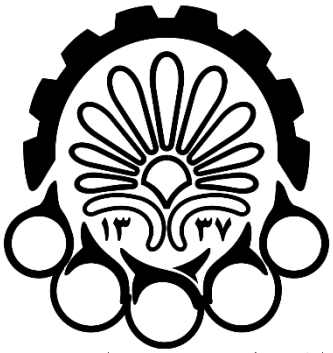
به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین درس شناسایی آماری الگو –سری سوم

فردین آیار

شماره دانشجویی: ۹۹۱۳۱۰۴۰

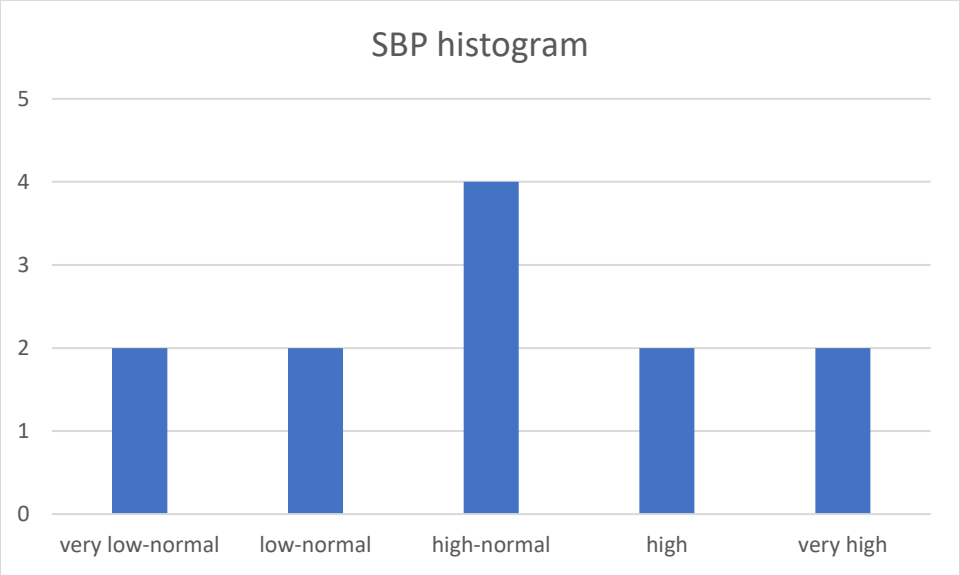
استاد: دکتر رحمتی

دانشکده کامپیوتر– زمستان۹۹

فایل های مربوط به تمریناتی که نیاز به پیاده سازی دارند، در پوشه هایی با شماره تمرین قراردارند.

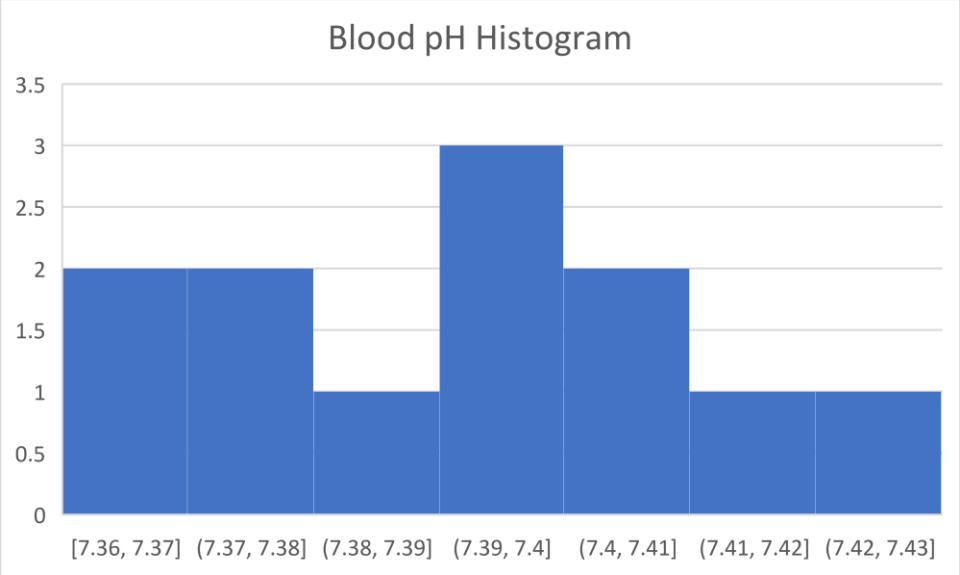
۱)

a) با توجه به دسته های داده شده، هیستوگرام مربوط به SBP به شکل زیر است:



برای رسم هیستوگرام مربوط به Blood pH ابتدا جدول بین‌ها و سپس نمودار را رسم می‌کنیم:

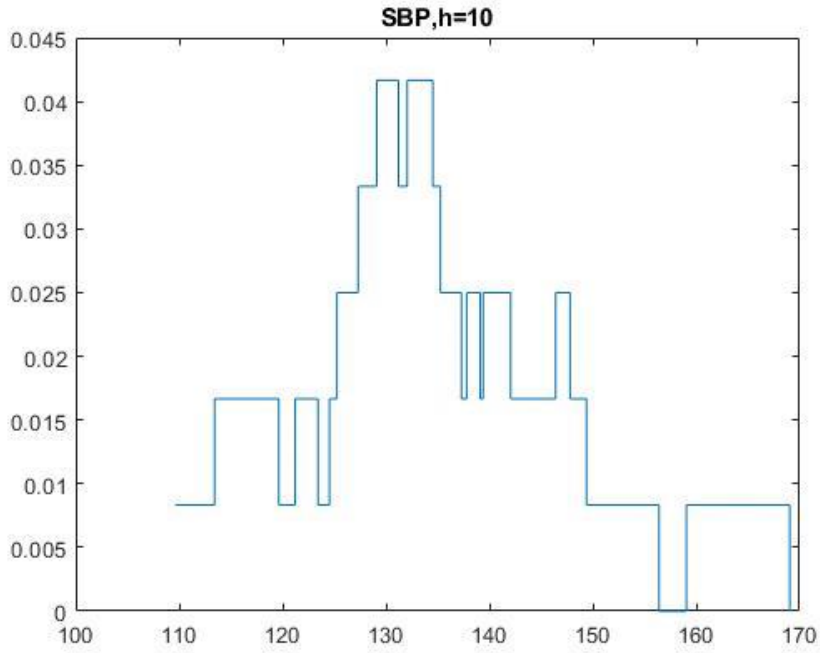
bin	lower bound	upper bound	bin center	count
1	7.36	7.37	7.365	2
2	7.37	7.38	7.375	2
3	7.38	7.39	7.385	1
4	7.39	7.4	7.395	3
5	7.4	7.41	7.405	2
6	7.41	7.42	7.415	1
7	7.42	7.43	7.425	1

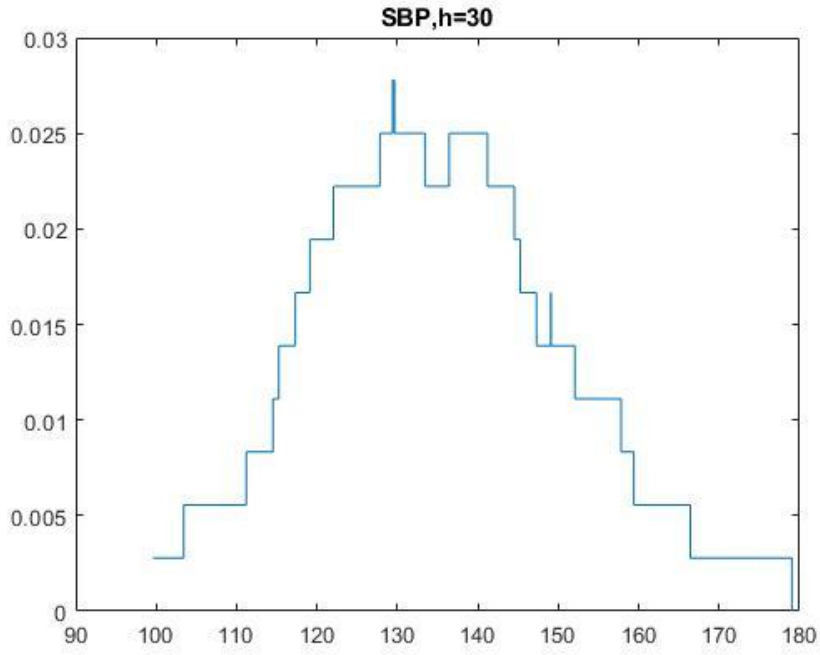


b) **برای انجام این بخش از خطاها صرفنظر شده است.** برای رسم تابع توزیع به این صورت عمل می‌کنیم: ابتدا نقاط را مرتب می‌کنیم. سپس با توجه به h ، محدوده مربوط به هر نقطه را مشخص می‌کنیم(ستون bound در جدول زیر). در ادامه نقاط شکست در نمودار که همان محدوده‌های بالا و پایین boundهستند را مشخص کرده و در ستون point می نویسیم. در انتها با توجه به ستون bound مقدار تابع را در نقاط شکست مشخص می‌کنیم(تعداد بازه هایی که point در آنها قرار دارد تقسیم برnh). جدول نهایی برای پارامتر SBP به صورت زیر است:

SBP							
h = 10				h=30			
bound		point	y	bound		point	y
109.63	119.63	109.63	0.008333	99.63	129.63	99.63	0.002778
113.43	123.43	113.43	0.016667	103.43	133.43	103.43	0.005556
121.2	131.2	119.63	0.008333	111.2	141.2	111.2	0.008333
124.53	134.53	121.2	0.016667	114.53	144.53	114.53	0.011111
125.26	135.26	123.43	0.008333	115.26	145.26	115.26	0.013889
127.29	137.29	124.53	0.016667	117.29	147.29	117.29	0.016667
129.11	139.11	125.26	0.025	119.11	149.11	119.11	0.019444
132.05	142.05	127.29	0.033333	122.05	152.05	122.05	0.022222
137.81	147.81	129.11	0.041667	127.81	157.81	127.81	0.025
139.4	149.4	131.2	0.033333	129.4	159.4	129.4	0.027778
146.42	156.42	132.05	0.041667	136.42	166.42	129.63	0.025
159.05	169.05	134.53	0.033333	149.05	179.05	133.43	0.022222
		135.26	0.025			136.42	0.025
		137.29	0.016667			141.2	0.022222
		137.81	0.025			144.53	0.019444
		139.11	0.016667			145.26	0.016667
		139.4	0.025			147.29	0.013889
		142.05	0.016667			149.05	0.016667
		146.42	0.025			149.11	0.013889
		147.81	0.016667			152.05	0.011111
		149.4	0.008333			157.81	0.008333
		156.42	0			159.4	0.005556
		159.05	0.008333			166.42	0.002778
		169.05	0			179.05	0

در انتها مقادیر point و y را به کمک تابع stairs در متلب رسم می‌کنیم:





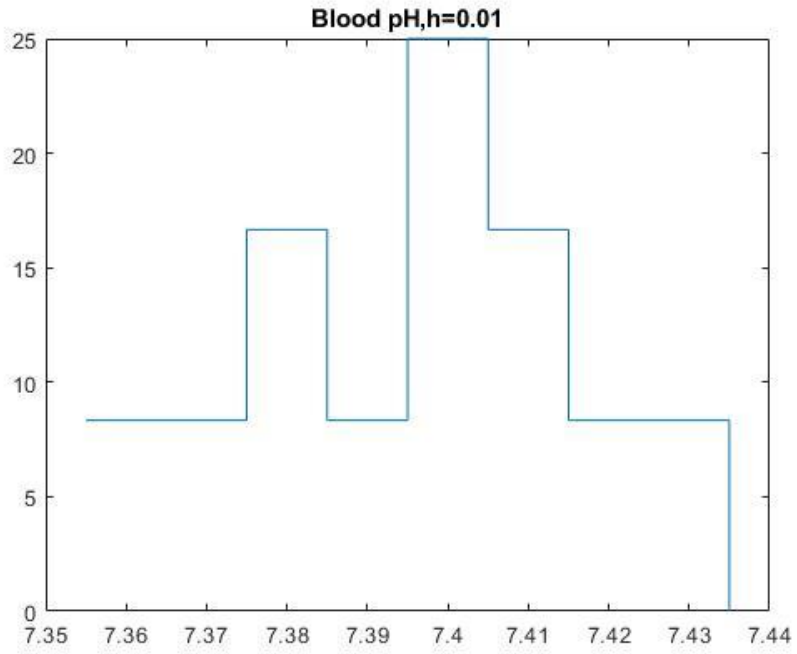
همانطور که مشاهده می‌شود با افزایش مقدار h، شکل تابع توزیع smoothتر شده‌است و به نظر می‌رسد دارای توزیع نرمال است. در ادامه روند فوق را برای Blood pH تکرار می‌کنیم:

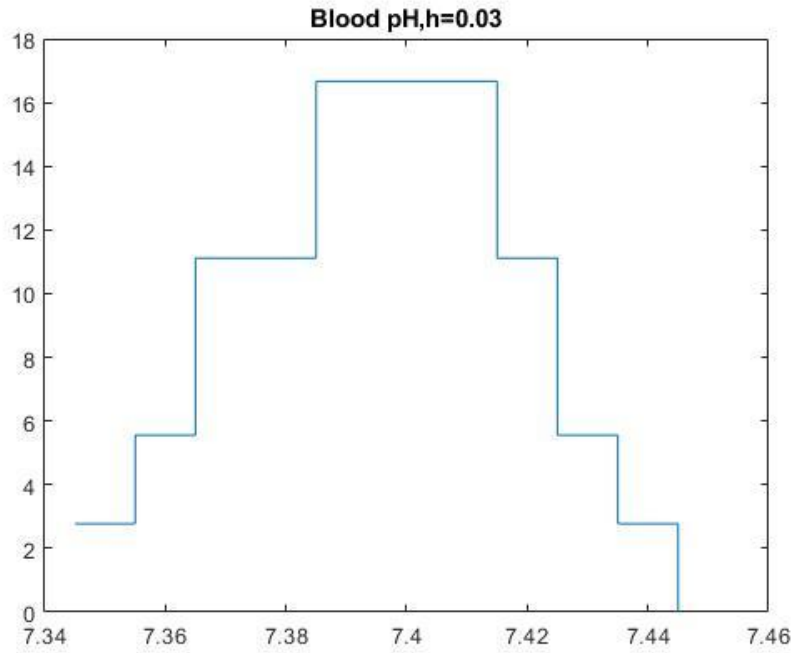
Blood pH	
h = 0.01	h=0.03

bound		point	y
7.355	7.365	7.355	8.333333
7.365	7.375	7.365	8.333333
7.375	7.385	7.375	16.66667
7.375	7.385	7.385	8.333333
7.385	7.395	7.395	25
7.395	7.405	7.405	16.66667
7.395	7.405	7.415	8.333333
7.395	7.405	7.425	8.333333
7.405	7.415	7.435	0
7.405	7.415		
7.415	7.425		
7.425	7.435		

bound		point	y
7.345	7.375	7.345	2.777778
7.355	7.385	7.355	5.555556
7.365	7.395	7.365	11.11111
7.365	7.395	7.375	11.11111
7.375	7.405	7.385	16.66667
7.385	7.415	7.395	16.66667
7.385	7.415	7.405	16.66667
7.385	7.415	7.415	11.11111
7.395	7.425	7.425	5.555556
7.395	7.425	7.435	2.777778
7.405	7.435	7.445	0
7.415	7.445		

نمودارهای مربوط به جداول فوق به صورت زیر است:





(c

d-1)محاسبات مربوط به این قسمت با استفاده از برنامه اکسل انجام‌شده است.فایل مربوطه در پوشه مربوط به تمرین ۱ موجود است. مقدار خواسته شده را به صورت زیر می‌یابیم:

$$P(SBP = 131) = \frac{1}{nh}\sum \varphi_i(u) \qquad where\ u = \frac{131 - x_i}{h}\ , h = 10$$

که $\varphi(u)$ نشان دهنده توزیع نرمال به میانگین 0 و واریانس 1 می‌باشد. محاسبات در جدول زیر ارائه می‌شود:

Volunteer	SBP (mmHg)	u	ø(u)
1	134.11	0.311	0.380108313
2	129.53	-0.147	0.394655111
3	142.81	1.181	0.1986285
4	130.26	-0.074	0.39785147
5	118.43	-1.257	0.181053442
6	144.4	1.34	0.162555055
7	126.2	-0.48	0.355532529
8	137.05	0.605	0.332222274
9	114.63	-1.637	0.104473374
10	151.42	2.042	0.049597218
11	132.29	0.129	0.395636652
12	164.05	3.305	0.001694359
P(SBP=131)			0.024616736

d-2)

$$P(125 < SBP < 145) = \frac{1}{n}\sum[\omega_i(a) - \omega_i(b)] \quad where\ a = \left(\frac{145 - x_i}{10}\right), b = \left(\frac{125 - x_i}{10}\right)$$

که $\omega_i(u)$ نشان دهنده توزیع نرمال تجمعی به میانگین 0 و واریانس 1 می‌باشد.

Volunteer	SBP (mmHg)	u_bound		w(a)-w(b)
1	134.11	1.089	-0.911	0.680775
2	129.53	1.547	-0.453	0.613794
3	142.81	0.219	-1.781	0.549219
4	130.26	1.474	-0.526	0.630315
5	118.43	2.657	0.657	0.251648
6	144.4	0.06	-1.94	0.497732
7	126.2	1.88	-0.12	0.517704
8	137.05	0.795	-1.205	0.672592
9	114.63	3.037	1.037	0.148673
10	151.42	-0.642	-2.642	0.256316
11	132.29	1.271	-0.729	0.665135
12	164.05	-1.905	-3.905	0.028343
P(125<SBP<145)		0.459353891		

d-3) مشابه قسمت‌های قبل عمل می‌کنیم:

$$P(pH = 7.42) = \frac{1}{nh}\sum \varphi_i(u) \qquad where\ u = \frac{7.42 - x_i}{h}, h = 0.01$$

که $\varphi(u)$ نشان دهنده توزیع نرمال به میانگین 0 و واریانس 1 می‌باشد. محاسبات در جدول زیر ارائه می‌شود:

Volunteer	Blood pH	u	ø(u)
1	7.38	4	0.00013383
2	7.43	-1	0.241970725
3	7.42	0	0.39894228
4	7.4	2	0.053990967
5	7.38	4	0.00013383
6	7.36	6	6.07588E-09
7	7.41	1	0.241970725
8	7.4	2	0.053990967
9	7.39	3	0.004431848
10	7.37	5	1.48672E-06
11	7.41	1	0.241970725
12	7.4	2	0.053990967

P(pH=7.42)		10.76273629

دلیل اینکه احتمال فوق بیش از ۱ بدست آمد این است که خروجی فوق در واقع مقدار تابع چگالی احتمال در نقطه 131 است. در متغیرهای پیوسته احتمال نقطه‌ای معنا ندارد.

d-4) مشابه قسمت‌های قبل عمل می‌کنیم:

$$P(pH < 7.38) = \frac{1}{n}\sum[\omega_i(a)] \quad where\ a = \left(\frac{7.38 - x_i}{0.01}\right)$$

که $\omega_i(u)$ نشان دهنده توزیع نرمال تجمعی به میانگین 0 و واریانس 1 می‌باشد.

Volunteer	Blood pH	u	w(u)
1	7.38	0	0.5
2	7.43	-5	2.87E-07
3	7.42	-4	3.17E-05
4	7.4	-2	0.02275
5	7.38	0	0.5
6	7.36	2	0.97725
7	7.41	-3	0.00135
8	7.4	-2	0.02275
9	7.39	-1	0.158655
10	7.37	1	0.841345
11	7.41	-3	0.00135
12	7.4	-2	0.02275
P(pH<7.38)			0.254019

۲)

---

a) نقاط را به ترتیب ۱ تا ۳ می‌نامیم. ابتدا توابع جداکننده را تعریف می‌کنیم. با توجه به K=1 و متفاوت بودن برچسب نقاط داریم:

$$g_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \qquad (x, y) \in label(c) \quad if \quad c = arg\min_i(g_i) \quad \forall i$$

با توجه به ویژگی توابع جداکننده می‌توانیم آن ها را به صورت زیر تغییر دهیم:

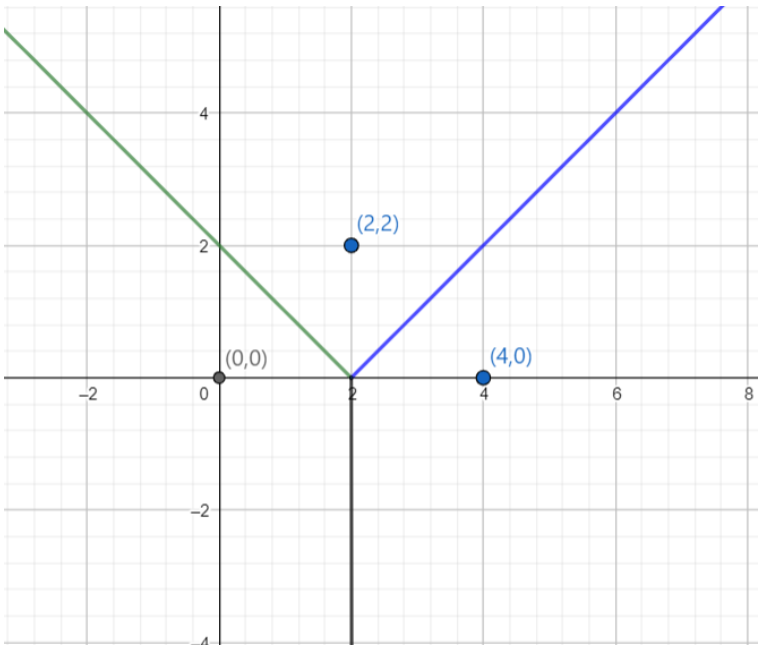
$$g_i = (x - x_i)^2 + (y - y_i)^2 \qquad (x, y) \in label(c) \quad if \quad c = arg\min_i(g_i) \quad \forall i$$

حال معادله مرزهای تصمیم به شکل زیر به دست می‌آید. منظور از H_{12} مرز تصمیم بین نقطه ۱ و ۲ است.

$$\begin{cases} H_{12} = (x-0)^2 + (y-0)^2 - (x-2)^2 - (y-2)^2 = 0 \Rightarrow H_{12} = x + y - 2 = 0 \\ H_{13} = (x-0)^2 + (y-0)^2 - (x-4)^2 - (y-0)^2 = 0 \Rightarrow H_{13} = x - 2 = 0 \\ H_{23} = (x-2)^2 + (y-2)^2 - (x-4)^2 - (y-0)^2 = 0 \Rightarrow H_{23} = x - y - 2 = 0 \end{cases}$$

معادلات بالا خطی هستند و نقطه برخورد آن‌ها (2,0) می‌باشد. بنابراین مرزهای تصمیم به صورت زیر رسم می‌شود:

$$H_{12} = x + y - 2 = 0 \qquad plot\ for\ x < 2$$
$$H_{13} = x - 2 = 0 \qquad plot\ for\ y < 0$$
$$H_{23} = x - y - 2 = 0 \qquad plot\ for\ x > 2$$



Draw with GeoGabra

در شکل بالا، برچسب هرناحیه برابر با برچسب نقطه مربوط به آن ناحیه است.

b) مشابه بخش قبل، توابع جداکننده را به صورت زیر تعریف می‌کنیم.

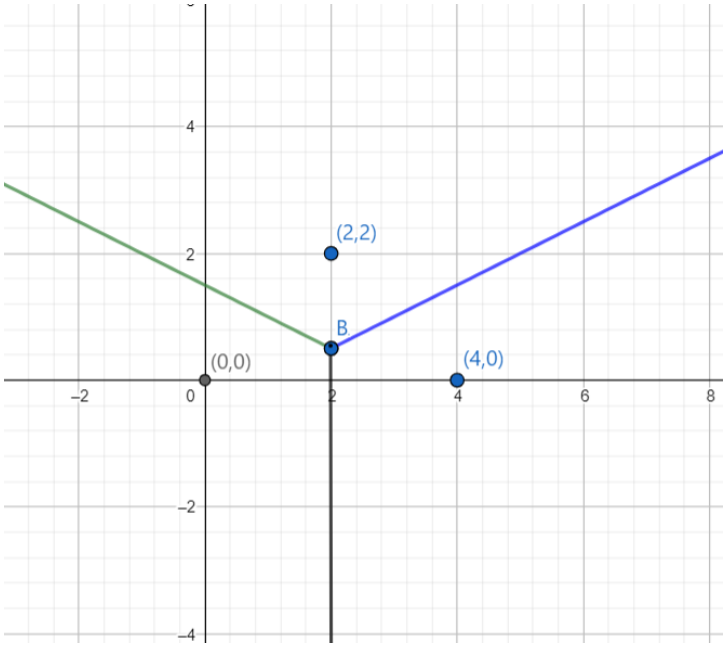
$$g_i = \frac{1}{2}(x - x_i)^2 + (y - y_i)^2 \qquad (x, y) \in label(c) \quad if \quad c = arg\min_i(g_i) \quad \forall i$$

بنابراین معادلات مرزتصمیم به شکل زیر است.

$$\begin{cases} H_{12} = \frac{1}{2}(x-0)^2 + (y-0)^2 - \frac{1}{2}(x-2)^2 - (y-2)^2 = 0 \Rightarrow H_{12} = x + 2y - 3 = 0 \\ H_{13} = \frac{1}{2}(x-0)^2 + (y-0)^2 - \frac{1}{2}(x-4)^2 - (y-0)^2 = 0 \Rightarrow H_{13} = x - 2 = 0 \\ H_{23} = \frac{1}{2}(x-2)^2 + (y-2)^2 - \frac{1}{2}(x-4)^2 - (y-0)^2 = 0 \Rightarrow H_{23} = x - 2y - 1 = 0 \end{cases}$$

معادلات بالا خطی هستند و نقطه برخورد آن‌ها (2,0.5) می‌باشد. بنابراین مرزهای تصمیم به صورت زیر رسم می‌شود:

$$H_{12} = x + 2y - 3 = 0 \qquad plot\ for\ x < 2$$
$$H_{13} = x - 2 = 0 \qquad plot\ for\ y < 0.5$$
$$H_{23} = x - 2y - 1 = 0 \qquad plot\ for\ x > 2$$



