EE 559 Project

Mathematical Pattern Recognition

Yu-Fu Chen

yufuchen@usc.edu

5-2-2017

Abstract

This project is to use “German credit dataset” to develop classifiers can predict whether or not the applicant is a good credit risk. The approach to the best performance in this project is from various aspects of pattern recognition, including: conditioning the dataset, decreasing the feature space dimensionality, training different classifier and improving their performance. Finally, the best classifier with best result is selected, which is **QDA (Quadratic Bayes)** with **0.635** F1 score.

Methodology

In order to design a better pattern recognition system for the given “German credit dataset”, The following approach is being applied:

- Investigating the given dataset for the dataset_1: To examine the data to explore is there any non-relevant feature to impair our system design, the data missing, and the distribution of different class samples.
- Preprocessing: Transform the given dataset_1 for the better use, including removing uninformative features, recasting non-numerical features to the form of reasonable numerical presentation/one-hot encoding, splitting different class samples for acquiring training/test dataset. Also data normalization is applied in order to construct more reliable classifier.
- Feature selection: Some techniques are being applied to given dataset_1 to select which features from original feature set are more influential to our pattern system design, in order to reduce the load of computation and acquire a better performance, and help us to understand the causal relationship between features and classes.
- Classification: Using different types of classifier to be trained, test the performance by different selected features, including using one-pass technique or cross-validation to test system performance, searching best parameter through cross-validation, for given dataset_1.

Preprocessing

It is noticeable the first feature in the given dataset_1 is just a index for identifying different applicant, which is non-relevant(trivial) to our pattern system design. For this reason, the first feature is discarded. Second, there are some categorical features which have natural order, as shown below:

categorical features with natural order		
Feature	Original data	Data after recasting
Saving Account/ Checking Account	rich	5
	quite rich	4
	moderate	3
	little	2
	NA	1

At the beginning, the mean of CheckingAccount/SavingAccount js applied to present "NA" However, after the test of QDA classifier, which has the best performance in this project, it is noticeable using "1" to present "NA" has better performance.

However, there are some features do not have natural order in the dataset_1. The technique of "One-Hot Encoding" was applied to these features, as we see below:

categorical features without natural order (One-Hot Encoding is applied)		
Feature	Original data	Data after one-hot encoding
Sex	male	[0 1]
	female	[1 0]

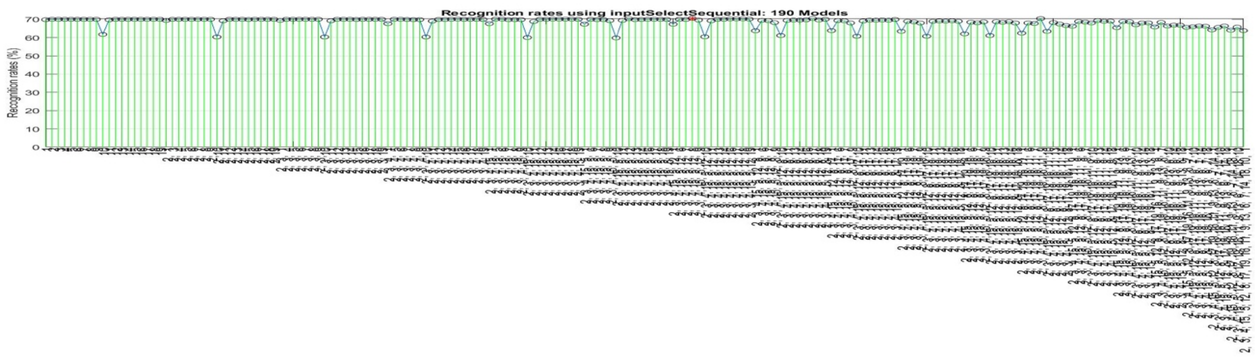
Housing	own	[1 0 0]
	rent	[0 1 0]
	free	[0 0 1]
Purpose	business	[1 0 0 0 0 0 0 0]
	education	[0 1 0 0 0 0 0 0]
	car	[0 0 1 0 0 0 0 0]
	vacation/others	[0 0 0 1 0 0 0 0]
	furniture/equipment	[0 0 0 0 1 0 0 0]
	domestic appliances	[0 0 0 0 0 1 0 0]
	radio/TV	[0 0 0 0 0 0 1 0]
	repairs	[0 0 0 0 0 0 0 1]

Feature Selection: <ee559_pro_fs.m>

For reducing the computation and acquire a pattern system with higher performance, a feature selection technique is applied, which is "Sequential Forward Selection using Heuristic method."

- K-NN Classifier is used with combination of leave-one-out test for performance estimation.
- First, normalize the dataset and select one feature with highest classification rate.
- Then, select other feature from unselected feature set combining with selected features which gives the highest classification rate. Repeat this process.
- Get the reduced feature space which gives the highest classification rate.
- The selected features after "SFS" is : 2, 3, 4, 5, 8, 12, 15, with classification rate of 70.5%.
- A effective system can be achieved by reducing 63.2% amount of total features.
- Function "inputSelectSequential" in "MLT" toolbox is applied.
  (Available on: https://mirlab.org/jang/matlab/toolbox/machineLearning/)

▪ The main purpose for feature selection here is for reducing computation load instead of acquiring a better performance because the amount of samples is greater than rule of thumb: 3 to 10 times degree of freedom (90). The situation of overfitting is with less probability to happen. Although one-hot encoding is applied to expand the original feature space, it is predictable the performance will probably be impaired after applying selected features to some classifiers compared with using all features.



Observation:

As we can see from above figure, it is noticeable that the classification rate drops whenever feature 10 (credit amount) is selected. There are two possible reasons: First, as we can see in the original dataset_1, the range of this feature is from 250 to 18424. The feature with wide-range values compared to other feature values will dominate Euclidean distance, more or less affect the system performance. However, before the "SFS" applied here, data normalization has been applied. In other words, the drop of classification rate with feature 10 (credit amount) selected is not because of the wide-range values. It is the natural result by choosing this feature.

Classification

(1) K-NN classifier: <ee559_project_fea_last_KNN.m>

- All features in dataset_1 is applied to train the classifier and test the performance, except first feature, which is index number identifying applicants.
- Train the classifier **with/without data normalization** of zero-mean, variance of one.
- Randomly set aside 20% amount of data to serve as test dataset.
- Using 20-folds cross validation in training dataset to **find best parameter k**.

- Using test data set to estimate the system performance (One-Pass Estimation).
- Using **10-folds cross validation** (subject to **constrain** that the percentage of samples in each class is unchanged in each subset: 70% from class 1; 30% from class 2) ,**20 times** with best parameter k we found above, to estimate system performance in order to get a more robust estimation of performance.
- Calculating the F1 score.
- Build-in Matlab Function "fitcknn()" is applied.
- Result:

K-Nearest Neighbor Classifier: **Normalized** Dataset_1 (Best Parameter k=19)		
Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 score
0.750 (fluctuating)	0.702 (more fixed)	0.524

K-Nearest Neighbor Classifier: **Non-Normalized** Dataset_1 (Best Parameter k=20)		
Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 score
0.655 (fluctuating)	0.695 (more fixed)	0.500

Observation:

- **The performance of the classifier improves if normalized dataset_1 is being used**.
- The result of One-Pass Estimation is fluctuating because the sample chosen to be training/testing dataset might be biased.
- As mentioned in feature selection above, The feature with wide-range values compared to other feature values will dominate Euclidean distance, more or less affect the system performance.

- The F1 score is relatively low compared to other method. The reason could be the unbiased dataset_1 in which a dominant class gives the classifier misleading probability estimation for establishing the model.
- The performance is influenced dramatically by different chosen parameter k.

(2) MSE / Perceptron with one vs one/ one vs all method <ee559pro_mse_per_linear.m>

- All features in dataset_1 is applied to train the classifier and test the performance, except first feature, which is index number identifying applicants.
- Train the classifier **with data normalization** of zero-mean, variance of one.
- Randomly set aside 20% amount of data to serve as test dataset.
- Using test data set to estimate the system performance (One-Pass Estimation).
- Using **10-folds cross validation** (subject to **constrain** that the percentage of samples in each class is unchanged in each subset: 70% from class 1; 30% from class 2) ,**20 times**, in order to acquire a more robust estimation for performance.
- Calculating the F1 score.
- Function “multiclass()” in “Classification toolbox” by David G.Stork and Elad Yom-Tov is applied.

Normalized Dataset_1 (one vs. rest)			
	Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 Score
MSE	0.725(fluctuating)	0.706	0.522
Perceptron	0.665(fluctuating)	0.686	0.601

Normalized Dataset_1 (one vs. one)			
	Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 Score
MSE	0.705(fluctuating)	0.691	0.594
Perceptron	0.665(fluctuating)	0.683	0.593

Observation:

- MSE has better performance in any scenarios, implying this dataset probably is not totally linear separable.

(3) Naïve Bayes Classifier / QDA <ee559pro_naïve_bayes.m>

- All features/**Reduced feature space calculated by "SFS"** in dataset_1 is applied to train the classifier and test the performance, except first feature, which is index number identifying applicants.
- Train the classifier **with/without data normalization** of zero-mean, variance of one.
- Randomly set aside 20% amount of data to serve as test dataset.
- Using test data set to estimate the system performance (One-Pass Estimation).
- Using **10-folds cross validation** (subject to **constrain** that the percentage of samples in each class is unchanged in each subset: 70% from class 1; 30% from class 2) ,**20 times**, in order to acquire a more robust estimation for performance.
- Calculating the F1 score.
- Matlab Built-in function "fitcib()" and "fitcdiscr()" is applied.
- Result:

Naïve Bayes Classifier: Normalized Dataset_1 with all features		
Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 score
0.650 (fluctuating)	0.672	0.615

Naïve Bayes Classifier: Non-Normalized Dataset_1 with all features		
Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 score
0.615 (fluctuating)	0.673	0.615

Observation:

- No matter normalized data is applied or not, the performance of the system is identical, because the decision rule is based on prior probability and class-conditional probability.

- Naïve Bayes classifier assumes all features are independent, which is hard to achieve in the real life dataset.

So here, **QDA (Quadratic Bayes)** is investigated:

- When we check the covariance matrix of each class, it is noticeable they are different. This is the reason Quadratic Bayes is applied instead of Linear Bayes.
- When using QDA, a inverting covariance matrix is required.
- Singular covariance matrices for 2 classes are acquired from dataset_1. The singularity issue need to be fixed.
- There are some techniques can solve the issue, such as using PCA to reduce dimensionality first, or regularize the covariance matrices.
- For here, “pseudoQuadratic” is applied to deal with this issue.
- Result:

QDA: Non-Normalized Dataset_1 with all features		
Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 score
0.690 (fluctuating)	0.695	0.635

QDA: Non-Normalized Dataset_1 with **reduced feature space**		
Classification Accuracy (One-Pass Estimation)	Classification Accuracy (Cross-Validation with 10 folds, with constraints, 20 times)	F1 score
0.670	0.693	0.498

Observation:

- The reduced feature space saves the computation load. However, it sacrifice the performance of the system. The reason here is, we are not using exhaustive method for “SFS”, so the reduced features set is not the most ideal reduced feature set, some lost of performance is predictable.

