Prediction of Hourly Earnings and Completion Time on a Crowdsourcing Platform

Anna Lioznova
Alexey Drutsa
Yandex, Moscow, Russia
lioznova.anna@,adrutsa@yandex.ru

Vladimir Kukushkin*
Huawei, Saint Petersburg, Russia
kukushkin.v@gmail.com

Anastasia Bezzubtseva
Yandex, Moscow, Russia
nstbezz@yandex-team.ru

ABSTRACT

We study the problem of predicting future hourly earnings and task completion time for a crowdsourcing platform user who sees the list of available tasks and wants to select one of them to execute. Namely, for each task shown in the list, one needs to have an estimated value of the user's performance (i.e., hourly earnings and completion time) that will be if she selects this task. We address this problem on real crowd tasks completed on one of the global crowdsourcing marketplaces by (1) conducting a survey and an A/B test on real users; the results confirm the dominance of monetary incentives and importance of knowledge on hourly earnings for users; (2) an in-depth analysis of user behavior that shows that the prediction problem is challenging: (a) users and projects are highly heterogeneous, (b) there exists the so-called "learning effect" of a user selected a new task; and (3) the solution to the problem of predicting user performance that demonstrates improvement of prediction quality by up to 25% for hourly earnings and up to 32% completion time w.r.t. a naive baseline which is based solely on historical performance of users on tasks. In our experimentation, we use data about 18 million real crowdsourcing tasks performed by 161 thousand users on the crowd platform; we publish this dataset. The hourly earning prediction has been deployed in Yandex.Toloka.

ACM Reference Format:
Anna Lioznova, Alexey Drutsa, Vladimir Kukushkin, and Anastasia Bezzubtseva. 2020. Prediction of Hourly Earnings and Completion Time on a Crowdsourcing Platform. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403369

1 INTRODUCTION

Crowdsourcing constitutes one of the most important technologies used by modern data-driven companies, e.g., search engines and social networks [2, 19, 28]. In order to obtain data labeled by humans, they either build and use their own in-house croudsourcing solutions [24], or process their tasks in crowdsourcing marketplaces (e.g. Amazon Mechanical Turk [19] and Yandex.Toloka [6, 7, 9, 11, 35]). A crowdsourcing marketplace represents a platform that operates in a two-sided market ecosystem of *requesters* (a.k.a. *customers*), that have tasks to be done, and *humans* (a.k.a. *users*, *performers*, *workers*, etc.), that are willing to perform tasks for some reward. The key goal of a two-sided market platform is to improve experience of each side of the market (users and requesters) and to make effective matching of their needs [12]. On the one hand, there is a large body of works devoted to needs of requesters: how to get completed tasks faster [5], cheaper [26], with higher quality [14, 33], etc. On the other hand, the following question is understudied in the current literature: how a crowdsourcing platform should help users to realize their incentives and to achieve their needs? Moreover, it is reported that a crowdsourcing marketplace usually have poor interfaces [17, 25], and it is argued that the larger the market grows the harder and more frustrating user interactions with the platform are [15]. The permanent appearance of external tools and instruments that help users to navigate better in the market environment may serve as a manifestation of this deficiency [16, 20, 21]. Hence, emerging and growing crowd marketplaces are quite interested in improving user interfaces to make labor supply sustainable.

A user in most of the major crowdsourcing marketplaces is free to select a task to execute among a large set of projects that are available to her and are provided by a variety of requesters [21]. In order to help such a user to make the best choice that align with her needs and incentives, a good crowdsourcing platform should provide reliable information about each task, instruments to search among available tasks, and tools to filter & sort lists of tasks. Most incentives of crowdsourcing platform users are monetary [18, 24], in particular, they select tasks based on expected earnings and completion time [21, 31]. We have also conducted a fresh survey on 5.4 thousand users of a crowdsourcing platform that reports: the main participation goal of 68% users is to make money in their spare time, while reward and hourly earnings are one of the most popular characteristics that affect task choice by a user (see Sec. 4). Information on hourly earnings of a user on tasks of a particular type may also be useful in effective pricing and task assignment mechanisms [34, 37], and knowledge of this information is hence valuable for crowdsourcing platform owners as well.

In this work, we study Yandex.Toloka [35], one of the largest global crowdsourcing marketplace platform. This platform connects requesters and performers as follows: (a) having a batch of tasks (referred to as a *project*) that should be performed by humans (e.g., to compare two images or to classify a text document), each requester provides a setting on how to execute them (a task instruction, reward per task, quality control, etc.) and posts the tasks to the marketplace; (b) each performer observes the list of all available

*This work was performed while working at Yandex.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403369

projects (with provided information) on the main web page of the platform, where she is able to select one of them to execute (see Fig. 1). Currently, 2 million users of this platform have already completed 4.5 billion tasks, and from 25 to 30 thousand users perform tasks of 500 different types every day in Toloka [35].

In order to additionally show the importance of hourly earnings for users, we provide the following empirical evidence from an A/B experiment conducted on new users of studied crowdsourcing platform. A user from the treatment group of this experiment saw expected hourly earnings for each task in the list of available ones and was able to sort the list by this parameter, while a user from the control group saw the list of tasks without these information and sorting. The treatment version demonstrated statistical significant improvements: first, the user return rate in the second week after registration and the average number of completed tasks per user were increased by +7.4% and by +10.8%, respectively, and, second, 9.7% of users utilized the sorting by hourly earnings (see more details in Sec. 4). Hence, providing a user with information on expected hourly earnings is in interest both of users (they are able to make decisions in line with their incentives) and of requesters (growth of the effective labor force supply in the two-sided market).

In the current study, we focus on the problem of *prediction of future hourly earnings* during execution of a task for a user who will select this task to perform among available ones. Since, at the moment of task selection, reward for a (completed) task is known in advance, prediction of hourly earnings is interconnected with prediction of the time that will be spent by the user to complete it[1]. Moreover, information on expected completion time before selecting a task may be of independent interest to the user: e.g., based on this, she could find a task which can be completed within a free time that she has. In this way, *prediction of task completion time* constitutes the second focus of our work.

For a task from a project that has been performed previously by a user, estimation of earnings and time for this user may be easily made with moderate accuracy both by the user and by the platform (just by inferring these parameters of the task from previous behavior of the user). However, in the opposite case of a new task type for a user, the prediction problem is much more challenging, because: (1) the nature of open marketplace implies that its projects and users are very heterogeneous [29]; and (2) there exists a so-called "learning effect" when, in an initial period, performance of a user on a new task type differs from her performance after she gets mastered tasks of this type. But at the same time this case is quite important since: (a) the number of available projects is much more larger than the number of the earlier performed ones by a particular user; (b) a user needs to know the task parameters when she wants change currently performed project to a new one; and (c) the market demand side permanently evolves what results in every day presence of new projects that should be chosen by users.

In this work, we conduct analysis and prediction of hourly earnings and task completion time by means of several models. We utilize features about a crowdsourcing task (e.g., readability of instruction, the number of input fields) and previous behavior of users (e.g., past hourly earnings). In our experimentation, we use

[1]**Do not confuse it** with prediction of throughput of a platform for a particular requester project (a.k.a. latency). The latter estimates how much time is needed to complete all the tasks in the project given the current state of market supply [5].

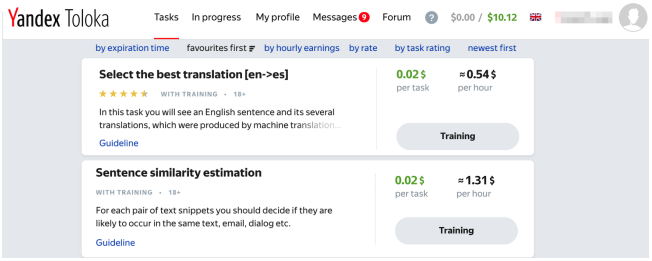

Figure 1: The main page of Yandex.Toloka with a list of projects and the experimental sorting by hourly earnings.

data about real crowdsourcing tasks performed by 161377 users on the studied platform Yandex.Toloka during 3 months in 2018. We publish this dataset for reproducibility of our results and to provide the research community with rich crowdsourcing data for future studies (see Sec. 6). Hourly earning predictions *have been deployed in Yandex.Toloka* and are available to users when they select tasks.

