VALA

When you’re done with AI…

This is what you can do with ML

“A review of flaky test management approaches, with experimental validation of ML-based solution”

Introduction

My background

Bardhyl Shatri

The all around “Test Guy”

Specialized in test automation with field experience in test management and DevOps

- A test automation engineer with keen interest in AI/ML and software testing
- 5 years in the field within different industries
- Strong dislike towards flaky tests

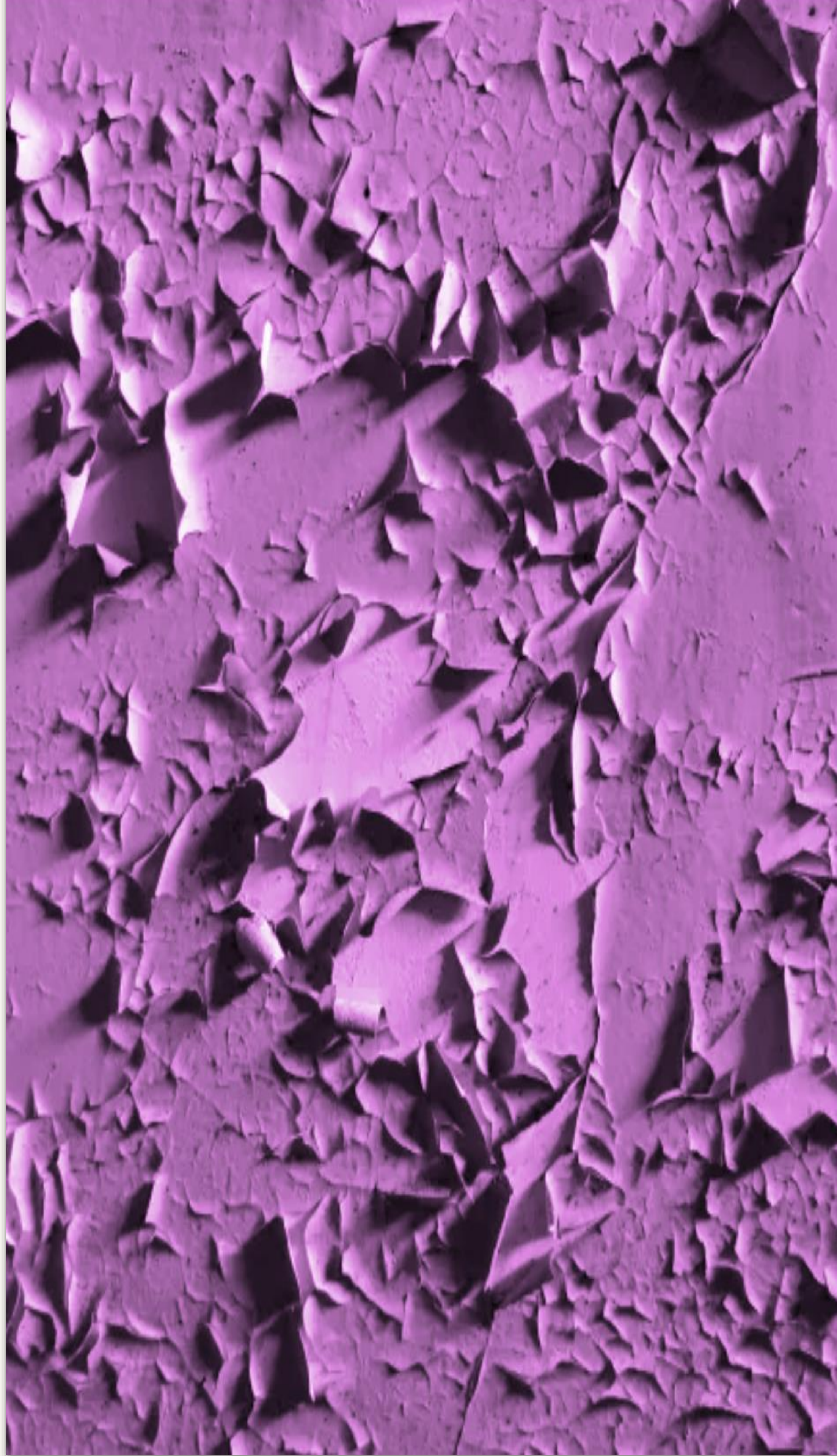