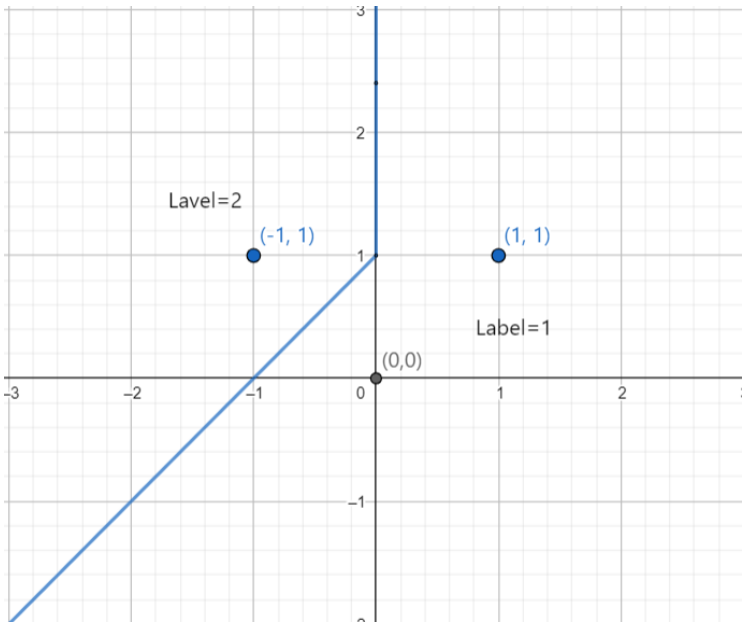
Draw with GeoGabra

c) نقاط را به ترتیب ۱ تا ۳ می‌نامیم. روند مشابه بخش a است با این تفاوت که برچسب نقاط ۱ و ۲ در این دیتاست یکسان است. بنابراین نیازی به رسم مرز تصمیم بین ۱و۲ نیست.

$$\begin{cases} H_{13} = (x-0)^2 + (y-0)^2 - (x+1)^2 - (y-1)^2 = 0 \Rightarrow H_{13} = -x + y - 1 = 0 \\ H_{23} = (x-1)^2 + (y-1)^2 - (x+1)^2 - (y-1)^2 = 0 \Rightarrow H_{23} = x = 0 \end{cases}$$

معادلات بالا خطی هستند و نقطه برخورد آن‌ها (0,1) می‌باشد. بنابراین مرزهای تصمیم به صورت زیر رسم می‌شود:

$$H_{13} = -x + y - 1 = 0 \qquad plot\ for\ x < 0$$
$$H_{23} = x = 0 \qquad plot\ for\ y > 1$$



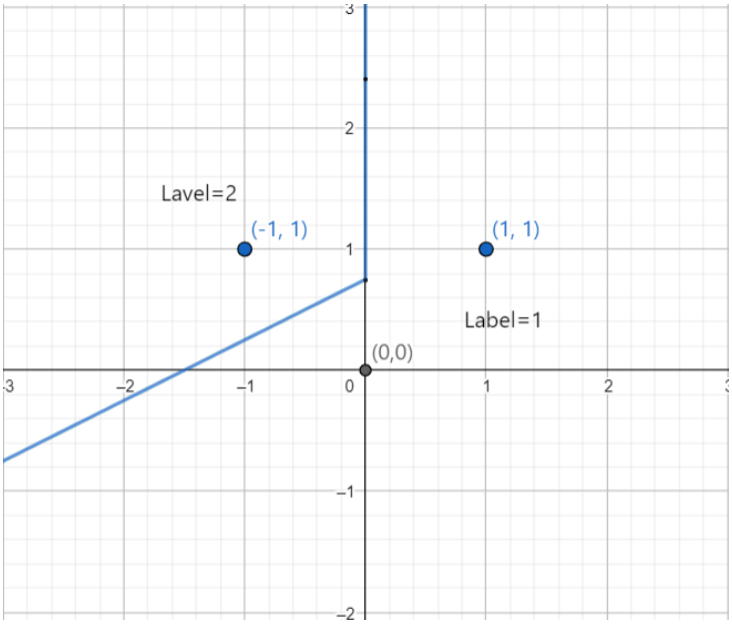
Draw with GeoGabra

d) مشابه بخش b و c ، معادله مرزهای تصمیم را به دست می‌آوریم:

$$\begin{cases} H_{13} = \frac{1}{2}(x-0)^2 + (y-0)^2 - \frac{1}{2}(x+1)^2 - (y-1)^2 = 0 \Rightarrow H_{13} = -x + 2y - 1.5 = 0 \\ H_{23} = \frac{1}{2}(x-1)^2 + (y-1)^2 - \frac{1}{2}(x+1)^2 - (y-1)^2 = 0 \Rightarrow H_{23} = x = 0 \end{cases}$$

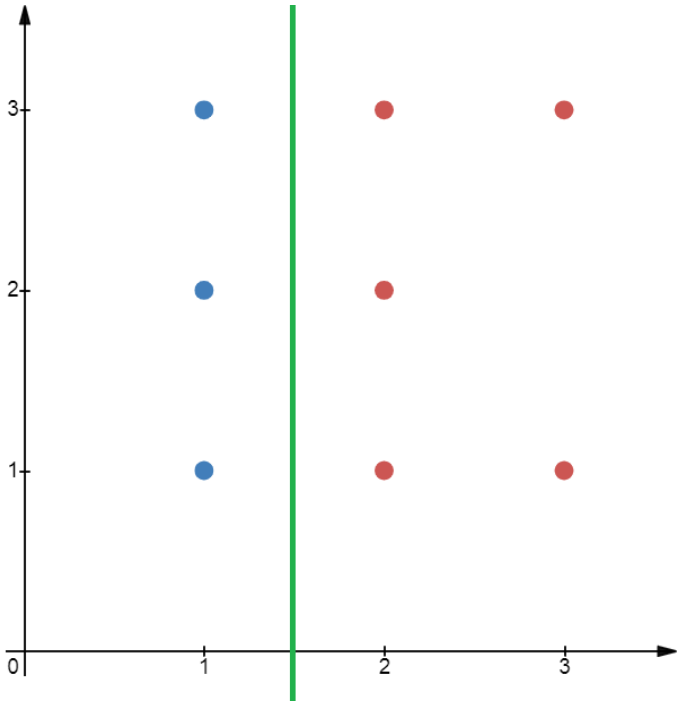
معادلات بالا خطی هستند و نقطه برخورد آن‌ها (0,0.75) می‌باشد. بنابراین مرزهای تصمیم به صورت زیر رسم می‌شود:

$H_{13} = -x + 2y - 1.5 = 0 \qquad plot\ for\ x < 0$
$H_{23} = x = 0 \qquad plot\ for\ y > 0.75$

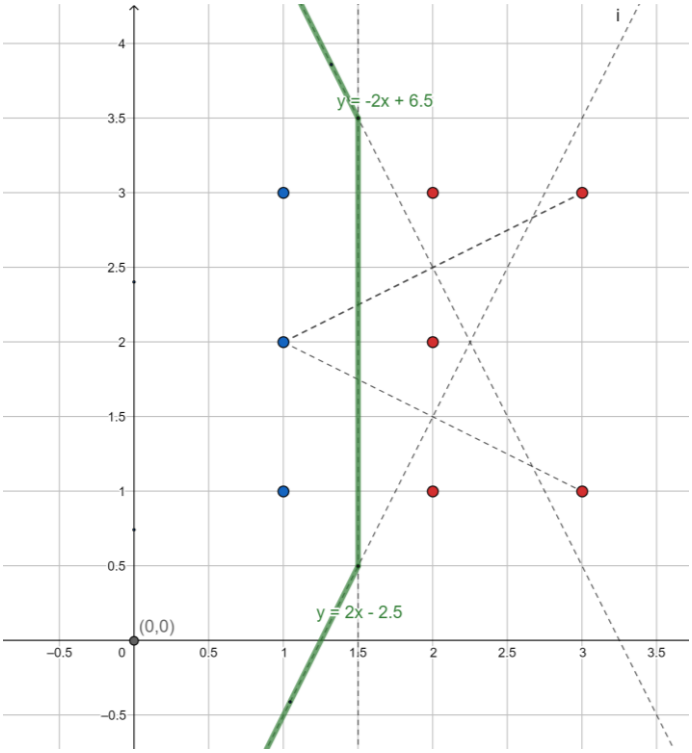


Draw with GeoGabra

e) برای k=1، مرز تصمیم به صورت زیر است. (خط x=1.5)



برای k=3 خطوط مرز تصمیم به صورت زیر رسم می‌شود: (مرزهای زیر به صورت دستی با نرم‌افزار GeoGabra کشیده‌شده‌اند.خطوط نقطه چین خطوط کمکی هستند)



f) با توجه به شکل‌های بالا، برچسب نقاط به این صورت می‌باشد:

x	y	Label (k=1)	Label (k=3)
1.3	4	Blue	Red
1.5	0.4	Blue & Red	Red
1.4	3.5	Blue	Blue

g) میانگین k نزدیک‌ترین همسایه هرنقطه، به عنوان برچسب نقطه در نظر گرفته می‌شود. در مواردی که ۲ نقطه فاصله یکسان دارند، نقطه‌ای که مقدار x آن کوچکتر است انتخاب می‌شود.

point	1-NN	Predicted (k=1)	3-NN	Predicted (k=3)	Real Values
0.5	(1,1.5)	1.5	(1,1.5) (1.5,3) (2,2.25)	2.25	0.75
1	(0.5,0.75)	0.75	(0.5,0.75) (1.5,3) (2,2.25)	2	1.5
1.5	(1,1.5)	1.5	(0.5,0.75) (1,1.5) (2,2.25)	1.5	3
2	(1.5,3)	3	(1,1.5) (1.5,3) (2.5,1)	1.833	2.25
2.5	(2,2.25)	2.25	(1.5,3) (2,2.25) (3,0.5)	1.916	1
3	(2.5,1)	1	(1.5,3) (2,2.25) (2.5,1)	2.083	0.5

$$MAE\ for\ 1-NN:\ error = \frac{1}{n}\sum_{1}^{n}|predicted - real| = 0.916$$

$$MAE\ for\ 3-NN:\ error = \frac{1}{n}\sum_{1}^{n}|predicted - real| = 1.069$$

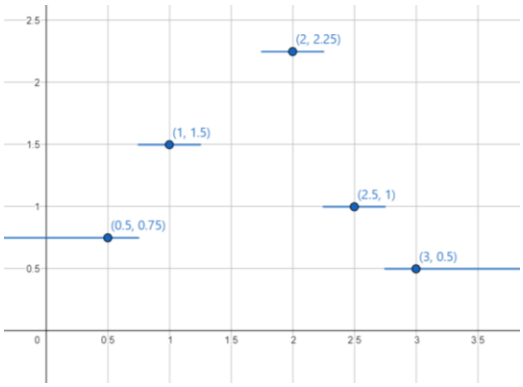
h) مانند بخش قبل، در مواردی که ۲ نقطه فاصله یکسان دارند، نقطه‌ای که مقدار x آن کوچکتر است انتخاب می‌شود.

Point	Predicted (k=1)	Predicted (k=3)
0.75	0.75	$\frac{0.75+1.5+3}{3} = 1.75$
1.75	3	$\frac{1.5+3+2.25}{3} = 2.25$
2.25	2.25	$\frac{3+2.25+1}{3} = 2.083$

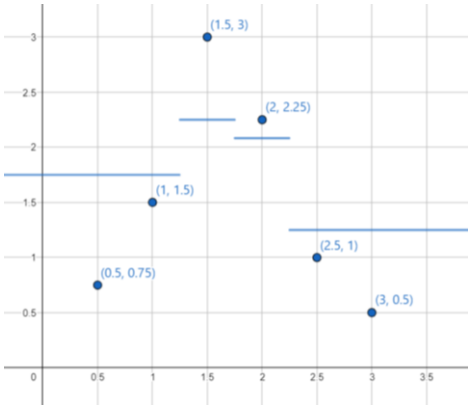
i) برای بازه های مختلف، مقادیر پیش‌بینی‌شده را مانند قسمت های قبل محاسبه می‌کنیم. فرضی که درمورد نقاط با فاصله یکسان وجود داشت در اینجا نیز برقرار است.

$$1-NN:\ y = \begin{cases} 0.75 & x \le 0.75 \\ 1.5 & 0.75 < x \le 1.25 \\ 3 & 1.25 < x \le 1.75 \\ 2.25 & 1.75 < x \le 2.25 \\ 1 & 2.25 < x \le 2.75 \\ 0.5 & 2.75 < x \end{cases} \qquad 3-NN:\ y = \begin{cases} 1.75 & x \le 1.25 \\ 2.25 & 1.25 < x \le 1.75 \\ 2.083 & 1.75 < x \le 2.25 \\ 1.25 & 2.25 < x \end{cases}$$

با استفاده از ابزار GeoGabra، توابع فوق را رسم می‌کنیم. برای k=1:



برای k=3:



۳)

---

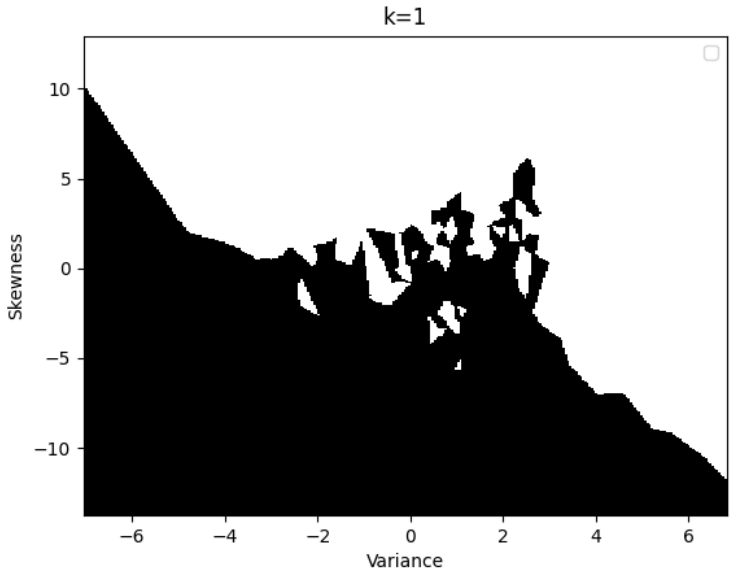
a)کد مربوط به این بخش در فایل a.py قرار دارد. داده های موجود در دیتاست، بر حسب خروجی مرتب هستند؛ یعنی داده هایی که خروجی آنها ۰ است در ابتدای دیتاست و داده‌هایی که خروجی آنها ۱ است در انتهای آن قرار دارند. بدیهی است که در این حالت الگوریتم به درستی کار نخواهد کرد؛ زیرا ۵۰۰ داده اول فقط شامل کلاس ۰ است. بنابراین داده ها را به کمک فایل shafling.py شافل کرده و در فایل s_data.csv ذخیره می‌کنیم. از این پس از s_data.csv به عنوان دیتاست استفاده می‌کنیم.

