2021

Assignment 3

MACHINE LEARNING CAP 6673

ASAM MAHMOOD

Introduction

Classification is a supervised learning concept in machine learning[1]. It breaks up instances specified in sets of data down to groups or classes. In this project we have a class(fp,nfp) we predict using a cost sensitive classifier combined with meta learners and learners. We are using 10-fold cross fold validation on the fit data set. While doing this we also iterate through different cost values which help us determine the optimal cost ratio which is the goal of this experiment.

Input

Our data FIT and Test data sets were given to us as a requirement to use. There are 9 attributes in and 188 instances in FIT data set. The Test data set has 94 instances and 9 attributes. Class attribute in each set has a requirement when attribute is less than 2 faults are considered nfp, and modules with 2 or more faults are considered fp .It was used in our previous assignment 1a where we engineered the input. I added command with configuration for each section after results I iterated through values for cost matrix which can be seen on table. I used command-line to get output metrics via terminal then I used python to execute command-line using python sys library then parse output metrics that were relevant to excel sheets for further analysis.

Metrics in tables

In Fit tables I included Cost matrix where I set values for cost. I included the Confusion matrix which helped determine the type I type II errors. I also included precision and recall which tell me the number of positives instances in a model or classifiers has correctly predicted and the accuracy of the positives out of total positives[1] I also included ROC area which in our textbook author discusses how its used to help look at area under curve[1] and I also included tp and fp rates value for all.

Bagging J48 10 iteration Fit

COST MATRIX	CONFUSION MATRIX	TP RATE	FP RATE	PRECISION	RECALL	ROC AREA	PRC AREA	TYPE I ERROR	TYPE II ERROR
.5	35 18	0.66	0.059	0.814	0.66	0.908	0.747	0.059259	0.339623
	8 127	0.941	0.34	0.876	0.941	0.908	0.943		
		0.862	0.261	0.858	0.862	0.908	0.887		
.75	39 14	0.736	0.074	0.796	0.736	0.92	0.783	0.074074	0.264151
	10 125	0.926	0.264	0.899	0.926	0.92	0.961		
		0.872	0.211	0.87	0.872	0.92	0.911		
1	42 11	0.792	0.081	0.792	0.792	0.933	0.81	0.081481	0.207547
	11 124	0.919	0.208	0.919	0.919	0.933	0.967		
		0.883	0.172	0.883	0.883	0.933	0.923		
2	48 5	0.906	0.156	0.696	0.906	0.922	0.781	0.155556	0.09434
	21 114	0.844	0.094	0.958	0.844	0.922	0.955		
		0.862	0.112	0.884	0.862	0.922	0.906		
2.5	47 6	0.887	0.156	0.691	0.887	0.916	0.76	0.155556	0.113208
	21 114	0.844	0.113	0.95	0.844	0.916	0.954		
		0.856	0.125	0.877	0.856	0.916	0.899		
3	47 6	0.887	0.141	0.712	0.887	0.92	0.756	0.140741	0.113208
	19 116	0.859	0.113	0.951	0.859	0.92	0.955		
		0.867	0.121	0.884	0.867	0.92	0.899		
4	49 4	0.925	0.156	0.7	0.925	0.923	0.774	0.155556	0.075472
	21 114	0.844	0.075	0.966	0.844	0.923	0.956		
		0.867	0.098	0.891	0.867	0.923	0.905		
5	49 4	0.925	0.17	0.681	0.925	0.918	0.778	0.17037	0.075472
	23 112	0.83	0.075	0.966	0.83	0.918	0.955		
		0.856	0.102	0.885	0.856	0.918	0.905		
6	47 6	0.887	0.178	0.662	0.887	0.919	0.764	0.177778	0.113208
	24 111	0.822	0.113	0.949	0.822	0.919	0.954		
		0.84	0.131	0.868	0.84	0.919	0.901		

Bagging J48 10 iteration Test

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION	19 8	17 10	22 5	24 3	23 4	25 2	24 3	25 2	24 3
MATRIX	5 62	7 60	8 59	14 53	9 58	11 56	13 54	14 53	15 52
TYPE I ERROR	0.074627	0.104478	0.119403	0.208955	0.134328	0.164179	0.19403	0.208955	0.223881
TYPE II ERROR	0.296296	0.37037	0.185185	0.111111	0.148148	0.074074	0.111111	0.074074	0.111111

