

# Automated Issue Assignment: Results and Insights from an Industrial Case

Ethem Utku Aktas · Cemal Yilmaz

Received: date / Accepted: date

**Abstract** *Softtech*, being a subsidiary of the largest private bank in Turkey, called *IsBank*, receives an average of 350 issue reports from the field every day. Manually assigning the reported issues to the software development teams is costly and cumbersome. We automate the issue assignments using data mining approaches and share our experience gained by deploying the resulting system at Softtech/IsBank. Automated issue assignment has been studied in the literature. However, most of these works report the results obtained on open source projects and the remaining few, although they use commercial, closed source projects, carry out the assignments in a retrospective manner. We, on the other hand, deploy the proposed approach, which has been making all the assignments since Jan 12, 2018. This presents us with an unprecedented opportunity to observe the practical effects of automated issue assignment in the field and to carry out user studies, which have not been done before in this context. We observe that it is not just about deploying a system for automated issue assignment, but also about designing/changing the assignment process around the system; the accuracy of the assignments does not have to be higher than that of manual assignments in order for the system to be useful; deploying such a system requires the development of additional functionalities, such as detecting deteriorations in assignment accuracies in an online manner and creating human-readable explanations for the assignments; stakeholders do not necessarily resist change; and gradual transition can help stakeholders build confidence.

---

Ethem Utku Aktas  
Softtech Inc., Research and Development Center,  
34947 Istanbul, Turkey  
E-mail: utku.aktas@softtech.com.tr

Cemal Yilmaz  
Faculty of Engineering and Natural Sciences,  
Sabanci University,  
34956 Istanbul, Turkey  
E-mail: cyilmaz@sabanciuniv.edu

**Keywords** Bug Triaging · Issue Report Assignment · Text Mining · Machine Learning · Accountable Machine Learning · Change Point Detection

## 1 Introduction

IsBank<sup>1</sup> is the largest private bank in Turkey with 7.5 million digital customers, 25 thousand employees, 6566 ATMs (Automated Teller Machines), and 1314 domestic and 22 foreign branches, providing a large variety of banking and financial services.

Softtech<sup>2</sup> is an ISO-9001-certified subsidiary of IsBank and the largest software company of Turkey owned by domestic capital, providing customer-oriented, business-critical solutions to IsBank by using universally-recognized lean techniques and agile processes with a diverse set of programming languages, platforms, and technologies, including COBOL, Java, C#, C++, main-frame platforms, mobile/wearable platforms, security- and privacy-related technologies, natural language processing technologies, speech technologies, image/video processing technologies, and artificial intelligence technologies.

When the wide range of software systems maintained by Softtech couple with the large user base owned by IsBank, who depend on these systems to carry out their day-to-day businesses, Softtech receives an average of 350 issue reports from the field every day (around 90 thousand reports per year). The reported issues range from bank clerks having software failures to bank customers facing software-related problems in any of the bank channels, including online, mobile, and ATM. Note that what we refer to as *issue reports* in this work are also often referred to as *bug reports* and *problem reports* in the literature.

Most of the reported issues concern business-critical software systems. Therefore, both Softtech and IsBank need to handle these issues with utmost importance and urgency. To this end, two dedicated teams of employees are employed: one at IsBank with 50 full-time clerks, called *IT Help Desk* (IT-HD), and the other at Softtech with 30 full-time employees, called *Application Support Team* (AST). IT-HD clerks collect the issues reported from the field, resolve the ones that they can (such as the issues addressed by basic troubleshooting guides), and if not, assign them to the software development teams at Softtech. AST members, on the other hand, being embedded in the development teams at Softtech, help the development teams manage the issue reports assigned to them.

An integral part of this process is to assign the issue reports to the right development teams, i.e., the ones that are responsible for resolving the reported issues. In the remainder of the paper, we refer to this task as *issue report assignment* (or *issue assignment*, in short).

It turned out that the manual issue assignments carried out by the IT-HD clerks were costly and cumbersome, mainly due to the large number of issues

<sup>1</sup> <https://www.isbank.com.tr>

<sup>2</sup> <https://softtech.com.tr>

received on a daily basis and the relatively large number of development teams that these issues should be assigned to (an average of 53 development teams at any point given in time). Furthermore, incorrect assignments, were not only causing friction between IT-HD and AST, but also increasing the turnaround time for resolutions as the respective issue reports tended to bounce back and forth between IT-HD and AST until the right development team was located.

In this work, we automate the process of issue assignment by using data mining approaches and share our experience gained by deploying the resulting system, called *IssueTAG*, at IsBank and Softtech, which has been making all the issue assignments since its deployment on Jan 12, 2018.

Automated issue assignment is indeed not a new idea (Murphy and Cubranic 2004; Anvik et al. 2006; Wang et al. 2008; Bhattacharya et al. 2012; Jonsson et al. 2016; Dedík and Rossi 2016). Most of the existing works, however, report the results obtained on open source projects, such as Eclipse, Mozilla, and Firefox (Murphy and Cubranic 2004; Anvik et al. 2006; Wang et al. 2008; Bhattacharya et al. 2012). Our work differs from these works in that we present an industrial case where we use the issue reports filed for commercial, closed-source software systems.

We, furthermore, assign issue reports to development teams, rather than to individual developers – a decision we made based on our discussions with the IT-HD and AST teams. The former is more practical and realistic in industrial setups, because the latter does not take into account 1) the current workloads owned by the individual developers, 2) the changes in the team structures, such as the developers leaving or joining the teams, and 3) the current status of developers, such as the developers who are currently on leave of absence. Therefore, especially in the presence of close-knit development teams, which is the case with Softtech, assigning issue reports to the development teams help the teams make more educated decisions as to which team member the report should be addressed by.

Moreover, rather than carrying out the issue assignments in the context of a single product, such as Eclipse, Mozilla, and Firefox, where the incoming issues are assigned to individual software engineers working on the product, we do the assignments at the level of an entire company (Softtech), which has 489 software products comprised of around 100 millions of lines of code (as of Feb 3, 2019). That is, we assign issue reports filed for any product owned by Softtech to the development teams responsible for resolving the reported issues. This is challenging because with the collection of software products maintained by Softtech, which heavily interact with each other in a business-critical environment by sharing many resources, such as databases, file systems, and GUI screens, the boundaries of the products from the perspective of issue reporting and management are not clear at all.

There are only few recent studies reporting the results obtained on closed-source, commercial software projects (Jonsson et al. 2016; Dedík and Rossi 2016; Lin et al. 2009; Helming et al. 2010). These studies, however, carry out the assignments in a retrospective and offline manner by simply treating the actual issue databases as historical data. We have, on the other hand, deployed

IssueTAG, which presented us with an unprecedented opportunity to observe the practical effects of automated issue assignment in practice as well as to carry out user studies, which (to the best of our knowledge) have not been done before in this context.

First, we observed that it is not just about deploying a data mining-based system for automated issue assignment, but also about designing/changing the assignment process around the system to get the most out of it. We, in particular, made simple, yet effective changes in the manual issue assignment process employed at IsBank and Softtech (Section 5).

Second, the accuracy of the assignments does not have to be higher than that of manual assignments in order for the system to be useful, which is further validated by the user studies we carried out on actual stakeholders in the field (Section 6). In a nutshell, although the daily assignment accuracy of IssueTAG was slightly lower than that of manual assignments (0.831 vs. 0.864), it reduced the manual effort required for the assignments by about 5 person-months per year and improved the turnaround time for resolving the reported issues by about 20% (Section 5.2.3). Furthermore, about 79% of the stakeholders participated in our user study “agreed” or “strongly agreed” that the system was useful (Section 6).

Third, we observed that deploying a data mining-based approach for automated issue assignments, requires the development of additional functionalities, which are not necessarily foreseen before the deployment. We have, in particular, developed two additional functionalities, both of which, to the best of our knowledge, have not been evaluated before in the context of issue assignment. One functionality we needed was to monitor the assignment accuracy of the system and detect deteriorations in an online manner, so that corrective actions, such as recalibrating the models, can be taken in time. To this end, we have developed a *change point detection*-based approach (Section 8). Another functionality we needed, which we also did not foresee before the deployment of IssueTAG, was to create human-readable, non-technical explanations for the assignments made by the system. This was indeed a need we came to realize when we received several phone calls from the stakeholders shortly after the deployment of IssueTAG, demanding explanations as to why certain issue reports (especially, the incorrect-assigned ones) were assigned to them. Note that this is not a trivial task at all, especially when the underlying data mining models are not human readable. To this end, we have generated model-agnostic explanations (Ribeiro et al. 2016) and carried out a user study to evaluate the quality of these explanations (Section 8).

Last but not least, we observed that stakeholders do not necessarily resist change. In particular, we did not receive any objection at all to the deployment of IssueTAG. We believe that this was because all the stakeholders believed that they would benefit from the new system and none of them felt threatened by it (Section 9). We, furthermore, observed that gradual transition helped stakeholders build confidence in IssueTAG, which, in turn, facilitated the acceptance of the system (Section 9).

More specifically, the research question we address in this work are:

- RQ1: How do the existing approaches for automated issue assignment compare with each other?
- RQ2: How do the amount and time locality of training data affect the assignment accuracies?
- RQ3: How does automated issue assignment compare to manual issue assignment in practice?
- RQ4: Is IssueTAG perceived as useful by the end-users?
- RQ5: Can the issue assignments made by the underlying data mining model be explained in a non-technical manner?
- RQ6: Can the deteriorations in the assignment accuracies be automatically detected in an online manner?

The remainder of the paper is organized as follows: Section 2 describes the issue assignment process employed at IsBank and Softtech before the deployment of IssueTAG; Section 3 evaluates the existing approaches for automated issue assignment on the collection of issue reports maintained by IsBank and Softtech (RQ1); Section 4 presents the empirical studies we carried out to determine the amount and time locality of the training data required for training/recalibrating the underlying data mining models(RQ2); Section 5 deploys IssueTAG and evaluates its effects in practice (RQ3); Section 6 carries out a user study on the end users of IssueTAG to evaluate whether the deployed system is perceived as useful (RQ4); Section 7 presents an approach for automatically generating explanations for the assignments and evaluates it by conducting a user study (RQ5); Section 8 describes and evaluates a change point detection-based approach for detecting deteriorations in assignment accuracies (RQ6); Section 9 presents lessons learnt; Section 10 discusses threats to validity; Section 11 presents related work; and Section 12 concludes with potential avenues for future work.

## 2 Case Description

At IsBank and Softtech, the issue reports, as they typically concern business-critical systems, are handled with utmost importance and urgency. To this end, two dedicated teams are employed, the sole purpose of which is to manage the reported issues, namely IT-HD (IT Help Desk) and AST (Application Support Team).

### 2.1 IT Help Desk

The IT-HD team is employed at IsBank and it consists of 50 full-time, (mostly) non-technical clerks, who are internally referred to as Level 1 employees, indicating the level of technical competency they have. When a bank employee or a bank customer faces with an IT-related issue, they call IT-HD on the

phone. The IT-HD clerk listens to the issue, collects the details as needed, records them, and resolves the reported issue right away if it is an issue that can be resolved by an IT-HD clerk, such as the ones documented in basic troubleshooting guides. If not, the clerk is responsible for dispatching the issue to the proper entity/unit in the company. In the case of a software-related issue, the clerk files an issue report to Softtech.

## 2.2 Issue Reports

An issue report, among other information, such as the date and time of creation, has two parts: a one-line summary and a description, both of which are written in Turkish. The former captures the essence of the issue, whereas the latter describes the issue, including the expected and observed behavior of the system, and provides information to reproduce the reported issue (Bettenburg et al. 2008a). Note that the aforementioned issue reports do not have any field conveying categorical information, such as product, component, and version information. The reason is that since the collection of software products maintained by Softtech are heavily interacting with each other in a business-critical environment, sharing many resources, such as databases, file systems, and GUI screens, the boundaries of the products/components from the perspective of issue reporting and management are not clear at all. For example, a single GUI screen can have multiple tabs, each of which is maintained by a different development team. A single tab can, in turn, have a number of widgets, each of which is under the responsibility of a different team. Almost all of the GUI screens interact with the core banking system, which is maintained by a different set of development teams. The core can be accessed via different banking channels, such as online, mobile, ATM, and SMS (Short Message Service), each of which has a dedicated set of development teams. Last but not least, financial transactions are typically carried out by using multiple GUI screens, widgets, and channels, crossing the boundaries of multiple development teams.

