Waseem Weshahi
206943391

Machine learning
Assignment 1

Bayan Farhan
208300145

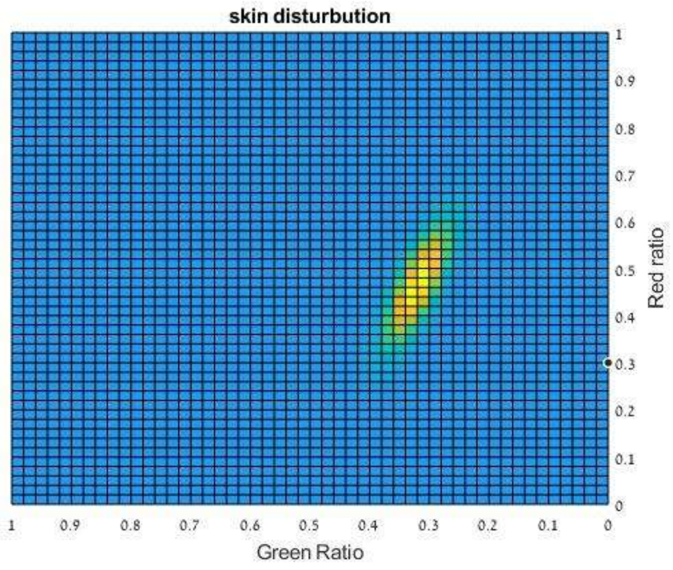
Exercise 1:

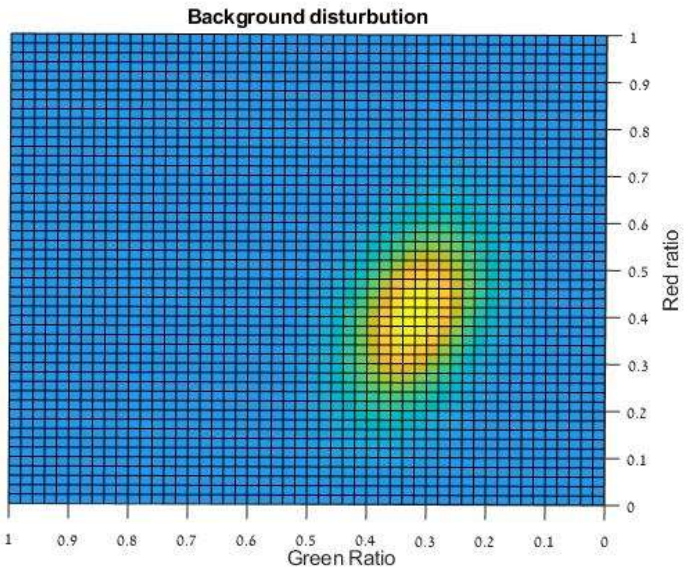
Skin Detection using MAP classifier

- First Part: Learning the Probabilities of the skin and background models:

We get a data which consists of the red and green ratios of the pixel (which is our 2x1 feature vector), we then learn the distribution of those features across skins and backgrounds, which is done using what we learned in class about simply averaging the feature vectors each class and calculating the sigma.

The distributions turn out to be :





Then we send the likelihood ratios of the validation set to compute the ROC curve and chose the optimal threshold.

Now we have tried choosing thresholds in many different approaches but only three seemed to work:

I) The "Trivial" threshold of choosing the threshold that minimizes the TPR and TNR gap

II) Choosing the threshold that leads to the closest point to the left top edge of the ROC graph

III) Computing the ROC the following way:

```
[~, r]=min(abs(skinPrior*ROC(:,1)-bgPrior*ROC(:,2)));
eer = ROC(r,:);
th = eer(3);
```

The idea behind this being that, we have a lot more background pixels than skin pixels (skinPrior being 21.21% and bgPrior 78.79%) meaning that we would like to guess more backgrounds correctly because guessing background falsely would be too obvious (there's a lot of background, so a lot of room for error) and not guessing skins right is more tolerable since there aren't that many of skin pixels.

