به نام خدا

امیرعلی شهریاری (۸۱۰۱۰۰۱۷۳)

CA 4

:۱

در این بخش میخواهیم حروف و علایمی که میخواهیم برای ارسال پیاممان از آن استفاده کنیم را به صورت باینری در آورده و مپ ست آنرا تشکیل دهیم که یک سلولی به ابعاد ۲*۳۲ تشکیل خواهد شد.

```
function Mapset = CreateMapset()
 Mapset = cell(2,32);


    alpha = char(97:122);
    for i = 1:26
        Mapset(1,i) = cellstr(alpha(i));
    end
    Mapset(1,27)=cellstr({' '});
    Mapset(1,28)=cellstr('.');
    Mapset(1,29)=cellstr(',');
    Mapset(1,30)=cellstr('!');
    Mapset(1,31)=cellstr(':');
    Mapset(1,32)=cellstr('"');


    for i = 1:32
        Mapset(2,i) = cellstr(dec2bin(i-1 , 5));
    end
end
```

که یک cell ایجاد شده که در سطر نخست آن حروف الفبا و علامت ها و در سطر دوم اعداد باینری آنها ، به آنها نسبت داده شده است.

2x32 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
2	00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110

:۱-۲

در این بخش میخواهیم به کمک تابع coding_amp ورودی پیام مد نظر ما با توجه به مپ ستی که درست کردیم و سرعت ارسال پیام را به عنوان ورودی دریافت نماید و پیام خروجی کد گذاری شده را به ما نمایش دهد.

```
function coding_amp = amp_coding(ClearText, rate)
Mapset = CreateMapset();

    BinStr = '';
    for i = 1:strlength(ClearText)
        index = find(strcmp(Mapset(1,:), ClearText(i)));
        BinStr = strjoin([BinStr , Mapset(2,index)],'');
    end

    coding_amp=[];
    i=0;
    while i < strlength(BinStr)
        i = i+rate;
        temp = BinStr((i-rate+1):i);
        factor = (bin2dec(temp))./(2.^rate - 1);

        x = (i-1) : 0.01 : (i-0.01);
        y = factor.*sin(2*pi.*x);
        coding_amp = [coding_amp , y];
    end
end
```

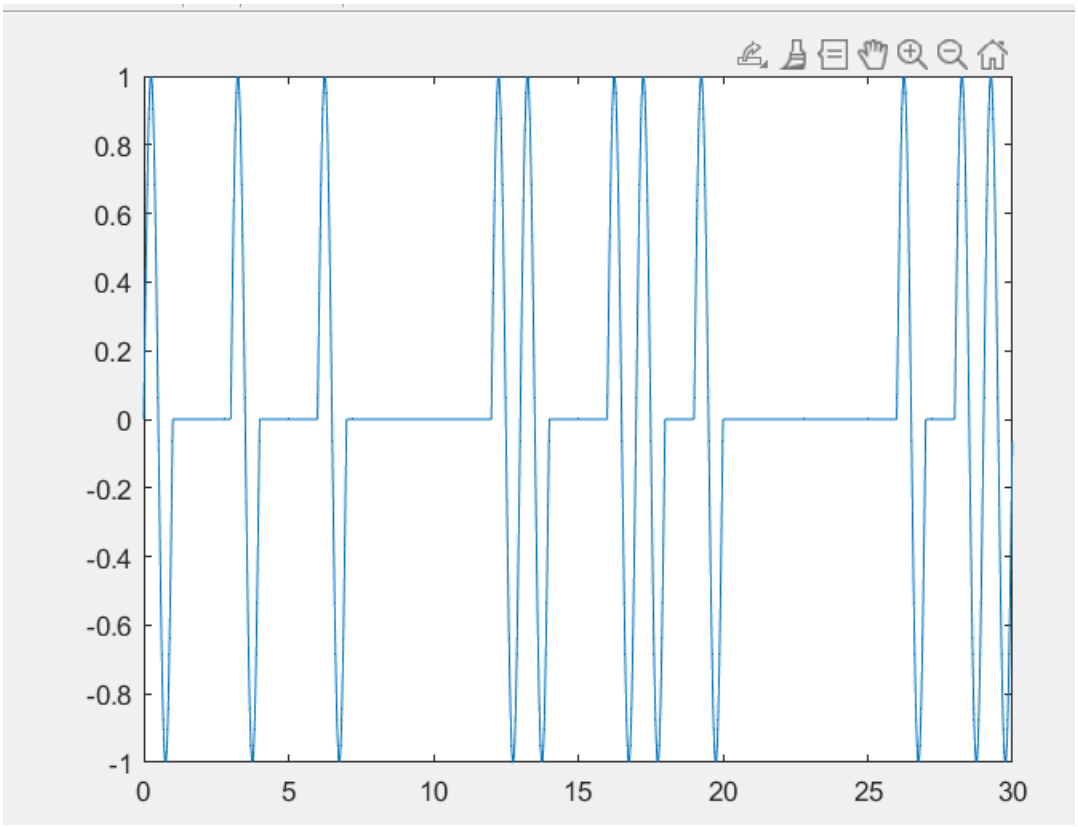
همانگونه که در کد مشخص است در نخستین حلقه رشته های متناظر با هر یک از کارکتر های رشته ورودی را یافته و به ترتیب قرار میدهیم. سپس با توجه به rateای که پیشتر به عنوان ورودی تابع به آن دادیم ، در هر اجرای حلقه تعدادی از بیت های رشته بدست آمده را یافته و سیگنال متناظر را در coding_amp ذخیره می کنیم .

۱-۳:

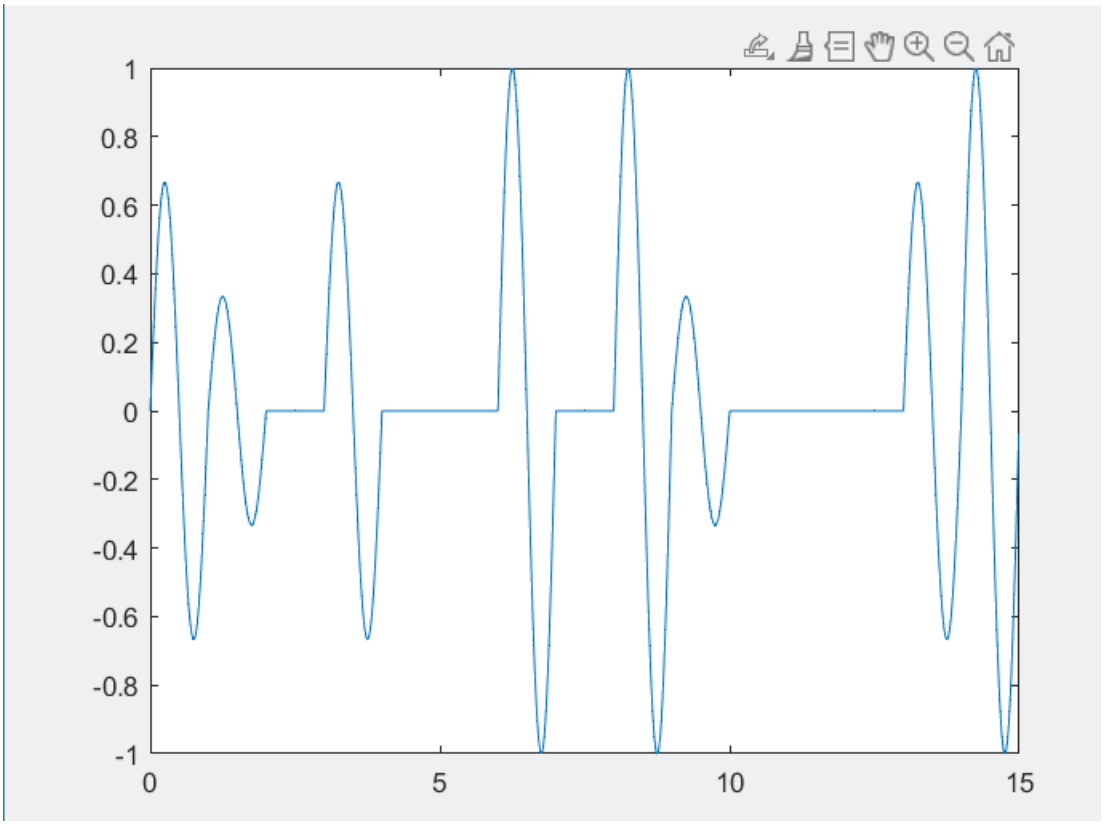
اکنون میخواهیم به کمک تابعی که در بخش پیشین نوشتیم ، کلمه signal را با سه سرعت رمز نگاری کنیم.

حال به کمک p1 rate را بنا به سه عدد خواسته شده تنظیم کرده و شکل موج را مشاهده می کنیم.

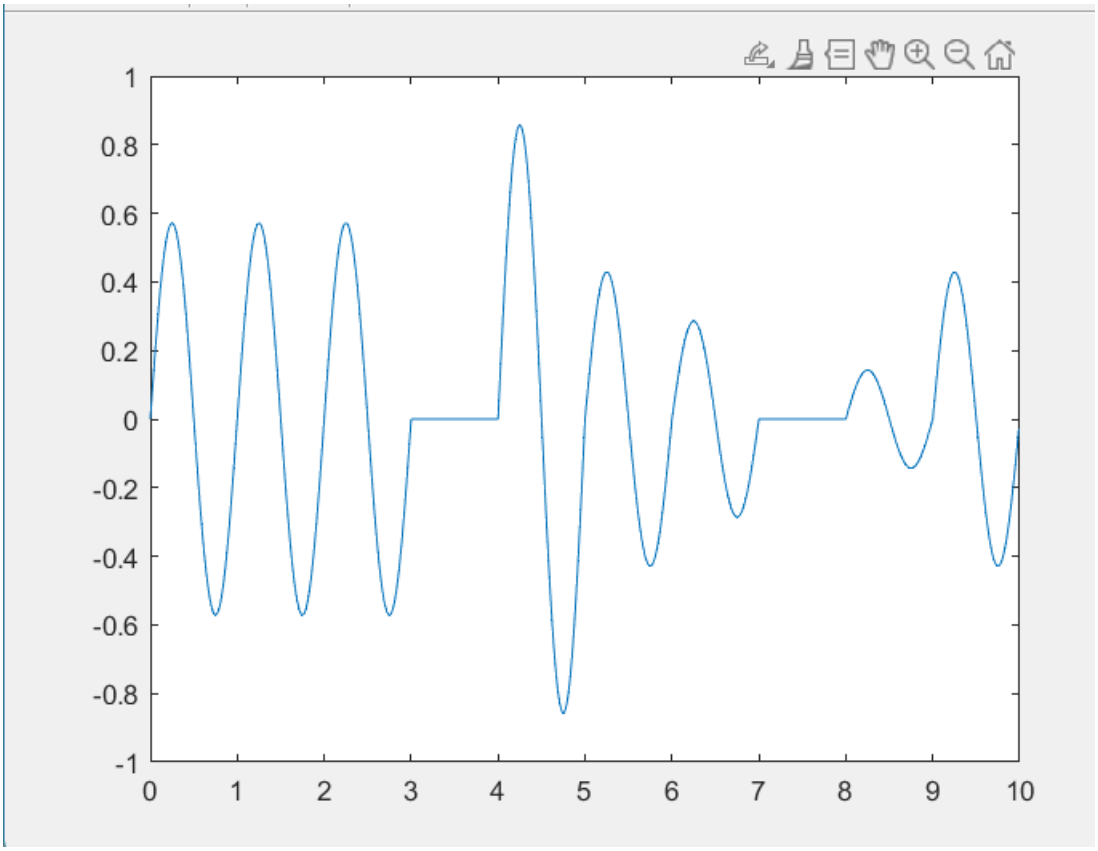
Rate = 1:


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25
30

Rate = 2:


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15

Rate = 3:



۴-۱:

اکنون در این بخش میخواهیم تابع decode آنرا ایجاد کنیم که ورودی آن پیام کدگذاری شده است و همچنین سرعت ریتی که پیام ارسال شده است و خروجی آن رمزگشایی شده پیام باشد

```
function DecodedText = amp_decoding(coding_amp , rate)
    CorrArr = [];

    for i = 1:100:length(coding_amp)
        x = 1 : 0.01 : (2-0.01);
        y = 2*sin(2*pi.*x);

        CorrArr = [CorrArr , 0.01*(coding_amp(i:(i+99))*y')];
    end
    tmp = dec2bin(uint8(CorrArr .* (2.^rate - 1)));

    tmp_str = '';

    [numRows,numCols] = size(tmp);
    for i = 1:numRows
        tmp_str = strcat (tmp_str, tmp(i,:));
    end


    Mapset = CreateMapset();

    DecodedText = '';
    for i = 1:5:strlength(tmp_str)
        index = find(strcmp(Mapset(2,:), tmp_str(i:i+4)));
        DecodedText = strjoin([DecodedText , Mapset(1,index)],'');
    end

    DecodedText
end
```

همانگونه که در کد مشخص است نخست به صورت گسسته کورولیشن گیری از ۱۰۰ نمونه انجام شده و با 2sin(2pi*t) حساب می شود عدد حاصل ضریب پایه ریتی که برای تقسیم بندی سیگنال ها استفاده کردیم است . اکنون عملا رویه معکوس آنچه پیشتر انجام دادیم را پیش میگیریم ، با ضرب 2^rate -1 در ضرایب حاصله و تبدیل آن به باینری و رشته باینری شکل می گیرد اکنون زمان تطبیق آن با مپ ستی است که تعبیه کرده بودیم و با دستور find هر حرف مربوطه را در حلقه آخری به یکدیگر متصل کرده و رشته نخستین حاصل می شود.

۵-۱:

برای شبیه سازی واقعیت در اثر گذاری سیگنال ها میخواهیم سیگنال را نویز دار کنیم و با توجه به دستور صورت سوال به کمک دستور randn نویز گوسی با میانگین صفر و واریانس یک هستند . که به کمک noiseTest نویز خواسته شده را شکل میدهیم.

```
Noise = randn(1, 3000);

mean = mean(Noise)
variance = var(Noise)
```

```
mean =

   -0.0089


variance =

    0.9318

fx >>
```

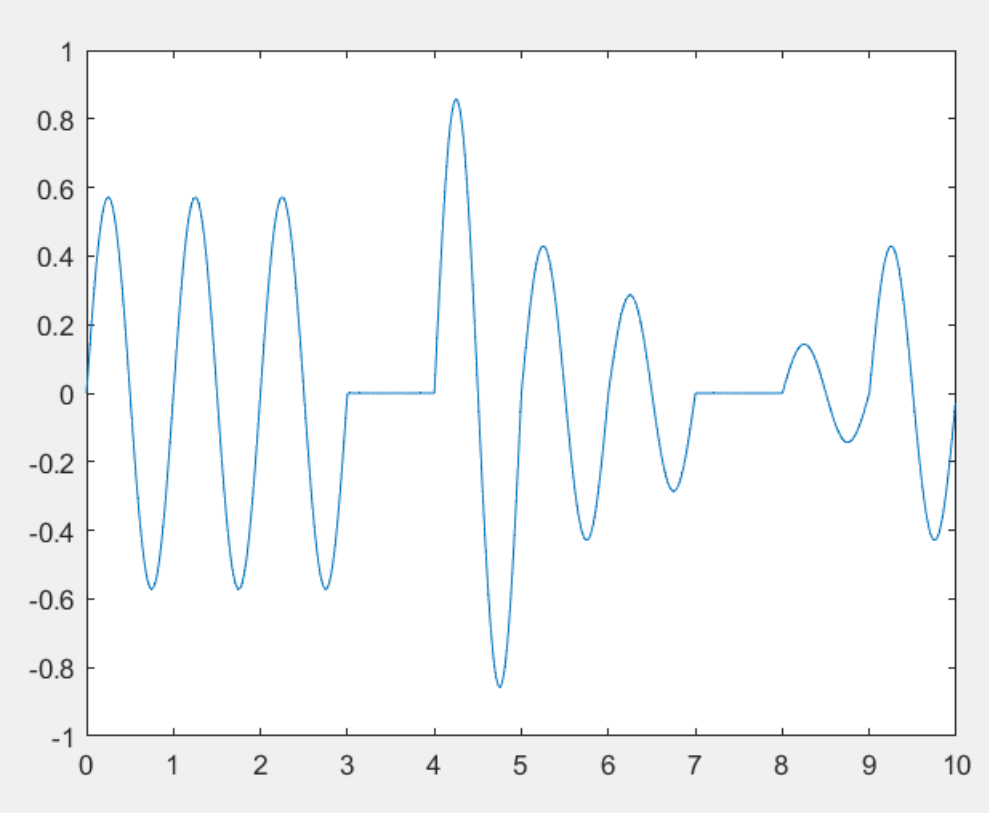
۱-۶:

اکنون مجدد به کمک p1 نویز را ۰.۰۰۱ تعیین کرده و با میانگین صفر آنرا به سیگنال اضافه می کنیم اکنون با توجه به بخش uint8 تعبیه شده در تابع دیکود این اعداد به صورت صحیح در آمده و عملا گرد شده و از float خارج می شوند که این مهم سبب می شود و اعداد نا دقیق مارو سبب می گردد که به نزدیک ترین عدد تبدیل بشوند. فلذا در ریت های پایین تر نیز جواب قابل قبول خواهد بود و خروجی signal به صورت صحیح نمایان می گردد.

۱-۷:

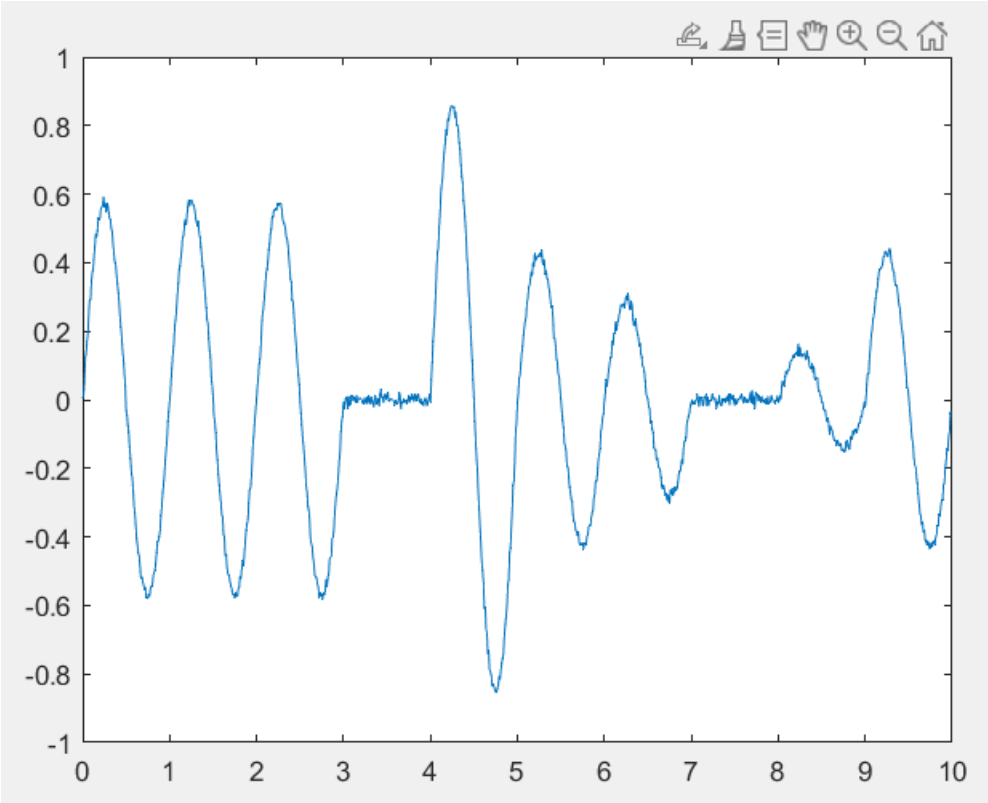
اکنون با تغییر واریانس نویز طی چند مرحله و افزایش قدرت نویز مشاهدات زیر را خواهیم داشت.

Rate = 3 & variance = 0.0001



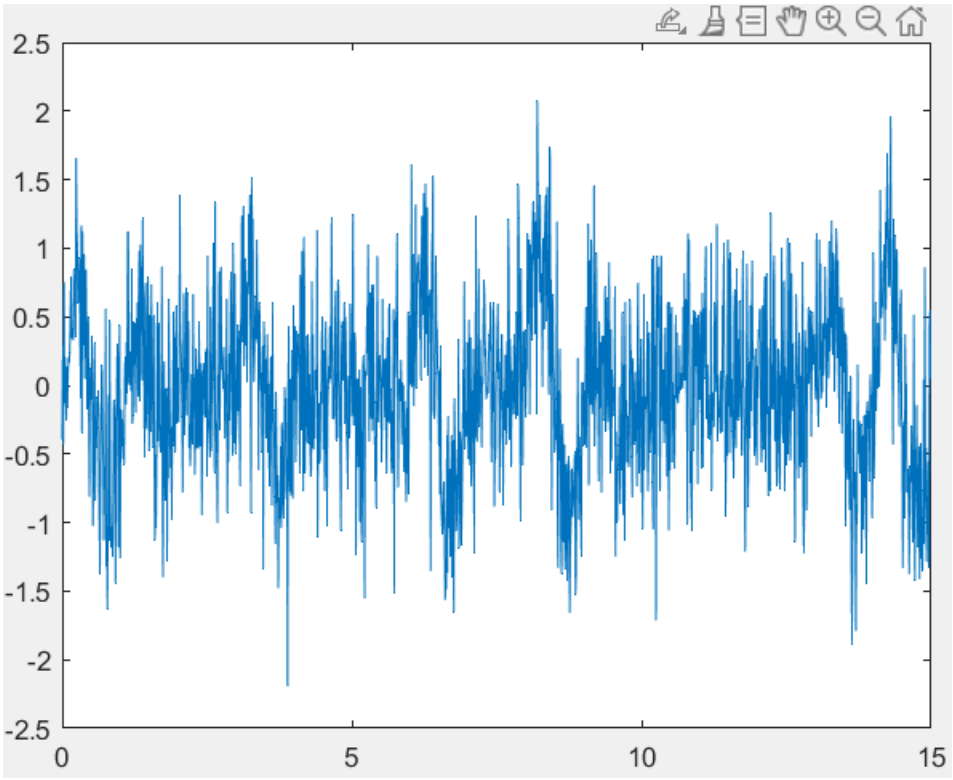
```
Command Window

DecodedText =

    'signal'

fx >>
```