الگوریتم را به وسیله تابع feature_validation که در **خط ۶۷ تعریف و در خط ۸۰ فراخونی** شده است، روی همه جفت ویژگی‌ها اجرا می‌کنیم و دقت را به ازای هر جفت‌ویژگی محاسبه می‌کنیم. خروجی به شرح زیر است:

	accuracy
Variance,Skewness	0.949541
Variance,Kurtosis	0.887615
Variance,Entropy	0.886468
Skewness,Kurtosis	0.896789
Skewness,Entropy	0.844037
Kurtosis,Entropy	0.701835

بنابراین دو ویژگی Variance و Skewness بهترین عملکرد را برای جداسازی کلاس‌ها دارند.

b) کد مربوط به این بخش در فایل b.py قراردارد. در خط آخر این فایل تابع اصلی برنامه فراخوانی می‌شود. آخرین آرگومان این تابع رزولوشن تصویر خروجی را مشخص می‌کند بنابراین تاثیر زیادی در سرعت اجرای برنامه دارد. لازم به ذکر است رنگ مشکی مربوط به داده های inauthentic است.

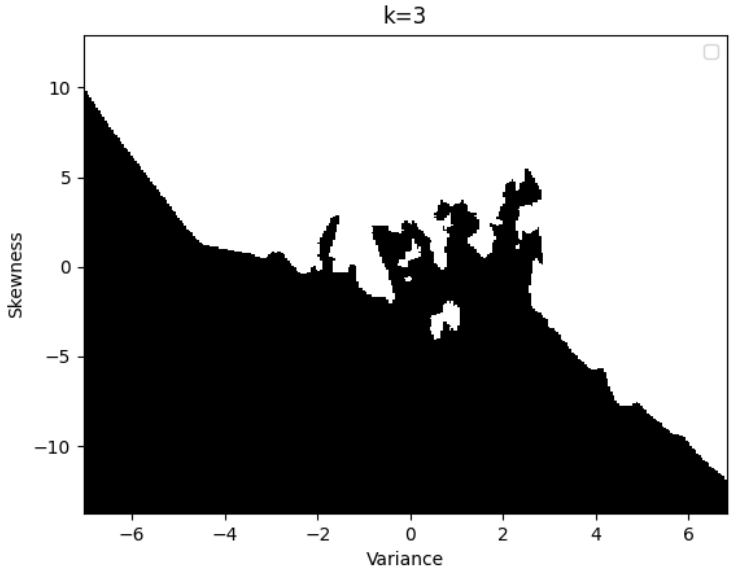


c) فایل a.py را مجددا اجرا می‌کنیم. بدین منظور تنها کافی است آرگومان مربوط به k در خط آخر را ۳ قرار دهیم. خروجی به صورت زیر است.

	accuracy
Variance,Skewness	0.949541
Variance,Kurtosis	0.892202
Variance,Entropy	0.900229
Skewness,Kurtosis	0.908257
Skewness,Entropy	0.877294
Kurtosis,Entropy	0.713303

مجددا دو ویژگی Variance و Skewness بهترین عملکرد را برای جداسازی کلاس‌ها دارند.

d) فایل b.py را مجددا اجرا می‌کنیم. این بار مقدار k را در فراخونی انتهای کد ۳ قرار می‌دهیم. خروجی به این صورت است:



همانطور که مشاهده می‌شود، پیچیدگی مرز تصمیم نسبت به حالت k=1 کمتر شده است. به بیان بهتر، واریانس مدل کاهش یافته است.

۴)

---

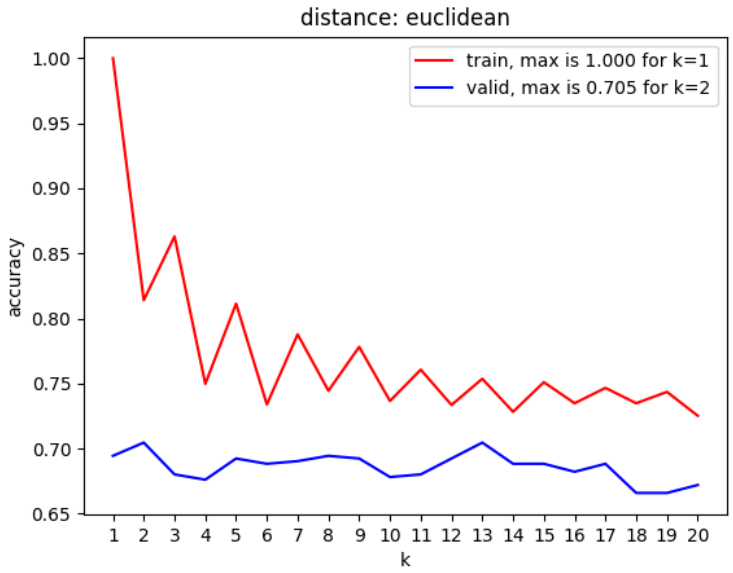
a) کد مربوط به این بخش در فایل a.py قرار دارد. تابع مورد نظر در **خط ۳۰ تعریف شده است**. همچنین برای تقسیم داده‌ها به مجموعه های آموزش، آزمون و اعتبارسنجی، تابعی به نام split_data در خط ۱۹ تعریف شده که در تابع prepare_data فراخوانی می‌شود. در نهایت تابع prepare_data سه مجموعه مجزا(طبق خواسته سوال) و برچسب‌های مربوط به آن‌ها را برمی‌گرداند.

b) کد مربوط به این بخش در فایل b.py قرار دارد. ابتدا خطای مربوط به دو مجموعه آموزش و اعتبارسنجی را نشان می‌دهیم:

k	Train Error	Validation Error
1	0.000000	0.325866
2	0.208224	0.344196
3	0.165792	0.319756
4	0.265529	0.360489
5	0.220035	0.325866
6	0.272966	0.354379
7	0.236220	0.334012
8	0.283465	0.358452
9	0.247594	0.338086
10	0.283027	0.340122
11	0.248031	0.327902
12	0.285652	0.340122
13	0.262905	0.315682
14	0.288714	0.344196
15	0.260280	0.311609
16	0.294401	0.317719
17	0.271216	0.315682
18	0.303150	0.329939
19	0.274278	0.307536
20	0.297900	0.342159

همانطور که مشاهده می‌شود برای مجموعه اعتبار سنجی، خطاها دارای نظم خاصی نیستند؛ بنابراین نمی‌توان با قطعیت گفت کدام مقدار k بهترین است. نکته قابل توجه دیگر این است خطا برای k=1 در مجموعه آموزش صفر است. دلیل این موضوع این است که داده هایی که از الگوریتم پرسیده می‌شود، همان داده هایی است که به عنوان آموزش به آن داده شده است. بنابراین عملا برای k=1 هر نقطه دقیقا با مقدار واقعی خود مقایسه می‌شود. به هرحال از آنجا که الگوریتم KNN دارای فاز آموزش نیست، تعریف خطای آموزش برای آن مناسب نیست.

در ادامه دقت بر حسب خطا را نمایش می‌دهیم:

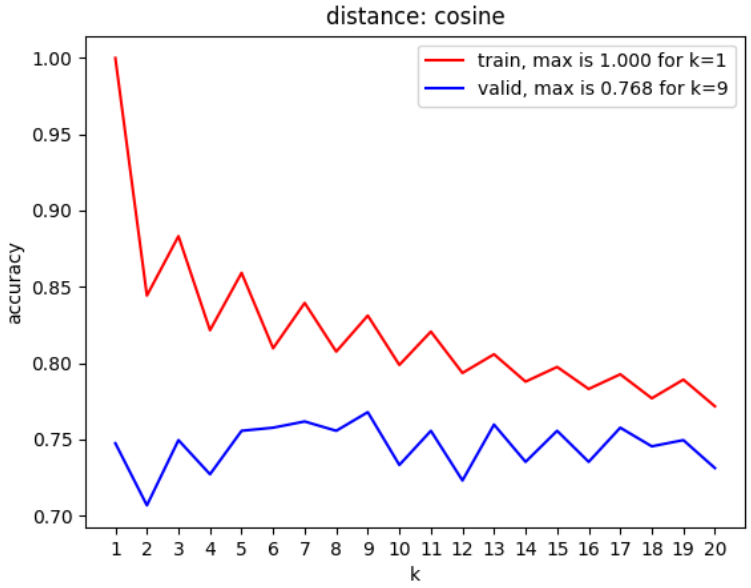


همانطورکه گفته شد، نمی‌توان به طور قطع مقدار مناسبی برای k پیشنهاد داد؛ اما در مجموعه فعلی، همانطور که در شکل مشخص است. K=2 با خطای 0.705 بهترین عملکرد را دارد. اما این مقدار با اجرای مجدد الگوریتم تغییر می‌کند. زیرا تابع prepare data مجموعه های آموزش، آزمون و اعتبارسنجی را تصادفی انتخاب می‌کند.

c) کد مربوط به این بخش در فایل c.py قراردارد. نتایج به صورت زیر است:

k	Train Error	Validation Error
1	0.000000	0.252546
2	0.155731	0.293279
3	0.116798	0.250509
4	0.178478	0.272912
5	0.140857	0.244399
6	0.190289	0.242363
7	0.160542	0.238289
8	0.192476	0.244399
9	0.168854	0.232179
10	0.201225	0.266802
11	0.179353	0.244399
12	0.206474	0.276986
13	0.194226	0.240326
14	0.212161	0.264766
15	0.202537	0.244399
16	0.216973	0.264766
17	0.207349	0.242363
18	0.223097	0.254582
19	0.210849	0.250509
20	0.228346	0.268839

برای فاصله کسینوسی هم نمی‌توان به طور قطع مقدار مناسبی برای k پیشنهاد داد. اما مشخصاً عملکرد آن به صورت کلی از فاصله اقلیدسی بهتر است. نمودار دقت بر حسب k به این صورت است.



در اینجا بهترین عملکرد را k=9 با خطای 0.768 دارد. البته قبلا نیز اشاره شد که چون الگوی خاصی برای نمودار خطا وجود ندارد، نمی‌توان به طور قطع بهترین مقدار را برای k اعلام کرد.

c) فاصله کیسنوسی زاویه بین دوبردار را نشان می‌دهد. اما فاصله اقلیدسی، فاصله آن‌ها را نشان می‌دهد. بنابراین فاصله کسینوسی صرف‌نظر از اندازه بردارها، شباهت آنها را نشان می‌دهد بنابراین در این مسئله بهتر است. برای توضیح بیشتر دیتاست نمونه ای که به عنوان راهنما داده‌شده‌است در نظر می‌گیریم.

Data	Feature 1 : count(dog)	Feature 2 : count(CR7)
‘dog’	1	0
‘CR7’	0	1
‘dog dog dog’	3	0

با توجه به جدول فوق، معیار اقلیدسی، فاصله بین نمونه‌های ۱ و ۲ را کمتر نشان می‌دهد. از طرفی فاصله کسینوسی فاصله بین نمونه های ۱ و ۳ را کمتر نشان می‌دهد (زیرا زاویه بین دو بردار صفر است). به وضوح مشخص است که نمونه ۳ و ۱ باهم شباهت بیشتری دارند؛ بنابراین فاصله کسینوسی بهتر عمل می‌کند. در واقع از آنجا که فاصله اقلیدسی طول نمونه ها را نیز در نظر می‌گیرد معیار مناسبی برای این مسئله نیست.

