Автоматическая рекомендация рефакторинга “Выделение метода” при копировании кода в IDE

Кириленко А. А.

Научный руководитель: к.т.н. Т. А. Брыксин

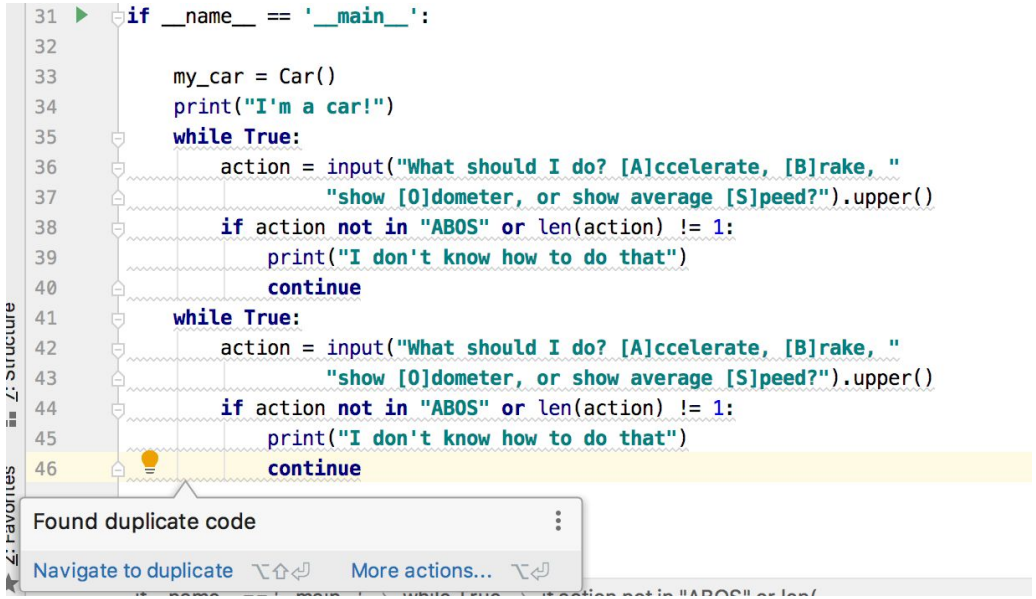
Санкт-Петербургская школа физико-математических и компьютерных наук

НИУ ВШЭ – Санкт-Петербург

Санкт-Петербург, 2020 год

Введение в область

- Дублирующийся код может содержать ошибки, которые придется долго искать по всему проекту [1]
- Для устранения используется рефакторинг “Выделение метода”



[1] Keisuke Hotta, Yui Sasaki, Yukiko Sano, Yoshiki Higo, and Shinji Kusumoto. 2012. An empirical study on the impact of duplicate code. Adv. Soft. Eng. 2012, Article 5 (January 2012), 22 pages. DOI:https://doi.org/10.1155/2012/938296

Обзор существующих решений

Генерация списка кандидатов и их ранжирование [1]

- Оценка кандидата на основе размеров, сложности кода, числу параметров

Извлечение признаков на основе истории проекта и обучение классификатора [2]

- Анализ изменений дублирований в коде при разработке и оценка их похожести

Алгоритмы на основе глубокого обучения [3]

- Рекуррентные нейронные сети

[1] Haas, Roman & Hummel, Benjamin. (2016). Deriving Extract Method Refactoring Suggestions for Long Methods. 144-155. 10.1007/978-3-319-27033-3_10.
[2] R. Yue, Z. Gao, N. Meng, Y. Xiong, X. Wang, and J. D. Morgenthaler, “Automatic clone recommendation for refactoring based on the present and the past,” in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018, pp. 115–126.
[3] White, Martin & Tufano, Michele & Vendome, Christopher & Poshyvanyk, Denys. (2016). Deep learning code fragments for code clone detection. 87-98. 10.1145/2970276.2970326.

