به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

سیستم های هوشمند

تمرین شماره 2

نیما زمان پور

810198407

آذر ماه 1401

فهرست سوالات

سوال 1

الف)

ساختن درخت تصمیم را با انتخاب یک ویژگی از بین 4 ویژگی مورد برای نظر انتخاب می‌کنیم. و برای هر کدام information gain را حساب می‌کنیم.

$$Entropy = E(S) = -\sum p_i \log_2 p_i$$

$$IG(S,A) = Entropy(S) - \sum_{V \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

انتخاب بر اساس محل زندگی:

	$\boldsymbol{S = [+5,-5]}$ $\boldsymbol{E = 1}$	
	محل زندگی	
آب		خشکی
$\boldsymbol{S = [+3,-3]}$ $\boldsymbol{E = 1}$		$S = [+2,-2]$ $E = 1$
$\boldsymbol{IG(S, habitat) = 1 - 0.4 * 1 - 0.6 * 1 = 0}$		

انتخاب بر اساس قد:

	$\boldsymbol{S = [+5,-5]}$ $\boldsymbol{E = 1}$	
	قد	
بلند		کوتاه
$S = [+3,-3]$ $E = 1$		$S = [+2,-2]$ $E = 1$
$IG(S, habitat) = 1 - 0.4 * 1 - 0.6 * 1 = 0$		

انتخاب بر اساس تعداد پا:

	$\boldsymbol{S = [+5, -5]}$ $\boldsymbol{E = 1}$	
	تعداد پا	
3		2
$S = [0, -3]$ $E = 0$		$\boldsymbol{S = [+5, -2]}$ $\boldsymbol{E = 0.863}$
$\boldsymbol{IG(S, habitat) = 1 - 0.3 * 0 - 0.7 * 0.863 = 0.396}$		

انتخاب بر اساس رنگ

	$\boldsymbol{S = [+5, -5]}$ $\boldsymbol{E = 1}$	
	رنگ	
سبز		قهوه‌ای
$S = [1, -3]$ $E = 0.811$		$\boldsymbol{S = [+4, -2]}$ $\boldsymbol{E = 0.811}$
$\boldsymbol{IG(S, habitat) = 1 - 0.4 * 0.811 - 0.6 * 0.811 = 0.125}$		

از بین مقادیر information gain برای انتخاب ویژگی ریشه درخت. 0، 0، 0.396، 0.082 بیشترین را برمی‌گزینیم. که تعداد پا می‌باشد.

حال در این ریشه به سراغ شاخه سمت چپ می‌رویم. در این شاخه تمام نمونه ها B هستند. پس این node تبدیل به برگ می‌شود.

در شاخه سمت راست حالا به سراغ انتخاب بهترین attribute برای شاخه می‌رویم:

انتخاب بر اساس محل زندگی:

	S = [+5,-2] E = 0.863	

	محل زندگی	
آب		خشکی
S = [+3,-1] E = 0.811		S = [2,-1] E = 0.918
IG(S, habitat) = $0.863-\frac{3}{7}*\mathbf{0.918}-\frac{4}{7}*\mathbf{0.811}=\mathbf{0.006}$		

انتخاب بر اساس محل قد:

	S = [+5,-2] E = 0.863	
	قد	
بلند		کوتاه
S = [+3,-1] E = 0.811		S = [2,-1] E = 0.918
IG(S, habitat) = $0.863-\frac{3}{7}*\mathbf{0.918}-\frac{4}{7}*\mathbf{0.811}=\mathbf{0.006}$		

انتخاب بر اساس رنگ:

	S = [+5,-2] E = 0.863	
	رنگ	
قهوه‌ای		سبز
S = [+4,-1] E = 0.722		S = [1,-1] E = 1
IG(S, habitat) = $0.863-\frac{2}{7}*\mathbf{1}-\frac{5}{7}*\mathbf{0.722}=\mathbf{0.062}$		

مقدار information gain ویژگی رنگ از همه بیشتر شد. پس شاخه بعدی ویژگی رنگ است.

حالا در شاخه سمت چپ به سراغ attribute بعدی می‌رویم.

انتخاب بر اساس قد

	S = [1,-1] E = 1	
	فد	
بلند		کوتاه
S = [0,-1] E = 0		S = [+1,0] E = 0
IG(S, habitat) = 1-0.5*0-0.5*0 = 1		

انتخاب بر اساس محل زندگی

	S = [1,-1] E = 1	
	محل زندگی	
آب		خشکی
S = [+1,-1] E = 1		S = [0,0] E = 0
IG(S, habitat) = 1-1*1 = 0		

واضحا ویژگی قد به راحتی داده ها را تقسیم کرده و 2 برگ به ما می‌دهد.

حالا به سراغ شاخه سمت راست رنگ می‌رویم.

انتخاب بر اساس قد

	S = [4,-1] E = 0722	
	فد	
بلند		کوتاه
S = [+3,0] E = 0		S = [+1,-1] E = 1
IG(S, habitat) = 0.722-*0.6*0-0.4*1 = 0.322		

انتخاب بر اساس محل زندگی

	S = [4,-1] E = 0722	
	محل زندگی	
آب		خشکی
S = [+2,0] E = 0		S = [2,-1] E = 0.918
IG(S, habitat) = 0.722 – 0.6*0.918-0.4*0 = 0.015		

ویژگی قد را به عنوان شاخه بعدی انتخاب می‌نماییم. شاخه سمت راست تبدیل به برگ می‌شود. چون فقط یک نوع داده دارد. حالا آخرین attribute را نیز به عنوان شاخه برای شاخه سمت چپ ویژگی قد می‌گذاریم.

انتخاب بر اساس محل زندگی

	S = [2,-1] E = 0.918	
	محل زندگی	
آب		خشکی
S = [+1,0] E = 0		S = [0,-1] E = 1
IG(S, habitat) = 0.918-0.5*0-0.5*0 = 0.918		

دوشاخه آخر تبدیل به برگ می‌شود. و ساختن درخت کامل می‌شود.

		تعداد پا				
	3		2			
B			رنگ			
		سبز		قهوه‌ای		
	قد				قد	
کوتاه		بلند		کوتاه		بلند
A		B		محل زندگی		A
			خشکی		آب	
		B				A

ب)

Table 1:داده های آزمون و نتایج مدل

شماره	رنگ	تعداد پا	قد	محل زندگی	جاندار	خروجی مدل
1	قهوه‌ای	3	بلند	خشکی	B	B
2	سبز	2	بلند	خشکی	A	B
3	سبز	2	کوتاه	خشکی	A	A
4	قهوه‌ای	2	کوتاه	آب	B	A
5	قهوه‌ای	2	بلند	خشکی	A	A

دقت مدل ٪60 شد.

Table 2: ماتریس آشفتگی[1] مدل

Label / Prediction	A	B
A	2	1
B	1	1

به دلیل کم بودن داده های آموزش، مدل عملکرد خوبی ندارد. و دو جاندار جای هم پیش‌بینی می‌شوند.

مدل توانست دقت داده های آموزش را صفر کند ولی دقت داده های آزمون پایین است. یعنی واریانس آن بالا است و مدل overfit شده است.