To sum up, our paper focuses on the problem, which coincides with the *present and emerging needs of crowdsourcing marketplace platforms*: help users to satisfy their need in reliable information on available human intelligence tasks. Specifically, the major contributions of our study include:

- Introduction of the problem to predict hourly earnings and task completion time for a crowdsourcing platform user who selects a project to execute among available ones.
- Addressing of this problem on real crowd tasks completed in the studied crowdsourcing marketplace Yandex.Toloka by
  - analysis of user behavior that shows specifics of the problem: (1) high heterogeneity of users and projects, (2) the "learning effect" of a user selected new projects;
  - the solution to the problem of predicting user performance that demonstrates improvement of prediction quality by up to 25% for hourly earnings and up to 32% completion time w.r.t. a naive baseline which is based solely on historical performance of users.
- A new public dataset containing information on 18 million real tasks completed by 161 thousand users for 767 crowd projects in the crowdsourcing marketplace Yandex.Toloka.

2 RELATED WORK

Task selection. According to [21], crowdsourcing platform users waste a lot of time wandering on the platform, looking for a proper task to accomplish. 22% of users reported that task search was "Very" or "Extremely" frustrating. Their findings suggest that improving the task search and selection process could potentially improve user experience and earnings for executors of all levels. [17] provided another estimate: up to 5% of time spent on a platform users spend looking for tasks. [21] also reported that the most important human intelligence tasks (HITs) selection criteria are "Pay per HIT", "Expected Task Completion Time" and "Requester Reputation". The latter two are rarely explicitly shown to users in crowd environments, which complicates the task selection process. [25] investigated crowdworker feedback on AMT on a specialized forum. They also found out that in order to make a choice a user often needs additional information not presented on the task page. Without this knowledge it is hard to decide if the task is worth it

for them. Our survey in Sec.4 extends those findings to our users. [3] developed a browser plugin which collects time tracks for each assignment, employs these data to estimate hourly rate for each assignment, and allows users to sort tasks by an estimated hourly rate. In contrast to our work they provide an external tool that has very limited access to crowd platform data (not all users install the browser plugin) and estimate only via aggregation of hourly earnings over historical performance data of executors on a project (while we use features of tasks that allow significantly boost prediction quality).

Time to completion and hourly earnings. In crowdsourcing literature time to completion (or *latency*) usually refers the time required to finish *a set of HITs* (or a project, in our terminology) given the throughput of a platform. Some of the works are devoted to latency. [13] presented a model that can be used to predict HIT set completion times and pricing policies. [32] fitted a Cox proportional hazard regression model to show the effects of various HIT set parameters (HIT price, HIT length in characters, the amount of money spent by requester, etc.) on latency. They concluded that the completion time is monotonically decreasing for increasing HIT reward values. [23] considered HIT set completion time prediction and its complications caused by assignments left by executors unfinished. They proposed an approach to identify the abandoned assignments and reassign them to other users.

There are few works devoted to *a single HIT* completion time, HIT reward, and hourly wage. [4] modeled the relationship between HIT completion time and error rate by manipulating the time that performers have to complete a task, and propose a data-driven effort metric, ETA (error-time area), that can be used to determine a task's fair price. Studies that pay attention to hourly wage usually emphasize that executors are severely underpaid. [17] conducted an extensive analysis of hourly wage in AMT via an external plugin. They revealed that only 4.2% of users earn more than the US federal minimum wage ($7.25) on average. Some crowdworkers report on their hourly wage claimed a pay rate of $4-6 to be reasonable [25]. The authors of [30] predict hourly wage, however their work significantly differs from our study in several aspects. First, they built a model by fitting completion time (minimizing RMSE for time) and used this model together with reward to predict wage; while, in our work, we show that a model which directly fits hourly wage (minimizes RMSE for wage) significantly outperforms the approach of [30] by 128%-299% in terms of RMSE (see Sec. 9, "Cross-target" learning). Second, that work did not use historical behavior of each individual executor, while we show that (a) executors are highly heterogeneous (see Sec. 7) and (b) historical information provides both strong baseline models and the most important features in our best models (see Sec. 9). Third, in the work [30], data were collected via a browser plugin, which was installed by performers who agreed to use it, what may result in a bias w.r.t. the distribution of the whole set of crowd-platform users. Fourth, the plugin asked users to do additional actions besides the ones required in a HIT, what affected their monotonic performance and, hence, may result in a bias w.r.t. their usual behavior (when they do not perform the additional actions). Finally, such data collection (external to a crowd platform) resulted in a very small dataset with 7.6K HITs and 83 executors. In contrast to that work, we (a) conduct our study using data with real (usual) behavior of users (their behavior has not been affected by our study), that are provided by the considered platform; (b) run an A/B experiment to show that providing of information on hourly wage to users significantly improves key online metrics of the studied platform (see Sec. 4); (c) deal with a dataset containing 18M HITs and 161K users, which is published for reproducibility and future studies.

3 FRAMEWORK: CROWD MARKETPLACE

We study Yandex.Toloka [35], a typical two-sided crowdsourcing marketplace, similar to Amazon Mechanical Turk [19], with *users* (referred to as *tolokers*) on one side and *customers* (referred to as *requesters*) on the other one. Customers provide users with so-called human intelligence tasks (referred to as *HITs* [19] or *microtasks*). Requesters are ready to pay for their execution, whereas tolokers look for opportunities to earn money. HITs usually refer to data collection for machine learning purposes, surveys, side-by-side comparisons, etc.: e.g. a large body of these microtasks relates to classification of texts/images/videos, text recognition from an image, speech recognition from an audio, image segmentation, etc. Task interface along with its instruction are associated with a *project* entity. Once a project is created, the requester is able to upload data to be processed within this particular interface with respect to this particular instruction.

When a user selects a project to execute, the platform assigns to him a batch of microtasks from the project; such batch is referred to as a *tasks* or an *assignment*. A task is a paid unit of user efforts[2]: all the microtasks in the assignment must be done in order to submit it and get the reward for this task. The number of microtasks within a task is set by the requester (often it is a project constant, but not necessarily). The requester also set the reward[3] for a completed task, which might depend on internal factors (e.g., user qualification, project complexity, etc.) and external ones (e.g., the state of supply and demand within the marketplace).

A requester can also apply several mechanisms for quality control [33], including the following well-known techniques: assign the same microtask to different tolokers (with further aggregation via, e.g., majority vote); microtasks with already known ground-truth answers (so-called *honeypots*); calculating of quality skills of users (e.g., accuracy); and limit the access for cheating or poor-qualified tolokers by means of *train* and *exam* tasks.

On the other side of the platform, there are tolokers who do the tasks and earn money. Each toloker arrives at *the main page*, looks at the list of available projects (see Fig. 1), and chooses any of them. The list is composed dynamically and individually for the toloker, depending on her skills and the requirement on these skills from projects. Then, the system assigns a task from the chosen project to the toloker. She is assumed to perform tasks from this project continuously until she decides to leave on her own or until available tasks are over. In the case of leaving, the toloker can go back to the main page and try to switch the project or leave the system at all.

[2]It is not convenient to consider a microtask as a paid unit of user efforts since, in practice, it might be executed quite quickly, in few seconds, and reward per a microtask would be thus less than $0.01.

[3]The studied crowd platform supports dynamic pricing (i.e., after task performing, the reward for future tasks can be modified). However, requesters usually use static rewards in favor of transparency.

4 MOTIVATING SURVEY & A/B EXPERIMENT

If the state of a market was static, tolokers would just stick to their already known favorite projects. However, most crowdsourcing marketplaces have dynamic nature, and ours is not an exception. First, there are several hundreds of requesters, several thousands of projects, few millions of tolokers, and their numbers are growing permanently. Second, each project has its own intensity of task issuing and own task complexity. Finally, rewards may vary significantly, and, e.g., requesters sometimes set inadequately low or excessively high price. So, in such environment, it may be quite difficult even for an experienced toloker to find and to select a project that fits her needs in the best way.

Survey. In order to reveal toloker needs and incentives, we conducted the following survey that lasted 10 days in April, 2018. We have got answers from 5402 users of the studied crowdsourcing platform that were randomly selected. The questionnaire includes the following questions: (1) "*When choosing a task in the platform, I first of all pay attention to (you can select one or more answers):*" with 15 available answers; and (2) "*My main goal in the platform is... (continue by selecting one of the answers)*" with 7 available answers. The survey results are the following. In the question (1), the top answers (with percent of users) are: task simplicity (60%), task reward (59%), type of a task (55%), clearness of task instruction (45%), hourly earnings during task performance (40%), task enjoyment (36%), similarity to previously performed tasks (34%), and each of 8 remaining answers had less than 30% votes. In the question (2), 68% of users answered '*make money in my spare time*', while the second popular answer '*learn something new*' had 14% votes. Hence, *we conclude that the dominant incentive of most of our users is monetary, and, moreover, they pay notable attention to their hourly earnings.*

A/B experiment. In order to evaluate a potential impact of the importance of hourly earnings for users, we conducted an online controlled experiment (A/B test) on new users of the studied crowd platform, following the state-of-the-art methodology [1, 22]. Our A/B test lasted 30 days in April-May, 2018 and was on 64 thousand users, registered at the platform in one of the 16 first days of the experiment period. The users were split into two groups randomly with equal probability. A user from *the treatment group B* (1) saw predicted[4] hourly earnings for each task in the list of available ones and (2) was able to sort the list by this parameter, while a user from *the control group A* saw the current version of the platform (i.e., the list of tasks without these information and sorting). See Fig. 1 for an example of the platform main page in the treatment group B. We emphasize that the experiment was conducted on new users and, hence, the users from the treatment group B had never seen the non-experimental (production/current) version of the platform. Thus, our A/B test is not affected by novelty effects [8, 22], i.e., observed behavior (via metrics) of a user cannot be just because the user investigates elements in an interface that are new with respect to a previously seen variant of the interface.