۵)

---

a) با توجه به داده های صورت سوال داریم:

$$S_w = \Sigma_1 + \Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \Rightarrow v = S_w^{-1}(\mu_2 - \mu_1) = \begin{bmatrix} -18 \\ 0 \end{bmatrix} \sim \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

توجه شود که تعداد نمونه ها برابر فرض شده است. بنابراین نیازی نیست برای محاسبه S_w، ماتریس‌های کوواریانس در تعداد نمونه ها ضرب شود، زیرا در این حالت تعداد نمونه ها به صورت ضریبی در v ظاهر می شود و می توان از آن صرف‌نظر کرد(جهت بردار v برای ما مهم است نه طول آن). بردار فوق، عمود بر خط جداکننده می‌باشد. بنابراین w=v

b) توزیع‌ها را روی بردار v قسمت قبل،تصویر می‌کنیم.

$$\begin{cases} \mu_1' = v^t \mu_1 = 3 \\ \Sigma_1' = v^t \Sigma_1 v = 1.5 \\ \mu_2' = v^t \mu_2 = -3 \\ \Sigma_2' = v^t \Sigma_2 v = 1.5 \end{cases}$$

با توجه به صورت سوال $p(c_1)=0.231$ و $p(c_2)=0.769$. از طرفی درفضای جدید، نقطه ای که در آن احتمال دو کلاس یکسان است از رابطه زیر به دست می‌آید.

$$p(c_1)f_1(x) - p(c_2)f_2(x) = 0 \Rightarrow 0.231e^{\frac{1}{2}\left(\frac{x-3}{1.5}\right)^2} = 0.769e^{\frac{1}{2}\left(\frac{x+3}{1.5}\right)^2} \Rightarrow x = -0.451$$

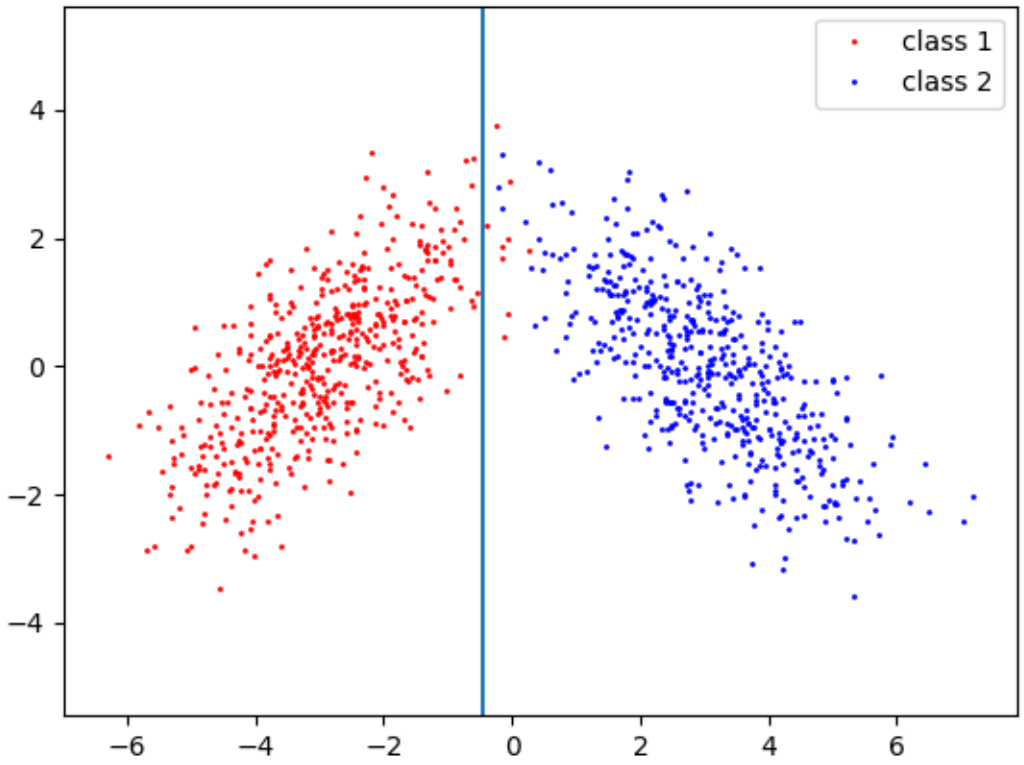
در معادله بالا منظور از f_i ، توزیع نرمال با پارامترهای کلاس i می‌باشد. مقدار به دست آمده برای x همان w_0 می باشد.بنابراین معادله جداکننده برابر است با:

$$d(x) = w^t x + w_0 = -x_1 - 0.451 \qquad if\ d > 0\ \ x \in 1\ \ o.w\ \ \ x \in 2$$

c) با توجه به معادله تابع جداکننده بالا:

$$d([-0.5,0.25]^t) = +0.5 - 0.451 = 0.049 > 1 \quad x \in 1$$

d) کد مربوط به این بخش در فایل d.py قرار دارد. خروجی به شکل زیر است:



مطابق انتظار، ازآنجاکه احتمال پیشین کلاس ۲ بیشتر است، مرز تصمیم به مرکز کلاس ۱ نزدیکتر است.

e) ابتدا ویژگی ۱ را به همه نقاط اضافه می‌کنیم و عمل نرمال‌سازی را روی بردارهای بدست آمده اعمال می‌کنیم:

$$\begin{cases} y_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ y_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ y_3 = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} \\ y_4 = \begin{bmatrix} -1 \\ -2 \\ 1 \end{bmatrix} \end{cases}$$

حال الگوریتم Perceptron به روش single sample اعمال می‌کنیم. در هرمرحله فقط اولین نمونه‌ای که اشتباه دسته‌بندی شده ذکر می شود:

$\boldsymbol{step\ 1}: \boldsymbol{w_0} = (\mathbf{0},\mathbf{0},\mathbf{0})^{\boldsymbol{t}} \quad \boldsymbol{\rho} = \mathbf{1}$

$w_0^t y_1 = 0 \Rightarrow\ w_1 = w_0 + y_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}$

$\boldsymbol{step\ 2}: \boldsymbol{w_1} = (\mathbf{1},\mathbf{0},\mathbf{0})^{\boldsymbol{t}}$

$w_1^t y_3 = -1 \Rightarrow\ w_2 = w_1 + y_3 = \begin{bmatrix}0\\-1\\0\end{bmatrix}$

$\boldsymbol{step\ 3}: \boldsymbol{w_2} = (\mathbf{0},\mathbf{-1},\mathbf{0})^{\boldsymbol{t}}$

$w_2^t y_1 = 0 \Rightarrow\ w_3 = w_2 + y_1 = \begin{bmatrix}1\\-1\\0\end{bmatrix}$

$\boldsymbol{step\ 4}: \boldsymbol{w_3} = (\mathbf{1},\mathbf{-1},\mathbf{0})^{\boldsymbol{t}}$

$w_3^t y_2 = 0 \Rightarrow\ w_4 = w_3 + y_2 = \begin{bmatrix}2\\0\\1\end{bmatrix}$

$\boldsymbol{step\ 5}: \boldsymbol{w_4} = (\mathbf{2},\mathbf{0},\mathbf{1})^{\boldsymbol{t}}$

$w_4^t y_3 = -2 \Rightarrow w_5 = w_4 + y_3 = \begin{bmatrix}1\\-1\\1\end{bmatrix}$

$\boldsymbol{step\ 6}: \boldsymbol{w_5} = (\mathbf{1},\mathbf{-1},\mathbf{1})^{\boldsymbol{t}}$

$w_5^t y_3 = 0 \Rightarrow w_6 = w_5 + y_3 = \begin{bmatrix}0\\-2\\1\end{bmatrix}$

$\boldsymbol{step\ 7}: \boldsymbol{w_6} = (\mathbf{0},\mathbf{-2},\mathbf{1})^{\boldsymbol{t}}$

$w_6^t y_1 = 0 \Rightarrow w_7 = w_6 + y_1 = \begin{bmatrix}1\\-2\\1\end{bmatrix}$

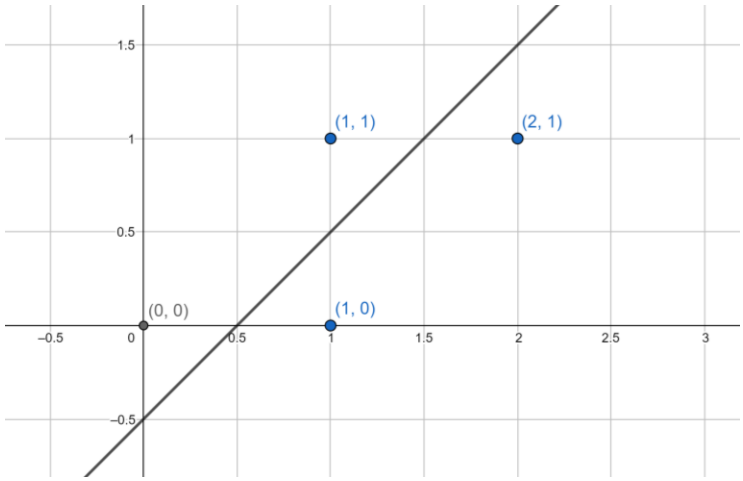
$\boldsymbol{step\ 8}: \boldsymbol{w_7} = (\mathbf{1},\mathbf{-2},\mathbf{1})^{\boldsymbol{t}}$

$w_7^t y_2 = 0 \Rightarrow w_8 = w_7 + y_2 = \begin{bmatrix}2\\-1\\2\end{bmatrix}$

$\boldsymbol{step\ 9}: \boldsymbol{w_8} = (\mathbf{2},\mathbf{-1},\mathbf{2})^{\boldsymbol{t}}$

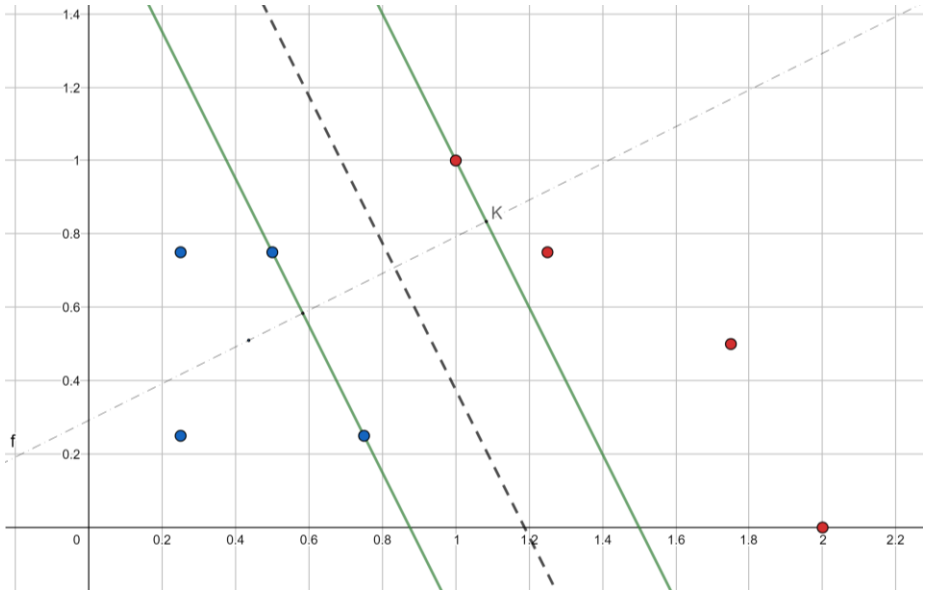
$w_8^t y_3 = -1 \Rightarrow w_9 = w_8 + y_3 = \begin{bmatrix}1\\-2\\2\end{bmatrix}$

بردار $\mathbf{w}_9$ همه نمونه‌ها را به درستی دسته‌بندی می‌کند. در ادامه آن را رسم می‌کنیم:



Draw with GeoGabrax

f) نقاط را رسم می‌کنیم و سعی می‌کنیم بردارهای پشتیبان را حدس بزنیم. سپس معادله خط را با رسم خطوط موازی می‌یابیم:



Draw with GeoGabrax

باتوجه به اینکه نقاط مربوط به حاشیه های خیابان را داریم به سادگی می توان معادله آن ها را بدست آورد. سپس معادله خط جداکننده(وسط خیابان) بدست می‌آید. با انجام این کار معادله خط جداکننده به صورت y=-2x+2.375 بدست می آید. بنابراین بردار وزن مربوط به این خط برابر است با $w = \begin{bmatrix} -2.375 \\ 2 \\ 1 \end{bmatrix}$

g) با توجه به قسمت قبل، بردارهای پشتیبان عبارتند از: (0.5,0.75) و (0.75,0.25) از کلاس یک و (1,1) از کلاس دو

h) ضرایب لاگرانژ را برای بردارهای پشتیبانی قسمت قبل می یابیم:

$$L(\alpha) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i . x_j + \sum_i \alpha_i \,,\ \ s.t.\ x_i\ is\ a\ SV\ \&\ a_i > 0\ \&\ \sum a_i y_i = 0$$

$$L(\alpha) = -\frac{1}{2}\left(\alpha_1 \begin{bmatrix} 0.5 \\ 0.75 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix} - \alpha_3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)\left(\alpha_1 \begin{bmatrix} 0.5 \\ 0.75 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix} - \alpha_3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) + \alpha_1 + \alpha_2 + \alpha_3$$

$$= -\frac{1}{2}\begin{bmatrix} 0.5\alpha_1 + 0.75\alpha_2 - \alpha_3 \\ 0.75\alpha_1 + 0.25\alpha_2 - \alpha_3 \end{bmatrix}^t \begin{bmatrix} 0.5\alpha_1 + 0.75\alpha_2 - \alpha_3 \\ 0.75\alpha_1 + 0.25\alpha_2 - \alpha_3 \end{bmatrix} + \alpha_1 + \alpha_2 + \alpha_3 =$$

$$= \frac{-13}{32}\alpha_1^2 - \frac{9}{16}\alpha_1\alpha_2 + \frac{5}{4}\alpha_1\alpha_3 - \frac{5}{16}\alpha_2^2 - \alpha_3^2 + \alpha_2\alpha_3 + \alpha_1 + \alpha_2 + \alpha_3 \xrightarrow{\alpha_1+\alpha_2-\alpha_3=0}$$

$$\frac{-13}{32}\alpha_1^2 - \frac{9}{16}\alpha_1\alpha_2 + \frac{5}{4}\alpha_1(\alpha_1+\alpha_2) - \frac{5}{16}\alpha_2^2 - (\alpha_1+\alpha_2)^2 + \alpha_2(\alpha_1+\alpha_2) + 2\alpha_1 + 2\alpha_2$$

$$\begin{cases} \dfrac{\partial y L(\alpha)}{\partial \alpha_1} = -\dfrac{26}{32}\alpha_1 - \dfrac{9}{16}\alpha_2 + \dfrac{10}{4}\alpha_1 + \dfrac{5}{4}\alpha_2 - 2(\alpha_1+\alpha_2) + \alpha_2 + 2 = 0 \\ \dfrac{\partial y L(\alpha)}{\partial \alpha_2} = -\dfrac{10}{16}\alpha_2 - \dfrac{9}{16}\alpha_1 + \dfrac{5}{4}\alpha_1 - 2(\alpha_1+\alpha_2) + \alpha_1 + 2\alpha_2 + 2 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} -\dfrac{5}{16}\alpha_1 - \dfrac{5}{16}\alpha_2 + 2 = 0 \\ -\dfrac{5}{16}\alpha_1 + \dfrac{5}{8}\alpha_2 + 2 = 0 \end{cases} \Rightarrow \begin{cases} \alpha_1 = \dfrac{32}{5} = 6.4 \\ \alpha_2 = 0 \\ \alpha_3 = 6.4 \end{cases}$$

