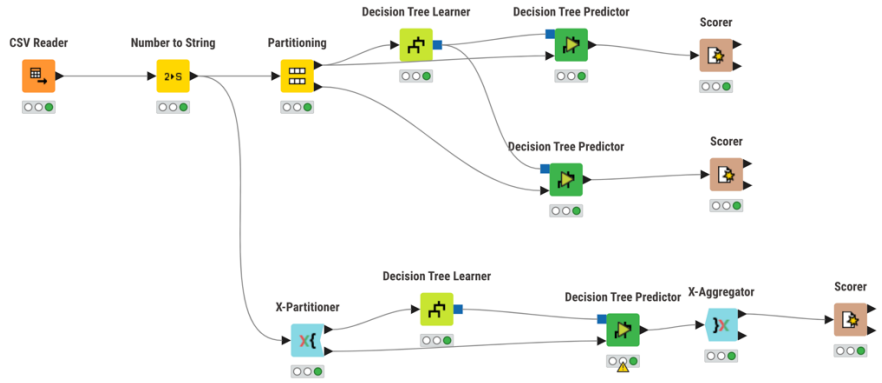
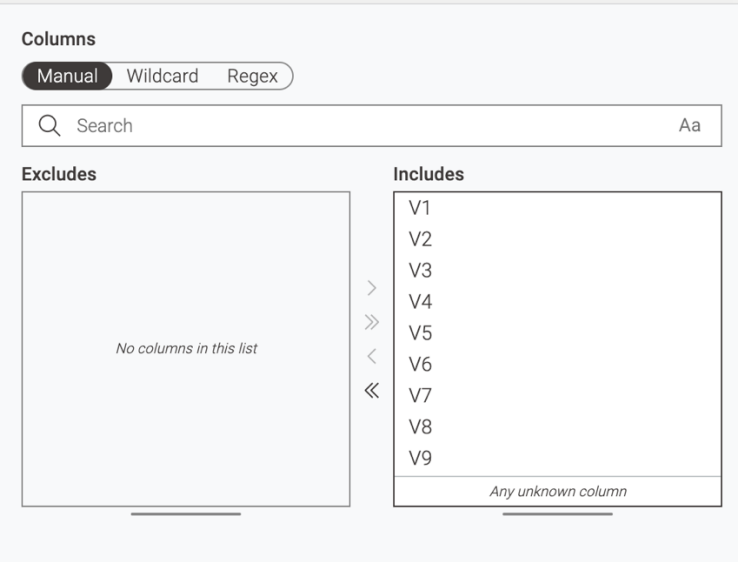
امیر مهدی اسلامی ۴۰۱۰۰۴۳۳

سوال ۱ عملی :

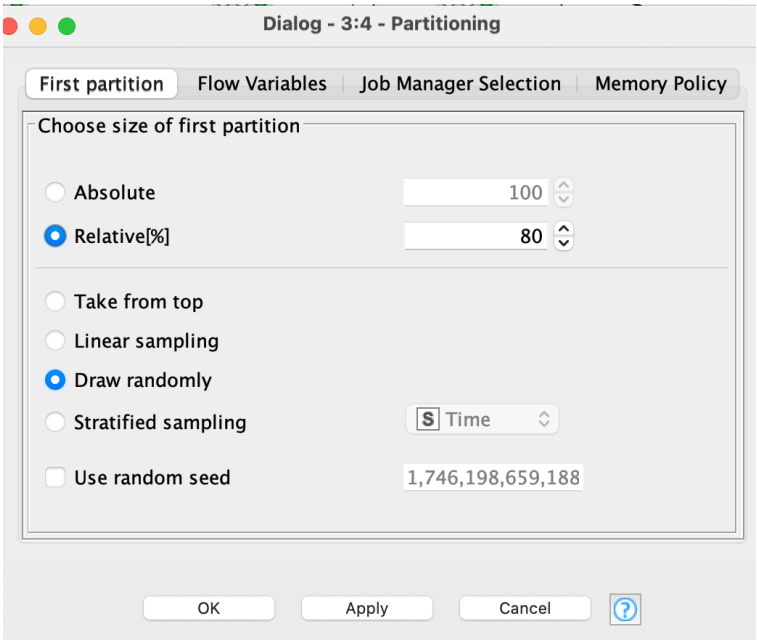
۱.۱.۲



ابتدای کار از نود number to string استفاده میکنیم برای تبدیل داده های numeric به string به خاظر اینکه نود decision tree learner فقط بر روی داده هایی که تایپشون string هستند میتواند کار بکند



با توجه به تصویر تمام داده های ما numeric بودند که ما ان ها را به string تبدیل کردیم

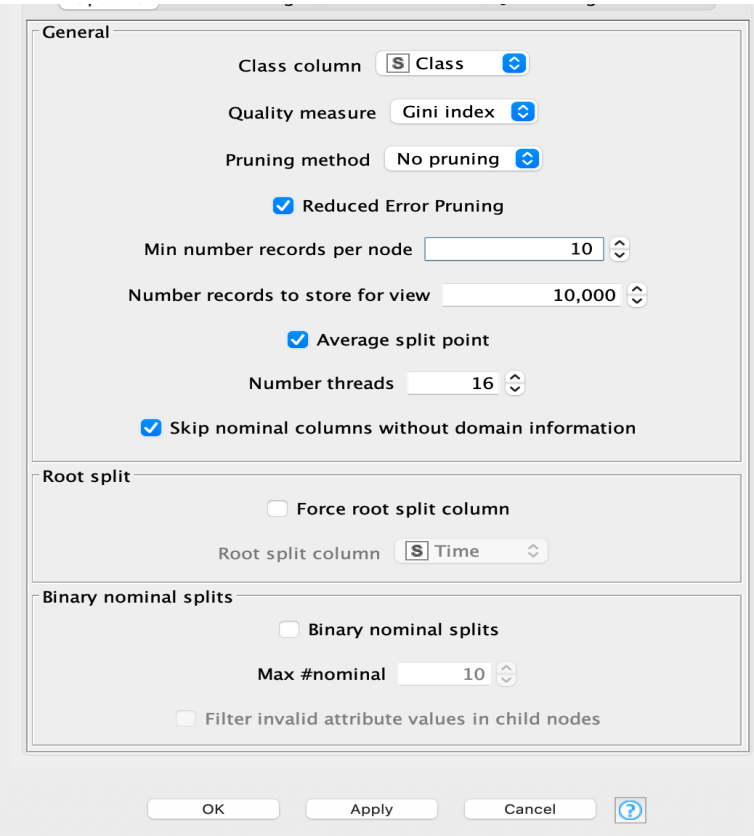


حالا با استفاده از نود partition داده هایمان را به train , test تقسیم میکنیم

در اینجا با انتخاب relative ٪ ۸۰ ما ۸۰ درصد از داده هایمان را به عنوان اموزش انتخاب کردیم