Существующие аналоги

- JDeodorant [1]: инструмент, обнаруживающий запахи в коде и рекомендующий их исправления
- JExtract [2]: плагин, анализирующий код и предлагающий рефакторинг “выделение метода”
- CRec [3]: инструмент, рекомендующий выделение методов на основе анализа эволюции клонов

Общий недостаток: **применение требует сознательного решения**

[1] Tsantalis, Nikolaos & Chaikalis, Theodore & Chatzigeorgiou, Alexander. (2008). JDeodorant: Identification and Removal of Type-Checking Bad Smells. 329-331. 10.1109/CSMR.2008.4493342.
[2] Silva, Danilo & Terra, Ricardo & Valente, Marco. (2015). JExtract: An Eclipse Plug-in for Recommending Automated Extract Method Refactorings.
[3] R. Yue, Z. Gao, N. Meng, Y. Xiong, X. Wang, and J. D. Morgenthaler, “Automatic clone recommendation for refactoring based on the present and the past,” in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018, pp. 115–126.

Цель и задачи

Цель: создать расширение для IntelliJ IDEA, которое будет рекомендовать рефакторинг “Выделение метода” при копировании кода

Задачи:

- Отобрать признаки кода для решения задачи классификации
- Создать датасет с копируемым кодом
- Обучить модель, которая будет рекомендовать выделение кода
- Создать плагин для Intellij IDEA, в который будет интегрирована полученная модель
- Провести апробацию решения на реальных пользователях

Выбранные признаки

Проанализировано более 300 признаков, из которых отобрано 122

- Признаки кода
  - Число строк, глубина вложенности, длина
- Признаки связности
  - Использование внешних переменных и методов
- Признаки на основе истории проекта
  - Время после последнего изменения
  - Количество авторов, менявших фрагмент

[1] Caulo, Maria. A taxonomy of metrics for software fault prediction. 1144-1147. 10.1145/3338906.3341462. (2019).
[2] P. Nordfors, ‘Prediction of Code Lifetime’, Dissertation, 2017.
[3] Haas Roman, Hummel Benjamin. Deriving Extract Method Refactoring Suggestions for Long Methods. — 2016. — 01. — P. 144 – 155.

Сбор данных: естественные данные

Refactoring Miner [1]

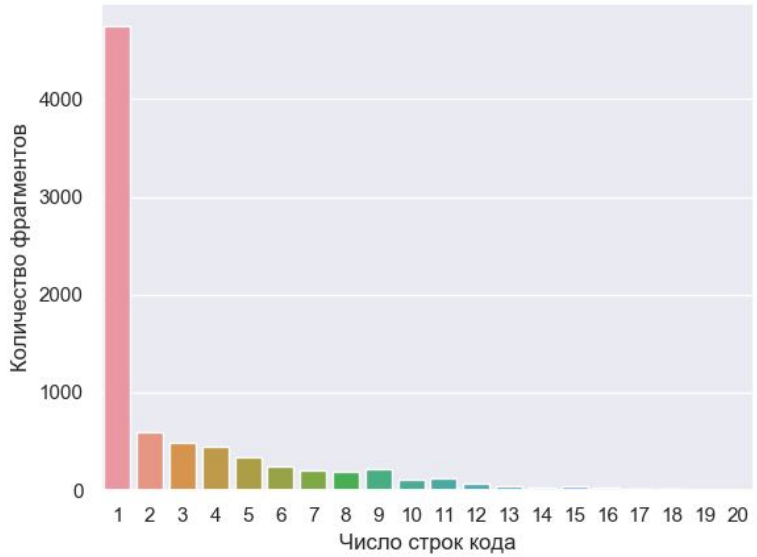
- Анализ истории коммитов в 23 проектах
- Сбор рефакторингов “Выделение метода”
- Фильтрация некорректных случаев

Результаты работы Refactoring Miner	
Репозиторий	**Извлечено выделений**
apache/*	2675
JetBrains/ intellij-community	1670
alibaba/*	673
Всего	**5018**

Полуавтоматический сбор данных

- Сбор логов операций копирования с пользователей в течении 5 месяцев (с явного согласия разработчиков)
- Разметка полученных данных вручную и путем кластеризации
- Собрано более 8000 фрагментов

Распределение собранных логов по числу строк



[1] Nikolaos Tsantalis, Matin Mansouri, Laleh Eshkevari, Davood Mazinanian, and Danny Dig, "Accurate and Efficient Refactoring Detection in Commit History," *40th International Conference on Software Engineering* (ICSE 2018), Gothenburg, Sweden, May 27 - June 3, 2018.

