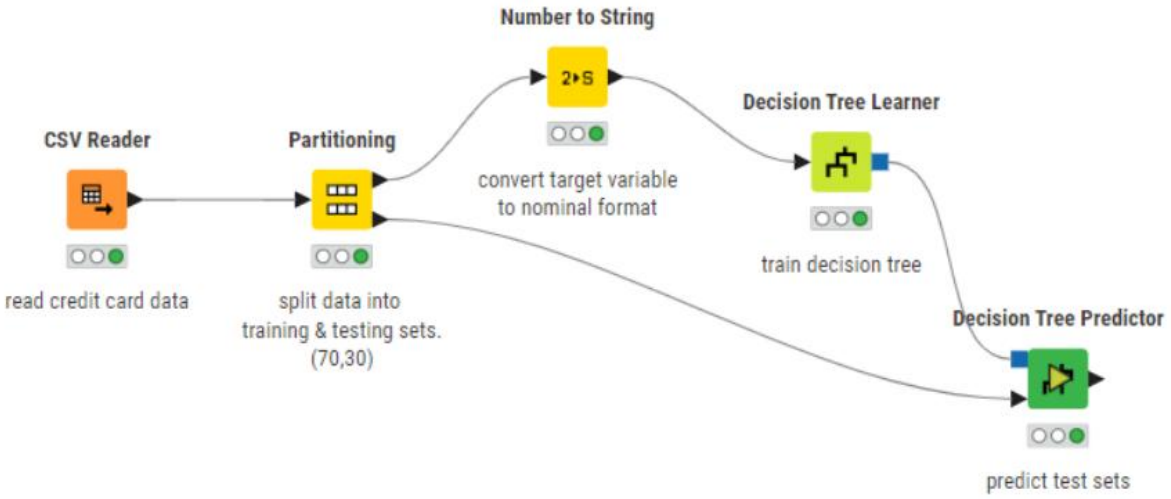
گزارش بخش عملی تکلیف دوم درس داده کاوی

کلاس‌بندي مجموعه داده کارت‌های اعتباری

1. با استفاده از درخت تصميم وضعيت كارتهاي اعتباري را پيش بيني كنيد.



برای ساخت درخت تصمیم ابتدا داده ها را با استفاده از نود csv reader میخوانیم و دیتا را با استفاده از نود partitioning به دو دسته training و testing به نسبت (70و30) تقسیم میکنیم. سپس به دلیل اینکه نود decision tree learner به این شکل هست که باید target variableهای دیتایی که دریافت میکند به شکل string باشد ، با استفاده از نود number to string ستون کلاس های دیتا را از حالت integer به فرمت string تبدیل میکنیم. سپس بعد از اینکه با decision tree learner مدل train کردیم باید مدل را به همراه داده های تست به decision tree predictorمیدهیم تا مدل را روی داده ها تست ، آزمایش کنیم.

برای درست کردن مدل از معیار gini indexاستفاده کرده ایم.

نتیجه پیش بینی مدل روی داده های تست برای 28 سطر اول اورده شده است.

Classified Data (Table)

Rows: 85443 | Columns: 32

#	RowID	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
		Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number
1	Row3	1	-0.966	-0.185	1.793	-0.863	-0.01	1.247	0.238	0.377	-1.387	-0.055	-0.226	0.178	0.508
2	Row16	12	1.103	-0.04	1.267	1.289	-0.736	0.288	-0.586	0.189	0.782	-0.268	-0.45	0.937	0.708
3	Row17	13	-0.437	0.919	0.925	-0.727	0.916	-0.128	0.708	0.088	-0.665	-0.738	0.324	0.277	0.253
4	Row18	14	-5.401	-5.45	1.186	1.736	3.049	-1.763	-1.56	0.161	1.233	0.345	0.917	0.97	-0.267
5	Row22	18	1.167	0.502	-0.067	2.262	0.429	0.089	0.241	0.138	-0.989	0.922	0.745	-0.531	-2.105
6	Row23	18	0.247	0.278	1.185	-0.093	-1.314	-0.15	-0.946	-1.618	1.544	-0.83	-0.583	0.525	-0.453
7	Row25	22	-2.074	-0.121	1.322	0.41	0.295	-0.96	0.544	-0.105	0.476	0.149	-0.857	-0.181	-0.655
8	Row28	23	-0.414	0.905	1.727	1.473	0.007	-0.2	0.74	-0.029	-0.593	-0.346	-0.012	0.787	0.636
9	Row29	23	1.059	-0.175	1.266	1.186	-0.786	0.578	-0.767	0.401	0.699	-0.065	1.048	1.006	-0.542
10	Row32	26	-0.53	0.874	1.347	0.145	0.414	0.1	0.711	0.176	-0.287	-0.485	0.872	0.852	-0.572
11	Row33	26	-0.53	0.874	1.347	0.145	0.414	0.1	0.711	0.176	-0.287	-0.485	0.872	0.852	-0.572
12	Row47	34	0.202	0.497	1.374	0.571	-0.631	-0.54	-0.076	-0.917	0.27	-0.48	-0.513	0.681	0.09
13	Row48	35	1.386	-0.794	0.778	-0.865	-1.064	0.351	-1.191	0.053	-0.304	0.577	-1.631	0.043	2.048
14	Row51	36	-1.005	-0.986	-0.038	3.71	-6.632	5.122	4.372	-2.007	-0.279	-0.231	0.145	-0.063	-0.8
15	Row60	41	0.986	-0.203	-0.493	0.408	0.306	-0.231	0.585	-0.208	-0.248	-0.192	-0.628	0.43	0.811
16	Row61	41	1.139	-1.193	1.407	-0.33	-2.07	-0.242	-1.307	0.105	0.135	0.494	-0.895	-0.183	0.146
17	Row66	44	-0.715	0.515	1.822	0.616	0.849	-0.112	1.506	-0.798	0.245	0.265	-1.108	-0.36	-0.163
18	Row69	46	-1.923	-0.87	2.32	1.989	0.417	-0.38	0.472	-0.557	-0.649	1.411	-0.518	-0.985	-0.401
19	Row71	46	-0.378	0.733	-0.12	0.186	2.594	3.797	0.059	0.977	-0.413	0.007	-0.625	-0.116	-0.215
20	Row72	47	1.198	0.237	0.51	0.658	-0.365	-0.745	0.079	-0.131	-0.052	-0.11	0.307	0.671	0.438
21	Row81	52	1.147	0.059	0.264	1.211	-0.044	0.301	-0.133	0.228	0.252	0.084	0.645	0.413	-1.468
22	Row83	53	-1.199	-1.474	1.84	-4.516	0.328	-0.174	0.96	-1.026	1.7	-0.079	1.663	0.486	-0.933
23	Row97	67	-0.653	0.16	1.592	1.297	0.997	-0.343	0.47	-0.132	-0.198	-0.105	-0.544	0.302	0.141
24	Row10	68	1.157	0.037	0.557	0.52	-0.48	-0.353	-0.222	0.158	0.011	0.106	1.612	0.354	-1.435
25	Row10	69	0.299	2.143	-1.542	1.561	0.938	-2.146	1.406	-0.778	0.329	0.127	1.613	-0.096	0.655
26	Row10	69	-0.608	0.307	1.473	1.191	0.021	0.888	1.19	0.042	-0.446	-0.08	0.66	-0.112	-1.469
27	Row10	73	1.239	0.293	0.086	1.178	0.373	0.326	0.119	-0.025	0.173	-0.154	-1.089	0.871	1.178
28	Row11	73	0.926	-0.358	1.377	1.901	-1.058	0.302	-0.462	0.25	1.254	-0.436	-0.681	1	-0.653