The results of the A/B test were evaluated by two metrics: the user return rate in the second week after registration (**RR**; measured as the fraction of users that completed at least one task in the second week after registration) and the average number of completed tasks per user (**CTpU**; measured as the mean quantity of tasks completed by users within 2 weeks since registration). The former metric represents *user loyalty*, while the latter one corresponds to *user activity* [10]. The treatment version B with information on hourly earnings demonstrated statistical significant improvements: the metric **RR** increased by +7.4% (p-value[5] = 0.016) and the metric **CTpU** increased by +10.8% (p-value = 0.015). We also investigated how the available sortings were used, and found that, in the treatment group, 9.7% of users utilized (clicked at least once) the new sorting by hourly earnings, while the percentage of users utilized the other sortings (already existed on the main page) are as follows[6]: by reward (11.8%), by time limit (3.9%), by project rating (5.4%), and by project novelty (5.9%). Hence, the new sorting by hourly earnings becomes the second most popular one.

So, we observe that, first, information on expected hourly earnings can improve the engagement of users since this is able to increase their loyalty and activity. Second, this information can boost the matching market due to the growth of the effective labor force supply, what is in interest both of requesters and of the platform owners. Therefore, *we conclude that providing reliable expected hourly earnings to users when they select tasks is an important development direction for a crowdsourcing platform.*

5 PROBLEM STATEMENT

First, from the previous section, we see that it is important to have a good estimation of future hourly earnings of a user who decided to visit the main page and wants to select a task to execute. In this situation, for each project shown in the list on this web page, *we need to have a predicted value of the user's hourly earnings that will be if she selects this project to execute.* Second, at the moment of project selection, reward for a (completed) task is known in advance, and we also know from our data that the reward is usually constant for tasks within one project (see Sec. 3). Therefore, an estimation of hourly earnings can be straightforwardly derived from a predicted amount of time that will be invested by the user to complete a task from a selected project (and vice versa). Note that information on expected completion time before selecting a task may be useful for the user as well (independently of hourly earnings)[7]. In this way, *we also consider the problem of task completion time prediction in the situation described above where user selects a project to execute.*

For a task from a project that has been performed previously by the user, prediction of earnings and time for her may be easily made with moderate accuracy both by the user and by the platform (just by utilizing previous behavior of the user to infer estimations of earnings and time). Therefore, a focus of our study is made to the opposite case of a new task type for the user, where the prediction problem is much more challenging and is quite important as discussed in Sec. 1 and will be shown in the next sections.

6 DATA

For the purposes of our study, we use the logs of user activity on the crowdsourcing platform Yandex.Toloka. We have collected toloker data from a 3-month period in 2018 (from September to November).

[4] The simple approach Baseline-P (see Sec. 9) was utilized here.

[5] P-values are measured by the widely used two-sample Student's t-test [10, 22].

[6] Percents for these "old" sortings are similar in both groups.

[7] E.g. if a user has 5 free minutes, information on completion time is useful to find a task that she is able to perform in 5 minutes.

Table 1: Basic statistics of the dataset.

statistic	mean	median	std.dev
# tasks in project	24142.5	1684	69394.3
# tasks performed by user	114.7	8	573.4
# users performed in project	2026.9	300	4601.2
# projects performed by user	9.6	3	18.3

Namely, we obtain (1) all user visits to the main web page of the platform where a task has been selected within the 3-month period (these events are referred to as *visits with selection* or, simply, *visits*); and (2) all tasks that have been assigned to and have been completed by users in the 3-month period[8] (these events are referred to as *assignments* or *tasks*). Each visit has its timestamp, the user id, the id of the selected project, the id of the task assigned to the user, and other side information. Each task has its start and completion (submission) timestamps, the project id, the user id, the reward, the number of microtasks in, the number of inputs & outputs, and other side information. We also provide some static features about users (e.g., registration date) and projects (e.g., the instruction length). More information on available fields of the visit, task, user, and project data are provided in the README.md file of our dataset.

In this way, we obtain the data that describe the core scheme of user interaction with the platform (see Sec. 3): a user visits the main page, selects a project to execute from the list, gets a task from the project, completes it, receives a next task from the project, and completes it, where two latter steps repeat until she leaves the platform or visits the main page to select another project. This dataset contains information on 161377 users, 767 projects and over 18 million tasks. Some basic statistic is presented in Table 1.

From here on we use only this dataset to make our analysis and experiments on prediction. *For reproducibility of our results and to provide the research community with rich crowdsourcing data for future studies, we publish*[9] *this dataset referred to as* **"Toloka Users & Tasks"** *available online at:* https://toloka.ai/datasets.

7 ANALYSIS

To understand specifics of our prediction problem, we conduct analysis of user hourly earnings and the time required to complete a task by a user. In our analysis, we consider all non-train/exam tasks from the dataset described in Sec. 6.

Notations. Formally, we have the set of assignments (tasks) $\mathcal{A}$, the set of users $\mathcal{U}$, and the set of projects $\mathcal{P}$. From here on, for each task $a \in \mathcal{A}$, we use the following notations: $\tau_a \in \mathbb{Z}_+$ is the completion time in seconds (is equal to the difference between the submission and start timestamps), $c_a \in \mathbb{R}$ is the reward (in US dollars), $m_a \in \mathbb{N}$ is the number of microtasks inside, $u_a \in \mathcal{U}$ is the user who performed this task, and $p_a \in \mathcal{P}$ is the project which contains this task. Let $\mathcal{A}_u = \{a \in \mathcal{A} \mid u_a = u\}$ be all tasks performed by a user $u \in \mathcal{U}$ and $\mathcal{A}_p = \{a \in \mathcal{A} \mid p_a = p\}$ be all completed tasks in a project $p \in \mathcal{P}$, then $\mathcal{A}_{u,p} = \mathcal{A}_u \cap \mathcal{A}_p$ are all tasks performed by the user in the project. We also have the set $Q = \{(u,p) \in \mathcal{U} \times \mathcal{P} \mid \mathcal{A}_{u,p} \neq \varnothing\}$ of all user-project pairs s.t. the user participated in the project, the set $\mathcal{U}_p = \{u \in \mathcal{U} \mid (u,p) \in Q\}$ of all participated in a project $p \in \mathcal{P}$, and the set $\mathcal{P}_u = \{p \in \mathcal{P} \mid (u,p) \in Q\}$ of all projects where a user $u \in \mathcal{U}$ performed tasks.