با انتخاب draw randomly هم گفتیم که بیاد و این ۸۰ درصد از داده ها را رندوم انتخاب بکند

البته میتوانستیم stratified sampling را بر روی class و با یک seed مشخص هم انتخاب بکنیم که اگه این کار را میکردیم توزیع داده هایمان طبق class تقسیم میشد

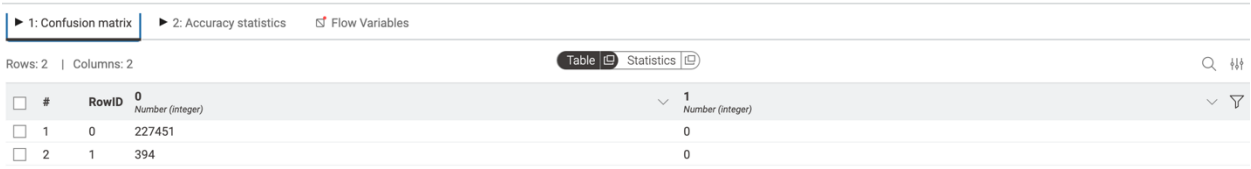


در نود decision tree learner هم تنظیمات را اینگونه انتخاب کردم

در قسمت class column ستونی که لیبل دار هست را انتحاب کردم و معیار تصمیم گیری هم ابتدا gini indedx انتخاب کردم

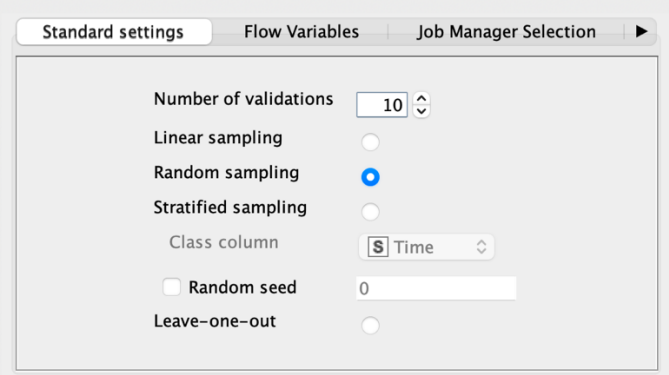
سپس این نود را به decision tree predictor وصل کردم که با توجه به الگوریتمی که بدست اوردیم بیاد لیبل کلاس را برای ما پیش بینی بکند

و بعد از ان برای اینکه بتوانیم accuracy مدل را بدست بیاورم به نود scorer وصل کردم .



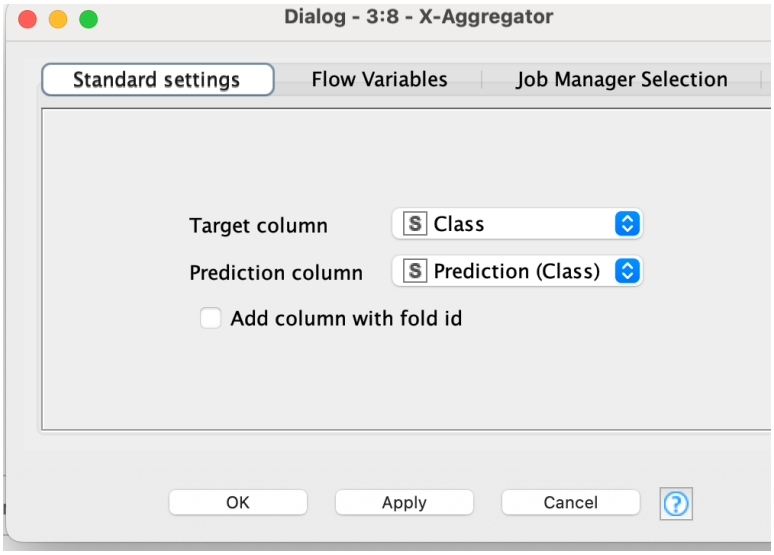
بخش دوم :

برای استفاده از cross validation هم دو نود x-partitioner , x-aggregator را اضافه کردم



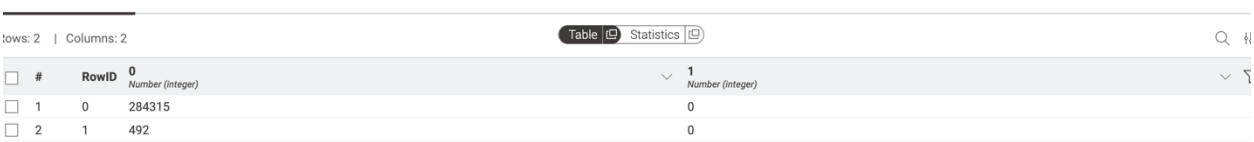
در اینجا در بخش number of validation تعداد validation را انتخاب میکنیم که من به صورت دیفالت خوده ۱۰ را انتخاب کردم و گذاشتم داده هایم به صورت رندوم انتخاب بشوند.

بقیه ی مراحل هم مانند قبل هست نود x-partitioner را به decision tree learner میدهیم تا مدل ما را اموزش بده و بعد از اینکه اموزش داد نود decision tree را به x-aggregator وصل میکنیم .



درنود x-aggregator هم ستون هدف و ستون پیش بینی را انتخاب میکنیم .

