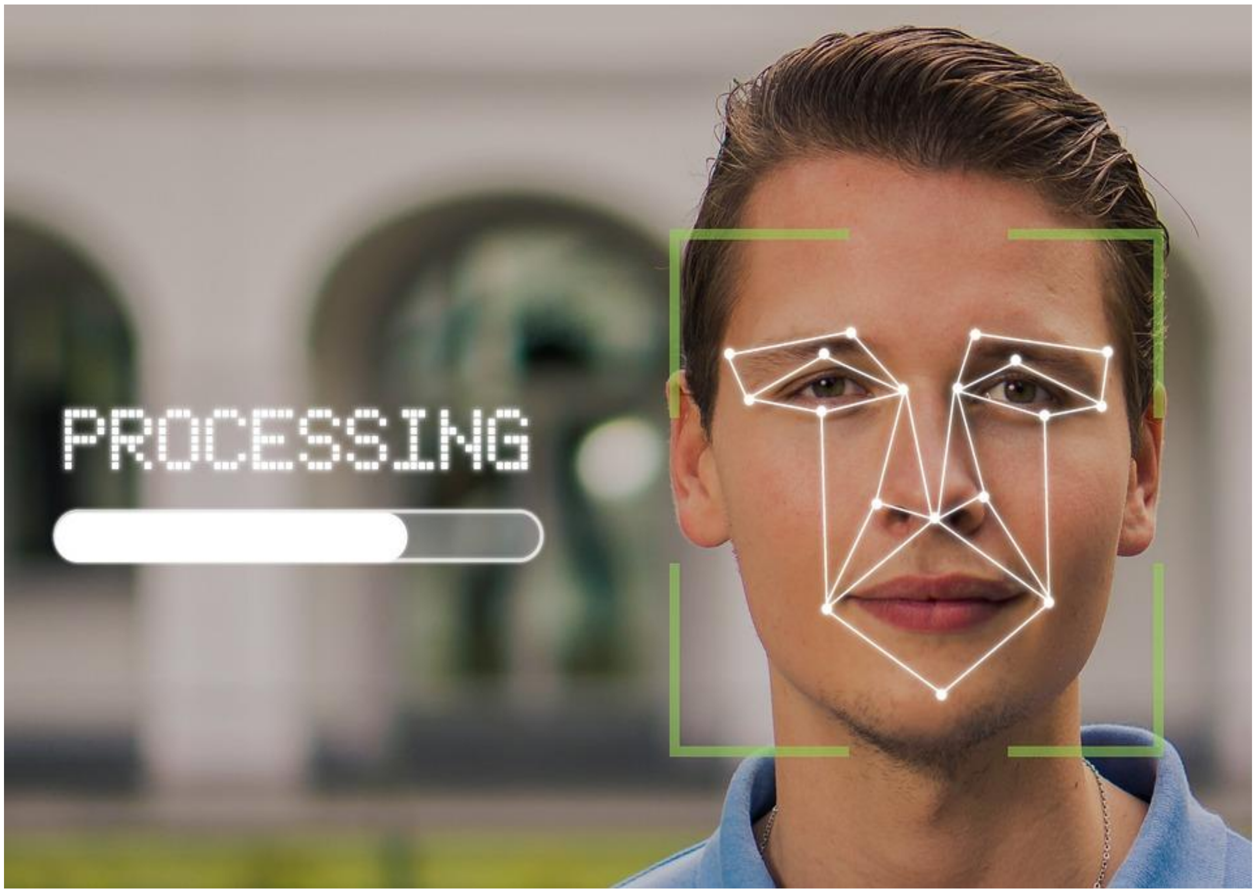
FACIAL EMOTION RECOGNITION

Machine Learning Project



Author: Rishi Arora 20C21003

Other Authors:

- Mihir Pandya 20C21036
- Pratham Bhatt 20C21005

TABLE OF CONTENTS

ABSTRACT

Face detection has been around for a long time. Moving ahead, human emotion shown by the face and felt by the brain, collected in video, electric signal (EEG), or picture form, can be approximated. Human emotion recognition is critical so that current artificial intelligence systems can mimic and assess reactions from facial expressions. This can help you make educated judgments about identifying intent, promoting offerings, or dealing with security issues. Recognizing emotions from images or video is a simple task for the human eye, but it is extremely difficult for machines and necessitates the use of numerous image processing techniques for feature extraction. Several machine learning methods are appropriate for this task. Any machine learning detection or recognition requires training algorithms and testing on an appropriate dataset. This project investigates a few machine learning algorithms as well as feature extraction techniques that can aid in the accurate identification of human emotions.

