A Model to Detect Readability Improvements in Incremental Changes

Devjeet Roy + Sarah Fakhoury + John Lee + Venera Arnaoudova @ Washington State University

Key Results

We developed a machine learning model that is able to correctly classify readability commits with a precision of 79.2% and recall of 67%

Readability improving commits are characterized by changes to existing lines of code rather than addition of new lines of code.

Our model outperforms existing SOTA approaches at detecting readability improvements in practice.

Source Code Readability

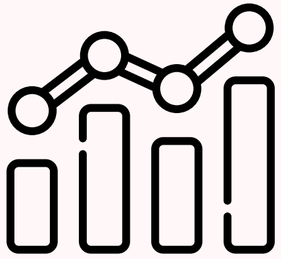
Fundamental aspect of software quality

- affects all parts of the software development process
- impacts program comprehension

One of the **most requested** metrics by software developers

Highly **subjective**

Several models have been proposed to measure readability



In Practice

Pantuichina et al. (ICSME '18) found that often developers perception of quality and source code metrics are not aligned

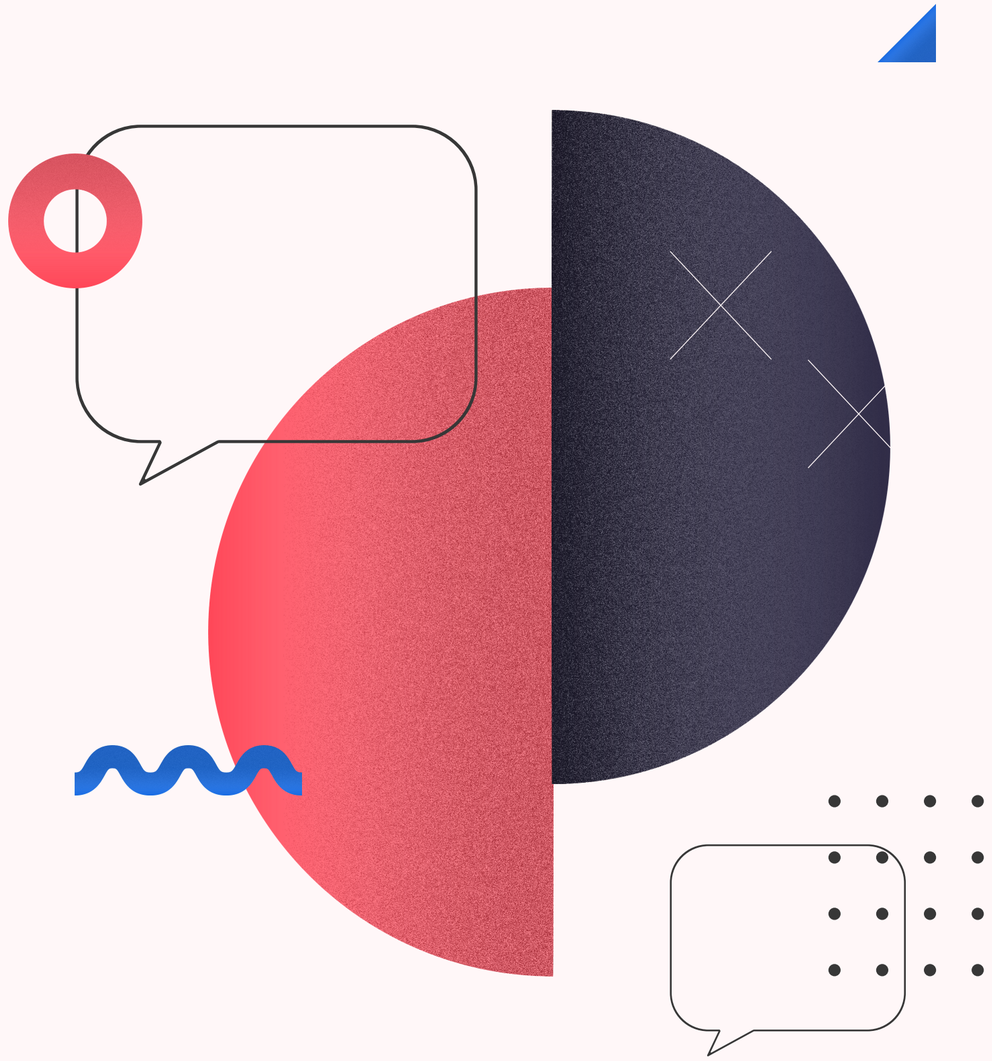
Fakhoury et al. (ICPC '19) found that state of the art readability models are not able to capture readability improvement in practice.

