Faculty of Engineering

Credit Hours Engineering Programs

Mechatronics Engineering and Automation

Academic Year 2019/2020 – Spring 2019

CSE 489

Machine Vision

Project No. (3)

Visual Recognition

Date due 16-Apr-19

Submitted by:

---- -

---- -

---- -

---- -

Contents

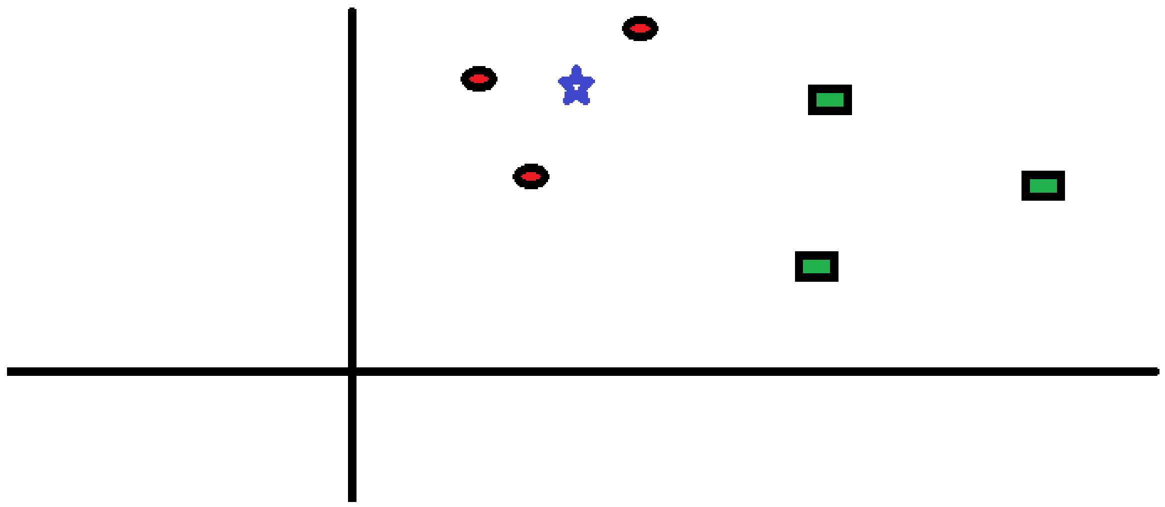
1 PROBLEM DEFINITON AND IMPORTANCE

The aim of this project is to build a machine learning classifier of different visual objects.
1. First select at least 10 visual objects and construct a data set for each object which at least has 100 images.
2. Divide the data sets to be 70% for training and 30% for testing.
3. Implement the NN, KNN, and MCSVM classifiers.
4. For each image, you consider the histogram of oriented gradients (HOG) feature.
5. Test your implemented classifiers and record the accuracy for each object. Comment on the results.
6. Repeat the above test but using any open source classification algorithm available on the authentic/trusted websites.