ج)

بطور کلی الگوریتم ID3 بصورت حریصانه[2] عمل کرده. و هر بار بهترین attribute را انتخاب می‌کند. تا به برگ های درخت برسد. روش Reduced-Error Pruning روشی برای هرس کردن درخت است. تا اندازه آن کوچک شود. این روش در ابتدا دقت مجموعه train را کاهش داده ولی دقت مجموعه test را افزایش می‌دهد. چون می‌خواهیم دقت مجموعه آموزش همچنان صفر بماند. حذف هر کدام از برگ ها باعث ضرر می‌شود. و نمی‌توان فقط با استفاده از 2 ویژگی درخت را ساخت. راه دیگر این است که $nCr\binom{4}{2}$ ویژگی را امتحان کرده و ببینیم که هیچ کدام دقت مجموعه آموزش را صفر نمی‌کند.

د)

درخت های تصمیم به دلیل آن که در مراحل ساخت تا جایی پیش می‌روند که یا ویژگی باقی نمانده باشد یا برگ ها همگی از یک نوع باشند. مقاومت به نویز کمی دارند. و وجود نویز باعث می‌شود که درخت بزرگتر شده تا اثر آن را خنثی کند. که این مسئله باعث بزرگ شدن درخت و overfitting می‌شود. دلیل دیگر نیز کمبود داده آزمون است که باعث خطای صفر آموزش و خطای بالای آزمون می‌شود.

برای حل این مشکل دو راهکار وجود دارد:

---

[1] Confusion matrix
[2] greedy

1. از ابتدا نگذاریم درخت به حد زیادی بزرگ شود. حدی برای حداکثر عمق درخت درنظر بگیریم.(max depth) یا اگر تعداد نمونه های باقی‌مانده در یک شاخه از حدی کمتر شد نیز درخت را متوقف کنیم. و دیگر تقسیم بندی انجام ندهیم(min sample split)
2. ابتدا بگذاریم درخت بطور کامل رشد کند. بعد از پایین شروع به "هرس" کردن درخت می‌کنیم. یک روش معمول Reduced-Error Pruning است. در این روش هر کدام از شاخه ها را به همراه زیرشاخه های آن حذف می‌کنیم. و جای آن پرتکرار ترین Label را می‌گذاریم. اگر حذف آن باعث بهبود دقت مجموعه آزمون شد آن را تبدیل به برگ می‌کنیم. در غیر این صورت آن را نگه می‌داریم این کار را تا جایی انجام می‌دهیم که هرگونه تغییر برای مدل مضر باشد.

سوال 2

الف)

ویژگی های موثر فرد سن، کلاس بلیط (که موقعیت فرد بر کشتی را مشخص می‌کند)، تعداد همراه(شامل فرزند، والدین، همسر) و جنسیت می‌باشد.

دیتاست شامل 891 مسافر کشتی تایتانیک است. ازین تعداد 700 مسافر را به عنوان داده آموزش و 191 نفر را داده آزمون انتخاب می‌کنیم. برای دادها ستون Survived را جدا می‌کنیم. بعد برای داده آزمون ستون Cabin را که شماره کابین افراد است. و اطلاعات خاصی به ما نمیدهد و همچنین در داده آموزش وجود ندارد را حذف می‌کنیم. در dataset تعدادی missing value وجود دارد. برای ستون های Age, Fare از میانگین بقیه ستون ها برای پر کردن نمونه‌های خالی استفاده می‌کنیم. و این دو متغیر پیوسته را از مقدار اطراف میانگین خود سه تکه کرده و گسسته می‌کنیم (افراد جوان میانسال و پیر براس ستون Age و بلیط های ارزان و متوسط و گران برای ستون Fare). برای بقیه ستون ها به دلیل گسسته بودن مقادیر، از مد داده‌های ستون استفاده می‌کنیم.

برای درخت تصمیم گیری، از معیار انتروپی استفاده می‌کنیم.

در depth = 3 دقت درخت برابر ٪81.68 و ماتریس آشفتگی بصورت زیر است:

Table 3: ماتریس آشفتگی درخت در عمق 3

Label / Prediction	1	0
1	114	6
0	29	42

در depth = 4 دقت درخت برابر ٪83.77 و ماتریس آشفتگی بصورت زیر است:

Table 4: ماتریس آشفتگی درخت در عمق 4

Label / Prediction	1	0
1	110	10
0	21	50

در depth = 5 دقت درخت برابر ٪81.68 و ماتریس آشفتگی بصورت زیر است:

Table 5: ماتریس آشفتگی درخت در عمق 5

Label / Prediction	1	0
1	108	12
0	23	48

در depth = 6 دقت درخت برابر ٪82.20 و ماتریس آشفتگی بصورت زیر است:

Table 6 : ماتریس آشفتگی در عمق 6

Label / Prediction	1	0
1	109	11
0	23	48

در depth = 7 دقت درخت برابر ٪91.62 و ماتریس آشفتگی بصورت زیر است:

Table 7: ماتریس آشفتگی در عمق 7

Label / Prediction	1	0
1	112	8
0	8	63

با افزایش عمق طبقه بند از عمق 3 تا 6 دقت مدل تقریبا یکسان و حدود ٪82 است. به این معنا که ویژگی های اضافه شده به درخت تاثیر زیادی بر دقت آن ندارند. اما در عمق 7 که حداکثر عمق ممکن است. دقت به ٪92 می‌رسد.

ویژگی ها به ترتیب از پایین به بالا Sex, Pclass, Fare, SibSp, Parch, Age, Embarked می‌باشند.

ب)

مهم‌ترین ایراد درخت تصمیم مستعد بودن به overfitting است. که باعث می‌شود. واریانس مدل زیاد شده. و نسبت به نویز حساس باشد.(برای داده های نویزی برای این که دقت مجموعه آموزش همچنان کم

باقی بماند، درخت شاخه های ضائد اضافه می‌کند که عمق آن را زیاده از حد می‌کند) ایراد دیگر آن سخت بودن اسفتاده در متغیر های پیوسته است. که زیاد به گسسته سازی دارد.

حالا برای حل مشکل اول باید به گونه ای از overfitting جلوگیری کنیم. و واریانس را کاهش دهیم. به جای یک درخت کامل که نویز بالایی دارد. از چند درخت باهم استفاده می‌کنیم. و با رای اکثریت[1] میانگین آن ها را می‌گیریم. از دو روش bootstrap aggregation و feature selection استفاده می‌کنیم. یعنی در دیتاست قسمتی از داده ها را تکثیر و جایگزین همان تعداد داده بصورت رندم می‌کنیم. این کار bootstrap aggregation نام دارد. سپس بر روی این داده ها درخت های خود را آموزش می‌دهیم. با این تفاوت که هر بار تعدادی از ویژگی ها را حذف می‌کنیم.(مثلا p ویژگی از کل m ویژگی) و سپس آموزش می‌دهیم. با این روش ها تعدادی درخت به وجود می‌آید. که پیش‌بینی نتایج را از طریق رای اکثریت بین جواب درخت ها انجام می‌دهیم.(در روش بالا اگر m = p بود به آن روش bagging می‌گویند. در غیر اینصورت جنگل تصادفی است.)