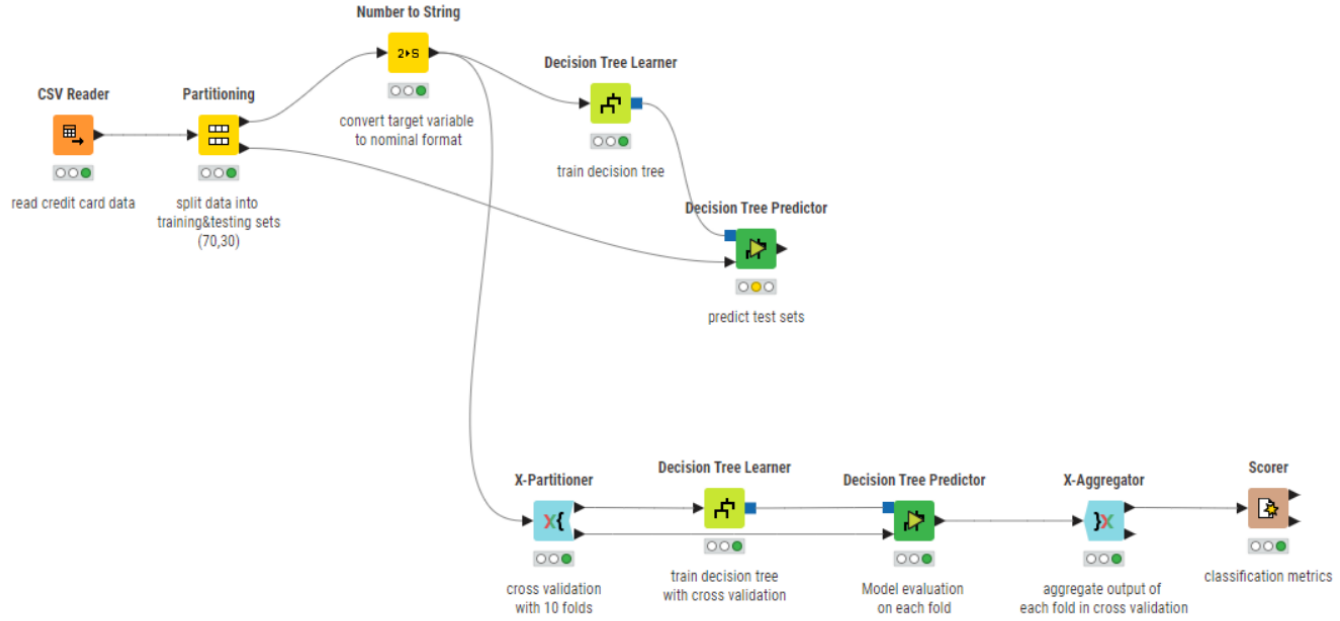
Rows: 85443 | Columns: 32

V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27
Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...	Number (dou...
0.508	-0.288	-0.631	-1.06	-0.684	1.966	-1.233	-0.208	-0.108	0.005	-0.19	-1.176	0.647	-0.222	0.063
0.708	-0.469	0.355	-0.247	-0.009	-0.596	-0.576	-0.114	-0.025	0.196	0.014	0.104	0.364	-0.382	0.093
0.253	-0.292	-0.185	1.143	-0.929	0.68	0.025	-0.047	-0.195	-0.673	-0.157	-0.888	-0.342	-0.049	0.08
-0.267	-0.479	-0.527	0.472	-0.725	0.075	-0.407	-2.197	-0.504	0.984	2.459	0.042	-0.482	-0.621	0.392
-2.105	1.127	0.003	0.424	-0.454	-0.099	-0.817	-0.307	0.019	-0.062	-0.104	-0.37	0.603	0.109	-0.041
-0.453	0.081	1.555	-1.397	0.783	0.437	2.178	-0.231	1.65	0.2	-0.185	0.423	0.821	-0.228	0.337
-0.655	-0.28	-0.212	-0.333	0.011	-0.488	0.506	-0.387	-0.404	-0.227	0.742	0.399	0.249	0.274	0.36
0.636	-0.086	0.077	-1.406	0.776	-0.943	0.544	0.097	0.077	0.457	-0.038	0.643	-0.184	-0.277	0.183
-0.542	-0.04	-0.219	0.004	-0.194	0.042	-0.278	-0.178	0.014	0.214	0.014	0.003	0.295	-0.395	0.081
-0.572	0.101	-1.52	-0.284	-0.311	-0.404	-0.823	-0.29	0.047	0.208	-0.186	0.001	0.099	-0.553	-0.073
-0.572	0.101	-1.52	-0.284	-0.311	-0.404	-0.823	-0.29	0.047	0.208	-0.186	0.001	0.099	-0.553	-0.073
0.09	-0.157	-0.589	-0.478	0.225	-0.669	-0.024	-0.222	0.719	-0.172	-0.166	0.776	0.818	0.443	0.143
2.048	-0.739	1.456	-0.272	-0.932	1.927	-0.66	-0.273	-0.229	-0.124	-0.131	-0.93	0.181	1.195	0.001
-0.8	-0.342	-0.931	0.511	0.092	0.824	1.19	-0.002	1.393	-0.382	0.97	0.019	0.571	0.333	0.857
0.811	0.372	1.026	-0.029	-0.308	-1.1	0.011	0.264	-0.306	-1.217	-0.078	-0.741	0.287	0.2	-0.075
0.146	-0.587	0.797	-0.892	-0.079	1.542	-0.984	-0.299	-0.156	-0.031	-0.02	0.434	-0.03	1.141	-0.009
-0.163	-0.784	-0.367	-0.653	-0.504	-0.261	0.435	0.107	-0.22	-0.018	-0.432	-0.13	0.337	-0.44	-0.675
-0.401	-0.831	0.338	0.03	0.371	-1.054	1.89	-0.369	-0.686	-0.779	1.086	0.519	-0.364	3.066	-0.589
-0.215	0.14	0.16	-0.602	-0.144	0.222	1.455	0.316	-0.108	-0.157	-0.195	1.014	0.146	-0.238	0.411
0.438	0.269	1.127	-0.197	0.008	-1.123	-0.591	-0.121	-0.15	-0.374	0.146	0.415	0.212	0.182	-0.016
-1.468	0.503	-0.627	-0.37	-0.032	-0.215	0.055	-0.256	-0.088	-0.111	-0.098	-0.323	0.633	-0.305	0.027
-0.933	-1.119	0.141	-2.812	-0.505	0.891	-1.512	-0.77	-0.453	0.335	-0.365	-0.31	-0.303	-1.244	-1.123
0.141	-0.208	0.024	-1.207	0.395	-0.751	0.441	0.226	0.038	0.336	-0.015	0.103	-0.265	-0.349	0.011
-1.435	0.797	0.745	0.223	-0.229	-0.365	-0.254	-0.222	-0.183	-0.612	0.197	0.175	0.032	0.099	-0.027
0.655	-4.318	1.039	0.405	2.729	1.116	-0.704	0.677	-0.291	-0	0.092	0.586	-0.397	-0.481	0.251
-1.469	0.285	-0.237	-0.913	0.243	0.123	1.175	0.374	0.047	0.165	0.064	-0.334	0.132	-0.218	-0.082
1.178	-0.218	-0.161	-0.534	-0.108	-0.848	-0.057	-0.104	-0.148	-0.112	-0.173	-0.715	0.817	-0.264	0.048
-0.653	-0.636	-1.703	-1.28	0.988	-1.479	-0.044	-0.124	-0.348	-0.677	0.101	0.596	0.322	-0.547	0.079

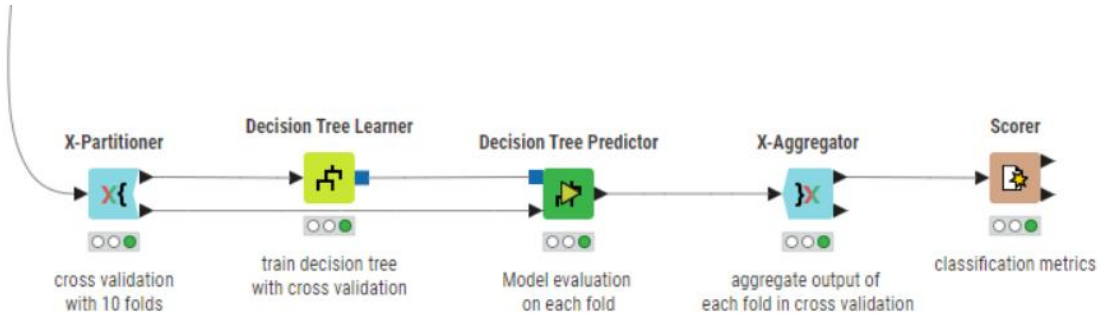
	V28 *Number (dou…*	Amount *Number (dou…*	Class *Number (inte…*	Predictio… *String*
	0.061	123.5	0	0
	0.037	12.99	0	0
	0.131	0.89	0	0
	0.95	46.8	0	0
	-0.011	2.28	0	0
	0.25	22.75	0	0
	0.243	26.43	0	0
	0.153	33	0	0
	0.024	12.99	0	0
	0.023	6.14	0	0
	0.023	6.14	0	0
	0.219	9.99	0	0
	0.02	30.9	0	0
	-0.076	1,402.95	0	0
	0.027	169.05	0	0
	0.042	96.94	0	0
	-0.473	59.9	0	0
	-0.396	35	0	0
	0.203	11.45	0	0
	0.016	3.63	0	0
	-0.001	6.67	0	0
	-0.734	89.17	0	0
	-0.049	19.85	0	0
	0.004	2.69	0	0
	-0.242	0.78	0	0
	-0.117	168.56	0	0
	0.009	1	0	0
	0.044	70.6	0	0

2. چگونه میتوانید با استفاده از validation-Cross در KNIME عملکرد این مدل دسته بندي را ارزیابي کنید؟ مراحل را به صورت عملي نشان دهید.

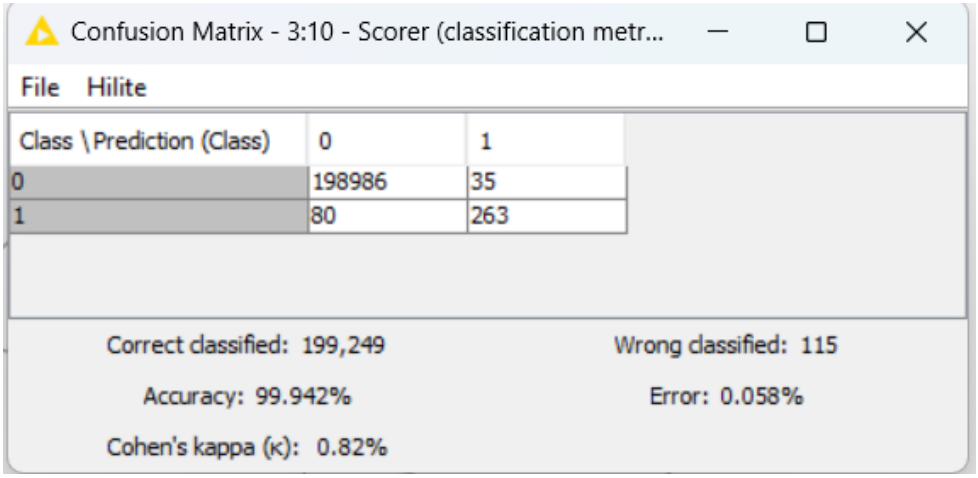
برای ارزیابی مدل با استفاده از cross validation طبق تصویر زیر باید ابتدا خروجی Number to string (که در اصل همان دیتا مربوط به credit card هست با این تفاوت که ستون مربوط به class به صورت string تعریف شده نه integer) را به ورودی x-partition متصل کنیم، این نود وظیفه این را دارد که داده ها را به چندین fold تقسیم کند، که در اصل هدف ما هم در cross validation همین می‌باشد که داده ها به بخش هایی با اندازه یکسان تقسیم شوند.



سپس خروجی مربوط به داده های train، x-partitioner را باید به decision tree learner بدهیم و خروجی تست x-partitioner را به decision tree predictor بدهیم تا نتیجه عملکرد درخت تصمیم را روی fold مربوط به داده های تست بدست آوریم. بعد از آن از X-Aggregator استفاده می‌کنیم تا نتایج مدل روی هربار آموزش و ارزیابی رو foldهای مختلف در cross validation را جمع آوری کنیم و بعد از نود scorer استفاده میکنیم تا معیار های ارزیابی را محاسبه کنیم.

