Statistical Model Update Optimization in Industrial Practice

Bachelor of Science Thesis in Software Engineering and Management

Maria-Bianca Cindroi
Robinson Iheanacho Mgbah

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Statistical Model Update Optimization in Industrial Practice
Maria-Bianca Cindroi
Robinson Iheanacho Mgbah

Supervisor: MOHAMMAD MOUSAVI
Examiner: RICHARD TORKAR

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Acknowledgements

We would like to thank our academic supervisor, Mohammad Mousavi, and our company supervisors, David Andersson and Michael West, for all the guidance and availability throughout the project. Additionally, we thank every participant from the troubleshooting teams from the company the thesis has been done in.

Statistical Model Update Optimization in Industrial Practice

Maria-Bianca Cindroi
guscincn@student.gu.se
University of Gothenburg, Software Engineering and Management
DIT 565 Software Engineering and Management Bachelor Thesis Project

Robinson Iheanacho Mgbah
gusmgbih@student.gu.se
University of Gothenburg, Software Engineering and Management
DIT 565 Software Engineering and Management Bachelor Thesis Project

Abstract—**This thesis presents a study done on optimizing machine learning model updates. The department of Quality and Functionality in a multinational telecommunication company is searching for an optimal solution to the problem of when, and how, to trigger a training cycle of a statistical model on their test execution dataset.**

We have investigated techniques regarding the possibilities of optimizing a statistical model update. A case-study has been conducted, using a telecommunication company as a case subject company.

Summary Here:

Keywords— ***machine learning model; optimization; changing models***

Introduction

Considering the attention that machine learning is receiving in the IT world today, effective techniques and best practices need to be established in regard to how and when the models that have been created should be built, discarded or updated in order for them to be relevant to the industry.
Thus, the department of Quality and Functionality in our case subject company is searching for an optimal solution to the problem of when, and how, to trigger a training cycle of a statistical model on their test execution dataset.
Massive amounts of data are collected from the internal failed test-cases performed on the radio base station in the continuous integration flow, continuous deployment sites, and customers' trouble reports. The case subject company is interested in finding out the optimal timing of when to refit a model on a recent dataset for the model in order to minimize troubleshooting time for the troubleshooting teams. The data is stored in an artifact storage, with references to it being summarized in datastore. From the datastore, a sample dataset is fetched and used for training the model. The training data in conjunction with the model fitting procedure produce a serialized model. Given the serialized model, a deserialization [1] process is performed, and the deserialized model is used to make predictions on incoming data in the fault-tickets. The data that will be used for this study is the data received from the ticketing system, and/or analysis results. For a model to predict accurately, the data on which the predictions are made must have a similar distribution as the data on which the model has been trained. Since data distributions change over time, deploying a model should be a continuous process. The department is interested in finding out the optimal timing of performing a model update in order to avoid redundant training cycles and adapt the model to shifting new data.

Generally, best practices in updating of machine learning models mostly involve observation drawn from experiments. A model update could be triggered on a pre-set time window [16].

1) *Case Company*

The thesis has been done in a multinational telecommunication company. Hundreds of development teams in the company are delivering code daily, which leads to a high probability of commits that introduce bugs and/or break legacy.

2) *Background*

In the company's ways of working, there exists a ticketing system where faults could be raised on specific categories. There can be currently three types of faults: product, environment, and test. The ticketing system facilitates product development by visualizing relevant data for the given specific discovered fault over all the product deliveries.
On a weekly basis, around 300000 data points are being piped from different data-sources to a common data cluster owned by a third-party company and visualized through the ticketing-system by posting queries to the cluster. The queries are being posted through an API that returns relevant data to a given query.

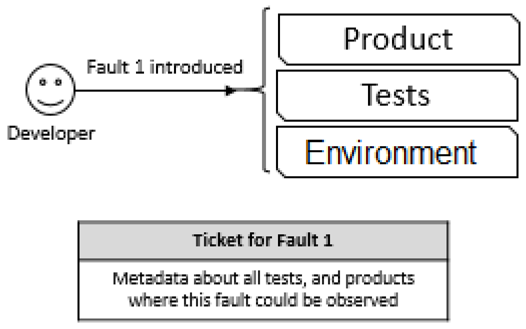

Fault 1 introduced
Developer
Product
Tests
Environment
Ticket for Fault 1
Metadata about all tests, and products where this fault could be observed

Fig 1. Example of how the ticketing system works

A product fault represents a fault observed in the hardware or the software of any of the products the case-study company delivers to its customers.
For the company to have happy-customers, fault-free products must be delivered. This is currently being achieved through extensive pre-delivery testing. A test-fault represents a fault observed in any of the failed tests on pre-delivery products.

An environment fault represents a fault observed in the development environment of the product, or in the internal-environment of the product itself.
Currently, troubleshooting takes a long time since a big part of the tickets that are being manually raised do not have the right fault tags. This makes tickets end up in wrong organizations, and software faults take longer time to be resolved due to redirected tickets around the company. A machine learning model is currently being implemented to automatically predict the accuracy of the software faults in the tickets that are being raised based on the metadata and the data provided in the ticket. This model will require an update on the new dataset that is being fed with, but an optimal time to perform the model update has not been established.

3) *Purpose of the study*

The purpose of this thesis is to devise a technique to determine when an update should be triggered for a machine learning model. When it comes to the above-mention project, the purpose of this thesis is to devise a method to when an update should be performed, in order to help in troubleshooting, and software testing.
Generally, best practices in machine learning model update procedures are mostly involving observation drawn from experiments assuming trial-and error methods. A model update is mostly triggered on a pre-set time window. The goal of our technique is to improve machine learning model updates. Experiments must be implemented for results to be drawn in this regard. This experiment can involve: simulating model update on historical data and applying the proposed solutions to it; but also proposing an optimized technique for the company's solution which improves the overall statistical accuracy of model predictions in the given project and the experienced quality of its predictions when used as input for problem solving performed by human-troubleshooters.

4) *Research questions*

For the model to remain relevant to solving the problem, model updates must be performed periodically. The right moment for the model update represents the time window when the update is performed at the most resource-optimal and model data-relevant moment. Model update requires resources such as time, data and CPU usage. By performing this update at the right moment, these resources are used optimally. A model data-relevant moment is when there is a significant change in the classification results.

Given a statistical model containing information about software, and software test failures, when, and how should an optimal model update be triggered for the model to still be accurate, and capable of classifying the failures accordingly?