بعد از این مورد برای ارزیابی این نود را به scorer وصل میکنیم



بخش سوم)

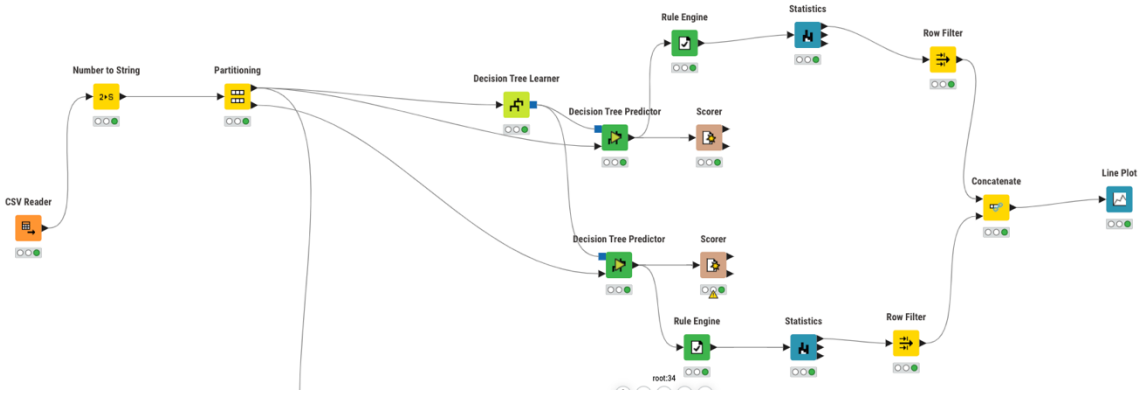
هرکدام از این معیار ها مزایا و معایب خاص خودش را دارد ولی اگر ما سرعت بیش تری در عملیات میخواهیم باید از معیار gini استفاده بکنیم چون سرعتش نسبت به entropy , gain ratio خیلی بیش تره و علتش هم بار عملیاتی کم ترش هست

ولی اگه دقت بیش تری میخواهیم معیار های entropy , gain ratio نسبت به gini خیلی بهتر عمل میکنند

یک مزیتی که gain ratio نسبت به entropy , gini دارد این است که دقت بهتر دارد و همچنین برای داده هایی که منحصر به فرد هستند مقاومت بالایی دارند در حقیقت Gain Ratioنرمالسازی شده است، بنابراین به ویژگی‌هایی که تعداد زیادی کلاس منحصربه‌فرد دارند، تنبیه می‌کند. در اینجا ما داده های منحصر به فرد نداشتیم که بخواهیم از gain ratio استفاده همچنین در این دیتاست دقت برای gini به اندازه ای خوب هست که به سراغ انتروپی و gain ratio نرویم

سوال ۲ عملی)

برای این قسمت کل نود های ما به این صورت هست

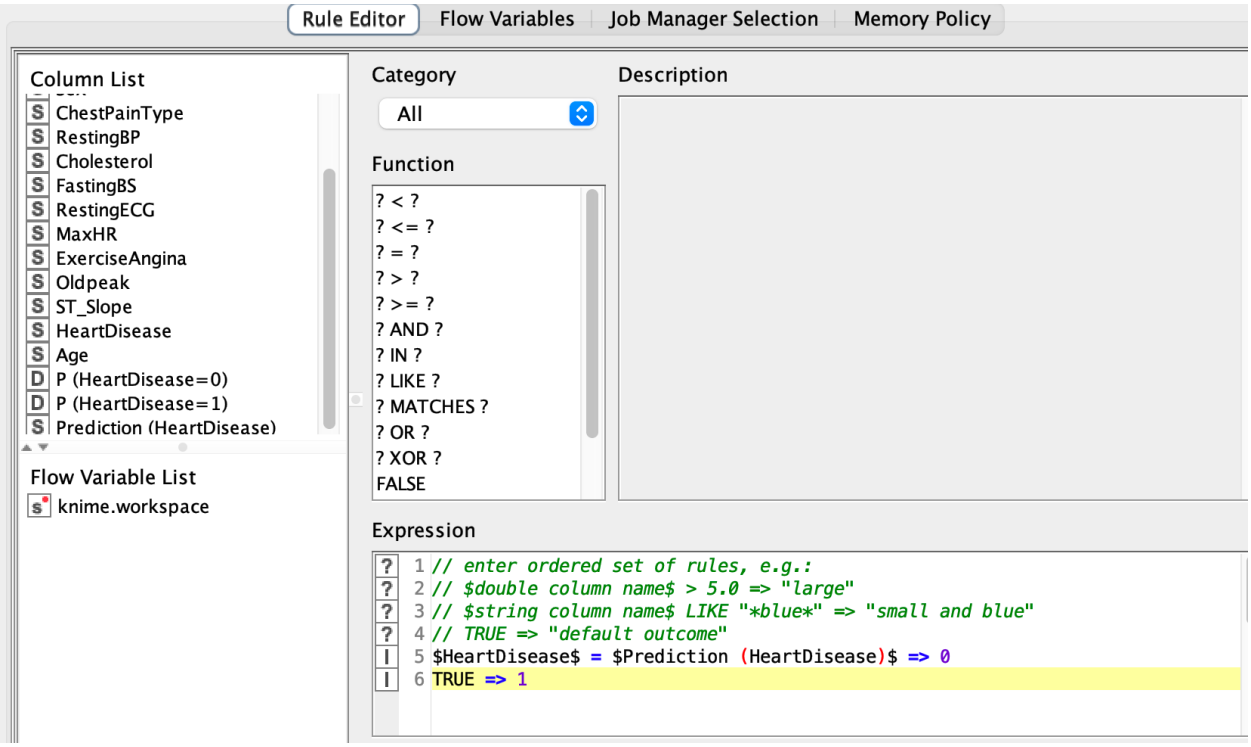


بسیاری از نود ها و مراحل در قسمت قبلی توضیح داده شده

ولی برای نشان دادن پیچیدگی مراحل و overfiting از نودهای rule engine , statistic , row filter , line plot استفاده کردم که در ادامه هرکدام از مراحل را توضیح میدم