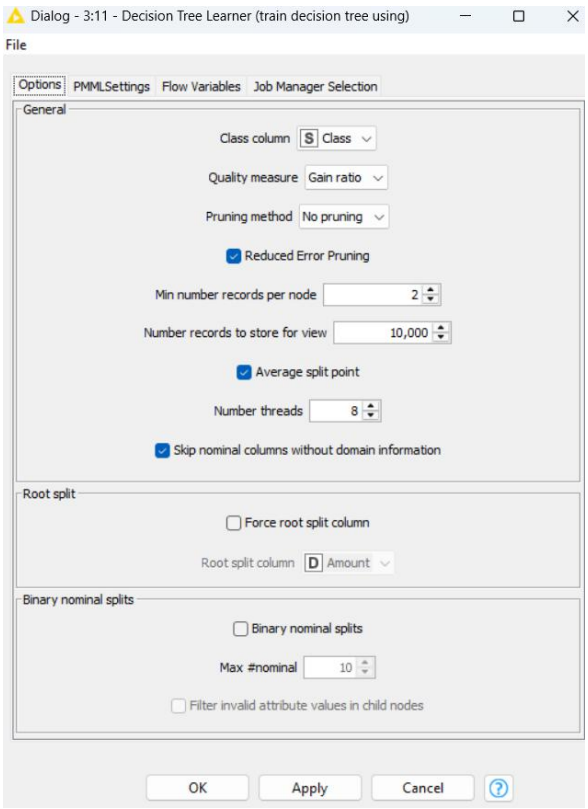


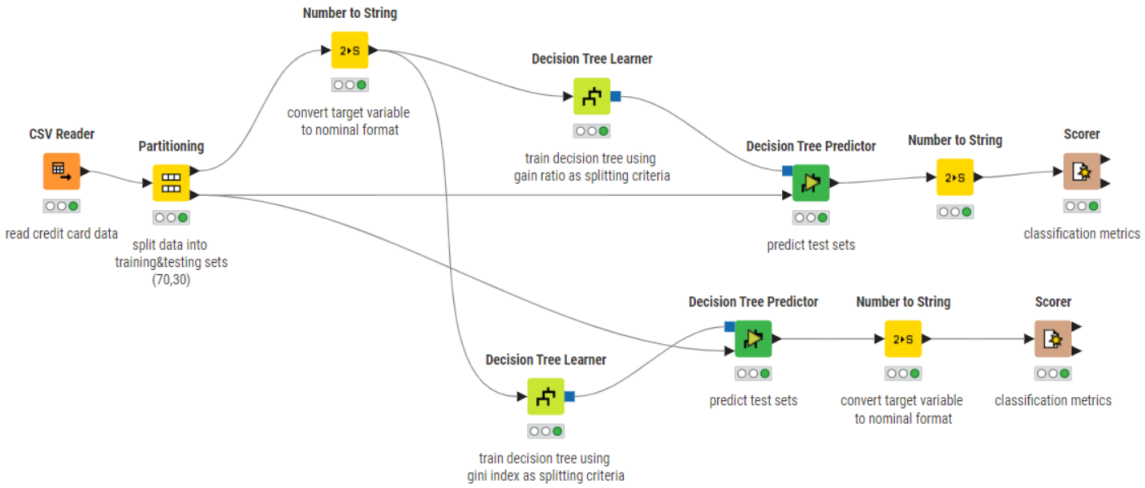
نتیجه بدست آمده از اجرای Scorer:



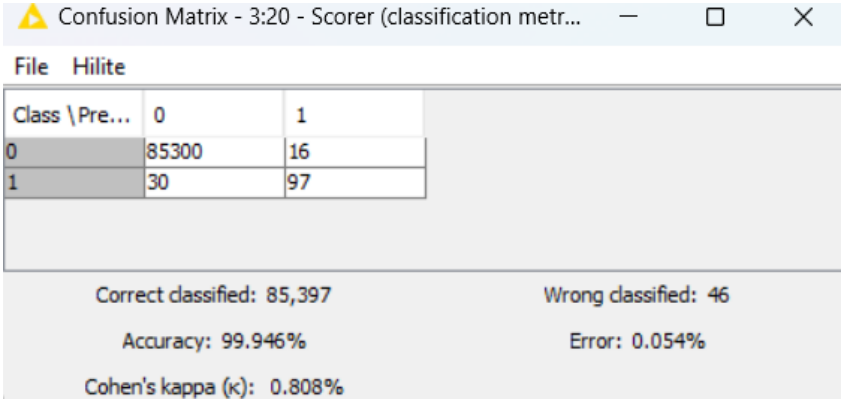
3. نحوه بهینه‌سازي تقسيم بندي داده ها (Criterion Splitting) در مدل هاي درخت تصميم در KNIME را بررسي كنيد و بهترين معيار تقسيم (... ، Entrop ،Gini) را تعيين كنيد.

برای اینکه بتوانیم از معیار های مختلف برای ایجاد درخت تصمیم استفاده کنیم می‌توانیم از داخل نود decision tree learner از قسمت Quality measure می‌توان بین gini index یا gain ratio یکی را انتخاب کرد. که در تصویر از زیر از هر دو معیار استفاده شده است.

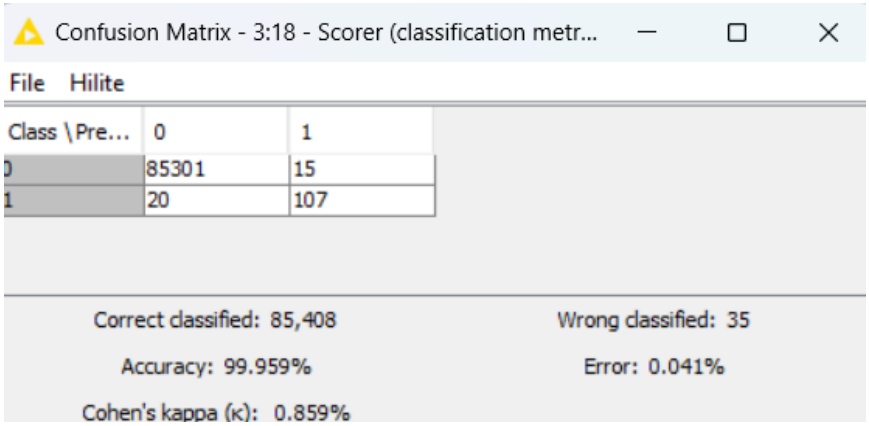




اگر خروجی هر دو مدل را به Scorer بدهیم نتیجه زیر حاصل میشود:



درخت تصمیم با معیار gini index



درخت تصمیم با معیار gain ratio

باتوجه به اینکه مقدار accuracy برای هر دو درخت به شدت نزدیک به هم بوده اما اگر بخواهیم دقیق تر بررسی کنیم، مقدار error و نمونه هایی که اشتباه دسته بندی شده اند برای درختی که معیار splitting آن ، gain ratio بوده از درختی با معیار gini index کمتر می‌باشد.

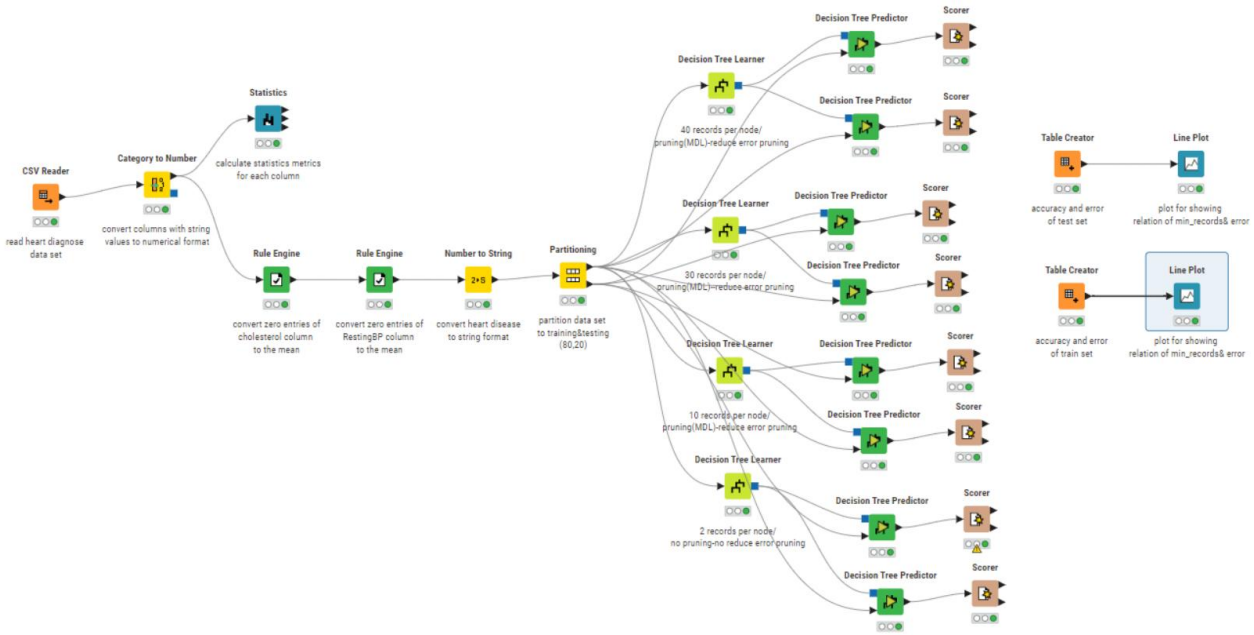
از لحاظ اینکه بین معیار های مختف کدام یک بهتر هست، به داده های ما گاها بستگی دارد به عنوان مثال: برای داده های متعاد(داده هایی که کلاس های تقریبا برابری دارند) gini index می‌تواند مناسب باشد ، چرا که نسبت به معیار هایی همچون entropy , gain ratio سریع تر عمل می‌کند. اما اگر کلاس ها نامتوان باشند بهتر هست که از entropy , gain ratio استفاده شود. معیار misclassification error هم معیار خوبی برای split کردن نیست چرا که صرفا می‌گوید اگر از اکثریت کلاس استفاده کنیم چند درصد خطا خواهیم داشت پس نسبت به تفاوت های کوچک در توزیع کلاس ها حساس نیست و بیشتر برای هرس کردن درخت مورد استفاده قرار می‌گیرد.