Сбор данных: синтетические данные

Автоматическая генерация на основе ранжирования

- Генерация последовательностей операторов
- Ранжирование кандидатов и разделение на классы
  - Оценивается упрощение исходного метода после извлечения кандидата

Обработка самостоятельных методов

- Код отдельных методов считается пригодным к извлечению
- Можно перебирать методы и вычислять их признаки

Результаты генерации датасета	
Репозиторий	**Сгенерировано фрагментов**
apache/*	23904
JetBrains/ intellij-community	6670
Всего	**30574**

Общее сравнение моделей

- Выборка с равным соотношением классов (5000 + 5000)
- Разбиение на обучающую и тестирующую выборку в соотношении 80:20
- Для моделей были подобраны оптимальные параметры
- Наиболее важна высокая точность

Результаты тестирования моделей					
Классификатор	Accuracy	Precision	Recall	F1	MCC
Random forest	**0.87**	**0.91**	**0.76**	**0.83**	**0.71**
SVM	0.67	0.82	0.59	0.68	0.53
Neural net	0.77	0.81	0.53	0.64	0.61
Naive Bayes	0.72	0.64	0.74	0.69	0.66

Модель на основе случайного леса была выбрана для дальнейшего использования

Тестирование качества

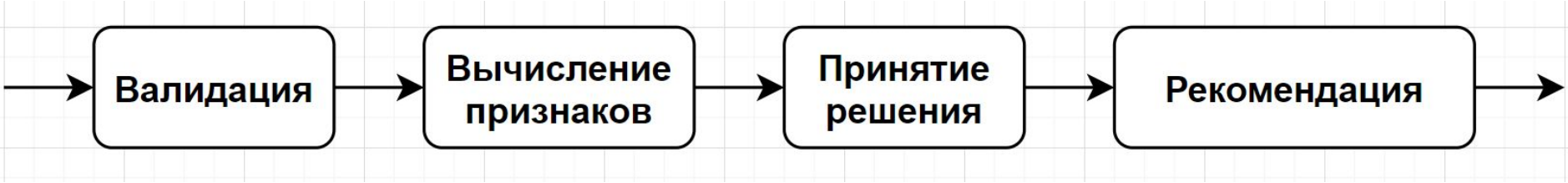
Требуется убедиться в работоспособности модели на различных целевых группах кандидатов на извлечение

Тестирование случайного леса на различных выборках					
Выборка	Accuracy	Precision	Recall	F1	MCC
Равные выборки, 6к, 10-fold CV	0.86	0.90	0.75	0.82	0.69
Тест на отдельном проекте, 10к+3к	0.95	0.83	0.64	0.72	0.71
Длинные фрагменты, 5к+1к	0.76	**0.94**	0.63	0.76	0.61
Короткие фрагменты, 5к+1к	0.84	0.85	0.80	0.83	0.69
Длинные + отдельный проект, 10к+500	0.74	**0.91**	0.60	0.73	0.63
Длинные + большая вложенность, 3к+500	0.81	0.83	0.75	0.79	0.62

