Machine Learning
(BITS F464)

Assignment 3
Naive Bayes Classifier

Team Members:

- **Kartik Srivastava** **2014B4A7755H**
- **Abhishek Singh** **2014B4A7752H**
- **Raj Aditya Kumar** **2014B4A7582H**
- **Shubham Jain** **2014A7PS085H**

Training the faces

The each of 70x60 pixels of each of the images were analysed and the following probabilities were computed for each pixel:

- P('#' | image=face)
- P('#' | image!=face)
- P(' ' | image=face)
- P(' ' | image!=face)

Also along with these probabilities, two more probabilities were computed:

- P(image=face)
- P(image!=face)

Thus after calculating these probabilities, Naive bayes Classifier was used to predict whether each of the images were faces or not by computing the following probabilities :

$$P(\text{image=face} \mid \text{pixels}) = \left(\prod P(\text{pixel('\#' or ' ')} \mid \text{image=face})\right) P(\text{image=face})$$

$$P(\text{image!=face} \mid \text{pixels}) = \left(\prod P(\text{pixel('\#' or ' ')} \mid \text{image!=face})\right) P(\text{image!=face})$$

and classifying image as face or not a face according to the greater of the two calculated probabilities.
To prevent the values of P(image=face | pixels) and P(image!=face | pixels) from reaching zero due to multiplication of individual probabilities, each of the probabilities P(pixel('#' or ' ') | image=face) or P(pixel('#' or ' ') | image! =face) were multiplied by a constant value of **1.42** .

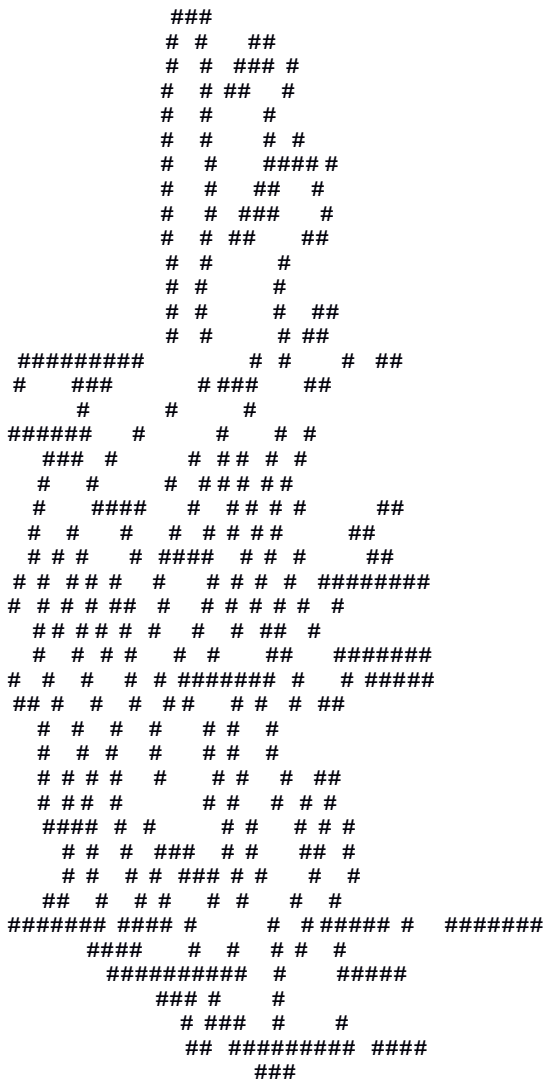
Confusion Matrix :

