**Report. Facial Recognition Project**



Universidad Autónoma de Occidente

Subject:

Image Processing

Teacher:

[Victor Adolfo Romero Cano](mailto:varomero@uao.edu.co)

Student:

Luis Fernando Pareja Bernal

Santiago de Cali, November 9th/2021

**PROBLEM STATEMENT**

In times of pandemic, biosecurity protocols have been implemented in all places where there may be a possible crowd of people, all this in an attempt to protect ourselves from the covid-19 virus. At the Universidad Autónoma de Occidente these protocols are seriously implemented and executed by the staff, unfortunately for the university community such seriousness with the implementation of these procedures has come to generate disagreements, more specifically when entering the campus, since to enter, you have to put your finger on the fingerprint detector of the turnstiles, this creates confusion since it is a bit inconsistent that biosecurity protocols are so strict, but when you enter the campus you have to put your finger where much more people have put it before. It is important to solve this problem since it would reduce the possibility of contagion of the virus and would make the actions carried out by the university to help combat the spread of the virus more coherent.

**SOLUTION**

After analyzing the problem, the team proposed to implement a facial recognition system, with this, people will not have to come into physical contact with any device to be able to make their recognition as a member of the university effective. Before a person enters the campus, a photo would be taken, with which the program would be in charge of comparing their facial features using PCA and a neural network trained with the dataset (dataset generated from information from exclusively members of the university ), and if the person is really a member of the university, the software will recognize them and the guard person will give the go-ahead to let this person in.

**IMPLEMENTATION**

For the implementation of this solution the following libraries were used:

* matplotlib.pyplot
* from sklearn.model\_selection import train\_test\_split
* from sklearn.datasets import fetch\_lfw\_people
* from sklearn.metrics import classification\_report
* from sklearn.decomposition import PCA
* from sklearn.neural\_network import MLPClassifier

**Matplotlib.pyplot**

It is a collection of functions that make matplotlib work like MATLAB.

**Train\_test\_split**

It is a function in Sklearn model selection to split data arrays into two subsets: one for training data and the other for test data, with this function you don't need to manually split the dataset.

**Classification\_report**

Build a report showing the main ranking metrics.

**Fetch\_lfw\_people**

Load the people dataset named “Labeled Faces in the Wild” (LFW).

