

به نام خدا

پروژه چهارم درس سیگنال‌ها و سیستم‌ها

فاطمه زهرا برومندنیا-۸۱۰۱۰۰۰۹۴

بخش اول

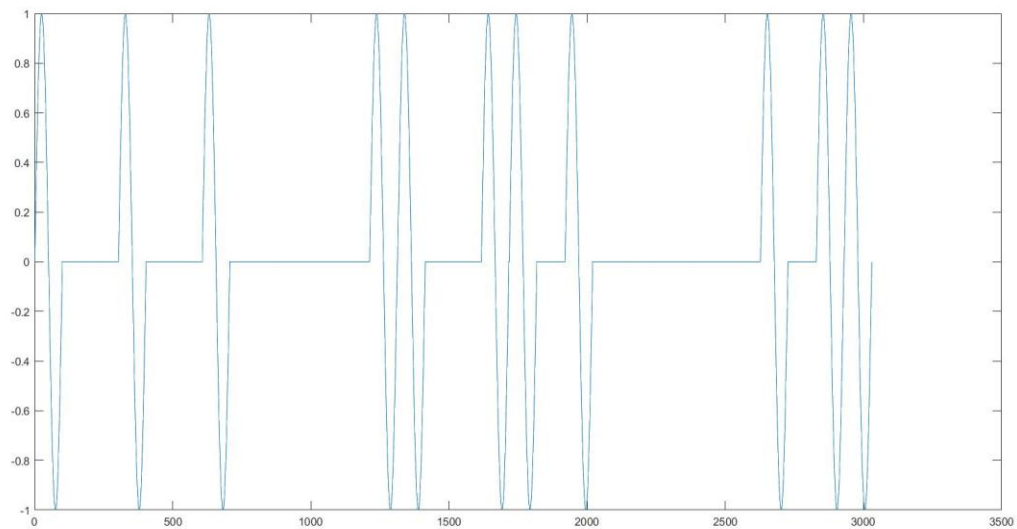
(۱-۱ و ۲-۱)

```
Editor - p1.m
1 - close all
2 - Alphabet='abcdefghijklmnopqrstuvwxyz .,!"';
3
4 - numChar=length(Alphabet);
5 - mapset=cell(2,numChar);
6 - for i=1:numChar
7 -     mapset{1,i}=Alphabet(i);
8 -     mapset{2,i}=dec2bin(i-1,5);
9 - end
10 - msg='sigal';
11
12 - outSig=coding_amp(msg,mapset,1);
13 - noise=0.0001*randn(1,3030);
14 - noisySig=noise+outSig;
15 - plot(1:length(noisySig),noisySig);
16
17 - decodedMsg=decoding_amp(outSig,mapset,1);
```

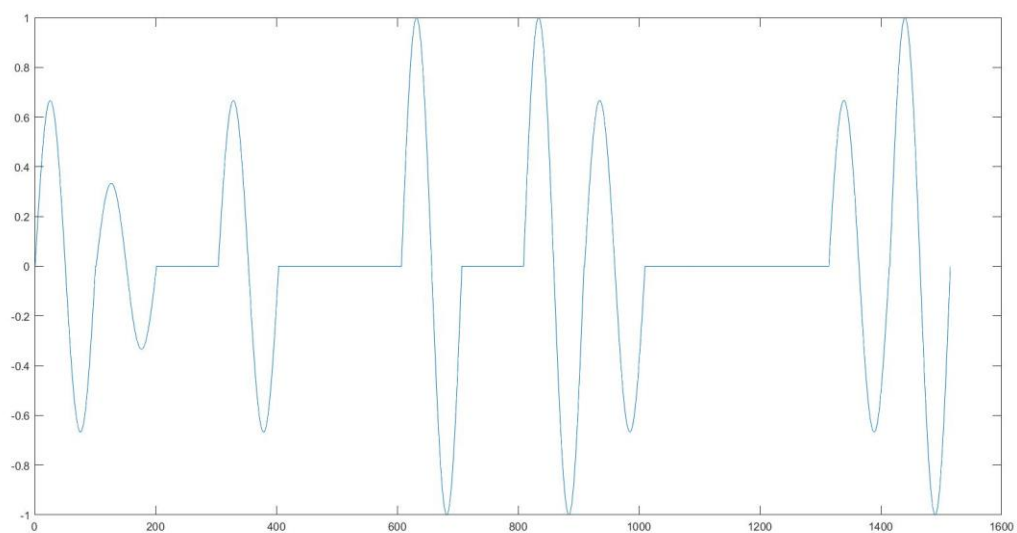
(۳-۱) این تابع ابتدا ایندکس‌های مربوط به هر کاراکتر را در 'mapset' پیدا می‌کند و آنها را در 'idx' ذخیره می‌کند. سپس پیام را به صورت دودویی در 'binMsg' ذخیره می‌کند.

سپس، تعداد نمونه‌های سیگنال را با استفاده از 'fs' (فرکانس نمونه‌برداری) و 'Ton' (مدت زمان سیگنال) تعیین می‌کند. سپس یک حلقه بر روی بیت‌های پیام ایجاد می‌کند و سپس عدد دودویی را به کسر معادل آن تبدیل کرده و در 'fraction' ذخیره می‌کند. سیگنال به صورت 'y' برابر با 'fraction' ضربدر سینوسی با فرکانس '2*pi*t' تعریف می‌شود. سیگنال نهایی با ادغام سیگنال‌های تولید شده در هر بیت در 'Y' ساخته می‌شود. در نهایت، سیگنال نهایی در 'signal' ذخیره می‌شود و به عنوان خروجی تابع برگردانده می‌شود.