حال با توجه به ضرایب فوق، بردار وزن را می‌یابیم.

$$w = \sum_i \alpha_i y_i x_i = 6.4 \times 1 \times \begin{bmatrix} 0.5 \\ 0.75 \end{bmatrix} + 6.4 \times -1 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -3.2 \\ -1.6 \end{bmatrix}$$

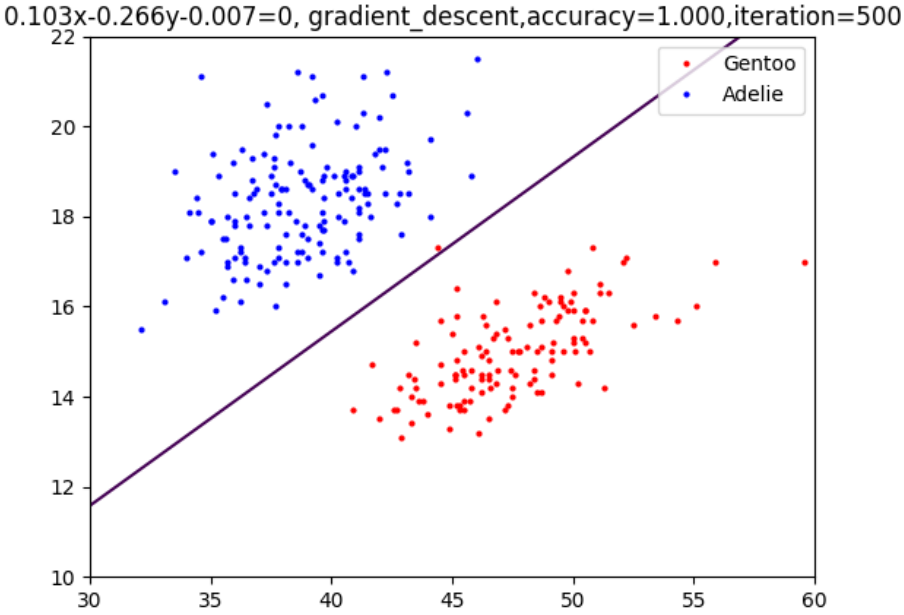
$$y_i(w.x_i + w_0) - 1 = 0 \Rightarrow -3.2 \times 0.5 - 1.6 \times 0.75 + w_0 - 1 = 0 \Rightarrow w_0 = 3.8$$

بنابراین بردار نهایی وزن به صورت $\begin{bmatrix} 3.8 \\ -3.2 \\ -1.6 \end{bmatrix}$ می‌باشد. درباره بردارهای وزن می‌دانیم که جهت آن‌ها مهم است نه مقدار آن‌ها؛ بنابراین با تقسیم آن بر 1.6- بردار وزن به $\begin{bmatrix} -2.375 \\ 2 \\ 1 \end{bmatrix}$ تبدیل می‌شود؛ که دقیقا برابر با مقدار پیش‌بینی شده در قسمت f است.

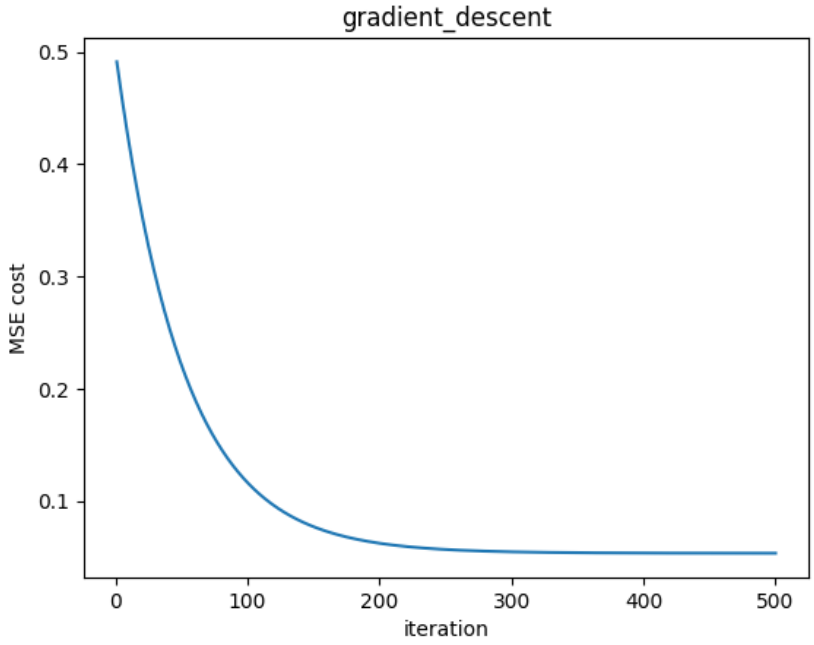
۶)

*** در تمام قسمت های این سوال از رویکرد batch استفاده شده است. همچنین ردیف های شامل missing values حذف شده اند.**

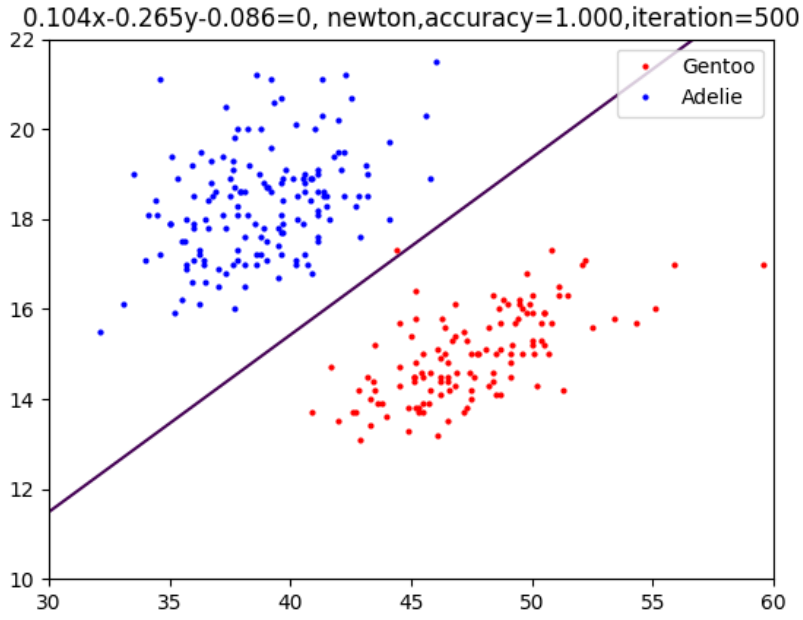
a)کد مربوط به این قسمت در فایل a.py قراردارد. داده‌ها نیاز به شافل شدن دارند. این کار به وسیله فایل shafling.py انجام شده و نتایج در فایل s_data.csv ذخیره شده‌اند. در این سوال از تابع هزینه MSE به صورت $\frac{1}{n}\sum(X\theta - b)^2$ استفاده می‌شود که در آن n تعداد نمونه‌ها، X ماتریس نمونه‌های Augmented شده و تتا بردار وزن می‌باشد. حال به بررسی خروجی ها می‌پردازیم. در شکل زیر شکل مرز تصمیم الگوریتم گرادیان نزولی به همراه توزیع داده‌ها و همچنین دقت روی داده‌های تست نشان داده‌شده‌است. لازم به ذکر است مقدار ضریب یادگیری ۰.۱ که در صورت سوال داده شده، منجر به واگرایی الگوریتم می‌شد. بنابراین پس از بررسی مقدار پارامتر ۰.۰۰۰۹ در نظر گرفته شد.



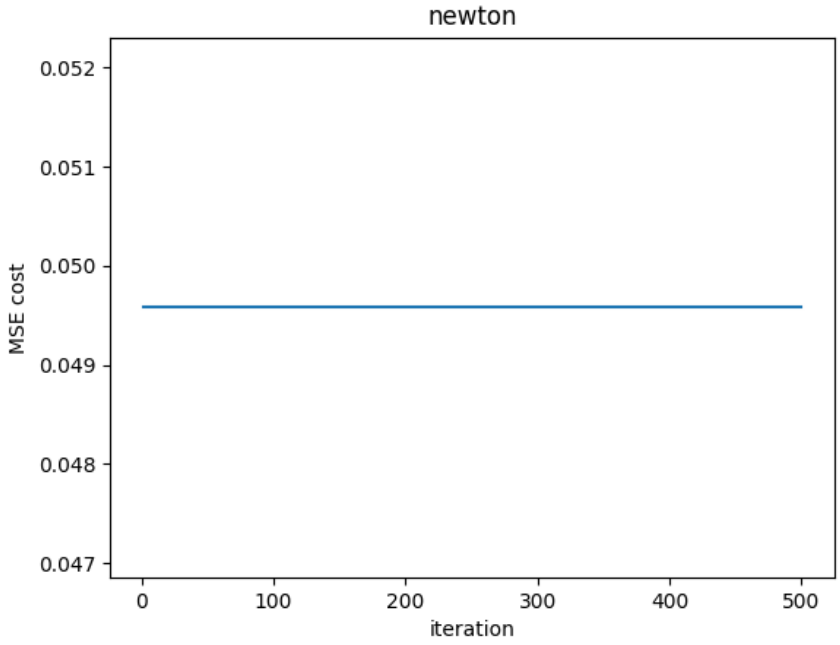
در این قسمت تعداد تکرار را به صورت دستی به الگوریتم می‌دهیم. همانطور که مشاهده می‌شود الگوریتم تنها یک نقطه را اشتباه دسته‌بندی کرده است. دلیل اشتباه دسته‌بندی شدن این داده، احتمالا این است که در این الگوریتم تمام درایه های ماتریس b ، برابر با ۱ در نظر گرفته شده است. برای دسته بندی کاملا دقیق می‌توان از نرمال سازی(با میانگین و واریانس) و یا رویکردهای پیشرفته تر با b متغیر استفاده کرد. به هرحال دقت داده‌های تست ۱۰۰ درصد می‌باشد که نشان می‌دهد داده ها به‌خوبی جداپذیر هستند. در ادامه نمودار هزینه برحسب تکرار را رسم می‌کنیم:



مطابق شکل الگوریتم پس از حدود ۲۰۰ تکرار به مینیمم تابع هزینه می‌رسد. در شکل زیر مرز تصمیم مربوط به الگوریتم نیوتون رسم می‌شود:



جهت مقایسه تعداد تکرار الگوریتم نیوتون را نیز ۵۰۰ وارد کرده‌ایم. مطابق شکل نتایج مشابه الگوریتم گرادیان نزولی می‌باشد. در انتها به بررسی نمودار تابع هزینه برحسب تکرار برای الگوریتم نیوتون می‌پردازیم:

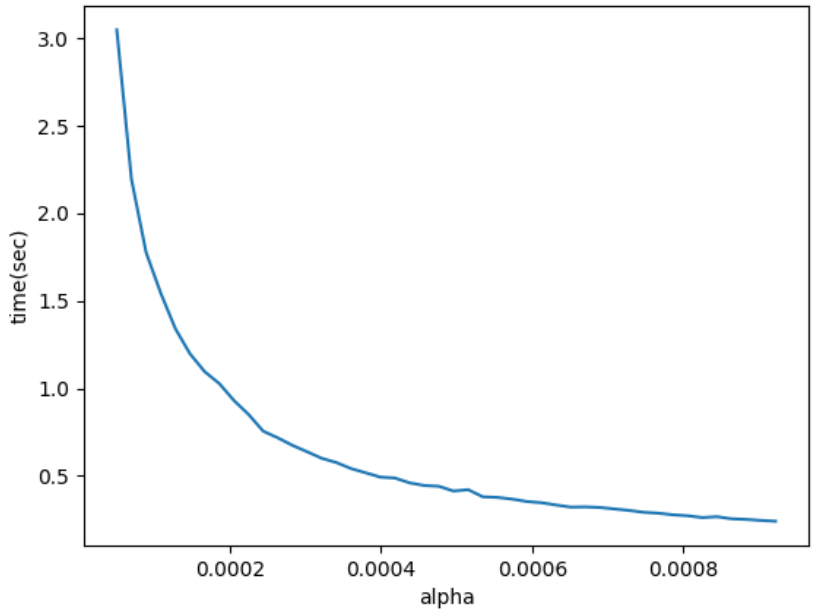


این نمودار نشان می دهد که الگوریتم نیوتون با یک تکرار، به مقدار مینیمم خود رسیده است. این نتیجه اگرچه عجیب به نظر می‌رسد، اما می‌توان اثبات کرد که کاملا منطقی است.اگر f را تابع هزینه MSE بنامیم. الگوریتم نیوتون برای مینیمم کردن MSE به صورت زیر عمل می‌کند.

$$\theta_{k+1} = \theta_k - \frac{f'(\theta_k)}{f''(\theta_k)} \xrightarrow{f(\theta_k)=\frac{1}{n}\sum(X\theta_k - b)^2} \theta_1 = \theta_0 - \frac{\frac{2}{n}X^t(X\theta_0 - b)}{\frac{2}{n}X^tX} \xrightarrow{\theta_0=0} \theta_1 = (X^tX)^{-1}X^tb$$

و می‌دانیم این معادله فرم نرمال برای جواب مقادیر تتا می‌باشد که مستقیما از صفر قراردادن مشتق به دست می‌آید و نیاز به تکرار ندارد. پس ثابت می‌شود الگوریتم نیوتون برای تابع MSE و مقدار اولیه صفر برای ضرایب تتا، در یک گام به جواب می‌رسد.

b) کد مربوط به این قسمت در فایل b.py قراردارد. می توان نشان داد با تعداد تکرار مناسب، مقدار حداقلی برای ضریب یادگیری وجود ندارد. بنابراین فرض می‌شود منظور سوال حداکثر ضریب یادگیری بوده است. با درنظر گرفتن ۵۰۰۰۰ تکرار برای حدبالای تکرار، نمودار را برای ضرایب آلفا در بازه‌ی ۰.۰۰۰۰۵ تا ۰.۰۰۱ رسم می‌کنیم. مطابق خروجی های برنامه، حداکثر ضریب یادگیری که موجب واگرایی می‌شود ۰.۰۰۰۹۶۱۲ می باشد.



