به نام خدا

دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

پاسخ تمرین سری چهارم یادگیری ماشین

استاد:

دکتر احسان ناظرفرد

دانشجو:

حلیمه رحیمی

شماره دانشجویی:

۹۹۱۳۱۰۴۳

زمستان ۱۳۹۹

سوالات تشریحی

۱- صحت هر یک از موارد زیر را بررسی کرده و دلایل خود را توضیح دهید.

الف) ماشین های بردار پشتیبان پارامتریک اند.

خیر، ماشین های بردار پشتیبان غیرپارامتریک اند. ما بر اساس نمونه ها، بردار های پشتیبان را مشخص و مرز تصمیم را تایین می کنیم و هیچ پارامتر ثابتی برای یادگیری در نظر نداریم و تعداد پارامترها با تعداد نمونه ها تغییر خواهد کرد. این پارامترها به توزیع داده های هر کلاس نیز ارتباطی ندارند.

ب) مقدار حاشیه ی به دست آمده برای دو ماشین بردار پشتیبان با هسته های متفاوت که برای داده های یکسان آموزش دیده اند، می تواند معیاری برای میزان کارایی مدل باشد.

در صورتی که هر دو مرز مشابه به دست دهند همینطور است اما هسته های متفاوت در بیشتر مواقع بسته به داده ها مرز متفاوتی خواهند داد و گاهی ممکن است هسته ی خاصی موجب بیش برازش بر داده های آموزش شوند. همچنین مسئله ی soft یا hard بودن حاشیه نیز وجود دارد.

ج) ماشین های بردار پشتیبان همواره در برابر بیش برازش مقاوم اند.

خیر، به هسته ی انتخاب شده و hard margin بودن بستگی دارد. هسته ی RBF با مقادیر مناسب و در حالت hard margin ممکن است دچار بیش برازش شود.

د) وجود داده های پرت و نویز بر روی ماشین های بردار پشتیبان بی تاثیر است.

داده های پرت تاثیری نخواهند داشت ولی داده ی نویز اگر به عنوان بردار پشتیبان انتخاب شود تاثیر گذار است. این تاثیرگذاری در حالت soft margin کم خواهد بود.

ه) الگوریتم آدابوست با استفاده از هر نوع دسته بند ضعیف و یا ترکیب چند دسته بند ضعیف، در نهایت به خطای آموزش صفر می رسد.

خیر، اگر توزیع داده ها به گونه ای باشد که نتوان با ترکیب خطی دسته بند های ضعیف آنها را از هم جدا کرد، به خطای آموزش صفر نخواهیم رسید. مثال xor را با decision stump و آدابوست در نظر بگیرید. هر بار داده ای یافت می شود که درست دسته بندی نشود؛ بنابراین در این مثال هیچ وقت به خطای آموزش صفر نخواهیم رسید.

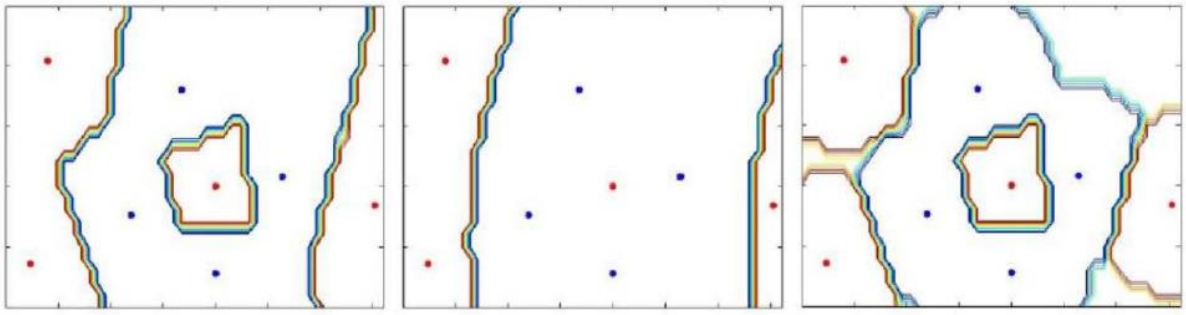
و) وزن های اختصاص داده شده به دسته بند ها در الگوریتم آدابوست همواره نامنفی هستند.

بله، به این علت که همواره وزن اولیه اختصاص داده شده به داده ها عددی مثبت بوده و مجموعشان برابر یک خواهد بود. خطایی که در هر بار اتفاق می افتد، برابر مجموع وزن های داده هایی است که به اشتباه کلاس بندی شده اند. این خطا نیز مثبت و کمتر از یک است.

