Effect of treating ordinal class variables as Regression problem v/s Classification



|  |  |  |
| --- | --- | --- |
| Ankur Garg North Carolina State University Raleigh, NC agarg12@ncsu.edu | Chinmoy Baruah North Carolina State University Raleigh, NC cbaruah@ncsu.edu | Sanket Shahane North Carolina State University Raleigh, NC svshahan@ncsu.edu |

Abstract

Class labels are not always nominal in nature they can sometimes have ordinal relationships among them. Bug priority prediction is one such problem. Such problems give rise to the question whether we treat these problems as classification problems or regression problems. In this paper we evaluate a technique which treats the problem as a regression problem and provide our critique on their conclusions based on some defined key criteria. We also propose our plan to statistically check their conclusions based on the key criteria.

***CCS Concepts*** •**Software Engineering** → Bugs; Bug Priorities; •**Machine Learning** → Classification; Regression; Evaluation; Cross-validation; •**Statistics** → Statistical measures, Bootstraping, Significance tests, Effect size tests.

***Keywords*** Ordinal Categorical Labels, Regression, Bug Prediction, Statistical Evaluation, Self-tuning models

1 Introduction

Assigning priority levels to bugs is a major factor contributing towards fixing it. High priority bugs are more important to be fixed than low priority bugs. Increasing complexity of the software systems is directly correlated to the number of bugs detected/reported. Human evaluation of every bug reported is not always feasible and thus using machine learning techniques to automatically assign appropriate priority levels is must. On a high level, Machine learning tasks are divided into supervised and unsupervised tasks depending upon what the nature of the data is. Having labelled data making predictions about it for the future makes it a supervised task whereas unsupervised tasks are generally grouping/clustering tasks where there is no label attribute attached to the data samples. Supervised ML tasks are further divided into Classification and Regression tasks having categorical and continuous labels respectively. Categorical labels are nominal attributes where ordering doesn’t make sense {boy,girl} for e.g. Continuous labels are numerical attributes where order does make sense. Heart rate for e.g. 72 bmp < 129 bpm.

Interesting fact about bug priorities is that these can be viewed as categories ranging from {p1 to p5}. However, the difference between p1 and p5 is not the same as difference between p1 and p2. Thus we can see that bug priorities are neither just ordinal nor just numerical. They are ordinal categorical in nature since we have a fixed number of categories but they have an ordering relationship between them {p1<p2<p3<p4<p5}. A natural question would be: What kind of Machine Learning technique should we use for such problems? Should we treat it as a pure classification problem or as a regression problem and bin the regression output into categories?

In this paper, we study an interesting approach proposed by Yuan Tian et.al.[1]. They treat this problem as a regression problem and have proposed a greedy algorithm to determine the appropriate bin ranges of the regression output to map it to bug classes. They compare their study with [2] and show that their approach is better than treating the problem as a classification problem. This conclusion is however, a complex technique and is supported by less statistical evidence. We have evaluated their technique according to the criteria presented in section 2 and focus on the key criteria presented in section 3 of the paper. In section 4 we provide our critique of [1] and in section 5 and 6 we propose the technologies, methodologies, and our plan to compare techniques which enable the key criteria and statistically compare it with the approach presented by [1].

2 Criteria

2.1 Model Readability

Readability is the ease with which a reader can decipher a written text. Some of the factors that increases readability are:

* Speed of perception
* Perceptibility at a distance
* Perceptibility in peripheral vision
* Visibility
* Reflex blink technique
* Rate of work (reading speed)
* Eye movements
* Fatigue in reading

Advantage

* A readable text always attracts a larger reader-base.

Disadvantage

* Readability is very different from under-stability. Just because something is highly readable doesn’t mean it could be easily understood.

2.2  Learnability and Repeatability of Results

Learnability is the capability of any item or program to enable the user to learn how to use it. In computational learning theory, learnability is the mathematical analysis of machine learning. The better the learnability of an application, the less training and time it will take for a person to use it.

Advantage

* Learnability is important, because it is closely linked to usability. It is vital that users can pick up how to use an application quickly. If you are creating software for professional use, this can be especially important, as employers are less likely to spend money on software that requires expensive training for staff members.

