Comprehensive Studies in Selected Topics in Computer Science (CS8692)

A Feasibility Study on Concept Drift in Empirical Software Engineering

---

Submitted To:
Dr. Jacky Keung
Department of Computer Science
City University of Hong Kong

Submitted By:
Md Alamgir Kabir
ID:55340650
Email:makabir4-c@my.cityu.edu.hk
Department of Computer Science

Contents

1 Motivation and Problem Identification

In these digital days, we are encompassed by large number of applications which are generating data at a surprising rate. This huge number of incoming data from different sources is referred to as streaming data [18]. The behavior of these streaming data is changing over time. The changes in the distribution or pattern or concept over time is referred to as concept drift [5]. In other word, concept drift refers as a time changing in probability distribution in the high speed data streams [15]. In this situation, the statistical properties of target variables or attributes changes over time. In predictive analytic and machine learning, the performance of prediction model degrades due to the concept drift [6]. According to Dong et al., if the incoming data starts to drift, a well-trained static leaning model could give more inaccurate output by losing its accuracy and the static learning model becomes outdated [6], [15].

Besides, learning under concept drift is very difficult. It is essential to know when, where, and how the concept is changes in a particular case. Over the years, many methods have been developed to identify the drift in non-stationary environments[6]. Among of them, the methods, which generates a signal when drift is identified in the datasets, are very famous in concept drift literature [6] that look over the data distributions and error-rate of the learner. The most widely used strategy of drift detection is concept drift reaction which updates the models [17]. When drift is occurred, it retains the models again based on the recent data that saved on cache memory. But, it won't work if the cache is full or insufficient [17]. Concept drift is a one-cut process while the data of new concept is used for updating the learners, the old concept data is discarded. The concept drift may occur within some specific regions [6].

Furthermore, being the environments non-stationary it exhibits class imbalance that causes the practical problem in classification i.e., one of the most major problem in data mining and machine learning[13]. It requires a lot of effort to observe the new minority class instances in new incoming data. If an instance changes in posterior distribution, the old distributions are still to increase the number of minority class instances[13]. The techniques of incremental learning in imbalance datasets (i.e.,

limited label data and plenty unlabeled data) with semi-supervised learning are more complex compare to the techniques of fixed distributions [5].

1.1 Implications in Empirical Software Engineering

In the field of empirical research in software engineering, many prediction model are developed by using historical datasets (i.e., fixed distributions). Many methods have been proposed over for years for predicting the numbers of bugs and also the location of the upcoming or future bugs in software source codes [3]. This types of prediction strategy can help the manager or leader of the project to lead the project by proper utilizing the testing resources knowing the location of predicted defects.

Unfortunately, in the field of software effort estimation and defect prediction, the prediction model may not work if the historical datasets changes overtime [8]. The accuracy of prediction model will deteriorate. Relying on these prediction models will be very risky for a project manager of software[9]. Wrong prediction may trouble to reach the goal of the software. Ekanayake et al. investigated the cause why the quality of prediction rising and and falling due to the fact of altering nature of the defect (or bug) fixing process [8]. Due to the change of some influencing features, bug generation process becomes more unsuitable. They adopted the notion of concept drift in bug prediction. They investigated the four famous open source projects. They collected the bug reports from the bug reporting repositories, Bugzilla and CVS (Concurrent Versioning Systems). The projects are Eclipse, open office, Netbeans, and Mozilla. They used the algorithm of a tree induction and linear regression model to observe the changes in streaming data. They observed that if the authors' number changes in a projects for fixing bugs, the number of bugs fixed by them that indicates the concept drift. Therefore, it influences the quality of defect prediction[9]. It reinforces the work of Ekanayake et al. [9] as follows:

> *"defect prediction quality varies over time exhibiting periods of stability and variability"* [9].

These literature works i.e., [8, 9, 3, 10] demand to recheck the prediction models that have been developed over the years by using streaming historical data. It

reveals bewilderment about the quality of defect prediction models. It also indicates the notion and implications of concept drift in the research of empirical software engineering.