## 2.3 Application Support Team (AST)

The AST team is employed at Softtech and it consists of 30 full-time, Level 2 employees. That is, in terms of technical competency, the AST employees are somewhere between Level 1 IT-HD clerks and Level 3 software engineers. AST employees are embedded in development teams, which are consisted of software engineers. The same AST member can work with multiple development teams and a development team can have multiple AST members. The sole responsibility of an AST member embedded in a development team is to manage the collection of issue reports assigned to the team. When a new issue report is assigned to a development team, the AST member embedded in the team is typically the first one to examine the report. If the AST member can resolve the reported issue, he/she first resolves it and then closes the report

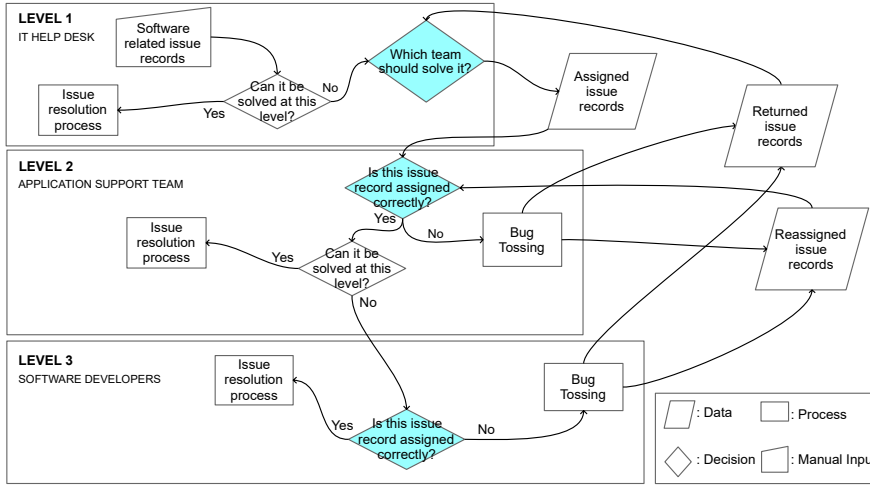


Fig. 1: Issue report assignment process before the deployment of IssueTAG.

on behalf of the team. Otherwise, the AST member notifies the development team about the newly reported issue by, for example, assigning it to a software engineer in the team or by creating a task for the team and linking it to the issue report. Note that AST members, although they are not considered to be software engineers, can still resolve some of the reported issues as not all of these issues may require changes in the code base. Some issues, for example, are resolved by running pre-existing scripts, which can automatically diagnose and fix the problems or by manually updating certain records in the database. Therefore, the ultimate goal of the AST members is to reduce the workload of software engineers by resolving the issues that do not require code changes.

## 2.4 Manual Issue Assignment Process

Before the deployment of IssueTAG, IT-HD clerks, after creating an issue report, was assigning it to a development team. To this end, they were maintaining a knowledge base, which was simply comprised of spreadsheets mapping certain keywords with development teams. In the presence of an incorrect assignment, although the AST member(s) or the software engineers in the respective development team could reassign the issue to a different team, the incorrectly assigned reports were often returned back to IT-HD for reassignment. Figure 1 summarizes the assignment process. The issue reports are managed by using Maximo<sup>3</sup> at IsBank and by using Jira<sup>4</sup> at Softtech.

<sup>3</sup> <https://www.ibm.com/products/maximo>

<sup>4</sup> <https://www.atlassian.com/software/jira>

## 2.5 Issues with the Manual Assignment Process

There were a number of issues with the aforementioned process. First, the learning curve for the IT-HD clerks (especially for the new hires) for excelling in team assignments was generally steep due to the large number of issue reports received on a daily basis (an average of about 350 issue reports) and the relatively large number of products and development teams present (more than 450 products and between 47 and 57 teams at any given point in time). Second, although IT-HD clerks were using a knowledge base to help with the assignments, it was maintained in an ad hoc manner, which was error prone, cumbersome, and time consuming. Last but not least, incorrect assignments were not only causing frictions between the IT-HD clerks and the AST members, but also increasing the turn around time for resolutions as the incorrectly-assigned issue reports would typically bounce back and forth between the IT-HD clerks and AST members (*bug tossing*) until the correct development team was located, which was causing a great deal of wasted time.

## 3 Evaluating Existing Issue Assignment Approaches

IsBank and Softtech wanted to improve their current practices. To this end, we, in this work, automate the issue assignments.

We start with investigating our first research question RQ1: “How do existing approaches for automated issue assignment compare with each other?” This is important as the results of this study will determine the approach to be used in the deployed system.

Note that our goal in this work is neither to propose yet another approach for automated issue assignment nor to evaluate all existing approaches to determine the best possible approach, but to identify an existing approach that can produce similar or better assignment accuracies with the manual assignment process employed at IsBank/Softtech and that can be developed and deployed with as little risk as possible. After all, most of the issue reports the system will process, concern business-critical software systems. Therefore, neither IsBank nor Softtech was willing to take too much risk.

### 3.1 Approach

To carry out the study, we have determined a number of approaches, which had been shown to be effective for automated issue assignment (Murphy and Cubranic 2004; Anvik et al. 2006; Bhattacharya et al. 2012; Anvik and Murphy 2011; Jonsson et al. 2016). We then empirically evaluated them by using the issue database, which has been maintained by Softtech since December 2016.

In particular, we cast the problem of issue assignment to a classification problem where the natural language descriptions in issue reports are analyzed by using various classification algorithms.



### 3.1.1 Representing Issue Reports

Given an issue report, we first combine the “description” and “summary” parts of the report, then tokenize the combined text into terms, and finally remove the non-letter characters, such as punctuation marks, as well as the stop words, which are extremely common words of little value in classifying issue reports (Manning et al. 2010), such as “the”, “a”, and “is.” We opt not to apply stemming in this work as an earlier work suggests that stemming has a little effect (if any at all) in issue assignments (Murphy and Cubranic 2004), which is also consistent with the results of our initial studies where stemming slightly reduced the assignment accuracies.

We then represent an issue report as an  $n$ -dimensional vector. Each element in this vector corresponds to a term and the value of the element depicts the weight (i.e., “importance”) of the term for the report. The weights are computed by using the well-known *tf-idf* method (Manning et al. 2010).

The *tf-idf* method combines two scores: term frequency (*tf*) and inverse document frequency (*idf*). For a given term  $t$  and an issue report  $r$ , the term frequency  $tf_{t,r}$  is the number of times  $t$  appears in  $r$ . The more  $t$  appears in  $r$ , the larger  $tf_{t,r}$  is. The inverse document frequency of  $t$  ( $idf_t$ ), on the other hand, is:

$$idf_t = \log\left(\frac{N}{df_t}\right), \quad (1)$$

where  $N$  is the total number of issue reports and  $df_t$  is the number of issue reports, in which  $t$  appears. The fewer the issue reports  $t$  appears in, the larger  $idf_t$  is.

Given  $tf_{t,r}$  and  $idf_t$ , the *tf-idf* score of the term  $t$  for the issue report  $r$  is computed as follows:

$$\text{tf-idf}_{t,r} = tf_{t,r} * idf_t. \quad (2)$$

Consequently, the more a term  $t$  appears in an issue report  $r$  and the less it appears in other issue reports, the more important  $t$  becomes for  $r$ , i.e., the larger  $\text{tf-idf}_{t,r}$  is.

### 3.1.2 Issue Assignments

Once an issue report is represented as an ordered vector of *tf-idf* scores, the problem of assignment is cast to a classification problem where the development team, to which the issue report should be assigned, becomes the class to be predicted and the *tf-idf* scores of the report become the attributes, on which the classification will be based on.

We train two types of classifiers: *level-0 classifiers*, each of which is comprised of an individual classifier, and *level-1 classifiers*, which were obtained by combining multiple level-0 classifiers using stacked generalization – an ensemble technique to combine multiple individual classifiers (Wolpert 1992).

All the classifiers we experiment with in this study have been shown to be effective for automated issue assignment (Murphy and Cubranic 2004; Anvik et al. 2006; Bhattacharya et al. 2012; Anvik and Murphy 2011; Jonsson et al. 2016).

For the level-0 classifiers, we use multinomial naive bayesian (Manning et al. 2010), decision tree (Breiman 2017), k-nearest neighbor (Manning et al. 2010), logistic regression (Bishop 2006), random forest (Breiman 2001), and linear support vector classifiers (SVCs) (Joachims 1998).

For the level-1 classifiers, we first train and evaluate our level-0 classifiers by using the same training and test sets for each classifier. We then use the prediction results obtained from these level-0 classifiers to train a level-1 classifier, which combines the probabilistic predictions of the level-0 classifiers using linear logistic regression (Wolpert 1992).

Inspired from (Jonsson et al. 2016), we, in particular, train two types of level-1 classifiers: *BEST* and *SELECTED*. The *BEST* ensemble is comprised of  $k$  (in our case,  $k = \{3, 5\}$ ) level-0 classifiers with the highest assignment accuracies, where as the *SELECTED* ensemble is comprised of a diversified set of  $k$  (in our case,  $k = \{3, 5\}$ ) level-0 classifiers, i.e., the ones with different representation and classification approaches, which are selected regardless of their classification accuracies, so that errors of individual classifiers can be averaged out by better spanning the learning space (Wolpert 1992). Note that the *BEST* and *SELECTED* ensembles are not necessarily the same because the best performing level-0 classifiers may not be the most diversified set of classifiers. More information on how these ensembles are created can be found in Section 3.2.

Furthermore, for the baseline classifier, which we use to estimate the baseline classification accuracy for our classifiers, we assign all issue reports to the team that have been assigned with the highest number of issue reports. That is, our baseline classifier always returns the class with the highest number of instances as the prediction.

### 3.2 Evaluation

We have conducted a series of experiments to evaluate the assignment accuracies of the level-0 and level-1 classifiers.

#### 3.2.1 Experimental Setup

In these experiments, we used the issue reports submitted to Softtech between June 1, 2017 and November 30, 2017 as the training set and the issue reports submitted in the month of December 2017 as the test set. We picked this time frame because it provided us with a representative data set in terms of the number of issue reports submitted, the number of teams present, and the distribution of the reported issues to these teams.

For the aforementioned time frame, we had a total number of 51,041 issue reports submitted to 65 different teams. Among all the issue reports of interest in this section as well as in the remainder of the paper, we only used the ones that were marked as “closed,” indicating that the reported issues had been validated and resolved. Furthermore, as the correct assignment for an issue report, we used the development team that had closed the report. The remainder of the issue reports were ignored as it was not yet certain whether these reports were valid or whether the development teams, to which they were currently assigned, were correct. After this filtering, a total of 47,123 issue reports submitted to 64 different development teams remained for analysis in this study.

To create the level-1 classifiers, we combined 3 or 5 individual classifiers, i.e.,  $k = 3$  or  $k = 5$ . We used the latter setting as it was also the setting used in a recent work (Jonsson et al. 2016). We used the former setting as it was the best setting we could empirically determine for ensemble learning, i.e., the one that produced the best assignment accuracies. In the remainder of the paper, these models are referred to as *BEST-3*, *SELECTED-3*, *BEST-5*, and *SELECTED-5*.

The BEST-3 and BEST-5 models were obtained by combining Linear SVC-Calibrated, Logistic Regression, and K-Neighbours; and Linear SVC-Calibrated, Logistic Regression, K-Neighbours, Random Forest, and Decision Tree classifiers, respectively, as these were the classifiers providing the best assignment accuracies. The SELECTED-3 and SELECTED-5 models, on the other hand, were obtained by combining Linear SVC-Calibrated, K-Neighbours, and Multinomial Naive Bayesian; and Linear SVC-Calibrated, Logistic Regression, K-Neighbours, Random Forest, and Multinomial Naive Bayesian classifiers, respectively, with the goal of better spanning the learning space by increasing the diversity of the classification algorithms ensembled. Note further that to include SVCs in level-1 classifiers, we used calibrated linear SVCs instead of linear SVCs as we needed to have class probabilities to ensemble individual classifiers (Ting and Witten 1999), which are not supported by the latter.

The classifiers were trained and evaluated by using the `scikit-learn` Python library (for level-0 classifiers) (Pedregosa et al. 2011) and `mlxtend` (for level-1 classifiers) (Raschka 2018) packages. All of the classifiers (unless otherwise stated) were configured with the default settings and the experiments were carried out on a dual-core Intel(R) Xeon(R) E5-2695 v4 2.10 GHz machine with 32 GB of RAM running Windows Server 2012 R2 as the operating system.

### 3.2.2 Evaluation Framework

To evaluate the quality of the assignments obtained from different classifiers, we used well-known metrics, namely *accuracy* and weighted *precision*, *recall*, and *F-measure* (Manning et al. 2010). Accuracy, which is also referred to as *assignment accuracy* in the remainder of the paper, is computed as the ratio

Table 1: Accuracy (A) and weighted precision (P), recall (R), and F-measure (F) values obtained from different classification models as well as the training times of these models.

classifier	using training set with 10-fold cross validation		using test set			
	A	training time	P	R	F	A
Baseline	0.10	-	0.01	0.12	0.03	0.12
Multinomial NB	0.47 (+/- 0.01)	31 s	0.70	0.52	0.50	0.52
Decision Tree	0.66 (+/- 0.02)	50 m 11 s	0.64	0.63	0.63	0.63
K-Neighbours	0.73 (+/- 0.02)	1 m 4 s	0.71	0.72	0.71	0.72
Logistic Regression	0.74 (+/- 0.01)	18 m 37 s	0.76	0.74	0.74	0.74
Random Forest	0.66 (+/- 0.02)	51 m 43 s	0.64	0.65	0.63	0.65
Linear SVC	<b>0.82 (+/- 0.01)</b>	3 m 32 s	<b>0.80</b>	<b>0.80</b>	<b>0.80</b>	<b>0.80</b>
Linear SVC-Calibrated	0.81 (+/- 0.01)	7 m 50 s	<b>0.80</b>	0.79	0.79	0.79
BEST-5	0.67 (+/- 0.02)	2 h 20 m 38 s	0.65	0.64	0.64	0.64
SELECTED-5	0.80 (+/- 0.01)	1 h 49 m 11 s	0.79	0.78	0.78	0.78
BEST-3	0.81 (+/- 0.01)	56 m 7 s	<b>0.80</b>	0.79	0.79	0.79
SELECTED-3	0.81 (+/- 0.01)	32 m 57 s	<b>0.80</b>	0.79	0.79	0.79

of correct issue assignments. Precision for a particular development team (i.e., class) is the ratio of the issue reports that are correctly assigned to the team to the total number of issue reports assigned to the team. Recall for a team is the ratio of the issue reports that are correctly assigned to the team to the total number of issue reports that should have been assigned to the team. F-measure is then computed as the harmonic mean of precision and recall, giving equal importance to both metrics. Note that each of these metrics takes on a value between 0 and 1 inclusive. The larger the value, the better the assignments are. Furthermore, we report the results obtained by both carrying out 10-fold cross validation on the training data and carrying out the analysis on the test set.