استفاده از gain ratio میتواند در برخی موارد حتی بهتر از entropy عمل کند چرا که صرف استفاده از entropy منجر به overfit شدن درخت خواهد شد که gain ratio می‌تواند از این عمل جلوگیری کند.

کلاس‌بندي مجموعه داده بیماران قلبی

1. با کمک knime نشان دهید که چگونه افزایش پیچیدگی درخت تصمیم باعث بروز پدیده‌ی overfitting می‌شود. این موضوع را بااستفاده از نمودار و داده های نمونه تفسیر کنید.

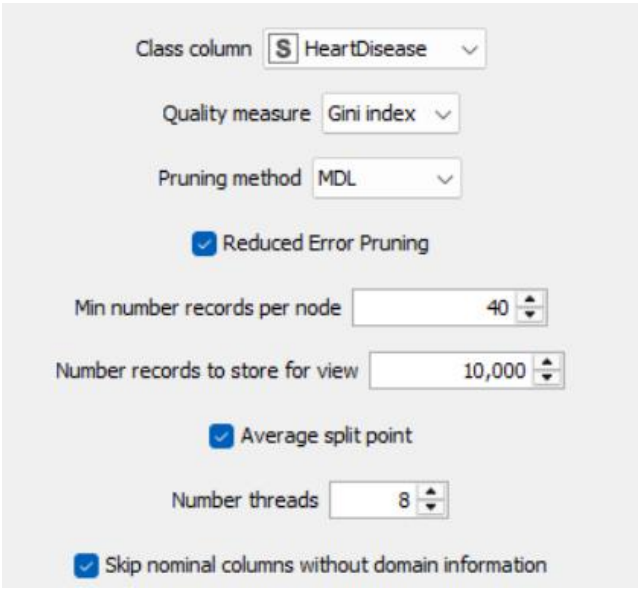
برای این کار با توجه به شکل زیر،ابتدا با استفاده از نودcsv reader داده ها را میخوانیم و سپس از نودcategory to number، attribute هایی که به صورت string هستند را به فرمت عددی تبدیل میکنیم مثلا برای جنسیت به جای male و female از 0و1 استفاده خواهیم کرد. سپس با توجه به اطلاعاتی که داخل فایل read me دیتا است، باید ستون هایی که مقادیر نامعتبری دارند(مانند attributeهای RestingBP, Cholesterol) که به صورت مقادیر صفر ظاهر شده اند را با میانگین رو آن ستون جایگزین کنیم. پس لازم است تا میانگین ستون های RestingBP, Cholesterol با استفاده از نود statistics گرفته شود و مقدار به دست امده با مقدار صفر این دو attribute جایگزین شود.
سپس داده های مربوط به کلاس را به فرمت string تبدیل میکنیم(چون نود decision tree داده های کلاس را به صورت string میپذیرد) و بعد از آن باید داده ها را به دو بخش training , testing به نسبت 80 به 20 تقسیم کنیم.
برای اینکه بتوانیم تاثیر پیچیدگی مدل بر رویoverfitting را متوجه شویم از 4 درخت با پیچیدگی های مختلف استفاده کرده ایم.



نحوه درست کردن درخت با پیچیدگی های مختلف از طریق تنظیم کردن پارامتر های داخل نود decision tree learner محقق می‌شود.

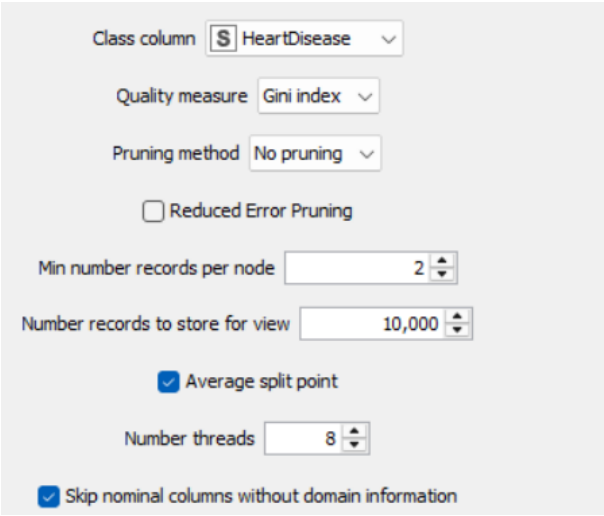
هرچقد که پارامتر min number records per node را بیشتر در نظر بگیریم مدل ساده تر بود و هرچقدر این مقدار کمتر شود مثلا برابر با 2 باشد عمق در خت و میزان پیچیدگی آن بیشتر خواهد شد. همچنین هرس کردن درخت چه قبل از ساخت درخت(پارامتر pruning method=MDL) و چه بعد از ساخت درخت(فعال کردن Reduced Error Pruning) می‌تواند از overfit شدن نیز جلوگیری کند.

در شکل زیر مثال ساخت درخت با 40 رکورد به ازای هر نود و فعال کردن هرس درخت قبل و بعد از ساختن درخت اورده شده است.



دو درخت بعدی را هم به همین منوال ساخته ایم با این تفاوت که برای یکی از آنها تعداد 30 رکورد به ازای هر نود در نظر گرفته ایم و برای دیگری 10رکورد به ازای هر نود.

در آخر برای پیچیده ترین درخت به این شکل عمل کردیم که تعداد 2 رکورد برای هر نود در نظر گرفتیم و عمل هرس کردن چه قبل و چه بعد از ساخته شدن مدل را غیر فعال کردیم.



سپس برای بدست اوردن میزان دقت و خطا روی داده های test , train برای هر درخت، مدل را یکبار روی داده های تست و بار دیگر روی خود داده ها train پیش بینی میکنیم(از نودdecision tree predictorاستفاده میکنیم) و بعد از آن خروجی هر predictor را به scorer میدهیم تا میزان دقت و خطا را بدست آوریم که نهایتا دو جدول زیر حاصل می‌شد.(داده های بدست امده از scorer ها به صورت دستی در دو جدول وارد شده اند، یک جدول برای داده های تست و دیگری برای داده های train)

1: Manually created table | Flow Variables

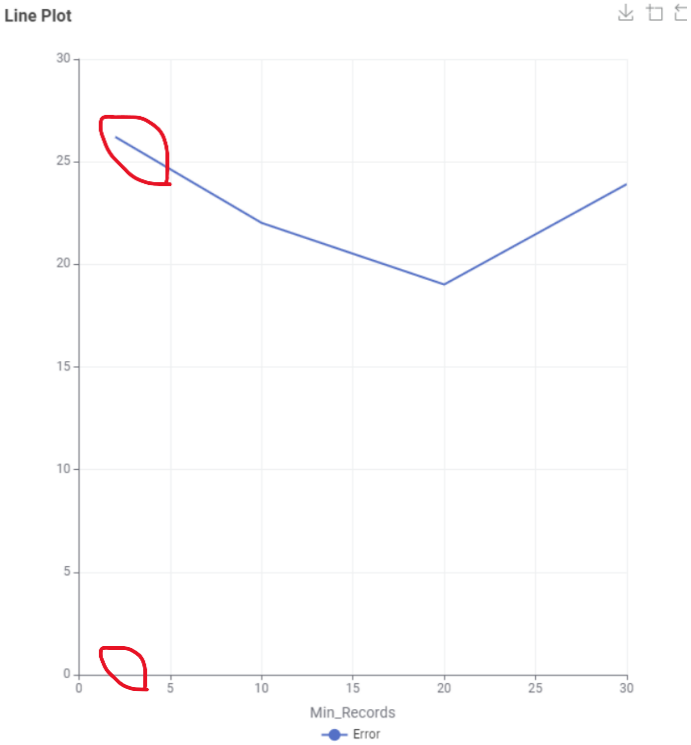
Rows: 8 | Columns: 4

Table | Statistics

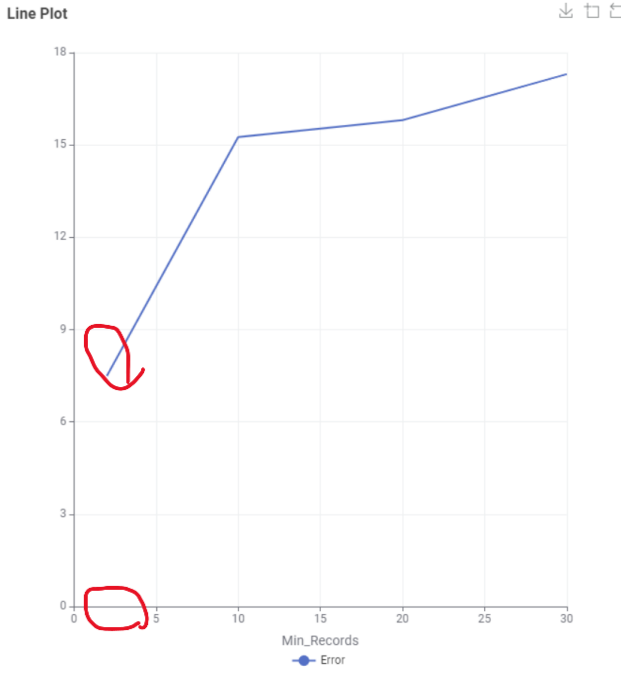
#	RowID	Dataset (String)	Min_Records (Number (integer))	Accuracy (Number (double))	Error (Number (double))
1	Row0	test	30	76.087	23.913
2	Row1	test	20	80.978	19.022
3	Row2	test	10	77.174	22.022
4	Row3	test	2	73.793	26.207
5	Row4	train	30	82.698	17.302
6	Row5	train	20	84.196	15.804
7	Row6	train	10	84.741	15.259
8	Row7	train	2	92.507	7.493

همان طور که از شکل پیدا است خطا برای داده train در ابتدا زیاد اما به مراتب کم میشود اما خطا برای داده تست در ابتدا زیاد بعد از آن کم میشود اما وقتی به مدل پیچیده تر میرسد ناگهان دوباره بسیار زیاد میشود.

