Signals and Systems

CA3

Rouja Aghajani – 810101380

سوال 1:

```
clc;
clear;
chars = ['a':'z', ' ', '.', ',', '!', '"', ';'];

charCell = cell(2, length(chars));

for i = 1:length(chars)
    charCell{1, i} = chars(i);
end

for i = 1:32
    if i <= length(chars)
        charCell{2, i} = dec2bin(i-1, 5);
    else
        break;
    end
end
inputMessage = 'hi there;';
pic=imread('Amsterdam.jpg');
grayPic = rgb2gray(pic);

threshold = 70;
blockSize = [5,5];

coddedPic = coding(charCell, grayPic, inputMessage, blockSize, threshold);
subplot(1, 2, 1);
imshow(coddedPic);
title('Codded');

% Plot second image on the second subplot
subplot(1, 2, 2);
imshow(grayPic);
title('original')
decodedMessage = decoding(coddedPic, charCell, blockSize, threshold);
disp(decodedMessage);
```

در اینجا ابتدا یک مپ ست از حرفو و اعداد 5 بیتی متناظرشان می سازیم. حال یک تصویر را خوانده و آن را تبدیل به عکس سیاه و سفید میکنیم. یک threshold برای انتخاب بلاک مناسب جهت قراردادن پیام انتخاب میکنیم؛ هرچقدر threshold بالاتر باشد، اگر بلاکی یافت شود پیام قرارداده شده بیشتر غیر قابل تشخیص خواهد شد.

```
function codedImage = coding(dataset, grayImage, message, blockSize, blockVarianceThreshold)
    BinaryMessage = messageBinarizer(message, dataset);
    semicolonIndex = find([dataset{1,:}] == ';', 1);
    BinaryMessage = [BinaryMessage dataset{2, semicolonIndex}];

    codedImage = grayImage;
    [rows, cols] = size(grayImage);
    Counter = 1;
    for i = 1:blockSize(1):rows-4
        for j = 1:blockSize(2):cols-4

            chosenBlock = grayImage(i:i+4, j:j+4);
            if var(double(chosenBlock(:))) > blockVarianceThreshold
                for x = i:i+4
                    for y = j:j+4
                        if Counter > length(BinaryMessage), break; end
                        codedImage(x, y) = bitset(codedImage(x, y), 1, BinaryMessage(Counter) - '0');
                        Counter = Counter + 1;
                    end
                end
            end
        end
    end

    if Counter <= length(BinaryMessage)
        error('Message length is too much; can not be fitted into picture');  % Raise an error if not all bits were encoded
    end
end
```

ابتدا با استفاده از تابع باینری کننده پیام که عکس آن در ادامه آمده است، پیام ورودی را باینری میکنیم تا بتوانیم با تغییر lsb هر پیکسل آنرا جایگذاری کنیم. سپس مکان قرارگرفتن ؛ جهت یافتن انتهای پیام را شناسایی میکنیم. در نهایت از گوشه بلاک شروع کرده، میبینیم آیا بلاک انتخابی واریانس بالاتر از حد تعیین کرده یعنی همان threshold دارد یا خیر؛ اگر داشت، کدگذاری را آغاز میکنیم. در بلاک انتخابی میگردیم و بیت آخر هر پیکسل را با توجه به پیام تغییر میدهیم و بدین ترتیب تا جایی که به انتهای پیام یعنی همان ؛ نرسیدیم کد گذاری را ادامه میدهیم. در نهایت نیز بررسی می شود که آیا تمام پیام در تصویر قرار داده شده یا نه، اگر نشده باشد، پیغام خطا نمایان می شود که بیان میکند پیام انتخابی از سایزی که بتوان آن را در تصویر قرار داد بزرگتر است.

```
function BinarizedMessage = messageBinarizer(message, dataset)
    BinarizedMessage = '';
    for char = message
        index = find([dataset{1,:}] == char, 1);
        if ~isempty(index)
            BinarizedMessage = [BinarizedMessage dataset{2, index}];
        else
            error(['Character ', char, ' does not exist in the dataset.']);
        end
    end
end
```

در این تابع برای هر کارکتر در دیتاست جست و جو می شود، اگر آن وجود داشت مقدار باینری آن به یک string اضافه می شود؛ و اگر نه پیغام خطا نمایش داده می شود.

Decoding:

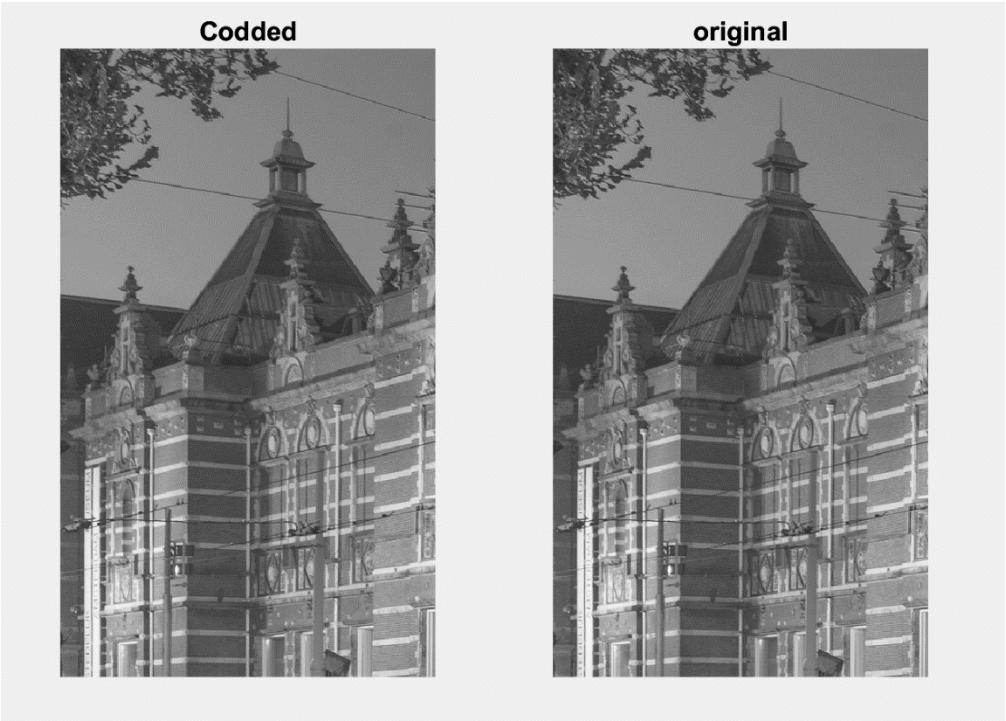
```
function message = decoding(codedImage, dataset, blockSize, Threshold)
    BinaryMessage = '';
    [rows, cols] = size(codedImage);

    for i = 1:blockSize(1):rows-4
        for j = 1:blockSize(2):cols-4
            block = codedImage(i:i+4, j:j+4);
            if var(double(block(:))) > Threshold
                for x = i:i+4
                    for y = j:j+4
                        BinaryMessage = [BinaryMessage, num2str(bitget(codedImage(x, y), 1))];
                    end
                end
            end
        end
    end

    message = BintoMsg(BinaryMessage, dataset);
end
```

تصویر کدگذاری شده، اندازه بلاک ها، threshold و دیتاست دریافت می شوند. حال برای هر بسته 5 تایی، اگر واریانس آن از threshold بالاتر بود، یعنی این بلاک کدگذاری شده است. سپس بیت آخر هر پیکسل بدست می آید و در رشته ای به نام binarymessage ذخیره شده تا در نهایت توسط تابع BintoMsg تبدیل به پیام قابل خواندن شود.

```
function Message = BintoMsg(BinaryMessage, dataset)
    Message = '';
    for j = 1:5:length(BinaryMessage) - 4
        BinaryChar = BinaryMessage(j:j+4);
        for i = 1:length(dataset(2, :))
            if strcmp(dataset{2, i}, BinaryChar)
                decodedChar = dataset{1, i};
                break;
            end
        end
        if exist('decodedChar', 'var') && decodedChar == ';'
            break;
        elseif exist('decodedChar', 'var')
            Message = [Message decodedChar];
        end
    end
    return;
end
```

در این تابع مسیج باینری دریافت می شود، آنگاه هر 5 عدد باینری با هم ذخیره شده و سپس در مپ ست میگردیم تا آن را پیدا کنیم، حرف معادل آن به string خروجی اضافه می شود تا در نهایت پیام دیکود شود. همچنین اگر حرف مدنظر یافت نشود، پیغام خطا نمایش داده می شود.

(1-3



تفاوتی دیده نمیشود، چرا که پیام در بلاک های 5*5 با بیشترین واریانس گنجانده شده تا تغییرات در آن زیاد باشد و تغییر lsb مشهود نشود.

5-1) در این روش چون بلام ها با واریانس زیاد را دیکود کرده ایم، به نظر می آید در دیکودینگ مشکلی ایجاد نشود. اگر از راه مقایسه عکس با عکس اولیه میرفتیم، احتمال ایجاد خطا رو دیکودینگ زیاد بود.

سوال 2:

برای سنتز:

روش کار در اینجا آن است که برای اعداد با توجه به مقدار انها میتواند با استفاده از mod و خارج قسمت تقسیم انها بر 3 جایگاه انها را تشخیص دهد. اما برای کاراکتر های* #ABCD باید به صورت دستی به انها یک شماره ستون و سطر تخصیص دهیم . سپس با این دیتای ساخته شده میتونیم شماره تلفن مد نظر را سنتز کنیم.

موجه تشکیل شده نیز مشابه آنچه در صورت پروژه نوشته شده بدست می آید.

```
testNumber='43218765';

fs=8000;
Ts=1/fs;
Ton=0.1;
Toff=0.1;
t=0:Ts:Ton;


fLow=[697 ,770 ,852 ,941];
fUp=[1209 ,1336 ,1477 ,1633];

silence=zeros(1,size(t,2)-1);


out=[];
for n=1:length(testNumber)

    switch testNumber(n)
        case 'A'
            row=1;
            column=4;
        case 'B'
            row=2;
            column=4;
        case 'C'
            row=3;
            column=4;
        case 'D'
            row=4;
            column=4;
        case '*'
            row=4;
            column=1;
        case '0'
            row=4;
            column=2;
        case '#'
            row=4;
            column=3;

        otherwise
            num=str2num(testNumber(n));
            row=ceil((num)/3);
            column=rem(num,3);

            if column==0
                column=3;
            end
    end
    %disp([row , column]);
    y1=sin(2*pi*fLow(row)*t);
    y2=sin(2*pi*fUp(column)*t);
    y=(y1+y2)/2;
    on=Ton*fs;
    out=[out y(1:on) silence];
end

sound(out,fs)
audiowrite('./y.wav',out,fs)
```