For any subset of tasks $\mathcal{A}' \subset \mathcal{A}$, one can calculate the average completion time per microtask (shortly, *Time per Microtask* or TpM) and the *hourly earnings* (HE) as follows

$$\mathtt{TpM}(\mathcal{A}') := \frac{\sum_{a\in\mathcal{A}'} \tau_a}{\sum_{a\in\mathcal{A}'} m_a} \text{ and } \mathtt{HE}(\mathcal{A}') := 3600\frac{\sum_{a\in\mathcal{A}'} c_a}{\sum_{a\in\mathcal{A}'} \tau_a}. \quad (1)$$

User and project heterogeneity. First, we analyse how a user $u \in \mathcal{U}$ performs and earns in a project $p \in \mathcal{P}$ w.r.t. all users $\mathcal{U}_p$ in this project. We do it by means of the relative completion time TpM and the relative hourly earnings HE defined as follows:

$$\mathtt{nTpM}^{\text{proj}}_{u,p} = \frac{\mathtt{TpM}(\mathcal{A}_{u,p})}{\mathtt{TpM}(\mathcal{A}_p)} \text{ and } \mathtt{nHE}^{\text{proj}}_{u,p} = \frac{\mathtt{HE}(\mathcal{A}_{u,p})}{\mathtt{HE}(\mathcal{A}_p)}.$$

The interpretation of these measures can be seen from the example: if $\mathtt{nTpM}^{\text{proj}}_{u,p} \ll 1$, then the user u executes tasks from the project p, on average, notably faster than other users. The relative measures $\mathtt{nTpM}^{\text{proj}}_{u,p}$ and $\mathtt{nHE}^{\text{proj}}_{u,p}$ are used in the axes oY of Fig.2(a)&(b), where the joint distribution of user-project pairs Q[10] are presented w.r.t. 3 project statistics that reflect *project popularity* and are used in the axes oX (the total project money $\sum_{a\in\mathcal{A}_p} c_a$, the total project time $\sum_{a\in\mathcal{A}_p} \tau_a$, and the number of participated users $|\mathcal{U}_p|$). Second, similarly, we analyse how a user $u \in \mathcal{U}$ performs and earns in a project $p \in \mathcal{P}$ w.r.t. all projects $\mathcal{P}_u$ where she participated. Again, we use the relative completion time TpM and the relative hourly earnings HE defined in another way:

$$\mathtt{nTpM}^{\text{user}}_{u,p} = \frac{\mathtt{TpM}(\mathcal{A}_{u,p})}{\mathtt{TpM}(\mathcal{A}_u)} \text{ and } \mathtt{nHE}^{\text{user}}_{u,p} = \frac{\mathtt{HE}(\mathcal{A}_{u,p})}{\mathtt{HE}(\mathcal{A}_u)}.$$

The interpretation of these measures can be seen from the example: if $\mathtt{nTpM}^{\text{user}}_{u,p} \ll 1$, then the user u spend, on average, notably less time on a task from the project p than on a task from other projects. The relative measures $\mathtt{nTpM}^{\text{user}}_{u,p}$ and $\mathtt{nHE}^{\text{user}}_{u,p}$ are used in the axes oY of Fig.2(c)&(d), where the joint distribution of user-project pairs Q are presented w.r.t. 3 user statistics that reflect *user popularity* and are used in the axes oX (the total user earnings $\sum_{a\in\mathcal{A}_u} c_a$, the total user time $\sum_{a\in\mathcal{A}_u} \tau_a$, and the number of projects in which user participated $|\mathcal{P}_u|$). We see that user-project pairs significantly differ in their completion time and hourly earnings w.r.t. average performance of a user or a project independently of both user popularity and of project popularity: a lot of log values of $\mathtt{nTpM}^{\text{user}}_{u,p}$, $\mathtt{nTpM}^{\text{proj}}_{u,p}$, $\mathtt{nHE}^{\text{user}}_{u,p}$, and $\mathtt{nHE}^{\text{proj}}_{u,p}$ significantly higher/lower than 0.

Finally, we show that (1) on each project $p \in \mathcal{P}$, different users usually perform differently and (2) each user $u \in \mathcal{U}$ has usually different performance on different projects. For this purpose, we use the coefficient of variation $\text{cv}(X) = \sigma(X)/\mu(X)$, where $\sigma(X)$ is the standard deviation and $\mu(X)$ is the mean value over a set of real values $X \subset \mathbb{R}$. In the case (1), we take $X_p = \{\mathtt{TpM}(\mathcal{A}_{u,p})\}_{u\in\mathcal{U}_p}$ (or $X_p = \{\mathtt{HE}(\mathcal{A}_{u,p})\}_{u\in\mathcal{U}_p}$) and plot the joint distribution of projects $\mathcal{P}$[11] w.r.t. the statistics $\text{cv}(X_p)$ and the number of participated users $|\mathcal{U}_p|$ in top (or bottom, respectively) of Fig.2(e).

[8] We filtered out tasks with no preceding visits by their toloker. This may happen if the visit where the task project has been selected was before the 3-month period.

[9] The data are anonymized: there are no real internal ids, personal user data, proprietary requester data (e.g. task results), and any textual info (e.g. project names, descriptions,..).

[10] We left only (u, p) pairs s.t. $|\mathcal{A}_{u,p}| \geq 5$ in the analysis.

[11] We left only projects p s.t. $|\mathcal{U}_p| \geq 10$ in the analysis.

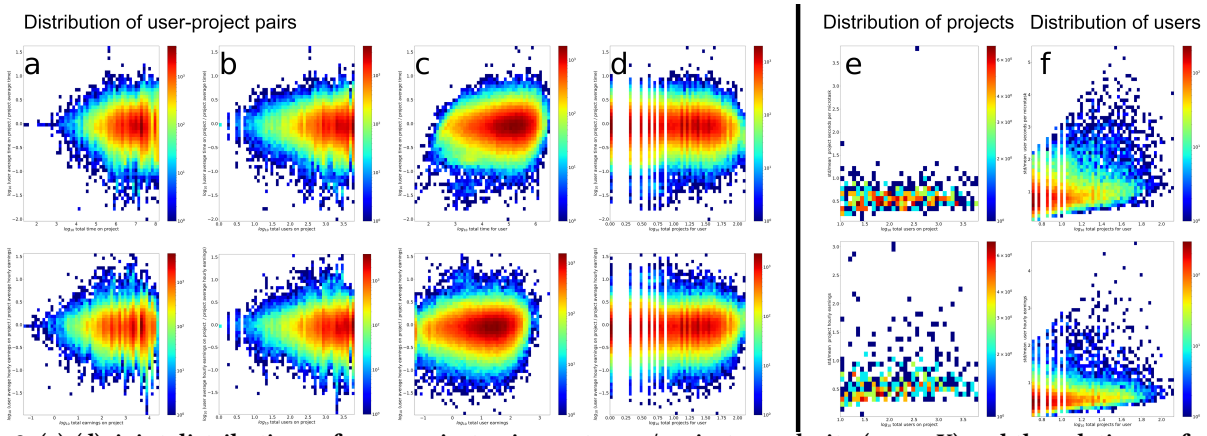

Figure 2: (a)-(d): joint distributions of user-project pairs w.r.t. user/project popularity (axes oX) and the relative performance metric $\mathtt{nM}^{\mathrm{user/proj}}$. (e)/(f): joint distributions of project/users w.r.t. number of users/projects (axes oX) and the coefficient of variation $\mathrm{cv}(\{\mathtt{TpM}(\mathcal{A}_{u,p})\}_{u\in\mathcal{U}_p/p\in\mathcal{P}_u})$. M=completion time TpM (top row) or hourly earnings HE (bottom row). See Sec. 7.

Similarly, in the case (2), we take $X_u = \{\mathtt{TpM}(\mathcal{A}_{u,p})\}_{p\in\mathcal{P}_u}$ (or $X_u = \{\mathtt{HE}(\mathcal{A}_{u,p})\}_{p\in\mathcal{P}_u}$) and plot the joint distribution of users $\mathcal{U}$[12] w.r.t. the statistics $\mathrm{cv}(X_u)$ and the number of projects in which user participated $|\mathcal{P}_u|$ in top (or bottom, resp.) of Fig.2(f). In both Fig.2(e) and Fig.2(f), we see that $\mathrm{cv}(X)$ notably larger than 0 and mostly larger than 0.5. Therefore, *we conclude that the projects and users in the studied crowdsourcing marketplace are highly heterogeneous what makes the problem of prediction challenging*, especially in the case when we have low information on a user-project pair.

"Learning effect". When a toloker starts to perform tasks from a new project for her, there may exist a so-called "learning effect". The performance of the user in the initial period (in few first tasks from the project) may differ from her performance when she gets mastered tasks of this project. In order to empirically verify this hypothesis, we make the following analysis. For a user-project pair (u, p), let the assignments $\mathcal{A}_{u,p}$ be sorted in ascending order of their start timestamps, then each task $a\in\mathcal{A}_{u,p}$ has its ordinal number i_a in this sorting. In Fig.3(a), we plot the joint distribution of tasks $a\in\mathcal{A}$[13] w.r.t. the ordinal number i_a in their user-project pairs and the relative time per microtask $\mathtt{TpM}(\{a\})/\mathtt{TpM}(\mathcal{A}_{u_a,p_a})$. In Fig.3(b), we plot similar heatmaps for tasks $\mathcal{A}_p$ of 4 most popular projects. In all 5 heatmaps, we observe the "learning effect": the completion time of few first assignments (at least 1-st, 2-nd, and 3-rd) is notably larger than the completion time of assignments away from the start. Hence, *we conclude that "learning effect" exists in the crowd platform.*

8 PREDICTION METHODS

Objects. As stated in Sec. 5, the user visits to the main page (where a user selects a project to execute) are the events (objects) when we need to have a predicted value of completion time or hourly earnings. Exactly these events are in our dataset, see Sec. 6. For the need to calculate some features on user behavior occurred previously to a visit (see below), we consider only the visits whose timestamp is later than 14 days since the first time moment of a period in which we have collected data[14] (i.e., the starting timestamp of dataset).

