Project 2 : Automatic classification of skin lesions (melanoma detection)

ESSAYEGH Nour

December 2020

The objective of this project is to automatically detect skin lesions, using multiple machine learning algorithms.

1 Database

1.1 Exploration of the database

The data is divided into 3 categories. We have the original image, the binary image and the segmentation of images using superpixels. Our database is composed of 200 samples, which are part of an original database from the world wide challenge: "ISIC 2017 : Skin Lesion Analysis Towards Melanoma Detection".

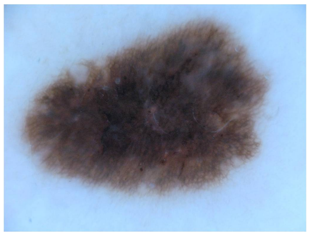


Figure 1: Original image

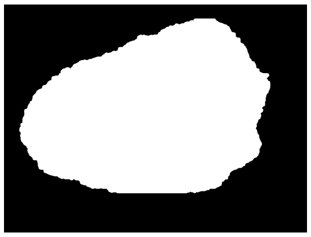


Figure 2: Binary image

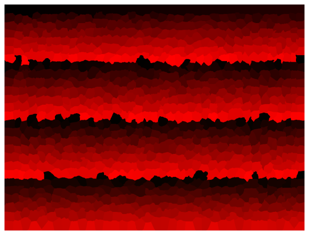
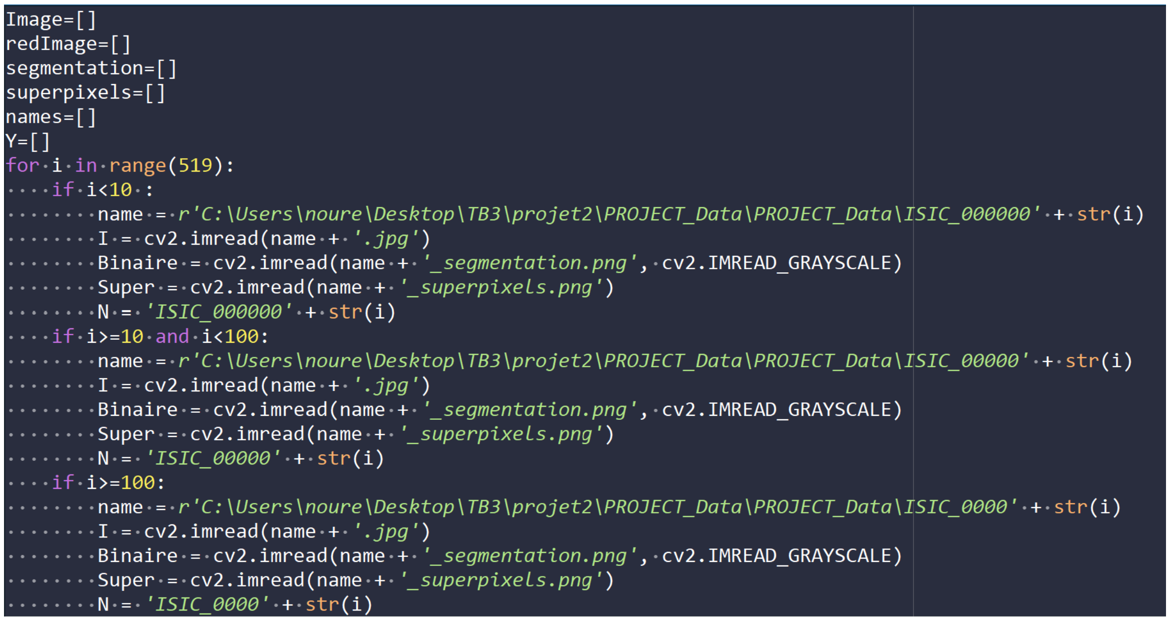


Figure 3: Superpixels

We also have at our disposition an Excel file where we can extract all the labels to determine if the image is a melanoma or not. In our case, we have to extract the labels using the names of the original images.

