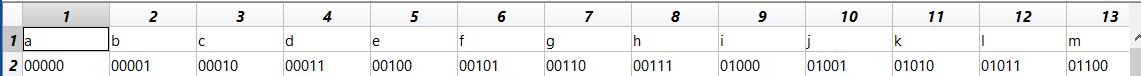
پروژه 4

سهیل حاجیان منش 810100119

بخش اول: تمرین1-1)

مشابه تمرین قبلی mapSet گفته شده را ایجاد میکنیم که قسمتی از آنرا در زیر قرار داده ام:



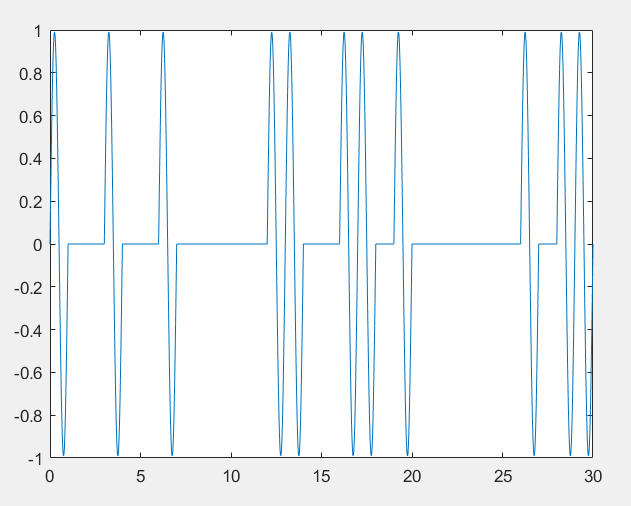
تمرین2-1)

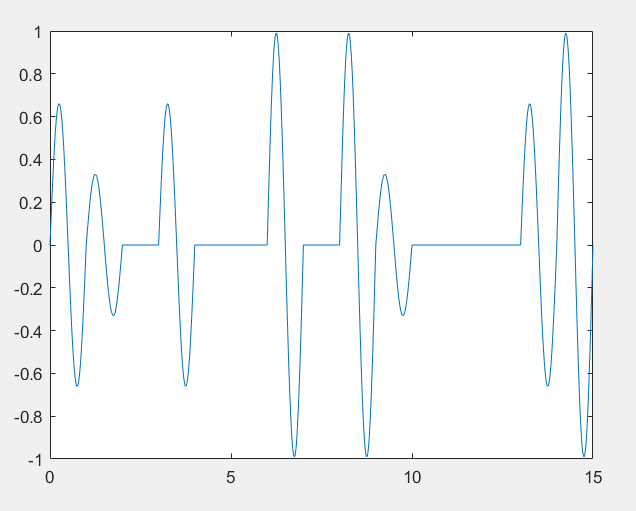
ابتدا از طریق mapSet به هرکاراکتر رشته باینری مرتبط با آنرا نسبت میدهیم و در انتهای یک رشته باینری که نشان دهنده پیام ورودی است میسازیم. سپس آرایه ای از ضراب میسازیم. برای اینکار با استفاده از تابع linspace بین 0 و 0.99 به تعداد speed^2 تا عدد میسازیم که همان ضرایب ما میباشند.

حال باید سیگنال نهایی را تشکیل دهیم. برای این کار رشته باینری ساخته شده را به بسته هایی با سایز speed تقسیم میکنیم. هر بسته نشان دهنده یک عدد باینری بین 0(000) و 7(111) میباشد.بترتیب بسته های را پیمایش میکنیم و با توجه به مقدار آن ضریب مربوط به آن را از آرایه ضرایب برداشته و در sin(2pi\*t) ضرب میکنیم و به سیگنال تولید شده تا اینجا اضافه میکنیم تا در نهایت سیگنال تکمیل شود.

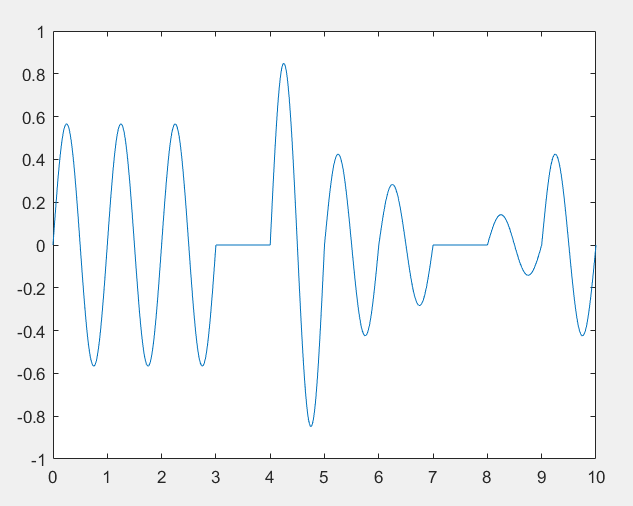
تمرین3-1)

خروجی این تابع به ازای ورودی signal و سرعت های 1و2و3 بترتیب بصورت زیر میباشد:

speed=1:

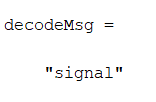


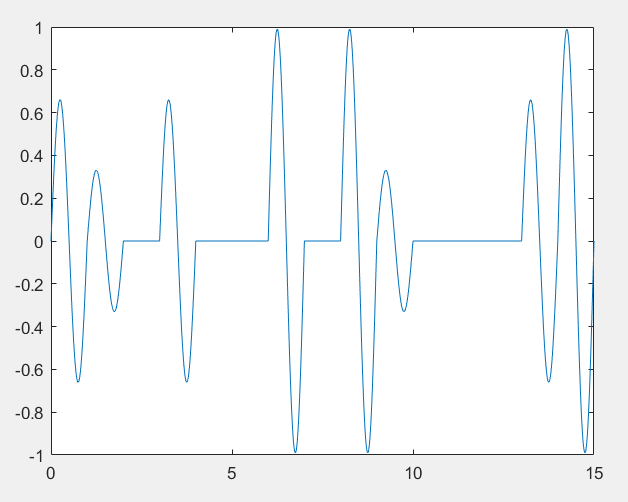
Speed=2:

Speed=3:

تمرین4-1)

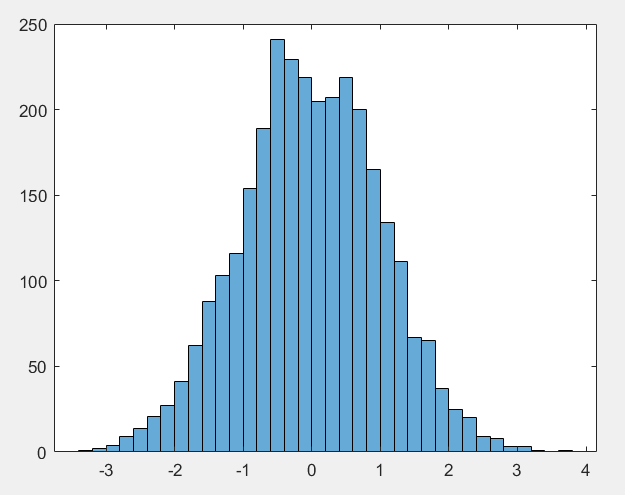
در ابتدا کل سیگنال را بخش بندی میکنیم که هر بخش شامل سیگنال مرتبط به یک ضریب است.سپس برای هر بخش به ازای تمام ثانیه ها با سیگنال sin(2pi\*t) کورولیشن میگیریم. در این مرحله ضرایب را بدست اورده ایم و کافیست برای اینکه ضرایب مقدار صیحیحی بگیرند آنها را در 2^speed-1 ضرب کنیم.حال خروجی ها برای مثال برای سرعت 3 همان 0و1و2و3و...و7 میباشند.سپس هرکدام از عدد های ضرایب را به ترتیب به مقدار باینریشان تبدیل میکنیم و در کنار هم قرار میدهیم تا یک رشته باینری ساخته شود در اینجا کافیست رشته باینری بدست آمده را پنج بیت پنج بیت دیکود کنیم تا کل پیام دیکود شود.