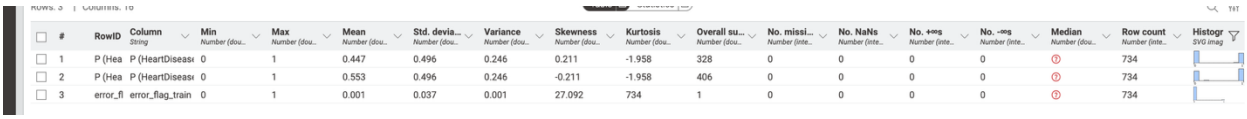
در rule engineاین دستورات وارد میکنیم برای اینکه مشخص بکنیم کدوم رکورد ها غلط پیش بینی شده اند

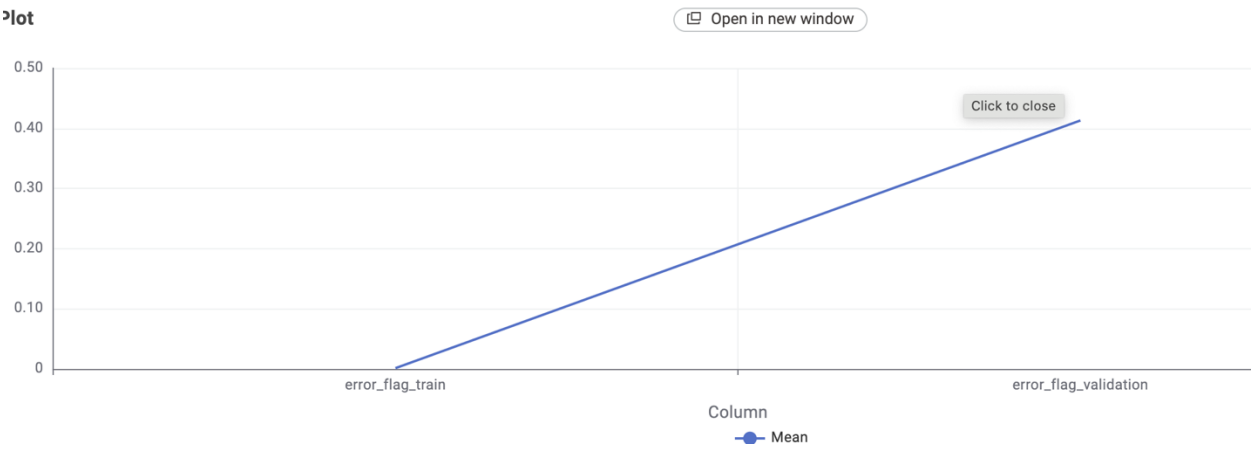
برای داده های تست هم مانند train همین کار را میکنیم سپس با استفاده از line plot خطای داده های تست و ترین را باهم. دیگر مقایسه میکنیم .



برای نشان دادن و محاسبه ی train , test مانند قبل دو نود decision tree predictor داریم که با استفاده از الگوریتم بدست امده از decision tree learner و داده های تقسیم بندی شده از partitioner برای ما پیش بینی میکند

برای اینکه داده های ما overfit بشوند یکبار min number record per node را برابر ۱ قرار میدهیم خروجی مانند شکل پایین میشود همانطور که مشاهده میکنید generalization error ما نسبت به train error ما خیلی بیش تر است که به ما overfit شدن داده هایمان را نشان میدهد





داده های train قبل از pruning با min number recored =۱

Rows: 2 | Columns: 2

#	RowID	0 Number (integer)	1 Number (integer)
1	0	328	0
2	1	1	405

Rows: 3 | Columns: 11

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	328	1	405	0	1	0.997	1	0.998	0.998		
2	1	405	0	328	1	0.998	1	0.998	1	0.999		
3	Overall										0.999	0.997

داده های تست قبل از pruning با min number record =۱

Rows: 2 | Columns: 2

#	RowID	0 Number (integer)	1 Number (integer)
1	0	46	14
2	1	9	62

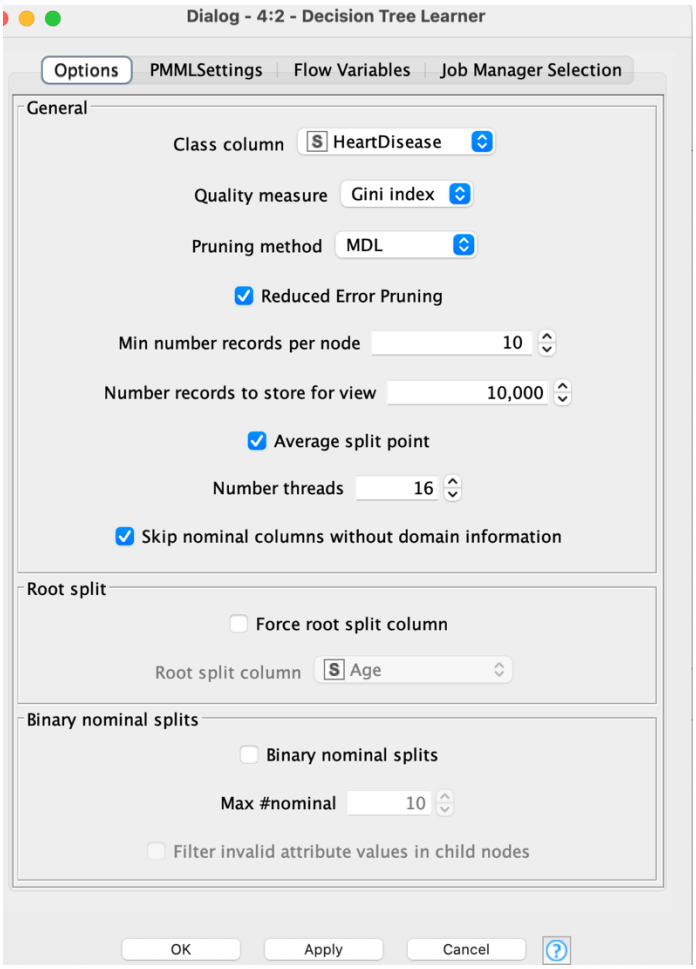
Rows: 3 | Columns: 11

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	46	9	62	14	0.767	0.836	0.767	0.873	0.8		
2	1	62	14	46	9	0.873	0.816	0.873	0.767	0.844		
3	Overall										0.824	0.644

