‘classifier\_Bayes.m’

clear;clc;close all;

%% import data

[features,labels]=preprocessing2();

costM=[0,1;3,0];

%% classification

disp('Naive-Bayes-Classification')

foldnum=5;

% set aside testdata(20%)

[dfeatures,dlabels]=Partition(features,labels,foldnum);

tcaccuracy=zeros(5,1);tcf\_measure=zeros(5,2);

for tindex=1:foldnum

    [traindata,trainlabels,testdata,testlabels]=redistribution(dfeatures,dlabels,tindex,0);

    % times for cross-validation

    ctimes=20;

    accuracymat=zeros(size(features,2),ctimes);

    for T=1:ctimes

        [tdfeatures,tdlabels]=Partition(traindata,trainlabels,foldnum);

        for fnum=1:size(features,2)

            % reinitial

            caccuracy=zeros(5,1);cf\_measure=zeros(5,2);

            for i=1:foldnum

                [trfeatures,trlabels,vfeatures,vlabels]=redistribution(tdfeatures,tdlabels,i,0);

                %% Normalization

                fmean=mean(trfeatures);

                fstd=std(trfeatures);

                trfeatures=normalize(trfeatures,fmean,fstd);

                vfeatures=normalize(vfeatures,fmean,fstd);

                %% PCA

                [coeff, score, latent]=pca(trfeatures);

                trfeatures=score(:,1:fnum);

                project\_v=vfeatures\*coeff;

                vfeatures=project\_v(:,1:fnum);

                %% classifier

                model=fitcnb(trfeatures,trlabels,'Cost',costM);

                pred\_labels=predict(model,vfeatures);

                %% evaluation

                [accuracy,F\_measure]=evaluation(pred\_labels,vlabels);

                caccuracy(i)=accuracy;

                cf\_measure(i,:)=F\_measure';

            end

            accuracymat(fnum,T)=mean(caccuracy);

        end

    end

    averageprecision=mean(accuracymat,2);

    [~,maxindex]=max(averageprecision);

    %% Normalization

     fmean=mean(traindata);

     fstd=std(traindata);

     traindata=normalize(traindata,fmean,fstd);

     testdata=normalize(testdata,fmean,fstd);

    %% PCA

     [coeff, score, latent]=pca(traindata);

     traindata=score(:,1:maxindex);

     project\_v=testdata\*coeff;

     testdata=project\_v(:,1:maxindex);

    %% classifier

     model=fitcnb(traindata,trainlabels,'Cost',costM);

     pred\_labels=predict(model,testdata);

     %% evaluation

     [accuracy,F\_measure]=evaluation(pred\_labels,testlabels);

     tcaccuracy(tindex)=accuracy;

     tcf\_measure(tindex,:)=F\_measure';

end

mf1scores=mean(tcf\_measure);

fprintf('Accuracy:%.2f%%  F\_meansure(Good;Bad):%.2f;%.2f \n',mean(tcaccuracy)\*100,mf1scores(1),mf1scores(2));

‘classifier\_KNN’

clear;clc;close all;

%% import data

[features,labels]=preprocessing2();

costM=[0,1;2,0];

%% classification

disp('K-Nerest-Neighborhood')

foldnum=5;

% set aside testdata(20%)

[dfeatures,dlabels]=Partition(features,labels,foldnum);

tcaccuracy=zeros(5,1);tcf\_measure=zeros(5,2);

for tindex=1:foldnum

    [traindata,trainlabels,testdata,testlabels]=redistribution(dfeatures,dlabels,tindex,0);

    ctimes=20;

    accuracymat=zeros(101,size(features,2),ctimes);

    for T=1:ctimes

        [tdfeatures,tdlabels]=Partition(traindata,trainlabels,foldnum);

        for knum=3:2:101

            for fnum=1:size(features,2)

                % reinitial

                caccuracy=zeros(5,1);cf\_measure=zeros(5,2);

                for i=1:foldnum

                    [trfeatures,trlabels,vfeatures,vlabels]=redistribution(tdfeatures,tdlabels,i,0);

                    %% Normalization

                    fmean=mean(trfeatures);

                    fstd=std(trfeatures);

                    trfeatures=normalize(trfeatures,fmean,fstd);

                    vfeatures=normalize(vfeatures,fmean,fstd);

                    %% PCA

                    [coeff, score, latent]=pca(trfeatures);

                    trfeatures=score(:,1:fnum);

                    project\_v=vfeatures\*coeff;

                    vfeatures=project\_v(:,1:fnum);

                    %% classifier

                    model=fitcknn(trfeatures,trlabels,'NumNeighbors',knum,'Cost',costM);

                    pred\_labels=predict(model,vfeatures);

                    %% evaluation

                    [accuracy,F\_measure]=evaluation(pred\_labels,vlabels);

                    caccuracy(i)=accuracy;

                    cf\_measure(i,:)=F\_measure';

                end

            accuracymat(knum,fnum,T)=mean(caccuracy);

            end

        end

    end

    averageprecision=accuracymat(:,:,1);

    for n=2:ctimes

        averageprecision=averageprecision+accuracymat(:,:,n);

    end

    averageprecision=averageprecision/ctimes;

    meanmax=max(averageprecision);

    meanmax=max(meanmax);

    [kmax,fmax]=find(averageprecision==meanmax);

    %% Normalization

     fmean=mean(traindata);

     fstd=std(traindata);

     traindata=normalize(traindata,fmean,fstd);

     testdata=normalize(testdata,fmean,fstd);

    %% PCA

     [coeff, score, latent]=pca(traindata);

     traindata=score(:,1:fmax(1));

     project\_v=testdata\*coeff;

     testdata=project\_v(:,1:fmax(1));

