Preliminary Comparison of Techniques for Dealing with Imbalance in Software Defect Prediction

Daniel Rodriguez[1] Israel Herraiz[2] Rachel Harrison[3]
J Dolado[4] JC Riquelme[5]

[1]University of Alcalá, Spain
[2]AMADEUS, Spain
[3]Oxford Brookes University, UK
[4]University of The Basque Country, Spain
[5]University of Seville, Spain

EASE 2014

Outline

Outline

Imbalance data

- Most publicly available datasets in software defect prediction are highly imbalanced, i.e., samples of non-defective modules vastly outnumber the defective ones.
- Data mining algorithms generate poor models because they try to optimize the overall accuracy but perform badly in classes with very few samples (minority class which is usually the one we are interested in). This is due to the fact that most data mining algorithms assume balanced datasets.
- The imbalance problem is known to affect many machine learning algorithms such as decision tress, neural networks or support vectors machines.

Outline

Sampling

Sampling techniques are classified as oversampling or undersampling and are based on adding or removing instances of the training dataset

- Random OverSampling (ROS) replicates instances from the minority class towards a more balanced distribution
- Random Under-Sampling (RUS) removes instances from the majority class

More intelligent approaches include:

- SMOTE (Synthetic Minority Over-sampling Technique) generates new instances based on a number of nearest neighbours (NN)
- There are other variations of SMOTE (Borderline SMOTE) or oversampling/undersampling based on NN

Cost-Sensitive Classifiers (CSC)

- The idea is to penalise differently the different types of error (in binary classification, the false positives and false negatives).
- Adapt classifiers to handle imbalanced datasets by either
  - adding weights to instances (if the base classifier algorithm allows this) or resampling the training data according to the costs assigned to each class in a predefined cost matrix
  - generating a model that minimises the expected cost

There is no systematic approach to do so. However, it is common practice to set the cost to equalize the class distribution.

Ensembles

- Bagging (Bootstrap aggregating). A base learner is applied to multiple equal size datasets created from the original data using bootstraping. Predictions are based on voting of the individual predictions
- Boosting techniques generate multiple models that complement each other inducing models that improve regions of the data where previous induced models preformed poorly. This is achieved by increasing the weights of instances wrongly classified, so new learners focus on those instances. Final classification is based on a weighted voted among all members of the ensemble
- Stacking (Stacked generalization) combines different types of models

Hybrid Approaches

- SMOTEBoost introduces SMOTE in each round of boosting to enable each learner to be able to sample more of the minority class cases.
- RUSBoost is similar to SMOTEBoost but RUSBoost applies Random Under Sampling instead of SMOTE in each iteration
- MetaCost combines bagging with cost-sensitive classification. Bagging is used to relabel training data so that each training instance is assigned the prediction that minimizes the expected cost. Based on the modified training data, MetaCost induces a single new classifier based on the new relabeled data which provides information about how a decision was reached

Outline

Datasets

We have used available software defect prediction datasets generated from projects carried out at NASA.
These datasets are available in two different versions from:

- the PROMISE repository[1]
- and the original one which has curated by Shepperd et al.[2] who analysed different problems and differences with these datasets and curated the repository.

[1] https://code.google.com/p/promisedata/
[2] http://nasa-softwaredefectdatasets.wikispaces.com/

MDP

Instances, Imbalance and Problems

	#Ins	%IR	%Dup	%Inc	#InsD'	%Prob	%IR D'	#Ins D"	%Ins D"	%IR D"
CM1	505	9.5	5.15	0	344	31.88	12.21	327	35.25	12.84
JM1	10878	19.32	24.16	8.17	9593	11.83	18.34	7720	29.03	20.88
KC1	2107	15.42	50.78	12.01	2095	0.57	15.51	1162	44.85	25.3
KC3	458	9.39	2.62	0	200	56.33	18	194	57.64	18.56
KC4	125	48.8	8	7.2	n.a	100	n.a	n.a	100	n.a
MC1	9466	0.72	84.22	1.12	8737	51.14	0.78	1952	80.49	1.84
MC2	161	32.3	2.48	0	127	21.12	34.65	124	22.36	35.48
MW1	403	7.69	3.72	1.24	264	34.49	10.23	250	37.72	10
PC1	1107	6.87	7.68	1.17	759	32.07	8.04	679	37.13	8.1
PC2	5589	0.41	17.61	0	1493	72.55	1.07	722	86.87	2.22
PC3	1563	10.24	5.05	0.38	1125	28.41	12.44	1053	31.35	12.35
PC4	1458	12.21	11.39	0.21	1399	7.68	12.72	1270	12.48	13.86
PC5	17186	3	91.53	10.04	16962	10.37	2.96	1694	90.23	27.04
Avg		13.53	24.18	3.2		35.26	12.25		51.18	15.71

