به نام خدا

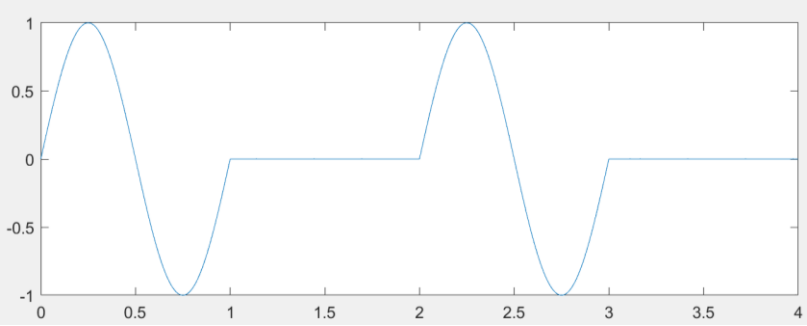
سیگنال ها و سیستم ها

تمرین کامپیوتری چهارم

مهلت تحویل: سه شنبه ۳۰ آبان ساعت **۱۷:۰۰**

بخش اول:

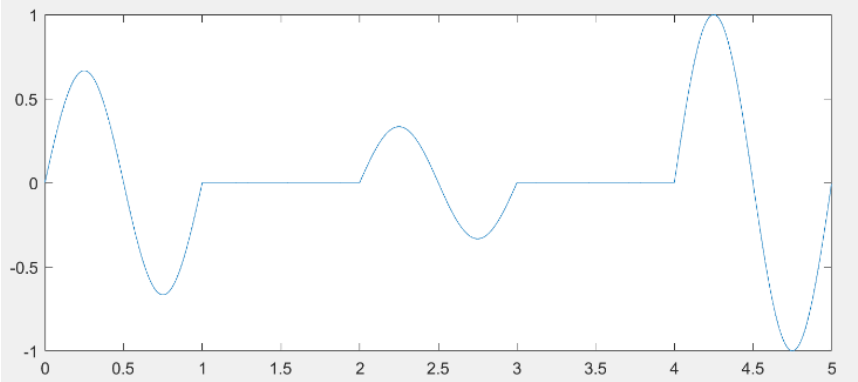
فرض کنید می خواهیم پیامی را به صورت بی سیم از طریق امواج برای شخصی ارسال کنیم. ابتدا پیام را در کامپیوتر به صورت یک رشته بیت باینری در آورده و سپس بیت ها را کدگذاری می کنیم. مثلا به جای هر بیت یک سیگنال خاص را ارسال می کنیم. فرض کنید به جای بیت صفر، سیگنال $x_0(t) = 0$ را به مدت یک ثانیه و به جای بیت یک، سیگنال $x_1(t) = \sin(2\pi t)$ را به مدت یک ثانیه ارسال کنیم. مثلاً رشته بیت 1010 به صورت زیر ارسال می شود:



فرض کنید قدرت فرستنده به گونه ای است که حداکثر دامنه ای که به سیگنال می تواند بدهد، دامنه ی یک است. همچنین فرکانس نمونه برداری سیگنال را $f_s = 100\,Hz$ در نظر بگیرید که باعث می شود هر سیگنال یک ثانیه ای شامل ۱۰۰ سمپل باشد. این فرض را تا آخر این تمرین کامپیوتری در نظر داشته باشید.