در نهایت فرمول بروزرسانی وزن ها به شکل زیر است که با جاگذاری یک عدد مثبت کوچکتر از یک به عنوان خطا، پاسخ همواره مثبت خواهد بود.

$$\begin{cases} w_i^{t+1} = \frac{w_i^t}{2} \times \frac{1}{1-\epsilon^t} & h^t(\boldsymbol{x_i}) = y_i \\ w_i^{t+1} = \frac{w_i^t}{2} \times \frac{1}{\epsilon^t} & h^t(\boldsymbol{x_i}) \neq y_i \end{cases}$$

۲- برای حس مسئله ی دسته بندی دو کلاسه از روش بردار پشتیبان با هسته ی RBF با $\boldsymbol{\sigma = \{0.2, 1, 10\}}$ استفاده کرده ایم. مشخص کنید که هر کدام از شکل های زیر حاصل دسته بندی با کدام مقدار سیگما است؟



تصاویر به ترتیب از راست به چپ مربوط به واریانس ۰/۲، ۱۰ و ۱ می باشد. به این دلیل که در هسته ی RBF هر چه سیگما بزرگتر باشد، به این معنی است که داده ها در منحنی توزیع نرمال به شکل smooth تر قرار می گیرند و این هسته مجموع بیشتری از داده ها را در یک توزیع جای می دهد، بنابراین وقتی یک صفحه داده ها را از هم جدا کند، در حالت قرارگیری اصلی داده ها مناطق وسیعتری ایجاد می شود. در واقع سیگمای بیشتر، بایاس بالاتر و واریانس پایینتر را نتیجه خواهد داد. وقتی سیگما بسیار کوچک باشد مطابق شکل یک، ممکن است حتی میان داده هایی که از یک کلاس هستند مرز ایجاد شود.

۳- تفاوت دو روش hard voting و soft voting در الگوریتم های مبتنی بر رای گیری چیست؟

در روش hard voting هر یک از دسته بند ها رای نهایی خود را اعلام می کنند و با میانگین گرفتن از این رای ها به نتیجه گیری نهایی می رسیم. درحالیکه در soft voting از احتمالی که هر دسته بند برای تعلق داده به هر کلاس بدست آورده، میانگین گرفته می شود.

۴- فرض کنید جدول زیر مربوط به یکی از داده های آزمون است. در هر یک از دو روش hard voting و soft voting، کلاس پیش بینی شده توسط الگوریتم کدام است؟ (w_3=2, w_2=1, w_1=2)

	Probabilities		
Classifier	**Class 1**	**Class 2**	**Class 3**
Classifier 1	0.1	0.5	0.4
Classifier 2	0.6	0.3	0.1
Classifier 3	0.4	0.3	0.3

Hard Voting:

Classifier 1: Class 2

Classifier 2: Class 1

Classifier 3: Class 1

با توجه به اینکه وزن دسته بند اول و سوم برابر ۲ می باشد، انتخاب اینها دو بار حساب می شود. بنابراین، دو بار کلاس ۲ و سه بار کلاس ۱ انتخاب شده است و نتیجه کلاس ۱ خواهد بود.

Soft Voting:

$$P(Class\ i) = \frac{w_1 \times P(Class\ i|Classifier\ 1) + w_2 \times P(Class\ i|Classifier\ 2) + w_3 \times P(Class\ i|Classifier\ 3)}{w_1 + w_2 + w_3}$$

$$P(Class\ 1) = \frac{2 \times 0.1 + 0.6 + 2 \times 0.4}{5} = 0.32$$

$$P(Class\ 2) = \frac{2 \times 0.5 + 0.3 + 2 \times 0.3}{5} = 0.38$$

$$P(Class\ 3) = \frac{2 \times 0.4 + 0.1 + 2 \times 0.3}{5} = 0.3$$

با توجه به محاسبات بالا، کلاس دوم نتیجه ی soft voting می باشد.

سوالات پیاده سازی

۱- مجموعه داده ی Parkinson.data که در فایل تمرین وجود دارد را بارگذاری کرده و داده ها را با استفاده از مدل SVM و کرنل های زیر دسته بندی کرده و به سوالات پاسخ دهید.

الف) کرنل خطی

ب) کرنل چند جمله ای (پارامترهای r و d)

ج) RBF (پارامتر گاما)

د) سیگموید (پارامتر r)

در ابتدای امر باید عرض کنم به دلیل مشکلاتی در لپتاپ خود، نتوانستم از فرم نرمالایز شده ی داده ها استفاده کنم و لپتاپ بنده در برابر محاسبات بسیار بزرگ یا مقادیر کوچک دچار مشکل می شد. تنها دلیل من برای استفاده نکردن از نرمالیزیشن همین است. در غیر این صورت به علت اینکه داده ها در یک بازه خاص نیستند و svm نیز یک کلاسیفایر بر اساس داده ها است از آن استفاده می کردم.

