

مبانی مهندسی پزشکی

پروژه دوم

استاد: دکتر بدیعی

محمد مهدی رحیمی

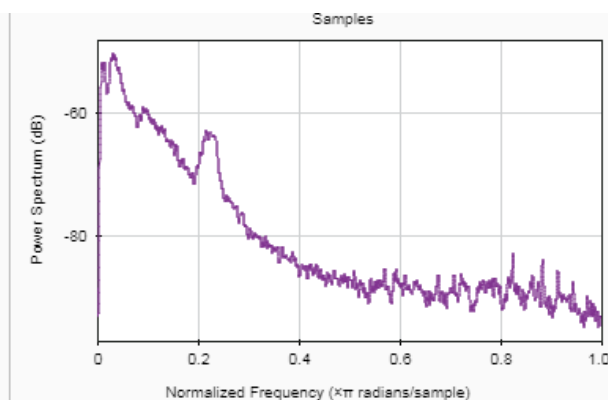
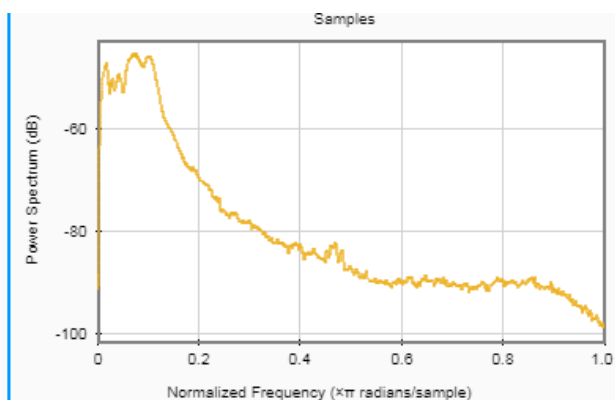
۸۱۰۱۹۷۵۱۰

• توضیحات مثال

در این پروژه ما از ماشین لرنینگ استفاده خواهیم کرد تا داده هایی که داریم را طبقه بندی کنیم. به طور خلاصه روند پروژه به این گونه است ما از تعدادی داده که صدای قلب می باشد استفاده می کنیم که این صدای قلب یا صدای طبیعی می باشد یا صدای قلبی که درست کار نمی کند می باشد. حال ما تعداد داده ای که شامل صدای های طبیعی و صدای هایی از عملکرد نادرست است را به برنامه داده و از وضعیت آن ها با خبر هستیم و حال با توجه به ویژگی هایی که هر کدام از دسته ها دارند دارند الگویی برای آن ها میابیم و برای پیدا کردن این الگو میتوان از الگوریتم ها و کدهای مختلف استفاده کرد و هرکدام نتیجه خد را دارد و پس از پایان دسته بندی داده هایی را به عنوان تست به آن می دهیم و با توجه به اینکه خودمان داده های تست را میدانیم میزان حساسیت و اختصاصیت قابل محاسبه است. حال توضیحات را با جزئیات بیشتر به شرح زیر داریم:

ماشین لرنینگ به چند دسته تسیم می شود که با توجه به خروجی می توان اینگونه دسته بندی کرد که یا بر اساس نوع داد ها جدا سازی می شوند و خروجی آن با توجه به ورودی داده شده در یکی از گروه های پیش بینی شده قرار میگیرد یا خطی و نموداری برازش می کنیم تا با دادن ورودی مقدار دقیقی به دست آوریم و مدل دیگر نیز پیدا کردن یک الگو می باشد که خروجی خاصی ندارد. که برای این پروژه نوع اول که classification است استفاده می شود چون یا صدا طبیعی بوده یا صدای عملکرد نا درست است.

حال در ابتدای کد داده هایی را از سایتی دانلود کرده به عنوان صدا های طبیعی و نادرست و حال نمونه ای از هرکدام را رسم کرده و اجرا می کند. در بخش بعد از ابزاری استفاده می شود که آنالیز کننده سیگنال می باشد که با استفاده از این ابزار می توان سیگنال و PSD آن را مشاهده کرد و با توجه به خروجی که نمایش می دهد بسیار واضح است که در صدای عملکرد نادرست در قسمتی از نمودار پیک وجود دارد که نشان می دهد این صدا در فرکانس های خاصی مقدار بیشتری دارد و همانطور که در شکل خود سیگنال نیز می توان دید بین دو تپش در صدای غیر طبیعی سیگنال هایی مانند نویز وجود دارد که احتمالاً همین پیک را به وجود آورده اند. که در شکل زیر می بینیم. (شکل سمت راست عملکرد نادرست است که پیک آن به طور واضح معلوم است)



سپس داده هایی که داریم را با توجه به لیبل آن ها خوانده. یعنی داده ها را با توجه به نوع آن ها وارد نرم افزار می کنیم. حال به سراغ تحلیل و بررسی ویژگی های داده ها می رویم که در اینجا منظورمان همان features می باشد و برای محاسبه این مشخصات تابعی وجود دارد که به عنوان مثال میانگین میانه و... محاسبه میشود.

حال با توجه به این ویژگی هایی که از سیگنال ها دریافت کردیم به مرحله بعدی پروژه ماشین لرنینگ می رویم . در این مرحله با توجه به ویژگی های یافته شده داده های را با روش ها یا الگوریتم های مختلف می توان این کار را کرد . در این مرحله با تعدادی از داده ها مدل مورد نظر را پیدا کرده و با باقی داده ها مدل را تست می کنیم. و می توان میزان حساسیت و اختصاصیت و ... را در مدل با توجه به جدول confusion matrix بررسی کرد. و مثال هایی از مدل های استفاده شده مانند Tree, KNN, SVM و... می باشد که هر کدام خصوصیت و مزایا و معایب خود را دارد.

حال با توجه به این که دیدیم مدل های مختلفی برای دسته بندی داده ها وجود دارد می توان با روش هایی مانند بهبود مدل های استفاده شده با تغییر پارامتر های آن مدل و یا استفاده از تعداد بیشتری داده برای آموزش یا استفاده از feature های بهینه تر یا تغییر روش محاسبه ان ها.

خب حال داده ها را تقسیم کرده تا تعدادی برای آموزش باشد و تعدادی برای تست آن باشد که به راحتی این کار را می کنیم. و در مرحله بعد آموزش را به نحوه ای تعیین می کنیم که بیشتر به این توجه شود که صدای غیر طبیعی با دقت بیشتری شناخته شود. و اگر در صدای طبیعی خطایی بود چشم پوشی می شود . و برای استفاده از مدل های مختلف پارامتر هایی وجود دارد که به تغییر آن ها می توان عملکرد را تغییر داد و باید توجه داشت که باید بهترین پارامتر ها انتخاب شوند و برای انتخاب این پارامتر ها میتوان از خود متلب استفاده کرد تا این پارامتر ها را پیدا کند یا در اصل tiun کند. در این بخش پس از پایان یافتن پارامتر ها مطابق انتظار عملکرد خوبی در یافتن صدا های غیر طبیعی دارد.