شکل زیر مربوط به ارتباط بین میزان error و کمترین رکورد برای هر نود می‌باشد که از شکل مشخص است که اگر از سمت راست به چپ به مدل نگاه کنیم میبینیم که خطای مدل روی داده های تست کم می‌شود اما وقتی پیچیدگی مدل زیاد شده دوباره خطای مدل روی داده هاس تست پیک زده است.

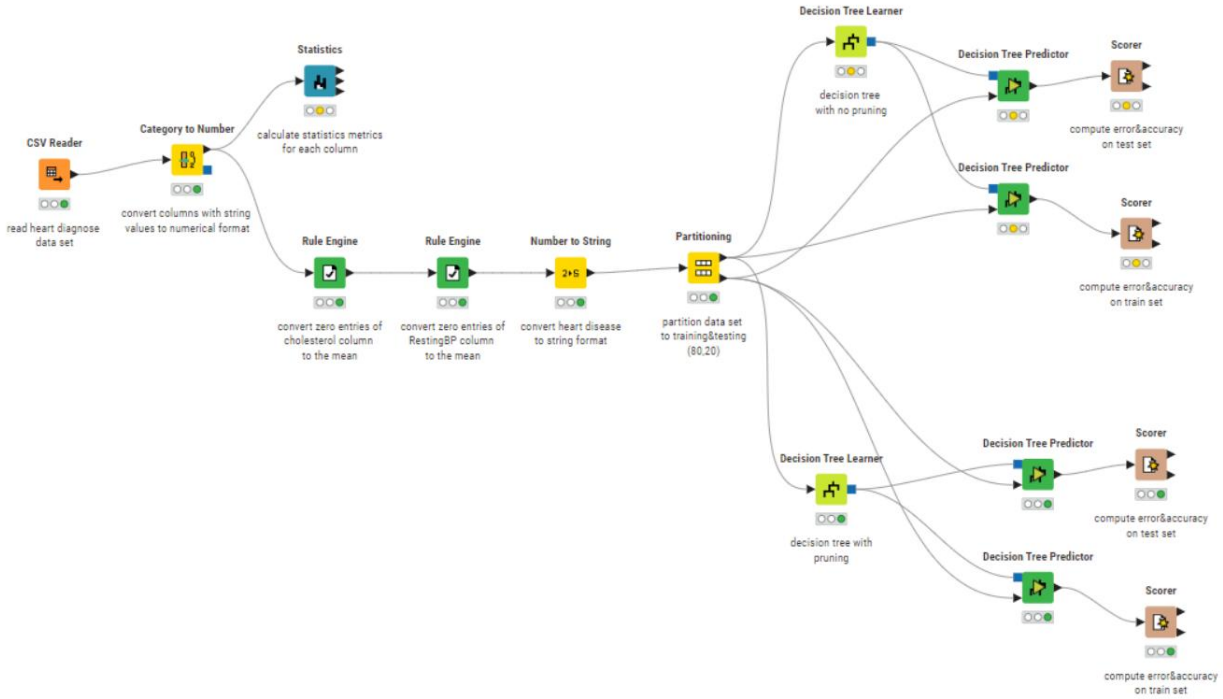


شکل زیر مربوط به ارتباط error با کمترین رکورد برای هر نود می‌باشد، همانطور که از شکل پیدا است خطا به مرور از سمت راست به چپ کاهش یافته(یعنی با افزایش پیچیدگی مدل) خطا روی داده هایtrain کاسته می‌شود می‌شود، تا اینکه در همان نقطه ای که خطای تست زیاد شده خطای train به شدت کاهش یافته.

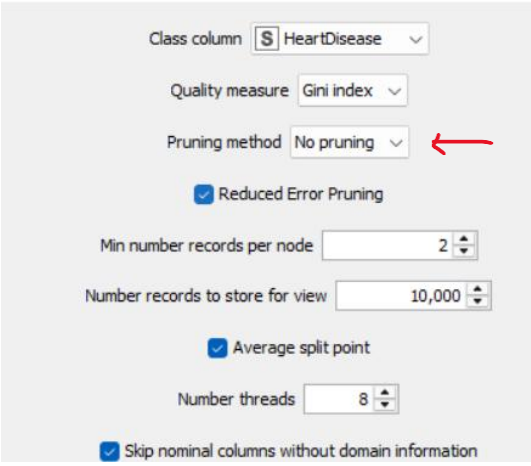


2. یک مثال عملی در Knime ارائه دهید تا تاثیر هرس کردن درخت تصمیم (pruning) را بر کاهش overfitting به وضوح نشان دهد.

با توجه به تصویر زیر عملیات خواندن و تقسیم داده همانند مثال قبل است ، برای نشان دادن اینکه pruning میتواند روی overfitting تاثیر بگذارد یا نه از دو درخت استفاده کرده ایم.



یکی از آنها بدون هرس کردن:



که میزان error , accuracy روی داده های train و تست به شرح زیر است:

HeartDisea...	0	1
0	54	20
1	19	55

Correct classified: 109 Wrong classified: 39

Accuracy: 73.649% Error: 26.351%

Cohen's kappa (κ): 0.473%

ارزیابی مدل روی داده های تست

File Hilite

HeartDisea...	0	1
0	310	19
1	36	369

Correct classified: 679 Wrong classified: 55

Accuracy: 92.507% Error: 7.493%

Cohen's kappa (κ): 0.849%

ارزیابی مدل رو داده های آموزشی

همانطور که از شکل های بالا پیداست خطای آموزش برابر با 7.493 و خطای تست برابر با 26.351می باشد.

اما اگر در مرحله بعد هرس کردن را هم به درخت اضافه کنیم نتیجه به شکل زیر خواهد شد:

File Hilite

HeartDisea...	0	1
0	56	25
1	19	84

Correct classified: 140 Wrong classified: 44

Accuracy: 76.087% Error: 23.913%

Cohen's kappa (κ): 0.511%

ارزیابی مدل روی داده های تست

File Hilite

HeartDisea...	0	1
0	261	68
1	59	346

Correct classified: 607 Wrong classified: 127

Accuracy: 82.698% Error: 17.302%

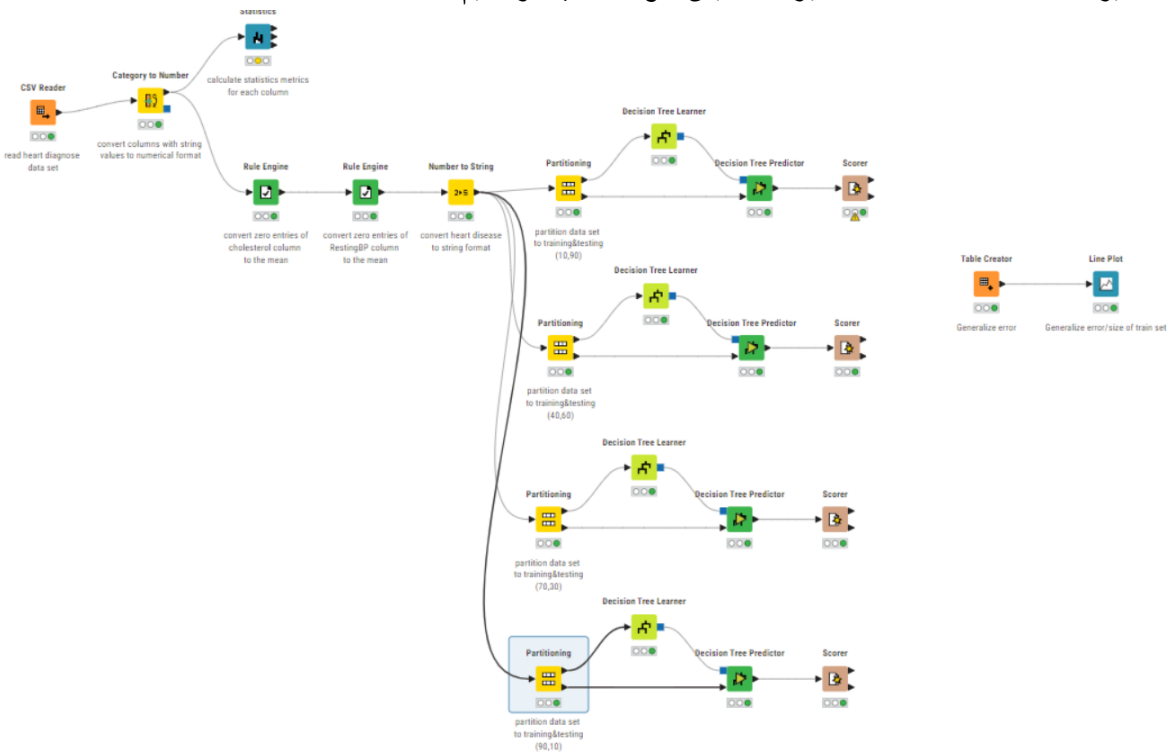
Cohen's kappa (κ): 0.649%

ارزیابی مدل رو داده های آموزشی

همان طور که از نتایج مشخص می‌باشد،pruning باعث شده است که خطای تست کم شود و به مقدار 23.913 برسیم و از طرفی خطای آموزش هم زیاد شده (17.302) و به خطای تست نزدیک شده که نشان میدهد از overfit شدن مدل کاسته شده است.