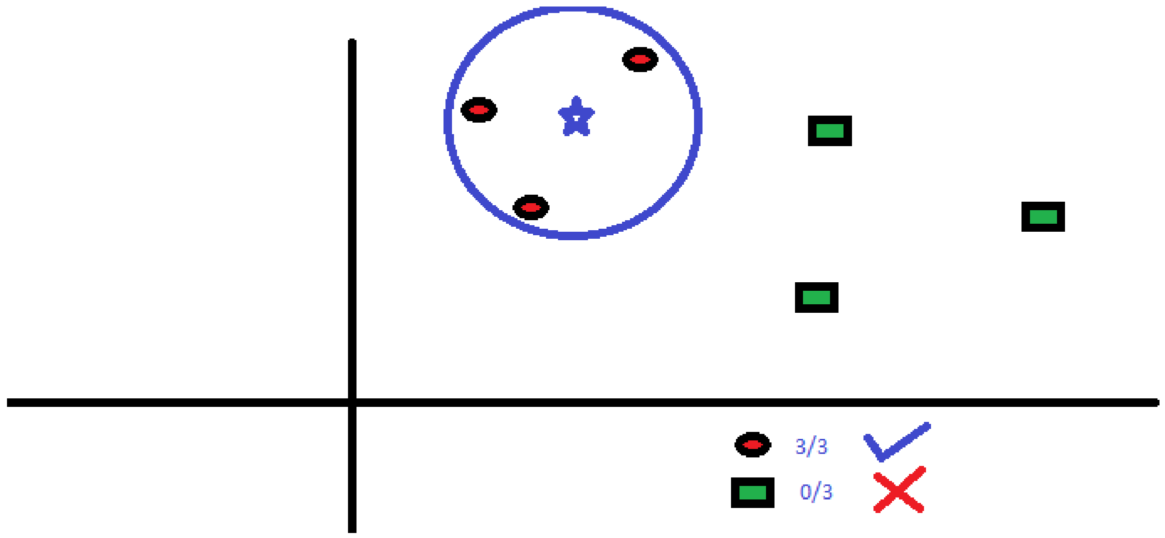
2 METHODS AND ALGORITHM

2.1 HOW DOES THE KNN ALGORITHM WORK?

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS) :



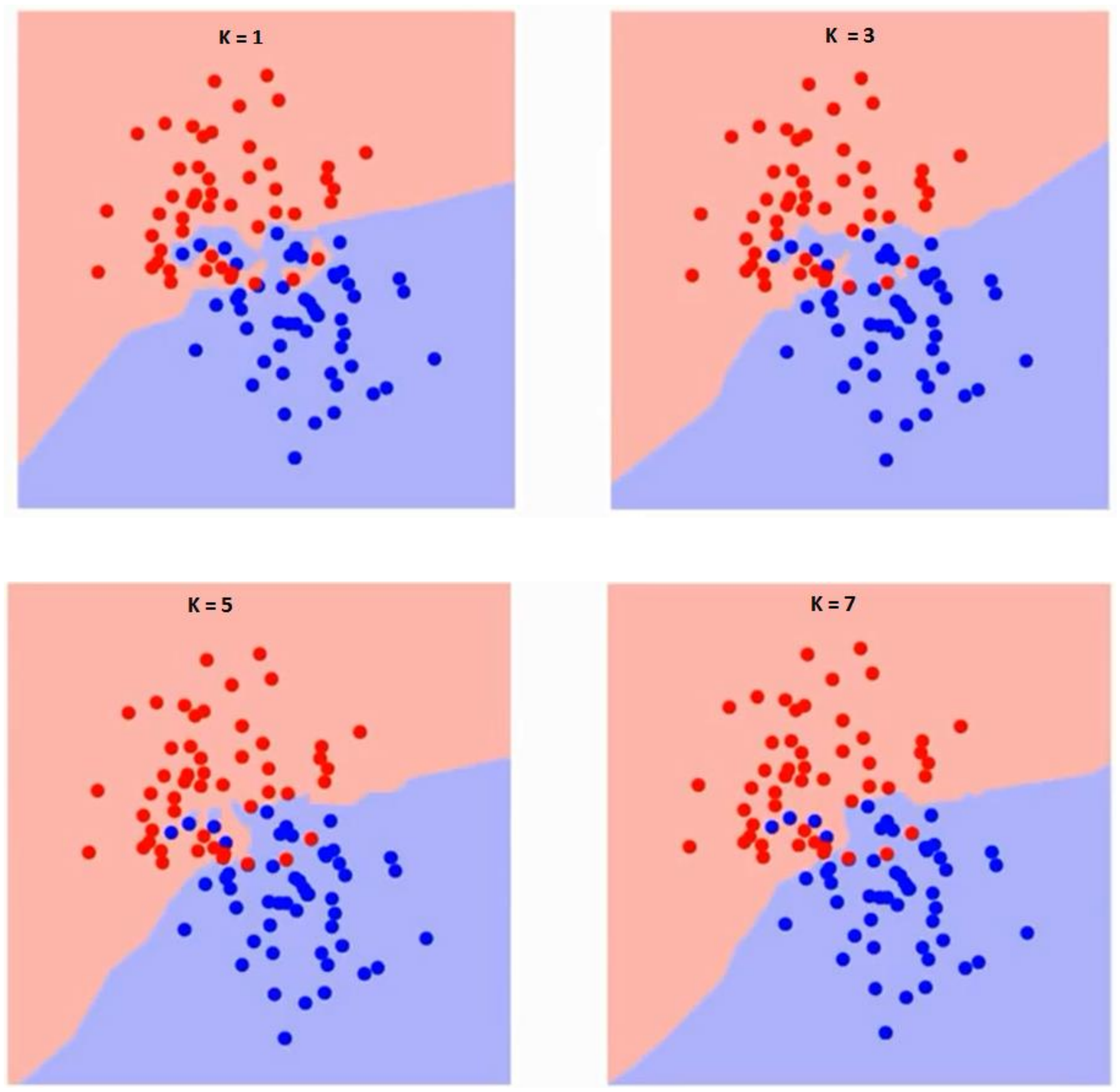
You intend to find out the class of the blue star (BS) . BS can either be RC or GS and nothing else. The "K" is KNN algorithm is the nearest neighbors we wish to take vote from. Let's say K = 3. Hence, we will now make a circle with BS as center just as big as to enclose only three datapoints on the plane. Refer to following diagram for more details:



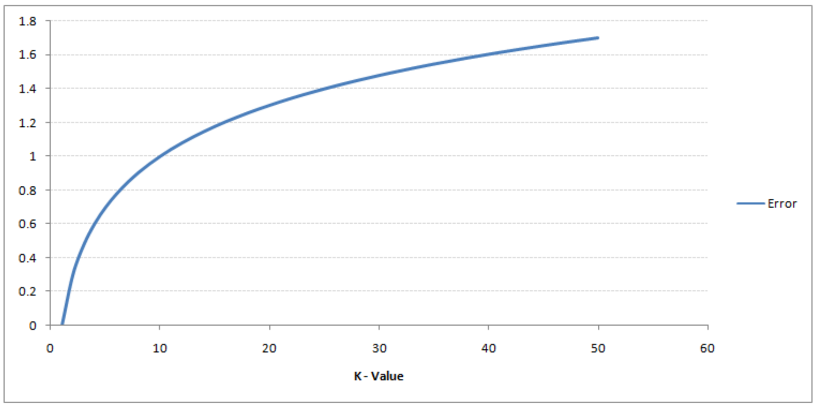
The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm. Next we will understand what are the factors to be considered to conclude the best K.

2.2 HOW DO WE CHOOSE THE FACTOR K?

First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. The same way, let’s try to see the effect of value “K” on the class boundaries. Following are the different boundaries separating the two classes with different values of K.



If you watch carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access on different K-value. Following is the curve for the training error rate with varying value of K:



```
knn(testingData,trainingData,trainingLabels,k)
 distance=[];
      % for j=1:length(testingData)
                for i=1: size(trainingData,1)
               distance = [distance; sqrt(sum(testingDatatrainingData(i,:)).^2)];
```

This Function get_KNN returns initially the labels that will be compared in order to see the Training classifier

2.3 HOG FEATURES EXTRACTION PROCESS:

- IMAGE IS DIVIDED IN CELLS OF SIZE N × N PIXELS. THE ORIENTATION OF ALL PIXELS IS COMPUTED AND ACCUMULATED IN AN M-BINS HISTOGRAM OF ORIENTATIONS. FINALLY, ALL CELL HISTOGRAMS ARE CONCATENATED IN ORDER TO CONSTRUCT THE FINAL FEATURES VECTOR. THE EXAMPLE REPORTS A CELL SIZE OF 4 PIXELS AND 8 ORIENTATION BINS FOR THE CELL HISTOGRAMS.
- In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions

[fvector, visualization]= extractHOGFeatures(img)

2.4 MULTICLASS CLASSIFICATION:

A classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multi-class classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time.