NEED FOR TITLE

Face recognition is an amazing technology used to detect the emotion of the human. Human emotion detection is implemented in many areas. Some of the areas are mentioned below

A. AUTHENTICATION

Human emotion detection is the need of the hour so that modern artificial intelligent systems can emulate and gauge reactions from face. This can be helpful to make informed decisions be it regarding identification of intent, promotion of offers or security related threats. Thus, human emotion detection is implemented in areas requiring additional security or information about the person. This can also be useful to verify that the person standing in front of the camera is not just a 2-dimensional representation.

B. BUSINESS

Another important domain where we see the importance of emotion detection is for business promotions. Most of the businesses thrive on customer responses to all their products and offers. If an artificial intelligent system can capture and identify real time emotions based on user image or video, they can make a decision on whether the customer liked or disliked the product or offer. Emotional analytics tells what people are feeling and why. It provides deep and contextual understanding with valuable insights to identify and resolve issues.

C. MEDICAL

In medical Face emotion system is used for identification of mental disorders, depression analysis, and health forecasting and criminal detection.

ALGORITHM USED

Once the dataset is created using the required features, the next important step is to use a good classification algorithm. Support Vector Machines (SVM) are used almost in all cases of multi-class classification of human expressions. They are combined with one or another feature extraction technique.

Why Support Vector Machines (SVM)?

K-Nearest Neighbors algorithm is Lazy learner algorithm and large amount of computational space is required to load the data which makes classification process very slow. So this algorithm is not suitable for FER system.

It is tough to obtain complex relationships using logistic regression and on high dimensional datasets, this may lead to the model being over-fit on the training set, which means overstating the accuracy of classification and prediction on the training set and thus the model may not be able to predict accurate results on the test set.

Naïve Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features. So, it isn’t ideal for image dataset.

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand. But at a same time it contains lots of layers, which makes it complex and may have an overfitting issue. Space and time Complexity of the decision tree is comparative high then SVM.

SVM works relatively well when there is a clear margin of separation between classes, more effective in high dimensional spaces and relatively memory efficient.

DATASET DESCRIPTION

Name and Source:

Because this project elicits the user's emotions, we must supply photographs as the dataset. We obtained our dataset from Kaggle (Face Emotion Recognition with EfficientNetB2). This dataset is in zip format and is divided into two subfolders: train and test. The train and test folders have seven folders for each expression, however we only utilised two for two emotions.

Filtering Techniques Used:

Dataset which is being used here is set of images. Each images is already gray-scaled and is of 48x48 size.

At first, we will store images in pandas dataframe. Then we will store this dataframe in .csv format by using dataframe.to_csv().

After this our dataset consists of 2304 features.

Just as physics at extremely small scales (Quantum Physics) behaves differently from physics that of everyday things and "normal" size (Classical Physics) the same happens with the **dimensions**.

A vector in **two dimensions** is composed of **two components**:

$$\vec{v}_{\mathbb{R}^2} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

a **three-dimensional** vector is made up of **three components**:

$$\vec{v}_{\mathbb{R}^3} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

and a vector of nn **dimensions** (also called a hypervector) is composed of nn **components**:

$$\vec{v}_{\mathbb{R}^n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Something interesting about the **high dimensions** arises two things:

Most of the points are very close to the surface of the hypercube.

The average distance between two points of the hypercube is high. For examples for a unit hypercube of 1×1061×106 dimensions is approximately 408.25, while for a unit cube three-dimensional is 0.66.

Why will all this happen? Well, there is a lot of space when you are in many dimensions, as a result, the data is very sparse: much of the training data is far from each other. This also means that new data may be too far from the training data making the predictions less reliable compared to least dimensional data.

How could it be solved?

One way to remove this obstacle is to increase the number of instances to have a higher density. of points, however in practice it is not very feasible.