Table of Contents

VALA

Flaky tests

Why, how and when

VALA

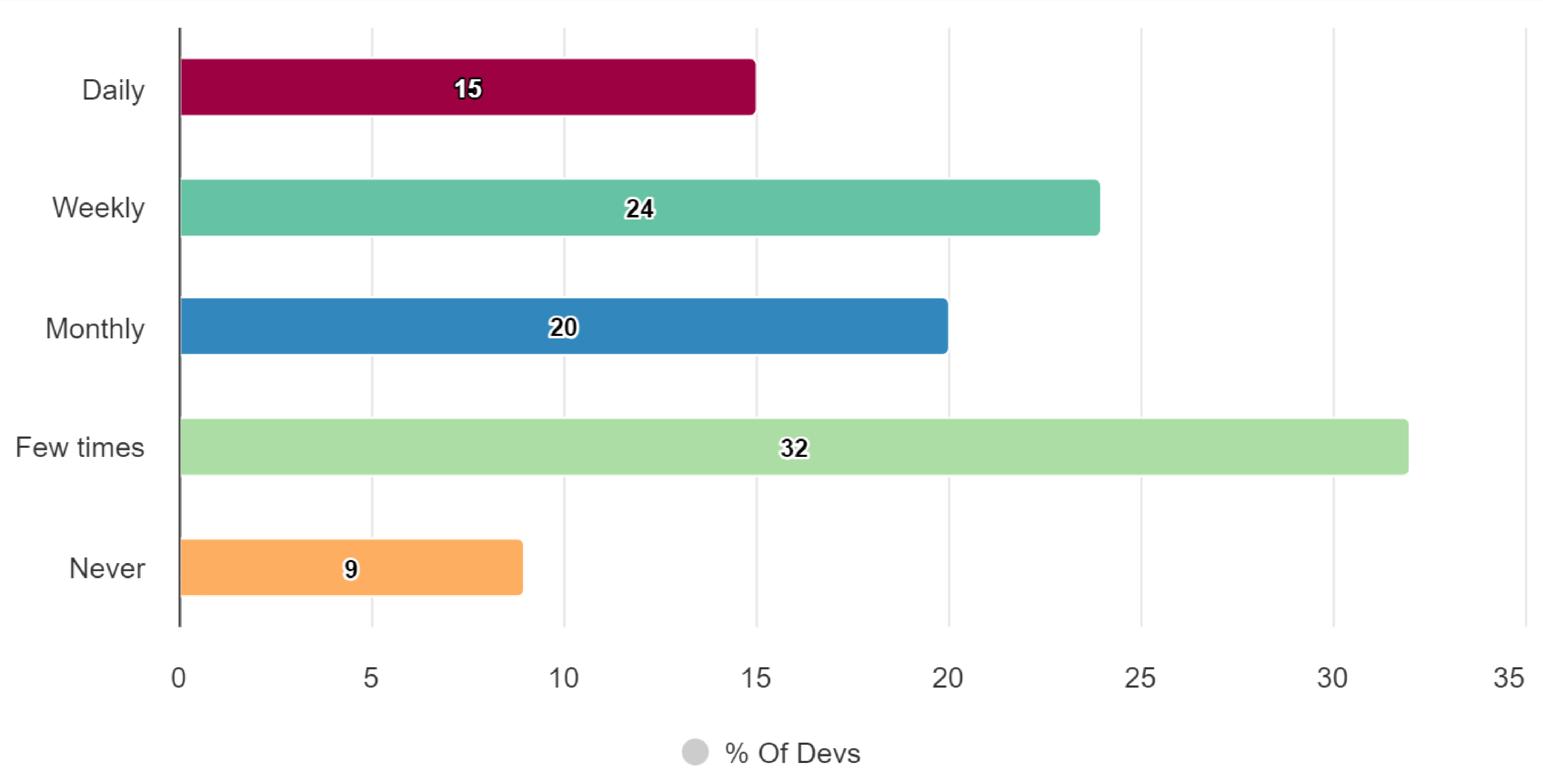


Flaky Tests

How often do they happen?

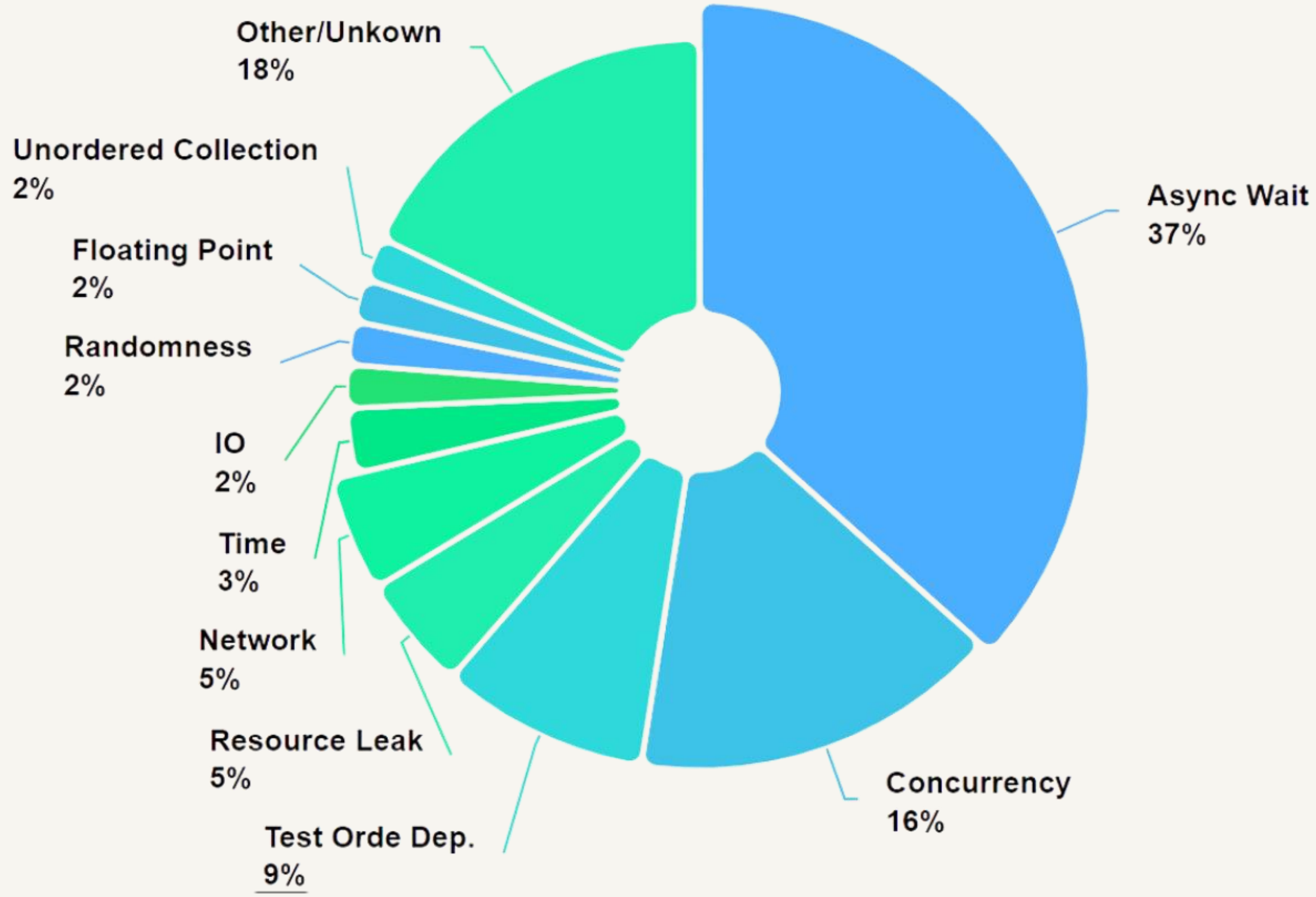
15% of developers encounter flakiness daily

24% on weekly basis



Flaky Tests

What causes flakiness?



Asynchronous Wait 37%

Concurrency 16%

Test Order Dependency 9%

Network 5%

Error: Timeout of 20ms exceeded.

Resource Leak 5%

Flaky Tests

The Consequences

Wastes developer time investigating false failures

Erodes trust in test suite reliability.