**PCA**

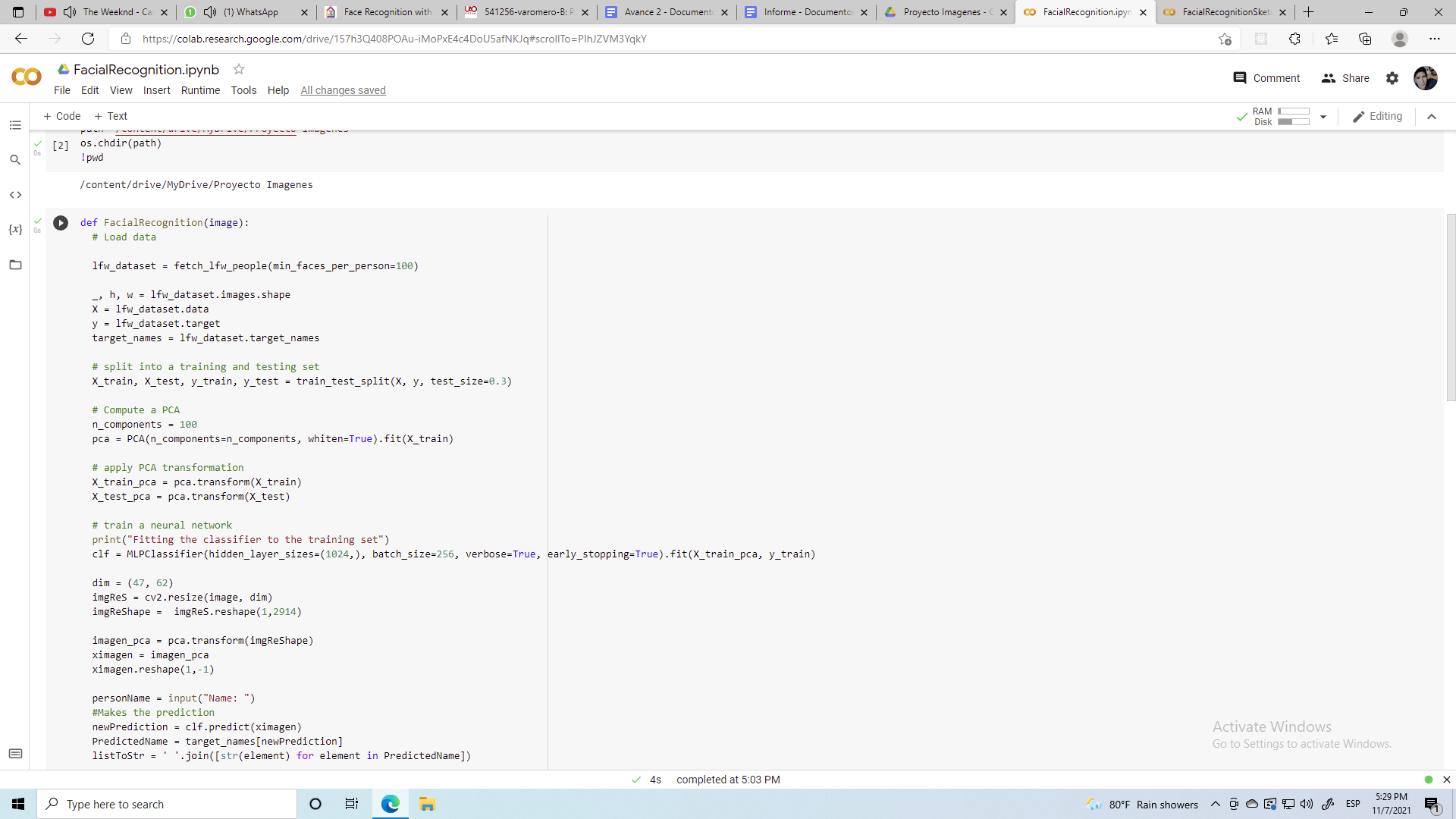
Linear dimensionality reduction using singular value decomposition of the data to project