برای آنالیز:

برای هر یک از کاراکتر های موجود در تلفن با توجه به فرکانس بالا و فرکانس پایین انها برای هر کدام یک موج تولید میکنیم که نمایانگر آن کاراکتر است ؛ در نتیجه یک مپ ست ساخته ایم . سپس برای تشخیص صدای ورودی، با استفاده از کورولیشین گیری تشخیص دهیم که یک تکه از صدای ورودی مشابه ترین به کدام موج برای کاراکتر است.

```
[a,Fs]=audioread("a.wav");
data=cell(2,16);
fLow=[697 ,770 ,852 ,941];
fUp=[1209 ,1336 ,1477 ,1633];
fs=8000;
Ts=1/fs;

Ton=0.1;
t=0:Ts:Ton;

Toff=0.1;
on=Ton*fs;

s=size(a);
dataLabel=['1','2','3','A','4','5','6','B','7','8','9','C','*','0','#','D'];

output=[];


for n=1:length(dataLabel)
    num=n;
    row=ceil(num/4);
    column=rem(num,4);
    if column==0
        column=4;
    end

    disp([row, column]);
    y1=sin(2*pi*fLow(row)*t);
    y2=sin(2*pi*fUp(column)*t);

    y=(y1+y2)/2;

    data(1,n)={dataLabel(n)};
    data(2,n)={y(1:on)};
end

for n=0:(s(1)/(0.2*Fs))-1
    samples=[(2*n*Fs*0.1)+1,(2*n*Fs*0.1)+on];

    [b,Fs]=audioread("a.wav",samples);
    ro=zeros(1,16);

    for i=1:16
        ro(i)=corr2(data{2,i},transpose(b));
    end
    [MAXRO,pos]=max(ro);
    na=cell2mat(data(1,pos));

    output=[output na];
end
disp(output);
```

Command Window

```
     4     4
```

سوال 3:

به شکل اسکریپت برای ران کردن:

```
IC = imread('IC.png');
PCB = imread('PCB.jpg');
grayIC = rgb2gray(IC);
grayPCB = rgb2gray(PCB);
rotatedGrayIC = imrotate(grayIC , 180);
out = Corrolator(grayPCB , grayIC);
outRotated = Corrolator(grayPCB , rotatedGrayIC);
ICSpotPlotting(PCB , IC , {out , outRotated});
```

تبدیل شده به فانکشن:

```
function ICRecognition(ICpath , PCBpath)
    IC = imread(ICpath);
    PCB = imread(PCBpath);
    grayIC = rgb2gray(IC);
    grayPCB = rgb2gray(PCB);
    rotatedGrayIC = imrotate(grayIC , 180);
    out = Corrolator(grayPCB , grayIC);
    outRotated = Corrolator(grayPCB , rotatedGrayIC);
    ICSpotPlotting(PCB , IC , {out , outRotated});

end
```

تابع ICRecognition دو ورودی دریافت میکنند که آنها که آدرس های ذخیره سازی تصاوری IC , PCB هستند . سپس دو عکس را میخواند و در متغیر های IC , PCB ذخیره میکند . برای ساده سازی و گرفتن correlation در ادامه، دو عکس با تابع gray2rgb تبدیل به عکس های خاکستری میشوند.

با توجه به اینکه در صورت پروژه ذکر شده که IC میتواند در حالت ۱۸۰ درجه هم چرخیده باشد بنابراین با استفاده از تابع imroatate یک عکس سیاه سفید مشابه قبلی ولی با 90 درجه چرخش ذخیره میکنیم. در نهایت کورولیشن گیری دو بار هم روی عکس اولیه و هم روی عکس ۱۸۰ درجه چرخیده باید لحاظ کنیم تا بتوانیم IC های چرخانده شده را تشخیص بدهیم

فانکشن جهت کورولیشن گرفتن:

```
function out=Corrolator (PCB,IC)
    [ICRow, ICCol] = size(IC);
    [PCBRow, PCBCol] = size(PCB);
    IC = double(IC);
    out=zeros(PCBRow-ICRow+1,PCBCol-ICCol+1);

    for i=1:(PCBRow-ICRow+1)
        for j=1:(PCBCol-ICCol+1)
            smallerPCB=double(PCB(i:i + ICRow - 1, j:j + ICCol - 1));
            out(i,j)=corrcoeff(IC,smallerPCB);
        end
    end

end
```

تابع نوشته شده تابع جهت محاسبه کورولیشن است. با توجه به صورت مسئله، باید کورولیشین نسبت به دو عکس نرمالیزه شود، پس از تابع corcoeff برای این امر استفاده میکنیم.