۱,۱) معیار Accuracy و F1-Measure را برای هر یک از دسته بندهای بالا به دست آورده و مقادیر بهینه را مشخص کنید. (برای هر یک از پارامترهای یاده شده، حداقل ۴ مقدار متفاوت در نظر بگیرید.)

در تمامی تمرین ها، از کتابخانه های آماده استفاده کرده و برای محاسبه Accuracy و F1-Measure هم از sklearn بهره برده ام.

نیاز به ذکر است که به دلیل unbalanced بودن دیتاست، لازم دیدم F1-Measure را برای هر دو کلاس نمایش دهم. البته با توجه به آنکه کلاس مثبت، کلاس ۱ می باشد، مقدار دوم در F1-Measure پاسخ خواهد بود اما در موقع آموزش باید حتما توجه کرد که به علت بیشتر بودن تعداد داده های کلاس ۱، یادگیرنده تمام داده های تست را به عنوان کلاس ۱ در نظر نگیرد؛ به همین دلیل هم مقدار را برای هر دو کلاس نمایش داده ام.

داده ها را با استفاده از sklearn به صورت ۷۰ و ۳۰ به ترتیب آموزش و آزمون تقسیم کردم. نتایجی که آورده ام مربوط به داده های آزمون می باشد.

الف) کرنل خطی:

```
Linear SVM Accuracy:  0.8813559322033898
Linear SVM F1-Measure:  [0.77419355 0.91954023]
```

۲,۱) تاثیر پارامترهای هر کرنل بر کارایی مدل ها را تحلیل کنید.

کرنل چند جمله ای:

```
Polynomial SVM Accuracy:  0.7627118644067796 for Degree=  3 for r=  0
Polynomial SVM F1-Measure:  [0.46153846 0.84782609] for Degree=  3 for r=  0
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7627118644067796 for Degree=  3 for r=  3
Polynomial SVM F1-Measure:  [0.46153846 0.84782609] for Degree=  3 for r=  3
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7627118644067796 for Degree=  3 for r=  6
Polynomial SVM F1-Measure:  [0.46153846 0.84782609] for Degree=  3 for r=  6
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7627118644067796 for Degree=  3 for r=  9
Polynomial SVM F1-Measure:  [0.46153846 0.84782609] for Degree=  3 for r=  9
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7627118644067796 for Degree=  6 for r=  0
Polynomial SVM F1-Measure:  [0.41666667 0.85106383] for Degree=  6 for r=  0
------------------------------------------------------------
Polynomial SVM Accuracy:  0.864406779661017 for Degree=  6 for r=  3
Polynomial SVM F1-Measure:  [0.73333333 0.90909091] for Degree=  6 for r=  3
------------------------------------------------------------
Polynomial SVM Accuracy:  0.864406779661017 for Degree=  6 for r=  6
Polynomial SVM F1-Measure:  [0.76470588 0.9047619 ] for Degree=  6 for r=  6
------------------------------------------------------------
Polynomial SVM Accuracy:  0.847457627118644 for Degree=  6 for r=  9
Polynomial SVM F1-Measure:  [0.76923077 0.88607595] for Degree=  6 for r=  9
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7457627118644068 for Degree=  7 for r=  0
Polynomial SVM F1-Measure:  [0.34782609 0.84210526] for Degree=  7 for r=  0
------------------------------------------------------------
Polynomial SVM Accuracy:  0.864406779661017 for Degree=  7 for r=  3
Polynomial SVM F1-Measure:  [0.75       0.90697674] for Degree=  7 for r=  3
------------------------------------------------------------
Polynomial SVM Accuracy:  0.8983050847457628 for Degree=  7 for r=  6
Polynomial SVM F1-Measure:  [0.84210526 0.925     ] for Degree=  7 for r=  6
------------------------------------------------------------
Polynomial SVM Accuracy:  0.8813559322033898 for Degree=  7 for r=  9
Polynomial SVM F1-Measure:  [0.82051282 0.91139241] for Degree=  7 for r=  9
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7288135593220338 for Degree=  9 for r=  0
Polynomial SVM F1-Measure:  [0.27272727 0.83333333] for Degree=  9 for r=  0
------------------------------------------------------------
Polynomial SVM Accuracy:  0.8983050847457628 for Degree=  9 for r=  3
Polynomial SVM F1-Measure:  [0.84210526 0.925     ] for Degree=  9 for r=  3
------------------------------------------------------------
Polynomial SVM Accuracy:  0.8813559322033898 for Degree=  9 for r=  6
Polynomial SVM F1-Measure:  [0.82051282 0.91139241] for Degree=  9 for r=  6
------------------------------------------------------------
Polynomial SVM Accuracy:  0.847457627118644 for Degree=  9 for r=  9
Polynomial SVM F1-Measure:  [0.7804878  0.88311688] for Degree=  9 for r=  9
------------------------------------------------------------
Polynomial SVM Accuracy:  0.7796610169491526 for Degree=  12 for r=  0
Polynomial SVM F1-Measure:  [0.48       0.86021505] for Degree=  12 for r=  0
------------------------------------------------------------
Polynomial SVM Accuracy:  0.8305084745762712 for Degree=  12 for r=  3
Polynomial SVM F1-Measure:  [0.75       0.87179487] for Degree=  12 for r=  3
------------------------------------------------------------
Polynomial SVM Accuracy:  0.8305084745762712 for Degree=  12 for r=  6
Polynomial SVM F1-Measure:  [0.75       0.87179487] for Degree=  12 for r=  6
------------------------------------------------------------
Polynomial SVM Accuracy:  0.847457627118644 for Degree=  12 for r=  9
Polynomial SVM F1-Measure:  [0.7804878  0.88311688] for Degree=  12 for r=  9
------------------------------------------------------------
```