2 Concept Drift

Social applications, sensor-based systems, network embedded systems have been used very frequently resulting intense amount of data. The data is changing over time. The pattern or concept is also changing in every moment [6]. The challenge is to learn from the data that changes over time. Even, it is essential to know when and how the concept or pattern has changed [15]. The formula of concept drift is explained by Gama et al. [12] with the time t_0 and t_1 as given below:

$$D_{t_0}(A, b) \neq D_{t_1}(A, b) \tag{1}$$

Where, D_{t_0} explains the distribution at the time t_0 between the input data X in its corresponding label y. Again, D_{t_1} prevails the distribution at the time t_0 between the input data X in its corresponding label y. These two distributions are not same in the context of pattern or concept.

2.1 Categories of Concept Drift

However, concept drifts can be classified into are four types [15, 17, 12]. These are incremental, sudden, reoccurring concepts, and gradual [15].

The figure 1 is self explanatory that describes the four types of concept drift [1].

[1]The figure 1 is extracted from the paper of Dong et al. [7] where an active ensemble method is presented to deal with concept drift.

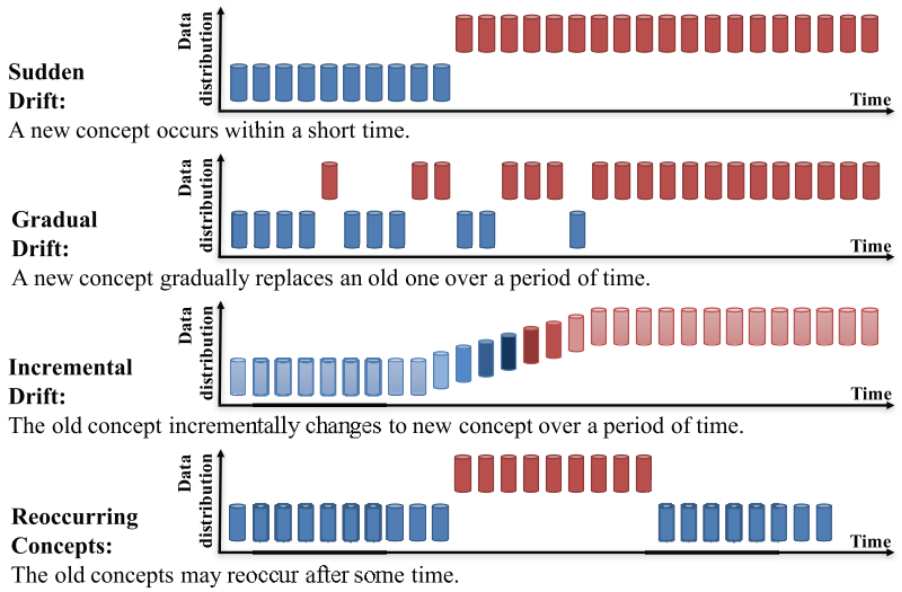

Figure 1: The types of concept drift [7]

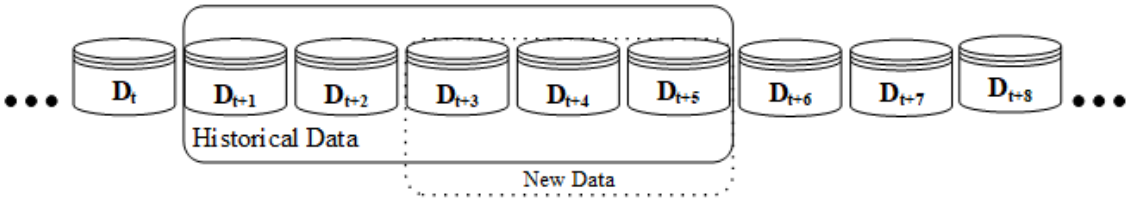

Figure 2: Two windows time where new data is defined by user

3 Concept Drift Detection

3.1 Concept Drift Detection Framework

The framework contains 4 stages i.e., data retrieval, data modeling, statistical test calculation, and hypothesis test. This framework is adpapted from Jei Lu et. al. [16]

Data Retrieval: In the stage of data retrieval, data chunks are retrieve from the streaming the data. The key features are retrieved from the data chunks in the stage of data modeling.

Data Modeling: As an optional stage 2, data modeling stage concerns about the

reduction of sample size.