3 EXPERIMENTAL RESULTS AND DISCUSSION

A sample from the classes we use and the Hog feature visualization

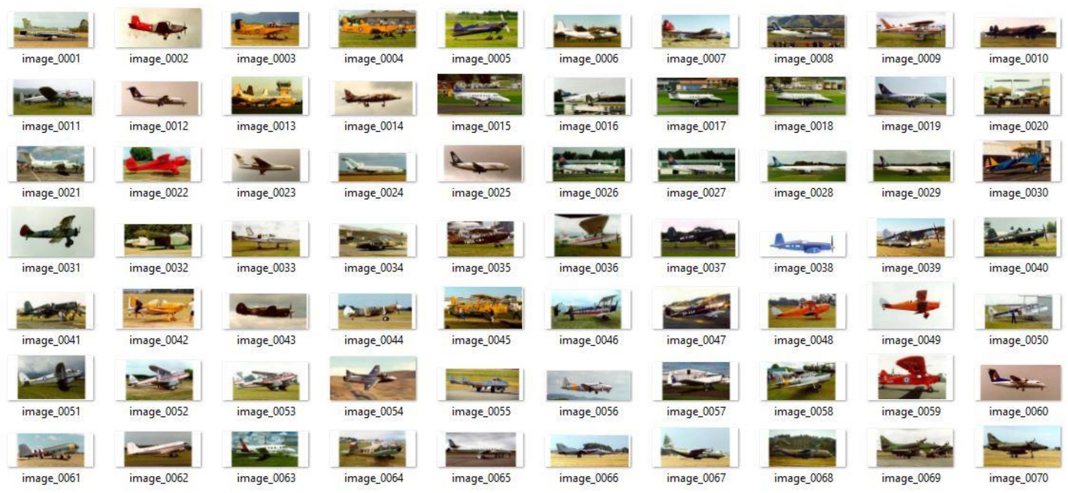


Figure 1 Class1 Airplanes

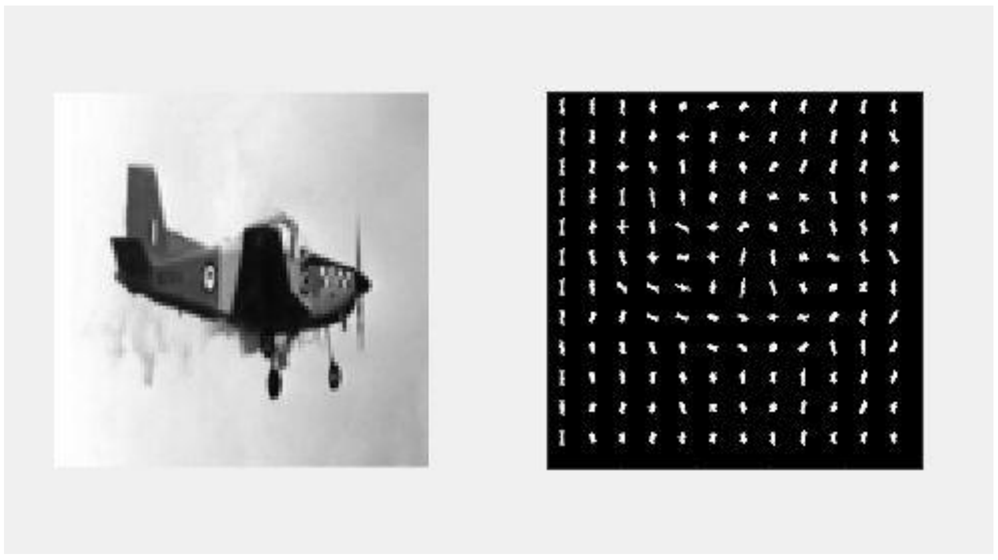


Figure 2 Hog Visualization

3.1 MCSVM RESULTS

Class Name	Accuracy
'airplanes';	86.667
bonsai	83.333
car_side	93.333
chandelier	90
Faces	93.333
hawksbill	83.333
ketch	76.667
Leopards	83.333
Motorbikes	53.333
watch	83.333