To evaluate the cost of creating the classification models, we measured the time it took to train the models. The smaller the training time, the better the approach is.

### 3.2.3 Data and Analysis

Table 1 summarizes the results we obtained. We first observed that all the classifiers we trained performed better than the baseline classifier. While the baseline classifier provided an accuracy of 0.10 on the training set and 0.12 on the test set, those of the worst-performing classifier were 0.47 and 0.52, respectively.

We then observed that the SELECTED ensembles generally performed similar or better than the BEST ensembles, supporting the conjecture that using diversified set of classifiers in an ensemble can help improve the accuracies by better spanning the learning space. For example, while the accuracy of the BEST-5 ensemble was 0.67 on the training set and 0.64 on the test set, those of the SELECTED-5 ensemble were 0.80 and 0.78, respectively. Fur-

thermore, the ensembles created by using 3 level-0 classifiers, rather than 5 level-0 classifiers, performed slightly better on our data set. For example, while the accuracy of the SELECTED-5 ensemble was 0.80 on the training set and 0.78 on the test set, those of the SELECTED-3 ensemble were 0.81 and 0.79, respectively.

Last but not least, among all the classifiers, the one that provided the best assignment accuracy (as well as the best F-measure) and did so at a fraction of the cost, was the linear SVC classifier (Table 1). While the linear SVC classifier provided an accuracy of 0.82 on the training data set and 0.80 on the test set with a training time of about three minutes, the runner-up classifiers, namely the SELECTED-3 and BEST-3 ensembles, provided the accuracies of 0.81 and 0.79, respectively, with a training time of about half an hour or more.

Based on both the assignment accuracies and the costs of training obtained from various classifiers using our data set, we have decided to employ linear SVC in IssueTAG. Consequently, all of the results presented in the remainder of the paper were obtained by using linear SVC classifiers.

## 4 Time Locality and Amount of Training Data

In this section, we investigate our second research question (RQ2): “How do the amount and time locality of training data affect the assignment accuracies?” Note that the results of this study will be used to determine the amount of training data (e.g., how many issue reports we should use) as well as the time locality of this data (i.e., how much back in time we should go) required for preparing the training data every time the underlying classification model needs to be retrained.

### 4.1 Approach

To answer these questions, we use the *sliding window* and *cumulative window* approaches introduced in (Jonsson et al. 2016). More specifically, we conjecture that using issue reports from “recent past” to train the prediction models, as opposed to using the ones from “distant past”, can provide better assignment accuracies since organizations, products, teams, and issues may change overtime.

To evaluate this hypothesis, we take a long period of time  $T$  (in our case, 13 months) and divide it into a consecutive list of calendar months  $T = [m_1, m_2, \dots]$ . For every month  $m_i \in T$ , we train and evaluate a linear SVC model. To this end, we use all the issue reports submitted in the month of  $m_i$  as the test set and all the issue reports submitted in the month of  $m_j$  as the training set, where  $i - j = \Delta$ , i.e., the sliding window approach in (Jonsson et al. 2016). Note that given  $m_i$  and  $\Delta$ ,  $m_j$  is the month, which is  $\Delta$  months away from  $m_i$  going back in time. For every month  $m_i \in T$ , we repeat this process for each possible value of  $\Delta$  (in our case,  $\Delta \in \{1, \dots, 12\}$ ). By fixing

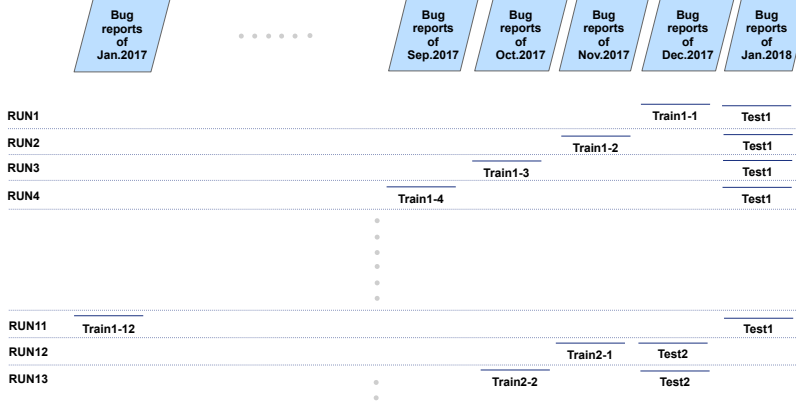


Fig. 2: Overview of the sliding window approach to study the effect of the time locality of training data on assignment accuracies.

the test set and varying the training sets, such that they come from different historical periods, we aim to measure the effect of time locality of the training data on the assignment accuracies.

Figure 2 illustrates the sliding window approach using the period of time from Jan 1, 2017 to Jan 31, 2018. For example, for the month of Jan 2018, we train a total of 12 classification models, each of which was trained by using all the issue reports submitted in a distinct month of 2017 (marked as Train1-1, Train1-2, ..., Train1-12) and separately test these models using all the issue reports submitted in the month of Jan, 2018 as the test set (marked as Test1). We then repeat this process for every month in the time period of interest, except for Jan 2017 as it does not have any preceding months. That is, for Dec 2017 (marked as Test2), we train and evaluate 11 models (marked as Train2-1, Train2-2, ...), for Nov 2017, we train and evaluate 10 models, etc.

To evaluate the effect of the amount of training data on the assignment accuracies, we use a related approach, called the cumulative window approach (Jonsson et al. 2016). This approach, as is the case with the sliding window approach, divides a period of interest  $T$  into a consecutive list of months  $T = [m_1, m_2, \dots]$ . Then, for every possible pair of  $m_i \in T$  and  $\Delta$ , we train and evaluate a classification model, where all the issue reports submitted in the month of  $m_i$  are used as the test set and all the issue reports submitted in the preceding  $\Delta$  months, i.e.,  $\{m_j \in T \mid 1 \leq i - j \leq \Delta\}$ , are used as the training set.

Figure 3 illustrates the approach. For example, for the month of Jan 2018, we train a total of 12 classification models. The first model is created by using

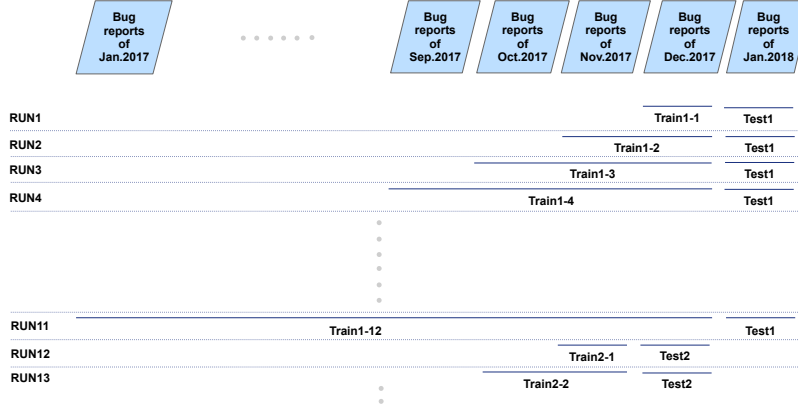


Fig. 3: Overview of the cumulative window approach to study the effect of the amount of training data on assignment accuracies.

the previous month's data (marked as Train1-1), the second model is created by using the previous two months' data (marked as Train1-2), and the last model is created by using the previous year's data (marked as Train1-12). The same process is repeated for every possible month in the period of interest.

## 4.2 Evaluation

We conducted a series of experiments to evaluate the effect of the amount and time locality of training data on assignment accuracies.

### 4.2.1 Experimental Setup

In these experiments, we used all the issue reports that were submitted during the period from Jan 1, 2017 to Jan 31, 2018. The summary statistics for this data set can be found in Table 2. All told, we have trained and evaluated a total of 144 linear SVC models for this study. All the experiments were carried out on the same platform with the previous study (Section 3.2.1).

### 4.2.2 Evaluation Framework

We used the assignment accuracies (Section 3.2.2) for evaluations.

Table 2: Number of issue reports submitted.

month	# of issue reports submitted	# of teams assigned
Jan 2017	6364	57
Feb 2017	5038	56
Mar 2017	7188	57
Apr 2017	6623	55
May 2017	6601	56
Jun 2017	6145	56
Jul 2017	6341	53
Aug 2017	6025	54
Sep 2017	5961	54
Oct 2017	6774	52
Nov 2017	7996	54
Dec 2017	7881	49
Jan 2018	7426	51
Total	86363	69

#### 4.2.3 Data and Analysis

Figures 4 and 5 represent the results we obtained from the sliding window and cumulative window approach, respectively. In these figures, the vertical and horizontal axes depict the assignment accuracies obtained and the  $\Delta$  values used in the experiments, respectively. The accuracies associated with a  $\Delta$  value were obtained from the classification models, each of which was created for a distinct month in the period of interest by using the same  $\Delta$  value. Furthermore, the polynomials in the figures are the second degree polynomials fitted to the data.

Looking at Figure 4, we first observed that using issue reports from recent past to train classification models, rather than the ones from distant past, provided better assignment accuracies; the accuracies tended to decrease as  $\Delta$  increased. For example, while the average assignment accuracy obtained when  $\Delta = 1$ , i.e., when the issue reports submitted in the immediate preceding months were used as the training sets, was 0.73, that obtained when  $\Delta = 12$ , i.e., when the issue reports submitted in Jan 2017 were used as the training set for the issue reports submitted in Jan 2018, was 0.52%.

Looking at Figure 5, we then observed that as we went back in time to collect the training data starting from the immediate preceding months (i.e., as  $\Delta$  increased in the cumulative window approach), the assignment accuracies tended to increase first and then stabilized around a year of training data. For example, while the average accuracy obtained when  $\Delta = 1$ , i.e., when the issue reports submitted only in the immediate preceding months were used as the training sets, was 0.73%, that obtained when  $\Delta = 12$ , i.e., when all the issue reports submitted in the preceding 12 months were used as the training data set, was 0.82%.

Based on the results of these studies, to train a prediction model at a given point in time, we decided to use all the issue reports that have been submitted



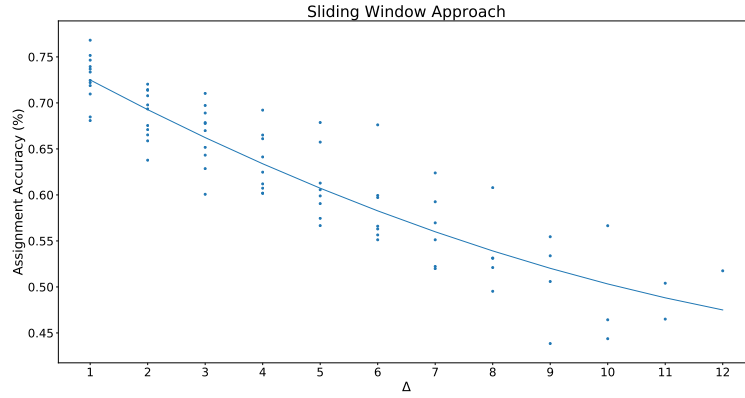


Fig. 4: Assignment accuracies obtained from the sliding window approach.

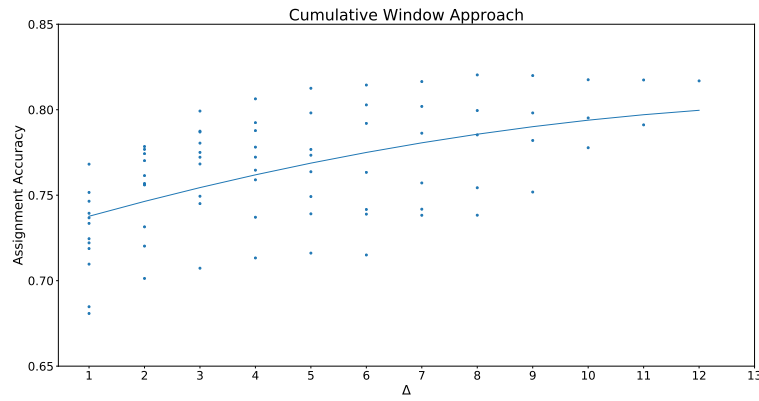


Fig. 5: Assignment accuracies obtained from the cumulative window approach.

in the last 12 months as the training set. Clearly, among all the issue reports of interest, we filter out the ones that have not yet been closed (Section 3.2.1).

## 5 Automated Issue Assignments in Practice

In this section, we investigate our third research question (RQ3): “How does automated issue assignment compare to manual issue assignment in practice?” Note that the results of this study will help evaluate the pros and cons of automated issue assignments in the field.

## 5.1 Approach

To deploy IssueTAG at IsBank and Softtech, we carried out a number of meetings with the IT-HD, AST, and software development teams. In these meetings, the problems with the manual issue assignment process were discussed, IssueTAG was presented, and the effect of automating the assignment process was demonstrated by using the results of a number of preliminary studies conducted on historical data collected from the field.

One commonly accepted observation, which was made numerous times in these meetings, was that automating the issue assignment process (i.e., deploying IssueTAG) would also require to modify the other parts of the process around the deployed system to improve the efficiency and effectiveness of the entire process to the extent possible.