ج)

برای p = 6 و تعداد درخت 5، دقت جنگل تصادفی برابر 84.82٪ و ماتریس آشفتگی آن بصورت زیر است:

Table 8 : ماتریس آشفتگی جنگل تصادفی

Label / Prediction	1	0
1	106	14
0	15	56

[1] Majority voting

سوال 3

طبقه بند k همسایه نزدیک

الف)

با پیاده سازی الگوریتم KNN نتایج زیر برای دیتاست آزمون بدست آمد

K = 1:

Accuracy = 83.33%

Table 9 : ماتریس آشفتگی برای **K = 1**

Label / Prediction	**0**	**1**	**2**
0	10	0	0
1	1	12	**1**
2	0	4	8

K = 5

Accuracy = 77.78%

Table 10 : ماتریس آشفتگی برای **K = 5**

Label / Prediction	**0**	**1**	**2**
0	10	0	0
1	1	11	2
2	2	3	7

K = 10

Accuracy = 72.22%

Table 11 : ماتریس آشفتگی برای K = 10

Label / Prediction	0	1	2
0	10	0	0
1	1	11	2
2	2	3	7

K = 20

Accuracy = 72.22%

Table 12 : ماتریس آشفتگی برای K = 20

Label / Prediction	0	1	2
0	10	0	0
1	1	10	3
2	1	5	6

بهترین عملکرد مدل در K = 1 بدست آمد.

ب)

با محاسبه احتمال کلاس پیش‌بینی شده و رسم هیستوگرام آن نمودار های زیر برای K های مختلف بدست می‌آید:

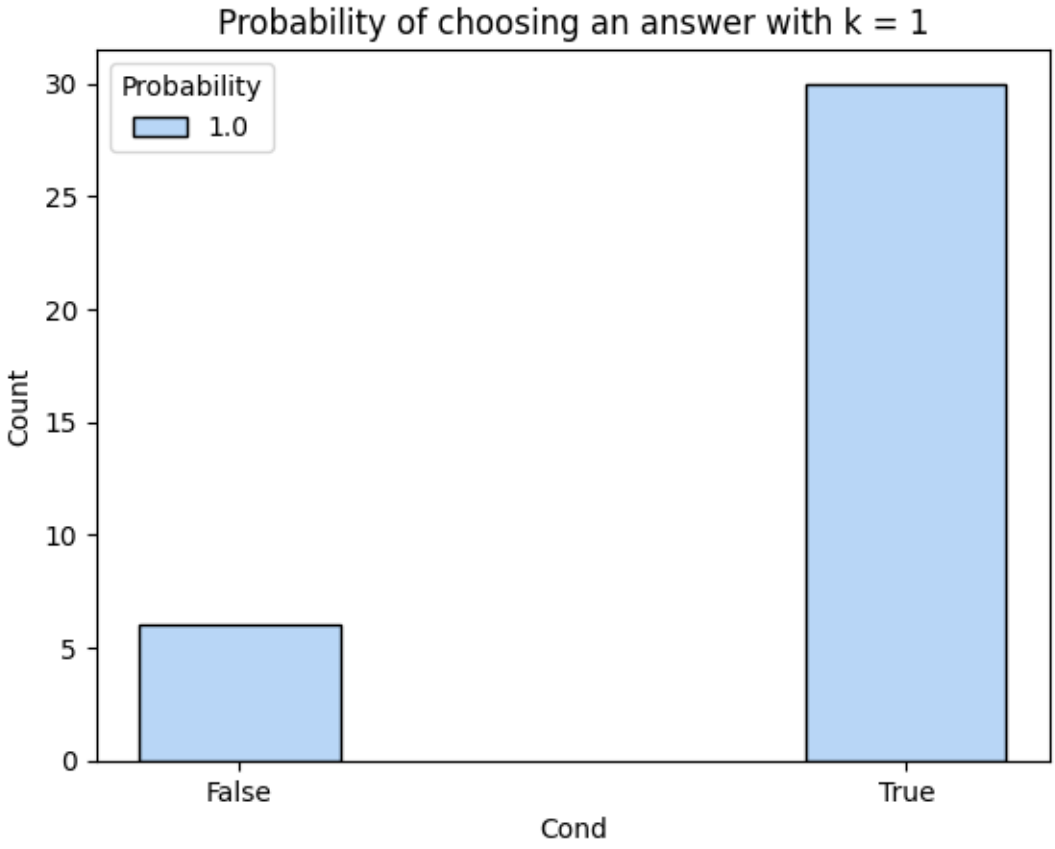

Figure 1 : نمودار توزیع احتمال متعلق به هر کلاس برای **K = 1**

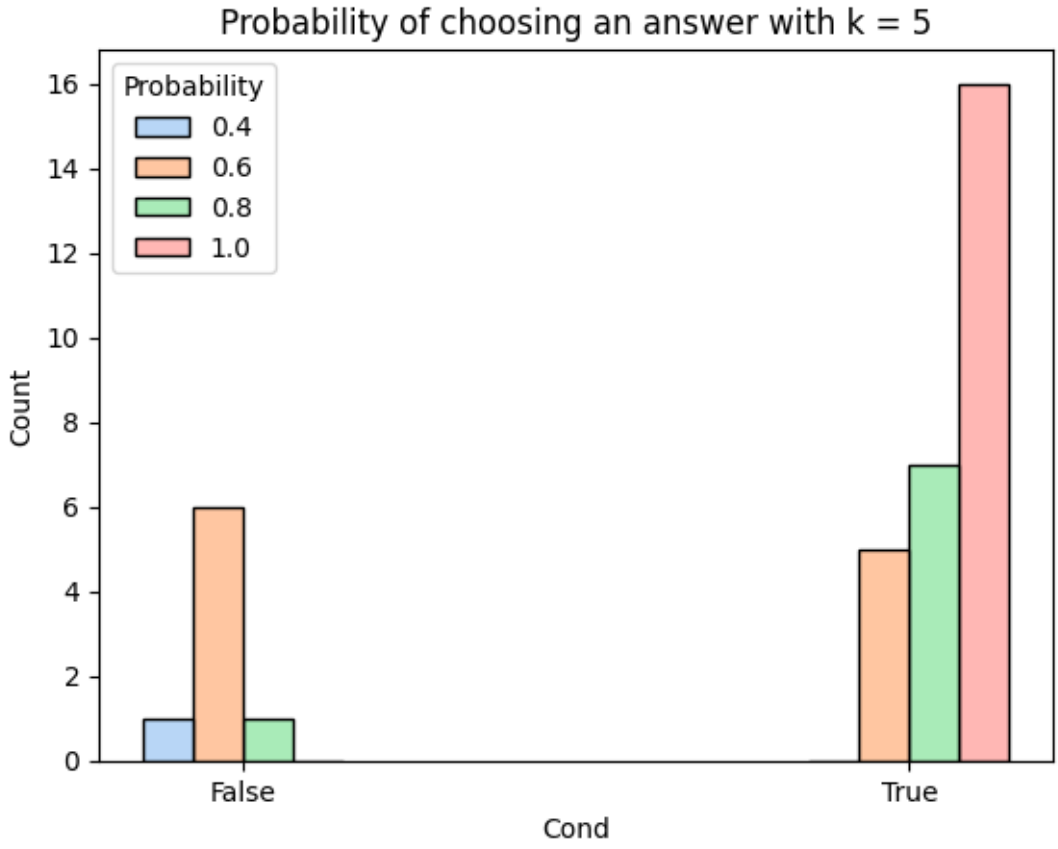

Figure 2 : نمودار توزیع احتمال متعلق به هر کلاس برای **K = 5**

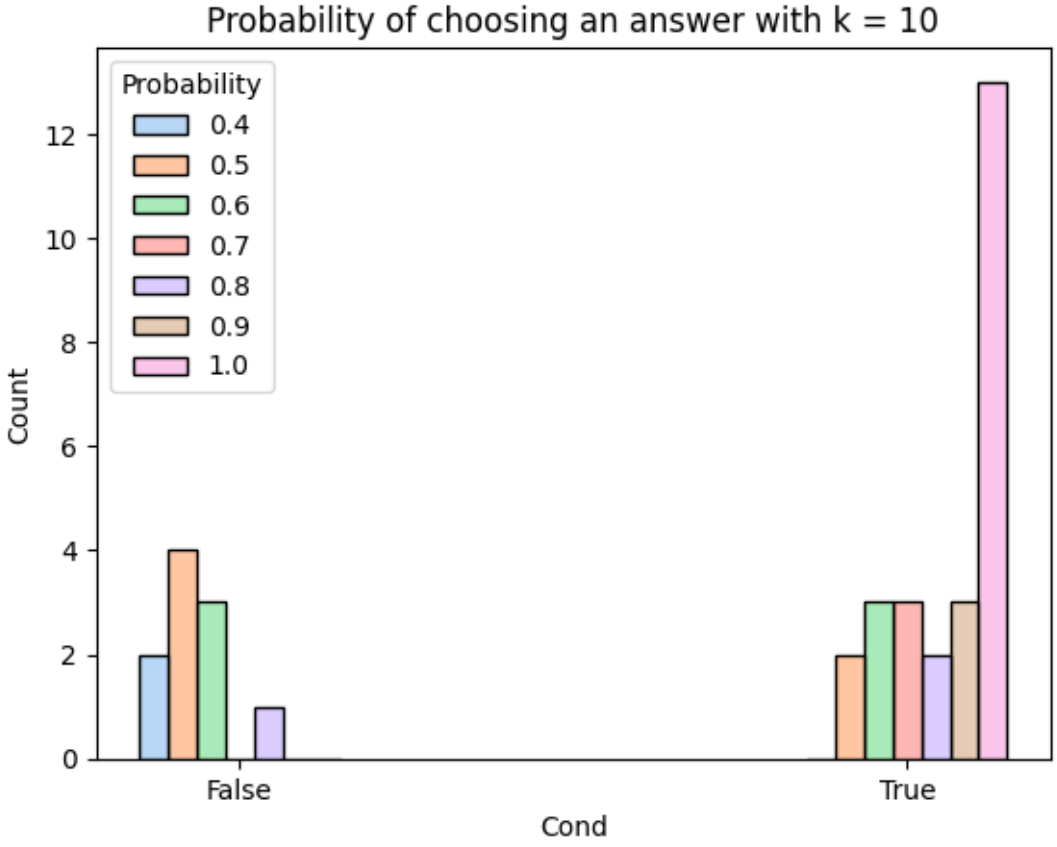

Figure 3 : نمودار توزیع احتمال متعلق به هر کلاس برای **K = 10**

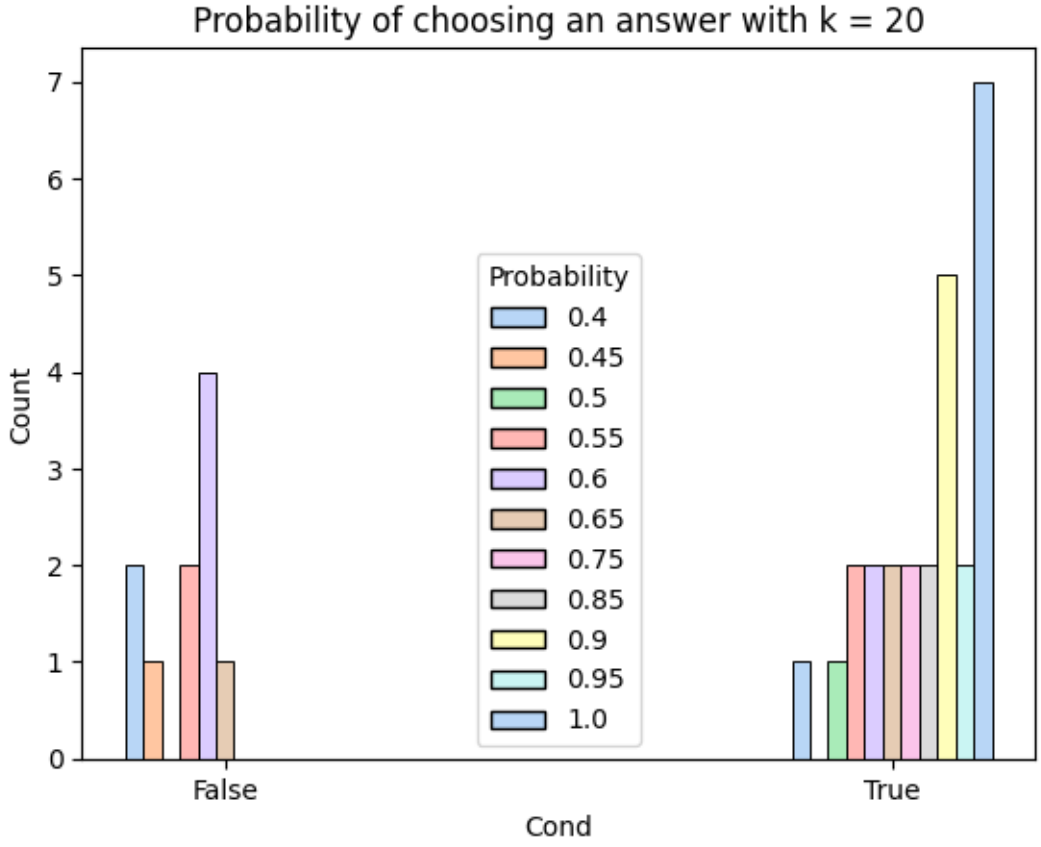

Figure 4 : نمودار توزیع احتمال متعلق به هر کلاس برای **K = 20**

با اتکا به درصد درستی به تنهایی، در K = 1 مدل بهتر عمل کرده است. ولی با توجه به نمودار ها می‌توان دید که در K = 10 احتمال درست بودن $p = 1$ به شکل چشم گیری از بقیه احتمال ها درست تر است. به این معنا که قطعیت آن بالا تر است. در حالی که در بقیه K ها احتمال $p = 1$ اختلاف چشم گیری ندارد.

یادگیری بر اساس معیار

الف)

بطور کلی اهداف یادگیری بر اساس معیار های LMNN, LFDA این است. که داده ها با یک تبدیل خطی به فضایی بروند که در آن فضا داده های مشابه به هم نزدیک تر و داده های غیر مشابه از هم دورتر شده باشند. در این معیار ها ما از فاصله ماهالانوبیس[1] به جای فاصله اقلیدسی استفاده می‌کنیم.

$$d(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} - \overrightarrow{x_j})^T M(\overrightarrow{x_i} - \overrightarrow{x_j})$$

[1] Mahalanobis distance

در معادله بالا M همان ماتریس تبدیل خطی ما است. که نیمه معین مثبت می‌باشد.