The next option seems to be very obvious but it is **reduce the dimensions** and for this we can take one of several paths, but for this work two are explored:

- ***Feature detection***
- ***Dimensionality Reduction***

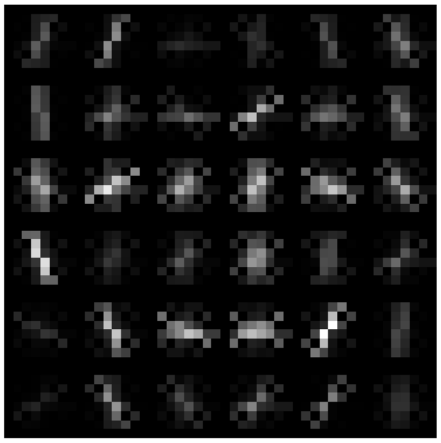
Feature Detector: Histogram of Oriented Gradients (HOG):

Histograms of Oriented Gradients or HOG is a feature descriptor proposed by Dalal et al in 2005 [2]. This descriptor computes the change in pixel values around the one of interest. This change is given by a **gradient** $G^{\rightarrow}$ (change vector) whose magnitude (mm) and angle (θθ) are given by equations 1 and 2:

$$m(x,y) = \sqrt{\vec{G}(x,y)\cdot\vec{G}(x,y)} = \sqrt{(G_x(x,y))^2 + \left(G_y(x,y)\right)^2}$$

$$\theta(x,y) = \tan^{1}\frac{G_y(x,y)}{G_x(x,y)}$$

Before



After

Dimensionality Reduction: Principal Component Analysis (PCA):

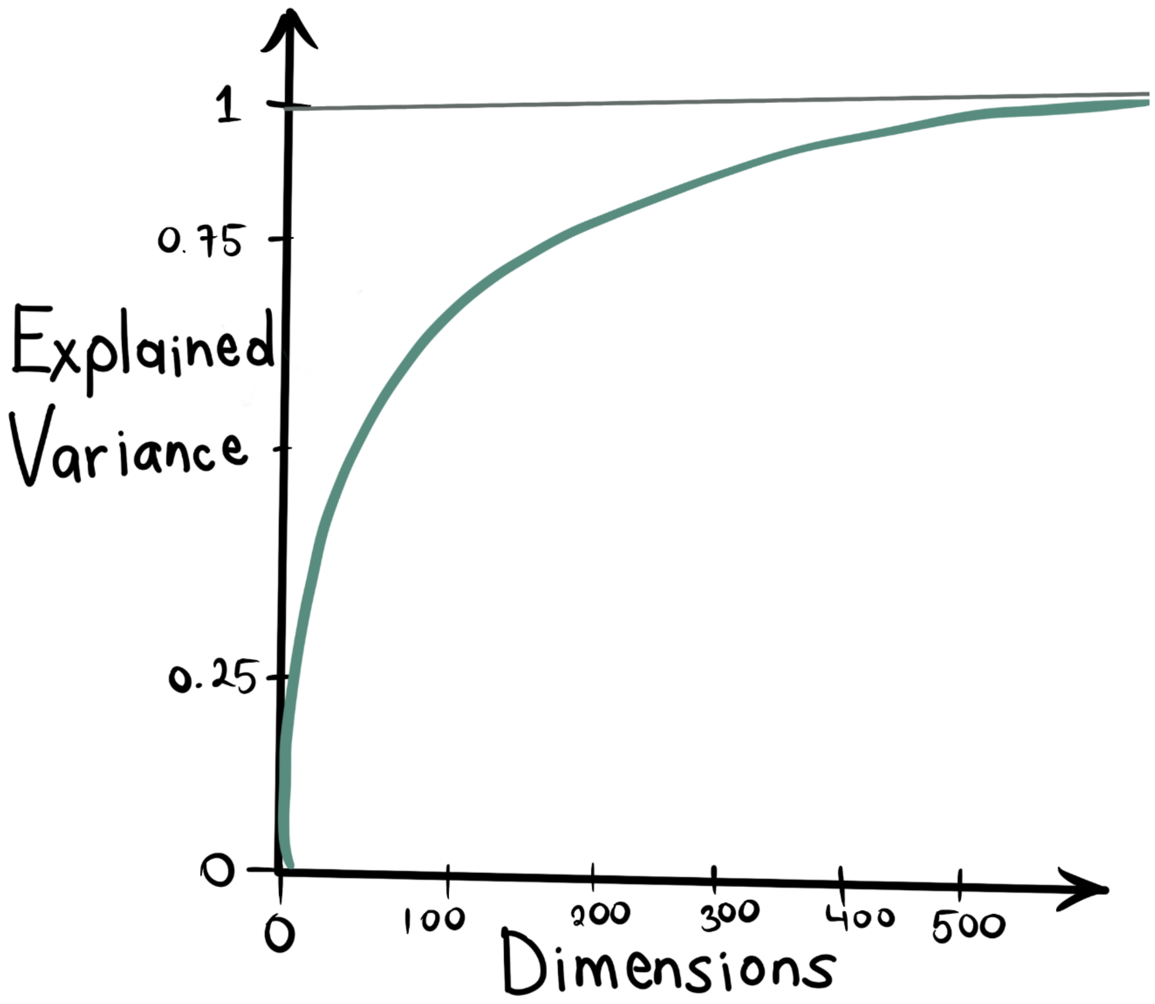
In order to optimize the training of the *Machine Learning* model, the following step will be taken: **Reduce the dimensions**.