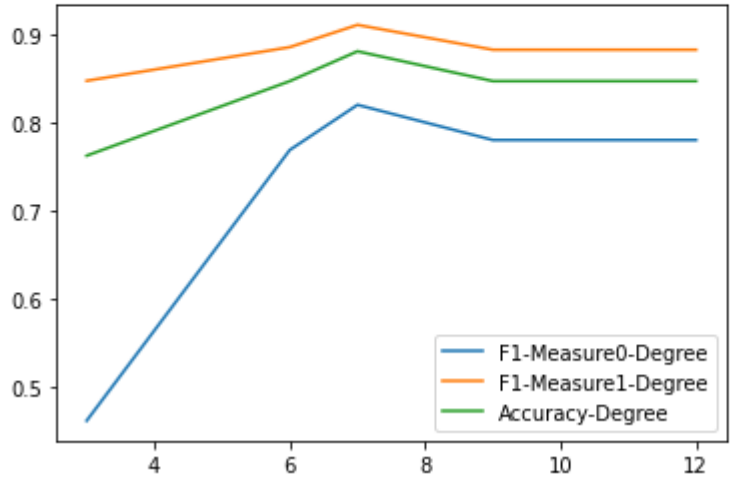
مسلما توزیع داده ها عامل مهمی در تعیین مقدار پارامترها است. در اینجا دو حالت درجه ۷ و عرض از مبدا ۶، و همچنین درجه ۹ و عرض از مبدا ۳ بهترین نتیجه را داده اند.

تاثیر d:

نمودار مربوط به مقدار r=9 می باشد.

```
plt.plot(degree, f10list[:,3],label='F1-Measure0-Degree')
plt.plot(degree, f11list[:,3],label='F1-Measure1-Degree')
plt.plot(degree, acclist[:,3],label='Accuracy-Degree')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1fb233c3438>
```



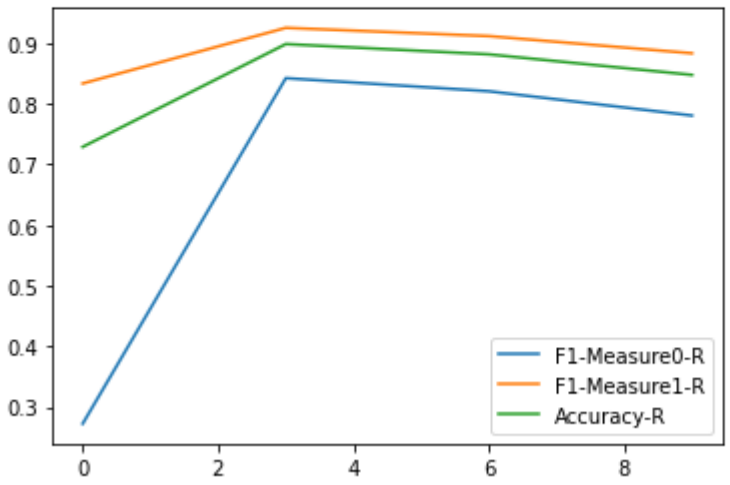
به طور کلی با افزایش مقدار d مرز برازش شده تری بر روی داده های آموزش خواهیم داشت و احتمال overfit شدن بالا می رود. با توجه به نتایج بالا واضح است که توزیع داده ها به گونه ای می باشد که درجه ۳ به خوبی روی داده ها برازش نشده است؛ به خصوص که تعداد داده های منفی (کلاس صفر) بیشتر است. در اینجا نشاندهنده ی این است که داده های منفی تا حدودی لابلای داده های مثبت قرار می گیرند که درجه ۳ قادر به جدا کردن آنها به شکل مناسبی نیست. درجات بالاتر نتایج بهتری داده اند.