حالا ماتریس M را طوری تعریف می‌کنیم که فاصله بین داده های مشابه را مینیمم کند.

$$\min_{M} \sum_{i,j \in N_i} d(\overrightarrow{x_i}, \overrightarrow{x_j})$$

و از طرف فاصله دیگر نمونه‌های غیر مشابه را که کمتر از یک واحد با نمونه های مشابه، با داده فاصله دارند را نیز حداکثر کند. یعنی یک margin بین داده های همشابه و همسایه با داده های غیر مشابه و همسایه ایجاد کند.(قسمت $[.]_+ = \max(.,0)$ برای این است که داده‌های مشابه که همینطوری از داده های غیر مشابه به نمونه نزدیک تر هستند. دیگر در بهینه سازی نقش نداشته باشند).

$$\min_{M} \sum_{i,j \in N_i, l, y_l \neq y_i} \left[d(\overrightarrow{x_i}, \overrightarrow{x_j}) + 1 - d(\overrightarrow{x_i}, \overrightarrow{x_l})\right]_+$$

معادله نهایی بصورت زیر است.

$$\min_{M} \sum_{i,j \in N_i} d(\overrightarrow{x_i}, \overrightarrow{x_j}) + \lambda \sum_{i,j,l} \xi_{ijl}$$

$$\forall_{i,j \in N_i, l, y_l \neq y_i}$$

$$d(\overrightarrow{x_i}, \overrightarrow{x_j}) + 1 - d(\overrightarrow{x_i}, \overrightarrow{x_l}) \leq \xi_{ijl}$$

$$\xi_{ijl} \geq 0$$

$$M: semi\ definite\ positive$$

در واقع ξ_{ijl} همان نقش تابع max را بازی می‌کند. و برای این که زیر رادیکال منفی نشود نیز ماتریس M باید نیمه معین مثبت باشد.

در روش LFDA ابتدا تعریف می‌کنیم.

$$S^{(w)} = \frac{1}{2} \sum_{i,j} W_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^T$$

$$S^{(b)} = \frac{1}{2} \sum_{i,j} W_{i,j}^{(b)} (x_i - x_j)(x_i - x_j)^T$$

که $S^{(w)}$ ماتریس خوشه درون-کلاسی[1] است. و $S^{(b)}$ ماتریس خوشه میان-کلاسی[2] است. داریم:

$$A_{i,j} = affininy\ of\ (x_i, x_j)$$

[1] within-class scatter
[2] between-class scatter matrix

$$W_{i,j}^{(w)} = \begin{cases} \frac{A_{i,j}}{n_l} & if\ y_i = y_j = l \\ 0 & y_i \neq y_j \end{cases}$$

$$W_{i,j}^{(b)} = \begin{cases} A_{i,j}(\frac{1}{n} - \frac{1}{n_l}) & if\ y_i = y_j = l \\ \frac{1}{n} & y_i \neq y_j \end{cases}$$

$$T_{LFDA} = argmax_{T \epsilon R}[tr(T^T S^{(w)} t)^{-1} T^T S^{(b)} T]$$

ب)

1. در طبقه بند k همسایه، پارامتر k تعداد همسایه نزدیکی را نشان می‌دهد. که با استفاده از آن ها لیبل نمونه ها مشخص می‌شود. ولی در روش های یادگیری بر اساس معیار پارامتر K در واقع تعداد همسایه با لیبل مشابه است که قرار است نزدیک نگه داشته شوند. و بقیه همسایه ها با ایجاد یک margin دور شوند.
2. روش اول را انتخاب نموده و به کمک کتابخانه mtric-learning بعد داده ها را کاهش دادیم.
3. به کمک دستور های fit, transform ابعاد داده‌ها کاهش یافت و به فضای 2 بعدی رفتند. علاوه بر آن تبدیل ماتریسی تغییر پایه نیز بر آن ها اعمال شد. برای K های مختلف نمودار های زیر بدست آمد:

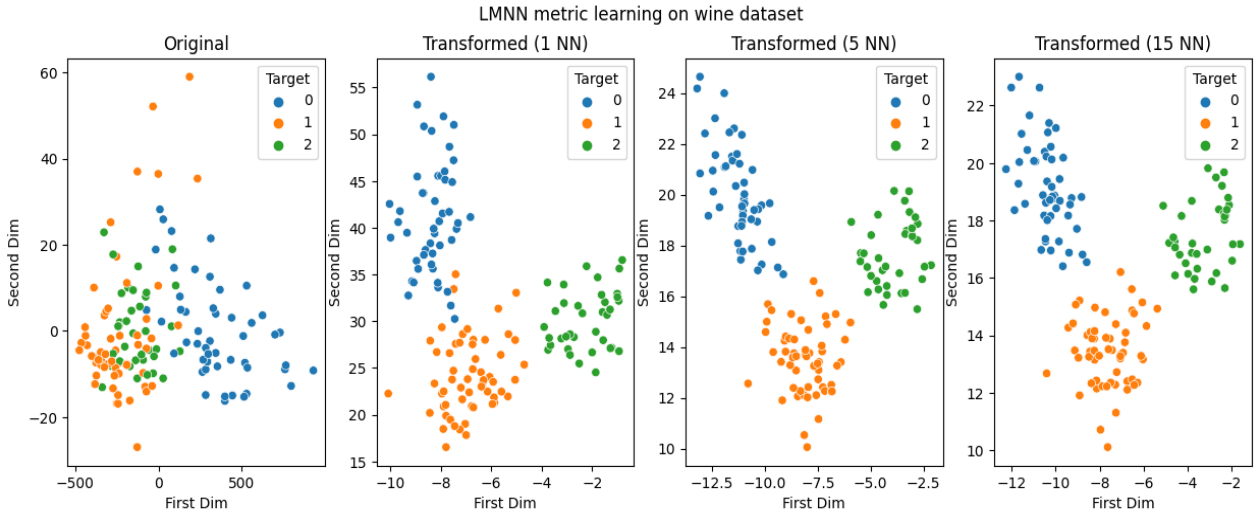

Figure 5 افراز دادگان اصلی و انتقال یافته به ازای مقادیر مختلف **K** با روش **LMNN**

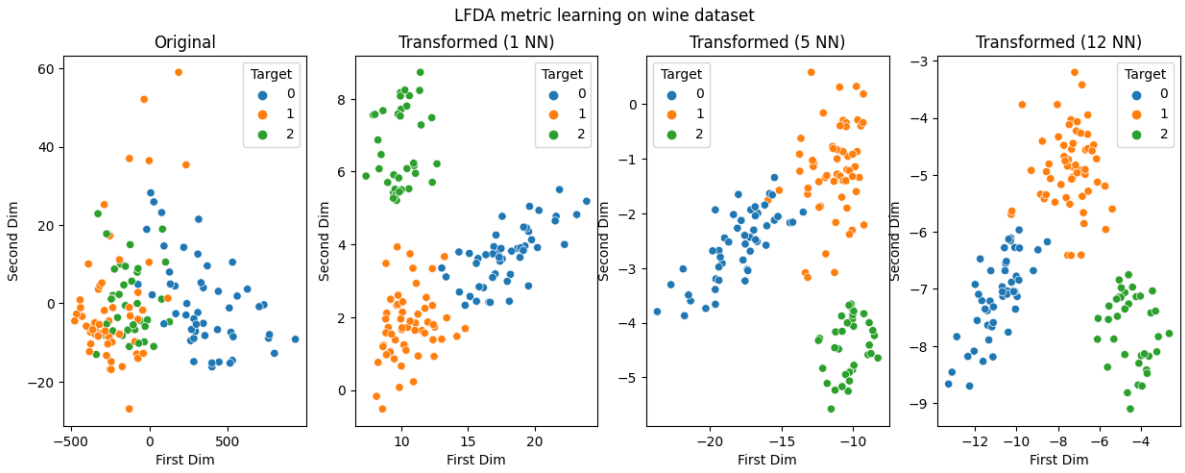

Figure 6 افراز دادگان اصلی و انتقال یافته به ازای مقادیر مختلف **K** با روش **LFDA**