It is clear that with the help of the HOG algorithm, the dimensions went from 2304 to 900 preserving a large amount of the information that contains the image, however, it still has a **huge** amount of *features*.

For this task, one of the most popular dimensionality reduction (and unsupervised learning) algorithms will be used. acquaintances: **PCA**

Generally speaking, the goal of **PCA** is to be able to find a linear projection that can **preserve the most information**. One of the ways in which the *preserving information* approach can be seen is that by looking for the plane where the new projection of the data will be made, find the one that can **maximize the variance**.

Just as in the ***K-Means*** algorithm, you have to give *a priori* the number of *clusters* you want and therefore there are to find a criterion to determine the optimal number of *clusters*, the same happens with **PCA**. To find the number of ideal dimensions, you have to know when the ***explained variance*** stops growing significantly, as shown in the following figure.



Another way to obtain the **optimal number of dimensions** is by specifying the percentage of information that we want to keep. This task is possible thanks to the **Scikit-Learn** library, since instead of specifying the integer number of dimensions in **`n_components`**, the percentage of information to keep can be written as a floating value.

For this project we will try to **preserve 90% of the information of the *data set***

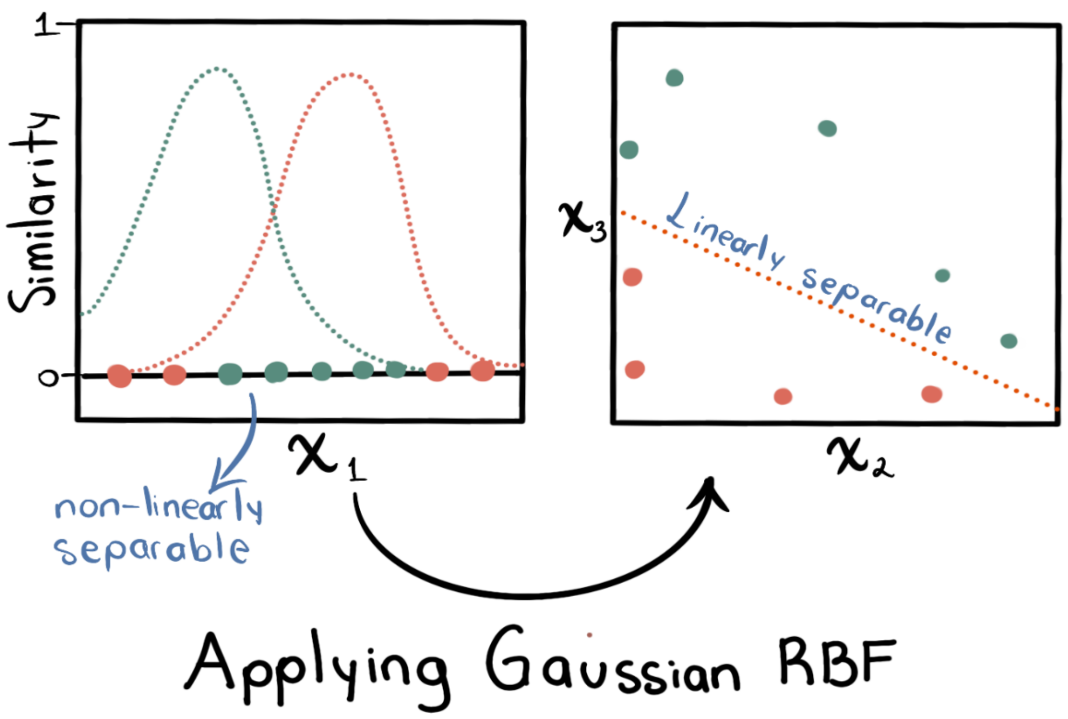
PROCESSING:

The ***Support Vector Machine*** algorithm is one of the most elegant algorithms in Machine Learning, also one of the most powerful and used in different problems; with a good pre-processing of the data, the predictions become similar to those of neural networks, an example is shown in the article Dimensionality Reduction for Handwritten Digit Recognition.