Bagging J48 10 iteration results

In Table Bagging J48 10 iteration fit I calculated the type errors for bot Type I and Type II. When comparing results between the models when cost matrix to 2.5 it was the most optimal for this table. The reason why it has the most optimal difference between the type errors[1] . Author discusses on page 172. It also has the mode optimal tp and fp rate. Which help me determine this. I also looked the recall which tells how many positive were positive and .88 was most optimal without sacrificing out type II and type I requirements. . As the requirement from instruction states TYPE II error is more serious than a type I error in our use case condition. So, the best would be lowest TYPE II with a balanced TYPE I which by diffing them can help you find the smallest difference as well.

command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

Bagging J48 25 iteration Fit

cost matrix	confusion matrix	TP Rate	FP Rate	Precision	Recall	ROC Area	PRC Area	Type I Error	Type II Error
.5	38 15 9 126	0.717 0.933 0.872	0.067 0.283 0.222	0.809 0.894 0.87	0.717 0.933 0.872	0.93 0.93 0.93	0.827 0.967 0.928	0.066667	0.283019
.75	42 11 9 126	0.792 0.933 0.894	0.067 0.208 0.168	0.824 0.92 0.893	0.792 0.933 0.894	0.932 0.932 0.932	0.809 0.959 0.917	0.066667	0.207547
1	43 10 11 124	0.811 0.919 0.888	0.081 0.189 0.158	0.796 0.925 0.889	0.811 0.919 0.888	0.936 0.936 0.936	0.813 0.961 0.919	0.081481	0.188679
2	48 5 20 115	0.906 0.852 0.867	0.148 0.094 0.11	0.706 0.958 0.887	0.906 0.852 0.867	0.931 0.931 0.931	0.785 0.959 0.91	0.148148	0.09434
2.5	48 5 19 116	0.906 0.859 0.872	0.141 0.094 0.107	0.716 0.959 0.89	0.906 0.859 0.872	0.933 0.933 0.933	0.81 0.96 0.918	0.140741	0.09434
3	50 3 21 114	0.943 0.844 0.872	0.156 0.057 0.084	0.704 0.974 0.898	0.943 0.844 0.872	0.93 0.93 0.93	0.778 0.959 0.908	0.155556	0.056604
4	50 3 21 114	0.943 0.844 0.872	0.156 0.057 0.084	0.704 0.974 0.898	0.943 0.844 0.872	0.929 0.929 0.929	0.777 0.959 0.907	0.155556	0.056604
5	50 3 25 110	0.943 0.815 0.851	0.185 0.057 0.093	0.667 0.973 0.887	0.943 0.815 0.851	0.927 0.927 0.927	0.785 0.957 0.909	0.185185	0.056604
6	50 3 25 110	0.943 0.815 0.851	0.185 0.057 0.093	0.667 0.973 0.887	0.943 0.815 0.851	0.924 0.924 0.924	0.775 0.956 0.905	0.185185	0.056604

Bagging J48 25 iteration Test

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION	16 11	17 10	21 6	24 3	23 4	25 2	24 3	25 2	25 2
MATRIX	5 62	6 61	6 61	11 56	10 57	12 55	12 55	12 55	16 51
TYPE I ERROR	0.074627	0.089552	0.089552	0.164179	0.149254	0.179104	0.179104	0.179104	0.238806
TYPE II ERROR	0.407407	0.37037	0.222222	0.111111	0.148148	0.074074	0.111111	0.074074	0.074074

Bagging J48 25 iteration results

In Table Bagging J48 25 iteration fit I calculated the type errors for both Type I and Type II. When comparing results between the models when cost matrix to 2.5 it was the most optimal for this table. The reason why it has the most optimal difference between the type errors[1] . Author discusses on page 172. It also has the mode optimal tp and fp rate. In This out put you can recall is most optimal at that point without risking type II error. It also has the optimal difference between the type I and Type II errors. I also wanted to look at test runs the optimal values were 2.5 as well. Also, the precision was one of the highest for fit without actually sacrificing room in type II. Precision and Recall also attributed to my decision making. Recall and Precision are both important factors to consider the author speaks about this on page 1. A precision at 1 is perfect result[1].