The following sub-questions have been deduced:
RQ1 When should the model update be automatically triggered?
RQ1.1 Is there a statistical relationship between model accuracy and the amount of data used during model fitting?
RQ1.2 Is there a co-occurrence between the model accuracy and the trend difference between new and old data?
RQ1.3 Should the optimal model update moment be triggered in a predefined time window?
RQ2 How should the update be triggered? Should the model be locked when the update is performed to avoid breaking it or is it safe to update the model without locking it to avoid high resource usage?
RQ3. Does the possibility of having a trade-off between the method in which the updating is performed (RQ2), and the time-window when it is performed (RQ1) provide the most optimal solution for the model update? Should the model update be custom-made for each scenario to improve resource efficiency**?**

In the above-mentioned questions, we assume that the data model will rapidly quantify since the data sources increase at a fast pace.

Literature Review

1) *Problem Domain Literature*

With the assistance of both supervisors, literature from the problem domain have been identified:

1. Hall Daumé III, A course in machine learning
2. Zi Yuan, Lili Yu, Chao Liu, Linghua Zhang, Predicting Bugs in Source Code Changes with Incremental Learning Method
3. Walter Daelemans, Véronique Hoste, Fien De Meulder, Bart Naudts, Combined Optimization of Feature Selection and Algorithm Parameters in Machine Learning of Language
4. Véronique Hoste, Optimization Issues in Machine Learning of Coreference Resolution
5. R. Polikar, L. Upda, S. S. Upda and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks"
6. Learning from Time-Changing Data with Adaptive Windowing(2006), Albert Bifet, Ricard Galvada
7. Concept Drift Detection and Model Selection with Simulated Recurrence and Ensembles of Statistical Detectors (2013), Piotr Sobolewski & Michal Wo´zniak
8. Learning under Concept Drift: an Overview (2010) ,Indr˙eˇ Zliobait˙e
9. The problem of concept drift: definitions and related work (2004), Alexey Tsymbal
10. Sample-based software defect prediction with active and semi-supervised learning Ming Li, Hongyu Zhang, Rongxin Wu, Zhi-Hua Zhou
11. Optimizing Classifier Performance via an Approximation to the Wilcoxon-Mann-Whitney Statistic, Lian Yan, Robert Dodier, Michael C. Mozer, Richard Wolniewicz

Within machine learning, one of the prevalent issues has been handling concept drift.
Overtime, underlying distribution datasets upon which models are built could change with time. When these changes occur,

the models built on the old data become inconsistent with new data which in turn requires regular update for these models [6]. This problem is known as concept drift.
Since models deal with information, they need to be updated whenever they are fed with new data which differs significantly from the data used to train them. When a model is built, it is immutable but certain parameters and information used in the building process are stored, these are then applied to the new data which results in a model retrain [7].
Some studies have been done in other to understand important factors that relate to model update. Since model updates require resources such as time and CPU power. It is important to learn and understand the factors which play important roles in model update. For example, how often should model updates be performed to ensure relevance in the model, should this update be done manually or automatically and how can this update be done in the most efficient way keeping the use of resources at an optimized level [20].

2) *Literature on potential solution approaches*

The above-mentioned literature on Machine Learning and bug prediction has been analyzed, and the literature on potential solution approaches implemented by other parties have been identified:

1. Hall Daumé III, A course in machine learning, Chapter 7
2. Albert Bifet, Ricard Galvada , Learning from Time-Changing Data with Adaptive Windowing(2006)
3. Piotr Sobolewski & Michal Wozniak, Concept Drift Detection and Model Selection with Simulated Recurrence and Ensembles of Statistical Detectors (2013),
4. ,Indr ̇eˇ Zliobait ̇e, Learning under Concept Drift: an Overview (2010)

The Categorization Project

The categorization project is a project meant to categorize failed test-cases under the main three labels: test, product, environment based on the failed test-cases metadata using a machine learning method. The method is currently implemented using multinomial logistic. The data has been collected from 2017, while the categorizing project has been in production since February 2018.
The project has been developed in Python3.6, while the database is implemented Hadoop HDFS, MongdoDB, and Elasticsearch solutions.

Research Methodology

In order to answer the research questions, research methodology techniques as questionnaires (in workshops) and experiments have been employed.

1) *Workshop Setup*

In order to draw observations in regards to what extend the user trust and uses the prediction algorithm in daily work, an initial workshop will be organized where users are asked to answer questions regarding how long the troubleshooting takes, and what the accuracy of the current predictions at the time.
In the case in which relevant information about approaches on how to optimize the model update, modifications will be done to the in-production model update method according to the deducted results. The prediction algorithm will be allowed to run for 1-2 weeks, after which a secondary workshop will be held in order to conclude if the users have observed any modifications in the predictions. The workshop setup and the set of question will be the same as that of the first workshop.

The workshops were designed to follow this pattern: developers from several troubleshooting teams were asked to answer a questionnaire. Considering that most of the participants are stationed in other countries, support has been made available for remote-participation.
A) Workshop 1
Workshop 1 follows the above-mentioned pattern. In order to support remote users, the questionnaire was introduced during the morning scrub meetings on the 27th of April and were asked to answer before 4th of May. The developers that are part of troubleshooting teams currently sitting in Gothenburg, were asked to join a locally organized workshop on the 4th of May.

The set of questions used in the workshop are given below:

1. How long does it take to currently troubleshoot with the help of the system (in hours)?
2. On a scale of 1 to 5, how much do you trust the predicted problem type presented in the ticket?
   1. I do not trust it at all.
   2. I trust it to some extent.
   3. I somehow trust it.
   4. I trust it to a great extent.
   5. I fully trust it.
3. As a ways of working, do you double check the accuracy of the predicted problem type presented in the ticket?
   1. Yes
   2. No
   3. Other
4. How often have you encountered a wrongly predicted problem type?
   a. 1 in 50
   b. 1 in 10
   c. 1 in 4
   d. 1 in 2
5. Is there anything that you will add in order to make the system easier to use?

2) *Simulation Setup*

Using the in-production model validator, simulations have been performed using different input parameters in a sliding-window manner. An initial step length was chosen to iterate over the input batch of training data. The validation data was predicted using the model's knowledge at the current state. The result of this simulation was a number representing the proportion of accurate predictions. The step length and the validation data were of the same size.

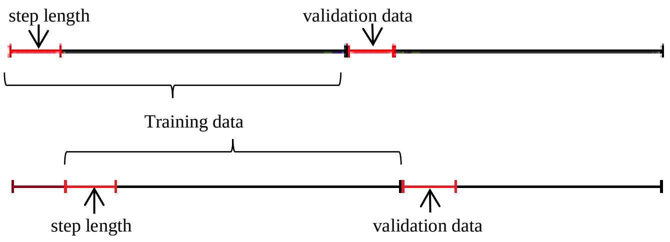


Fig 2. Model validating in a sliding window manner

A statistical test was used to validate these hypotheses. A non-parametric test was chosen for the analysis since no assumptions could be made about the normality of the data distribution. Due to concept drift, no assurance that data remains the same overtime can be expected. Randomness in data affects the reliability of the conclusion, therefore it is important to employ statistical tests to rigorously assess whether these results are indeed reliable [8]. The Mann-Whitney U test has been chosen in order to validate the results of the Kruskal-Wallis test performed above. because it is a preferred choice when comparing more than 2 independent data samples [9] in order to check if they come from identical populations.

