Motivation Research Using Labeling Functions

Idan Amit
idan.amit@mail.huji.ac.il
The Hebrew University
Jerusalem, Israel

Dror G. Feitelson
feit@cs.huji.ac.il
The Hebrew University
Jerusalem, Israel

ABSTRACT

Motivation is an important factor in software development. However, it is a subjective concept that is hard to quantify and study empirically. In order to use the wealth of data available about real software development projects in GitHub, we represent the motivation of developers using labeling functions. These are validated heuristics that need only be better than a guess, computable on a dataset. We define four labeling functions for motivation based on behavioral cues like working in diverse hours of the day. We validated the functions by agreement with respect to a developers survey, per person behavior, and temporal changes. We then apply them to 150 thousand developers working on GitHub projects. Using the identification of motivated developers, we measure developer performance gaps. We show that motivated developers have up to 70% longer activity period, produce up to 300% more commits, and invest up to 44% more time per commit.

CCS CONCEPTS

• **Applied computing → Psychology**; • **Computing methodologies → Machine learning**.

KEYWORDS

methodology, weak supervision, software engineering, motivation

ACM Reference Format:
Idan Amit and Dror G. Feitelson. 2024. Motivation Research Using Labeling Functions. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3661167.3661224

1 INTRODUCTION

Human aspects of software engineering are usually studied using tools like experiments, case studies, interviews, and surveys. These can be costly in money and effort to apply, limiting the data to only a small number of samples. Machine learning on large datasets can complement the research done using such methods and leverage the data available in open-source code repositories. However, to investigate a concept using such datasets one needs to identify it. This is difficult to do when the concept is abstract and subjective, such as motivation.

Machine learning copes with this problem by using labeled samples instead of a precise definition. A model that can predict the labels of unseen samples demonstrates that it has captured the concept that they represent. In supervised learning, one builds the model using labeled train samples. But we do not have enough labeled samples to build a robust model.

The solution we propose is to use labeling functions as models. Labeling functions are heuristics for generating labels, correlated with motivation, and validated to predict it better than a guess [50]. In our case the labeling functions use developers' behaviors to predict whether they are motivated. Even a single labeling function is helpful, but it might be biased. By using multiple labeling functions that capture diverse perspectives of motivation, we increase our confidence in results that reproduce for all of them.

Our labeling functions include refactoring (investing in design improvement reflects motivation) and working diverse hours (a developer that occasionally works in late hours is assumed to be motivated). Using them we quantify the behavior of developers with and without motivation, reproduce prior work regarding its benefits [13, 33, 38], and predict future developer retention.

Our main contributions are the following:

- We provide a new methodology, complementing surveys and experiments, to investigate motivation.
- The method enables large scale, long term, quantitative, and reproducible investigation of motivation in actual behavior in a natural setting.
- We provide and validate four labeling functions for motivation, two new and two from prior work.
- We show that our labeling functions can be used to predict churn in advance, allowing intervention.
- We show that motivation is correlated with more activity, more output, and investing more time in each task.

2 THE VISION: LABELING FUNCTIONS AS A RESEARCH FRAMEWORK

The current toolbox of motivation researchers contains experiments [13], interviews, surveys [44], and case studies [52]. While these methods allow control, the cost of each sample is high, and therefore there is a validity threat due to the small datasets [58]. This small dataset problem can be solved by mining software repositories, leveraging the millions of activities performed by many thousands of developers.

However, to apply supervised learning and investigate a concept (such as motivation), we need to label the data and distinguish cases where the concept applies from those where it does not. Usually, one can obtain labels for the concept using manual human work (e.g., asking all the developers about their motivation). Manual labeling is limited in capacity, preventing leveraging the power of big data.

EASE 2024, June 18–21, 2024, Salerno, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1701-7/24/06
https://doi.org/10.1145/3661167.3661224

Labeling functions, providing predictions that need be only slightly better than a guess [50], provide an alternative to manually labeling the concept. The use of multiple labeling functions allows us to raise our confidence in the labeling when all or most of them agree. Given a good enough representation of the concept, one can investigate it at scale. This can provide supporting evidence for relations of interest (e.g., activity period is longer given all motivation functions).

Labeling functions are especially beneficial in concepts that are not well defined. **The essence of the difficulty** is motivation being an internal, subjective, hard to measure concept. Literature surveys — including one spanning 75 years of motivation research — found that there is no canonical method to measure motivation, and the suggested methods have limited agreement [44, 54].

Hence, we do not aim to provide a new definition of motivation or measure our work with respect to an existing one. Instead, we evaluate our functions with respect to answers about motivation, specifically self-reporting on motivation and working hours [16].

2.1 Labeling Functions

A labeling function [19] is a validated computable weak classifier, a heuristic that one can apply computationally to a dataset and get predictions that are better than a guess.

The concept of weak learnability [37], learning slightly better than a guess, was suggested as a way to relax the high requirements of PAC learning [56]. Surprisingly, it was shown that the concepts are equivalent, and one can boost weak learners to regular PAC learners [50]. Since then boosting became an important learning method [29, 50]. Weak classifiers were also found powerful in coping with the lack of labeled data. They were part of both theoretical and practical work like co-training [19] and weakly-supervised learning [9, 11, 23].

Labeling functions can be either learned from a dataset or just a fixed rule. An example of such a rule is "people that participate in popular projects tend to be motivated by recognition". This rule is not perfectly accurate, yet it encapsulates knowledge which improves our prediction. Given a single labeling function, one can use it as a proxy for the concept. For example, the labeling function of retention in a project can be used to investigate the concept of motivation. Of course, the same investigation can also be framed as the investigation of retention as an object of interest on its own.

Behavioral cues [25] and even labeling functions have been used previously in motivation research, though not formally. For example, coming to work in a snowstorm is predictive of high satisfaction [52]. In open-source the projects' license openness level is predictive of the developer ideology-based motivation [18].

Our goal in this paper is not to reach the best predictive power, but to find relations between motivation and performance. For this it is beneficial to have different and diverse functions capturing different aspects of motivation. An increase of a behavior of interest, given an increase in a motivation labeling function, is a hint of the relation between them. If the result reproduces for several different motivation labeling functions, it increases the likelihood that motivation indeed increases this behavior.

3 MOTIVATION LABELING FUNCTIONS AND THE INTUITION BEHIND THEM

We looked for motivation labeling functions that represent activities which are not mandatory and require some cost. We looked for actions for which usually there is no external enforcement. For example, writing tests requires extra investment, but in many projects they are enforced by cultural norms or technological means. Therefore, tests may not be a good labeling function for motivation. Yet, the length of the commit message documenting development can be a good labeling function, since colleagues and managers usually are not aware of the message length and therefore do not enforce it.