Naïve Bayes / QDA Comparison:

- QDA performs better than Naïve Bayes Classifier.
- Using multivariate Gaussian has better performance, the reason might be this probabilistic

model is more suitable to the nature of this given dataset_1.

- The reason Naïve Bayes performs worse than QDA is because the assumption (all features are independent) Naïve Bayes make is too strong for real-data world.

Conclusion

The methods to improve the system performance is believable. For this project, I believe the way to replace “NA” in feature “Saving/CheckingAccount” would have some influence on the performance. Although feature selection is performed by Sequential Forward Selection, from the perspective of rule of thumb for dimensionality reduction, I believe there is no need to train systems with reduced amount of features because the situation of overfitting is less possible, and less information classifier can learn from. As shown in the last trial, reduced feature space is applied on QDA classifier and a worse outcome appears.

The best classifier in this project is Quadratic Bayes Classifier with F1 score 0.635. This dataset has unbiased samples with different labels (70% from class 1 and 30% from class 2), which has more potential negative effect on statistical classifier.

References

[1] “Mathematics & Statistics Lecture Notes” by Dr. Guangliang Chen, San José State University

Retrieved from: http://www.math.sjsu.edu/~gchen/Math285S16/lec1knn.pdf

[2] “Data Clustering and Pattern Recognition” by Roger Jang

Retrieved from: https://mirlab.org/jang/books/dcpr/

[3] “Q&A from stackoverflow.com”

Retrieved from:

http://stackoverflow.con/questions/22915003/is-there-ant-function-to-calculate_precision-and-recall-using-matlab