One of the most interesting features of this algorithm is the kernel trick, which in a few words, can transform the dimensions of the data so that you can count the hyperplane that can classify the data, without compromising the effort computational of the algorithm.

There are different types of kernels, but for this problem we will try the **Gaussian Radial Basis Function kernel**. Is it is a technique used when the available data are not linearly separable.

What this kernel does is add features via a similarity function (the **Gaussian Radial Basis Function**) that measures how much an instance resembles a certain region or landmark.



In the image above you can see how a set of data that **are not linearly separable** is transformed into another that **if it is linearly separable** with the function **Gaussian Radial Basis Function**

($x1$ is the original feature, $x2$ and $x3$ are the features created by the function)

The implementation of this algorithm in Scikit-Learn is by declaring **kernel = 'rbf'** in the model

Hyper-parameters of SVC with kernel = 'rbf'

The Equation 3 is the kernel of **Gaussuan Radial Basis Function**

$$\mathrm{K}(\mathbf{a}, \mathbf{b}) = \exp\left(-\gamma ||\mathbf{a} - \mathbf{b}||^2\right)$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors, one of them (usually the second) is the landmark; and γ **is one of the hyper-parameters**.

Increasing γ makes the bell shape of the curve narrower; as a result the range of influence of each instance is smaller causing the decision boundary to be a bit more jagged, sticking to individual instances. case contrary to when γ decreases, the shape of the curve is broader; instances have a greater range of influence and the decision boundary it's a bit more "soft".

In this project SVM is perfectly suitable for FER System Because of below advantages:

1. *SVM works relatively well when there is a clear margin of separation between classes.*
2. *SVM is more effective in high dimensional spaces.*
3. *SVM is effective in cases where the number of dimensions is greater than the number of samples.*
4. *SVM is relatively memory efficient*

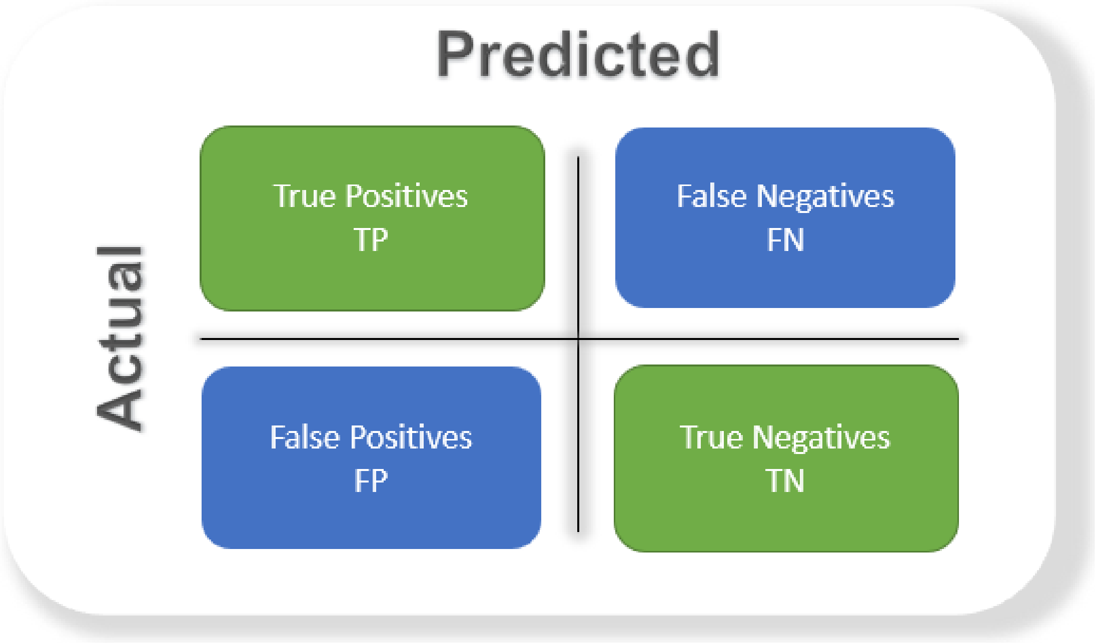
DATASET DISTRIBUTION:

We tried many other splits such as 80:20 and 70:30. 70:30 split seemed more appealing as our assumption was all classes will be equally represented in the test set. For cross-validation score we initially tested with 4 splits. To improve the results we chose the value 5 and 10, which are standard values for cross-validation. Random Forest Classifier and Decision Trees were also run on our dataset, but resulted into low accuracy as compared to other algorithms in our experiment; hence we decided to continue with SVM with 80:20 split.