One refinement suggestion came from us (Process Improvement Team at Softtech). In our preliminary studies, we observed that wrong assignments made by IssueTAG were often caused due to the difficulty of distinguishing related, but different development teams from each other, such as the teams working on related products or working on different components of the same product. That is, when an issue report was assigned to a wrong team, the assignee and the correct team (i.e., the one, to which the report should have been assigned) were often related to each other, e.g., they were aware of each other's works. Consequently, we suggested that in the presence of an incorrect assignment, rather than returning the issue report to IT-HD for reassignment, which was typically the case in the manual assignment process (Section 2), letting the assignee (e.g., the AST member embedded in the incorrectly assigned team) do the reassignment, could profoundly speed up the process.

Another refinement suggestion came from the IT-HD management. They simply suggested to prevent IT-HD clerks from modifying the issue assignments made by IssueTAG. On one hand, this was a natural consequence of the design decision discussed above in the sense that when the reassignments are made by the current assignee, IT-HD clerks will not necessarily be aware of these modifications, thus may not learn from them to improve their assignment accuracies. On another hand, we observed that IT-HD was actually looking forward to deferring the responsibility of issue assignments. One reason was that, especially for the new IT-HD clerks, the learning curve for excelling in assignments was generally steep due to the large number of issue reports received on a daily basis and the relatively large number of development teams present (Section 5.2.3). In fact, IT-HD was maintaining a knowledge base (comprised mostly of spreadsheets) to help the clerks with the assignments. However, it was cumbersome and costly for them to keep this knowledge base up to date. Nevertheless, incorrect assignments were often causing friction between the IT-HD clerks and AST members as well as the development teams.

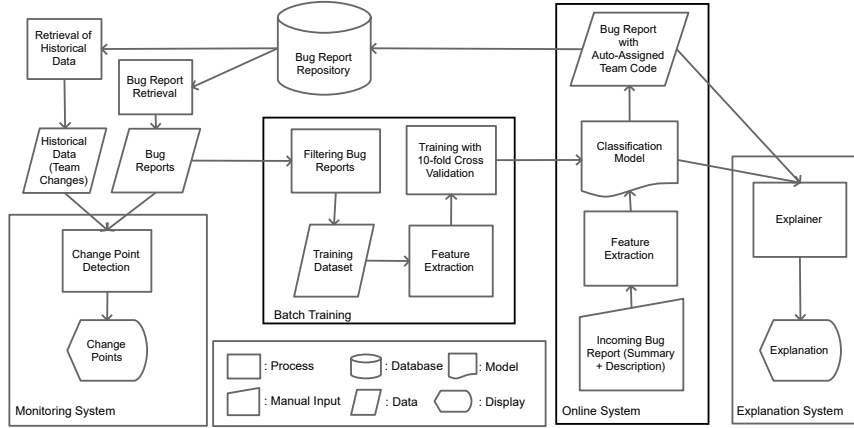


Fig. 6: High level architecture of IssueTAG.

Table 3: Summary statistics regarding the operations of IssueTAG, starting from its deployment on Jan 12, 2018 till June 30, 2019.

Item	Value
Total number of issue reports assigned	134,622
Average number of issue reports per day	380
Total number of distinct teams	62
Average time it takes to train the model (with one year of data)	7m 42s
Average response time of the system	746 msec
Size of the trained model (trained with one year of data)	588 MB

## 5.2 Evaluation

We deployed IssueTAG on Jan 12, 2018. The system has been fully operational since then, making automated assignments for all the issue reports submitted. Figure 6 presents the overall system architecture. Furthermore, Table 3 reports some summary statistics regarding the operations of the deployed system.

### 5.2.1 Deployment Setup

Based on the results of our empirical studies in Section 3, IssueTAG was configured to use Linear SVC to train the classification models. And, based on the results obtained in Section 4, the models have been trained by using the issue reports submitted in the last 12-month time frame. Furthermore, as all the process improvement suggestions discussed in Section 5.1 were accepted by all the stakeholders involved, we configured IssueTAG such that once an

issue report was created by an IT-HD clerk for the first time, the report was automatically assigned to a development team by the deployed system and the IT-HD clerk did not have any means of interfering with the assignment process and/or modifying the assignment.

The system is deployed on a Dual-Core Intel(R) Xeon(R) E5-2695 v4 2.10 GHz machine with 32 GB of RAM running Windows Server 2012 R2 as the operating system.

### 5.2.2 Evaluation Framework

To evaluate the quality of the assignments over a period of time, we compute the assignment accuracy on a daily basis, which we refer to as *daily assignment accuracy*. More specifically, the daily assignment accuracy achieved on a day  $d$ , is the ratio of the assignments that are correctly made for the issue reports opened on the day  $d$ . Note that we compute the daily accuracies based on the dates, on which the issue reports are opened, rather than they are closed. This is because the automated assignments are made as soon as the issue reports are created (i.e., opened) by using the underlying classification model, which was available at the time of the creation.

To evaluate the reduction in the amount of manual effort required for the issue assignments, we measure the person-months saved by automating the process. To this end, a survey we conducted on the IT-HD clerks revealed that, given an issue report, it takes about 30 seconds on average for an IT-HD clerk to assign the report to a development team, which is mostly spent for reasoning about the issue report and (if needed) performing a keyword-based search in the knowledge base. Note that this effort does not include the effort needed to maintain the knowledge base. Therefore, the actual amortized manual effort is expected to be higher than 30 seconds. IssueTAG, on the other hand, requires no human intervention to make an assignment once an issue report has been created.

To evaluate the effect of the deployed system as well as the improvements made in the issue assignment process, we compute and compare the *solution times* before and after the deployment of IssueTAG. In particular, we define the solution time for an issue report as the time passed between the report is opened and it is closed. The shorter the solution times, the better the proposed approach is. Furthermore, as the characteristics of the reported issues, thus the solution times, can change over time, we, in the evaluations, compute and compare the solution times for the issue reports that were opened within two months before and after the deployment of IssueTAG.

### 5.2.3 Data and Analysis

Figure 7 presents the daily assignment accuracies achieved between December 2016 and June 2019. The time point 0 in this figure represents the date, on which the manual issue assignment process as it is described in Section 2, was started. Furthermore, the vertical dashed lines in the figure represent the

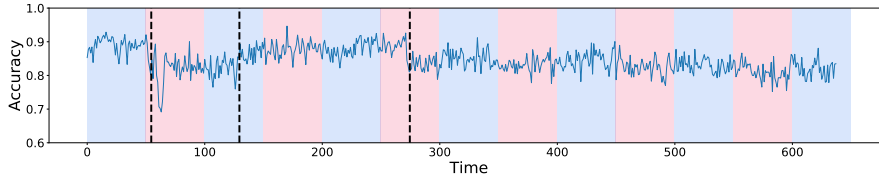


Fig. 7: Daily assignment accuracies achieved between December 2016 and June 2019. The time point 0 represents the date, on which the manual issue assignment process as it is described in Section 2 was started. The vertical dashed lines represent the points in time where a shift in daily accuracies was automatically detected by our change point detection approach (Section 8). IssueTAG was deployed at the third dashed line, i.e., all the accuracies before this line were obtained by manual assignments, whereas those after were obtained by automated assignments. The first dashed line represents the date, on which significant changes in team responsibilities occurred due to migrating certain functionalities from mainframes to state-of-the-art platforms. Therefore, the time gap between the first and second dashed lines (i.e., about 2.5 months) represent the amount of time it took for the IT-HD clerks to adapt to these changes.

points in time where a shift in daily accuracies was automatically detected by the change point detection approach we had developed (Section 8). IssueTAG was, indeed, deployed exactly at the 273th time point where the third vertical dashed line resides. That is, all the accuracies before this dashed line were obtained by manual assignments, whereas those after were obtained by automatic assignments. The other vertical dashed lines will be discussed below in this section.

We first observed that after IssueTAG was deployed, the daily assignment accuracies dropped slightly (Figure 7). More specifically, the average daily accuracies before and after the deployment were 0.864 ( $min = 0.691$ ,  $max = 0.947$ ,  $stddev = 0.040$ ) and 0.831 ( $min = 0.752$ ,  $max = 0.912$ ,  $stddev = 0.027$ ), respectively.

We, however, observed that the accuracy of an automated issue assignment system does not have to be higher than that of manual assignments in order for the system to be useful. First, we observed that IssueTAG reduced the manual effort required for the assignments. In particular, given that it takes an average of 30 seconds for an IT-HD clerk to assign an issue report to a development team and an average of 8,000 issue reports are received on a monthly basis, IssueTAG has been saving 5 person-months yearly, on average (8,000 issue reports \* 30 seconds = 240,000 seconds per month = 5 person-months per year).

Second, we observed that the deployed system together with the process improvements we implemented, profoundly reduced the turnaround time for

closing the issue reports. More specifically, the average solution times before and after the deployment were 3.26 days and 2.61 days, respectively.

Third, we observed that it can take quite a while for a human stakeholder to excel in the issue assignment task, which is, in deed, a problem, especially in the presence of high employee turn over rates. For example, the first vertical dashed line in Figure 7, represents the date on which an integral part of the core banking system was migrated from mainframes to state-of-the-art hardware and software platforms. As a result of this migration, the structure and the responsibilities of the related development teams changed significantly. In particular, the responsibilities of one development team working on mainframes were migrated to 3 development teams working on state-of-the-art platforms, which consisted of completely different software engineers. Evidently, the assignment accuracies were affected by this change; the daily accuracies dropped at the first vertical dashed line (i.e., 55th time point) and stayed low until the second vertical dashed line (i.e., the 130th time point). More specifically, the average daily accuracies obtained from the manual assignments before the first dashed line, in between the first and second dashed lines, and after the second dashed line until IssueTAG was deployed at the third dashed line were, 0.889 ( $min = 0.825$ ,  $max = 0.929$ ,  $stddev = 0.024$ ), 0.819 ( $min = 0.691$ ,  $max = 0.900$ ,  $stddev = 0.039$ ), and 0.879 ( $min = 0.822$ ,  $max = 0.947$ ,  $stddev = 0.024$ ), respectively. That is, it took the IT-HD clerks about 2.5 months to adapt to the new development teams. Therefore, this time frame can be considered to be a lower bound on the amount of time a new hire would require to learn to make accurate assignments. It is a lower bound in the sense that only 19% of the issue reports were affected by the changes in the team responsibilities during the aforementioned period of time and that the IT-HD clerks already had a great deal of experience; for a new hire, everything will be new.

Note further that the 0th time point in Figure 7 represents the date, on which Jira was started to be used for storing and managing the issue reports. That is, IT-HD clerks had been making manual assignments before this date, but had different means of managing the reports, which explains the high daily assignment accuracies event at the 0th time point in the figure. As we didn't have any access to the issue databases maintained before the 0th time point, we used only the issue reports managed by Jira in this research.

## 6 User Evaluations

In this section, we investigate our fourth research question (RQ4): "Is IssueTAG perceived as useful by the end-users?"

### 6.1 Approach

To carry out the study, we created a survey by following a survey template frequently used at Softtech. It had a total of 8 questions from two categories:

Table 4: Survey questions used for evaluating IssueTAG.

No	Question	Type	Category
Q1	I know the business requirements that the system is supposed to meet.	Likert scale	requirements satisfaction
Q2	The system (as a software product) is reliable.	Likert scale	requirements satisfaction
Q3	The system is useful.	Likert scale	requirements satisfaction
Q4	The system reduces the solution times for issue reports.	Likert scale	product quality
Q5	The issue assignments made by the system are trustworthy.	Likert scale	product quality
Q6	The system is robust.	Likert scale	product quality
Q7	I recommend the system to other companies.	Likert scale	product quality
Q8	What do you like and don't like about the system? Do you have any suggestions for improvement?	open-ended	product quality

*requirement satisfaction* and *product quality*. The former category aims to evaluate the extent to which the deployed system meets its requirements, whereas the latter category aims to evaluate the quality of the final product. All questions, except for the last one, were Likert scale questions each with answer options: *no opinion*, *1 - strongly disagree*, *2 - disagree*, *3 - agree*, and *4 - strongly agree*. The last question was an open-ended question. Furthermore, for the Likert scale questions, we asked the participants to elaborate on their responses, if they had “disagreed” or “strongly disagreed.” Table 4 presents the questions we used in the survey.

## 6.2 Evaluation

We conducted the survey on the AST members. We chose this group of stakeholders as the recipients of the survey because, being embedded in the development teams, they were the direct end-users of IssueTAG. That is, they, as the first recipients of the issue reports, were the ones to validate whether the assignments were correct or not and to reassign them as needed. The IT-HD clerks, on the other hand, could not participate in the survey because they were not considered to be the end-users of the deployed system in the sense that they neither made use of the assignments automatically made by the deployed system nor had a control over them.

### 6.2.1 Experimental Setup

About half of the AST members (more specifically, 14 out of 30) voluntarily agreed to participate in the study. The participants filled out the survey online at their spare time.