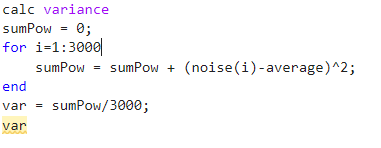
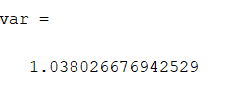
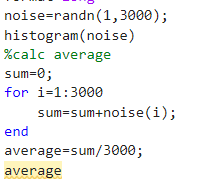
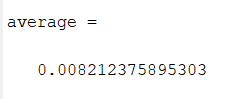
برای مثال به ازای ورودی سیگنال زیر و که سرعت آن سه است خروجی signal بدست می آید:



تمرین5-1)

نویز گوسی با میانگین صفر و واریانس یک را به کمک دستور گفته شده تولید میکنیم و گوسی بودن آن را با رسم نمودار هسیتوگرام زیر اثبات میکنیم. همچنین میانگین و واریانس آنرا نیز با زدن کد مربوطه بدست می آوریم تا ثابت کنیم مقادیرشان بترتیب صفر و یک است.

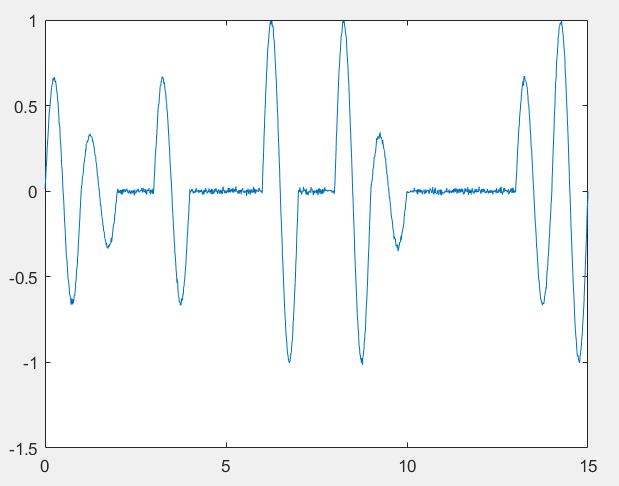
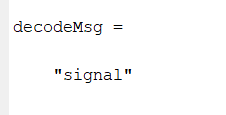




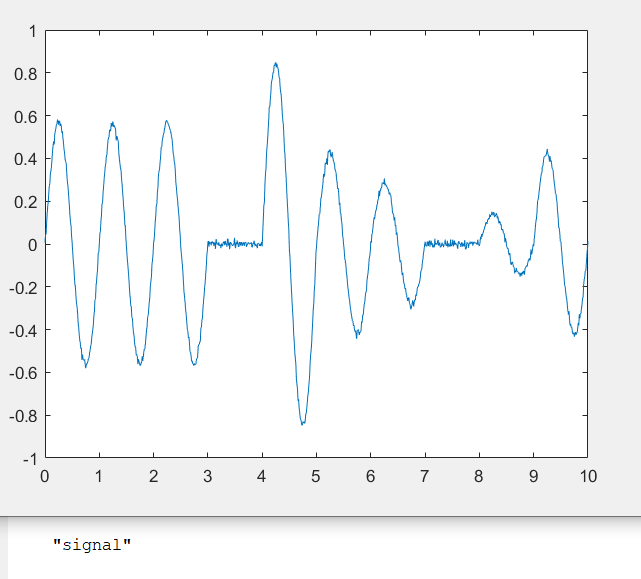
تمرین6-1)

وقتی به سیگنال کدگذاری شده سیگنال گوسی با واریانس 0.0001 را اضافه میکنیم شکل سیگنال بصورت زیر میشود. خروجی هم چنان درست است:

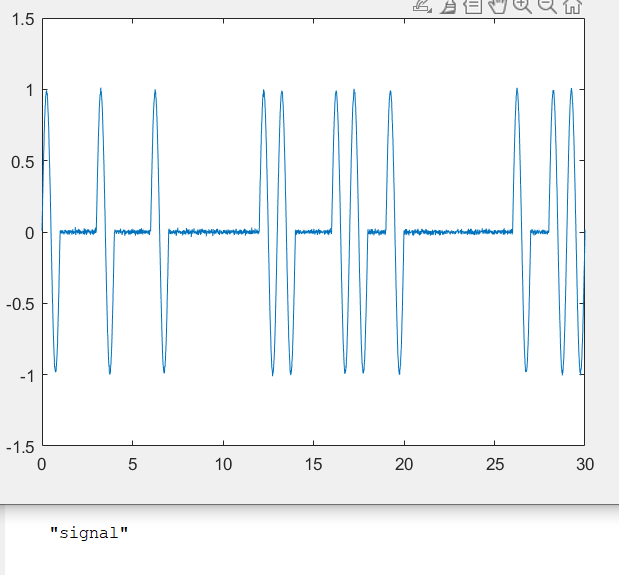
Bit rate=2



Bit rate=3:



Bit rate=1:

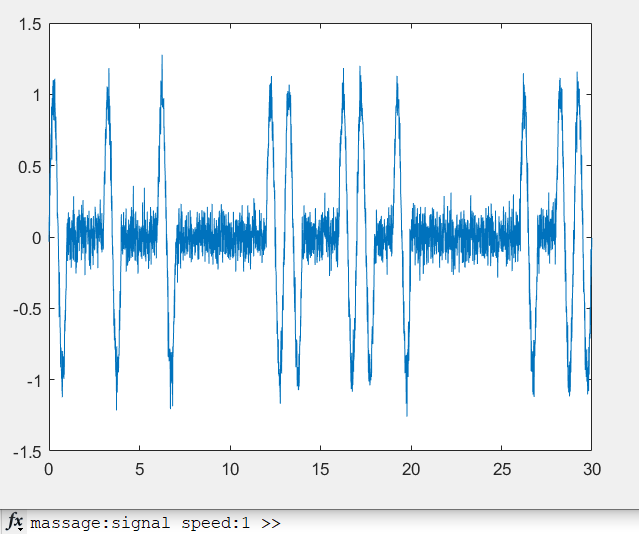


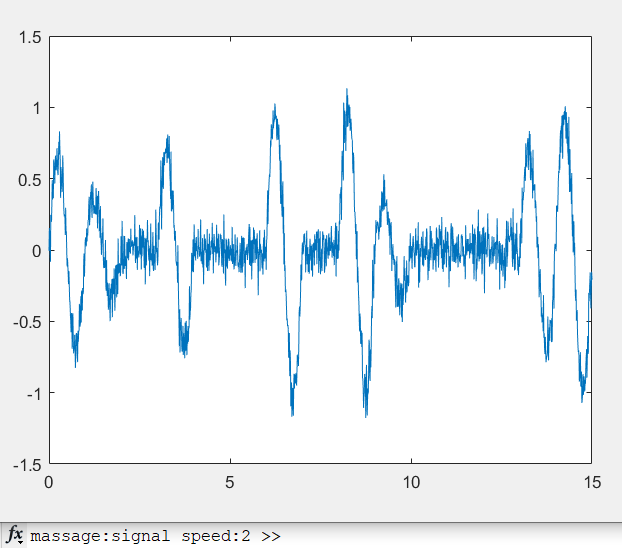
در هر سه سرعت ، پیام درست دیکود میشود.

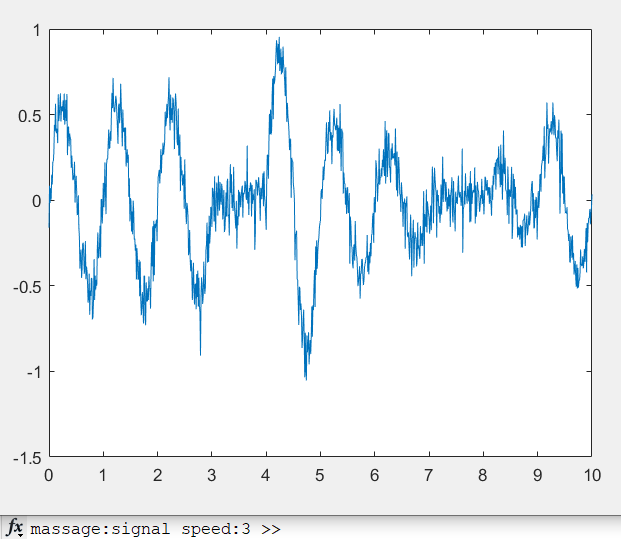
تمرین7-1)

از noise rate 0.1 شروع کرده و هرمرحله 0.1 به آن اضافه میکنیم.