We also wanted the functions to fit open-source development, where many of the participants volunteer as a part-time hobby. Thus, counting working days could be a labeling function for full time employees, but it might underestimate the motivation of a volunteer working on weekends.

We deliberately choose simple, one-variable binary functions. We need a binary function to distinguish motivated from unmotivated developers. Hence, we need to choose a cut-off value for the continuous labeling functions. For simplicity, we uniformly use the mean of each metric. In Section 6.1 we investigate the benefit of using the raw metrics without a cut-off and show that there is no major difference.

Last, we wanted the functions to be diverse and capture motivation in different ways. By doing so we increase the robustness of estimation and validity of relations. The labeling functions we selected are retention in a project, working diverse hours, performing refactoring, and writing detailed commit messages. The first two are based on prior work and the last two are new.

3.1 Retention

Out of a project's developers in a given year, we define the retained ones as those that continue in the next year. Given the prior work supporting the relation between motivation and retention [12, 42, 49], we label retained developers to be motivated in the current year and developers who did not continue as not motivated.

Argyle found modest correlation between job satisfaction and retention, tending to be stronger among white collar workers. [12]

Note, though, that ending activity in a project due to reasons not related to motivation poses a threat. Reasons to take a break are personal in 78.3% of the cases, of which 36.2% are due to a life event or a financial issue, unrelated to the developer's desires or the project [20]. To reduce external influence, we examine only active projects, hence the reason for leaving the project is not project termination.

Another problem is that retention is a binary function, so it lacks the fidelity of a continuous function. For a developer working for years in a project we can only claim a similar level of motivation in the first years, and lower motivation in the last year. Continuous labeling functions do not have this limitation and allow us to better quantify the motivation level. Another limitation of retention as a labeling function is that we can know it only in retrospect. In research we can indeed use past data for investigation, but this labeling function cannot help estimating the current motivation.

Other than the developer, retention is influenced by the project characteristics. The retention in large projects is 65% of the retention in medium ones and only 18% of that in single person projects. Retention in new projects is 25% higher than in old ones. The retention in projects belonging to companies is 79% that in projects owned by other organizations. Retention in extraordinarily popular projects is 21% of that in projects of low popularity. All these effects are not surprising, as they are associated with feelings of community and ownership, known motivators — and thus retention [14, 38, 45].

Nevertheless, such influences of the project characteristics might lead to systematic biases of the labeling function. To cope with this, we first identify these biases. We use control variables to verify that a behavior is not due to the control. We verify that the function is predictive despite the biases, and we use several functions to have diversity and reduce the impact of each specific bias.

3.2 Refactoring

While quitting a project can be a strong indication of lack of motivation, it is a binary indicator that usually happens only once. We wanted the other functions to allow continuous monitoring of motivation and its levels. Therefore, we base them on recurring activities and measure the level of activity.

Refactoring is improvement of the design of code while keeping the same functionality [26]. Since refactoring does not add functionality, its value is not always seen from an external point of view (e.g., of the developers' managers or customers). Investing in refactoring therefore reflects motivation on the part of the developer. As far as we know, we are the first to use refactoring as an indication of motivation. However, it is known that improvement activities in other contexts are hard and require motivation [39].

We identify refactoring activity using a linguistic-analysis-based classifier applied to the commit messages documenting the change done [6, 21]. In order not to be dependent on the number of commits, we use the refactoring probability, namely the ratio of refactoring commits to total commits [4].

In some of the use cases we need a binary function instead of a continuous one. We turned all our activity-based labeling functions into binary ones by comparing each developer's activity to the mean activity in our survey dataset (See Section 5.1). For refactoring this cut-off is at 20% refactoring probability, which is the 69^{th} percentile of cases in GitHub. This threshold is chosen for its simplicity, and we show in Sections 5.2 and 6.1 that the influence of the specific threshold is limited.

Refactoring is also influenced by the project characteristics. The refactoring probability in old projects is 37% higher than in new ones. In large projects (with many developers), the refactoring probability is 13% higher than in a single person project, and 37% higher than in small ones. In extraordinarily popular projects, the refactoring rate is 60% higher than in low popularity projects. We use control variables to avoid false impact attribution.

3.3 Diverse Working Hours

A motivated worker might start working earlier or stay later when needed. The correlation between motivation and overtime [16] also supports this metric.

Using the commits' timestamp, we identified each developer's working hours in a whole calendar year. We used the number of distinct hours of the day in which commits were performed as our metric. Hence the maximal value is 24 hours, and a single sleepless working day should be enough to reach it. On the other hand, a person working 9 to 5 will have the value of 8. We did not use the sum of working hours since an unmotivated full-time employee will probably still work more hours than a week-end motivated hobbyist. The cut-off value for binarization (the mean) is working at least 18 hours, reached at the 71^{st} percentile.

The longer the period a person contributes to a project, the more likely the person is to contribute in diverse hours, regardless of motivation. This is supported by the Pearson correlation between activity days and distinct hours, which is 0.59. The threat due to this correlation increases since activity days also have 0.19 correlation with retention. However, diverse hours have a higher 0.26 correlation with retention, so at least part of it is not due to activity days.

Moreover, the activity in a specific hour of the day, per developer, year, and project, is due to a single commit in 43% of the hours, and at most two in 59%. Hence, the contribution in these hours is very sensitive and a single commit might change the number of distinct hours. Indeed, a single sleepless night is enough to reach 24 distinct hours. However, deciding to devote your sleepless night to programming might be a strong indicator of motivation.

3.4 Long Commit Messages

Commit messages are used to document the change done when committing code. The content of the message might contain the change, the reason to perform it, administrative details, etc. [21]. As far as we know, we are the first to use message length as an indicator for motivation. Yet, documentation is considered to be a tedious task and requires motivation [51]. We use the average length of the messages as indication to the motivation and investment in writing them. The cut-off value for high average message length is above 84 characters, reached at the 59^{th} percentile.

Note that messages can be very long. The 99^{th} percentile is 1,204 characters and there are also messages of millions of characters. These are probably the result of mechanisms that automatically generate very long messages. In "Squash commits" the work in several commits is aggregated into one and their messages are combined. Some tools automatically add the "git diff", the summary of the modifications to the code. 9% of the messages above 10k characters mention squash, compared to only 0.1% in shorter messages; diff is mentioned in 37% compared to 1.2%. In such cases the long message is not an indication of motivation and investing effort in writing it. One could take it further and claim this is an indication of lack of motivation to remove a "git log polluting" too long message.