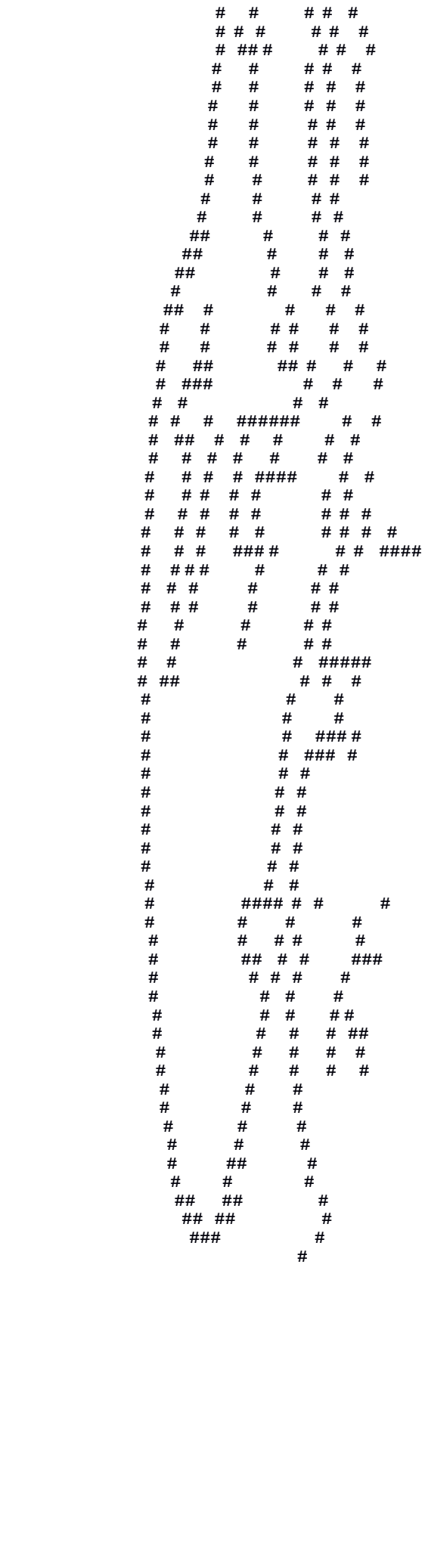
Classification / Truth	True	False
Positive	63	6
Negetive	71	10