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 25 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

Bagging Decision stump 10 Fit

COST MATRIX	CONFUSION MATRIX	TP RATE	FP RATE	PRECISION	RECALL	ROC AREA	PRC AREA	TYPE I ERROR	TYPE II ERROR
.5	33 20	0.623	0.089	0.733	0.623	0.891	0.728	0.088889	0.377358
	12 123	0.911	0.377	0.86	0.911	0.891	0.944		
		0.83	0.296	0.824	0.83	0.891	0.883		
.75	39 14	0.736	0.096	0.75	0.736	0.878	0.69	0.096296	0.264151
	13 122	0.904	0.264	0.897	0.904	0.878	0.919		
		0.856	0.217	0.856	0.856	0.878	0.854		
1	45 8	0.849	0.126	0.726	0.849	0.892	0.683	0.125926	0.150943
	17 118	0.874	0.151	0.937	0.874	0.892	0.95		
		0.867	0.144	0.877	0.867	0.892	0.874		
2	49 4	0.925	0.2	0.645	0.925	0.873	0.635	0.2	0.075472
	27 108	0.8	0.075	0.964	0.8	0.873	0.935		
		0.835	0.111	0.874	0.835	0.873	0.851		
2.5	47 6	0.887	0.178	0.662	0.887	0.89	0.657	0.177778	0.113208
	24 111	0.822	0.113	0.949	0.822	0.89	0.946		
		0.84	0.131	0.868	0.84	0.89	0.864		
3	49 4	0.925	0.215	0.628	0.925	0.884	0.675	0.214815	0.075472
	29 106	0.785	0.075	0.964	0.785	0.884	0.938		
		0.824	0.115	0.869	0.824	0.884	0.864		
4	49 4	0.925	0.23	0.613	0.925	0.879	0.633	0.22963	0.075472
	31 104	0.77	0.075	0.963	0.77	0.879	0.942		
		0.814	0.119	0.864	0.814	0.879	0.854		
5	49 4	0.925	0.259	0.583	0.925	0.878	0.632	0.259259	0.075472
	35 100	0.741	0.075	0.962	0.741	0.878	0.941		
		0.793	0.127	0.855	0.793	0.878	0.854		
6	50 3	0.943	0.259	0.588	0.943	0.879	0.631	0.259259	0.056604
	35 100	0.741	0.057	0.971	0.741	0.879	0.942		
		0.798	0.114	0.863	0.798	0.879	0.854		

Bagging Decision stump 10 Test

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION MATRIX	13 14 2 65	24 3 7 60	23 4 8 59	25 2 15 52	25 2 20 47	25 2 13 54	25 2 16 51	25 2 20 47	25 2 20 47
TYPE I ERROR	0.029851	0.104478	0.119403	0.223881	0.298507	0.19403	0.238806	0.298507	0.298507
TYPE II ERROR	0.518519	0.111111	0.148148	0.074074	0.074074	0.074074	0.074074	0.074074	0.074074

Bagging decision stump 10 iteration results

In Table Bagging decision stump 10 iteration fit I calculated the type errors for both Type I and Type. In this experiment we used meta learner bagging with learner decision stump. I iterated the cost matrix to find the optimal cost ratio model. I use the performance metrics generated with some metrics I calculated to determine optimal values. Funny enough the optimal cost for this set was 2 as and on test 2 was the optimal. My factor was the difference between type I and type II were which was around .6 on FIT it also had a small type II against type I error. Which met our requirements. The recall was highest at 2 which mean how many of positive instance in the model or classifications was correctly predicted. The precision weighted average I thought was one of the best without sacrificing TYPE II values not being optimal enough. The decision stump learner performs worse than j48 because it classifies based off one attribute[1] .