The case company has started collecting relevant data in 2016(See Appendices 5. Distribution of data points from the beginning of time), but relevant data points to the categorization project are being observed from September 2017 (See appendices 6. Distribution of relevant data points to the categorization project). The data used in this thesis is mostly data collected since January 2018(see Appendices 7. Distribution of data relevant to this thesis) due to the fact that in January there have been efforts put into standardizing the collected data among all stakeholders of the data warehouse.

In order to verify the results, a Kruskal-Wallis test has been used to check if there is any simulation group that comes from a different population distribution than the others. A two-tailed Man-Whitney U test was also employed to check if a statistical relationship against each simulation result combination may be observed.
The Mann-Whitney U test is a preferred choice when comparing 2 independent, small [10], ordinal data samples [11].

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2}, \text{ where:}$$

R = the sum of ranks in the sample
n_i = sample size for sample i

Fig 3. Mann-Whitney U test formula

The Kruskal-Wallis [17] test has been chosen due to the fact it is a generalization of the Mann-Whitney U test which allows comparison between more than two data samples.

$$H = (N-1)\frac{\sum_{i=1}^{g} n_i(\bar{r}_{i\cdot} - \bar{r})^2}{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(r_i - \bar{r})^2}, \text{ where:}$$

n_i = is the number of observations in group i
r_{ij} = is the rank (among all observations) of observation j from group i
N = total number of observations across all groups
$\bar{r}_{i\cdot} = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i}$ is the average rank of all observations in group i
$\bar{r}_i = \frac{1}{2}(N+1)$ is the average of all the r_{ij}

Fig 4. Kruskal-Wallis test formula

The tests were analyzed using the standard critical values of the Mann-Whitney U table [12].
A significance level(α) of 0.05 indicates a 5% risk of concluding that a difference exists when there is no actual difference.

In the case in which large sample groups will be observed(n>20) where Mann-Whitney U two-tailed tests cannot be applied, and a statistically significant difference between the medians of the specific data group will be observed in the Wallis-Kruskal test result, a Z-test will be applied as the value of U approaches a normal distribution [22].

Result Analysis

We assume that the solution improves the overall statistical accuracy of model predictions and the experienced quality of its predictions when used as input for problem solving performed by human troubleshooters.

1) *Results Workshops*

A) Workshop 1

25 subjects participated in a questionnaire, among which 4 people sat in Gothenburg, and the rest were distributed over China, Poland, Stockholm, and Croatia.

1. How long does it take to currently troubleshoot with the help of the system (in hours)?

Out of the 100 subjects that have been invited to the workshop, 25% (25 subjects) have participated. Out of the 25 respondents, 32% of the respondents (8) have been unsure about the number of hours that currently takes to troubleshoot; the rest have replied according to the table below:

No of respondents	Percentage	Hours
1	4%	0.5
2	8%	2
2	8%	3
1	4%	3-4
1	4%	0.5-4
2	8%	4
2	8%	5
1	4%	6

1	4%	8
2	8%	12
1	4%	16-32
1	4%	40

Fig 5. Troubleshooting hours according to respondents

According to the table above, troubleshooters spend an average of 7.21 hours on troubleshooting a fault using the ticketing system.

2. On a scale of 1 to 5, how much do you trust the predicted problem type presented in the ticket?

Answer	Percentage	Number of respondents
I do not trust it at all.	0%	0
I trust it to some extent.	16%	4
I somehow trust it.	40%	10
I trust it to a great extent.	36%	9
I fully trust it.	8%	2

Fig 6. Users trust in the ticketing system

3. As a ways of working, do you double check the accuracy of the predicted problem type presented in the ticket?

Answer	Percentage	Number of respondents
Yes	80%	20
No	20%	5
Other	0%	0

Fig 7. Troubleshooting teams' usage percentage of the ticketing system

4. How often have you encountered a wrongly predicted problem type?

Answer	Percentage	Number of respondents
1 in 50	24%	6
1 in 10	44%	11
1 in 4	28%	7
1 in 2	4%	1

Fig 8. Percentage of observed wrongly set problem type

5. Is there anything that you will add in order to make the system easier to use?

Out of the 25 respondents, 22 respondents replied that there is no improvement they could add to the system, while 3 have suggested that more information of the failing test level in the ticketing system would be useful.

B) Workshop 2

In section 2 of this chapter, the simulation results have been analyzed and no statistically significant results have been observed that will enable an optimization technique for the predicting algorithm discussed in this thesis, thus attempting to analyze how the user perspective changes after the update method has been modified is not possible.

Conclusion: Even though the project in discussion is relatively new, it has succeeded in having international awareness with the company, but a slight reluctance to its results can still be observed among the 25 respondents, out of which 76% trust the system to a smaller or a greater extend.

Useful feedback has been received from the user about the system, where 3 of the respondents would appreciate a more extensive information of the failing test level in the tickets.

According to user feedback, the prediction algorithm has a low failure rate, with 68% of the users replying that wrongly predicted types are not commonly seen.

2)Simulation Results

The below-mentioned results have been deducted with the help of the validator of the algorithm model. This validator is meant to predict the test-data using the training-data set as a basis for training the model.

For RQ1.1 a csv file containing the dataset has been read from and used to validate the model, while for RQ1.3 the data is being fetched from the database using a date-based query. The date-based query is in the form of a json rest API, where a filter is being pushed through the API to the database and a result is received based on it.

RQ1 When should the model updating be automatically triggered?

RQ1.1 Is there a statistical relationship between model accuracy and the amount of data used during model fitting?

An observed statistical relationship between the different combinations of parameters used in the simulation is expected, whereas simulations with higher values as parameters are most likely to trigger a model update.

An initial training data set of 25686 data points was chosen. This data represents the same data sample the model has been trained on when it was first being developed. The model has been initially trained on a randomly chosen step length of 1000, with a validation data set of 10000. The reason this number has been chosen as a valid approach at this step is in order to simulate how the model has been working in production. This training data represents data fetched from the beginning of a training data batch size of different sizes (5000, 10000, 15000) was also chosen to be validated against a test data size of different step lengths in a sliding window manner. Different step lengths (1000, 2000, 3000) have been applied to each of the data batch sizes simulations, thus resulting into 9 combinations of sample populations. (See Appendices 1 *Results from simulations for RQ1.1* table for results).

In the following analysis, we note D(S) as the distribution of the sample X of a given population.

Kruskal-Wallis test:
In the below-mentioned hypotheses, S = {S1, S2, S3, S4, S5, S6, S7, S8 S9}, where S_a represents a sample found in Appendices 2 *Validation scores per sample group* table.

