Санкт-Петербургский государственный университет

Сергеев Егор Федорович

Выпускная квалификационная работа

Оценка потенциального количества пользователей для различной функциональности приложения

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2017 «Программная инженерия»*

Научный руководитель:
доцент кафедры СП, к.ф.-м.н., Д.В. Луцив

Рецензент:
Программист ООО «Интеллиджей Лабс» Е.А. Ушаков

Санкт-Петербург
2022

Saint Petersburg State University

Egor Sergeev

Bachelor's Thesis

Potential users number evaluation for application features

Education level: bachelor

Speciality *09.03.04 «Software Engineering»*

Programme *CB.5080.2017 «Software Engineering»*

Scientific supervisor:
docent, C.Sc. D.V. Luciv

Reviewer:
Software Developer at "IntelliJ Labs" E.A. Ushakov

Saint Petersburg
2022

Оглавление

1. Введение

По ходу разработки и сопровождения программных приложений количество требований и идей для улучшения продукта стремительно растет. Накапливающиеся запросы пользователей и компаний-клиентов, желание не уступать конкурентам в качестве продукта — все это и многое другое формирует длинный список задач для разработчиков системы. Зачастую в проекте не хватает временных и денежных ресурсов для воплощения всех имеющихся идей и запросов. В связи с этим остро встает вопрос приоритезации — выбора наиболее важных запросов на изменения, которые сильнее всего способствуют улучшению пользовательских характеристик продукта и достижению поставленных бизнес-показателей [40].

Наиболее распространенные подходы к приоритезации идей, такие как RICE [22] или модель Кано [19], основываются на оценке потенциальной ценности, которую получат пользователи продукта после реализации этих запросов, и сравнении этих оценок для имеющихся потенциальных улучшений с учетом затрат на их реализацию. Проблема таких методик заключается в субъективности оценки — у нескольких членов команды мнения насчет значимости насчет одного и того же запроса на изменения могут отличаться. Это приводит к необходимости создания объективных критериев, которые позволят оперировать фактами, а не субъективными мнениями.

Такой запрос существует и у команды IntelliJ IDEA — известной многоязыковой интегрированной среды разработки (IDE). Ключевой особенностью этого продукта является большое количество доступной пользователю функциональности, которая может быть полезна в разнообразных сценариях. Как следствие, многие пользователи не знают о существовании той или иной полезной функциональности продукта. Таким образом, задача приоритезации особенно актуальна для развития уже существующей функциональности и привлечения внимания пользователей к ней за счет улучшения опыта взаимодействия с IDE.

В данной работе представлена система оценки потенциального ко-

личества пользователей для различной функциональности приложения на примере IntelliJ IDEA. Сравнивая такую оценку для разных подсистем и возможностей (features) продукта, можно выбрать те, улучшение которых сможет дать наибольший прирост числа регулярных пользователей. Выбор данной метрики обусловлен бизнес-целями продукта: чем больше людей смогли получить ценность от использования определенной функциональности и начали ее регулярно использовать в своей работе, тем лучше IDE справляется со своей задачей — сократить время пользователя, которое он тратит на выполнение своих задач. Такой подход позволяет команде разработчиков получить агрегированную информацию по рассматриваемой функциональности на основании статистики использования продукта и использовать ее в качестве объективного критерия в процессе приоритезации.

2. Постановка задачи

Целью работы является разработка системы оценки потенциального количества пользователей для различной функциональности приложения на примере IntelliJ IDEA. Такая система позволит команде разработки получить объективный критерий для сравнения эффективности работы над различными подсистемами и функциональностью в рамках приоритезации и планирования задач. Для её выполнения были поставлены следующие задачи.

1. Изучить существующие методологии ранжирования функциональности и идей, работа над которыми может привлечь наибольшее число пользователей, и подходы к предсказанию потенциального количества пользователей для функциональности приложения.

2. Сформировать коллекцию данных, характеризующую поведение пользователей в IntelliJ IDEA.

3. Реализовать систему предсказания потенциального количества пользователей для различной функциональности IDE.

4. Провести оценку точности предсказаний и проанализировать полученные результаты.

5. Разработать и апробировать внутрикорпоративный сервис для просмотра результатов оценки потенциального числа пользователей.

3. Обзор

Чаще всего качество продукта зависит от успешности определения потребностей пользователя. Неверно расставленные приоритеты могут привести к печальному результату — снижению позиций продукта на рынке [4]. Исследователи в этой области отмечают, что принятие решений о требованиях к продукту очень усложнено недостатком информации, в том числе, о привычках, поведении и ожиданиях пользователей [26].

Одним из применений данной работы является снижение неопределенности в процессе приоритезации за счет получения дополнительных знаний. Это знание — количество пользователей, которые потенциально стали бы регулярно применять функциональность — основывается на предсказании поведения группы пользователей, сформированном по историческим данным IntelliJ IDEA.

В этой главе рассмотрены распространенные принципы, по которым может определяться порядок реализации идей. Это даст возможность понять контекст использования критериев приоритезации и сформировать набор требований к ним. Затем будут приведены примеры решения похожих задач по анализу и предсказанию поведения пользователей.

3.1. Методологии приоритезации

Все популярные методологии приоритезации требований и идей можно разделить на три группы: сегментирующие, порядковые, численные [35].

Сегментирующие (nominal scale) механизмы предполагают определение нескольких групп и категоризацию всех идей и требований по этим группам на основе значимости. Группы имеют определенный приоритет, который присваивается и идеям. В рамках одной группы все требования имеют одинаковый приоритет. MoSCoW [16], Top ten [1] и модель Кано [38] являются примерами таких подходов.

Порядковые (ordinal scale) методологии основываются на создании порядка для всего списка идей с точки зрения важности их реализации.

Ranking [4], Bubble sort [31] и Cumulative voting [32] принадлежат этой группе.

Численные (ratio scale) подходы проецируют значимость идеи на шкалу, основываясь на одном или нескольких параметрах соответствующей задачи. Например, охват пользователей (reach), влияние (impact), уверенность в оценках (confidence) и затраты на реализацию (effort) в методологии RICE [22]. Сюда же относятся Cost-value approach [32], Wieger's matrix [34], Analytical Hierarchy Process [39].

Данная классификация методов приоритезации была рассмотрена и изучена с разных сторон в нескольких работах [36, 35, 27, 11]. Среди прочих были рассмотрены следующие характеристики.

1. Простота использования.
2. Уверенность команды в полученном результате.
3. Масштабируемость похода на большое количество идей и требований.
4. Надежность подхода при участии нескольких членов команды разработки в процессе приоритезации.

В таблице 1 приведено сравнение вышеупомянутых классов методологий по этим характеристикам.

Класс методологий	Простота	Уверенность	Масштабируемость	Надежность
Сегментирующие	Высокая	Высокая	Низкая	Низкая
Порядковые	Низкая	Высокая	Низкая	Низкая
Численные	Средняя	Средняя	Низкая	Низкая

Таблица 1: Сравнение методологий приоритезации

Для того, чтобы справиться с проблемой низкой масштабируемости, были предложены алгоритмы приоритезации задач, основанные на машинном обучении [33, 41, 30]. Они значительно упрощают работу

с большим количеством критериев и идей, но проявляется проблема отсутствия прозрачности результатов, поскольку процесс приоритезации оказывается скрыт в алгоритме машинного обучения. Это подрывает уверенность команды разработки в результате.

Именно поэтому целью данной работы является создание критерия, который с одной стороны будет понятным и однозначно интерпретируемым членами команды, и с другой агрегировать информацию из реального мира, то есть быть объективным и применимым на большом числе идей. В таком случае, процесс приоритезации не изменяется, а лишь дополняется. Это позволит сохранить привычки команды и эффективнее внедрить использование такого критерия.