Code

```
%% change categoriclas features into corresponding numerical value AND make Dataset
into matrix

% '[0 1]' presenting 'male'; '[1 0]'presenting 'female'
dm = dataset('File','Proj_dataset_1.csv','Delimiter',',');
for i=1:length(dm)
    if length(dm{i,3})==4
        dm{i,3}= [0 1];
    elseif length(dm{i,3})==6
        dm{i,3}=[1 0];
    end
end

% '[0 0 1]' represting 'free'; '[0 1 0]' presenting 'rent'; '[1 0 0]' presenting
'own'
for i=1:length(dm)
    if length(dm{i,5})==3
        dm{i,5}=[1 0 0];
    elseif length(dm{i,5})==4
        if  dm{i,5}=='rent'
            dm{i,5}=[0 1 0];
        else
            dm{i,5}=[0 0 1];
        end
    end
end

% let feature 'saving accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; '1' presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,6})==1
        dm{i,6}=5;
    elseif strcmp('quite rich',dm{i,6})==1
        dm{i,6}=4;
    elseif strcmp('moderate',dm{i,6})==1
        dm{i,6}=3;
    elseif strcmp('little',dm{i,6})==1
        dm{i,6}=2;
    elseif strcmp('NA',dm{i,6})==1
        dm{i,6}=1;
    else
        dm{i,6}=0;
    end
end
% let feature 'checking accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; '1' presenting 'NA'
for i=1:length(dm)
```

```
    if strcmp('rich',dm{i,7})==1
        dm{i,7}=5;
    elseif strcmp('quite rich',dm{i,7})==1
        dm{i,7}=4;
    elseif strcmp('moderate',dm{i,7})==1
        dm{i,7}=3;
    elseif strcmp('little',dm{i,7})==1
        dm{i,7}=2;
    elseif strcmp('NA',dm{i,7})==1
        dm{i,7}=1;
    else
        dm{i,7}=0;
    end
end
% let feature 'purpose' be value from 8 to 1
% '[1 0 0 0 0 0 0 0]' presenting 'business'; '[0 1 0 0 0 0 0 0]' presenting
'education'
% '[0 0 1 0 0 0 0 0]' presenting 'car'; '[0 0 0 1 0 0 0 0]' presenting
'vacation/others'
% '[0 0 0 0 1 0 0 0]' presenting 'furniture/equipment'; '[0 0 0 0 0 1 0 0]'
presenting 'radio/TV'
% '[0 0 0 0 0 0 1 0]' presenting 'repairs'; '[0 0 0 0 0 0 0 1]' presenting 'domestic
appliances'
for i=1:length(dm)
    if strcmp('business',dm{i,10})==1
        dm{i,10}=[1 0 0 0 0 0 0 0];
    elseif strcmp('education',dm{i,10})==1
        dm{i,10}=[0 1 0 0 0 0 0 0];
    elseif strcmp('car',dm{i,10})==1
        dm{i,10}=[0 0 1 0 0 0 0 0];
    elseif strcmp('vacation/others',dm{i,10})==1
        dm{i,10}=[0 0 0 1 0 0 0 0];
    elseif strcmp('furniture/equipment',dm{i,10})==1
        dm{i,10}=[0 0 0 0 1 0 0 0];
    elseif strcmp('domestic appliances',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 1 0 0];
    elseif strcmp('radio/TV',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 1 0];
    elseif strcmp('repairs',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 0 1];
    end
end
% Make Dataset into Matrix Form
dm= dataset2cell(dm);
dm(1,:) = [];
dm=cell2mat(dm);
%delete the first feature, it is just a index for appliances.
dm(:,1)=[];
dm_wt_label= dm;
```

```
dm_wt_label(:,20)=[];

%%normalized Data
feat_mean=mean(dm);
std_vec_dm=std(dm);
dm_normalized=zeros(1000,20);
for j=1:size(dm,2)
    dm_normalized(:,j)= (dm(:,j)-feat_mean(j))./std_vec_dm(j);%make data zero mean
and std of 1
end
clear j feat_mean std_vec_dm
dm_normalized(:,20)=[];
dm_wt_label=dm_normalized;


DS1.dataName= 'project';
for i=1:19
DS1.inputName{1,i}=i;
end
for i=1:2
DS1.outputName{1,2}=i;
end
DS1.input= dm_wt_label';
DS1.output= dm(:,20)';

inputNum=size(DS1.input, 1);
DS1.inputName=cellstr(int2str((1:inputNum)'));
inputSelectSequential(DS1);%%adjust the code for my use
%% FROM Reference [3]
```

```
%% Final_Project_for_EE559 File 3
%% KNN
%% change categoriclas features into corresponding numerical value AND make Dataset
into matrix

% let feature 'sex' be value of '1' and '2'
% '1' presenting 'male'; '2'presenting 'female'
dm = dataset('File','Proj_dataset_1.csv','Delimiter',',');
for i=1:length(dm)
    if length(dm{i,3})==4
        dm{i,3}=1;
    elseif length(dm{i,3})==6
        dm{i,3}=2;
    else
        dm{i,3}=0
    end
end

% let feature 'housing' be value of '1' , '2', and'3'
% '1' represting 'free'; '2' presenting 'rent'; '3' presenting 'own'
for i=1:length(dm)
    if length(dm{i,5})==3
        dm{i,5}=3;
    elseif length(dm{i,5})==4
        if  dm{i,5}=='rent'
            dm{i,5}=2;
        else
            dm{i,5}=1;
        end
    end
end

% let feature 'saving accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; '1' presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,6})==1
        dm{i,6}=5;
    elseif strcmp('quite rich',dm{i,6})==1
        dm{i,6}=4;
    elseif strcmp('moderate',dm{i,6})==1
        dm{i,6}=3;
    elseif strcmp('little',dm{i,6})==1
        dm{i,6}=2;
    elseif strcmp('NA',dm{i,6})==1
        dm{i,6}=1;
    else
        dm{i,6}=0;
    end
```

```
end
% let feature 'checking accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; '1' presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,7})==1
        dm{i,7}=5;
    elseif strcmp('quite rich',dm{i,7})==1
        dm{i,7}=4;
    elseif strcmp('moderate',dm{i,7})==1
        dm{i,7}=3;
    elseif strcmp('little',dm{i,7})==1
        dm{i,7}=2;
    elseif strcmp('NA',dm{i,7})==1
        dm{i,7}=1;
    else
        dm{i,7}=0;
    end
end
% let feature 'purpose' be value from 8 to 1
% '8' presenting 'business'; '7' presenting 'education'
% '6' presenting 'car'; '5' presenting 'vacation/others'
% '4' presenting 'furniture/equipment'; '3' presenting 'radio/TV'
% '2' presenting 'repairs'; '1' presenting 'domestic appliances'
for i=1:length(dm)
    if strcmp('business',dm{i,10})==1
        dm{i,10}=8;
    elseif strcmp('education',dm{i,10})==1
        dm{i,10}=7;
    elseif strcmp('car',dm{i,10})==1
        dm{i,10}=6;
    elseif strcmp('vacation/others',dm{i,10})==1
        dm{i,10}=5;
    elseif strcmp('furniture/equipment',dm{i,10})==1
        dm{i,10}=4;
    elseif strcmp('domestic appliances',dm{i,10})==1
        dm{i,10}=3;
    elseif strcmp('radio/TV',dm{i,10})==1
        dm{i,10}=2;
    elseif strcmp('repairs',dm{i,10})==1
        dm{i,10}=1;
    end
end
% Make Dataset into Matrix Form
dm= dataset2cell(dm);
dm(1,:) = [];
dm=cell2mat(dm);
%delete the first feature, it is just a index for appliances.
dm(:,1)=[];
```

```
dm_wt_label= dm;
dm_wt_label(:,10)=[];

%% Randomly Generate 200 samples as test dataset, 800 samples as traing dataset
r1=randperm(1000,200);
for i=1:200
per20_test(i,:)=dm(r1(i),:);
end
per20_test_without_label=per20_test;
per20_test_without_label(:,10)=[];
per80_train=dm;
per80_train(r1,:)=[];
per80_train_without_label=per80_train;
per80_train_without_label(:,10)=[];

%% Using Cross-Validation OF 20 folds in traing dataset to find Best Parameter K
error=1;
for i=1:20
model=fitcknn(per80_train_without_label, per80_train(:,10),'NumNeighbors',i);
c=crossval(model,'kfold',20); %% adjust from reference [1]
error_i = kfoldLoss(c);
if error_i<error
    error=error_i;
    k_best=i;
end
end
model=fitcknn(per80_train_without_label, per80_train(:,10),'NumNeighbors',k_best);
t = predict(model,per20_test_without_label);
fprintf('Best K is %d\n',k_best)
fprintf('Accuracy for KNN using Best K found by cross-validation with 20 folds: %5.3
f\n',sum(per20_test(:,10)==t)/200);
clear k_loss k_loss_i r1 per20_test per20_test_without_label
per80_train_without_label per80_train kloss_i kloss dm_wt_label






%% Cross-Validation of 10-folds of Performace Estimation for KNN using Best K found
by cross-validation with 20 folds with normalized data
feat_mean=mean(dm);
std_vec_dm=std(dm);
dm_normalized=zeros(1000,10);
for j=1:size(dm,2)
    dm_normalized(:,j)= (dm(:,j)-feat_mean(j))./std_vec_dm(j);%make data zero mean
and std of 1
end
clear j feat_mean std_vec_dm
```

```
dm_normalized(:,10)=[];

class1_dm=dm_normalized; %building class 1 dataset
k=0;
for i=1:1000
    if dm(i,10)==2
        k=k+1;
        l(k)=i;
    end
end
class1_dm(l,:)=[];
clear k l

class2_dm=dm_normalized; %building class 2 dataset
k=0;
for i=1:1000
    if dm(i,10)==1
        k=k+1;
        l(k)=i;
    end
end
class2_dm(l,:)=[];
clear k l


rv1_test= randperm(size(class1_dm,1),700);
rv2_test= randperm(size(class2_dm,1),300);
for i=1:70
    D1_c1(i,:)= class1_dm(rv1_test(i),:);
    label_D1_c1(i)= 1;
end
for i=1:30
    D1_c2(i,:)= class2_dm(rv2_test(i),:);
    label_D1_c2(i)= 2;
end
D1=[D1_c1;D1_c2];
label_D1=[label_D1_c1';label_D1_c2'];
D_set{1,1}=D1;
D_set{2,1}=label_D1;
for i=1:70
    D2_c1(i,:)= class1_dm(rv1_test(i+70),:);
    label_D2_c1(i)= 1;
end
for i=1:30
    D2_c2(i,:)= class2_dm(rv2_test(i+30),:);
    label_D2_c2(i)= 2;
end
D2=[D2_c1;D2_c2];
label_D2=[label_D2_c1';label_D2_c2'];
```

```
D_set{1,2}=D2;
D_set{2,2}=label_D2;
for i=1:70
    D3_c1(i,:)= class1_dm(rv1_test(i+140),:);
    label_D3_c1(i)= 1;
end
for i=1:30
    D3_c2(i,:)= class2_dm(rv2_test(i+60),:);
    label_D3_c2(i)= 2;
end
D3=[D3_c1;D3_c2];
label_D3=[label_D3_c1';label_D3_c2'];
D_set{1,3}=D3;
D_set{2,3}=label_D3;
for i=1:70
    D4_c1(i,:)= class1_dm(rv1_test(i+210),:);
    label_D4_c1(i)= 1;
end
for i=1:30
    D4_c2(i,:)= class2_dm(rv2_test(i+90),:);
    label_D4_c2(i)= 2;
end
D4=[D4_c1;D4_c2];
label_D4=[label_D4_c1';label_D4_c2'];
D_set{1,4}=D4;
D_set{2,4}=label_D4;

for i=1:70
    D5_c1(i,:)= class1_dm(rv1_test(i+280),:);
    label_D5_c1(i)= 1;
end
for i=1:30
    D5_c2(i,:)= class2_dm(rv2_test(i+120),:);
    label_D5_c2(i)= 2;
end
D5=[D5_c1;D5_c2];
label_D5=[label_D5_c1';label_D5_c2'];