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

Bagging Decision Stump 25 - FIT

COST MATRIX	CONFUSION MATRIX	TP RATE	FP RATE	PRECISION	RECALL	ROC AREA	PRC AREA	TYPE I ERROR	TYPE II ERROR
.5	32 21	0.604	0.089	0.727	0.604	0.9	0.73	0.088889	0.396226
	12 123	0.911	0.396	0.854	0.911	0.9	0.952		
		0.824	0.31	0.818	0.824	0.9	0.889		
.75	43 10	0.811	0.096	0.768	0.811	0.904	0.713	0.096296	0.188679
	13 122	0.904	0.189	0.924	0.904	0.904	0.954		
		0.878	0.163	0.88	0.878	0.904	0.886		
1	44 9	0.83	0.126	0.721	0.83	0.896	0.703	0.125926	0.169811
	17 118	0.874	0.17	0.929	0.874	0.896	0.95		
		0.862	0.157	0.871	0.862	0.896	0.881		
2	48 5	0.906	0.207	0.632	0.906	0.896	0.678	0.207407	0.09434
	28 107	0.793	0.094	0.955	0.793	0.896	0.948		
		0.824	0.126	0.864	0.824	0.896	0.871		
2.5	44 9	0.83	0.126	0.721	0.83	0.896	0.703	0.125926	0.169811
	17 118	0.874	0.17	0.929	0.874	0.896	0.95		
		0.862	0.157	0.871	0.862	0.896	0.881		
3	49 4	0.925	0.207	0.636	0.925	0.896	0.685	0.207407	0.075472
	28 107	0.793	0.075	0.964	0.793	0.896	0.947		
		0.83	0.113	0.872	0.83	0.896	0.873		
4	48 5	0.906	0.237	0.6	0.906	0.893	0.667	0.237037	0.09434
	32 103	0.763	0.094	0.954	0.763	0.893	0.946		
		0.803	0.135	0.854	0.803	0.893	0.868		
5	48 5	0.906	0.259	0.578	0.906	0.893	0.676	0.259259	0.09434
	35 100	0.741	0.094	0.952	0.741	0.893	0.946		
		0.787	0.141	0.847	0.787	0.893	0.87		
6	50 3	0.943	0.252	0.595	0.943	0.887	0.655	0.251852	0.056604
	34 101	0.748	0.057	0.971	0.748	0.887	0.944		
		0.803	0.112	0.865	0.803	0.887	0.863		

Bagging Decision Stump 25 Test

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION MATRIX	23 4 7 60	23 4 9 58	24 3 9 58	25 2 16 51	24 3 9 58	25 2 16 51	25 2 16 51	25 2 20 47	25 2 20 47
TYPE I ERROR	0.104478	0.134328	0.134328	0.238806	0.134328	0.238806	0.238806	0.298507	0.298507
TYPE II ERROR	0.148148	0.148148	0.111111	0.074074	0.111111	0.074074	0.074074	0.074074	0.074074

Bagging Decision Stump 25 Results

In Table Bagging decision stump 25 iteration fit I calculated the type errors for both Type I and Type. In this experiment we used meta learner bagging with learner decision stump. I iterated the cost matrix to find the optimal cost ratio model with number iterations at 25. I use the performance metrics TYPE I and TYPE II help determine optimal cost per requirements I also look at precision and recall [1] which are also key factors. I saw the divergence in the FIT test set between when the cost is set to 2. I deep dived more and between values 1 and 2 but I realized the table would get super long if go more smaller increments to show divergence. So the optimal in fit is at 2 and on test is 2.5. There is a inverse relationship pattern we continue to see as values increase. The optimal in 10 iteration FIT did better than this optimal selection.

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 25 -W weka.classifiers.trees.DecisionStump

ADA Boost J48 10 -FIT

cost matrix	confusion matrix	TP Rate	FP Rate	Precision	Recall	ROC Area	PRC Area	Type I Error	Type II Error
.5	41 12 11 124	0.774 0.919 0.878	0.081 0.226 0.186	0.788 0.912 0.877	0.774 0.919 0.878	0.92 0.92 0.92	0.819 0.956 0.917	0.081481	0.226415
.75	41 12 10 125	0.774 0.926 0.883	0.074 0.226 0.183	0.804 0.912 0.882	0.774 0.926 0.883	0.912 0.912 0.912	0.829 0.951 0.916	0.074074	0.226415
1	40 13 14 121	0.755 0.896 0.856	0.104 0.245 0.205	0.741 0.903 0.857	0.755 0.896 0.856	0.893 0.893 0.893	0.733 0.939 0.881	0.103704	0.245283
2	43 10 13 122	0.811 0.904 0.878	0.096 0.189 0.163	0.768 0.924 0.88	0.811 0.904 0.878	0.927 0.927 0.927	0.796 0.96 0.914	0.096296	0.188679
2.5	44 9 13 122	0.83 0.904 0.883	0.096 0.17 0.149	0.772 0.931 0.886	0.83 0.904 0.883	0.914 0.914 0.914	0.783 0.943 0.898	0.096296	0.169811
3	42 11 11 124	0.792 0.919 0.883	0.081 0.208 0.172	0.792 0.919 0.883	0.792 0.919 0.883	0.926 0.926 0.926	0.832 0.949 0.916	0.081481	0.207547
4	46 7 11 124	0.868 0.919 0.904	0.081 0.132 0.118	0.807 0.947 0.907	0.868 0.919 0.904	0.926 0.926 0.926	0.839 0.958 0.925	0.081481	0.132075
5	43 10 14 121	0.811 0.896 0.872	0.104 0.189 0.165	0.754 0.924 0.876	0.811 0.896 0.872	0.923 0.923 0.923	0.76 0.961 0.904	0.103704	0.188679
6	42 11 16 119	0.792 0.881 0.856	0.119 0.208 0.182	0.724 0.915 0.861	0.792 0.881 0.856	0.908 0.908 0.908	0.771 0.95 0.9	0.118519	0.207547