3.2. Предсказание поведения пользователей

Построение долгосрочных отношений с клиентами (customer relationship management, CRM) является важным звеном в работе любого бизнеса. От того, насколько эффективно выстроена работа с пользователями, зависит успех всей компании. Предсказание оттока пользователей — одна из наиболее актуальных задач в этой области. Она заключается в заблаговременном нахождении группы пользователей, склонных через некоторое время отказаться от использования продукта.

Большая часть открытых исследований в этой области использует наборы данных телекоммуникационных компаний для обучения алгоритмов машинного обучения [7, 3, 12]. Как правило, в них содержится несколько десятков параметров: описание клиента (пол, возраст и т.д.), агрегированная информация о поведении (среднее количество звонков в месяц, использование текстовых сообщений) и другие.

3.2.1. Решающие деревья

Классическим подходом в этой задаче является использование решающих деревьев и лесов (decision tree) [15]. Основным преимуществом данного подхода является возможность визуализировать обученный классификатор. Это позволяет компаниям понять взаимосвязь признаков

клиента с вероятностью его оттока, ответить на вопрос «Почему пользователь отказался от использования продукта» и применить это знание для увеличения прибыли. Тем не менее точность предсказания таких алгоритмов достаточно невелика (Accuracy = 0,859) [8]. Можно улучшить результат, сохранив интерпретируемость модели, с помощью методов отбора признаков (feature selection) [18]. Это позволяет оценить влияние каждого признака на получение итогового результата, сократить их количество в обучающей выборке и тем самым снизить вероятность переобучения модели. Использование Information Gain [43] и Correlation Attributes Ranking Filter [7] и в качестве алгоритмов feature selection позволило повысить точность деревьев до 0,896 [8].

Тем не менее интерпретируемость модели несущественна в задаче оценки потенциального количества регулярных пользователей, поскольку интерес представляет лишь агрегированное значение, а не влияние атрибутов на класс конкретного пользователя. Помимо этого в рамках работы с IntelliJ IDEA есть возможность получить более подробный набор данных, на котором решающие деревья и леса показывают результаты хуже. Это проявилось на наборе данных Prozorro [28], который содержит данные о 140 тысячах пользователей и двух миллионах объектах, с которыми они взаимодействовали. В эксперименте [21] были получены значения AUC 0,65 и 0,83 для решающих деревьев и алгоритмов с использованием бустинга (boosting algorithms) соответственно.

3.2.2. Алгоритмы бустинга

Подход бустинга заключается в итеративном обучении нескольких слабых классификаторов с целью их объединения в более сильный классификатор [42]. При добавлении нового алгоритма в цепочку, ему некоторым образом присваиваются веса (веса остальных моделей пересчитываются), определяющие степень его влияния на итоговый результат предсказания в цепочке классификаторов. XGBoost, LightGBM и CatBoost являются зарекомендовавшими себя алгоритмами этого типа [5]. Сравнение последних двух в рамках задачи предсказания оттока

пользователя для банковского продукта [2] приведено в таблице 2.

	Accuracy	AUC	Recall	F1-score
Catboost	0,88	0,84	0,88	0,87
LightGBM	0,85	0,81	0,86	0,86

Таблица 2: Сравнение метрик алгоритмов предсказания оттока для банковского продукта

3.3. Алгоритмы рекомендаций

Задачу определения числа потенциальных пользователей функциональности приложения можно рассмотреть и с точки зрения следующего подхода. С помощью рекомендательных систем можно предсказать вероятность того, что человек воспользуется исследуемой функциональностью. Используя это значение, будем его считать потенциальным пользователем этого действия, если соответствующее значение вероятности превышает определенный порог. Таким образом, рекомендательные системы могут быть применимы в данной работе.

Рекомендательные системы принято делить на три класса: collaborative filtering (CF), content-based filtering и гибридные алгоритмы [14]. Фильтрация на основе контента предполагает анализ характеристик объектов, с которыми взаимодействует пользователи, и поиск взаимосвязи между взаимодействиями и этими характеристиками. Такой подход не применим при оценке потенциального количества пользователей в IntelliJ IDEA. Функциональность IDE не обладает достаточным количеством характеристик, поскольку весь исходный код и контекст использования IntelliJ не записывается. Имеющихся данных недостаточно для применения как фильтрации на основе контента, так и гибридных алгоритмов, поскольку последние совмещают информацию о контенте с агрегацией поведения пользователей.

Группу подходов коллаборативной фильтрации можно в свою очередь разделить на алгоритмы, основанные на соседстве (nearest neighbors algorithm), и на подходы, основанные на модели.

3.3.1. Алгоритмы, основанные на соседстве

Подходы, основанные на соседстве, предсказывают отношение пользователя к новому объекту, используя его оценки другими пользователями, похожими на данного. Степень похожести можно оценивать как по свойствам объектов (что не применимо в рамках IntelliJ IDEA), так и по распределению оценок других пользователей [6]. В таком случае строится матрица отношения пользователей к объектам, где в каждой строке находится вектор оценок некоторого пользователя. Оценки могут быть как явными (рейтинг фильма или товара в онлайн-магазине), так и неявными [25] (время, частота, интенсивность взаимодействия). Затем для выдачи рекомендаций пользователю в построенной матрице находятся вектора, наиболее похожие на вектор оценок данного пользователя. Из них извлекаются объекты с наивысшими оценками и сортируются в порядке убывания интереса группы похожих пользователей — чем выше и чаще был оценен объект, тем выше он окажется в списке рекомендаций.

Классические примеры таких алгоритмов — метод k-ближайших соседей [29], а также подходы, основанные на сходстве Пирсона [24] и косинусном сходстве [23].

3.3.2. Алгоритмы, основанные на модели

В случае основанных на модели подходов используются алгоритмы машинного обучения для определения наиболее релевантных объектов. Входные данные все так же представляют собой матрицу оценок объектов пользователями (user-item matrix). С их помощью обучается модель, которая будет определять предполагаемую оценку объектов конкретных пользователей.

Поскольку количество объектов в таких наборах данных как правило велико, часто для реализации такого подхода используются алгоритмы кластеризации и понижения размерности [20, 17].

Алгоритмы кластеризации [10, 9] предполагают разбиение всей выборки пользователей на группы, используя входные данные. Затем в

приоритете для рекомендаций оказываются наиболее популярные объекты в группе, к которой принадлежит пользователь. Это позволяет заметно ускорить определение рекомендованных объектов, но сократить точность предсказаний на небольшом количестве объектов. В рамках задачи оценки потенциального количества пользователей функциональности IntelliJ IDEA преимущество по скорости предсказания не играет существенной роли, поскольку в этой задаче не требуется высокая скорость подсчета оценки в реальном времени, ведь основное ее применение в процессе приоритезации происходит раз в цикл планирования. С другой стороны, количество записываемых действий IntelliJ значительно уступает распространенным наборам данных, что может привести к снижению точности, поэтому алгоритмы кластеризации не применимы в рамках поставленной задачи.

Методы понижения размерности наоборот, представляют особый интерес. Многие записываемые события, производимые пользователями IntelliJ IDEA, носят массовый характер, например, открытие проекта, использование функций копирования, вставки, отмены и другие. С большой вероятностью они не вносят значимый вклад в использование пользователем исследуемой функциональности. Использование методов снижения размерности (сингулярное разложение [37], матричная факторизация [13]) позволит снизить зависимость от этих функций, снизить переобучение модели и повысить итоговую точность.