Rate = 3 & variance=0.01




```
DecodedText =

    'signal'
```

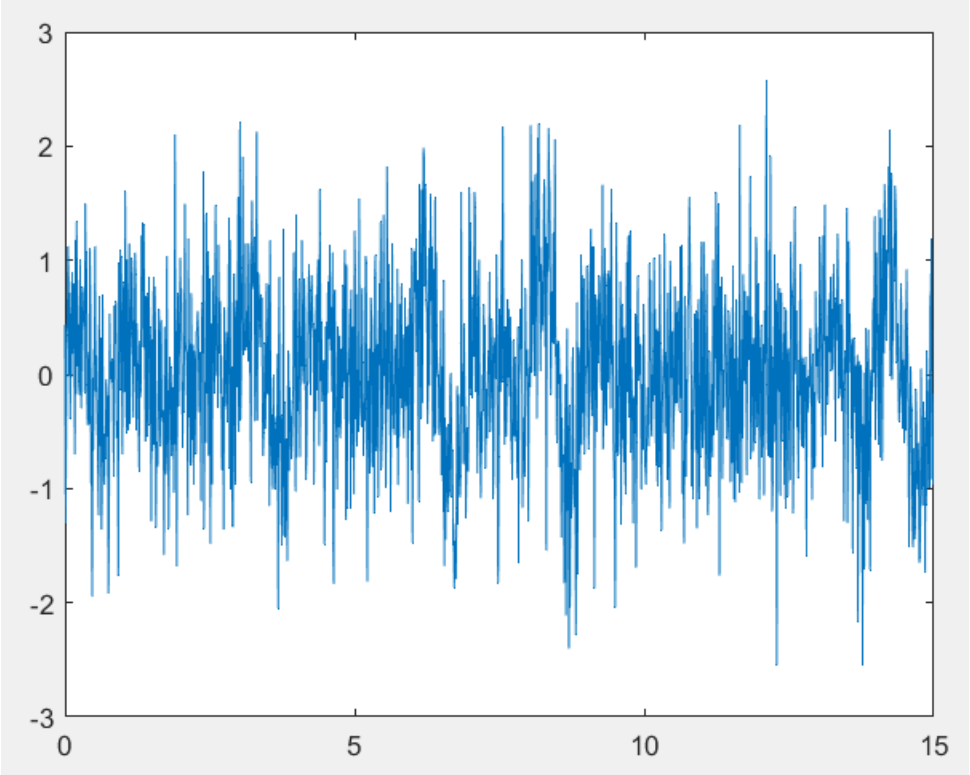
Rate = 2 & variance = 0.5:




```
DecodedText =

    'signal'
```

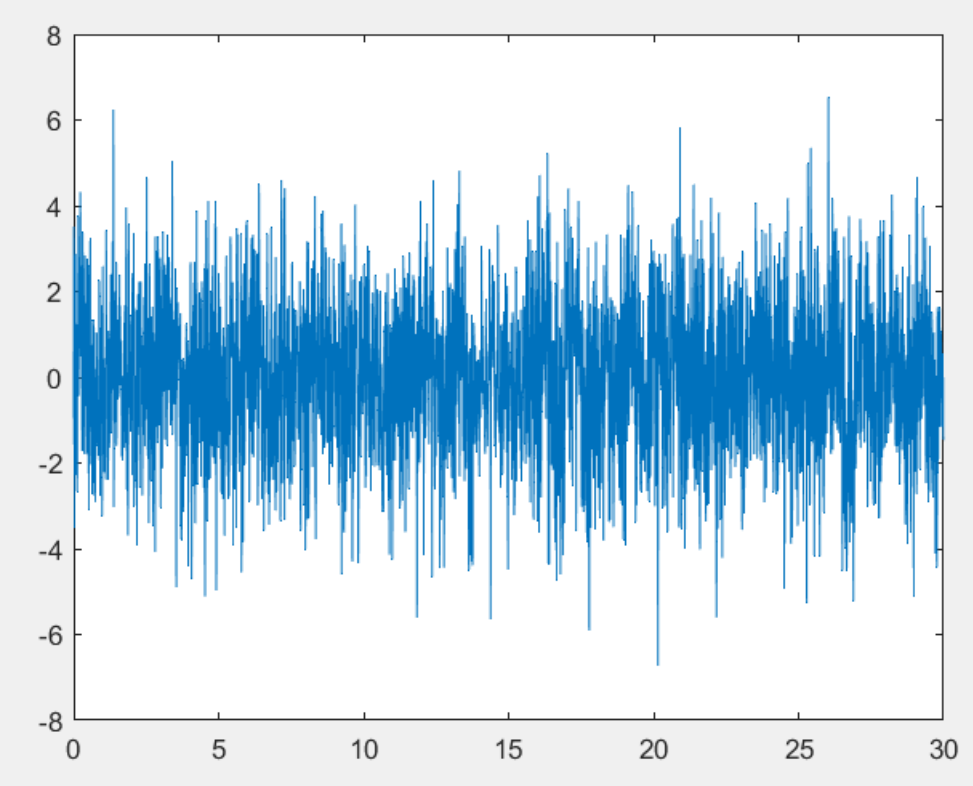
و اما در ریت و واریانس هایی مانند زیر نویز از حد گذشته و پیام به صورت اشتباه دیکود می شود :

Rate = 2 & variance = 0.7




```
DecodedText =

    'kjgnal'
```

Rate = 1 & variance = 1.8




```
DecodedText =

    'signak'
```

۱۰-۹-۸:

همانطور که مشاهده شد ، بیت ریت های کمتر مقاومت بیشتری از خود در برابر نویز نشان می دهند به دلیل اینکه ناحیه ۰ و ۱ در آن کمتر و طول هر یک از بازه ها بیشتر می شود و قدرت نویز بیشتری نیاز است تا آنها را از محدوده گرد شدن به آن عدد اصلی خارج کرده و باعث خطا شود .

با بالاتر بردن قدرت فرستنده تا حداکثر دامنه سیگنال نیز افزایش یافته و طور بازه های تقسیم بندی بیشتری می شود تا هر بیت ریت مقام تر باشد پس طبیعتا برای بالاتر بردن سرعت بایستی قدرت فرستنده نیز افزایش یابد تا تاثیر پذیری از نویز کاهش بیابد.

در عدم حضور نویز، ماکسیمم بیت ریت مطلوب به تعداد بیت های متغیر بستگی دارد، چون تعداد سطح هایی را مشخص میکند که میتوانیم داشته باشیم.

در بخش دیکود این پروژه چون از unit8 استفاده کردیم پس ۸ بیت داریم و نمیتوانیم از سرعتی استفاده کنیم که تقسیم بندی آن مربوط به تعداد بیت و در نتیجه تعداد سطح بیشتری می طلبد.

۱-۱۱:

در بخش پیشین چون افزایش دامنه تنها بر خود سیگنال اعمال می شود ، نسبت سیگنال به نویز افزایش می یافت و خطا کم بنابراین حساسیتی در برابر نوییز تغییر ایجاد میکرد ؛ اما در اینجا در فرآیند دیکود چنانچه از 10 sin (2pi * t) به جای کورولیشن گیری با

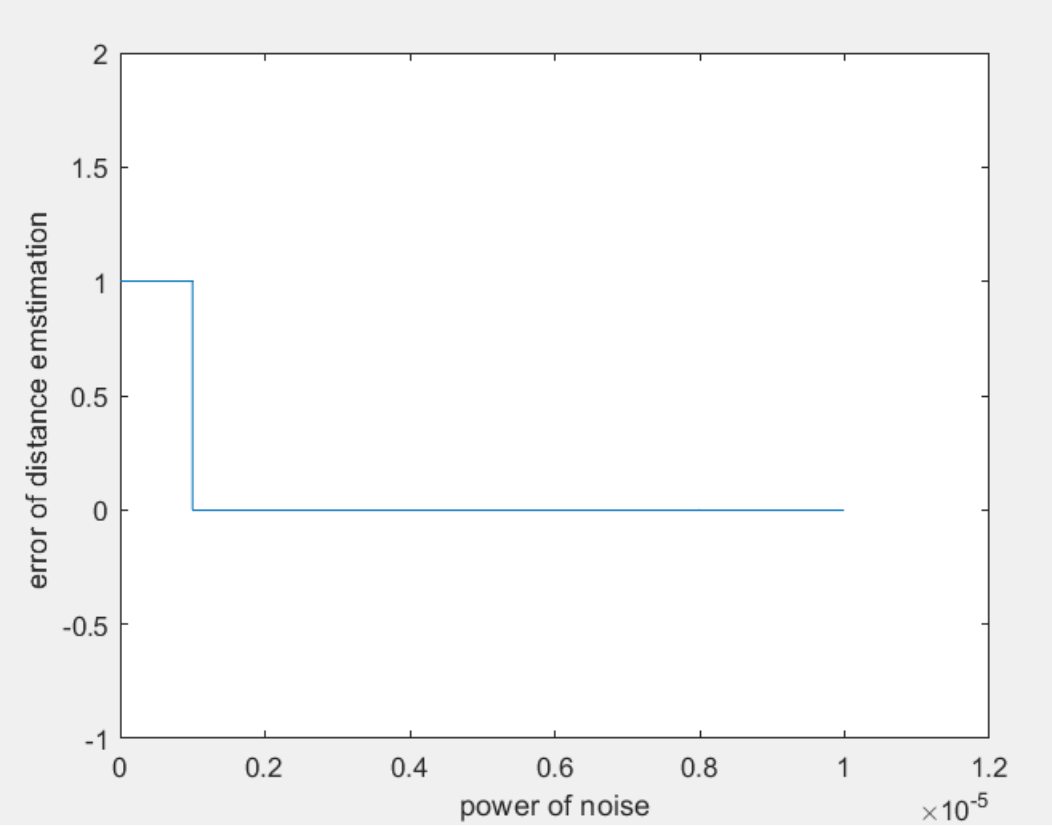
2sin(2pi*t) استفاده کنیم ، چون نسبت سیگنال به نویز عوض نمی شود ، پس تفاوتی هم در دیکودینگ و مقاومت در برابر نویز اتفاق نمی افتد .