EVALUATION METHODS:

Confusion Matrix:

The confusion matrix forms the basis for the other types of classification metrics. It’s a matrix that fully describes the performance of the model. A confusion matrix gives an in-depth breakdown of the correct and incorrect classifications of each class.



The four terms represented in the image above are very important.

Let’s define them:

- ***True positives*** *– a scenario where positive predictions are actually positive.*
- ***True negatives*** *– negative predictions are actually negative.*
- ***False positives*** *– positive predictions are actually negative.*
- ***False negatives*** *– a scenario where negative predictions are actually positive.*

From the definition of the four terms above, the takeaway is that it’s important to amplify true positives and true negatives. False positives and false negatives represent misclassification, that could be costly in real-world applications. Consider instances of misdiagnosis in a medical deployment of a model.

A model may wrongly predict that a healthy person has cancer. It may also classify someone who actually has cancer as cancer-free. Both these outcomes would have unpleasant consequences in terms of the well being of the patients after being diagnosed (or finding out about the misdiagnosis), treatment plans as well as expenses. Therefore it’s important to minimize false negatives and false positives.

The green shapes in the image represent when the model makes the correct prediction. The blue ones represent scenarios where the model made the wrong predictions. The rows of the matrix represent the actual classes while the columns represent predicted classes.

We can calculate accuracy from the confusion matrix. The accuracy is given by taking the average of the values in the “true” diagonal.

That is:

Accuracy = (True Positive + True Negative) / Total Sample

That translates to:

Accuracy = Total Number of Correct Predictions / Total Number of Observations

Since the confusion matrix visualizes the four possible outcomes of classification mentioned above, aside from accuracy, we have insight into precision, recall, and ultimately, F-score. They can easily be calculated from the matrix. Precision, recall, and F-score are defined in the section below.

F-score

F-score is a metric that incorporates both the precision and recall of a test to determine the score. It defines as the harmonic mean of recall and precision. F-score is also known as F-measure or F1 score.

Let’s define precision and recall.

Precision refers to the number of true positives divided by the total positive results predicted by a classifier. Simply put, precision aims to understand what fraction of all positive predictions were actually correct.

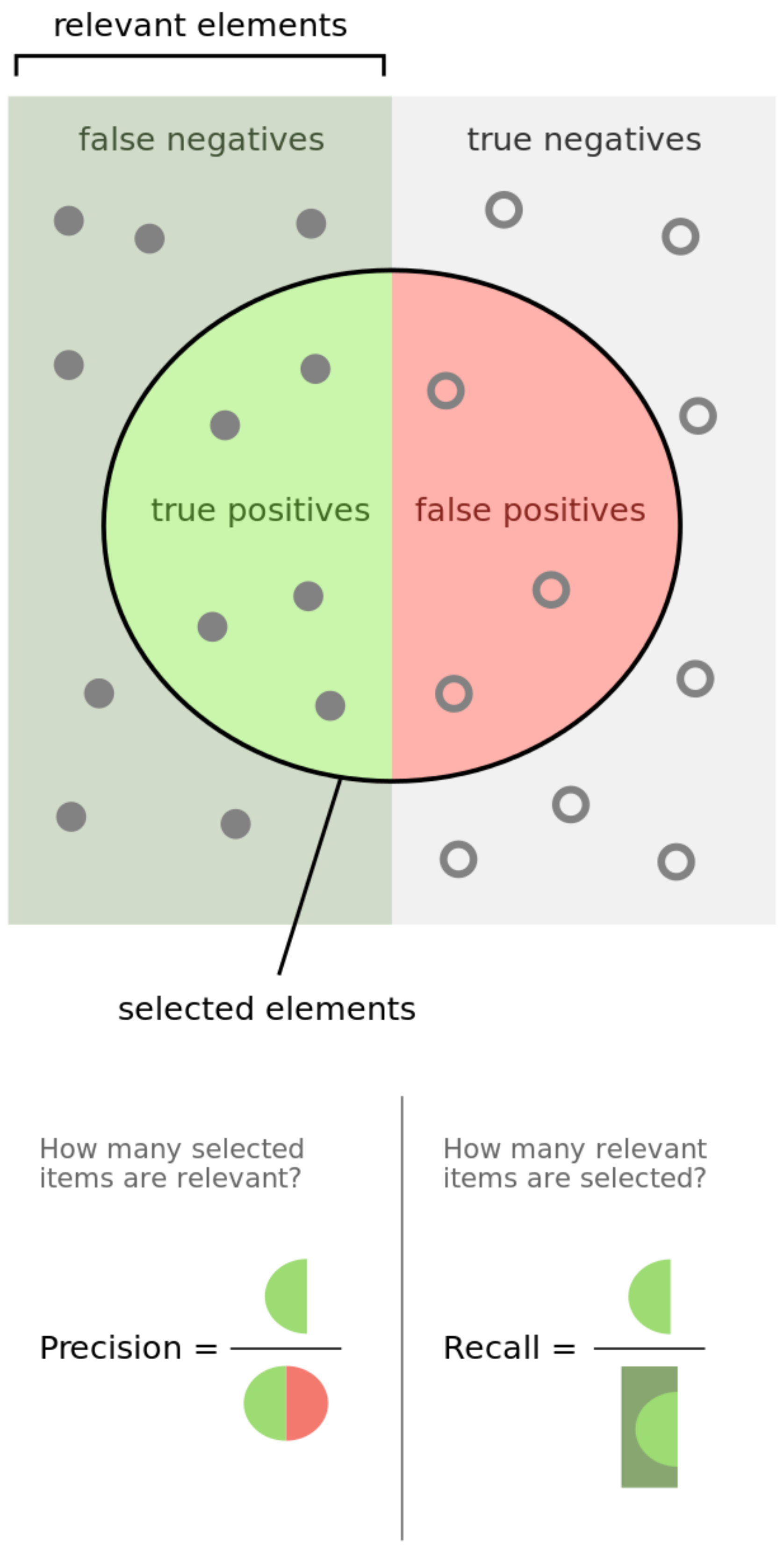
Precision = True Positives / (True Positives + False Positives)