3.3.3. Формулировка задачи в терминах рекомендаций

Таким образом, задача оценки потенциального количества пользователей с точки зрения рекомендательных систем может быть поставлена следующим образом. Рассмотрим входные данные в качестве матрицы $A(u, v)$, в которой ряд соответствует конкретному пользователю $u \in \{1, ..., U\}$, а столбец определенному действию (функциональности) $v \in \{1, ..., V\}$ внутри продукта. Значение $A(u_1, v_1)$ отражает количество раз, когда пользователь u_1 применял действие v_1 за определенный период. Тогда необходимо предсказать вероятность того, что пользователь $\bar{u}$ активирует действие $\bar{v}$. При этом, если известно, что этот человек еще

не сталкивался с этой функциональностью (или сталкивался мало), а предсказанная вероятность достаточно высока, его можно считать потенциальным пользователем. Получив предсказания для всех u, можно оценить количество потенциальных пользователей среди них.

3.4. Выводы

Несмотря на то, что найти описанный подход для оценки потенциального количества пользователей не удалось, были выявлены классы похожих задач, результаты решения которых можно использовать в данной работе. Например, стоит уделить внимание алгоритмам бустинга, и, в частности, LightGBM, поскольку они показывают одни из лучших результатов в задаче оценки оттока пользователей. Помимо этого, большое количество исследований в области планирования и приоритезации, а также целый список методологий ранжирования задач, дополнительно подтверждают актуальность этих задач в мировом сообществе.

4. Описание подхода

Для того, чтобы оценить число пользователей, которые потенциально могли бы воспользоваться определенной функциональностью приложения, предложен следующий подход. Имея статистику использования приложения, для каждого пользователя собирается набор параметров, характеризующий его поведение в выбранный период времени. Этот набор специфичен для разных приложений в зависимости от их сложности и доступной статистики. Затем выбирается исследуемое действие (функциональность, для которой мы хотим посчитать число потенциальных пользователей), и для каждого пользователя определяется частота применения этого действия в рамках того же периода времени. Затем на основании этого значения мы определяем, является ли пользователь регулярным для выбранной функциональности, то есть использует ли он ее достаточно часто. Далее применяются алгоритмы машинного обучения в попытке предсказать, является ли пользователь регулярным на основании характеристик его поведения. Таким образом предполагается найти неявные маркеры того, что исследуемая функциональность может помочь или быть интересной пользователям. Наконец, те пользователи, для которых алгоритм предсказал, что они являются регулярными, но которые, на самом деле, таковыми не являются, и называются потенциальными. Это можно интерпретировать как то, что модель нашла их поведение в достаточной степени похожим на фактических регулярных пользователей. При этом количество таких людей и является значением потенциального числа пользователей исследуемой функциональности.

5. Архитектура и реализация

В данной главе рассматривается разработанная система предсказания потенциального количества пользователей для различной функциональности IDE, описывается ее архитектура, реализация, подход к оценке точности предсказаний, а также сервис для предоставления доступа к результатам предсказания.

5.1. Архитектура системы

В основе разработанной системы лежит клиент-серверная архитектура. Само веб-приложение состоит из следующих компонентов: база данных, хранящая информацию о совершенных пользователями действиях в IntelliJ IDEA, сервер для обучения модели предсказания на исторических данных, их регулярного обновления и подсчета метрик, описывающих потенциальное количество пользователей, и клиент, который позволяет просматривать потенциальное количество пользователей для различной функциональности IntelliJ IDEA.

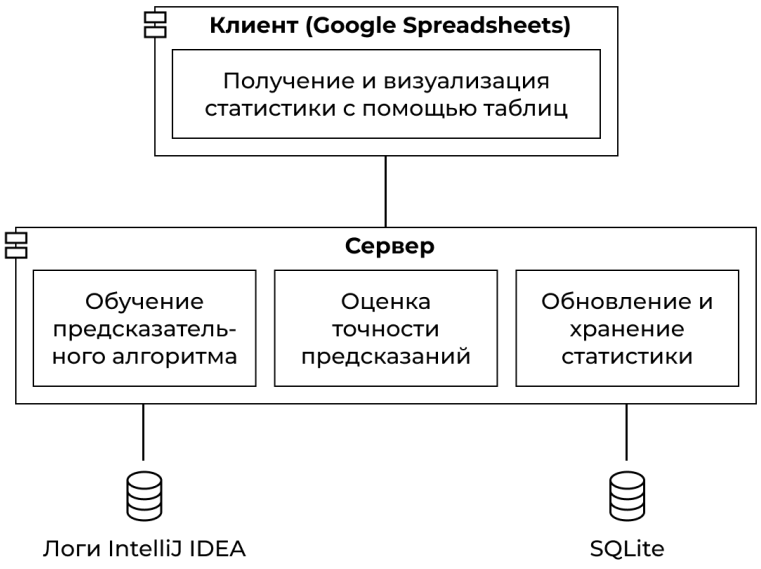


Рис. 1: Архитектура системы оценки потенциального количества пользователей

Рассмотрим каждый из компонентов подробнее.

5.2. База данных

JetBrains использует облачный интерактивный сервис запросов, позволяющий анализировать и манипулировать данными с помощью диалекта SQL. Он служит для предоставления внутрикорпоративного доступа к данным о действиях пользователей в большинстве продуктов, включая IntelliJ IDEA.

Эти данные записываются самой IDE при наличии согласия пользователя на сохранение и обработку анонимной статистики. В контексте данной работы наиболее интересны две категории регистрируемых продуктом событий, примеры записей в базе данных для которых приведены в таблице 3.

1. Взаимодействие с элементами интерфейса с помощью мыши, а также с помощью горячих клавиш. Такие записи содержат анонимизированный идентификатор пользователя, название действия, дату и время события, а также идентификатор сессии работы с продуктом, в рамках которой было совершено это взаимодействие. Каждая сессия — это период времени, до и после которого пользователь не совершал никаких действий в течении 30 минут.

2. Факт наличия в одном из открытых проектов файлов с различными расширениями и количество таких файлов. Помимо названия расширения, в таких записях содержатся анонимизированный идентификатор проекта и количество файлов с соответствующим расширением. Запись событий такого типа происходит при изменении структуры текущего проекта — при открытии нового, добавлении файлов, переименовании и так далее.

ID польз.	Тип	Событие	Количество или горячая клавиша	Дата, время	ID сессии
1	Action	OpenProject	—	14.01 15:19	1ef4a...
1	FileInProject	.java	17	14.01 15:20	1ef4a...
1	Action	Debug	Ctrl+R	14.01 16:37	s5h73...
2	FileInProject	.txt	1	14.01 17:11	jf0ge...
2	Action	Find	Ctrl+F	14.01 17:11	jf0ge...

Таблица 3: Упрощенный пример записей с логами регистрируемых событий IntelliJ IDEA

Рассмотренная в данной работе система агрегирует данные, приведенные выше, для каждого пользователя на основании его идентификатора при помощи SQL запросов и Python. Таким образом, формируется набор данных, необходимый для обучения предсказательной модели.

Условия тарификации пользователей облачного сервиса запросов подразумевают плату пропорциональную объему данных, которые сервис просканировал для того, чтобы исполнить SQL запрос. В совокупности это приводит к необходимости оптимизации SQL запросов для экономии денежных затрат в процессе функционирования системы.

В данной работе удалось снизить эти затраты на порядок благодаря поэтапной фильтрации данных с созданием промежуточных таблиц, размер которых существенно меньше основных хранилищ данных о действиях пользователей в различных продуктах JetBrains.

У такого подхода есть недостаток, заключающийся в значительно выросшем времени исполнения запросов. Однако специфика работы системы не предполагает отслеживания данных в реальном времени. Обновления продуктов, способные влиять на реализацию функциональности и, как следствие, пользу для пользователей, происходят не чаще раза в неделю. Таким образом, длительное время выполнения запросов никак не влияет на эффективность работы системы в целом.

5.3. Обработка данных

Используя описанное выше хранилище данных и сервис запросов, для каждого пользователя IntelliJ IDEA, работавшего с продуктом в течение исследуемого периода времени, извлекается следующая статистика.