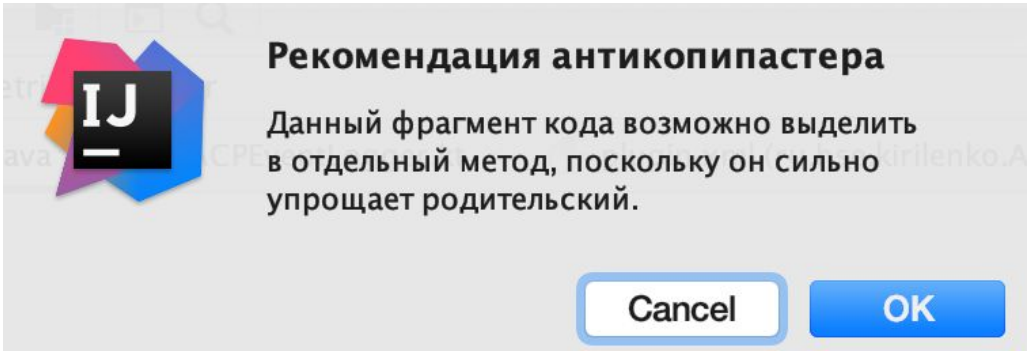
Модель демонстрирует высокую точность на данных из отдельного проекта и фрагментах большого размера

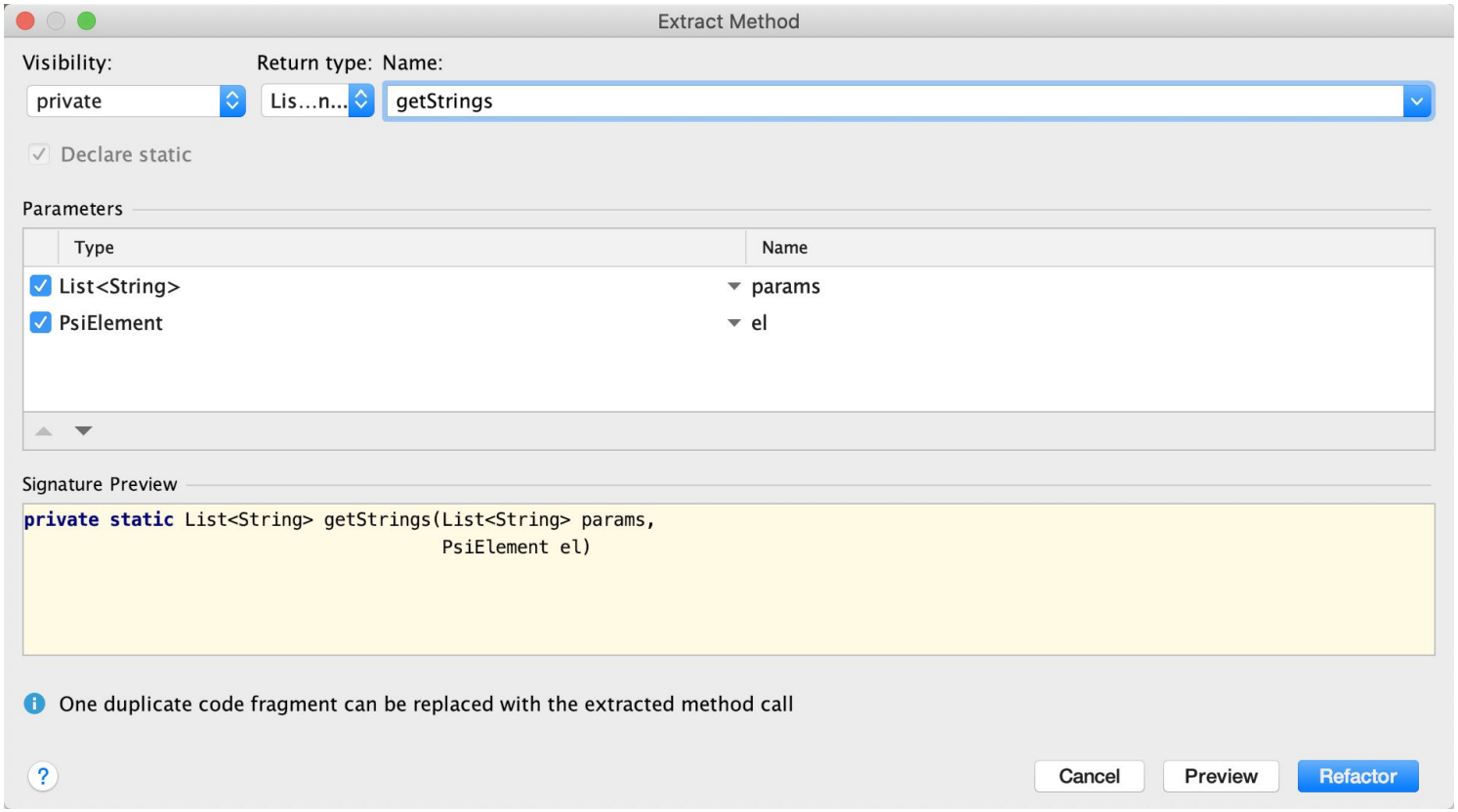
Разработанное расширение к IntelliJ IDEA