ADA Boost J48 10 TEST

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION MATRIX	16 11 4 63	22 5 5 62	16 11 6 61	14 13 4 63	17 10 4 63	17 10 3 64	20 7 6 61	20 7 4 63	22 5 6 61
TYPE I ERROR	0.059701	0.074627	0.089552	0.059701	0.059701	0.044776	0.089552	0.059701	0.089552
TYPE II ERROR	0.407407	0.185185	0.407407	0.481481	0.37037	0.37037	0.259259	0.259259	0.185185

ADA Boost J48 10 Results

In Table Bagging decision stump 10 iteration fit I calculated the type errors for both Type I and Type. In this experiment we used meta learner bagging with learner decision stump. I iterated the cost matrix to find the optimal cost ratio model with number iterations at 10. I tried high value, but I did not get them to diverge where type II is lower than type I the closest in the current FIT table is where type II is the lowest is at 4 for fit and 5 for test. The reason is because it has the lowest type 2 with optimal recall and precision values.

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

ADA Boost J48 25 - FIT

cost matrix	confusion matrix	TP Rate	FP Rate	Precision	Recall	ROC Area	PRC Area	Type I Error	Type II Error
.5	41 12 12 123	0.774 0.911 0.872	0.089 0.226 0.188	0.774 0.911 0.872	0.774 0.911 0.872	0.924 0.926 0.925	0.797 0.964 0.917	0.088889	0.226415
.75	42 11 11 124	0.792 0.919 0.883	0.081 0.208 0.172	0.792 0.919 0.883	0.792 0.919 0.883	0.931 0.927 0.928	0.807 0.96 0.917	0.081481	0.207547
1	43 10 10 125	0.811 0.926 0.894	0.074 0.189 0.156	0.811 0.926 0.894	0.811 0.926 0.894	0.915 0.923 0.921	0.789 0.959 0.911	0.074074	0.188679
2	43 10 8 127	0.811 0.941 0.904	0.059 0.189 0.152	0.843 0.927 0.903	0.811 0.941 0.904	0.934 0.928 0.929	0.814 0.96 0.919	0.059259	0.188679
2.5	45 8 16 119	0.849 0.881 0.872	0.119 0.151 0.142	0.738 0.937 0.881	0.849 0.881 0.872	0.923 0.918 0.92	0.793 0.957 0.911	0.118519	0.150943
3	44 9 11 124	0.83 0.919 0.894	0.081 0.17 0.145	0.8 0.932 0.895	0.83 0.919 0.894	0.933 0.932 0.932	0.816 0.962 0.921	0.081481	0.169811
4	44 9 13 122	0.83 0.904 0.883	0.096 0.17 0.149	0.772 0.931 0.886	0.83 0.904 0.883	0.932 0.941 0.938	0.816 0.973 0.929	0.096296	0.169811
5	47 6 10 125	0.887 0.926 0.915	0.074 0.113 0.102	0.825 0.954 0.918	0.887 0.926 0.915	0.936 0.927 0.929	0.822 0.96 0.921	0.074074	0.113208
6	42 11 14 121	0.792 0.896 0.867	0.104 0.208 0.178	0.75 0.917 0.87	0.792 0.896 0.867	0.925 0.92 0.922	0.807 0.957 0.914	0.103704	0.207547

ADA Boost J48 25 TEST

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION MATRIX	16 11 3 64	21 6 5 62	16 11 5 62	15 12 5 62	17 10 4 63	19 8 4 63	20 7 5 62	20 7 5 62	20 7 5 62
TYPE I ERROR	0.044776	0.074627	0.074627	0.074627	0.059701	0.059701	0.074627	0.074627	0.074627
TYPE II ERROR	0.407407	0.222222	0.407407	0.444444	0.37037	0.296296	0.259259	0.259259	0.259259

