Classification of Human Facial Expressions

Mariachiara Acconcia, Natasha Lalwani

Scope:

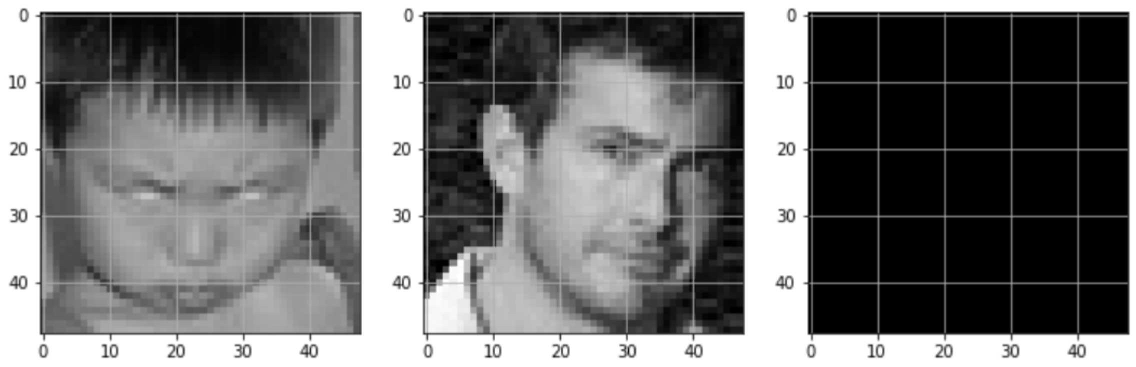
The purpose of this Project is to build a Machine Learning model that will classify human expressions from a dataset of images.
For the Emotion Recognition we will be using the FER2013 (https://www.kaggle.com/msambare/fer2013) dataset, which consists of a set ot training and testing images of people expressing seven different emotions: anger, disgust, fear, happiness, neutrality, sadness and surprise.

The images were downloaded in a "jpg" format and in this project, we will be working with the pixel "Gray-scale" values of each image.

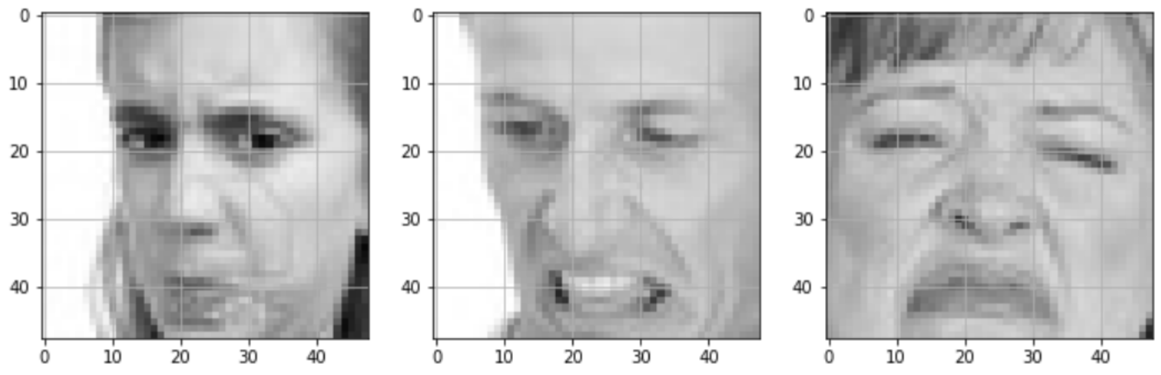
Image Sample:

Here is a sample of three images from each of the seven expressions.
The images were opened and displayed in 'RGB' values by using Python's Pillow (PIL) library.

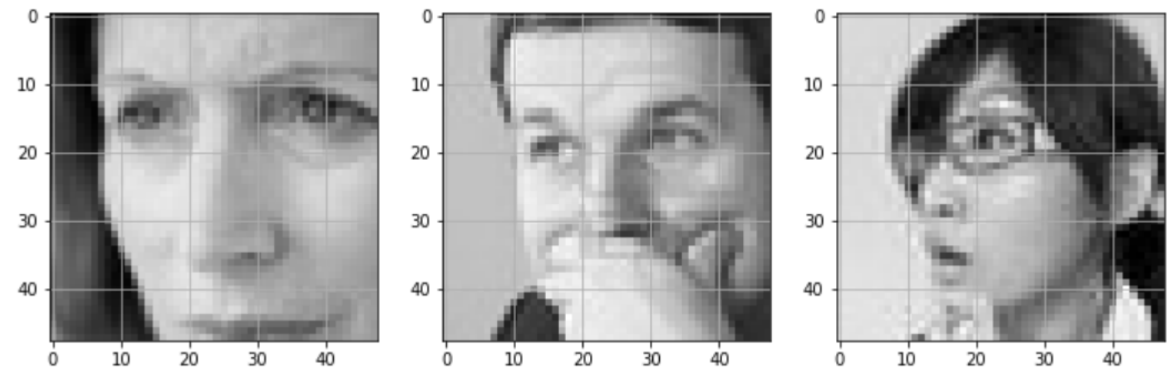
"Angry" images:



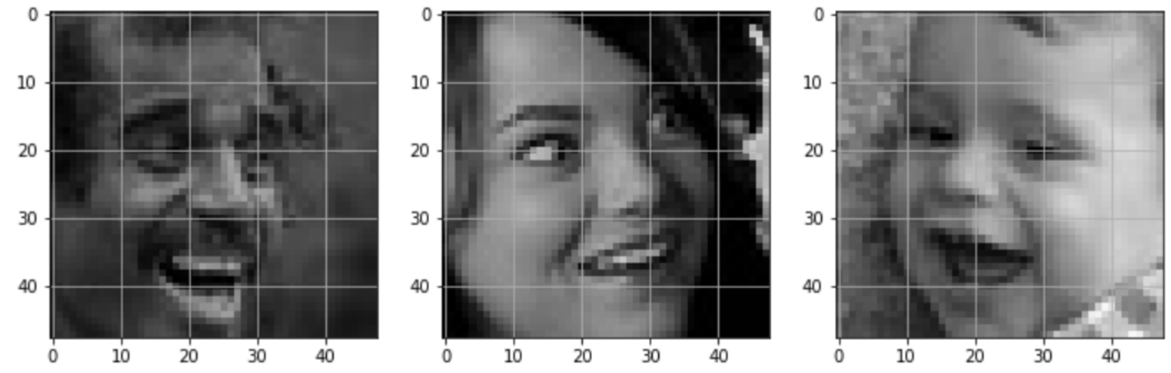
"Disgust" images:



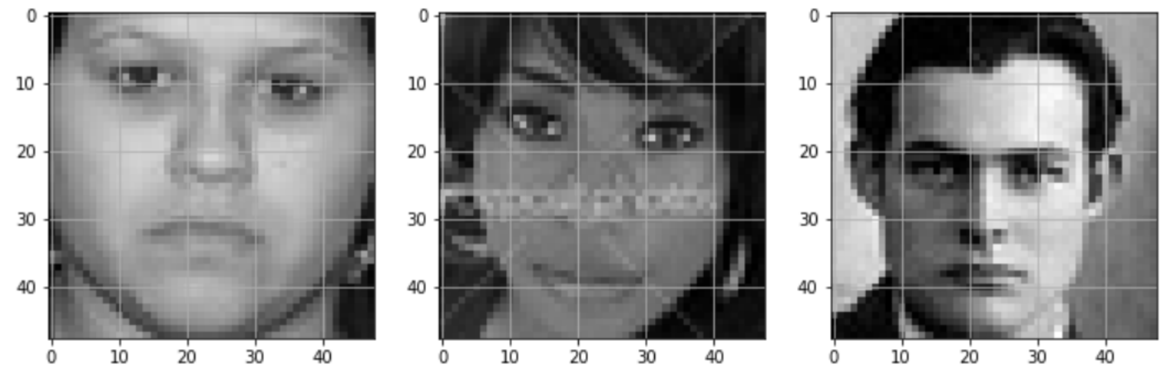
“Fear” images:



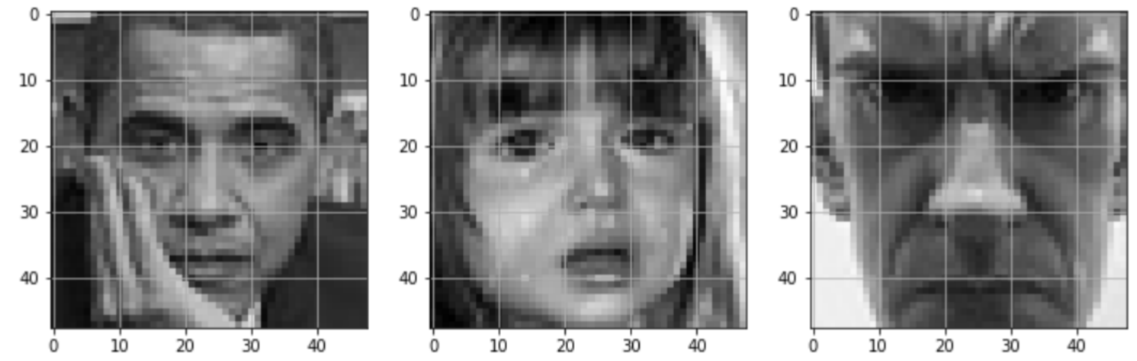
“Happy” images:



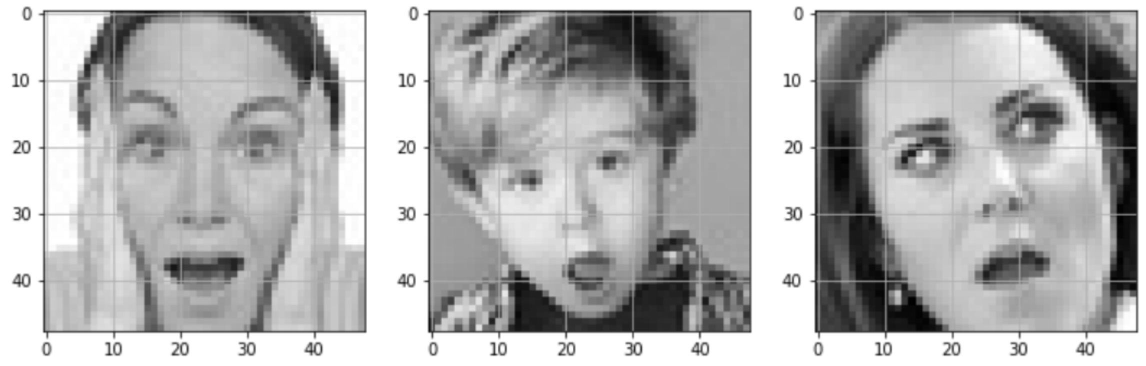
“Neutral” images:



“Sad” images:

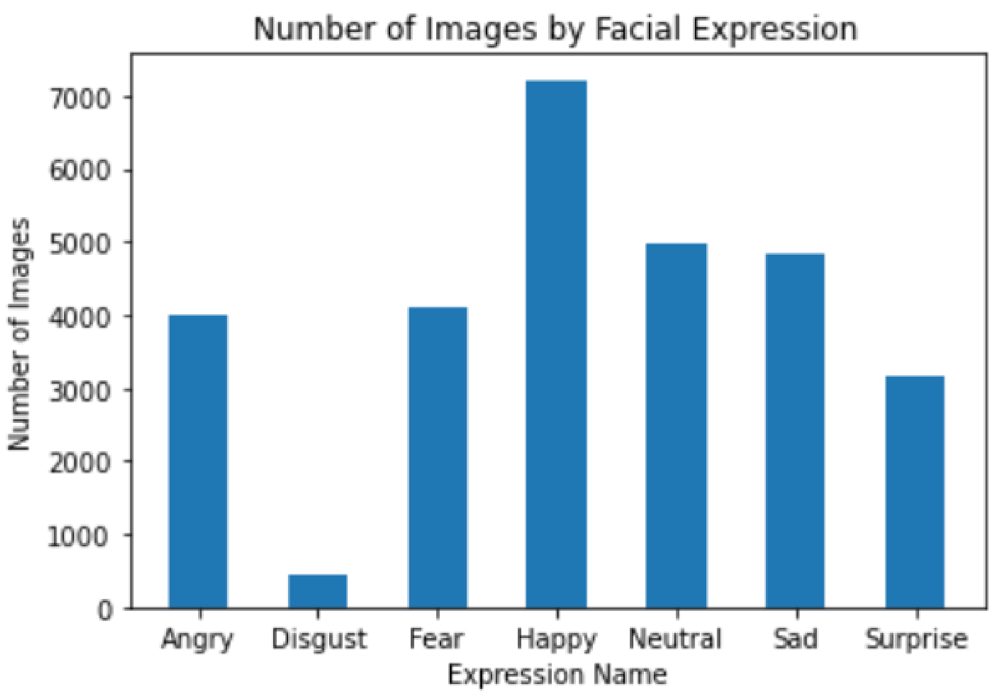


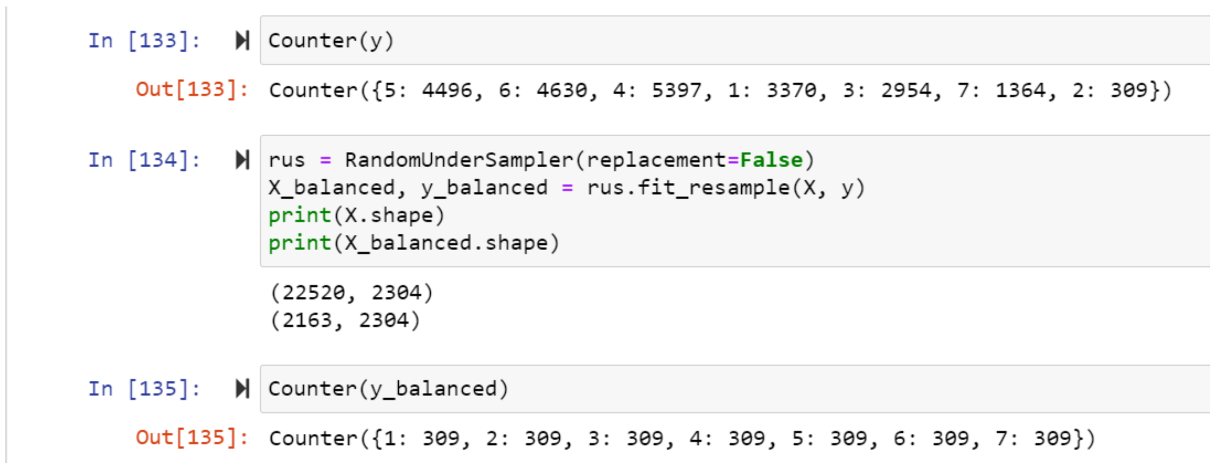
“Surprise” images:



Preprocessing:

By looking at the bar plots in the following pictures, we can see that the distribution of classes is imbalanced. The most populated class (“Happy”) has 7215 images, while the least populated class “Disgust” only has 436 images. Therefore, we decided to Down Sample the classes, in order to reduce the dimensionality of our dataset and also to balance all classes.

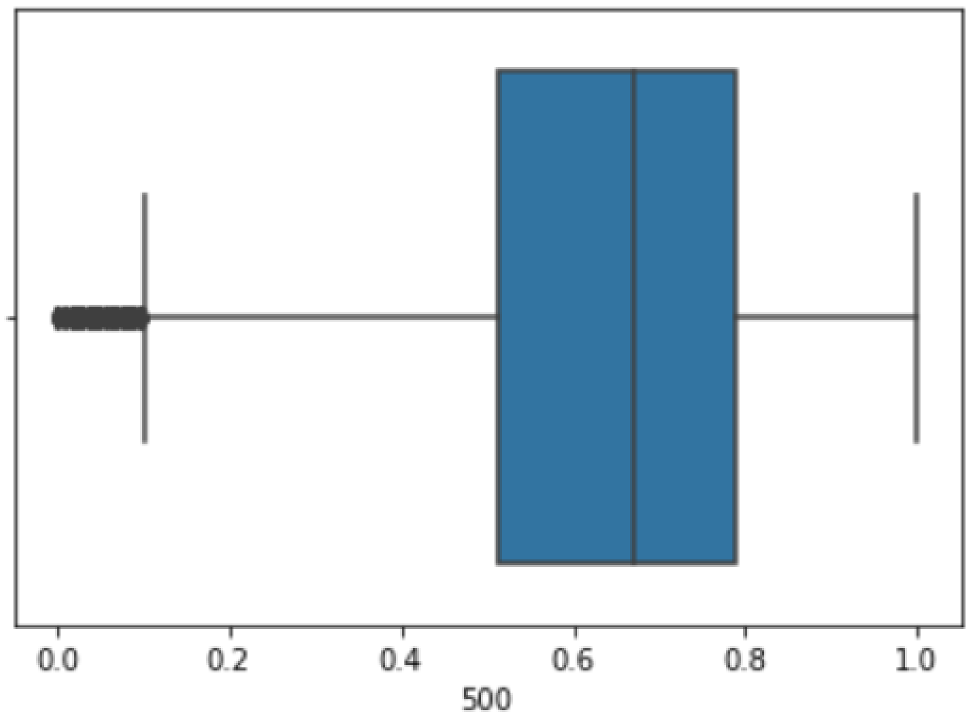




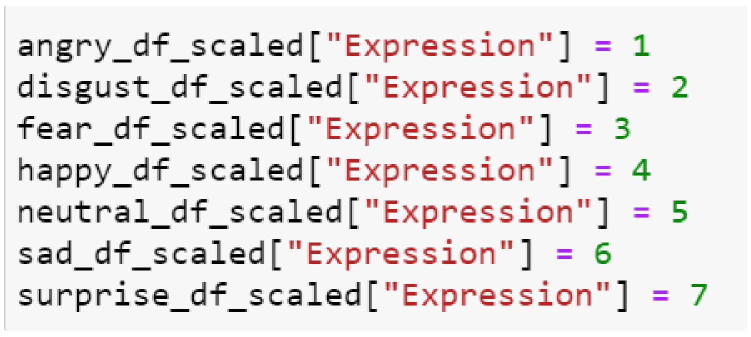
```
In [133]:  Counter(y)

Out[133]: Counter({5: 4496, 6: 4630, 4: 5397, 1: 3370, 3: 2954, 7: 1364, 2: 309})

In [134]:  rus = RandomUnderSampler(replacement=False)
           X_balanced, y_balanced = rus.fit_resample(X, y)
           print(X.shape)
           print(X_balanced.shape)

           (22520, 2304)
           (2163, 2304)

In [135]:  Counter(y_balanced)

Out[135]: Counter({1: 309, 2: 309, 3: 309, 4: 309, 5: 309, 6: 309, 7: 309})
```

To create our final dataset that we used for the implementation of different Machine Learning algorithms, we first created a single dataframe for each expression by taking in the images' pixels and flattening the 48*48 pixels into a 1 dimensional array and inserting the array into a Pandas dataframe. Each row represents one image.
We then preprocessed our data by scaling all the pixel values of each image using MinMax Scaler.
In order to get better accuracy scores with the implementation part of our project, we took care of the outliers of each 'facial expression' dataset by using IQR.
The interquartile range method gets rid of all the values that don't fall into a specific range of values, hence the outliers.