1. Количество возникновения событий взаимодействия с интерфейсом (для каждого события).
2. Количество использований горячих клавиш (для каждой комбинации).
3. Количество файлов в проектах (для каждого расширения).
4. Количество проектов.
5. Частота использования исследуемой функциональности (доля пользовательских сессий, в рамках которых исследуемое событие было зарегистрировано).
6. Суммарное количество сессий.
7. Операционная система.

При этом среди пользователей IntelliJ IDEA существуют люди, которые незаконно используют продукт, сбрасывая длительность бесплатного пробного периода IDE с помощью сторонних инструментов. Это приводит к тому, что при таком сбросе создается новый идентификатор пользователя. Как следствие это негативно влияет на корректность данных: вместо одного пользователя появляются два или более, запись событий для которых обрывается в момент сброса пробного периода. Для повышения точности работы алгоритма в данной работе используется фильтрация таких пользователей. Помимо такой фильтрации существует и ряд других проверок на корректность и полноту данных в хранилище, включая проверку соответствия всех идентификаторов

шаблону, чтобы избежать попадания в итоговый набор данных дефектов хранилища и ошибок регистрации событий.

Целью данного этапа обработки является создание единого набора данных, включающего вышеперечисленную статистику для каждого пользователя. Возможности используемого сервиса запросов не позволяют добиться такой цели, поскольку в его диалекте SQL отсутствует функция pivot для преобразования списка троек (id пользователя, название события, число событий) в колоночный вид, это проиллюстрировано в таблице 4. Для достижения этой цели используется язык Python и библиотека Pandas, являющаяся общепринятым стандартом в области работы с табличными данными, что в свою очередь на этапе проектирования позволило быстро проверять гипотезы с точки зрения формата данных и оптимизации алгоритмов машинного обучения.

ID польз.	Событие	Время
1	GotoDeclaration	12:57
1	GotoDeclaration	12:59
1	Debug	13:01
1	Resume	13:01
1	ToggleBreakpoint	13:01
1	Resume	13:05
1	Resume	13:06
1	Resume	13:09
1	FindAndReplace	13:12

→

ID польз.	Событие	Кол-во
1	Resume	4
1	GotoDeclaration	2
1	Debug	1
1	ToggleBreakpoint	1
1	FindAndReplace	1

Таблица 4: Иллюстрация использования функции pivot для преобразования данных

Финальный набор данных выглядит так, как показано в таблице 5.

ID пользователя	Действие 1	...	Действие N	Всего действий	...
1	5		176	4391	...
2	17		0	10232	...

...	Файл 1	...	Файл M	Всего файлов	ОС	Проектов	Сессий
...	0		13	102	Mac	2	15
...	0		0	2071	Win	8	41

Таблица 5: Пример значений и список параметров для финального набора данных

Количество параметров значительно превосходит стандартный для задач машинного обучения размер набора данных. Были предприняты попытки снижения размерности, поскольку это могло привести к повышению точности дальнейшего предсказания. Однако это привело к обратному эффекту, поскольку для данной задачи естественно, что какое-то одно действие сильно влияет на использование другого. Например, наличие файлов с расширением .gradle сильно повышает шансы полезности использования конфигурации параметров системы сборки Gradle[1].

Многие события в IntelliJ IDEA взаимосвязаны или продублированы. В качестве самого простого примера таких событий можно привести использование горячих клавиш, вызывающих действие в IDE: активация комбинации неизбежно приводит к срабатыванию самого события. Такие параметры и характеристики необходимо удалить, прежде чем преступать к обучению модели. В противном случае, они предоставят модели нечестное преимущество, которое приведет к значительному повышению точности, однако сделает применение подхода невозможным для оценки потенциального числа пользователей. Список таких взаимосвязанных событий определяется вручную с опорой на знание устройства сбора статистики в IntelliJ IDEA, и затем проверяется при помощи подсчета корреляции суммарного числа срабатываний в качестве одного из этапов обработки данных. В случае если было найдено событие с

[1]https://gradle.org/

корреляцией 0,7 или больше, будет отправлено предупреждение администратору сервиса, а процесс обработки приостановлен.

Финальным этапом обработки данных является расчет частоты использования исследуемой функциональности и определение классов пользователей: регулярных и не регулярных. На основе этого параметра будет обучен предсказательный алгоритм, цель которого решить задачу бинарной классификации — предсказать принадлежность пользователя к одному из двух вышеупомянутых классов.

5.4. Алгоритм предсказания числа потенциальных пользователей

Для решения задачи классификации и предсказания, является ли пользователь регулярным, используется алгоритм LightGBM [5]. Он хорошо зарекомендовал себя в похожих задачах, например, для предсказания оттока пользователей (user churn [2]). В обзоре было показано, что он наряду с CatBoost[2] и XGBoost[3] показывает наиболее высокие метрики F меры (F1-score) и точности(Accuracy).

LightGBM обучается предсказывать принадлежность пользователя к классу регулярных или не регулярных на основании вектора параметров, описанного в предыдущем разделе. Затем считается количество пользователей, предсказанных регулярными, однако на момент конца периода исследования таковыми не являющихся (то есть число ложноположительных предсказаний). Это значение и является числом потенциальных пользователей с точки зрения той функциональности, для которой определялась частота использования и класс регулярности пользователя. Для того, чтобы получить оценку потенциального количества пользователей для разных действий IDE, предлагается обучить отдельный классификатор для каждого из таких событий.

[2]https://catboost.ai/

[3]https://xgboost.readthedocs.io/en/stable/

5.5. Оценка точности предсказаний

Для того, чтобы интерпретировать пользователей из класса ложно положительных предсказаний модели в качестве потенциальных, необходимо показать, что такие пользователи, действительно, со временем начинают регулярно применять исследуемое действие. Для такой оценки эффективности модели предложена следующая метрика трансформаций классов пользователей (1), визуализация которой представлена на Рис. 3.

$$\frac{TP_2 \mid FP_1}{(TP_2 \mid FP_1) + (TN_2 \mid FP_1) + (FN_2 \mid FP_1) + (Churn_2 \mid FP_1)} \quad (1)$$

- TP — класс истинно положительных (True Positive, Рис. 2) предсказаний
- TN — класс истинно отрицательных (True Negative, Рис. 2) предсказаний
- FP — класс ложно положительных (False Positive, Рис. 2) предсказаний, класс потенциальных пользователей
- FN — класс ложно отрицательных (False Negative, Рис. 2) предсказаний
- Churn — класс пользователей, прекративших использование продукта
- Индекс (TP_2) — номер периода времени для формирования набора данных, который использовался для обучения модели и, соответственно, получения предсказаний

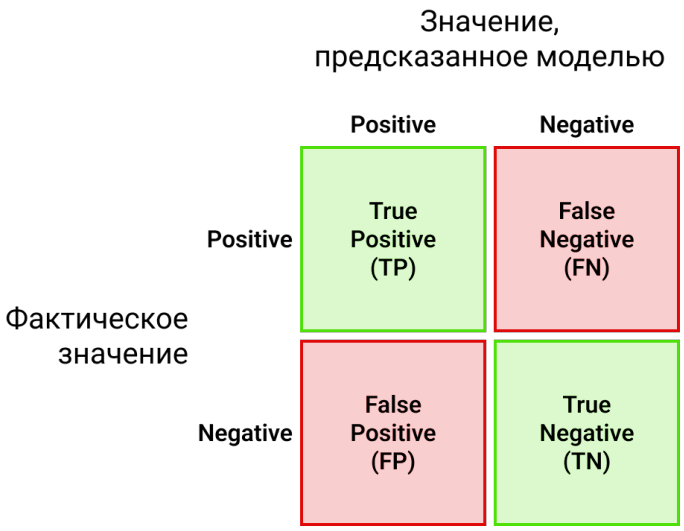


Рис. 2: Иллюстрация определения классов для результатов предсказаний

Данная метрика лежит в интервале $[0, 1]$ и оценивает долю пользователей, которые действительно стали регулярными во втором периоде времени после того, как они были обозначены потенциальными в первом периоде. Это значение можно интерпретировать как вероятность того, что пользователь станет из потенциального регулярным (в случае, если он перестанет быть потенциальным). Чем выше эта метрика, тем лучше. Причем значение 1, хоть и является идеальным результатом, в реальности едва ли достижимо, поскольку многие люди прекращают пользоваться продуктом из-за внешних обстоятельств, меняют со временем свои привычки (например, при покупке устройства с другой операционной системой), а также обладают разной скоростью обучения, из-за чего выбранного периода времени может быть недостаточно для того, чтобы пользователь стал регулярным.