the data into a lower dimensionality.

**MLPClassifier**

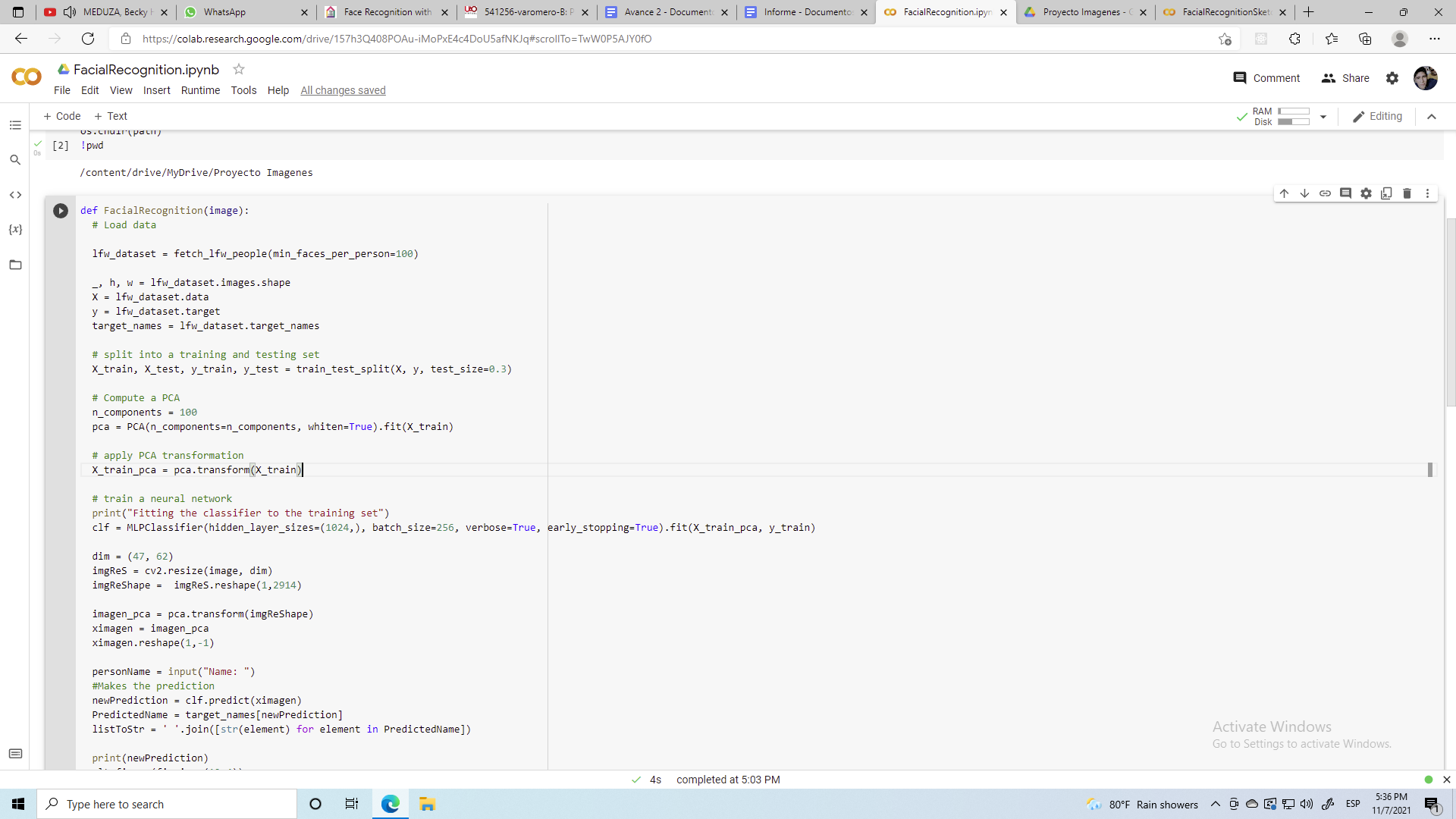
Multilayer Perceptron Classifier.