    %% classifier

     model=fitcknn(traindata,trainlabels,'NumNeighbors',kmax(1),'Cost',costM);

     pred\_labels=predict(model,testdata);

     %% evaluation

     [accuracy,F\_measure]=evaluation(pred\_labels,testlabels);

     tcaccuracy(tindex)=accuracy;

     tcf\_measure(tindex,:)=F\_measure';

end

mf1scores=mean(tcf\_measure);

fprintf('Accuracy:%.2f%%  F\_meansure(Good;Bad):%.2f;%.2f \n',mean(tcaccuracy)\*100,mf1scores(1),mf1scores(2));

‘classifier\_MAQ.m’

clear;clc;close all;

%% import data

[features,labels]=preprocessing();

costM=[0,1;1,0];

%% classification

disp('Discriminant analysis(Quadratic)')

foldnum=5;

% set aside testdata(20%)

[dfeatures,dlabels]=Partition(features,labels,foldnum);

tcaccuracy=zeros(5,1);tcf\_measure=zeros(5,2);

for tindex=1:foldnum

    [traindata,trainlabels,testdata,testlabels]=redistribution(dfeatures,dlabels,tindex,0);

    % times for cross-validation

    ctimes=20;

    accuracymat=zeros(size(features,2),ctimes);

    for T=1:ctimes

        [tdfeatures,tdlabels]=Partition(traindata,trainlabels,foldnum);

        for fnum=1:size(features,2)

            % reinitial

            caccuracy=zeros(5,1);cf\_measure=zeros(5,2);

            for i=1:foldnum

                [trfeatures,trlabels,vfeatures,vlabels]=redistribution(tdfeatures,tdlabels,i,0);

                %% Normalization

                fmean=mean(trfeatures);

                fstd=std(trfeatures);

                trfeatures=normalize(trfeatures,fmean,fstd);

                vfeatures=normalize(vfeatures,fmean,fstd);

                %% PCA

                [coeff, score, latent]=pca(trfeatures);

                trfeatures=score(:,1:fnum);

                project\_v=vfeatures\*coeff;

                vfeatures=project\_v(:,1:fnum);

                %% classifier

                mdllinear=fitcdiscr(trfeatures,trlabels,'Cost',costM,'DiscrimType','pseudoquadratic');

                pred\_labels=predict(mdllinear,vfeatures);

                %% evaluation

                [accuracy,F\_measure]=evaluation(pred\_labels,vlabels);

                caccuracy(i)=accuracy;

                cf\_measure(i,:)=F\_measure';

            end

            accuracymat(fnum,T)=mean(caccuracy);

        end

    end

    averageprecision=mean(accuracymat,2);

    [~,maxindex]=max(averageprecision);

    %% Normalization

     fmean=mean(traindata);

     fstd=std(traindata);

     traindata=normalize(traindata,fmean,fstd);

     testdata=normalize(testdata,fmean,fstd);

    %% PCA

     [coeff, score, latent]=pca(traindata);

     traindata=score(:,1:maxindex);

     project\_v=testdata\*coeff;

     testdata=project\_v(:,1:maxindex);

    %% classifier

     mdllinear=fitcdiscr(traindata,trainlabels,'Cost',costM,'DiscrimType','pseudoquadratic');

     pred\_labels=predict(mdllinear,testdata);

     %% evaluation

     [accuracy,F\_measure]=evaluation(pred\_labels,testlabels);

     tcaccuracy(tindex)=accuracy;

     tcf\_measure(tindex,:)=F\_measure';

end

mf1scores=mean(tcf\_measure);

fprintf('Accuracy:%.2f%%  F\_meansure(Good;Bad):%.2f;%.2f \n',mean(tcaccuracy)\*100,mf1scores(1),mf1scores(2));

‘classifier\_MDC.m’

clear;clc;close all;

%% import data

[features,labels]=preprocessing2();

costM=[0,1;1,0];

%% classification

disp('Minimum-Distance-Classification')

foldnum=5;

% set aside testdata(20%)

[dfeatures,dlabels]=Partition(features,labels,foldnum);

tcaccuracy=zeros(5,1);tcf\_measure=zeros(5,2);

for tindex=1:foldnum

    [traindata,trainlabels,testdata,testlabels]=redistribution(dfeatures,dlabels,tindex,0);

    % times for cross-validation

    ctimes=20;

    accuracymat=zeros(size(features,2),ctimes);

    for T=1:ctimes

        [tdfeatures,tdlabels]=Partition(traindata,trainlabels,foldnum);

        for fnum=1:size(features,2)

            % reinitial

            caccuracy=zeros(5,1);cf\_measure=zeros(5,2);

            for i=1:foldnum

                [trfeatures,trlabels,vfeatures,vlabels]=redistribution(tdfeatures,tdlabels,i,0);

                %% Normalization

                fmean=mean(trfeatures);

                fstd=std(trfeatures);

                trfeatures=normalize(trfeatures,fmean,fstd);

                vfeatures=normalize(vfeatures,fmean,fstd);

                %% PCA

                [coeff, score, latent]=pca(trfeatures);

                trfeatures=score(:,1:fnum);

                project\_v=vfeatures\*coeff;

                vfeatures=project\_v(:,1:fnum);

                %% classifier

                centroids=zeros(2,fnum);

                for ci=1:2

                    centroids(ci,:)=mean(trfeatures(find(trlabels==ci),:));

                end

                distance=pdist2(vfeatures,centroids);

                [~, pred\_labels] = min(distance, [], 2);

                %% evaluation

                [accuracy,F\_measure]=evaluation(pred\_labels,vlabels);

                caccuracy(i)=accuracy;

                cf\_measure(i,:)=F\_measure';

            end

            accuracymat(fnum,T)=mean(caccuracy);

        end

    end

    averageprecision=mean(accuracymat,2);