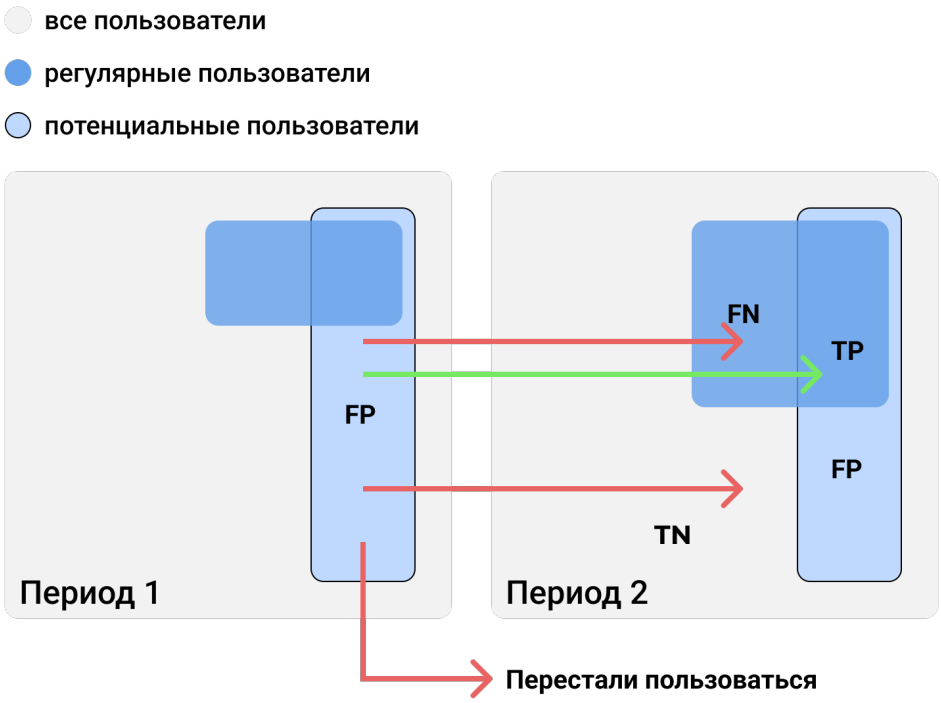


Рис. 3: Иллюстрация принципа вычисления метрики трансформаций

Итоговые значения метрики трансформаций потенциальных пользователей для моделей, обученных для пяти разных событий в IntelliJ IDEA, представлены в таблице 6 (столбец 2). Были использованы данные IntelliJ IDEA Ultimate за 4 недели в сентябре и следующие 4 недели в октябре 2021 года, соответствующие периодам 1 и 2 в формуле. Несмотря на то, что не удалось вплотную приблизиться к значениям 1, данный результат показывает применимость разработанного подхода, поскольку вероятность стать регулярным пользователем значительно превосходит доли регулярных пользователей для соответствующей функциональности.

В дополнении к оценке метрики трансформаций для потенциальных пользователей было проведено сравнение ее значения со значением похожей метрики для пользователей, которые не являются ни потенциальными, ни регулярными. При наличии превосходства метрики для потенциальных пользователей, можно говорить о том, что вероятность стать регулярным для потенциального пользователя превосходит вероятность такого изменения для остальных. Этого удалось добиться в

процессе данной работы, что в свою очередь подтверждает работоспособность предложенного подхода. Степень превосходства вероятности стать регулярным пользователем для потенциальных характеризуется значением мультипликатора, который получается делением метрики трансформаций для регулярных пользователей на такую же метрику для не потенциальных и не регулярных одновременно. Примеры значений метрик приведены в таблице 6.

Функциональность	Метрика трансформаций для классов пользователей		Мультипликатор
	Потенциальные	Не потенциальные	
Feature1	0,15 ± 0,003	0,008 ± 0,0002	18,84 ± 0,53
Feature2	0,15 ± 0,005	0,02 ± 0,0006	7,71 ± 0,35
Feature3	0,19 ± 0,002	0,03 ± 0,0002	6,99 ± 0,1
Feature4	0,37 ± 0,004	0,09 ± 0,002	4,06 ± 0,07
Feature5	0,23 ± 0,011	0,06 ± 0,002	3,97 ± 0,18

Таблица 6: Метрики трансформаций для классов потенциальных, не потенциальных пользователей и мультипликатор

Что касается классических метрик для оценки моделей машинного обучения, в процессе оптимизаций основное внимание было уделено F1 мере (F1-score) и значению точности (Accuracy).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$F_1\ score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3}$$

Использование F1 меры обусловлено низкой сбалансированностью классов. Большинство пользователей IntelliJ IDEA ограничиваются лишь небольшим числом действий, которые они используют регулярно.

Метрика Accuracy же описывает общую точность предсказаний модели и является общепринятой при решении задач классификации.

Дополнительно было проведено сравнение модели LightGBM со случайными классификаторами (dumb classifiers[4]). Они представляют со-

[4]https://rikunert.com/dumb$_c lassifier_p erformance$

бой самые простые предсказательные модели, сравнение с которыми может выявить случайность полученных результатов и факт распознавания выбранной моделью неочевидных взаимосвязей. Сравнение было проведено с классификатором, предсказывающим случайный класс пользователя (random classifier) и моделью постоянно предсказывающей наиболее популярный класс (majority classifier). Результаты подтвердили, что использование модели LightGBM действительно позволило найти дополнительные взаимосвязи между параметрами и вырастить метрики (Таблица 7).

Классификатор	Accuracy	F1-score
LightGBM	0,83 ± 0,01	0,66 ± 0,02
Random classifier	0,5 ± 0,002	0,5 ± 0,002
Majority classifier	0,92 ± 0	0 ± 0

Таблица 7: Сравнение модели LightGBM со случайными классификаторами

5.5.1. Ограничения подхода

Несмотря на относительно высокие значения метрик трансформации, приведенные выше, у предложенного подхода есть значительное ограничение. У некоторой функциональности IntelliJ IDEA, не пользующейся большой популярностью у пользователей, количество регулярных пользователей очень мало. Как следствие, классы пользователей в задаче бинарной классификации получаются крайне несбалансированными со значительным превосходством в числе класса нерегулярных пользователей. Это приводит к снижению метрик качества и точности предсказательной модели для такой функциональности, что, в свою очередь, делает невозможным исследование нераспространенной функциональности при помощи данного подхода. Тем не менее этот недостаток снижает ценность применения оценки потенциального количества пользователей при приоритезации задач незначительно, поскольку функциональность, которой отводится больше всего внимания на планировании итераций разработки, все еще доступна для оценки с

точки зрения потенциального количества пользователей.