**CODE AND RESULTS**

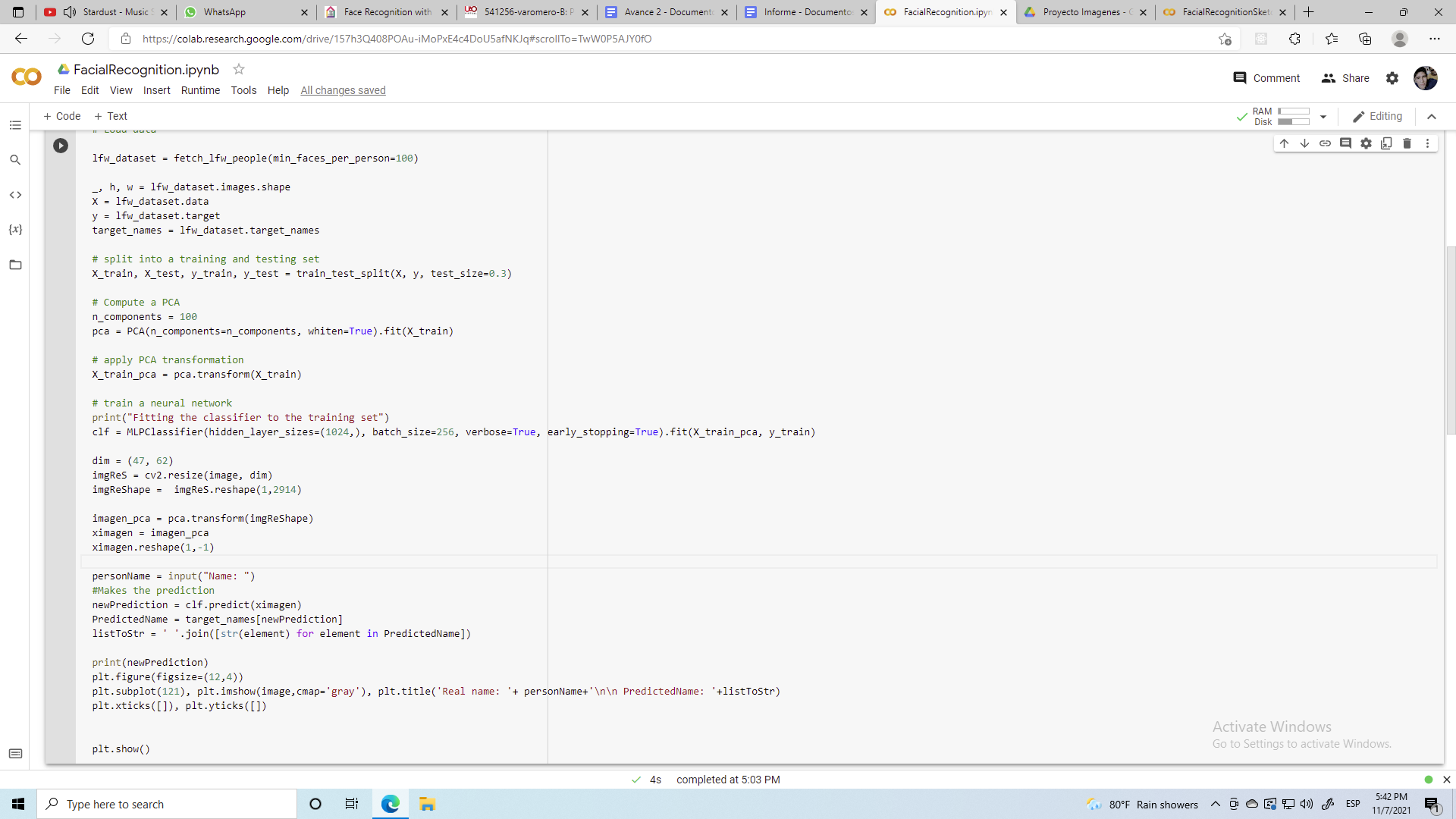
Initially we define our function that will be in charge of carrying out the entire facial recognition process. In this case we call it FacialRecognition to which we pass as a parameter the image that we are going to compare with the dataset, we will see this later. Inside this function we start by loading the dataset that we are going to use and split it into its various components (so we will have the images and the corresponding names separately), then using Train\_test\_split, we split the dataset in two, one for training and the other for tests.