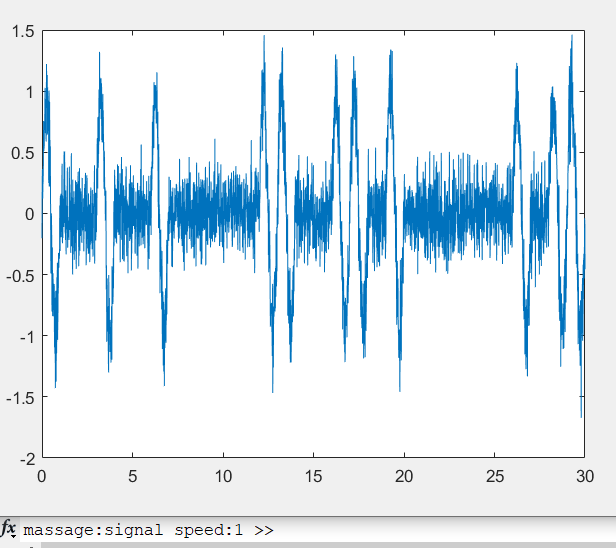
Noise rate=0.1:

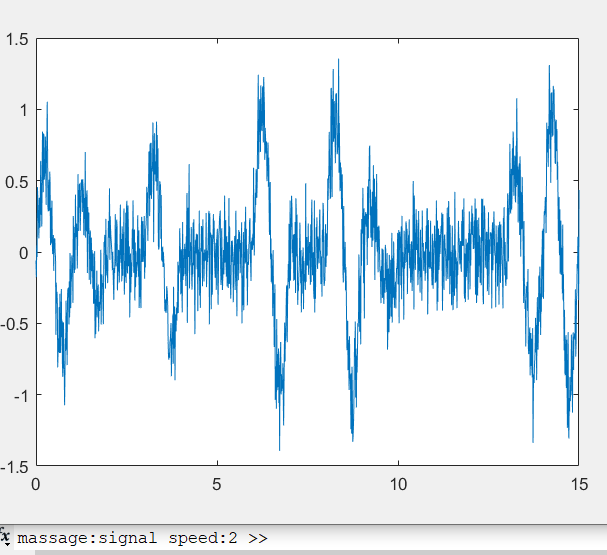


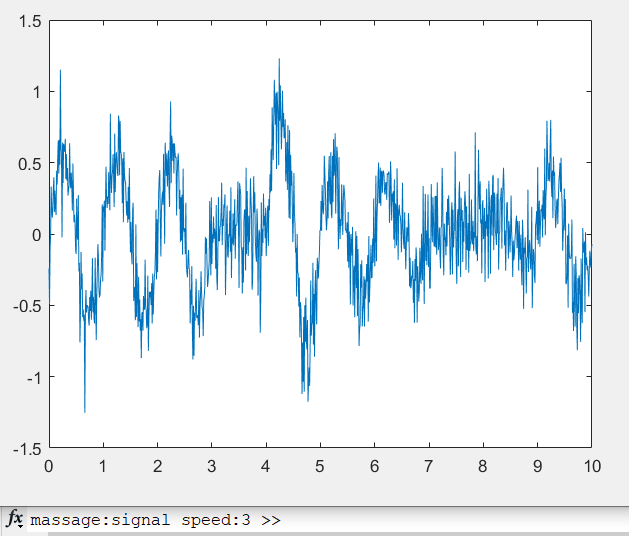




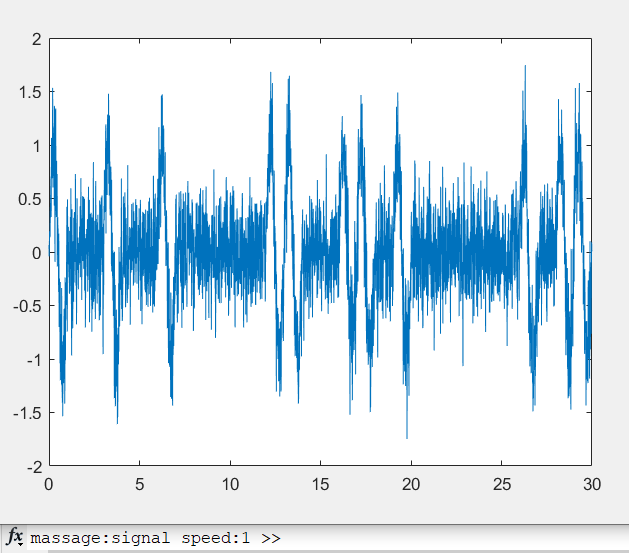
noise rate=0.3:

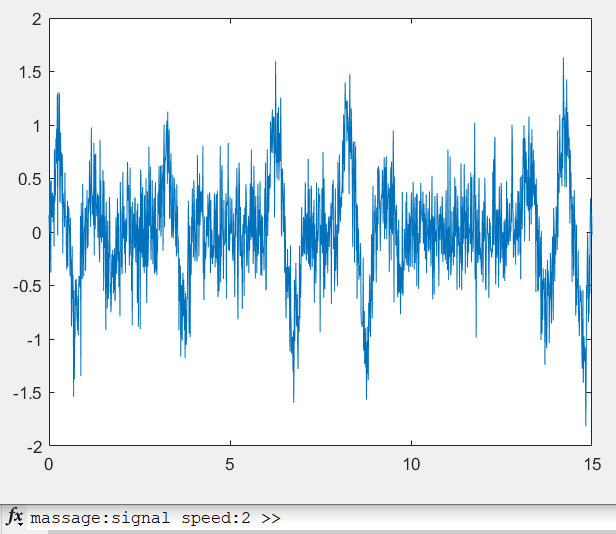


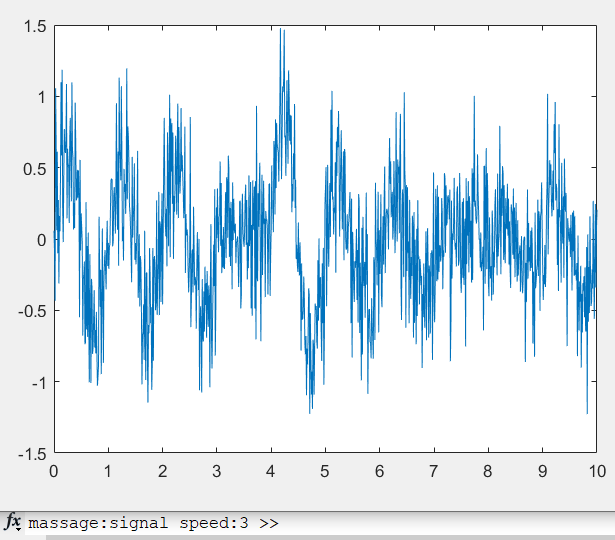




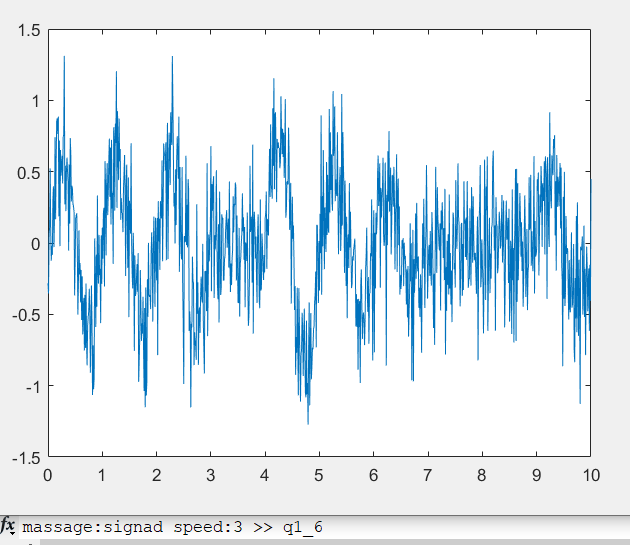
Noise rate=0.3:



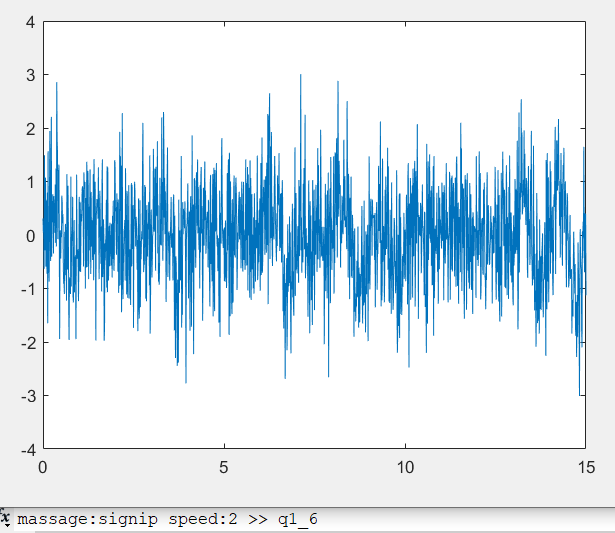




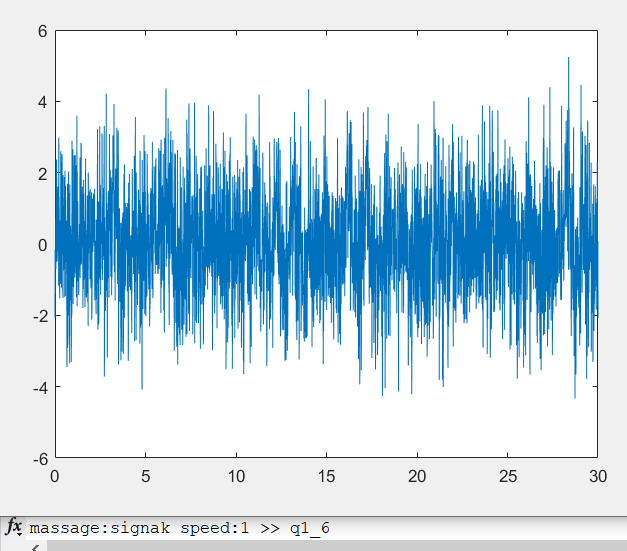
در noise bit=0.3 تنها برای speed=3 مقدار پیام دیکود شده اشتباه است:



به ازای noise rate= 0.8 برای speed=2 هم مقدار پیام دیکود شده غلط میشود:



در noise rate=1.4 هم در بعضی مواقع پیام دیکود شده برای سرعت 1 غلط است:



تمرین8-1)

برای bit rate=3 برابر 0.04 برای bit rate=2 برابر 0.49 و برای bit rate=1 برابر 1.69 بدست آمد.

تمرین9-1)

همانطور که بطور کامل در صورت پروژه گفته شده است با افزایش قدرت سیگنال و دامنه سیگنال میتوان bit rate را نسبت به نویز مقاوم تر کرد.

تمرین10-1)

از لحاظ تئوری اگه نویزی نداشته باشیم می توانی م تا هر چه قدر دلمان م ی خواهد بیت ریت را تا ب ی نهایت افزایش داد ولی دستگاه هایمان در آن حد برای تول ید و تشخی ص سیگنال از هم دقیق نیستند و اگر بر فرض همان چی زی که می فرست یم را عینا و بدون کم و کاست دریافت کنیم. در کامیپیوتر برای هر عدد اعشاری مقداری بیت جهت ذخی ره سازی آن استفاده می کنیم که اگر عددمان اعشار داشته باشد و از حدی بیشتر شود دو عدد در داخل کامپیوتر به یک صورت ذخیره م ی شوند ولی در واقعیت در مقداری بسیار جزیی با هم تفاوت دارند که این ناشی از دیجیتال بودن دستگاه های ما می باشد در حالی که در فضا موج آنالوگ می باشد.

تمرین11-1)

اگر بدون تغییر دامنه صرفا در محاسباتمان سیگنال را 5 برابر کنیم با این که threshold هایمان را 5 برابر می کنیم برای مثال اگر threshold هایمان .1 .1، باشند پس از اعمال این تغییر .5 ، .5 تا می شوند که اندکی کار ما را برای decoding راحت تر میکند ولی مشکل اینجاست مقدار noise ها را نیز 5 برابر می کن یم و عمال تغییر خاصی ایجاد نمی شود و تغییر خاصی در مقاومت به نویز ایجاد نمی کند و کمکی به ما نمی کند.

تمرین12-1)

سرعت اینترنت خانگی در بیشترین حالت 16مگابایت بر ثانیه و در حالت معمولی 8 مگابایت بر ثانیه است که برابر 8388608 بیت بر ثانیه است که تفاوت قابل توجهی با سرعت ارسال اطلاعات در پروژه ما که 1 تا 3 بیت بر ثانیه بود دارد.

بخش دوم: تمرین2-1)

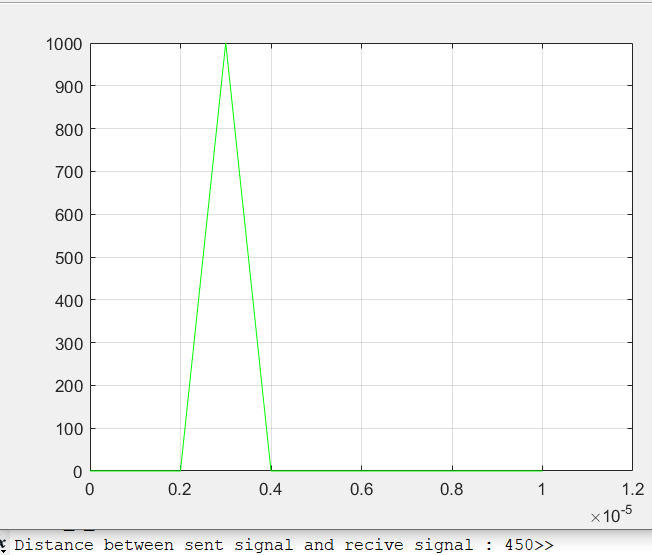
ابتدا محور افقی(t) را طبق مقادیر گفته شده در سوال معین می کنیم. سپس با تابع zeros ماتریسی به اندازه t ایجاد میکنیم و در نهایت به اندازه گفته شده پالس با دامنه یک را در ماتریس zeros قرار میدهیم.بعد از این مراحل سیگنال ورودی درست شده است.

برای سیگنال برگشتی ابتدا از طریق فرمول مقدار td را بدست می آوریم و سپس مشابه قسمت قبل عمل میکنیم. با این تفاوت که سیگنال پالس را باید از td وارد ماتریس zeros کنیم.



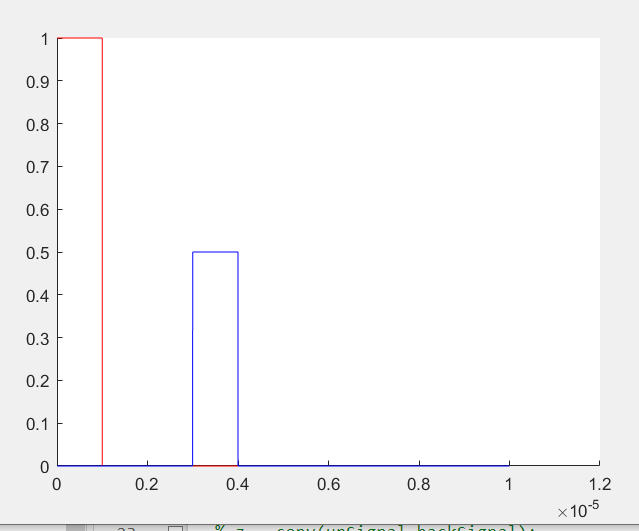
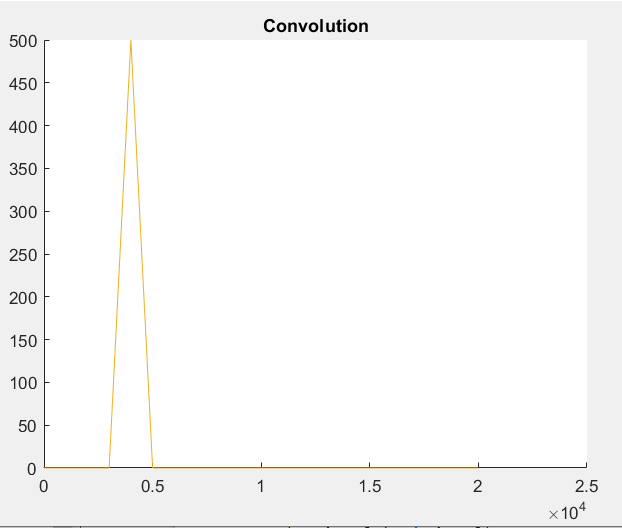
برای حالت برعکس هم از کورولیشن گیری و template matching استفاده میکنیم و سیگنال برگشتی را با تمام سیگنال های ممکن برای پالس کورولیشن میگیریم و در انتها ایندکس ماکزیمم کورولیشن ها را بعنوان جوابtd برمیگردانیم که با استفاده از فرمول میتوان به R هم رسید.