Test Statistics Calculation: The stage 3 is very important. In this stage, the similarity and dissimilarity are measured of the datasets. The selection of the similarity or dissimilarity measures is still a challenging task in the area of drift detection. The last stage is hypothesis test. In this stage, the hypothesis test is evaluated to observe the statistical significance of the measures selected in stage 3 or the p-value.

Hypothesis Test: Without the stage 4, we cannot say that the statistical tests applied in stage 3 are helpful for drift detection. With the stage 4, it guarantees that drift is really occurred. Without this stage, it seems that the drift may be occurred because of noisy data or random sample selection. The famous hypothesis tests are bootstrapping, test statistics, permutation tests, and Hoeffding's test.

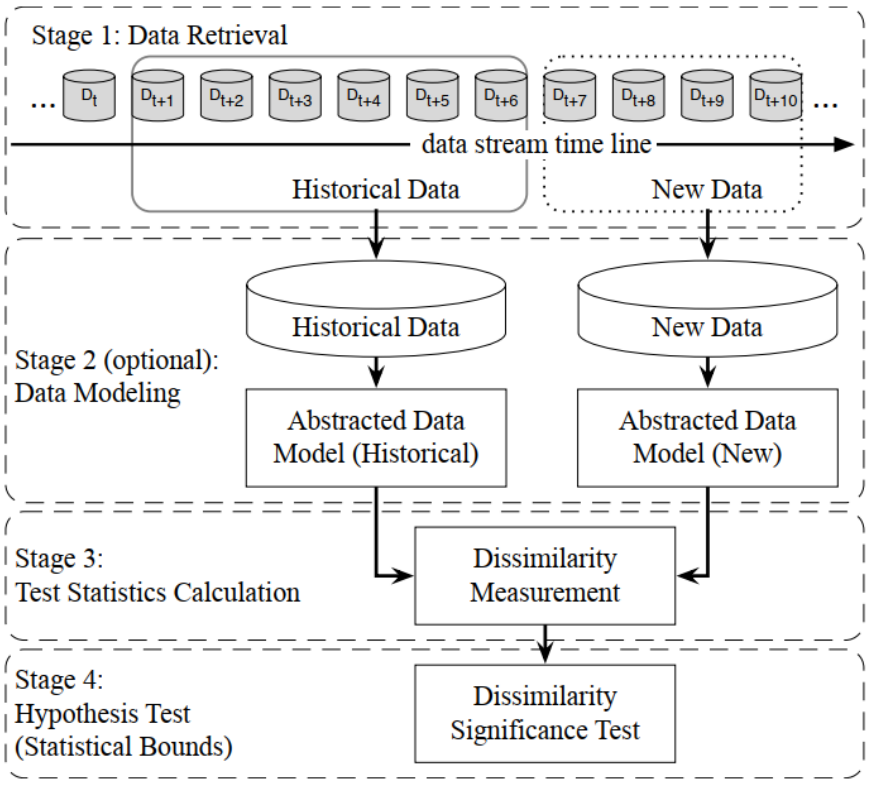

Figure 3: Concept drift detection framework [16]

3.2 Concept Drift Detection Methods

With the help of case-based techniques, David et al. [1]endeavor was Instance-based (IB) learning to handle the concept drift. Later, Lu et al. discovered that their attempt was to handle sudden drift [17]. However, The concept drift detection method is categorized into three ways [17]. These are drift detection by monitoring (1) data distributions, (2) parameters of learners, and (3) prediction errors of learners. The brief description of drift detection method is given below:

3.2.1 By monitoring error-rate of learning algorithms

1. Tracking changes in the online error rate of base classifiers
2. Increase or decrease of the error rate is proven to be statistically significant, an upgrade process (drift alarm) will be triggered
3. Stage 1: landmark time window is used. Landmark time window-
4. Stage 2: Training model to learning model
5. Stage 3: The statistics test constitute the online error rate
6. Stage 4 (Hypothesis test): Estimation the distribution of the online error rate and also the calculation the warning level and drift threshold.

In this section, we have included the well-accepted drift detection method, Drift Detection Method (DDM).