    [~,maxindex]=max(averageprecision);

    %% Normalization

     fmean=mean(traindata);

     fstd=std(traindata);

     traindata=normalize(traindata,fmean,fstd);

     testdata=normalize(testdata,fmean,fstd);

    %% PCA

     [coeff, score, latent]=pca(traindata);

     traindata=score(:,1:maxindex);

     project\_v=testdata\*coeff;

     testdata=project\_v(:,1:maxindex);

    %% classifier

     centroids=zeros(2,maxindex);

     for ci=1:2

        centroids(ci,:)=mean(traindata(find(trainlabels==ci),:));

     end

     distance=pdist2(testdata,centroids);

     [~, pred\_labels] = min(distance, [], 2);

     %% evaluation

     [accuracy,F\_measure]=evaluation(pred\_labels,testlabels);

     tcaccuracy(tindex)=accuracy;

     tcf\_measure(tindex,:)=F\_measure';

end

mf1scores=mean(tcf\_measure);

fprintf('Accuracy:%.2f%%  F\_meansure(Good;Bad):%.2f;%.2f \n',mean(tcaccuracy)\*100,mf1scores(1),mf1scores(2));

‘classifier\_MPL.m’

clear;clc;close all;

%% import data

[features,labels]=preprocessing2();

costM=[0,1;1,0];

%% classification

disp('Multiple Layers Neural Network')

% define architecture

NodeNum1=128;% 1st layer

NodeNum2=128;% 2nd layer

NodeNum3=128;% 3rd layer

NodeNum4=128;% 4th layer

NodeNum5=64;% 5th layer

TypeNum=2; % number of class

TF1='tansig';%'tansig';  %tansig(n) = 2/(1+exp(-2\*n))-1

TF2='purelin';

TF3='tansig';

TF4='purelin';

TF5='logsig';

OUT='tansig';

net=newff(minmax(sn),[NodeNum1,NodeNum2,NodeNum3,NodeNum4,NodeNum5,TypeNum],{TF1 TF2 TF3,TF4,TF5,OUT});%,'trainscg','learngdm'

net.trainParam.show=100;

net.trainParam.epochs=100000;

net.trainParam.goal=3e-5;

net.trainParam.lr=0.1;

foldnum=5;

% set aside testdata(20%)

[dfeatures,dlabels]=Partition(features,labels,foldnum);

tcaccuracy=zeros(5,1);tcf\_measure=zeros(5,2);

for tindex=1:foldnum

    [traindata,trainlabels,testdata,testlabels]=redistribution(dfeatures,dlabels,tindex,0);

    % times for cross-validation

    ctimes=20;

    accuracymat=zeros(size(features,2),ctimes);

    for T=1:ctimes

        [tdfeatures,tdlabels]=Partition(traindata,trainlabels,foldnum);

        for fnum=1:size(features,2)

            % reinitial

            caccuracy=zeros(5,1);cf\_measure=zeros(5,2);

            for i=1:foldnum

                [trfeatures,trlabels,vfeatures,vlabels]=redistribution(tdfeatures,tdlabels,i,0);

                %% Normalization

                fmean=mean(trfeatures);

                fstd=std(trfeatures);

                trfeatures=normalize(trfeatures,fmean,fstd);

                vfeatures=normalize(vfeatures,fmean,fstd);

                %% PCA

                [coeff, score, latent]=pca(trfeatures);

                trfeatures=score(:,1:fnum);

                project\_v=vfeatures\*coeff;

                vfeatures=project\_v(:,1:fnum);

                %% classifier

                targets=zeros(size(trlabels,1),2);

                for ti=1:size(trlabels,1)

                    if trlabels(ti)==1

                        targets(ti,:)=[1,0];

                    end

                    if trlabels(ti)==2

                        targets(ti,:)=[0,1];

                    end

                end

                [sn,mins,maxs,tn,mint,maxt]=premnmx(trfeatures',targets');%pn = 2\*(p-minp)/(maxp-minp) - 1

                net=train(net,sn,tn,'useGPU','yes');

                % test

                s2n=tramnmx(vfeatures',mins,maxs);

                an=sim(net,s2n);

                testout=postmnmx(an,mint,maxt);

                [~, pred\_labels] = max(testout', [], 2);

                %% evaluation

                [accuracy,F\_measure]=evaluation(pred\_labels,vlabels);

                caccuracy(i)=accuracy;

                cf\_measure(i,:)=F\_measure';

            end

            accuracymat(fnum,T)=mean(caccuracy);

        end

    end

    averageprecision=mean(accuracymat,2);

    [~,maxindex]=max(averageprecision);

    %% Normalization

     fmean=mean(traindata);

     fstd=std(traindata);

     traindata=normalize(traindata,fmean,fstd);

     testdata=normalize(testdata,fmean,fstd);

    %% PCA

     [coeff, score, latent]=pca(traindata);

     traindata=score(:,1:maxindex);

     project\_v=testdata\*coeff;

     testdata=project\_v(:,1:maxindex);

    %% classifier

     targets=zeros(size(trainlabels,1),2);

     for ti=1:size(trainlabels,1)

         if trainlabels(ti)==1

             targets(ti,:)=[1,0];

         end

         if trainlabels(ti)==2

             targets(ti,:)=[0,1];

         end

     end

     [sn,mins,maxs,tn,mint,maxt]=premnmx(traindata',targets');%pn = 2\*(p-minp)/(maxp-minp) - 1

     net=train(net,sn,tn,'useGPU','yes');

     % test

     s2n=tramnmx(testdata',mins,maxs);

     an=sim(net,s2n);

     testout=postmnmx(an,mint,maxt);

     [~, pred\_labels] = max(testout', [], 2);