نمودار مقادیر کورولیشن ها و همچنین مقدار R خروجی بدست آمده که برابر 450 شده است:

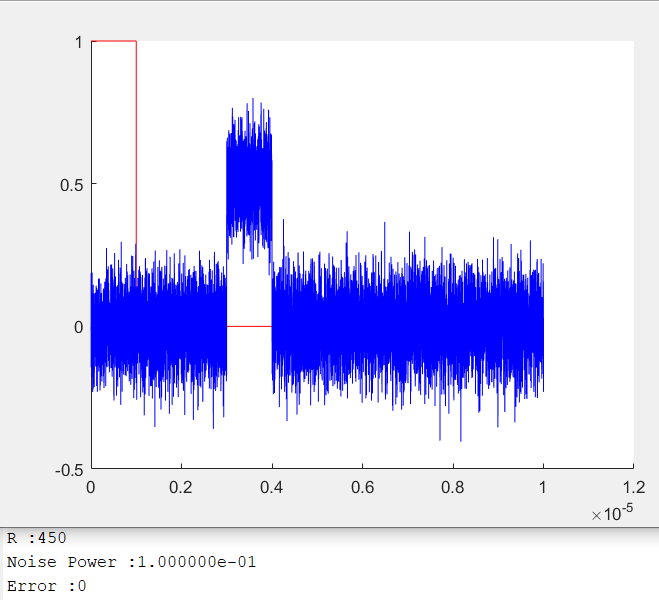


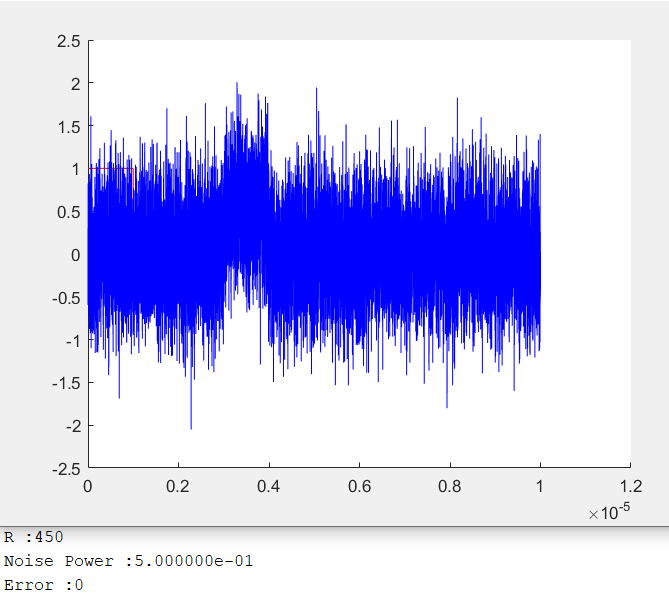
تمرین 2-2)

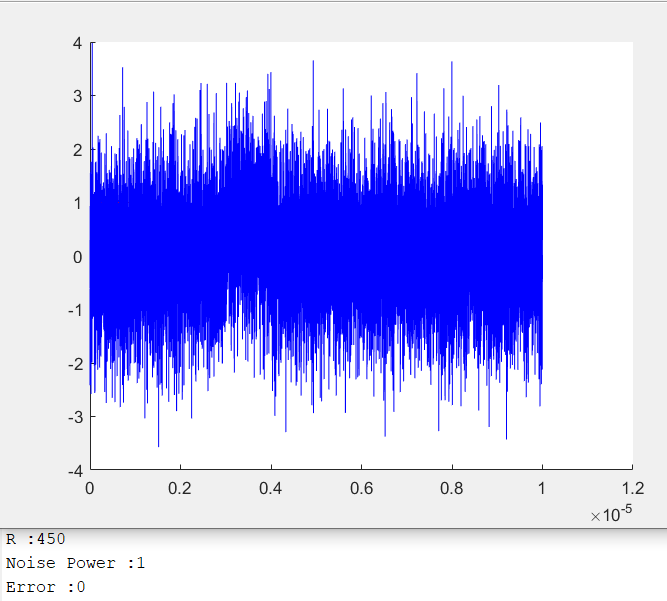
پالس را از چپ به راست تکان میدهیم تا هر دو سیگنال ورودی و دریافتی را داشته باشیم.سپس این دو سیگنال را به تابع گفته شده در صورت سوال میدهیم. تابع conv یکی از سیگنال ها را قرینه می کند و روی سیگنال دیگر حرکت می دهد و اندازه خروجی نهایی اش دو برابر سیگنال های ورودی آن است. البته برای این که مساحت را دقیق حساب کرده باشد باید آن را ضرب در دلتاx بازه کنیم. با پیدا کردن ماکسیمم آن نیز می توان فاصله مان از جسم را تشخیص بدیم.

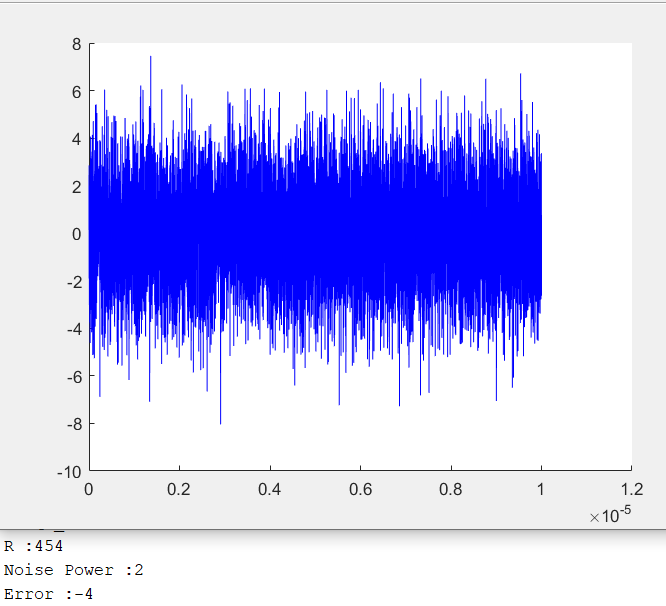


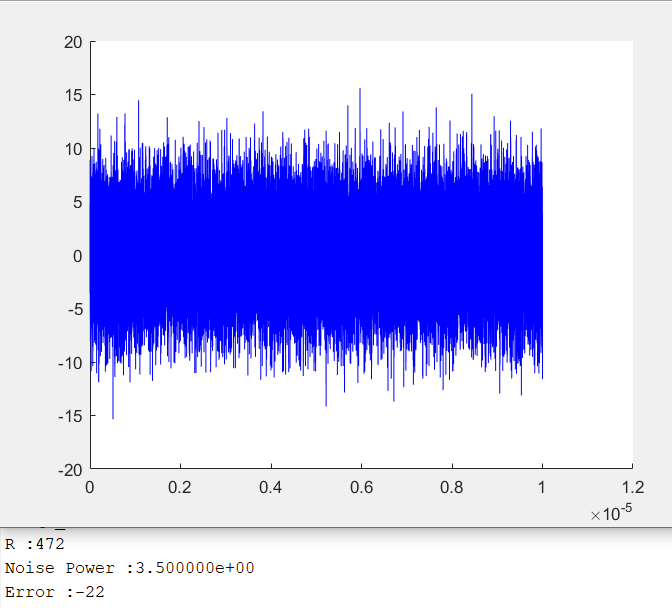
تمرین 3-2)