تاثیر r:

نمودار مربوط به درجه ۹ می باشد.

```
plt.plot(coef0list, f10list[3,:],label='F1-Measure0-R')
plt.plot(coef0list, f11list[3,:],label='F1-Measure1-R')
plt.plot(coef0list, acclist[3,:],label='Accuracy-R')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1fb23433940>
```



مشاهده می شود که در حالتی که درجه ۳ می باشد، این پارامتر اثر خاصی نگذاشته است. در درجات بالاتر کاملا مشهود است که مقدار صفر برای این پارامتر مناسب این داده ها نبوده است. مقادیر ۳، ۶ و ۹ در رسیدن به جواب خوب یاری کننده بوده اند. مقادیر بالاتر گاه موجب بهتر و گاه موجب بدتر شدن نتایج می شد.

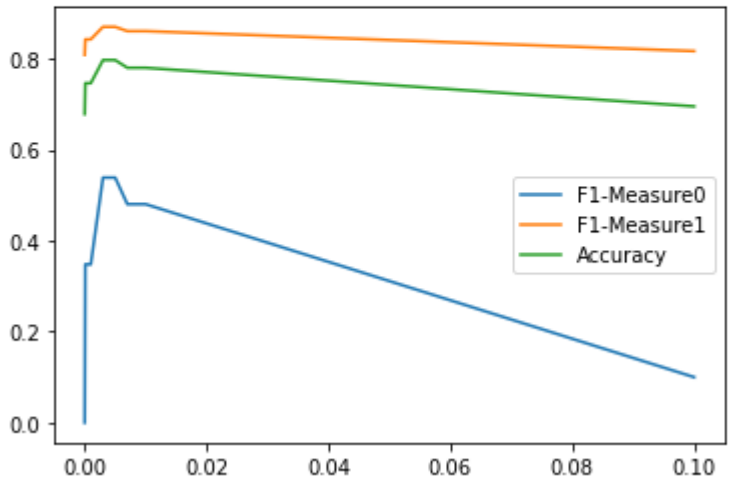
کرنل RBF:

```
RBF SVM Accuracy:  0.6779661016949152 for Gamma=  1e-06
RBF SVM F1-Measure:  [0.         0.80808081] for Gamma=  1e-06
------------------------------------------------------------
RBF SVM Accuracy:  0.7457627118644068 for Gamma=  0.0001
RBF SVM F1-Measure:  [0.34782609 0.84210526] for Gamma=  0.0001
------------------------------------------------------------
RBF SVM Accuracy:  0.7457627118644068 for Gamma=  0.001
RBF SVM F1-Measure:  [0.34782609 0.84210526] for Gamma=  0.001
------------------------------------------------------------
RBF SVM Accuracy:  0.7966101694915254 for Gamma=  0.003
RBF SVM F1-Measure:  [0.53846154 0.86956522] for Gamma=  0.003
------------------------------------------------------------
RBF SVM Accuracy:  0.7966101694915254 for Gamma=  0.005
RBF SVM F1-Measure:  [0.53846154 0.86956522] for Gamma=  0.005
------------------------------------------------------------
RBF SVM Accuracy:  0.7796610169491526 for Gamma=  0.007
RBF SVM F1-Measure:  [0.48       0.86021505] for Gamma=  0.007
------------------------------------------------------------
RBF SVM Accuracy:  0.7796610169491526 for Gamma=  0.01
RBF SVM F1-Measure:  [0.48       0.86021505] for Gamma=  0.01
------------------------------------------------------------
RBF SVM Accuracy:  0.6949152542372882 for Gamma=  0.1
RBF SVM F1-Measure:  [0.1        0.81632653] for Gamma=  0.1
------------------------------------------------------------
```

تاثیر گاما:

```
plt.plot(gammalist, f10list,label='F1-Measure0')
plt.plot(gammalist, f11list,label='F1-Measure1')
plt.plot(gammalist, acclist,label='Accuracy')
plt.legend()
plt.show()
```

```
<matplotlib.legend.Legend at 0x1fb230914a8>
```



