ASSIGNMENT 5

Asam Mahmood

Florida Atlantic University

PROF. TAGHI M KOSHGOFTAR

Course CAP 6673 Data Mining and Machine Learning

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 logistic regression. 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. In Part 2 of this project we will use multi-layer Perceptron on our fit data and test data sets for evaluation.

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. We use weka as a tool to train and test models in this project. In my case for this project, I was able to use the command line tool of weka to run multiple results to find the most optimal cost using python system library and parse out the output from the results to quickly create graphs and tables.

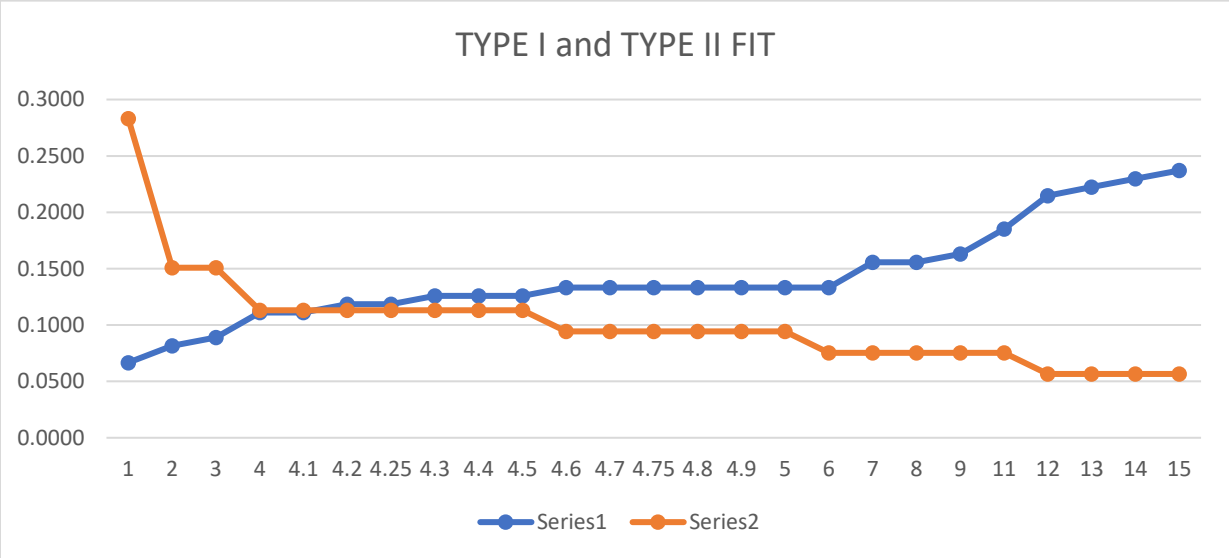
Metrics in tables

In Fit tables I included Cost matrix where I set values for cost. I included the Confusion matrix values and calculated type II and type II values in the metrics within tables. I also include graphs as artifacts of evidence for optimal value analysis.

Evaluation on fit data set

cost	*TP*	*FN*	*FP*	*TN*	*TPR*	*TNR*	*Type I Error*	*Type II Error*
1	38	15	9	126	0.716981	0.933333	0.066667	0.283019
2	45	8	11	124	0.849057	0.918519	0.081481	0.150943
3	45	8	12	123	0.849057	0.911111	0.088889	0.150943
4	47	6	15	120	0.886792	0.888889	0.111111	0.113208
4.1	47	6	15	120	0.886792	0.888889	0.111111	0.113208
4.2	47	6	16	119	0.886792	0.881481	0.118519	0.113208
4.25	47	6	16	119	0.886792	0.881481	0.118519	0.113208
4.3	47	6	17	118	0.886792	0.874074	0.125926	0.113208
4.4	47	6	17	118	0.886792	0.874074	0.125926	0.113208
4.5	47	6	17	118	0.886792	0.874074	0.125926	0.113208
4.6	48	5	18	117	0.90566	0.866667	0.133333	0.09434
4.7	48	5	18	117	0.90566	0.866667	0.133333	0.09434
4.75	48	5	18	117	0.90566	0.866667	0.133333	0.09434
4.8	48	5	18	117	0.90566	0.866667	0.133333	0.09434
4.9	48	5	18	117	0.90566	0.866667	0.133333	0.09434
5	48	5	18	117	0.90566	0.866667	0.133333	0.09434
6	49	4	18	117	0.924528	0.866667	0.133333	0.075472
7	49	4	21	114	0.924528	0.844444	0.155556	0.075472
8	49	4	21	114	0.924528	0.844444	0.155556	0.075472
9	49	4	22	113	0.924528	0.837037	0.162963	0.075472
11	49	4	25	110	0.924528	0.814815	0.185185	0.075472
12	50	3	29	106	0.943396	0.785185	0.214815	0.056604
13	50	3	30	105	0.943396	0.777778	0.222222	0.056604
14	50	3	31	104	0.943396	0.77037	0.22963	0.056604
15	50	3	32	103	0.943396	0.762963	0.237037	0.056604