Null Hypothesis: The sample comes from populations with the same distribution, which makes the mean ranks coming from the same group, and model updating moments not being impacted by the tested step length parameters variation in combination with the size of the training data.

H₀: $\nexists S_a, S_o for\ which\ D(S_a) \neq D(S_o)$ where $S_o, S_a \in S$ and $S_o \neq S_a$

Alternative Hypothesis: At least one of the samples comes from a population with a different distribution than the others, which makes the model updating moments to be impacted by the values tested step length parameters variation in combination with the size of the training data.

H₀: $\exists\ S_a.\ \phi(S_a), S_o\ for\ which\ D(S_a)\ D(S_o)$ where $S_o, S_a \in S$ and $S_o \neq S_a$

Wallis-Kruskal test results	
H statistic	13.180578758965984
P-value	0.10578565764728853

Fig 13. Wallis-Kruskal test results

Since the p-value > α, there is not enough evidence to reject the null hypothesis that the differences between the medians are not statistically significant.

Combinations of simulation samples were tested against each other with a Mann-Whitney U test. The following hypothesis were considered for all tests;

Mann-Whitney U test:
In the below-mentioned hypotheses, T = {T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T20, T21, T22, T23, T24, T25, T26, T27, T28, T29, T30, T31, T32, T33, T34, T35, T36}, where T_a represents a sample found in Appendices 3 *Mann-Whitney U results* table.

Null Hypothesis: There is no statistical relationship observed in comparing the two data samples, which makes model updating moments are redundantly performed on sliding windows of a length less than 3000.

H₀: $\nexists T_o\ for\ which\ D(T_o) \geq Critical\ U(T_o), \forall\ T_o \in T$

Alternative hypothesis: Model updates should be performed on small samples (1000, 2000 respectively 3000) since a randomly selected value from S1 does not have an equal distribution with a randomly selected value from S2.

Hₐ: $\exists\ T_o\ for\ which\ D(T_o) < Critical\ U(T_o), \forall\ T_o \in T$

Statistical tests have been run on the following combinations of data samples.

$$ {}^{9}_{2}C = \frac{9!}{2!\,(9-2)!} = 36\ combinations $$

The results retrieved from the *Mann-Whitney U results* table in Appendices 2 have been compared against the critical values of the Mann-Whitney U two-tailed testing table [21].

Conclusion: Since none of the values retrieved from the simulations have a U statistic result smaller than the critical U for the specific samples, we conclude that none of the results have statistically significant evidence at α =0.05 to show that any two populations of data points are not equal. This implies that retrains could be done with step lengths higher than 3000 as this does not affect the model update.

RQ1.2 Is there a co-occurrence between the model updating moments and the trend difference between new and old data?

Since no statistical relationship has been observed at RQ1.1 between the model updating moments and the amount of new data, a co-occurrence between the trend difference between new and old data could not be observed either.

RQ1.3 Should the optimal model update moment be triggered in a predefined time window?

An observed statistical relationship between the different combinations of data-based updates, whereas simulations with higher values as parameters are most likely to trigger a model update.

Since the model has been in production for approximately 5 months (1st January – 1st July 2018), we have considered that the data collected for the past 5 months (5540534 data points) is being relevant to the prediction model. A training data batch size of different sizes (2-month, 3-months, 4-months) has been chosen to be validated against a test data size of different step lengths in a sliding window manner. Different step lengths (1-day, 1-week, 1-month) have been applied to each of the data batch sizes simulations, thus resulting into 9 combinations of sample populations. (See Appendices 4 *Validation scores per sample group for date-wise run simulations* for results).

In the following analysis, we note D(S) as the distribution of the sample X of a given population.

Kruskal-Wallis test:
In the below-mentioned hypotheses, S = {S1, S2, S3, S4, S5, S6, S7, S8 S9}, where S_a represents a sample found in Appendices 4 *Validation scores per sample group for date-wise run simulations* table.

Null Hypothesis: The samples come from populations with the same distribution, which makes the mean ranks coming from the same group, and model updating moments not being impacted by the tested step length parameters variation in combination with the size of the training data.

H₀: $\nexists S_a, S_o \, for \; which \; D(S_a) \neq D(S_o)$ where $S_o, S_a \in S$ and $S_o \neq S_a$

Alternative Hypothesis: At least one of the samples comes from a population with a different distribution than the others, which makes the model updating moments to be impacted by the values tested step length parameters variation in combination with the size of the training data.

H₀: $\exists \, S_a. \phi(S_a), S_o \; for \; which \; D(S_a) \; D(S_o)$ where $S_o, S_a \in S$ and $S_o \neq S_a$

Wallis-Kruskal test results	
H statistic	2
P-value	0.3679

Fig 14. Wallis-Kruskal test results

Since the p-value > α, there is not enough evidence to reject the null hypothesis that the differences between the medians are not statistically significant.

Combinations of simulation samples were tested against each other with a Mann-Whitney U test. The following hypothesis were considered for all tests:
Mann-Whitney U test:
In the below-mentioned hypotheses, T = {T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T20, T21, T22, T23, T24, T25,}, where T_a represents a sample found in Appendices 5 *Mann-Whitney U results for date-wise run simulations* table.

Null Hypothesis: There is no statistical difference observed in comparing the two data samples, which makes model updating moments are redundantly performed on sliding windows formed on date-based queries.

H₀: $!\exists T_o \; for \; which \; D(T_o) \geq Critical \; U(T_o), \forall \, T_o \in T$

Alternative hypothesis: Model updates should be performed on dynamically build date-based queries since a randomly selected value from a sample does not have an equal distribution with a randomly selected value from another sample.

Hₐ: $\exists \, T_o \; for \; which \; D(T_o) < Critical \; U(T_o), \forall \, T_o \in T$

25 test combinations based on 8 simulation data results have been tested against each other using Mann-Whitney U test-

The results retrieved from the *Mann-Whitney U results* table in Appendices 2 have been compared against the critical values of the Mann-Whitney U two-tailed testing table [21].

Conclusion: Since none of the values retrieved from the simulations have a U statistic result smaller than the critical U for the specific samples and the result of the Kruskal-Wallis test is not showing any significant difference between the medians of the data groups, we conclude that none of the results have statistically significant evidence at α =0.05 to show that any two populations of data points are not equal. After a quick-glance over the results, most of the refitting accuracy hit a score of less than random. This implies that a new prediction algorithm update is independent on the amount of time that has passed from the former update and cannot be optimized based on date-based queried. This could be due to the significant difference between the data points distribution (see Appendices 6 and 7) or due to the fact that over a period of time a specific fault could not be seen in the training data, thus the prediction algorithm unable to classify it accordingly in the test-data.

RQ2 How should the update be triggered? Should the model be locked when the update is performed to avoid breaking it or is it safe to update the model without locking it to avoid high resource usage?