3.2 K-NEIGHREST NEIGHBOUR RESULTS

Class Name	Accuracy
'airplanes';	30
bonsai	16.667

car_side	20
chandelier	16.667
Faces	13.333
hawksbill	20
ketch	10
Leopards	33.333
Motorbikes	26.667
watch	16.667

3.3 NEARST NEIGHBOUR RESULTS

Class Name	Accuracy
'airplanes';	30
bonsai	16.667
car_side	20
chandelier	16.667
Faces	13.333

hawksbill	20
ketch	10
Leopards	33.333
Motorbikes	26.667
watch	16.667

3.4 DISCUSSION

The multiclass support vector machine algorithm showed a superior accuracy over the nearest neighbor algorithm. The MCSVM reach accuracy of 93.3 % while the Nearest neighbor maximum accuracy was only 33.333% !. the MCSVM was implemented using a ready matlab code. No wonder it achieves a high result than the implemented by ours.

4 APPENDIX

4.1 FOR EACH IMAGE, YOU CONSIDER THE HISTOGRAM OF ORIENTED GRADIENTS (HOG) FEATURE.

```
h=100;w=100;
NAirplanes=2;
s='datasets\SelectedDataSet\airplanes';
C1=[];
for i=1:NAirplanes
    [i]
    str=sprintf('%s/image_%04d.jpg',s,i);
    im=imread(str);
    im=rgb2gray(im);
    im=imresize(im,[h w]);
    [hog,visualization] = extractHOGFeatures(im);
    subplot(1,2,1);imshow(im);
    subplot(1,2,2);plot(visualization);
    pause(0.1)
    C1(i,:)=hog;
end
%%%%%%%%%%%%%%%%%%%%%
save('Airplanes.mat','C1')
%%%%%%%%%%%%%%%%%%%%%
```

4.2 IMPORTIMAGES

```
function [cTrainD, cTrainL, cTestD, cTestL] =
importImages(Dir,cName)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
cTrainD = [];
cTrainL = {};
cTestD = [];
cTestL = {};
for img_idx = 1:100
    img = imread(sprintf('%s/image_%04d.jpg' , Dir, img_idx));
    %      imshow(img);
    %      pause(0.5
    img= imresize(img,[128 192]);

    fVector = extractHOGFeatures(img);% x1 x2 point in 2-d space x1
x2 x3  point in 3-d space      x-n point in nd space

    if img_idx < 71      %3shan ye5ally awel 70 img training we ba2y
el 30 testing
        cTrainD = [cTrainD;fVector];  %img = img(:);      %returns
column representation of the matrix/kant ma7tota makan fVector
        cTrainL = [cTrainL;cName];
    else
        cTestD = [cTestD;fVector];
        cTestL = [cTestL;cName];
    end
end
end
```

4.3 ACCURACY

```
function [ acc ] = Accuracy(MCVM_1,cName1 )
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
acc=0;
for i=1:30
    if strcmp(MCVM_1(i),cName1)
        acc=acc+1;
    end
end

acc = (acc/30)*100;

end
```

4.4 K-NEARST NEIGHBOUR

```
function [Label] =
K_NearestNeighbor(TestingD,TrainingD,TrainingL,k)
distance=[];
                for i=1: size(TrainingD,1)
                distance = [distance; sqrt(sum(TestingD-
TrainingD(i,:)).^2)];
                end
    min_dist= 99999999*ones(1,12420);
    Index=0;
        for i=0:k
            for l=1:length(distance)
                 if distance(l,:)< min_dist(1,:)
                     min_dist(1,:)=distance(l,:);
                     Index=l;
                 end
            end
            distance(Index,:)=999999999;
        end
         %the KNN Algorithm
  if k>1
            C= zeros(10,1);
            if Index<=70
                C(1)=C(1)+1;
            elseif Index>70 && Index<=140
                C(2)=C(2)+1;
            elseif Index>140 && Index<=210
```

```
                C(3)=C(3)+1;
            elseif Index>210 && Index<=280
                C(4)=C(4)+1;
            elseif Index>280 && Index<=350
                C(5)=C(5)+1;
            elseif Index>350 && Index<=420
                C(6)=C(6)+1;
            elseif Index>420 && Index<=490
                C(7)=C(7)+1;
            elseif Index>490 && Index<=560
                C(8)=C(8)+1;
            elseif Index>560 && Index<=630
                C(9)=C(9)+1;
            elseif Index>630 && Index<=700
                C(10)=C(10)+1;
            end
           [~,Ind]=max(C);
           Label=TrainingL(Ind*70);
  else
      Label= TrainingL(Index);
  end

end
```