5.6. Серверная часть

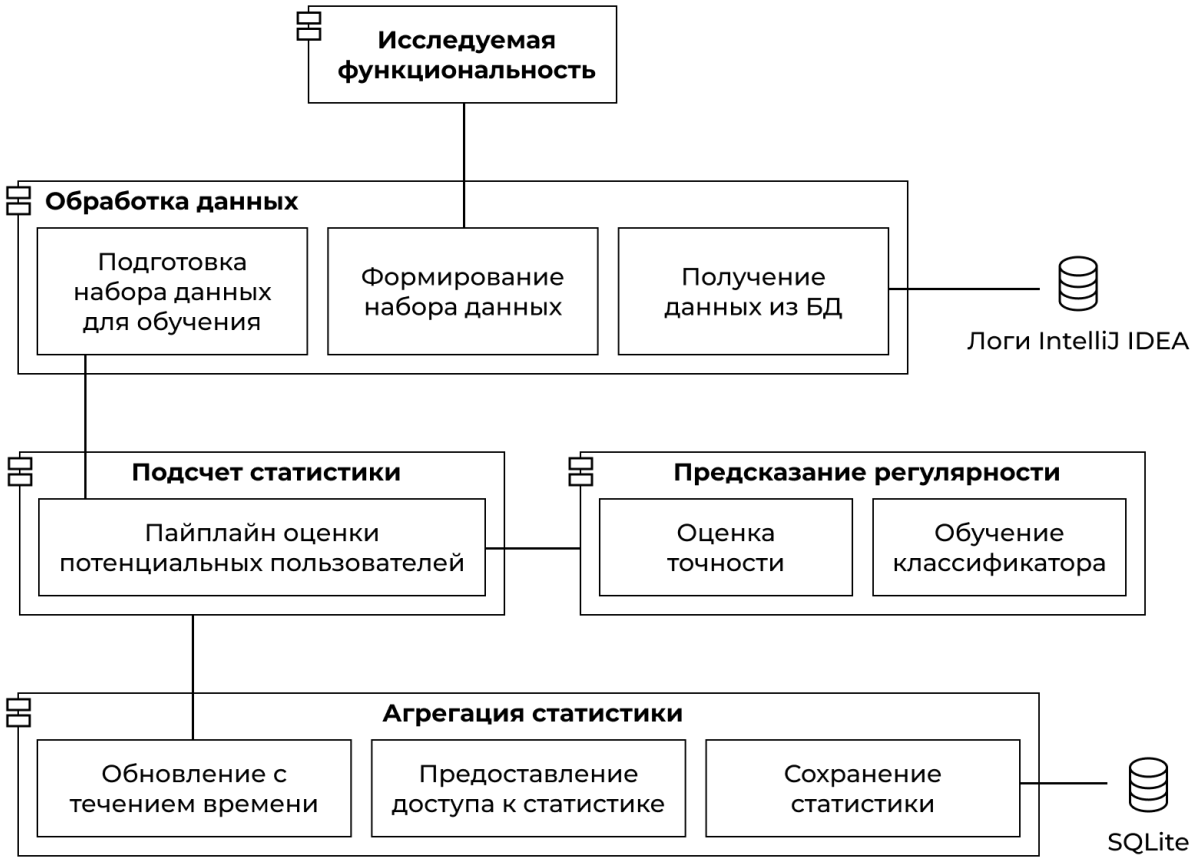


Рис. 4: Архитектура сервера системы оценки потенциального количества пользователей

Описанные выше этапы получения оценки потенциального количества пользователей объединены серверной частью разработанного приложения. Ее архитектура представлена на (Рис. 4).

Исходя из набора функциональности (компонент «Исследуемая функциональность» на Рис. 4), которую предполагается исследовать и в конечном счете сравнивать для приоритезации, получаются данные из базы данных с логами IntelliJ IDEA через облачный сервис запросов. Затем формируется полный набор данных («Обработка данных» на Рис. 4, который подготавливается для обучения для каждой функциональ-

ности специальным образом, учитывая ее специфику и набор взаимосвязанных действий («Исследуемая функциональность» на Рис. 4).

Компонент «Предсказание регулярности» предоставляет интерфейс для обучения предсказательной модели — классификатора. Помимо этого он оценивает точность полученной модели, выбирает лучшую и предоставляет эти данные для дальнейшей интерпретации. За эту интерпретацию с точки зрения формирования конечных метрик и сборку последовательности этапов подхода отвечает «Подсчет статистики» на Рис. 4.

Главная задача сервера заключается в предоставлении доступа по REST API к актуальным результатам анализа и подсчитанной статистике («Агрегация статистики» на Рис. 4). Это реализовано при помощи библиотек для Python Flask[5] и APscheduler[6]. При помощи Flask реализованы аутентификация клиента и доступ к статистике, сохраняющейся в базу данных SQLite. Эта СУБД была выбрана из-за легковесности и простоты использования. При этом ее недостатки вроде низкой масштабируемости и скорости выполнения запросов в контексте выбранного подхода несущественны, поскольку количество исследуемой функциональности и, как следствие, и количество хранимых записей, ограничено требованиями к системе: человек должен иметь возможность оценить весь список, поэтому его размеры невелики. При помощи APscheduler реализовано периодическое обновление статистики: раз в неделю система проверяет наличие новых, еще не оцененных действий IDE, которые незамедлительно оценивает, а также раз в месяц собирает новые данные для анализа. Таким образом статистика поддерживается в актуальном состоянии и обновляется для периодов длительностью в 4 недели.

[5]https://flask.palletsprojects.com/en/2.1.x/

[6]https://apscheduler.readthedocs.io/en/3.x/

5.7. Реализация сервиса для просмотра результатов предсказания

Для использования системы с точки зрения приоритезации задач, с помощью Google Spreadsheets[7] был реализован интерфейс клиента. Данная технология была выбрана для максимально быстрого способа предоставить работающий прототип. С ее помощью удалось избежать длительной разработки фронтенд части системы и в то же время предоставить интерфейс, с которым пользователи сталкиваются ежемесячно из-за различных рабочих задач. Пользователями выступают члены команды разработки и их руководители, которые заинтересованы в приоритезации задач.

С помощью метода REST API клиент получает с сервера актуальную статистику и отображает ее в привычном для пользователей табличном виде. Возможности Google Sheets изначально не предоставляют функциональности для получения данных в соответствии с этим архитектурным стилем и отображения JSON файлов. Однако широкая возможность расширения функциональности за счет подключаемых Javascript файлов Apps Script[8] позволила использовать клиент-серверный подход с использованием REST API.

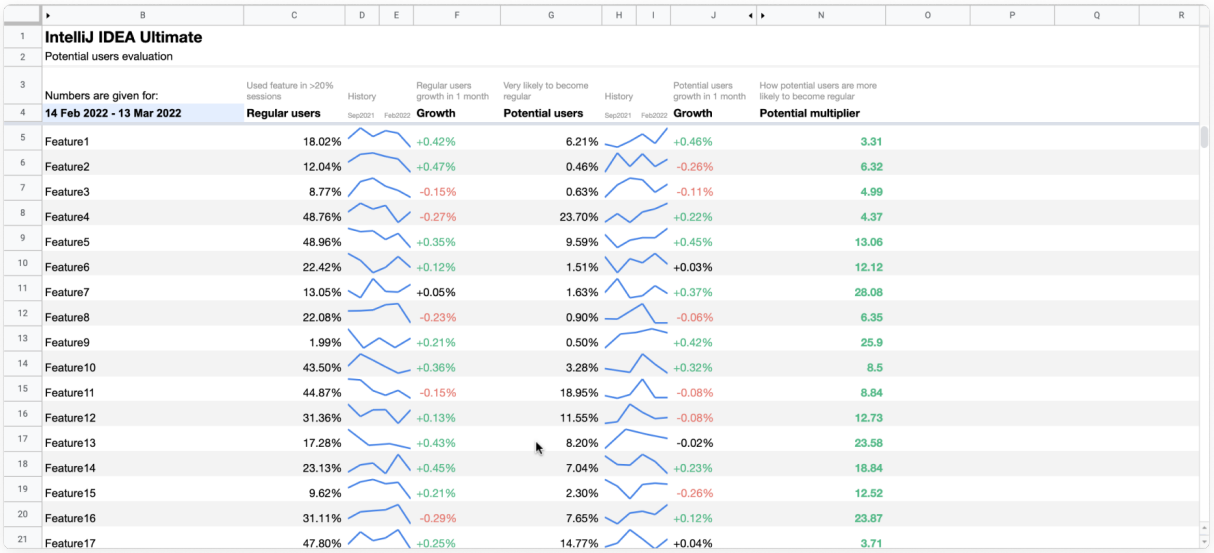
Интерфейс клиента (Рис. 5) предоставляет агрегированную информацию, которая была подсчитана на сервере. Каждая функциональность, для которой была завершена оценка числа потенциальных пользователей, отображается в списке. Для каждого действия из списка доступны текущие (за последний подсчитанный период длительностью 4 недели) значения регулярных и потенциальных пользователей. Также для каждого из них доступна статистика за несколько предыдущих периодов и изменение с последнего из них для отслеживания динамики их изменений. Помимо этого отображается мультипликатор значения метрики трансформаций для потенциальных пользователей по отношению к такой же метрике для не потенциальных и не регулярных. Это позво-