D_set{1,5}=D5;
D_set{2,5}=label_D5;

for i=1:70
    D6_c1(i,:)= class1_dm(rv1_test(i+350),:);
    label_D6_c1(i)= 1;
end
for i=1:30
    D6_c2(i,:)= class2_dm(rv2_test(i+150),:);
    label_D6_c2(i)= 2;
end
D6=[D6_c1;D6_c2];
```

```
label_D6=[label_D6_c1';label_D6_c2'];
D_set{1,6}=D6;
D_set{2,6}=label_D6;

for i=1:70
    D7_c1(i,:)= class1_dm(rv1_test(i+420),:);
    label_D7_c1(i)= 1;
end
for i=1:30
    D7_c2(i,:)= class2_dm(rv2_test(i+180),:);
    label_D7_c2(i)= 2;
end
D7=[D7_c1;D7_c2];
label_D7=[label_D7_c1';label_D7_c2'];
D_set{1,7}=D7;
D_set{2,7}=label_D7;
for i=1:70
    D8_c1(i,:)= class1_dm(rv1_test(i+490),:);
    label_D8_c1(i)= 1;
end
for i=1:30
    D8_c2(i,:)= class2_dm(rv2_test(i+210),:);
    label_D8_c2(i)= 2;
end
D8=[D8_c1;D8_c2];
label_D8=[label_D8_c1';label_D8_c2'];
D_set{1,8}=D8;
D_set{2,8}=label_D8;
for i=1:70
    D9_c1(i,:)= class1_dm(rv1_test(i+560),:);
    label_D9_c1(i)= 1;
end
for i=1:30
    D9_c2(i,:)= class2_dm(rv2_test(i+240),:);
    label_D9_c2(i)= 2;
end
D9=[D9_c1;D9_c2];
label_D9=[label_D9_c1';label_D9_c2'];
D_set{1,9}=D9;
D_set{2,9}=label_D9;

for i=1:70
    D10_c1(i,:)= class1_dm(rv1_test(i+630),:);
    label_D10_c1(i)= 1;
end
for i=1:30
    D10_c2(i,:)= class2_dm(rv2_test(i+270),:);
    label_D10_c2(i)= 2;
end
```

```
D10=[D10_c1;D10_c2];
label_D10=[label_D10_c1';label_D10_c2'];
D_set{1,10}=D10;
D_set{2,10}=label_D10;
clear D1_c1 D1_c2 D2_c1 D2_c2 D3_c1 D3_c2 D4_c1 D4_c2 D5_c1 D5_c2 D6_c1 D6_c2 D7_c1
D7_c2 D8_c1 D8_c2 D9_c1 D9_c2 D10_c1 D10_c2
clear label_D1_c1  label_D1_c2 label_D2_c1  label_D2_c2 label_D3_c1  label_D3_c2
label_D4_c1  label_D4_c2 label_D5_c1  label_D5_c2 label_D6_c1  label_D6_c2
label_D7_c1  label_D7_c2 label_D8_c1  label_D8_c2 label_D9_c1  label_D9_c2
label_D10_c1  label_D10_c2



%   total_classfication_rate=total_classfication_rate+m;
% end
%crs_vrid_perfm=total_classfication_rate/10;

total_classfication_rate=0;
for i=1:10
model=fitcknn([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)
+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set
{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}],    [D_set{2,(mod
(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)
+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set
{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}],'NumNeighbors',k_best);
t = predict(model, D_set{1,(mod(i+9,10)+1)});
m=mean(D_set{2,(mod(i+9,10)+1)}==t);
total_classfication_rate=total_classfication_rate+m;
end
crs_vrid_perfm=total_classfication_rate/10;
fprintf('Cross-Validation of 10-folds of Performace Estimation for KNN using Best K
found by cross-validation with 20 folds: %5.3f\n',crs_vrid_perfm);
%clear k_loss k_loss_i r1 per20_test per20_test_without_label
per80_train_without_label per80_train kloss_i kloss dm_wt_label
```

```
%%Native Bayes Classifier


%% Randomly Generate 200 samples as test dataset, 800 samples as traing dataset
%% and also do cross-validation
% t_total1=0;
% t_total2=0;
% t_total3=0;
% for j=1:30
% r1=randperm(1000,200);
% for i=1:200
% per20_test(i,:)=dm(r1(i),:);
% end
% per20_test_without_label=per20_test;
% per20_test_without_label(:,20)=[];
% per80_train=dm;
% per80_train(r1,:)=[];
% per80_train_without_label=per80_train;
% per80_train_without_label(:,20)=[];
% %% Naive Bayers Classifier
% model=fitcnb(per80_train_without_label,per80_train(:,20),'Distribution','normal');
% t1 = predict(model,per20_test_without_label);
% t_total1= t_total1 +sum(per20_test(:,20)==t1)/200;
%
% %% Improving Naive Bayes
% model=fitcnb(per80_train_without_label,per80_train(:,20),'Distribution','kernal');
% t2 = predict(model,per20_test_without_label);
% t_total2= t_total2 +sum(per20_test(:,20)==t2)/200;

%
% end
% fprintf('Accuracy for Naive Bayers Classifier: %5.3f\n',t_total1/30);
% fprintf('Accuracy for Improving Naive Bayers Classifier: %5.3f\n',t_total2/30);

% %% F1 Score for Naive Bayers Classifier
% model=fitcnb(dm_wt_label,dm(:,20),'Distribution','normal');
% t_total = predict(model, dm_wt_label);
% CM=confusionmat(dm(:,20),t_total);
% precision=diag(CM)./sum(CM,2);
% recall=diag(CM)./sum(CM,1)';
% f1Score_eachclass=2*(precision.*recall)./(precision+recall); %F1 Score for each
class
% F1_Score=mean(f1Score_eachclass); % F1 score for all classes
% fprintf('F1 Score for Naive Bayers Classifier with Normalized Dataset is %5.3f\n',
F1_Score)
% %% F1 Score for Improvined Naive Bayers Classifier
% model=fitcnb(dm_wt_label,dm(:,20),'Distribution','kernal');
% t_total = predict(model, dm_wt_label);
% CM=confusionmat(dm(:,20),t_total);
```

```
% precision=diag(CM)./sum(CM,2);
% recall=diag(CM)./sum(CM,1)';
% f1Score_eachclass=2*(precision.*recall)./(precision+recall); %F1 Score for each
class
% F1_Score=mean(f1Score_eachclass); % F1 score for all classes



% fprintf('F1 Score for improved Naive Bayers Classifier with Normalized Dataset is %
5.3f\n',F1_Score)
% %% F1 Score for LDA Classifier
% model=fitcdiscr(dm_wt_label, dm(:,20),'DiscrimType','Linear');
% t_total = predict(model, dm_wt_label);
% CM=confusionmat(dm(:,20),t_total);
% precision=diag(CM)./sum(CM,2);
% recall=diag(CM)./sum(CM,1)';
% f1Score_eachclass=2*(precision.*recall)./(precision+recall); %F1 Score for each
class
% F1_Score=mean(f1Score_eachclass); % F1 score for all classes
% fprintf('F1 Score for LDA Classifier with Normalized Dataset is %5.3f\n\n',
F1_Score)
%
















%%
%% Final_Project_for_EE559 File 2
%% Naive Bayers Classifier
%% change categoriclas features into corresponding numerical value AND make Dataset
into matrix
%% one-hot encoding
clc;
```

```
clear all
% '[0 1]' presenting 'male'; '[1 0]'presenting 'female'
dm = dataset('File','Proj_dataset_1.csv','Delimiter',',');
for i=1:length(dm)
    if length(dm{i,3})==4
        dm{i,3}= [0 1];
    elseif length(dm{i,3})==6
        dm{i,3}=[1 0];
    end
end

% '[0 0 1]' represting 'free'; '[0 1 0]' presenting 'rent'; '[1 0 0]' presenting
'own'
for i=1:length(dm)
    if length(dm{i,5})==3
        dm{i,5}=[1 0 0];
    elseif length(dm{i,5})==4
        if  dm{i,5}=='rent'
            dm{i,5}=[0 1 0];
        else
            dm{i,5}=[0 0 1];
        end
    end
end


% let feature 'saving accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; "1" presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,6})==1
        dm{i,6}=5;
    elseif strcmp('quite rich',dm{i,6})==1
        dm{i,6}=4;
    elseif strcmp('moderate',dm{i,6})==1
        dm{i,6}=3;
    elseif strcmp('little',dm{i,6})==1
        dm{i,6}=2;
    elseif strcmp('NA',dm{i,6})==1
        dm{i,6}=1;
    else
        dm{i,6}=0;
    end
end

% let feature 'checking accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; "1" presenting 'NA'
for i=1:length(dm)
```

```
    if strcmp('rich',dm{i,7})==1
        dm{i,7}=5;
    elseif strcmp('quite rich',dm{i,7})==1
        dm{i,7}=4;
    elseif strcmp('moderate',dm{i,7})==1
        dm{i,7}=3;
    elseif strcmp('little',dm{i,7})==1
        dm{i,7}=2;
    elseif strcmp('NA',dm{i,7})==1
        dm{i,7}=1;
    else
        dm{i,7}=0;
    end
end

% let feature 'purpose' be value from 8 to 1
% '[1 0 0 0 0 0 0 0]' presenting 'business'; '[0 1 0 0 0 0 0 0]' presenting
'education'
% '[0 0 1 0 0 0 0 0]' presenting 'car'; '[0 0 0 1 0 0 0 0]' presenting
'vacation/others'
% '[0 0 0 0 1 0 0 0]' presenting 'furniture/equipment'; '[0 0 0 0 0 1 0 0]'
presenting 'radio/TV'
% '[0 0 0 0 0 0 1 0]' presenting 'repairs'; '[0 0 0 0 0 0 0 1]' presenting 'domestic
appliances'
for i=1:length(dm)
    if strcmp('business',dm{i,10})==1
        dm{i,10}=[1 0 0 0 0 0 0 0];
    elseif strcmp('education',dm{i,10})==1
        dm{i,10}=[0 1 0 0 0 0 0 0];
    elseif strcmp('car',dm{i,10})==1
        dm{i,10}=[0 0 1 0 0 0 0 0];
    elseif strcmp('vacation/others',dm{i,10})==1
        dm{i,10}=[0 0 0 1 0 0 0 0];
    elseif strcmp('furniture/equipment',dm{i,10})==1
        dm{i,10}=[0 0 0 0 1 0 0 0];
    elseif strcmp('domestic appliances',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 1 0 0];
    elseif strcmp('radio/TV',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 1 0];
    elseif strcmp('repairs',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 0 1];
    end
end
% Make Dataset into Matrix Form
dm= dataset2cell(dm);
dm(1,:) = [];
dm=cell2mat(dm);
%delete the first feature, it is just a index for appliances.
dm(:,1)=[];
```

```
dm_wt_label= dm;
dm_wt_label(:,20)=[];

%% Randomly Generate 200 samples as test dataset, 800 samples as traing dataset
r1=randperm(1000,200);
for i=1:200
per20_test(i,:)=dm(r1(i),:);
end
per20_test_without_label=per20_test;
per20_test_without_label(:,20)=[];
per80_train=dm;
per80_train(r1,:)=[];
per80_train_without_label=per80_train;
per80_train_without_label(:,20)=[];


%% Naive Bayers Classifier for non-nomalized data. One-Pass Accuracy estimation
model=fitcnb(per80_train_without_label, per80_train(:,20),'Distribution','normal');
t1 = predict(model,per20_test_without_label);
fprintf('One-Path Accuracy for Naive Bayers Classifier without normalization: %5.3
f\n',sum(per20_test(:,20)==t1)/200);


%% QDA Classifier
model=fitcdiscr(per80_train_without_label,per80_train(:,
20),'DiscrimType','pseudoQuadratic');
t3 = predict(model,per20_test_without_label);
fprintf('One-Path Accuracy for for QDA without normalization: %5.3f\n',sum(per20_test
(:,20)==t3)/200);



%% normalization
feat_mean=mean(dm);
std_vec_dm=std(dm);