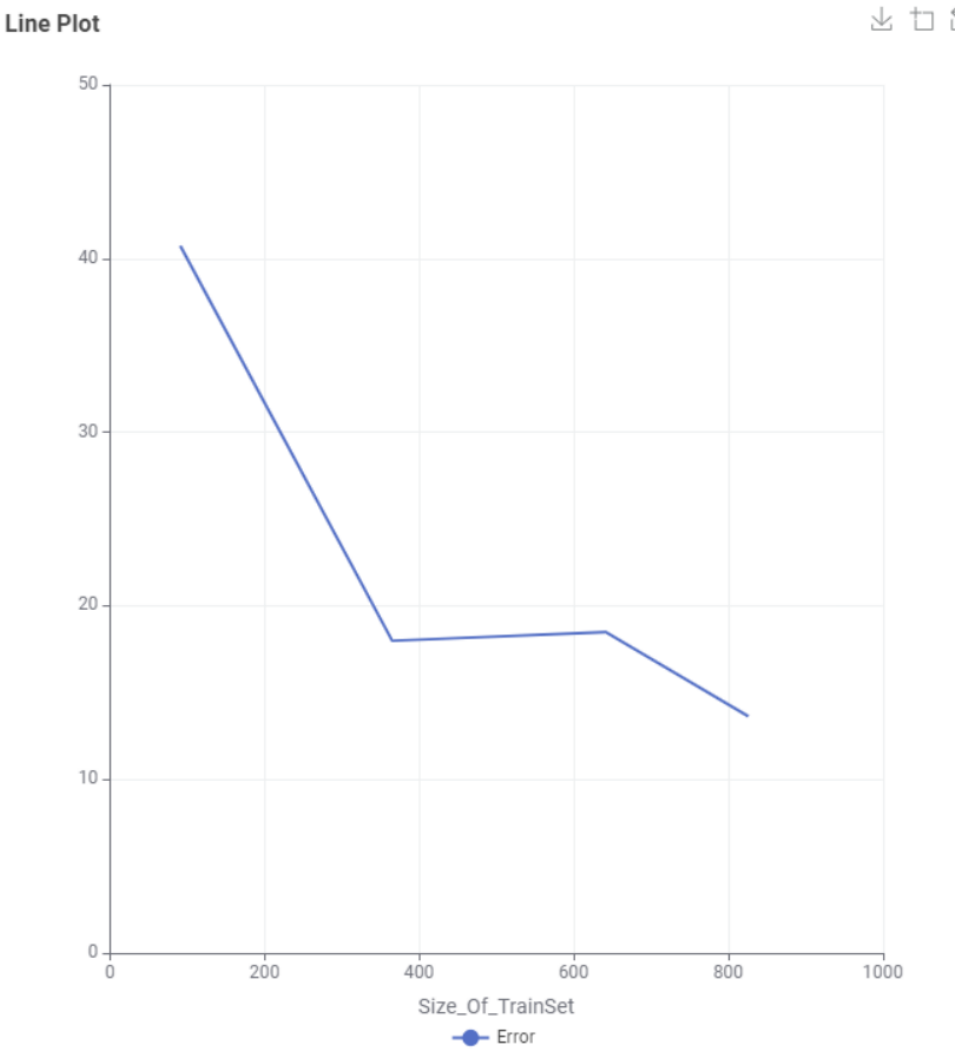
3. در knime تاثیر اندازه ی داده های آموزشی را بر خطای تعمیم(generalization error) بررسی کرده و نتیجه را با رسم نمودار تفسیر کنید.

برای این کار لازم است تا از partition بندی متفاوت استفاده کنیم برای ازمودن اینکه سایز داده های آموزشی می‌تواند تاثیر به سزایی در خطایgeneralize بگذارد از 4 partition بندی متفاوت استفاده کرده ایم، که به ترتیب سایز ها 91و365و642و826 را برای آزمایش خود انتخاب کرده ایم.



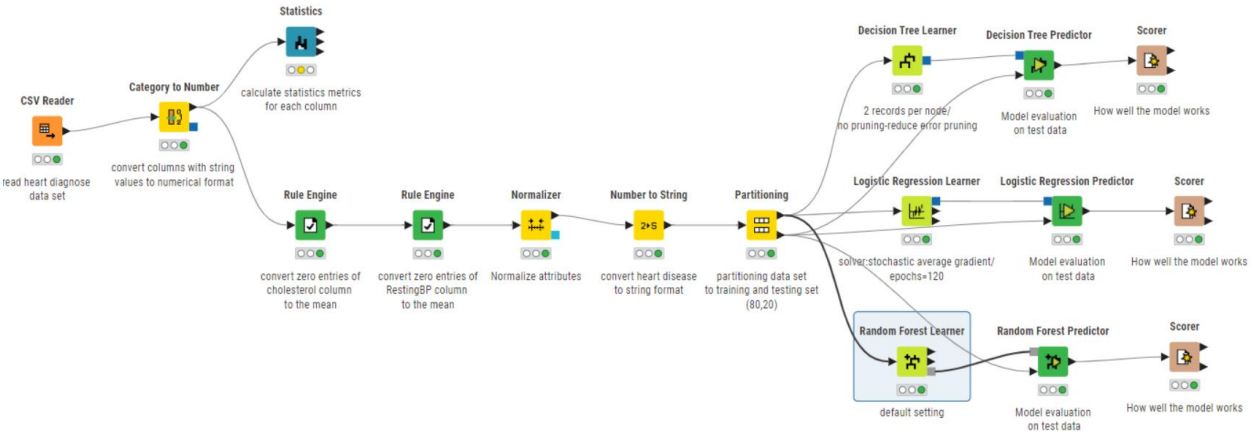
که نتیجه generalize error برای هر کدام از این partitioning هابه شکل زیر است، که میبینم اگر داده های آموزشی زیادی را به مدل نشان بدهیم خطای generalize میتواند کاهش پیدا کند، که افزایش داده های آموزشی خودش یک روش برای کاهش underfitting و در برخی موارد حتی overfitting میباشد.

Size_Of_TrainSet *Number (integer)*	Error *Number (double)*
91	40.727
365	17.986
642	18.478
826	13.643



4. چگونه می‌توانید در knime با ایجاد یک مجموعه اعتبار سنجیvalidation set بهترین مدل را انتخاب کنید؟ مراحل را به صورت عملی و با استفاده از داده های نمونه نشان دهید.

برای اینکار پس از خواندن داده ها و سپس انجام پیش پردازش های لازم از قبیل جایگزینی داده های نامعتبر با مقادیری همچون میانگین و نرمال کردن ویژگی ها میتوان داده ها را به دو قسمت اموزش و اعتبار سنجی تقسیم کرد برای اینکار 20 درصد از داده ها را برای مجموعه اعتبار سنجی از مجموعه کل داده ها جدا کردیم، این داده ها به مدل نشان داده نخواهد شد تا بتوانیم عملکرد مدل را روی داده هایی که از قبل ندیده ایم ارزیابی کنیم.
برای ارزیابی مدل از سه مدل مختلف از جمع درخت تصمیم ، logistic regression و random forest استفاده کرده ایم.
بعد از اینکه داده های اموزشی را بر هر کدام از این مدل ها دادیم باید از نودpredictor هر کدام از این مدل ها استفاده کنیم تا بتوانیم مدل را روی داده های اعتبار سنجی که مدل پیش از ان، آنها را ندیده ازمایش کنیم. سپس خروجی predictor ها را به نودscorer میدهیم تا بتوانیم معیار هایی همچون ..accuracy, recall, را بدست آوریم و بنا به کاربرد هر کدام که معیار بالاتری برای ما دارند را انتخاب کنیم.
به عنوان مثال در برخی موارد بهتر هست که معیار انتخاب ما بر اساس recall باشد یا در برخی موارد accuracy بالاتر میتواند مدل بهتری برای ما باشد.
مراحل گفته شده در بالا در زیر اورده شده است:

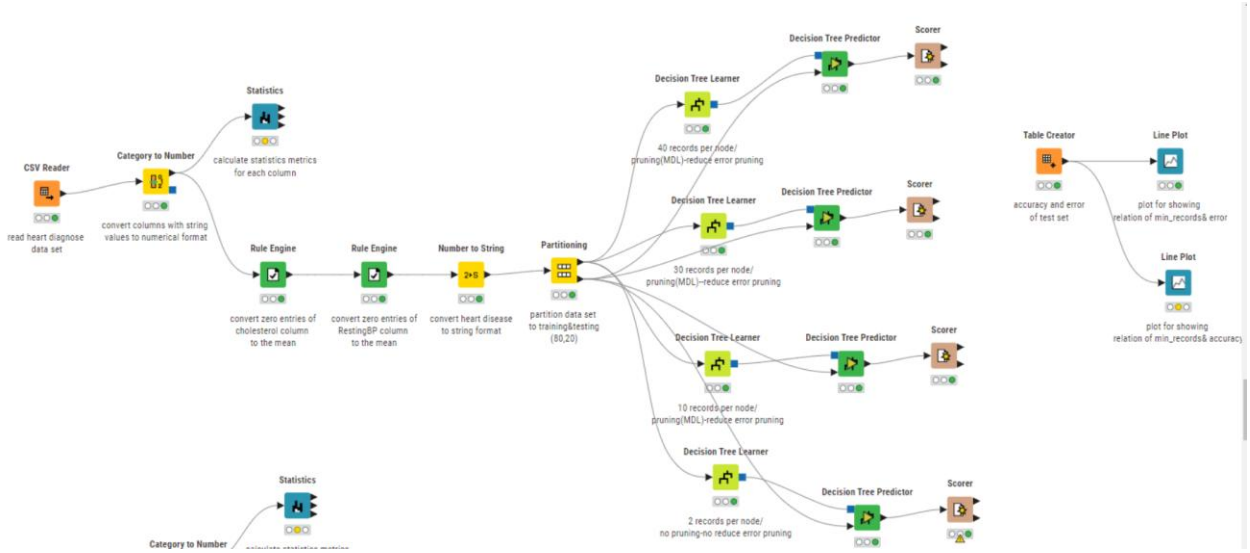


که خروجی scorer برای هر کدام از مدل ها در زیر اورده شده است، که اگر معیار ما بیشتر بودن accuracy هست میتوان گفت مه مدل random forest بهتر از دو مدل دیگر می باشد.

model *String*	accuracy *Number (double)*	error *Number (double)*
Decision Tree	81.522	18.478
Logistic Regressiion	87.5	12.5
Random Foret	88.587	11.413

5. با استفاده از روش های ارزیابی مدل در knime مانند holdout و cross-validation بهترین مدل درخت تصمیم را از نظر پیچیدگی و دقت انتخاب کرده و دلایل خود را شرح دهید.

برای انتخاب مدل میتوان از هر دو روش holdout و cross validation استفاده کرد . برای روش holdout میتوان درخت ساخته شده را روی دیتا test امتحان کرد و score ان را بدست اورد و به همین شکل برای دیگر درخت ها با پیچیدگی های مختلف می‌توان همین کار را کرد.