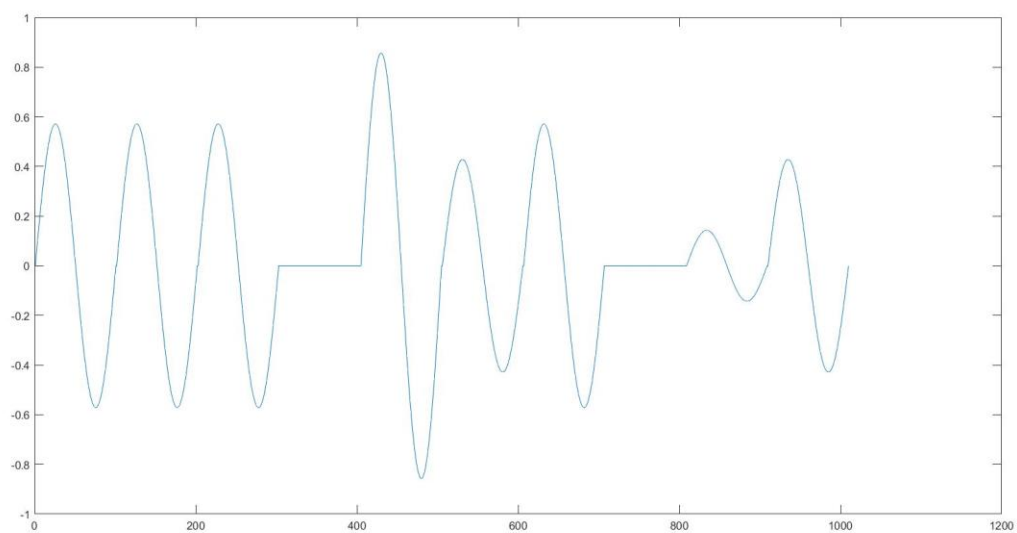
```
Editor - coding_amp.m
1  function signal=coding_amp(msg,mapset,bitrate)
2
3      idx=[];
4      for i=1:length(msg)
5          ch=msg(i);
6          idx=[idx, find(strcmp(ch,mapset(1,:))==1)];
7      end
8      binMsg=cell2mat(mapset(2,idx));
9      fs=100;
10     Ts=1/fs;
11     Ton=1;
12     t=0:Ts:Ton;
13     Y=[];
14     for i=1:bitrate:length(binMsg)
15         m=bin2dec(binMsg(i:i+bitrate-1));
16         fraction=bin2dec(binMsg(i:i+bitrate-1))/(2^bitrate-1);
17         y=fraction.*sin(2*pi*t);
18         plot(1:length(y),y);
19         Y=[Y y];
20     end
21     Y;
22     plot(1:length(Y),Y);
23     signal=Y;
24     end
```



نرخ بیت ۱



نرخ بیت ۲



نرخ بیت ۳

۴-۱) سیگنال، سیگنال ورودی است که باید رمزگشایی شود.

در تابع مقدار آستانه بر اساس بیتیت محاسبه و برای تعیین مقدار کورلیشن حداقل برای دیکودینگ استفاده می شود.

سپس تابع وارد یک حلقه می شود که در هر بار از آن، سیگنال را در قطعات **chunk** مشخص شده استخراج کرده و با یک سیگنال (**template**) که همان سینوس دو پی تی است کورلیشن را محاسبه می کند.

مقدار کورلیشن با عناصر آرایه **fraction_arr** که شامل کسرهایی بین ۰ تا ۱ هستند، مقایسه شده و با استفاده از توابع **min** و **abs**، شاخص نزدیک ترین کسر به مقدار کورلیشن بدست می آید.

اگر مقدار کورلیشن کمتر از آستانه باشد، نمایش دودویی شاخص به رشته **binMsg** با استفاده از تابع **dec2bin** الحاق می شود. در غیر این صورت، علامت سوال ('?') برای نشان دادن خطا در رمزگشایی الحاق می شود.

پس از پایان حلقه، طول رشته **binMsg** به ۵ تقسیم شده و حروف معادل آن محاسبه می شود. با استفاده از تابع **find**، شاخص نماد متناظر در آرایه **mapset** بر اساس مقایسه با ردیف دوم آرایه به دست می آید.

در نهایت، با ادغام مقادیر متناظر از ردیف اول آرایه **mapset**، پیام رمزگشایی شده به متغیر **msg** اختصاص داده می شود.

به طور خلاصه، این کد با استفاده از الگوی مطابقت الگو و کورلیشن، سیگنال را رمزگشایی کرده و پیام رمزگشایی شده را بر اساس مجموعه نگاشت بازمی گرداند.

```
Editor - decoding_amp.m
1 function msg=decoding_amp(signal,mapset,bitrate)
2     Fs=100;
3     Ts=1/Fs;
4     Ton=1;
5     chunk = round(Ton * Fs);
6     t=0:Ts:Ton;
7
8     binMsg='';
9     threshold = 2*(2^bitrate-1);
10
11     fraction_arr=[];
12     for i=0:1:(2^bitrate-1)
13         fraction=i/(2^bitrate-1);
14         fraction_arr = [fraction_arr fraction];
15     end
16
17     for i = 1:chunk:(length(signal) - chunk)
18         sig_part = signal(i:i+chunk-1);
19
20         template = 2*sin(2*pi*t);
21         correlation = max(0.01*xcorr(sig_part, template));
22         m=abs(fraction_arr - correlation);
23         [u, idx] = min(abs(fraction_arr - correlation));
24
25         if threshold > u
26             binMsg=strcat(binMsg,dec2bin(idx-1,bitrate));
27         else
28             binMsg=strcat(binMsg,'?');
29         end
30     end
```

(۵-۱)

همان طور که در تصاویر زیر مشخص است، میانگین داده‌ها منفی ۰.۰۱۷۶ گزارش شده که به صفر نزدیک است و واریانس هم ۰.۹۹۵ که به یک نزدیک است. در خروجی پلات هم نشان داده شده که توزیع گاوسی است، و با خط قرمز روی آن توزیع گاوسی واقعی نشان داده شده است.

```

18 %%%%%%%%%%%%%%% part 1-5
19 data = randn(1,3000);
20 histogram(data, 'Normalization', 'pdf');
21 hold on;
22 x = -4:0.1:4;
23 y = normpdf(x, 0, 1);
24 plot(x, y, 'r', 'LineWidth', 2);
25 mu = mean(data);
26 sigma = std(data);
27 disp(['mean: ', num2str(mu)]);
28 disp(['variance', num2str(sigma^2)]);