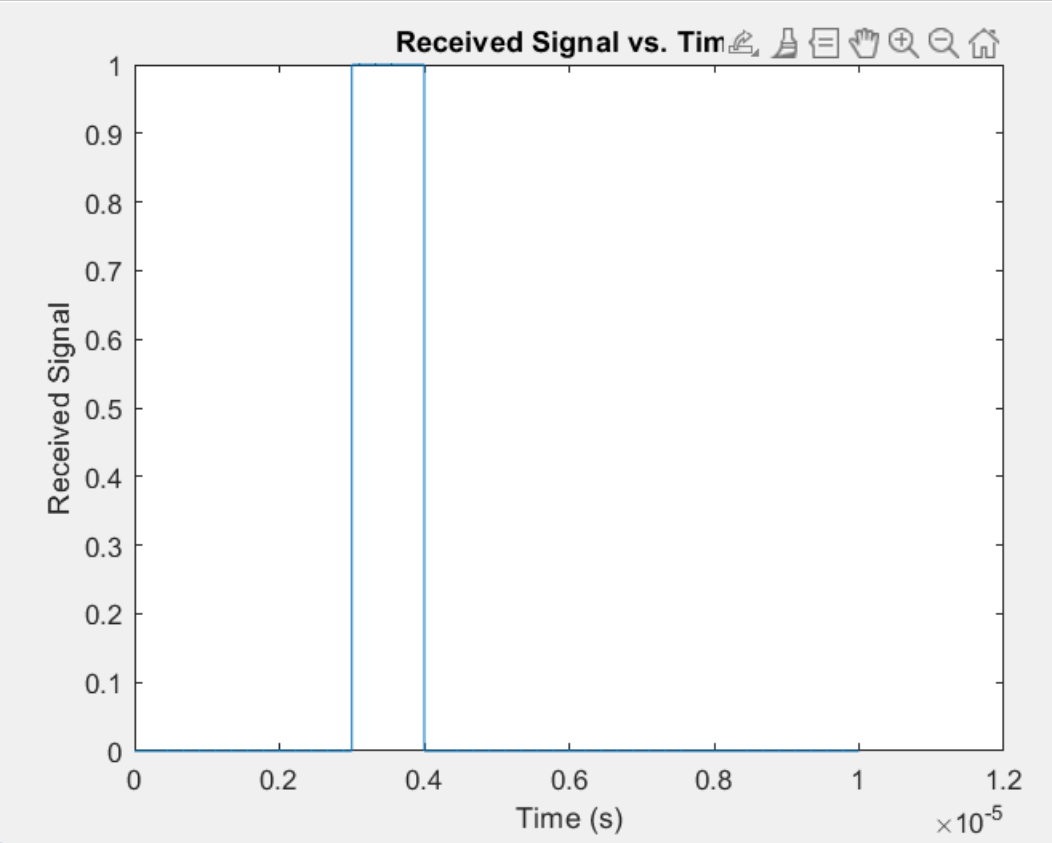
۱-۱۲:

در اینترنت های خانگی adsl سرعت در بهترین حالت 16 Mb/sec و در حالت عادی 10Mb/sec است که ما در این تمرین با سرعت حداکثر 3b/sec ارسال کردیم.

۲:

چند خواسته بخش ۱-۲ عملا مشابه آن چیزی است که در CA نخست انجام دادیم و که حالات خواسته شده در دو فایل مجزا پسوند ۱ قرار داده شده اند و خروجی آنها به شکل زیر می باشد:



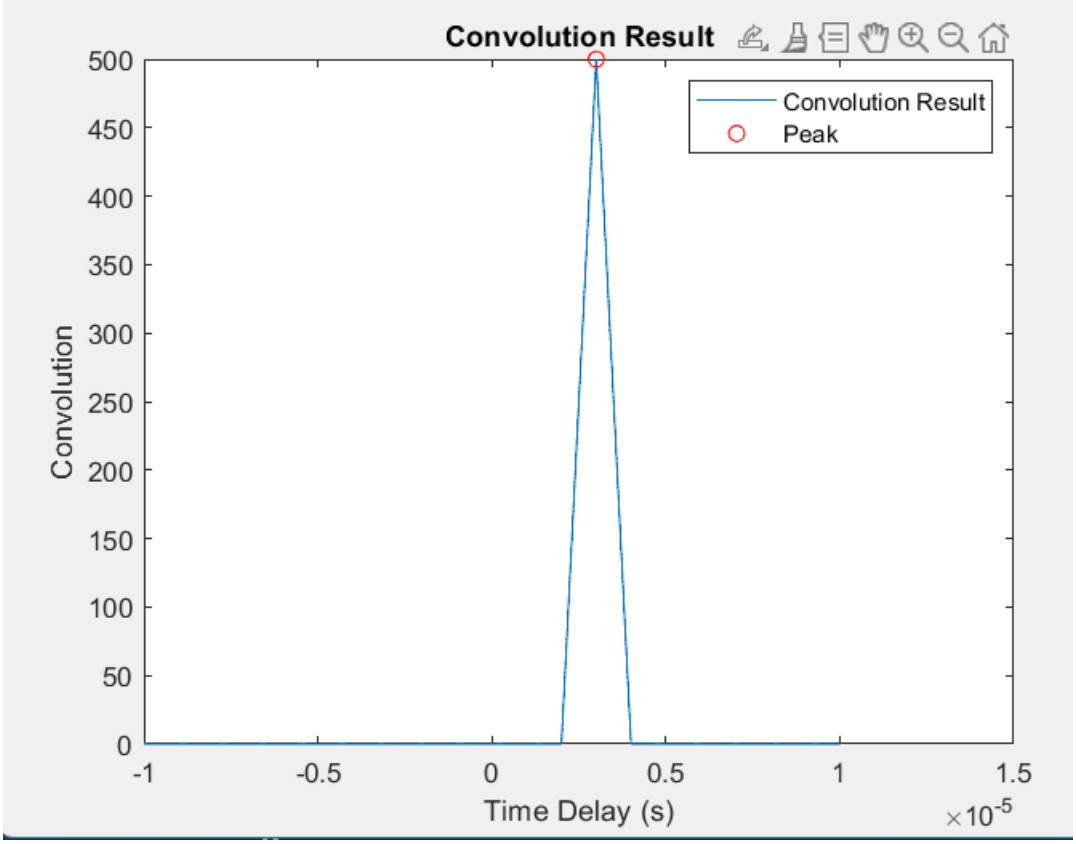


۲-۲:

در این بخش اما می خواهیم آنچه را پیشتر با کورولیشن گیری در CA1 پیش بردیم اینبار با کانولوشن و به کمک دستور conv اجرا بکنیم .

```
tau = 1e-6;

t = 0:ts:T;
tlen = length(t);
sent = zeros(1, tlen);
sent(1:round(tau/ts)) = 1;

R = 450;
a = 0.5;
c = 3e8;

td = 2 * R / c;

received = circshift(0.5 * sent, round(td/ts));
% Use convolution instead of correlation
convolution_result = conv(received, flip(sent));

convolution_t = -T:ts:T;
convolution_t = convolution_t(1:length(convolution_result));

figure;

plot(convolution_t, convolution_result);
xlabel('Time Delay (s)');
ylabel('Convolution');
title('Convolution Result');
hold on;

[max_conv, max_idx] = max(convolution_result);
plot(convolution_t(max_idx), max_conv, 'ro');
legend('Convolution Result', 'Peak');

time_delay = convolution_t(max_idx);
estimated_R = time_delay * c / 2;

disp(['Estimated Distance (R): ', num2str(estimated_R), ' meters']);
```

در اینجا در ادامه توضیحات CA1 ، xcorr را با conv تغییر دادیم و خروجی آن به شکل زیر می گردد:

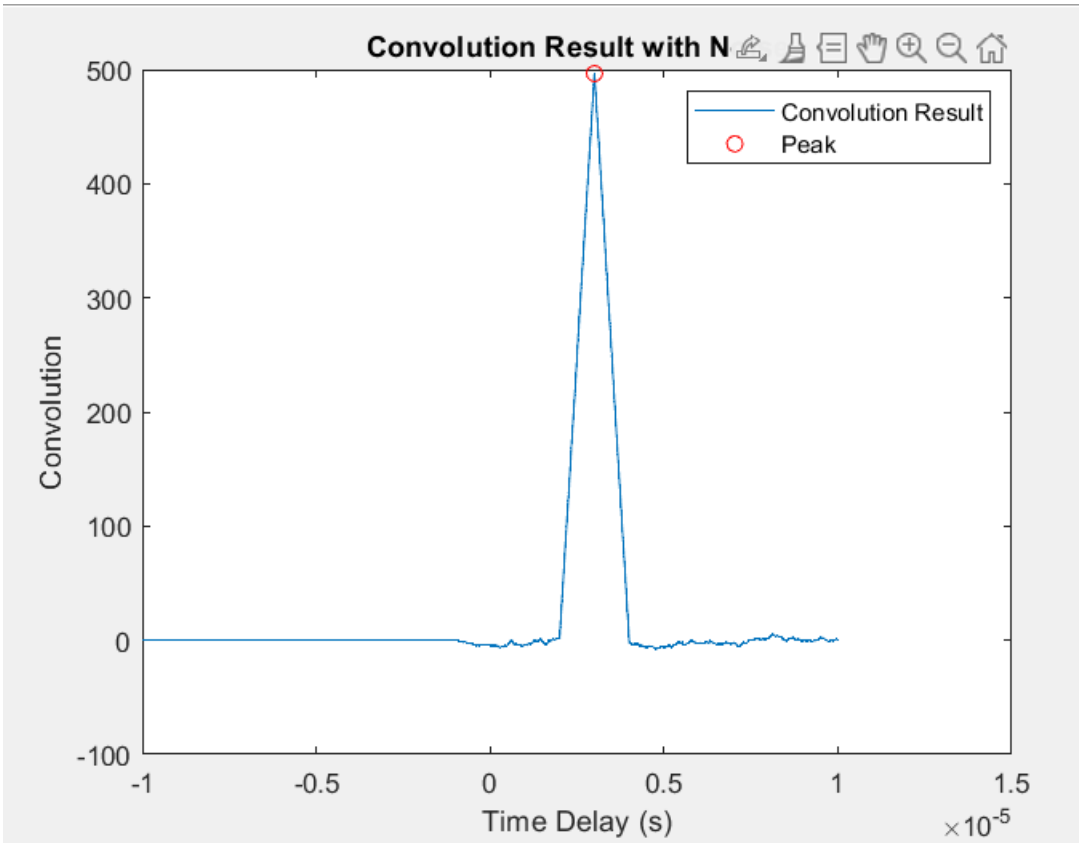



```
>> p2_2
Estimated Distance (R): 450 meters
```

۲-۳:

اینبار در این بخش با افزودن نویز به بخش قبلی و افزودن نویز به آن؛ و بررسی می کنیم که اگر خطای فاصله یابی آن کمتر از ۱۰ متر بوده باشه تخمین درستی است با آن نویز لولی که برایش در نظر گرفته بودیم.

```
t = 0:ts:T;
tlen = length(t);
sent = zeros(1, tlen);
sent(1:round(tau/ts)) = 1;

R = 450;
a = 0.5;
c = 3e8;

td = 2 * R / c;

received = circshift(0.5 * sent, round(td/ts));

noise_level = 0.1; % addjust the noise level
received_with_noise = received + noise_level * randn(size(received));

% use convolution
convolution_result = conv(received_with_noise, flip(sent));

convolution_t = -T:ts:T;
convolution_t = convolution_t(1:length(convolution_result));

figure;

plot(convolution_t, convolution_result);
xlabel('Time Delay (s)');
ylabel('Convolution');
title('Convolution Result with Noise');
```