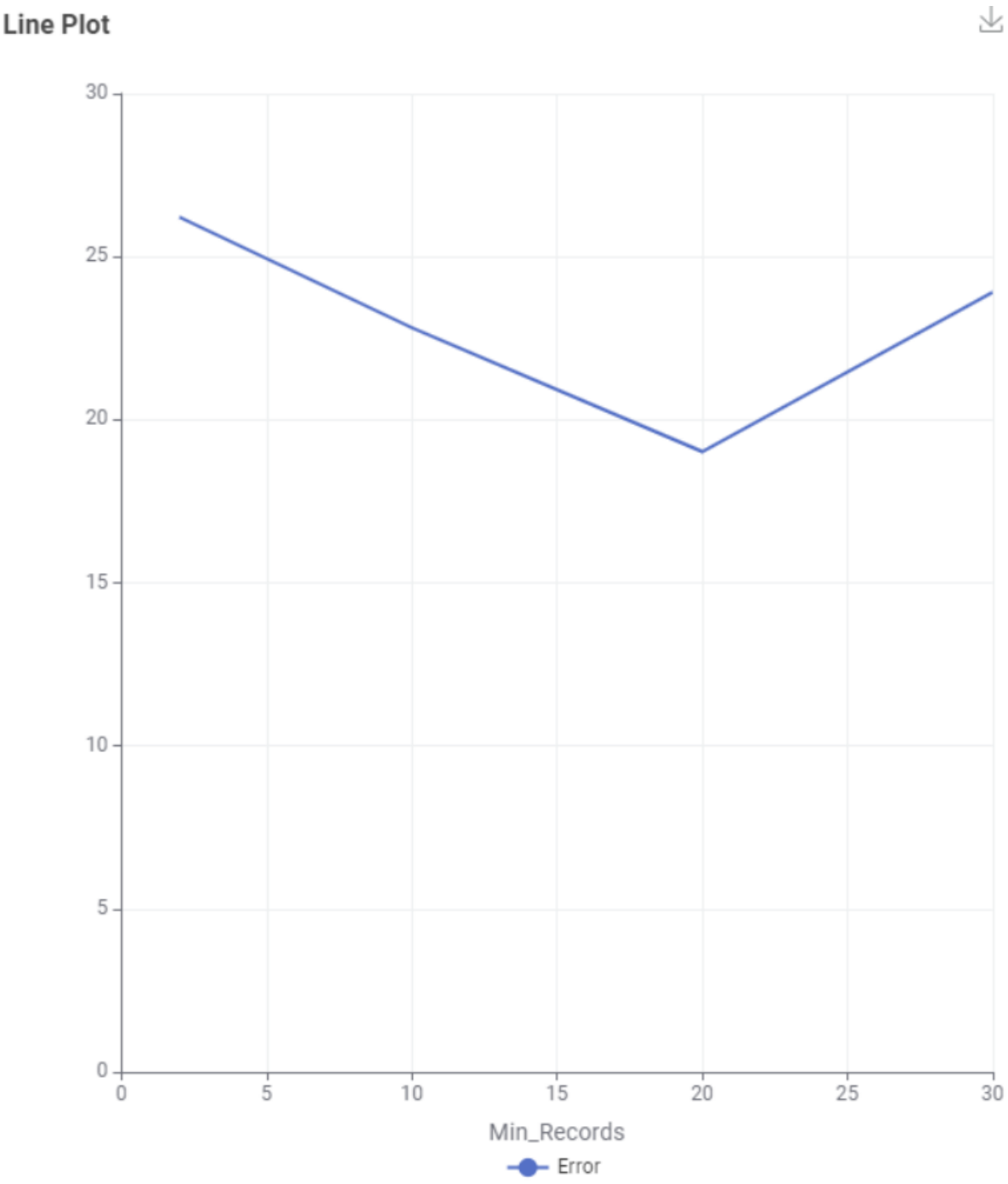


بعد از آن باید نتیجه همهscore ها را ارزیابی کرد که هم میتوان جدولی از آن کشید و هم میتوان به صورت نمودار خطی آن را نشان داد.

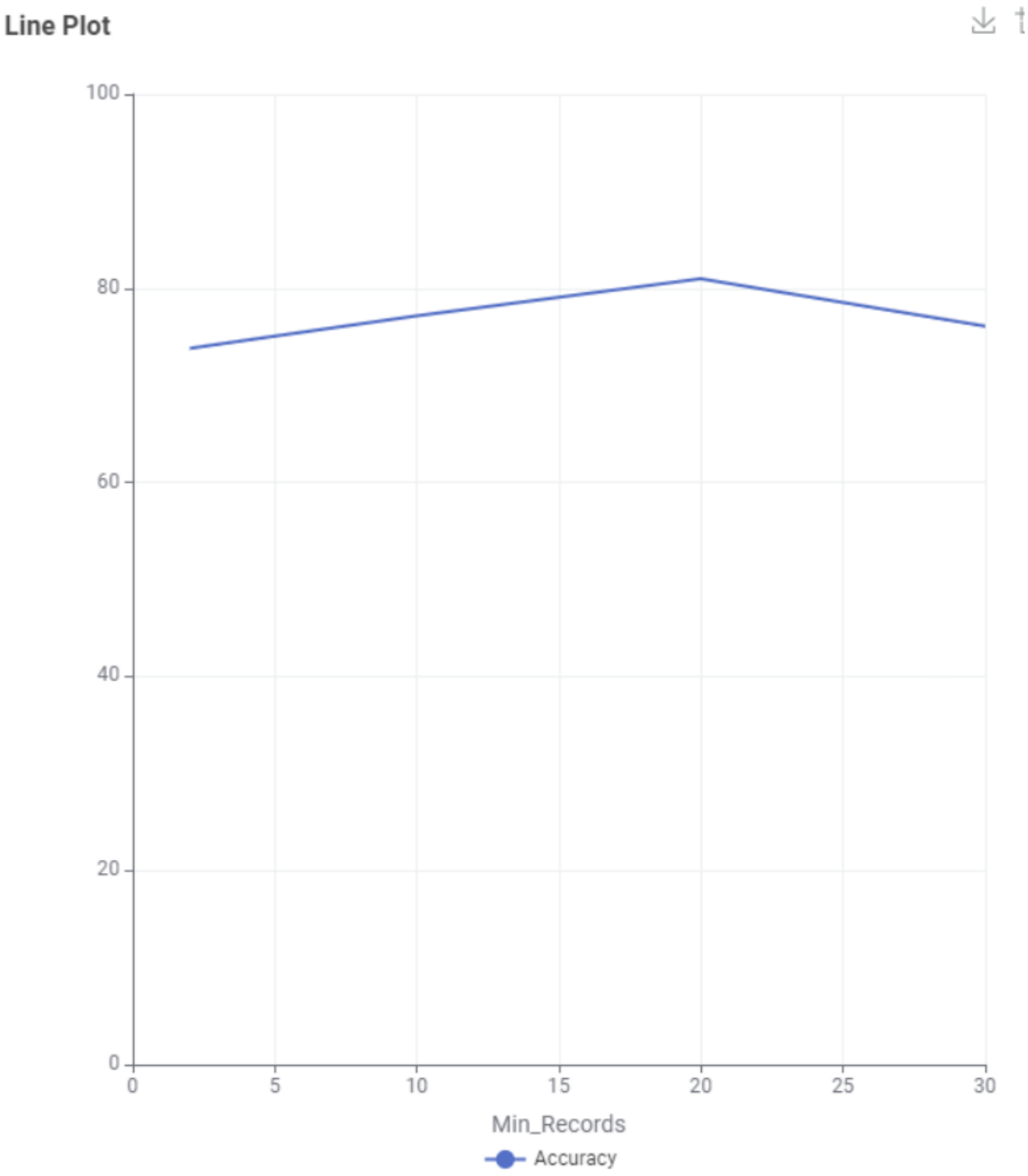
شکل زیر نتیجه تمامی score ها برای تک تک درخت ها است:

Dataset *String*	Min_Records *Number (integer)*	Accuracy *Number (double)*	Error *Number (double)*
test	30	76.087	23.913
test	20	80.978	19.022
test	10	77.174	22.826
test	2	73.793	26.207

که میتوان از همین جدول برای رسم نموداری بر اساس تعداد رکورد برای هر نود و خطای تعمیم ، دقت تعمیم استفاده کرد.



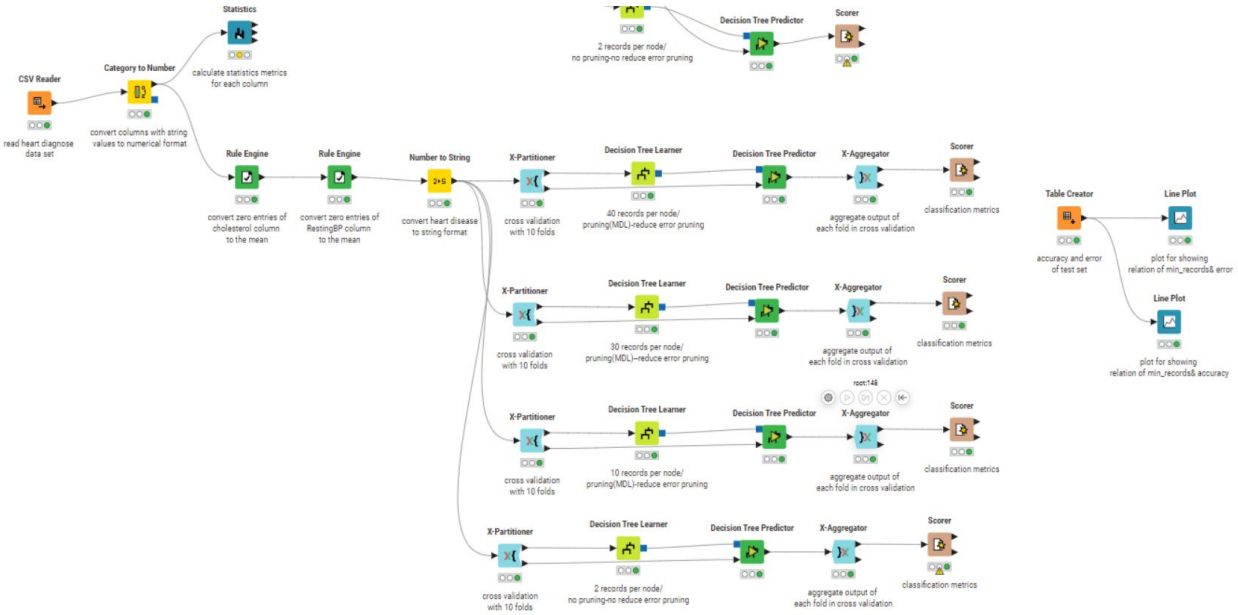
ارور بر اساس تعداد رکورد برای هر نود



دقت بر اساس تعداد رکورد برای هر نود

که اگر براساس تکنیک holdout برویم نشان میدهد که بهترین پیچیدگی ممکن برای درخت تعداد 20 رکورد برای هر نود می باشد و کمتر از این مقدار با پدیده overfittingمواجه میشویم و برای مقادیر بیشتر با پدیدهunderfitting.