Current state of the art readability models are **not effective** at detecting readability improvements in **incremental changes**

Readability and incremental changes

Most of the everyday work performed by developers involves incremental changes

The ability to detect readability improvements in this scenario would allow us to conceive better tools to support software developers in their day to day work.



RQ 1

Can we use machine learning to capture readability improvements in practice?

Dataset

We use a dataset of **2,665 commits** from **76 engineered Java projects**

These commits consist of commits intended to improve readability as well as non-readability related commits

Each commit is manually validated and detangled

github

Metrics

- Traditional source code quality metrics such as Halstead Metrics.
- Fine grained code changes
- Poor programming practices/style conformance using PMD and Checkstyle
- Detailed line diff metrics using RSM

Model Building

We split the data randomly into train & test sets

10-fold cross validation used on the train set for model building process

Feature selection, model selection & hyperparameter optimization all performed automatically via cross-validation

Best configuration trained on entire training set, and evaluated on holdout set

RQ1 Results

Our model is able to classify readability improvements with a precision of **79.2%**, recall of **67%** and MCC of **0.39** (fair)

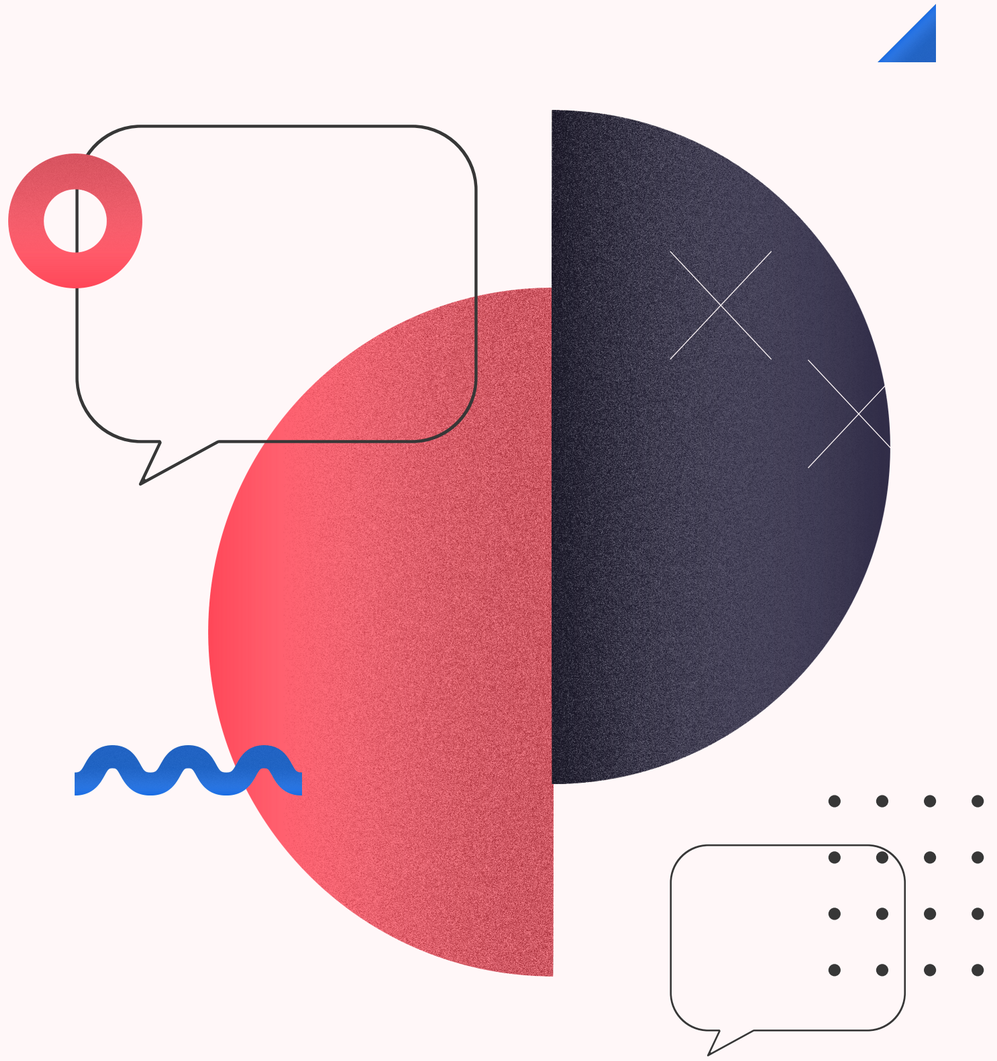
	CV	Test
Precision	69.4%	79.2%
Recall	59.3%	67%
F-1	63.9%	72.6%
Mathew's Correlation	0.41	0.39
ROC-AUC	70%	70%