The potential case in which performing a model swap could corrupt the model itself is when parallel programming is employed in development since an expensive calculation runs on less resources.
Two common approaches in parallel programming are either to run code via threads (*multithreading module* [14].) or multiple processes (*multiprocessing module* [13]), respectively. Comparative to the processes approach, which do not threaten with breaking the model due to the fact that each process runs completely independent from the others; threads could potentially cause conflicts in case of improper synchronization due to the fact that each thread has access to the same memory area.
The project used as a case study in this paper has been, as stated above, developed in Python3.6. Multiple threads do not run concurrently in Python due to the global interpreter lock (GIL [15]).
According to the Python Software Foundation [15], GIL is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once. The lock is necessary due to the fact that the *multithreading* library is development using CPython which does not have a thread-safe memory management.

Conclusion: Considering the points raised above, we conclude that the model will always be locked when the update is performed due to the global interpreter lock.

RQ3 Does the possibility of having a trade-off between the method in which the updating is performed (RQ2), and the time-window when it is performed (RQ1) be the most optimal solution for the model update? Should the model update be custom-made for each scenario to improve resource efficiency**?**
Using the data from RQ1 and RQ2, in RQ3, we will combine the results from the previous research questions and attempt to find a trade-off between the two.

As no statistical relationship has been observed regarding model updates done either date-wise or data-amount wise, and the fact that the specific model is always locked when the update is triggered due to the programming language it has been implemented in, no trade-off scenario applies to this case.

DISCUSSION AND OBSERVATIONS

While conducting the simulations, several important notes have been observed and communicated to the developing team.
1. The database queries used in the validator fail to return the same amount of data for the same query at different point in time. This might be due to other stakeholders updating relevant entries in the datastore at the same time, and thus locking the data entries. The observed marge of difference has been fluctuating up to 10000 entries over the course of a week when queries are done real time.
2. The database query API returns different number of columns per record when queried date-wise which could affect the results of RQ1.2 significantly due to inadequacies in data points. This issue does not appear when it comes to the results analyzed in RQ1.1 since the data used in RQ1.1 has been cleaned and columns amount made even before validations have been run.
3. Several software bugs have been observed in the validator when it comes to querying the database date-wise, out of which several have been patched and solved. Worth mentioning is the case in which the validator was found finishing execution when no data points could be observed over a timespan of several days, and the case in which the validator could not find all the fault-types in the test-data, thus failing to conclude the attempted update.

These observations have the capability of nullifying the results analyzed at RQ1.2.

4. In order to be able to isolate the problem to data-source only bug, we suggest that more extensive testing be done in order to avoid bugs like the above-mentioned one in the validator before the project goes from development to production.

5. On the other hand, the categorizer project has high popularity within the company with a high user-trust in its results.

CONCLUSION

Considering the results and observations discussed above, we can conclude that the current method of updating the prediction algorithm is the safest and most optimal, and no other more optimal approach can be applied with the current development tools of the team.

REFERENCES

[1] Marshall Cline. "C++ FAQ: "What's this "serialization" thing all about?"". Archived from the original on 2015-04-05.

[2] A course in machine learning, Hal Daume III, 2017

[3] Multi-label classification, Grigorios Tsoumakas & Ioannis Katakis

[4] Selection of relevant features and examples in machine A.L. Blum, P Langley/Artificial Intelligence 97 (1997) 245-271

[5] A Survey on Feature Selection Jianyu Miaoa,c, Lingfeng Niub,c,∗

[6] The problem of concept drift: definitions and related work , Alexey Tsymbal April 29, 2004

[7] https://blog.bigml.com/2018/02/06/retraining-machine-learning-models/

[8] A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering, Andrea Arcuri and Lionel Briand

[9] The Kruskal-Wallis Test and Stochastic Homogeneity , Andras Vargha & Harold D. Delaney

[10] W. LaMorte, W. (2017). *Mann Whitney U Test (Wilcoxon Rank Sum Test)*. [online] Sphweb.bumc.bu.edu. Available at: http://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_nonparametric/BS704_Nonparametric4.html [Accessed 13 May 2018].

[11] Fay, Michael P.; Proschan, Michael A. (2010). "Wilcoxon–Mann–Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules". Statistics Surveys. 4: 1–39. doi:10.1214/09-SS051. MR 2595125. PMC 2857732 Freely accessible. PMID 20414472.

[12] Ocw.umb.edu. (n.d.). Critical Values of the Mann-Whitney U test Table. [online] Available at: http://ocw.umb.edu/psychology/psych-270/other-materials/RelativeResourceManager.pdf [Accessed 13 May 2018].

[13] Docs.python.org. (2018). *17.2. multiprocessing — Process-based parallelism — Python 3.6.5 documentation*. [online] Available at: https://docs.python.org/3/library/multiprocessing.html [Accessed 16 May 2018].

[14] Docs.python.org. (2018). *17.1. threading — Thread-based parallelism — Python 3.6.5 documentation*. [online] Available at: https://docs.python.org/3/library/threading.html [Accessed 16 May 2018].

[15] Wiki.python.org. (2018). *GlobalInterpreterLock - Python Wiki*. [online] Available at: https://wiki.python.org/moin/GlobalInterpreterLock [Accessed 16 May 2018].

[16] Learning from Time-Changing Data with Adaptive Windowing(2006), Albert Bifet, Ricard Galvada

[17] Daniel, Wayne W. (1990). "Kruskal–Wallis one-way analysis of variance by ranks". Applied Nonparametric Statistics (2nd ed.). Boston: PWS-Kent. pp. 226–234. ISBN 0-534-91976-6.

[18] A Sliding Window Solution for the On-line Implementation of the Levenberg-Marquardt Algorithm ,Fernando Morgado Dias, Ana Antunes, José Vieira, Alexandre Mota

[19] No Free Lunch For Early Stopping, Zehra Cataltepe, Yaser S. Abu-Mostafa, Malik Magdon-Ismail

[20] The Seven Steps to Model Management, https://www.knime.com/blog/the-seven-steps-to-model-management

[21] Real-statistics.com. (2018). *Mann-Whitney Table | Real Statistics Using Excel*. [online] Available at: http://www.real-statistics.com/statistics-tables/mann-whitney-table/ [Accessed 22 May 2018].

[22] Sprinthall, R. C. (2011). Basic Statistical Analysis (9th ed.). Pearson Education. ISBN 978-0-205-05217-2.