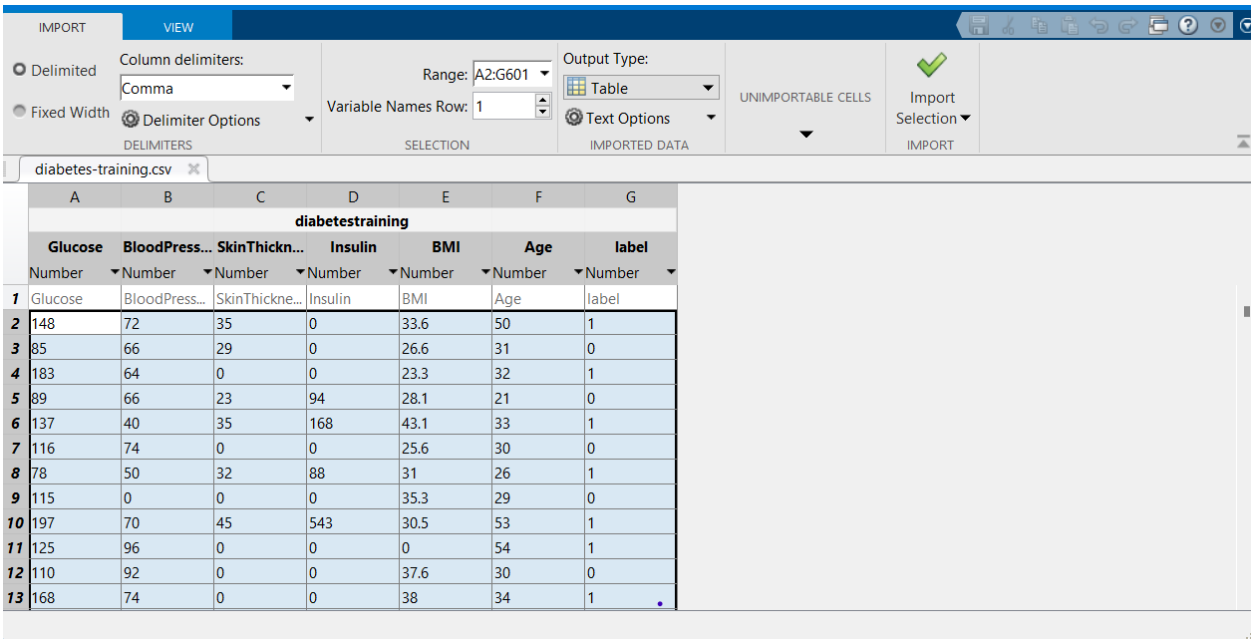



```
>> p2_3
Estimated Distance (R): 450 meters
The difference is less than 10.
```

:۳

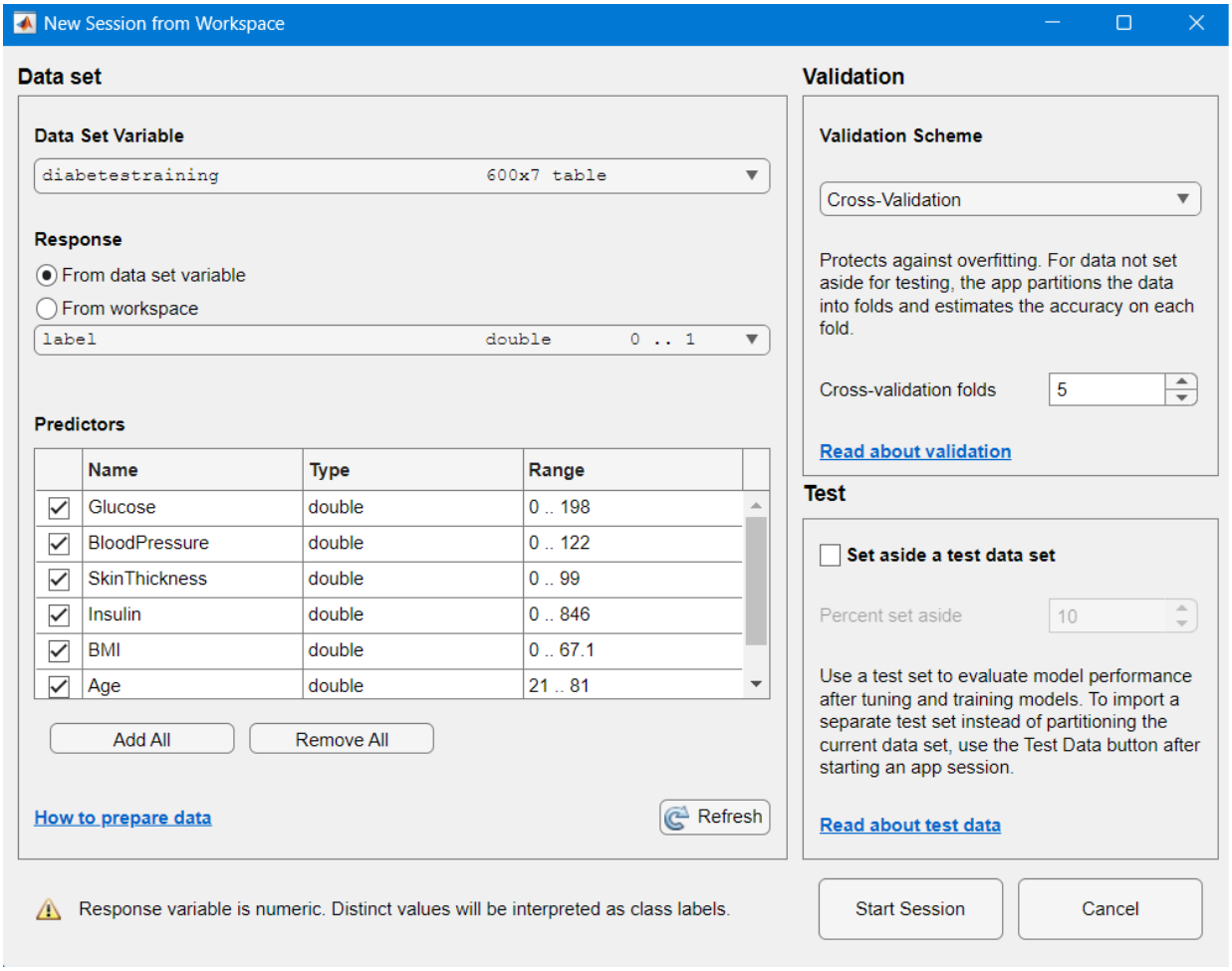
در این بخش برای ساخت سیستم فراخوانی مشتریان تابع فوق دو ورودی شماره باجه و مشتری را دریافت کرده و در مپ ست که در پوشه مجزا آنها ضبط شده اند در یک دیکشنری ذخیره سازی می کند وعملا خواندن هر فایل صوتی را برایش ممکن می سازد . در ادامه برای هندل ارور ها شرایطی که را که صورت سوال برای بازه اعداد برای شماره مشتری و باجه ها مطرح کرده است را راستی آزمایی می کند سپس در get number text فراخوانی میشود تا متن مربوط به مشتری را از دیکشنری دریافت نماید و سپس تابع بر اساس مقدار این شماره تصمیم گیری می کند و اعداد ۱ تا ۲۰ که به صورت مجزا بیان شده اند و اعداد ۲۰ تا ۹۰ و یکان آنها از دو بخش به کمک واسط o بیان شده اند که در این مرحله آنها از یکدیگر تفکیک می شوند و عملا تصمیم گیری مبنی بر شکل کلی آن عدد بر اساس مقدار ورودی شکل می گیرد در ادامه concatenate audio می آید و بخش های مختلف را به یکدیگر متصل می کند و به کمک sound آن پلی می شود و دستور فراخوانی خواسته شده بر اساس اعدادی که به تابع ورودی دادیم صورت می پذیرد.