بعد از این مرحله از NCA استفاده کرده که روشی است برای بهبود و پیدا کردن features و این روش تعدادی از feature هایی که بیشتر در دسته بندی ها مورد توجه است و تاثیر دارد انتخاب می شود و برای جلوگیری از زیاد شدن حجم اطلاعات این کار را می کنیم. برای مثال در اینجا از ۷۱ feature که داشتیم فقط ۱۱ تای ان انتخاب می شود. زیرا این feature ها داده های به درد بخوری دارند.

حال در مرحله بعد برای با توجه به feature های انتخاب شده مرحله قبلی یعنی آموزش و پیدا کردن پارامتر را انجام می دهیم و این دفعه هم نتایج قابل قبول تر هستند هم سریع تر محاسبات انجام میشود زیرا حجم داده را کم کرده و تنها از اطلاعاتی که مفید هستند استفاده کردیم.

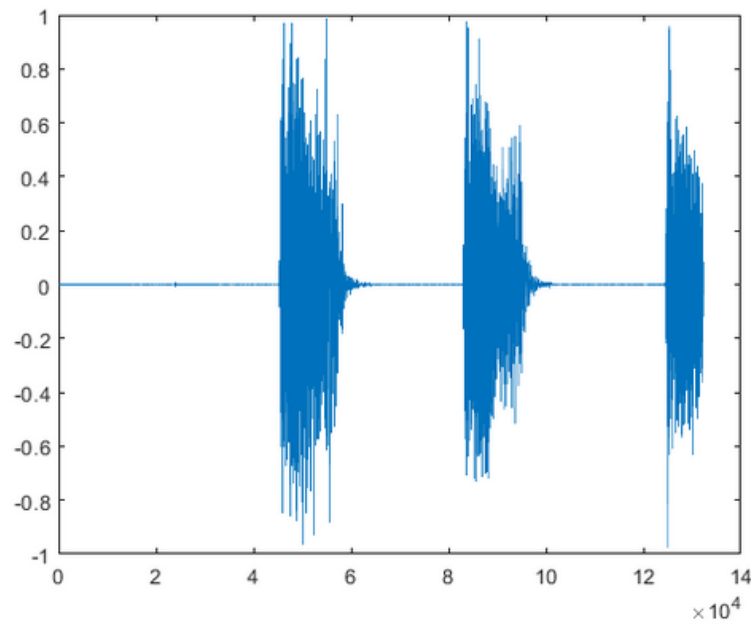
در انتها نیز کد را به زبان دیگری مانند C خروجی گرفته تا در دستگاه ها و جاهای مختلف دیگر قابل استفاده باشد.

اگر بخواهیم به طور خلاصه بیان کنیم ابتدا تعدادی داده که می دانیم وضعیت آن ها به چه شکلی است را به برنامه داده و ویژگی هایی از سیگنال های داده شده استخراج می کنیم و با توجه به این ویژگی ها شروع به دسته بندی میکنیم. و برای بهبود عملکرد از کدی استفاده می کنیم که خود متلب آن ویژگی هایی که بیشترین تاثیر را دارند استفاده کرده تا حجم داده کمتر شود سرعت پردازش بیشتر شود و عملکرد نیز بهبود یابد و برای تعیین پارامتر های مدل استفاده شده خود متلب این پارامتر ها را پیدا می کند. و در نهایت با این روش به نتیجه مطلوبی می رسیم.

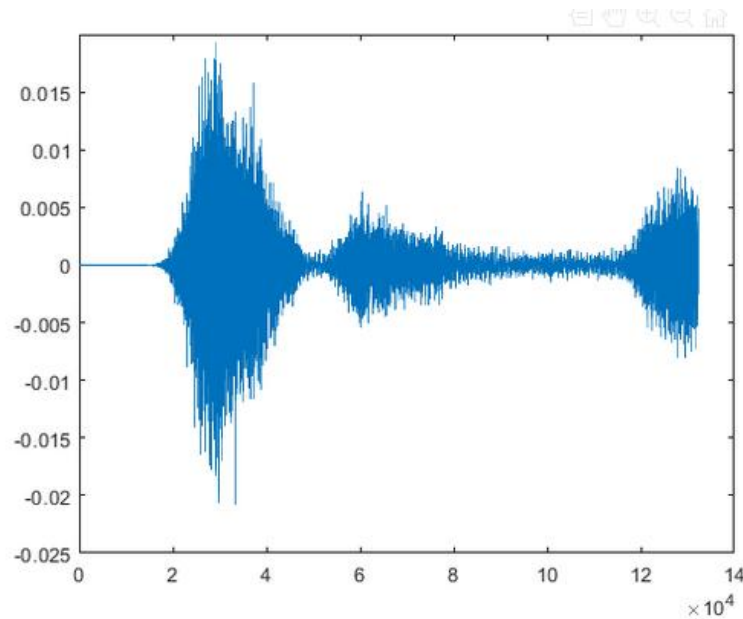
• تکرار پروژه با دیتاست جدید

برای این بخش دیتاست ایجاد کرده ایم که در آن ۲۵ صدای سرفه و ۲۵ صدای نفس کشیدن ضبط کردیم. حال مطابق کد اصلی مراحل را تکرار می‌کنیم. یعنی ابتدا یک فایل از هر کدام از داده‌ها را اجرا کرده و آن را در نمودار رسم می‌کنیم و نمودار آن مطابق شکل می‌باشد.