RQ1 Results

Our model struggled with changes that affect only one line of code

```
-           inflater.inflate(R.menu.call_log_options_new, menu);
+           inflater.inflate(R.menu.call_log_options, menu);
```

In some instances, non-readability changes also included **secondary readability improvements**



RQ 2

What features align with developer's perception of readability improvements in practice?

SHapley Additive exPlanations (SHAP)

SHAP is a gametheoretic approach to measuring the contribution of each feature to the predictions made by a machine learning model

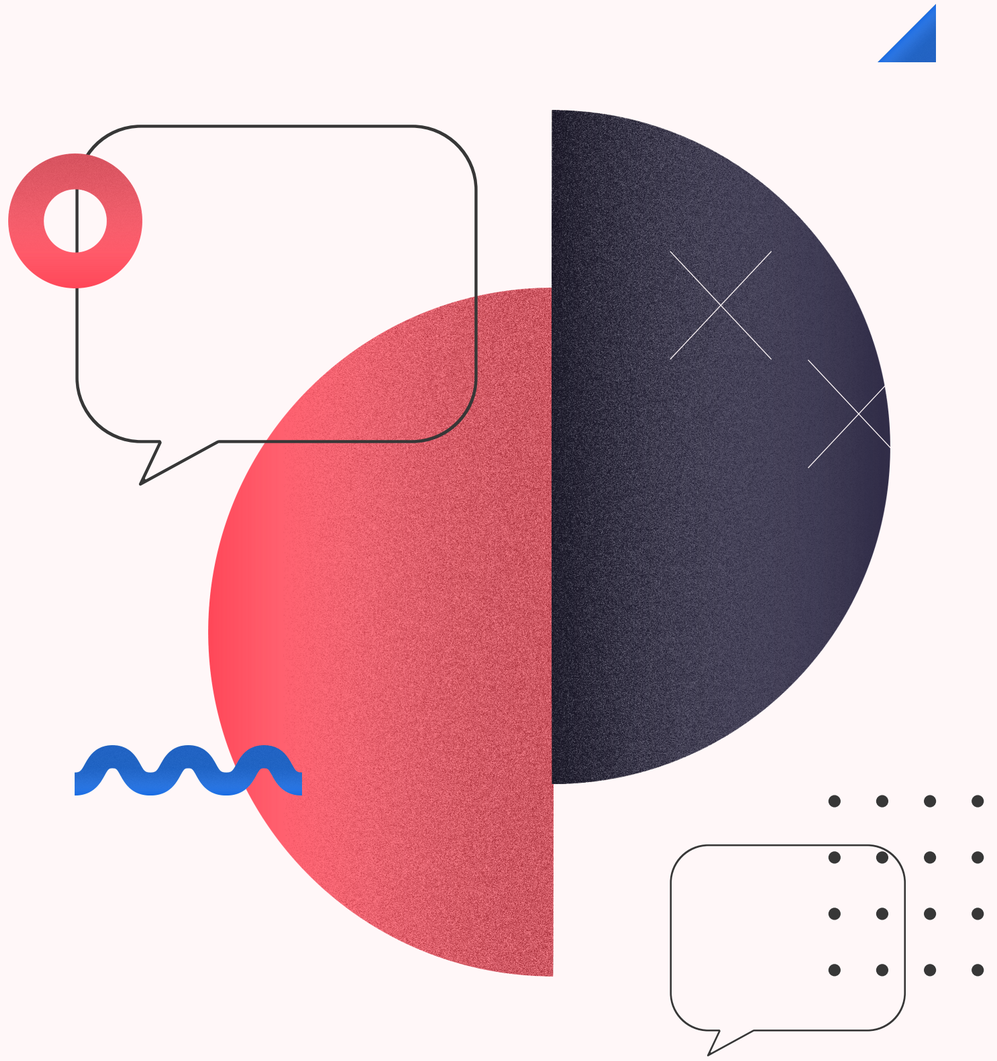
Able to identify **both the magnitude and direction** of the influence of a feature on a given prediction

For tree based methods, it provides an **exact algorithm** to determine the contribution of each feature of the inputs

RQ2 Results

Readability improving commits tend to make **changes to existing lines** of code rather than introducing new code.