APPENDICES:

1. **Results from Simulations RQ1.1**

Training Data Batch Size/ Step Length	1000	2000	3000
5000	Step number: 0, Validation score: 0.957	Step number: 0, Validation score: 0.957	Step number: 0, Validation score: 0.957
	Step number: 1, Validation score: 0.945	Step number: 1, Validation score: 0.856	Step number: 1, Validation score: 0.363
	Step number: 2, Validation score: 0.856	Step number: 2, Validation score: 0.618	Step number: 2, Validation score: 0.808
	Step number: 3, Validation score: 0.363	Step number: 3, Validation score: 0.808	Step number: 3, Validation score: 0.258
	Step number: 4, Validation score: 0.618	Step number: 4, Validation score: 0.585	Step number: 4, Validation score: 0.974
	Step number: 5, Validation score: 0.408	Step number: 5, Validation score: 0.065	Step number: 5, Validation score: 0.896
	Step number: 6, Validation score: 0.808	Step number: 6, Validation score: 0.974	
	Step number: 7, Validation score: 0.773	Step number: 7, Validation score: 0.789	
	Step number: 8, Validation score: 0.585	Step number: 8, Validation score: 0.772	
	Step number: 9, Validation score: 0.258	Step number: 9, Validation score: 0.678	
	Step number: 10, Validation score: 0.065		
	Step number: 11, Validation score: 0.348		
	Step number: 12, Validation score: 0.974		
	Step number: 13, Validation score: 0.965		
	Step number: 14, Validation score: 0.789		
	Step number: 15, Validation score: 0.896		
	Step number: 16, Validation score: 0.772		
	Step number: 17, Validation score: 0.687		
	Step number: 18, Validation score: 0.678		
	Step number: 19, Validation score: 0.47		
10000	Step number: 0, Validation score: 0.902	Step number: 0, Validation score: 0.902	Step number: 0, Validation score: 0.902
	Step number: 1, Validation score: 0.884	Step number: 1, Validation score: 0.788	Step number: 1, Validation score: 0.737
	Step number: 2, Validation score: 0.788	Step number: 2, Validation score: 0.258	Step number: 2, Validation score: 0.301
	Step number: 3, Validation score: 0.737	Step number: 3, Validation score: 0.301	Step number: 3, Validation score: 0.644
	Step number: 4, Validation score: 0.258	Step number: 4, Validation score: 0.919	Step number: 4, Validation score: 0.732
	Step number: 5, Validation score: 0.065	Step number: 5, Validation score: 0.811	
	Step number: 6, Validation score: 0.301	Step number: 6, Validation score: 0.732	
	Step number: 7, Validation score: 0.418		

	Step number: 8, Validation score: 0.919 Step number: 9, Validation score: 0.644 Step number: 10, Validation score: 0.811 Step number: 11, Validation score: 0.719 Step number: 12, Validation score: 0.732 Step number: 13, Validation score: 0.681 Step number: 14, Validation score: 0.586		
15000	Step number: 0, Validation score: 0.36 Step number: 1, Validation score: 0.367 Step number: 2, Validation score: 0.403 Step number: 3, Validation score: 0.874 Step number: 4, Validation score: 0.639 Step number: 5, Validation score: 0.735 Step number: 6, Validation score: 0.604 Step number: 7, Validation score: 0.774 Step number: 8, Validation score: 0.802 Step number: 9, Validation score: 0.623	Step number: 0, Validation score: 0.36 Step number: 1, Validation score: 0.403 Step number: 2, Validation score: 0.639 Step number: 3, Validation score: 0.604 Step number: 4, Validation score: 0.802	Step number: 0, Validation score: 0.36 Step number: 1, Validation score: 0.874 Step number: 2, Validation score: 0.604

2. Validation scores per sample group

Group	Validation Score for Model Refit/Simulations
(5000, 1000)	[0.957, 0.945, 0.856, 0.363, 0.618, 0.408, 0.808, 0.773, 0.585,0.258, 0.065, 0.348, 0.974, 0.965, 0.789, 0.896, 0.772, 0.687, 0.678, 0.47]
(5000, 2000)	[0.957, 0.7955, 0.5195, 0.7905, 0.4215, 0.0895, 0.9675, 0.767, 0.7295, 0.6315]
(5000, 3000)	[0.957, 0.45733333333333333, 0.722,0.14566666666666667,0.91033333333333333,0.8166666666666667]
(10000, 1000)	[0.902, 0.884, 0.788, 0.737, 0.258, 0.065, 0.301, 0.418, 0.919, 0.644, 0.811, 0.719, 0.732, 0.681, 0.586]
(10000, 2000)	[0.893, 0.762, 0.1615, 0.292, 0.799, 0.706, 0.742]
(10000, 3000)	[0.8676666666666667,0.369, 0.42833333333333334,0.6376666666666667,0.69733333333333334]
(15000, 1000)	[0.36, 0.367, 0.403, 0.874, 0.639,0.735, 0.604, 0.774, 0.802, 0.623]
(15000, 2000)	[0.337, 0.6275, 0.677, 0.6505, 0.7015]
(15000, 3000)	[0.32033333333333336, 0.75133333333333333,0.66]

3. Mann-Whitney U results

Test	Samples used for the test	Step lengths of sample data	Critical U from table	U-Statistics from test	P-Value
T1	sample_step_length1000_batch_size5000 against sample_step_length2000_batch_size5000	20 and 10	55	97.5	0.4649391098256198
T2	sample_step_length1000_batch_size5000 against sample_step_length1000_batch_size10000	20 and 15	90	136.0	0.3263324935480303
T3	sample_step_length1000_batch_size5000 against sample_step_length2000_batch_size10000	20 and 7	34	61.0	0.3190736222938987
T4	sample_step_length1000_batch_size5000 against sample_step_length3000_batch_size10000	20 and 5	20	42.0	0.305192256937339
T5	sample_step_length1000_batch_size5000 against sample_step_length3000_batch_size5000	20 and 6	27	56.5	0.4275538222841562
T6	sample_step_length3000_batch_size5000 against sample_step_length2000_batch_size5000	6 and 10	11	28.5	0.45678166516889307
T7	sample_step_length1000_batch_size5000 against sample_step_length1000_batch_size15000	20 and 10	55	84.0	0.24764822023820338
T8	sample_step_length1000_batch_size5000 against sample_step_length3000_batch_size15000	20 and 5	20	36	0.17953348933015717
T9	sample_step_length1000_batch_size5000 against sample_step_length2000_batch_size15000	20 and 3	8	20	0.19290871783341984
T10	sample_step_length2000_batch_size5000 against sample_step_length1000_batch_size15000	10 and 10	23	40	0.23633779675579358
T11	sample_step_length2000_batch_size5000 against sample_step_length2000_batch_size15000	10 and 5	8	16.0	0.14893008377490308
T12	sample_step_length2000_batch_size5000 against sample_step_length3000_batch_size15000	10 and 3	3	10	0.2234364103554154
T13	sample_step_length3000_batch_size5000 against sample_step_length3000_batch_size15000	6 and 3	1	6.0	0.2593025082143628
T14	sample_step_length3000_batch_size5000 against sample_step_length1000_batch_size15000	6 and 10	11	22	0.20796881311651105
T15	sample_step_length3000_batch_size5000 against sample_step_length1000_batch_size10000	6 and 15	19	30	0.25407402747406904
T16	sample_step_length3000_batch_size5000 against sample_step_length2000_batch_size15000	6 and 5	3	9.0	0.15765122604087278