گاما هر چه بزرگتر باشد، یعنی سیگمای توزیع های گاوسی در نظر گرفته شده کوچکتر بوده اند و فضای کمی از فضای داده ها را شامل شده اند و احتمال بیش برازش روی داده های آموزشی بالا می رود. از طرفی مقدار کوچک گاما، یعنی سیگمای بزرگ که داده های بیشتری را در یک توزیع گاوسی در نظر گرفته و بایاس بالاتر می شود؛ بنابراین احتمال کم برازش روی داده ها بالا می رود. بنابراین حتما باید مقداری بینابین در نظر گرفته شود. دلیل آنکه کلاس یک همواره F-Measure بالایی دارد، تعداد بالای داده های مربوط به این کلاس است؛ چه دچار کم برازش شویم و همه را کلاس یک بدانیم و چه دچار بیش برازش شویم و بیشتر بخش های فضا را متعلق به کلاس یک بدانیم، مقدار F-Measure برای کلاس صفر پایین و حتی صفر خواهد بود.

کرنل سیگموید:

برای این کرنل، گاما را برابر 0.01 در نظر گرفتم.

البته براساس آنچه در جستجوهایم یافتم[1]، مقدار $\frac{1}{23}$ یا به عبارتی یک بر روی تعداد ابعاد داده ها را نیز امتحان کردم که به مقداری به خوبی نتایج ارائه شده دست نیافتم. سپس به صورت تصادفی مقدار 0.01 را در نظر گرفتم. با توجه به Lin 2003 [2] مقدار گاما را مثبت و مقدار r را منفی و کوچک در نظر داشتم.

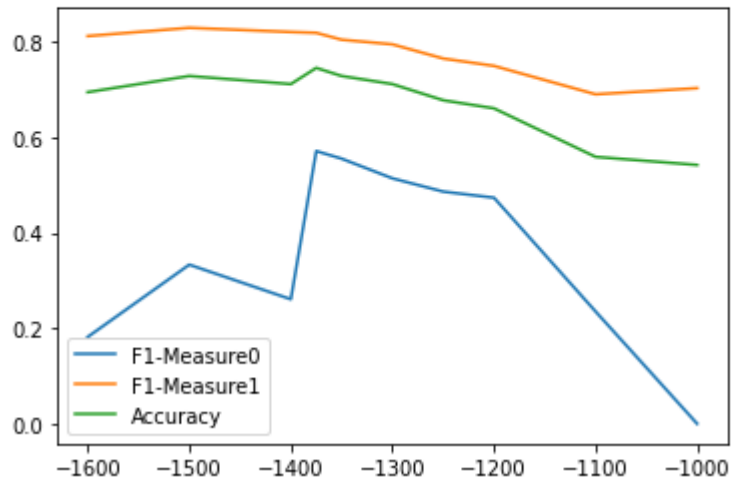
[1] http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

[2] Lin, H.-T. and C.-J. Lin (2003). "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods." submitted to Neural Computation 3(1-32): 16.

```
Sigmoid SVM Accuracy:  0.5423728813559322 for r=  -1000
Sigmoid SVM F1-Measure:  [0.        0.7032967] for r=  -1000
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.559322033898305 for r=  -1100
Sigmoid SVM F1-Measure:  [0.23529412 0.69047619] for r=  -1100
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.6610169491525424 for r=  -1200
Sigmoid SVM F1-Measure:  [0.47368421 0.75      ] for r=  -1200
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.6779661016949152 for r=  -1250
Sigmoid SVM F1-Measure:  [0.48648649 0.7654321 ] for r=  -1250
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.711864406779661 for r=  -1300
Sigmoid SVM F1-Measure:  [0.51428571 0.79518072] for r=  -1300
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.7288135593220338 for r=  -1350
Sigmoid SVM F1-Measure:  [0.55555556 0.80487805] for r=  -1350
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.7457627118644068 for r=  -1375
Sigmoid SVM F1-Measure:  [0.57142857 0.81927711] for r=  -1375
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.711864406779661 for r=  -1400
Sigmoid SVM F1-Measure:  [0.26086957 0.82105263] for r=  -1400
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.7288135593220338 for r=  -1500
Sigmoid SVM F1-Measure:  [0.33333333 0.82978723] for r=  -1500
------------------------------------------------------------
Sigmoid SVM Accuracy:  0.6949152542372882 for r=  -1600
Sigmoid SVM F1-Measure:  [0.18181818 0.8125    ] for r=  -1600
------------------------------------------------------------
```

پارامتر r:

```
plt.plot(coef0list, f10list,label='F1-Measure0')
plt.plot(coef0list, f11list,label='F1-Measure1')
plt.plot(coef0list, acclist,label='Accuracy')
plt.legend()
plt.show()
```



این پارامتر روی threshold اثرگذار است و به دلیل جابجایی تابع تانژانت هایپربولیک موجب تغییر threshold می شود.