dm_normalized=zeros(1000,20);
for j=1:size(dm,2)
    dm_normalized(:,j)= (dm(:,j)-feat_mean(j))./std_vec_dm(j);%make data zero mean
and std of 1
end
clear j feat_mean std_vec_dm

class1_dm=dm_normalized; %building class 1 dataset
k=0;
for i=1:1000
    if dm(i,20)==2
        k=k+1;
        l(k)=i;
    end
```

```
end
class1_dm(l,:)=[];
clear k l

class2_dm=dm_normalized; %building class 2 dataset
k=0;
for i=1:1000
    if dm(i,20)==1
        k=k+1;
        l(k)=i;
    end
end
class2_dm(l,:)=[];
clear k l
%% Naive Bayers Classifier for nomalized data. One-Pass Accuracy estimation
r1=randperm(1000,200);
for i=1:200
per20_test(i,:)=dm_normalized(r1(i),:);
per20_test_label(i)= dm(r1(i),20);
end
per20_test_without_label=per20_test;
per20_test_without_label(:,20)=[];
per80_train=dm_normalized;
per80_train(r1,:)=[];
per80_train_without_label=per80_train;
per80_train_without_label(:,20)=[];
per80_train_label= dm(:,20);
per80_train_label(r1,:)=[] ;
model=fitcnb(per80_train_without_label, per80_train(:,20),'Distribution','normal');
t1 = predict(model,per20_test_without_label);
fprintf('One-Path Accuracy for Naive Bayers Classifier with normalization: %5.3f\n',
sum(per20_test(:,20)==t1)/200);
%% Naive Bayers Classifier , F1 score for non normalized data
model=fitcnb(dm_wt_label, dm(:,20),'Distribution','normal');
t_total = predict(model, dm_wt_label);
CM=confusionmat(dm(:,20),t_total);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for Naive Bayers Classifier without Normalized Dataset is %5.3
f\n\n',F1_Score)
%% QDA Classifier , F1 score for non normalized data
model=fitcdiscr(dm_wt_label, dm(:,20),'DiscrimType','pseudoQuadratic');
t_total = predict(model, dm_wt_label);
CM=confusionmat(dm(:,20),t_total);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
```

```
F1_Score=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for QDA Classifier without Normalized Dataset is %5.3f\n\n',F1_Score)

%% Naive Bayers Classifier , F1 score for normalized data
dm_normalized(:,20)=[];
model=fitcnb(dm_normalized, dm(:,20),'Distribution','normal');
t_total = predict(model, dm_normalized);
CM=confusionmat(dm(:,20),t_total);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for Naive Bayers Classifier with Normalized Dataset is %5.3f\n\n',
F1_Score)
%%  Cross-Validation of 10-folds, 20 times with constraint of same percentage of
class, 20 times for Performace Estimation for Naive Bayers Classifier with normalized
data
crs_vrid_perfm1=0;
class1_dm(:,20)=[];
class2_dm(:,20)=[];
for j=1:20
rv1_test= randperm(size(class1_dm,1),700);
rv2_test= randperm(size(class2_dm,1),300);
for i=1:70
    D1_c1(i,:)= class1_dm(rv1_test(i),:);
    label_D1_c1(i)= 1;
end
for i=1:30
    D1_c2(i,:)= class2_dm(rv2_test(i),:);
    label_D1_c2(i)= 2;
end
D1=[D1_c1;D1_c2];
label_D1=[label_D1_c1';label_D1_c2'];
D_set{1,1}=D1;
D_set{2,1}=label_D1;
for i=1:70
    D2_c1(i,:)= class1_dm(rv1_test(i+70),:);
    label_D2_c1(i)= 1;
end
for i=1:30
    D2_c2(i,:)= class2_dm(rv2_test(i+30),:);
    label_D2_c2(i)= 2;
end
D2=[D2_c1;D2_c2];
label_D2=[label_D2_c1';label_D2_c2'];
D_set{1,2}=D2;
D_set{2,2}=label_D2;
for i=1:70
```

```
    D3_c1(i,:)= class1_dm(rv1_test(i+140),:);
    label_D3_c1(i)= 1;
end
for i=1:30
    D3_c2(i,:)= class2_dm(rv2_test(i+60),:);
    label_D3_c2(i)= 2;
end
D3=[D3_c1;D3_c2];
label_D3=[label_D3_c1';label_D3_c2'];
D_set{1,3}=D3;
D_set{2,3}=label_D3;
for i=1:70
    D4_c1(i,:)= class1_dm(rv1_test(i+210),:);
    label_D4_c1(i)= 1;
end
for i=1:30
    D4_c2(i,:)= class2_dm(rv2_test(i+90),:);
    label_D4_c2(i)= 2;
end
D4=[D4_c1;D4_c2];
label_D4=[label_D4_c1';label_D4_c2'];
D_set{1,4}=D4;
D_set{2,4}=label_D4;

for i=1:70
    D5_c1(i,:)= class1_dm(rv1_test(i+280),:);
    label_D5_c1(i)= 1;
end
for i=1:30
    D5_c2(i,:)= class2_dm(rv2_test(i+120),:);
    label_D5_c2(i)= 2;
end
D5=[D5_c1;D5_c2];
label_D5=[label_D5_c1';label_D5_c2'];
D_set{1,5}=D5;
D_set{2,5}=label_D5;

for i=1:70
    D6_c1(i,:)= class1_dm(rv1_test(i+350),:);
    label_D6_c1(i)= 1;
end
for i=1:30
    D6_c2(i,:)= class2_dm(rv2_test(i+150),:);
    label_D6_c2(i)= 2;
end
D6=[D6_c1;D6_c2];
label_D6=[label_D6_c1';label_D6_c2'];
D_set{1,6}=D6;
D_set{2,6}=label_D6;
```

```
for i=1:70
    D7_c1(i,:)= class1_dm(rv1_test(i+420),:);
    label_D7_c1(i)= 1;
end
for i=1:30
    D7_c2(i,:)= class2_dm(rv2_test(i+180),:);
    label_D7_c2(i)= 2;
end
D7=[D7_c1;D7_c2];
label_D7=[label_D7_c1';label_D7_c2'];
D_set{1,7}=D7;
D_set{2,7}=label_D7;
for i=1:70
    D8_c1(i,:)= class1_dm(rv1_test(i+490),:);
    label_D8_c1(i)= 1;
end
for i=1:30
    D8_c2(i,:)= class2_dm(rv2_test(i+210),:);
    label_D8_c2(i)= 2;
end
D8=[D8_c1;D8_c2];
label_D8=[label_D8_c1';label_D8_c2'];
D_set{1,8}=D8;
D_set{2,8}=label_D8;
for i=1:70
    D9_c1(i,:)= class1_dm(rv1_test(i+560),:);
    label_D9_c1(i)= 1;
end
for i=1:30
    D9_c2(i,:)= class2_dm(rv2_test(i+240),:);
    label_D9_c2(i)= 2;
end
D9=[D9_c1;D9_c2];
label_D9=[label_D9_c1';label_D9_c2'];
D_set{1,9}=D9;
D_set{2,9}=label_D9;

for i=1:70
    D10_c1(i,:)= class1_dm(rv1_test(i+630),:);
    label_D10_c1(i)= 1;
end
for i=1:30
    D10_c2(i,:)= class2_dm(rv2_test(i+270),:);
    label_D10_c2(i)= 2;
end
D10=[D10_c1;D10_c2];
label_D10=[label_D10_c1';label_D10_c2'];
D_set{1,10}=D10;
```

```
D_set{2,10}=label_D10;
clear D1_c1 D1_c2 D2_c1 D2_c2 D3_c1 D3_c2 D4_c1 D4_c2 D5_c1 D5_c2 D6_c1 D6_c2 D7_c1
D7_c2 D8_c1 D8_c2 D9_c1 D9_c2 D10_c1 D10_c2
clear label_D1_c1  label_D1_c2 label_D2_c1  label_D2_c2 label_D3_c1  label_D3_c2
label_D4_c1  label_D4_c2 label_D5_c1  label_D5_c2 label_D6_c1  label_D6_c2
label_D7_c1  label_D7_c2 label_D8_c1  label_D8_c2 label_D9_c1  label_D9_c2
label_D10_c1  label_D10_c2





total_classfication_rate=0;
for i=1:10
model=fitcnb([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)
+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set
{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}],    [D_set{2,(mod
(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)
+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set
{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}],'distribution','normal');
t = predict(model, D_set{1,(mod(i+9,10)+1)});
m=mean(D_set{2,(mod(i+9,10)+1)}==t);
total_classfication_rate=total_classfication_rate+m;
end
crs_vrid_perfm=total_classfication_rate/10;
crs_vrid_perfm1=crs_vrid_perfm+crs_vrid_perfm1;

end
fprintf('Cross-Validation of 10-folds,20 times of Performace Estimation for naive
bayers classifier with normalization: %5.3f\n',crs_vrid_perfm1/20);
%clear k_loss k_loss_i r1 per20_test per20_test_without_label
per80_train_without_label per80_train kloss_i kloss dm_wt_label



%%  Cross-Validation of 10-folds, 20 times with constraint of same percentage of
class, 20 times for Performace Estimation for naive bayers with normalization data
without normalized data
crs_vrid_perfm1=0;


class1_dm=dm; %building class 1 dataset
k=0;
for i=1:1000
    if dm(i,20)==2
        k=k+1;
        l(k)=i;
    end
end
```

```
class1_dm(l,:)=[];
clear k l

class2_dm=dm; %building class 2 dataset
k=0;
for i=1:1000
    if dm(i,20)==1
        k=k+1;
        l(k)=i;
    end
end
class2_dm(l,:)=[];
clear k l


class1_dm(:,20)=[];
class2_dm(:,20)=[];
for j=1:20
rv1_test= randperm(size(class1_dm,1),700);
rv2_test= randperm(size(class2_dm,1),300);
for i=1:70
    D1_c1(i,:)= class1_dm(rv1_test(i),:);
    label_D1_c1(i)= 1;
end
for i=1:30
    D1_c2(i,:)= class2_dm(rv2_test(i),:);
    label_D1_c2(i)= 2;
end
D1=[D1_c1;D1_c2];
label_D1=[label_D1_c1';label_D1_c2'];
D_set{1,1}=D1;
D_set{2,1}=label_D1;
for i=1:70
    D2_c1(i,:)= class1_dm(rv1_test(i+70),:);
    label_D2_c1(i)= 1;
end
for i=1:30
    D2_c2(i,:)= class2_dm(rv2_test(i+30),:);
    label_D2_c2(i)= 2;
end
D2=[D2_c1;D2_c2];
label_D2=[label_D2_c1';label_D2_c2'];
D_set{1,2}=D2;
D_set{2,2}=label_D2;
for i=1:70
    D3_c1(i,:)= class1_dm(rv1_test(i+140),:);
    label_D3_c1(i)= 1;
end
for i=1:30
```

```
    D3_c2(i,:)= class2_dm(rv2_test(i+60),:);
    label_D3_c2(i)= 2;
end
D3=[D3_c1;D3_c2];
label_D3=[label_D3_c1';label_D3_c2'];
D_set{1,3}=D3;
D_set{2,3}=label_D3;
for i=1:70
    D4_c1(i,:)= class1_dm(rv1_test(i+210),:);
    label_D4_c1(i)= 1;
end
for i=1:30
    D4_c2(i,:)= class2_dm(rv2_test(i+90),:);
    label_D4_c2(i)= 2;
end
D4=[D4_c1;D4_c2];
label_D4=[label_D4_c1';label_D4_c2'];
D_set{1,4}=D4;
D_set{2,4}=label_D4;

for i=1:70
    D5_c1(i,:)= class1_dm(rv1_test(i+280),:);
    label_D5_c1(i)= 1;
end
for i=1:30
    D5_c2(i,:)= class2_dm(rv2_test(i+120),:);
    label_D5_c2(i)= 2;
end
D5=[D5_c1;D5_c2];
label_D5=[label_D5_c1';label_D5_c2'];
D_set{1,5}=D5;
D_set{2,5}=label_D5;

for i=1:70
    D6_c1(i,:)= class1_dm(rv1_test(i+350),:);
    label_D6_c1(i)= 1;
end
for i=1:30
    D6_c2(i,:)= class2_dm(rv2_test(i+150),:);
    label_D6_c2(i)= 2;
end
D6=[D6_c1;D6_c2];
label_D6=[label_D6_c1';label_D6_c2'];
D_set{1,6}=D6;
D_set{2,6}=label_D6;

for i=1:70
    D7_c1(i,:)= class1_dm(rv1_test(i+420),:);
    label_D7_c1(i)= 1;
```

```
end
for i=1:30
    D7_c2(i,:)= class2_dm(rv2_test(i+180),:);
    label_D7_c2(i)= 2;
end
D7=[D7_c1;D7_c2];
label_D7=[label_D7_c1';label_D7_c2'];
D_set{1,7}=D7;
D_set{2,7}=label_D7;
for i=1:70
    D8_c1(i,:)= class1_dm(rv1_test(i+490),:);
    label_D8_c1(i)= 1;
end
for i=1:30
    D8_c2(i,:)= class2_dm(rv2_test(i+210),:);
    label_D8_c2(i)= 2;
end
D8=[D8_c1;D8_c2];
label_D8=[label_D8_c1';label_D8_c2'];
D_set{1,8}=D8;
D_set{2,8}=label_D8;
for i=1:70