T17	sample_step_length3000_batch_size5000 against sample_step_length3000_batch_size10000	6 and 5	3	10	0.20565689588812947
T18	sample_step_length3000_batch_size5000 against sample_step_length2000_batch_size10000	6 and 7	6	17	0.3085375387259869
T19	sample_step_length1000_batch_size10000 against sample_step_length2000_batch_size10000	15 and 7	24	51	0.47190153623465564
T20	sample_step_length1000_batch_size10000 against sample_step_length3000_batch_size10000	15 and 5	14	31	0.3002356300616926
T21	sample_step_length2000_batch_size10000 against sample_step_length3000_batch_size10000	7 and 5	5	14	0.3130587399583693
T22	sample_step_length1000_batch_size15000 against sample_step_length2000_batch_size15000	10 and 5	8	23	0.42711980288404666
T23	sample_step_length1000_batch_size15000 against sample_step_length3000_batch_size15000	10 and 3	3	13	0.39992305283123664
T24	sample_step_length2000_batch_size15000 against sample_step_length3000_batch_size15000	5 and 3	0	7.0	0.5
T25	sample_step_length2000_batch_size5000 against sample_step_length1000_batch_size10000	10 and 15	39	66	0.31864358516179914
T26	sample_step_length2000_batch_size5000 against sample_step_length2000_batch_size10000	10 and 7	14	32	0.4036250839660036
T27	Sample_step_length2000_batch_size5000 against sample_step_length3000_batch_size10000	10 and 5	8	19	0.2502797489090697
T28	sample_step_length1000_batch_size10000 against sample_step_length1000_batch_size15000	15 and 10	39	66.0	0.31864358516179914
T29	sample_step_length1000_batch_size10000 against sample_step_length2000_batch_size15000	15 and 5	14	27.0	0.19136654444261297
T30	sample_step_length1000_batch_size10000 against sample_step_length3000_batch_size15000	15 and 3	5	19	0.3611414813580961
T31	sample_step_length2000_batch_size10000 against sample_step_length1000_batch_size15000	7 and 10	14	32	0.4036250839660036
T32	sample_step_length2000_batch_size10000 against sample_step_length2000_batch_size15000	7 and 5	5	10	0.1278115537732063
T33	sample_step_length2000_batch_size10000 against sample_step_length3000_batch_size15000	7 and 3	1	8	0.324251689826488
T34	sample_step_length3000_batch_size10000 against sample_step_length1000_batch_size15000	5 and 10	8	25	0.4755851346484678

T35	sample_step_length3000_batch_size10000 against sample_step_length2000_batch_size15000	5 and 5	2	12	0.5
T36	sample_step_length3000_batch_size10000 against sample_step_length3000_batch_size15000	5 and 3	0	7	0.5

4. Validation scores per sample group for date-wise run simulations[1]

Days of training data / Days of validation data	1-day	1-week	1-month
4 months	[0,358779, 0.498567, 0.212121, 0.517544, 0.469484, 0.596429, 0.343023, 0.555249, 0.127726, 0.583732, 0.198068, 0.215094, 0.285333, 0.185764, 0.250464, 0.370813, 0.293976, 0.514874, 0.290466, 0.234783, 0.317778]	[0.524068, 0.392324, 0.41385]	[0.51959]
3 months	[0.236601, 0.174971, 0.220708, 0.181575, 0.15736, 0.164311, 0.528302, 0.439614, 0.24898, 0.259398, 0.581206, 0.556769, N/A, N/A, 0.3664, 0.262048, 0.23756, 0.60733, 0.587332, 0.342541, 0.5427, 0.391221, 0.274964, 0.478146, 0.474277, 0.269841, 0.271768, 0.368421, 0.450581, 0.2125, 0.291878, 0.402299, 0.400612, 0.176471, 0.199029, 0.346715, 0.338192, 0.11202, 0.0644172, 0.752336, 0.145631, 0.168627, 0.459384, 0.326241, 0.506494, 0.309002, 0.340686, 0.231121, 0.150224, 0.518931, 0.579812]	[0.488116, 0.33376, 0.258386, 0.339385, 0.51087, 0.138421, 0.248014]	[0.278428]
2 months	[2]Results could not be retrieved	[0.498594 ¸ 0.370629, N/A, 0.855362, 0.475115, 0.525646, 0.45068, 0.662122, 0.430505, 0.680892¸ 0.519067, 0.5055, 0.494492, 0.608241, 0.57017, 0.680685, 0.426067]	[0.348034, 0.303679, 0.478957, 0.260702]

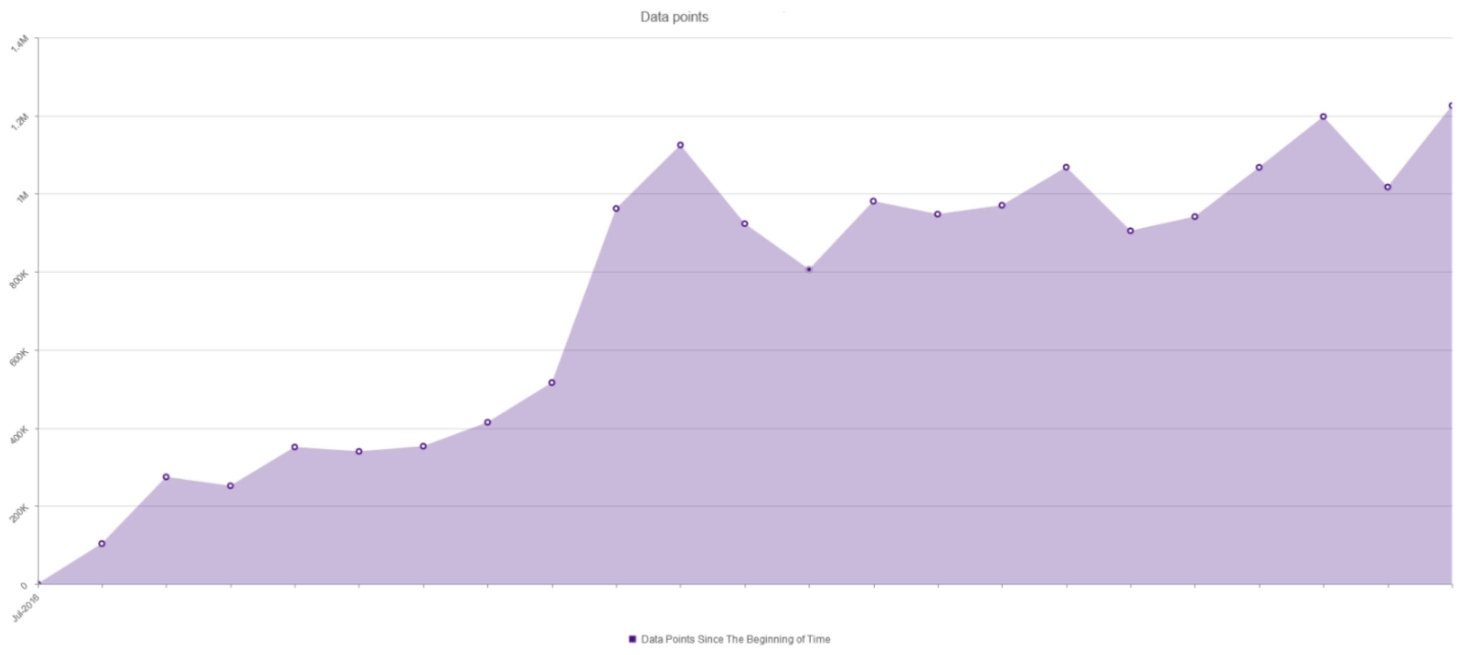
[1] This table has a slight difference in output than the previous one due to space