### 6.2.2 Evaluation Framework

For the Likert scale questions, we use the frequencies and the average scores obtained to quantitatively analyze the results. The average scores were computed as the arithmetic average of the scores with the “no opinion” responses

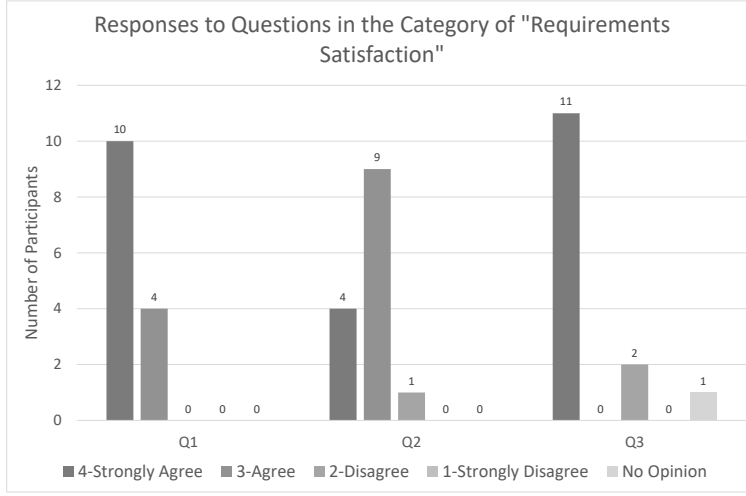


Fig. 8: Responses to questions in the category of “requirements satisfaction.”

excluded. For the open-ended question, we present the answers we received (Table 6) and qualitatively discuss them.

### 6.2.3 Data and Analysis

The results of the survey strongly suggest that IssueTAG meets its business needs with high quality. Regarding the questions in the category of “requirements satisfaction,” we observed that the majority of the participants thought IssueTAG was useful and reliable (Figure 8). More specifically, all of the participants “strongly agreed” or “agreed” to Q1, indicating that they knew the business requirements that IssueTAG was supposed to meet. And, 92.86% (13 out of 14) of the participants for Q2 and 78.57% (11 out of 14) of the participants for Q3, responded “agree” or higher. The average scores were 3.71, 3.21, and 3.69 (out of 4) for these questions, respectively.

Only 1 participant for Q2 and 2 participants for Q3 “disagreed.” The comments that they provided as to why they disagreed are given in Table 5. Evidently, part of the reason was that these participants were unrealistically expecting to have perfect assignments (with no incorrect assignments) from the deployed system.

Regarding the other quality aspects of the system, 100% (14 out of 14) of the participants for Q4, 78.57% (11 out of 14) for Q5, 92.86% (13 out of 14) for Q6, and 100% (14 out of 14) for Q7 responded “agree” or higher (Figure 9). The average scores were 3.36, 3.0, and 3.0, and 3.43 (out of 4) for these questions,



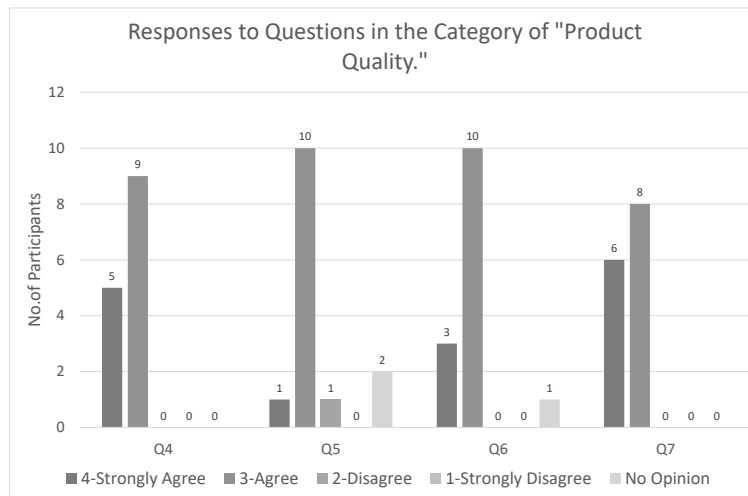


Fig. 9: Responses to questions in the category of “product quality.”

Table 5: Comments that the participants provided as to why they “disagreed.”

Question No	Comment No	Comment
Q2	1	Due to some keywords [appearing in issue reports], the system sometimes make incorrect assignments to my team.
Q3	1	In fact, the application is useful. However, in the presence of a wrong assignment made by the system, reassigning the bug report to the correct team, especially when we don't know which team it should really be assigned to or when the other team refuses to take the responsibility for the issue report, causes delays in the process.
Q3	2	The system sometimes makes incorrect assignments.
Q5	1	I can't say "I agree" because I sometimes encounter incorrect assignments.

respectively. Only 1 participant disagreed with Q5, the comment of whose can be found in Table 5. We, furthermore, observed that all the participants would recommend the system to other companies; all responded “agree” or higher to Q7 (Figure 9).

Last but not least, the responses given to the open-ended question Q8 can be found in Table 6. All of these comments can be considered as generally positive. A couple of them actually make some suggestions for future improvements. For example, the last comment basically suggests that the system should provide an explanation as to why a given issue report is assigned

Table 6: Responses given to the open-ended question Q8.

Comment No	Comment
1	I think that the assignments are made rapidly and accurately. I have not been having any issues with the system. We [as a team] rarely receive incorrect assignments. I, however, believe that this is normal because the same words [terms] can be related with multiple development teams. It is quite normal for the system not being able to distinguish between teams in such situations.
2	Most of the time, the system works for us. Sometimes, however, it assigns irrelevant issue reports to my team.
3	General words, such as "problem", should not be used by the system when assigning issue reports to development teams.
4	The system profoundly reduced the loss of time by rapidly assigning issue reports to development teams with high accuracy.
5	I think that it takes a while for an AI algorithm to learn about new issue reports. Improvements can be made in this area.
6	I believe that the system had a profound effect on assigning the issue reports to the right development teams.
7	It is a nice and practical system, better results can be obtained with further development. One area for improvement could be to explain the keywords [terms] used for the assignments.

to the selected development team. As a matter of fact, this request turned out to be a common one, for which we have developed an automated approach (Section 7.1).

## 7 Explaining Team Assignments

One interesting observation we made after IssueTAG had been deployed was that, occasionally, especially for incorrect assignments, the stakeholders demanded some explanations as to why and how certain issue reports had been assigned to their teams. This was an issue we didn't expect to face before deploying the system. As a matter of fact, based on the informal discussions we had with the stakeholders, we quickly realized that explaining the assignments could further improve the trust in IssueTAG.

In this section, we develop and empirically evaluate (by conducting a survey on actual stakeholders) an approach for automatically generating explanations for the issue assignments made by the underlying classification model, answering our fifth research question (RQ5): "Can the issue assignments made by the underlying data mining model be explained in a non-technical manner?"

Note that since the classification models we use, namely the linear SVC models, are not human-readable, providing such explanations is a non-trivial task. To the best of our knowledge, there is, indeed, no work in the literature of automated issue assignment, addressing this problem.

One requirement we have is that the explanations should easily be interpreted and understood even by non-technical stakeholders as the recipients of these explanations are not necessarily technical stakeholders. Another requirement is that they should be given in terms of the natural language descriptions present in the issue reports, so that stakeholders can relate to them.

With all these in mind, we conjecture that providing a list of most influential (positive or negative) words for an issue assignment together with their relative impact scores as an explanation for the assignment, could help stakeholders understand the rationale behind the assignments.

Interestingly enough, we observe that such explanations could also be used in an interactive manner to enable the stakeholder creating the issue report to provide feedback to the classification model. Although such human-in-the-loop assignments are out of the scope of this paper, we, nevertheless, added additional questions to our survey to evaluate the plausibility of the idea.

### 7.1 Approach

We use LIME (Local Interpretable Model-Agnostic Explanations) to automatically produce explanations for the issue assignments made by IssueTAG. LIME is a model-agnostic algorithm for explaining the predictions of a classification or regression model (Ribeiro et al. 2016). In this work, we, (to the best of our knowledge) for the first time, use LIME in the context of automated issue assignment and evaluate it by carrying out a survey on actual stakeholders in the field. Next, we briefly describe the LIME algorithm without any intention to provide all the mathematics behind it. The interested reader can refer to (Ribeiro et al. 2016) for further details.

LIME, in our context, aims to identify a human-interpretable, locally faithful model, which provides qualitative understanding between the terms used in issue reports and the development teams, to which they are assigned. In a nutshell, given an issue report, the assignment made for this report, and the underlying classification model, LIME first represents the report as a bag of words and samples instances around the report by drawing subsets of the words in the bag uniformly at random. Then, the samples are weighted by their proximities to the original issue report and fed to the classification model to label them. Next, all the samples together with their associated labels are used to learn a linear model comprised of  $K$  terms (in our case,  $K = 6$ ), which distinguishes the labels. Finally, the linear model learnt is reported as an explanation for the assignment.

The explanation generated for an assignment is, indeed, a set of  $K$  terms selected from the original issue report together with their relative weights, indicating the influential terms that either contribute to the assignment or are evidence against it. Figure 13a presents an example explanation created for an assignment made by IssueTAG in the field. The vertical axis reports the most influential terms selected, whereas the horizontal axis denotes their relative weights. The terms with positive weights depict the terms that contribute to the assignment, whereas those with negative weights depict the ones that are evidence against it. That is, in a sense, the former set of terms vote for the assignment, whereas the latter ones vote against it in an attempt to change the assignment.

### 7.2 Evaluation

We could not simply ask the stakeholders to evaluate each and every explanation created for the issue assignments, which were of interest to them. The

Table 7: Survey questions related to selected issue reports and their explanations.

No	Question	Type
Q1	Is the explanation helpful in understanding the assignment?	Yes/No
Q2	Given the issue report, the assignment, and the explanation for the assignment, how would you rate the trustworthiness of the assignment?	Likert scale
Q3	Which terms in the explanation make you think that the assignment is not trustworthy?	Open ended
Q4	What are the, additional terms that you would like to see in the explanation; or the terms, the impact factor of which you would like to increase in the explanation, before you can trust the assignment?	Open ended

reason was that there was a large number of issue reports submitted on a daily basis (Section 5.2) and that checking out the explanations was optional, i.e., the stakeholders were not required to have a look at the explanations. Therefore, forcing the stakeholders to evaluate the explanations as the issue assignments were made, could have adversely affected their performance.

### 7.2.1 Experimental Setup

We, therefore, carried out an empirical study with the same participants involved in our survey in Section 6, after having their consensus to voluntarily participate in this additional study, which were accepted by all of them.

For each participant, we randomly picked 10 issue assignments, which were handled by the participant in the last week before the study, such that the ratio of correctly and incorrectly assigned issue reports roughly resembled the average daily assignment accuracy. When there were less than 10 issue assignments for a participant, we selected all of the available ones. All told, we picked a total of 130 issue assignments (10 for each participant, except for two, for whom we could have only 5 assignments each). Out of all the selected assignments, 13 (10%) were incorrect.

We then created a questionnaire for each participant by using the issue assignments selected for the participant. For each assignment in the questionnaire, we included 1) the issue report, 2) the assignment made by IssueTAG, 3) the explanation automatically created by the proposed approach, using the 6 most influential terms involved in the assignment, and 4) four questions (Table 7).

The first two questions, namely Q1 and Q2, were directly concerned with our main research question in this study, i.e., whether or not the automatically generated explanations could help stakeholders understand the rationale behind the assignments. Q1 was a “yes” or “no” question, whereas Q2 was a Likert scale question with answer options: *2 - very trustworthy*, *1 - trustworthy*, *0 - not sure*, *-1 - untrustworthy*, *-2 - very untrustworthy*. The last two questions, namely Q3 and Q4, on the other hand, aimed to evaluate the plausibility of using the explanations to get feedback from the stakeholders in an attempt to further improve the assignment accuracies. These questions were open-ended questions, which were conditional on Q2; the participants

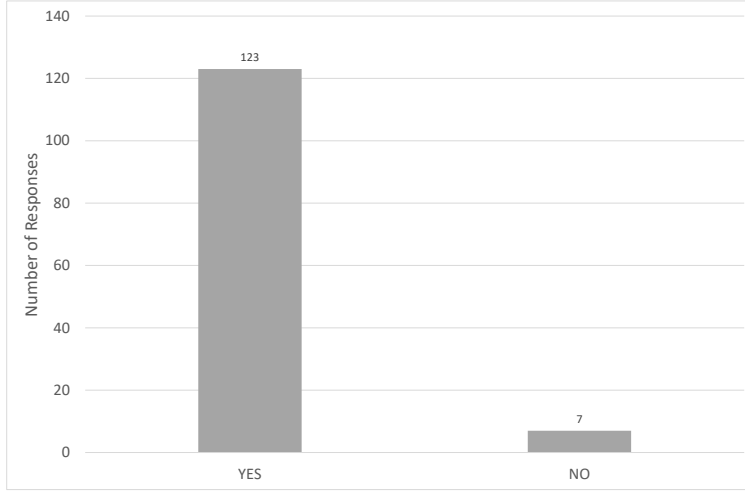


Fig. 10: Responses to Q1: “Is the explanation helpful in understanding the assignment?”

were asked to answer these questions only when the response to Q2 was either “untrustworthy” or “very untrustworthy.”

All the explanations were created by using the LIME Python library (Ribeiro et al. 2016) with  $K = 6$  – a decision we made based on the maximum number of terms that we thought a stakeholder could efficiently and effectively reason about.

### 7.2.2 Evaluation Framework

For Q1 and Q2, we use the frequencies of responses to quantitatively analyze the results. For Q3 and Q4 (when answered), we manually investigate how the feedbacks can be used to further improve the accuracies.

### 7.2.3 Data and Analysis

Regarding Q1, we observed that participants found 95% (123 out of 130) of the explanations, each of which was created for a distinct issue assignment, helpful in understanding the rationale behind the assignments (Figure 10).