Leads to ignore test failures, potentially missing real bugs

Increases infrastructure costs due to reruns

Masks real bugs leading to false negatives

ML - Basic concepts

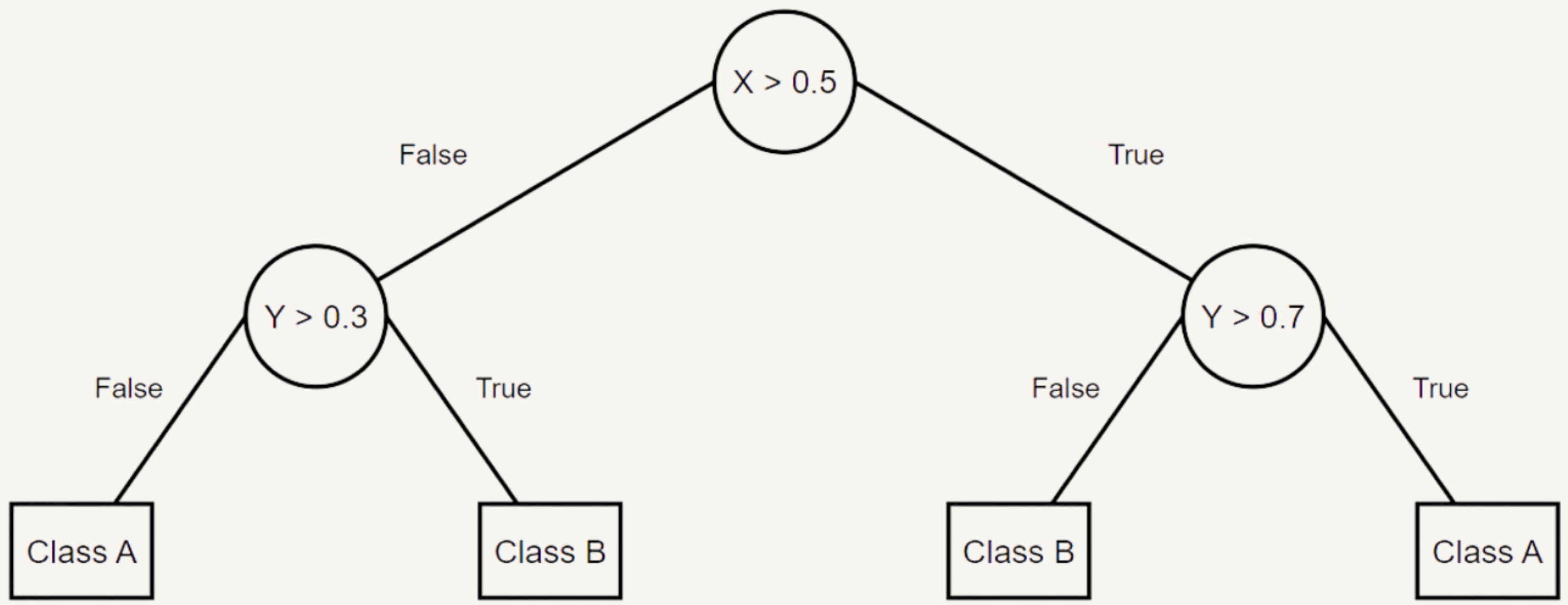
A brief overview on relevant concepts

Supervised learning classifiers

Decision Tree Classifier

Decision Tree Classifier

- Creates a flowchart-like structure where each node represents a decision based on features
- Easy to interpret and visualize
- Prone to overfitting on complex data