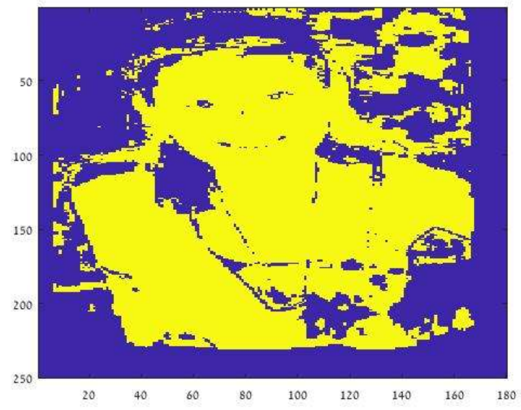
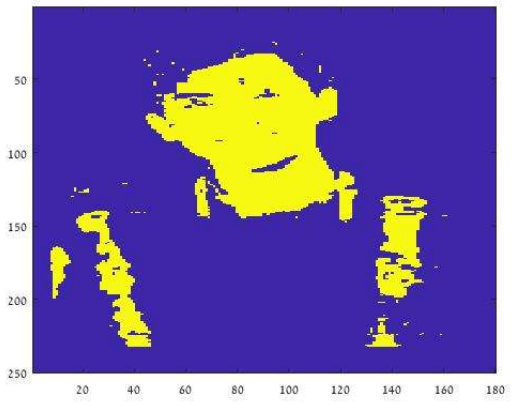
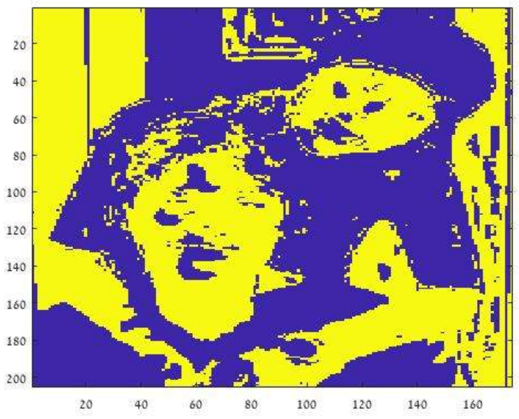
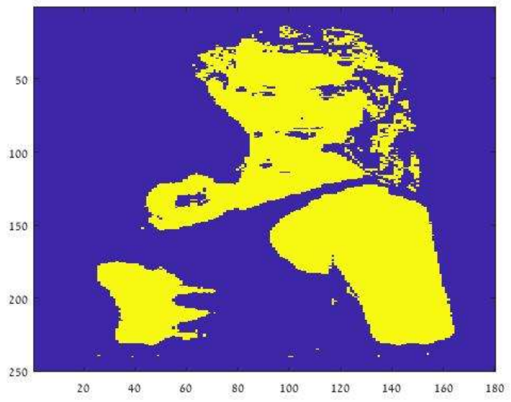
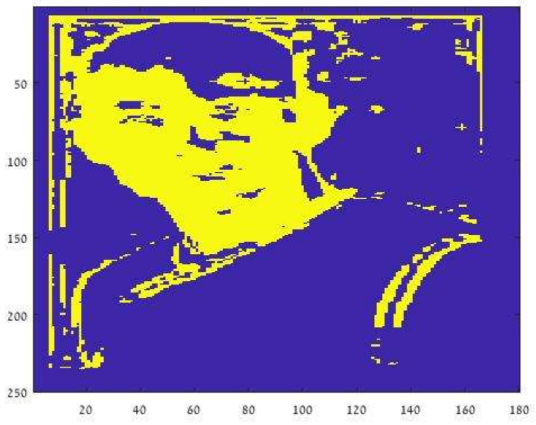
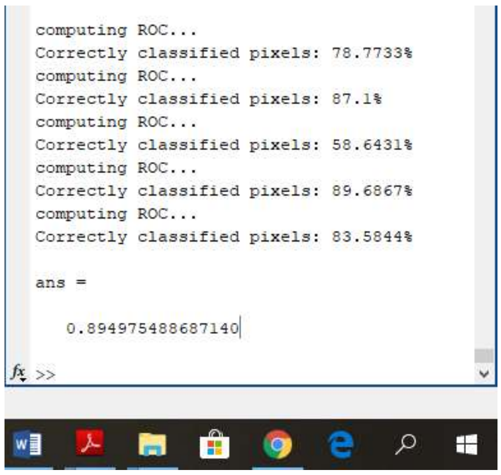
- Second Part: Classifying images

in This part we run over the each individual pixel and check the likelihood ratio for it being a skin over the probability of it being a background and If the likelihood ratio turned out to be bigger than the threshold we assume it's a skin, otherwise it's a background.

- Results:

We ran the classifier on 5 images, results were as follow:

- Choosing the trivial Threshold:

```
computing ROC...
Correctly classified pixels: 78.7733%
computing ROC...
Correctly classified pixels: 87.1%
computing ROC...
Correctly classified pixels: 58.6431%
computing ROC...
Correctly classified pixels: 89.6867%
computing ROC...
Correctly classified pixels: 83.5844%

ans =

   0.894975488687140

fx >>
```

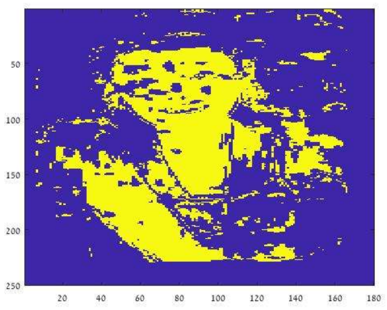
** the second way of choosing threshold (the closest point to the top left corner) has lead to a very similar results to the first way of choosing the threshold, the only difference being a couple of percentages more of correctly classified pixels.

