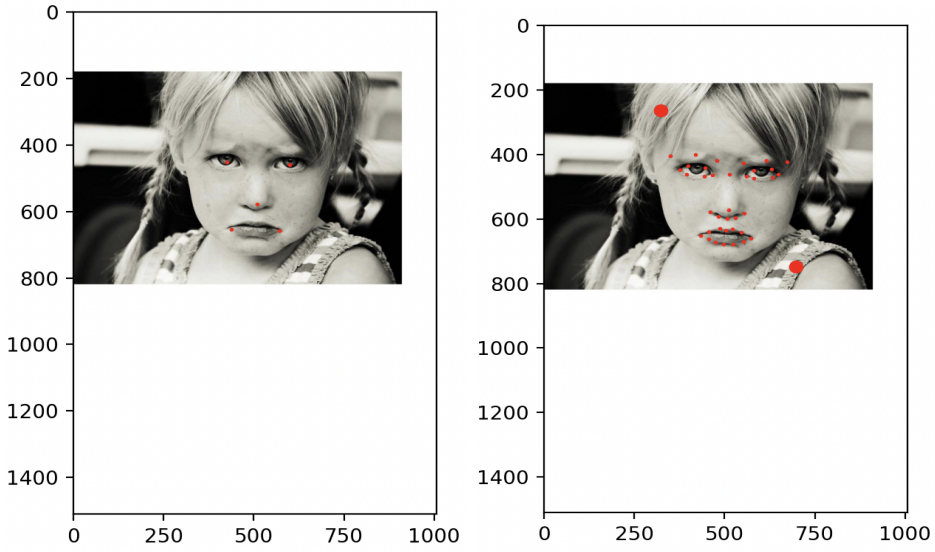
The number and distribution of classes are as follows:



Label	Emotion	#Samples in Train	#Samples in Test
0	Surprise	1290	329
1	Fear	281	74
2	Disgust	717	160
3	Happiness	4772	1185
4	Sadness	1982	478
5	Anger	705	162
6	Neutral	2524	680

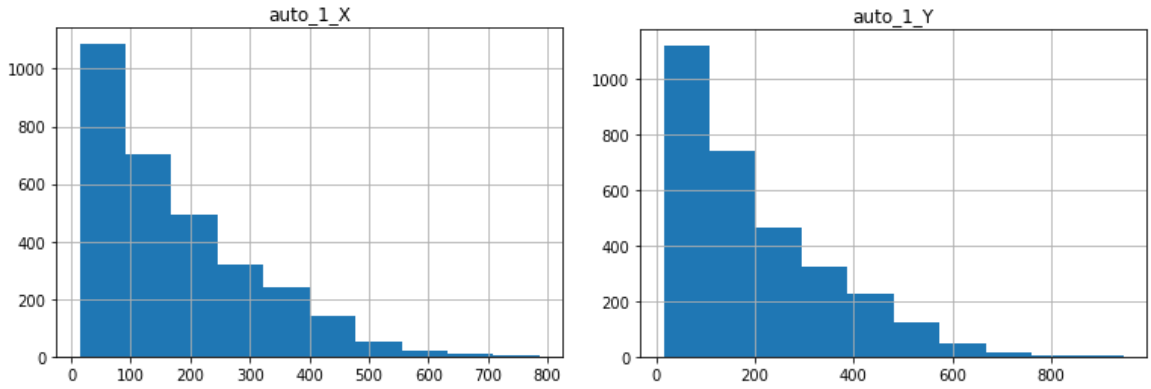
The distribution is quite skewed and this fact significantly affects classification performance.

Below we have visualized the annotations present in the dataset. On the left are the 5 manual annotations, and on the right are the 37 automatic annotations. The two large dots are the top left corner and the bottom right corner of the bounding box.



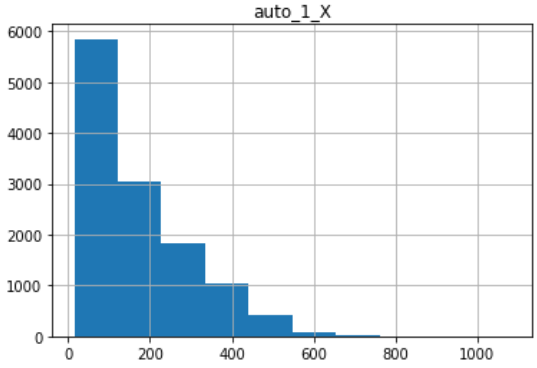
Below are the histograms of the distribution of the first automatically generated landmark's x and y coordinates. All the landmarks display a similarly clustered distribution, so we have not presented them in the report explicitly.