```

calling_customer.m × training_loading.m × trainClassifierDiabet

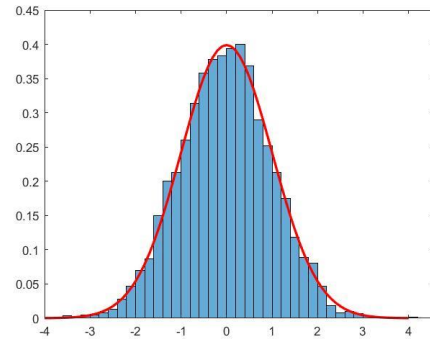
Command Window

New to MATLAB? See resources for [Getting Started](#).

```

>> p1
mean: -0.017673
variance0.99552

```



۶-۱) برای هر سه نرخ درست تشخیص داده شد. در تصاویر زیر خروجی هرکدام آمده است:

```

30 %%%%%%%%%%%%%%% part 1-6
31 bitRate=1;
32 outSig=coding_amp(msg,mapset,bitRate);
33 noise=0.0001*randn(1,length(outSig));
34 noisySig=noise+outSig;
35 plot(1:length(noisySig),noisySig);
36 decodedNoisyMsg1=decoding_amp(noisySig,mapset,bitRate)

```

calling_customer.m × training_loading.m × trainClassifierDiabetes.m × p4.m ×

Command Window

```

decodedNoisyMsg1 =

    'signal'

```

نرخ بیت ۱

```

30 %%%%%%%%%%% part 1-6
31 - bitRate=2;
32 - outSig=coding_amp(msg,mapset,bitRate);
33 - noise=0.0001*randn(1,length(outSig));
34 - noisySig=noise+outSig;
35 - plot(1:length(noisySig),noisySig);
36 - decodedNoisyMsg1=decoding_amp(noisySig,mapset,bitRate)

```

calling_customer.m x training_loading.m x trainClassifierDiabetes.m x p4.m

Command Window

```

decodedNoisyMsg1 =

'signal'

```

نرخ بیت ۲

```

30 %%%%%%%%%%% part 1-6
31 - bitRate=3;
32 - outSig=coding_amp(msg,mapset,bitRate);
33 - noise=0.0001*randn(1,length(outSig));
34 - noisySig=noise+outSig;
35 - plot(1:length(noisySig),noisySig);
36 - decodedNoisyMsg1=decoding_amp(noisySig,mapset,bitRate)
37

```

calling_customer.m x training_loading.m x trainClassifierDiabetes.m x p4.m x

Command Window

```

decodedNoisyMsg1 =

'signal'

```

نرخ بیت ۳

۷-۱) طبق جدول گزارش زیر، به ترتیب بیت ریت ۱ سپس ۲ و سپس ۳ مقاوم بودند. بله نتایج با آنچه در مقدمه گفته شد همخوانی دارد، چون داریم دامنه ثابت را به بازه های کوچک تری می شکنیم احتمال اینکه تشخیص غلط داده شود و در بازه اشتباهی بیفتد بیشتر است.

واریانس/بیت ریت	بیت ریت ۱	بیت ریت ۲	بیت ریت ۳
۰.۰۰۱	صحیح	صحیح	صحیح
۰.۰۱	صحیح	صحیح	صحیح
۰.۱	صحیح	صحیح	صحیح
۰.۴	صحیح	صحیح	غلط

۰.۸	صحیح	غلط	غلط
۱.۵	غلط	غلط	غلط

۸-۱) برای بیت ریت سه، تقریباً تا ۰.۳ مقاوم بود و پس از آن با ۰.۴ نتیجه غلط داد. برای بیت ریت ۲، تا ۰.۷ مقاوم بود و پس از آن با ۰.۸ نتیجه غلط داد. برای بیت ریت ۱، تا ۱.۴ مقاوم بود و پس از آن با ۱.۵ نتیجه غلط داد.

۱۰-۱) محدودیتی ندارد.

۱۱-۱) به نظر نمی رسد مقاوم تر شود، چون اگر فقط در دریافت این کار را کنیم، در واقع نویزها را هم به همان نسبت تشدید کرده ایم و اثری روی خوانا تر شدن سیگنال نخواهد داشت.

۱۲-۱) ما با نرخ ۳ بیت بر ثانیه ارسال کردیم اما برای اینترنت خانگی بین ۲ تا ۲۴ مگابیت بر ثانیه است. چیزی حدود ۱ تا ۸ میلیون برابر.

بخش دوم

۲-۲) بخش های اول، همانند تمرین کامپیوتری اول انجام شده است. در نهایت از کانولوشن خود متلب استفاده شده ولی بازه مشخصی از آن اسلایس شده؛ چون در تمرین قبلی مراحل شیفت دادن و کورلیشن گیری از صفر آغاز می شد و وقتی ابتدای تمپلت به انتهای سیگنال می رسید کار خاتمه پیدا می کرد. اما در کانولوشن تا انتهای سیگنال در حال شیفت به انتهای سیگنال ثابت نرسد کار تمام نمی شود. به همین خاطر اسلایس کرده ایم، در نقطه ای که کانولوشن ماکسیمم می شود یعنی بیشترین روی هم افتادگی و فضای مشترک را دارند و در واقع همان زمان درستی است که به دنبال آن می گردیم، ایندکس آن را در تابع زمان می گذاریم در سرعت ضرب می کنیم تا فاصله به دست آید.

```

Editor - p2.m
1  clc, clear, close all
2  ts=1e-9;
3  T=1e-5;
4  tau=1e-6;
5
6
7  t=0:ts:T;
8  tlen=length(t);
9  x=zeros(1,tlen);
10 t2=0:ts:tau;
11 x(1:length(t2))=ones(1,length(t2));
12 plot(t,x);
13 title('sent signal');
14 xlim([0, T]);
15 ylim([-1, 2]);
16 xlabel('time')
17 ylabel('amplitude')
18
19 R=450;
20 lightSpeed=3e+8;
21 index=2*R/lightSpeed;
22 td=0:ts:index;
23
24 y=zeros(1,tlen);
25 t2=0:ts:tau;
26 y(length(td):length(td)+length(t2))=0.5;
27 figure
28 plot(t,y);
29 title('recieved signal');
30
31 plot(t,y);
32 title('recieved signal');
33 xlim([0, T]);
34 ylim([-1, 2]);
35 xlabel('time')
36 ylabel('amplitude')
37
38 figure
39 conv_result=conv(y,x);
40 conv_result=conv_result(1:10001);
41 plot(t,conv_result);
42 title('convolve result');
43
44 [maximum2,idx2]=max(conv_result)
45 distance=(t(idx2)-tau)*lightSpeed/2
46