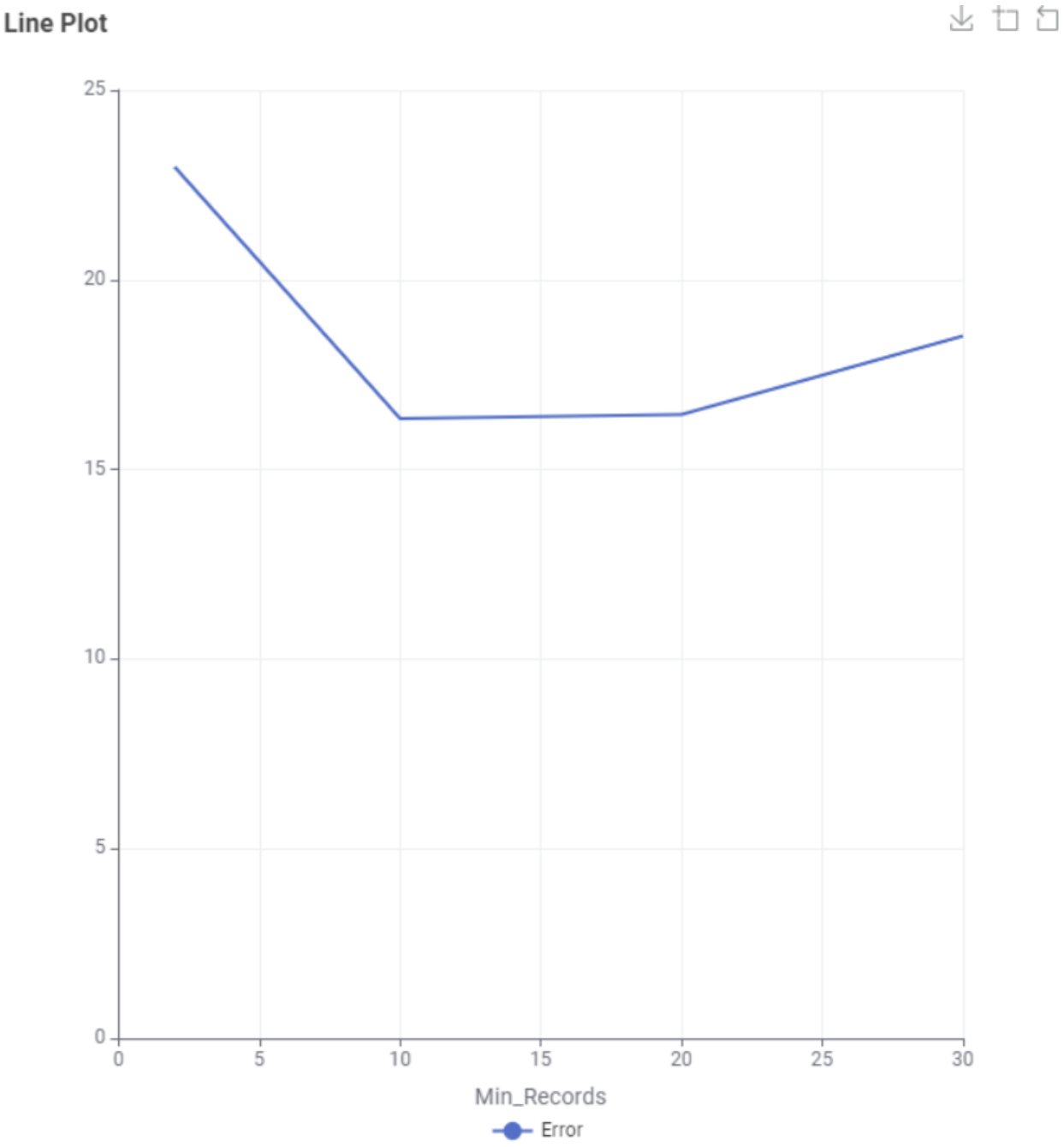
اما اگر بخواهیم که مدل را بر اساس cross-validation تست کنیم باز هم میتوان به ازای هر درخت از cross validation با تعداد 10 فولد استفاده کرد و بعد از هر کدام یک score گرفت و مقادیر بدست امده از scoreها را باهم مقایسه و ارزیابی کرد.



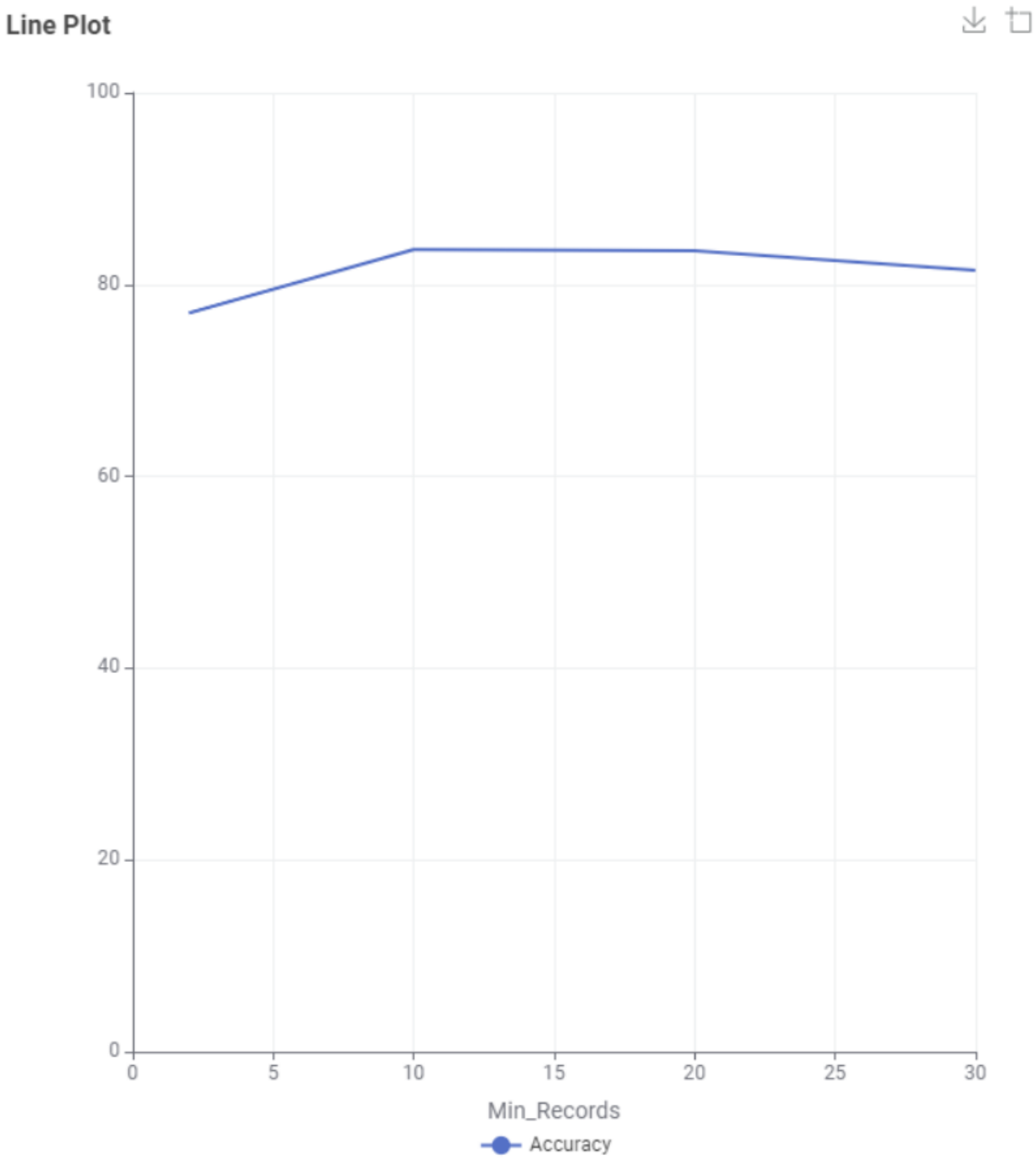
سپس میتوان از نتایج بدست امده از scoreها استفاده کرد برای رسم نمودار و کشیدن یک جدول.

Dataset *String*	Min_Records *Number (integer)*	Accuracy *Number (double)*	Error *Number (double)*
test	30	81.481	18.519
test	20	83.551	16.449
test	10	83.66	16.34
test	2	77.025	22.975

با استفاده از نتایج بالا نمودار خطی رسم میکنیم.



ارور بر اساس تعداد رکورد برای هر نود



دقت بر اساس تعداد رکورد برای هر نود

با توجه به نتایج بدست امده در بالا که حاصل استفاده از تکنیک cross validation هست میتوان نتیجه گرفت که بهترین حالت برابر با 10 و 20 رکورد می باشد .اما چون اختلاف بین دقت و خطای تعمیم برای این دو مقدار در حالتی که از cross validation استفاده کرده ایم کم می باشد میتوان از روش holdout استفاده کرد ، که دیدم در این حالت بهترین

مدل ، درختی است که تعداد رکورد هر نود ان برابر با 20 باشد در این حالت مدل چیزی بین overfitting و underfitting هست و میتواند ایده آل عمل کند.