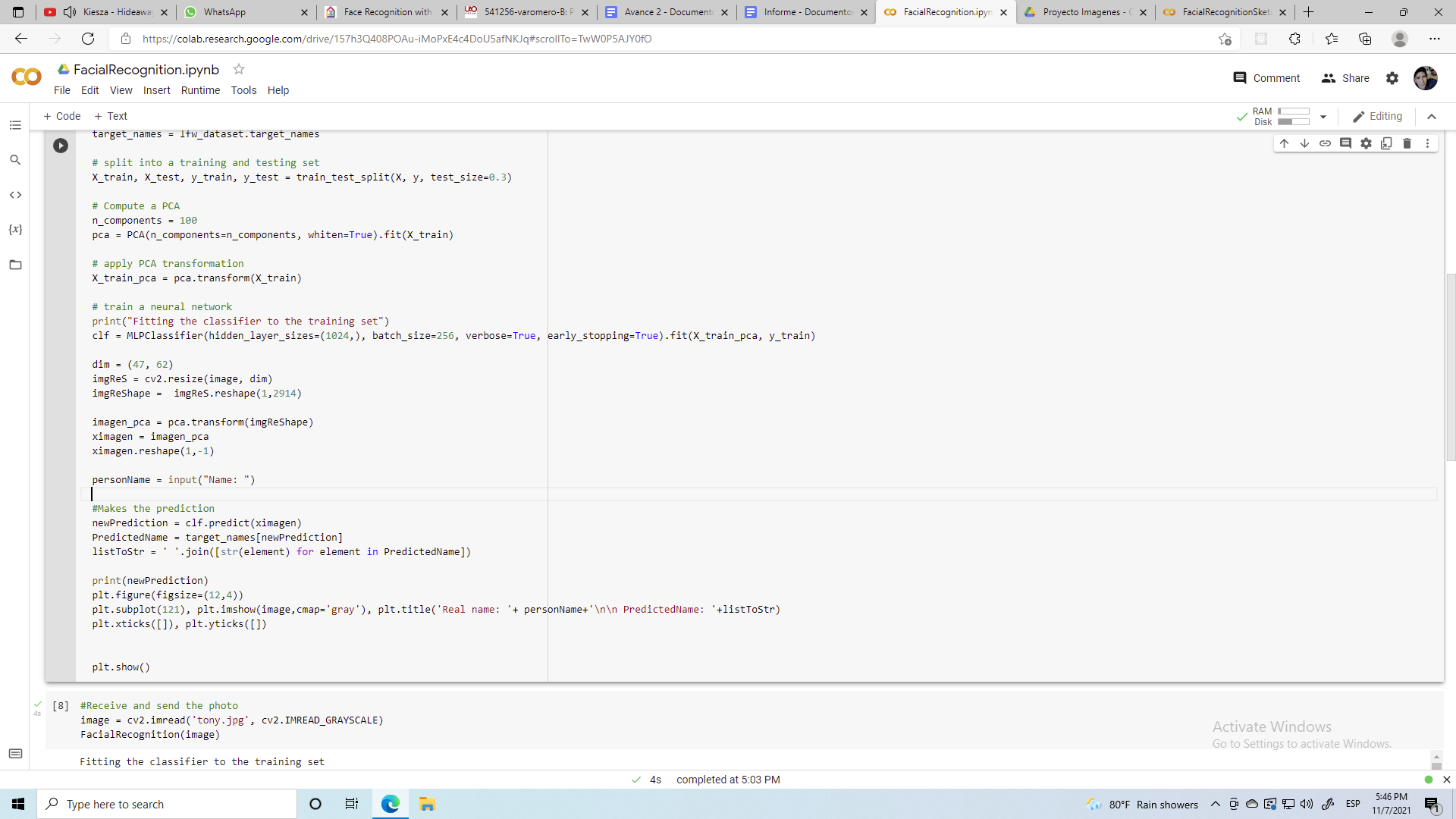
We then compute PCA, define 100 components and using **PCA** from **sklearn.decomposition** to train this PCA on the training dataset (obtained above). With the pca trained, we apply the PCA transformation to the training dataset, that is, we are going to reduce its dimensionality.