```


۳-۲) در این قسمت به کمک یک حلقه، ضریب نویز را از ۰.۱ تا عدد ۱۰ آرام آرام بالا می‌بریم، مراحل قبلی را طی می‌کنیم و در نهایت حساب می‌کنیم فاصله به دست آمده چقدر از حد آستانه که ۱۰ متر است فاصله دارد. اگر بیشتر بود، آن ضریب را به عنوان نتیجه که تا آن جا می‌توان افزایش داد خروجی می‌دهیم.

```

46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Part 2-3
47 Threshold=10;
48 for coef=0.1:0.1:10
49     noise=coef*rand(1,tlen);
50     z=y+noise;
51
52     conv_result=conv(z,x);
53     conv_result=conv_result(1:10001);
54     title('convolve result');
55
56     [maximum2,idx2]=max(conv_result);
57
58     distance=(t(idx2)-tau)*lightSpeed/2;
59     if (abs(distance-R)>Threshold)
60         coef
61         break
62     end
63 end

```

بخش سوم

(۳)

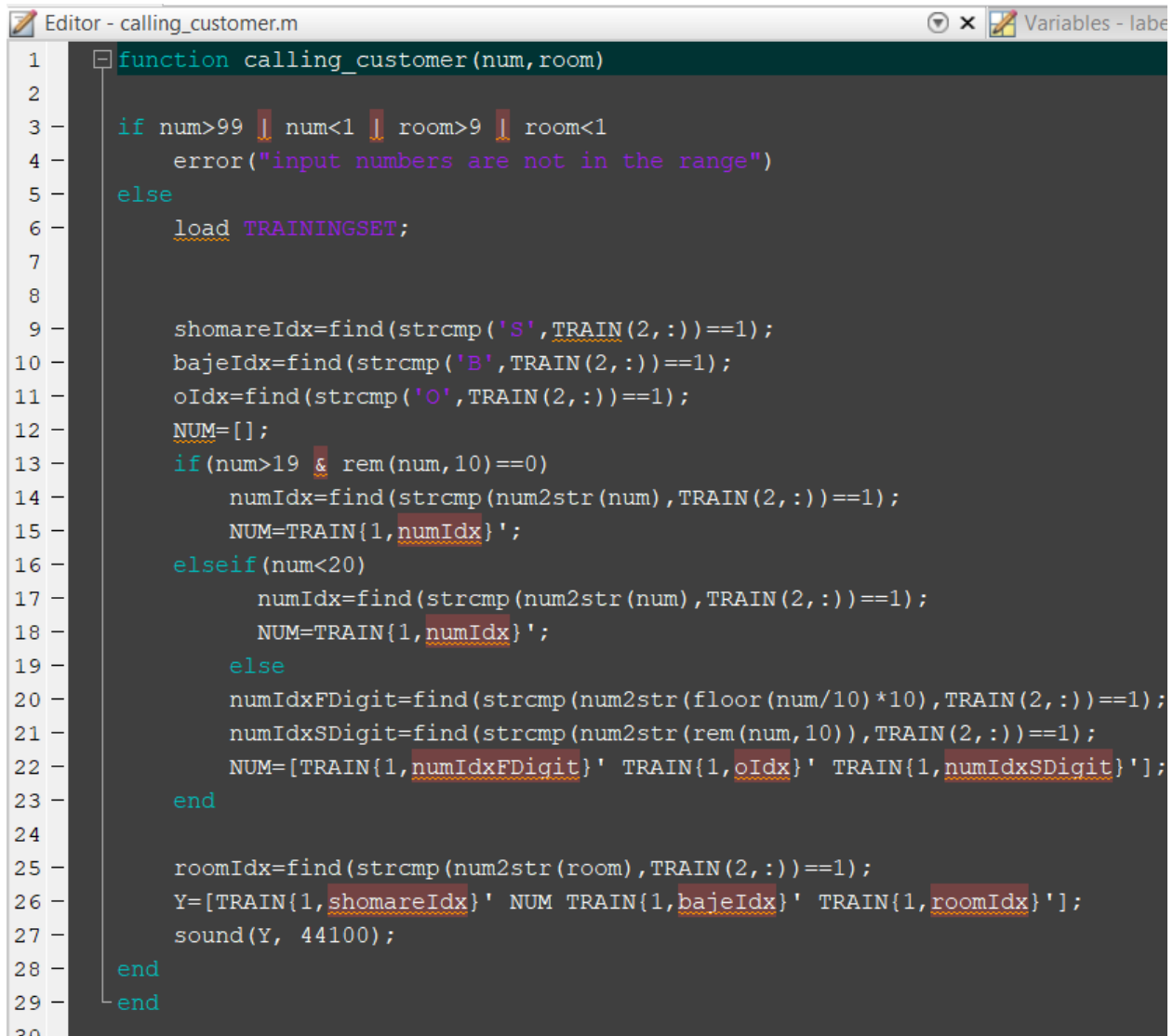
در این قسمت، ابتدا، کد فایل TRAININGSET را لود می‌کند. سپس با استفاده از توابع `find` و `strcmp`، اندیس‌های ستون‌های مربوط به شماره، باجه و دستگاه در مپ ست TRAININGSET را مشخص می‌کند.

