Прогнозирование ошибок методами машинного обучения

Лобанов Артём
Научный руководитель: Брыксин Тимофей

Введение

- Прогнозирование ошибок – поиск ошибкоопасных мест в коде
- Важно для тестирования
- Единицей программы в данном случае является метод

Цель работы

Целью работы было написание плагина для IntelliJ IDEA, который бы находил в коде ошибкоопасные места

Для это нужно было решить следующие задачи:

- Выбрать тип классификатора
- Найти обучающую выборку
- Выбрать параметры обучения
- Написать сам плагин

Использованные материалы

Прочитаны статьи про различные подходы к прогнозированию ошибок с помощью машинного обучения:

- A Study on Software Defect Prediction Using Fuzzy Decision Trees
  - Studia Univ. Babes Bolyal, Informatica, Volume LXI, Number 2, 2016
  - Zsuzsanna Marian, Istvan-Gergely Czibula, Ioan-Gabriel Mircea and Vlad-Sebastian Ionescu
- A Novel Approach for Software Defect Prediction Using Fuzzy Decision Trees
  - Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016 18th International Symposium
  - Zsuzsanna Marian ; Ioan-Gabriel Mircea ; Istvan-Gergely Czibula ; Gabriela Czibula
- Comparative analysis of statistical and machine learning methods for predicting faulty modules.
  - Applied Soft Computing, 2014.
  - Ruchika Malhotra. Comparative analysis of statistical and machine learning methods
- Empirically Guided Software Development Using Metric-Based Classification Trees
  - University of California at lrvine 1990
  - Adam A. Porter and Richard W. -/by,

Выбор классификатора

Первоначально планировалось использовать нечёткие деревья решений

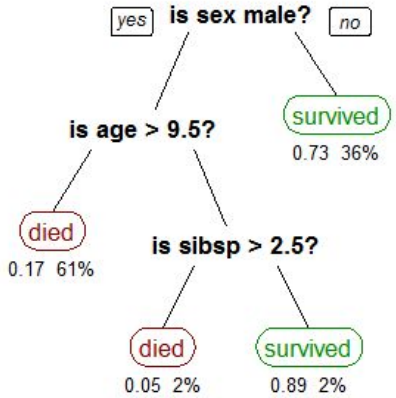
Однако выяснилось, что обычные деревья решений решают задачу поиска ошибок лучше

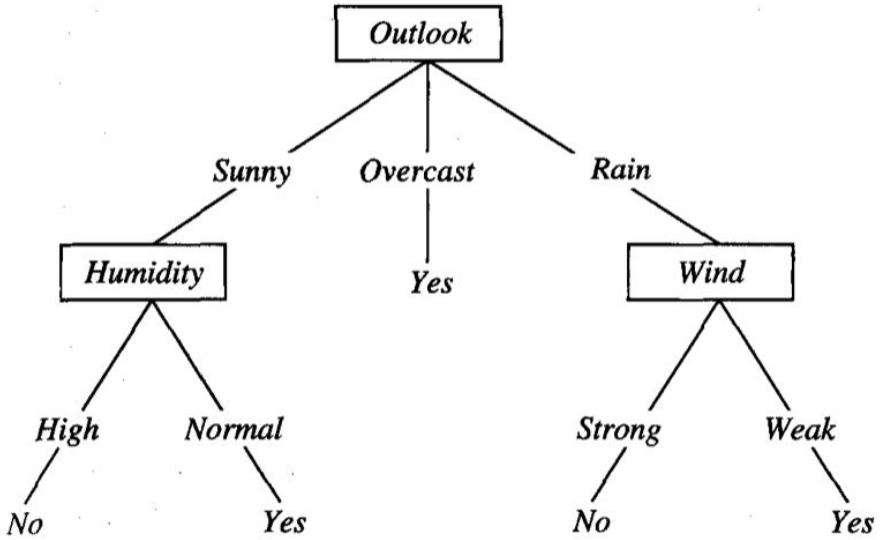
Approach	Ar1	Ar3	Ar4	Ar5	Ar6
Our FuzzyDT	0.807	0.706	0.776	0.893	0.804
Genetic Programming [1]	0.530	0.67	0.65	0.67	0.630
Multiple Linear Regression [1]	0.550	0.61	0.62	0.55	0.590
Binary Logistic Regression [15]	0.551	**0.87**	0.73	0.39	0.722
Logistic Regression [12]	0.734	0.82	**0.82**	**0.91**	0.640
Logistic Regression [9]	0.494	n/a	n/a	n/a	0.538
Artificial Neural Networks [9]	0.711	n/a	n/a	n/a	0.774
Support Vector Machines [9]	0.717	n/a	n/a	n/a	0.721
Decision Trees [9]	**0.865**	n/a	n/a	n/a	**0.948**
Cascade Correlation Networks [9]	0.786	n/a	n/a	n/a	0.758
GMDH Network [9]	0.744	n/a	n/a	n/a	0.702
Gene Expression Programming [9]	0.547	n/a	n/a	n/a	0.688