     %% evaluation

     [accuracy,F\_measure]=evaluation(pred\_labels,testlabels);

     tcaccuracy(tindex)=accuracy;

     tcf\_measure(tindex,:)=F\_measure';

end

mf1scores=mean(tcf\_measure);

fprintf('Accuracy:%.2f%%  F\_meansure(Good;Bad):%.2f;%.2f \n',mean(tcaccuracy)\*100,mf1scores(1),mf1scores(2));

‘classifier\_SVM.m’

clear;clc;close all;

%% import data

[features,labels]=preprocessing2();

costM=[0,1;1,0];

%% classification

disp('Support-Vector-Machine')

g=logspace(-3,3,50);c=g;

foldnum=5;

% set aside testdata(20%)

[dfeatures,dlabels]=Partition(features,labels,foldnum);

tcaccuracy=zeros(5,1);tcf\_measure=zeros(5,2);

for tindex=1:foldnum

    [traindata,trainlabels,testdata,testlabels]=redistribution(dfeatures,dlabels,tindex,0);

    ctimes=5;

    accuracymat=zeros(50,50,ctimes);

    for T=1:ctimes

        [tdfeatures,tdlabels]=Partition(traindata,trainlabels,foldnum);

        for cnum=1:50

            for gnum=1:50

                % reinitial

                caccuracy=zeros(5,1);cf\_measure=zeros(5,2);

                for i=1:foldnum

                    [trfeatures,trlabels,vfeatures,vlabels]=redistribution(tdfeatures,tdlabels,i,0);

                    %% Normalization

                    fmean=mean(trfeatures);

                    fstd=std(trfeatures);

                    trfeatures=normalize(trfeatures,fmean,fstd);

                    vfeatures=normalize(vfeatures,fmean,fstd);

%                     %% PCA

%                     [coeff, score, latent]=pca(trfeatures);

%                     trfeatures=score(:,1:fnum);

%                     project\_v=vfeatures\*coeff;

%                     vfeatures=project\_v(:,1:fnum);

                    %% classifier

                    options=sprintf('%s %d %s %d %s','-t 2 -c',c(cnum),'-g',g(gnum),'-d 3 -q');

                    model=svmtrain(trlabels,trfeatures,options);

                    %validation result

                    [pred\_labels, vaccuracy, vdecision\_values]=svmpredict(vlabels, vfeatures, model, '-q');

                    %% evaluation

                    [accuracy,F\_measure]=evaluation(pred\_labels,vlabels);

                    caccuracy(i)=accuracy;

                    cf\_measure(i,:)=F\_measure';

                end

            accuracymat(cnum,gnum,T)=mean(caccuracy);

            end

        end

    end

    averageprecision=accuracymat(:,:,1);

    for n=2:ctimes

        averageprecision=averageprecision+accuracymat(:,:,n);

    end

    averageprecision=averageprecision/ctimes;

    surf(c,g,averageprecision); title('ACC');

    set(gca,'Xscale','log','Yscale','log');

    meanmax=max(averageprecision);

    meanmax=max(meanmax);

    [cmax,gmax]=find(averageprecision==meanmax);

    %% Normalization

     fmean=mean(traindata);

     fstd=std(traindata);

     traindata=normalize(traindata,fmean,fstd);

     testdata=normalize(testdata,fmean,fstd);

%     %% PCA

%      [coeff, score, latent]=pca(traindata);

%      traindata=score(:,1:fmax(1));

%      project\_v=testdata\*coeff;

%      testdata=project\_v(:,1:fmax(1));

     %% classifier

     options=sprintf('%s %d %s %d %s','-t 2 -c',c(cmax),'-g',g(gmax),'-d 3 -q');

     model=svmtrain(trainlabels,traindata,options);

     %validation result

     [pred\_labels, vaccuracy, vdecision\_values]=svmpredict(testlabels, testdata, model, '-q');

     %% evaluation

     [accuracy,F\_measure]=evaluation(pred\_labels,testlabels);

     tcaccuracy(tindex)=accuracy;

     tcf\_measure(tindex,:)=F\_measure';

end

mf1scores=mean(tcf\_measure);

fprintf('Accuracy:%.2f%%  F\_meansure(Good;Bad):%.2f;%.2f \n',mean(tcaccuracy)\*100,mf1scores(1),mf1scores(2));

‘evaluation.m’

function [accuracy,F\_measure]=evaluation(pred\_labels,vlabels)

accuracy=size(find(pred\_labels==vlabels),1)/(size(vlabels,1));

C=confusionmat(vlabels,pred\_labels);

precision=diag(C)./sum(C,2);

recall=diag(C)./sum(C,1)';

F\_measure=2\*(precision.\*recall)./(precision+recall);

end

‘normalize.m’

function [ feature ] = normalize(infeature,fmean,fstd)

%normalize

    for n=1:size(infeature,2)

    feature(:,n)=infeature(:,n)-fmean(n);

    if fstd~=0

    feature(:,n)=feature(:,n)/fstd(n);

    end

    end

end

‘observefeaturesm.m’

clear;clc;close all;

[features,labels]=preprocessing2;

x=features(:,9)';y=labels';

content=unique(x);

y1=zeros(1,size(content,2));

y2=y1;y=y1;

for i=1:size(content,2)

    tmp=labels(find(x==content(i)));

    y1(i)=size(find(tmp==1),1);

    y2(i)=size(find(tmp==2),1);

    y(i)=size(tmp==2,1);

end

% hold on

% stem(y1/700,'go');

% stem(-y2/300,'rx');

% hold off

y/10

‘importdata.m’

%% Import data from text file.

% Script for importing data from the following text file:

%

%    /Users/youqilin/Documents/USC/2rd Semester/EE559/Homework/Project/Proj\_dataset\_1.csv

%

% To extend the code to different selected data or a different text file,

% generate a function instead of a script.