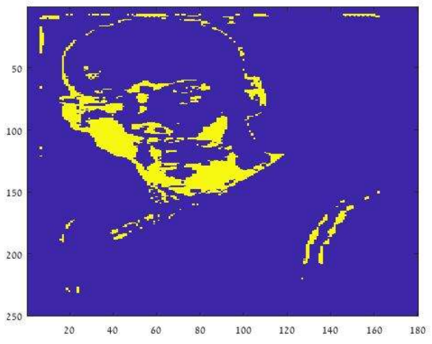
Waseem Weshahi
206943391

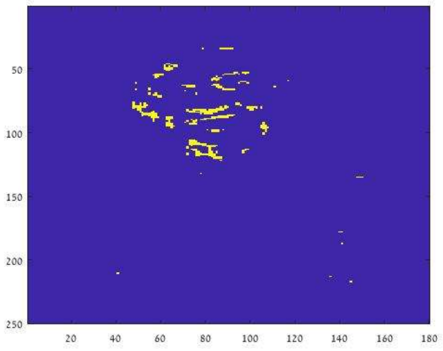
Machine learning
Assignment 1

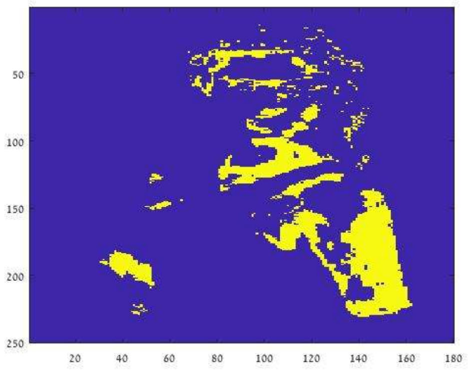
Bayan Farhan
208300145

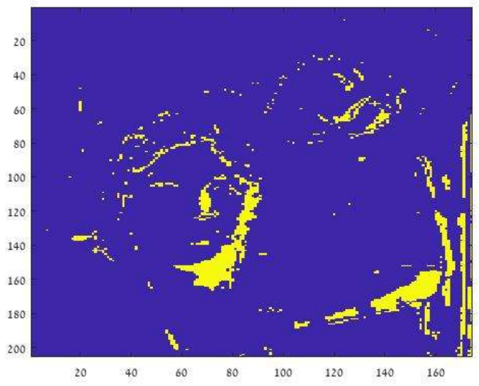
❖ Choosing the threshold that counts for the priors of the classes:

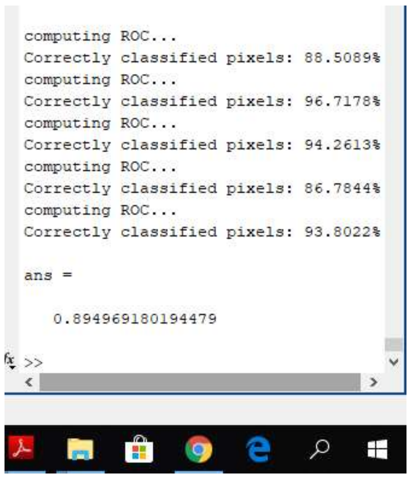












```
computing ROC...
Correctly classified pixels: 88.5089%
computing ROC...
Correctly classified pixels: 96.7178%
computing ROC...
Correctly classified pixels: 94.2613%
computing ROC...
Correctly classified pixels: 86.7844%
computing ROC...
Correctly classified pixels: 93.8022%

ans =

   0.894969180194479

>>
```

As we can see from the results, choosing the trivial threshold leads to a "*better visual*" detection of skin, while the other threshold classifies a bigger *percentage* of pixels correctly but it looks off.

Waseem Weshahi
206943391

Machine learning
Assignment 1

Bayan Farhan
208300145

Exercise 2:

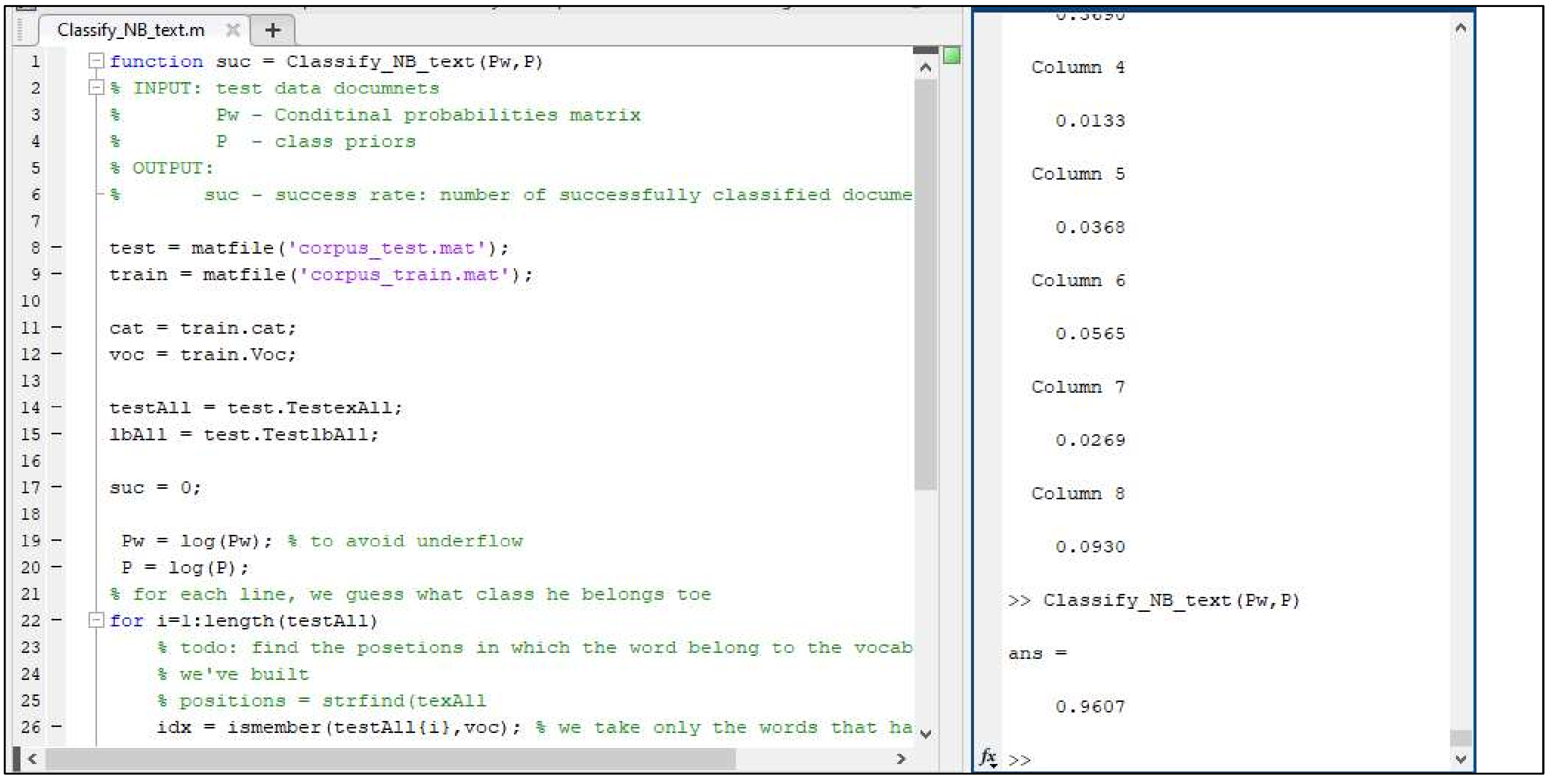
Text Classification Using Naïve Bayes Algorithm

In the Attached .m Files we can see the implementation of the Naïve Bayes algorithm for learning and classifying the category of given a text, under the assumptions of the “Bag of words Model” that we learned in class (independence between words and their positions in the document).

After learning the probabilities of words and categories using the training text. we ran the classifier on the testing text, and the success rate turned out to be:

Success Rate: 96.07%

Success rate in `format long`
0.960712654179991


Classify_NB_text.m

```
function suc = Classify_NB_text(Pw,P)
% INPUT: test data documnets
%         Pw - Conditinal probabilities matrix
%         P  - class priors
% OUTPUT:
%        suc - success rate: number of successfully classified docume

test = matfile('corpus_test.mat');
train = matfile('corpus_train.mat');

cat = train.cat;
voc = train.Voc;

testAll = test.TestexAll;
lbAll = test.TestlbAll;

suc = 0;

 Pw = log(Pw); % to avoid underflow
 P = log(P);
% for each line, we guess what class he belongs toe
for i=1:length(testAll)
    % todo: find the posetions in which the word belong to the vocab
    % we've built
    % positions = strfind(texAll
    idx = ismember(testAll{i},voc); % we take only the words that ha
```

```
    0.3690

  Column 4

    0.0133

  Column 5

    0.0368

  Column 6

    0.0565

  Column 7

    0.0269

  Column 8

    0.0930

>> Classify_NB_text(Pw,P)

ans =

    0.9607

>>
```