[2] Results could not be retrieved for the given training data period due to the fact that not all the product faults (product, environment, test) could be found in the training data. Having this parameter is a mandatory condition for the training algorithm to be able to produce results.

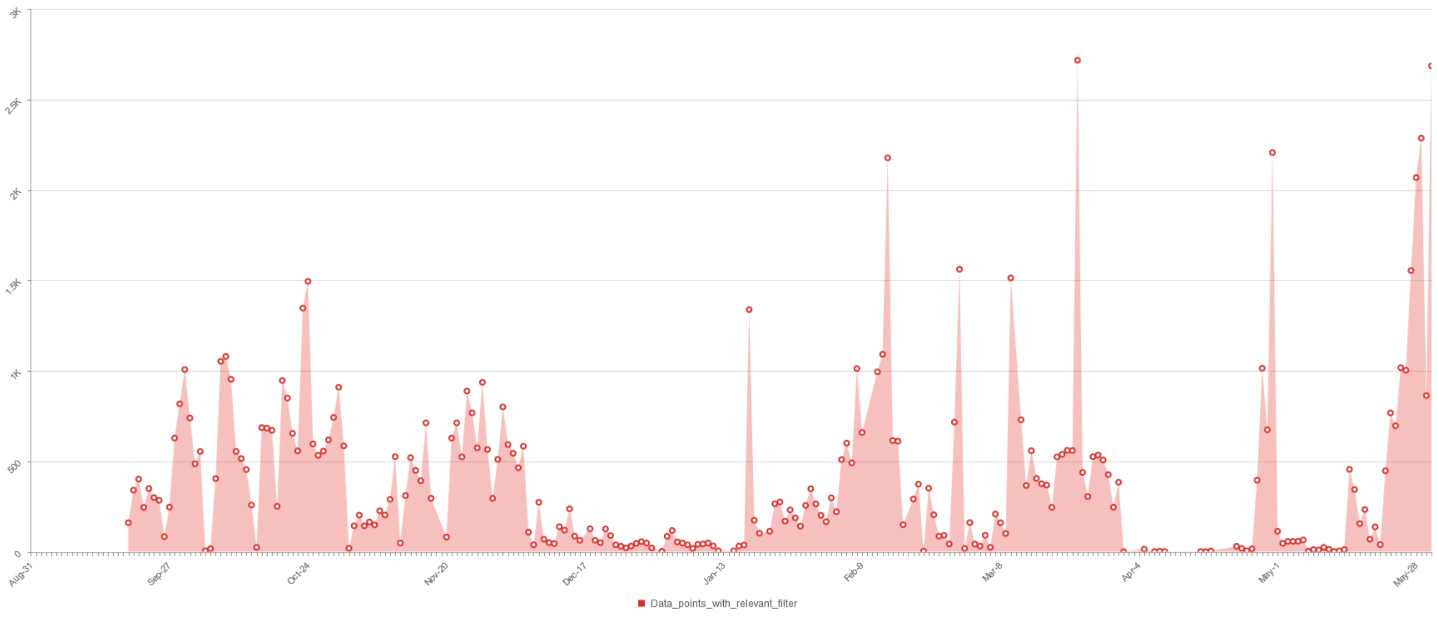
5. Mann-Whitney U results

Test	Samples used for the test	Step lengths of sample data	Critical U from table	U-Statistics from test	P-Value
T1	sample_step_length1 day batch_size3months against sample_step_lenght7 days_batch_size3months	49 and 7	-	178	0.8844
T2	sample_step_length1 day_batch_size3months against sample_step_length1month_batch_size3months	49 and 1	-	27	0.92
T3	sample_step_length7days_batch_size3months against sample_step_length1month_batch_size3months	7 and 1	-	4	1
T4	step1day_batch120 and step30days_batch120	21 and 1	-	20	0.1818
T5	step1day_batch120 and step7days_batch120	21 and 3	-	17	0.2342
T6	step7days_batch120 and step30days_batch120	3 and 1	-	3	0.5
T7	step7days_batch60 and step30days_batch60	10 and 3	31	30	0.006993
T8	step1day_batch90 and step1day_batch120	49 And 21	-	482	0.684
T9	step7days_batch90 and step7days_batch120	7 And 3	11	4	0.1833
T10	step7days_batch90 and step30days_batch120	7 and 1	-	6	0.5
T11	step7days_batch90 and step7days_batch60	7 and 10	14	10	0.01357
T12	step7days_batch90 and step30days_batch60	7 and 3	16	14	0.5167
T13	step1day_batch90 and step7days_batch120	49 And 3	-	40	0.2065
T14	step1day_batch90 and step30days_batch120	49 and 1	-	40	0.4
T15	step1day_batch90 and step7days_batch60	49 and 10	-	73	0.0002302
T16	step1day_batch90 and step30days_batch60	49 and 3	-	87	0.6285

T17	step30days_batch90 and step1day_batch120	1 And 21	-	7	0.7273
T18	step30days_batch90 and step7days_batch120	1 And 3	-	0	0.5
T19	step30days_batch90 and step30days_batch120	1 And 1	-	1	1
T20	step1day_batch120 and step7days_batch60	21 and 10	-	34	0.001881
T21	step1day_batch120 and step30days_batch60	21 and 3	-	38	0.6196
T22	step7days_batch120 and step7days_batch60	3 and 10	13	6	0.1608
T23	step7days_batch120 and step30days_batch60	3 and 3	-	9	0.1
T24	step30days_batch120 and step7days_batch60	1 and 10	-	0	0.1818
T25	step30days_batch120 and step30days_batch60	1 and 3	-	0	0.5



6. Distribution of relevant data points



7. **Distribution of the data relevant to this thesis**