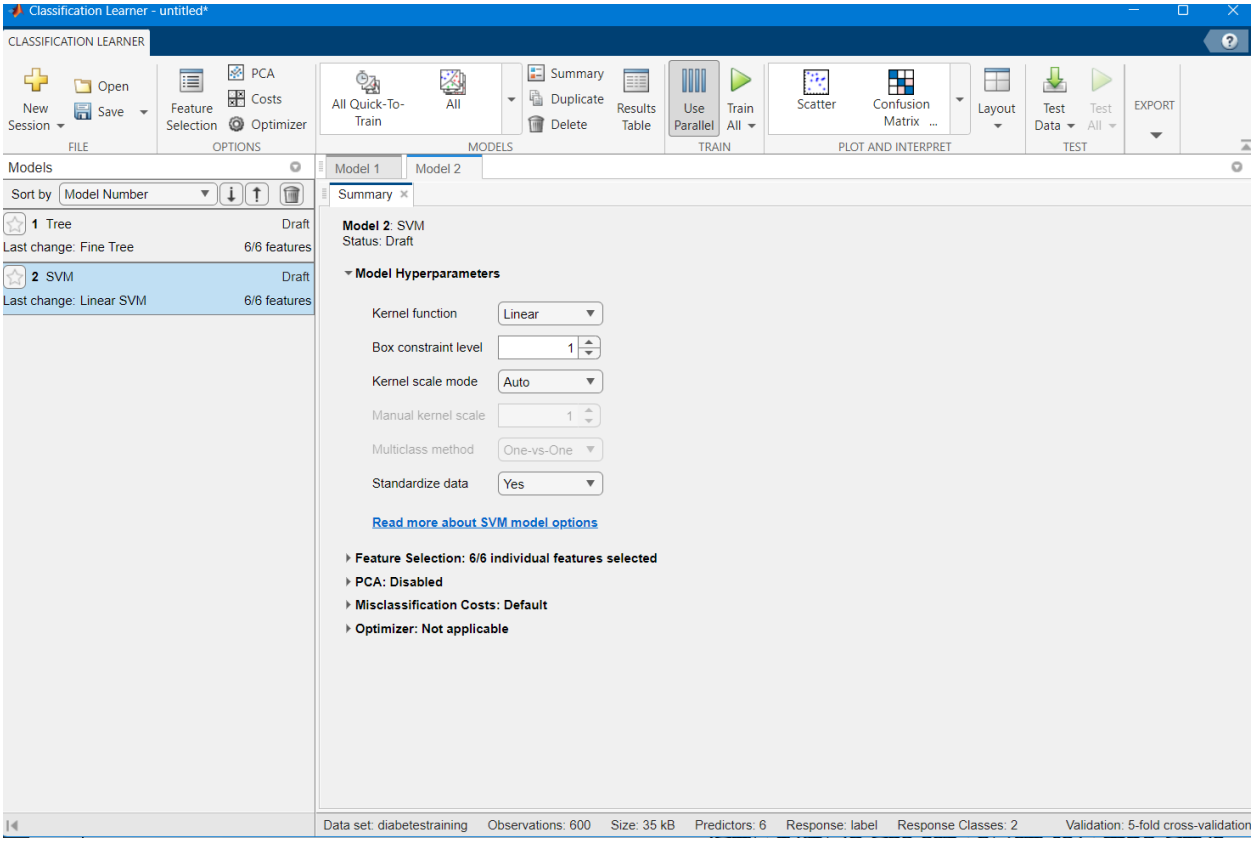
:۴

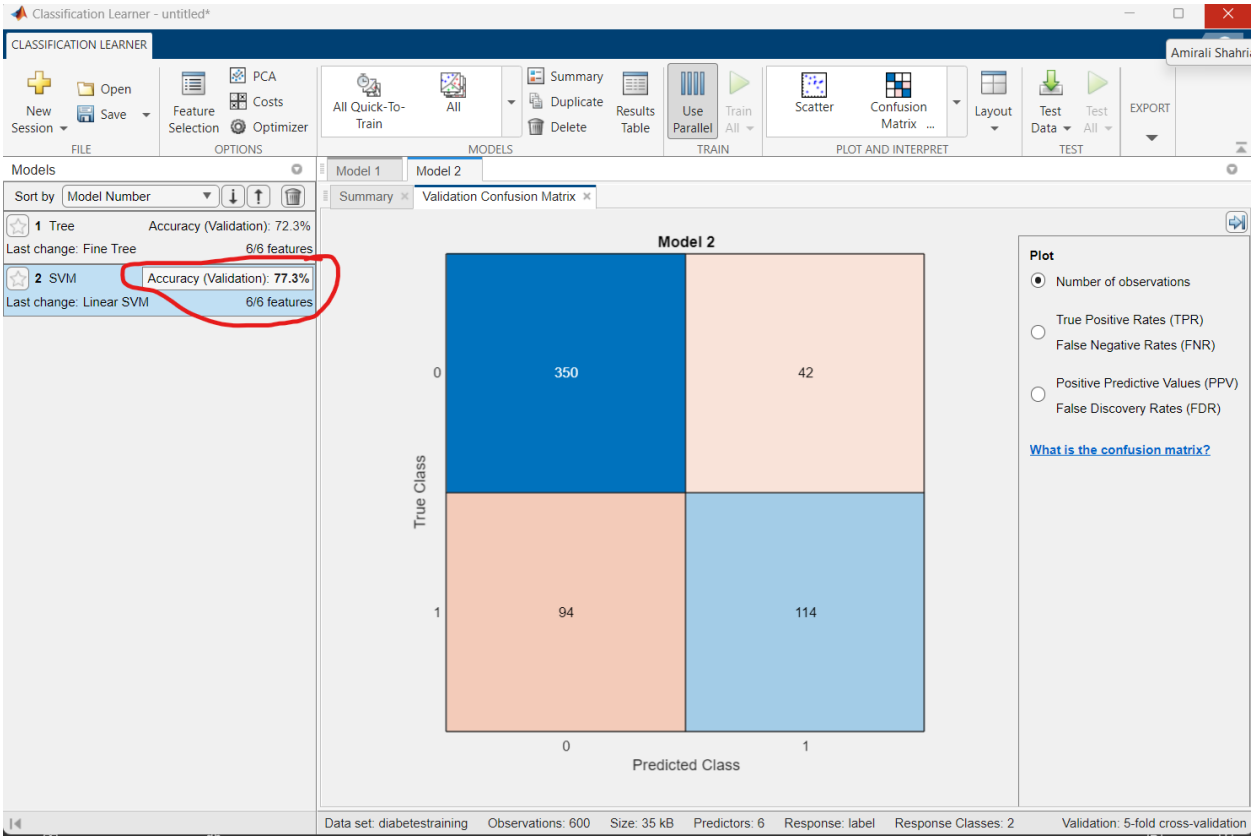


diabetestraining

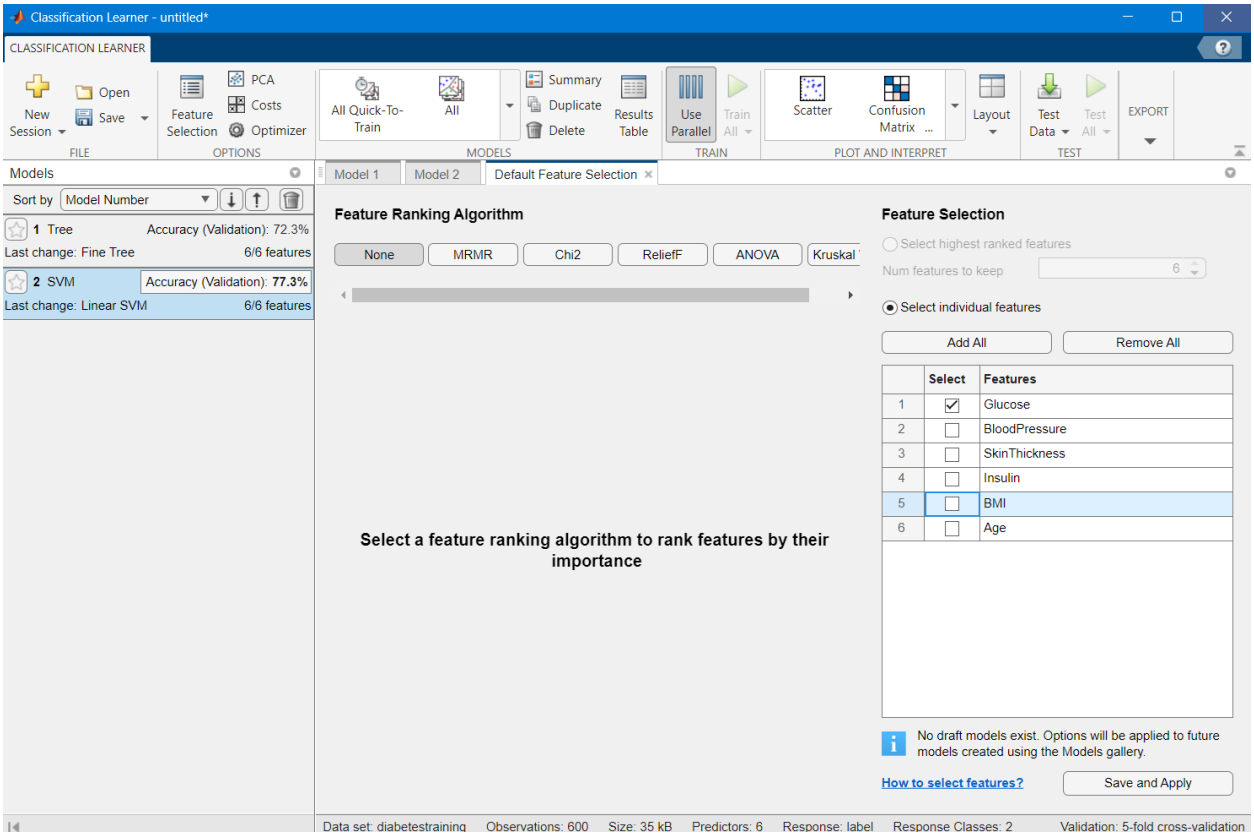
	Glucose	BloodPress...	SkinThickn...	Insulin	BMI	Age	label
	Number	Number	Number	Number	Number	Number	Number
1	Glucose	BloodPress...	SkinThickne...	Insulin	BMI	Age	label
2	148	72	35	0	33.6	50	1
3	85	66	29	0	26.6	31	0
4	183	64	0	0	23.3	32	1
5	89	66	23	94	28.1	21	0
6	137	40	35	168	43.1	33	1
7	116	74	0	0	25.6	30	0
8	78	50	32	88	31	26	1
9	115	0	0	0	35.3	29	0
10	197	70	45	543	30.5	53	1
11	125	96	0	0	0	54	1
12	110	92	0	0	37.6	30	0
13	168	74	0	0	38	34	1