ADA Boost J48 25 Results

In Table Bagging decision stump 25 iteration fit I calculated the type errors for both Type I and Type. In this experiment we used meta learner bagging with learner decision stump. I iterated the cost matrix to find the optimal cost ratio model with number iterations at 25. I tried high values as well, but I was not able to diverge where type II is lower than type I the closest in the current FIT table is where type II is the lowest is at 5 for fit and 3 for test. The reason is because it has the lowest type 2 with optimal recall and precision values. Also, in the book it speaks about ROC value and slight some researchers use it to determine at point like this. costs [1] page 173. ROC value is not ideal evaluating machine learning models in situations with known error costs. Also, people use the AUC because the larger the area the better the model which the author speaks about as well on page 173. Also, another determining factor was the precision and recall precision was one of the one closest to 1[1] without sacrificing type II.

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -I 25 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

ADA Boost Decision Stump 10 – FIT

COST MATRIX	CONFUSION MATRIX	TP RATE	FP RATE	PRECISION	RECALL	ROC AREA	PRC AREA	TYPE I ERROR	TYPE II ERROR
.5	36 17	0.679	0.111	0.706	0.679	0.895	0.743	0.111111	0.320755
	15 120	0.889	0.321	0.876	0.889	0.895	0.949		
		0.83	0.262	0.828	0.83	0.895	0.891		
.75	35 18	0.66	0.111	0.7	0.66	0.897	0.735	0.111111	0.339623
	15 120	0.889	0.34	0.87	0.889	0.897	0.943		
		0.824	0.275	0.822	0.824	0.897	0.885		
1	36 17	0.679	0.126	0.679	0.679	0.893	0.733	0.125926	0.320755
	17 118	0.874	0.321	0.874	0.874	0.893	0.942		
		0.819	0.266	0.819	0.819	0.893	0.883		
2	46 7	0.868	0.193	0.639	0.868	0.925	0.785	0.192593	0.132075
	26 109	0.807	0.132	0.94	0.807	0.925	0.955		
		0.824	0.149	0.855	0.824	0.925	0.907		
2.5	45 8	0.849	0.185	0.643	0.849	0.908	0.753	0.185185	0.150943
	25 110	0.815	0.151	0.932	0.815	0.908	0.949		
		0.824	0.161	0.851	0.824	0.908	0.894		
3	48 5	0.906	0.17	0.676	0.906	0.922	0.774	0.17037	0.09434
	23 112	0.83	0.094	0.957	0.83	0.922	0.956		
		0.851	0.116	0.878	0.851	0.922	0.905		
4	49 4	0.925	0.193	0.653	0.925	0.927	0.777	0.192593	0.075472
	26 109	0.807	0.075	0.965	0.807	0.927	0.966		
		0.84	0.108	0.877	0.84	0.927	0.913		
5	49 4	0.925	0.193	0.653	0.925	0.934	0.841	0.192593	0.075472
	26 109	0.807	0.075	0.965	0.807	0.934	0.969		
		0.84	0.108	0.877	0.84	0.934	0.933		
6	49 4	0.925	0.193	0.653	0.925	0.929	0.809	0.192593	0.075472
	26 109	0.807	0.075	0.965	0.807	0.929	0.967		
		0.84	0.108	0.877	0.84	0.929	0.922		

ADA Boost Decision Stump 10 Test

COST MATRIX	.5	.75	1	2	2.5	3	4	5	6
CONFUSION	23 4	23 4	20 7	25 2	25 2	25 2	24 3	25 2	25 2
MATRIX	10 57	10 57	6 61	16 51	14 53	14 53	14 53	14 53	14 53
TYPE I ERROR	0.149254	0.149254	0.089552	0.238806	0.208955	0.208955	0.208955	0.208955	0.208955
TYPE II ERROR	0.148148	0.148148	0.259259	0.074074	0.074074	0.074074	0.111111	0.074074	0.074074

ADA Boost Decision Stump 10 Results

In Table ADA Boost Decision Stump 10 we were able to validate where the divergence happened and validate optimal cost ratio. The optimal cost ration for FIT was 2.5 the difference calc was small also it had a solid recall when comparing other type II and ROC was highest without removing TYPE II out of the equation which is not a major factor in decision making but it helps referring to page 171 on roc values[1]. In test which did real well at .75 was the optimal result which had optimal values are almost the same which was surprising considering how it did with .75 in fit. The value for type I and Type II are almost the same for .75 in test.