در صورتی که ورودی‌ها خارج از محدوده معتبر باشند، یک خطا با پیام "input numbers are not in the range" ایجاد می‌شود.

سپس ماتریس NUM را تعریف می‌کند. اگر عدد `num` بزرگتر از ۱۹ و بخش پذیر بر ۱۰ باشد، اندیس مربوط به `num` را در TRAININGSET پیدا کرده و ستون متناظر را به ماتریس NUM اضافه می‌کند. اگر عدد `num` کوچکتر از ۲۰ باشد، به طور مشابه عمل می‌کند. در غیر این صورت، ابتدا اندیس عدد دهگان `num` را پیدا کرده و در NUM اضافه می‌کند، سپس اندیس عدد یکان `num` را پیدا کرده و نیز به NUM اضافه می‌کند.

سپس اندیس مربوط به باجه 'room' را در TRAININGSET پیدا کرده و ستون متناظر را به ماتریس Y اضافه می‌کند. ستون‌های مربوط به شماره، دستگاه باجه و شماره باجه در Y قرار می‌گیرند.

در نهایت، تابع 'sound' با ورودی Y و فرکانس نمونه‌برداری 44100 فراخوانی می‌شود تا صدای متناظر با Y پخش شود.



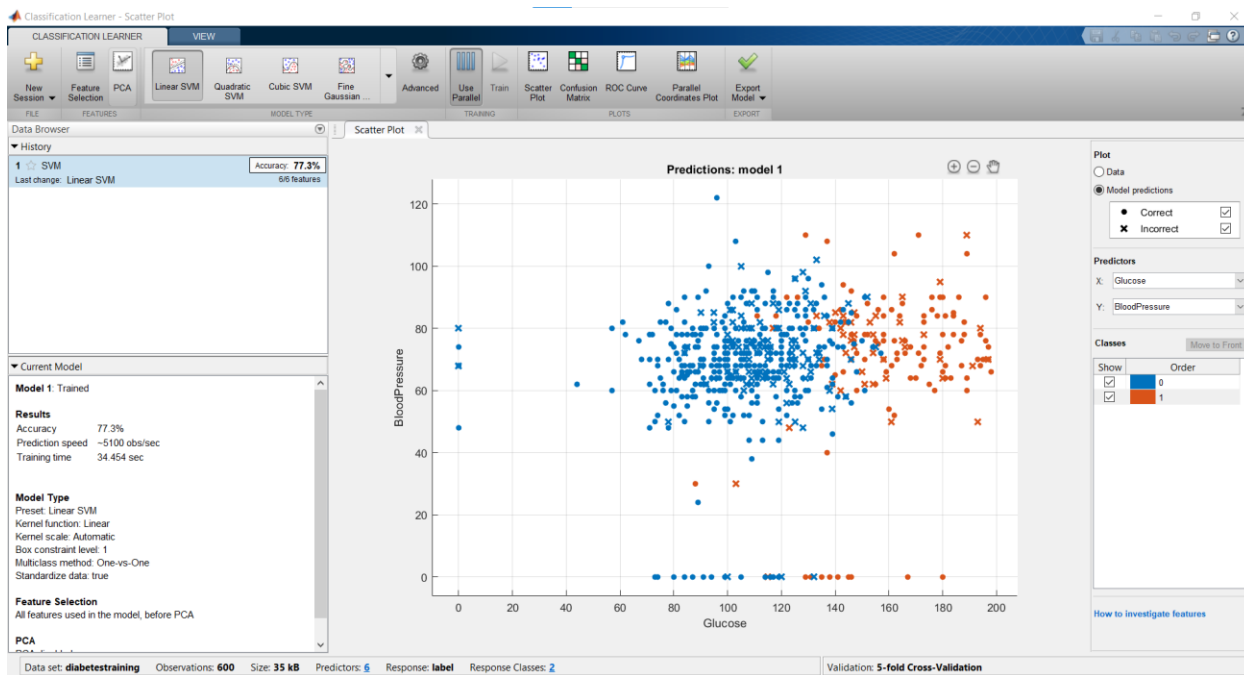
```
1 function calling_customer(num,room)
2
3 if num>99 || num<1 || room>9 || room<1
4     error("input numbers are not in the range")
5 else
6     load TRAININGSET;
7
8
9     shomareIdx=find(strcmp('S',TRAIN(2,:))==1);
10    bajeIdx=find(strcmp('B',TRAIN(2,:))==1);
11    oIdx=find(strcmp('O',TRAIN(2,:))==1);
12    NUM=[];
13    if(num>19 & rem(num,10)==0)
14        numIdx=find(strcmp(num2str(num),TRAIN(2,:))==1);
15        NUM=TRAIN{1,numIdx}';
16    elseif(num<20)
17        numIdx=find(strcmp(num2str(num),TRAIN(2,:))==1);
18        NUM=TRAIN{1,numIdx}';
19    else
20        numIdxFDigit=find(strcmp(num2str(floor(num/10)*10),TRAIN(2,:))==1);
21        numIdxSDigit=find(strcmp(num2str(rem(num,10)),TRAIN(2,:))==1);
22        NUM=[TRAIN{1,numIdxFDigit}' TRAIN{1,oIdx}' TRAIN{1,numIdxSDigit}'];
23    end
24
25    roomIdx=find(strcmp(num2str(room),TRAIN(2,:))==1);
26    Y=[TRAIN{1,shomareIdx}' NUM TRAIN{1,bajeIdx}' TRAIN{1,roomIdx}'];
27    sound(Y, 44100);
28 end
29 end
30
```