۳,۱) آیا روشی هوشمند برای تنظیم پارامترها وجود دارد؟ به طور خلاصه توضیح دهید.

می توان از روش های جستجوی رندوم، الگوریتم ژنتیک و ... استفاده کرد. مقالاتی نیز در مورد پیدا کردن بازه ی مناسب برای جستجوی exhaustive ارائه شده است که پارامترهای مختلفی را برای تعیین این بازه در نظر می گیرند؛ از جمله استفاده از فاصله ی درون کلاسی و بین کلاسی.

۲- مجموعه داده ی pima_indians_diabetes.csv که در فایل تمرین وجود دارد را بارگذاری کرده و به کمک مدل های زیر داده ها را دسته بندی کنید.

با توجه به اینکه از جنگل تصادفی استفاده می کنیم، نیازی به نرمالایز کردن داده ها نیست. لازم دیدم به دلیل unbalanced بودن دیتاست صرفا برای اینکه از نتایج در هنگام آموزش آگاه بشوم، مقادیر F1-Measure را نیز داشته باشم.

داده ها را مطابق سوال پیش به صورت ۷۰، ۳۰ جدا کردم.

۱,۲) به ازای حداقل ۳ مقدار برای هر یک از پارامترهای زیر، دسته بند جنگل تصادفی را بر روی این مجموعه داده آموزش دهید و دقت مدل بر روی مجموعه آموزش و آزمون را گزارش و بهترین مدل را مشخص کنید.

(n_estimators, max_features, max_depth)

برای هر کدام از این پارامترها ۴ مقدار در نظر گرفته ام. تمامی نتایج حاصل را به علت طولانی بودن در یک فایل pdf گذاشته ام و در اینجا به برخی نتایج که برای پاسخ بخش ۲,۲ نیاز است اکتفا کرده ام. با توجه به اینکه جنگل تصادفی ممکن است با پارامترهای یکسان نتایج متفاوتی بدهد (به علت تعداد ویژگی های انتخابی و تصادفی بودن داده ها در حالت استفاده از bootstrap) باید عرض کنم که با چند بار اجرا گرفتن، بهترین دقت تقریبا در یک بازه ی بین ۷۲ تا ۸۱ درصد متغیر بود.

بهترین نتیجه:

```
Best Accuracy for Test:  0.7575757575757576 for n_estimators=  10 for max_features=  2 for max_depth=  5
for n_estimators=  10 for max_features=  2 for max_depth=  5
Random Forest Accuracy for Train:  0.8677839851024208
Random Forest F1-Measure for Train:  [0.89929078 0.80758808]
---------------
Random Forest Accuracy for Test:  0.7575757575757576
Random Forest F1-Measure for Test:  [0.83030303 0.57575758]
```

نمونه ای از بیش برازش:

```
for n_estimators=  50 for max_features=  8 for max_depth=  20
Random Forest Accuracy for Train:  1.0
Random Forest F1-Measure for Train:  [1. 1.]
---------------
Random Forest Accuracy for Test:  0.7316017316017316
Random Forest F1-Measure for Test:  [0.80379747 0.57534247]
```

۲,۲) نقش و تاثیر پارامترهای یاد شده بر عملکرد مدل ها را تحلیل کنید.

همانطور که در نتایج می بینید، تعداد ۵۰ درخت موجب بیش برازش بر روی داده های آموزش شده است و دقت ۱ را برای داده های آموزش و دقت ۷۳-۷۸ را برای داده های آزمون نتیجه می دهد. این حالت در بیشتر مواقع از جمله در نتایج نهایی ارائه شده، در مقدار عمق ۲۰ اتفاق می افتاد. در بیشتر مواقع، مقدار ۱۰ و ۱۵ برای تعداد estimator و عمق بهترین دقت ها را حاصل می شدند.

بنابراین تاثیر هر یک از این پارامترها را اینطور بیان می کنم:

تاثیر n_estimator: با بالا رفتن آن، احتمال بیش برازش بر روی داده های آموزش بالاست.