- Проверка корректности фрагмента
- Параллельное вычисление признаков
- Отложенная рекомендация — проверка неизменности фрагмента
- Обоснование необходимости рефакторинга по дереву решения



Разработанное расширение к IntelliJ IDEA





Апробация

- Приняло участие 5 разработчиков
- Оценивались удобство использования, производительность, а также качество рекомендаций
- В процессе апробации было дано 334 рекомендации, из которых принято 67

Вывод: полученными рекомендациями пользуются, созданный в рамках работы инструмент приносит пользу

Результаты

- Выбран подходящий набор признаков
- Создан датасет копирований кода и инструменты, позволяющие их собирать
  - Общий размер датасета ≈60000 элементов
- Обучена модель, предсказывающая необходимость сделать выделение метода
  - На собранном датасете точность классификации достигает 90%
- Реализовано расширение, интегрирующее полученную модель и реализующее рекомендацию рефакторинга и выделение метода
- Планируется публикация расширения в маркетплейсе IntelliJ

Реализация: https://github.com/JetBrains-Research/anti-copy-paster

Рефакторинг “Выделение метода”

- Улучшает читаемость кода
- Убирает дублирования кода
- Изолирует независимые части

```
void printOwing() {
  printBanner();

  // Print details.
  System.out.println("name: " + name);
  System.out.println("amount: " + getOutstanding());
}
```

```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}

void printDetails(double outstanding) {
  System.out.println("name: " + name);
  System.out.println("amount: " + outstanding);
}
```

Общая схема подхода

- Каждому фрагменту кода сопоставляется вектор численных признаков
- Собирается большое количество векторов каждого из классов
- Обучается модель, которая классифицирует новые векторы