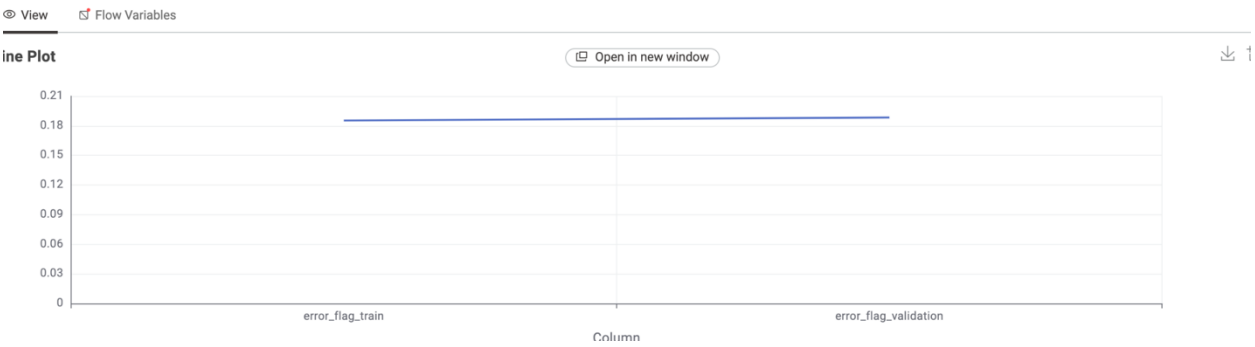
همانطور که از شکل و داده هایماز مشخص هست مدل ما خیلی پیچیده شده است و برای داده های train ما دقت خیلی بالایی داربم ولی برای داده های تست دقت ما خیلی کم هست

بخش ۲)

حالا اگر در تنظیمات decision tree learner ما بیاییم و peruning method = mdl را انتخاب بکنیم خطای تست و ترین ما خیلی کمتر میشود



اکر تنظیمات ب این گونه تغییر بکند نتایج ما به صورت زیر است :



خطای داده های train بعد از pruning

Rows: 2 | Columns: 2

#	RowID	0 Number (integer)	1 Number (integer)
1	0	251	77
2	1	63	343

Rows: 3 | Columns: 11

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	251	63	343	77	0.765	0.799	0.765	0.845	0.782		
2	1	343	77	251	63	0.845	0.817	0.845	0.765	0.831		
3	Overall										0.809	0.613

Missing Value

داده های تست بعد از pruning:

Rows: 2 | Columns: 2

#	RowID	0 Number (integer)	1 Number (integer)
1	0	66	16
2	1	15	87

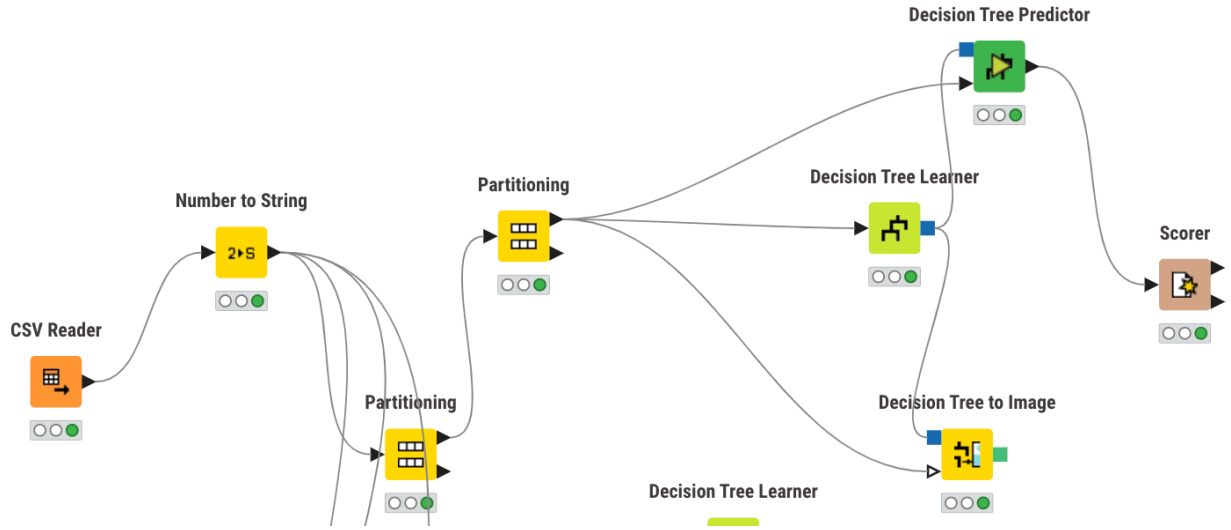
15

Rows: 3 | Columns: 11

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	66	15	87	16	0.805	0.815	0.805	0.853	0.81		
2	1	87	16	66	15	0.853	0.845	0.853	0.805	0.849		
3	Overall										0.832	0.659

همانطور که از تصاویر بالا مشخص هست pruning باعث میشود مدل ما overfit نشود و تعادل بین داده های train , test باشه

بخش ۳)



برای این بخش ما از دو partitioning استفاده کردیم در ابتدا ۸۰ درصد داده ها را برای train و ۲۰ درصد داه ها را برای test اختصاص میدهیم و سپس در partitioning دوم یک تقسیم بندی دوباره روی داده های train انجام میدهیم

و از کم به زیاد درصد داده ها را زیاد میکنیم و بررسی میکنیم ک خطای generalization ما چگونه است

وقتی از ۸۰ درصد داده های اموزش استفاده میکنیم خروجی نود scorer ب این صورت میشود :

Rows: 3 | Columns: 11 — Table | Statistics

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	222	21	304	40	0.847	0.914	0.847	0.935	0.879		
2	1	304	40	222	21	0.935	0.884	0.935	0.847	0.909		
3	Overall										0.896	0.788

وقتی از ۷۰ درصد از داده های اموزش استفاده میکنیم :

Rows: 3 | Columns: 11 — Table | Statistics

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	192	22	262	37	0.838	0.897	0.838	0.923	0.867		
2	1	262	37	192	22	0.923	0.876	0.923	0.838	0.899		
3	Overall										0.885	0.766