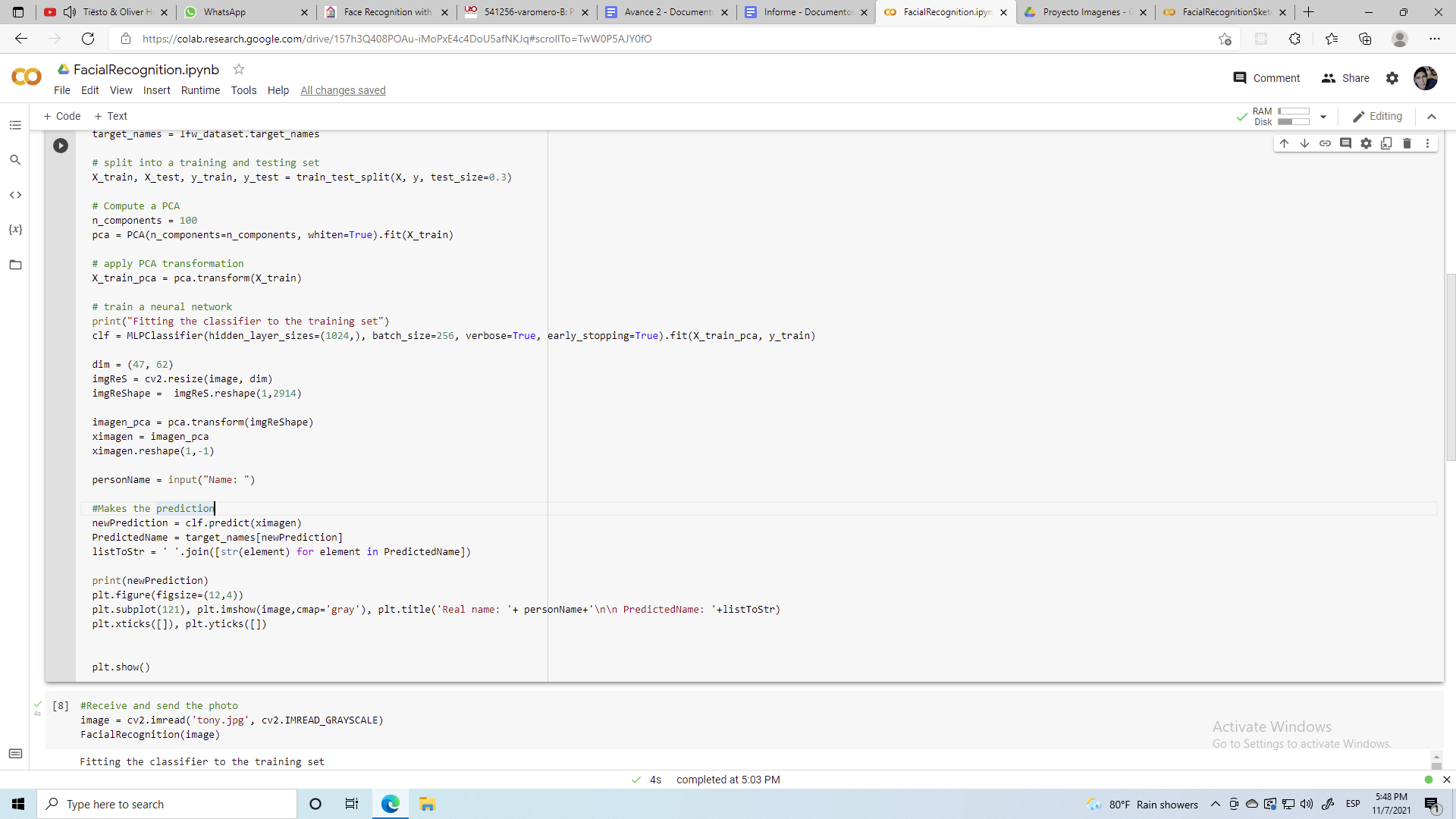
Next we go on to train the neural network with the help of **MLPClassifier**, we train this neural network with the divided training dataset with its reduced dimensionality, that is, the newly created variable X\_train\_pca, and the y\_train component of the dataset (obtained previously using the **Train\_test\_split**).