4. هر چقدر مقدار K بیشتر شود تفکیک پذیری بیشتری دارد. دلیل آن این است. که مقدار k همانطور که در بخش الف گفته شد. K عدد از همسایه های مشابه نزدیک را در نظر می‌گیرد. و ماتریسی میسازد که داده های مشابه همسایه را نزدیک تر و داده های غیر مشابه را با یک margin دور می‌سازد. طبیعتا هر چه مقدار k بیشتر باشد. همسایه های بیشتری به هم نزدیک می‌شوند و غیر همسان ها دور تر و قابلیت تفکیک پذیری بالاتر می‌رود.

ج)

در بخش قبل مقدار K = 1 بهترین عملکرد را داشت. با دادگان انتقال یافته در فضای جدید برای هر دو روش LMNN, LFDA داریم:

Accuracy = 94.44 %

Table 13 ماتریس آشفتگی مدل در فضای انتقال یافته

Label / Prediction	0	1	2
0	12	0	0
1	1	16	1
2	0	0	6

طبقه بند در فضای انتقال یافته جدید بسیار بهتر عمل کرده که دلیل آن تفکیک پذیری بهتر با استفاده از یادگیری معیار است.

د)

با رسم ماتریس همبستگی دیتاست برای حالت بدون انتقال می‌بینیم که تعدادی از داده ها با هم همبستگی مثبت و تعدادی همبستگی مثبت دارند. با اعمال روش LMNN می‌بینیم که همبستگی داده های که مثبت بوده بیشتر شده و همبستگی داده هایی که منفی بوده اند بیشتر شده است. اما در ماتریس همبستگی روش LFDA می‌بینیم که تقریبا همبستگی بین داده ها صفر شده است. یعنی ستون های ویژگی از هم مستقل شده اند. که باعث می‌شود. تفکیک پذیری داده ها بیشتر شود.