    D9_c1(i,:)= class1_dm(rv1_test(i+560),:);
    label_D9_c1(i)= 1;
end
for i=1:30
    D9_c2(i,:)= class2_dm(rv2_test(i+240),:);
    label_D9_c2(i)= 2;
end
D9=[D9_c1;D9_c2];
label_D9=[label_D9_c1';label_D9_c2'];
D_set{1,9}=D9;
D_set{2,9}=label_D9;

for i=1:70
    D10_c1(i,:)= class1_dm(rv1_test(i+630),:);
    label_D10_c1(i)= 1;
end
for i=1:30
    D10_c2(i,:)= class2_dm(rv2_test(i+270),:);
    label_D10_c2(i)= 2;
end
D10=[D10_c1;D10_c2];
label_D10=[label_D10_c1';label_D10_c2'];
D_set{1,10}=D10;
D_set{2,10}=label_D10;
clear D1_c1 D1_c2 D2_c1 D2_c2 D3_c1 D3_c2 D4_c1 D4_c2 D5_c1 D5_c2 D6_c1 D6_c2 D7_c1↙
D7_c2 D8_c1 D8_c2 D9_c1 D9_c2 D10_c1 D10_c2
clear label_D1_c1  label_D1_c2 label_D2_c1  label_D2_c2 label_D3_c1  label_D3_c2↙
```

```
label_D4_c1 label_D4_c2 label_D5_c1 label_D5_c2 label_D6_c1 label_D6_c2
label_D7_c1 label_D7_c2 label_D8_c1 label_D8_c2 label_D9_c1 label_D9_c2
label_D10_c1 label_D10_c2
total_classfication_rate=0;
for i=1:10
model=fitcnb([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)
+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set
{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}],    [D_set{2,(mod
(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)
+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set
{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}],'distribution','normal');
t = predict(model, D_set{1,(mod(i+9,10)+1)});
m=mean(D_set{2,(mod(i+9,10)+1)}==t);
total_classfication_rate=total_classfication_rate+m;
end
crs_vrid_perfm=total_classfication_rate/10;
crs_vrid_perfm1=crs_vrid_perfm+crs_vrid_perfm1;

end
fprintf('Cross-Validation of 10-folds,20 times of Performace Estimation for Naive
bayers without normalization: %5.3f\n',crs_vrid_perfm1/20);


%%  Cross-Validation of 10-folds, 20 times with constraint of same percentage of
class, 20 times for Performace Estimation for QDA without normalized data
crs_vrid_perfm1=0;


class1_dm=dm; %building class 1 dataset
k=0;
for i=1:1000
    if dm(i,20)==2
        k=k+1;
        l(k)=i;
    end
end
class1_dm(l,:)=[];
clear k l

class2_dm=dm; %building class 2 dataset
k=0;
for i=1:1000
    if dm(i,20)==1
        k=k+1;
        l(k)=i;
    end
end
class2_dm(l,:)=[];
clear k l
```

```
class1_dm(:,20)=[];
class2_dm(:,20)=[];
for j=1:20
rv1_test= randperm(size(class1_dm,1),700);
rv2_test= randperm(size(class2_dm,1),300);
for i=1:70
    D1_c1(i,:)= class1_dm(rv1_test(i),:);
    label_D1_c1(i)= 1;
end
for i=1:30
    D1_c2(i,:)= class2_dm(rv2_test(i),:);
    label_D1_c2(i)= 2;
end
D1=[D1_c1;D1_c2];
label_D1=[label_D1_c1';label_D1_c2'];
D_set{1,1}=D1;
D_set{2,1}=label_D1;
for i=1:70
    D2_c1(i,:)= class1_dm(rv1_test(i+70),:);
    label_D2_c1(i)= 1;
end
for i=1:30
    D2_c2(i,:)= class2_dm(rv2_test(i+30),:);
    label_D2_c2(i)= 2;
end
D2=[D2_c1;D2_c2];
label_D2=[label_D2_c1';label_D2_c2'];
D_set{1,2}=D2;
D_set{2,2}=label_D2;
for i=1:70
    D3_c1(i,:)= class1_dm(rv1_test(i+140),:);
    label_D3_c1(i)= 1;
end
for i=1:30
    D3_c2(i,:)= class2_dm(rv2_test(i+60),:);
    label_D3_c2(i)= 2;
end
D3=[D3_c1;D3_c2];
label_D3=[label_D3_c1';label_D3_c2'];
D_set{1,3}=D3;
D_set{2,3}=label_D3;
for i=1:70
    D4_c1(i,:)= class1_dm(rv1_test(i+210),:);
    label_D4_c1(i)= 1;
end
for i=1:30
    D4_c2(i,:)= class2_dm(rv2_test(i+90),:);
```

```
    label_D4_c2(i)= 2;
end
D4=[D4_c1;D4_c2];
label_D4=[label_D4_c1';label_D4_c2'];
D_set{1,4}=D4;
D_set{2,4}=label_D4;

for i=1:70
    D5_c1(i,:)=class1_dm(rv1_test(i+280),:);
    label_D5_c1(i)= 1;
end
for i=1:30
    D5_c2(i,:)=class2_dm(rv2_test(i+120),:);
    label_D5_c2(i)= 2;
end
D5=[D5_c1;D5_c2];
label_D5=[label_D5_c1';label_D5_c2'];
D_set{1,5}=D5;
D_set{2,5}=label_D5;

for i=1:70
    D6_c1(i,:)=class1_dm(rv1_test(i+350),:);
    label_D6_c1(i)= 1;
end
for i=1:30
    D6_c2(i,:)=class2_dm(rv2_test(i+150),:);
    label_D6_c2(i)= 2;
end
D6=[D6_c1;D6_c2];
label_D6=[label_D6_c1';label_D6_c2'];
D_set{1,6}=D6;
D_set{2,6}=label_D6;

for i=1:70
    D7_c1(i,:)=class1_dm(rv1_test(i+420),:);
    label_D7_c1(i)= 1;
end
for i=1:30
    D7_c2(i,:)=class2_dm(rv2_test(i+180),:);
    label_D7_c2(i)= 2;
end
D7=[D7_c1;D7_c2];
label_D7=[label_D7_c1';label_D7_c2'];
D_set{1,7}=D7;
D_set{2,7}=label_D7;
for i=1:70
    D8_c1(i,:)=class1_dm(rv1_test(i+490),:);
    label_D8_c1(i)= 1;
end
```

```
for i=1:30
    D8_c2(i,:)= class2_dm(rv2_test(i+210),:);
    label_D8_c2(i)= 2;
end
D8=[D8_c1;D8_c2];
label_D8=[label_D8_c1';label_D8_c2'];
D_set{1,8}=D8;
D_set{2,8}=label_D8;
for i=1:70
    D9_c1(i,:)= class1_dm(rv1_test(i+560),:);
    label_D9_c1(i)= 1;
end
for i=1:30
    D9_c2(i,:)= class2_dm(rv2_test(i+240),:);
    label_D9_c2(i)= 2;
end
D9=[D9_c1;D9_c2];
label_D9=[label_D9_c1';label_D9_c2'];
D_set{1,9}=D9;
D_set{2,9}=label_D9;

for i=1:70
    D10_c1(i,:)= class1_dm(rv1_test(i+630),:);
    label_D10_c1(i)= 1;
end
for i=1:30
    D10_c2(i,:)= class2_dm(rv2_test(i+270),:);
    label_D10_c2(i)= 2;
end
D10=[D10_c1;D10_c2];
label_D10=[label_D10_c1';label_D10_c2'];
D_set{1,10}=D10;
D_set{2,10}=label_D10;
clear D1_c1 D1_c2 D2_c1 D2_c2 D3_c1 D3_c2 D4_c1 D4_c2 D5_c1 D5_c2 D6_c1 D6_c2 D7_c1
D7_c2 D8_c1 D8_c2 D9_c1 D9_c2 D10_c1 D10_c2
clear label_D1_c1  label_D1_c2 label_D2_c1  label_D2_c2 label_D3_c1  label_D3_c2
label_D4_c1  label_D4_c2 label_D5_c1  label_D5_c2 label_D6_c1  label_D6_c2
label_D7_c1  label_D7_c2 label_D8_c1  label_D8_c2 label_D9_c1  label_D9_c2
label_D10_c1  label_D10_c2
total_classfication_rate=0;
for i=1:10
model=fitcdiscr([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)
+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set
{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}],    [D_set{2,(mod
(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)
+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set
{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}],'DiscrimType','pseudoQuadratic');
t = predict(model, D_set{1,(mod(i+9,10)+1)});
m=mean(D_set{2,(mod(i+9,10)+1)}==t);
```

```
total_classfication_rate=total_classfication_rate+m;
end
crs_vrid_perfm=total_classfication_rate/10;
crs_vrid_perfm1=crs_vrid_perfm+crs_vrid_perfm1;

end
fprintf('Cross-Validation of 10-folds,20 times of Performace Estimation for QDA
without normalization: %5.3f\n',crs_vrid_perfm1/20);
```

```
%% Final_Project_for_EE559 File 2

%% change categoriclas features into corresponding numerical value AND make Dataset
into matrix

clc;
clear all;
% '[0 1]' presenting 'male'; '[1 0]'presenting 'female'
dm = dataset('File','Proj_dataset_1.csv','Delimiter',',');
for i=1:length(dm)
    if length(dm{i,3})==4
        dm{i,3}= [0 1];
    elseif length(dm{i,3})==6
        dm{i,3}=[1 0];
    end
end

% '[0 0 1]' represting 'free'; '[0 1 0]' presenting 'rent'; '[1 0 0]' presenting
'own'
for i=1:length(dm)
    if length(dm{i,5})==3
        dm{i,5}=[1 0 0];
    elseif length(dm{i,5})==4
        if  dm{i,5}=='rent'
            dm{i,5}=[0 1 0];
        else
            dm{i,5}=[0 0 1];
        end
    end
end

% let feature 'saving accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; MEAN presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,6})==1
        dm{i,6}=5;
    elseif strcmp('quite rich',dm{i,6})==1
        dm{i,6}=4;
    elseif strcmp('moderate',dm{i,6})==1
        dm{i,6}=3;
    elseif strcmp('little',dm{i,6})==1
        dm{i,6}=2;
    elseif strcmp('NA',dm{i,6})==1
        dm{i,6}=1;
    else
        dm{i,6}=0;
    end
end
```

```
k=0;
c=0;
for i=1:length(dm)
    if dm{i,6}~=1
        k=k+dm{i,6};
        c=c+1;
    end
end

for i=1:length(dm)
    if dm{i,6}==1
        dm{i,6}=k/c; %replace 'NA' with the mean of this feature
    end
end
% let feature 'checking accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; MEAN presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,7})==1
        dm{i,7}=5;
    elseif strcmp('quite rich',dm{i,7})==1
        dm{i,7}=4;
    elseif strcmp('moderate',dm{i,7})==1
        dm{i,7}=3;
    elseif strcmp('little',dm{i,7})==1
        dm{i,7}=2;
    elseif strcmp('NA',dm{i,7})==1
        dm{i,7}=1;
    else
        dm{i,7}=0;
    end
end

k=0;
c=0;
for i=1:length(dm)
    if dm{i,7}~=1
        k=k+dm{i,7};
        c=c+1;
    end
end

for i=1:length(dm)
    if dm{i,7}==1
        dm{i,7}=k/c; %replace 'NA' with the mean of this feature
    end
end
% let feature 'purpose' be value from 8 to 1
% '[1 0 0 0 0 0 0 0]' presenting 'business'; '[0 1 0 0 0 0 0 0]' presenting
```

```
'education'
% '[0 0 1 0 0 0 0 0]' presenting 'car'; '[0 0 0 1 0 0 0 0]' presenting
'vacation/others'
% '[0 0 0 0 1 0 0 0]' presenting 'furniture/equipment'; '[0 0 0 0 0 1 0 0]'
presenting 'radio/TV'
% '[0 0 0 0 0 0 1 0]' presenting 'repairs'; '[0 0 0 0 0 0 0 1]' presenting 'domestic
appliances'
for i=1:length(dm)
    if strcmp('business',dm{i,10})==1
        dm{i,10}=[1 0 0 0 0 0 0 0];
    elseif strcmp('education',dm{i,10})==1
        dm{i,10}=[0 1 0 0 0 0 0 0];
    elseif strcmp('car',dm{i,10})==1
        dm{i,10}=[0 0 1 0 0 0 0 0];
    elseif strcmp('vacation/others',dm{i,10})==1
        dm{i,10}=[0 0 0 1 0 0 0 0];
    elseif strcmp('furniture/equipment',dm{i,10})==1
        dm{i,10}=[0 0 0 0 1 0 0 0];
    elseif strcmp('domestic appliances',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 1 0 0];
    elseif strcmp('radio/TV',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 1 0];
    elseif strcmp('repairs',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 0 1];
    end
end