Also, the average message length in single-developer projects is 47 characters, compared to the almost 3 times more (140 characters) in others. This is probably since single developers write messages for 'future me' while in larger projects the community needs and enforces this documentation. In twins experiment, comparing the same developer in a single developer project and larger ones, the average length is shorter in the single developer project in 59% of the cases.

4 METHODOLOGY

4.1 Analysis

The goal of our methodology is to validate that we represent motivation well and capture the relations between motivation and developer performance. We start by using 4 labeling functions, to reduce the influence of a specific function's artifacts. As ground truth we use developers self-reporting on two motivation questions. We verify that the functions weakly predict them. Once being weak classifiers is established, we validate the functions on a large scale, with respect to each other, using the following methods:

- We measure monotonicity with respect to retention.
- We validate agreement at the developer level using twin experiments, showing that a higher value in one function tends to agree with higher values in the others.
- We validate temporal agreement, that an improvement in one function is predictive of improvements in the others, as expected when measuring improvement in the same concept.
- We use control variables to verify that the relations are not due to co-founding, both with a single control and with all in a supervised model.

Once we validate the representation of motivation, we investigate its relation to developer performance. We measure activity in project, output, and process motivation [55], each with two metrics. We evaluate the relation between each of the metrics and the functions. Other than predictive analysis, we used co-change and twin analysis and the control variables. The redundancy in the representation, and investigation in the population level, developer level, and temporal level, reduce the threat of misidentification.

We use the following control variables. Age groups, divided into projects before GitHub creation (before 2008), the oldest 25% (before 2014), the youngest 25% (since 2017), and the middle 50% [7]. Developers were grouped into 'Single', 'Small' (at most 10), 'Medium' (at most 100) and 'Large'. Popularity groups were divided into 'Low' (lowest 25%, at most 8 stars), 'Medium' (next 50%, at most 422 stars), 'High' (next 20%, at most 5027) and 'Extraordinary' for the top 5%. Projects belonging to a company were identified by manually labeling the 100 users with most projects. For programming languages, we control for: Python, JavaScript, Java, C++, PHP, and 'other'.

4.2 Motivation Survey Dataset

As a first validation of the labeling functions, we wanted to compare them to answers regarding motivation. We used a survey by Amit and Feitelson asking various questions regarding motivation [8]. To match the answers with the actual behavior the survey also asked for the GitHub profile and projects.

The survey included 66 questions about motivation and software development, covering 11 motivators as learning, recognition, etc. Our goal here is to establish first the ability to measure motivation and its influence in general. Therefore, we use only a few relevant questions here and leave labeling functions for motivators (e.g., people in popular projects report high recognition motivation) to future work.

The questions used in the labeling functions validation are:

- I regularly have a high level of motivation to contribute to the repository (based on [46])
- How many hours a week do you work on the repository (average)?
- I'm being paid for my work in this repository

The survey was conducted from December 2019 to March 2021. It obtained 1,724 responses, 521 of them finished the survey. The participants provided the names of 484 projects and 303 personal GitHub profiles.

After a year, a follow-up survey was sent to the participants that provided their emails in the original survey. In the follow up survey, 124 out of the 341 participants answered (36.3%).

4.3 GitHub Dataset

GitHub is a platform for source control and code development projects, used by millions of users. Our dataset is based on the BigQuery GitHub schema, which includes the commit history of select projects. We start with all projects with 50 or more commits during 2021. We excluded forks, redundant projects, and non-software projects [7], ending with 18,958 projects.

Many of the developers contributing to a project make only occasional, sporadic contributions, sometimes a single commit. For example, they may fix a bug found while working with the project or add a small functionality for self-use. These developers do not represent well the typical motivations of involved developers. Our focus is on the developers who make significant contributions to the project and are in some way invested in it. We choose to use the threshold of 12 commits per year, an average of one commit per month, as a lower bar for involvement [7]. While this omits 62% of the developers, they are responsible for only 6% of the commits.

To reduce the threat of bots [31], we also filtered out developers with 1,000+ commits per year, 0.04% of the developers. Note that since we started with projects active during 2021 and examined their history, none of the developers stopped working on a project because the project was terminated.

5 VALIDATION OF LABELING FUNCTIONS

5.1 Labeling Functions Validation by the Survey

In this section we validate the labeling functions by comparison to answers in the survey regarding motivation. We asked survey respondents for their GitHub profile, which allows us to match their actual behavior with their answers.

We perform the validation in a supervised learning manner, using a classifier to predict a concept. The 'Concept' column in Table 1 is the question for which we try to predict a high answer. We used both the motivation question and the working hours question. 'Classifier' is the way we predict the concept — by using a high value in either a labeling function, or in the other motivation question (That is, we used the motivation question to predict the working hour question and vice versa). Also, we compared high answers to motivation questions in the original and the follow-up surveys of the same developer contributing to the same project.

It is generally accepted that motivated workers work longer hours [16]. This result may be tainted by mixing data about paid developers with data about volunteers, both common in open-source projects. We checked this by separating the groups using the survey question about payment. For unpaid workers the reported average working hours were 10.8 (high motivation) and 4.5 (low), while for

Table 1: Validation of labeling functions using survey answers

Concept	Classifier	Cases	Accuracy	Accuracy Lift	Precision	Precision Lift
Motivation Answer	Retention	28	0.57	0.19	0.85	0.13
	High Refactoring	28	0.54	0.15	0.83	0.11
	High Hours	28	0.68	0.27	0.88	0.17
	Long Messages	28	0.43	0.04	0.78	0.04
	Working Hours Answer	245	0.65	0.30	0.70	0.35
Motivation Answer Follow-Up	Motivation Answer	46	0.72	0.47	0.63	0.45
Working Hours Answer	Retention	21	0.52	0.05	0.56	0.06
	High Refactoring	21	0.57	0.15	0.60	0.15
	High Hours	21	0.52	0.05	0.55	0.04
	Long Messages	21	0.52	0.07	0.60	0.15
	Motivation Answer	245	0.65	0.30	0.56	0.35
Working Hours Answer Follow-Up	Working Hours Answer	47	0.81	0.53	0.65	1.04

paid workers they were 27.9 (high) and 25.8 (low). Therefore in Table 1 we used only the behavior of unpaid developers to compare to the working hours question. In the rest of the analysis we do not do this filtering and use all developers.

'Cases' is the number of developers whose data was used in each row. When comparing questions, we have nearly 250 cases, and when comparing to the follow-up we have nearly 50 cases. However, when comparing answers to actual behavior the numbers are lower, since this requires a combination that occurs for only a small fraction of the survey respondents: they need to both provide their profile, and we need to have their project in our dataset. Note that we reached these numbers from a relatively big survey completed by 521 people with partial replies from 1,724 (a big drop due to not contributing to GitHub). Hence, it will be hard to enlarge this dataset significantly. Instead, to increase validity, we analyze in the next sections the full GitHub dataset, having years of activity in 18,958 projects by 151,775 developers.