On the other hand, recall is the number of true positives divided by all the samples that should have been predicted as positive. Recall has the goal to perceive what fraction of actual positive predictions were identified accurately.

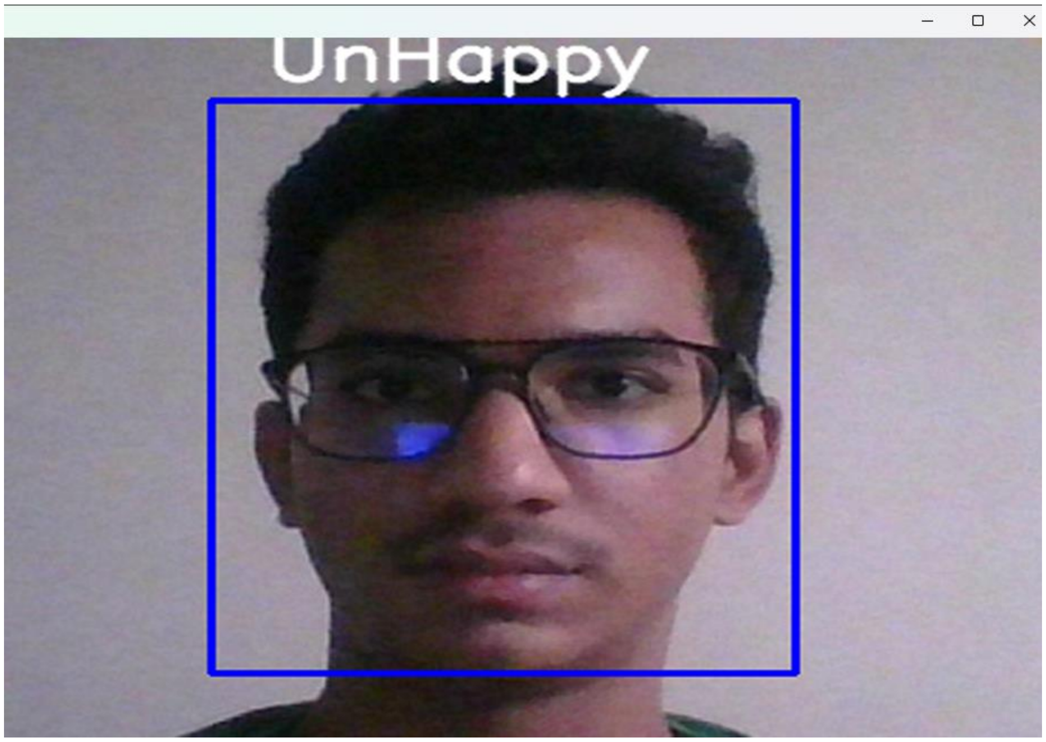
Recall = True Positives / (True Positives + False Negatives)