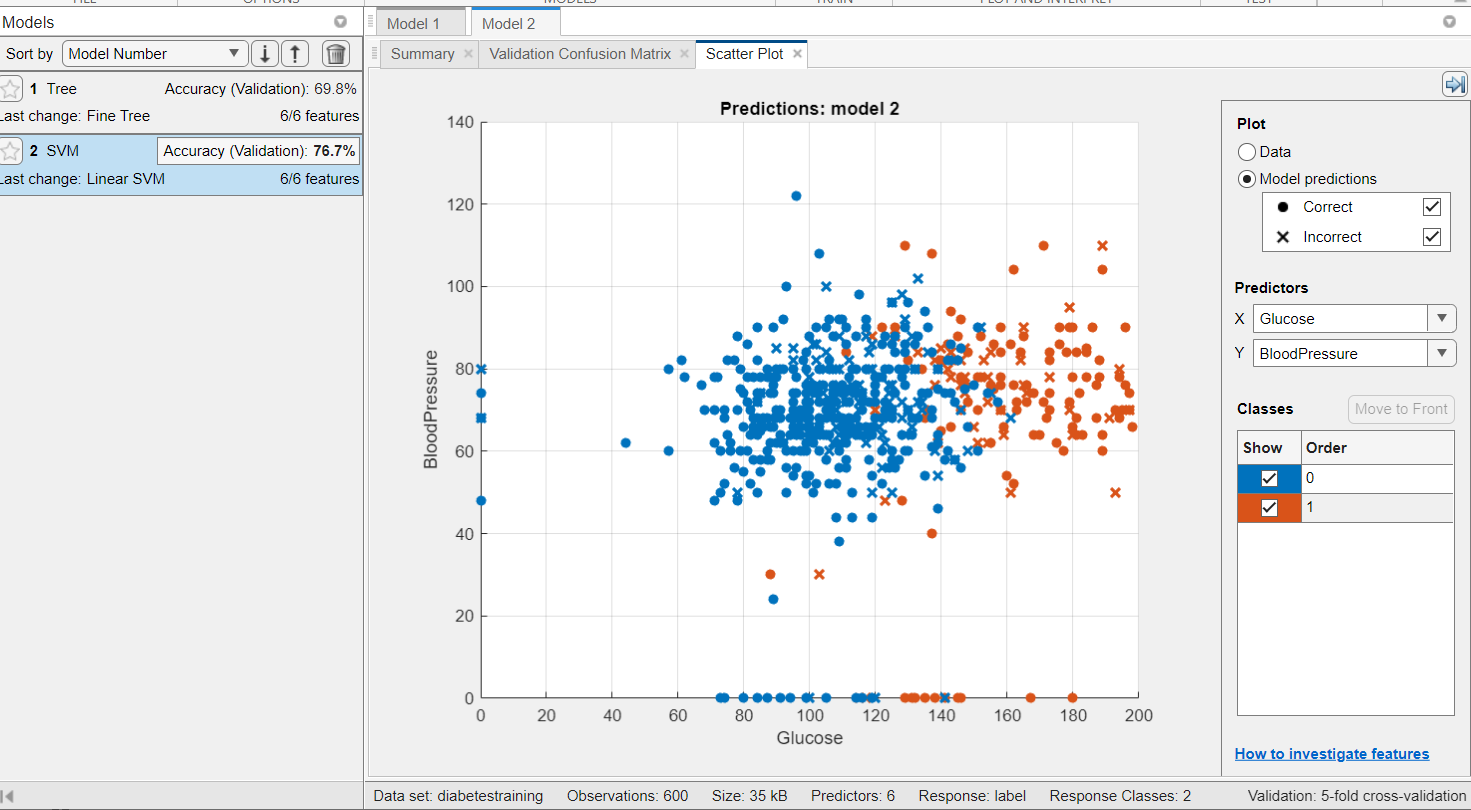


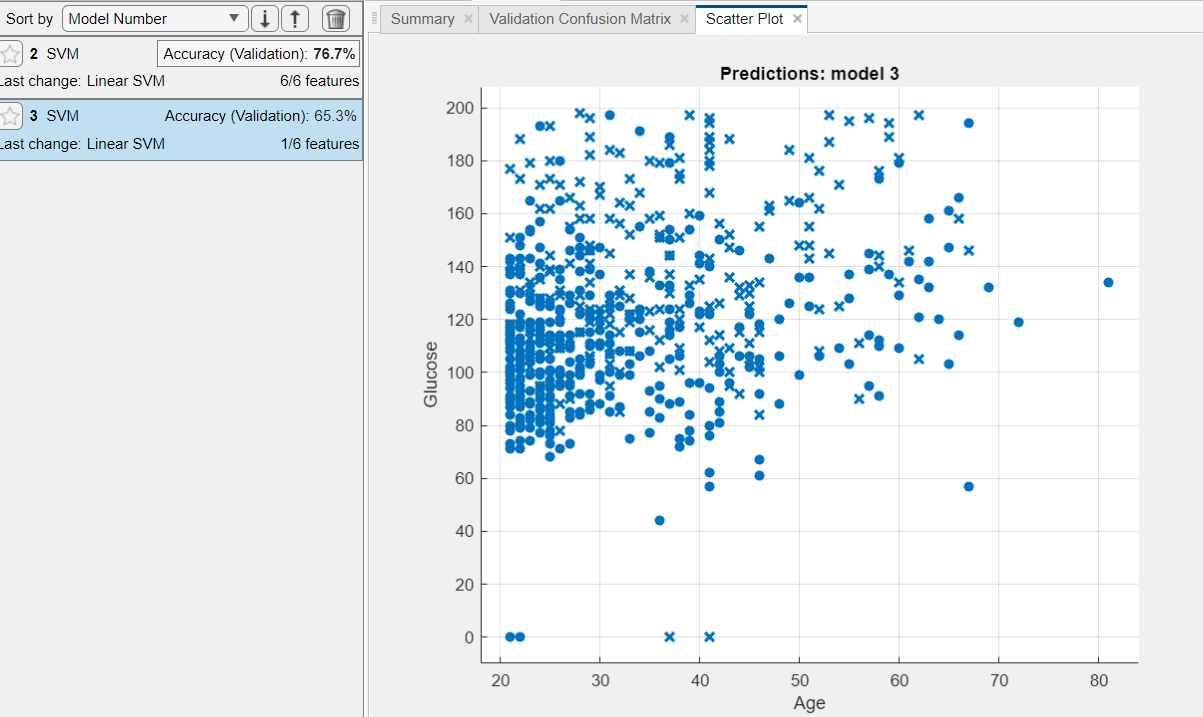
همانطور که در تصاویر دیدید با زیاد شدن نویز خطای تشخیص هم به مرور زیاد میشود و تقریبا میتوان گفت از حدود قدرت نویز 3.5 به بعد تشخیص R ممکن است با احتمال بالایی با خطا مواجه شود.

بخش سوم:

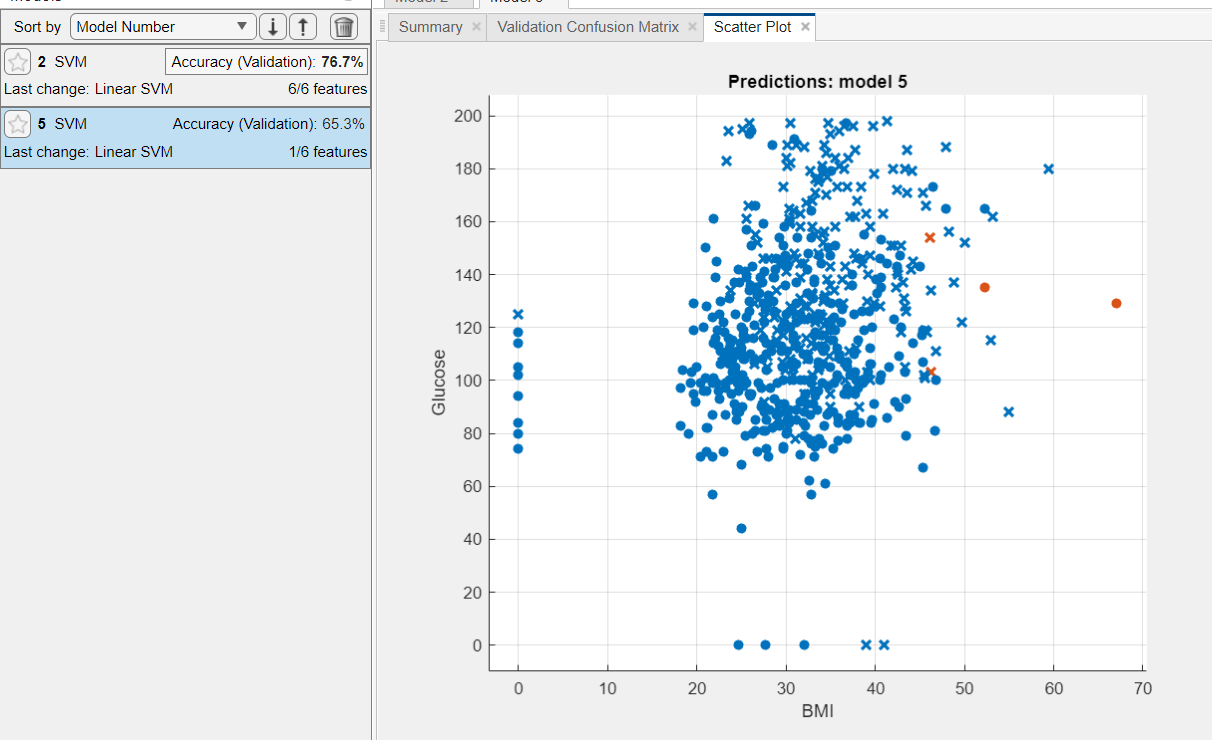
تمامی فایل های این بخش در پوشه part3 قرار دارد.

ابتدا با دستور audioread هر فایل را خواندیم و سیگنال مربوط به آنرا به همراه نام آن در مپ ست ذخیره کردیم و سپس در تابع calling\_customer سیگنال صدای نهایی را بوجود می آوریم.در این تابع ابتدا سیگنال مربوط به صدای شماره را ایجاد میکنیم و سپس با توجه به این که عدد شماره مکمل است یا خیر سیگنال مربوط به آنرا اضافه میکنیم و سپس سیگنال به باجه و درانتها سیگنال مربوط به شماره باجه را اضافه میکنیم. با استفاده از دستور sound هم میتوانیم صدای تولید شده را گوش دهیم.

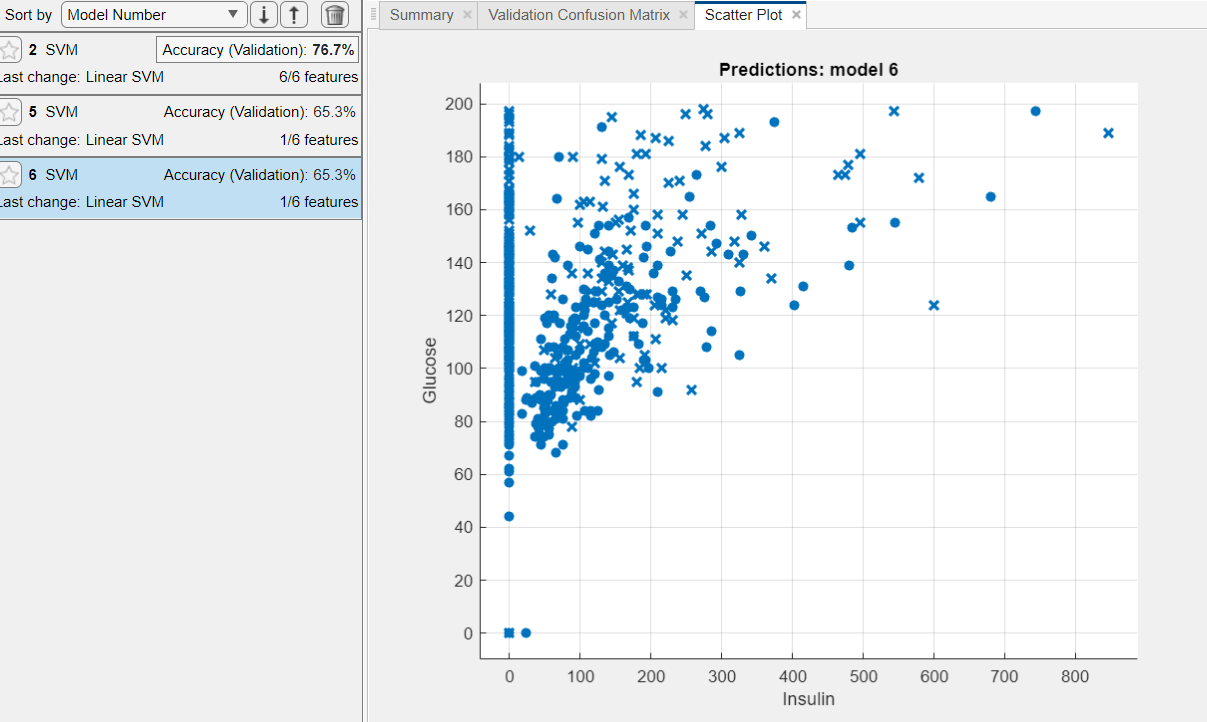
بخش چهارم: تمرین4-1)بعد از import کردن داده ها در برنامه با استفاده از مراحل گفته شده دقت بدست آمده در الگوریتم linear SVM برابر 76.7% گزارش شده است.

تمرین4-2) برای age دقت 65.3% بدست آمد.