В таблице показаны результаты (AUC) разных подходов к решению задачи. Ar1 и т.д. - наборы данных

Деревья решений

- Общая идея: разделение примеров по какому-то атрибуту, рекурсивное построение для получившихся групп примеров
- Алгоритм CART для обучения дерева решений





Использованные технологии

- Плагин MetricsReloaded для IntelliJ IDEA для вычисления метрик
- Библиотека Weka для создания и обучения классификатора
- Обучающая выборка (http://openscience.us/repo/)

name	number of instances	defects	% defects
Ar*	427	60	14,05%
kc3.arff	193	36	18,65%
mc1.arff	1987	46	2,32%
mc2.arff	124	43	34,68%
mw1.arff	252	27	10,71%
pc1.arff	704	61	8,66%
pc2.arff	744	16	2,15%
pc3.arff	1076	134	12,45%
pc4.arff	1457	178	12,22%
pc5.arff	17185	516	3,00%

Обучающая выборка

- Итоговая обучающая выборка была собрана из перечисленных на прошлом слайде наборов данных
- Содержит огромное число повторений, что вызывает переобучение
- Около 7 тысяч уникальных примеров
- Использовались значения метрик, посчитанных во всех наборах
- Уникальные примеры были случайным образом разделены на обучающую (70% примеров) и тестовую (30%) выборки