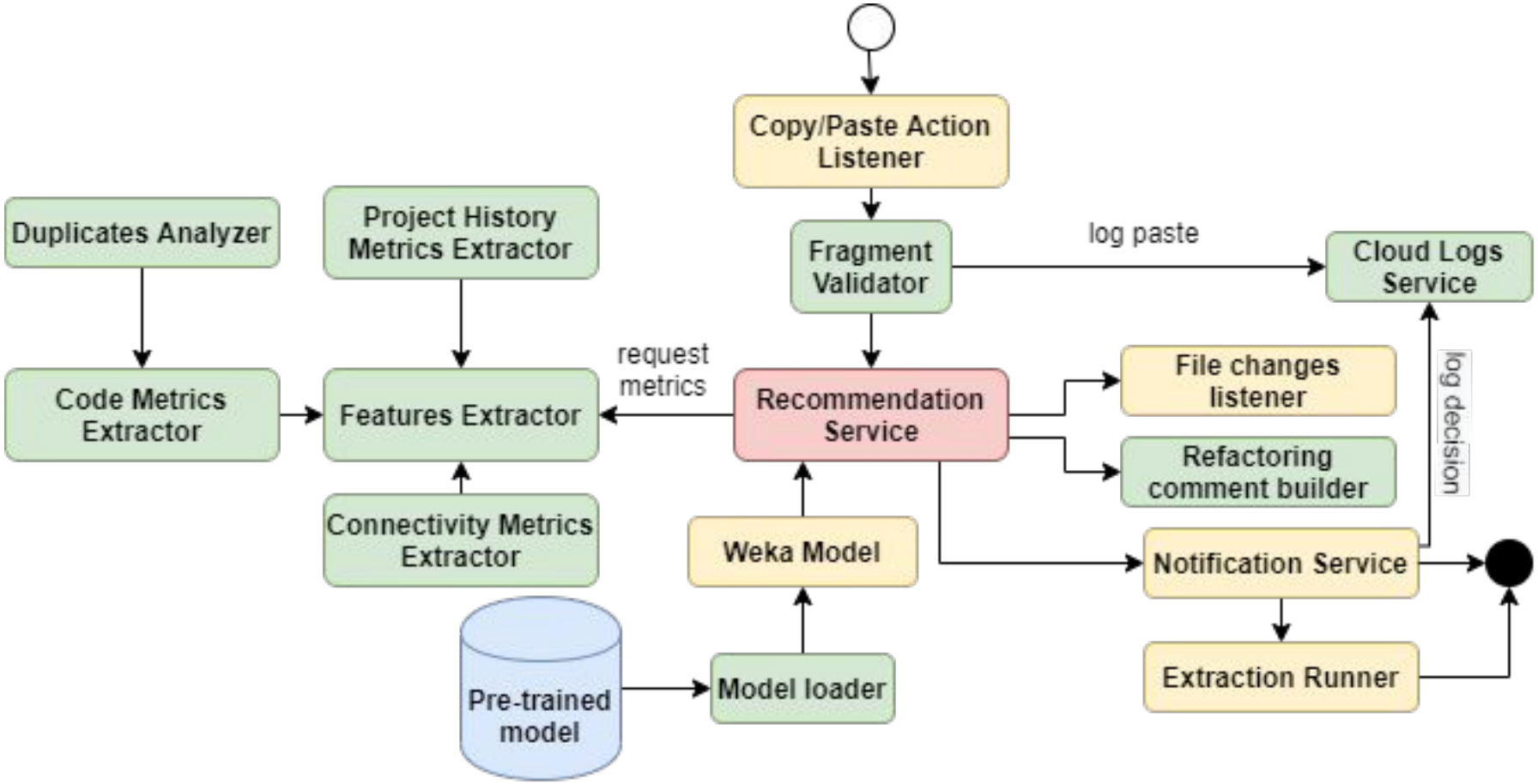
Метрики: детали

- Число строк и символов, средние и абсолютные значения
- Количество дублирований в проекте
- Глубина вложенности - средняя максимальная и суммарная
- Количество локальных переменных внутри фрагмента, объявленных снаружи
- Количество объявленных переменных внутри фрагмента, используемых снаружи
- Количество разработчиков, менявших фрагмент
- Среднее время жизни строки во фрагменте

Параметры моделей

- Случайный лес
  - Подбор числа деревьев и максимальной глубины (50, 100, 200, 500 деревьев и максимальная глубина 6, 8, 10)
  - Наилучший результат: 200 деревьев глубины не более 6
- Нейронная сеть
  - Конфигурации из 1, 2, 3 слоев размерами 32, 64, 128; функция активации - ReLU, tanh
  - Наилучший результат: 2 слоя размера 64
- Метод опорных векторов
  - Проверка различных ядер: полиномиальное(степеней 1, 2, 3) и RBF, подбор параметра регуляризации
  - Наилучший результат: RBF ядро с C=2

Плагин: архитектура



Архитектура

Плагин: возможности

- Валидация копируемых фрагментов
- Параллельное вычисление метрик
- Отслеживание изменений фрагментов в течении некоторого промежутка времени
- Описание необходимости рефакторинга по пути принятия решения
- Отслеживание дублирований в проекте
- Отправка пользовательских действий в облако для оценки качества
- Замена всех дублирований на вызов метода, где это возможно

Метрики качества

- Достоверность (accuracy)
- Точность (precision)
- Полнота (recall)
- F1-оценка
- Коэффициент корреляции Мэтьюса (MCC)