% Make Dataset into Matrix Form
dm= dataset2cell(dm);
dm(1,:) = [];
dm=cell2mat(dm);
%delete the first feature, it is just a index for appliances.
dm(:,1)=[];
dm_wt_label= dm;
dm_wt_label(:,20)=[];
%% MSE/Perceptron Learning and Using one vs.rest/one vs.one method with normalized
dataset, randomly choosing 20% of data as Test dataset
feat_mean=mean(dm);
std_vec_dm=std(dm);
dm_normalized=zeros(1000,20);
for j=1:size(dm,2)
    dm_normalized(:,j)= (dm(:,j)-feat_mean(j))./std_vec_dm(j);%make data zero mean
and std of 1
end
clear j feat_mean std_vec_dm
dm_normalized(:,20)=[]; % This is just class labels, not a feature. It need to be
discarded. When we need to use labels, we can then pick this for use
randm_20_percent_vec=randperm(size(dm,1),size(dm,1)*0.2);%ramdomly generate a vector
for picking up 20% of dataset for test dataset
traing_dm= dm_normalized;
```

```matlab
for j=1:size(dm,1)*0.2
    test_dm(j,:)=dm_normalized(randm_20_percent_vec(j),:);% Randomly generate 20% of data for Test Dataset
    label_test(j,1)= dm(randm_20_percent_vec(j),20);%Build test Labels
end
clear i j;
traing_dm(randm_20_percent_vec,:)=[]; % The rest of the data are for training
label_traing=dm(:,20);
label_traing(randm_20_percent_vec,:)=[]; % Build Traing Labels
t1= multiclass(traing_dm',label_traing',test_dm','[''OAA'',0,''LS_modified'',[]]');
fprintf('Normalized Dataset with zero mean and std of 1 with 20 percent of data set aside as test dataset: \n')
fprintf('Correct Classification rate for MSE with one vs.rest method is %6.5f\n',mean(t1==label_test'))
t4= multiclass(traing_dm',label_traing',test_dm','[''all-pairs'',0,''LS_modified'',[]]');
fprintf('Correct Classification rate for MSE with one vs.one method is %6.5f\n',mean(t4==label_test'))
t2= multiclass(traing_dm',label_traing',test_dm','[''OAA'',0,''Perceptron'',[]]');
fprintf('Correct Classification rate for perceptron with one vs.rest method is %6.5f\n',mean(t2==label_test'))
t3= multiclass(traing_dm',label_traing',test_dm','[''all-pairs'',0,''Perceptron'',[]]');
fprintf('Correct Classification rate for perceptron with one vs.one method is %6.5f\n\n',mean(t3==label_test'))

%% Caculating F1 Score
t_total1= multiclass(dm_normalized',dm(:,20)',dm_normalized','[''OAA'',0,''LS_modified'',[]]');
t_total1=t_total1';
CM=confusionmat(dm(:,20),t_total1);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for MSE (One vs. Rest) is %5.3f\n',F1_Score)

t_total2= multiclass(dm_normalized',dm(:,20)',dm_normalized','[''all-pairs'',0,''Perceptron'',[]]');
t_total2=t_total2';
CM=confusionmat(dm(:,20),t_total2);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score2=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for MSE (One vs.One) is %5.3f\n',F1_Score2)

t_total3= multiclass(dm_normalized',dm(:,20)',dm_normalized','[''OAA'',0,''Perceptron'',[]]');
```

```
t_total3=t_total3';
CM=confusionmat(dm(:,20),t_total3);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score3=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for Perceptron (One vs. Rest) is %5.3f\n',F1_Score3)

t_total4= multiclass(dm_normalized',dm(:,20)',dm_normalized',['''all-pairs'',
0,''LS_modified'',[]]');
t_total4=t_total4';
CM=confusionmat(dm(:,20),t_total4);
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score4=mean(f1Score_eachclass); % F1 score for all classes from reference [3]
fprintf('F1 Score for Perceptron (One vs. ONE) is %5.3f\n\n',F1_Score4)




%% MSE/Perceptron Learning and Using one vs.rest/one vs.one method with normalized
dataset, using cross-validation for estimation of perfprmance (with 10 subsets,
subject to constrain that percentage of samples in each class is unchanged in each
subset)

class1_dm=dm_normalized; %building class 1 dataset
k=1;
for i=1:size(dm_normalized,1)
    if dm(i,20)==2
        l(k)=i;
        k=k+1;
    end
end
class1_dm(l,:)=[];
clear k
class2_dm=dm_normalized; %building class 2 dataset
k=1;
for i=1:size(dm_normalized,1)
    if dm(i,20)==1
        y(k)=i;
        k=k+1;
    end
end
class2_dm(y,:)=[];
clear k

crs_vrid_perfm1_1=0;
```

```
crs_vrid_perfm2_1=0;
crs_vrid_perfm3_1=0;
crs_vrid_perfm4_1=0;
for j=1:20
rv1_test= randperm(size(class1_dm,1),700);
rv2_test= randperm(size(class2_dm,1),300);
for i=1:70
    D1_c1(i,:)= class1_dm(rv1_test(i),:);
    label_D1_c1(i)= 1;
end
for i=1:30
    D1_c2(i,:)= class2_dm(rv2_test(i),:);
    label_D1_c2(i)= 2;
end
D1=[D1_c1;D1_c2];
label_D1=[label_D1_c1';label_D1_c2'];
D_set{1,1}=D1;
D_set{2,1}=label_D1;
for i=1:70
    D2_c1(i,:)= class1_dm(rv1_test(i+70),:);
    label_D2_c1(i)= 1;
end
for i=1:30
    D2_c2(i,:)= class2_dm(rv2_test(i+30),:);
    label_D2_c2(i)= 2;
end
D2=[D2_c1;D2_c2];
label_D2=[label_D2_c1';label_D2_c2'];
D_set{1,2}=D2;
D_set{2,2}=label_D2;
for i=1:70
    D3_c1(i,:)= class1_dm(rv1_test(i+140),:);
    label_D3_c1(i)= 1;
end
for i=1:30
    D3_c2(i,:)= class2_dm(rv2_test(i+60),:);
    label_D3_c2(i)= 2;
end
D3=[D3_c1;D3_c2];
label_D3=[label_D3_c1';label_D3_c2'];
D_set{1,3}=D3;
D_set{2,3}=label_D3;
for i=1:70
    D4_c1(i,:)= class1_dm(rv1_test(i+210),:);
    label_D4_c1(i)= 1;
end
for i=1:30
    D4_c2(i,:)= class2_dm(rv2_test(i+90),:);
    label_D4_c2(i)= 2;
```

```
end
D4=[D4_c1;D4_c2];
label_D4=[label_D4_c1';label_D4_c2'];
D_set{1,4}=D4;
D_set{2,4}=label_D4;

for i=1:70
    D5_c1(i,:)= class1_dm(rv1_test(i+280),:);
    label_D5_c1(i)= 1;
end
for i=1:30
    D5_c2(i,:)= class2_dm(rv2_test(i+120),:);
    label_D5_c2(i)= 2;
end
D5=[D5_c1;D5_c2];
label_D5=[label_D5_c1';label_D5_c2'];
D_set{1,5}=D5;
D_set{2,5}=label_D5;

for i=1:70
    D6_c1(i,:)= class1_dm(rv1_test(i+350),:);
    label_D6_c1(i)= 1;
end
for i=1:30
    D6_c2(i,:)= class2_dm(rv2_test(i+150),:);
    label_D6_c2(i)= 2;
end
D6=[D6_c1;D6_c2];
label_D6=[label_D6_c1';label_D6_c2'];
D_set{1,6}=D6;
D_set{2,6}=label_D6;

for i=1:70
    D7_c1(i,:)= class1_dm(rv1_test(i+420),:);
    label_D7_c1(i)= 1;
end
for i=1:30
    D7_c2(i,:)= class2_dm(rv2_test(i+180),:);
    label_D7_c2(i)= 2;
end
D7=[D7_c1;D7_c2];
label_D7=[label_D7_c1';label_D7_c2'];
D_set{1,7}=D7;
D_set{2,7}=label_D7;
for i=1:70
    D8_c1(i,:)= class1_dm(rv1_test(i+490),:);
    label_D8_c1(i)= 1;
end
for i=1:30
```

```
    D8_c2(i,:)= class2_dm(rv2_test(i+210),:);
    label_D8_c2(i)= 2;
end
D8=[D8_c1;D8_c2];
label_D8=[label_D8_c1';label_D8_c2'];
D_set{1,8}=D8;
D_set{2,8}=label_D8;
for i=1:70
    D9_c1(i,:)= class1_dm(rv1_test(i+560),:);
    label_D9_c1(i)= 1;
end
for i=1:30
    D9_c2(i,:)= class2_dm(rv2_test(i+240),:);
    label_D9_c2(i)= 2;
end
D9=[D9_c1;D9_c2];
label_D9=[label_D9_c1';label_D9_c2'];
D_set{1,9}=D9;
D_set{2,9}=label_D9;

for i=1:70
    D10_c1(i,:)= class1_dm(rv1_test(i+630),:);
    label_D10_c1(i)= 1;
end
for i=1:30
    D10_c2(i,:)= class2_dm(rv2_test(i+270),:);
    label_D10_c2(i)= 2;
end
D10=[D10_c1;D10_c2];
label_D10=[label_D10_c1';label_D10_c2'];
D_set{1,10}=D10;
D_set{2,10}=label_D10;
clear D1_c1 D1_c2 D2_c1 D2_c2 D3_c1 D3_c2 D4_c1 D4_c2 D5_c1 D5_c2 D6_c1 D6_c2 D7_c1
D7_c2 D8_c1 D8_c2 D9_c1 D9_c2 D10_c1 D10_c2
clear label_D1_c1  label_D1_c2 label_D2_c1  label_D2_c2 label_D3_c1  label_D3_c2
label_D4_c1  label_D4_c2 label_D5_c1  label_D5_c2 label_D6_c1  label_D6_c2
label_D7_c1  label_D7_c2 label_D8_c1  label_D8_c2 label_D9_c1  label_D9_c2
label_D10_c1  label_D10_c2

total_classfication_rate=0;

for i=1:10
t=    multiclass([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)
+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set
{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}]',[D_set{2,(mod
(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)
+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set
{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}]',D_set{1,(mod(i+9,10)+1)}','[''OAA'',
0,''LS_modified'',[]]');
```

```
m=mean(t==D_set{2,(mod(i+9,10)+1)}');
total_classfication_rate=total_classfication_rate+m;
end
crs_vrid_perfm1=total_classfication_rate/10;
crs_vrid_perfm1_1=crs_vrid_perfm1_1+crs_vrid_perfm1;

total_classfication_rate2=0;
for i=1:10
t2=    multiclass([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}]',[D_set{2,(mod(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}]',D_set{1,(mod(i+9,10)+1)}','[''OAA'',0,''Perceptron'',[]]');
m2=mean(t2==D_set{2,(mod(i+9,10)+1)}');
total_classfication_rate2=total_classfication_rate2+m2;
end
crs_vrid_perfm2=total_classfication_rate2/10;
crs_vrid_perfm2_1=crs_vrid_perfm2_1+crs_vrid_perfm2;

total_classfication_rate3=0;
for i=1:10
t3=    multiclass([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}]',[D_set{2,(mod(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}]',D_set{1,(mod(i+9,10)+1)}','[''all-pairs'',0,''Perceptron'',[]]');
m3=mean(t3==D_set{2,(mod(i+9,10)+1)}');
total_classfication_rate3=total_classfication_rate3+m3;
end
crs_vrid_perfm3=total_classfication_rate3/10;
crs_vrid_perfm3_1=crs_vrid_perfm3_1+crs_vrid_perfm3;

total_classfication_rate4=0;
for i=1:10
t4=    multiclass([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}]',[D_set{2,(mod(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}]',D_set{1,(mod(i+9,10)+1)}','[''all-pairs'',0,''LS_modified'',[]]');
m4=mean(t4==D_set{2,(mod(i+9,10)+1)}');
total_classfication_rate4=total_classfication_rate4+m4;
end
crs_vrid_perfm4=total_classfication_rate4/10;
```

```
crs_vrid_perfm4_1=crs_vrid_perfm4_1+crs_vrid_perfm4;

clear i m total_classfication_rate total_classfication_rate2