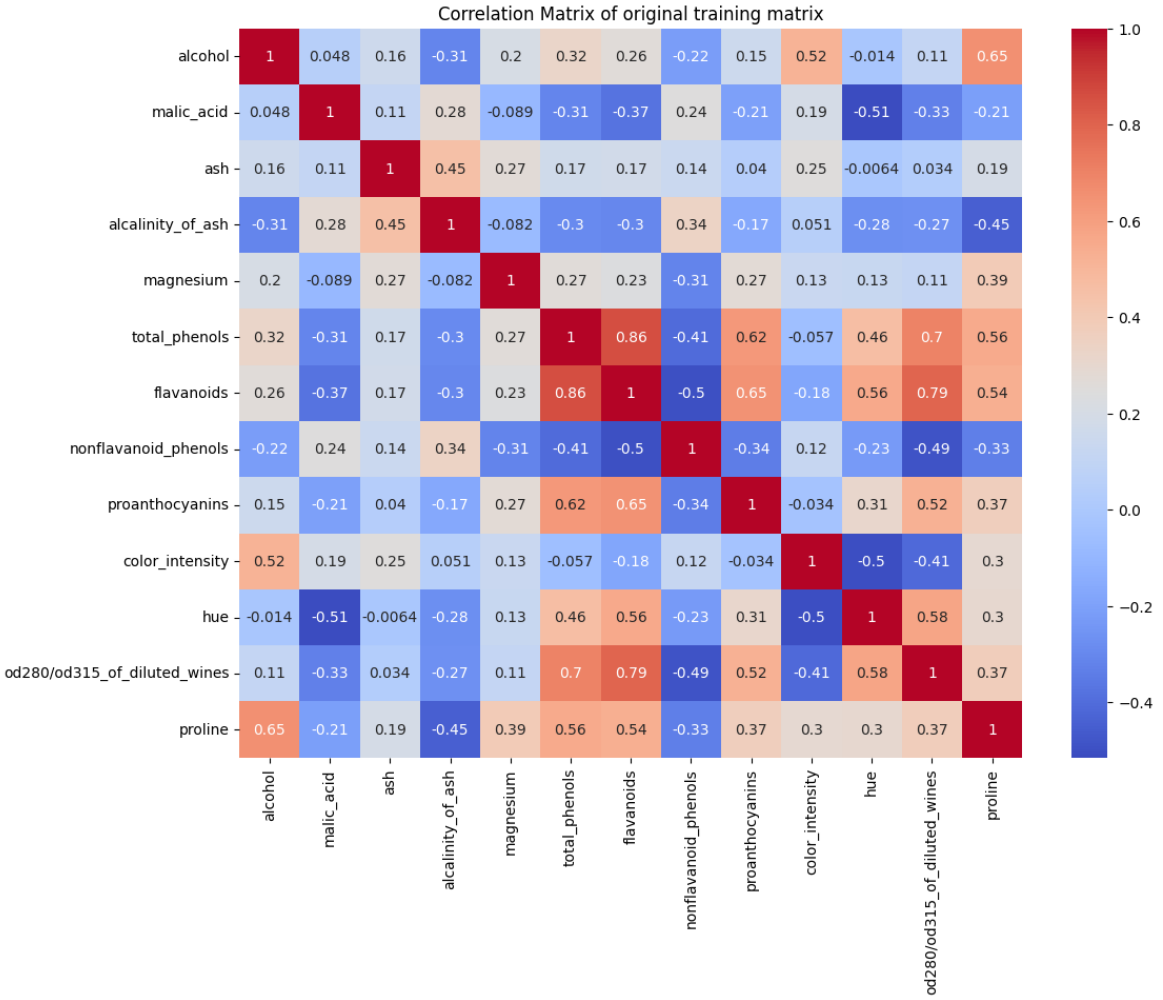

Figure 7 ماتریس همبستگی دیتاست اصلی

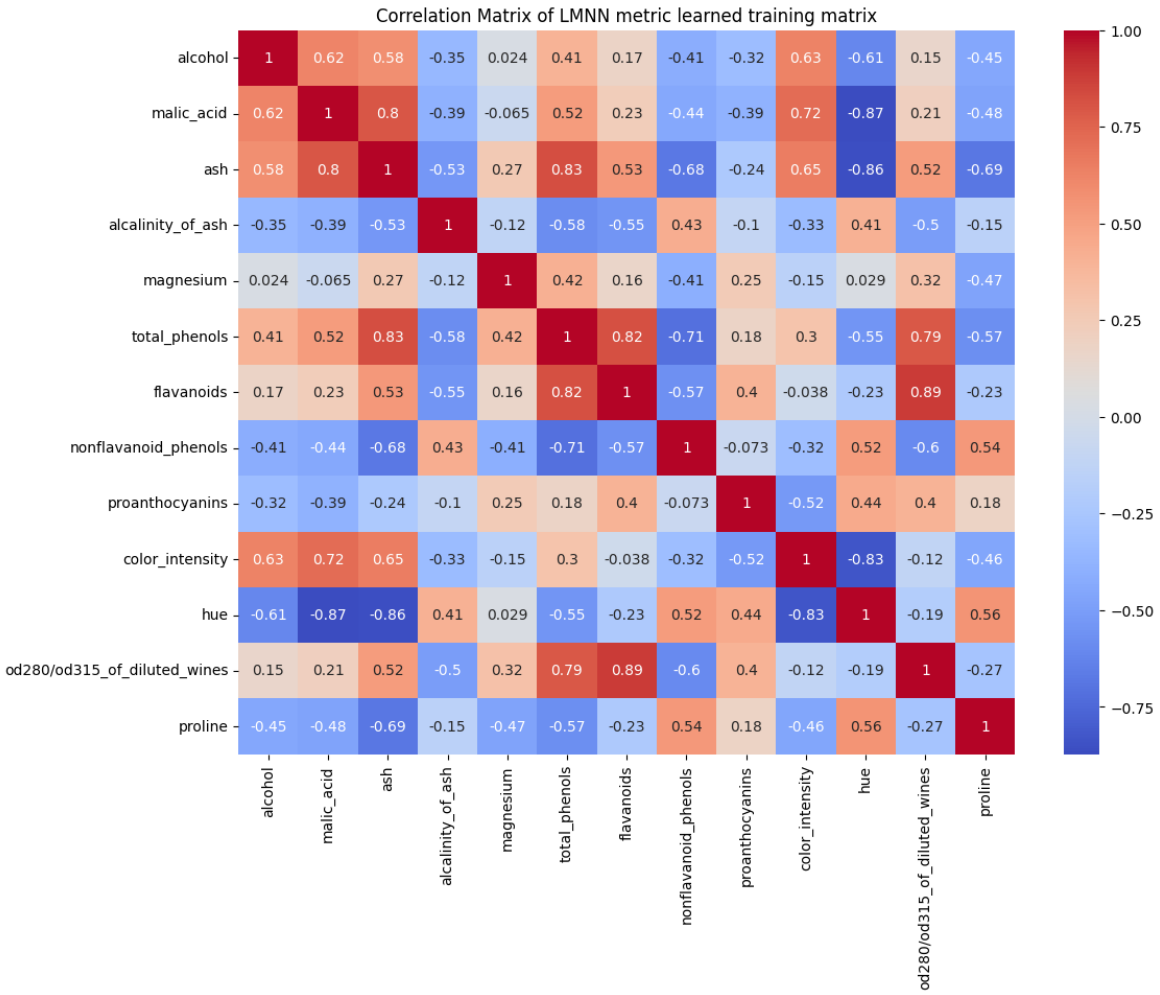

Figure 8 ماتریس همبستگی دیتاست انتقال یافته به روش **LMNN**

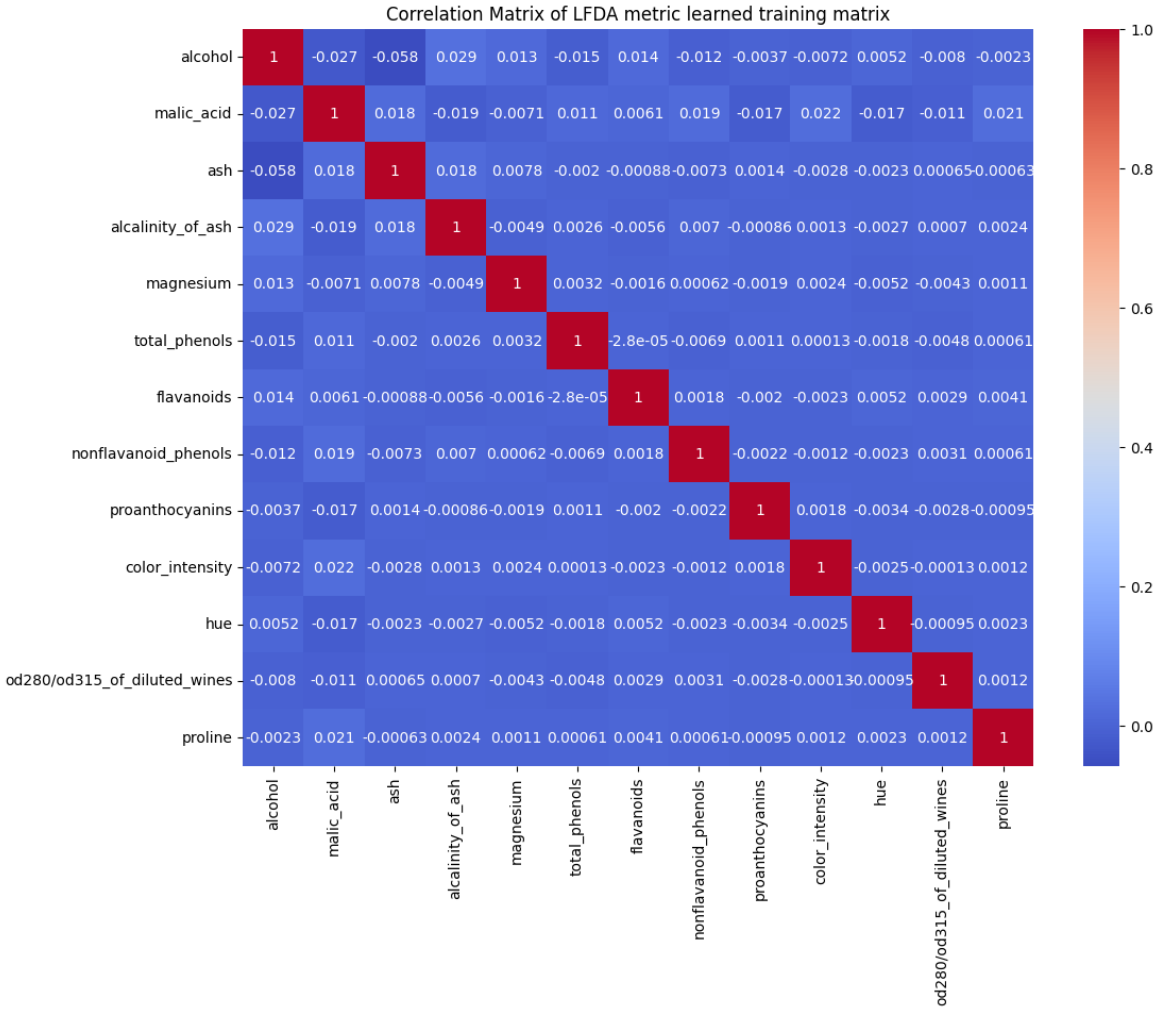

Figure 9 ماتریس همبستگی دیتاست انتقال یافته به روش **LFDA**

۵)

در روش GMLL پایه روش همان دور کردن نقاط غیر مشابه و نزدیک کردن نقاط مشابه به هم است. همانند روش هایی مثل MMC این روش نیز یک بخش دارد که فاصله داده‌های نزدیک را کم می‌کند.

$$\min_{M} \sum_{(x_i,x_j)\in S} d_A(\overrightarrow{x_i},\overrightarrow{x_j})$$