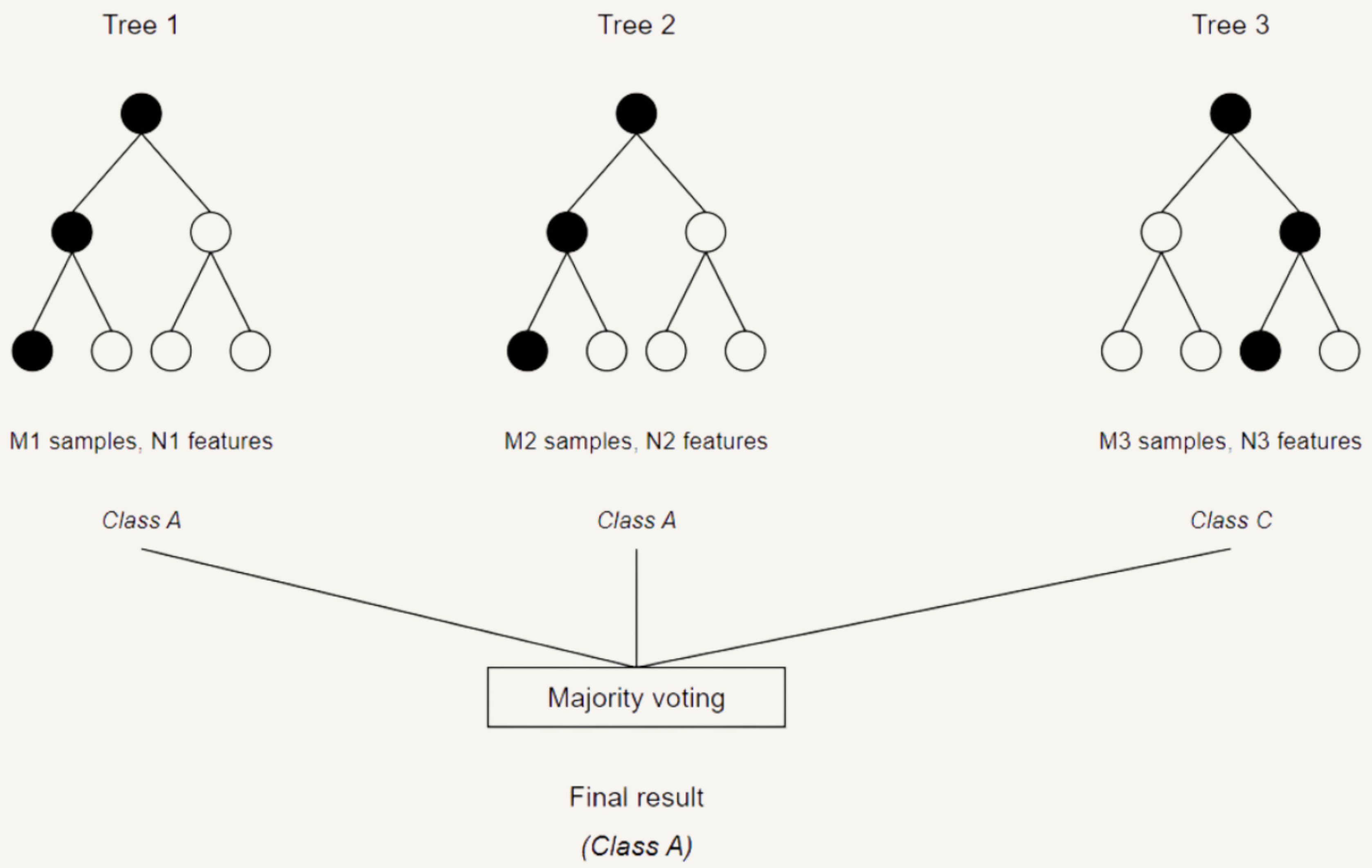
VALA

Supervised learning classifiers

Random Forest Classifier

Random Forest Classifier

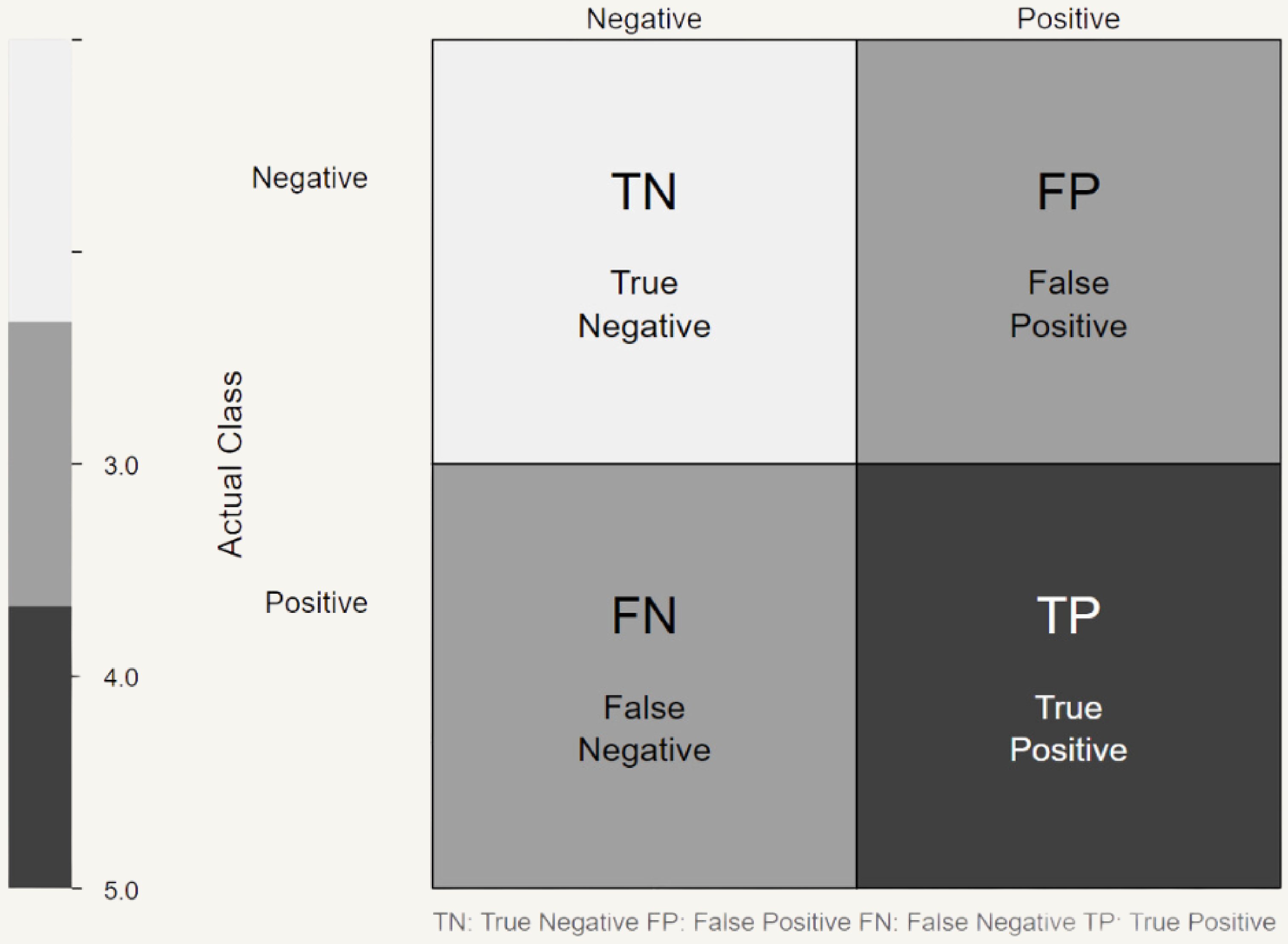
- Ensemble of decision trees using random subsets of features
- Good balance of accuracy and overfitting resistance
- Works well for many types of problems



Confusion Matrix

Shows how well a classification model performs

- True Positives (TP): Correctly identified positive cases
- True Negatives (TN): Correctly identified negative cases
- False Positives (FP): Incorrectly labeled as positive
- False Negatives (FN): Incorrectly labeled as negative



Evaluation Metrics

Accuracy

$$\frac{TP + TN}{TP+TN+FP+FP}$$

Out of all predictions made, how many were true?

Precision

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

Out of all positive predictions made, how many were true?

Recall

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

Out of all the data points that should be predicted as true, how many did we correctly predict as true?

F1-score

$$\frac{2 \times (\text{Precision} \times \text{Recal})}{\text{Precision} + \text{Recall}}$$

More widely used accuracy metric which combines precision and recall

Stratified K-Fold Cross-Validation

Stratified K-Fold Cross-Validation maintains the same percentage of samples for each class across all folds as found in the complete dataset

This advanced cross-validation technique is particularly effective for unbalanced datasets, ensuring fair and reliable training and validation processes

ML - Predicting flakiness

Data, features and training

VALA

ML - Predicting flakiness

Practical Flaky Test Prediction using Common Code Evolution and Test History Data - 2023

In 2023 a case study was conducted in a real-world environment.

Simple approach using only commonly available data:

Test run and version control history.

ML - Predicting flakiness

Practical Flaky Test Prediction using Common Code Evolution and Test History Data - 2023

"...We trained several established classifiers on the suggested features and evaluated their performance on a large-scale industrial software system, from which we collected a data set of 100 flaky and 100 non-flaky test- and code-histories.

The best model was able to achieve an F1-score of 95.5 % using only 3 features..."

Feature extraction

Test history

Test duration

Mean test execution time of all test executions

Long running tests are more likely to emit flaky behavior

Mean diff PASS/FAIL duration

Difference between the mean duration of passing and failing runs

Sniff out test failures that occur due to fast failure or waiting out timeout

Feature extraction

Test history

Flip Rate

How many times the test outcome changes between test runs

“Google reports that about 84 % of the transitions they observed from pass to fail involved a flaky test”