Regarding Q2, based on the explanations created for the correct assignments, the participants found 93% of the assignments (109 out of 117) “trustworthy” or “very trustworthy” (Figure 11). And, for the remaining 7% of the assignments (8 out of 117), they were “not sure” whether the explanations

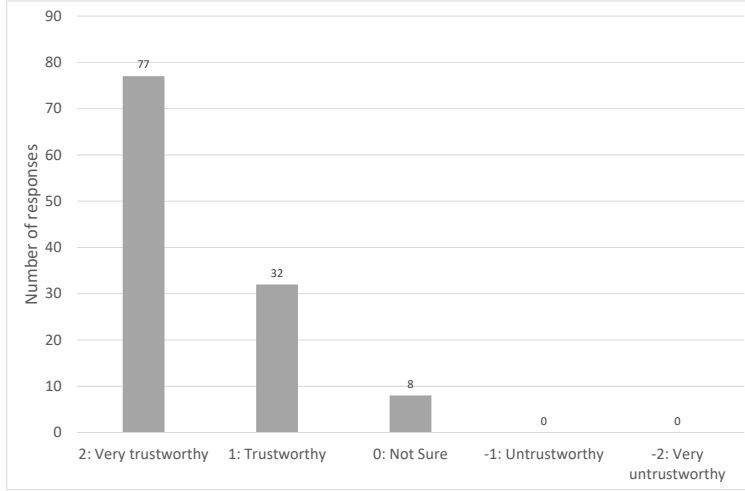


Fig. 11: Responses to Q2 (for the correct assignments): “Given the issue report, the assignment, and the explanation for the assignment, how would you rate the trustworthiness of the assignment?”

helped them decide if the assignments were reliable or not. None of the assignments was found “untrustworthy” or “very untrustworthy.”

Interestingly enough, based on the explanations created for the incorrect assignments, we observed that the participants found 77% of the assignments (10 out of 13) “trustworthy” or “very trustworthy,” suggesting that given the same issue reports, these participants would have made the same or similar mistakes in assigning the reports. We believe that this was because of some missing information in these issue reports, which was required for accurate assignments (Figure 12). Furthermore, the participants were “not sure” about the trustworthiness of the 15% of the assignments (2 out of 13).

Regarding Q3 and Q4, among all the responses given to Q2, only one was scored below 0. That is, based on the explanations created for the assignments, only one of the assignments was found “untrustworthy.” And, this assignment was, indeed, an incorrect assignment made by IssueTAG.

The explanation created for the aforementioned assignment is given in Figure 13a. Given this explanation, the participant argued in her response that the term “telegram,” which is a domain specific term used when creating a credit account, was an important term for the issue report at question. Therefore, it should have positively, rather than negatively, affected the assignment. As a matter of fact, this argument was also well-aligned with the automatically generated explanation given in Figure 13a in the sense that “telegram,”

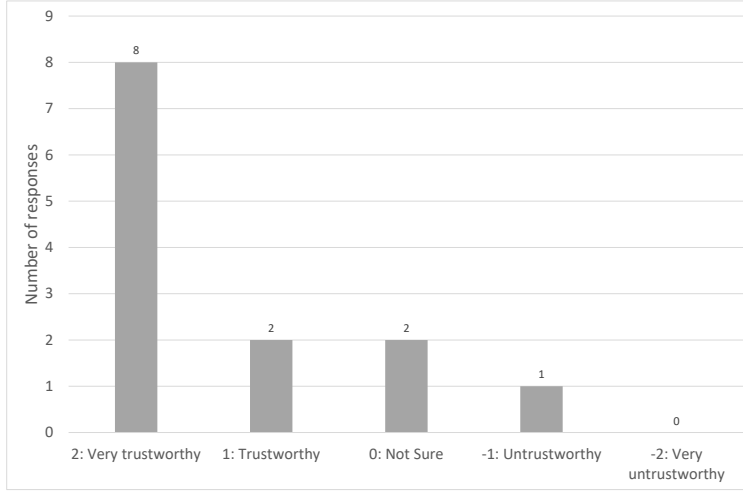


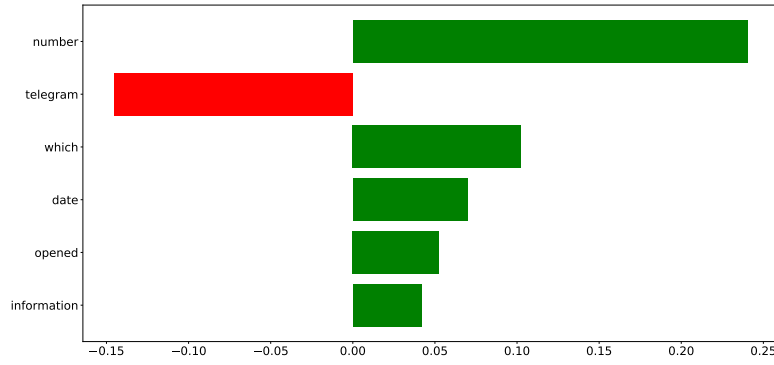
Fig. 12: Responses to Q2 (for the incorrect assignments): “Given the issue report, the assignment, and the explanation for the assignment, how would you rate the trustworthiness of the assignment?”

being a term with a large negative impact, voted against the assignment in an attempt to change it. It was, however, not strong enough to modify the outcome.

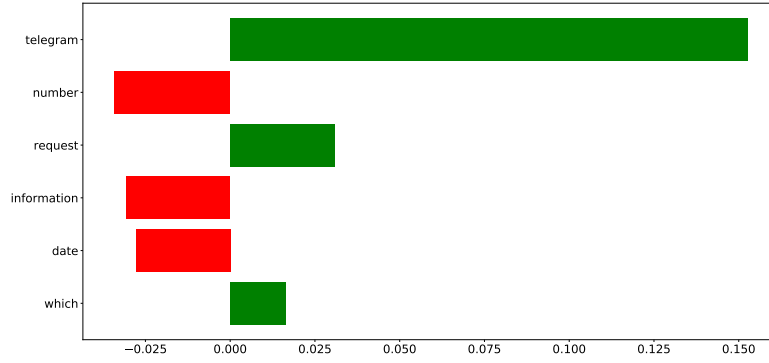
Interestingly enough, Figure 13b presents the explanation created for the second likely assignment made by the underlying classification model, which turned out to be the correct assignment. Note that in this assignment, the term “telegram” had the largest positive impact on selecting the correct team, which was also suggested by the stakeholder. Therefore, had the participant presented with the explanations created for the top two most likely assignments, she could have selected the second assignment, thus increased the assignment accuracy. Note that the aforementioned type of approaches are beyond the scope of this work. However, as the results of this study are promising, we, as a future work, plan to develop “human-in-the-loop” approaches, which leverage the automatically created explanations to further improve the assignment accuracies.

## 8 Monitoring Deterioration

In this study, we address our sixth research question (RQ6): “Can the deteriorations in the assignment accuracies be automatically detected in an online manner?” This was, indeed, another issue we faced after the deployment of



(a)



(b)

Fig. 13: The explanations created for the assignment marked as “untrustworthy” by a participant: a) the explanation created for the original assignment, which was incorrect and b) the explanation created for the second likely assignment, which was correct.

IssueTAG. It is important because such a mechanism not only increases the confidence of the stakeholders in the system, but also helps determine when the underlying classification model needs to be recalibrated by, for example, retraining the model (Sections 3-4).



## 8.1 Approach

One observation we make is that every issue report at Softtech is closed by the development team, who has fixed the reported issue. Therefore, in the presence of an incorrect assignment made by IssueTAG, the report is reassigned and the history of the reassignments is stored in the issue tracking system. Consequently, at any point in time, the assignment accuracy of IssueTAG can automatically be computed using the history of the issue reports that have been closed. Therefore, deteriorations in the accuracy can be analyzed in an online manner.

To this end, we use an online change point detection approach, called Pruned Exact Linear Time (PELT) (Killick et al. 2012). In a nutshell, PELT is a statistical analysis technique to identify when the underlying model of a signal changes (Truong et al. 2018b). In our context, we feed PELT with a sequence of daily assignment accuracies (Section 8.2) as the signal. The output is a set of points in time (if any) where mean shifts. PELT, being an approach based on dynamic programming, detects both the number of change points and their locations with a linear computational cost under certain conditions (Killick et al. 2012). Further information can be found in (Killick et al. 2012) and (Truong et al. 2018b).

PELT has been used for change point detection in many application domains, including DNA sequence data, financial time series, and oceanographic data (Hocking et al. 2013; Lavielle and Ere 2007; Killick et al. 2012). In this work, we, on the other hand, use it (to the best of our knowledge) for the first time in the context of automated issue assignment to detect the deteriorations in the assignments made by a data mining model.

## 8.2 Evaluation

We applied the PELT approach to the daily assignment accuracies collected from the field. PELT detected three change points, each of which was depicted by a vertical dashed line in Figure 7. It turned out that these change points, indeed, coincided with some important events that affected the assignment accuracies, validating the results obtained from the proposed approach. The first dashed line represents the date, on which significant changes in the team responsibilities occurred due to migrating certain functionalities from mainframes to state-of-the-art platforms. The time gap between the first and second dashed lines (i.e., about 2.5 months) represent the amount of time it took for the IT-HD clerks to adapt to these changes. And, the third dashed line represents the date on which IssueTAG was deployed. Further discussion on these change points can be found in Section 5.2.3.

We observed that PELT did not detect any other change point after IssueTAG was deployed. We believe that this was because the underlying classification model had been regularly retrained at every month as a part of Softtech's

policy by using the issue reports submitted in the last 12 months before the calibration (Section 4).

To further evaluate the proposed approach, we, therefore, carried out additional experiments where we systematically varied the nature of the deteriorations and evaluated whether the proposed approach detected them or not. Note that controlling the nature of the deteriorations in this study allows us to reliably evaluate the results, because when the true nature of a deterioration, such as the exact point in time at which the deterioration occurred, is not known, which is typically the case with the data collected from the field, the analysis may suffer from the lack of ground truth. Note further that even if the underlying classification model is regularly trained, monitoring for deteriorations is still relevant as the assignment accuracies can still deteriorate in between the calibrations.

### 8.2.1 Experimental Setup

In each experimental setup, we used an ordered sequence of 200 daily assignment accuracies. The first 100 of these accuracies came from a normal distribution representing the accuracies expected from IssueTAG, whereas the remaining 100 accuracies came from a distribution (or a number of distributions) representing a deterioration. That is, the change point in each experiment was the 100th time point as the deterioration was introduced after this point in time.

For each experimental setup, we then mimicked the real-life operations of IssueTAG. More specifically, given a sequence of 200 daily assignment accuracies, we fed them to the proposed approach one daily accuracy after another in the order they appeared in the sequence. After every daily accuracy, a decision was made whether a deterioration had occurred, and if so, when. We finally determined how long it took for the proposed approach to detect the deterioration. For each experimental setup, we repeated the experiments 1000 times.

As an implementation of the PELT approach, we used the **ruptures** Python library (Truong et al. 2018a). As the *penalty level*, i.e., the only parameter to calibrate in PELT, we used the empirically determined value of 0.05. The penalty level is a mechanism used for guarding against overfitting, determining to which extent a shift in the accuracies should be considered as a change point. The larger the penalty level, the fewer (and more significant) change points are detected.

To model the daily accuracies expected from the system, we used a normal distribution with mean of 0.85 and standard deviation of 0.025 (i.e.,  $\mu = 0.85$  and  $\sigma = 0.025$ ), mimicking the daily accuracies of the deployed system observed in the field (Section 5.2.3). To model the deteriorations, we experimented with two types of changes: *sudden deteriorations* and *gradual deteriorations*. In either case, we used 5-, 10-, 15-, and 20-point drops in daily accuracies, such that the mean accuracy (i.e., the mean of the distribution,

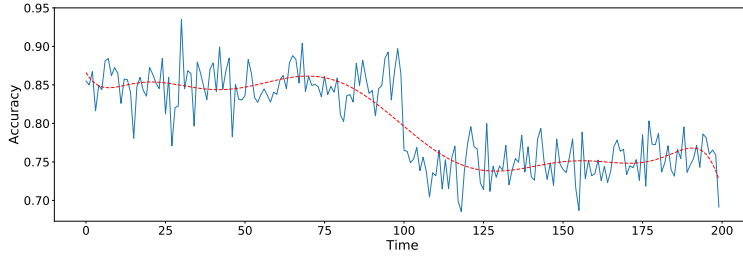


Fig. 14: An example sequence of daily assignment accuracies showing a sudden 10-point deterioration at the 100th time point.

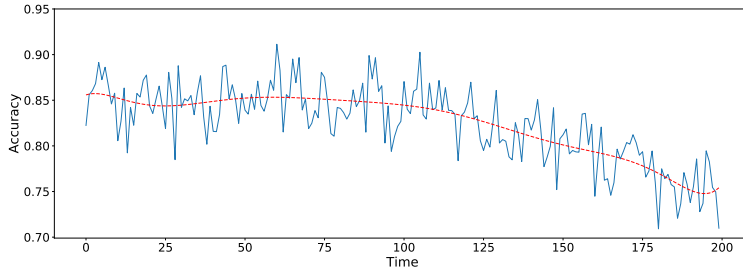


Fig. 15: An example sequence of daily assignment accuracies showing a gradual 10-point deterioration starting from the 100th time point.

from which the accuracies were drawn) eventually became 0.80, 0.75, 0.70, and 0.65, respectively.

For the sudden deteriorations, we abruptly dropped the mean accuracy from 0.85 to the requested level (i.e., 0.80, 0.75, 0.70, or 0.65, depending on the choice) right after the change point at the 100th time point and kept it intact until and including the 200th time point (i.e., until the end of the experiment). Figure 14 presents an example sequence of daily assignment accuracies showing a sudden 10-point deterioration.