بخش چهارم

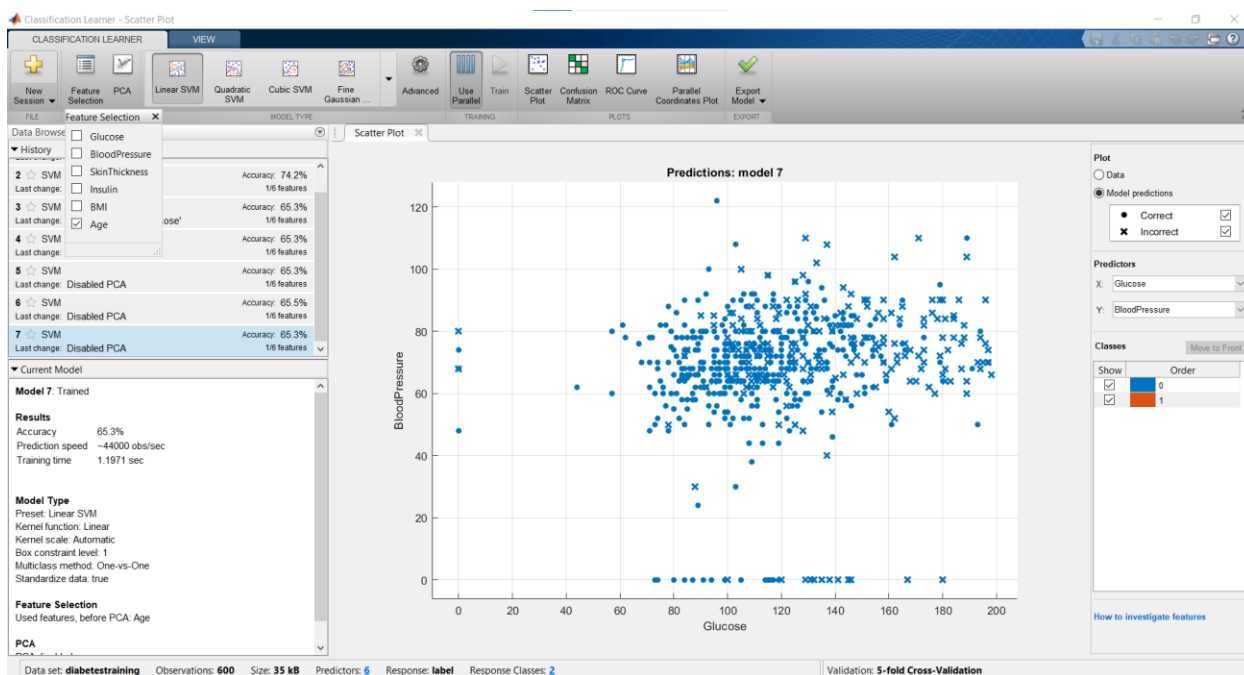
۲-۴

و

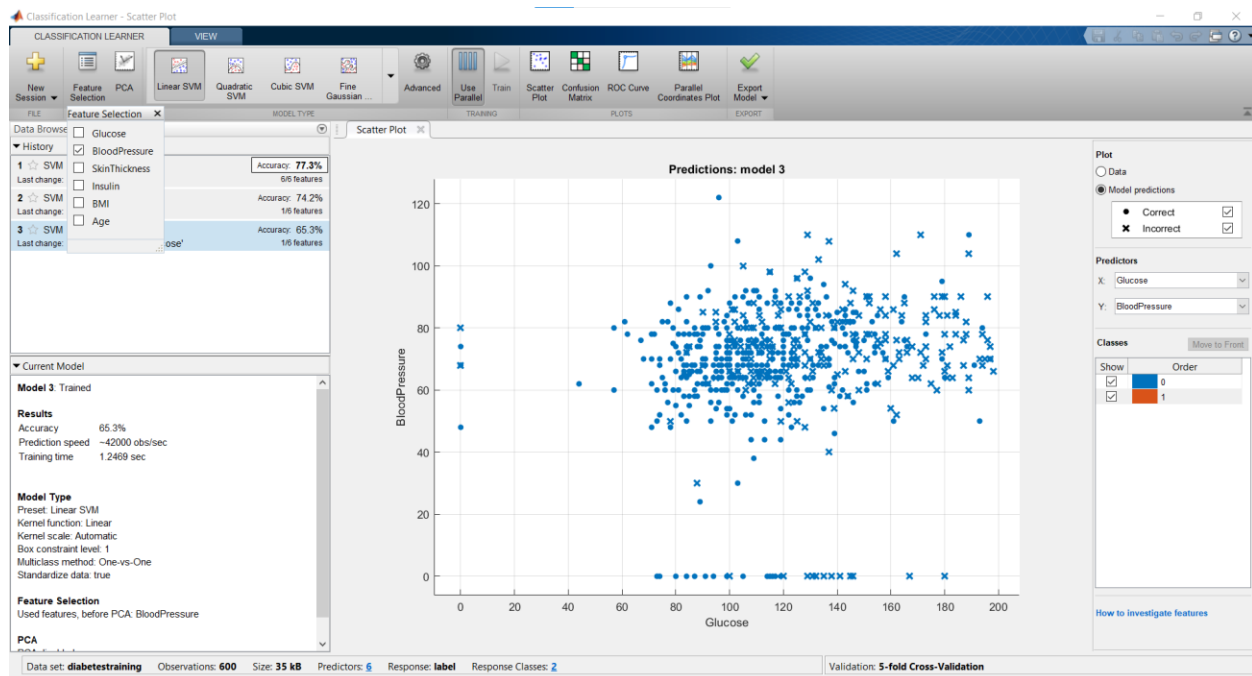
۱-۴



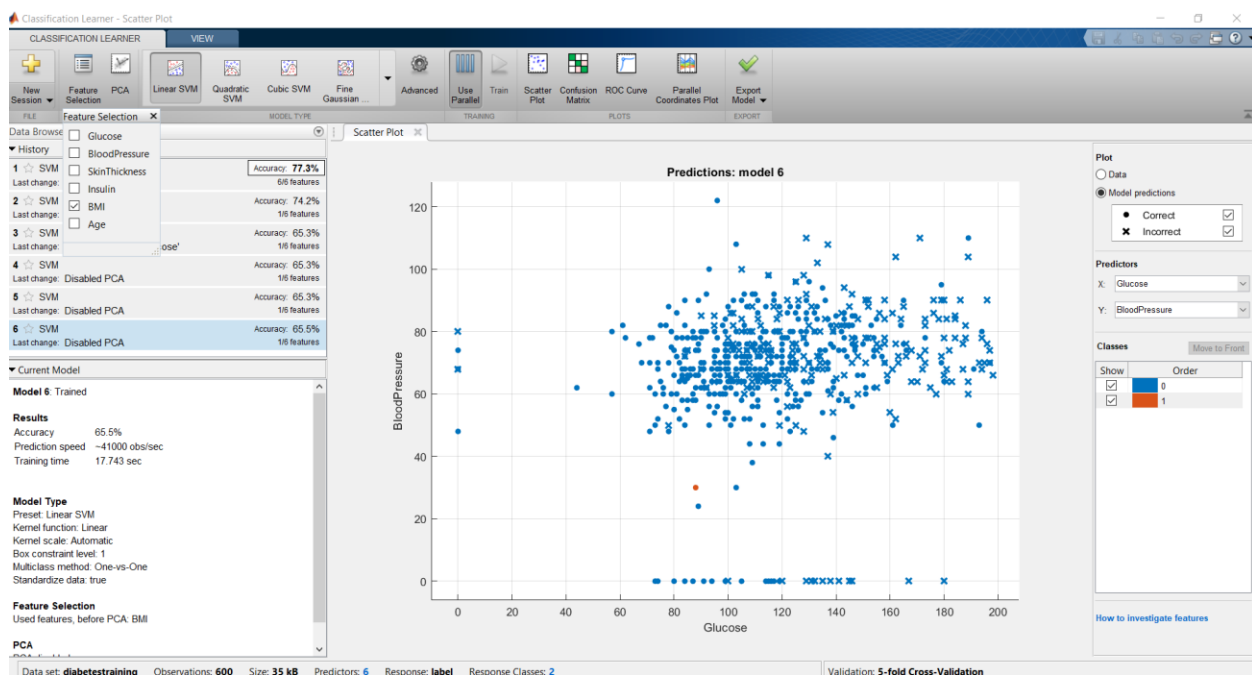
کل: ۷۷.۳ درصد



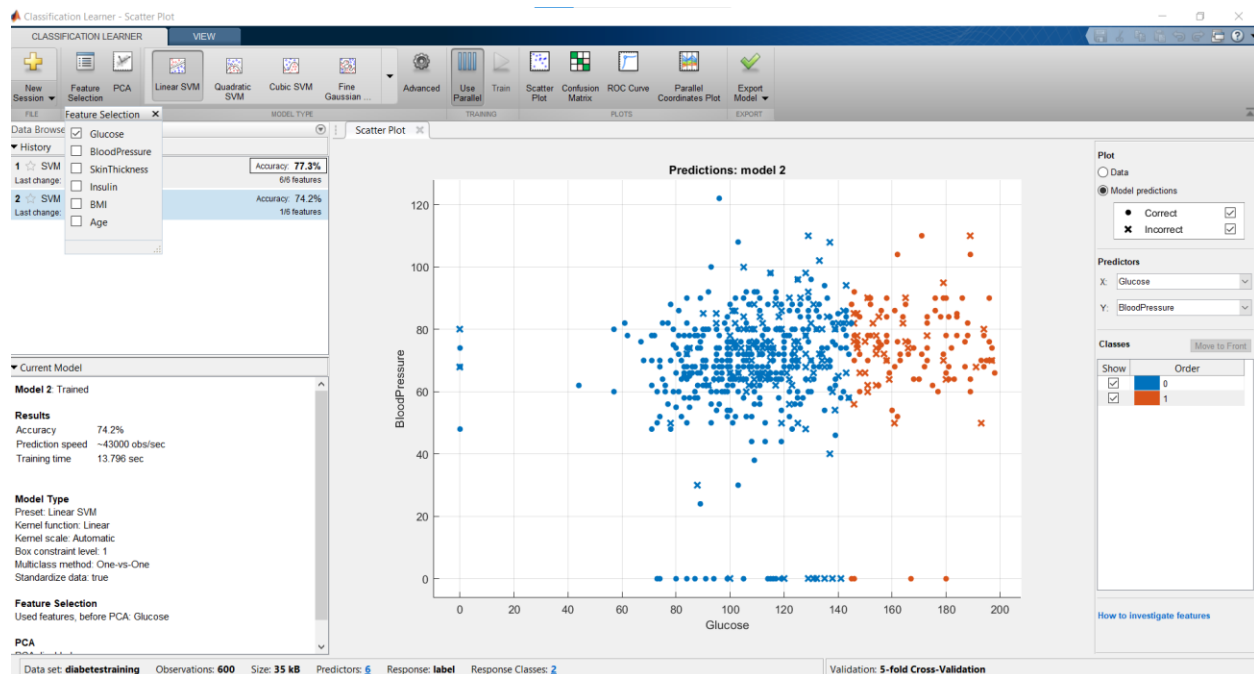
سن: ۶۵.۳ درصد



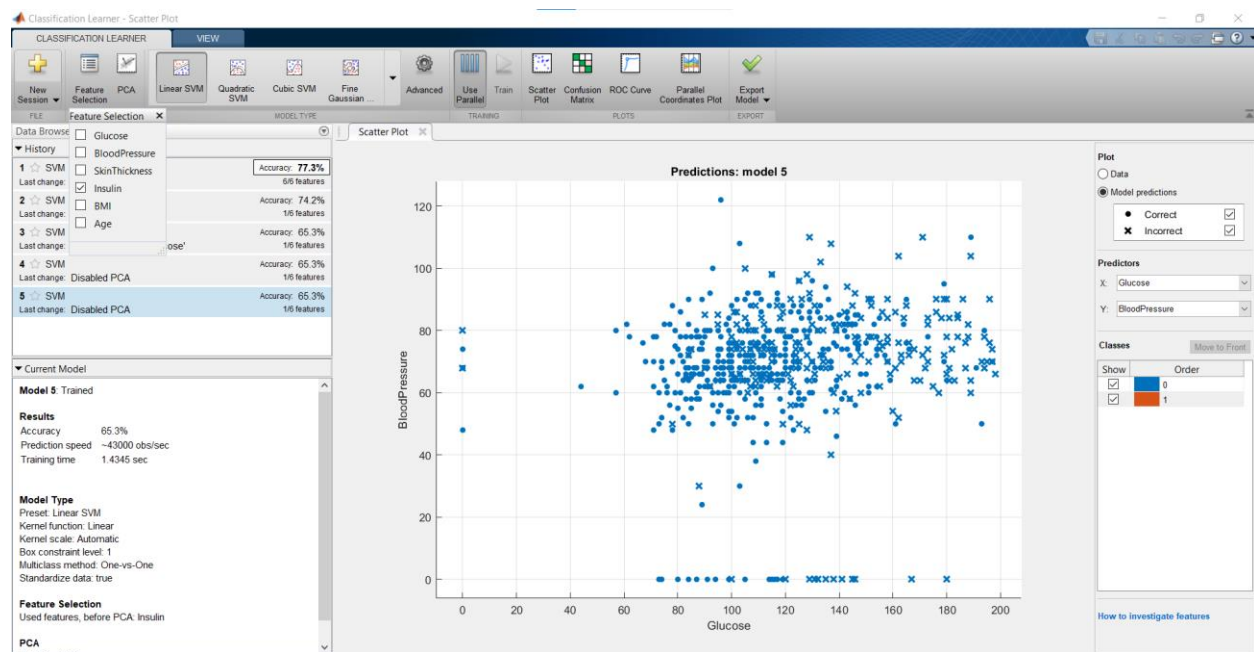
فشار خون: ۶۵.۳ درصد



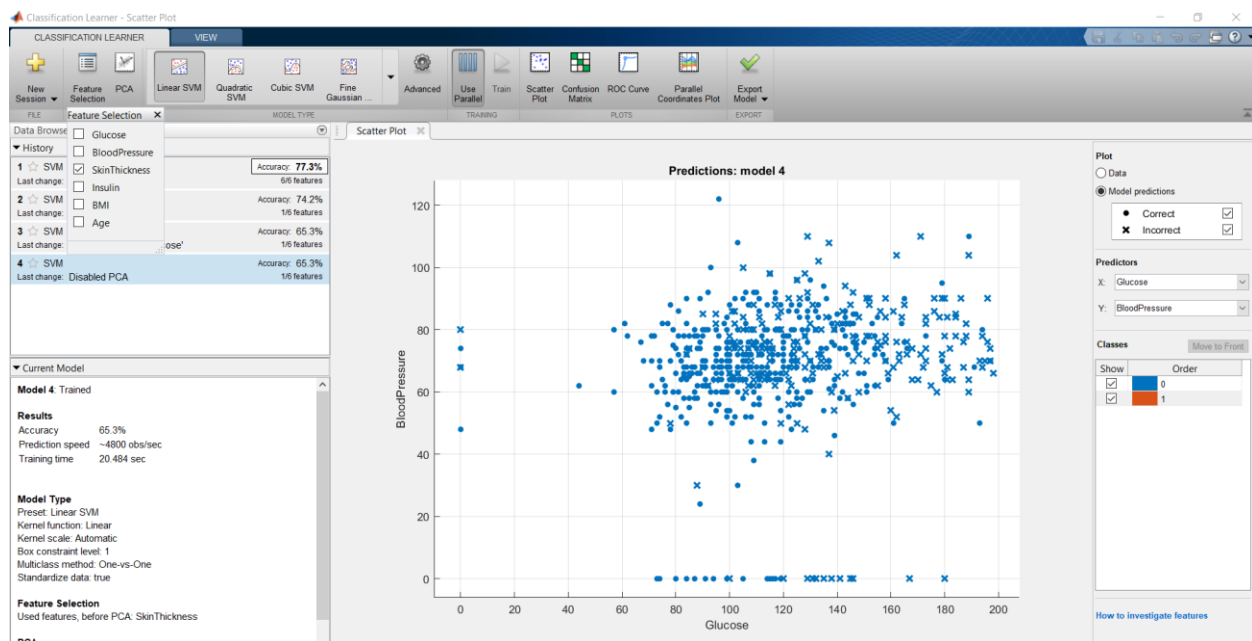
نسبت BMI: ۶۵.۵ درصد



گلوکز: ۷۴.۲ درصد



انسولین: ۶۵.۳ درصد



پوست کلفتی: ۶۵.۳ درصد

ویژگی میزان قند خون فرد بیش از همه به دیابتی بودن فرد ارتباط دارد.