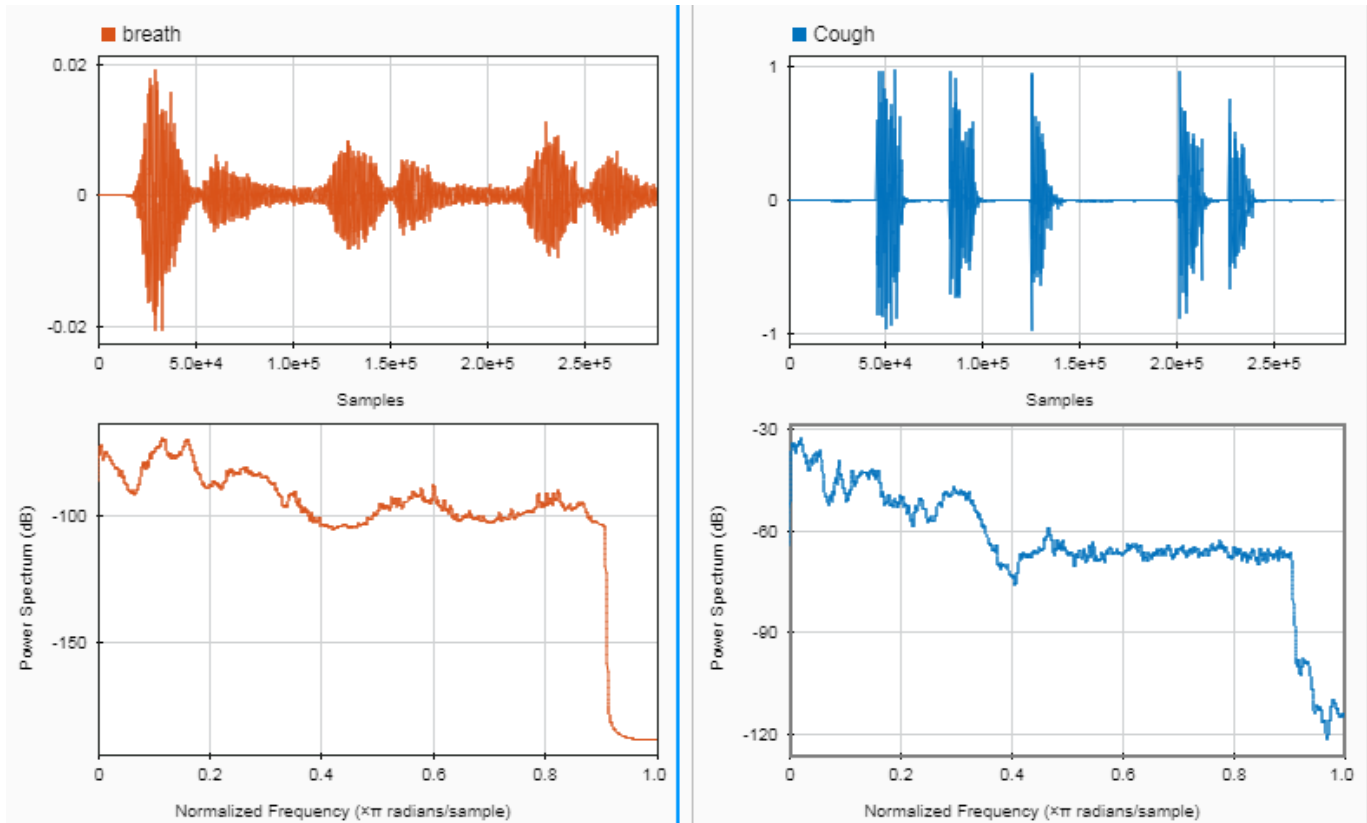
برای سرفه داریم:



برای نمودار نفس کشیدن داریم:



حال با استفاده از ابزار آنالیز سیگنال متلب داریم:



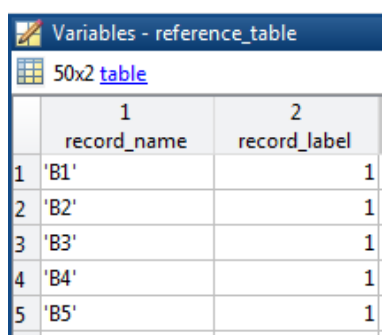
به طور واضح تفاوت ها دیده می شود در قسمت هایی پیک وجود دارد و در جاهایی نمودار مقدار کمی دارد.

بعد از این مرحله نوبت وارد کردن داده ها به نرم افزار می باشد و باید فایل اکسلی ایجاد کرد که در آن نام فایل ها با لیبل برای آن ها وجود داشته باشد. فایل هایی با نام های B و لیبل ۱ برای فایل های نفس بوده و فایل های S با لیبل ۱- برای سرفه می باشد. سپس باید فایل با پسوند cvs ذخیره شود در پوشه دیتاست. در قسمتی از کد که مربوط به وارد کردن اطلاعات است (خط ۲۸ کد اصلی) مشکلی وجود دارد. که رفع مشکل به شرح زیر است: (باید در ماتریس reference_table این تغییر را انجام داد)

زمانی که کد اجرا شود داده به شکل زیر خوانده می شود:

Variables - reference_table		
50x2 table		
	1 record_name	2 record_label
1	'i»B1'	1
2	'B2'	1
3	'B3'	1
4	'B4'	1
5	'B5'	1

همانطور که مشاهده می شود داده اول اشتباه خوانده می شود. بنده سعی کردم مشکل را حل کنم اما به دلیل کمبود وقت از آن صرف نظر کرده و به طور دستی به شکل زیر تغییر می دهیم:



	1	2
	record_name	record_label
1	'B1'	1
2	'B2'	1
3	'B3'	1
4	'B4'	1
5	'B5'	1

اگر این داده اصلاح نشود کد به مشکل خواهد خورد البته اگر فایل FeatureTable.mat وجود داشته باشد که در فایل نیز ضمیمه خواهد شد مشکلی در این بخش نخواهیم داشت.

در تابع extractFeatures.m نیز تغییر زیر را انجام می دهیم:

```
13 - keySet = {-1, 1};
14 - valueSet = {'cough', 'breath'};
```

پس از وارد کردن اطلاعات به صورت صحیح حال feature ها را محاسبه می کنیم البته برای سه تا از آن ها قابل محاسبه نمی باشد بنابراین آن ها را محاسبه نکرده که عبارت اند از:

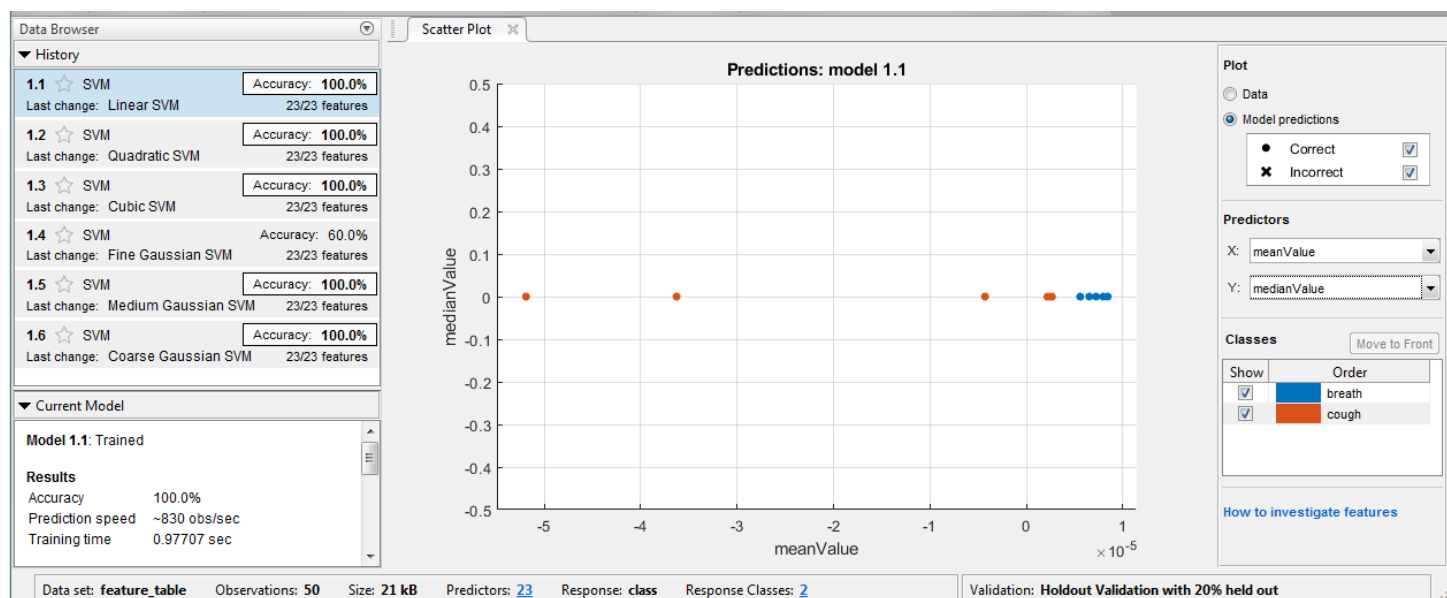
```
spectralEntropy , dominantFrequencyValue , dominantFrequencyMagnitude
dominantFrequencyRatio
```