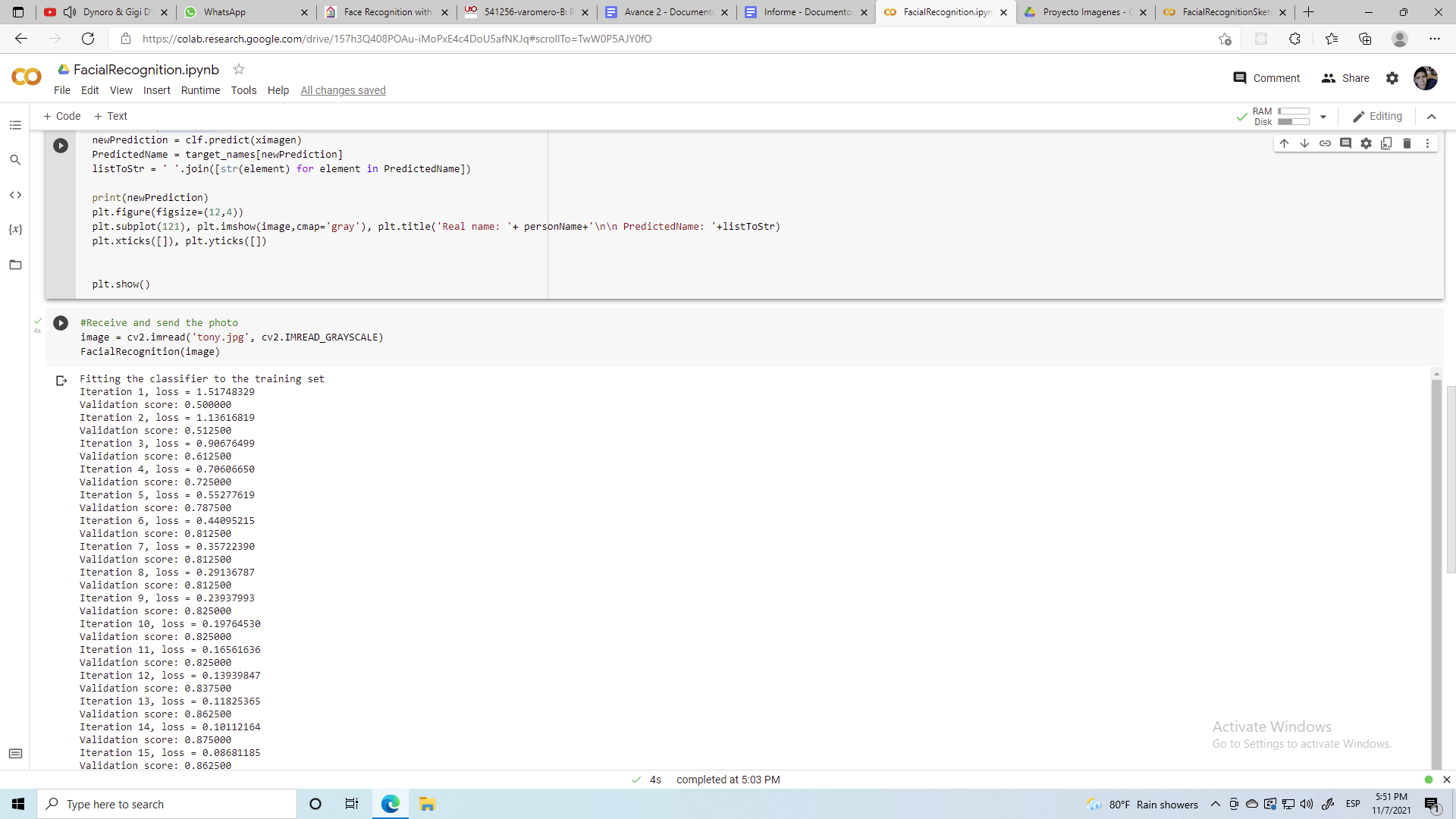
The only thing left is to pass the image on, and call the neural network to make the prediction and identify the person. To correctly pass the image and make it usable by the neural network, you have to apply PCA to the image, and to be able to do this you have to resize and reshape the image, in such a way that it is equal to the dimensionality of the image. dataset with which the PCA was trained.

