Story and Task Issue Analysis for Agile Machine Learning Projects

Kushal Singla, Vinayak TM, Arpitha AS, Chetan Naik, Joy Bose
Samsung R&D Institute
Bangalore, India
vinayak.m@samsung.com

Abstract— **The usage of Agile methodology in planning and executing machine learning (ML) and data science related software engineering projects is increasing. However, there are very few studies using real data on how effective such planning is or guidelines on how to plan such projects. In this paper, we analyze data taken from several software projects using Scrum tools. We compare the data for data science/ML and non-ML projects, in an attempt to understand if data science and ML projects are planned or executed any differently compared to normal software engineering projects. We also perform a story classification task using machine learning to analyze story logs for agile tasks for several teams. We find there are differences in what makes a good ML story as opposed to a non ML story. After analyzing this data, we propose a few ways in which software projects, whether machine learning related or not, can be better logged and executed using Scrum tools like Jira.**

Keywords— *scrum, machine learning project, software engineering, agile methodology*

I. INTRODUCTION

The number of software engineering projects having data science or machine learning (ML) is growing at a fast pace [1]. Agile methodology [2] has been very popular recently as a means for software development, because of advantages such as flexibility and rapid prototyping cycles. It has become the de facto standard for many new projects [3] at many big companies. Each of these requirements or wish list of the software product is tracked as the "Product Backlog" in agile. The requirement is termed as "User Story" [4]. In this methodology, pending work is divided into units of stories, tasks and subtasks, to be completed in fixed time units or cycles called sprints. This makes it easier to efficiently track the work progress and get customer feedback.

There have been a few studies so far at the effectiveness of agile methodology for software development, but to our knowledge they do not focus on data science or ML projects. Since ML is still an emerging field with its own best practices and processes, it is useful to examine the effectiveness of agile methodology when applied to such projects.

In this paper, we analyze logs of agile tools from teams within our organization, incorporating data from machine learning (ML) as well as non-machine learning (non-ML) software teams. All teams practice agile methodology in executing their projects. This approach enables us to isolate and study the characteristics of machine learning projects executed by following agile methodology. We are interested in understanding what are the unique features of agile ML projects as compared to non-ML, and how do such projects perform when measured with various parameters. We use data from stories, tasks and subtasks collected from a popular scrum tool, named Jira [14], for our analysis. Further, we use machine learning techniques to determine what makes a good story as opposed to a bad story

The rest of this paper is organized as follows: in section 2 we review related work in the area of agile methodology and machine learning projects. In section 3 we examine our method for data collection on ML and non-ML agile projects. Sections 4 and 5 describes some results from our data analysis of issues using logs taken from several teams. Section 6 summarizes the learnings, discussing ways in which agile methodology could be made more useful and relevant for machine learning projects. Finally, section 7 concludes the paper.

II. RELATED WORK

There are a number of online articles and blog posts [5-8] examining the feasibility and issues connected with agile machine learning. Most of them provide advice on how to get a successful machine learning project that takes advantage of agile methodology.

Lucassen [9] proposed a theoretical framework to consider while creating stories using NLP techniques. Developers at Microsoft proposed a framework called Team Data Science Process (TDSP) [15] to plan a data science, machine learning or analytics project efficiently using Agile principles. Their approach incorporated built-in machine learning specific terminology such as feature engineering, data exploration etc. Bullock [16] argued that ML projects can benefit from boosted productivity gained from agile processes.

However, the above works lack in studies and data from real agile software development systems and analysis of the issues faced by them.

Schleier-Smith [10] studied the feasibility of agile machine learning in real time apps, taking as example the development and design of a dating application on a mobile phone. The study flagged challenges such as difficulty in getting training data and long deployment cycles. They also proposed a generic agile cycle for machine learning projects including continuously observing problems with existing ML models, defining new features, followed by training, testing and redeploying new ML models with the new features. However, this too did not collect data on the project lifecycle characteristics of real life agile ML related development projects.

There are also a few studies of machine learning techniques applied to different use cases related to software engineering. The study by Wen et. al. [11] reviewed the use of machine learning models applied to estimation of how long it might take to complete software projects. Malhotra [12] applied machine learning methods to improve software quality by predicting faults. In this paper too, we use ML to classify story logs taken from multiple teams.