وقتی از ۵۰ درصد از داده های اموزش استفاده میکنیم :

Rows: 3 | Columns: 11

Table | Statistics

#	RowID	TruePositives Number (integer)	FalsePositives Number (integer)	TrueNegatives Number (integer)	FalseNegatives Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's kappa Number (double)
1	0	139	20	183	25	0.848	0.874	0.848	0.901	0.861		
2	1	183	25	139	20	0.901	0.88	0.901	0.848	0.891		
3	Overall										0.877	0.751

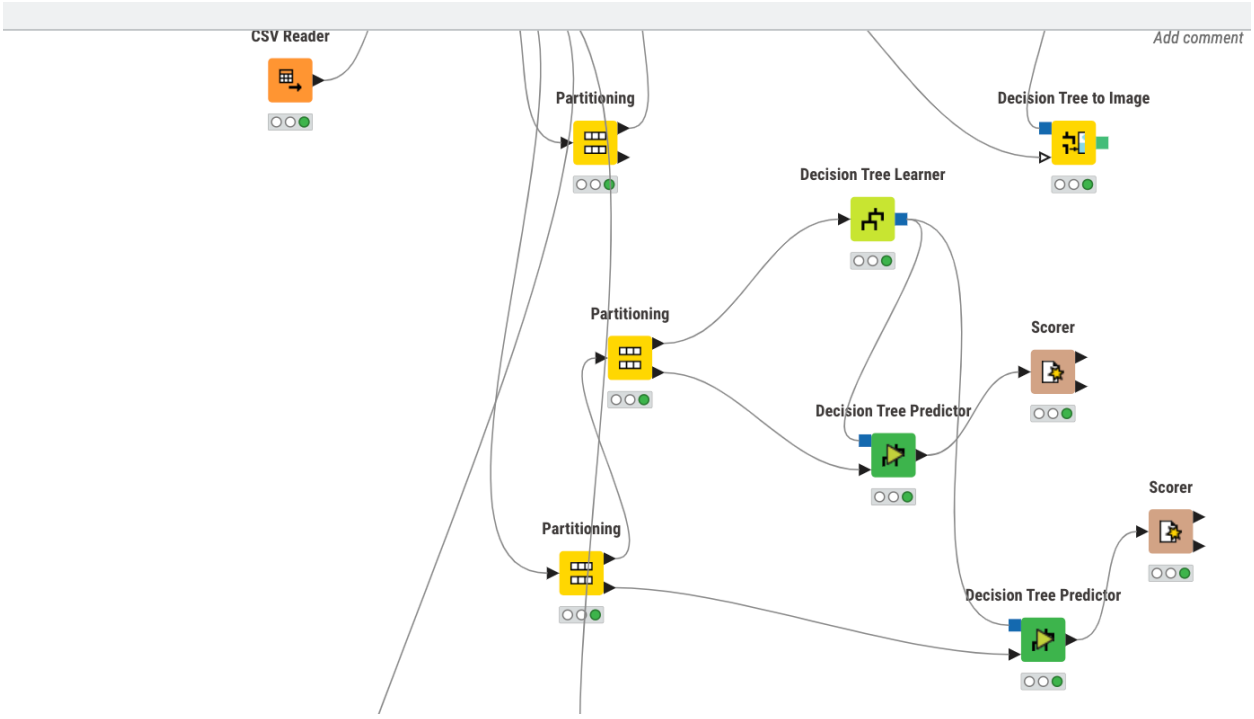
همانطور که مشاهده میکنید وقتی از داده های کم استفاده میکنیم دقت داده های اموزشی ما دارد کم تر میشود همچنین اگر تعداد درصد داده های مدل را کمتر بکنیم مدل ما به خاطر تعداد داده های کمتر به سمت overfiting می رود و تفاوت بین train error , test error بیش تر می شود

یکی از راه های حساب کردن خطای generalization error تفاوت بین train error , test errorاسن که در بالا گفته شده ولی یکی دیگر از این موارد در درس امده شده است و اینگونه حساب میشود :

$$\text{Error generalization} = \text{error(t)} + x * k/N \text{ train}$$

با توجه به visiual کردن درخت تصمیم و محاسبه ی مقادیر بالا هم به این مورد میرسیم که تعداد داده های اموزشی بیش تر باشد احتمال overfiting کم تر و generalization error هم کمتر است .

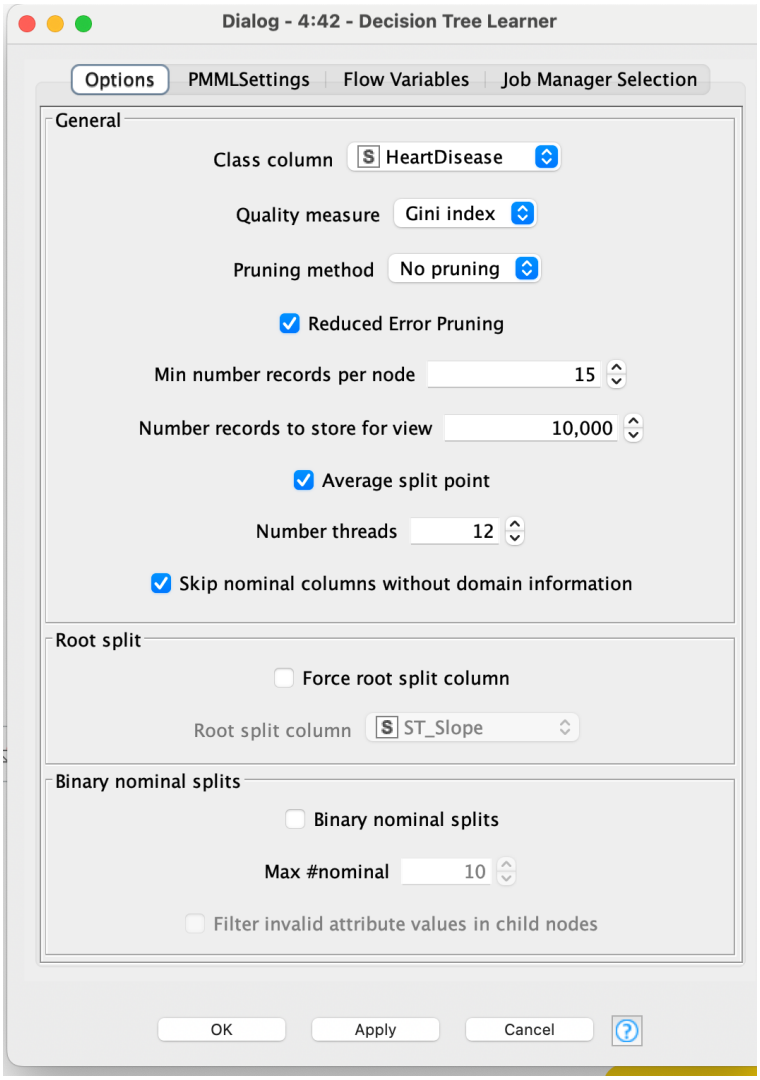
بخش ۴)