% Auto-generated by MATLAB on 2017/04/24 18:28:48

%% Initialize variables.

filename = 'Proj\_dataset\_1.csv';

delimiter = ',';

startRow = 2;

%% Read columns of data as text:

% For more information, see the TEXTSCAN documentation.

formatSpec = '%\*s%s%s%s%s%s%s%s%s%s%s%[^\n\r]';

%% Open the text file.

fileID = fopen(filename,'r');

%% Read columns of data according to the format.

% This call is based on the structure of the file used to generate this

% code. If an error occurs for a different file, try regenerating the code

% from the Import Tool.

dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, 'HeaderLines' ,startRow-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');

%% Close the text file.

fclose(fileID);

%% Convert the contents of columns containing numeric text to numbers.

% Replace non-numeric text with NaN.

raw = repmat({''},length(dataArray{1}),length(dataArray)-1);

for col=1:length(dataArray)-1

    raw(1:length(dataArray{col}),col) = dataArray{col};

end

numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,3,7,8,10]

    % Converts text in the input cell array to numbers. Replaced non-numeric

    % text with NaN.

    rawData = dataArray{col};

    for row=1:size(rawData, 1);

        % Create a regular expression to detect and remove non-numeric prefixes and

        % suffixes.

        regexstr = '(?<prefix>.\*?)(?<numbers>([-]\*(\d+[\,]\*)+[\.]{0,1}\d\*[eEdD]{0,1}[-+]\*\d\*[i]{0,1})|([-]\*(\d+[\,]\*)\*[\.]{1,1}\d+[eEdD]{0,1}[-+]\*\d\*[i]{0,1}))(?<suffix>.\*)';

        try

            result = regexp(rawData{row}, regexstr, 'names');

            numbers = result.numbers;

            % Detected commas in non-thousand locations.

            invalidThousandsSeparator = false;

            if any(numbers==',');

                thousandsRegExp = '^\d+?(\,\d{3})\*\.{0,1}\d\*$';

                if isempty(regexp(numbers, thousandsRegExp, 'once'));

                    numbers = NaN;

                    invalidThousandsSeparator = true;

                end

            end

            % Convert numeric text to numbers.

            if ~invalidThousandsSeparator;

                numbers = textscan(strrep(numbers, ',', ''), '%f');

                numericData(row, col) = numbers{1};

                raw{row, col} = numbers{1};

            end

        catch me

        end

    end

end

%% Split data into numeric and cell columns.

rawNumericColumns = raw(:, [1,3,7,8,10]);

rawCellColumns = raw(:, [2,4,5,6,9]);

%% Allocate imported array to column variable names

Age = cell2mat(rawNumericColumns(:, 1));

Sex = rawCellColumns(:, 1);

Job = cell2mat(rawNumericColumns(:, 2));

Housing = rawCellColumns(:, 2);

Savingaccounts = rawCellColumns(:, 3);

Checkingaccount = rawCellColumns(:, 4);

Creditamount = cell2mat(rawNumericColumns(:, 3));

Duration = cell2mat(rawNumericColumns(:, 4));

Purpose = rawCellColumns(:, 5);

Class = cell2mat(rawNumericColumns(:, 5));

%% Clear temporary variables

clearvars filename delimiter startRow formatSpec fileID dataArray ans raw col numericData rawData row regexstr result numbers invalidThousandsSeparator thousandsRegExp me rawNumericColumns rawCellColumns;

‘importdata2.m’

%% Import data from text file.

% Script for importing data from the following text file:

%

%    /Users/youqilin/Documents/USC/2rd Semester/EE559/Homework/Project/Proj\_dataset\_2.csv

%

% To extend the code to different selected data or a different text file,

% generate a function instead of a script.

% Auto-generated by MATLAB on 2017/04/26 22:30:03

%% Initialize variables.

filename = 'Proj\_dataset\_2.csv';

delimiter = ',';

%% Read columns of data as text:

% For more information, see the TEXTSCAN documentation.

formatSpec = '%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%[^\n\r]';

%% Open the text file.

fileID = fopen(filename,'r');

%% Read columns of data according to the format.

% This call is based on the structure of the file used to generate this

% code. If an error occurs for a different file, try regenerating the code

% from the Import Tool.

dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,  'ReturnOnError', false);

%% Close the text file.

fclose(fileID);

%% Convert the contents of columns containing numeric text to numbers.

% Replace non-numeric text with NaN.

raw = repmat({''},length(dataArray{1}),length(dataArray)-1);

for col=1:length(dataArray)-1

    raw(1:length(dataArray{col}),col) = dataArray{col};

end

numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]

    % Converts text in the input cell array to numbers. Replaced non-numeric

    % text with NaN.

    rawData = dataArray{col};

    for row=1:size(rawData, 1);

        % Create a regular expression to detect and remove non-numeric prefixes and

        % suffixes.

        regexstr = '(?<prefix>.\*?)(?<numbers>([-]\*(\d+[\,]\*)+[\.]{0,1}\d\*[eEdD]{0,1}[-+]\*\d\*[i]{0,1})|([-]\*(\d+[\,]\*)\*[\.]{1,1}\d+[eEdD]{0,1}[-+]\*\d\*[i]{0,1}))(?<suffix>.\*)';

        try

            result = regexp(rawData{row}, regexstr, 'names');

            numbers = result.numbers;

            % Detected commas in non-thousand locations.

            invalidThousandsSeparator = false;

            if any(numbers==',');

                thousandsRegExp = '^\d+?(\,\d{3})\*\.{0,1}\d\*$';

                if isempty(regexp(numbers, thousandsRegExp, 'once'));

                    numbers = NaN;

                    invalidThousandsSeparator = true;

                end

            end

            % Convert numeric text to numbers.