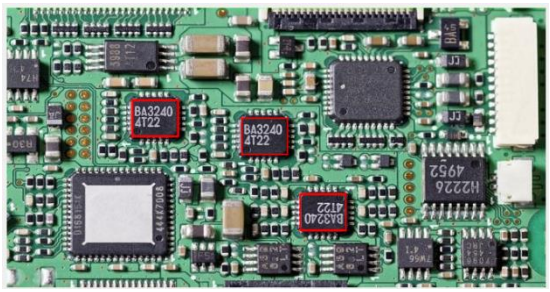
تابع نرمالیزه کردن:

```
function coeff = corrcoeff(p1 , p2)
    coeff = sum((p1.*p2)) / sqrt( sum(p1.*p1) .* sum(p2.*p2) );

end
```

.رای این که اثر یک مشاهده که صرفا دامنه ی زیادی دارد با مشاهده ی دیگری که دامنه ی زیادی ندارد مشابه باشد، این نرمالیزاسیون انجام می شود .

درنهایت پس از کورولیشین گیری باید نتیجه ی کورولیشین گیری را یک بار برای حالت عادی ic و یک بار برای ic با ۱۸۰ درجه دوران را در یک CELL میریزیم و با توجه به مختصات درونorrelationResult مستطیل های دیتکت شده در آن را روی PCB نشان میدهیم.

```
function result=ICSpotPlotting(PCB,IC,correlationResult)
    [IC_row,IC_col , ~]=size(IC);
    threshold=0.94;
    figure,imshow(PCB);
    hold on

    for l=1:length(correlationResult)
        CorResultM=cell2mat(correlationResult(1,l));
        Detected=find(CorResultM>threshold);
        [rows, columns]=ind2sub(size(CorResultM),Detected);
        for k=1:length(rows)
            disp([columns(k) rows(k) IC_col IC_row ])
            rectangle('Position',[columns(k) rows(k) IC_col IC_row],'EdgeColor','r');
        end
    end

    F=getframe(gcf);
    result=frame2im(F);
end
```



سوال چهارم:

```
clc
clear
success=calling_costumer(11,9);
if success~=1
    error('invalid input');
end
```