که از محاسبه آن ها صرف نظر می کنیم. اگر جدول features وجود نداشته باشد بعد از مدت کوتاهی به دست می آید که قسمتی از آن به شرح زیر است:

meanValue	medianValue	standardDeviation	meanAbsoluteDeviation	quantile25
9.0422e-06	0	0.0016485	0.00089877	-0.00043208
7.9631e-06	0	0.0010394	0.00064907	-0.0003777
5.5792e-06	0	0.0015738	0.00086774	-0.00042685
7.7862e-06	0	0.00077686	0.00051864	-0.00035693
8.2667e-06	0	0.00089229	0.00057521	-0.00038503

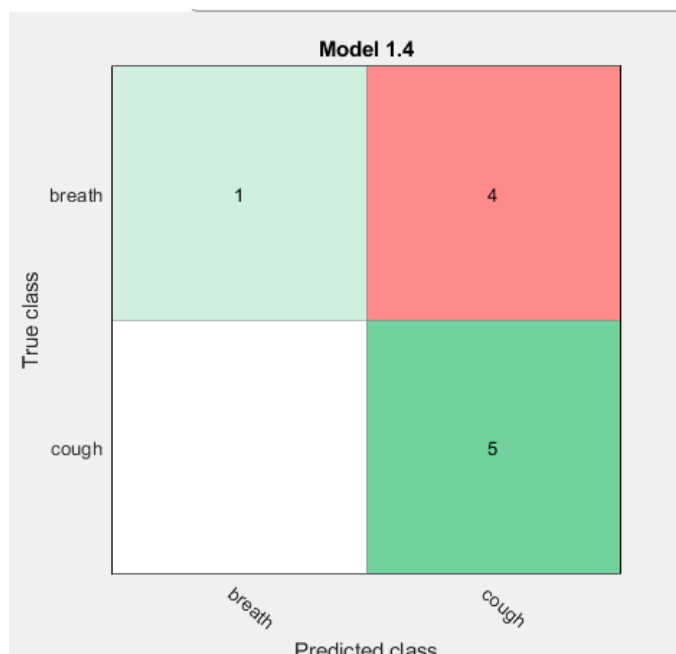
تعداد feature های محاسبه شده ۲۳ عدد می باشد.

حال مانند مثال حل شده از ابزار classificationLearner استفاده کرد تا با توجه به feature های به دست آمده داده ها را دسته بندی کند. مانند مثال اگر مدل را بر روی All SVMs قرار دهیم نتیجه مطابق زیر خواهد شد:

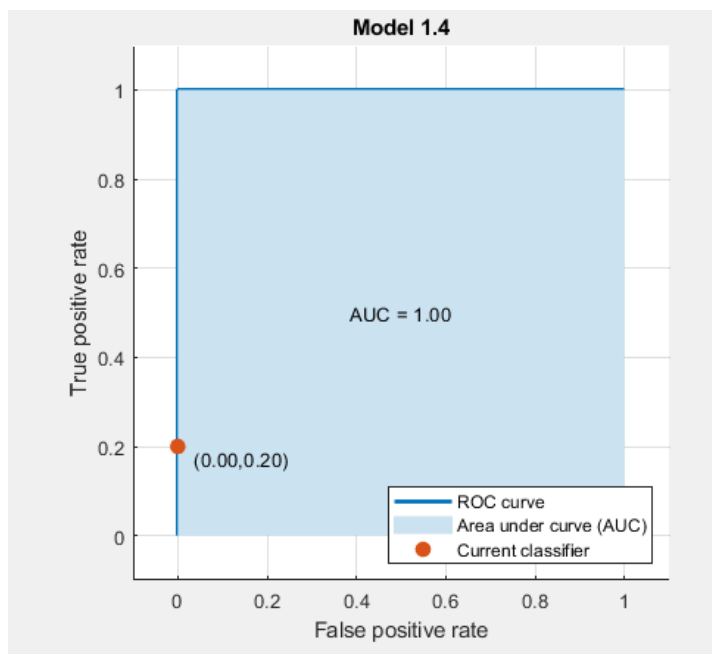


با توجه به خواسته صورت پروژه باید از ۵ عدد به عنوان تست استفاده کرد بنابراین با توجه به اینکه ۵۰ داده داریم اگر ۲۰ درصد داده ها برای تست نگه داشته شوند تعداد داده هایی که برای تست نگه داشته شده اند برابر است با ۵ تا از هر کدام از دسته ها.

نمودار مدل های ۱۰۰ درصد که به شکل کامل است اما برای حالت ۶۰ درصد داریم:

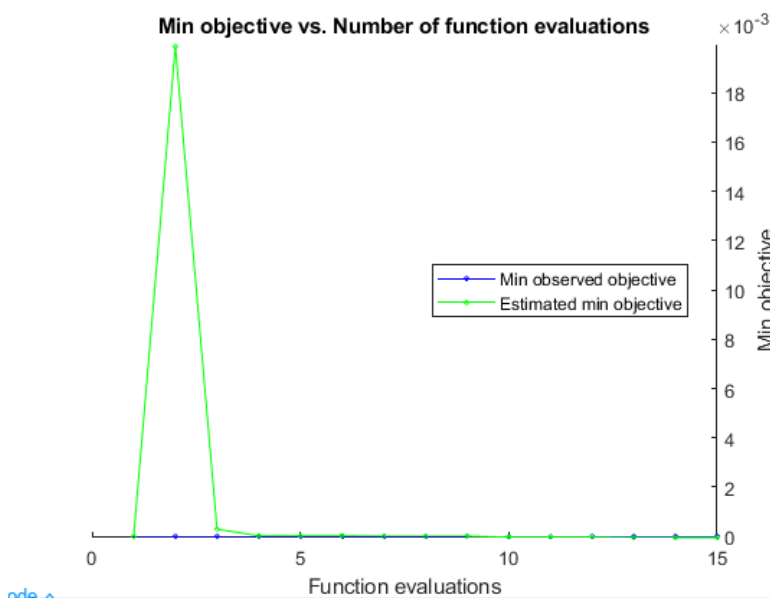


و نمودار ROC آن به شکل زیر است:



سپس با استفاده از تابع `splitDataSet` به تعداد ۴۰ تا از داده ها را برای آموزش جدا کرده و ۱۰ تای آن را برای تست نگه میداریم.