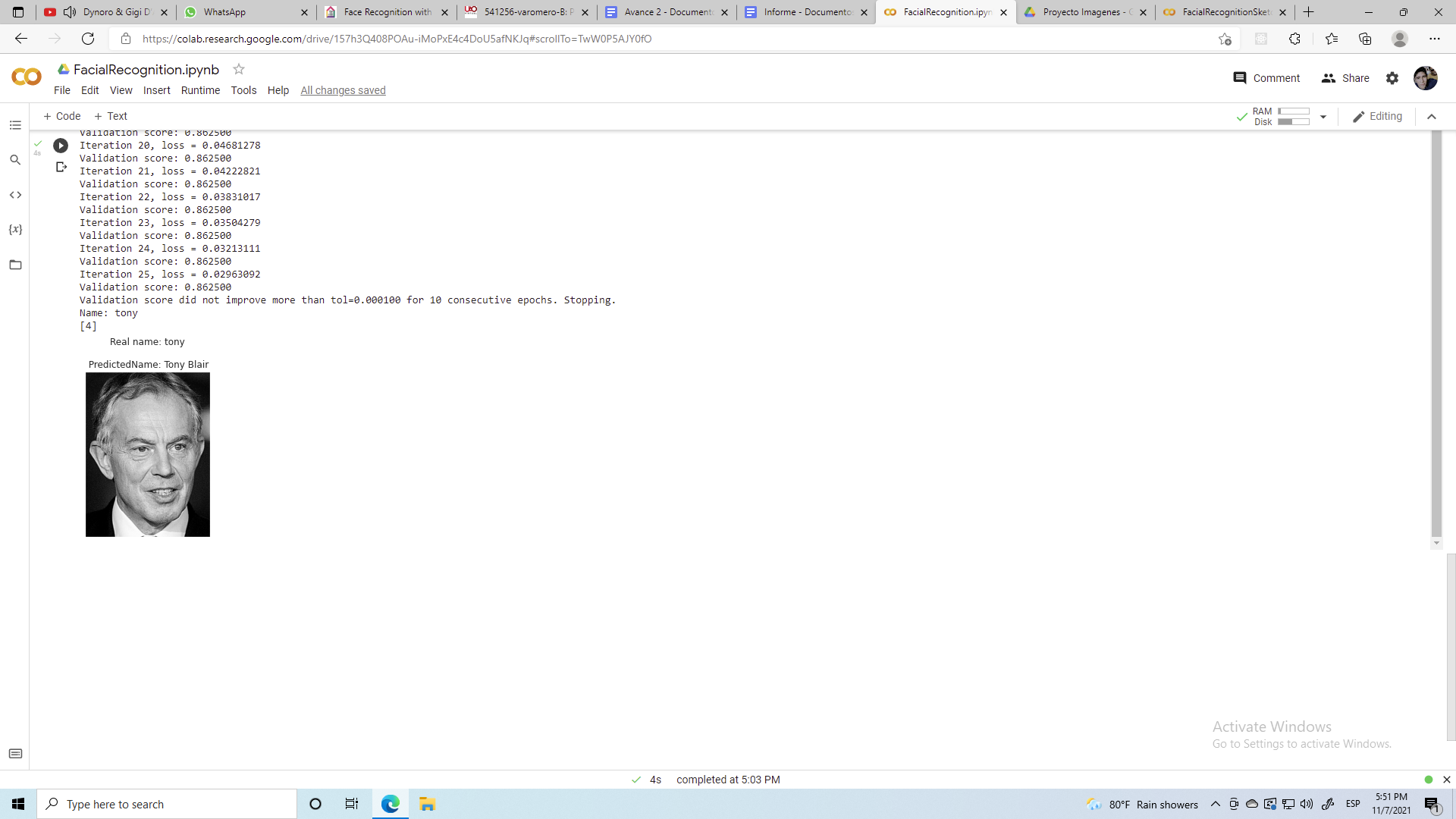


Once this is done, the image is passed to the neural network, we capture the prediction in a variable and print the results.



In order to make the entire process described above effective, the function is called and the image we want to compare is passed as a parameter. The result is the image that we pass as a parameter, the name of the person who is going to enter, and finally the prediction of the name that the neural network made.





**PCA DISADVANTAGES**

At the time of implementing the solution, we realized that when entering images of other presidents who should be recognized, this one did not. After researching about the PCA method and its scope, we found several disadvantages that apply in this situation:

* Independent variables become less interpretable: After implementing PCA on the dataset, the original features become principal components. The main components are the linear combination of the original features and these are not as readable and interpretable as the original features.
* The dimensionality of the images is very large for methods for the multilayer perceptron and classical machine learning methods such as PCA.
* Although PCA tries to cover the maximum variance between the features in the dataset, if we do not select the number of principal components carefully, it may lose some information when compared to the original list of features.

Additionally, the recognition accuracy also depends on the training dataset, the more information about a person you have, the more effective the recognition will be, which happened with the fetch\_lfw\_people dataset, which has more information about some presidents than others, it is the In the case of President George Bush, about whom there is more information than about other presidents, when entering an image different from George's, in some cases, the neural network predicted (erroneously) that the image provided was of him, that is , the neural network is biased since the training dataset has more information about some presidents than others.

References

* *LFW Face Database : Main*. Vis-cs.umass.edu. (2021). Retrieved 9 November 2021, from <http://vis-www.cs.umass.edu/lfw/>.
* *Face Recognition with Eigenfaces – Python Machine Learning*. Pythonmachinelearning.pro. (2021). Retrieved 9 November 2021, from <https://pythonmachinelearning.pro/face-recognition-with-eigenfaces/>.
* *sklearn.model\_selection.train\_test\_split*. scikit-learn. (2021). Retrieved 9 November 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>.
* *sklearn.metrics.classification\_report*. scikit-learn. (2021). Retrieved 9 November 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>.
* *sklearn.decomposition.PCA*. scikit-learn. (2021). Retrieved 9 November 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>.
* *sklearn.neural\_network.MLPClassifier*. scikit-learn. (2021). Retrieved 9 November 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html.