Параметры обучения

Основная проблема для обучения – несбалансированность данных

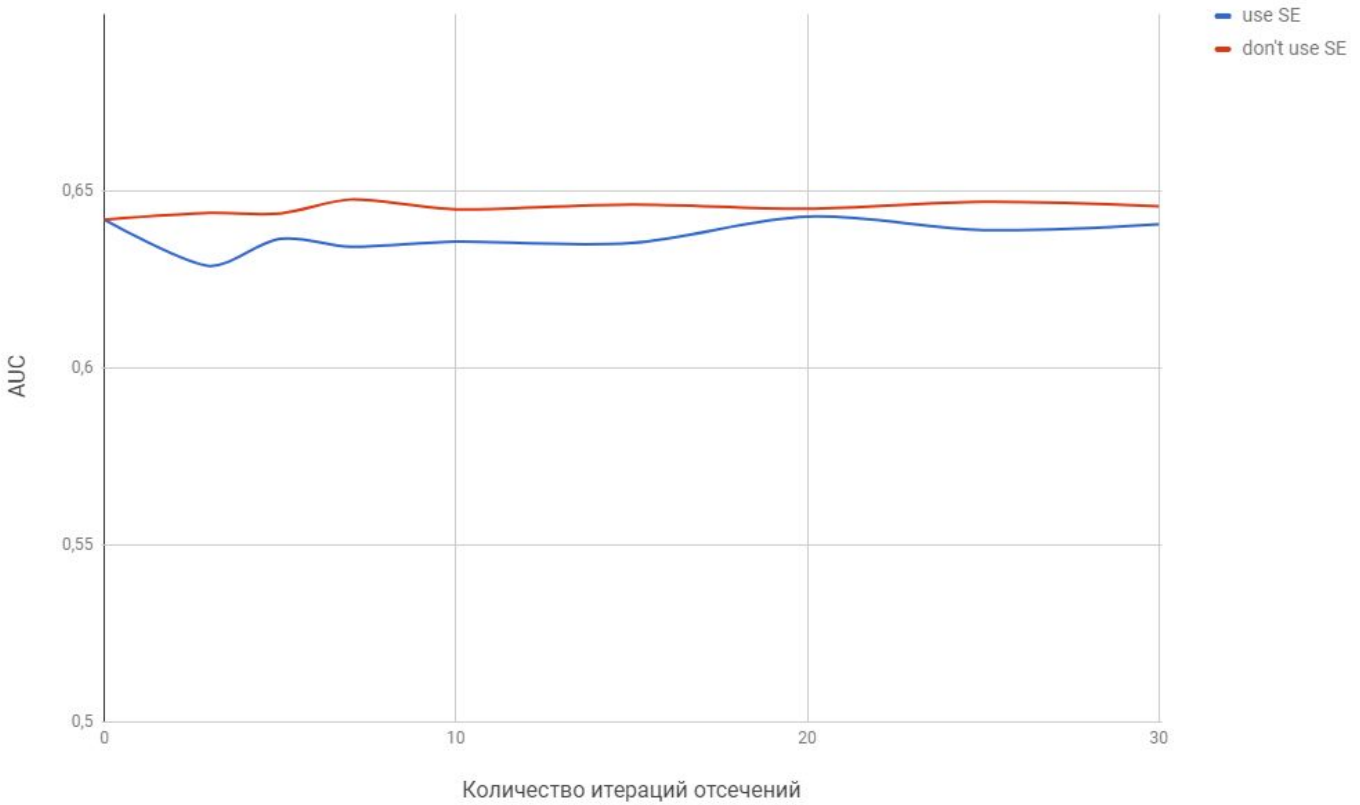
Варьируемые параметры при обучении:

- Процент отсеянных хороших примеров
- Коэффициент веса дефектных методов
- Количество итераций отсечений
- Минимальное количество (суммарный вес) примеров в листьях
- Стратегия выбора лучшего дерева при отсечениях (два варианта)