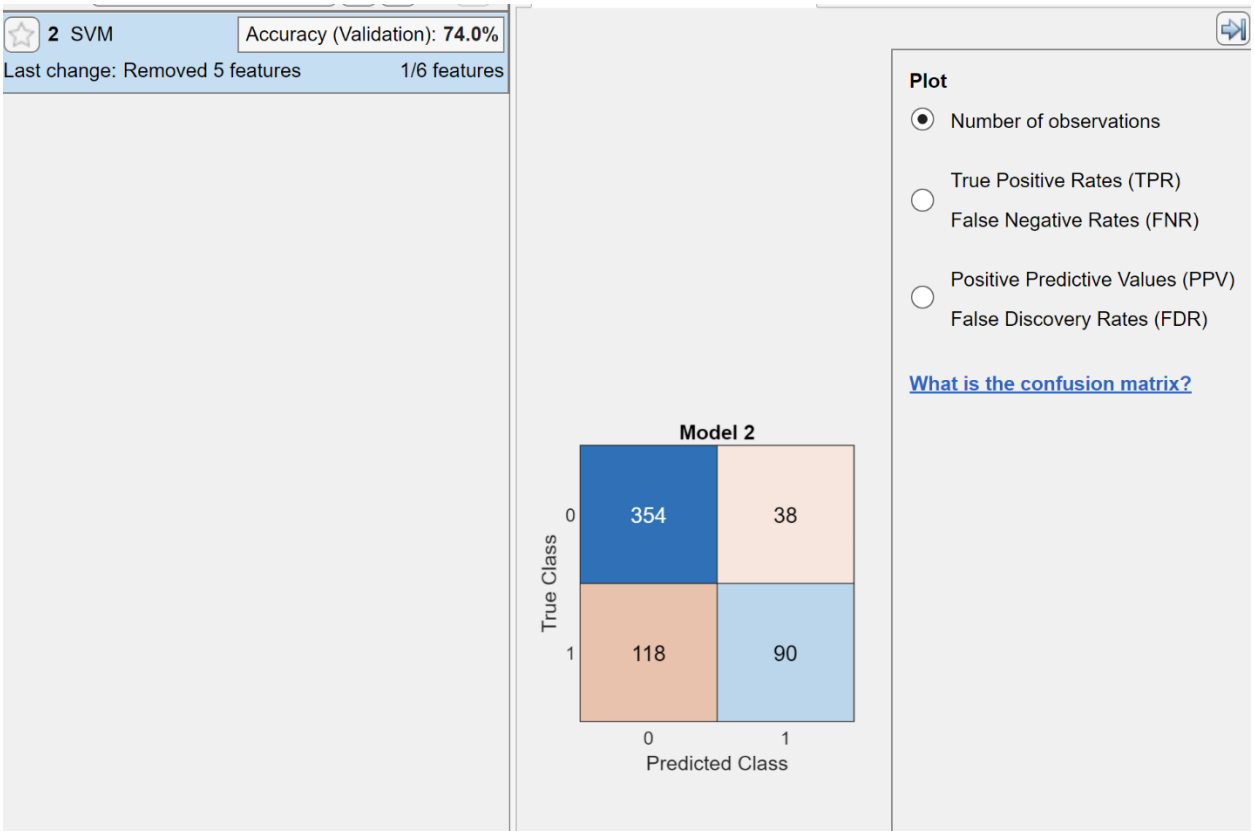
```
function called = calling_costumer(number1,number2)
files = dir('voice/*.wav');
len = length(files);
TRAIN = cell(2, len);

input = number1;

for i = 1:len
    filename = fullfile('voice', files(i).name);
    [a, Fs] = audioread(filename);
    TRAIN{1, i} = a;
    [~, name, ~] = fileparts(files(i).name);
    TRAIN{2, i} = str2double(name);
end

save('TRAININGSET.mat', 'TRAIN');

digits = zeros(2,1);
digits(2,1) = rem(input,10);
digits(1,1) = floor(input/10)*10;
newX = zeros(1000,2);
starter =1;
fade_length = 0.001 * Fs;
fade_window = hann(fade_length * 2);

if input <= 20
    for i = 1:28
        if TRAIN{2, i} == input
            audio = TRAIN{1, i};
            fadeIn = audio(1:fade_length, :) .* fade_window(1:fade_length);
            fadeOut = audio(end-fade_length+1:end, :) .* flipud(fade_window(fade_length+1:end));
            crossfade = linspace(1, 0, fade_length).' .* fadeOut(1:fade_length, :) + linspace(0, 1, fade_length).' .* fadeIn(end-fade_length+1:end,
            newX(1:length(audio), :) = [fadeIn; audio(fade_length+1:end-fade_length, :); crossfade];
            starter = starter + length(audio);
            break;
        end
    end
else
    for j = 1:2
        for i = 1:size(TRAIN, 2)
            if TRAIN{2, i} == digits(j, 1)
                audio = TRAIN{1, i};
                fadeIn = audio(1:fade_length, :) .* fade_window(1:fade_length);
                fadeOut = audio(end-fade_length+1:end, :) .* flipud(fade_window(fade_length+1:end));
                crossfade = linspace(1, 0, fade_length).' .* fadeOut(1:fade_length, :) + linspace(0, 1, fade_length).' .* fadeIn(end-fade_length+1:
                newX(starter:starter+length(audio)-1, :) = [fadeIn; audio(fade_length+1:end-fade_length, :); crossfade];
                starter = starter + length(audio) - fade_length;
                break;
            end
        end
        if digits(j, 1) > 10
            audio = TRAIN{1, 28};
            fadeIn = audio(1:fade_length, :) .* fade_window(1:fade_length);
            fadeOut = audio(end-fade_length+1:end, :) .* flipud(fade_window(fade_length+1:end));
            crossfade = linspace(1, 0, fade_length).' .* fadeOut(1:fade_length, :) + linspace(0, 1, fade_length).' .* fadeIn(end-fade_length+1:end,
            newX(starter:starter+length(audio)-1, :) = [fadeIn; audio(fade_length+1:end-fade_length, :); crossfade];
            starter = starter + length(audio) - fade_length;
        end
    end
```

```
        if digits(j, 1) > 10
            audio = TRAIN{1, 28};
            fadeIn = audio(1:fade_length, :) .* fade_window(1:fade_length);
            fadeOut = audio(end-fade_length+1:end, :) .* flipud(fade_window(fade_length+1:end));
            crossfade = linspace(1, 0, fade_length).' .* fadeOut(1:fade_length, :) + linspace(0, 1, fade_length).' .* fadeIn(end-fade_length+1:end,
            newX(starter:starter+length(audio)-1, :) = [fadeIn; audio(fade_length+1:end-fade_length, :); crossfade];
            starter = starter + length(audio) - fade_length;
        end
    end
end

audio = TRAIN{1, 29};
fadeIn = audio(1:fade_length, :) .* fade_window(1:fade_length);
fadeOut = audio(end-fade_length+1:end, :) .* flipud(fade_window(fade_length+1:end));
crossfade = linspace(1, 0, fade_length).' .* fadeOut(1:fade_length, :) + linspace(0, 1, fade_length).' .* fadeIn(end-fade_length+1:end, :);
newX(starter:starter+length(audio)-1, :) = [fadeIn; audio(fade_length+1:end-fade_length, :); crossfade];
starter = starter + length(audio) - fade_length;
for i = 1:28
        if TRAIN{2, i} == number2
            audio = TRAIN{1, i};
            fadeIn = audio(1:fade_length, :) .* fade_window(1:fade_length);
            fadeOut = audio(end-fade_length+1:end, :) .* flipud(fade_window(fade_length+1:end));
            crossfade = linspace(1, 0, fade_length).' .* fadeOut(1:fade_length, :) + linspace(0, 1, fade_length).' .* fadeIn(end-fade_length+1:end,
            newX(starter:starter+length(audio)-1, :) = [fadeIn; audio(fade_length+1:end-fade_length, :); crossfade];
            break;
        end
end
called=1;
audiowrite("final.wav",newX,Fs);
end
```