For the gradual deteriorations, on the other hand, the changes were obtained by linearly dropping the mean accuracy starting from right after the change point at the 100th time point until and including the 200th time point, such that the mean accuracy at end of the experiment became 0.80, 0.75, 0.70, or 0.65, depending on the choice. For example, if the requested level of accuracy was 0.80, then starting from the mean accuracy of 0.85, the mean accuracy would be dropped by 0.05-point each day (5-point drop/100 days) until it would become 0.80 at the 200th time point. Figure 15 presents an example sequence of daily assignment accuracies showing a gradual 10-point deterioration starting from the 100th time point.

### 8.2.2 Evaluation Framework

To evaluate the proposed approach, we first determine whether the deteriorations are detected or not. If so, we measure *detection time* as the number of days past after the change point (i.e., after the 100th time point) until the deterioration is detected. The smaller the detection time, the better the proposed approach is.

### 8.2.3 Data and Analysis

Table 8 presents the data we obtained on the sudden deteriorations used in the study. We first observed that the proposed approach detected all the deteriorations. We then observed that as the deterioration amount increased, the detection time tended to decrease, i.e., the proposed approach tended to detect the deteriorations faster. On average, the deteriorations were detected in 1.33, 1.60, 1.84, 2.67 days after there was a 20-, 15-, 10-, and 5-point sudden drop in the mean assignment accuracies, respectively.

Table 9 presents the data we obtained on the gradual deteriorations. As was the case with the sudden deteriorations, the proposed approach detected all the deteriorations and as the deterioration amount (thus, the deterioration rate) increased, the detection time tended to decrease. Compared to the sudden deteriorations, however, the detection times for gradual deteriorations increased, which is to be expected. To better evaluate the quality of the detections, we, therefore, analyzed the mean accuracies that were present when the deteriorations were detected. We observed that throughout all the experiments, the proposed approach detected the deteriorations before the mean accuracy dropped more than 5-points (the last column in Table 9).

## 9 Lessons Learnt

**Stakeholders do not necessarily resist change.** To deploy IssueTAG, we carried out a number of meetings with the IT-HD, AST, and software development teams. One thing we repeatedly observed in these meetings was that all the stakeholders, although they had some rightful concerns, such as what if the proposed approach adversely affects the issue-resolution process –

Table 8: Results obtained on sudden deteriorations. The experiments were repeated 1000 times.

Deterioration	min	Detection Time		
		avg	max	stddev
5-point	1	2.67	6	1.13
10-point	1	1.84	3	0.38
15-point	1	1.60	2	0.49
20-point	1	1.33	2	0.47

Table 9: Results obtained on gradual deteriorations. The experiments were repeated 1000 times.

Deterioration	min	Detection Time			Minimum Mean Accuracy at the Point of Detection
		avg	max	stddev	
5-point	1	31.03	55	13.64	0.8125
10-point	1	20.14	35	8.41	0.8070
15-point	1	15.70	26	6.49	0.8035
20-point	1	13.36	21	4.95	0.8000

a major concern for a company developing business-critical software systems – were actually willing to automate the process of issue assignments as much as possible.

We, indeed, had no objection at all. The AST members and the development teams were looking forward to reducing the turnaround time for issue resolutions; the incorrectly assigned issue reports were bouncing back and forth between the IT-HD clerks and the AST members, causing a great deal of wasted time. The IT-HD clerks were looking forward to 1) avoiding the costly and cumbersome process of maintaining a knowledge base about the development teams and their responsibilities and 2) deferring the responsibility of making assignments as much as possible since incorrect assignments were often causing friction with the AST members and the development teams.

Another reason behind the absence of any resistance was that none of the stakeholders felt threatened by the new system. The IT-HD clerks were still needed as they were the ones communicating with both the bank costumers and employees to collect the issues, resolving the ones that they could, and creating issue reports for the remaining ones. The AST members were still needed as they were the ones helping the development teams manage the issue reports. The development teams were still needed as they were the ones developing the software products.

**Gradual transition helps stakeholders build confidence, facilitating the acceptance of the system.** To address the rightful concerns of the stakeholders regarding the accuracy of the proposed system, we followed a gradual transition strategy. First, we simply added a single button to the screen, which the IT-HD clerks used to create the issue reports. We initially did not modify the assignment process at all in the sense that the use of this button was optional. If the IT-HD clerk chose to arm the button after creating an issue report, it would simply display the assignment made by IssueTAG. The clerk could then accept the assignment as it was or modify it. We observed that 3 months after the deployment of this button, enough confidence was built among the stakeholders to fully deploy the system.

**It is not just about automating the issue assignments, but also about changing the process around it.** One observation we made numerous times during the meetings with the stakeholders was that automating the issue assignments also requires to modify the other parts of the assignment

process to improve the efficiency and effectiveness of the entire process to the extent possible. This was because most of the steps in the assignment process were dependent on the fact that issue assignments were made by the IT-HD clerks. Changing this, therefore, necessitated other changes. In particular, we prevented the IT-HD clerks from modifying the issue assignments made by IssueTAG and the incorrectly assigned issue reports from being returned back to the IT-HD clerks for a reassignment. All of these changes were based on the discussions we had with the stakeholders as well as the analysis of the results we obtained from a number of feasibility studies (Section 5).

**The accuracy of the deployed system does not have to be higher than that of manual assignments in order for the system to be useful.**

Although the assignment accuracy of IssueTAG was slightly lower than that of manual assignments, it reduced the manual effort required for the assignments and improved the turnaround time for closing the issue reports. All of these helped improve the usability of IssueTAG, which was also evident from the survey we conducted on the stakeholders in the field (Section 6).

**Deploying a data mining-based automated issue assignment system requires the development of additional functionalities.** When the issue assignments are automatically made by using a data mining model, the accuracy of the assignments needs to be monitored and deteriorations need to be detected in an online manner, so that corrective actions, such as recalibrating the underlying model, can be taken in time. To this end, we have developed a change point detection-based approach using PELT (Killick et al. 2012) (Section 8). Furthermore, stakeholders may demand some explanations as to why certain issue reports (especially the incorrectly assigned ones) have been assigned to their teams. Note that since the data mining models used for predicting the assignments are not necessarily readable and interpretable by human beings (as was the case in this work), generating such explanations can be a non-trivial task. To this end, we have developed a LIME-based (Ribeiro et al. 2016) approach for automatically generating explanations that can easily be interpreted even by non-technical stakeholders.

## 10 Threats to Validity

In this section, we discuss threats to validity.

### 10.1 Construct Validity

To circumvent the construct threats, we used the well-known *accuracy* metric (Manning et al. 2010) throughout the paper to evaluate the quality of the issue assignments. We have also complemented the accuracy results with other well-known metrics, namely precision, recall, and F-measure, as we see fit (Section 3). We mainly focused on the accuracies because 1) it was the choice of a recent related work in the literature (Jonsson et al. 2016) and 2) the

assignment accuracies and F-measures (computed by giving equal importance to both precision and recall) we obtained in the experiments were comparable (Table 1).

To measure the amount of effort saved by automating the issue assignments (Section 5), we used the person-month metric, which is also a well-known metric to quantify effort in software engineering projects (Pressman 2005).

To measure the effect of the proposed approach on the issue-resolution process, we compared the average times required to close the issue reports before and after the deployment of IssueTAG (Section 5). To this end, we used the dates and times recorded by the issue report management tool (namely, Jira). Furthermore, since the characteristics of the reported issues, thus the times it takes to resolve them, can change over time, we used the issue reports submitted within two months before and after the deployment of the system for this purpose.

To further evaluate the usefulness of the deployed system, we carried out a survey on the actual users of the system (Sections 6-7). The survey had both Likert scale and open-ended questions and about half of the actual users of the deployed system voluntarily participated in the survey.

Throughout the paper, we used the actual database of issue reports maintained by Softtech. Furthermore, all the survey results were obtained from the actual users in the field. We followed the same approach to evaluate our PELT-based technique to detect deteriorations in assignment accuracies, which, indeed, successfully detected three deteriorations each with a different cause (Section 8). To further evaluate the proposed approach, we also carried out controlled experiments, each of which was repeated 1000 times. We did this because in the data collected from the field, it was not always possible to determine whether there really were some deteriorations or not, and if so, what the nature of these deteriorations were. Therefore, the controlled experiments helped us further evaluate the proposed approach, as in these experiments, we knew both the nature of the deteriorations (e.g., sudden or gradual) and the exact point in time where they occurred.

## 10.2 Internal Validity

To circumvent the internal threats that may be caused by implementation errors, we used well-known and frequently used libraries. In particular, we used the Python scikit-learn (Pedregosa et al. 2011) library for preprocessing the issue reports and extracting the features; the scikit-learn (Pedregosa et al. 2011) and mlxtend (Raschka 2018) libraries for training the classification models; the lime (Ribeiro et al. 2016) library for creating the LIME-based explanations for the assignments; and the ruptures (Truong et al. 2018a) library for PELT-based change point detection.

In Section 3, we performed the same preprocessing steps and extracted the same set of features for all the classification algorithms used in the study. However, the performances of these classifiers might have been dependent on

the preprocessing steps used and the features extracted. On the other hand, we used well-known preprocessing steps, such as tokenization and removal of non-letter characters as well as stop words and extracted frequently used features, such as the bag-of-words model.

A related concern is that we used the default configurations of the aforementioned classifiers, except for the  $k$ -nearest neighbor and the stacked generalization classifiers where we used cosine similarity and empirically tuned  $k$  for the former; and logistic regression as the level-1 algorithm together with the probabilities emitted by the level-0 classifiers for the latter. On the other hand, the performance of these classifiers might have been dependent on the underlying configurations. Note, however, that optimizing the configurations for these classifiers could have only generated better accuracies.

In the evaluations, as the correct team for a given issue report (i.e., as the ground truth), we used the team who actually closed the report. Some reports, however, might have needed to be processed by multiple teams before the reported issues could be fixed. Since in these situations, typically the last team in the chain closed the report, even if the initial assignment of the report was considered to be correct, it was counted as incorrect when computing the assignment accuracies. Note, however, that counting such assignments as correct could have only increased the accuracies.

When computing the amount of manual effort required for issue assignments, we did not take the amount of effort required for maintaining the knowledge base used by the IT-HD clerks into account. Therefore, the actual savings in person-months can be larger than the ones reported.

### 10.3 External Validity

One external threat is that IssueTAG was deployed at Softtech/IsBank only. Softtech, however, being a subsidiary of IsBank – the largest private bank in Turkey – is the largest software company of Turkey owned by domestic capital, maintaining around 100 millions of lines of code with 1.200 employees. Consequently, it shares many characteristics of large software development houses, especially the ones producing custom, business-critical software systems, such as having a large, continuously evolving code base maintained by dozens of development teams involving hundreds of software developers with hundreds of issue reports filed daily, each of which needs to be addressed with utmost importance and urgency.

Another possible threat is that issue reports at IsBank (thus, the ones used in this work) are created by the IT-HD clerks (Section 2). Although, this team is a non-technical team, they are specialized in creating issue reports by listening to the bank customers and employees. Therefore, the quality of the issue reports used in this study may differ from the ones directly created by, for example, the end-users of a system. However, many companies, especially the ones that produce business-critical software systems and that need to deal with a large number of issue reports, employ similar call centers. Furthermore,



all the issue reports used in this work were written in Turkish. However, we used simple text processing steps, such as tokenization and removal of non-letter characters and stop words. Therefore, the proposed approach can also be used with issue reports written in other languages.

#### 10.4 Conclusion Validity

All the issue reports we used in the experiments were the real issue reports collected from the field. After the deployment of IssueTAG, once an issue report was created by an IT-HD clerk, the assignment was automatically made by the system. There was no means that the deployed system could be bypassed or that the assignments made by the system could be changed by an IT-HD clerk. Note that the AST members could then reassign the issue reports if needed, in which case the initial assignments made by the system were considered as incorrect. The number of issue reports closed was an important performance metric for the AST members as well as for the development teams at Softtech. Consequently, as a part of the company's policy, the issue reports were required to be closed by the development teams, who actually resolved the reported issues. The stakeholders payed utmost attention to this matter. Therefore, the assignment accuracies reported in this work, reflect the actual accuracies obtained by IssueTAG in the field.

To further evaluate the deployed system, we carried out two surveys (Sections 6-7). Although 14 participants were involved in these surveys, these participants constituted about half (14 out of 30) of the AST members, who are the direct end-users of IssueTAG, dealing with the issue reports on a daily basis.

### 11 Related Work

Several works in the literature studied the issue assignment problem. These works use a variety of approaches to make the assignments, including Naive Bayes classifiers (Murphy and Cubranic 2004; Anvik et al. 2006), Bayesian Networks (Jeong et al. 2009), Support Vector Machines (Anvik et al. 2006; Jonsson et al. 2016), and information retrieval-based approaches (Chen et al. 2011; Kagdi et al. 2012; Nagwani and Verma 2012; Shokripour et al. 2012; Canfora and Cerulo 2006; Linares-Vásquez et al. 2012; Xie et al. 2012; Xia et al. 2013), Expectation Maximization (Anvik 2007), Nearest Neighbor classifiers (Anvik and Murphy 2011), Decision Trees (Ahsan et al. 2009), Random Forests (Ahsan et al. 2009), REPTrees (Ahsan et al. 2009), Radial Basis Function Networks (Ahsan et al. 2009), Neural Networks (Helming et al. 2010) and Ensemble-based classification (Jonsson et al. 2016).