Flip Rate Decay Functions

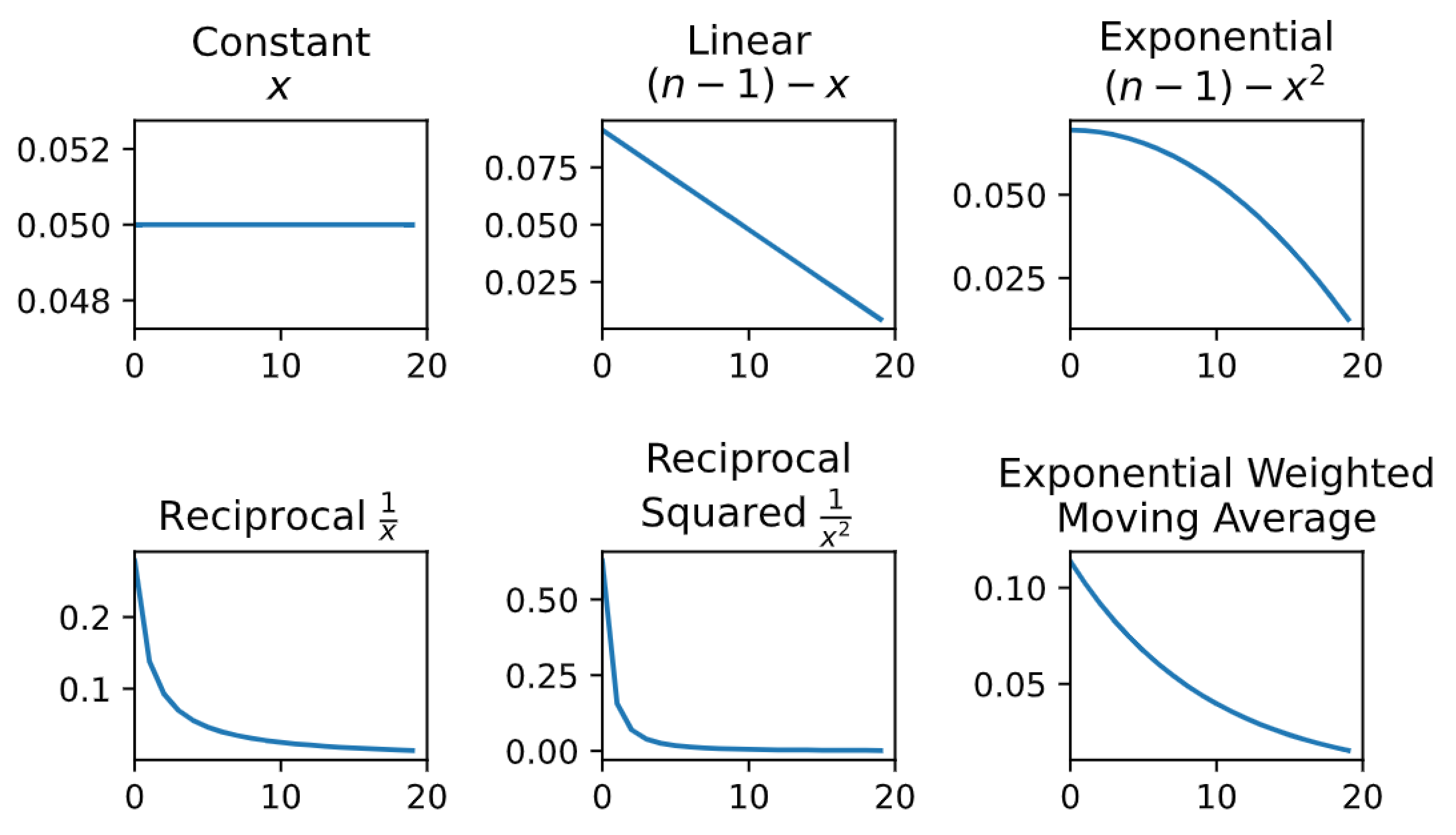
How much weight/importance to put on the most recent runs compared to older test runs

What if a test has been flaky before but is now stable?

~70% of tests are flaky from the start

Feature extraction

Flip rate - Decay functions



Feature extraction

Version Control

File Extension

Group files by extension or other characteristic pattern in filename

“.java”, “.gitignore”, “README”

How changes in different files are associated with flakiness

If only README changed compared to files with business logic

Changes made in 3, 14 and 54 days

Given a git revision that has changed “.java” and “.py” files

Get the changes done on the files with same extensions in the past

How development speed on those files correlates to flakiness

Feature extraction

Version Control

Number of modified files in current PR

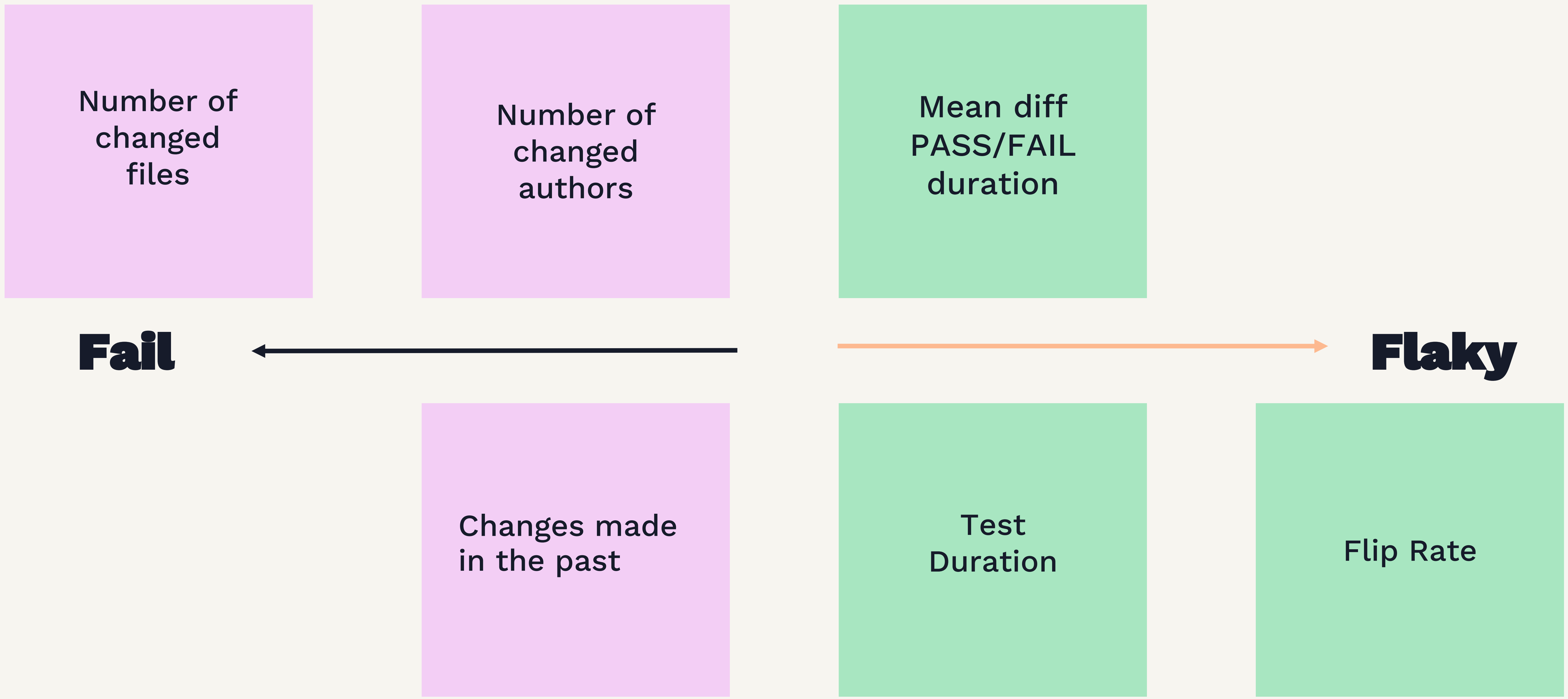
The more files have changed the more likely it is that a test failure is genuine

Number of authors of the current PR

The more authors on a given PR
the higher the chance the test failure is genuine

Feature extraction

Feature impact



The data

Java Websockets - Open source project

For training our model we have a dataset of test runs with **30 reruns** across **75 commits**