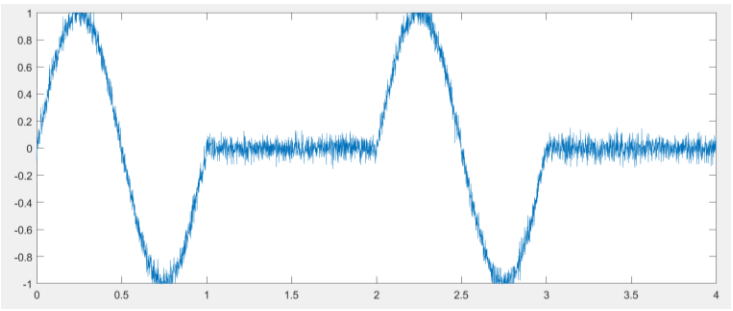
در گیرنده این سیگنال دریافت می شود. با فرض این که فرستنده و گیرنده از نظر زمانی کاملاً سنکرون باشند، برای رمزگشایی از پیام در گیرنده کافی است در هر یک ثانیه، correlation سیگنال دریافتی را با $2\sin(2\pi t)$ حساب کنیم. هر جایی correlation مقدار داشته باشد (مقدار $\int 2sin^2(2\pi t)\mathrm{dt} = 1$) می فهمیم بیت ارسالی برابر یک بوده و هر جا مقدار correlation صفر شود می فهمیم بیت ارسالی برابر صفر بوده است. توجه داشته باشید عدد 2 در دامنه ی $2\sin(2\pi t)$ هیچ اهمیتی ندارد و فقط برای رُند شدن مقدار correlation است.

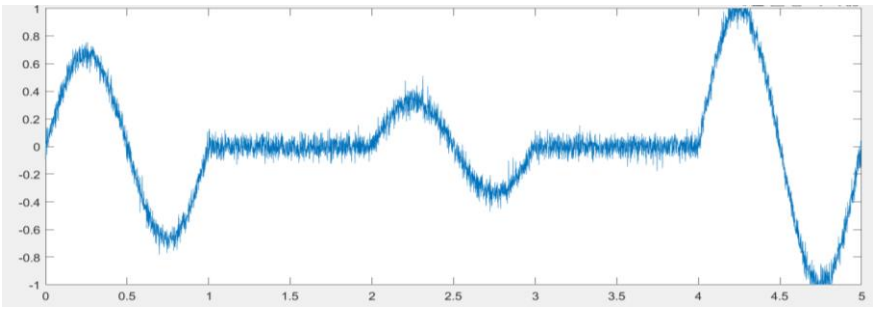
در سناریوی بالا سرعت ارسال پیام (bit rate) $1\ \frac{bit}{sec}$ است. حال فرض کنید بخواهیم bit rate را افزایش دهیم. برای این کار باید کدگذاری را روی بیت های بیشتری انجام دهیم. مثلا به جای هر دو بیت، یک سیگنال خاص را ارسال کنیم. برای ارسال 00 سیگنال $x_0(t) = 0$ را به مدت یک ثانیه، برای ارسال 01 سیگنال $x_1(t) = \frac{1}{3}\sin(2\pi t)$ را به مدت یک ثانیه، برای ارسال 10 سیگنال $x_2(t) = \frac{2}{3}\sin(2\pi t)$ را به مدت یک ثانیه و برای ارسال 11 سیگنال $x_3(t) = \sin(2\pi t)$ را به مدت یک ثانیه ارسال می کنیم. مثلا رشته بیت 1000010011 به صورت زیر ارسال می شود.



در گیرنده این سیگنال دریافت می شود. مجدداً برای رمزگشایی از پیام در گیرنده کافی است در هر ثانیه، correlation سیگنال دریافتی را با $2\sin(2\pi t)$ حساب کنیم و بسته به خروجی correlation که می تواند مقادیر صفر، $\frac{1}{3}$ ، $\frac{2}{3}$ و یک بگیرد راجع به بیت های ارسالی تصمیم گیری کنیم. در این سناریو سرعت ارسال پیام (bit rate) $2\ \frac{bit}{sec}$ است.