تاثیر max_features: بسته به اینکه چه ویژگی هایی را در نظر گرفته ایم و چه ویژگی هایی با هم در نظر گرفته شده اند نتیجه متفاوت است. گاه ممکن است یک یا دو ویژگی چنان خوب باشد که به راحتی بیشتر داده ها را از هم جدا کند. مسئله اینجاست که در جنگل تصادفی به علت استفاده از چندین درخت، تاثیر ویژگی ها به گونه ای خواهد بود که اگر ویژگی بخصوصی همواره بهترین تمایز را ایجاد کند، در بیشتر درخت ها انتخاب خواهد شد و در غیر این صورت در برخی از آنها ظاهر می شود. البته انتخاب ویژگی ها به داده هایی که به هر درخت داده شده نیز بستگی دارد. در صورتی که تعداد estimator ها بالا باشد، تعداد کم ویژگی ها ممکن است باعث واریانس بالا در نتایج درخت ها شود، در حالیکه تعداد ویژگی زیاد موجب می شود به احتمال بالایی درخت ها پاسخ مشابه بدهند و بایاس افزایش می یابد.

تاثیر max_depth: با بالا رفتن عمق، احتمال بیش برازش بر روی داده های آموزش بالا می رود.

۳,۲) از هر یک از روش های ترکیبی و یا پایه ی دلخواه استفاده کرده و سعی کنید دقت بر روی مجموعه داده ی آزمون را افزایش دهید. (برای این بخش تنها ۳ مدل نهایی کفایت می کند. اما بهتر است دقت به دست آمده، بیشتر یا مساوی دقت بهترین مدل بخش پیشین باشد. بهترین دقت مشمول نمره امتیازی خواهد بود.)

از روش پایه ی SVM با کرنل چند جمله ای و Logistic Regression استفاده کردم که در بیشتر مواقع نتایج قابل قبولی را حاصل می شدند.

SVM یک یادگیرنده ی قدرتمند است که در اینجا با کرنل چند جمله ای با درجه ۵ بهترین نتیجه را می داد. مقدار عرض از مبدا بین ۳ تا ۵ برای بهترین نتیجه متغیر بود.

```
Polynomial SVM Accuracy:  0.7402597402597403 for Degree=  5 for r=  4
Polynomial SVM F1-Measure:  [0.81595092 0.55882353] for Degree=  5 for r=  4
```

Logistic Regression به همراه SVM بهترین نتایج را در تمامی اجراهایم داشت.

```
clf = LogisticRegression()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred)
f1s = f1_score(y_test, y_pred, average=None)
print(acc)
print(f1s)
```

```
0.7359307359307359
[0.8189911 0.512    ]
```

این یادگیرنده در برخی اجرا ها نتیجه بهتری از جنگل تصادفی هم می داد که نمونه اش را در زیر مشاهده می کنید.

```
Random Forest Accuracy:  0.8008658008658008 for n_estimators=  10 for max_features=  6 for max_depth=  20
Random Forest F1-Measure:  [0.86309524 0.63492063] for =  10 for =  6 for max_depth=  20
```

```
clf = LogisticRegression()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred)
f1s = f1_score(y_test, y_pred, average=None)
print(acc)
print(f1s)
```

```
0.8095238095238095
[0.86419753 0.68115942]
```

روش های ترکیبی که به کار بردم را در زیر شرح می دهم:

با وجود آنکه در روش های ترکیبی از دسته بند های ضعیف استفاده می شود، به دلیل ماهیت تصادفی بودن داده ها در bagging به نظرم رسید بهتر است با یادگیرنده های قوی نیز آن را امتحان کنم. همچنین با توجه به آنکه svm بسته به داده ها عمل می کند، با کم کردن مقدار max_sample سعی بر این داشتم تا کمی از قدرت آن کاسته شود.

```
clf = BaggingClassifier(base_estimator=svm.SVC(kernel='poly', degree=5, coef0=5), n_estimators=100, max_samples=50)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred)
f1s = f1_score(y_test, y_pred, average=None)
print(acc)
print(f1s)
```

```
0.7532467532467533
[0.82461538 0.58394161]
```

مسئله این است که با وجود نتیجه بسیار خوبی که در اینجا می بینید، گاهی نتیجه از پاسخ یک دسته بند svm به تنهایی کمی بدتر می شود؛ چرا که بسته به انتخاب هایی است که با داده ها داشته ایم.

Logistic Regression به همراه آدابوست به شکل کاهش گرادیان نیز نتیجه قابل قبولی داشت.

```
clf = AdaBoostClassifier(base_estimator=LogisticRegression(),algorithm='SAMME', n_estimators=100, random_state=0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred)
fls = f1_score(y_test, y_pred, average=None)
print(acc)
print(fls)
```

```
0.7402597402597403
[0.82035928 0.53125   ]
```

پس از اینها، آدابوست با decision stump نتیجه مناسبی داشت.

```
clf = AdaBoostClassifier(n_estimators=300)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred)
fls = f1_score(y_test, y_pred, average=None)
print(acc)
print(fls)
```

```
0.7316017316017316
[0.80254777 0.58108108]
```