The analysis is in the supervised framework, and we present accuracy, precision, and their lifts. The lift, $\frac{Accuracy - Independent\ Prob}{Independent\ Prob}$ (and likewise for precision with respect to positive rate), measures how much larger the result is relative to the base probability. Note that all labeling functions have positive accuracy lift and precision lift with respect to both questions, thereby validating them.

As expected, comparing with questions leads to higher performance than comparing with labeling functions; In particular, comparing with the same question in the follow-up survey leads to the highest scores. This provides a benchmark with which to compare the results for the labeling functions. While the performance achieved by the labeling functions is lower, it is not too low, and it is sufficient for the requirement from weak classifiers that can be applied at scale.

5.2 Labeling Functions Validation by Monotonicity

Correlation between two binary variables (e.g. high distinct hours and retention) is indicative of agreement, since the probability of accidental agreement decreases with the number of samples and level of agreement. We use retention as a **proxy** for motivation and compare the continuous labeling function to it. Retention has Pearson correlation of 0.26 with hours, 0.006 with message length, and 0.005 with refactoring probability. The first value is medium-low and the last two are almost zero.

Monotonicity [35, 53] adds to correlation by requiring a step by step increase and not only a general increase in both variables together. This is since even when there is some correlation between the variables, the probability of having an increase in every step by mere chance is low.

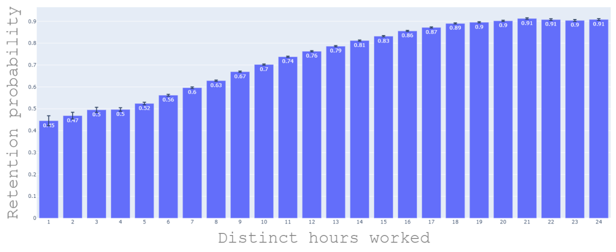

Figure 1: Retention in next year given distinct hours worked in the current one.

Figure 1 presents a "text-book graph" of the monotonicity of working hours and retention probability. The probability of retention also increases with average message length, but only in the lower part and then it is rather flat.

With refactoring there is no positive monotonicity. The differences in retention between deciles are small, sometimes decreasing, and inconsistent.

Hence, we see monotonicity of retention with respect to diverse working hours, partially with message length, but not with refactoring.

5.3 Labeling Functions Validation by Twins Experiments

In Figure 1 we show correlation between working hours and retention, in the population. Twin experiments allow us to investigate the correlation at the individual person level.

When a person is motivated in a project, it might be due to the person, the project, or the interaction between them. The ability to factor out possible influencing variables, by equating them, helps focusing on the other variables. A popular method for that in psychology is "twin experiments" [57]. Identical twins have the same genetic background, so a difference in their behavior is attributed to another variable (e.g., being raised differently). This idea was used in software to profile malware of the same developer [11], and to investigate software quality [5, 7].

We analyze the results of two types of twin experiments, where the twins differ in their retention. In the first one we observe the same developer in the same year in two projects. We choose pairs of projects such that in one the developer continued and in the other the developer left. This setting factors out the developer, yet the projects are different.

In the second type we observe the same developer in the same project in two consecutive years: the developer's last year in the project, and the one before it. Here we factor out the developer and the project, but not completely, because people and projects change over time.

Table 2: Validation of labeling functions by same developer twins experiments, using function agreement with retention being more than 0.5

Twins Type	Devs.	Pairs	Refactor Prob.	Distinct Hours	Msg. Length
Same year, different proj.	7,856	1,314,536	0.95	0.59	0.56
Same proj., consecutive years	42,087	51,549	0.58	0.68	0.49

Table 2 presents the twin experiments results. The columns show the probability that the labeling function is at least as good in the continuing case (indicating retention and implying higher motivation). Taking no influence as the null hypothesis, we expect a probability of 50%. Note that the probability is always higher, other than for message length in consecutive years, which is close to 50% from below. The very high probability of 95% for refactoring in the same year is due to developers not doing refactoring in both cases. When we ignore cases where both are zero the probability is 58%. The probability of equality was small in all other cases.

We also analyzed the twin experiments subject to the control variables. In the 'Same year' case controlling for company, number of developers, popularity, and programming language lead to the same behavior as without controlling. In the 'Consecutive years' case we got the same behavior when controlling for age, number of developers, and popularity. We sometimes got better message length when controlling a company and a programming language. Overall, this shows that the results are rather robust to the controls.

We also compared advantages in one continuous labeling function given advantage in another. In the first analysis we used twins which were the same developer, in the same year, in different projects, regardless of retention. For example, we check the probability of higher distinct hours in project A relative to project B, given a higher refactoring probability in project A relative to project B. In all cases there was a positive lift. Using controls, there was a positive lift in 200 out of 210 cases. In a similar way, we analyzed the same developer, in the same project, in consecutive years regardless of retention. All cases had a positive precision lift too. 118 out of 126 controls had positive lift.

5.4 Labeling Functions Validation by Co-Change

If a person becomes more motivated, we expect an improvement in all our labeling functions. However, we cannot measure the person's motivation directly but only using our labeling functions. If all the labeling functions reflect the person's motivation, an increase in one function is expected to correlate with an increase in the other functions. In co-change analysis [5, 7], we check this expected correlation between the labeling functions, omitting the person's motivation which is hidden from us.

We compare the change of two metrics on the same developer in the same project in two consecutive years. We use an improvement in one metric as a classifier predicting an improvement in the other concept and measure the precision and precision lift. Metric improvement is defined as an increase of our continuous labeling functions.

Table 3 presents the results of the co-change analysis. 'Classifier' is the metric that improved. 'Concept' is the metric that we checked its probability of improvement given an improvement in the classifier. In each cell we present the precision and in parenthesis the precision lift. Note that precision lift is a symmetric function and stays the same when replacing the classifier with the concept. The diagonal is empty since it represents the comparison of a metric with itself.

Table 3: Validation of labeling functions using co-change precision (precision lift in parenthesis)

	Classifier		
Predicted concept	Messages	Refactoring	Hours
Messages		0.20 (0.16)	0.46 (0.04)
Refactoring	0.65 (0.16)		0.45 (0.02)
Hours	0.58 (0.04)	0.18 (0.02)	

In all cases the precision lift is positive, indicating an increased probability of improvement in one metric, given an improvement in the other. Using controls, out of the 186 control cases, only in 18 cases there was a negative lift. Hence, the co-change analysis also shows high robustness with respect to controls.