            if ~invalidThousandsSeparator;

                numbers = textscan(strrep(numbers, ',', ''), '%f');

                numericData(row, col) = numbers{1};

                raw{row, col} = numbers{1};

            end

        catch me

        end

    end

end

%% Allocate imported array to column variable names

A1 = cell2mat(raw(:, 1));

A2 = cell2mat(raw(:, 2));

A3 = cell2mat(raw(:, 3));

A4 = cell2mat(raw(:, 4));

A5 = cell2mat(raw(:, 5));

A6 = cell2mat(raw(:, 6));

A7 = cell2mat(raw(:, 7));

A8 = cell2mat(raw(:, 8));

A9 = cell2mat(raw(:, 9));

A10 = cell2mat(raw(:, 10));

A11 = cell2mat(raw(:, 11));

A12 = cell2mat(raw(:, 12));

A13 = cell2mat(raw(:, 13));

A14 = cell2mat(raw(:, 14));

A15 = cell2mat(raw(:, 15));

A16 = cell2mat(raw(:, 16));

A17 = cell2mat(raw(:, 17));

A18 = cell2mat(raw(:, 18));

A19 = cell2mat(raw(:, 19));

A20 = cell2mat(raw(:, 20));

Class = cell2mat(raw(:, 21));

%% Clear temporary variables

clearvars filename delimiter formatSpec fileID dataArray ans raw col numericData rawData row regexstr result numbers invalidThousandsSeparator thousandsRegExp me;

‘Partition.m’

% Qilin You 2017/03/11

% example:[dfeature,dlabel]=Partition(feature,label,foldnum)

function [dfeature,dlabel]=Partition(feature,label,foldnum)

numclass=size(unique(label),1);

dfeature=zeros(1,size(feature,2),foldnum);dlabel=zeros(1,1,foldnum);

for i=1:numclass

    subdata=feature((find(label==i)),:);sublabel=label((find(label==i)),:);

    bsize=size(subdata,1);rbsize=bsize;

    for m=1:foldnum

    %divide into bins

    for numpick=1:bsize/foldnum

    randpick=unidrnd(rbsize); %generate random index

    for col=1:size(subdata,2)

    dfeature(size(find(dlabel(:,1,m)~=0),1)+1,col,m)=subdata(randpick,col);%store feature

    end

    dlabel(size(find(dlabel(:,1,m)~=0),1)+1,1,m)=sublabel(randpick,1);%store label

    subdata(randpick,:)=[]; sublabel(randpick,:)=[]; %remove picked data

    rbsize=rbsize-1; %record the rest data

    end

    end

    %divide the rest data

    for m=1:mod(bsize,foldnum)

    randpick=unidrnd(rbsize); %generate random index

    for col=1:size(subdata,2)

    dfeature(size(find(dlabel(:,1,m)~=0),1)+1,col,m)=subdata(randpick,col);%store feature

    end

    dlabel(size(find(dlabel(:,1,m)~=0),1)+1,1,m)=sublabel(randpick,1);%store label

    subdata(randpick,:)=[]; sublabel(randpick,:)=[]; %remove picked data

    rbsize=rbsize-1;

    end

end

end

‘preprocessing.m’

function [features,labels]= preprocessing()

% Author: Qilin You

%Usage: [features,labels]= preprocessing()

    importdata;

    %% labels

    labels=Class;

    %% features

    features=zeros(1000,9);

    % Age

    features(:,1)=Age;

    % Sex

    for i=1:size(Sex,1)

        if  isequal(Sex{i},'female')

            features(i,2)=0;

        end

        if  isequal(Sex{i},'male')

            features(i,2)=1;

        end

    end

    % Job

    features(:,3)=Job;

    % Housing

    for i=1:size(Housing,1)

        if  isequal(Housing{i},'free')

            features(i,4)=0;

        end

        if  isequal(Housing{i},'rent')

            features(i,4)=1;

        end

        if  isequal(Housing{i},'own')

            features(i,4)=2;

        end

    end

    % Savingaccounts

    for i=1:size(Savingaccounts,1)

        if  isequal(Savingaccounts{i},'NA')

            features(i,5)=-1;

        end

        if  isequal(Savingaccounts{i},'little')

            features(i,5)=0;

        end

        if  isequal(Savingaccounts{i},'moderate')

            features(i,5)=1;

        end

        if  isequal(Savingaccounts{i},'rich')

            features(i,5)=2;

        end

        if  isequal(Savingaccounts{i},'quite rich')

            features(i,5)=3;

        end

    end

    % Checkingaccount

    for i=1:size(Checkingaccount,1)

        if  isequal(Checkingaccount{i},'NA')

            features(i,6)=-1;

        end

        if  isequal(Checkingaccount{i},'little')

            features(i,6)=0;

        end

        if  isequal(Checkingaccount{i},'moderate')

            features(i,6)=1;

        end

        if  isequal(Checkingaccount{i},'rich')

            features(i,6)=2;

        end

    end

    % Creditamount

    features(:,7)=Creditamount;

    % Duration

    features(:,8)=Duration;

    % Purpose

    for i=1:size(Purpose,1)

        if  isequal(Purpose{i},'radio/TV')

            features(i,9)=0;

        end

        if  isequal(Purpose{i},'repairs')

            features(i,9)=1;

        end

        if  isequal(Purpose{i},'domestic appliances')

            features(i,9)=2;

        end

        if  isequal(Purpose{i},'furniture/equipment')

            features(i,9)=3;

        end

        if  isequal(Purpose{i},'vacation/others')

            features(i,9)=4;

        end

        if  isequal(Purpose{i},'education')

            features(i,9)=5;

        end

        if  isequal(Purpose{i},'business')

            features(i,9)=6;

        end

        if  isequal(Purpose{i},'car')

            features(i,9)=7;

        end

    end

%% Handle missing data

    for col=1:size(features,2)

        temp=features(:,col);

        temp(find(temp==-1))=[];

        sdata=mean(temp);

        for row=1:size(features,1)

            if features(row,col)==-1

                features(row,col)=sdata;

            end

        end

    end

%% drop out invalid data

    fstd=std(features);

    features(:,find(fstd==0))=[];

end

‘preprocessing\_expanded.m’

function [features,labels]= preprocessing()