این روند را می توان به همین ترتیب ادامه داد و سرعت ارسال پیام را افزایش داد. یعنی هر سه بیت را کدگذاری کرد تا سرعت ارسال سه برابر شود و ... اما این اضافه شدن سرعت در عمل یک مشکل اساسی دارد که به ما اجازه نمی دهد سرعت را از حدی بیشتر کنیم. مشکل نویز است! وقتی فرستنده سیگنالی را به گیرنده ارسال می کند، در گیرنده علاوه بر سیگنال اصلی، نویز نیز به سیگنال اصلی اضافه می شود. مثلا در سناریوی اول و دوم سیگنال های زیر به گیرنده می رسد:





حال اگر correlation سیگنال دریافتی را با $2\sin(2\pi t)$ حساب کنیم، خروجی ها دقیقاً اعدادی که انتظار داشتیم نمی شوند و بنابراین ممکن است به اشتباه رمز گشایی کنیم. هر چه سرعت ارسال بیت (bit rate) بیشتر باشد این خطا بیشتر خواهد بود. چرا؟ قدری فکر کنید و بعد پاسخ این سوال را که در ادامه آمده است بخوانید.

به عنوان مثال سناریوی اول که سرعت ارسال $1\ \frac{bit}{sec}$ بود را در نظر بگیرید. انتظار داشتیم مقادیر خروجی correlation عدد صفر یا عدد یک باشد ولی در حضور نویز این اعداد کمی تغییر می کنند. پس به نظر می رسد باید عدد 0.5 را به عنوان threshold (آستانه) تعریف کنیم. اگر correlation بیش از 0.5 شد اعلام کنیم بیت مورد نظر یک بوده و اگر کمتر از 0.5 شد اعلام کنیم بیت مورد نظر صفر بوده است.

در سناریوی دوم که سرعت ارسال $2\ \frac{bit}{sec}$ بود، انتظار داشتیم مقادیر خروجی correlation عدد صفر، $\frac{1}{3}$ ، $\frac{2}{3}$ و یک شود، اما به خاطر وجود نویز این اعداد ممکن است هر مقداری بگیرند. بنابراین باید سه آستانه ی $\frac{1}{6}$ ، $\frac{3}{6}$ و $\frac{5}{6}$ را تعریف کنیم و با توجه به آنها راجع به بیت ها تصمیم گیری کنیم. چون فاصله ی این اعداد از هم کمتر شده است بنابراین احتمال خطا در تصمیم گیری بیشتر می شود.

پس در محیط عملیاتی، در روش پیشنهاد شده، یک trade off بین افزایش سرعت انتقال پیام و مقاوم بودن نسبت به نویز وجود دارد. حال می خواهیم مفاهیم بیان شده در این بخش را شبیه سازی کنیم.

هدف این تمرین ارسال پیام به زبان انگلیسی از فرستنده به گیرنده است. هر پیام فقط شامل حروف کوچک انگلیسی، فاصله، نقطه، ویرگول، علامت تعجب، سمی کالن (;) و کوتیشن (") است. بنابراین در مجموع ۳۲ کاراکتر داریم. به هر کاراکتر ۵ بیت مرتبط می کنیم.

تمرین ۱-۱) یک سلول به اسم Mapset با ابعاد 32×2 درست کنید. در سطر اول خود کارکترها را قرار دهید و در سطر دوم ۵ بیتی که به آنها مرتبط کردید را قرار دهید. به عنوان مثال شکل زیر قسمت ابتدایی Mapset را نشان می دهد. (دستور dec2bin برای باینری کردن اعداد کمک کننده خواهد بود.)

	1	2	3
1	'a'	'b'	'c'
2	'00000'	'00001'	'00010'

تمرین ۱-۲) تابعی (*function*) به نام *coding_amp* بنویسید که ورودی های آن ۱) پیام مورد نظر برای ارسال و ۲) سرعت ارسال اطلاعات باشد و خروجی آن پیام کدگذاری شده باشد.