There are also a number of empirical studies of agile systems, such as by Dyba [13], but they do not cover machine learning projects as a separate category. Also, the backlog and the stories are analyzed differently. This is the gap we seek to fill in this paper. In our approach, we analyze the story with all required and related attributes such as story title and acceptance criteria. The user story is complete when it has all the attributes including Acceptance Criteria, Story Points, Description, Epic attached to it.

III. METHODOLOGY

In order to study issues and characteristics related to machine learning projects using agile methodology, we collected data over a period of 6 months from multiple teams of two kinds: the first kind working on exclusively machine learning projects and consisting mainly of data engineers and data scientists, and the other working on conventional software projects (mainly analytics related) not involving machine learning specifically, and consisting largely of software engineers.

All the teams followed agile methodology using scrum, with daily stand up meetings, two week long sprints, a sprint planning meeting at the beginning of each sprint and a demo and retrospective meeting at the end of each sprint.

We collected data using the logs from the Jira software [14], a popular issue tracking and product management software developed by Atlassian for Agile projects. We exported the JIRA logs to Microsoft Excel and finally to CSV format, and then analyzed them using machine learning tools.

For our analysis, we used manually labelled data by the software engineering team, which used certain criteria to label a story or task in Jira as properly logged or not. We trained the dataset using the manually labelled data, using machine learning, and used it to predict the labels on the test set. Finally, we analyzed the results to determine which features from the data constitute a properly logged story, for both ML and non ML projects. Our intention was to use this knowledge to formulate guidelines for proper planning and logging of projects using the Agile tools, as well as to use the trained model to automate the labeling of a story or task as properly logged or not.

In the following section, we describe the results of our analysis and the trends detected.

IV. STORY CLASSIFICATION

As mentioned, we analyse story and task logs taken from multiple teams, both ML and non-ML. The logs are taken from the Jira tool [14]. We classify the logs using machine learning by analysing the words used and to judge what counts as a 'good' story as opposed to a 'bad' or poorly logged story. This is important since good descriptions allow for more efficient tracking and hence better software engineering practice, and vice versa.

In deciding whether to manually label a story as good or bad, we followed standard software quality guidelines for scrum. These include criteria such as the story and task descriptions have to be complete, there should be acceptance criteria clearly defined and so on.

We use machine learning to classify the labelled story dataset in order to understand which features make for a good or bad story. This knowledge can be used in the future to automate the labeling process with high accuracy.

A. Data Collection and Preprocessing

For this experiment, we collected data from 59 scrum projects for 4 sprints with a total of 1022 stories. Selected Scrum projects are of one of the following types: Development, Research, and Optimization. The Scrum projects are taken from a variety of software domains including Internet of Things (IoT), Voice Service, Payment, Security, Camera, OS Platform, Cloud platform, Big data analytics, Services, etc.

We considered the following:

- 16 Scrum projects of ML teams for 4 sprints with a total of 254 stories
- 43 Scrum projects of non ML teams with a total of 768 stories.

Each story is manually labelled by a dedicated team into 'good' or 'bad', based on how well or how precisely the story has been described in the Jira tool. A 'good' story is one which explains the who/what/why/validation aspect of the task, and a 'bad' story is one that has an incomplete description.

For our analysis, we used the following features to construct our dataset: Story title, story description text, story acceptance criteria, title length, description length, acceptance criteria length, title punctuation count, description punctuation count, acceptance criteria punctuation count, is title same as description.

We have used a combination of the textual features with numeric features. We obtained the textual features from the story title, description etc. We obtained the numeric features by running some functions such as length on the text features.

The pre-processing steps we used are the following:

- Story title : stop word removal, stemming
- Story description text: stop word removal, stemming
- Story acceptance criteria: stop word removal, stemming

We used the nltk package of Python for the preprocessing steps. As part of the pre-processing, we also perform standard scaling for the following numerical features - title length, description length, acceptance criteria length, title punctuation count, description punctuation count, acceptance criteria punctuation count.

B. Training and Testing

For the classification of the story text into good or bad, we use a support vector machine (SVM) model. SVM is commonly used as a fast and reasonably accurate machine learning algorithm suitable for most classification tasks.

For the story logs, we divided the dataset into 80% training set and 20% test set. We used a linear SVM with parameter C = 0.1 using Python's Scikit-learn library. We trained the mode on the training set and tested using the testing set.

We built 3 models – the first using only the ML stories, second using the non-ML stories and the third using both types of stories. Threshold of the model is set at 50%.

