Surgical mask detection

Machine Learning Project

Author : Badea Adrian Cătălin, group 235

Competition link: Kaggle link

Short Description

In this report, I will make a short description about audio data features extraction, as well as best models available from Scikit-learn, easily fitted for the given set's samples. The first remarks will be about *"tester.py",* which will be found in the archive, along with *"main.py"*, which was used for the validation dataset.

As accessing the files is very easy with python, the paths to the given files were put in a list using list comprehensions. Afterwards, the function librosa.load was used for accessing the data. Then, other functions from the library librosa were used in order to convert the audio file to a dB-scaled spectrogram, as follows. An ensemble of the best suited classifiers were put together in order to maximize the accuracy, which we will discover shortly.

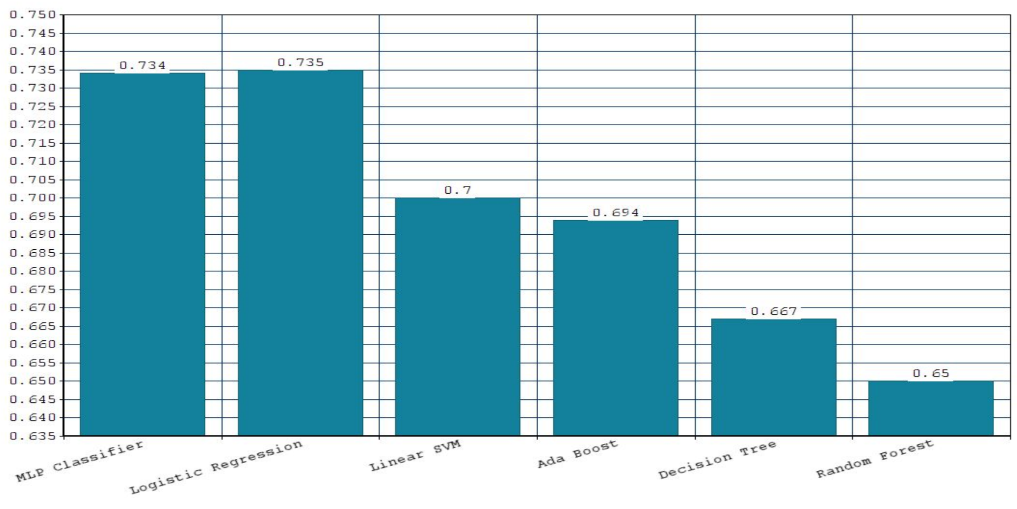
Audio Files

Every audio file was loaded and transformed into a spectrogram using librosa.stft with parameters such as n_fft and win_length as 8192 (one of the biggest powers of two, strictly smaller than the sampling rate(22500).

After loading the data into memory, I converted the amplitude spectrogram to a dB-scaled spectrogram, in which the features we had in mind were shown better.

Classifiers

The initial ensemble of classifiers was put together. Having chosen the best from more than 10 classifiers, some which were not productive at all, I gathered the list which was more suited for audio data.



After more submissions, the ones that remained on spot were *Linear SVM* and *Adaboost.* MLP classifiers probably overfitted resulting in 0.615 on submission, due to the big amount of data. The ensemble used at first was :

```
 LogisticRegression(),
SVC(kernel="linear", C=0.025),
DecisionTreeClassifier(max_depth=7),
RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
MLPClassifier(alpha=1, max_iter=1000),
AdaBoostClassifier(),
```

For the best submissions, the classifiers used were *Linear SVM, Decision Tree* and *AdaBoost.* In the next paragraphs I will present the best two, the first and the last.
The confusion matrix for the ensemble at validation: (On the first column there are the true zeros, and the second column the true ones.

out-marked by 0 classifiers : ***158***, 19
out-marked by 1 classifier: ***129***, 57
out-marked by 2 classifiers: 112, ***160***
out-marked by all 3 classifiers: 73, ***292***
So, the total precision is (158 + 129 + 160 + 292) / 1000 = 0.739

Linear SVM

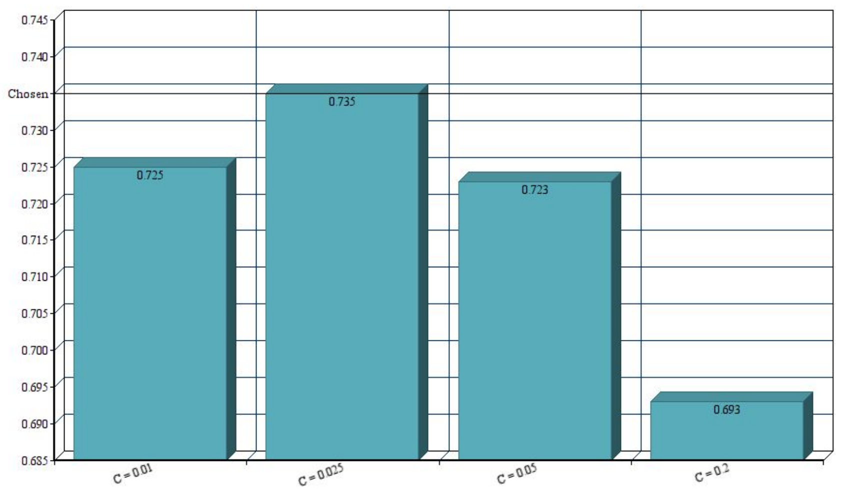
Linear SVM, was faster than a non-linear one, and proved to have a close accuracy. The linearity of the classifier meant being able to model more times with different types of C in a smaller amount of time. A very small value of C would cause the optimizer to look for a larger-margin separating hyperplane, as the strength of the regularization is inversely proportional to C.

The Confusion matrix for validation :

323, 150
149, 378

So, the recall was ~ 378 / 528 = 0.71, and the precision is ~ 0.378 / 0.527 = 0.71
Then, the F1 score is also, 0.71.

For optimization of the classifier on validation set, I used *Grid-Search* in order to outline the best C values, as follows:



Remarkably, all the chosen C values were very close on accuracy, but still the winner was *C = 0.025.* It can not be surely stated that the best hyperparameter for the validation data is the same for the testing data, as there was a slight difference in these sets.

Adaboost

Adaboost was really fast, fitting in approximately 15 minutes. Seeing as this classifier is itself an ensemble of classifiers with weights, the accuracy went up with only 0.003, when Decision Tree, and Linear SVM were taken into account. Here, individual classifiers vote and a final prediction label is returned by performing majority voting.
It is quite a popular boosting technique which helps you combine multiple “weak classifiers” into a single “strong classifier”. A weak classifier is simply a classifier that performs poorly, but performs better than random guessing. A simple example would be classifying a person as male or female based on their height. Seeing how this classifier is making its way up in the kaggle competitions, I imagined it would be worth a try.

Confusion matrix at validation:

305, 150
155, 390

So, the recall was 390 / 540 ~ 0.72, and the precision 390 / 545 ~ 0.715
Then, the F1 becomes, also : 0.71, very close to both precision and recall.
Here, the number of estimators are default, as no system message was provided when modeling the classifier, which led me to the idea it was fully fitted.

Additional Information:

Github project link
Audio manipulation link
Email address : ADRIAN-CATALIN.BADEA@my.fmi.unibuc.ro