Accuracy: 89.33%

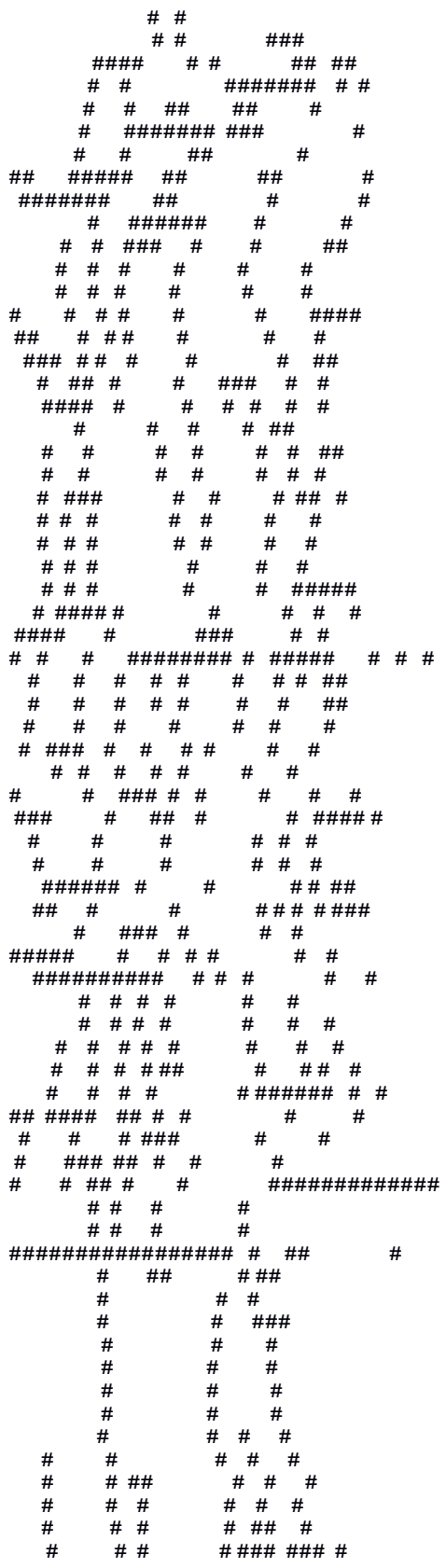
Examples of Misclassification

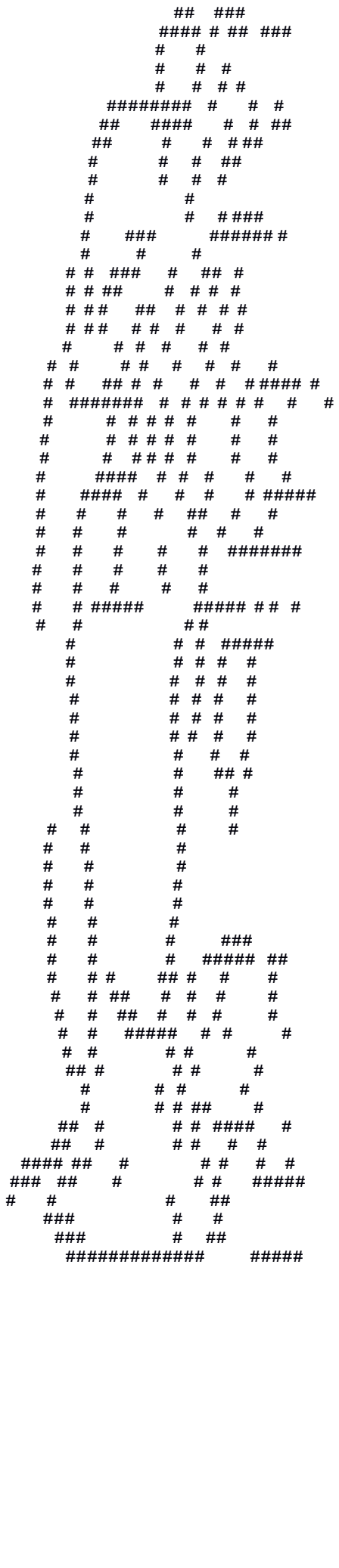
False Positive





False Negetive





Code:

```
import java.io.*;
class NaiveBayes
{
     public static void main(String [] args)throws IOException
     {
          double P_h1[][]=new double[70][60];//array to store
P('#'|(image=face))
          double P_h0[][]=new double[70][60];//array to store
P('#'|(image=not face))
          double P_s1[][]=new double[70][60];//array to store P('
'|(image=face))
          double P_s0[][]=new double[70][60];//array to store P('
'|(image=not face))
          double acc=0;// to store accuracy
          double s_const=1.42;// smoothing constant to prevent the
probabilities from reaching zero
          int conf_mat[][]=new int[2][2];// confusion matrix
          double P_1=0;// to store probability of face images in
the training data
          double P_0=0; // to store the probility that image is not
of a face in the training data
          double P1=1.0;// probability that image to be tested is
of a face
          double P0=1.0;// probability that image to be tested is
not of a face
          int count=0;

          for(int i=0;i<2;i++)
          for(int j=0;j<2;j++)
          conf_mat[i][j]=0;//to initialize confusion matrix
elements to zero

          for(int i=0;i<70;i++)
               for(int j=0;j<60;j++)
               {
                    P_h1[i][j]=0.0;
                    P_h0[i][j]=0.0;
                    P_s1[i][j]=0.0;
                    P_s0[i][j]=0.0;
               }// to initialize the probablities to zero before
training

          FileReader tr = new FileReader("facedatatrain");//opening
the training image file
          BufferedReader br=new BufferedReader(tr);
          String tr_img=br.readLine();//stores each line of file

          FileReader tr_label=new
FileReader("facedatatrainlabels");// opening the training result
file
          BufferedReader br1=new BufferedReader(tr_label);
          String tr_res=br1.readLine();//stores result of image

          while((tr_img)!=null && (tr_res)!=null)
          {
               if(tr_res.equals("1"))
```

```
                P_1+=1.0;
                else
                P_0+=1.0;

                for(int i=0;i<70;i++)
                {

                     for(int j=0;j<60;j++)
                     {

                          char ch=tr_img.charAt(j);

                          if(tr_res.equals("1"))
                          {
                               if(ch=='#')
                               P_h1[i][j]+=1.0;
                               else if(ch==' ')
                               P_s1[i][j]+=1.0;
                          }
                          else if(tr_res.equals("0"))
                          {
                               if(ch=='#')
                               P_h0[i][j]+=1.0;
                               else if(ch==' ')
                               P_s0[i][j]+=1.0;
                          }

                     }

                     tr_img=br.readLine();
                }//to scan each image of each image and increment
probabilities accordingly

                count++;
                tr_res=(br1.readLine());
           }

           tr.close();
           tr_label.close();

           // computing probabilities