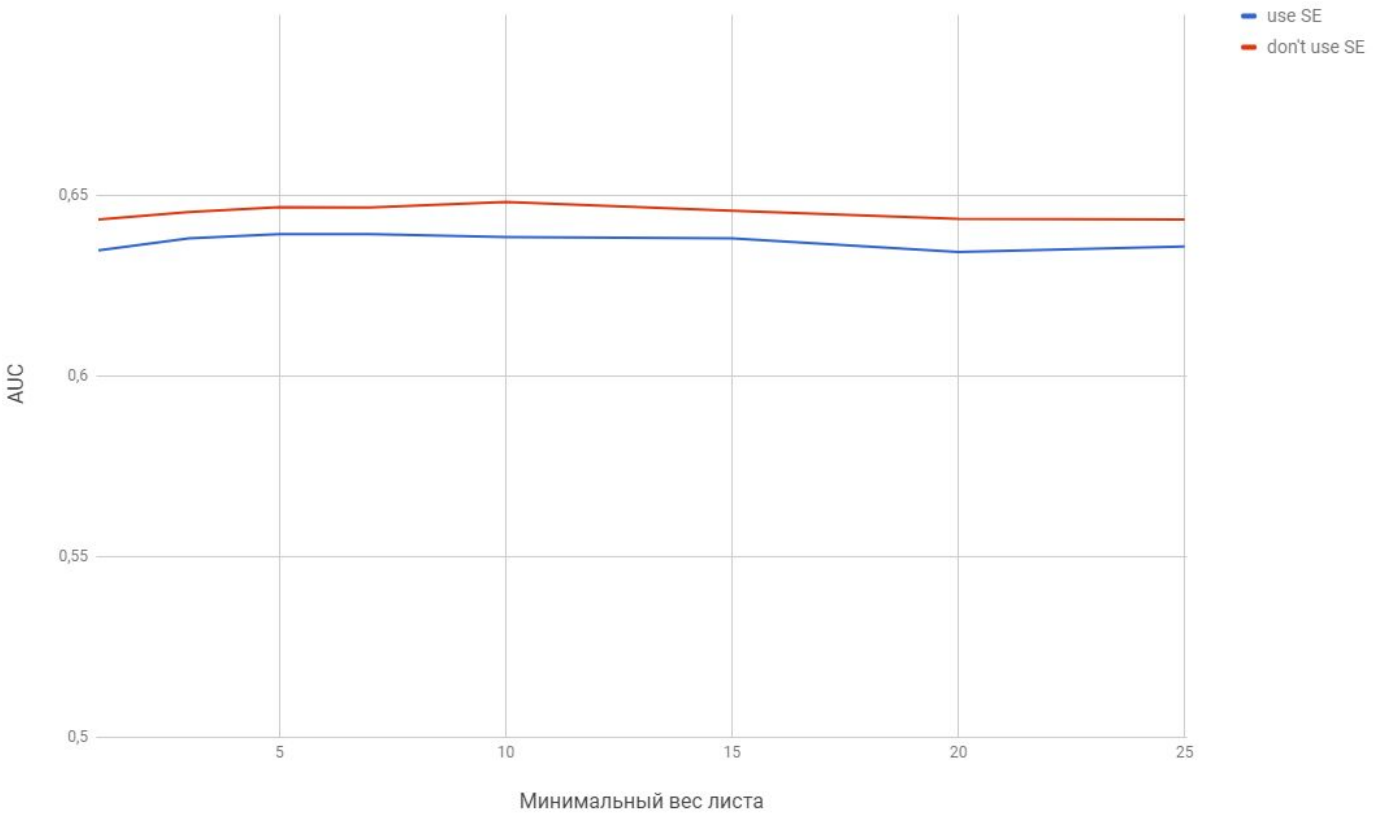
Перебор параметров

1. Количество итераций отсечения $\in$ {0, 3, 5, 7, 10, 15, 20, 25, 30}
2. Процент дефектных примеров $\in$ {0.13 (естественный процент), 0.15, 0.2}
3. Вес дефектных примеров $\in$ {1, 2, 3, 4, 5, 7, 10, 15}
4. Минимальный вес в листьях $\in$ {1, 3, 5, 7, 10, 15, 20, 25}
5. 2 варианта стратегий выбора оптимального дерева после серии отсечений