Drift Detection Method (DDM) [11]: They studied the learning problem under this situation, distributions changes over time. They proposed a method to detect the changes in probabilities distributions. Their idea is to control the online error rate-based technique to detect the changes in probabilities. By using the statistical analysis, they claimed that if the distributions are stationary, the error will diminish. The error become make greater in size, if the distribution is non-stationary or it changes in times. Their developed method monitors the online error rate. If the distributions changes over time, the error rate becomes greater or reaches their declared warning level, K_w. This is a sign of changes in distributions. They have tested their method with eight datasets and three learning algorithms i.e., perception, neural network, and decision tree. In machine learning, the concept drift is managed or handled with time windows or weighted examples. The approaches that

Table 1: Concept Drift Detection by monitoring data distributions

SN	Statistical Tests	Data Dimension	Findings
1	Wilcoxon and Kolmogorov-Smirnov tests	One-dimensional data	Limitations of scalability [4]
2	Modified Kolmogorov-Smirnov tests	Multi-dimensional data	Finding classes in higher dimensions [14]
3	Kullback–Leibler divergence and percentile bootstrap statistical method	N/A	Detected regions may not explained easily [4, 17]

handle concept drift are classified into two ways or categories i.e., the learners with considering changes and without considering changes. If the error rate of the algorithm increases, the class distribution changes.

3.2.2 By monitoring data distributions

For monitoring the data distributions, there have been applied some statistical tests in literature. Comparing two sample that are drawn from the same sample, the famous non-parametric methods, wilcoxon and Kolmogorov-Smirnov tests are applied in one dimensional data [17]. These tests are not suitable for multi-dimentional data due to the limitation of scalability [4]. Table 1 describes the concept drift detection methods by monitoring data distributions.

3.3 Effect of Ensemble Classifiers for Drift Detection

Data stream mining plays an important role in recent days. The main challenge in data streaming mining is concept drift which means the pattern or concept of the incoming data is changing is over time. According to Nagendran et al., [22], the ensemble classifiers works well in the streaming data. They have conducted a survey on online, block-based ensemble, and hybrid learning approaches [22] by comparing accuracy, memory, and time on different streaming datasets.
Unfortunately, tradition learning algorithms do not well dealing to concept drift in streaming datasets.

4 Bellwether Effect and Concept Drift

Window size (relevant number) and window age (elapsed time). These two things have occurred difficulty or effect the prediction accuracy for software effort estimation or empirical software engineering. [20, 21]
Bellwether moving window means the selection of relevant and recently completely projects
There is the need to consider the best selection of training and validation sets prior to modeling[19].
Selection of training and validation sets from a subset of chronologically arranged projects can further improve the prediction accuracy
Window size: The number of most recently completed historical projects
Window age: The elapsed time of projects (within the moving window)
Motivation: Current study requires a comparative study with bellwether effect and concept drift.

5 Evaluation

In this section, the experiment setup is discussed. This section also discusses the test statistic and more about the datasets that have been considered in this experiment.

5.1 Experimental Design

5.1.1 Dataset Descriptions

52 release of 14 open source projects were employed in this experiments. Table 2 and 3 describes about the datasets.

Table 2: Summary of selected 34 datasets

Project Name	Metric	Release	Module	# Defects
Ant 1	Static	1.3	125	20
Ant 1	Static	1.4	178	40
Ant 2	Static	1.5	293	31
Ant 2	Static	1.6	351	92
Ant 2	Static	1.7	745	166
Ivy1	Static	1.1	111	63
Ivy1	Static	1.4	241	16
Ivy2	Static	2	352	40
Jedit3	Static	3.2	272	90
Jedit4	Static	4	306	76
Jedit4	Static	4.1	312	79
Jedit4	Static	4.2	367	48
Jedit4	Static	4.3	492	11
Log4j1	Static	1	135	34
Log4j1	Static	1.1	109	37
Log4j1	Static	1.2	205	189
Lucene2	Static	2	195	91
Lucene2	Static	2.2	247	144
Lucene2	Static	2.4	340	203
Poi1	Static	1.5	237	141
Poi2	Static	2	314	37
Poi2	Static	2.5	385	248
Poi3	Static	3	442	281
Velocity1	Static	1.4	196	147
Velocity1	Static	1.5	214	142
Velocity1	Static	1.6	229	78
Xalan2	Static	2.4	723	110
Xalan2	Static	2.5	803	387
Xalan2	Static	2.6	885	411
Xalan2	Static	2.7	909	898
Xerces1	Static	1	162	77
Xerces1	Static	1.2	440	71
Xerces1	Static	1.3	453	69
Xerces1	Static	1.4	588	437