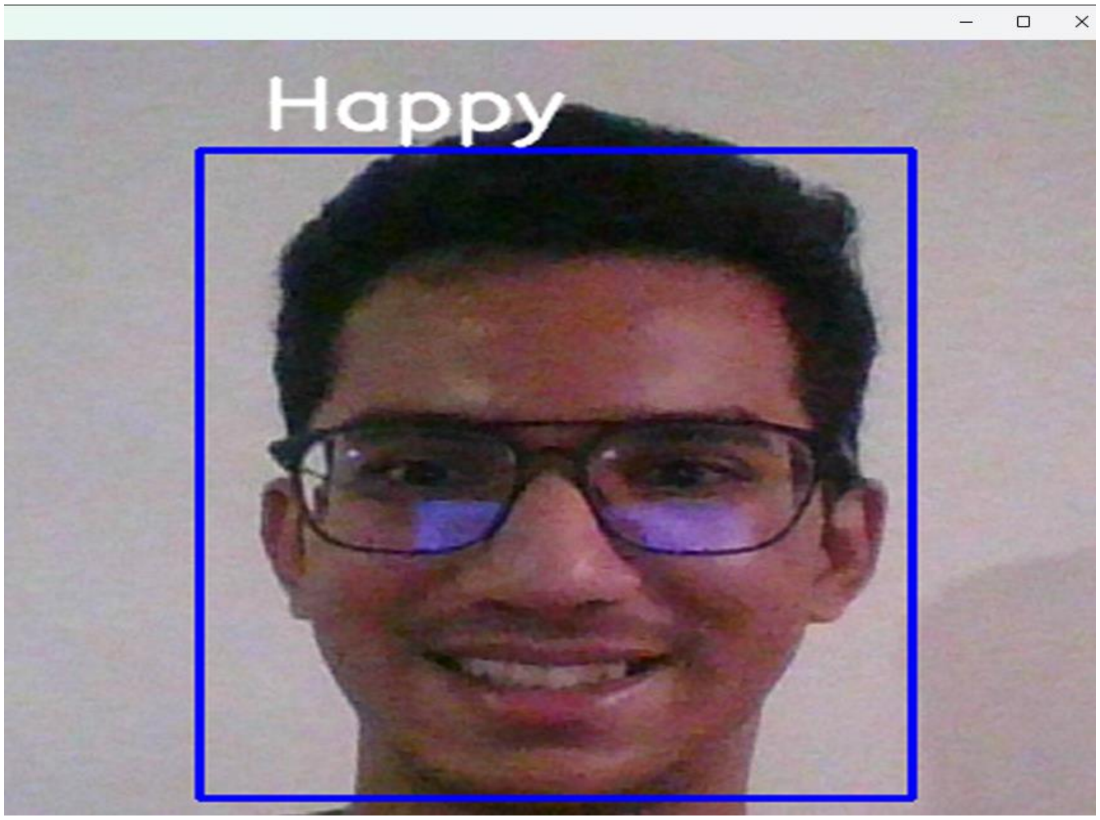
In addition to robustness, the F-score shows us how precise a model is by letting us know how many correct classifications are made. The F-score ranges between 0 and 1. The higher the F-score, the greater the performance of the model.

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many selected items are relevant?
How many relevant items are selected?
Precision =
Recall =

OUTPUT:





CONCLUSION:

- *Reducing the dimensions has helped to reduce the training time, since using the more than four thousand features Initially, the SVM algorithm would have been saturated.*
- *Knowing what hyperparameters are and what they mean in each type of Machine Learning algorithm is key to being able to adjust them and thus achieve a better performance of the model.*
- *Learned new and/or reinforced knowledge of concepts such as*
  - ***Histograms of Oriented Gradients**: A quite useful feature descriptor to know what is most relevant in an image. The creation of a class so that this descriptor could enter inside the Pipeline has future applications, since its hyper-parameters to improve the performance of some model.*
  - ***Principal Component Analysis**: Although choosing the number of dimensions to reduce can be a very subjective task (detect elbow of the graph), it is very useful to be able to choose the proportion of information that you want to keep.*
- *The best classification model is the second, where there is a high value of **F1-Score**, the ideal metric to prioritize equally the **Precision** and **Recall**.*
- *Take complex problems, which have been treated with Deep Learning, and try to solve them with classic Machine Learning algorithms It can be of great help for a better understanding of concepts seen and the application of new ones. I also think that doing this shows the ability to solve problems by being able to take different perspectives in the same situation.*