1.2 Loading the Database

We tried to extract the images into 3 lists depending on their nature (Original, segmented or superpixel image). We used the library cv2 to extract the images without having to deal with the missing images in the automatic process we put in place to load the Data.



```python
Image=[]
redImage=[]
segmentation=[]
superpixels=[]
names=[]
Y=[]
for i in range(519):
    if i<10 :
        name = r'C:\Users\noure\Desktop\TB3\projet2\PROJECT_Data\PROJECT_Data\ISIC_000000' + str(i)
        I = cv2.imread(name + '.jpg')
        Binaire = cv2.imread(name + '_segmentation.png', cv2.IMREAD_GRAYSCALE)
        Super = cv2.imread(name + '_superpixels.png')
        N = 'ISIC_000000' + str(i)
    if i>=10 and i<100:
        name = r'C:\Users\noure\Desktop\TB3\projet2\PROJECT_Data\PROJECT_Data\ISIC_00000' + str(i)
        I = cv2.imread(name + '.jpg')
        Binaire = cv2.imread(name + '_segmentation.png', cv2.IMREAD_GRAYSCALE)
        Super = cv2.imread(name + '_superpixels.png')
        N = 'ISIC_00000' + str(i)
    if i>=100:
        name = r'C:\Users\noure\Desktop\TB3\projet2\PROJECT_Data\PROJECT_Data\ISIC_0000' + str(i)
        I = cv2.imread(name + '.jpg')
        Binaire = cv2.imread(name + '_segmentation.png', cv2.IMREAD_GRAYSCALE)
        Super = cv2.imread(name + '_superpixels.png')
        N = 'ISIC_0000' + str(i)
```

The if inside the loop detect the nonexistent images so that we won't add them to our lists. We also saved the names of our images to extract the labels and save them in the Y vector.

2 Feature extraction

Before testing the classification methods we have to describe our data and gather information to differentiate between a healthy mole and a melanoma. Since the 4 ground rules for the detection of melanoma are asymmetry, border irregularity, color irregularity and differential structure we have to split the feature extraction into 2 categories geometrical/morphological features and intensity/texture descriptor.

2.1 Geometrical/morphological features

In this section, we will be extracting some geometrical features from the segmented images since understanding shapes prove to be a useful tool. We have to start to identify the features that would describe the best our segmented images and help us detect skin lesions. Before working on the images we have to make sure they all have the same resolution so that the transformations based on the number of pixels will be accurate.

We chose to work with 9 features: area, eccentricity, parameter, diameter, extent, the difference between the Feret diameters, elongation, roundness, and circularity.

We used regionprops as seen in the first project to determine the area, perimeter, diameter, extent and eccentricity. To complete the shape description and evaluate the asymmetry of our binerazed images we used the difference between the Feret diameters, the elongation, the roundness (1) of our lesion and the circularity (2).

$$Roundness = \frac{4A}{\pi D^2} \tag{1}$$

$$Circularity = \frac{4\pi A}{P^2} \tag{2}$$

With A the area, P the perimeter and D the Feret Diameter.

Conclusion : We know have an X_{bin} that sums up the geometrical and morphological features of our binerazed images.