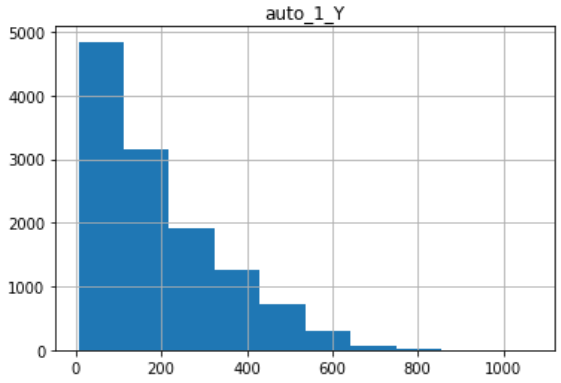
Test Set:



index	auto_1_x	auto_1_y
mean	174.245925684485	200.8368970013038
std	129.94727251917033	150.50781519046464
min	14.0	15.0
max	787.0	945.9

Train Set:





index	auto_1_x	auto_1_y
mean	169.738750	195.341651
std	126.907844	146.743158
min	15.0	6.0
max	1080.5	1067.4

3. Models

Process Flow

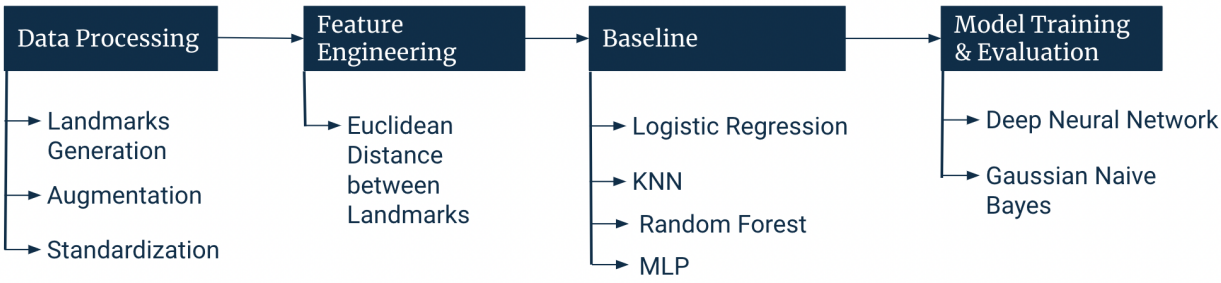


Fig. Pipeline diagram

Data Preprocessing

First, we tried to generate landmarks from the images using MediaPipe's FaceMesh generator. It produced normalized x, y, and z (depth from the camera) coordinates for 478 different landmarks on each image and it worked pretty well with most of the images. Here are some sample output landmarks plotted on the respective images:

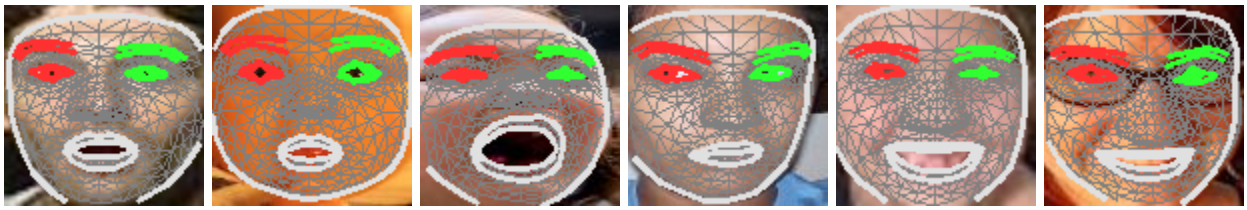


Fig. MediaPipe Facemesh Output

However, we've noticed that MediaPipe wasn't able to detect landmarks for more than 10% of our dataset, primarily due to poor picture quality or bad face alignments. Therefore, we decided to use the landmarks provided by the creators of RAF-DB[1].

The dataset that we were working with had some automatically generated landmarks as well manually selected landmarks[2]. We combined both to augment our feature space. Next, we standardized the landmark coordinates to get the relative values irrespective of the location of the faces in the picture.

Feature Engineering

We calculated pairwise Euclidean distances between all the landmarks as a set of additional features to our classification models.

Baseline

We used sklearn's out-of-the-box models - to set a baseline for our classification task. We trained logistic regression, KNN, random forest, and multi-layer perceptron (MLP) models on our dataset. The results of these models are summarized in the results section below.

