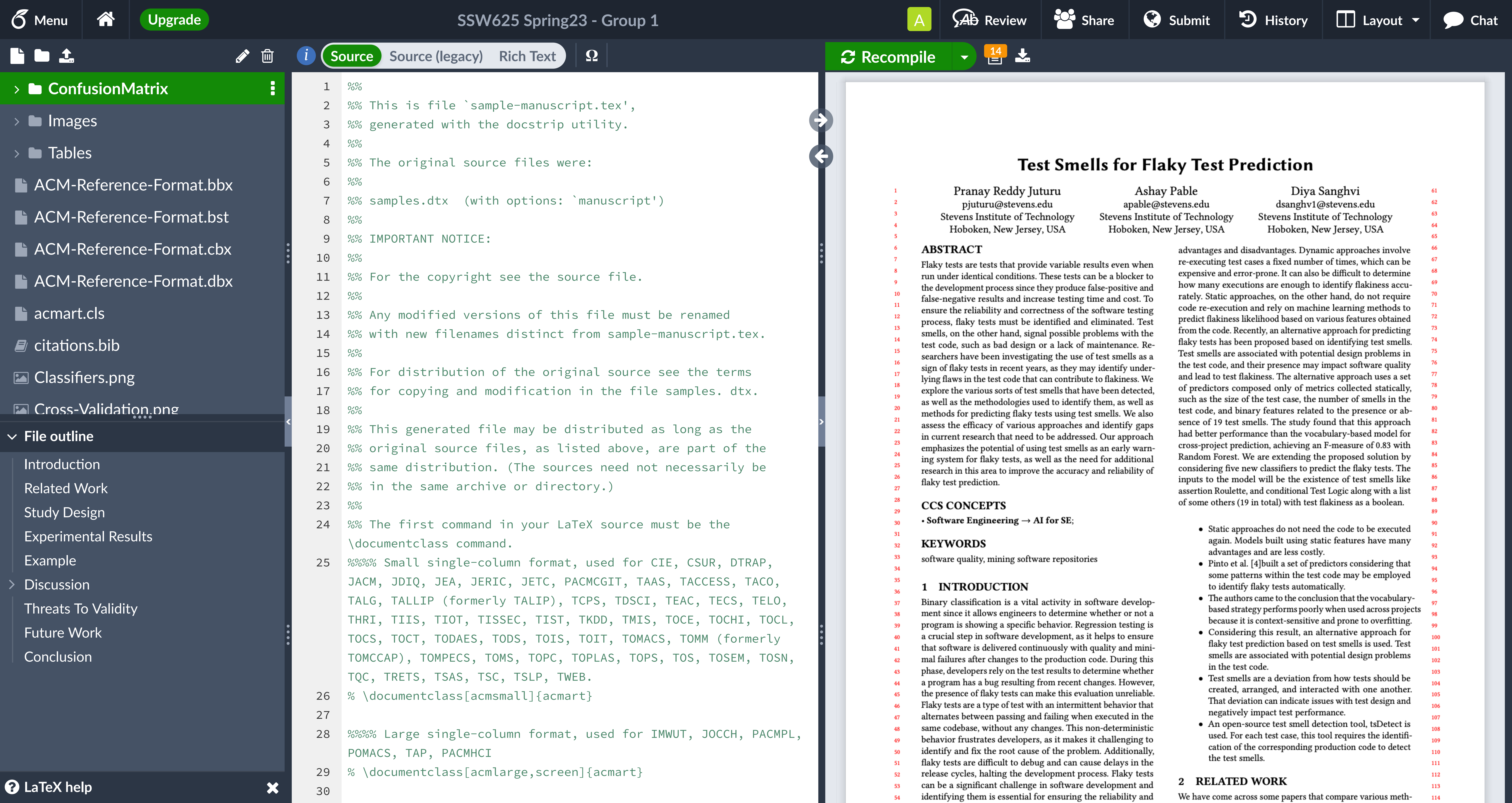

```
%%
%% This is file `sample-manuscript.tex',
%% generated with the docstrip utility.
%%
%% The original source files were:
%%
%% samples.dtx  (with options: `manuscript')
%%
%% IMPORTANT NOTICE:
%%
%% For the copyright see the source file.
%%
%% Any modified versions of this file must be renamed
%% with new filenames distinct from sample-manuscript.tex.
%%
%% For distribution of the original source see the terms
%% for copying and modification in the file samples. dtx.
%%
%% This generated file may be distributed as long as the
%% original source files, as listed above, are part of the
%% same distribution. (The sources need not necessarily be
%% in the same archive or directory.)
%%
%% The first command in your LaTeX source must be the \documentclass command.
%%%% Small single-column format, used for CIE, CSUR, DTRAP, JACM, JDIQ, JEA, JERIC, JETC, PACMCGIT, TAAS, TACCESS, TACO, TALG, TALLIP (formerly TALIP), TCPS, TDSCI, TEAC, TECS, TELO, THRI, TIIS, TIOT, TISSEC, TIST, TKDD, TMIS, TOCE, TOCHI, TOCL, TOCS, TOCT, TODAES, TODS, TOIS, TOIT, TOMACS, TOMM (formerly TOMCCAP), TOMPECS, TOMS, TOPC, TOPLAS, TOPS, TOS, TOSEM, TOSN, TQC, TRETS, TSAS, TSC, TSLP, TWEB.
% \documentclass[acmsmall]{acmart}

%%%% Large single-column format, used for IMWUT, JOCCH, PACMPL, POMACS, TAP, PACMHCI
% \documentclass[acmlarge,screen]{acmart}