Non-readability commits tend to introduce **new lines of code** and are associated with a degradation of certain software quality metrics (Halstead Effort, McCabe's Cyclomatic Complexity etc.)



RQ 3

How does the proposed model perform when compared to state of the art readability models?

State of the art readability models

Scalabrino's Model - uses metrics that measure quality of source code lexicon as a proxy for readability [1]

Dorn's Model - uses visual, spatial, alignment and linguistic aspects of the source code [2]

Combined Model - proposed by Scalabrino et al. as a combination of multiple state of the art readability models considering both linguistic and structural aspects of source code [1]

[1] Scalabrino, S., Linares-Vásquez, M., Oliveto, R. and Poshyvanyk, D., 2018. A comprehensive model for code readability. Journal of Software: Evolution and Process, 30(6), p.e1958.

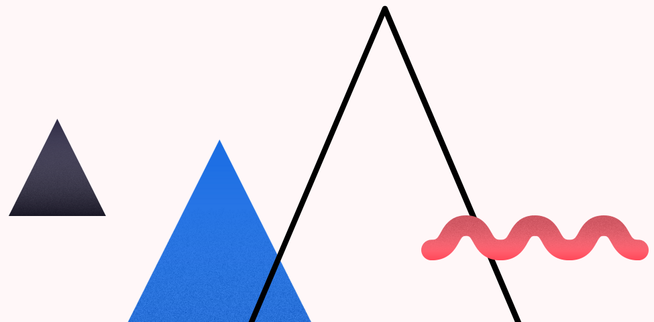
[2] Jonathan Dorn. A general software readability model. MCS Thesis. 2012.

RQ 3 Results

Overall, our model outperforms existing SOTA appraoches on all metrics

	Dorn	Scalabrino	Combined	Ours
Precision	60%	64%	62.2%	**79.2%**
Recall	41%	47.14%	37%	**67%**
F-1	49%	54.47%	46.96%	**72.6%**
Mathew's Correlation	-0.01	0.07	0.02	**0.39**
ROC-AUC	49.46%	53.62%	51.24%	**70%**

Moreover, all the SOTA models have MCC < 0.1(low), while our model attains **0.39 (fair)**.

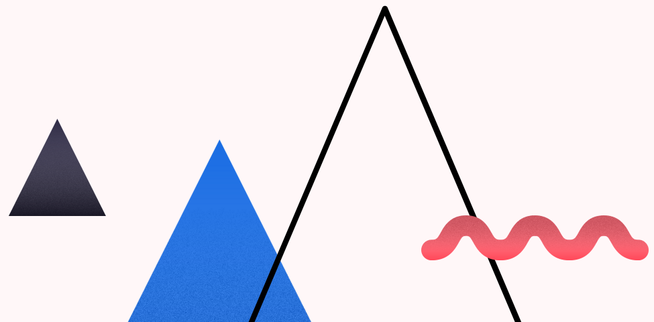


RQ 3 Results

Qualitative investigation shows that SOTA readability models struggled with readability improvements where **only comments** are changed.

SOTA models also did not check for **style violations** unlike our proposed model, which used data from Checkstyle and PMD

	Dorn	Scalabrino	Combined	Ours
Precision	60%	64%	62.2%	**79.2%**
Recall	41%	47.14%	37%	**67%**
F-1	49%	54.47%	46.96%	**72.6%**
Mathew's Correlation	-0.01	0.07	0.02	**0.39**
ROC-AUC	49.46%	53.62%	51.24%	**70%**



Thank You

RQ2 Results

This figure shows the SHAP values for the most important features (for test set predictions)

The shading of each point indicates the magnitude of the value of the feature, while its location on the x-axis indicates whether the contribution of the feature was positive or negative.