The following subsections give the results of the test. Class is the label of a story (1 is for good story, 0 for bad story). Precision is the fraction of stories which are labeled correctly out of the total number of stories labeled to a certain class. Recall is the amount of stories labeled correctly out of all the stories of a class. F1 score is an evaluation measure which is the harmonic mean of the precision and the recall. Support specifies the number of items of a specific class.

TABLE I. CLASSIFICATION ACCURACY FOR THE ML STORY ANALYSIS TASK USING SVM

Class	Precision	Recall	F1 score	Suppor t
0	1.00	0.94	0.97	18
1	0.95	1.00	0.97	19
Total	0.97	0.97	0.97	37

TABLE II. CLASSIFICATION ACCURACY FOR THE NON-ML STORY ANALYSIS TASK USING SVM

Class	Precision	Recall	F1 score	Suppor t
0	1.00	0.95	0.98	44
1	0.97	1.00	0.98	59
Total	0.98	0.98	0.98	103

TABLE III. CLASSIFICATION ACCURACY FOR THE ML AND NON-ML STORY ANALYSIS TASK USING SVM

Class	Precision	Recall	F1 score	Suppor t
0	1.00	0.94	0.97	65
1	0.95	1.00	0.98	82
Total	0.97	0.97	0.97	147

C. *Classification Results for ML, non-ML and Combined*

The results of the classification task for the stories of the test set for ML (data taken from ML teams), non-ML (data taken from non ML teams) and combined are given in tables 1-3. As we can see, we obtain overall F1 scores of 97%, indicating the SVM model has been able to classify the data with high accuracy.

V. LIST OF FEATURES FOR A GOOD (ML AND NON ML) STORY AND BAD STORY

Table 4 gives the list of top 3 features that make a good ML story verses those that make a bad ML story, as obtained from our trained SVM model. We also mentioned the model coefficients for each term that we get from the SVM model.

TABLE IV. LIST OF FEATURES FOR A GOOD ML STORY AND BAD STORY

Features for a bad ML story	Features for a good ML story
-0.10 quality	1.02 acceptance criteria length
-0.09 improve	0.18 title length
-0.08 domain	0.15 tech

As we can see, good ML stories have in common the following features:

- They have a good count of acceptance criteria
- Their title length is sufficiently high, meaning the story titles are sufficiently descriptive
- They use technical terms to make the story more precise

Bad stories on the other hand, have the following salient features:

- The quality of the story descriptions is poor
- The domain of the story is often not clear

However, the bad features have lower model coefficient scores from the SVM model as compared to the good features, indicating that the weight of the good features like acceptance criteria is higher, while deciding if a story is good.

The above gives a clue as to how stories involving ML can be written more effectively in an Agile tool.

Table 5 gives the list of top 3 features that make a good Non-ML story Vs those that make a bad non-ML story.

TABLE V. LIST OF FEATURES FOR A GOOD NON-ML STORY AND BAD STORY

Features for a bad non-ML story	Features for a good non-ML story
-0.14 approved	1.2 acceptance criteria length
-0.13 logger	0.55 title length
-0.12 quick	0.25 submission

As we can see, good Non-ML stories have in common the following features:

- They have a good count of acceptance criteria
- Their title length is sufficiently high, meaning the story titles are sufficiently descriptive

This is somewhat similar to the features identified in the case of ML stories, except that the domain and technical content is more emphasized in ML stories.

Finally, table 6 gives the list of top 3 features taken from the combined dataset (ML plus Non-ML) to story Vs those that make a bad story.

TABLE VI. LIST OF FEATURES FOR A GOOD (ML AND NON-ML) STORY AND BAD STORY

Features for a bad story	**Features for a good story**
-0.21 responses	1.04 acceptance criteria count
-0.19 user	0.46 title length
-0.17 working	0.30 want

As we can see, good combined (ML and Non-ML) stories have in common the following features:

- They have a good count of acceptance criteria
- Their title length is sufficiently high, meaning the story titles are sufficiently descriptive

Bad stories on the other hand, have the following salient features:

- The responses are not well recorded

On the basis of this, we can form guidelines as to how to write stories properly, by logging proper schedule in the tool, with proper and precise language of titles, acceptance criteria and description.