Repeatability is the variation in measurements taken by a single person or instrument on the same item, under the same conditions, and in a short period of time. Repeatability is a critical building block of a collaborative data science process. Repeatability is the idea that a given process (whether it be a data cleaning script, a feature engineering pipeline, or a modeling algorithm) will produce the same (or nearly the same) output given similar inputs. For data scientists to be able to collaborate, they must be able to rely on the instruments and procedures they have being consistent. This often surfaces itself as a challenge in data science collaboration with environmental and data instability.

Advantage/Disadvantage of using less CPU and less disk/RAM

2.3  Anomaly Detection

Anomaly detection is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset. Three broad categories of anomaly detection techniques exist.[1] Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then testing the likelihood of a test instance to be generated by the learnt model.

Few techniques for anomaly detection are:

* Density-based techniques (k-nearest neighbor, local outlier factor, and many more variations of this concept).
* Subspace and correlation-based] outlier detection for high-dimensional data.]
* One class support vector machines.]
* Replicator neural networks.
* Cluster analysis-based outlier detection.
* Deviations from association rules and frequent itemset.
* Fuzzy logic based outlier detection.
* Ensemble techniques, using feature bagging, score normalization and different sources of diversity

Use for Anomaly Detection

Anomaly detection is applicable in a variety of domains, such as intrusion detection, fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting Eco-system disturbances. It is often used in preprocessing to remove anomalous data from the dataset. In supervised learning, removing the anomalous data from the dataset often results in a statistically significant increase in accuracy.

2.4  Context Aware

Context-aware computing refers to a very peculiar kind of application that can sense the physical environment and behave accordingly. This feature of context awareness can be further enhanced using machine learning. This is possible by providing services not only related to the usage pattern of the users but also the environmental context of the user. Devices may have information about the circumstances under which they are able to operate and based on rules, or an intelligent stimulus, react accordingly. Context awareness is regarded as an enabling technology for ubiquitous computing systems. Context awareness is used to design innovative user interfaces, and is often used as a part of ubiquitous and wearable computing. It is also beginning to be felt in the internet with the advent of hybrid search engines.

Human factors related context is structured into three categories: information on the user (knowledge of habits, emotional state, bio physiological conditions), the user’s social environment (co-location of others, social interaction, group dynamics), and the user’s tasks (spontaneous activity, engaged tasks, general goals). Likewise, context related to physical environment is structured into three categories: location (absolute position, relative position, co-location), infrastructure (surrounding resources for computation, communication, task performance), and physical conditions (noise, light, pressure, air quality).

Advantage/Use of Context Aware. A few situations where context aware would be useful are:

* Operations will be easier as surgeons will be able to view relevant important data like blood count as well as have anatomical references like the blood vessel location.
* Context Aware Expertise Location and Management (ELM) systems will ensure that service techs can interact with subject matter experts when needed.
* Expect further changes on how you work with your smart phones. Google is continually working on enhancing the context awareness quotient and we will see many changes soon.
* You don’t need to tell your TVs what you want to watch or what you want to do — they can predict what you intend to do, which saves you time.

2.3  Incremental Learning

Incremental learning is a machine learning paradigm where the learning process takes place whenever new example(s) emerge and adjusts what has been learned according to the new example(s). It represents a dynamic technique of supervised learning and unsupervised learning that can be applied when training data becomes available gradually over time or its size is out of system memory limits.

Usage of Incremental algorithms: Frequently applied to data streams or big data, addressing issues in data availability and resource scarcity respectively. Stock trend prediction and user profiling are some examples of data streams where new data becomes continuously available. Applying incremental learning to big data aims to produce faster classification or forecasting times.

2.4  Self-Tuning

In control theory a self-tuning system is capable of optimizing its own internal running parameters in order to maximize or minimize the fulfilment of an objective function; typically, the maximization of efficiency or error minimization. Self-tuning systems typically exhibit non-linear adaptive control. Self-tuning systems have been a hallmark of the aerospace industry for decades, as this sort of feedback is necessary to generate optimal multi-variable control for non-linear processes. In the telecommunications industry, adaptive communications are often used to dynamically modify operational system parameters to maximize efficiency and robustness. Self-tuning systems are typically composed of four components: expectations, measurement, analysis, and actions. The expectations describe how the system should behave given exogenous conditions.