Table 3: Summary of selected 18 datasets

Project Name	Metric	Release	Module	# Defects
Cm1	Static	1	327	42
Cm1	Static	1.2	344	42
Cm1	Static	1.3	327	42
Jm1	Static	1	7782	1672
Jm1	Static	1.2	9593	1759
Jm1	Static	1.3	7782	1672
Kc3	Static	3	194	36
Kc3	Static	3.2	200	36
Kc3	Static	3.3	194	36
Mc1	Static	1	1988	46
Mc2	Static	2	125	44
Mc1	Static	1.2	9277	0
Mc1	Static	1.3	1988	46
Mc2	Static	2.2	127	44
Mc2	Static	2.3	125	44
Mw	Static	1	253	27
Mw	Static	1.2	237	27
Mw	Static	1.3	226	27

5.1.2 Prediction Models

In this experiment, I have considered naive bayes and decision tree classifiers to monitor the error-rate of the learners. Concept drift literature suggests that if the error-rate fluctuates during the learning time, the concept drift happens in that datasets.

5.1.3 Performance Measures

Most of the classification of defect prediction models are evaluated using confusing matrix. In the matrix, the defect-prone class is considered as positive instance and non defect-prone class is classified as negative instance. From table 5.1.3, the outcomes are described as follows:

- True Positives (TP): the number of defective modules predicted as defective.
- False Positives (FP): the number of non-defective modules predicted as defective.

Table 4: Confusion Matrix

	Positive Prediction	Negative Prediction
Actual Positive	TP	FN
Actual Negative	FP	TN

- True Negatives (TN) : the number of non-defective modules predicted as non-defective.
- False Negatives (FN) : the number of defective modules predicted as non-defective.

In this study, two evaluation measures, F-measure and AUC (Area Under the ROC Curve) are adopted to check the performance of all the methods of conventional learners and semi- supervised learners. F-measures computes the harmonic mean of precision and recall [2]. The performance measure, the Area Under the ROC Curve (AUC) are computed from the Receiver Operating Characteristics (ROC). It handles the trade-offs between error rates of the true and false positives. The details description of these measures can be found in [3] and [2].

$$Recall(R) = \frac{TP}{TP + FP} \tag{2}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{3}$$

$$F = \frac{2PR}{P + R} \tag{4}$$

High values of F-measures prevails the better performance. High AUC denotes the classifier produces good classification of the training datasets. The mathematical definitions of these measures are presented in Eqs. 2 to 4.

5.2 Statistical test

For detecting drift, the well accepted statistical t-test is adapted to conduct the hypothesis test in this experiment. The chi-square test with Yates's continuity correction test [23] is adapted in this experiment. The statistic test is performed by the follwing:

$$T(r_o, r_r, n_o, n_n) = \frac{|a| - 0.5b}{\sqrt{\widehat{p}(1-\widehat{p})b}} \tag{5}$$

where,

$$a = |r_o/n_o - r_r/n_r|,$$

$$b = (1/n_o + 1/n_r),$$

and

$$\widehat{p} = (r_o + r_r)/(n_o + n_r).$$

The test is performed by following the equation 5. The test value is compared with the significant value (P-value). In equation 5, the r_o is the number of correct classifications among the overall examples, n_o. r_r is the number of correct classifications among the recent examples, n_r. If the P-value is less than the significant level (0.05), the null hypothesis is rejected and alternative hypothesis is accepted.

6 Results and Analysis

Table 9 and 10 illustrates the accuracy of the considered datasets. There have been considered *JM* datasets of Tim Menzies and *prop* datasets of Jureczko. Figure 4 and 5 shows the error data with the number of training examples. We conduct hypothesis test assuming the data of two window times. We observe the significant level (P-value) of the datasets, *JM* and *prop*. Table 5 to 8 describes the p-value of the training examples of two window times.