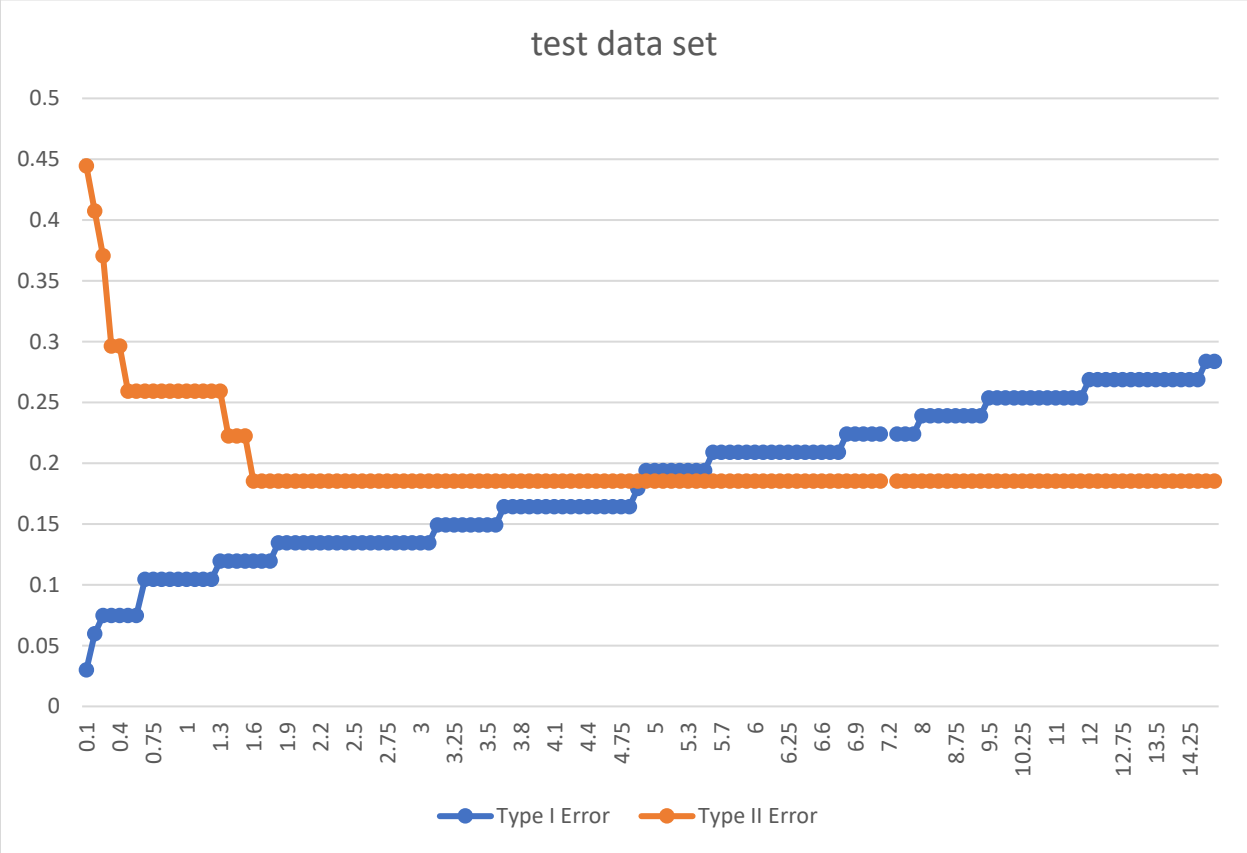
Fit Diagram



Evaluation on test data set

c	tp	fn	fp	tn	TPR	TNR	Type I Error	Type II Error
1	20	7	7	60	0.740741	0.895522	0.104478	0.259259
2	22	5	9	58	0.814815	0.865672	0.134328	0.185185
3	22	5	9	58	0.814815	0.865672	0.134328	0.185185
4	22	5	11	56	0.814815	0.835821	0.164179	0.185185
5	22	5	13	54	0.814815	0.80597	0.19403	0.185185
6	22	5	14	53	0.814815	0.791045	0.208955	0.185185
7	22	5	15	52	0.814815	0.776119	0.223881	0.185185
8	22	5	16	51	0.814815	0.761194	0.238806	0.185185
9	22	5	16	51	0.814815	0.761194	0.238806	0.185185
10	22	5	17	50	0.814815	0.746269	0.253731	0.185185
11	22	5	17	50	0.814815	0.746269	0.253731	0.185185
12	22	5	18	49	0.814815	0.731343	0.268657	0.185185
13	22	5	18	49	0.814815	0.731343	0.268657	0.185185
14	22	5	18	49	0.814815	0.731343	0.268657	0.185185

Test diagram



Evaluation of FIT and Test

This project for part 1 demonstrates a method for using logistic regression in software quality modeling to predict whether each module will be fault-prone or not[1][2]. I chose c at 4.1-4.2 (when you round to hundredth 4.1 is .11 and .11 exactly the same which is the best) is the most optimal. The reason I chose this because type I and type II are the most similar at those points and when you validate through graph above it is the point where they intersect. For test set the most optimal was at 5. Because at that point to in the graph they intersect, and the type I and type II are the most similar.