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

Оценка упрощения метода при извлечении фрагментов

Сумма трех слагаемых: оценок размера, вложенности и параметризации

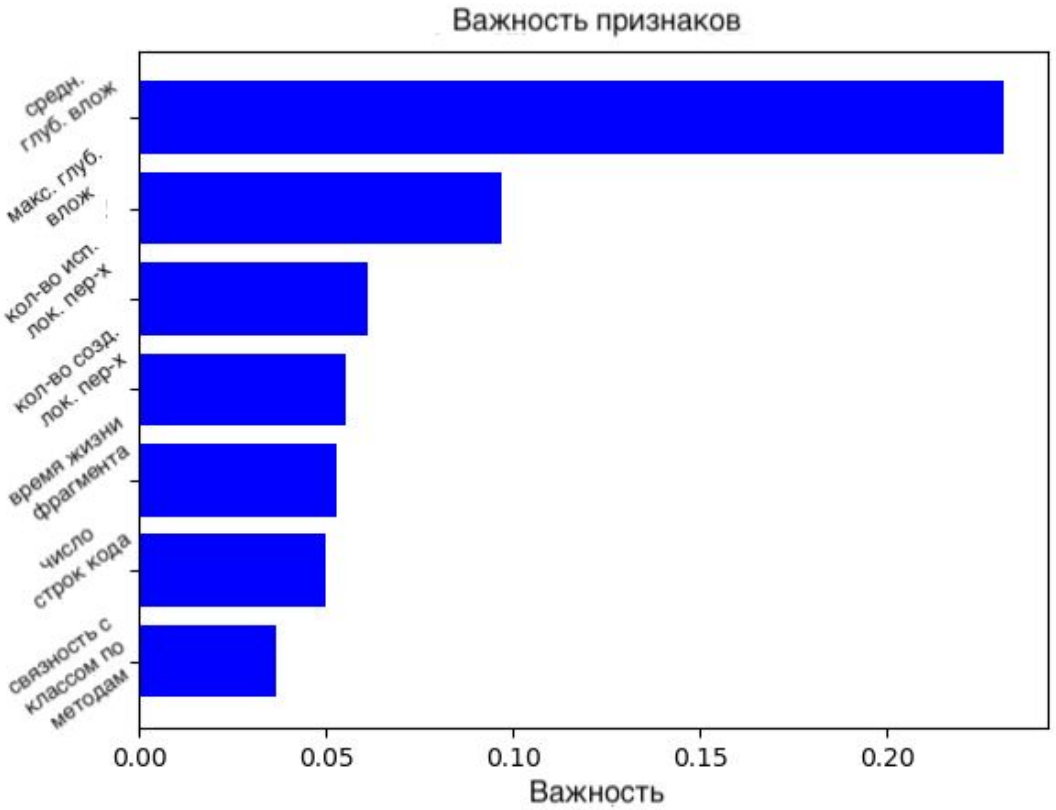
- $S_{размер} = \min(0.1*\min(L_{ex}, L_{rest}), MAX\_SCORE\_S), \quad MAX\_SCORE\_S = 3$
- $S_{влож} = \min(D_{before} - D_{ex}, D_{before} - D_{rest})$
- $S_{сум.\ влож} = 2 * D_{before} * \min(A_{before} - A_{ex}, A_{before} - A_{rest}) / A_{before}$
- $S_{парам} = MAX\_SCORE\_P - P_{in} - P_{out}, \quad MAX\_SCORE\_P = 4$

$S = S_{размер} + S_{влож} + S_{сум.\ влож} + S_{парам}$

Важные признаки

Обучившись на 10 самых важных признаках:

- Достоверность: 0.77
- Точность: 0.76
- Полнота: 0.63
- F1: 0.69
- MCC: 0.53



Разбиение данных по времени

Тренировочная выборка: первые 80% фрагментов по времени

Тестирующая выборка: оставшиеся 20%

Проблема: для синтетических данных время создания не определено

Классификатор	Accuracy	Precision	Recall	F1	MCC
Random forest	**0.85**	**0.89**	**0.75**	**0.82**	**0.70**
SVM	0.64	0.81	0.62	0.70	0.54
Neural net	0.78	0.80	0.51	0.62	0.62
Naive Bayes	0.73	0.62	0.75	0.68	0.67