By assuming 95% confidence, the observed p-values of the training examples of 3000

Table 5: p-value of the naive bayes classifier on jm data

Overall Examples	Recent Examples	p-value
500	1000	0.263804
1000	1500	0.248608
1500	2000	0.295549
2000	2500	0.717368
2500	3000	0.10155
3000	3500	**0.027421**
3500	4000	0.121493
4000	4500	**0.041416**
4500	2657	0.312574

Table 6: p-value of the decision tree classifier on jm data

Overall Examples	Recent Examples	p-value
500	1000	0.263804
1000	1500	0.248608
1500	2000	0.295549
2000	2500	0.717368
2500	3000	0.10155
3000	3500	**0.027421**
3500	4000	0.121493
4000	4500	**0.041416**
4500	2657	0.312574

to 3500 and 4000 to 4500 less than the significant level that reject null hypothesis and accept alternative hypothesis for the dataset, *JM*. For the dataset, *prop*, the null hypothesis is rejected and alternative hypothesis is accepted for the two window training example of 5000 to 5500, 5500 to 6000, and 6500 to 7000. In this training eamples, the concept drift happens and statistically proved. The table 7 and 8 illustrate the p-value of the two window training examples.

7 Future Plan

Current studies demand to do the several experiments and further study on drift detection methods. Here, a list is given below:

Table 7: p-value of the naive bayes classifier on prop data

Overall Examples	Recent Examples	p-value
500	1000	0.571133
1000	1500	0.382681
1500	2000	0.206242
2000	2500	0.090992
2500	3000	0.118753
3000	3500	0.086954
3500	4000	0.511022
4000	4500	0.094793
4500	5000	0.163062
5000	5500	**0.029752**
5500	6000	**0.027926**
6000	6500	0.207741
6500	7000	**0.03753**
7000	7500	0.06989
7500	8000	0.060321
8000	2053	0.063944

Table 8: p-value of the decision tree classifier on prop data

Overall Examples	Recent Examples	p-value
500	1000	0.519514
1000	1500	0.497514
1500	2000	0.151435
2000	2500	0.084665
2500	3000	0.065097
3000	3500	0.0562
3500	4000	0.315242
4000	4500	0.089983
4500	5000	0.145608
5000	5500	**0.033338**
5500	6000	**0.029807**
6000	6500	0.147845
6500	7000	**0.04183**
7000	7500	0.079372
7500	8000	0.074134
8000	2053	0.095129

Table 9: Accuracy measures per prediction model in metric-type jureczko dataset

Datasets	Naive Bayes	Decision Tree
ant.1.3	0.8571	0.7778
ant.1.4	0.6854	0.7079
ant.1.5	0.9041	0.8836
ant.1.6	0.7543	0.72
ant.1.7	0.7849	0.7527
ivy.1.1	0.6364	0.6182
ivy.1.4	0.9339	0.9091
ivy.2.0	0.8523	0.8636
jedit.3.2	0.75	0.6618
Jedit.4.0	0.7829	0.6974
Jedit.4.1	0.7756	0.6731
Jedit.4.2	0.8743	0.8251
Jedit.4.3	0.9756	0.9715
Log4j-1.0	0.75	0.7206
Log4j-1.1	0.8182	0.6909
Log4j-1.2	0.9029	0.8932
Lucene-2.0	0.6082	0.4742
Lucene-2.2	0.626	0.6016
Lucene-2.4	0.6882	0.6176
poi.1.5	0.6891	0.6723
poi.2.0	0.8544	0.8418
poi.2.5	0.7668	0.6891
poi.3.0	0.7793	0.6667
Velocity-1.4	0.8367	0.7347
Velocity-1.5	0.7196	0.729
Velocity-1.6	0.7281	0.6667
xalan-2.4	0.7956	0.8177
xalan-2.5	0.6035	0.5461
xalan-2.6	0.7127	0.638
xalan-2.7	0.978	0.989
xerces-1.0	0.6463	0.6463
xerces-1.2	0.7955	0.8318
xerces-1.3	0.837	0.815
xerces-1.4	0.8299	0.9218