راهنمایی: مثلا فرض کنید می خواهیم پیام *bc* را با سرعت $1\ \frac{bit}{sec}$ ارسال کنیم. ابتدا حروف پیام را به صورت باینری در آورید که (با توجه به شکل بالا) می شود: *0000100010* ، سپس رشته باینری به دست آمده را مطابق آنچه در مقدمه توضیح داده شد کدگذاری کنید. توجه داشته باشید پیام ممکن است هر کلمه یا جمله ای با طول دلخواه شامل ۳۲ کاراکتر ذکر شده باشد. در بخش های بعدی بیشترین سرعت ارسال اطلاعات را ۳ بیت برثانیه در نظر خواهیم گرفت گرچه کد شما می تواند هر عدد طبیعی دلخواهی را پشتیبانی کند! (دستورات *extract* و *strlength* برای جداسازی کاراکترهای پیام کمک کننده خواهند بود.)

توجه: فرض کنید طول رشته بیت با سرعت اطلاعات همخوانی دارد به این معنی که پیش نمی آید طول رشته بیت بر تعداد بیت ارسالی در هر ثانیه تقسیم پذیر نباشد!

تمرین ۱-۳) خروجی تابع *coding_amp* را برای پیام (کلمه‌ی) *signal* با سرعت ارسال اطلاعات یک، دو و سه بیت بر ثانیه به صورت جداگانه رسم کنید.

تمرین ۱-۴) تابعی به نام *decoding_amp* بنویسید که ورودی های آن ۱) پیام کدگذاری شده (سیگنال زمانی تولید شده در قسمت قبل) و ۲) سرعت ارسال اطلاعات باشد و خروجی آن پیام رمز گشایی شده باشد. تابعی که نوشتید را روی همان پیام *signal* که در قسمت ۱-۳ با سرعت ارسال های مختلف کُد کردید تست کنید تا مطمئن شوید کدتان درست کار می کند.

توجه: اگر می خواهید نتایج *correlation* کاملاً شبیه آنچه که در مقدمه بیان شد باشد، عدد *0.01* را پشت *correlation* ضرب کنید. چون به صورت گسسته *correlation* را حساب می کنید و تعداد نمونه ها در هر بازه‌ی *correlation* گیری ۱۰۰ نمونه است این اتفاق رخ می دهد.

تمرین ۱-۵) در این قسمت می خواهیم به سیگنال دریافتی در گیرنده نویز اضافه کنیم تا شبیه سازی مشابه شرایط واقعی شود. برای تولید نویز از دستور *randn* استفاده کنید. این دستور نویز گوسی با میانگین صفر و واریانس یک تولید می کند. نشان دهید خروجی این دستور یک نویز الف) گوسی، ب) با میانگین صفر و ج) با واریانس یک است. برای مثال این سه نکته را روی خروجی *randn(1,3000)* نشان دهید.

تمرین ۱-۶) برای زیاد یا کم کردن قدرت نویز کافی است در پشت دستور *randn* عدد دلخواه σ (بدون توان ۲) را ضرب کنید. این عدد کاری می کند تا واریانس نویز σ^2 شود. به پیام *signal* بعد از این که کدگذاری شد نویز گوسی با واریانس *0.0001* و میانگین صفر اضافه کنید و بعد آن را *decode* کنید. آیا بازهم پیام *signal* استخراج شد؟ نتایج *decoding* را برای هر سه *bit rate* یک، دو و سه به صورت جداگانه گزارش کنید.

تمرین ۱-۷) قدرت نویز را کم کم و طی چندین مرحله افزایش دهید و هر بار قسمت ۱-۶ را تکرار کنید. مشاهدات خود را گزارش کنید. کدام یک از سه *bit rate* به نویز مقاوم تر بودند؟ آیا نتایج با آنچه در مقدمه بیان شد همخوانی داشتند؟

تمرین ۱-۸) طی شبیه سازی هایی که در قسمت ۱-۷ انجام دادید، بیشترین واریانس نویز که *bit rate* سه به آن مقاوم بود به صورت تقریبی چند بود؟ برای *bit rate* دو و یک چه طور؟

تمرین ۱-۹) به نظر شما با چه راهکاری می توان *bit rate* را نسبت به نویز مقاوم تر کرد؟ پاسخ خود را در هایلات قرمز جستجو کنید. حال در ادامه پاسخ را مطالعه کنید.