حال به آن قسمتی از کد می رسیم که در مثال نیز آن را تکرار کردیم. در این بخش پیدا کردن پارامترها یا همان `tiuning` را با استفاده از متلب انجام داده و داریم:

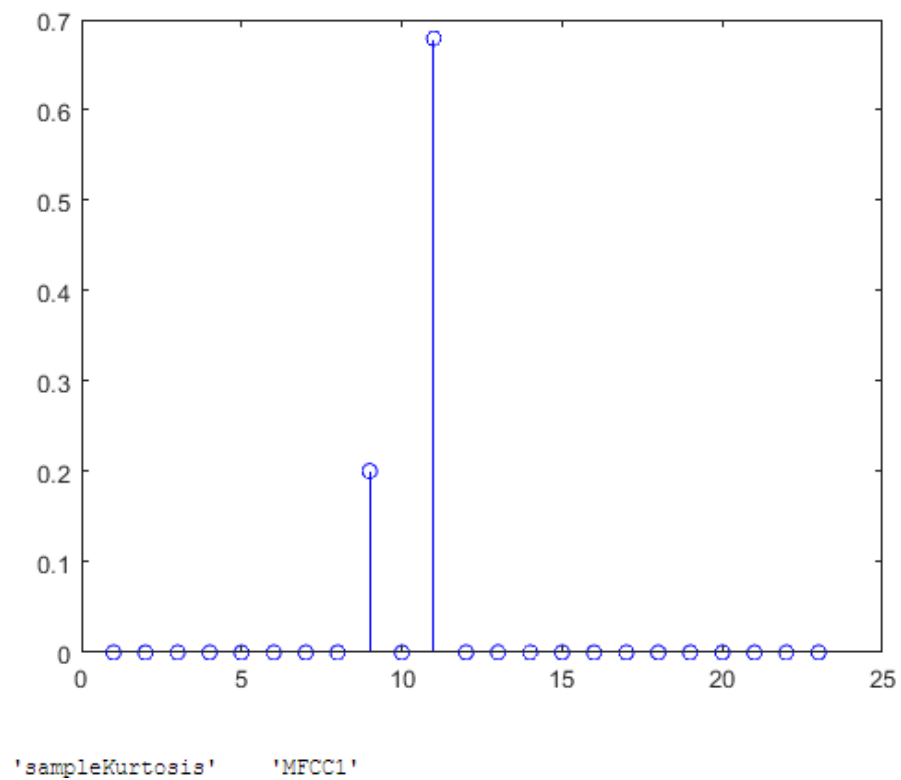


Iter	Eval	Objective	Objective	BestSoFar	BestSoFar	Method	NumLearningC-	LearnRate
	result		runtime	(observed)	(estim.)		ycles	
1	Best	0	121.55	0	0	Bag	421	-
2	Accept	0.5	2.9733	0	0.01988	RUSBoost	277	0.0028086
3	Accept	0	5.1355	0	0.00029257	GentleBoost	13	0.0099514
4	Accept	0.5	1.1516	0	2.4999e-05	RUSBoost	42	0.27696
5	Accept	0	7.0044	0	1.9989e-05	LogitBoost	27	0.0012786
6	Accept	0.5	1.5003	0	2.4978e-05	AdaBoostM1	188	0.017091
7	Accept	0	96.685	0	1.1895e-05	GentleBoost	499	0.0021077
8	Accept	0	2.8088	0	1.1151e-05	Bag	10	-
9	Accept	0.025	47.692	0	1.0544e-05	LogitBoost	239	0.9984
10	Accept	0	58.907	0	-2.8056e-05	GentleBoost	289	0.99824
11	Accept	0	20.004	0	-2.8058e-05	LogitBoost	112	0.0010041
12	Accept	0	18.538	0	-2.9582e-05	GentleBoost	108	0.99341
13	Accept	0	6.1928	0	-4.9048e-05	GentleBoost	27	0.0010004
14	Accept	0	7.6894	0	-4.8979e-05	LogitBoost	42	0.0010075
15	Accept	0	114.52	0	-4.8917e-05	Bag	488	-

cough	100	0
breath	0	100
	cough	breath

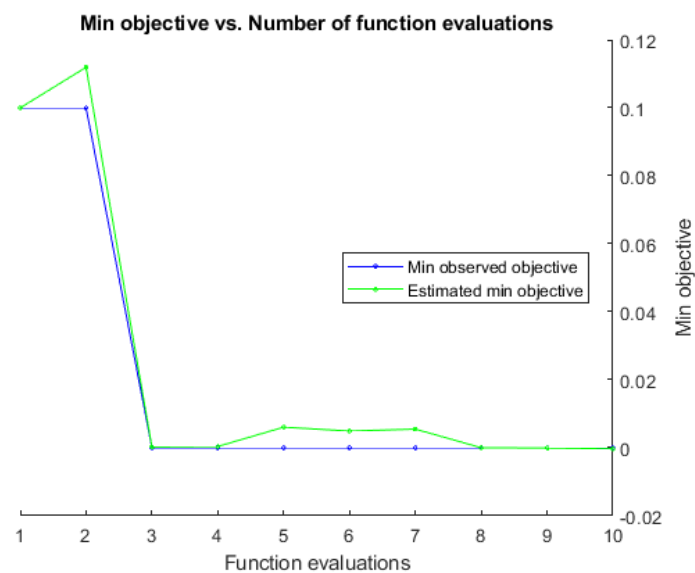
خروجی بسیار مطلوب می باشد اما برای اینکه پروژه کامل باشد بقیه مراحل را نیز طی می کنیم.

در مرحله بعد باید feature هایی که مناسب هستند و بیشترین دقت و بیشترین فایده را دارند پیدا کنیم و برای این کار از خود متلب استفاده می کنیم و در این بخش بهترین feature ها استخراج می شوند تا در مرحله بعد از آن ها برای train استفاده شود. در اصل این بخش استفاده از NCA می باشد که خروجی آن به شرح زیر است:



حال دوباره با استفاده از feature های مشخص شده به سراغ train و پیدا کردن پارامتر ها می رویم و با خود متلب پارامتر ها را tiun می کنیم.