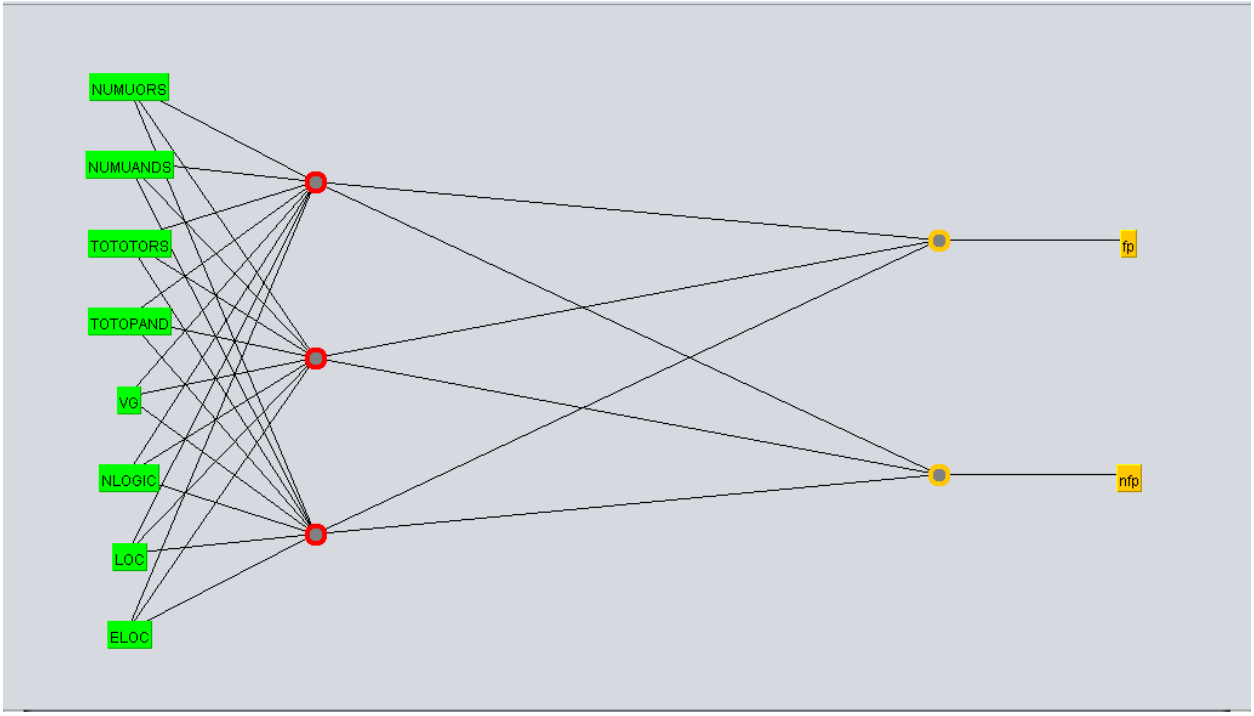
When looking at coefficients Totopands, vg are our negative values means that it's associated with more non fault prone classification. A positive coefficient meant its associated with more fault prone classification. We are more interested in the positive class of fp. NUMOUR,ELOC,NLOGIC are the ones that contributed the most when classifying as fault prone. ELOC will increase the chances of being fault prone by 11(rounding up) because it is at 110.65%. NLOGIC will increase the chances of being fault prone by 9(rounding up) because it is at 109%. NUMOURS will increase the chances of being fault prone by 18 because it is at 118.9%.