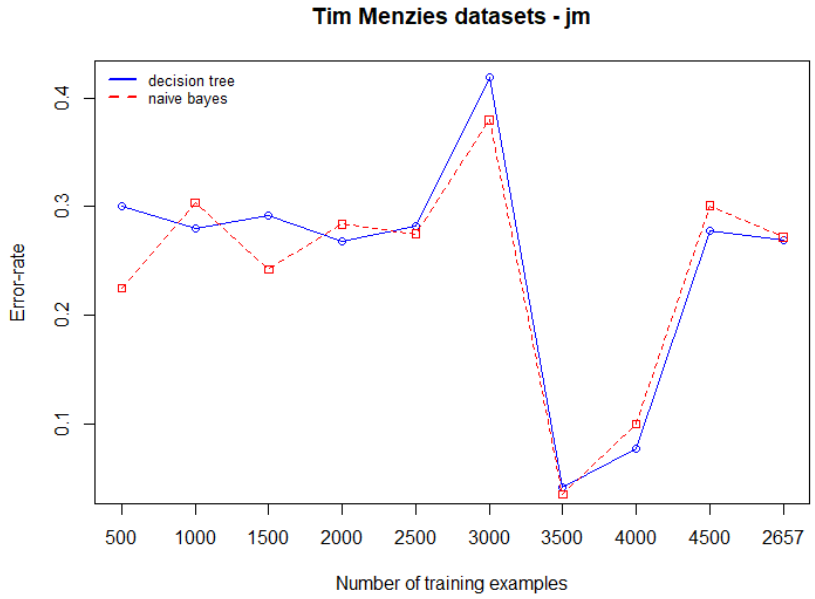

Figure 4: Observation of the error-rate of the learning algorithms for the datasets, jm

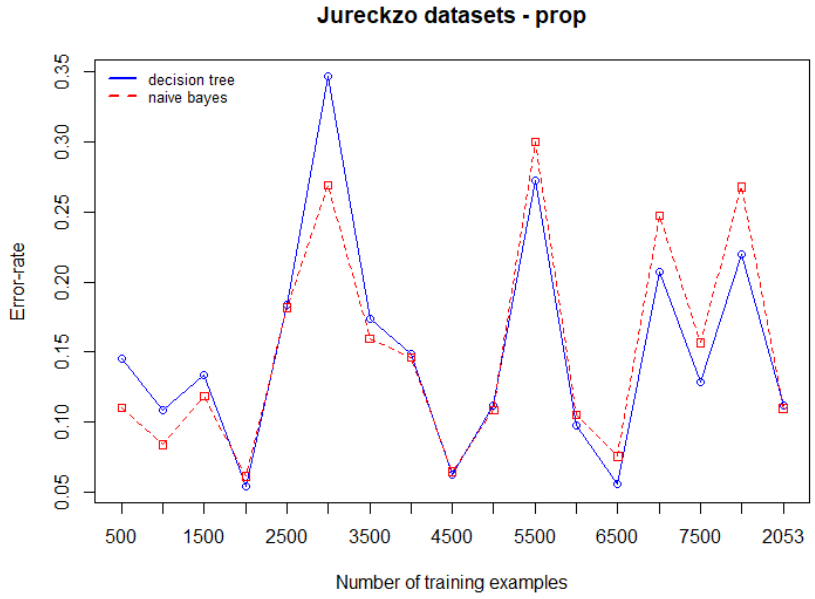

Figure 5: Observation of the error-rate of the learning algorithms for the datasets, prop

Table 10: Accuracy measures per prediction model in metric-type mccabe dataset

Datasets	Naive Bayes	Decision Tree
cm1	0.8476	0.8354
cm12	0.8605	0.8314
cm13	0.8537	0.8476
jm1	0.6746	0.7129
jm12	0.6797	0.7706
jm13	0.6746	0.7129
kc3	0.7938	0.7113
kc32	0.77	0.71
kc33	0.7938	0.7113
mc1	0.9708	0.9698
mc2	0.6667	0.6349
mc12	0.9457	0.9918
mc13	0.9708	0.9698
mc22	0.6349	0.5079
mc23	0.6667	0.5714
mw1	0.8889	0.8254
mw12	0.8788	0.8333
mw13	0.8889	0.8254

1. Explore the concept drift detection methods by monitoring the parameters of learners and also find the limitations of the methods.
2. Explore the methods of concept drift detection by monitoring the prediction errors of learners and also find the limitations of it.
3. Being the environments non-stationary, it exhibits class imbalance[13]. Our plan is also to explore the techniques that handle class imbalance in streaming data.
4. Prepare a well trained prediction model in streaming software engineering datasets. We have planned do the experiments on open source projects.
5. Conduct an experiment to find the difference between bellwether effect and concept drift. Current studies require a comparative between concept drift and bellwether moving window.