Targets. Let us use the notations introduced in Sec. 7. For a given visit v of a user u^v, let p^v be the selected project and a_1^v be the first task assigned to the user u^v in this project p^v after his visit v. In order to define the performance measures (i.e., completion time and hourly earnings) of the user u^v on the project p^v at this visit, we use the formulas in Eq. (1)[15]. However, it is not enough to get the values, because we need to specify the subset $\mathcal{A}'$ of assignments over which performance measures are calculated. In this paper, we consider two approaches to select this subset[16]. First, it is natural to take all tasks performed by the user between the visit v and the next one v^{next} (or the end of dataset if the user has no more visits after v). We refer to the set of all these tasks as the session $S(v) \subset \mathcal{A}$ after the visit v. Second, it is seen (from our dataset) that a number of tasks in a session may be quite small (its median is 3). Hence, a low number of tasks may result in higher noise of target. To get more statistical data for the formulas Eq. (1), one can use information on tasks from the project p^v performed later than the user's next visit v^{next} (e.g., she may select the same project, or return to it after another one). So, the second approach is to use all tasks performed by the user u^v on the project p^v in the d-length time window that starts from the visit v (we refer to these tasks as $W_d(v)$). Summarizing, we study the following targets for a visit v:

$$\begin{aligned} \mathbf{TpM}_{\mathrm{s}}(v) &:= \mathtt{TpM}(S(v)), \quad \mathbf{HE}_{\mathrm{s}}(v) := \mathtt{HE}(S(v)), \\ \mathbf{TpM}_{\mathrm{w}}(v) &:= \mathtt{TpM}(W_d(v)), \quad \mathbf{HE}_{\mathrm{w}}(v) := \mathtt{HE}(W_d(v)). \end{aligned} \tag{2}$$

Note that, when we predict time, errors in magnitudes are much more important than errors in absolute values[17]. For this purpose, time is usually logarithnically transformed to significantly improve models [10]. We also $\log_{10}$–transform our completion time targets and denote them by $\mathbf{lgTpM}_{\mathrm{s}}(v)$ and $\mathbf{lgTpM}_{\mathrm{w}}(v)$.

Loss function and performance metric. We consider our prediction problem as a regression, and thus use the standard

[12] We left only users u s.t. $|\mathcal{P}_u| \geq 5$ in the analysis.
[13] We left only tasks a s.t. $|\mathcal{A}_{u_a,p_a}| \geq 20$ in the analysis.
[14] This 14-day filtering is not a limitation of our approach. We do it due to absence of history prior to Sept. 2018 in our dataset. In production, we can use the whole history.
[15] Note that we calculate completion time per microtask, because the number of microtasks may vary between tasks in a project.
[16] In fact, there is a large variety of approaches to select the subset $\mathcal{A}'$ (e.g., to filter out firs/last tasks in a session, to remove outliers, to tune the length of a window, etc.), but we provide only the two most representative ones due to the space constraints
[17] For instance, when the true time is 1 min, and we predict 2 min (the absolute error = 1 min) it is more critical (more notable) for a user than the same absolute error, but when the true time is 60 minutes, and we predict 61 minutes.

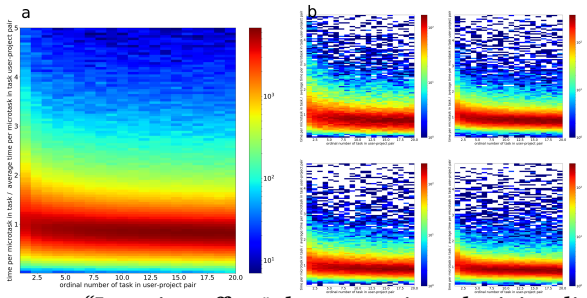

Figure 3: "Learning effect" demonstration: the joint distributions of a task a w.r.t. the ordinal number i_a in its user-project pair (u_a, p_a) and the relative time per microtask $\texttt{TpM}(\{a\})/\texttt{TpM}(\mathcal{A}_{u_a,p_a})$. (a) heatmap for tasks of all projects, (b) heatmap for tasks of each of 4 most popular projects.

Root Mean Square Error (RMSE) both as the loss function to learn our prediction models (on a training set), and as the main performance/quality metric for them (on a test set).

Slices: study of "cold start" and "learning effect". First, let a visit v be referred to as a "*cold-start*" one when it is the first visit for the user-project pair (u^v, p^v) in a considered dataset. As we discuss in Sec. 5, prediction on "cold-start" visits is much more important and challenging than on non-"cold-start" ones. Second, as we observe in Sec. 7, the "learning effect" presents in the performance of a user on first tasks after a "cold-start" visit. In fact, a user is interested in information on her performance (hourly earnings, especially) on a project that is measured in the scenario: she is engaged by the project, gets mastered its tasks, and performs a lot of them. Hence, we will also measure prediction quality (RMSE) on the subset of the "cold-start" visits such that each visit v has at least $k \in \mathbb{N}$ future tasks in the d-length window, i.e., $|W_d(v)| \geq k$. We refer to this subset as the "learning effect" free visits (the *CS-LEF slice*). We will also learn some models on a similar slice of train data. Analysis in Sec. 7 suggests that taking $k = 10$ is sufficient to notably reduce the main part of the "learning effect".

Simple prediction methods. First, we propose a couple of constant-like basic models. The first model (Baseline-C1) always outputs the average target value over the whole train dataset. The second one (Baseline-C2) is piecewise constant function in the sense that it outputs the average target value over one of 5 types of a visit in a train dataset: the average target over all non-"cold-start" visits and the one over "cold-start" visits where [project]/[user] [has]/[does not have] visits before (4 combinations), see App.A.5 to reproduce it. Second, we build more advanced model (Baseline-P) which, however, still does not use machine learning at all. For a given visit v, this model outputs the similar to the target statistics (TpM, log TpM, or HE), but calculated over several tasks submitted before the visit v. Namely, we choose no more than 20 last tasks in the following sequence until we are able to get at least one task: seek for tasks of the given user-project pair (u^v, p^v) in the train dataset; only if there is no any, seek for tasks for the project p^v; then, tasks of the toloker u^v; finally, tasks of any project and user[18].

Feature-based prediction method. As we see in Sec. 7, users and projects have high heterogeneity w.r.t. completion time and hourly earnings, what should result in a low quality of the constant baselines and, in the case of "cold-start" visits, in a low one of the latter baseline. Therefore, we build a model based on features. We use CatBoost, one of the most popular implementation of the state-of-the-art gradient boosting decision tree method [27].

First, 5 different statistics about completion time and earnings have been calculated for 3 sets of assignments (with the same project, with the same user, and with the same project-user pair) over rolling windows of 4 different lengths (20 & 100 assignments and 3600 & 86400 seconds): overall 96 "rolling window" features are used. Second, we use user device category, OS family, seconds since registration, session start time, and cold start type. Third, in order to describe projects, we use parameters of the instruction text (its length, readability, and language) and of the task's HTML inputs/outputs (the number of checkboxes, radio buttons, text inputs, images, etc), as well as project creation timestamp and average cost of the assignment. So, we get 130 features in total. Finally, outputs of our baselines are also used as features (stacking of models). All details are given in Appendix A.1.

9 EXPERIMENTAL EVALUATION

Experimental setup. In order to evaluate the performance of proposed prediction models in Sec. 8, we consider the following experimental pipeline that utilizes our dataset from Sec. 6. First, in order to measure the statistical significance of reported quality improvements, we utilize a technique similar to 10-fold cross-validation one to split the data into train and test sets (but with a slight modification as our data are time sensitive). We split the data as follows:

(1) Let $\mathcal{U}$ be a set of all users. We break it randomly into 10 non-intersecting parts of nearly equal size: $\mathcal{U} = \bigsqcup_{i=1}^{10} \mathcal{U}_i$.
(2) Split the whole available 3-month period of time into 2 parts: $T = T_{[1,2]} \sqcup T_{[3]}$, where $T_{[1,2]}$ denotes a period of the first and second months, $T_{[3]}$ is the third month.
(3) Let $\mathcal{W}_i = \mathcal{U} \setminus \mathcal{U}_i$ be all users except the ones from $\mathcal{U}_i$.
(4) Set $\{(S^i_{train}, S^i_{test})\}_{i=1}^{10}$ as 10 user-time splits, where $S^i_{train} = \mathcal{W}_i \times T_{[1,2]}$ and $S^i_{test} = \mathcal{U}_i \times T_{[3]}$.
(5) Each user-time split $(S^i_{train}, S^i_{test})$ uniquely defines the split $(D^i_{train}, D^i_{test})$ of all data in our dataset from Sec. 6.