PROMISE

Instances, Imbalance and Problems

	#Ins	%IR	%Dup	%Inc	#InsD'	%Prob	%IR D'	#Ins D"	%Ins D"	%IR D"
CM1	498	9.84	18.88	0.4	495	0.6	9.7	437	12.25	10.53
JM1	10885	19.35	24.14	8.17	9591	11.89	18.34	7720	29.08	0.28
KC1	1783	18.28	60.01	14.19	2095	0.79	15.51	1162	53.11	25.3
KC2	522	20.5	34.87	22.61	484	7.28	20.66	325	37.74	28.31
KC3	458	9.39	37.12	0.44	458	6.33	9.39	324	31	12.96
MC1	9398	0.72	84.83	1.13	8737	51.51	0.78	1952	81.07	1.84
MC2	161	32.3	3.73	1.24	159	0	32.7	155	3.11	32.9
MW1	403	7.69	8.93	1.74	402	0	7.71	375	6.7	7.47
PC1	1109	6.94	21.64	1.17	1083	6.67	6.65	919	17.67	6.53
PC2	5589	0.41	82.68	1.79	5356	20.81	0.43	1362	76.88	1.54
PC3	1563	10.24	12.09	0.58	1535	3.45	10.29	1409	8.83	10.5
PC4	1458	12.21	11.39	0.21	1379	7.68	12.91	1270	12.48	13.86
PC5	17186	3	91.53	10.04	16962	10.37	2.96	1694	90.23	27.04
Avg		11.61	37.83	4.9		9.8	11.39		35.4	13.77

MDP
Attributes

MDP	# Att	# Probl	% Prob	# Att D'	Removed
CM1	41	6	14.63	38	3
JM1	22	9	40.91	22	0
KC1	22	4	18.18	22	0
KC3	41	3	7.32	40	1
KC4	41	30	73.17	0	41
MC1	40	5	12.5	39	1
MC2	41	2	4.88	40	1
MW1	41	4	9.76	38	3
PC1	41	8	19.51	38	3
PC2	41	8	19.51	37	4
PC3	41	7	17.07	38	3
PC4	41	11	26.83	38	3
PC5	40	5	12.5	39	1
Avg			21.29		

PROMISE

Attributes

PROMISE	# Att	Prob	%Prob	Att D'	Removed
CM1	22	15	68.18	21	1
JM1	22	16	72.73	22	0
KC1	22	16	72.73	22	0
KC2	22	15	68.18	22	0
KC3	40	1	2.5	40	0
MC1	39	4	10.26	39	0
MC2	40	0	0	40	0
MW1	38	0	0	38	0
PC1	22	15	68.18	22	0
PC2	37	3	8.11	23	14
PC3	38	3	7.89	38	0
PC4	38	8	21.05	38	0
PC5	39	4	10.26	39	0
Avg			31.54		

Outline

Binary classifiers Evaluation

Confusion matrix

		Actual Pos	Neg	
Pred	*Pos*	True Positive (TP)	False Positive (FP) Type I error (False alarm)	*Positive Predictive Value (PPV)*= *Confidence* = *Precision* = $= \frac{TP}{TP+FP}$
	Neg	False Negative (FN) Type II error	True Negative (TN)	*Negative Predictive Value (NPV)*= $\frac{TN}{FN+TN}$
		Recall = *Sensitivity* = $TP_r = \frac{TP}{TP+FN}$	*Specificity* = $TN_r = \frac{TN}{FP+TN}$	

Evaluation measures

Common used measures with imbalance data include ROC (AUC), MCC, and the $f-measure$, which are defined as:

- f-measure is the harmonic mean of precision and recall:
  $f1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$
- Area Under the ROC Curve
  $AUC = \frac{1 + TP_r - FP_r}{2}$
- Matthews Correlation Coefficient (MCC)
  $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Outline

Running of the Experiments

- All algorithms were run using the WEKA environment, the Experimenter tool. the t-test was used to compare with the base classifier
- Results were obtained with 5 runs, each run is a 5-fold CV, i.e., 5x5CV.
- Based classifiers
  - C4.5 (called J48 in Weka) is a decision tree where the leaves of the tree correspond to classes, nodes correspond to features, and branches to their associated values
  - The naïve Bayes classifier assigns a set of attributes $A_1, A_2, \ldots, A_n$ to a class C such that $P(C|A_1, A_2, \ldots, A_n)$ is maximum, that is the probability of the class description value given the attribute instances, is maximal.

Outline

D’ MCC with J48

	J48	RUS	ROS	SMOTE	SMOTEBoost	RUSBoost	MetaCost	CSC-Resamp	CSC-MinCost	AdaBoostM1	Bagging	RF	
CM1	.10	.18	.17	.17	.16	.16	.23	.23	.13	.19	.12	.04	
JM1	.23	.24	.23	.24	.26	.24	.25	.24	.21	.24	.26	.18	•
KC1	.28	.32	.30	.31	.33	.34	◦ .33	.32	.21	.31	.36	◦ .31	
KC3	.22	.25	.24	.29	.29	.26	.22	.24	.18	.24	.30	.28	
MC1	.44	.20	• .40	.43	.42	.17	• .35	.44	.41	.59	◦ .45	.45	
MC2	.21	.21	.21	.20	.34	.36	.16	.16	.18	.32	.33	.38	
MW1	.32	.22	.10	• .15	.19	.27	.22	.20	.31	.25	.20	.30	
PC1	.24	.29	.24	.26	.29	.33	.29	.30	.25	.23	.25	.22	
PC2	.00	.16	◦ .07	.09	.08	.12	.11	.09	.00	.01	.01	.00	
PC3	.24	.25	.22	.22	.30	.31	.32	◦ .29	.29	.29	.23	.19	
PC4	.51	.52	.47	.52	.56	.55	.53	.51	.54	.53	.51	.54	
PC5	.50	.52	.51	.54	◦ .55	◦ .48	.56	◦ .52	.52	.52	.52	.52	
Avg	.27	.28	.26	.29	.31	.30	.30	.29	.27	.31	.30	.28	

◦, • statistically significant improvement or degradation

D” MCC with J48

	J48	RUS	ROS	SMOTE	SMOTEBoost	RUSBoost	MetaCost	CSC-Resamp	CSC-MinCost	AdaBoostM1	Bagging	RF
CM1	0.11	0.12	0.13	0.16	0.16	0.18	0.19	0.17	0.08	0.13	0.09	0.05
JM1	0.19	0.21	0.20	0.20	0.22	0.21	0.21	0.18	0.07	•0.20	0.22	0.17
KC1	0.23	0.23	0.23	0.23	0.29	0.22	0.19	0.18	0.07	•0.26	0.28	0.27
KC3	0.23	0.25	0.24	0.24	0.30	0.23	0.27	0.25	0.18	0.20	0.33	0.29
MC1	0.07	0.13	0.22	0.18	0.29	◦0.14	0.15	0.23	0.08	0.28	0.07	0.08
MC2	0.21	0.21	0.21	0.18	0.35	0.31	0.21	0.13	0.15	0.30	0.36	0.35
MW1	0.16	0.27	0.18	0.17	0.31	0.32	0.26	0.17	0.24	0.25	0.37	0.26
PC1	0.23	0.26	0.27	0.31	0.33	0.31	0.27	0.30	0.22	0.31	0.26	0.28
PC3	0.22	0.26	0.21	0.25	0.30	0.26	0.29	0.26	0.28	0.23	0.22	0.16
PC4	0.46	0.50	0.45	0.50	0.54	0.53	0.50	0.51	0.51	0.51	0.52	0.53
PC5	0.33	0.33	0.33	0.34	0.39	◦0.37	0.34	0.33	0.29	0.37	0.37	0.36
Avg	0.22	0.25	0.24	0.25	0.32	0.28	0.26	0.25	0.20	0.28	0.28	0.25