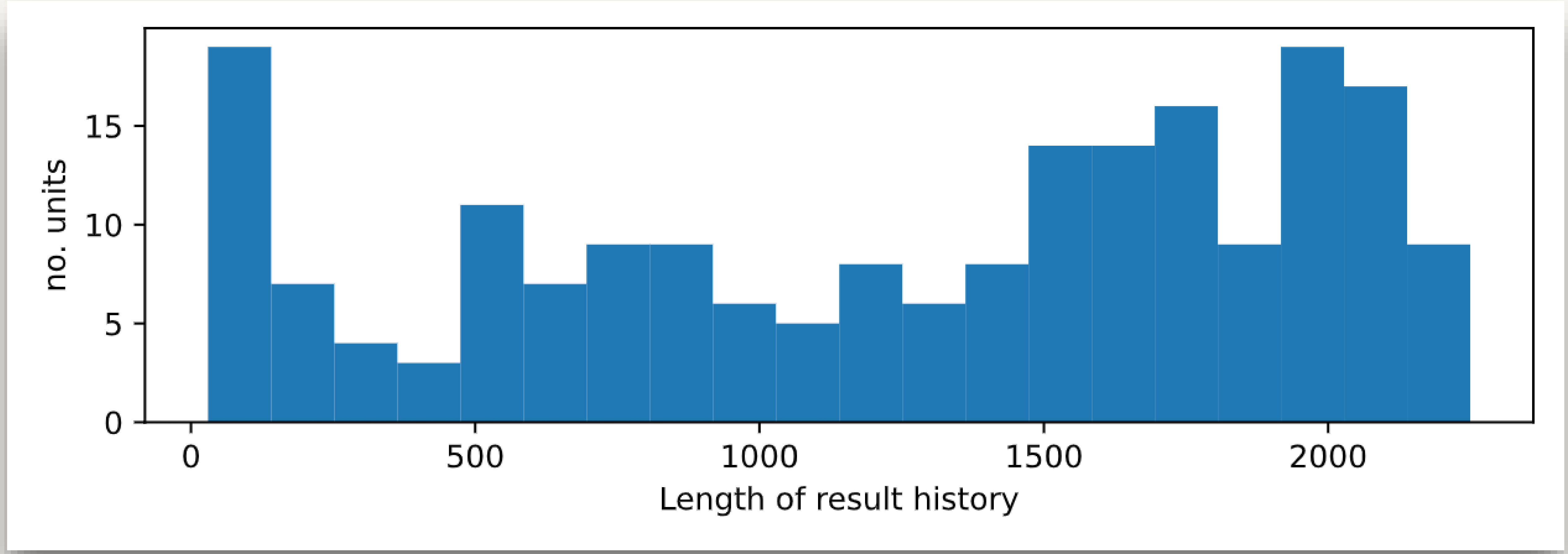
The open source Java WebSockets project has **146 unique tests**

In total we'll have 30 x 75 x 146 **= 328 500 test runs**

Data sampling

Java Websockets - Open source project

We collect a dataset of **100 non-flaky** tests and **100 flaky tests**



Training the model

We train the model using different

- Combinations of feature sets
  - **12** different sets of features

- Decay functions for the flip-rate feature
  - **6** of the 12 features sets are different flip rate decay functions

- Supervised learning classifiers
  - **0** different well established classifiers