Model Training & Evaluation

a. Deep Neural Network

We implemented a deep neural network with three hidden layers having 8, 32, and 8 neurons each, respectively. We used Cross Entropy[3] as the loss function and adaptive movement estimation (Adam)[4] as the optimizer to handle noisy data and somewhat sparse gradients. After a randomized search & tuning, we selected the following additional hyper-parameters for our model:

Hyper-Parameter	**Value**
Learning rate	2e-4
Weight decay	0.0001
Activation Functions	Sigmoid (hidden layers)
Number of epochs	1500

b. Gaussian Naive Bayes

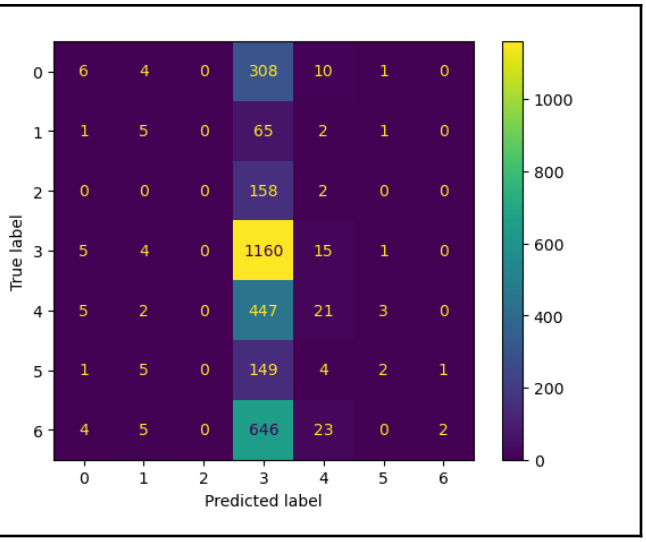
Since we were working with the continuous data (in the form of landmark coordinates and the Euclidean distances between the landmarks), we implemented a Gaussian Naive Bayes model to fit on our data. Taking into consideration the possibility of getting sparse features, we replaced the standard deviations of 0s with 0.01s to avoid "divide by zero" errors in the Gaussian distribution formula.

4. Results

	Model	Accuracy	F1-Score (Macro)	F1-Score (Weighted)
Baseline Models from Scikit-Learn	**Logistic Regression**	0.65	0.45	0.62
	K-Nearest Neighbors	0.51	0.35	0.49
	Random Forest	0.62	0.42	0.58
	Multilayer Perceptron	**0.71**	**0.56**	**0.69**
	Gaussian Naive Bayes	0.36	0.27	0.37
Models Implemented by Us	**Neural Network**	**0.68**	**0.46**	**0.65**
	Naive Bayes	0.39	0.29	0.39

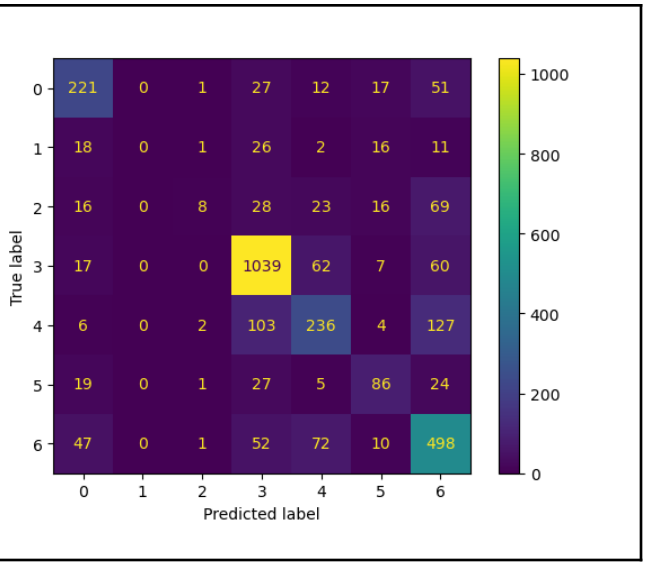
Naive Bayes

class	precision	recall	f1-score	support
0: Surprise	0.27	0.02	0.03	329
1: Fear	0.2	0.07	0.1	74
2: Disgust	**0**	**0**	**0**	160
3: Happiness	0.4	**0.98**	0.56	1185
4: Sadness	0.27	0.04	0.08	478
5: Anger	0.25	**0.01**	0.02	162
6: Neutral	0.67	**0**	**0.01**	680