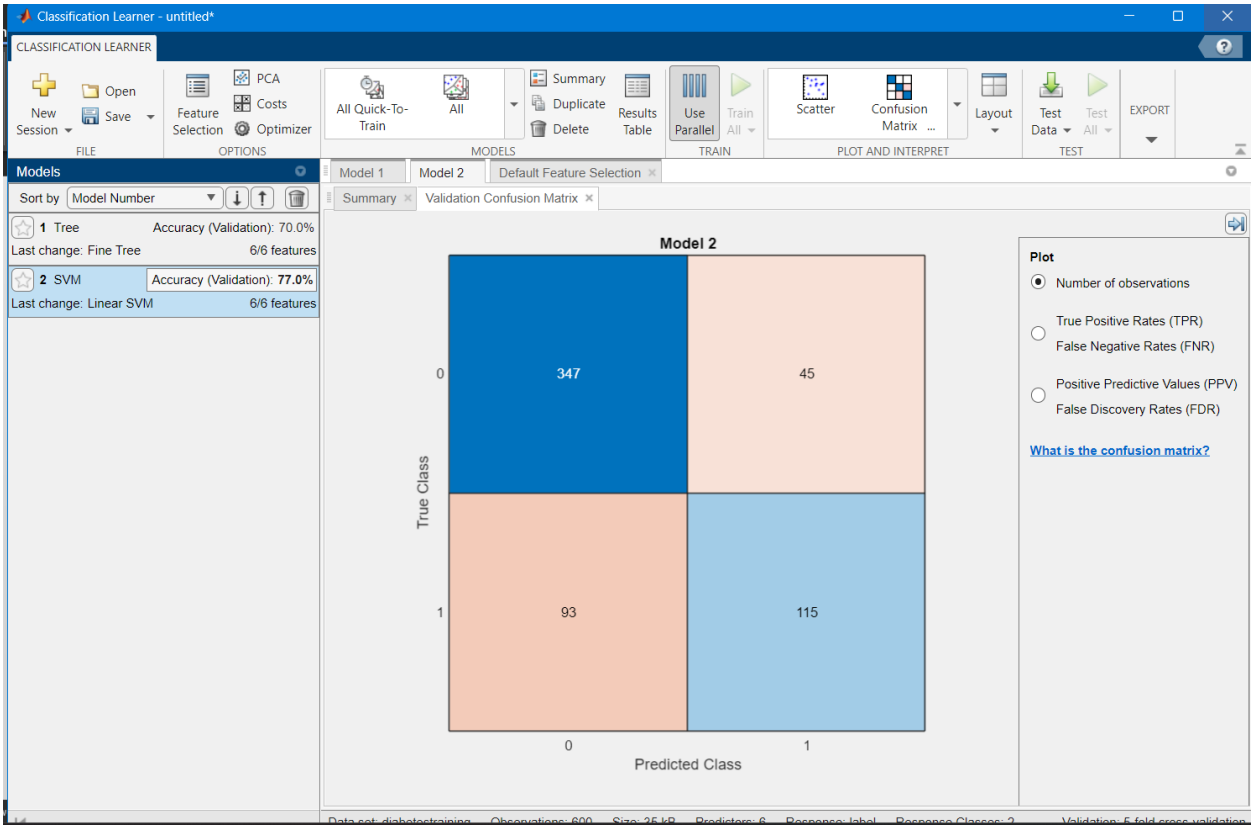


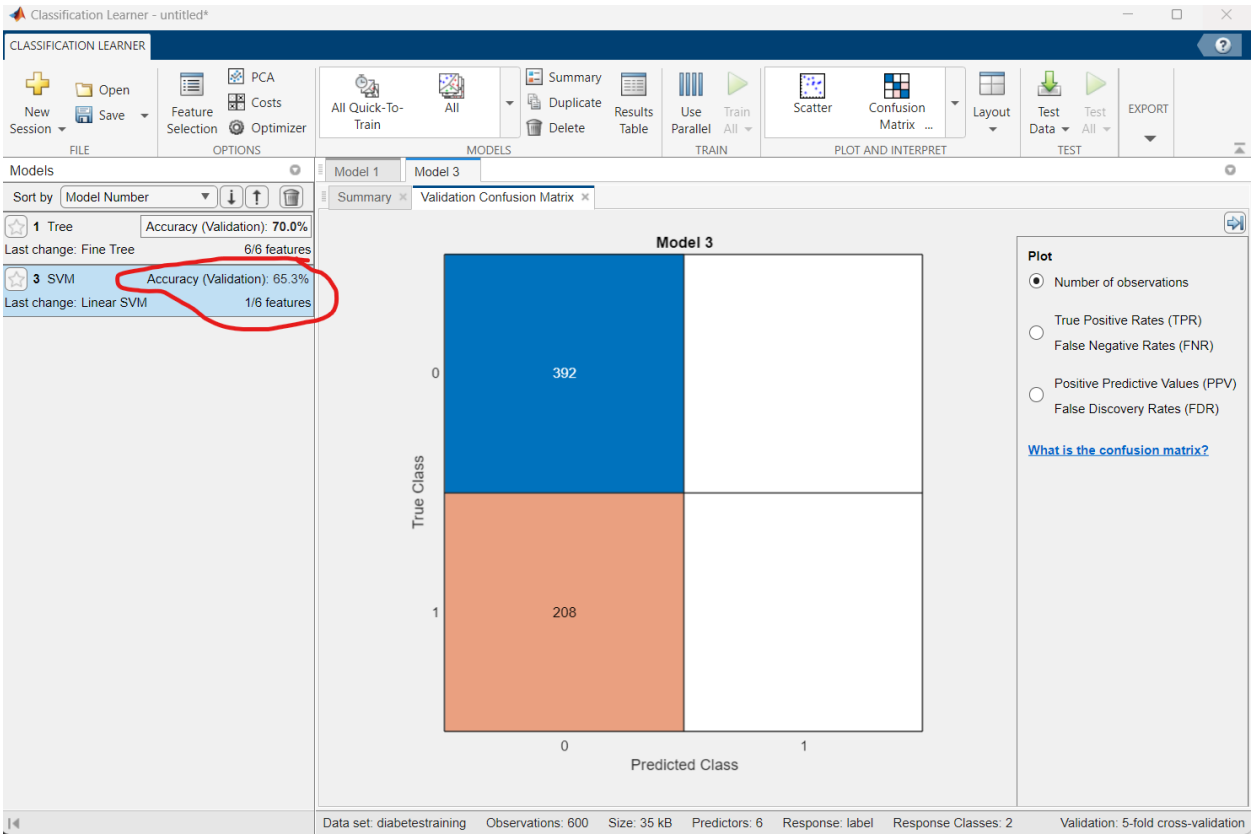

Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
TEST
EXPORT
Models
Sort by Model Number
1 Tree Draft
Last change: Fine Tree 6/6 features
2 SVM Draft
Last change: Linear SVM 6/6 features
Model 1
Model 2
Summary
Model 2: SVM
Status: Draft
Model Hyperparameters
Kernel function Linear
Box constraint level 1
Kernel scale mode Auto
Manual kernel scale 1
Multiclass method One-vs-One
Standardize data Yes
Read more about SVM model options
Feature Selection: 6/6 individual features selected
PCA: Disabled
Misclassification Costs: Default
Optimizer: Not applicable
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation

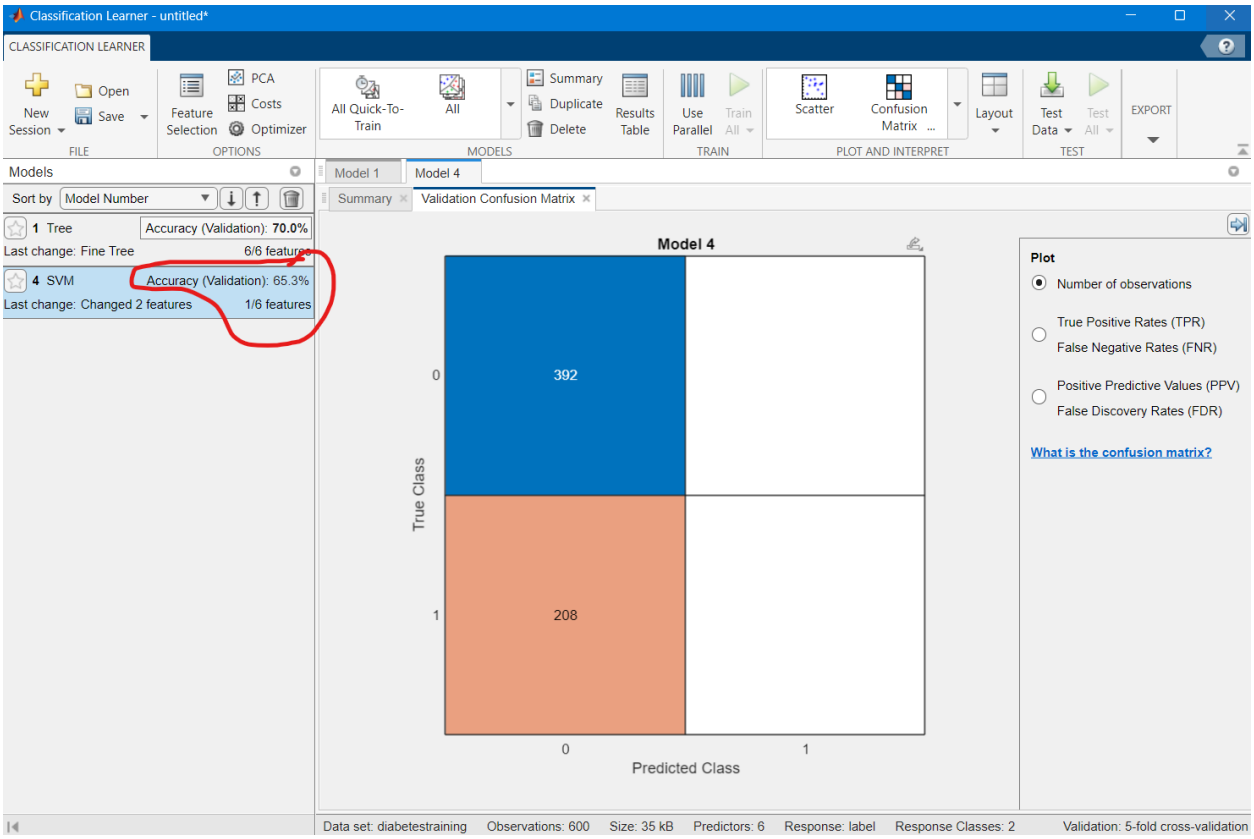

Classification Learner - untitled*
CLASSIFICATION LEARNER
Amirali Shahri
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
EXPORT
TEST
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 72.3%
Last change: Fine Tree
6/6 features
2 SVM
Accuracy (Validation): 77.3%
Last change: Linear SVM
6/6 features
Model 1
Model 2
Summary
Validation Confusion Matrix
Model 2
350
42
94
114
0
1
True Class
0
1
Predicted Class
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?
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation

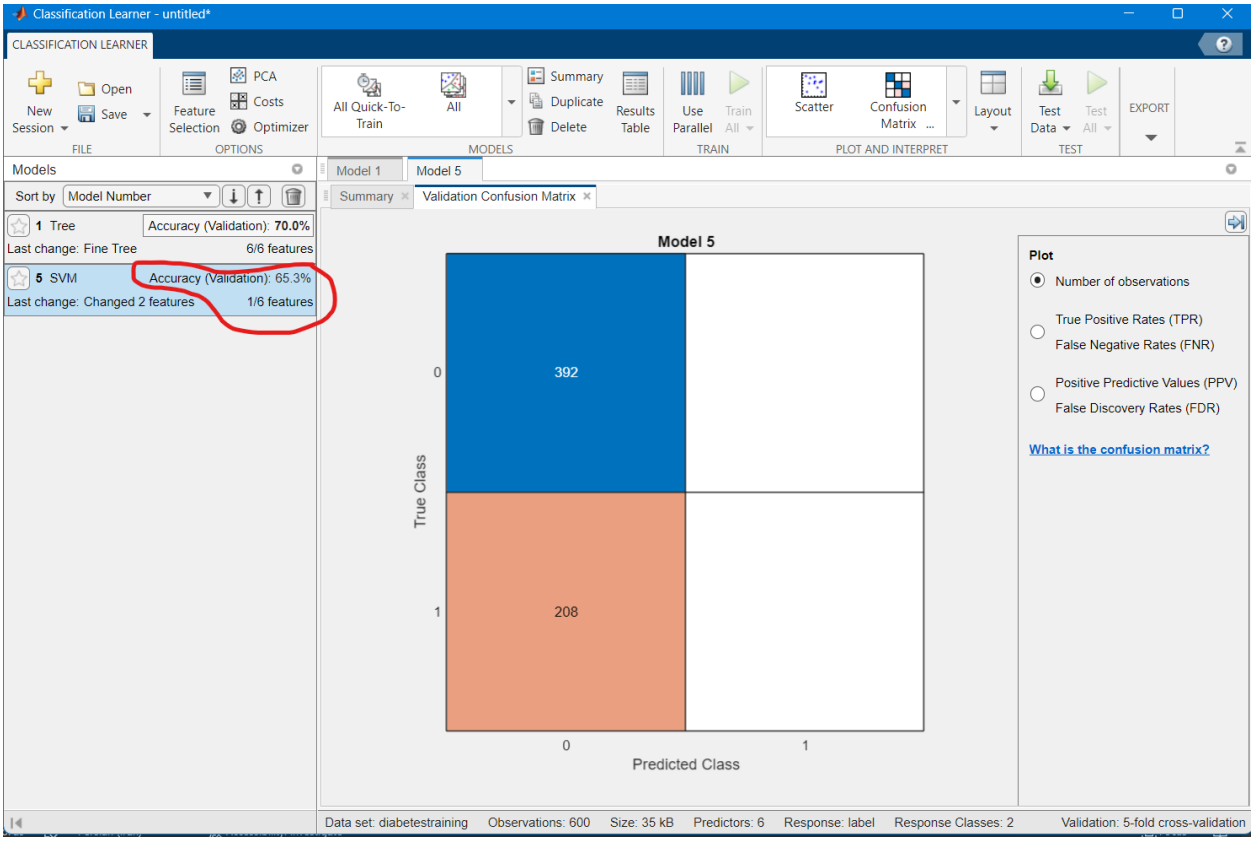
۲-۴:

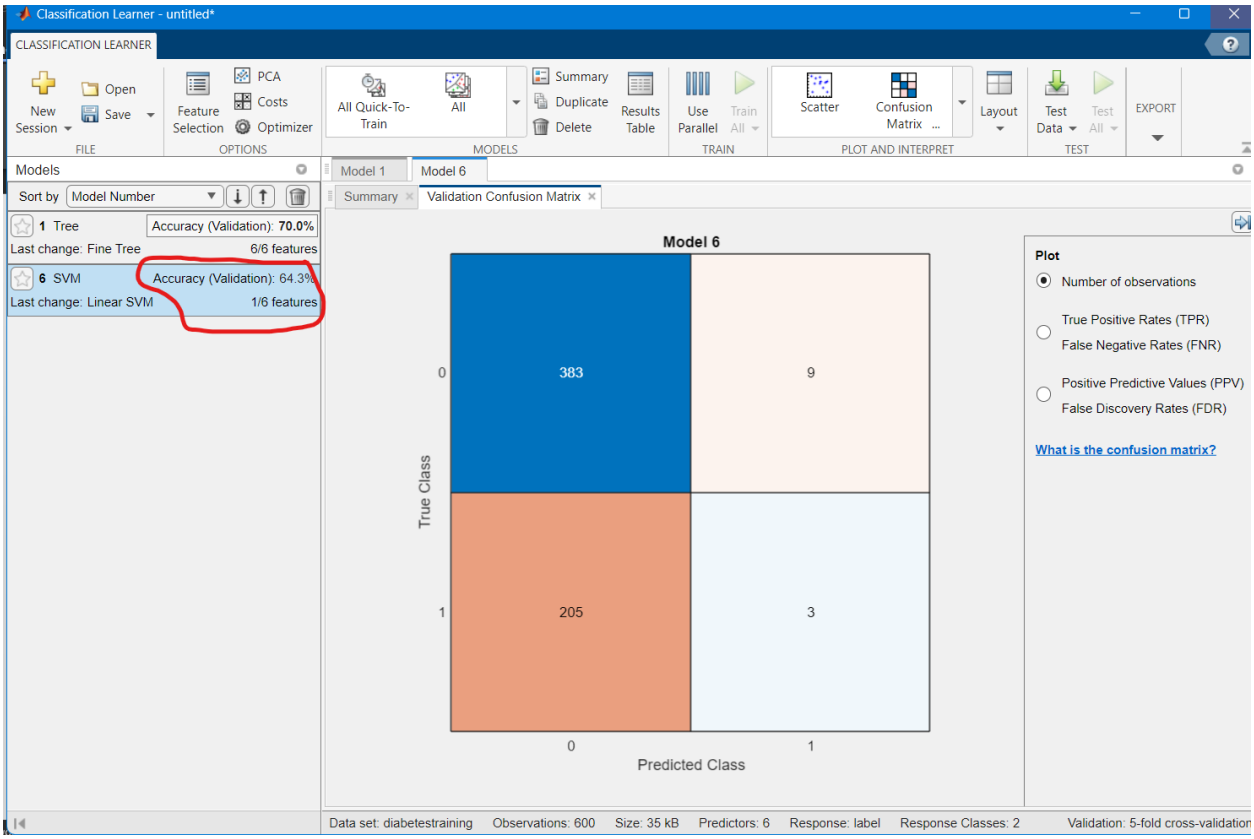

Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
TEST
EXPORT
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 72.3%
Last change: Fine Tree
6/6 features
2 SVM
Accuracy (Validation): 77.3%
Last change: Linear SVM
6/6 features
Model 1
Model 2
Default Feature Selection
Feature Ranking Algorithm
None
MRMR
Chi2
ReliefF
ANOVA
Kruskal
Select a feature ranking algorithm to rank features by their importance
Feature Selection
Select highest ranked features
Num features to keep
6
Select individual features
Add All
Remove All
Select
Features
1 Glucose
2 BloodPressure
3 SkinThickness
4 Insulin
5 BMI
6 Age
No draft models exist. Options will be applied to future models created using the Models gallery.
How to select features?
Save and Apply
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation

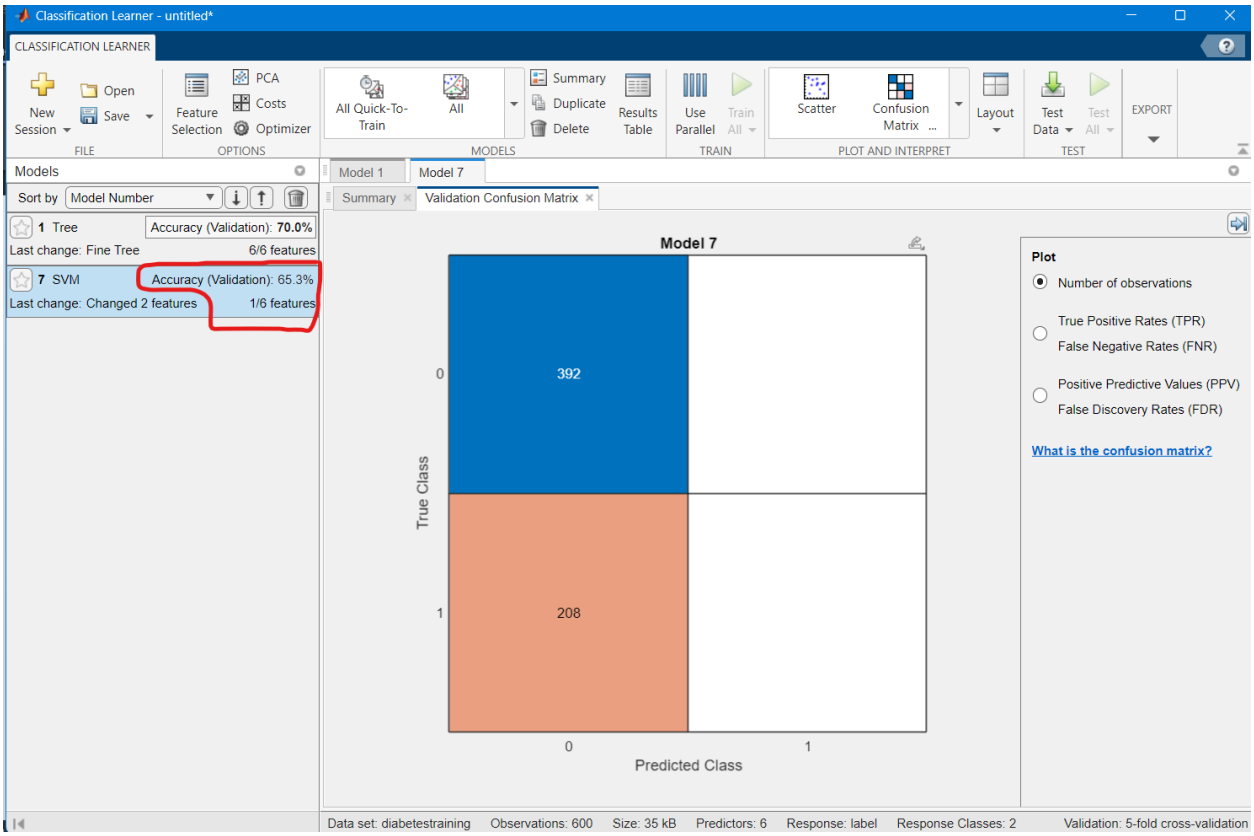

Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
EXPORT
TEST
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 70.0%
Last change: Fine Tree
6/6 features
2 SVM
Accuracy (Validation): 77.0%
Last change: Linear SVM
6/6 features
Model 1
Model 2
Default Feature Selection
Summary
Validation Confusion Matrix
Model 2
347
45
93
115
0
1
True Class
0
1
Predicted Class
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?


Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
EXPORT
TEST
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 70.0%
Last change: Fine Tree
6/6 features
3 SVM
Accuracy (Validation): 65.3%
Last change: Linear SVM
1/6 features
Model 1
Model 3
Summary
Validation Confusion Matrix
Model 3
392
208
0
1
True Class
0
1
Predicted Class
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?
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation


Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
EXPORT
TEST
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 70.0%
Last change: Fine Tree
6/6 features
4 SVM
Accuracy (Validation): 65.3%
Last change: Changed 2 features
1/6 features
Model 1
Model 4
Summary
Validation Confusion Matrix
Model 4
392
208
0
1
True Class
0
1
Predicted Class
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?
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation


Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
EXPORT
TEST
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 70.0%
Last change: Fine Tree
6/6 features
5 SVM
Accuracy (Validation): 65.3%
Last change: Changed 2 features
1/6 features
Model 1
Model 5
Summary
Validation Confusion Matrix
Model 5
392
208
0
1
True Class
0
1
Predicted Class
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?
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation


Classification Learner - untitled*
CLASSIFICATION LEARNER
New Session
Open
Save
FILE
Feature Selection
PCA
Costs
Optimizer
OPTIONS
All Quick-To-Train
All
Summary
Duplicate
Delete
Results Table
MODELS
Use Parallel
Train All
TRAIN
Scatter
Confusion Matrix ...
PLOT AND INTERPRET
Layout
Test Data
Test All
TEST
EXPORT
Models
Sort by Model Number
1 Tree
Accuracy (Validation): 70.0%
Last change: Fine Tree
6/6 features
6 SVM
Accuracy (Validation): 64.3%
Last change: Linear SVM
1/6 features
Model 1
Model 6
Summary
Validation Confusion Matrix
Model 6
383
9
205
3
0
1
True Class
0
1
Predicted Class
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?
Data set: diabetestraining
Observations: 600
Size: 35 kB
Predictors: 6
Response: label
Response Classes: 2
Validation: 5-fold cross-validation



۴-۳:

```
a = trainedModel.predictFcn(diabetestraining);

correctPredictions = 0;

for i = 1:600
    if (a(i,1) == diabetestraining{i, 'label'})
        correctPredictions = correctPredictions + 1;
    end
end

% accuracy
accuracy = correctPredictions / 600 * 100;

disp(['Data percentage label Estimated correctly ( for training.csv): ', num2str(accuracy), '%']);
```

در این بخش ماشینی که آموزش دادیم را اینبار با همه فیچر هایش به نام trainedmodel ذخیره کردیم و اکنون مستقیما ۶۰۰ داده فایل training را بررسی می کنیم ، که در حلقه یکی یکی این صحت را بررسی کرده و در نهایت درصد گیری می کنیم که چند درصد داده ها صحیح پیش بینی شده اند و انتظار داریم که عددی نزدیک به آنچه در classification learner با SVM ایجاد کردیم باشد.

```
>> p4
Data percentage label Estimated correctly ( for training.csv): 77.5%
>>
```

که همانگونه که انتظار می رفت عددی که بدست آمده دقت فاز آموزش بوده و مشابه عدد بخش قبل می باشد.

۴-۴:

اکنون برای راستی آزمایی مدل ایجاد شده ، اما اینبار برای فایل validation ، مشابه بخش قبلی با این تفاوت که تعداد کل داده هایمان ۱۰۰ نفر هستند کد را بازنویسی کرده و درصد صحت پیش بینی مان را می آزماییم.

```
b= trainedModel.predictFcn(diabetesvalidation);

correctPredictions = 0;

for i = 1:100
    if (a(i,1) == diabetesvalidation{i, 'label'})
        correctPredictions = correctPredictions + 1;
    end
end

% accuracy
accuracy = correctPredictions / 100 * 100;

disp(['Data percentage label Estimated correctly ( for validation.csv): ', num2str(accuracy), '%']);
```

```
>> p4
Data percentage label Estimated correctly ( for validation.csv): 50%
```

پایان