در این مرحله با feature های انتخاب شده داریم:



Iter	Eval	Objective	Objective	BestSoFar	BestSoFar	Me
	result		runtime	(observed)	(estim.)	
1	Best	0.1	88.416	0.1	0.1	
2	Accept	0.4	0.94126	0.1	0.11193	RUSB
3	Best	0	3.3701	0	0.00024013	GentleB
4	Accept	0.5	0.91616	0	0.00038357	RUSB
5	Accept	0	103.77	0	0.0060676	GentleB
6	Accept	0	19.736	0	0.0049232	LogitB
7	Accept	0.5	0.83437	0	0.0054927	AdaBoo
8	Accept	0.175	2.593	0	2.2961e-05	LogitB
9	Accept	0	42.559	0	-6.642e-05	LogitB
10	Accept	0	95.968	0	-0.00028339	GentleB

و دوباره نیز نتیجه بسیار مطلوب است:

cough	100	0
breath	0	100
	cough	breath

البته یکی از دلایلی که از همان ابتدا نتایج بسیار مناسب بود چون در این پروژه از صدای سرفه و نفس استفاده شده بود که تفاوت های آن ها زیاد بوده با توجه به ویژگی های آن ها بنابراین تشخیص آن بسیار راحت تر از مثال ابتدای پروژه بود.

و برای این حالت مقادیر عبارت اند از

$$TP = TN = 100\% \quad FP = FN = 0\% \quad sensitivity = specificity = 100\%$$

فایل این بخش در پوشه شماره یک می باشد.

حال می خواهیم نتیجه مطلوبی مانند دفعه قبل بگیریم و چند داده را اشتباه لیبل می زنیم و این کار را در فایل cvs که لیبل های داده ها در آن است می کنیم و در ماتریس `reference_table` چک میکنیم. ما ۴ داده را اشتباه لیبل می زنیم مانند شکل زیر: (توجه داشته باشید دوباره داده ی اول یعنی B1 را باید اصلاح کنیم)

Variables - reference_table		
50x2 table		
1	2	3
record_name	record_label	
22 'B22'	1	
23 'B23'	1	
24 'B24'	-1	
25 'B25'	-1	
26 'S1'	1	
27 'S2'	1	
28 'S3'	-1	
29 'S4'	-1	
30 'S5'	-1	
31 'S6'	-1	
32 'S7'	-1	
33 'S8'	-1	
34 'S9'	-1	

21	B21,1
22	B22,1
23	B23,1
24	B24,-1
25	B25,-1
26	S1,1
27	S2,1
28	S3,-1
29	S4,-1

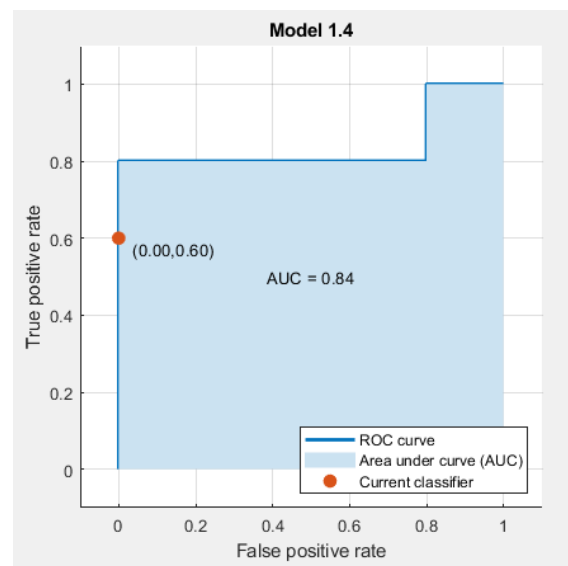
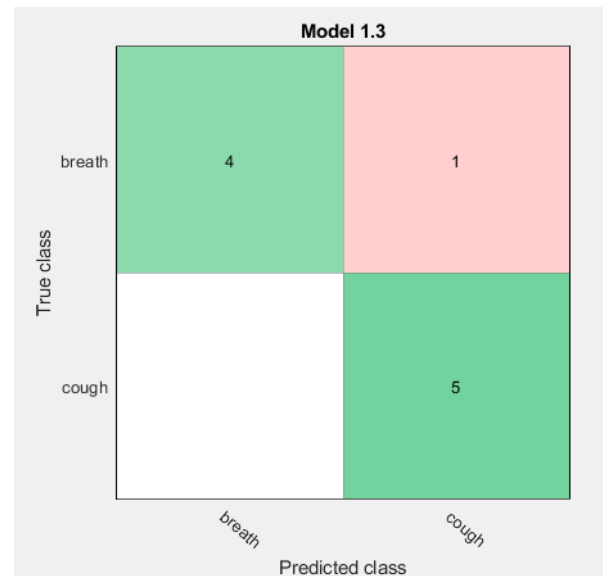
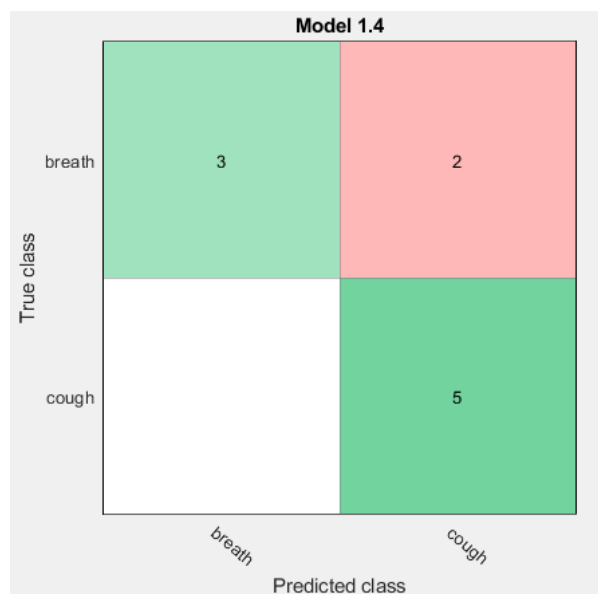
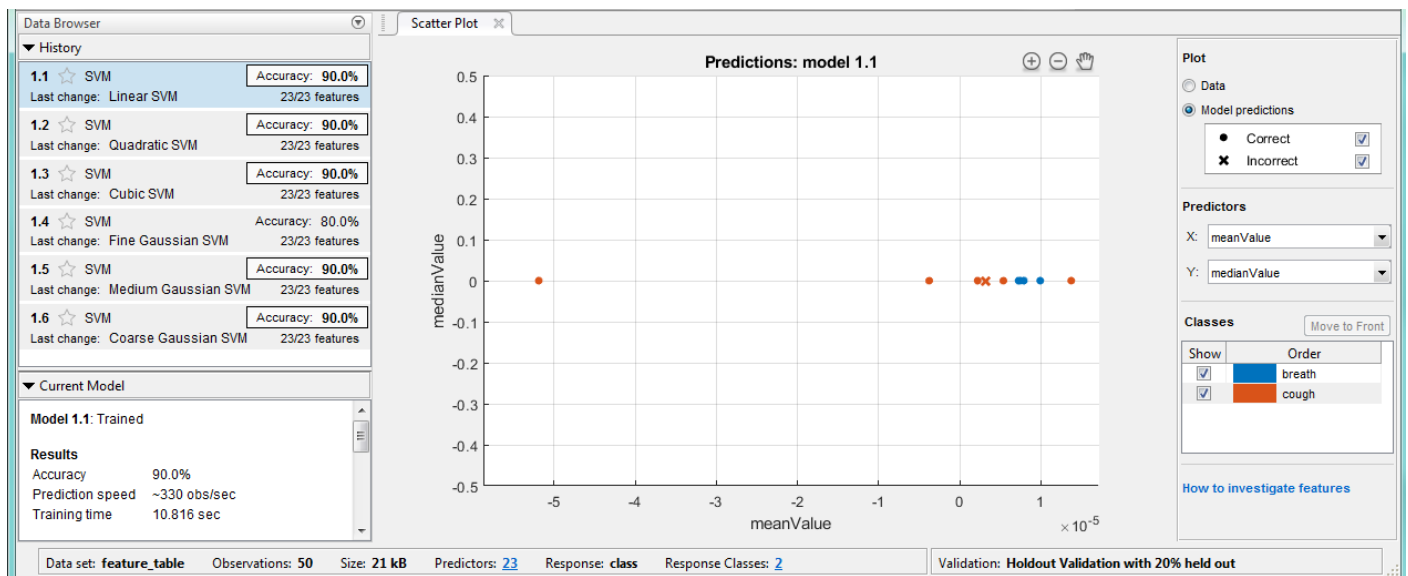
دو بخش اول که مانند حالت ایده آل بوده و از تکرار آن اجتناب می کنیم.