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump

ADA Boost Decision Stump 25 FIT

cost matrix	confusion matrix	TP Rate	FP Rate	Precision	Recall	ROC Area	PRC Area	Type I Error	Type II Error
.5	32 21 12 123	0.604 0.911 0.824	0.089 0.396 0.31	0.727 0.854 0.818	0.604 0.911 0.824	0.9 0.9 0.9	0.73 0.952 0.889	0.088889	0.396226
.75	43 10 13 122	0.811 0.904 0.878	0.096 0.189 0.163	0.768 0.924 0.88	0.811 0.904 0.878	0.904 0.904 0.904	0.713 0.954 0.886	0.096296	0.188679
1	44 9 17 118	0.83 0.874 0.862	0.126 0.17 0.157	0.721 0.929 0.871	0.83 0.874 0.862	0.896 0.896 0.896	0.703 0.95 0.881	0.125926	0.169811
2	48 5 28 107	0.906 0.793 0.824	0.207 0.094 0.126	0.632 0.955 0.864	0.906 0.793 0.824	0.896 0.896 0.896	0.678 0.948 0.871	0.207407	0.09434
2.5	44 9 17 118	0.83 0.874 0.862	0.126 0.17 0.157	0.721 0.929 0.871	0.83 0.874 0.862	0.896 0.896 0.896	0.703 0.95 0.881	0.125926	0.169811
3	49 4 28 107	0.925 0.793 0.83	0.207 0.075 0.113	0.636 0.964 0.872	0.925 0.793 0.83	0.896 0.896 0.896	0.685 0.947 0.873	0.207407	0.075472
4	48 5 32 103	0.906 0.763 0.803	0.237 0.094 0.135	0.6 0.954 0.854	0.906 0.763 0.803	0.893 0.893 0.893	0.667 0.946 0.868	0.237037	0.09434
5	48 5 35 100	0.906 0.741 0.787	0.259 0.094 0.141	0.578 0.952 0.847	0.906 0.741 0.787	0.893 0.893 0.893	0.676 0.946 0.87	0.259259	0.09434
6	50 3 34 101	0.943 0.748 0.803	0.252 0.057 0.112	0.595 0.971 0.865	0.943 0.748 0.803	0.887 0.887 0.887	0.655 0.944 0.863	0.251852	0.056604

ADA Boost Decision Stump 25 Test

cost matrix	.5	.75	1	2	2.5	3	4	5	6
confusion matrix	23 4 7 60	23 4 9 58	24 3 9 58	25 2 16 51	24 3 9 58	25 2 16 51	25 2 16 51	25 2 20 47	25 2 20 47
Type I Error	0.104478	0.134328	0.134328	0.238806	0.134328	0.238806	0.238806	0.298507	0.298507
Type II Error	0.148148	0.148148	0.111111	0.074074	0.111111	0.074074	0.074074	0.074074	0.074074

ADA Boost Decision Stump 25 Results

In Table ADA Boost Decision Stump 25 we were able to validate where the divergence happened and validate optimal cost ratio. The optimal cost ration for FIT was 2 the difference calc was optimal also precision and recall was most optimal when comparing other type II and ROC was highest without removing TYPE II out of the equation which is not a major factor in decision making but it helps referring to page 171 on roc values[1]. In Test this most optimal was 1 where it did well. It did well when comparing to other iternum 25 runs in comparison.

Command: weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -I 25 -W weka.classifiers.trees.DecisionStump

Evaluation Table

Classifier	Iter	cost	Type I Error	Type II Error
Bagging j48	10	2.5	0.16	0.11
Bagging j48	25	2	0.14	0.09
Bagging decision stump	10	2	0.2	0.08
Bagging decision stump	25	2	0.2	0.09
adaboost j48	10	4	0.08	0.13
adaboost j48	25	5	0.07	0.11
adaboost decision stump	10	2.5	0.18	0.15
adaboost decision stump	25	2	0.2	0.09

Evaluation Results

Above in the evaluation table after running through all classifiers which is time consuming process, we have the final optimal table which shows all optimal in each table and run from FIT data set We can see just by looking at the type I and type II values Bagging decision stump did the best out of all of them. It is usually used in conjunction with a boosting algorithm page 391[1].It still feels like a rough model when you take account what attributes it looks at. This also proves when iterating through value in the cost matrix it helps achieve a more improved classification and helps lowering the Type II error which is the harmful one in our use case.

References

[1] Witten, Ian H. et al. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2017