Measurements gather data about the conditions and behaviour. Analysis helps determine whether the expectations are being met- and which subsequent actions should be performed. Common actions are gathering more data and performing dynamic reconfiguration of the system.

Advantage

* Facilitates controlling critical processes of systems;
* Approaches optimum operation regimes;
* Facilitates design unification of control systems;
* Shortens the lead times of system testing and tuning;
* Lowers the criticality of technological requirements on control systems by making the systems more robust
* Saves personnel time for system tuning.

2.5  Multi Goal Reasoning - Did not understand

Put some text

2.6  Shareable - Did not understand

Put some text

3 Key Criteria

3.1 Self Tuning

4 Critique

The paper compares their complex yet wonderful approach with Menzies’ algorithms and shows that their’ is better. However, they have not shown enough statistical evidence to support it. One of the things missing is their validation approach does not contain cross validation.

Handling Imbalanced data: The problem discussed in the paper is a class imbalance problem. Before using any kind of model or training technique, we need to find ways to handle the class imbalance between the various classes present in the data. The paper mentioned above doesn’t explicitly use any methods to handle class imbalance. The paper has claimed that the approach used for training inherently handles class imbalance problem but does not provide enough evidence to support that. We did not find its comparison with cases, where pre-training techniques to handle class-imbalance problems were used along with some classification techniques.

The paper doesn’t use any sampling (under-sampling/over-sampling of majority/minority classes) to balance the classes in the dataset. It is observed that more analysis is required with different class imbalance techniques to understand how well the proposed technique performs in comparison to the existing classification techniques.

Using Regression for Ordinal data: The paper concludes that the proposed approach of treating output variable (ordinal variable) as continuous regression problem works better than treating the output variable as categorical variable and using standard classification techniques have multiple problems. One of the results stated in the paper is that Naive Bayes Could not be run despite of supplying 9GB RAM. This is highly surprising and we would like to reproduce the results. Also their statement is that RIPPER could not be completed after running for 8 hours but they report Menzies’ algorithm which uses Ripper had training time of 812.18 sec. Our second hypothesis is that if we properly implement the Naive Bayes algorithm, Random Forest, etc with proper parameter tuning and handling imbalanced dataset, we can statistically compare the methods and have much simpler models.

5 Review

For the above hypotheses, we review 2 simple yet effective technologies in this section.

5.1 SMOTE

SMOTE algorithm to handle the imbalanced dataset.

1. Explain the algorithm and its working here with some citations.

5.2 DE (Differential Evolution)

DE algorithm for Hyperparameter tuning. Many studies have shown the effect that hyperparameter tuning paper by Menzies.

6 Planning

References

|  |  |
| --- | --- |
| [1] | Patricia S. Abril and Robert Plant. 2007. The patent holder’s dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan. 2007), 36–44. DOI: http://dx.doi.org/10.1145/1188913.1188915 |
| [2] | I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. 2002. Wireless Sensor Networks: A Survey. *Comm. ACM* 38, 4 (2002), 393–422. |
| [3] | David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle*. Master’s thesis. Royal Institute of Technology (KTH), Stockholm, Sweden. |
| [4] | P. Bahl, R. Chancre, and J. Dungeon. 2004. SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks. In *Proceeding of the 10th International Conference on Mobile Computing and Networking* (MobiCom’04). ACM, New York, NY, 112–117. |
| [5] | Kenneth L. Clarkson. 1985. *Algorithms for Closest-Point Problems (Computational Geometry)*. Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171. |
| [6] | Jacques Cohen (Ed.). 1996. Special Issue: Digital Libraries. *Commun. ACM* 39, 11 (Nov. 1996). |
| [7] | Bruce P. Douglass. 1998. Statecarts in use: structured analysis and object-orientation. In *Lectures on Embedded Systems*, Grzegorz Rozenberg and Frits W. Vaandrager (Eds.). Lecture Notes in Computer Science, Vol. 1494. Springer-Verlag, London, 368–394. DOI: http://dx.doi.org/10.1007/3-540-65193-429 |
| [8] | Ian Editor (Ed.). 2008. *The title of book two* (2nd. ed.). University of Chicago Press, Chicago, Chapter 100. DOI: http://dx.doi.org/10.1007/3-540-09237-4 |