Neural Network

class	precision	recall	f1-score	support
0: Surprise	0.64	0.67	0.66	329
1: Fear	**0.00**	**0.00**	**0.00**	74
2: Disgust	0.57	0.05	0.09	160
3: Happiness	**0.80**	**0.88**	**0.84**	1185
4: Sadness	0.57	0.49	0.53	478
5: Anger	0.55	0.53	0.54	162
6: Neutral	0.59	0.73	0.66	680



5. Analysis

Naive Bayes Model

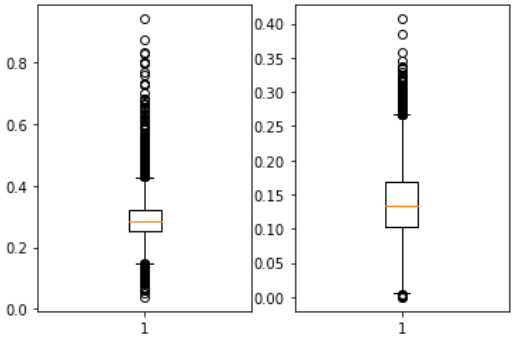
The Naive Bayes model gave **accuracy of 0.39** and **F1 score of 0.29** on the test data.
(**F1-score of 0.39** if label imbalance is accounted for).

We were able to get the least misclassification among all the labels for Class-3 (Happiness), with recall of 0.98. Class-6 (Neutral) and Class-2 (Disgust) were misclassified by the model almost 100% of the time, giving f1-score of 0.01 and 0 respectively.
Misclassifications for all the labels are skewed towards Class-3 (Happiness), bringing the f1-score down to 0.56, despite having a very high recall.

Class-3 (Happiness) dominated our model and Class-1 & Class-2 were severely underrepresented. Upon looking at the class distribution in the training dataset:

1. We have almost **17 times** as many samples in our dataset with Class-3 as the ones with Class-1, the class with the lowest number of samples. Since the Naive Bayes model highly depends on the values of prior probabilities, we suspect that this is one of the primary reasons why our Naive Bayes model is highly skewed towards Class-3. We will need to implement some sort of smoothing/undersampling/oversampling/augmentation techniques to handle this vast class imbalance.

2. There is a presence of possible outliers in the dataset. For example, below are the boxplots of two of the columns (‘`auto_5_28_sq_dist`’ and ‘`manual_4_5_sq_dist`’, respectively).



Because of these outliers, we believe the data is not close to being in the normal form, and that’s why the Gaussian Naive Bayes model is not able to predict correct class labels. We can try to eliminate the outliers, but we are unsure if they are really outliers. As these features are representative of Euclidean distance between landmarks, it is unclear why there are outliers. Perhaps, some of the landmarks may be farther from each other in comparison to the others, in which case we should focus on how we can generate better features from landmarks that can account for these issues.

Neural Network Model

The neural network model gave **accuracy of 0.68** and **F1 score of 0.46** on the test data.
(**F1-score of 0.65** if label imbalance is accounted for).

Similar to Naive Bayes model, we got the most accurate classification among all the labels for Class-3 (Happiness), with 0.84 f1-score. The model misclassified Class-2 (Disgust) most of the time; majority of these were classified as Class-6 (Neutral). Class-1 was completely mislabeled by the model, with f1-score of 0. Generally, the misclassifications are skewed towards Class-3 (Happiness) and Class-0 (Surprise).

Class-3 (Happiness) and Class-6 (Neutral) dominated our model and Class-1 & 2 were severely underrepresented in the model. Upon looking at the class distribution in our train set:

1. Just like in Naive Bayes, the model reflects the data distribution.
2. Class-6 (Neutral) has better f1-score 0.66, despite having similar sample size as Class-2 (f1-score 0), this could be due to the model classifying images as Neutral, if it isn't able to assign any other emotion labels to it.
3. We will need to use oversampling / augmentation / undersampling techniques for the classes in the dataset, to balance our dataset and get better classification results.

We tried using oversampling to balance out the classes, but it ended up making the performance even worse, so we abandoned the approach.

References

1. Real-world Affective Faces Database (RAF-DB): http://whdeng.cn/RAF/model1.html [Last Visited: 12/03/2022].
2. S. Li, W. Deng and J. Du, "Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2584-2593, doi: 10.1109/CVPR.2017.277.
3. Cross Entropy Loss- https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html [Last Visited: 12/03/2022].
4. Adam Optimizer - https://pytorch.org/docs/stable/generated/torch.optim.Adam.html [Last Visited: 12/03/2022].