Thus, a decomposition $(D^i_{train}, D^i_{test})$ keeps all the events isolated between the i-th train and i-th test data partitions (both in user and time dimensions[19]), while the sequences of user actions are kept solid. All the features are calculated within these datasets. Hence, we are sure that the features from the train set do not use any information from the test set as if it does not exist at all (and vice versa). Note that by the construction of dataset partitions (and the dataset itself), there are cases where any user from the beginning of a test (train) set (if ordered by the timestamp) would be considered as a newcomer since we do not know her history which may be located in the period $T_{[1,2]}$ (before the period $T_{[1,2]}$, resp.). In order to equalize visits (our prediction objects, see Sec. 8) in the amount of available information, we (1) in both training and test sets, consider only visits happened after 14 days since the beginning of a set; (2) all features that exploit history before a visit use only events that happen within the 14-day window prior to the visit.

[18] This logic is similar to SQL function COALESCE.

[19] In particular, users from a training set D^i_{train} do not present in the corresponding test set D^i_{test}.

Table 2: RMSE performance of predictors: % are w.r.t. the model in the previous row; stat. signif. is in boldface ($\alpha = 0.05$).

	(A) @ visits of all types			(B) @ "Cold-start" visits with reduced "learning effect" (CS-LEF slice)		
Model ↓ / Target →	$\mathbf{TpM}_s$	$\mathbf{lgTpM}_s$	$\mathbf{HE}_s$	$\mathbf{TpM}_w$	$\mathbf{lgTpM}_w$	$\mathbf{HE}_w$
Baseline-C1	174.99	0.541	5.84	50.81	0.558	7.92
Baseline-C2	**174.85 (-0.08%)**	**0.542 (+0.09%)**	**1.52 (-73.94%)**	**49.83 (-1.94%)**	**0.553 (-0.85%)**	**3.51 (-55.70%)**
Baseline-P	172.46 (-1.37%)	**0.155 (-71.31%)**	**1.17 (-22.78%)**	**38.65 (-22.43%)**	**0.154 (-72.23%)**	**0.80 (-77.58%)**
Catboost	**113.87 (-33.98%)**	**0.115 (-25.92%)**	**0.93 (-18.85%)**	**29.50 (-23.66%)**	**0.104 (-32.09%)**	**0.60 (-25.05%)**

Table 3: RMSE performance of predictors with feature ablation: % are w.r.t. the model trained on all features. Statistical significant differences w.r.t. the model trained on all features are highlighted in boldface ($\alpha = 0.05$).

	@ visits of all types			@ "Cold-start" visits with reduced "learning effect" (CS-LEF slice)		
Model ↓ / Target →	$\mathbf{TpM}_s$	$\mathbf{lgTpM}_s$	$\mathbf{HE}_s$	$\mathbf{TpM}_w$	$\mathbf{lgTpM}_w$	$\mathbf{HE}_w$
Catboost	113.87	0.115	0.93	29.50	0.104	0.60
Catboost without user features	**122.91 (+7.94%)**	**0.127 (+10.43%)**	**1.11 (+19.35%)**	30.58 (+3.66%)	**0.110 (+5.77%)**	**0.65 (+8.33%)**
Catboost without project features	**159.83 (+40.36%)**	**0.499 (+333.91%)**	**1.05 (+12.9%)**	**49.41 (+67.49%)**	**0.610 (+486.54%)**	**0.71 (+18.33%)**

So, given a train-test decomposition $(D^i_{train}, D^i_{test})$, we calculate baselines and learn the CatBoost models[20] on D^i_{train}, while calculate the quality metric (RMSE) of resulting models on the test set D^i_{test}. The resulting quality scores for $i = 1, .., 10$ are averaged to be reported further in this section and are used in *the paired two-sample t-test* to measure the significance level of the obtained results. We consider the 3 session targets $\mathbf{TpM}_s$, $\mathbf{lgTpM}_s$, $\mathbf{HE}_s$ and the 3 window targets $\mathbf{TpM}_w$, $\mathbf{lgTpM}_w$, $\mathbf{HE}_w$ (with $d = 24$ hours), see Sec. 8. For each target, we apply Baseline-C1, Baseline-C2, Baseline-P, and the CatBoost model over all available features. CatBoost is learned with 1000 iterations and 1000 trees of depth 10[21].

Results on visits of all types. In Table 2.A, results of our models are reported w.r.t. RMSE calculated over all considered visits in a test dataset. We see that, first, according to our expectations, the increasing complexity of the models positively affects the accuracy of predictions. Second, *the most advanced CatBoost model outperforms the others with respect to all targets*: it statistically significantly improves RMSE by up to 34% for prediction of task completion time ($\mathbf{TpM}_s$); by up to 25.9% for prediction of logarithmic task completion time ($\mathbf{lgTpM}_s$); and by up to 18.8% for hourly earnings prediction ($\mathbf{HE}_s$). We also see the strength of baselines that improve prediction quality over constant Baseline-C1 by a huge margin. The reason of baseline strength here is that non-"cold-start" visits comprise up to 90% of their total number in our dataset.

"Cold-start" visits with reduced "learning effect". As we argue in Sec. 5, 7, and 8, we are primarily interested in obtaining good predictions in the CS-LEF slice, which contains all "cold-start" visits with at least $k = 10$ assignments after. For this case, we train CatBoost model on this slice of a set D^i_{train}, measure the quality (RMSE) of our models on the CS-LEF slice of the set D^i_{test}, and report the aggregated results over our 10 folds in Table 2.B. First, in the CS-LEF slice, *the CatBoost model statistically significantly outperforms all the baselines as well: it provides RMSE reduction by* 23.7% *for task completion time* $\mathbf{TpM}_w$*; by* 32.1% *for its logarithm* $\mathbf{lgTpM}_w$*; and by* 25.1% *for hourly earnings* $\mathbf{HE}_w$. Second, note that the relative improvement of the CatBoost model w.r.t. the constant Baseline-C1 are notably larger than the ones in the case of all type visits (Table 2.A vs. B). Third, from the absolute RMSE values, we conclude that *the best method (CatBoost) allows us to achieve (a) the average error of a half of a minute for the completion time prediction and (b) the average error of* \$0.6 *for hourly earnings prediction.*

Feature importance and ablation. According to the feature importance (fstr) outputted by CatBoost trained on the CS-LEF slice for the target $\mathbf{HE}_w$: 6 of 10-top features are dynamically calculated window-based hourly earnings over some history before a visit, instruction readability metric is on the 2-nd position, dynamically calculated window-based completion time per microtask for user is on the 7th position, the length of project specification is on the 8th position, and baseline-P (as a stacked model) is on the 9th position.

We also evaluate the effect of features related to (a) users and (b) projects on RMSE quality by means of ablation of corresponding feature groups while training CatBoost. Namely, we train CatBoost on reduced sets of features: (a) the set of features without all user-related ones and (b) the set of features without all project-related ones. These feature sets are listed in App. A.2. We compare these reduced models w.r.t. the performance of the original Catboost model in Table 3 both for visits of all types and for the "cold-start" slice. We see that both user- and project-related features (both dynamically calculated and static ones) are found significantly important for prediction quality. For instance, for prediction of hourly earnings $\mathbf{HE}_w$ (on CS-LEF slice), ablation of the user features raises RMSE by 8.33% and ablation of the project ones raises RMSE by 18.33%. These results serve as an additional confirmation of our analysis from Sec. 7, where we found high heterogeneity of users and projects. Overall, *we conclude that features on task instruction, inputs of the task, window features on past history, user expertise, and past quality are importantly valuable to provide best prediction quality.*

"Cross-target" learning. *We claim that a model learned to predict completion time cannot be effectively used to predict hourly earnings* (as was done in [30]). In order to demonstrate this, we predict the value of one of our hourly earnings targets ($\mathbf{HE}_s$ or $\mathbf{HE}_w$) by combining the reward for a microtask and the output of the CatBoost model trained to fit one of the corresponding completion time targets ($\mathbf{TpM}_s$, $\mathbf{TpM}_w$, $\mathbf{lgTpM}_s$, and $\mathbf{lgTpM}_w$). To do so, we used the feature 'project_window_20_assignments_avg_cost_to_microtask' (which we denote from here on by `reward`). Note that `reward` is available to the platform at the moment when we need to show

[20] When we learn the CatBoost models, we also split D^i_{train} in the same way to obtain a train part and validation part. We emphasize that these parts are used to fit hyperparameters of the model; we do not use the test set D^i_{test} for this purpose.
[21] Python package v.0.12.2 is used with loss_function='RMSE' & use_best_model=True.