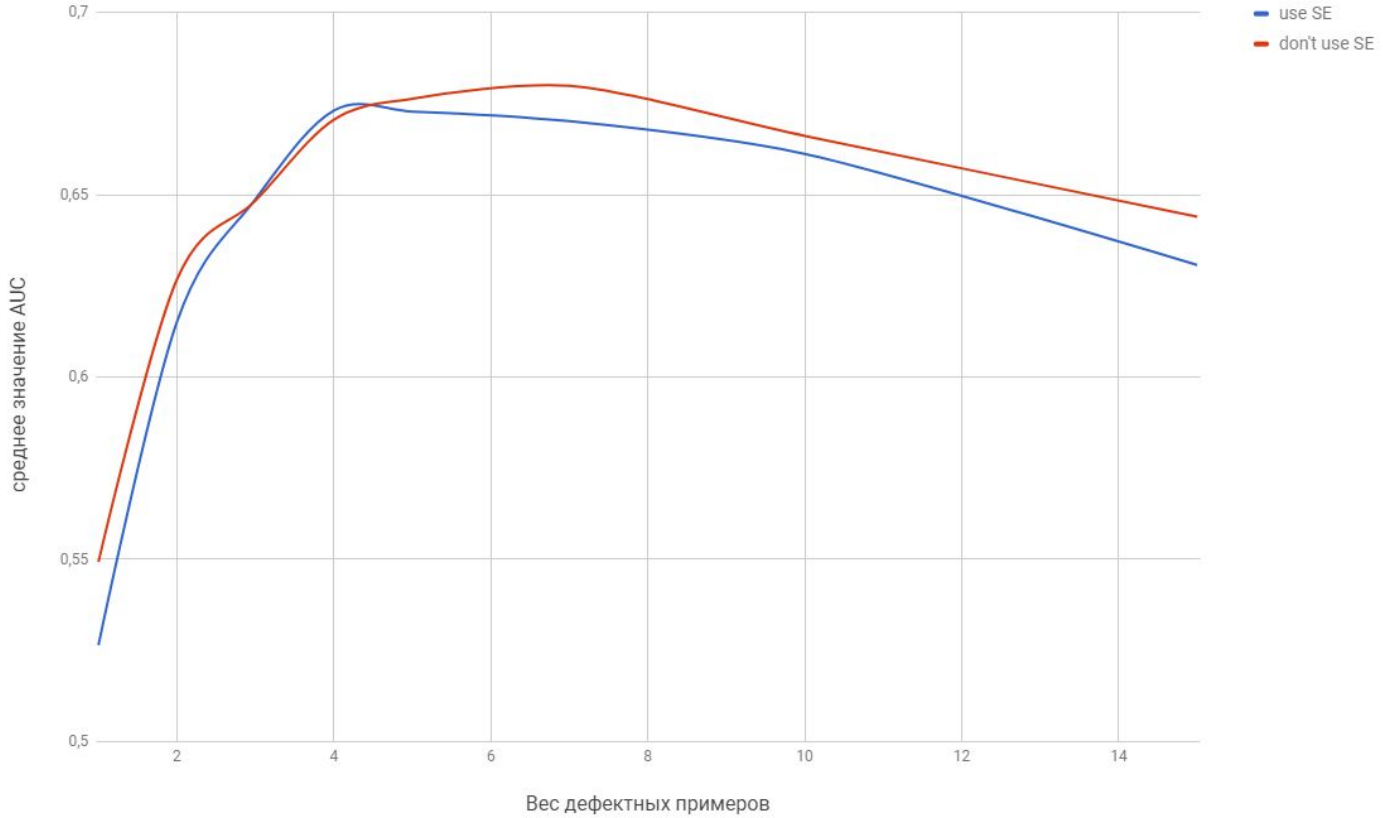
Количество итераций процедуры отсечений



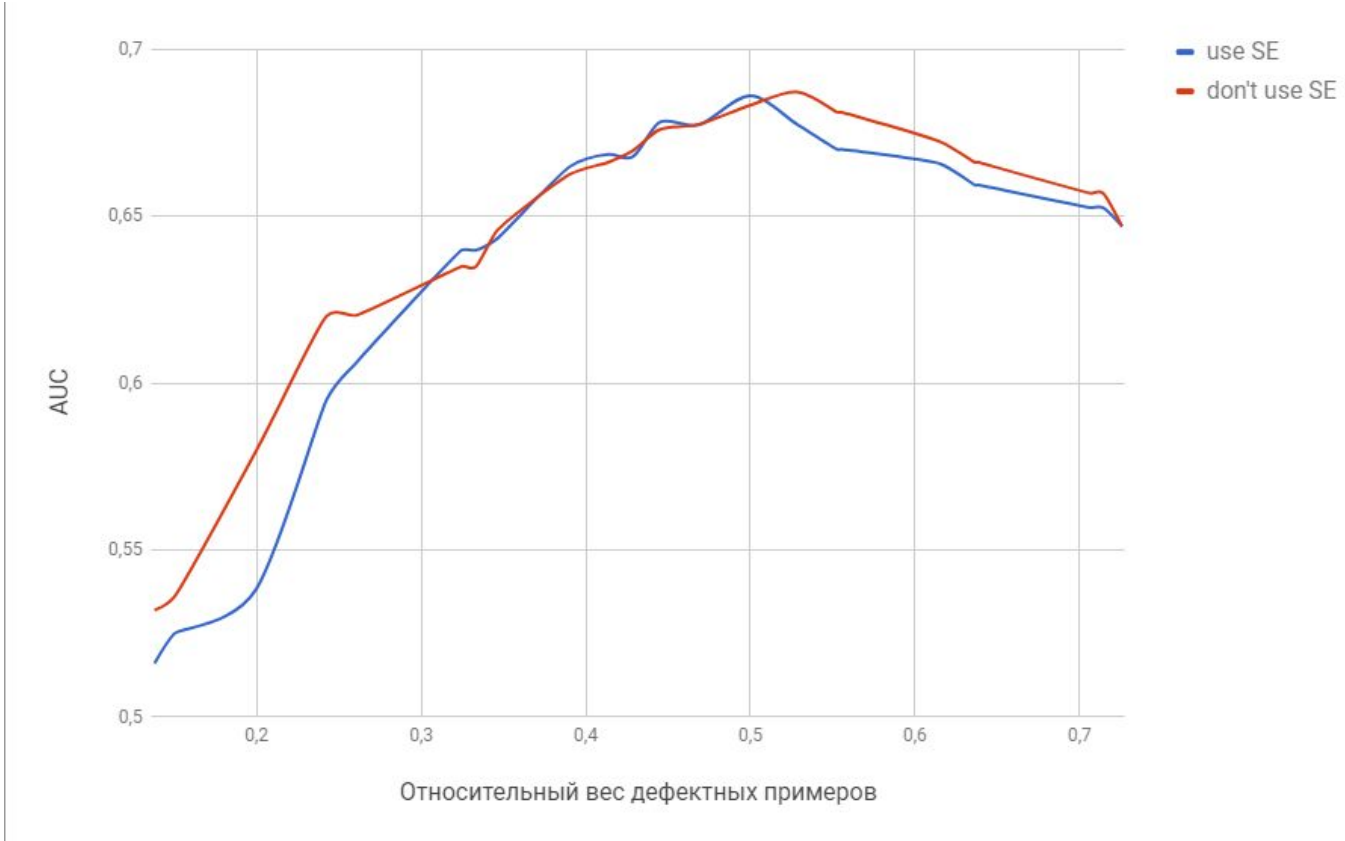
Минимальный вес листа



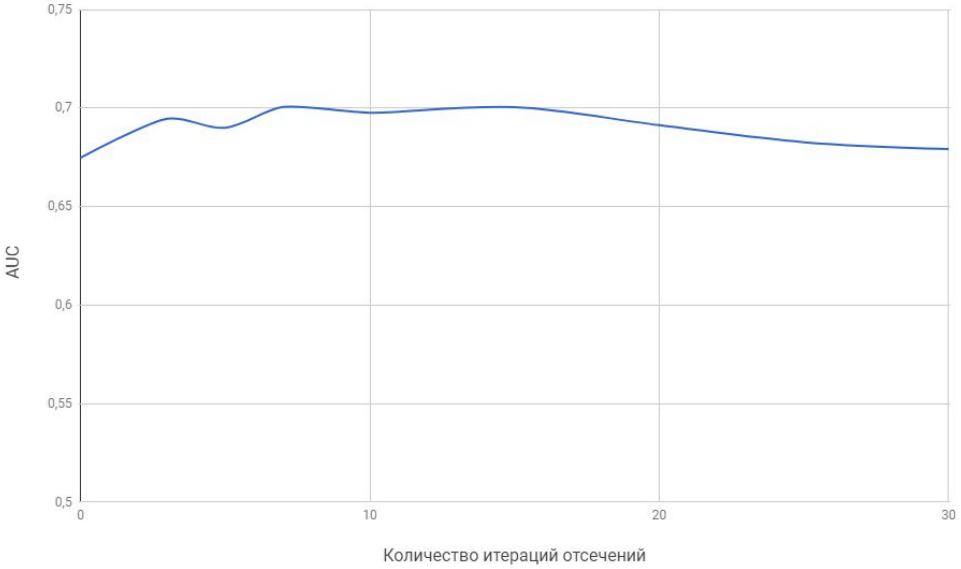
Вес дефектным примеров

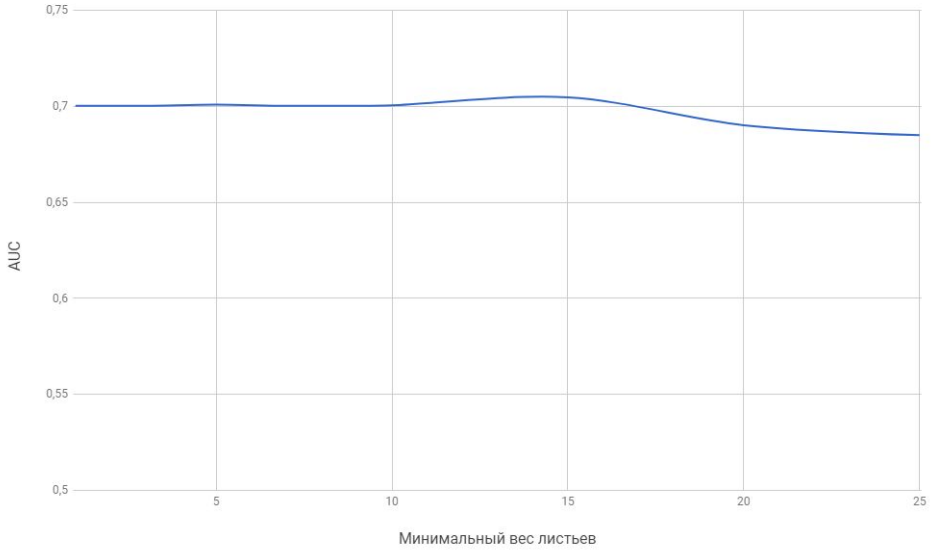


Относительный вес дефектных примеров