One of things to remember about the **odds ratio** is that when a **odds ratio value is** greater than 1 it is considered a positive association[1]. When you look at the negative values VG and TOTOPANDS in the **odd ratios** table they both are under the threshold of 100% both will lead to reduction on the probability of classification as fault prone by 2% and 16%.

odd ratios	
variable	class : fp
NUMUORS	1.1885
NUMUANDS	1.0283
TOTOTORS	1.0017
TOTOPAND	0.9857
VG	0.8468
NLOGIC	1.0982
LOC	1.0003
ELOC	1.1065

coefficient	
variable	class : fp
NUMUORS	0.1727
NUMUANDS	0.0279
TOTOTORS	0.0017
TOTOPAND	-0.0144
VG	-0.1663
NLOGIC	0.0937
LOC	0.0003
ELOC	0.1012
Intercept	-7.4834

Evaluation Part 2



Comparison table MLP and Logistic Regression

Type	tp	fn	fp	tn	TPR	TNR	Type I Error	Type II Error	Misclassification
MLP:TEST	**21**	**6**	**8**	**59**	**0.777778**	**0.880597**	**0.12**	**0.22**	**14.8936**
MLP:FIT	**37**	**16**	**9**	**126**	**0.698113**	**0.933333**	**0.06**	**0.30**	**13.2979**
MLP-non-normalize-fit	**0**	**53**	**0**	**135**	**0**	**1.0**	**0**	**1.0**	**28.1915**
MLP-non-normalize-test	**0**	**27**	**0**	**67**	**0**	**1.0**	**0**	**1.0**	**28.1915**
Logistic optimal	**47**	**6**	**15**	**120**	**0.886792**	**0.888889**	**0.11**	**0.11**	**7.4468**

MLP Evaluation and comparison of logistic regression

Looking at comparison table above we can see the logistic regression with the selected most optimal c has more balanced type error than all mlp results. The logistic regression has a lower misclassification rate as well. The mlp results were not balanced between type I and type II. I also validated the normalized set for fit and test performed better than and non-normalized results for type I and type II.

When comparing the weight with logistic regression with mlp, ***VG*** is the only positive weight. It is like logistic regression that ***VG*** was associated with reduction on the probability of classification as fault prone. I also compared against a non-normalized mlp output and normalized output as well and they both were the same conclusions. When comparing normalized and non-normalized I had different confusion matrix and type I and type II errors. In my curiosity I compared both. When I was comparing NN I basically came to conclusion it is just a method that takes parameters and produces a result. As with all methods, it has a set of parameters. You must normalize the values that you want to pass to the NN to make sure it is within the parameters list. As with every method, if the arguments that are passed in are not in the parameters list, the result is not guaranteed to be correct. The exact behavior of the NN on arguments passed in are outside of the parameters depends on the implementation of the NN. In the end the final output is useless if arguments passed in are not within the parameter list [1]. Also If a attribute in the Dataset is larger in scale when comparing to other attributes then this large attribute becomes dominating then as a result, the results of the NN will not be as accurate[1]. That is why you see non-normalized perform as bad as it did vs normalized when comparing type errors. Normalized performed better then non normalized.

Sigmoid Node table 1 – normalize

Sigmoid Node 0		**Sigmoid Node 2**		**Sigmoid Node 3**		**Sigmoid Node 4**	
Inputs	Weights	Inputs	Weights	Inputs	Weights	Inputs	Weights
Threshold	2.797558	Threshold	-1.93934	Threshold	-2.95959	Threshold	-2.18868
Node 2	-1.733	NUMUORS	-2.52134	NUMUORS	-3.72243	NUMUORS	-2.78097
Node 3	-2.99962	NUMUANDS	-0.87892	NUMUANDS	-1.36078	NUMUANDS	-1.01521
Node 4	-2.01306	TOTOTORS	-0.34367	TOTOTORS	-0.56469	TOTOTORS	-0.42257
		TOTOPAND	-0.30717	TOTOPAND	-0.46463	TOTOPAND	-0.39555
		VG	1.557417	VG	2.573262	VG	1.772838
		NLOGIC	-0.00723	NLOGIC	0.126706	NLOGIC	0.013256
		LOC	-0.91602	LOC	-1.45658	LOC	-1.0183
		ELOC	-0.35033	ELOC	-0.44659	ELOC	-0.32276

Sigmoid Node table 2 – non normalized

Sigmoid Node 0		**Sigmoid Node 2**		**Sigmoid Node 3**		**Sigmoid Node 4**	
Inputs	Weights	Inputs	Weights	Inputs	Weights	Inputs	Weights
Threshold	-0.47077	Threshold	0.047450112	Threshold	-0.01399	Threshold	-0.02886
Node 2	-0.01755	NUMUORS	-0.00681797	NUMUORS	-0.0079	NUMUORS	-0.01471
Node 3	-0.48002	NUMUANDS	-0.018905209	NUMUANDS	0.013376	NUMUANDS	-0.04159
Node 4	-0.0502	TOTOTORS	-0.002620618	TOTOTORS	0.109682	TOTOTORS	-0.01391
		TOTOPAND	-0.045587031	TOTOPAND	0.073894	TOTOPAND	-0.07669
		VG	-0.018244076	VG	0.013958	VG	0.008506
		NLOGIC	-0.044132458	NLOGIC	0.025469	NLOGIC	-0.02793
		LOC	-0.099917138	LOC	0.168355	LOC	-0.03426
		ELOC	-0.042366155	ELOC	0.007536	ELOC	0.028609

References

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