همانطور که میدانیم validation set به ما کمک میکنه تا بهترین تنظیمات را پیدا بکنیم همچنین در بهینه سازی مدل و جلوگیری از overfiting به ما کمک میکنه تا ببینیم تا کجا اموزش را جلو ببریم

برای این بخش هم مانند قبل ۲ قسمت partitioner داریم در partitioner اول ۸۰ درصد داده ها را برای اموزش داریم و در partitioner دوم میاییم از این ۸۰ درصد داده ای که جدا کردیم ۳۰ درصدش را برای validation و ۷۰ درصد را برای train انتخاب میکنیم و در ادامه سعی میکنیم با تغییر دادن پارامتر ها بهترین مدل را که بیش ترین دقت را دارد انتخاب کنیم

مثلا اولین تنظیم را اینگونه فرض میکنیم :



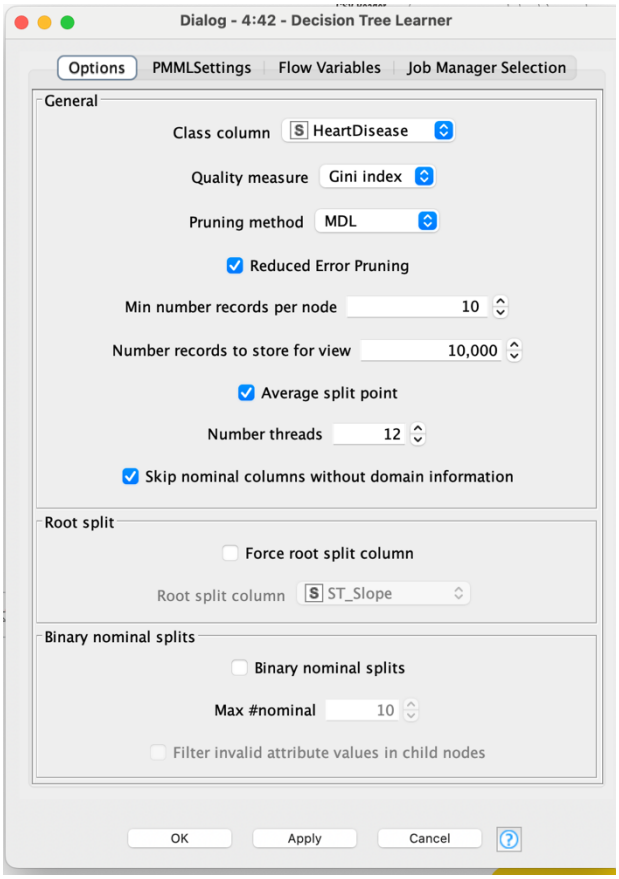
مقدار دقت داده ی تست اینگونه است :

#	RowID	TruePositiv... Number (integer)	FalsePositi... Number (integer)	TrueNegati... Number (integer)	FalseNegat... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's ka... Number (double)
1	0	79	12	125	32	0.712	0.868	0.712	0.912	0.782	?	?
2	1	125	32	79	12	0.912	0.796	0.912	0.712	0.85	?	?
3	Overall	?	?	?	?	?	?	?	?	?	0.823	0.635

مقدار دقت داده ی validation هم اینگونه است :

#	RowID	TruePositiv... Number (integer)	FalsePositi... Number (integer)	TrueNegati... Number (integer)	FalseNegat... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's ka... Number (double)
1	0	30	6	45	11	0.732	0.833	0.732	0.882	0.779		
2	1	45	11	30	6	0.882	0.804	0.882	0.732	0.841		
3	Overall										0.815	0.621

حال دوباره تنظیمات را تغییر میدهیم :



برای داده های تست مقدار دقت زیر را داریم :

#	RowID	TruePositiv... Number (integer)	FalsePositi... Number (integer)	TrueNegati... Number (integer)	FalseNegat... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's ka... Number (double)
1	0	90	22	115	21	0.811	0.804	0.811	0.839	0.807		
2	1	115	21	90	22	0.839	0.846	0.839	0.811	0.842		
3	Overall										0.827	0.65

برای داده های validation مقدار دقت زیر را داریم :

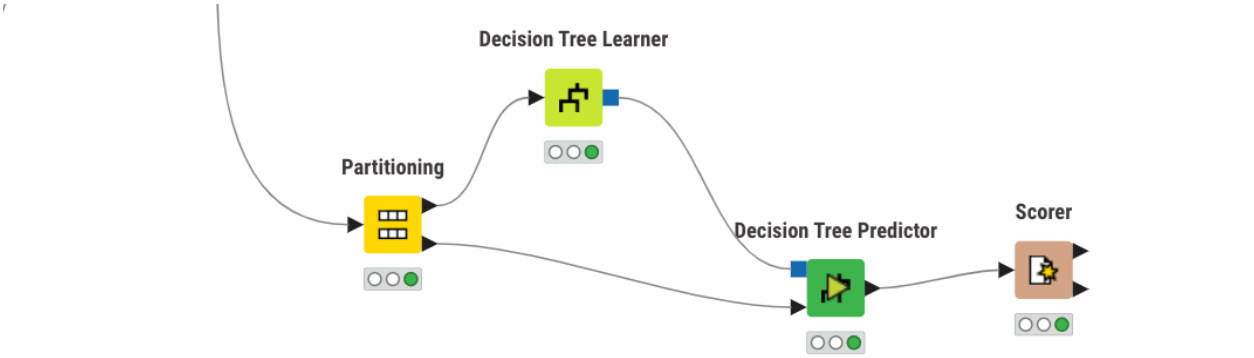
#	RowID	TruePositiv... Number (integer)	FalsePositi... Number (integer)	TrueNegati... Number (integer)	FalseNegat... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's ka... Number (double)
1	0	34	8	43	7	0.829	0.81	0.829	0.843	0.819		
2	1	43	7	34	8	0.843	0.86	0.843	0.829	0.851		
3	Overall										0.837	0.671