```

Test Smells for Flaky Test Prediction

Pranay Reddy Juturu
pjuturu@stevens.edu
Stevens Institute of Technology
Hoboken, New Jersey, USA

Ashay Pable
apable@stevens.edu
Stevens Institute of Technology
Hoboken, New Jersey, USA

Diya Sanghvi
dsanghv1@stevens.edu
Stevens Institute of Technology
Hoboken, New Jersey, USA

ABSTRACT

Flaky tests are tests that provide variable results even when run under identical conditions. These tests can be a blocker to the development process since they produce false-positive and false-negative results and increase testing time and cost. To ensure the reliability and correctness of the software testing process, flaky tests must be identified and eliminated. Test smells, on the other hand, signal possible problems with the test code, such as bad design or a lack of maintenance. Researchers have been investigating the use of test smells as a sign of flaky tests in recent years, as they may identify underlying flaws in the test code that can contribute to flakiness. We explore the various sorts of test smells that have been detected, as well as the methodologies used to identify them, as well as methods for predicting flaky tests using test smells. We also assess the efficacy of various approaches and identify gaps in current research that need to be addressed. Our approach emphasizes the potential of using test smells as an early warning system for flaky tests, as well as the need for additional research in this area to improve the accuracy and reliability of flaky test prediction.

CCS CONCEPTS

• **Software Engineering → AI for SE**;

KEYWORDS

software quality, mining software repositories

1 INTRODUCTION

Binary classification is a vital activity in software development since it allows engineers to determine whether or not a program is showing a specific behavior. Regression testing is a crucial step in software development, as it helps to ensure that software is delivered continuously with quality and minimal failures after changes to the production code. During this phase, developers rely on the test results to determine whether a program has a bug resulting from recent changes. However, the presence of flaky tests can make this evaluation unreliable. Flaky tests are a type of test with an intermittent behavior that alternates between passing and failing when executed in the same codebase, without any changes. This non-deterministic behavior frustrates developers, as it makes it challenging to identify and fix the root cause of the problem. Additionally, flaky tests are difficult to debug and can cause delays in the release cycles, halting the development process. Flaky tests can be a significant challenge in software development and identifying them is essential for ensuring the reliability and

advantages and disadvantages. Dynamic approaches involve re-executing test cases a fixed number of times, which can be expensive and error-prone. It can also be difficult to determine how many executions are enough to identify flakiness accurately. Static approaches, on the other hand, do not require code re-execution and rely on machine learning methods to predict flakiness likelihood based on various features obtained from the code. Recently, an alternative approach for predicting flaky tests has been proposed based on identifying test smells. Test smells are associated with potential design problems in the test code, and their presence may impact software quality and lead to test flakiness. The alternative approach uses a set of predictors composed only of metrics collected statically, such as the size of the test case, the number of smells in the test code, and binary features related to the presence or absence of 19 test smells. The study found that this approach had better performance than the vocabulary-based model for cross-project prediction, achieving an F-measure of 0.83 with Random Forest. We are extending the proposed solution by considering five new classifiers to predict the flaky tests. The inputs to the model will be the existence of test smells like assertion Roulette, and conditional Test Logic along with a list of some others (19 in total) with test flakiness as a boolean.

- Static approaches do not need the code to be executed again. Models built using static features have many advantages and are less costly.
- Pinto et al. [4]built a set of predictors considering that some patterns within the test code may be employed to identify flaky tests automatically.
- The authors came to the conclusion that the vocabulary-based strategy performs poorly when used across projects because it is context-sensitive and prone to overfitting.
- Considering this result, an alternative approach for flaky test prediction based on test smells is used. Test smells are associated with potential design problems in the test code.
- Test smells are a deviation from how tests should be created, arranged, and interacted with one another. That deviation can indicate issues with test design and negatively impact test performance.
- An open-source test smell detection tool, tsDetect is used. For each test case, this tool requires the identification of the corresponding production code to detect the test smells.

2 RELATED WORK

We have come across some papers that compare various meth-