[7]https://spreadsheets.google.com/

[8]https://developers.google.com/apps-script

ляет команде учитывать точность подсчитанной системой статистики в процессе приоритезации. Значения мультипликатора ниже определенного порога изменяют цвет на красный, чтобы показать пониженный уровень доверия к представленному результату. Функциональность, которая не получила достаточно высокую точность предсказаний (которая выходит за рамки ограничений подхода), не будет отображаться в списке.

Таким образом, сравнивая значения потенциального количества пользователей, команда разработки может оценить, насколько сильным может быть эффект от улучшения определенных действий. Чем выше число потенциальных пользователей для определенной функциональности, тем более эффективным будет реализация задач, направленных на ее улучшение и распространение в продукте.



IntelliJ IDEA Ultimate

Potential users evaluation

Numbers are given for:	Used feature in >20% sessions	History	Regular users growth in 1 month	Very likely to become regular	History	Potential users growth in 1 month	How potential users are more likely to become regular
14 Feb 2022 - 13 Mar 2022	**Regular users**	Sep2021 Feb2022	**Growth**	**Potential users**	Sep2021 Feb2022	**Growth**	**Potential multiplier**
Feature1	18.02%		+0.42%	6.21%		+0.46%	3.31
Feature2	12.04%		+0.47%	0.46%		-0.26%	6.32
Feature3	8.77%		-0.15%	0.63%		-0.11%	4.99
Feature4	48.76%		-0.27%	23.70%		+0.22%	4.37
Feature5	48.96%		+0.35%	9.59%		+0.45%	13.06
Feature6	22.42%		+0.12%	1.51%		+0.03%	12.12
Feature7	13.05%		+0.05%	1.63%		+0.37%	28.08
Feature8	22.08%		-0.23%	0.90%		-0.06%	6.35
Feature9	1.99%		+0.21%	0.50%		+0.42%	25.9
Feature10	43.50%		+0.36%	3.28%		+0.32%	8.5
Feature11	44.87%		-0.15%	18.95%		-0.08%	8.84
Feature12	31.36%		+0.13%	11.55%		-0.08%	12.73
Feature13	17.28%		+0.43%	8.20%		-0.02%	23.58
Feature14	23.13%		+0.45%	7.04%		+0.23%	18.84
Feature15	9.62%		+0.21%	2.30%		-0.26%	12.52
Feature16	31.11%		-0.29%	7.65%		+0.12%	23.87
Feature17	47.80%		+0.25%	14.77%		+0.04%	3.71

Рис. 5: Интерфейс прототипа системы оценки потенциального количества пользователей (числа статистики заменены правдоподобными случайными значениями)

6. Апробация

Для того, чтобы проверить разработанный сервис в деле, одна из команд IntelliJ IDEA использовала его при планировании задач на следующую итерацию разработки. Предварительно для них был подготовлен необходимый список функциональности для сравнения и автоматически подсчитаны необходимые метрики.

Использование системы оценки потенциального числа пользователей позволило команде взглянуть на приоритет улучшения действия «Run to Cursor» под другим углом и в конечном счете включить соответствующие задачи в план. Несмотря на то, что оценить эффект на бизнес метрики от такого решения пока не представляется возможным, команда высоко оценила предоставленный им прототип: он дал им «ощущение уверенности в составленном плане» и «снизил объем неконструктивных обсуждений, основывающихся на личном опыте коллег». Тем не менее, прототип оказался «не слишком очевидным» для быстрого погружения и потребовал некоторое время, чтобы разобраться с метрикой качества оценки предсказаний (метрика трансформаций). Для того, чтобы исправить данный недочет, в конечном счете был использован мультипликатор вместо самого значения метрики трансформаций для потенциальных пользователей: «С множителем уже значительно понятнее, что происходит, понятно, насколько потенциальные пользователи вообще отличаются от остальных».

Таким образом, получилось применить прототип системы оценки потенциального количества пользователей в планировании, что позволило команде сместить приоритет и быть более уверенными в результате. Изначальные цели в создании понятного, интерпретируемого и объективного критерия для приоритезации задач были выполнены. Также на основе обратной связи были исправлены некоторые недостатки системы.

7. Заключение

В ходе данной работы была разработана система оценки потенциального количества пользователей для различной функциональности приложения на примере IntelliJ IDEA. Такая система позволит команде разработки получить объективный критерий для сравнения эффективности работы над различными подсистемами и функциональностью в рамках приоритезации и планирования задач.

В ходе работы были достигнуты следующие результаты.

1. Проанализированы существующие методологии ранжирования функциональности и идей, работа над которыми может привлечь наибольшее число пользователей. Выявлен ключевой недостаток таких методологий — отсутствие объективности критериев приоритезации.

2. Изучены существующие подходы к предсказанию потенциального количества пользователей для функциональности приложения, выявлены классы похожих задач и определены наиболее многообещающие алгоритмы для их решения.

3. Сформирована коллекция данных, характеризующая поведение пользователей в IntelliJ IDEA, а также подсистема для получения, обработки и подготовки этой коллекции к дальнейшему использованию в системе.

4. Реализована система предсказания потенциального количества пользователей для различной функциональности IntelliJ IDEA с использованием алгоритмов бустинга.

5. Проведена оценка точности предсказаний, предложена метрика оценки точности предсказаний и проанализированы значения результатов.

6. Разработан и апробирован внутрикорпоративный сервис для просмотра результатов оценки потенциального числа пользователей.

Список литературы

[1] Aasem Muhammad, Ramzan Muhammad, Jaffar Arfan. Analysis and optimization of software requirements prioritization techniques // 2010 International Conference on Information and Emerging Technologies. — 2010. — P. 1–6.

[2] Analysis and prediction of bank user churn based on ensemble learning algorithm / Yihui Deng, Dingzhao Li, Lvqing Yang et al. // 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA). — 2021. — P. 288–291.

[3] Archaux C., Martin Arnaud, Khenchaf Ali. An SVM based Churn Detector in Prepaid Mobile Telephony. — 2004. — 05. — P. 459 – 460.

[4] Babar Muhammad Imran, Ramzan Muhammad, Ghayyur Shahbaz A. K. Challenges and future trends in software requirements prioritization // International Conference on Computer Networks and Information Technology. — 2011. — P. 319–324.

[5] Bentéjac Candice, Csörgő Anna, Martínez-Muñoz Gonzalo. A comparative analysis of gradient boosting algorithms // Artificial Intelligence Review. — 2021. — 03. — Vol. 54.