Table 4: Comparison of models trained to fit different targets, but used to predict hourly earnings (see Sec. 9 on how these models are used): all diffs are stat. signif. ($\alpha = 0.05$).

@ visits of all types		@ "Cold-start" visits (CS-LEF slice)	
Catboost trained for ↓	RMSE for $\mathbf{HE}_s$	Catboost trained for ↓	RMSE for $\mathbf{HE}_w$
$\mathbf{HE}_s$	0.93	$\mathbf{HE}_w$	0.60
$\mathbf{TpM}_s$	**3.71 (+299%)**	$\mathbf{TpM}_w$	**1.37 (+128%)**
$\mathbf{lgTpM}_s$	**1.36 (+46%)**	$\mathbf{lgTpM}_w$	**3.36 (+460%)**

hourly earnings to a user. Hence, combining estimation of completion time and `reward`, one can obtain estimator (predictor) of hourly earnings. Namely, if `time` is an estimator of $\mathbf{TpM}_s$ or $\mathbf{TpM}_w$ (obtained from learning CatBoost to fit $\mathbf{TpM}_s$ or $\mathbf{TpM}_w$, resp.), then we use the following equation to get a predictor of hourly earnings (denoted by `earnings`): $\texttt{earnings} = \texttt{reward} \cdot 3600/\texttt{time}$. Similarly, given `logtime` (an estimator of $\mathbf{lgTpM}_s$ or $\mathbf{lgTpM}_w$), we use the formula: $\texttt{earnings} = \texttt{reward} \cdot 3600/10^{\texttt{logtime}}$.

So, we compare these estimators (predictors) of hourly earnings with the CatBoost model learned to fit the targets $\mathbf{HE}_s$ and $\mathbf{HE}_w$, resp. The results are in Table 4. We see that the model fitted to completion time $\mathbf{TpM}_w$ and used as described above has prediction quality of hourly earnings $\mathbf{HE}_w$ statistically significantly worser by 128% in terms of RMSE than the CatBoost model directly learned to predict $\mathbf{HE}_w$. Similarly, the model fitted to logarithmic time $\mathbf{lgTpM}_w$ statistically significantly raises the RMSE of predicting $\mathbf{HE}_w$ by 460%; the one fitted to $\mathbf{TpM}_s$ stat. sign. raises RMSE of $\mathbf{HE}_s$ by 299%; while the model fitted to $\mathbf{lgTpM}_s$ statistically significantly raises RMSE of $\mathbf{HE}_s$ by 46.2%.

10 CONCLUSIONS

We studied the problem of predicting future hourly earnings and task completion time of a crowdsourcing platform user who visits the list of available tasks and wants to select one of them to execute. We addressed this problem on real crowd tasks completed in one of the largest crowdsourcing marketplaces. First, our survey and an A/B test on real users showed that monetary incentives dominate among users; and user engagement significantly improves when performers can use expected hourly earnings in the platform UI. Second, we provided user behavior analysis to show that the prediction problem is challenging due to high heterogeneity of users/projects and the presence of the "learning effect". Finally, we predicted user performance (hourly earnings and completion time) by a gradient boosting decision tree model that outperformed a naive baseline (which is based solely on historical performance of users) by up to 25% for hourly earnings (and up to 32% completion time) in the most important and difficult case when user selects a new task. We published the dataset that we used in our experimentation for reproducibility of our results and to provide the research community with rich crowdsourcing data for future studies.

REFERENCES

[1] R. Budylin, A. Drutsa, G. Gusev, P. Serdyukov, and I. Yashkov. 2018. Online evaluation for effective web service development. *Tutorial at KDD'2018.*

[2] Ricardo Buettner. 2015. A systematic literature review of crowdsourcing research from a human resource management perspective. In *HICSS*.

[3] Chris Callison-Burch. 2014. Crowd-Workers: Aggregating Information Across Turkers to Help Them Find Higher Paying Work. In *HCOMP*.

[4] Justin Cheng, Jaime Teevan, and Michael S. Bernstein. 2015. Measuring Crowdsourcing Effort with Error-Time Curves. In *CHI*.

[5] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G Ipeirotis, and Philippe Cudré-Mauroux. 2015. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In *WWW*. 238–247.

[6] A. Drutsa, V. Farafonova, V. Fedorova, O. Megorskaya, E. Zerminova, and O. Zhilinskaya. 2019. Practice of Efficient Data Collection via Crowdsourcing at Large-Scale. *Tutorial at KDD'2019.*

[7] A. Drutsa, V. Fedorova, D. Ustalov, O. Megorskaya, E. Zerminova, and D. Baidakova. 2020. Practice of Efficient Data Collection via Crowdsourcing: Aggregation, Incremental Relabelling, and Pricing. In *WSDM'2020.* 873–876.

[8] Alexey Drutsa, Gleb Gusev, and Pavel Serdyukov. 2017. Using the Delay in a Treatment Effect to Improve Sensitivity and Preserve Directionality of Engagement Metrics in A/B Experiments. In *WWW'2017.*

[9] A. Drutsa, D. Rogachevsky, O. Megorskaya, A. Slesarev, E. Zerminova, D. Baidakova, A. Rykov, and A. Golomedov. 2020. Efficient Data Annotation for Self-Driving Cars via Crowdsourcing on a Large-Scale. In *CVPR'2020.*

[10] Alexey Drutsa, Anna Ufliand, and Gleb Gusev. 2015. Practical Aspects of Sensitivity in Online Experimentation with User Engagement Metrics. In *CIKM'2015.*

[11] A. Drutsa, D. Ustalov, E. Zerminova, V. Fedorova, O. Megorskaya, and D. Baidakova. 2020. Crowdsourcing Practice for Efficient Data Labeling: Aggregation, Incremental Relabeling, and Pricing. In *SIGMOD'2020.* 2623–2627.

[12] David S Evans and Richard Schmalensee. 2016. *Matchmakers: the new economics of multisided platforms.* Harvard Business Review Press.

[13] Siamak Faradani, Björn Hartmann, and Panagiotis G Ipeirotis. 2011. What's the Right Price? Pricing Tasks for Finishing on Time. *Human comp.* 11 (2011), 11.

[14] Tanya Goyal, Tyler McDonnell, Mucahid Kutlu, Tamer Elsayed, and Matthew Lease. 2018. Your Behavior Signals Your Reliability: Modeling Crowd Behavioral Traces to Ensure Quality Relevance Annotations. In *HCOMP*.

[15] B.V Hanrahan, D. Martin, J. Willamowski, and J.M Carroll. 2018. Investigating the Amazon Mechanical Turk Market Through Tool Design. *CSCW* 27, 3-6 (2018).

[16] Benjamin V Hanrahan, Jutta K Willamowski, Saiganesh Swaminathan, and David B Martin. 2015. TurkBench: Rendering the market for Turkers. In *HFCS*.

[17] Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P Bigham. 2018. A Data-Driven Analysis of Workers' Earnings on Amazon Mechanical Turk. In *CHI*.

[18] Simo Hosio, Jorge Goncalves, Vili Lehdonvirta, Denzil Ferreira, and Vassilis Kostakos. 2014. Situated crowdsourcing using a market model. In *UIST*. ACM.

[19] Panagiotis G Ipeirotis. 2010. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students* 17, 2 (2010), 16–21.

[20] Lilly C Irani and M Silberman. 2013. Turkopticon: Interrupting worker invisibility in amazon mechanical turk. In *CHI*.

[21] T. Kaplan, S. Saito, K. Hara, and J. P Bigham. 2018. Striving to earn more: a survey of work strategies and tool use among crowd workers. In *HCOMP'2018*.

[22] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M Henne. 2009. Controlled experiments on the web: survey and practical guide. *DMKD* 18, 1 (2009).

[23] Pavel Kucherbaev, Florian Daniel, Stefano Tranquillini, and Maurizio Marchese. 2016. ReLauncher: crowdsourcing micro-tasks runtime controller. In *CSCWSC*.

[24] Adam Marcus, Aditya Parameswaran, et al. 2015. Crowdsourced data management: Industry and academic perspectives. *FTD* 6, 1-2 (2015), 1–161.

[25] David Martin, Benjamin V Hanrahan, Jacki O'Neill, and Neha Gupta. 2014. Being a turker. In *CSCWSC*.