% Author: Qilin You

%Usage: [features,labels]= preprocessing()

    importdata;

    %% labels

    labels=Class;

    %% features

    % Age

    features=Age;

    % Sex

    subf=zeros(0,0);

    for i=1:size(Sex,1)

        if  isequal(Sex{i},'female')

           subf(i,:)=0;

        end

        if  isequal(Sex{i},'male')

           subf(i,:)=1;

        end

    end

    features=[features,subf];

    % Job

    features=[features,Job];

    % Housing

    subf=zeros(0,0);

    for i=1:size(Housing,1)

        if  isequal(Housing{i},'free')

            subf(i,:)=[1,0,0];

        end

        if  isequal(Housing{i},'rent')

            subf(i,:)=[0,1,0];

        end

        if  isequal(Housing{i},'own')

            subf(i,:)=[0,0,1];

        end

    end

    features=[features,subf];

    % Savingaccounts

    subf=zeros(0,0);

    for i=1:size(Savingaccounts,1)

        if  isequal(Savingaccounts{i},'NA')

            subf(i,:)=-1;

        end

        if  isequal(Savingaccounts{i},'little')

            subf(i,:)=0;

        end

        if  isequal(Savingaccounts{i},'moderate')

            subf(i,:)=1;

        end

        if  isequal(Savingaccounts{i},'rich')

            subf(i,:)=2;

        end

        if  isequal(Savingaccounts{i},'quite rich')

            subf(i,:)=3;

        end

    end

    features=[features,subf];

    % Checkingaccount

    subf=zeros(0,0);

    for i=1:size(Checkingaccount,1)

        if  isequal(Checkingaccount{i},'NA')

            subf(i,:)=-1;

        end

        if  isequal(Checkingaccount{i},'little')

            subf(i,:)=0;

        end

        if  isequal(Checkingaccount{i},'moderate')

            subf(i,:)=1;

        end

        if  isequal(Checkingaccount{i},'rich')

            subf(i,:)=2;

        end

        if  isequal(Checkingaccount{i},'quite rich')

            subf(i,:)=3;

        end

    end

    features=[features,subf];

    % Creditamount

    features=[features,Creditamount];

    % Duration

    features=[features,Duration];

    % Purpose

    subf=zeros(0,0);

    for i=1:size(Purpose,1)

        if  isequal(Purpose{i},'business')

            subf(i,:)=[1,0,0,0,0,0,0,0];

        end

        if  isequal(Purpose{i},'car')

            subf(i,:)=[0,1,0,0,0,0,0,0];

        end

        if  isequal(Purpose{i},'domestic appliances')

            subf(i,:)=[0,0,1,0,0,0,0,0];

        end

        if  isequal(Purpose{i},'education')

            subf(i,:)=[0,0,0,1,0,0,0,0];

        end

        if  isequal(Purpose{i},'furniture/equipment')

            subf(i,:)=[0,0,0,0,1,0,0,0];

        end

        if  isequal(Purpose{i},'radio/TV')

            subf(i,:)=[0,0,0,0,0,1,0,0];

        end

        if  isequal(Purpose{i},'repairs')

            subf(i,:)=[0,0,0,0,0,0,1,0];

        end

        if  isequal(Purpose{i},'vacation/others')

            subf(i,:)=[0,0,0,0,0,0,0,1];

        end

    end

    features=[features,subf];

%% Handle missing data

    for col=1:size(features,2)

        temp=features(:,col);

        temp(find(temp==-1))=[];

        sdata=mean(temp);

        for row=1:size(features,1)

            if features(row,col)==-1

                features(row,col)=sdata;

            end

        end

    end

%% drop out invalid data

    fstd=std(features);

    features(:,find(fstd==0))=[];

end

‘preprocessing2.m’

function [features,labels]= preprocessing2()

% Author: Qilin You

%Usage: [features,labels]= preprocessing()

    importdata2;

    %% labels

    labels=Class;

    %% features

    features=zeros(1000,20);

    % Status of existing checking account

    features(:,1)=mod(A1,10);

    % Duration in month

    features(:,2)=A2;

    % Credit history

    features(:,3)=mod(A3,10);

    % Purpose

    for index=1:size(A4,1)

        if A4(index)<100

            features(index,4)=mod(A4(index),10);

        else

            features(index,4)=10;

        end

    end

    % Credit amount

    features(:,5)=A5;

    % Savings account/bounds

    features(:,6)=mod(A6,10);

    % Present employment since

    features(:,7)=mod(A7,10);

    % Installment rate in percentage of disposable income

    features(:,8)=A8;

    % Personal status and sex

    features(:,9)=mod(A9,10);

    % Other debtors/guarantors

    features(:,10)=mod(A10,10);

    % present residence since

    features(:,11)=A11;

    % Property

    features(:,12)=mod(A12,10);

    % Age in years

    features(:,13)=A13;

    % Ohter installment plans

    features(:,14)=mod(A14,10);

    % Housing

    features(:,15)=mod(A15,10);

    % Number of existing credits at this bank

    features(:,16)=A16;

    % Job

    features(:,17)=mod(A17,10);

    % Number of people being liable to provide

    features(:,18)=A18;

    % Telephone

    features(:,19)=mod(A19,10);

    % foreign worker

    features(:,20)=mod(A20,10);