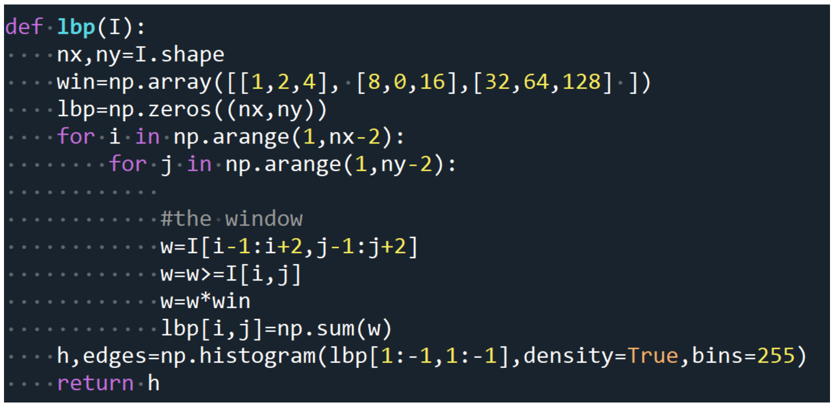
2.2 Intensity/texture descriptors

2.2.1 Local Binary Patterns descriptor

The Local Binary Patterns descriptor is a very efficient texture and intensity operator. It labels the pixels of an image by thresholding the neighborhood of each pixel abs considers the result as a binary number. In our function file, we defined an LBP function to compute the frequency of each number occurring in each of the red component of the original image since the texture description is the same for the green and blue components as well.

This step is the one step that takes a considerable amount of time to run all the 200 images and that is due to the structure of our function (2 loops) and the fact that our images have a high resolution (767x1022 and higher). To make the running time less significant we can reshape our images into a much smaller resolution, but we could risk the loss of important information linked to the texture.

In our program, we reshaped the images to a size 767x1022 so that the running time for our function would be equal for all the images. We also printed the loop number to track the evolution since it can take a considerable amount of time.



```python
def lbp(I):
    nx,ny=I.shape
    win=np.array([[1,2,4], [8,0,16],[32,64,128] ])
    lbp=np.zeros((nx,ny))
    for i in np.arange(1,nx-2):
        for j in np.arange(1,ny-2):
            
            #the window
            w=I[i-1:i+2,j-1:j+2]
            w=w>=I[i,j]
            w=w*win
            lbp[i,j]=np.sum(w)
    h,edges=np.histogram(lbp[1:-1,1:-1],density=True,bins=255)
    return h
```

Figure 4: LBP function

Conclusion : We know have an $X_{texture}$ that regroups all the frequencies of each number occurring in all our 200 images.

2.2.2 Intensity extraction for the superpixels images

We extract from the red and green components another feature to evaluate the color irregularity of our images and with that, we combined all the different features and constructed our descriptive data. Since the data we gathered is quite important and some feature might be more important than other we decided to shorten the running time if our classifier to select 200 best-describing features of our data using the function $data = SelectKBest(chi2, k = 200).fit\_transform(X, Y)$ This function selects the 200 best features with the higher chi-squared stats and returns a new X with 200 features.