Below there is an example of a Box Plot showing outliers for the Facial Expression “Happy”:



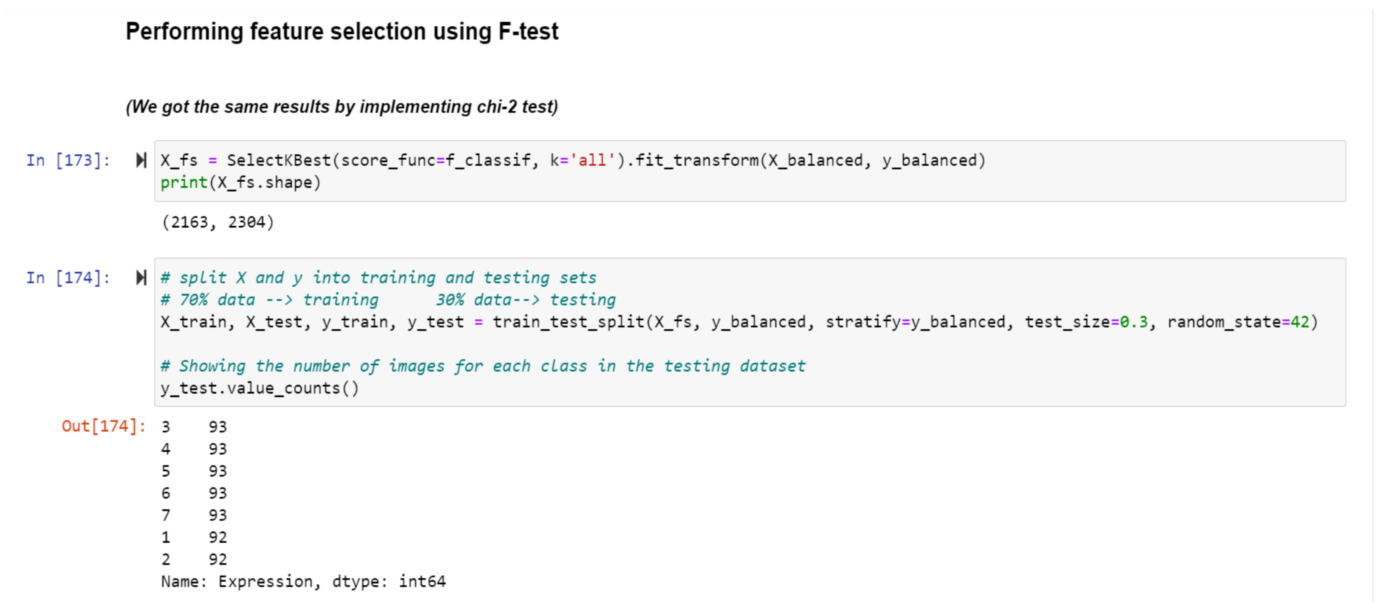
Finally we added the Label column "Expression" to each dataframe to indicate the target value to predict.



```
angry_df_scaled["Expression"] = 1
disgust_df_scaled["Expression"] = 2
fear_df_scaled["Expression"] = 3
happy_df_scaled["Expression"] = 4
neutral_df_scaled["Expression"] = 5
sad_df_scaled["Expression"] = 6
surprise_df_scaled["Expression"] = 7
```

We concatenated all data frames from all expressions together before implementing the different algorithms. After this step we also shuffled all the rows of the dataframe before moving on with the splitting of train and test.

After balancing our dataset, we performed feature selection using F-test. We implemented the same using the chi-2 test and got the same results, hence we decided to go with the F-test. After that, we split our dataset into training and testing sets.



Performing feature selection using F-test

(We got the same results by implementing chi-2 test)