Результаты подбора





Результаты обучения

Для выбранного набора параметров:

- Точность - 67%
- Количество распознанных дефектных методов - 74%
- Значение AUC - 0.700

При этом максимальные замеченные значения:

- AUC - 0.707
- Точность - 87%, при условии распознавания минимум 50% ошибок - 77%

Результаты обучения

Пример получающегося правила:

(if *total_loc* > 21.5) -> (if *total_loc* < 92.5) -> (if *design_density* > 0.025) ->

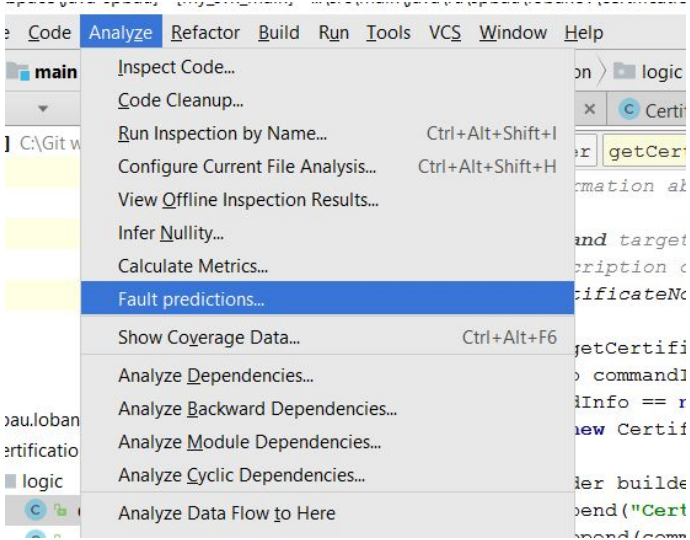
(if *condition_count* > 2.5) -> (if *normalized_cyclomatic_complexity* > 0.175) ->