5.5 Labeling Functions Reliability

We would like our functions to be reliable, returning similar results when measuring the same entity again. However, we cannot measure the same labeling function on the same developer twice at the same time.

As a second best, we compare the value of the labeling functions for the same developer, in the same project, in consecutive years [7]. Consecutive years are not the same time yet not too far from it. Given the value in one year and the consecutive year, we compute two metrics. Self-Pearson is the Pearson correlation of the pairs of metrics for all developers in all projects. Hence, it measures the relative change in the developer ranking as time goes by. Relative difference is the average of the difference between the metric values in the consecutive years, divided by the value in the first year. This metric ignores the rest of the population and measures the average difference between two measurements. Hours had self Pearson of 0.59, and average relative difference of 5%. Refactoring had self Pearson of 0.63, and average relative difference of -13%. Message length had self Pearson of 0.11, rather low, and average relative difference of 15%. Note that average message length is very sensitive and even a single long message can change it dramatically.

6 RESULTS

Given the labeling functions for motivation, we now turn to seeing how they can be used with the GitHub dataset.

6.1 Predicting Retention

The first use is to predict the retention using the other labeling functions. Early prediction of abandonment is important because it allows intervention and possibly avoiding churn. Table 4 presents retention probabilities given the labeling functions. The 'Classifier' column is a labeling function. The 'Retention' column is the precision of predicting the retention concept, given that the classifier is high. In the 'Two Years Retention', we extend the co-change analysis and the classifier is **an improvement** in the metric from one year of a developer in a project to the next year; the column shows the retention rate in the next year.

Table 4: Labeling functions' predictions of retention

Classifier	Retention	Retention Lift	Two Years Retention	Two Years Lift
None	71	-0.058	75	-0.073
High Refactoring	75	0.003	81	-0.001
Long Messages	77	0.025	81	0.009
High Hours	90	0.196	86	0.061
All	90	0.204	86	0.063
Positive Rate	75	0.0	81	0.0

The baseline for each analysis is the positive rate, the probability of retention. In both cases, when none of the labeling functions is high, the retention rate is lower, and when all are high it is the highest. The lift of high hours is almost as high as for all labeling functions together. Long messages have 2.5% lift in retention and the lift in the other cases is close to zero.

To learn if the predictive power is due to the labeling functions or the controls, we compared the performance of models built on the functions, the controls, and both. Aiming for precision, logistic regression models reached precision of 91% and recall of 21% on both the labeling functions alone and when adding the controls. Using the controls alone it reached precision of 90% yet with recall of only 7%, hence the labeling functions have better predictive power, above the contribution from the controls.

We also investigated if the prediction can be improved by using the raw metrics and not the labeling functions (e.g., knowing of 24 distinct hours and not just a Boolean value). Logistic regression had a precision of 89% (2 percentage points less) yet recall of 37% (16 percentage points higher), higher Jaccard, and higher mutual information. This indicates that their use might be beneficial in some settings yet without a dramatic change.

Note that logistic regression is a low-capacity model and our dataset is large, reducing the threat of over-fitting. We also checked high-capacity models such as random forests, boosting, and neural networks to build models of higher representation ability and performance. They had lower performance, indicating that representation power is not the limiting constraint.

We used the 'Two Years Retention' to try to predict retention in the second year. Our co-change analysis (Section 5.4) used a single metric. We now apply the full power of supervised learning to predict changes. We allow more inputs and provide the functions in one year, the year afterwards, the difference (to ease representation), and controls. On this dataset we can apply any supervised learning classifier, learning complex and powerful representations. We were able to reach precision of 92% with recall of 33%, higher than the precision of 86% when all metrics improve. Hence, the application of the more powerful method is beneficial, yet since the baseline result is high it is not dramatic.

6.2 Developer Performance by Labeling Functions

We next use the labeling functions to estimate the relation between motivation and various metrics of developer performance. Table 5 has a 'Metric' column and a 'Description' column, explaining the metrics. There is an additional column per labeling function. The cell intersecting a labeling function column and a metric row represents $\frac{Avg(metric|function=High)}{Avg(metric|function=Low)}$, where 1.0 means similar averages. Note that the labeling functions are weak classifiers, labeling some motivated people as unmotivated and vice versa. Therefore the results are not accurate yet indicate the nature of the relations with motivation.

The first four rows match the labeling functions with each other. When a column labeling function is high, the average of the other labeling functions raw metrics is expected to be high, if they all capture motivation. This indeed happens in all cases, except the disagreement between hours and message length, providing an additional validation of the functions.

We are interested in three performance aspects: activity, output, and process motivation. We measure each aspect with two metrics to reduce the influence of a single metric's artifacts. Activity is measured by activity days, and, to better fit work by both full-time employees and volunteers, activity period. Output is measured by commits and files as units of work. Commits and other output units in software engineering are not of a single size (e.g., a commit might represent different amounts of work). However, commits are a common way to measure output [1]. We did not used issues or pull requests, which are not available in our dataset, yet measuring by either commits, issues, or pull requests tend to agree [7].

Developers might be driven by output motivation (wanting to produce more) or process motivation (produce better) [55]. Corrective commit probability is influenced by both the existence of bugs and their detection effectiveness. Process motivation might lead to investing more in detection effectiveness (e.g., by writing tests)

Table 5: Developer performance by labeling functions

Metric	Description	Reten.	Hours	Msgs.	Refact.
Retention	Probability of continuing in the next year		1.29	1.05	1.01
Commit Hours	Number of distinct hours of the day during a year	1.25	1.85	0.99	1.04
Message Length	Average number of characters in a commit message	1.13	0.91	5.75	2.41
Refactor Prob.	Attempt to improve the software [6, 7]	1.04	1.12	2.02	6.70
Activity Period	Days between the first and last commits in a year	1.70	1.41	1.11	1.01
Activity Days	Number of distinct days in which a commit was made	2.09	4.06	0.79	1.03
Commits	The number of commits, modifications of the code	2.08	4.03	0.80	1.03
Files Edited	Number of Files modified (or created)	1.62	2.99	0.84	0.99
Commit Duration	Average gross duration of a commit [3, 5, 7]	1.07	1.26	1.44	1.19
Corrective Commit Prob.	The ratio of bug fixing commits, Measures bug fixing effort [7]	1.08	1.03	1.98	1.57

which can lead to finding more bugs. Core developers abandoning the project (possibly lacking motivation) reduce effectiveness [7]. Similarly, commit duration is influenced by productivity and process, e.g., writing tests increase commit duration by 18% [3].