اگر قدرت فرستنده ی ما بیشتر می بود می توانستیم دامنه ی سیگنال بیشتری داشته باشیم و بنابراین فاصله ی *threshold* هایی که برای تصمیم گیری در نظر گرفته بودیم بیشتر می شد و در نتیجه به نویز حساسیت کمتری ایجاد می شد. مثلا برای ارسال اطلاعات با سرعت $2\ \frac{bit}{sec}$ ، اگر قدرت فرستنده به گونه ای می بود که بیشینه ی دامنه ی سیگنال سه می شد، شرایط بهتر می شد. به عبارت دیگر، برای ارسال 00 سیگنال $x_0(t) = 0$ را به مدت یک ثانیه، برای ارسال 01 سیگنال $x_1(t) = \sin(2\pi t)$ را به مدت یک ثانیه، برای ارسال 10 سیگنال $x_2(t) = 2\sin(2\pi t)$ را به مدت یک ثانیه و برای ارسال 11 سیگنال $x_3(t) = 3\sin(2\pi t)$ را به مدت یک ثانیه ارسال می کردیم.

پس در کل می توان نتیجه گرفت در روش کدگذاری دامنه اگر می خواهیم سرعت ارسال اطلاعات را افزایش دهیم باید *power* بیشتری نیز مصرف کنیم تا در برابر نویز مقاوم باشیم.

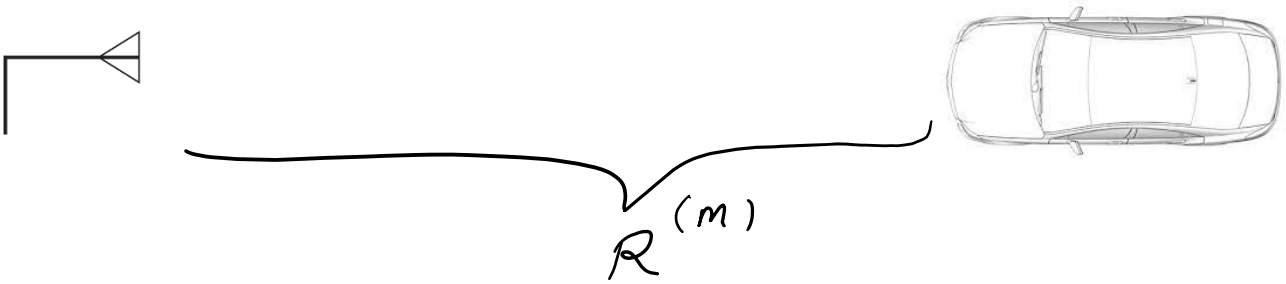
تمرین ۱-۱۰) به نظر شما در کدینگ دامنه، اگر نویز نباشد، بیت ریت را تا کجا می توان افزایش داد؟

تمرین ۱-۱۱) در کدینگ دامنه با فرض این که همچنان بیشترین دامنه ی ارسالی فرستنده برابر یک است، اگر سیگنال دریافتی را به جای $2\sin(2\pi t)$ در $10\sin(2\pi t)$ ضرب کنیم، آیا عملکرد روش به نویز مقاوم تر می شود؟ توضیح دهید.