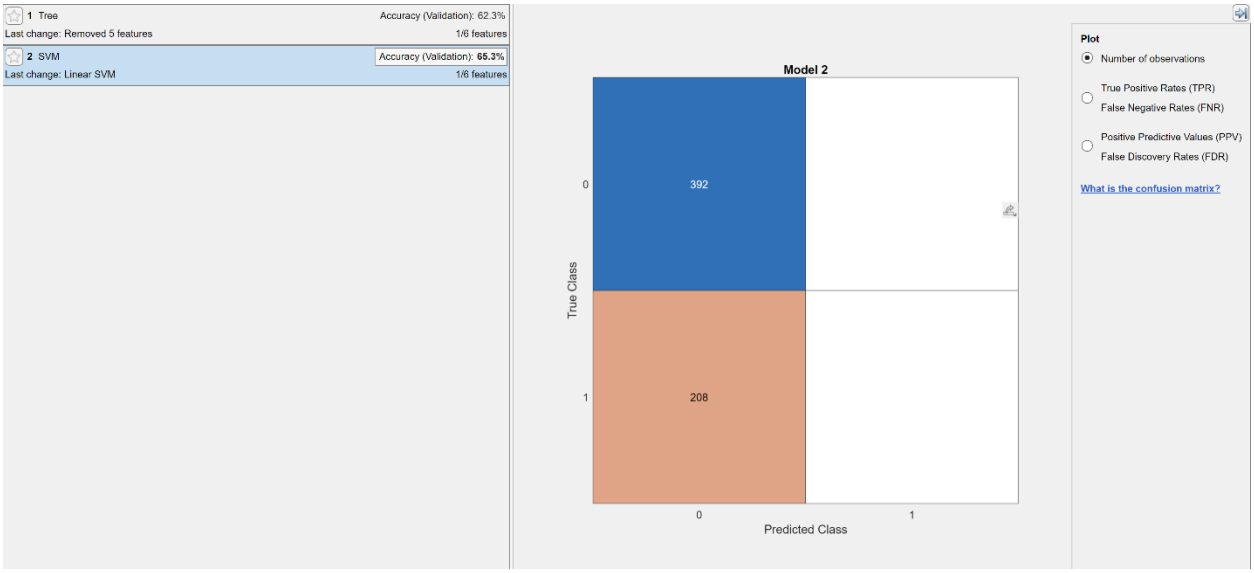
ابتدا با ویس های ضبط شده برای هر عدد و عدد متناسب با آن، یک مپ ست تشکیل می شود. سپس عدد مد نظر بررسی می شود؛ اگر کمتر از بیست بود به صورت یک عدد خوانده می شود و در نتیجه ذخیره و سپس به صورت یک فایل wav. سیو می شود. اگر عدد بزرگتر از 20 بود، عدد دو بخش شده و به صورت یکان + و + دهگان سیو می شود. همچنین از کراسفید مقداری بهره برده شده است تا صدا بیش از حد قطه قطعه نشود.

سوال پنجم:

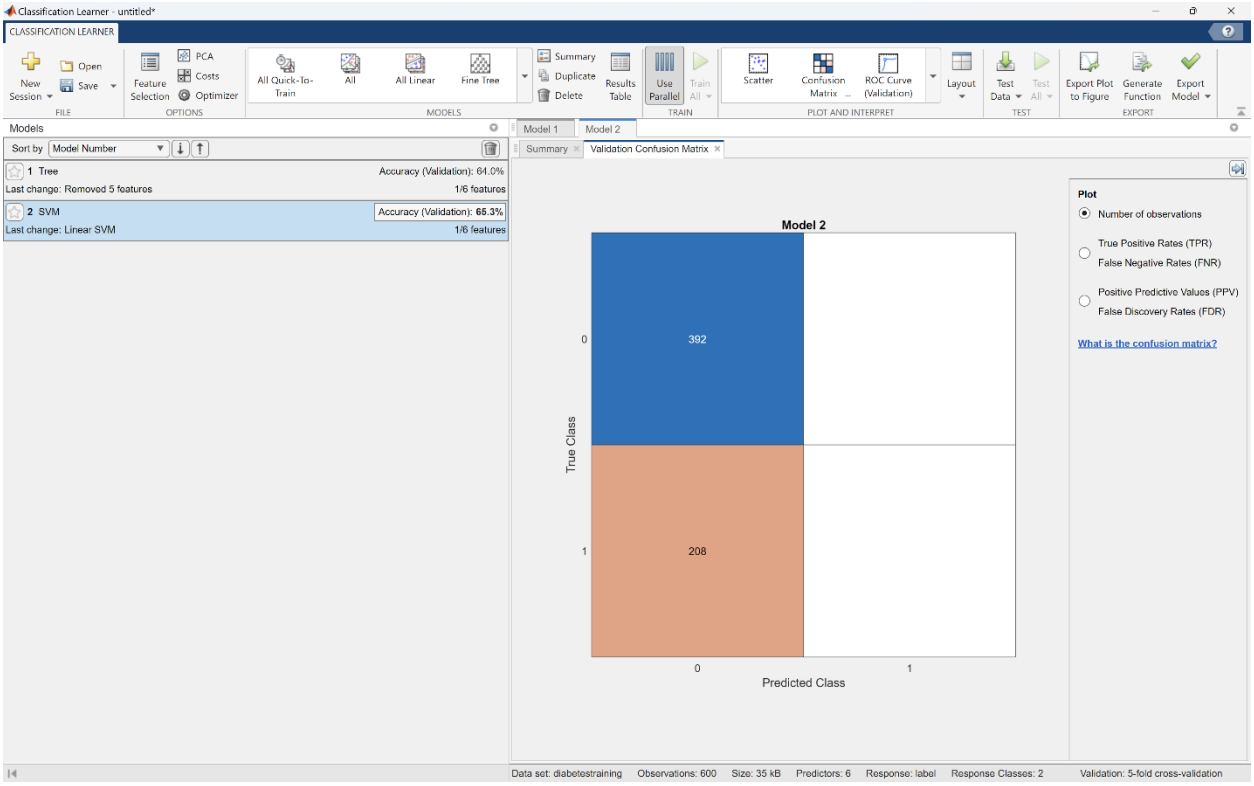
با glucose:



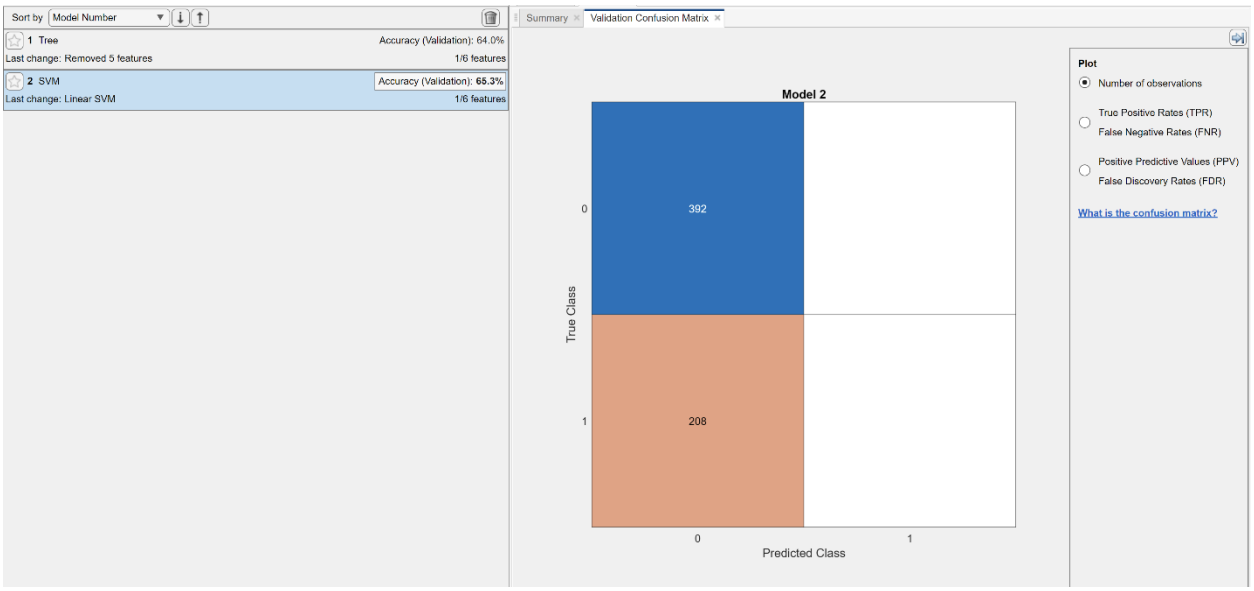
با blood pressure:



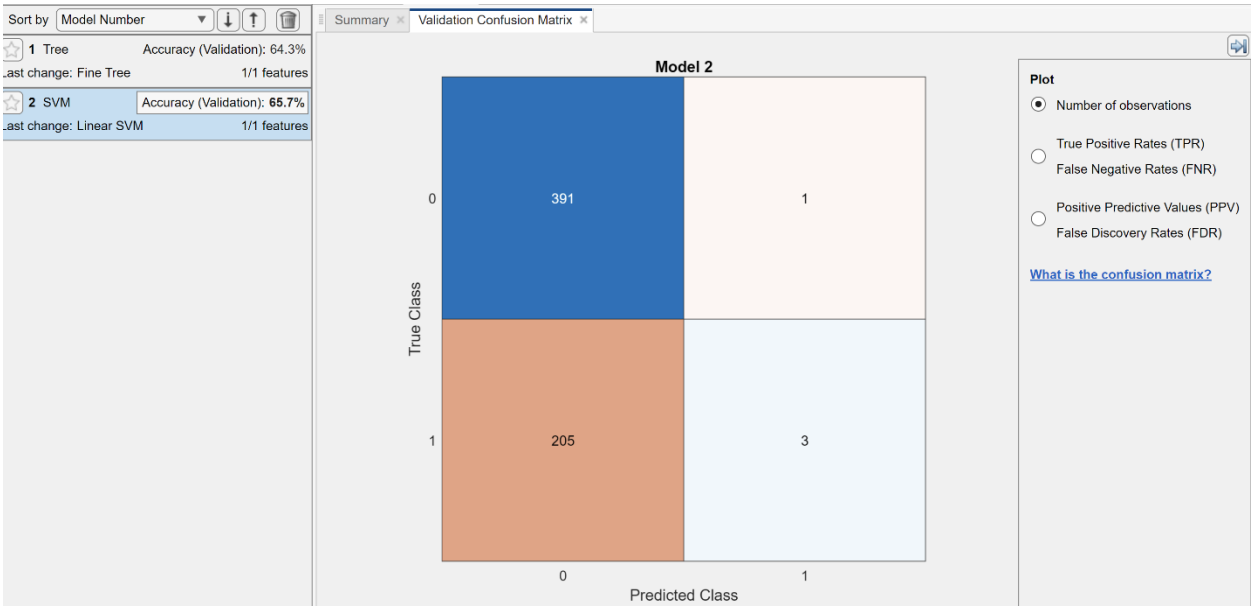
با skin thickness:



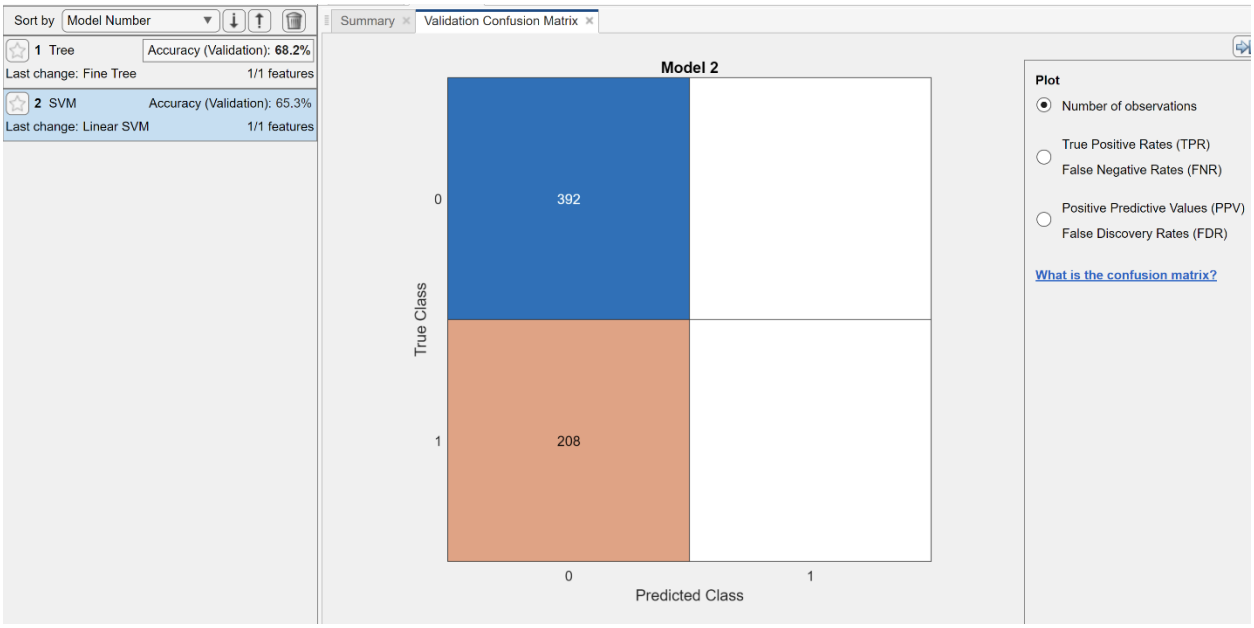
با Insulin:



با BMI:


Sort by Model Number
1 Tree
Accuracy (Validation): 64.3%
Last change: Fine Tree
1/1 features
2 SVM
Accuracy (Validation): 65.7%
Last change: Linear SVM
1/1 features
Summary
Validation Confusion Matrix
Model 2
391
1
205
3
True Class
Predicted Class
0
1
Plot
Number of observations
True Positive Rates (TPR)
False Negative Rates (FNR)
Positive Predictive Values (PPV)
False Discovery Rates (FDR)
What is the confusion matrix?

با age:


Sort by Model Number
1 Tree
Accuracy (Validation): 68.2%
Last change: Fine Tree
1/1 features
2 SVM
Accuracy (Validation): 65.3%
Last change: Linear SVM
1/1 features
Summary
Validation Confusion Matrix
Model 2
392
208
True Class
Predicted Class
0
1
Plot
Number of observations
True Positive Rates (TPR)
False Negative Rates (FNR)
Positive Predictive Values (PPV)
False Discovery Rates (FDR)
What is the confusion matrix?

بخش 5-3)

```
data = readtable('diabetes-training.csv');

features = data(:, 1:end-1);
trueLabels = table2array(data(1:600, end));
predictedLabels = trainedModel.predictFcn(features);

correctPredictions = predictedLabels == trueLabels;

accuracy = sum(correctPredictions) / size(data, 1);

disp(['Accuracy: ', num2str(accuracy * 100), '%']);
```

Command Window

```
>> p5
Accuracy: 77.5%
```

در این بخش دقت 77.5% بدست آمد.

بخش 5-4)

```
%data = readtable('diabetes-training.csv');
data = readtable('diabetes-validation.csv');

features = data(:, 1:end-1);
trueLabels = table2array(data(:, end));
predictedLabels = trainedModel.predictFcn(features);

correctPredictions = predictedLabels == trueLabels;

accuracy = sum(correctPredictions) / size(data, 1);

disp(['Accuracy: ', num2str(accuracy * 100), '%']);
```

Command Window

```
>> p5
Accuracy: 78%
```

دقت این بخش 78% است.