باتوجه به نمودار، با افزایش ضریب یادگیری، زمان لازم برای همگرایی کاهش یافته است. این کاهش تا نقطه ۰.۰۰۰۹۶۱۲ ادامه دارد و پس از آن الگوریتم واگرا می‌شود.

c) تعداد عملیات ضرب و جمع ماتریسی را حساب می‌کنیم. حلقه اصلی مربوط به گرادیان نزولی به صورت زیر است:

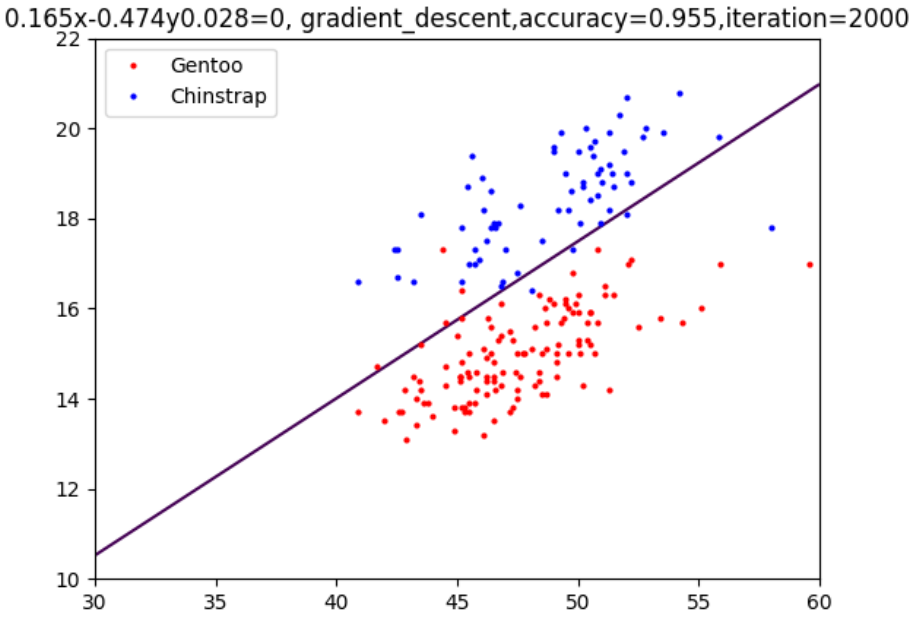
```
49      for iteration in range(iterations):
50          h = X_np.dot(theta)
51          gradient = X_np.T.dot(h-b)*1/(m)
52          theta = theta - alpha*gradient
53          cost_history[iteration] = cost(X,b,theta)
54      return theta, cost_history
```

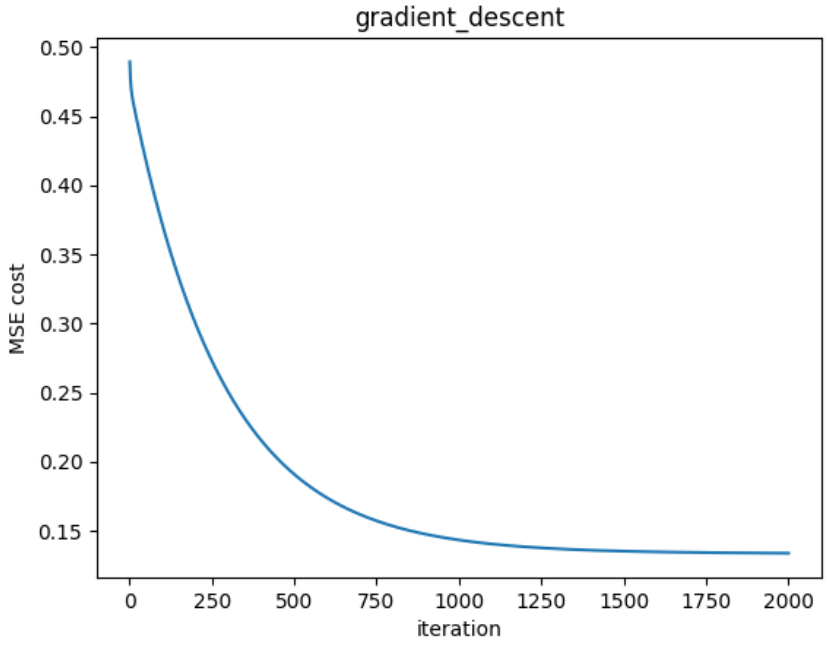
خط ۵۰ یک ضرب ماتریسی، خط ۵۱ یک ضرب و یک جمع ماتریسی و خط ۵۲ یک جمع ماتریسی دارد. فرض می‌شود در حالت عادی نیاز به تاریخچه هزینه نیست بنابراین از خط ۵۳ صرفنظر می‌کنیم. درنهایت اگر تعداد تکرار برای همگرا شدن k باشد، تعداد کل عملیات ماتریسی 2k جمع ماتریسی و 2k ضرب ماتریسی می‌باشد.

برای الگوریتم نیوتون طبق تصویر زیر در خط ۶۲ یک ضرب ماتریسی، در خط ۶۳ یک جمع و یک ضرب ماتریسی، در خط ۶۴ یک ضرب ماتریسی و در خط ۶۵ یک جمع و یک ضرب ماتریسی وجود دارد. قبلا ثابت شد که برای این الگوریتم در این مسئله، با شروع از صفر با یک تکرار به همگرایی می‌رسیم بنابراین تعداد کل عملیات ماتریسی عبارت‌اند از ۴ ضرب ماتریسی و ۲ جمع ماتریسی.

```
61      for iteration in range(iterations):
62          h = X_np.dot(theta)
63          d1 = X_np.T.dot(h-b)*1/(m)
64          d2 = X_np.T.dot(X_np)*(1/m)
65          theta = theta - np.linalg.inv(d2).dot(d1)
66          print(theta)
67          cost_history[iteration] = cost(X,b,theta)
68      return theta, cost_history
```

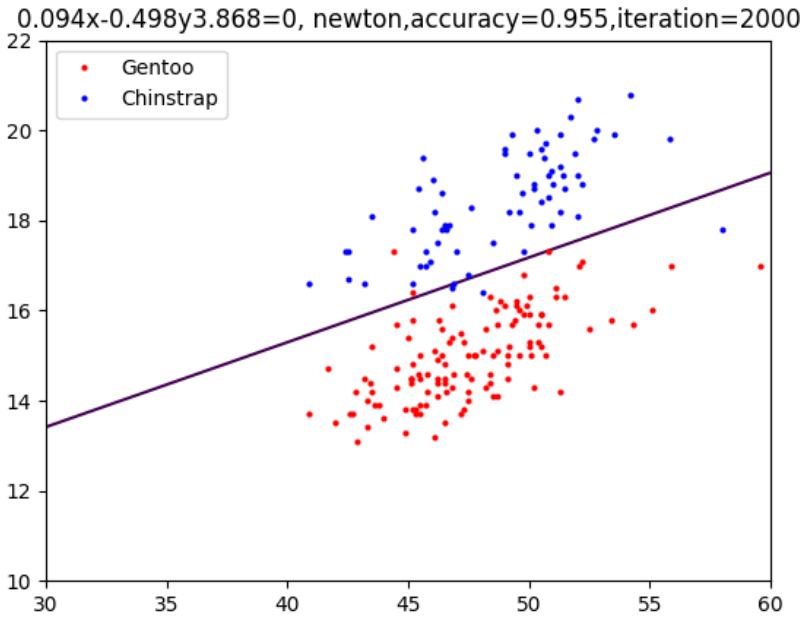
d) از همان فایل های a.py و b.py استفاده می‌کنیم. با این تفاوت که در خط آخر هر دو برنامه، کلاس‌های هدف را به Chinstrap و Gentoo تغییر می‌دهیم.(ضریب یادگیری ۰.۰۰۰۷ و تعداد تکرار ۲۰۰۰) ابتدا مرز تصمیم و سپس تابع هزینه برحسب تکرار را رسم می‌کنیم.

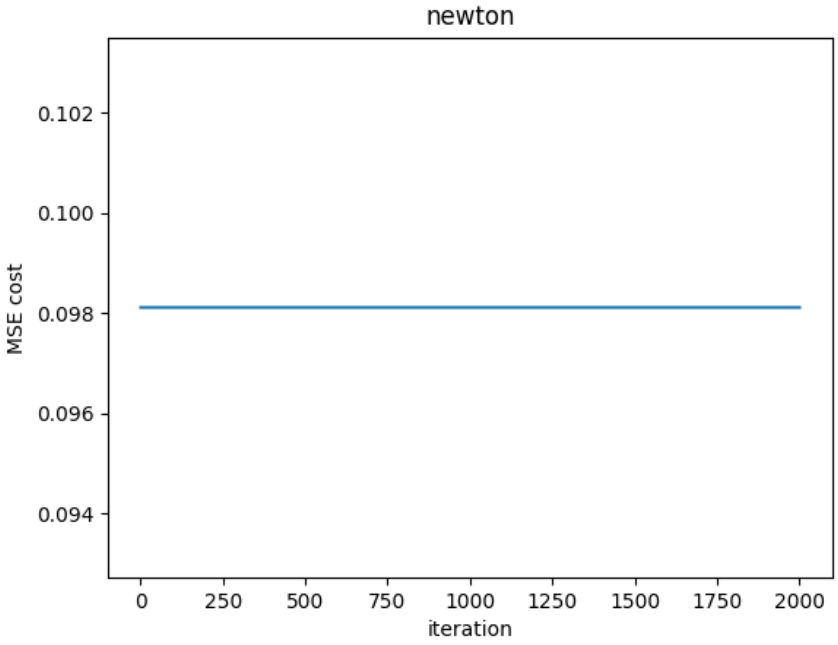




مطابق شکل‌ها، این دو کلاس کمتر جداپذیر هستند. این مورد دو نتیجه مستقیم در نمودارهای فوق داشته است:۱)کاهش دقت روی داده‌های تست ۲)افزایش تعداد تکرار برای همگرا شدن

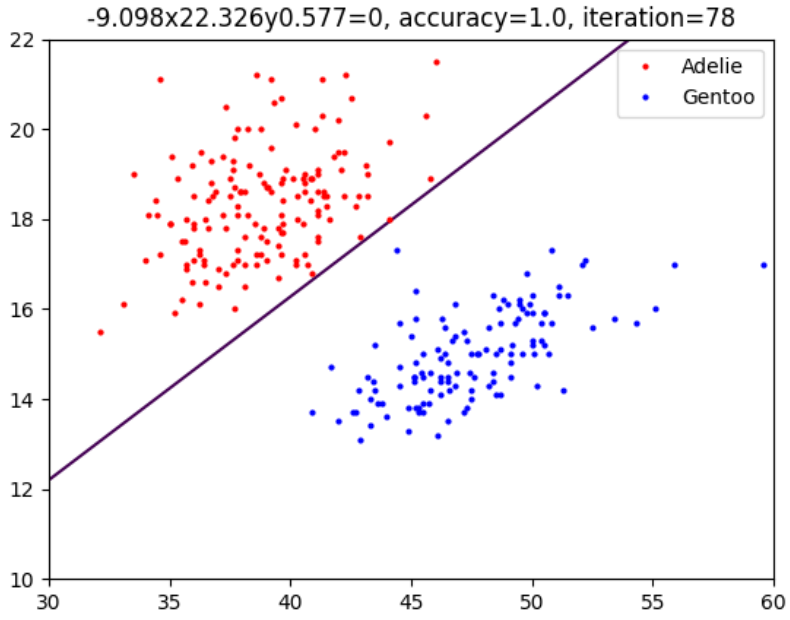
نمودارهای مربوط به الگوریتم نیوتون به این صورت است:



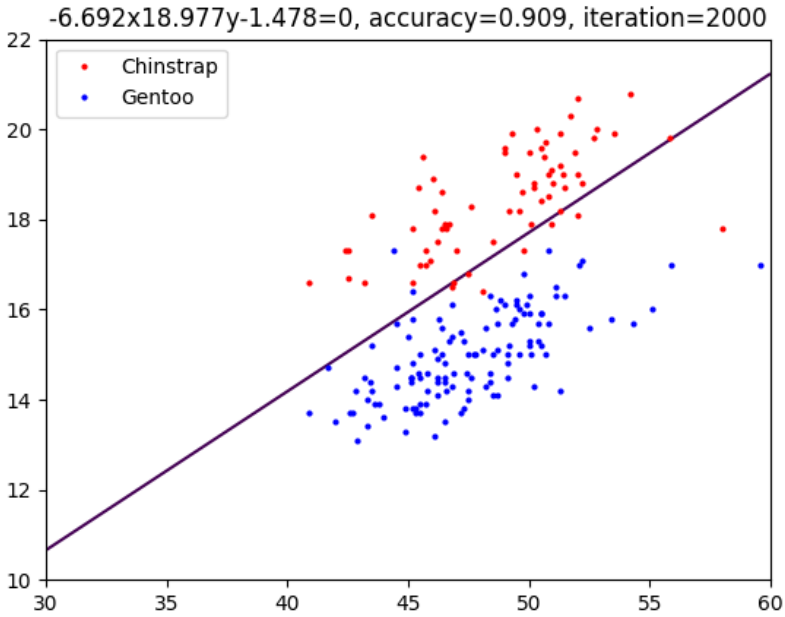


خروجی الگوریتم نیوتون با روش گرادیان نزولی تفاوت اندکی دارد. همچنین دلیل همگرا شدن الگوریتم نیوتون در یک گام قبلا توضیح داد شد.

e) کد مربوط به این قسمت در فایل e.py قراردارد. برای جلوگیری از حلقه بی‌نهایت حدبالای تکرار را ۲۰۰۰ قرار می‌دهیم. تمامی پارامترهای درخواست شده توسط سوال در نمودار زیر و عنوان آن مشخص شده اند:



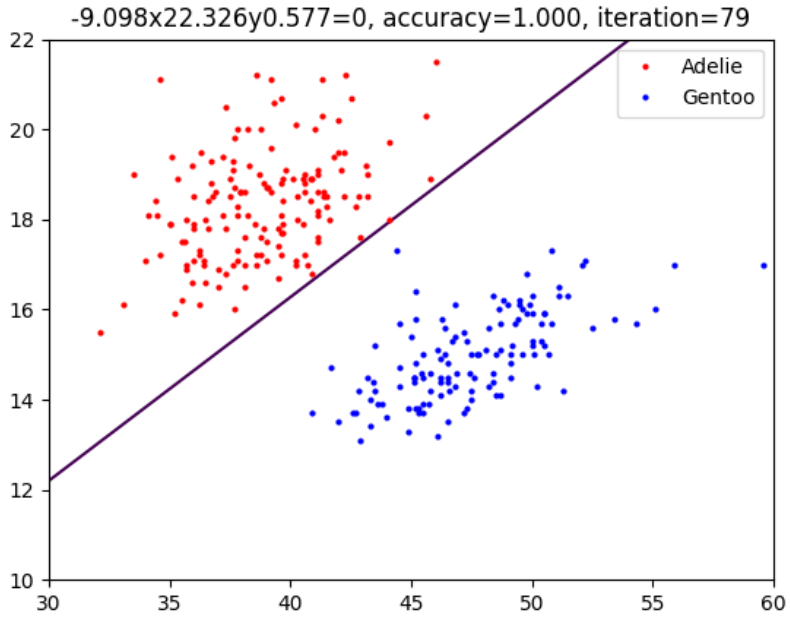
f) از همان کد قسمت قبل استفاده می‌کنیم. با این تفاوت که در فراخوانی تابع کلاس‌های هدف را تغییر می‌دهیم:



با توجه به نمودار، مدل با ۲۰۰۰ تکرار همچنان همگرا نشده است. دلیل این موضوع با توجه به اینکه داده‌ها کاملا خطی جداپذیر نیستند واضح است.بنابراین نیازی به افزایش تعداد تکرار نیست. اگرچه کارایی الگوریتم با توجه به مقدار دقت در این حالت مناسب است، اما تضمین نمی‌شود که مرز تصمیم فوق بهترین جواب باشد. برای یافتن بهترین جواب باید از روش pocket algorithm استفاده کرد.

g) همانطور که گفته شده مدل بین Gentoo و Adélie با ۷۸ تکرار همگرا شد. اما مدل بین Gentoo و Chinstrap همگرا نمی‌شود.

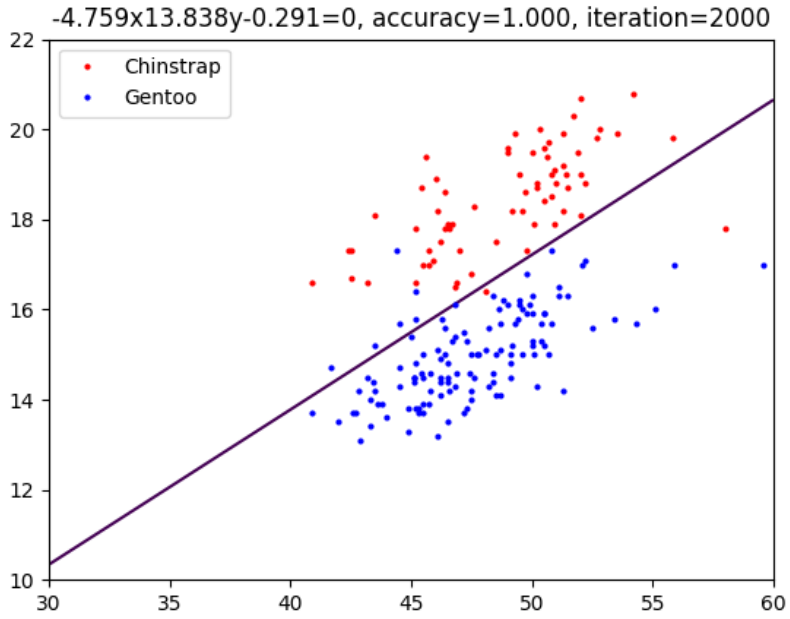
h) کد مربوط به این قسمت در فایل h.py قراردارد. معیار الگوریتم pocket برای بروزرسانی، تعداد نمونه های اشتباه دسته‌بندی‌شده در نظر گرفته شده‌است. یعنی در هربار اجرای حلقه اگر تعداد نمونه های اشتباه دسته‌بندی‌شده از کمترین مقدار، کمتر باشد؛ pocket بروز می‌شود. خروجی‌ها به این صورت است.



pocket updated 9 times and best answer was in 79th iteration

از آنجایی که برای این حالت الگوریتم همگرا می‌شود، مطابق انتظار بهترین مقدار pocket در تکرار آخر بوده است.

i) از همان کد مربوط به قسمت قبل استفاده می‌کنیم. با این تفاوت که در فراخوانی تابع، کلاس‌های هدف را تغییر می‌دهیم.



pocket updated 6 times and best answer was in 183th iteration

مقدار pocket ۶ بار بروزرسانی شده‌است و بهترین مقدار آن در تکرار ۱۸۳ بوده‌است. اگر معیار کارایی الگوریتم را دقت روی داده‌های آزمون در نظر بگیریم،با توجه به شکل این الگوریتم بهترین عملکرد را برای دو کلاس Chinstrap و Gentoo داشته است(در مقایسه با الگوریتم های قبلی برای همین دوکلاس)

۷)

---

a) یک روش برای این کار پیدا کردن عرض مناسب(h)، انتخاب آن به گونه ای است که انتگرال مجذور خطاها(MISE) را با توجه به یک حدس اولیه برای تابع توزیع مینیمم کند:

$$h_{opt} = argmin_h\ MISE\big(P_{KDE}(x)\big) = argmin_h\ \{E[\int (P_{KDE}(x) - P(x))^2\, dx]\}$$

در رابطه بالا P_{KDE} توزیع تخمینی و P توزیع واقعی می‌باشد. به طور خاص، اگر تابع P دارای توزیع نرمال باشد و از kernel گاووسی برای تخمین‌گر استفاده کنیم؛ می توان نشان داد عرض بهینه از رابطه $h_{opt} = 1.06\sigma N^{-1/5}$ بدست می آید که N تعداد نمونه‌های آموزشی است. یک روش دیگر ماکسیمم کردن pseudo-likelihood با روش leave-one-out-cross-validation است.

$$\begin{cases} h_{MLCV} = argmax_h \left\{ \frac{1}{n} \sum_n log P_{-n}(x^{(n)}) \right\} \\ where\ P_{-n}(x^{(n)}) = \frac{1}{(N-1)h} \sum_{m=1, n\neq m}^{N} K\left(\frac{x^{(n)} - x^{(m)}}{h} \right) \end{cases}$$

طبق روابط بالا هربار از یک نمونه به عنوان تست استفاده می‌کنیم و احتمال آن را به وسیله روش parzen window انتخاب می‌کنیم. h بهینه مقداری است که مجموع لگاریتم تمام تخمین ها برای نمونه های آموزش را ماکسیمم کند.(مشابه روش تخمین پارامتر می‌توان با مشتق گرفتن h بهینه را حساب کرد) .یک روش عمومی دیگر انتخاب عرض به گونه ای است که تعداد سلول ها برابر مجذور تعداد نمونه ها باشد.

b) بله. در روش 1NN اگر هر نمونه را به عنوان مرکز یک کلاس در نظر بگیریم، به دنبال کلاسی هستیم که مرکز آن با نمونه آزمون کمترین فاصله را دارد. یعنی به ازای هر داده آموزش می توان یک تابع جداکننده g تعریف کرد. در این حالت برچسب داده برابر با برچسب کلاس(نقطه)ای است که کمترین(بیشترین) g را دارد. به طور مشابه ، اگر در روش MDC، مرکز داده ها را به عنوان داده آزمون در نظر بگیریم و سایر نقاط را حذف کنیم، مسئله مانند الگوریتم 1NN می باشد. بدین معنی که می‌توانیم توابع جداکننده را تعریف نکنیم و مانند روش 1NN به صورت تنبل برچسب داده ها را با توجه به مرکز نقاط تعیین کنیم.

قسمت a تا d تمرین ۲ مثالی برای توضیحات فوق است.

c) خیر. خطی بودن مرزهای تصمیم به معیار فاصله در الگوریتم knn بستگی دارد. به عنوان مثال اگر معیار فاصله را به صورت $d(x,y) = \sqrt{(x_i x - x_i)^2 + (y_i y - y_i)^2}$ تعریف کنیم. به وضوح مرز تصمیم غیرخطی است.

d) با افزایش k واریانس مدل knn کاهش یافته و در نتیجه مرز تصمیم آن smooth تر می‌شود. از طرف دیگر با کاهش k، مرز تصمیم پیچیده تر می‌شود.

e) شرایط متفاوتی وجود دارد که این اتفاق رخ دهد. به عنوان یک مثال، وقتی نمونه‌های هر دو کلاس در یک راستا قرارداشته‌باشند و از فاصله کسینوسی به عنوان معیار فاصله استفاده شود؛ فاصله کسینوسی از هر دو کلاس یکسان خواهد بود. در این شرایط اگر انتخاب کلاس به صورت تصادفی نباشد ممکن است به ازای همه داده های تست، الگوریتم یک خروجی یکسان برگرداند.

f) یکی از اشکالات الگوریتم knn این است که فاز آموزش ندارد و اصطلاحاً تنبل است؛ درنتیجه فاز آزمون آن ممکن است کند باشد. برای رفع این مشکل می‌توان در فاز آموزش به وسیله دایاگرام ورونوی مرز تصمیم را ایجاد و ذخیره کرد. با این کار در فاز آزمون داده ها با توجه به مرز تصمیم برچسب گذاری می‌شوند و مشکل کند بودن آن حل می‌شود.

g) الگوریتم svm یک الگوریتم غیرپارامتریک است. یعنی با افزایش تعداد داده‌های آموزش، تعداد پارامترهای آن(بردارهای پشتیبان) افزایش می‌یابد ؛اما تعداد پارامترهای MLP با افزایش داده های آموزش ثابت است.الگوریتم MLP برای داده‌های آموزشی زیاد بهتر عمل می‌کند. از طرف دیگر الگوریتم svm برای مسائل غیرخطی نیاز به تعیین kernel دارد در حالی که الگوریتم MLP می‌تواند مستقیما بر روی مسائل غیرخطی استفاده شود. تفاوت دیگر این دو الگوریتم این است که svm ذاتاً برای مسائل دو کلاسه طراحی شده و برای استفاده از آن برای مسائل n کلاسه باید از n مدل svmمختلف استفاده شود. برتری دیگر MLP نسبت به svm، توانایی استفاده از آن در مسائل online learning است. در صورتی که svm(یا حداقل نوع kernel آن) این قابلیت را ندارد.

الگوریتم svm نسبت به MLP دارای برتری‌هایی نیز هست. الگوریم svm نسبت به MLP شانس کمتری برای گیر افتادن در مینیمم‌های محلی دارند. تعداد هایپر پارامتر ها در MLP بسیار زیاد است و ممکن است مدل دچار بیش‌برازش شود. الگوریتم svm برای مسائلی با داده‌های آموزش کم، بهتر عمل می‌کند و مرحله آموزش آن سریع‌تر است.

به طور خلاصه در مسائلی که تعداد داده ها کم است و نیاز به آموزش سریع‌تر است الگوریتم svm و در غیر اینصورت الگوریتم MLP ارجحیت دارد. همچنین در مسائل online learning استفاده از MLP ارجحیت دارد.

(h