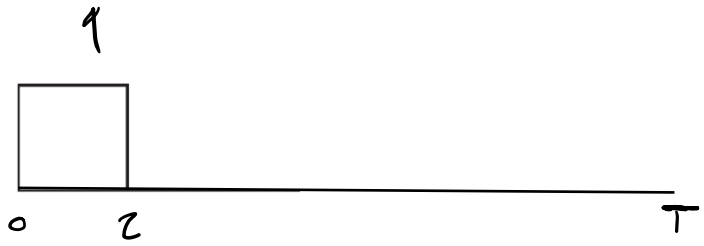
تمرین ۱-۱۲) سرعت اینترنت ADSL خانگی معمولاً چند $\frac{bit}{sec}$ است؟ ما در این تمرین با چه سرعتی اطلاعات را ارسال کردیم ؟!!!

بخش دوم:

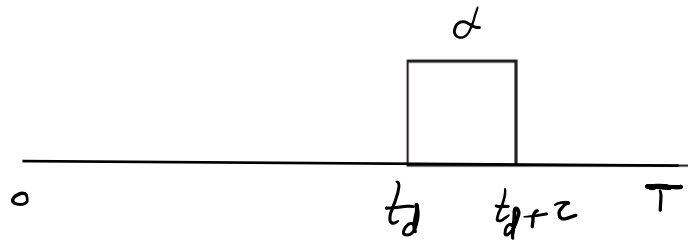
می خواهیم توسط یک رادار، فاصله ی یک جسم از رادار را بیابیم.



سیگنال ارسالی رادار به صورت زیر است:



این سیگنال پس از برخورد به جسم و بازگشت به سمت رادار، به صورت زیر دریافت می شود:



که در شکل بالا $t_d = \frac{2R}{C}$ است و مقدار $\alpha < 1$ اهمیتی ندارد.

پارامتر مجهول مساله t_d می باشد زیرا با به دست آوردن آن، فاصله نیز مشخص می شود. برای به دست آوردن این پارامتر، ایده ی correlation یا template matching مطرح شد و در تمارین کامپیوتری قبلی، این مساله را حل کردیم. در این تمرین می خواهیم با استفاده از مفهوم کانوولشون همان روش را پیاده سازی کنیم.

تمرین ۲-۱) سیگنال ارسالی را با فرضیات زیر تولید کرده و رسم کنید (t_s فاصله ی دو نمونه ی زمانی و یا عکس فرکانس نمونه برداری f_s است).

```
ts=1e-9;T=1e-5;tau=1e-6;
```

سپس سیگنال دریافتی را با فرض $R = 450\,m$ تولید کرده و رسم کنید ($\alpha = 0.5$). حال به صورت برعکس به مساله نگاه کنید. یعنی فرض کنید سیگنال دریافتی را داریم و می خواهیم از روی آن فاصله را محاسبه کنیم.

تمرین ۲-۲) آیا می توان با استفاده از مفهوم کانوولشون روش پیشنهادی در تمارین کامپیوتری قبلی را پیاده سازی کرد؟ با استفاده از دستور conv ، آن را پیاده سازی کنید.

تمرین ۲-۳) حال به سیگنال دریافتی کمی نویز اضافه کنید و تمرین ۲-۲ را تکرار کنید. قدرت نویز را کم کم و طی چندین مرحله افزایش دهید و ببینید تا کجا همچنان می توانید فاصله را به درستی تشخیص دهید. فرض کنید اگر خطای فاصله یابی کمتر از ۱۰ متر باشد، بگوییم فاصله درست تخمین زده شده است.

بخش سوم:

در این تمرین می خواهیم تابعی با نام calling_customer بنویسیم که عددی بین ۱ تا ۹۹ و شماره ی باجه (بین ۱ تا ۹) را دریافت کند و سپس شماره را مشابه دستگاه های فراخوان مشتری در بانک ها اعلام کند:

مثلاً: با اجرای دستور calling_customer(25,2) "شماره ی بیست پنج به باجه ی دو" اعلام شود.