4.5 MAIN

```
clear;close;clc;
Dir1= 'datasets\SelectedDataSet\airplanes';
Dir2= 'datasets\SelectedDataSet\bonsai';
Dir3= 'datasets\SelectedDataSet\car_side';
Dir4= 'datasets\SelectedDataSet\chandelier';
Dir5= 'datasets\SelectedDataSet\Faces';
Dir6= 'datasets\SelectedDataSet\hawksbill';
Dir7= 'datasets\SelectedDataSet\ketch';
Dir8= 'datasets\SelectedDataSet\Leopards';
Dir9= 'datasets\SelectedDataSet\Motorbikes';
Dir10= 'datasets\SelectedDataSet\watch';

cName1= 'airplanes';
cName2= 'bonsai';
cName3= 'car_side';
cName4= 'chandelier';
cName5= 'Faces';
cName6= 'hawksbill';
cName7= 'ketch';
cName8= 'Leopards';
cName9= 'Motorbikes';
cName10= 'watch';


[c1TrainD,c1TrainL, c1TestD, c1TestL]   =
importImages(Dir1,cName1);
```

```
[c2TrainD,c2TrainL, c2TestD, c2TestL]   =
importImages(Dir2,cName2);
[c3TrainD,c3TrainL, c3TestD, c3TestL]   =
importImages(Dir3,cName3);
[c4TrainD,c4TrainL, c4TestD, c4TestL]   =
importImages(Dir4,cName4);
[c5TrainD,c5TrainL, c5TestD, c5TestL]   =
importImages(Dir5,cName5);
[c6TrainD,c6TrainL, c6TestD, c6TestL]   =
importImages(Dir6,cName6);
[c7TrainD,c7TrainL, c7TestD, c7TestL]   =
importImages(Dir7,cName7);
[c8TrainD,c8TrainL, c8TestD, c8TestL]   =
importImages(Dir8,cName8);
[c9TrainD,c9TrainL, c9TestD, c9TestL]   =
importImages(Dir9,cName9);
[c10TrainD,c10TrainL,c10TestD,c10TestL] =
importImages(Dir10,cName10);

MDL_MCSVM=
fitcecoc([c1TrainD;c2TrainD;c3TrainD;c4TrainD;c5TrainD;c6TrainD;c7T
rainD;c8TrainD;c9TrainD;c10TrainD],[c1TrainL;c2TrainL;c3TrainL;c4Tr
ainL;c5TrainL;c6TrainL;c7TrainL;c8TrainL;c9TrainL;c10TrainL]);



 %Accuracy
MCVM_1=predict(MDL_MCSVM,c1TestD);
Accuracy_MCVM_1=Accuracy(MCVM_1,cName1);

MCVM_2=predict(MDL_MCSVM,c2TestD);
Accuracy_MCVM_2=Accuracy(MCVM_2,cName2);

MCVM_3=predict(MDL_MCSVM,c3TestD);
Accuracy_MCVM_3=Accuracy(MCVM_3,cName3);

MCVM_4=predict(MDL_MCSVM,c4TestD);
Accuracy_MCVM_4=Accuracy(MCVM_4,cName4);

MCVM_5=predict(MDL_MCSVM,c5TestD);
Accuracy_MCVM_5=Accuracy(MCVM_5,cName5);

MCVM_6=predict(MDL_MCSVM,c6TestD);
Accuracy_MCVM_6=Accuracy(MCVM_6,cName6);

MCVM_7=predict(MDL_MCSVM,c7TestD);
Accuracy_MCVM_7=Accuracy(MCVM_7,cName7);

MCVM_8=predict(MDL_MCSVM,c8TestD);
Accuracy_MCVM_8=Accuracy(MCVM_8,cName8);

MCVM_9=predict(MDL_MCSVM,c9TestD);
Accuracy_MCVM_9=Accuracy(MCVM_9,cName9);

MCVM_10=predict(MDL_MCSVM,c10TestD);
Accuracy_MCVM_10=Accuracy(MCVM_10,cName10);
for i=1:size(c1TestL,1)
    %k=3
```

```