◦, • statistically significant improvement or degradation

MPD
D' ROC

	J48	RUS	ROS	SMOTE	SMOTEBoost	RUSBoost	MetaCost	CSC-Resamp	CSC-MinCost	AdaBoostM1	Bagging	RF	
CM1	.56	.62	.56	.59	.73	○.74	○.68	○.64	.57	.73	○.77	○.75	○
JM1	.67	.65	.60	●.66	.70	○.70	○.67	.66	.63	●.69	.72	○.73	○
KC1	.67	.70	.62	.69	.77	○.77	○.73	.66	.64	.75	○.81	○.82	○
KC3	.59	.61	.60	.65	.72	○.71	○.65	.67	.59	.71	.69	.72	
MC1	.77	.88	○.80	.81	.96	○.93	○.74	.81	.65	●.94	○.91	○.88	○
MC2	.62	.62	.62	.61	.73	○.73	○.59	.58	.59	.72	○.72	○.75	○
MW1	.58	.63	.55	.59	.69	.72	.67	.63	.64	.67	.73	○.74	○
PC1	.70	.73	.59	.68	.83	○.82	○.70	.68	.66	.82	○.83	○.84	○
PC2	.52	.77	○.53	.56	.79	○.89	○.63	.56	.50	.76	○.78	○.70	
PC3	.65	.68	.59	.64	.80	○.81	○.72	○.68	.68	.80	○.81	○.83	○
PC4	.77	.79	.70	.75	.93	○.92	○.84	.81	.81	.92	○.92	○.94	○
PC5	.77	.91	○.64	●.79	.95	○.96	○.89	○.67	●.80	.95	○.96	○.96	○

○, ● statistically significant improvement or degradation

MPD
D' MCC

	NB	RUS	ROS	SMOTE	SMOTEBoost	RUSBoost	MetaCost	CSC-Resamp	CSC-MinCost	AdaBoostM1	Bagging	RF
CM1	.21	.21	.21	.21	.17	.18	.20	.21	.21	.21	.22	.20
JM1	.22	.23	.22	.23	○.18	●.23	.24	○.23	.23	.22	.22	.21
KC1	.29	.30	.30	.31	○.26	.27	.33	.31	○.31	○.29	.30	.31
KC3	.26	.26	.28	.28	.27	.26	.21	.29	.29	.24	.29	.29
MC1	.20	.18	.19	●.18	●.15	●.14	.17	.19	●.19	●.20	.19	.18
MC2	.31	.31	.31	.33	.31	.24	.32	.32	.32	.38	.33	.33
MW1	.32	.31	.31	.31	.30	.24	●.23	●.31	.31	.33	.32	.33
PC1	.28	.26	.27	.27	.16	●.16	●.16	●.27	.27	.26	.28	.27
PC2	.08	.13	.09	.09	.03	.13	.15	.08	.08	.06	.08	.09
PC3	.15	.23	.14	.13	.05	.08	.02	●.11	●.11	●.16	.18	.14
PC4	.32	.31	.34	.38	○.28	.05	●.25	●.35	.35	.37	.34	.28
PC5	.42	.44	.42	.42	.27	●.21	●.41	.42	.42	.42	.42	.41

○, ● statistically significant improvement or degradation

Outline

Conclusions

- There are differences depending on the base classifier, evaluation metrics used and the preprocessing (cleaning) of the data.
- There are some *questions* about the quality of the data
- Remove duplicates?
  - They should not be removed if they come from the *actual* distribution of the data but this is unknown in this case.
  - but are those missing values?
- Meta-learners algorithms as in general they seem to work quite well, but they do not explain why a module can be defect prone (compared to rules or decision trees)

Outline

Future Work

- Used other datasets and better statistical tests
- Duplicates are not the only problem
  - Dataset shift (training and test data follow different distributions)
  - Distribution of the cross validation data
  - Small disjuncts, the lack of density or small sample size, class overlapping, the correct management of borderline examples or noisy data.
  - How to measure the quality of the data?
- Combine it with Feature Selection
  - A reduced volume of data allows different data mining or searching techniques to be applied.
  - Irrelevant and redundant attributes generate less accurate and more complex models.