[6] Breese John S., Heckerman David, Kadie Carl. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. — 2013. — 1301.7363.

[7] A Churn Prediction Model Using Random Forest: Analysis of Machine Learning Techniques for Churn Prediction and Factor Identification in Telecom Sector / Irfan Ullah, Basit Raza, Ahmad Kamran Malik et al. // IEEE Access. — 2019. — Vol. 7. — P. 60134–60149.

[8] A Churn Prediction Model Using Random Forest: Analysis of Machine Learning Techniques for Churn Prediction and Factor Identification in Telecom Sector / Irfan Ullah, Basit Raza, Ahmad Kamran Malik et al. // IEEE Access. — 2019. — Vol. 7. — P. 60134–60149.

[9] Collaborative Filtering Recommendation Algorithm Based on Item Clustering and Global Similarity / Suyun Wei, Ning Ye, Shuo Zhang et al. // 2012 Fifth International Conference on Business Intelligence and Financial Engineering. — 2012. — P. 69–72.

[10] Collaborative Filtering Recommender Systems / Ben Schafer, Ben J, Dan Frankowski et al. — 2007. — 01.

[11] Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique / Javed Khan, Izaz Rehman, Yawar Khan et al. // International Journal of Modern Education and Computer Science. — 2015. — 11. — Vol. 7. — P. 53–59.

[12] A Customer Churn Prediction Model in Telecom Industry Using Boosting / Ning Lu, Hua Lin, Jie Lu, Guangquan Zhang // Industrial Informatics, IEEE Transactions on. — 2014. — 05. — Vol. 10. — P. 1659–1665.

[13] Denise Chen. Recommender System — Matrix Factorization. — [Online; accessed 10-12-2021]. URL: https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b.

[14] Elkhelifi Aymen, Kharrat Firas Ben, Faiz Rim. Recommendation Systems Based on Online User's Action // 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. — 2015. — P. 485–490.

[15] Enterprise subscription churn prediction / Ramakrishna Vadakattu, Bibek Panda, Swarnim Narayan, Harshal Godhia // 2015 IEEE International Conference on Big Data (Big Data). — 2015. — P. 1317–1321.

[16] Fuzzy MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements / Khadija Sania Ahmad, Nazia Ahmad,

Hina Tahir, Shaista Khan // 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). — 2017. — P. 433–437.

[17] Item-based Collaborative Filtering Recommendation Algorithms / Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl // Proceedings of ACM World Wide Web Conference. — 2001. — 08. — Vol. 1.

[18] Jović A., Brkić K., Bogunović N. A review of feature selection methods with applications // 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). — 2015. — P. 1200–1205.

[19] Lai X., Xie M., Tan T.C. Optimizing product design using the Kano model and QFD // 2004 IEEE International Engineering Management Conference (IEEE Cat. No.04CH37574). — Vol. 3. — 2004. — P. 1085–1089 Vol.3.

[20] Lee Joonseok, Sun Mingxuan, Lebanon Guy. A Comparative Study of Collaborative Filtering Algorithms. — 2012. — 1205.3193.

[21] Malyar Mykola, Mykola Robotyshyn M.V., Sharkadi Maryana. Churn Prediction Estimation Based on Machine Learning Methods // 2020 IEEE 2nd International Conference on System Analysis Intelligent Computing (SAIC). — 2020. — P. 1–4.

[22] Mcbride Sean. RICE: Simple prioritization for product managers. — URL: https://www.intercom.com/blog/rice-simple-prioritization-for-product-managers/ (online; accessed: 2021-11-30).

[23] Neo4j graph data science. Cosine Similarity. — [Online; accessed 10-12-2021]. URL: https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/cosine/.

[24] Neo4j graph data science. Pearson Similarity. — [Online; accessed 10-12-2021]. URL: https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/pearson/.

[25] Neural Autoregressive Collaborative Filtering for Implicit Feedback / Yin Zheng, Cailiang Liu, Bangsheng Tang, Hanning Zhou // Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. — 2016. — Sep. — URL: http://dx.doi.org/10.1145/2988450.2988453.

[26] Ngo-The An, Ruhe Günther. Decision Support in Requirements Engineering // Engineering and Managing Software Requirements / Ed. by Aybüke Aurum, Claes Wohlin. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2005. — P. 267–286. — ISBN: 978-3-540-28244-0. — URL: https://doi.org/10.1007/3-540-28244-0_12.

[27] Olaronke Iroju, Ikono Rhoda, Gambo Ishaya. An Appraisal of Software Requirement Prioritization Techniques // Asian Journal of Research in Computer Science. — 2018. — 04. — P. 1–16.

[28] Oleksa Stepaniuk. ProZorro. Ukrainian public procurement dataset. — [Online; accessed 9-12-2021]. URL: https://www.kaggle.com/oleksastepaniuk/prozorro-public-procurement-dataset.

[29] Onel Harrison. Machine Learning Basics with the K-Nearest Neighbors Algorithm. — [Online; accessed 10-12-2021]. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-

[30] Perini Anna, Susi Angelo, Avesani Paolo. A Machine Learning Approach to Software Requirements Prioritization // IEEE Transactions on Software Engineering. — 2013. — Vol. 39, no. 4. — P. 445–461.

[31] Persis Voola, Babu A. Comparison of requirements prioritization techniques employing different scales of measurement // ACM SIGSOFT Software Engineering Notes. — 2013. — 07. — Vol. 38. — P. 1–10.

[32] Prioritization of quality requirements: State of practice in eleven companies / Richard Berntsson Svensson, Tony Gorschek, Björn Regnell et al. // 2011 IEEE 19th International Requirements Engineering Conference. — 2011. — P. 69–78.

[33] Qayyum Shamaila, Qureshi Ahsan. A Survey on Machine Learning Based Requirement Prioritization Techniques. — 2018. — 11. — P. 51–55.

[34] Rahim Md Shamsur, Chowdhury A Z M Ehtesham, Das Shovra. Rize: A proposed requirements prioritization technique for agile development // 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). — 2017. — Dec. — P. 634–637.

[35] Requirements Prioritization Techniques Comparison / Amjad Hudaib, Raja Masadeh, Mais Haj Qasem, Abdullah Alzaqebah // Modern Applied Science. — 2018. — 01. — Vol. 12.

[36] Requirements Prioritization Techniques in the last decade: A Systematic Literature Review / Juan Carlos B. Somohano-Murrieta, Jorge Octavio Ocharán-Hernández, Angel J. Sánchez-García, Maria de los Ángeles Arenas-Valdés // 2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT). — 2020. — P. 11–20.

[37] Reza Bagheri. Understanding Singular Value Decomposition and its Application in Data Science. — [Online; accessed 10-12-2021]. URL: https://towardsdatascience.com/understanding-singular-value-decomposition-and-its-application-i

[38] Rotar Laura, Kozar Mitja. The Use of the Kano Model to Enhance Customer Satisfaction // Organizacija. — 2017. — 12. — Vol. 50.

[39] Saaty R.W. The analytic hierarchy process—what it is and how it is used // Mathematical Modelling. — 1987. — Vol. 9, no. 3. — P. 161–176. — URL: https://www.sciencedirect.com/science/article/pii/0270025587904738.

[40] Sommerville I. Software Engineering. — Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1996. — ISBN: 0-13-703515-2.

[41] Talele Pratvina, Phalnikar Rashmi. Classification and Prioritisation of Software Requirements using Machine Learning – A Systematic Review // 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence). — 2021. — P. 912–918.

[42] Wikipedia contributors. Boosting (machine learning). — [Online; accessed 9-12-2021]. URL: https://en.wikipedia.org/wiki/Boosting_(machine_learning).

[43] Wikipedia contributors. Information gain in decision trees. — [Online; accessed 9-12-2021]. URL: https://en.wikipedia.org/wiki/Information_gain_in_decision_trees.