Total number of combinations: **120**

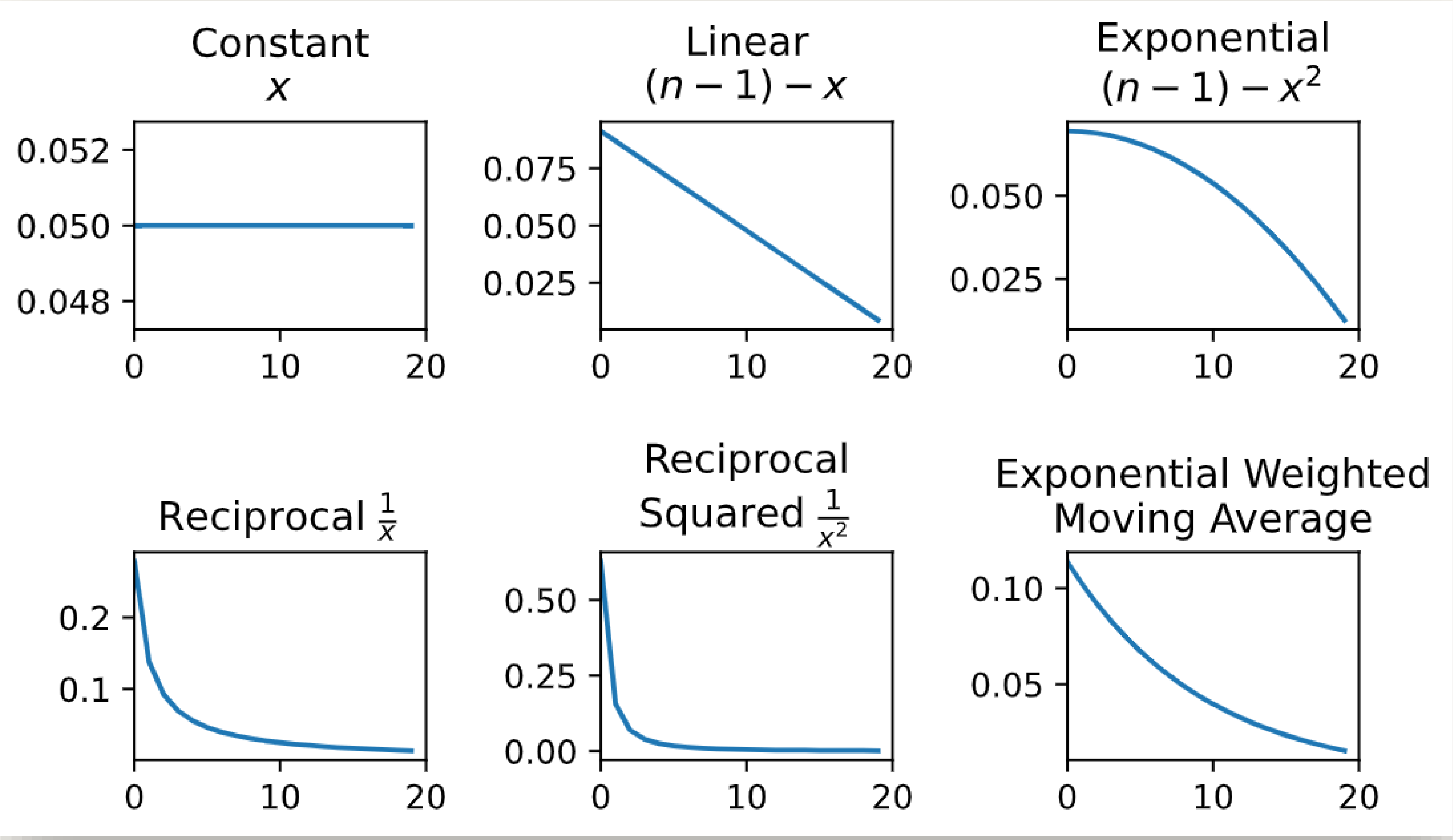
ML - Results

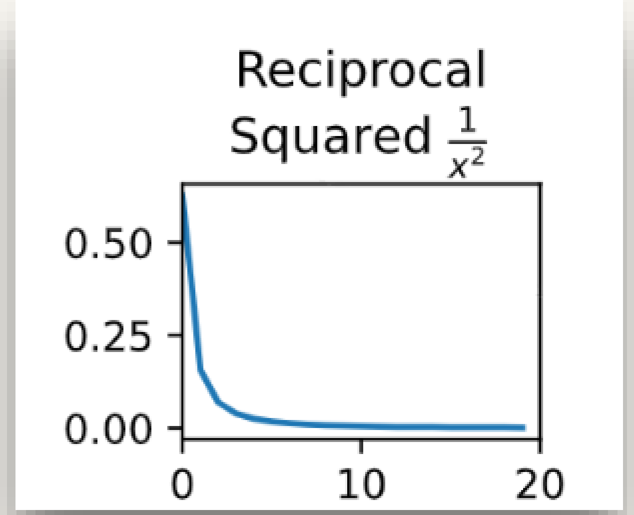
Top classifiers, feature sets and decay functions

Best performing models

Best performing Decay Functions

Reciprocal Squared





VALA

Best performing models

Best performing classifiers	Mean F1-Score across all sets
1. Decision Tree Depth 1	F1-Score: **0.907506**
2. Random Forest Classifier	F1-Score: **0.882765**
3. AdaBoost Classifier	F1-Score: **0.877389**

Best performing models

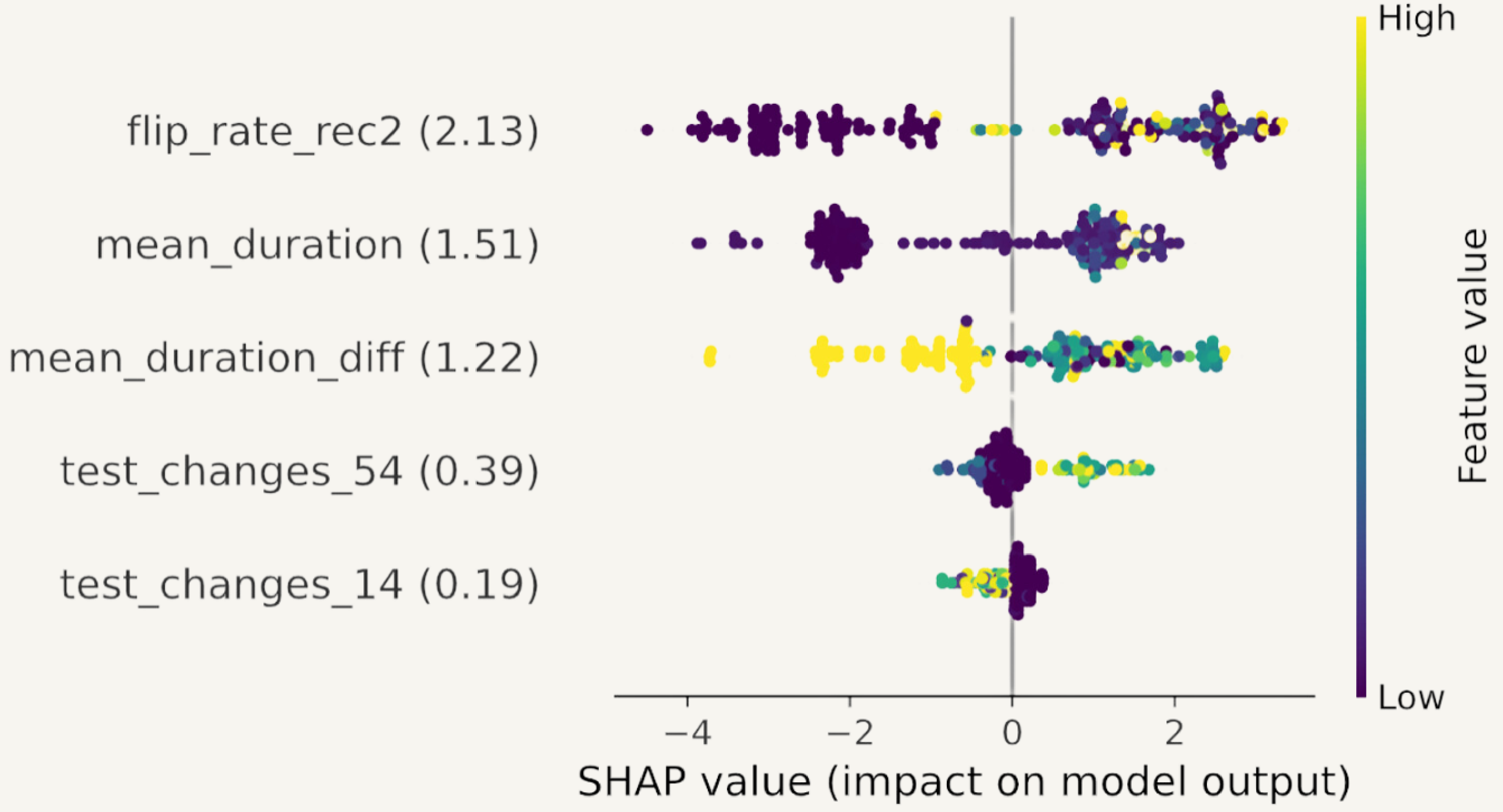
Using the best performing classifier - **Decision Tree Depth 1**

Best set of features	Mean F1-Score
1. All features together	F1-Score: **0.910032**
2. Flip-rate **+** Test duration **+** Mean Diff	F1-Score: **0.910032**
3. All features **-** Test duration **-** Mean Diff	F1-Score: **0.904981**