و بخش دیگر که فاصله داده های غیر مشابه را کم می‌کند. با این تفاوت که از خاصیت ماتریس های مثبت معین استفاده می‌کند که $A \succ b \Longrightarrow B^{-1} \succ A^{-1}$ یعنی همزمان هر چقدر که ماتریس A فاصله Mahalanobis را کم می‌کند. ماتریس A^{-1} فاصله Mahalanobis را زیاد می‌کند. که این دقیقا خواسته ما از مسئله است.

$$\min_{M} \sum_{(x_i,x_j)\in S} = d_A(\overrightarrow{x_i},\overrightarrow{x_j}) + \sum_{(x_i,x_j)\in D} d_{A^{-1}}(\overrightarrow{x_i},\overrightarrow{x_j})$$

$$= \sum_{(x_i,x_j)\in S} tr(A(x_i-x_j)(x_i-x_j)^T) + \sum_{(x_i,x_j)\in D} tr(A^{-1}(x_i-x_j)(x_i-x_j)^T)$$

فرق روش GMML و LMNN این است که روش LMNN به شکل غیر متقارن با داده ها رفتار می‌کند. و هر کدام را جداگانه از تابع max عبور می‌دهد. ولی روش GMML فاصله Mahalanobis را بصورت یک حاصل جمع بر روی کل تابع هزینه اعمال می‌کند.

پیوست

سوال 2) توضیح کد

در قسمت اول کد درخت تصمیم ابتدا داده‌ها را وارد کرده. و بر اساس توضیحات سوال 2 پیش پردازش را انجام می‌دهیم. توابع entropy, information_gain, best_attribute را تعریف می‌کنیم. این توابع برای انتخاب بهترین ویژگی برای تقسیم درخت هستند. تابع best_attribute برای هر ستون دیتاست ورودی بهره اطلاعات را به کمک معیار آنتروپی حساب می‌کند. و بهترین ویژگی را برمی‌گرداند. جهت تعریف بهتر درخت، یک کلاس Node تعریف می‌کنیم. که شامل attribute های پدر، فرزند و زیر درخت و ... است.

حالا به سراغ تعریف تابع درخت تصمیم گیری می‌رویم. این درخت بصورت بازگشتی پیاده سازی شده. در ابتدا یک نود ریشه می‌سازیم این نود دارای بهترین ویژگی گرفته شده از تابع best_attribute است. برای هر مقدار از این ویژگی به کمک for یک نود فرزند ساخته و قسمتی از دیتاست که از پدر آمده و دارای این ویژگی است به عنوان subtree نود فرزند قرار داده می‌شود. حالا برای قسمت بازگشتی به هر نود فرزند همانند یک ریشه نگاه می‌کنیم. و دوباره تابع decision_tree را روی آن فرا می‌خوانیم. این بازگشت تا جایی ادامه پیدا می‌کند. تا عمق درخت به حداکثر عمق تعیین شده برسد. در این صورت Node های آخر تبدیل به برگ می‌شوند. و مقدار آن ها برابر بیشترین مقدار تکرار شده ستون هدف می‌شود.

بعد از ساخت درخت به سراغ تست آن میرویم. تابع predict مسافر و درخت ساخته شده را میگیرد و نتیجه را برمی‌گرداند. طرز کار آن بازگشتی است. به این صورت که از ریشه شروع کرده و هر بار مقدار value فرزند های آن را نگاه می‌کند. هر کدام که مقدار value در آن ویژگی ستون خودش برابر بود. از همان نود فرزند دوباره تابع را فرا می‌خواند تا جایی که به برگ درخت برسد و مقدار پیشبینی را برگرداند.

تابع score نیز برای همه نمونه های دیتاست آزمون نتیجه را برمی‌گردند. در یک لیست ریخته؛ تبدیل به دیتافریم می‌کند. و دقت تابع را همراه با پیش بینی ها برمی‌گرداند. در تابع confusion_matrix() نیز

پیش بینی ها را گرفته و به کمک دستور crosstab در کتابخانه pandas به راحتی ماتریس آشفتگی آن را برمی‌گرداند.

برای قسمت دوم سوال که پیاده سازی جنگل تصادفی است. آرگومان های حداکثر عمق، تعداد ویژگی ها و تعداد درختان را می‌گیرد. در یک حلقه for به تعداد درختان ورودی ابتدا به شکل رندوم تعدادی ویژگی را انتخاب می‌کنیم. و بقیه ویژگی ها را از دیتاست دور انداخته. سپس bootstrap aggregation انجام می‌دهیم. و دیتاست آماده شده را به تابع ()decision_tree می‌دهیم. تابع همه درخت ها را در یک لیست به اسم forest ریخته و برمی‌گرداند.

حالا برای درخت نیز باید تابع ()forest_predict بسازیم. میتوان به راحتی از تابع ()predict درخت تصادفی استفاده کرد. و پیش بینی ها را در یک لیست ریخت و مد آن را برگرداند.

در آخر نیز از مدل خود run می‌گیریم.

سوال 3) توضیح کد

ابتدا کتابخانه های مورد نظر را import می‌کنیم. داده ها به ترتیب هستند و نیاز به بر خوردن دارند. برای این کار ابتدا دیتاست را به دو قسمت x, y تقسیم می‌کنیم. سپس x, y را زیپ کرده و در یک لیست آن را shuffle می‌کنیم. و دوباره unzip می‌کنیم. دیتاست را به نسبت 1 به 4 تقسیم می‌کنیم.

تابع KNN را تعریف می‌کنیم. برای هر نقطه در دیتاست آزمون، فاصله از هر نقطه در دیتاست آموزش را به کمک تابع ()np.linalg.norm حساب کرده و در یک لیست می‌ریزیم. لیست را مرتب کرده و از بین K داده اول، لیبل مد آن را به همراه لیبل همه k نقطه برمی‌گردانیم.

برای مشخص شدن دقت مدل تابع ()score را تعریف می‌کنیم. نتایج و لیبل دیتاست تست را می‌گیرد و تعداد درست ها بر تعداد کل نمونه هارا برمی‌گرداند. تابع ()confusion_matrix نیز لیبل پیش‌بینی ها را گرفته و به کمک دستور crosstab در ماتریس آشفتگی آن را برمی‌گرداند.

حالا برای قسمت ب سوال 3 تابع ()result_prob را تعریف می‌کنیم. این تابع برای هر پیش‌بینی، تعداد لیبل های پیش‌بینی را به تعداد k تقسیم کرده و احتمال لیبل خود را برمی‌گرداند. حالا این تابع را برای K های مختلف فراخوانده و نمودار هیستوگرام بر اساس درست بودن یا غلط بودن پیش‌بینی ها رسم می‌کنیم.

برای قسمت دوم که یادگیری بر اساس معیار است. ابتدا از پکیج LMNN داده ها را fit و سپس transform می‌کنیم. این کار را برای K(پارامتر روش LMNN) های مختلف انجام داده و نمودار آن را رسم

می‌کنیم. سپس برای k = 15 داده های انتقال یافته را به مدل KNN داده و دقت و ماتریس آشفتگی آن را چاپ می‌کنیم. مشابه این اعمال را برای LFDA انجام می‌دهیم.

منابع:

Large Margin Nearest Neighbors (LMNN) Algorithm (open-instruction.com)

Deep Large Margin Nearest Neighbor for Gait Recognition (degruyter.com)