میبینیم با تغییر دادن پارامتر ها به این صورت دقت الگوریتم ما بیش تر شده

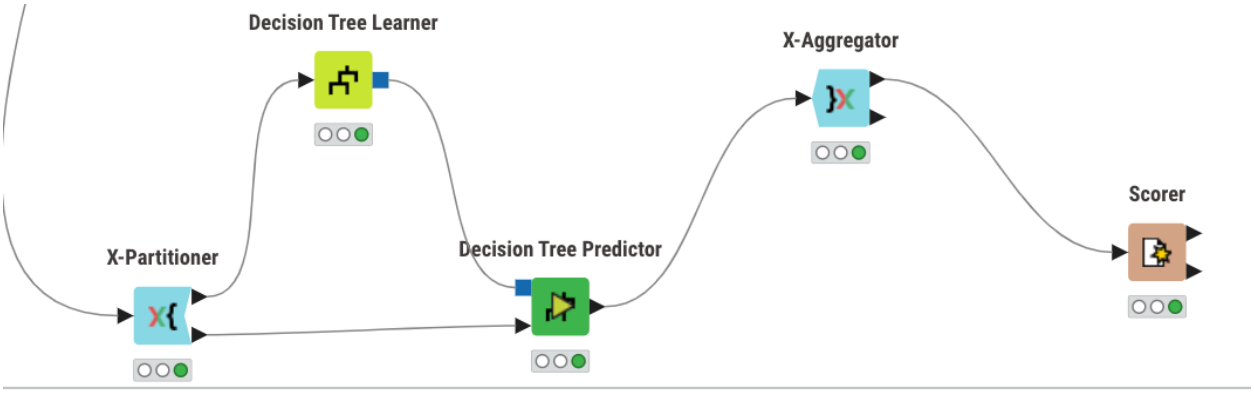
در حقیقت در تنظیم اول حدود ۸۱ درصد دقتش بود در تنظیم دوم حدود ۸۴ درصد دقتش هست

بخش ۵)

روش holdout



روش cross validation



دقت روش holdout

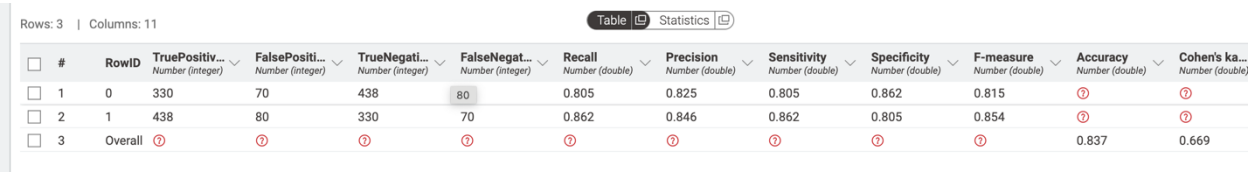
Rows: 3 | Columns: 11

Table | Statistics

#	RowID	TruePositiv... Number (integer)	FalsePositi... Number (integer)	TrueNegati... Number (integer)	FalseNegat... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's ka... Number (double)
1	0	68	19	83	14	0.829	0.782	0.829	0.814	0.805	?	?
2	1	83	14	68	19	0.814	0.856	0.814	0.829	0.834	?	?
3	Overall	?	?	?	?	?	?	?	?	?	0.821	0.639

Open table column header submenu

دقت روش cross validation



Rows: 3 | Columns: 11

Table | Statistics

#	RowID	TruePositiv... Number (integer)	FalsePositi... Number (integer)	TrueNegati... Number (integer)	FalseNegat... Number (integer)	Recall Number (double)	Precision Number (double)	Sensitivity Number (double)	Specificity Number (double)	F-measure Number (double)	Accuracy Number (double)	Cohen's ka... Number (double)
1	0	330	70	438	80	0.805	0.825	0.805	0.862	0.815	?	?
2	1	438	80	330	70	0.862	0.846	0.862	0.805	0.854	?	?
3	Overall	?	?	?	?	?	?	?	?	?	0.837	0.669

روش holdout اینگونه است که ما داده هایماز را به ۲ بخش تقسیم میکنیم مثلا در اینجا من ۷۰ درصد داده ها را برای اموزش و ۳۰ درصد داده ها را برای تست قرار دادم حالا میام اموزش رو روی داده های اموزش انجام میدم و تست رو روی داده های تست انجام میدم

روش cross validation هم اینجوری هست که من داده هام رو به k بخش تقسیم میکنم مثلا در اینجا به ۱۰ قسمت تقسیم کردم سپس مدلم به صورت چرخه ای بین بخش های مختلف اموزش داده می شود و سپس مدلم در بخش تست ارزیابی میشود

هرکدام از این روش‌ها مزایا و معایب خود را دارند، بنابراین انتخاب بهتر بستگی به شرایط پروژه و نیازهای خاص شما دارد.

اگر داده‌ها محدود یا کوچک باشند، Cross Validationمعمولاً بهتر است، زیرا از تمام داده‌ها برای آموزش و ارزیابی استفاده می‌کند و دقت بیشتری در ارزیابی مدل ایجاد می‌کند.

اگر سرعت و سادگی مهم باشند، Holdoutمی‌تواند انتخاب بهتری باشد، زیرا سریع‌تر است و نیازی به اجرای چندین بار فرآیند آموزش ندارد.

به طور کلی، Cross Validationبه دلیل دقت بالاتر و استفاده بهینه‌تر از داده‌ها ترجیح داده می‌شود، مگر در مواقعی که نیاز به سرعت و زمان کمتر دارید.