We expect that motivation will lead to higher involvement, as reflected by a longer activity period, more distinct activity days, more commits, and more files edited [55]. Our results indicate that in general this indeed happens. Activity period is longer for all functions. The increase in activity days is higher than the one of activity period in all cases other than messages where it is even lower than one. The ratios for commits are almost identical to those of activity days. However, since commit duration is longer, more time was invested to perform these commits. The ratios of files edited are lower than for commits but follow a similar pattern.

When we look at the table with respect to the labeling functions, metrics are always higher for retention and hours. For refactoring it holds other than the 0.99 for files edited. Long messages have a significant drop to 0.79 in activity days. Part of this is due to confounding variables like the tendency to short messages and long activity periods in projects of few developers. When controlling by developer group, the activity period is higher given long messages. Commits and files edited also have a drop yet per active day they improve.

The results can also be used to address cases where motivation can be hypothesized to have opposite effects [55]. For example, it may be claimed that motivation compensates for the tedious effort of fixing bugs, and therefore motivated individuals will perform more bug fixing. Alternatively, motivated individuals might apply more attention to their work, increasing quality, and therefore will have less bugs and require less bug fixes. The results indicate that the first hypothesis dominates: using all four labeling functions, higher motivation seems to go with higher corrective commit probability.

The same goes for commit duration, which is longer for all functions. One could expect a decrease due to output motivation and higher productivity. An increase can be explained by process motivation, higher attention and standards. Hence, our results better fit process motivation, aiming to produce better, than they fit output motivation, aiming to produce more [55].

As an additional validation, we compare the metrics in same-year twins experiments, comparing a project in which the developer continued to one abandoned. Commits are higher in the continued project in 65% of the twins pairs, files edited in 56%, activity period in 74%, activity days in 65%, and commit duration is higher in 55% (at random 50% is expected). Opposed to the table, CCP is lower in 86%. Results hold when controlled by any of our control variables.

Co-change analysis showed the improvements in the labeling functions lead to higher metric values for all the metrics. These results are the same as in the table.

7 RELATED WORK

Demarco and Lister [24], and also Frangos [27], claim that the important software problems are human and not technological. So there has been intensive investigation of motivation in software engineering [17, 28, 40].

Our work, and specifically the labeling functions, were designed to align with psychological motivation theories. Commits, refactoring, and hours are aligned with McClelland's [45] affiliation and achievement, and in certain contexts authority. Vroom's Expectancy Theory [59] predicts higher outcome from refactoring and documentation for motivated developers planning to stay. All our functions are aligned with ownership (e.g., Motivation-Hygiene Theory [34], others [14, 38]) and more motivators.

Open-source development is the collaborative development of software that is free to use and further modify [48]. It is common to develop open source software as a volunteer, which means that salary is not a motivator [41]. Therefore, the motivation of open source developers was investigated as a specific domain, in an effort to uncover other motivators [22, 60].

Ownership and autonomy are important motivators [14, 38]. We saw in Section 3.1 that retention is significantly higher in smaller projects, in which ownership and autonomy are high. Recognition [30] is another motivator. Recognition is stronger in popular projects that have lower retention. This is aligned with external motivators like recognition being weaker than internal ones like ownership [2].

Touré-Tillery and Fishbach distinguish between output motivation (producing more) and process motivation (producing well) [55]. We saw that commit duration increases, by all labeling functions, indicating process motivation — giving more attention to each commit instead of trying to finish them faster. They also claim that motivation is demonstrated in choice, speed, and performance.

8 THREATS TO VALIDITY AND LIMITATIONS

We discussed the limitations and biases of each labeling function when presenting it.

We base our work on machine learning frameworks [10, 47, 50]. However, the application of these frameworks to motivation and software engineering is new and therefore there are no benchmarks or other labeling functions to which we can compare. Instead, we validated the functions in many ways to reduce the risk of error.

Studying motivation poses a challenge since motivation is an internal abstract concept.

For example, survey answers might have reliability problems due to ego defenses [15], subjectivity, different personal scales, etc. Comparison of self-rating to those of a related person (e.g., supervisor, co-worker, or a spouse) showed only moderate correlation [12, 36, 43]. Therefore, we also independently evaluated the actual behavior working at GitHub on 151,775 developers.

The number of survey answers that were matched with the behavior in GitHub is low. Note that this is despite the survey itself being completed by 521 developers. Hence, there was a low probability of being able to match a person. This led to threats of both response bias [32] and incorrect statistical estimation problems [58]. Since the survey is already large it seems it will be hard to do a larger survey. Cooperation with companies, which have behavior information and might agree to conduct surveys is a possible option.

Motivation might be due to many motivators, like enjoyment, self-use, community, etc. We did not consider all these motivators and possible relations between them but only the outcome as motivation. For example, it is possible that people motivated by self-use will contribute only a single modification that helps them and therefore retention is irrelevant to their motivation. The survey we used included questions on 11 motivators and in future work we can apply the same methodology and validate labeling functions for specific motivators, getting a finer-grained picture of people's motivations.

A hard to notice threat is due to the motivation level. All the developers that we analyze contributed at GitHub hence are somewhat motivated in the first place. Hence, instead of comparing motivated and unmotivated people, we might have compared motivated and highly motivated people. This might turn out to be a benefit since members of organizations, and communities also have minimal motivation, as in our scenario.

9 CONCLUSIONS

GitHub contains years of data on thousands of developers in their natural every-day software development. We suggest four labeling functions based on behavioral cues [25] which enable using this data to study motivation and its effects in software development.

We first validated that the labeling functions are weak classifiers by predicting developers' answers to two motivation questions from a survey. We then checked agreement between the functions, monotonicity, agreement per person using twin experiments, and temporal agreement using co-change. We used control variables, alone and combined in a supervised learning model, to verify that the labeling functions add predictive power beyond these variables.

Our results reproduce prior work on the positive impact of motivation: the activity period is up to 70% longer, up to 44% higher time investment in commit, and up to 300% more commits. We also built models for developer retention. A high precision retention model can be used to identify dedicated developers on which a project can rely. A high recall model can be used to identify developers lacking motivation (those not identified by the model), allowing intervention that might increase it.

Our application of the methodology to motivation is just one example. Additional labeling functions can be used to obtain even better characterizations. Specifically, our survey included questions about 11 motivators, for which labeling functions can be built allowing a drill down into motivation details.

More importantly, the same methodology can be used for studying other concepts, especially when one cannot obtain a precise labeling of the concept due to its ambiguity, the cost of labeling, or noise. Using our methodology facilitates quantified, reproducible, long-term investigation, based on large-scale data from real projects.

EXPERIMENTAL MATERIALS

The replication package (DOI 10.5281/zenodo.10519880) can be found at https://zenodo.org/records/10519880. Most up-to-date version is available at https://github.com/evidencebp/motivation-labeling-functions.