           for(int i=0;i<70;i++)
           {
                for(int j=0;j<60;j++)
                {
                     P_h1[i][j]/=P_1;
                     P_h0[i][j]/=P_0;
                     P_s1[i][j]/=P_1;
                     P_s0[i][j]/=P_0;
                }
           }


           P_1/=count;
           P_0/=count;

           FileReader test = new
FileReader("facedatatest");//opening the training image file
```

```
            BufferedReader br2=new BufferedReader(test);
            String ts_img=br2.readLine();//stores each line of file

            FileReader test_label=new
FileReader("facedatatestlabels");// opening the image testing file
            BufferedReader br3=new BufferedReader(test_label);
            String ts_res=br3.readLine();//stores the result of the
image

            count =0;

            System.out.println("Actual Output\tClassified Output");

            while((ts_img)!=null && (ts_res)!=null)
            {
                  P1=P_1;
                  P0=P_0;//initializing the probailities
                  int res=-1;

                  for(int i=0;i<70;i++)
                  {

                        for(int j=0;j<60;j++)
                        {
                              char ch=ts_img.charAt(j);
                              switch(ch)
                              {
                                    case '#': P1=P1*P_h1[i][j]*s_const;
                                              P0=P0*P_h0[i][j]*s_const;
                                              break;

                                    case ' ': P1=P1*P_s1[i][j]*s_const;
                                              P0=P0*P_s0[i][j]*s_const;
                                              break;
                              }


                        }//computing probabilities P(face|data) and
P(not face|data)

                        ts_img=br2.readLine();
                  }

                  if(P1>P0)
                  res=1;
                  else
                  res=0;//computing result based on the probabilities

                  if(res==0 && ts_res.equals("0"))
                  conf_mat[0][0]++;
                  else if(res==0 && ts_res.equals("1"))
                  conf_mat[0][1]++;
                  else if (res==1 && ts_res.equals("0"))
                  conf_mat[1][0]++;
                  else if (res==1 && ts_res.equals("1"))
                  conf_mat[1][1]++;
                  //incrementing values of the confusion matrix
```

```
     System.out.println("\t"+ts_res+"\t\t"+res);//printing results

                count++;
                ts_res=(br3.readLine());
           }

           System.out.println("True Negetives: "+conf_mat[0][0]);
           System.out.println("True Positives: "+conf_mat[1][1]);
           System.out.println("False Negetives: "+conf_mat[0][1]);
           System.out.println("False Positives: "+conf_mat[1][0]);
           // printing value of confusion matrix

           acc=(double)(conf_mat[0][0]+conf_mat[1][1])/
((double)count);//calculating accuracy
           System.out.println("Accuracy: "+acc*100+"%");

           if(count==(conf_mat[0][0]+conf_mat[0][1]+conf_mat[1]
[0]+conf_mat[1][1]))
           System.out.println("All files Read
Correctly");//Verifying that all files have been read


     }
}
```