total_classfication_rate3  total_classfication_rate4
end
fprintf('Normalized Dataset with zero mean and std of 1 with cross-validation
estimation of performance: \n')
fprintf('Correct Classification rate for MSE with one vs.rest method is %6.5f\n',
crs_vrid_perfm1_1/20)
fprintf('Correct Classification rate for MSE with one vs.one method is %6.5f\n',
crs_vrid_perfm4_1/20)
fprintf('Correct Classification rate for perceptron with one vs.rest method is %6.5
f\n',crs_vrid_perfm2_1/20)
fprintf('Correct Classification rate for perceptron with one vs.one method is %6.5
f\n',crs_vrid_perfm3_1/20)
close all
```

```
%% change categoriclas features into corresponding numerical value AND make Dataset
into matrix
%% one-hot encoding
clc;
clear all
% '[0 1]' presenting 'male'; '[1 0]'presenting 'female'
dm = dataset('File','Proj_dataset_1.csv','Delimiter',',');
for i=1:length(dm)
    if length(dm{i,3})==4
        dm{i,3}= [0 1];
    elseif length(dm{i,3})==6
        dm{i,3}=[1 0];
    end
end

% '[0 0 1]' represting 'free'; '[0 1 0]' presenting 'rent'; '[1 0 0]' presenting
'own'
for i=1:length(dm)
    if length(dm{i,5})==3
        dm{i,5}=[1 0 0];
    elseif length(dm{i,5})==4
        if  dm{i,5}=='rent'
            dm{i,5}=[0 1 0];
        else
            dm{i,5}=[0 0 1];
        end
    end
end


% let feature 'saving accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; "1" presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,6})==1
        dm{i,6}=5;
    elseif strcmp('quite rich',dm{i,6})==1
        dm{i,6}=4;
    elseif strcmp('moderate',dm{i,6})==1
        dm{i,6}=3;
    elseif strcmp('little',dm{i,6})==1
        dm{i,6}=2;
    elseif strcmp('NA',dm{i,6})==1
        dm{i,6}=1;
    else
        dm{i,6}=0;
    end
end
```

```
% let feature 'checking accounts' be value from '5' to '1'
% '5' presenting 'rich'; '4' presenting 'quite rich';
% '3' presenting 'moderate'; '2' presnting 'little' ; "1" presenting 'NA'
for i=1:length(dm)
    if strcmp('rich',dm{i,7})==1
        dm{i,7}=5;
    elseif strcmp('quite rich',dm{i,7})==1
        dm{i,7}=4;
    elseif strcmp('moderate',dm{i,7})==1
        dm{i,7}=3;
    elseif strcmp('little',dm{i,7})==1
        dm{i,7}=2;
    elseif strcmp('NA',dm{i,7})==1
        dm{i,7}=1;
    else
        dm{i,7}=0;
    end
end

% let feature 'purpose' be value from 8 to 1
% '[1 0 0 0 0 0 0 0]' presenting 'business'; '[0 1 0 0 0 0 0 0]' presenting
'education'
% '[0 0 1 0 0 0 0 0]' presenting 'car'; '[0 0 0 1 0 0 0 0]' presenting
'vacation/others'
% '[0 0 0 0 1 0 0 0]' presenting 'furniture/equipment'; '[0 0 0 0 0 1 0 0]'
presenting 'radio/TV'
% '[0 0 0 0 0 0 1 0]' presenting 'repairs'; '[0 0 0 0 0 0 0 1]' presenting 'domestic
appliances'
for i=1:length(dm)
    if strcmp('business',dm{i,10})==1
        dm{i,10}=[1 0 0 0 0 0 0 0];
    elseif strcmp('education',dm{i,10})==1
        dm{i,10}=[0 1 0 0 0 0 0 0];
    elseif strcmp('car',dm{i,10})==1
        dm{i,10}=[0 0 1 0 0 0 0 0];
    elseif strcmp('vacation/others',dm{i,10})==1
        dm{i,10}=[0 0 0 1 0 0 0 0];
    elseif strcmp('furniture/equipment',dm{i,10})==1
        dm{i,10}=[0 0 0 0 1 0 0 0];
    elseif strcmp('domestic appliances',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 1 0 0];
    elseif strcmp('radio/TV',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 1 0];
    elseif strcmp('repairs',dm{i,10})==1
        dm{i,10}=[0 0 0 0 0 0 0 1];
    end
end
% Make Dataset into Matrix Form
dm= dataset2cell(dm);
```

```
dm(1,:) = [];
dm=cell2mat(dm);
%delete the first feature, it is just a index for appliances.
dm(:,1)=[];
dm(:,[1,6,7,9,10,11,13,14,16,17,18,19])=[];
dm_wt_label= dm;
dm_wt_label(:,8)=[];



%% Randomly Generate 200 samples as test dataset, 800 samples as traing dataset
r1=randperm(1000,200);
for i=1:200
per20_test(i,:)=dm(r1(i),:);
end
per20_test_without_label=per20_test;
per20_test_without_label(:,8)=[];
per80_train=dm;
per80_train(r1,:)=[];
per80_train_without_label=per80_train;
per80_train_without_label(:,8)=[];
%% QDA Classifier for non-nomalized data. One-Pass Accuracy estimation
model=fitcdiscr(per80_train_without_label, per80_train(:
8),'DiscrimType','pseudoQuadratic');
t3 = predict(model,per20_test_without_label);
fprintf('One-Path Accuracy for for QDA without normalization: %5.3f\n',sum(per20_test
(:,8)==t3)/200);

%% QDA Classifier , F1 score for non normalized data
model=fitcdiscr(dm_wt_label,dm(:,8),'DiscrimType','pseudoQuadratic');
t_total = predict(model, dm_wt_label);
CM=confusionmat(dm(:,8),t_total);%  from reference [2]
precision=diag(CM)./sum(CM,2);
recall=diag(CM)./sum(CM,1)';
f1Score_eachclass=2*(precision.*recall)./(precision+recall);%F1 Score for each class
F1_Score=mean(f1Score_eachclass); % F1 score for all classes
fprintf('F1 Score for QDA Classifier without Normalized Dataset is %5.3f\n\n',
F1_Score)

%%  Cross-Validation of 10-folds, 20 times with constraint of same percentage of
class, 20 times for Performace Estimation for QDA without normalized data
crs_vrid_perfm1=0;


class1_dm=dm; %building class 1dataset
k=0;
for i=1:1000
    if dm(i,8)==2
        k=k+1;
```

```
        l(k)=i;
    end
end
class1_dm(l,:)=[];
clear k l

class2_dm=dm; %building class 2 dataset
k=0;
for i=1:1000
    if dm(i,8)==1
        k=k+1;
        l(k)=i;
    end
end
class2_dm(l,:)=[];
clear k l


class1_dm(:,8)=[];
class2_dm(:,8)=[];
for j=1:20
rv1_test= randperm(size(class1_dm,1),700);
rv2_test= randperm(size(class2_dm,1),300);
for i=1:70
    D1_c1(i,:)= class1_dm(rv1_test(i),:);
    label_D1_c1(i)= 1;
end
for i=1:30
    D1_c2(i,:)= class2_dm(rv2_test(i),:);
    label_D1_c2(i)= 2;
end
D1=[D1_c1;D1_c2];
label_D1=[label_D1_c1';label_D1_c2'];
D_set{1,1}=D1;
D_set{2,1}=label_D1;
for i=1:70
    D2_c1(i,:)= class1_dm(rv1_test(i+70),:);
    label_D2_c1(i)= 1;
end
for i=1:30
    D2_c2(i,:)= class2_dm(rv2_test(i+30),:);
    label_D2_c2(i)= 2;
end
D2=[D2_c1;D2_c2];
label_D2=[label_D2_c1';label_D2_c2'];
D_set{1,2}=D2;
D_set{2,2}=label_D2;
for i=1:70
    D3_c1(i,:)= class1_dm(rv1_test(i+140),:);
```

```
    label_D3_c1(i)= 1;
end
for i=1:30
    D3_c2(i,:)= class2_dm(rv2_test(i+60),:);
    label_D3_c2(i)= 2;
end
D3=[D3_c1;D3_c2];
label_D3=[label_D3_c1';label_D3_c2'];
D_set{1,3}=D3;
D_set{2,3}=label_D3;
for i=1:70
    D4_c1(i,:)= class1_dm(rv1_test(i+210),:);
    label_D4_c1(i)= 1;
end
for i=1:30
    D4_c2(i,:)= class2_dm(rv2_test(i+90),:);
    label_D4_c2(i)= 2;
end
D4=[D4_c1;D4_c2];
label_D4=[label_D4_c1';label_D4_c2'];
D_set{1,4}=D4;
D_set{2,4}=label_D4;

for i=1:70
    D5_c1(i,:)= class1_dm(rv1_test(i+280),:);
    label_D5_c1(i)= 1;
end
for i=1:30
    D5_c2(i,:)= class2_dm(rv2_test(i+120),:);
    label_D5_c2(i)= 2;
end
D5=[D5_c1;D5_c2];
label_D5=[label_D5_c1';label_D5_c2'];
D_set{1,5}=D5;
D_set{2,5}=label_D5;

for i=1:70
    D6_c1(i,:)= class1_dm(rv1_test(i+350),:);
    label_D6_c1(i)= 1;
end
for i=1:30
    D6_c2(i,:)= class2_dm(rv2_test(i+150),:);
    label_D6_c2(i)= 2;
end
D6=[D6_c1;D6_c2];
label_D6=[label_D6_c1';label_D6_c2'];
D_set{1,6}=D6;
D_set{2,6}=label_D6;
```

```
for i=1:70
    D7_c1(i,:)=class1_dm(rv1_test(i+420),:);
    label_D7_c1(i)= 1;
end
for i=1:30
    D7_c2(i,:)=class2_dm(rv2_test(i+180),:);
    label_D7_c2(i)= 2;
end
D7=[D7_c1;D7_c2];
label_D7=[label_D7_c1';label_D7_c2'];
D_set{1,7}=D7;
D_set{2,7}=label_D7;
for i=1:70
    D8_c1(i,:)=class1_dm(rv1_test(i+490),:);
    label_D8_c1(i)= 1;
end
for i=1:30
    D8_c2(i,:)=class2_dm(rv2_test(i+210),:);
    label_D8_c2(i)= 2;
end
D8=[D8_c1;D8_c2];
label_D8=[label_D8_c1';label_D8_c2'];
D_set{1,8}=D8;
D_set{2,8}=label_D8;
for i=1:70
    D9_c1(i,:)=class1_dm(rv1_test(i+560),:);
    label_D9_c1(i)= 1;
end
for i=1:30
    D9_c2(i,:)=class2_dm(rv2_test(i+240),:);
    label_D9_c2(i)= 2;
end
D9=[D9_c1;D9_c2];
label_D9=[label_D9_c1';label_D9_c2'];
D_set{1,9}=D9;
D_set{2,9}=label_D9;

for i=1:70
    D10_c1(i,:)=class1_dm(rv1_test(i+630),:);
    label_D10_c1(i)= 1;
end
for i=1:30
    D10_c2(i,:)=class2_dm(rv2_test(i+270),:);
    label_D10_c2(i)= 2;
end
D10=[D10_c1;D10_c2];
label_D10=[label_D10_c1';label_D10_c2'];
D_set{1,10}=D10;
D_set{2,10}=label_D10;
```

```
clear D1_c1 D1_c2 D2_c1 D2_c2 D3_c1 D3_c2 D4_c1 D4_c2 D5_c1 D5_c2 D6_c1 D6_c2 D7_c1 D7_c2 D8_c1 D8_c2 D9_c1 D9_c2 D10_c1 D10_c2
clear label_D1_c1  label_D1_c2 label_D2_c1  label_D2_c2 label_D3_c1  label_D3_c2 label_D4_c1  label_D4_c2 label_D5_c1  label_D5_c2 label_D6_c1  label_D6_c2 label_D7_c1  label_D7_c2 label_D8_c1  label_D8_c2 label_D9_c1  label_D9_c2 label_D10_c1  label_D10_c2
total_classfication_rate=0;
for i=1:10
model=fitcdiscr([D_set{1,(mod(i,10)+1)};D_set{1,(mod(i+1,10)+1)};D_set{1,(mod(i+2,10)+1)};D_set{1,(mod(i+3,10)+1)};D_set{1,(mod(i+4,10)+1)};D_set{1,(mod(i+5,10)+1)};D_set{1,(mod(i+6,10)+1)};D_set{1,(mod(i+7,10)+1)};D_set{1,(mod(i+8,10)+1)}],    [D_set{2,(mod(i,10)+1)};D_set{2,(mod(i+1,10)+1)};D_set{2,(mod(i+2,10)+1)};D_set{2,(mod(i+3,10)+1)};D_set{2,(mod(i+4,10)+1)};D_set{2,(mod(i+5,10)+1)};D_set{2,(mod(i+6,10)+1)};D_set{2,(mod(i+7,10)+1)};D_set{2,(mod(i+8,10)+1)}],'DiscrimType','pseudoQuadratic');
t = predict(model, D_set{1,(mod(i+9,10)+1)});
m=mean(D_set{2,(mod(i+9,10)+1)}==t);
total_classfication_rate=total_classfication_rate+m;
end
crs_vrid_perfm=total_classfication_rate/10;
crs_vrid_perfm1=crs_vrid_perfm+crs_vrid_perfm1;

end
fprintf('Cross-Validation of 10-folds,20 times of Performace Estimation for QDA without normalization: %5.3f\n',crs_vrid_perfm1/20);
```