ACKNOWLEDGMENTS

We thank David Amit, Daniel Shir, Assa Bentzur, Yaniv Mama, Yinnon Meshi, Aviad Baron, and Gil Shabtai for the discussions and their insights.

REFERENCES

[1] Abdulkareem Alali, Huzefa Kagdi, and Jonathan I. Maletic. 2008. What's a Typical Commit? A Characterization of Open Source Software Repositories. In *2008 16th IEEE International Conference on Program Comprehension*. 182–191. https://doi.org/10.1109/ICPC.2008.24

[2] Teresa M. Amabile, Karl G. Hill, Beth A. Hennessey, and Elizabeth M. Tighe. 1994. The work preference inventory: Assessing intrinsic and extrinsic motivational orientations. *Journal of Personality and Social Psychology* 66, 5 (1994), 950–967. https://doi.org/10.1037/0022-3514.66.5.950

[3] Idan Amit. 2020. Software development task effort estimation. U.S. patent application #US20220122025A1. https://patents.google.com/patent/US20220122025A1/

[4] Idan Amit. 2021. End to End Software Engineering Research. *arXiv preprint arXiv:2112.11858* (2021). arXiv:2112.11858 [cs.SE]

[5] Idan Amit, Nili Ben Ezra, and Dror G. Feitelson. 2021. Follow Your Nose – Which Code Smells are Worth Chasing? *arXiv preprint arXiv:2103.01861* (2021). arXiv:2103.01861 [cs.SE]

[6] Idan Amit and Dror G. Feitelson. 2019. Which Refactoring Reduces Bug Rate?. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering* (Recife, Brazil) *(PROMISE'19)*. Association for Computing Machinery, New York, NY, USA, 12–15. https://doi.org/10.1145/3345629.3345631

[7] Idan Amit and Dror G. Feitelson. 2021. Corrective commit probability: a measure of the effort invested in bug fixing. *Software Quality Journal* 29, 4 (Aug 2021), 817–861. https://doi.org/10.1007/s11219-021-09564-z

[8] Idan Amit and Dror G. Feitelson. 2024. A Large Scale Survey of Motivation in Software Development and Analysis of its Validity. *arXiv preprint arXiv:2404.08303* (2024). arXiv:2404.08303 [cs.SE]

[9] Idan Amit, Eyal Firstenberg, Jonathan Allon, and Yaron Neuman. 2020. Identifying changes in use of user credentials. US Patent 10,686,829.

[10] Idan Amit, Eyal Firstenberg, and Yinnon Meshi. 2017. Framework for semi-supervised learning when no labeled data is given. U.S. patent #US11468358B2. https://patents.google.com/patent/US11468358B2

[11] Idan Amit, John Matherly, William Hewlett, Zhi Xu, Yinnon Meshi, and Yigal Weinberger. 2019. Machine Learning in Cyber-Security - Problems, Challenges and Data Sets. *arXiv preprint arXiv:1812.07858* (2019). arXiv:1812.07858 [cs.LG]