8 Conclusions

Software defect prediction models are developed based on the historical data. If the historical data changes in times, the prediction models outdated. Even, a well trained prediction model could give more inaccurate results, if the historical data changes over time.

The target of our experiment is to investigate to check the concept drift existence in software defect datasets. Based on our experiment, we observe that the concept drift exits in the defect datasets. In the dataset, *JM* and *prop*, the statistical test proves that the concept drift happens in these datasets. The prediction models would not give accurate results if the training examples are taken from these datasets.

References

[1] David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[2] Kwabena Ebo Bennin, Jacky W Keung, and Akito Monden. On the relative value of data resampling approaches for software defect prediction. *Empirical Software Engineering*, pages 1–35, 2018.

[3] Abraham Bernstein, Jayalath Ekanayake, and Martin Pinzger. Improving defect prediction using temporal features and non linear models. In *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, pages 11–18. ACM, 2007.

[4] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

[5] Gregory Ditzler and Robi Polikar. Semi-supervised learning in nonstationary environments. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2741–2748. IEEE, 2011.

[6] Fan Dong, Jie Lu, Kan Li, and Guangquan Zhang. Concept drift region identification via competence-based discrepancy distribution estimation. In *Intelligent Systems and Knowledge Engineering (ISKE), 2017 12th International Conference on*, pages 1–7. IEEE, 2017.

[7] Fan Dong, Jie Lu, Guangquan Zhang, and Kan Li. Active fuzzy weighting ensemble for dealing with concept drift. *Int. J. Comput. Intell. Syst.*, 11(1):438–450, 2018.

[8] Jayalath Ekanayake, Jonas Tappolet, Harald C Gall, and Abraham Bernstein. Tracking concept drift of software projects using defect prediction quality. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 51–60. IEEE, 2009.

[9] Jayalath Ekanayake, Jonas Tappolet, Harald C Gall, and Abraham Bernstein. Time variance and defect prediction in software projects. *Empirical Software Engineering*, 17(4-5):348–389, 2012.

[10] Jacqui Finlay, Russel Pears, and Andy M Connor. Data stream mining for predicting software build outcomes using source code metrics. *Information and Software Technology*, 56(2):183–198, 2014.

[11] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[12] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44, 2014.

[13] Thomas Ryan Hoens and Nitesh V Chawla. Learning in non-stationary environments with class imbalance. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–176. ACM, 2012.

[14] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[15] Anjin Liu, Yiliao Song, Guangquan Zhang, and Jie Lu. Regional concept drift detection and density synchronized drift adaptation. In *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence*, pages 2280–2286. Accept, 2017.

[16] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[17] Ning Lu, Guangquan Zhang, and Jie Lu. Concept drift detection via competence models. *Artificial Intelligence*, 209:11–28, 2014.

[18] Shikha Mehta et al. Concept drift in streaming data classification: Algorithms, platforms and issues. *Procedia computer science*, 122:804–811, 2017.

[19] Solomon Mensah, Jacky Keung, Michael Franklin Bosu, Kwabena Ebo Bennin, and Patrick Kwaku Kudjo. A stratification and sampling model for bellwether moving window. In *SEKE*, pages 481–486, 2017.

[20] Solomon Mensah, Jacky Keung, Stephen G MacDonell, Michael F Bosu, and Kwabena E Bennin. Investigating the significance of bellwether effect to improve software effort estimation. In *Software Quality, Reliability and Security (QRS), 2017 IEEE International Conference on*, pages 340–351. IEEE, 2017.

[21] Solomon Mensah, Jacky Keung, Stephen G MacDonell, Michael Franklin Bosu, and Kwabena Ebo Bennin. Investigating the significance of the bellwether effect to improve software effort prediction: Further empirical study. *IEEE Transactions on Reliability*, 67(3):1176–1198, 2018.

[22] Nalini Nagendran, H Parveen Sultana, and Amitrajit Sarkar. A comparative analysis on ensemble classifiers for concept drifting data streams. In *Soft Computing and Medical Bioinformatics*, pages 55–62. Springer, 2019.

[23] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In *International conference on discovery science*, pages 264–269. Springer, 2007.