These works, except for (Lin et al. 2009; Helming et al. 2010; Jonsson et al. 2016; Dedík and Rossi 2016), evaluated the proposed approaches by using the issue databases of open source projects. We, on the other hand, used the issue

reports filed for commercial, closed-source projects. Although the remaining works (Lin et al. 2009; Helming et al. 2010; Jonsson et al. 2016; Dedík and Rossi 2016), report on the results obtained on closed-source, commercial software projects, they do so by carrying out a retrospective analysis in an offline manner. We, on the other hand, deployed the proposed approach and shared both the results we obtained and the lessons we learnt regarding the practical effects of automated issue assignment in the field. Furthermore, to the best of our knowledge, our work is the first work carrying out user studies in this context.

Some of the aforementioned works use natural language explanations present in issue reports for assignments, such as one-line summary and description (Murphy and Cubranic 2004; Anvik et al. 2006; Canfora and Cerulo 2006; Ahsan et al. 2009; Baysal et al. 2009; Jeong et al. 2009; Lin et al. 2009; Matter et al. 2009; Helming et al. 2010; Anvik and Murphy 2011; Chen et al. 2011; Park et al. 2011; Bhattacharya et al. 2012; Linares-Vásquez et al. 2012; Nagwani and Verma 2012; Alenezi et al. 2013; Jonsson et al. 2016; Bettenburg et al. 2008a). Others also leverage categorical information, such as product, component, and version (Ahsan et al. 2009; Lin et al. 2009; Park et al. 2011; Jonsson et al. 2016).

In this work, we used natural language descriptions present in the issue reports, more specifically the one-line summaries and descriptions. We did not use any categorical information, e.g., product, component, and version information, because such information was not included in the issue reports; there were no fields in the issue reporting tool, requesting these types of categorical information. The reason was that with the collection of software products maintained by Softtech, which heavily interact with each other in a business-critical environment, sharing many resources, such as databases, file systems, and GUI screens, the boundaries of the products from the perspective of issue reporting were not clear at all. Further discussion on this can be found in Section 2.

Different sources of information have been also used for making the assignments. For example, (Tamrawi et al. 2011) model the technical expertise of individual developers and use these models together with the information about the developers who recently made changes in the code base. (Wu et al. 2011) infer a social network model of the developers using the comments they make on historical issue reports as well as the comments automatically generated at the time of the source code commits, to help with the assignments. (Baysal et al. 2009) use developers' preferences as an additional source of information, which are expressed by the ratings the developers gave for the issues they resolved.

We, in this work, deliberately used a single source of information, i.e., the natural language descriptions present in the issue reports, to simplify the design and implementation of the proposed system to the extent possible. This was a design decision we made to increase the reliability of the proposed system as the system needed to be deployed, making hundreds of assignments per day in a business-critical environment. However, we are currently in the

process of figuring out what types of additional sources of information could be used in an industrial setup to further improve the assignment accuracies.

There are also automated approaches for dealing with various other aspects of the issue report management process. One type of approaches aim to identify duplicate issue reports, which can help developers with 1) figuring out the number of actual issues reported; 2) assigning priorities; and 3) debugging (Podgurski et al. 2003). Generally speaking the problem of duplicate identification is casted to a clustering problem where similar reports are grouped together with the assumption that similar descriptions report the same (or similar) issues (Podgurski et al. 2003; Bettenburg et al. 2008b; Wang et al. 2008; Jalbert and Weimer 2008).

Other types of approaches mainly focus on better utilizing the available resources for resolving the reported issues. For example, some approaches aim to predict the severities of the issues (Lamkanfi et al. 2010; Menzies and Marcus 2008; Antoniol et al. 2008; Pandey et al. 2017), which, in this context, indicate the levels of impact the issues have on the development and release process. Others aim to predict the effort required to resolve the issues (Weiss et al. 2007; Giger et al. 2010; Zhang et al. 2013).

Note that the aforementioned problems, i.e., duplicate detection, severity identification, and effort prediction are different than the issue assignment problem addressed in this work. We, however, plan to conduct industrial-strength studies at IsBank and Softtech to evaluate the efficiency and effectiveness of these approaches.

## 12 Conclusion and Future Work

In this work, we have developed and deployed a system to automate the process of issue assignments at Softtech/IsBank. To this end, we first cast the problem to a classification problem and determined the classifier to be used in the deployed system by empirically evaluating a number of existing classifiers, which are known to perform well for the problem at hand, on the actual database of issues maintained by the company. We then carried out further studies to determine both the amount and time locality of the historical data required for training the underlying classification models. We finally deployed the proposed system by configuring it based on the results we obtained from these studies.

We observed that 1) it is not just about deploying a data mining-based system for automated issue assignment, but also about designing/changing the assignment process around the system to get the most out of it; 2) the accuracy of the system does not have to be higher than that of manual assignments in order for the system to be useful, which was further validated by the user studies we carried out on actual stakeholders in the field; 3) deploying such a system also requires the development of additional functionalities, such as detecting deteriorations in assignment accuracies in an online manner and creating human-readable, non-technical explanations for the assignments made,

for both of which we developed and empirically evaluated different approaches; 4) stakeholders do not necessarily resist change; and 5) gradual transitions can help stakeholders build confidence, which, in turn, facilitates the acceptance of the system.

One avenue for future research is to use additional sources of information to further improve the assignment accuracy; we are, in particular, interested in developing “human-in-the-loop” type of approaches. Another avenue is to carry out industrial-strength studies using the deployed system to evaluate the efficiency and effectiveness of the other related approaches in the field, including duplicate detection, severity identification, and effort prediction.

## References

- Ahsan SN, Ferzund J, Wotawa F (2009) Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine. In: 2009 Fourth International Conference on Software Engineering Advances, IEEE, pp 216–221
- Alenezi M, Magel K, Banitaan S (2013) Efficient bug triaging using text mining. *JSW* 8(9):2185–2190
- Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc YG (2008) Is it a bug or an enhancement?: a text-based approach to classify change requests. In: *CASCON*, vol 8, pp 304–318
- Anvik J (2007) Assisting bug report triage through recommendation. PhD thesis, University of British Columbia
- Anvik J, Murphy GC (2011) Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 20(3):10
- Anvik J, Hiew L, Murphy GC (2006) Who should fix this bug? In: *Proceedings of the 28th international conference on Software engineering*, ACM, pp 361–370
- Baysal O, Godfrey MW, Cohen R (2009) A bug you like: A framework for automated assignment of bugs. In: 2009 IEEE 17th International Conference on Program Comprehension, IEEE, pp 297–298
- Bettenburg N, Just S, Schröter A, Weiss C, Premraj R, Zimmermann T (2008a) What makes a good bug report? In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, ACM, pp 308–318
- Bettenburg N, Premraj R, Zimmermann T, Kim S (2008b) Duplicate bug reports considered harmful... really? In: 2008 IEEE International Conference on Software Maintenance, IEEE, pp 337–345
- Bhattacharya P, Neamtiu I, Shelton CR (2012) Automated, highly-accurate, bug assignment using machine learning and tossing graphs. *Journal of Systems and Software* 85(10):2275–2292
- Bishop CM (2006) *Pattern recognition and machine learning*. springer
- Breiman L (2001) Random forests. *Machine Learning* 45(1):5–32
- Breiman L (2017) *Classification and regression trees*. Routledge
- Canfora G, Cerulo L (2006) Supporting change request assignment in open source development. In: *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, pp 1767–1772
- Chen L, Wang X, Liu C (2011) An approach to improving bug assignment with bug tossing graphs and bug similarities. *JSW* 6(3):421–427
- Dedík V, Rossi B (2016) Automated bug triaging in an industrial context. In: 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, pp 363–367
- Giger E, Pinzger M, Gall H (2010) Predicting the fix time of bugs. In: *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, ACM, pp 52–56

- Helming J, Arndt H, Hodaie Z, Koegel M, Narayan N (2010) Automatic assignment of work items. In: International Conference on Evaluation of Novel Approaches to Software Engineering, Springer, pp 236–250
- Hocking TD, Schleiermacher G, Janoueix-Lerosey I, Delattre O, Bach F, Vert JP (2013) Learning smoothing models of copy number profiles using breakpoint annotations. *BMC bioinformatics* 14:164
- Jalbert N, Weimer W (2008) Automated duplicate detection for bug tracking systems. In: 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), IEEE, pp 52–61
- Jeong G, Kim S, Zimmermann T (2009) Improving bug triage with bug tossing graphs. In: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM, pp 111–120
- Joachims T (1998) Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning, Springer-Verlag, ECML'98, pp 137–142
- Jonsson L, Borg M, Broman D, Sandahl K, Eldh S, Runeson P (2016) Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. *Empirical Software Engineering* 21(4):1533–1578
- Kagdi H, Gethers M, Poshyvanyk D, Hammad M (2012) Assigning change requests to software developers. *Journal of Software: Evolution and Process* 24(1):3–33
- Killick R, Fearnhead P, Eckley IA (2012) Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association* 107(500):1590–1598
- Lamkanfi A, Demeyer S, Giger E, Goethals B (2010) Predicting the severity of a reported bug. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE, pp 1–10
- Lavielle M, Ere G (2007) Adaptive detection of multiple change-points in asset price volatility. *Long Memory in Economics*
- Lin Z, Shu F, Yang Y, Hu C, Wang Q (2009) An empirical study on bug assignment automation using chinese bug data. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE, pp 451–455
- Linares-Vázquez M, Hossen K, Dang H, Kagdi H, Gethers M, Poshyvanyk D (2012) Triage incoming change requests: Bug or commit history, or code authorship? In: 2012 28th IEEE International Conference on Software Maintenance (ICSM), IEEE, pp 451–460
- Manning C, Raghavan P, Schütze H (2010) Introduction to information retrieval. *Natural Language Engineering* 16(1):100–103
- Matter D, Kuhn A, Nierstrasz O (2009) Assigning bug reports using a vocabulary-based expertise model of developers. In: 2009 6th IEEE international working conference on mining software repositories, IEEE, pp 131–140
- Menzies T, Marcus A (2008) Automated severity assessment of software defect reports. In: 2008 IEEE International Conference on Software Maintenance, IEEE, pp 346–355
- Murphy G, Cubranic D (2004) Automatic bug triage using text categorization. In: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering, Citeseer
- Nagwani NK, Verma S (2012) Predicting expert developers for newly reported bugs using frequent terms similarities of bug attributes. In: 2011 Ninth International Conference on ICT and Knowledge Engineering, IEEE, pp 113–117
- Pandey N, Sanyal DK, Hudait A, Sen A (2017) Automated classification of software issue reports using machine learning techniques: an empirical study. *Innovations in Systems and Software Engineering* 13(4):279–297
- Park Jw, Lee MW, Kim J, Hwang Sw, Kim S (2011) Costriage: A cost-aware triage algorithm for bug reporting systems. In: Twenty-Fifth AAAI Conference on Artificial Intelligence
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. (2011) Scikit-learn: Machine learning in python. *Journal of machine learning research* 12(Oct):2825–2830
- Podgurski A, Leon D, Francis P, Masri W, Minch M, Jiayang Sun, Bin Wang (2003) Automated support for classifying software failure reports. In: 25th International Conference on Software Engineering, 2003. Proceedings., pp 465–475

- Pressman RS (2005) Software engineering: a practitioner's approach. Palgrave Macmillan
- Raschka S (2018) Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *J Open Source Software* 3(24):638
- Ribeiro MT, Singh S, Guestrin C (2016) Why should i trust you?: Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, pp 1135–1144
- Shokripour R, Kasirun ZM, Zamani S, Anvik J (2012) Automatic bug assignment using information extraction methods. In: *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, IEEE, pp 144–149
- Tamrawi A, Nguyen TT, Al-Kofahi JM, Nguyen TN (2011) Fuzzy set and cache-based approach for bug triaging. In: *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, ACM, pp 365–375
- Ting KM, Witten IH (1999) Issues in stacked generalization. *Journal of artificial intelligence research* 10:271–289
- Truong C, Oudre L, Vayatis N (2018a) ruptures: change point detection in python. *arXiv preprint arXiv:180100826*
- Truong C, Oudre L, Vayatis N (2018b) Selective review of offline change point detection methods. *arXiv preprint arXiv:180100718*
- Wang X, Zhang L, Xie T, Anvik J, Sun J (2008) An approach to detecting duplicate bug reports using natural language and execution information. In: *Proceedings of the 30th international conference on Software engineering*, ACM, pp 461–470
- Weiss C, Premraj R, Zimmermann T, Zeller A (2007) How long will it take to fix this bug? In: *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, IEEE, pp 1–1
- Wolpert DH (1992) Stacked generalization. *Neural networks* 5(2):241–259
- Wu W, Zhang W, Yang Y, Wang Q (2011) Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking. In: *2011 18th Asia-Pacific Software Engineering Conference*, IEEE, pp 389–396
- Xia X, Lo D, Wang X, Zhou B (2013) Accurate developer recommendation for bug resolution. In: *2013 20th Working Conference on Reverse Engineering (WCRE)*, IEEE, pp 72–81
- Xie X, Zhang W, Yang Y, Wang Q (2012) Dretom: Developer recommendation based on topic models for bug resolution. In: *Proceedings of the 8th international conference on predictive models in software engineering*, ACM, pp 19–28
- Zhang H, Gong L, Versteeg S (2013) Predicting bug-fixing time: an empirical study of commercial software projects. In: *Proceedings of the 2013 international conference on software engineering*, IEEE Press, pp 1042–1051