%% Handle missing data

%% drop out invalid data

    fstd=std(features);

    features(:,find(fstd==0))=[];

end

‘preprocessing2\_expanded.m’

function [features,labels]= preprocessing2\_expanded()

% Author: Qilin You

%Usage: [features,labels]= preprocessing()

    importdata2;

    %% labels

    labels=Class;

    %% features

    % Status of existing checking account

    features=mod(A1,10);

%     features(find(features(:,end)==4))=3;

    % Duration in month

    features=[features,A2];

    % Credit history

    features=[features,mod(A3,10)];

%     features(find(features(:,end)==1))=0;

%     features(find(features(:,end)==2))=1;

%     features(find(features(:,end)==3))=2;

%     features(find(features(:,end)==4))=2;

    % Purpose

    subf=zeros(0,0);

    for i=1:size(A4,1)

        if  A4(i)==40

            subf(i,:)=[1,0,0,0,0,0,0,0,0,0,0];

        end

        if  A4(i)==41

            subf(i,:)=[0,1,0,0,0,0,0,0,0,0,0];

        end

        if  A4(i)==42

            subf(i,:)=[0,0,1,0,0,0,0,0,0,0,0];

        end

        if  A4(i)==43

            subf(i,:)=[0,0,0,1,0,0,0,0,0,0,0];

        end

        if  A4(i)==44

            subf(i,:)=[0,0,0,0,1,0,0,0,0,0,0];

        end

        if  A4(i)==45

            subf(i,:)=[0,0,0,0,0,1,0,0,0,0,0];

        end

        if  A4(i)==46

            subf(i,:)=[0,0,0,0,0,0,1,0,0,0,0];

        end

        if  A4(i)==47

            subf(i,:)=[0,0,0,0,0,0,0,1,0,0,0];

        end

        if  A4(i)==48

            subf(i,:)=[0,0,0,0,0,0,0,0,1,0,0];

        end

        if  A4(i)==49

            subf(i,:)=[0,0,0,0,0,0,0,0,0,1,0];

        end

        if  A4(i)==410

            subf(i,:)=[0,0,0,0,0,0,0,0,0,0,1];

        end

    end

    features=[features,subf];

    % Credit amount

    features=[features,A5];

    % Savings account/bounds

    features=[features,mod(A6,10)];

%     features(find(features(:,end)==4))=3;

%     features(find(features(:,end)==5))=4;

    % Present employment since

    features=[features,mod(A7,10)];

%     features(find(features(:,end)==2))=1;

    % Installment rate in percentage of disposable income

    features=[features,A8];

    % Personal status and sex

    subf=zeros(0,0);

    for i=1:size(A9,1)

        if  A9(i)==91

            subf(i,:)=[1,0,0,0,0];

        end

        if  A9(i)==92

            subf(i,:)=[0,1,0,0,0];

        end

        if  A9(i)==93

            subf(i,:)=[0,0,1,0,0];

        end

        if  A9(i)==94

            subf(i,:)=[0,0,0,1,0];

        end

        if  A9(i)==95

            subf(i,:)=[0,0,0,0,1];

        end

    end

    features=[features,subf];

    % Other debtors/guarantors

    features=[features,mod(A10,10)];

%     features(find(features(:,end)==3))=2;

    % present residence since

    features=[features,A11];

    % Property

    features=[features,5-mod(A12,10)];

    % Age in years

    features=[features,A13];

    % Other installment plans

    subf=zeros(0,0);

    for i=1:size(A14,1)

        if  A14(i)==141

            subf(i,:)=[1,0,0];

        end

        if  A14(i)==142

            subf(i,:)=[0,1,0];

        end

        if  A14(i)==143

            subf(i,:)=[0,0,1];

        end

    end

    features=[features,subf];

    % Housing

    subf=zeros(0,0);

    for i=1:size(A15,1)

        if  A15(i)==151

            subf(i,:)=[1,0,0];

        end

        if  A15(i)==152

            subf(i,:)=[0,1,0];

        end

        if  A15(i)==153

            subf(i,:)=[0,0,1];

        end

    end

    features=[features,subf];

    % Number of existingcredits at this bank

    features=[features,A16];

%     features(find(features(:,end)==3))=2;

%     features(find(features(:,end)==4))=2;

    % Job

    features=[features,mod(A17,10)];

%     features(find(features(:,end)==2))=1;

    % Number of people being liable to provide

    features=[features,A18];

    % Telephone

    features=[features,mod(A19,10)];

    % foreign worker

    features=[features,mod(A20,10)];

%% Handle missing data

%% drop out invalid data

    fstd=std(features);

    features(:,find(fstd==0))=[];

end

‘redistribution.m’

function [bdata,blabels,sdata,slabels]=redistribution(ddata,dlabels,i,sflag)

% number of fold

foldnum=size(ddata,3);

% data

bdata=zeros(0,0);blabels=zeros(0,0);

sdata=zeros(0,0); slabels=zeros(0,0);

% small bin

sdata=ddata(find(dlabels(:,1,i)~=0),:,i);slabels=dlabels(find(dlabels(:,1,i)~=0),:,i);

% big bins

for j=1:foldnum

    if j~=i

    bdata=cat(1,bdata,ddata(find(dlabels(:,1,j)~=0),:,j)); blabels=cat(1,blabels,dlabels(find(dlabels(:,1,j)~=0),:,j));

    end

end

if sflag

    % shuffle data

    % small bin

    oldt=[sdata,slabels];

    newt=oldt(randperm(size(oldt,1)),:);

    slabels=newt(:,end);

    newt(:,end)=[];

    sdata=newt;

    % big bin

    oldt=[bdata,blabels];

    newt=oldt(randperm(size(oldt,1)),:);

    blabels=newt(:,end);

    newt(:,end)=[];

    bdata=newt;

end

end