VI. GUIDELINES TO MAKE AGILE MORE USEFUL FOR SOFTWARE PROJECTS (BOTH ML AND NON ML)

In this section, based on the previous analysis, we propose a few suggestions or guidelines in which agile methodology can be made more useful for machine learning projects and teams. Some of our recommendations are as follows:

- Clearly defined acceptance criteria for stories, both ML and non-ML. Ideally it should be something that is objective and measurable.
- Bigger titles that are more descriptive in relation to the actual content of the work
- Using more technical terms in the titles and descriptions, as opposed to generic terms
- Clearly labeling the domain of the story or task

VII. CONCLUSION AND FUTURE WORK

In this paper, we have studied the Jira logs for multiple machine learning (ML) and non ML teams, analyzing the trends and their reasons. We have also provided a few suggestions by which Agile can be made more useful or relevant for machine learning teams and projects.

In future, we plan to have a bigger study with more teams to get a more generalized insight on the trends. We also plan to study the impact of our recommendations by conducting an A/B test, and studying the Jira logs with and without following these recommendations.

REFERENCES

[1] L. Columbus, “Roundup Of Machine Learning Forecasts And Market Estimates,” Forbes, Feb 2018 [Online]. Available: forbes.com/sites/louiscolumbus/2018/02/18/roundup-of-machine-learning-forecasts-and-market-estimates-2018/#5a32a37d2225. [Accessed: October 20, 2019]

[2] Wikipedia, “Agile software development.” [Online]. Available: en.wikipedia.org/wiki/Agile_software_development. [Accessed: October 20, 2019]

[3] C. Cardoza, “Agile is becoming the de facto standard for software development, according to VersionOne.” SD Times, March 2015. [Online]. Available: sdtimes.com/agile/agile-is-becoming-the-de-facto-standard-for-software-development-according-to-versionone/

[4] Wang, X., Zhao, L., Wang, Y. and Sun, J. “The role of requirements engineering practices in agile development: an empirical study.” *Requirements Engineering* (pp. 195- 209). Springer, Berlin, Heidelberg.

[5] G. Villamin, “How to Use Agile For Successful Machine Learning Projects.” B2C, May 2018. [Online]. Available: business2community.com/strategy/how-to-use-agile-for-successful-machine-learning-projects-02067366. [Accessed: October 20, 2019]

[6] R. Hinds. “Agile Machine Learning: From Theory to Production.” DZone, June 2017. [Online]. Available: dzone.com/articles/agile-machine-learning-from-theory-to-production[Accessed: October 20, 2019]

[7] YellowRoad, “When Agile Meets Machine Learning.” Towards Data Science, June 2017. [Online]. Available: towardsdatascience.com/when-agile-meets-machine-learning-2af111bddeec. [Accessed: October 20, 2019]

[8] P. Barba, “The Case for Agile Machine Learning.” Dataversity, June 2018 [Online] Available: dataversity.net/case-agile-machine-learning. [Accessed: October 20, 2019]

[9] Lucassen, G., Dalpiaz, F., van der Werf, J.M.E. and Brinkkemper, S., 2015, August. Forging high-quality user stories: towards a discipline for agile requirements. In Requirements Engineering Conference (RE), 2015 IEEE 23rd International (pp. 126-135). IEEE

[10] J. Schleier-Smith. “An architecture for Agile machine learning in real-time applications.” In *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2059-2068). August 2015. ACM.

[11] J. Wen, S. Li, Z. Lin, Y. Hu and C. Huang. “Systematic literature review of machine learning based software development effort estimation models.” *Information and Software Technology*, 54(1), pp.41-59, 2012.

[12] R. Malhotra and A. Jain. “Fault prediction using statistical and machine learning methods for improving software quality.” *Journal of Information Processing Systems*, 8(2), pp.241-262. 2012

[13] T. Dyba and T. Dingsoyr. “Empirical studies of agile software development: A systematic review.” *Information and software technology*, Elsevier, 50(9-10), pp.833-859.

[14] Altassian. “Agile tools for software teams.” Jira software| Atlassian. [Online]. Available: atlassian.com/software/jira/agile. [Accessed: October 20, 2019]

[15] Microsoft Azure documentation, “Agile development of data science projects.” 28 Nov 2017 [Online]. Available: docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/agile-development. [Accessed: October 20, 2019]

[16] T. Bullock, J. Justice, A. Sutherland. Scruminc, “Why Scrum is the Best Way to Create AI.” [Online]. Available: scruminc.com/scrum-in-ai-artificial-intelligence. [Accessed: October 20, 2019]