۳-۴ و ۴-۴

در این قسمت، از ستون `label` که واقعیت است و از ستون `predictedLabel` که به کمک ماشین لرنینگ به دست آمده، ابتدا `diff` گرفته شده است تا تفاوت آن‌ها مشخص شود. سپس درصدگیری صورت گرفته تا `accuracy` و صحت آن به دست آید. این کار در بخش ۴.۴ نیز تکرار شده است.

```
Editor - p4.m
1 %%%%%%%%%%%%%%% part 4-3
2 table = readtable('diabetes-training.csv');
3 label=table.label
4 table = removevars(table, 'label');
5 predictedLabel=trainedModel.predictFcn(table)
6
7
8 diff = sum(abs(predictedLabel-label));
9 percentage= (1 - diff/length(label))*100;
10
11 %%%%%%%%%%%%%%% part 4-4
12 table = readtable('diabetes-validation.csv');
13 label=table.label
14 table = removevars(table, 'label');
15 predictedLabel=trainedModel.predictFcn(table)
16
17
18 diff = sum(abs(predictedLabel-label));
19 percentage= (1 - diff/length(label))*100
```

برای قسمت ۴-۳، عدد ۷۷.۵ درصد و برای قسمت ۴-۴، عدد ۷۸ درصد گزارش شد.