نکات زیر را برای حل این تمرین در نظر داشته باشید:

- اگر هر یک از اعداد ورودی در بازه های مذکور نبودند پیغام خطا در command چاپ شود.
- صدای تولید شده حتما باید صدای شما باشد.
- می بایست یک mapset شامل موارد زیر از صدای شما ضبط و تشکیل شود و در نهایت از ترکیب آنها پیام اعلام شود:

۱- کلمات "شماره ی" و "به باجه ی"

۲- اعداد یک تا نوزده – اعداد بیست، سی ، ... نود

۳- صدای "و"

- حتما mapset و تابع را آپلود کنید تا بتوان کد شما را تست کرد.

بخش چهارم:

یک دیتاست با نام diabetes-training در اختیار شما قرار داده شده است. این دیتاست را در محیط matlab وارد کنید. با دبل کلیک کردن روی آن و زدن کلید import selection دیتاست وارد workspace می شود. همان طور که مشاهده می کنید این دیتاست شامل شش اطلاعات و علامت پزشکی (یا فیچر) اندازه گیری شده از ۶۰۰ نفر مختلف است. ستون آخر نشان دهنده ی این است که فرد دیابت دارد (برچسب ۱) و یا خیر (برچسب ۰).

می خواهیم به یک ماشین آموزش دهیم که چگونه با استفاده از شش ویژگی استخراج شده از هر فرد تصمیم بگیرد که فرد مورد نظر دیابت دارد یا خیر.

تمرین ۴-۱) اپلیکیشن Classification Learner را اجرا کرده و داده ها را با استفاده از آیکون New Session وارد اپلیکیشن کنید. هنگام ورود داده ها به تنظیمات پیش فرض از جمله cross validation folds و ... دست نزنید. بعد از لود داده ها در فضای اپلیکیشن، از منوی بالای صفحه طبقه بند Linear SVM را انتخاب کنید و در نهایت روی گزینه Train کلیک کنید. دقت به دست آمده روی این داده ها را گزارش کنید.

تمرین ۴-۲) با استفاده از گزینه ی Feature Selection یکی یکی فیچر ها را به تنهایی انتخاب کنید و مجدد گزینه ی Train را کلیک کنید و برای هر فیچر دقت به دست آمده را گزارش کنید (شش عدد مختلف). کدام یک از ویژگی ها به دیابتی بودن فرد بیشتر ربط دارد؟

تمرین ۴-۳) ماشین آموزش دیده (با استفاده از همه ی فیچرها) را با استفاده از گزینه ی Export Model در Workspace با نام TrainedModel ذخیره کنید. حال فرض کنید می خواهیم با ماشین آموزش دیده برچسب داده های diabetes-training را مجدد تخمین بزنیم و خودمان دقت ماشین آموزش دیده را بسنجیم. با استفاده از TrainedModel و آرگومان predictFcn آن، برچسب داده ها را تخمین بزنید. برچسب چند درصد داده ها درست تخمین زده شد؟ این عدد دقت فاز آموزش است.

توجه داشته باشید جواب شما باید با جواب تمرین ۴-۱ مطابقت داشته باشد. همچنین برای محاسبه ی دقت اگر احتیاج شد می توانید ستون آخر Table داده ها را با استفاده از دستور table2array به array تبدیل کنید تا بتوانید از آن برای محاسبات استفاده کنید.

تمرین ۴-۴) فایلی با نام diabetes-validation در اختیار شما قرار داده شده است که فیچرهای ۱۰۰ نفر جدید است و در ستون آخر آن دیابتی بودن و یا نبودن هر فرد مشخص شده است. با استفاده از ماشین آموزش دیده در قسمت قبل، برچسب این داده ها را تخمین بزنید. برچسب چند درصد داده ها درست تخمین زده شد؟ این عدد دقت فاز تست یا ارزیابی است.

نکات کلی:

• درصورت وجود هرگونه پرسش و ابهام به محمد امین کشمیری و استاد ایمیل بزنید.

• فایل نهایی شما باید به صورت یک فایل زیپ شامل گزارشکار به فرمت PDF و کد های متلب باشد.