KNN_1(i,1)=K_NearestNeighbor([c1TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_2(i,1)=K_NearestNeighbor([c2TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_3(i,1)=K_NearestNeighbor([c3TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_4(i,1)=K_NearestNeighbor([c4TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_5(i,1)=K_NearestNeighbor([c5TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_6(i,1)=K_NearestNeighbor([c6TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_7(i,1)=K_NearestNeighbor([c7TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_8(i,1)=K_NearestNeighbor([c8TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_9(i,1)=K_NearestNeighbor([c9TestD(i,:)],[c1TrainD;c2TrainD;c3Tr
ainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Train
D],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;
c8TrainL;c9TrainL;c10TrainL],3);
KNN_10(i,1)=K_NearestNeighbor([c10TestD(i,:)],[c1TrainD;c2TrainD;c3
TrainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Tra
inD],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7Train
L;c8TrainL;c9TrainL;c10TrainL],3);
%K=1
NN_1(i,1)=K_NearestNeighbor([c1TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_2(i,1)=K_NearestNeighbor([c2TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_3(i,1)=K_NearestNeighbor([c3TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_4(i,1)=K_NearestNeighbor([c4TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_5(i,1)=K_NearestNeighbor([c5TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
```

```
NN_6(i,1)=K_NearestNeighbor([c6TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_7(i,1)=K_NearestNeighbor([c7TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_8(i,1)=K_NearestNeighbor([c8TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_9(i,1)=K_NearestNeighbor([c9TestD(i,:)],[c1TrainD;c2TrainD;c3Tra
inD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10TrainD
],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL;c
8TrainL;c9TrainL;c10TrainL],1);
NN_10(i,1)=K_NearestNeighbor([c10TestD(i,:)],[c1TrainD;c2TrainD;c3T
rainD;c4TrainD;c5TrainD;c6TrainD;c7TrainD;c8TrainD;c9TrainD;c10Trai
nD],[c1TrainL;c2TrainL;c3TrainL;c4TrainL;c5TrainL;c6TrainL;c7TrainL
;c8TrainL;c9TrainL;c10TrainL],1);

end
%
ACC_NN_1=Accuracy(NN_1,cName1);
ACC_NN_2=Accuracy(NN_2,cName2);
ACC_NN_3=Accuracy(NN_3,cName3);
ACC_NN_4=Accuracy(NN_4,cName4);
ACC_NN_5=Accuracy(NN_5,cName5);
ACC_NN_6=Accuracy(NN_6,cName6);
ACC_NN_7=Accuracy(NN_7,cName7);
ACC_NN_8=Accuracy(NN_8,cName8);
ACC_NN_9=Accuracy(NN_9,cName9);
ACC_NN_10=Accuracy(NN_10,cName10);
ACC_KNN_1=Accuracy(KNN_1,cName1);
ACC_KNN_2=Accuracy(KNN_2,cName2);
ACC_KNN_3=Accuracy(KNN_3,cName3);
ACC_KNN_4=Accuracy(KNN_4,cName4);
ACC_KNN_5=Accuracy(KNN_5,cName5);
ACC_KNN_6=Accuracy(KNN_6,cName6);
ACC_KNN_7=Accuracy(KNN_7,cName7);
ACC_KNN_8=Accuracy(KNN_8,cName8);
ACC_KNN_9=Accuracy(KNN_9,cName9);
ACC_KNN_10=Accuracy(KNN_10,cName10);
```