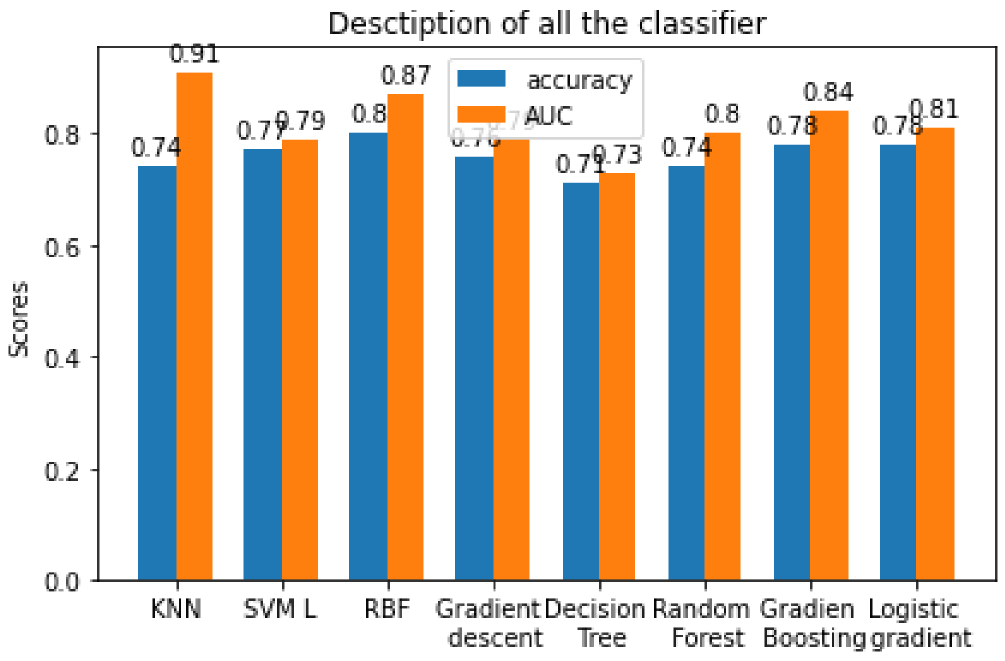
3 Classification

In this section, we will be comparing different classification algorithms to determine which one gives the best predictions. For that, we will be dividing our data sets into 2 categories, the training set (75% of the data) and the testing set (25%). We will be using the accuracy, the F score, and the ROC curve to determine which one classified the best our testing set.

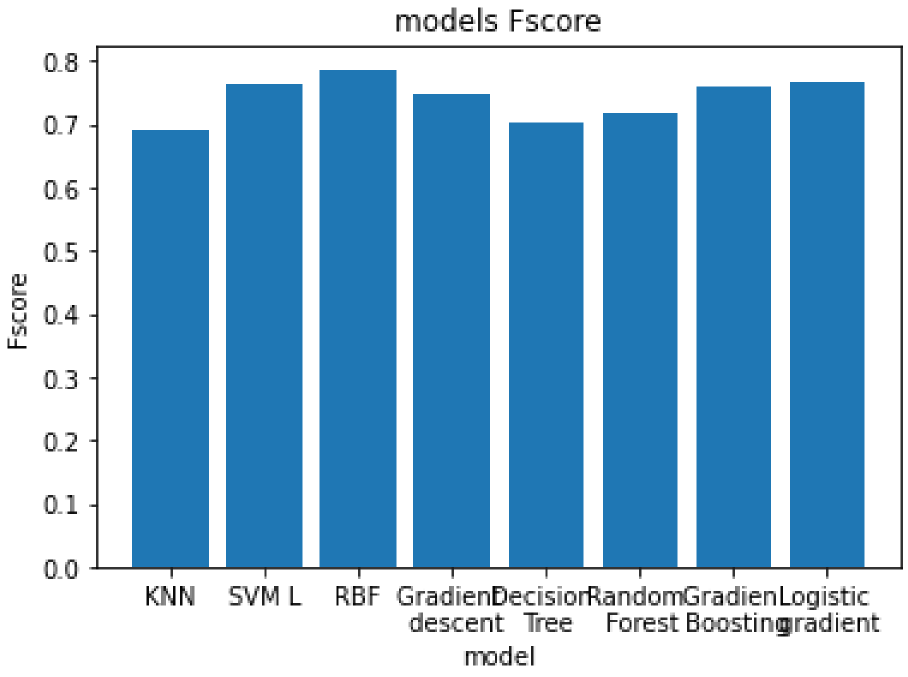
We chose to call different classifiers from the library skealrn and we used the grid search to determine the optimal parameter of our classifiers so that it will suit our data the best way possible and avoid the overfitting.

The classifiers we used are K nearest neighbors, linear svm and RBF svm, stochastic gradient descent, Decision tree, random forest, gradient boosting and Logistic gradient.

We tried our classifiers on 100 samples to extract the the mean of the accuracy, F_score, and the area under the curve ROC.



We can see from this plot that the best classifier regarding the accuracy and the area under the curve ROC is the SVM using RBF. The F_score diagram is in coherence with the result we obtained.



We can still explore other classification methods with a more technical aspect as the neural networks for example. This deep learning method is a good classifier but is much more complicated to put in place.

4 Conclusion

The gathering of the features from our images is the most important step of our classification since we can test multiple classifiers with different parameters. We should also think of a way to select the best features. The Kbestfeature method is an efficient way, but we can also study the correlation between the features and select fewer features without losing information.

An accuracy of 80% is good yet not sufficient regarding the seriousness of our problem since it's very dangerous to miss diagnose a melanoma tumor as a healthy mole. That is why we need to explore more classifiers to find the best one and avoid over fitting.