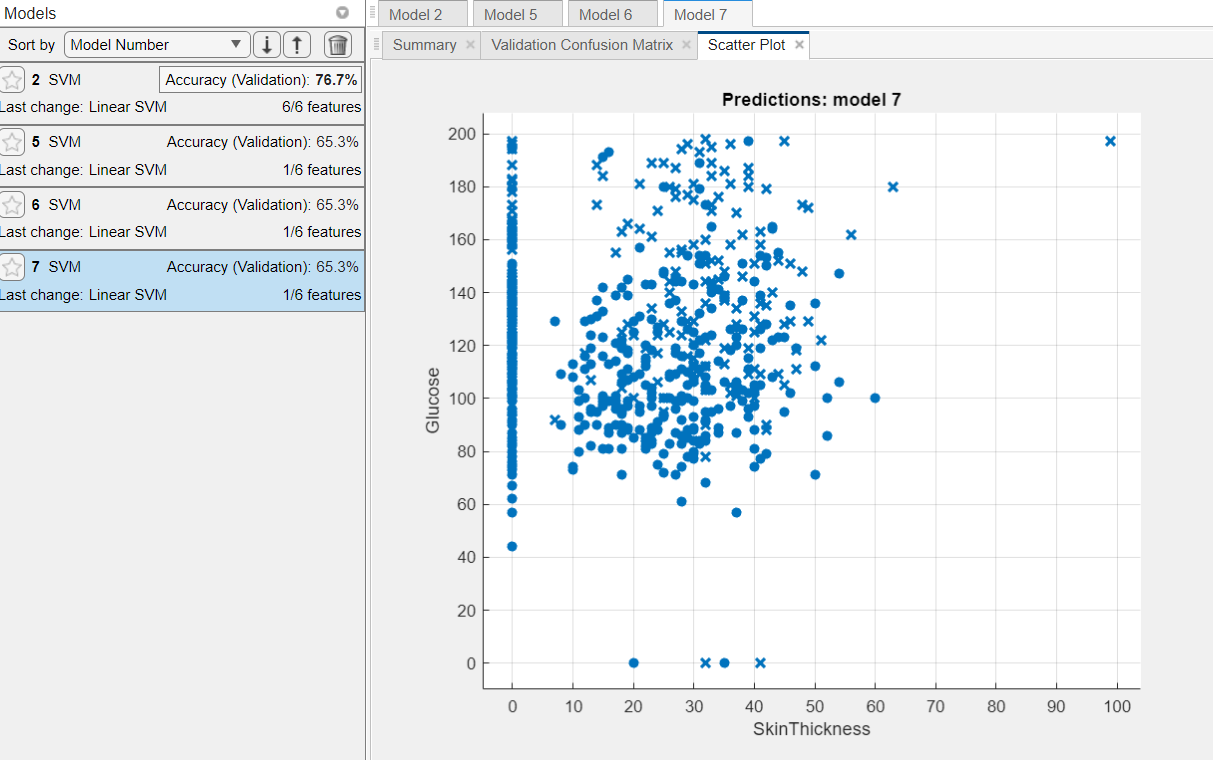
Bmi:65.3%



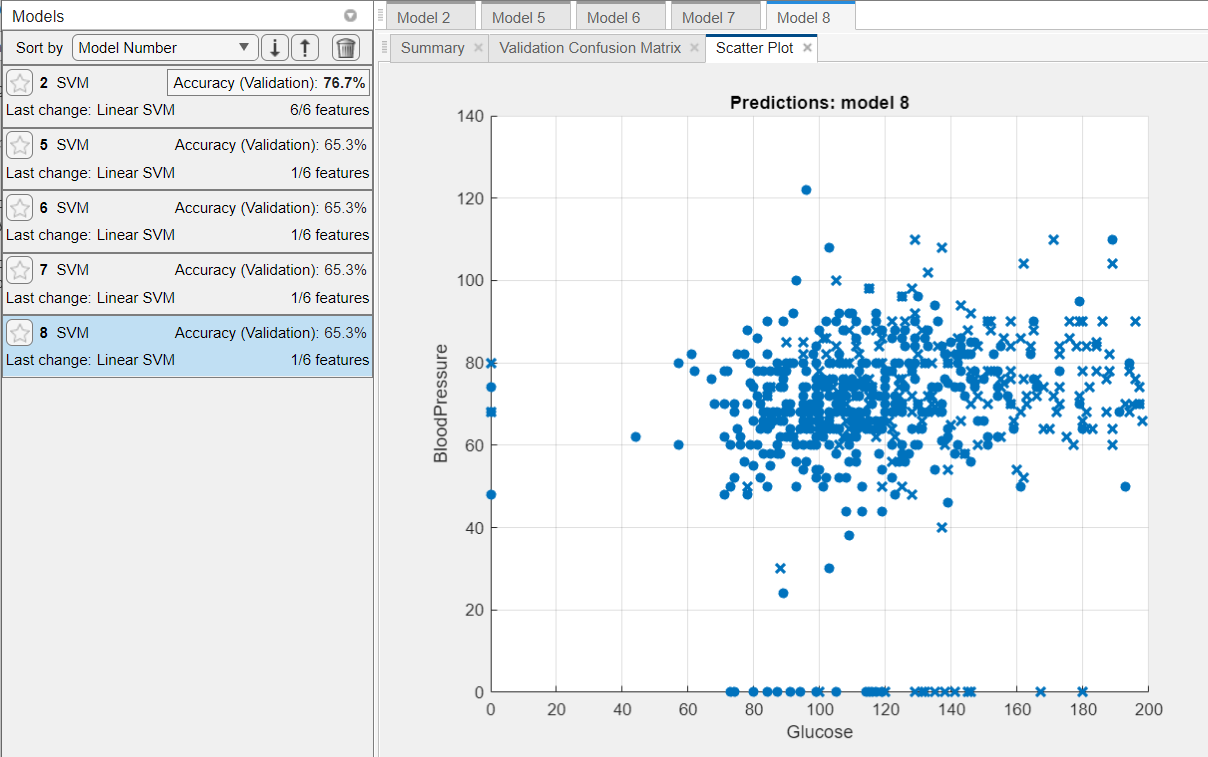
Insulin:65.3%



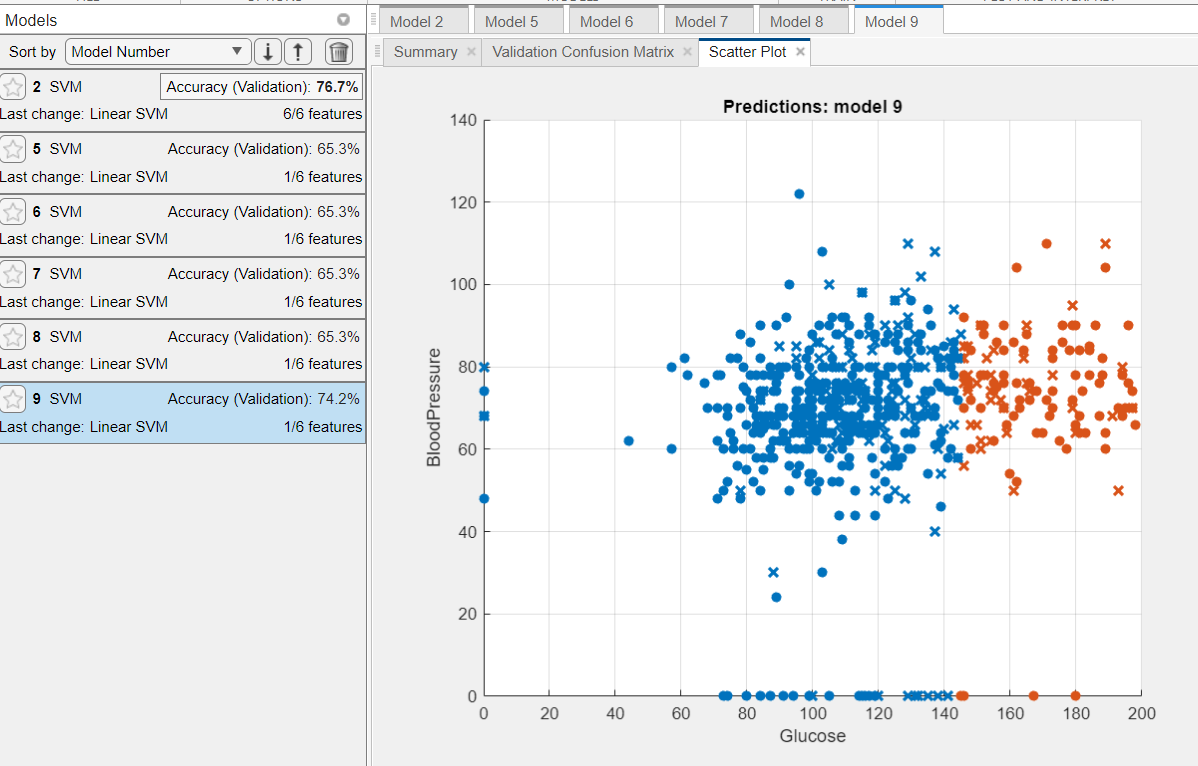
Skin thickness:65.3%



Blood pressure:65.3%



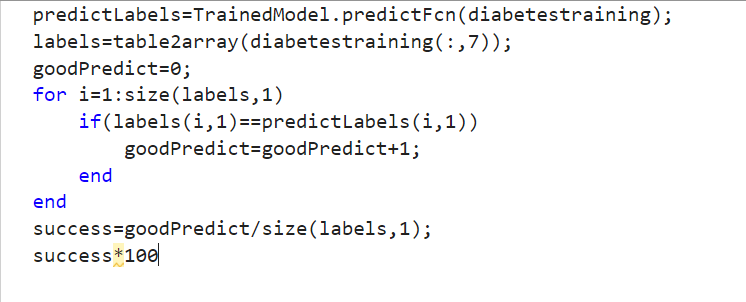
Glucose:74.2%

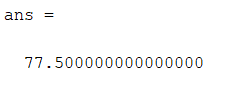


همانطور که از دقت های بدست گلوکز نسبت به باقی فیوچر ها با دقت بیشتر میتواند داده هارا تخمین بزند.

تمرین4-3)

ابتدا با استفاده از predictFcn برچسب داده ها را با استفاده از مدل آموزش دیده تخمین میزنیم.سپس در مرحله بعد با برچسب های خود مدل مقایسه میکنیم تا میزان دقت در تخمین زده شده بدست آید.





دقت بدست آمده همانطور که مشاهده میکنید برابر با 77.5 درصد درآمد که تقریبا با خطای کمی برابر دقت قسمت 4-1 است.

تمرین4-4)

مشابه قسمت قبلی عمل میکنیم و نتیجه درست میگیریم و دقت 78 درصد بدست می اید که نشان میدهد مدل آموزش دیده بر روی افراد جدید هم درست عمل میکند و دقت تخمین مشابه دقت فاز تست میباشد.