[26] P. Minder, S. Seuken, A. Bernstein, and M. Zollinger. 2012. Crowdmanager-combinatorial allocation and pricing of crowdsourcing tasks with time constraints. In *Workshop on Social Computing and User Generated Content in ACM EC*.

[27] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*. 6638–6648.

[28] John Prpić, Araz Taeihagh, and James Melton. 2015. The fundamentals of policy crowdsourcing. *Policy & Internet* 7, 3 (2015), 340–361.

[29] Mejdl Safran and Dunren Che. 2018. Efficient Learning-Based Recommendation Algorithms for Top-N Tasks and Top-N Workers in Large-Scale Crowdsourcing Systems. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 2.

[30] Susumu Saito, Chun-Wei Chiang, Saiph Savage, Teppei Nakano, Tetsunori Kobayashi, and Jeffrey P Bigham. 2019. TurkScanner: Predicting the Hourly Wage of Microtasks. In *The World Wide Web Conference*. ACM, 3187–3193.

[31] Thimo Schulze, Simone Krug, and Martin Schader. 2012. WorkersâĂŹ task choice in crowdsourcing and human computation markets. (2012).

[32] Jing Wang, Siamak Faridani, and Panagiotis Ipeirotis. 2011. Estimating the completion time of crowdsourced tasks using survival analysis models. *Crowdsourcing for search and data mining (CSDM 2011)* 31 (2011).

[33] Jing Wang and Panagiotis Ipeirotis. 2013. A framework for quality assurance in crowdsourcing. (2013).

[34] Chaolun Xia and Shan Muthukrishnan. 2017. Revenue-Maximizing Stable Pricing in Online Labor Markets. (2017).

[35] Yandex. Since 2014. *Yandex.Toloka.* http://toloka.ai

[36] Yandex.Toloka. 2020. *Dataset "Toloka Users & Tasks".* https://toloka.ai/datasets

[37] Y. Zhang, H. Qin, B. Li, J. Wang, S. Lee, and Zh. Huang. 2018. Truthful mechanism for crowdsourcing task assignment. *Tsinghua Science and Technology* 23, 6 (2018).

A APPENDIX

A.1 Features

In order to make our results reproducible, we provide a comprehensive description of features used in our study. We use Baseline-C1, Baseline-C2, Baseline-P for each target (as a stacked model). We also use session start time, seconds since user registration, user device category, OS family, and cold start type (one of [no_cold_start, new_project, new_user, new_user_project, new_user_new_project]). To describe projects we use all the project features described in README.md of our dataset [36] (except for the project id) and cost which is task price averaged over all tasks in the project.

Window features. By window features we mean those which are calculated through a lagging window with respect to a current visit, similar to window $W_d(v)$ defined in Sec. 8. These features have the following pattern in their names: <prefix>_window_<window size>_<window units>_<statistics>.

- <prefix>: one of [user_project, project, user]. Denotes what type of tasks are taken into the window. Similar to the definition of Baseline-P model. See Sec. 8.
- <window size>: window width.
- <window units>: one of [seconds, assignments]. Units of window width measurment. In the study we used 20 and 100 assignment windows, and 3600, 86400 seconds.
- <statistics>. Denotes the statistic which is calculated over a given window. It is one of:
  - avg_time_to_microtask,
  - avg_cost_to_microtask,
  - avg_log_time_to_microtask,
  - gs_accuracy (quality on golden sets),
  - gs_total (golden set quantity),
  - hourly_wage,
  - total_assignments,
  - total_microtasks.

Also we had one special window feature calculated for <prefix>=project, <window size>=1000, <window units>=assignments, <statistics>was equal to avg_time_to_microtask for $\mathbf{TpM}_s$, avg_log_time_to_microtask for $\mathbf{lgTpM}_s$, hourly_wage for $\mathbf{HE}_s$ if there were 5 or more assignments in the window and similar for $\mathbf{TpM}_w$, $\mathbf{lgTpM}_w$, $\mathbf{HE}_w$, but the average was taken only over session in CS-LEF slice and the feature was calculated regardless of the number of tasks in the window.

A.2 Feature ablation

We trained CatBoost on reduced sets of features. Namely, we trained on two sets: (a) the set of features without all user-related ones and (b) the set of features without all project-related ones.

Namely, **for the case (a) 'without user-related features'** we eliminated all features that exploit information about the user u^v of a visit v (but aggregated information from all users is left). So, in this case, we train the Catboost model on the following features:

- cost
- project_created_timestamp
- project_has_audio
- project_has_button
- project_has_buttonClicked_input
- project_has_checkbox_input
- project_has_externalHtml
- project_has_fileAudio_input
- project_has_fileImg_input
- project_has_fileVideo_input
- project_has_file_input
- project_has_iframe
- project_has_image
- project_has_imageAnnotation_input
- project_has_radio_input
- project_has_sbs
- project_has_select_input
- project_has_sourcesRecorder_input
- project_has_string_input
- project_has_suggest_input
- project_has_textarea_input
- project_has_video
- project_instruction_FK
- project_instruction_len
- project_instruction_wordCount
- project_required_fields
- project_spec_length
- cold_start_type
- project_instruction_language
- baseline1_for_current_target
- baseline2_for_current_target
- project_window_XX_avg_cost_to_microtask
- project_window_XX_avg_log_time_to_microtask
- project_window_XX_avg_time_to_microtask
- project_window_XX_gs_accuracy
- project_window_XX_gs_total
- project_window_XX_hourly_wage
- project_window_XX_total_assignments
- project_window_XX_total_microtasks,

where "XX" ∈ { 20_assignments, 100_assignments, 3600_seconds, and 86400_seconds }.

For the case (b) 'without projects-related features' we eliminated all features that exploit information about the project p^v of a visit v (but aggregated information from all projects is left). So, in this case, we train the Catboost model on the following features:

- seconds_since_registration
- session_start_time
- cold_start_type
- device_category
- os_family
- baseline1_for_current_target
- baseline2_for_current_target
- user_window_XX_avg_cost_to_microtask
- user_window_XX_avg_log_time_to_microtask
- user_window_XX_avg_time_to_microtask
- user_window_XX_gs_accuracy
- user_window_XX_gs_total
- user_window_XX_hourly_wage
- user_window_XX_total_assignments
- user_window_XX_total_microtasks,

where "XX" ∈ { 20_assignments, 100_assignments, 3600_seconds, and 86400_seconds }.

A.3 Learning parameters

To fit the Catboost (python package v.0.12.2) model we used the following parameters:

- depth=10
- loss_function='RMSE'
- use_best_model=True
- iterations=1000
- learning_rate=0.03

A.4 "Cross-target" learning

We used Catboost models trained for $\mathbf{TpM}_s$, $\mathbf{lgTpM}_s$, $\mathbf{TpM}_w$, $\mathbf{lgTpM}_w$ to predict $\mathbf{HE}_s$ and $\mathbf{HE}_w$, to make this prediction we used the feature project_window_20_assignments_avg_cost_to_microtask (which we denote from here on by `price`). Note that `price` is available to the platform at the moment when we need to show hourly earnings to a user. Hence, combining estimation of completion time and `price`, one can obtain estimator (predictor) of hourly earnings.

Namely, for instance, if `time` is an estimator of $\mathbf{TpM}_{w/s}$ (obtained from learning CatBoost to fit $\mathbf{TpM}_{w/s}$), then we use the following equation to get a predictor of hourly earnings (denoted by `earnings`):

$$\texttt{earnings} = \texttt{price} \cdot 3600/\texttt{time}.$$

Similarly, given `logtime` (an estimator of $\mathbf{lgTpM}_{w/s}$), we use the formula:

$$\texttt{earnings} = \texttt{price} \cdot 3600/10^{\texttt{logtime}}.$$

A.5 Formal description of Baseline-C2

Baseline-C2 is piecewise constant function in the sense that it outputs the average target value over one of 5 types of a visit in a train dataset:

- the average target over all non-"cold-start" visits;
- the average target over all "cold-start" visits $\{v\}$ s.t. their projects p^v do not have visits before[22] and their users u^v do not have visits before[23] as well;
- the average target over all "cold-start" visits $\{v\}$ s.t. their projects p^v do not have visits before, but their users u^v have visits before;
- the average target over all "cold-start" visits $\{v\}$ s.t. their projects p^v have visits before, but their users u^v do not have visits before;
- the average target over all "cold-start" visits $\{v\}$ s.t. their projects p^v have visits before and their users u^v have visits before as well.

[22] This means that the visit v is the first one (in dataset), where the project p^v has been selected by a user.
[23] This means that the visit v is the first one (in dataset), where the user u^v has selected a project.