حال مانند حالت قبل مراحل را تکرار می کنیم. ابتدا `feature` ها را محاسبه می کنیم که بخشی از آن به شکل زیر است

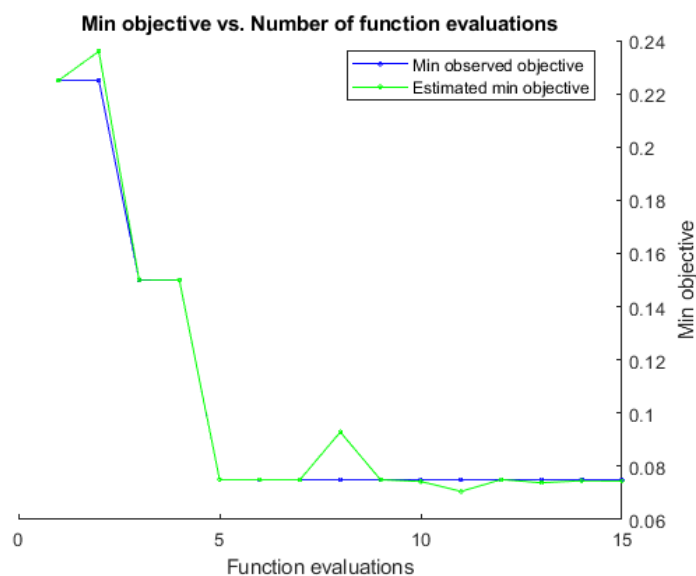
FeatureTable.mat loaded

meanValue	medianValue	standardDeviation	meanAbsoluteDeviation
9.0422e-06	0	0.0016485	0.00089877
7.9631e-06	0	0.0010394	0.00064907
5.5792e-06	0	0.0015738	0.00086774
7.7862e-06	0	0.00077686	0.00051864
8.2667e-06	0	0.00089229	0.00057521

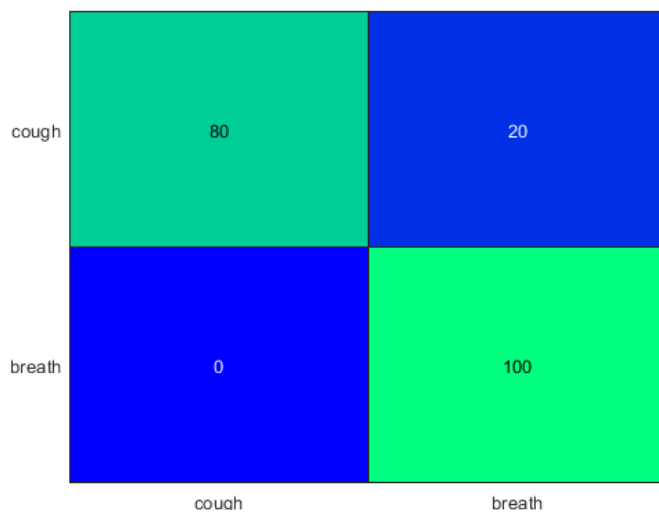
حال با این داده ها عملیات `classification learner` را تکرار می کنیم:



و پس از آن داده های تست و آموزش را جدا کرده و حال به مرحله ای میرسیم که متلب پارامترها را پیدا کند و مدل مناسب ارائه دهد و داریم:

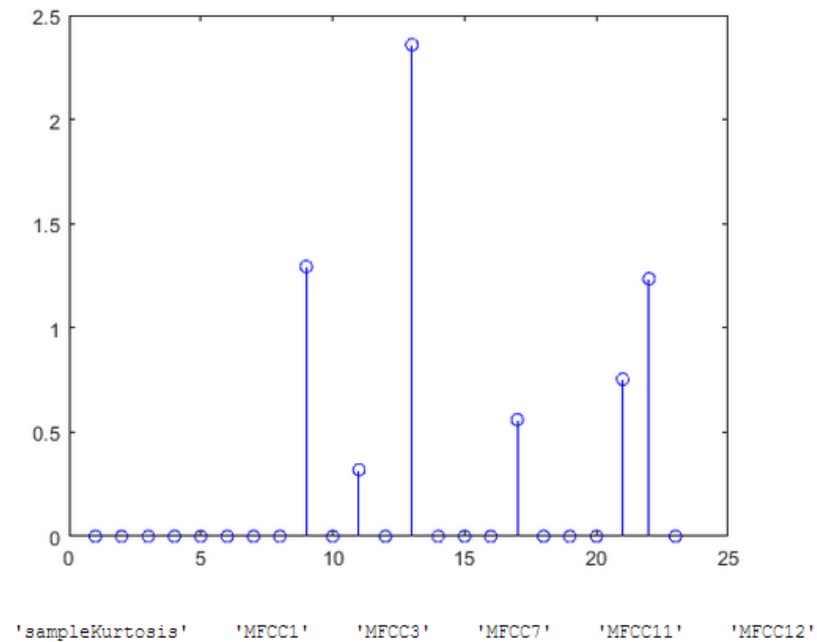


Iter	Eval	Objective	Objective	BestSoFar	BestSoFar	Method	Nu
	result		runtime	(observed)	(estim.)		yc
=====							
1	Best	0.225	107.57	0.225	0.225	Bag	
2	Accept	0.5	2.1337	0.225	0.23593	RUSBoost	
3	Best	0.15	4.5057	0.15	0.15013	GentleBoost	
4	Accept	0.5	1.1461	0.15	0.15002	RUSBoost	
5	Best	0.075	4.0792	0.075	0.075025	LogitBoost	
6	Accept	0.5	1.2153	0.075	0.075027	AdaBoostM1	
7	Accept	0.075	89.256	0.075	0.075014	LogitBoost	
8	Accept	0.125	11.762	0.075	0.092903	LogitBoost	
9	Accept	0.075	98.635	0.075	0.074964	LogitBoost	
10	Accept	0.075	5.097	0.075	0.074322	LogitBoost	
11	Accept	0.075	23.675	0.075	0.070554	LogitBoost	
12	Accept	0.075	2.7771	0.075	0.075034	LogitBoost	
13	Accept	0.125	3.542	0.075	0.073825	GentleBoost	
14	Accept	0.5	0.902	0.075	0.074599	AdaBoostM1	
15	Accept	0.15	2.8068	0.075	0.074454	GentleBoost	

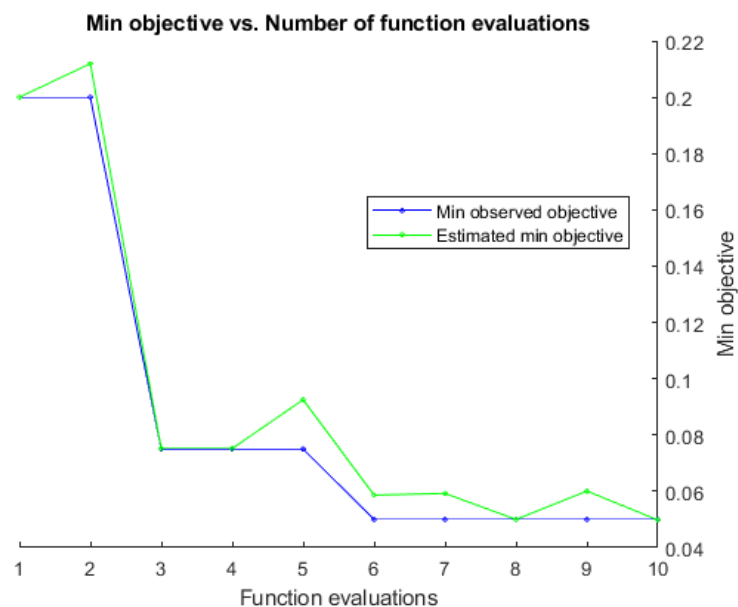


همانطور که مشاهده می شود نتایج از حالت قبل طبیعی تر شده و دیگر ایده آل نمی باشد.

حال مانند قبل به سراغ پیدا کردن feature های مناسب میرویم و داریم:



حال با توجه به feature های به دست آمده به سراغ این می رویم که با استفاده از آن ها دوباره پارامتر ها را به دست آوریم و مدل های مختلف اینجا تست می شوند. داریم:



Iter	Eval	Objective	Objective	BestSoFar	BestSoFar	
	result		runtime	(observed)	(estim.)	
1	Best	0.2	223.62	0.2	0.2	
2	Accept	0.5	7.8549	0.2	0.21193	RL
3	Best	0.075	10.126	0.075	0.075198	Gentl
4	Accept	0.425	2.7222	0.075	0.075198	RL
5	Accept	0.1	135.83	0.075	0.09255	Gentl
6	Best	0.05	4.8322	0.05	0.058655	Logi
7	Accept	0.5	3.3551	0.05	0.059242	AdaB
8	Accept	0.05	147.46	0.05	0.050012	Logi
9	Accept	0.075	5.9318	0.05	0.060078	Logi
10	Accept	0.05	131.01	0.05	0.049803	Logi

cough	80	20
breath	0	100
	cough	breath

نتیجه با حالت قبل زیاد تفاوت نکرد و دلیل آن بخاطر دیتاستی است که داریم چون سرفه و نفس هستند و چون لیبیل اشتباه زدیم نتیجه اینگونه شد و با بهبود روش ها نتیجه زیاد تغییری نخواهد کرد.

اگر نفس کشیدن را P در نظر بگیریم و سرفه را N داریم:

$$TN = 80 \quad TP = 100 \quad FP = 0 \quad FN = 20 \quad sensitivity = 100\% \quad specificity = 80\%$$

فایل این قسمت از پروژه در پوشه ای با نام ۲ ذخیره شده است که فایل اصلی آن با نام breath_cough_wrong می باشد.

❖ توجه : در هر دو پروژه بعد از وارد کردن اطلاعات فایل CSV باید سطر اول آن را از درون متلب اصلاح کرد .

❖ آزمایش را دوبار تکرار کردیم چون در ابتدا دیتاستی که داشتیم صدای سرفه و نفس بود و تفاوت آن بسیار واضح است خروجی بسیار ایده آل بود و برای واقعی تر شدن آن مدل غیر ایده آل را ایجاد کردیم که در آن چنتا از داده ها را اشتباه لیبل زدیم و آزمایش را تکرار کردیم.

❖ فایل های مربوط به حالت ایده آل در پوشه یک می باشد و فایل های مربوط به حالت غیر ایده آل در پوشه دو.

❖ در آزمایش غیر ایده آل نتیجه حالت های ابتدایی با حالت هایی که بهبود دادیم یعنی مدل یابی را به طور خودکار توسط متلب انجام دادیم و features ها را بهینه کردیم تفاوتی نکرد و علت آن این است که در اینجا دیتا ست ما خیلی گسترده نبود و بسیار دیتاست های هر دسته شبیه بودند و با دسته دیگر تفاوت زیادی داشتند و خب خطایی که به طور دستی (لیبل اشتباه زدن) به آن وارد کردیم باعث این موضوع می شود.

❖ فایل هایی که همراه پروژه اند الزامی می باشند اما اگر جدول هایی که با محاسبات به دست می آیند نباشد خود کد بررسی میکند و در صورت نبودن آن ها آن ها را ایجاد می کند.