Feature impact

Feature set: **All features together**

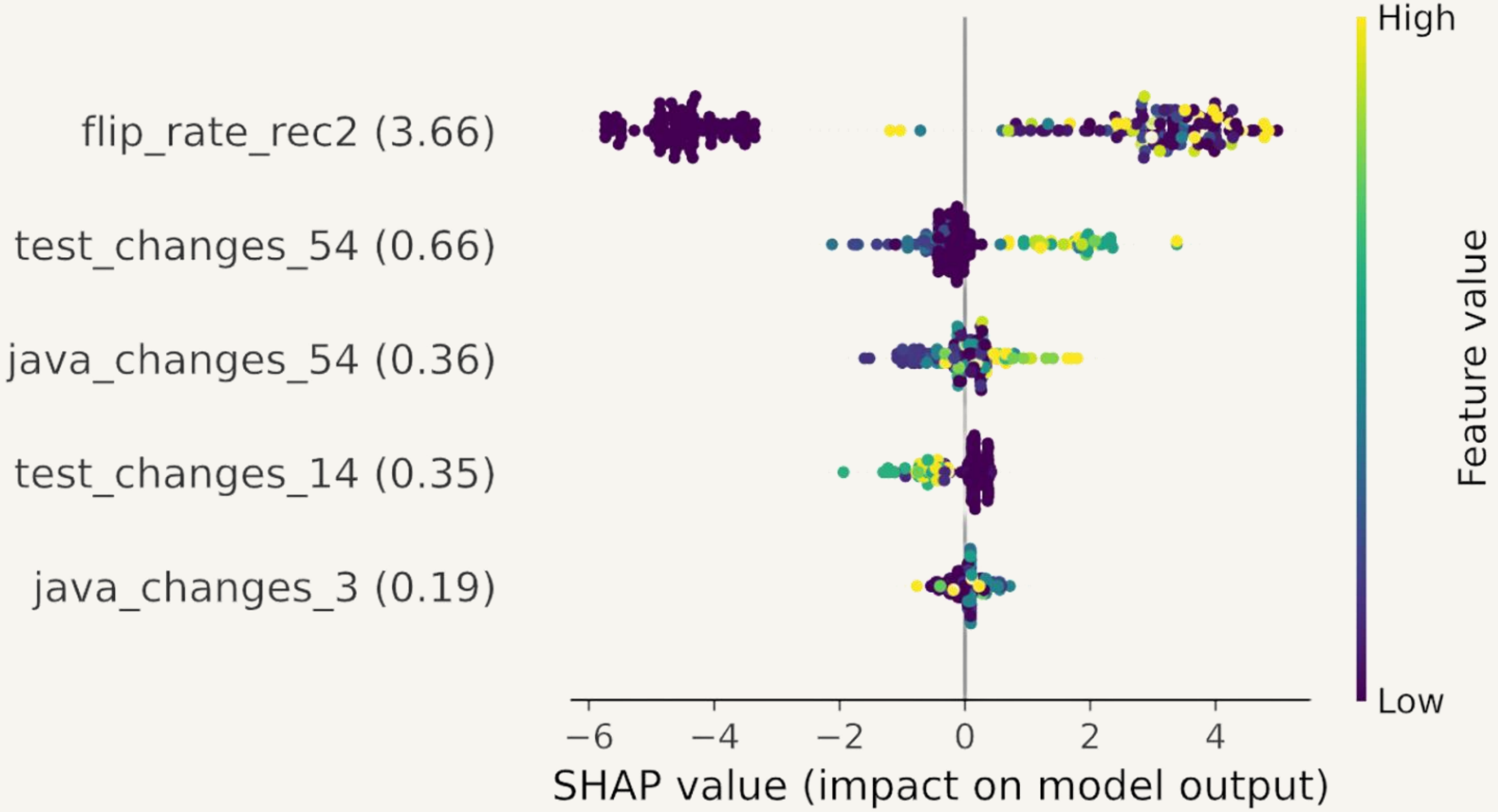
Top 5 features



Feature impact

Feature set: **All features - Test duration - Mean Diff**

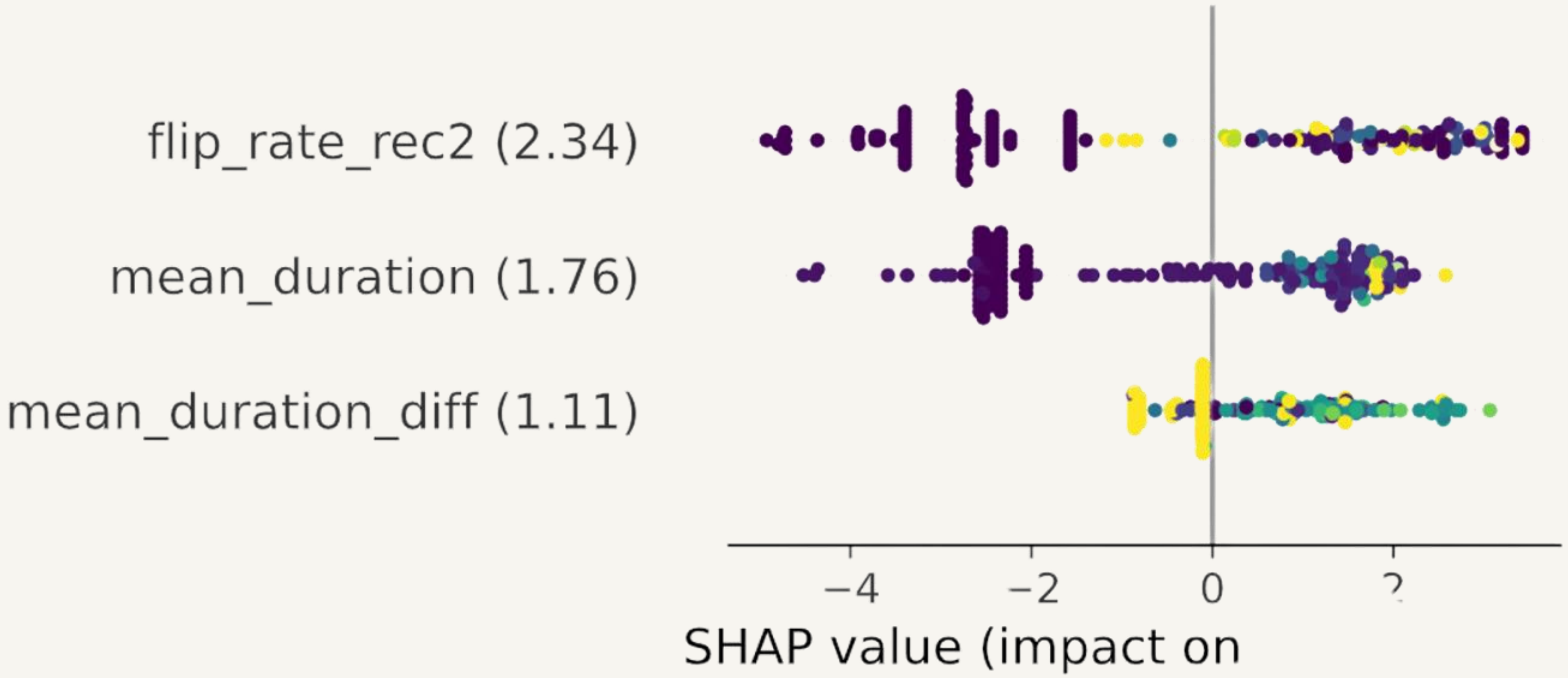
Top 5 features



Feature impact

Feature set: **Flip-rate + Test duration + Mean Diff**

Top 5 features



VALA

Interpretation of results

All top performers performed very well

Most impact in all sets made the **flip-rate**

More interestingly Test history alone performed as well as the full set with our best classifier **Decision Tree Depth 1**

Thread of validity

Replication dataset contains only one genuine failure

Over representation of trivial non-flaky samples

No representation of genuine failures in between flaky failures

Final thoughts

Q&A