[12] M. Argyle. 1989. Do happy workers work harder? The effect of job satisfaction on job performance. In *How harmful is happiness? Consequences of enjoying life or not*, Ruut Veenhoven (Ed.). Universitaire Pers, Rotterdam, The Netherlands.
[13] Dan Ariely, Emir Kamenica, and Dražen Prelec. 2008. Man's search for meaning: The case of Legos. *Journal of Economic Behavior & Organization* 67, 3-4 (2008), 671–677. https://doi.org/10.1016/j.jebo.2008.01.004
[14] Nathan Baddoo and Tracy Hall. 2002. Motivators of Software Process Improvement: an analysis of practitioners' views. *Journal of Systems and Software* 62, 2 (2002), 85 – 96. https://doi.org/10.1016/S0164-1212(01)00125-X
[15] Nigel Bassett-Jones and Geoffrey C. Lloyd. 2005. Does Herzberg's motivation theory have staying power? *Journal of Management Development* 24 (12 2005), 929–943. https://doi.org/10.1108/02621710510627064
[16] Debby GJ Beckers, Dimitri van der Linden, Peter GW Smulders, Michiel AJ Kompier, Marc JPM van Veldhoven, and Nico W van Yperen. 2004. Working overtime hours: relations with fatigue, work motivation, and the quality of work. *Journal of Occupational and Environmental Medicine* (2004), 1282–1289.
[17] Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. 2008. Motivation in Software Engineering: A systematic literature review. *Information & Software Technology* 50, 9-10 (2008), 860–878. https://doi.org/10.1016/j.infsof.2007.09.004
[18] Sharon Belenzon and Mark A Schankerman. 2008. Motivation and sorting in open source software innovation. *Available at SSRN 1311136* (2008).
[19] Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (Madison, Wisconsin, USA) *(COLT' 98)*. ACM, New York, NY, USA, 92–100. https://doi.org/10.1145/279943.279962
[20] Fabio Calefato, Marco Aurelio Gerosa, Giuseppe Iaffaldano, Filippo Lanubile, and Igor Steinmacher. 2021. Will You Come Back to Contribute? Investigating the Inactivity of OSS Core Developers in GitHub. arXiv:2103.04656 [cs.SE]
[21] Leshem Choshen and Idan Amit. 2021. ComSum: Commit Messages Summarization and Meaning Preservation. *arXiv preprint arXiv:2108.10763* (2021). arXiv:2108.10763 [cs.CL]
[22] Andrea Bonaccorsi Cristina and Cristina Rossi. 2004. Altruistic individuals, selfish firms? The structure of motivation in Open Source software. *First Monday* 9 (2004), 9. https://doi.org/10.5210/fm.v9i1.1113
[23] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R Bradski. 2006. Self-supervised monocular road detection in desert terrain.. In *Robotics: science and systems*, Vol. 38. Philadelphia.
[24] Tom DeMarco and Tim Lister. 2013. *Peopleware: productive projects and teams*. Addison-Wesley.
[25] Giel Dik and Henk Aarts. 2007. Behavioral cues to others' motivation and goal pursuits: The perception of effort facilitates goal inference and contagion. *Journal of Experimental Social Psychology* 43, 5 (2007), 727–737. https://doi.org/10.1016/j.jesp.2006.09.002
[26] M. Fowler. 2018. *Refactoring: Improving the Design of Existing Code*. Pearson Education. https://books.google.de/books?id=2H1_DwAAQBAJ
[27] S. A. Frangos. 1997. Motivated Humans for Reliable Software Products. In *Reliability, Quality and Safety of Software-Intensive Systems*, Dimitris Gritzalis (Ed.). Springer US, Boston, MA, 83–91. https://doi.org/10.1007/978-0-387-35097-4_7
[28] C. França, F. Q. B. da Silva, and H. Sharp. 2020. Motivation and Satisfaction of Software Engineers. *IEEE Transactions on Software Engineering* 46, 2 (Feb 2020), 118–140. https://doi.org/10.1109/TSE.2018.2842201
[29] Yoav Freund and Robert E Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. System Sci.* 55, 1 (1997), 119–139. https://doi.org/10.1006/jcss.1997.1504
[30] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. 2021. The Shifting Sands of Motivation: Revisiting What Drives Contributors in Open Source. arXiv:2101.10291 [cs.SE]
[31] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. 2021. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software* 175 (May 2021), 110911. https://doi.org/10.1016/j.jss.2021.110911
[32] Walter R Gove and Michael R Geerken. 1977. Response bias in surveys of mental health: An empirical investigation. *American journal of Sociology* 82, 6 (1977), 1289–1317.
[33] Adam Grant. 2008. The Significance of Task Significance: Job Performance Effects, Relational Mechanisms, and Boundary Conditions. *The Journal of Applied Psychology* 93 (Feb 2008), 108–24. https://doi.org/10.1037/0021-9010.93.1.108
[34] F. Herzberg, B. Mausner, and B. B. Snyderman. 1959. *Motivation to Work*. Wiley, New York.
[35] Austin Bradford Hill. 1965. The environment and disease: association or causation?
[36] Timothy A. Judge, Edwin A. Locke, Cathy C. Durham, and Avraham N. Kluger. 1998. Dispositional effects on job and life satisfaction: The role of core evaluations. *Journal of Applied Psychology* 83 (1998), 17–34. https://pdfs.semanticscholar.org/9912/e58168ca993de3fa8105bd1c64fd63f7ddb3.pdf
[37] Michael Kearns. 1988. Learning Boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory* (1988).
[38] Chak Fu Lam and Suzanne T. Gurland. 2008. Self-determined work motivation predicts job outcomes, but what predicts self-determined work motivation? *Journal of Research in Personality* 42, 4 (2008), 1109–1115. https://doi.org/10.1016/j.jrp.2008.02.002
[39] BA Lameijer, J Antony, A Chakraborty, RJMM Does, and JA Garza-Reyes. 2021. The role of organisational motivation and coordination in continuous improvement implementations: an empirical research of process improvement project success. *Total Quality Management & Business Excellence* 32, 13-14 (2021), 1633–1649.
[40] Per Lenberg, Robert Feldt, and Lars Göran Wallgren. 2015. Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software* 107 (2015), 15 – 37. https://doi.org/10.1016/j.jss.2015.04.084
[41] Josh Lerner and Jean Tirole. 2002. Some Simple Economics of Open-Source. *Journal of Industrial Economics* 50 (02 2002), 197–234. http://www.people.hbs.edu/jlerner/simple.pdf
[42] Brenda L. Mak and Hy Sockel. 2001. A confirmatory factor analysis of IS employee motivation and retention. *Information & Management* 38, 5 (2001), 265–276. https://doi.org/10.1016/S0378-7206(00)00055-0
[43] Christina Maslach, Susan Jackson, and Michael Leiter. 1997. The Maslach Burnout Inventory Manual. *Evaluating Stress: A Book of Resources* 3 (01 1997), 191–218.
[44] John D Mayer, Michael A Faber, and Xiaoyan Xu. 2007. Seventy-five years of motivation measures (1930–2005): A descriptive analysis. *Motivation and Emotion* 31 (2007), 83–103.
[45] D. C. McClelland. 1961. *The Achieving Society.* Van Nostrand, Princeton, NJ.
[46] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde. 2019. What Predicts Software Developers' Productivity? *IEEE Transactions on Software Engineering* (2019), 1–1. https://doi.org/10.1109/TSE.2019.2900308
[47] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data Programming: Creating Large Training Sets, Quickly. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 3567–3575. http://papers.nips.cc/paper/6523-data-programming-creating-large-training-sets-quickly.pdf
[48] Eric S. Raymond. 1999. *The Cathedral and the Bazaar* (1st ed.). O'Reilly & Associates, Inc., Sebastopol, CA, USA.
[49] Bishal Sainju, Chris Hartwell, and John Edwards. 2021. Job satisfaction and employee turnover determinants in Fortune 50 companies: Insights from employee reviews from Indeed.com. *Decision Support Systems* 148 (2021), 113582. https://doi.org/10.1016/j.dss.2021.113582
[50] Robert E Schapire. 1990. The strength of weak learnability. *Machine learning* 5, 2 (1990), 197–227.
[51] Yulia Shmerlin, Irit Hadar, Doron Kliger, and Hayim Makabee. 2015. To Document or Not to Document? An Exploratory Study on Developers' Motivation to Document Code. In *Advanced Information Systems Engineering Workshops*, Anne Persson and Janis Stirna (Eds.). Springer International Publishing, Cham, 100–106.
[52] Frank J Smith. 1977. Work attitudes as predictors of attendance on a specific day. *Journal of applied psychology* 62, 1 (1977), 16.
[53] Sonja A Swanson, Matthew Miller, James M Robins, and Miguel A Hernán. 2015. Definition and evaluation of the monotonicity condition for preference-based instruments. *Epidemiology (Cambridge, Mass.)* 26, 3 (2015), 414.
[54] Kristi Toode, Pirkko Routasalo, and Tarja Suominen. 2011. Work motivation of nurses: A literature review. *International Journal of Nursing Studies* 48, 2 (2011), 246–257. https://doi.org/10.1016/j.ijnurstu.2010.09.013
[55] Maferima Touré-Tillery and Ayelet Fishbach. 2014. How to Measure Motivation: A Guide for the Experimental Social Psychologist. *Social and Personality Psychology Compass* 8 (07 2014). https://doi.org/10.1111/spc3.12110
[56] Leslie G Valiant. 1984. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.
[57] Steven G Vandenberg. 1966. Contributions of twin research to psychology. *Psychological Bulletin* 66, 5 (1966), 327.
[58] V. Vapnik and A. Chervonenkis. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications* 16, 2 (1971), 264–280. https://doi.org/10.1137/1116025 arXiv:https://doi.org/10.1137/1116025
[59] Victor. H. Vroom. 1964. *Work and Motivation.* Wiley, New York.
[60] Yunwen Ye and Kouichi Kishida. 2003. Toward an Understanding of the Motivation Open Source Software Developers. In *Proceedings of the 25th International Conference on Software Engineering* (Portland, Oregon) *(ICSE '03)*. IEEE Computer Society, Washington, DC, USA, 419–429. http://dl.acm.org/citation.cfm?id=776816.776867