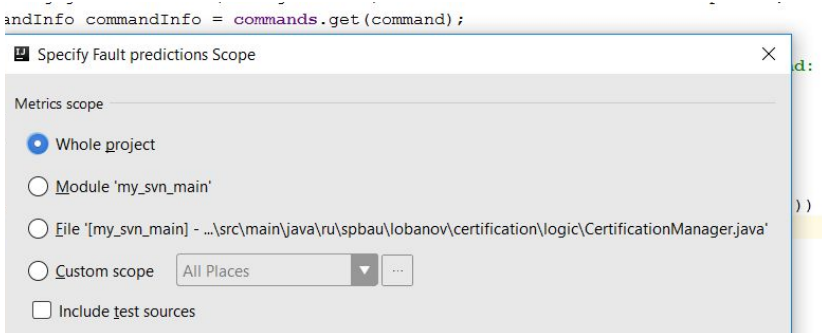
(if *formal_parameters* < 1.5) -> (if *cyclomatic_complexity* > 6.5) ->

(if *design_complexity* < 19.5) -> (if *executable_loc* > 43.5) -> *defect* = 0.839

Плагин

Функционал плагина: можно запустить поиск ошибок, указать какие файлы надо проверять. В конце работы плагин показывает таблицу с результатами..





Плагин

```
*/
public String getCertificate(String command) throws CertificateNotFountException {
    CommandInfo commandInfo = commands.get(command);
    if (commandInfo == null) {
        throw new CertificateNotFountException("Certificate wasn't found for command: " + command);
    }
    StringBuilder builder = new StringBuilder();
    builder.append("Certificate for command: ")
            .append(commandInfo.getName())
            .append("\n\t\t");
    builder.append(commandInfo.getGeneralInfo().replaceAll(LINE_SEPARATOR, "\n\t\t"))
            .append('\n');
    if (!commandInfo.getArguments().isEmpty()) {
        builder.append("\tArguments:\n");
        for (ArgumentInfo argument : commandInfo.getArguments()) {
            builder.append("\t\t")
                    .append(argument.getArgumentName())
                    .append(" - ")
                    .append(argument.getDescription().replaceAll(LINE_SEPARATOR, "\n\t\t\t"))
                    .append('\n');
        }
    }
    if (!commandInfo.getArguments().isEmpty()) {
        builder.append("\tTips:\n");
        for (Tip tip : commandInfo.getTips()) {
            builder.append("\t\t- ")
                    .append(tip.getMessage().replaceAll(LINE_SEPARATOR, "\n\t\t"))
                    .append('\n');
        }
    }
```

Fault predictions window Fault predictions result

Method	Defectiveness	Comment
ru.spbau.lobanov.certification.logic.CertificationManager.CertificateNotFountException.CertificateNotFountException(String)	2	no comment
ru.spbau.lobanov.certification.logic.CertificationManager.CertificationManager(String)	7	no comment
ru.spbau.lobanov.certification.logic.CertificationManager.formatCommand(CommandInfo)	2	no comment
ru.spbau.lobanov.certification.logic.CertificationManager.getCertificate(String)	8	no comment
ru.spbau.lobanov.certification.logic.CertificationManager.getCommands()	6	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.CertificationFormatException.CertificationFormatException(String)	2	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.CertificationParser(int)	2	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.parse(InputStream)	3	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.parse(String)	2	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.parseArgument(Scanner)	2	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.parseCommand(Scanner)	1	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.parseMessage(Scanner)	3	no comment
ru.spbau.lobanov.certification.logic.CertificationParser.parseTip(Scanner)	2	no comment

Fault prediction succeed

Analysis completed

Fault prediction succeed: Analysis completed (moments ago)

9:1 CRLF UTF-8 Git: task7

Результаты

- Найдена и обработана обширная обучающая выборка
- Подобраны параметры обучения
- Создан классификатор
- Реализован плагин

https://github.com/ArtyomLobanov/MetricsReloaded/tree/fault_predictions

https://docs.google.com/spreadsheets/d/1YO7coU5YrwcQ7j0AfPXuueIpXI16B_fP6kANeJ4Ynmo/edit?usp=sharing