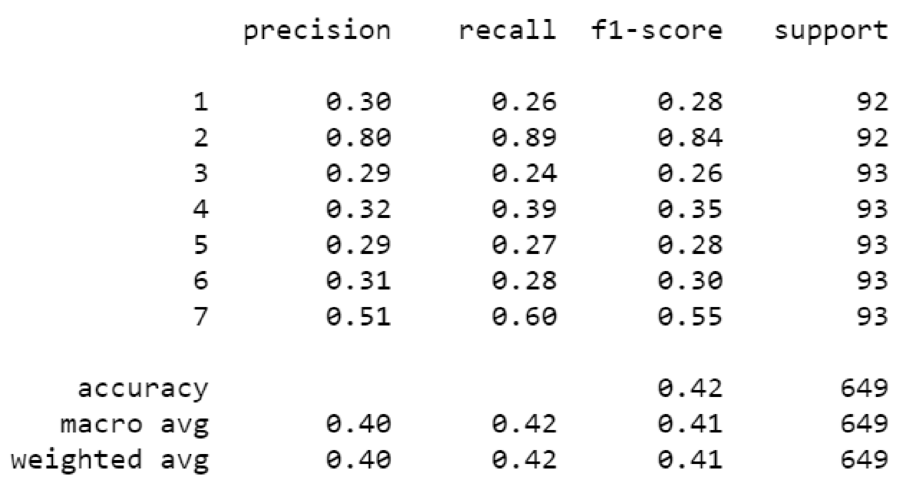
```
In [173]:  X_fs = SelectKBest(score_func=f_classif, k='all').fit_transform(X_balanced, y_balanced)
           print(X_fs.shape)

           (2163, 2304)

In [174]:  # split X and y into training and testing sets
           # 70% data --> training      30% data--> testing
           X_train, X_test, y_train, y_test = train_test_split(X_fs, y_balanced, stratify=y_balanced, test_size=0.3, random_state=42)

           # Showing the number of images for each class in the testing dataset
           y_test.value_counts()

Out[174]:  3    93
           4    93
           5    93
           6    93
           7    93
           1    92
           2    92
           Name: Expression, dtype: int64
```

ML Models Implementation and Results:

Since we have seven different emotions, we are dealing with a classification problem, and we implemented six different classification algorithms in this project: Logistic Regression, KNN, SVC, Naive Bayes, OneVsRest and Decision Tree Classifier.
For each machine learning algorithm we implemented Grid Search in order to find the optimal hyperparameters.

Here we show the classification reports of each implemented algorithm.

Logistic Regression Classifier:



```
              precision    recall  f1-score   support

           1       0.30      0.26      0.28        92
           2       0.80      0.89      0.84        92
           3       0.29      0.24      0.26        93
           4       0.32      0.39      0.35        93
           5       0.29      0.27      0.28        93
           6       0.31      0.28      0.30        93
           7       0.51      0.60      0.55        93

    accuracy                           0.42       649
   macro avg       0.40      0.42      0.41       649
weighted avg       0.40      0.42      0.41       649
```

KNN Classifier:

```
           1       0.18      0.12      0.14        92
           2       0.44      0.61      0.51        92
           3       0.19      0.14      0.16        93
           4       0.22      0.26      0.24        93
           5       0.23      0.28      0.25        93
           6       0.27      0.17      0.21        93
           7       0.38      0.46      0.42        93

    accuracy                           0.29       649
   macro avg       0.27      0.29      0.28       649
weighted avg       0.27      0.29      0.28       649
```

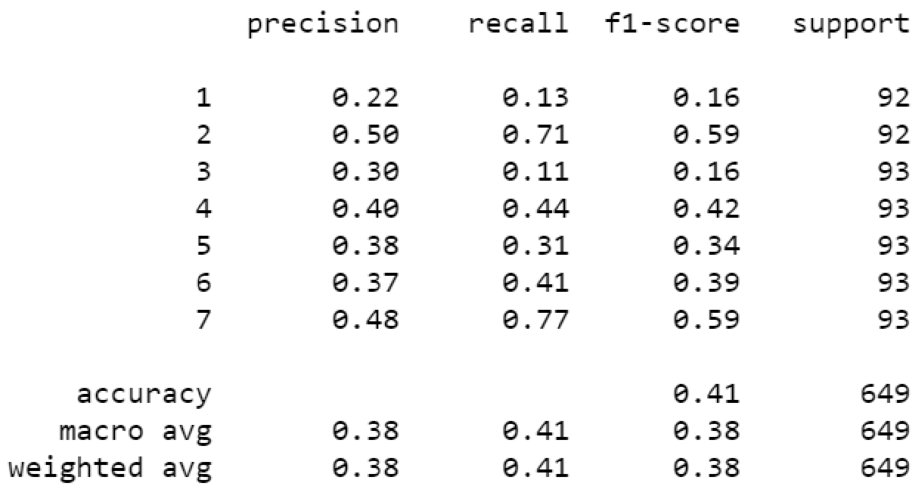
Support Vector Classifier:

```
           1       0.23      0.30      0.26        92
           2       0.66      0.70      0.68        92
           3       0.27      0.28      0.28        93
           4       0.33      0.38      0.35        93
           5       0.28      0.22      0.24        93
           6       0.40      0.31      0.35        93
           7       0.48      0.44      0.46        93

    accuracy                           0.37       649
   macro avg       0.38      0.37      0.37       649
weighted avg       0.38      0.37      0.37       649
```

Naive Bayes (Gaussian) Classifier:

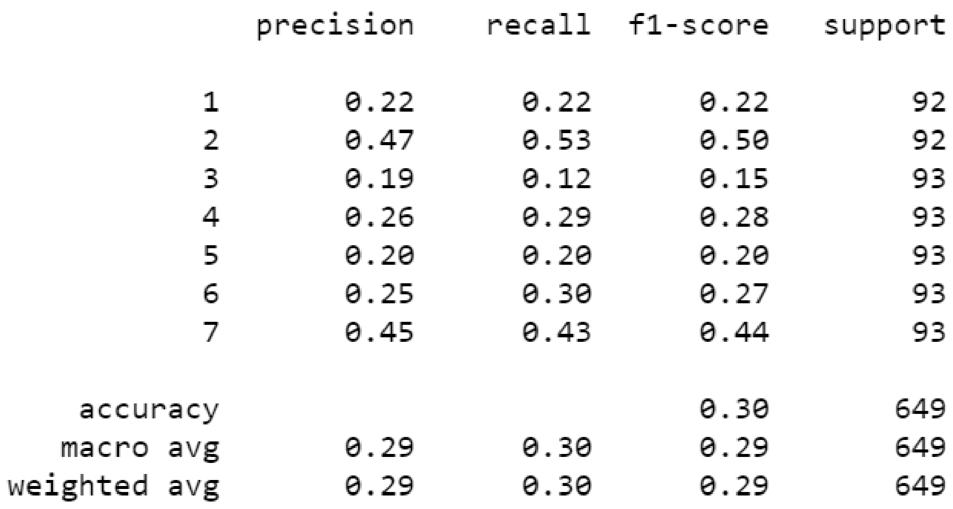
```
              precision    recall  f1-score   support

           1       0.18      0.07      0.10        92
           2       0.10      0.05      0.07        92
           3       0.22      0.12      0.15        93
           4       0.23      0.24      0.23        93
           5       0.19      0.13      0.15        93
           6       0.32      0.51      0.39        93
           7       0.30      0.68      0.42        93

    accuracy                           0.26       649
   macro avg       0.22      0.26      0.22       649
weighted avg       0.22      0.26      0.22       649
```

One vs Rest Classifier:



```
              precision    recall  f1-score   support

           1       0.22      0.13      0.16        92
           2       0.50      0.71      0.59        92
           3       0.30      0.11      0.16        93
           4       0.40      0.44      0.42        93
           5       0.38      0.31      0.34        93
           6       0.37      0.41      0.39        93
           7       0.48      0.77      0.59        93

    accuracy                           0.41       649
   macro avg       0.38      0.41      0.38       649
weighted avg       0.38      0.41      0.38       649
```

Decision Tree Classifier:

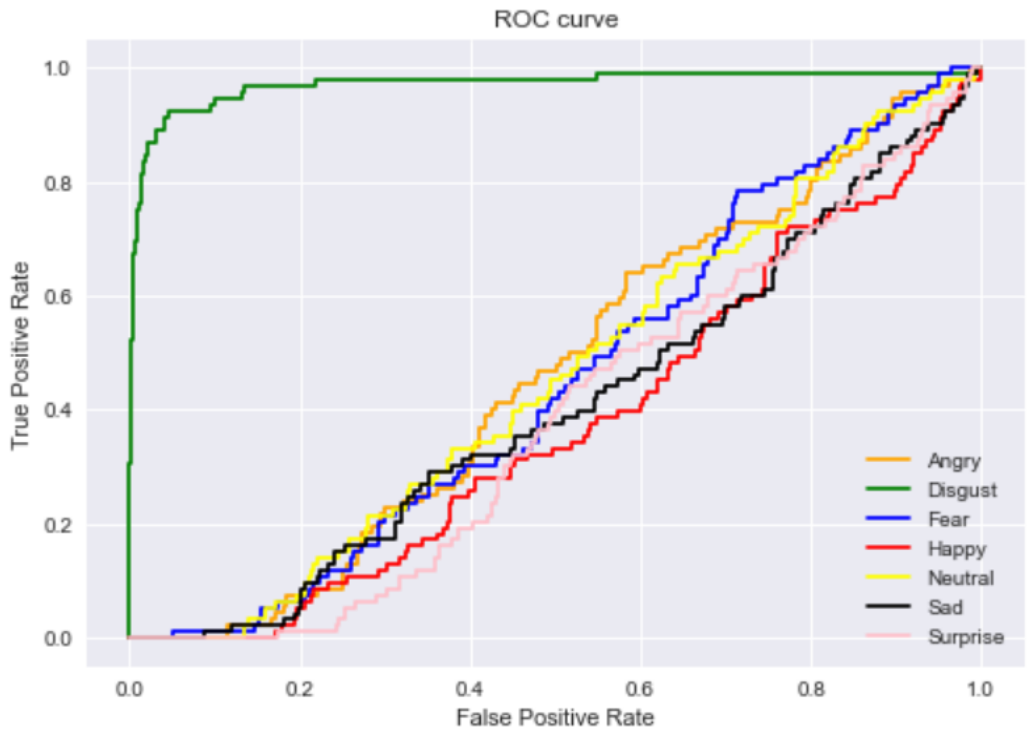


```
              precision    recall  f1-score   support

           1       0.22      0.22      0.22        92
           2       0.47      0.53      0.50        92
           3       0.19      0.12      0.15        93
           4       0.26      0.29      0.28        93
           5       0.20      0.20      0.20        93
           6       0.25      0.30      0.27        93
           7       0.45      0.43      0.44        93

    accuracy                           0.30       649
   macro avg       0.29      0.30      0.29       649
weighted avg       0.29      0.30      0.29       649
```

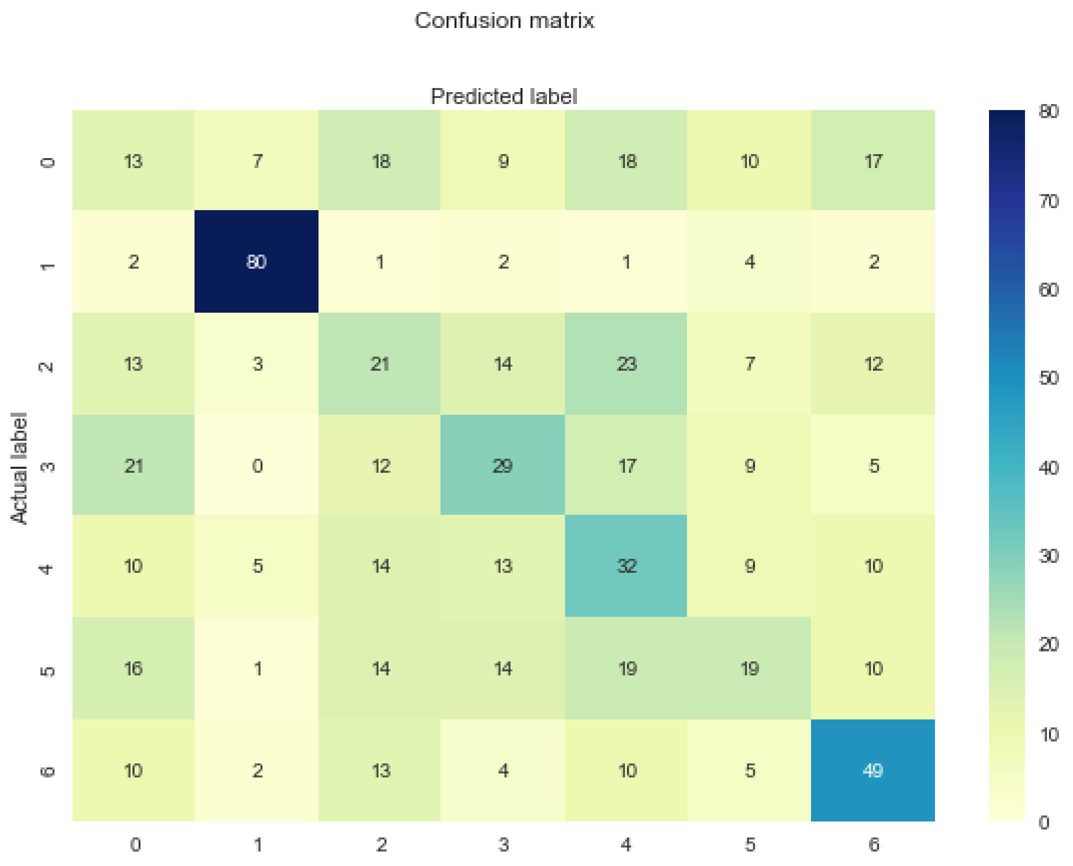
The Accuracy tells us how often the classifier is correct. The Precision Score is the accuracy of the positive predictions while the Recall Score is the ratio of positive instances that are correctly detected by a classifier.

As we can see from the classification reports, the Logistic Regression Model is the algorithm that gives us the highest scores and accuracies, so it’s the model that we considered for further analysis and plots.

Here is the ROC curve for each facial expression to represent the rate between the false positive and true positive.



Here below we show the confusion matrix, which represents how often a class gets identified correctly and how often it gets identified incorrectly, and which class it gets mistaken for.



	0	1	2	3	4	5	6
0	13	7	18	9	18	10	17
1	2	80	1	2	1	4	2
2	13	3	21	14	23	7	12
3	21	0	12	29	17	9	5
4	10	5	14	13	32	9	10
5	16	1	14	14	19	19	10
6	10	2	13	4	10	5	49

Limitations:
After proving that our dataset is imbalanced, we performed SMOTE, down-sampling and up sampling. We even took care of the outliers and performed feature selection. For feature selection we did not decrease the number of features drastically since each feature represents a pixel value which is an integral part of the process of representing each facial expression. Despite all the measures taken, the accuracy is not what we had hoped for and at the same time the running time of the algorithms was longer, which is not supported by local systems.

Future Scope:
The future scope of this project could be to implement a Deep Learning CNN model to classify the expressions, and mapping each expression to corresponding emojis or avatars. OpenCV tools could be used to detect the bounding boxes of faces in the webcam to predict the emotions.