Code Defect AI

User Manual

V1.0

Contents

[Revision History 3](#_Toc20746900)

[Code Defect AI 4](#_Toc20746901)

[1. Introduction 4](#_Toc20746902)

[2. Methodology 5](#_Toc20746903)

[2.1 Explore 5](#_Toc20746904)

[2.1.1 Business Understanding 5](#_Toc20746905)

[2.1.2 Data Understanding 5](#_Toc20746906)

[2.1.3 Data Preparation 6](#_Toc20746907)

[2.2 Evaluate 6](#_Toc20746908)

[2.2.1 Modelling 6](#_Toc20746909)

[3 Code Defect AI User Interface & Workflow Process 7](#_Toc20746910)

[3.1 Overview 7](#_Toc20746911)

[3.2 Project Dashboard Screen 7](#_Toc20746912)

[3.3 Prediction Details Page 9](#_Toc20746913)

[3.3.1 Commit Id 10](#_Toc20746914)

[3.3.2 Commit Time 12](#_Toc20746915)

[3.3.3 File Name 12](#_Toc20746916)

[3.3.3.1 Bug Prediction Explanation Screen 12](#_Toc20746917)

[3.3.4 Confidence Score 13](#_Toc20746918)

[3.3.5 Bug Risk Prediction 14](#_Toc20746919)

[3.3.6 Commit Details 15](#_Toc20746920)

[3.4 View Trend – Multivariate Trend Analysis 15](#_Toc20746921)

[4 Appendix 17](#_Toc20746922)

[4.1 GitHub Links 17](#_Toc20746923)

[4.2 Feature List 18](#_Toc20746924)

[4.3 Screen Footer links 18](#_Toc20746925)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2019/09/27 | V1.0 | Code Defect AI User Manual | Altran |

# Code Defect AI

# Introduction

Code Defect AI solution leverages Machine Learning to train a classifier for predicting potentially buggy commits through the historical analysis of the code committed and bugs raised.

This document describes the use of the Graphical User Interface provided by the Code Defect AI.

The table below shows the three GitHub projects for which the predictions are shown by the Code Defect AI.

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Programming Language** | **GitHub Project Name** |
| 1 | Java | Spring Boot |
| 2 | C/C++ | OpenCV |
| 3 | C# | CoreFX |

# Methodology

Figure 1 depicts the three-step methodology followed in developing the Code Defect AI.



Figure 1: Methodology

## Explore

Explore deals with business & data understanding of the project, followed by preparation of data for evaluation phase.

## Business Understanding

Microsoft intends to enable its large and fast-growing developer community get started on AI/ML technologies. Microsoft AI Lab promotes innovative solutions based on Microsoft AI/ML technologies. Microsoft has selected the Altran Code Defect AI solution to be promoted as part of its AI Lab initiative and make the solution available to its Microsoft AI/ML developer community.

## Data Understanding

Data collection of the Spring Boot, OpenCV & CoreFX projects in GitHub is performed using the RESTful interfaces along with CLI (Command Line Interface).

Issues data is collected from GitHub using Restful API’s followed by labelling of raw data collected from complete repository.

Preprocessing of data includes cleansing, normalizing, vectorizing, labelling and feature reduction using python

## Data Preparation

Data Cleaning process follows steps

* Feature Identification – Refer appendix for used set of features for model training
* Label Encoding – Conversion of the Categorical data to numerical data.
* One hot encoding – Splitting of categorical data with more than two values into multiple columns.
* Data Scaling & Normalization – Scale the features to a range centered around 0.
* Labelling:

The labelling process involves understanding of the pattern in which the fix commits are tagged for each of the closed issue. After the fix commits are collected, the commits which introduced the bugs are identified through historically backtracking changes for each file in a fix commit. This is done using python scripts in an automated fashion. However, as each project has varying keywords used for commit messages and tagging of fixed commits, initial identification of the bug pattern is required to translate algorithm to automated scripts.

## Evaluate

The evaluate procedure takes in multiple machine learning algorithms on the input training data and deploy the one with high precision and high recall

## Modelling

Following ML Classification Algorithms have been evaluated on the training data. These algorithms have been evaluated with different data balancing techniques.

* Naïve-Bayes
* Random Forest
* Decision Tree
* Logistic Regression
* SVM (Support Vector Machines)
* LDA (Linear Discriminant Analysis)
* MLP (Multi-Layer Perceptron)

As the data set contains small amount of bug data in comparison to clean records, synthetic data generation is an important step to avoid bias towards the majority class. Here, the minority class is of more importance. Data set balancing techniques used for improving imbalanced data are:

* Synthetic Minority Over-Sampling Technique (SMOTE)
* Adaptive Synthetic (ADASYN) Sampling Approach
* TRIM-SMOTE (Pruning based Smote variant)

# Code Defect AI User Interface & Workflow Process

## Overview

Figure 2 depicts the User Interface navigation flow modelled in Code Defect AI.

Figure 2: Logical flow of User Interface in Code Defect AI

## Project Dashboard Screen

Figure 3 shows the Project Dashboard Page providing the consolidated view of the three projects. For each of the projects it shows the prediction of the number of files having a high bug risk and the total number of committed files during the period of 15 days. The solution has been built using three open source projects from GitHub; SpringBoot, Open CV and CoreFX.

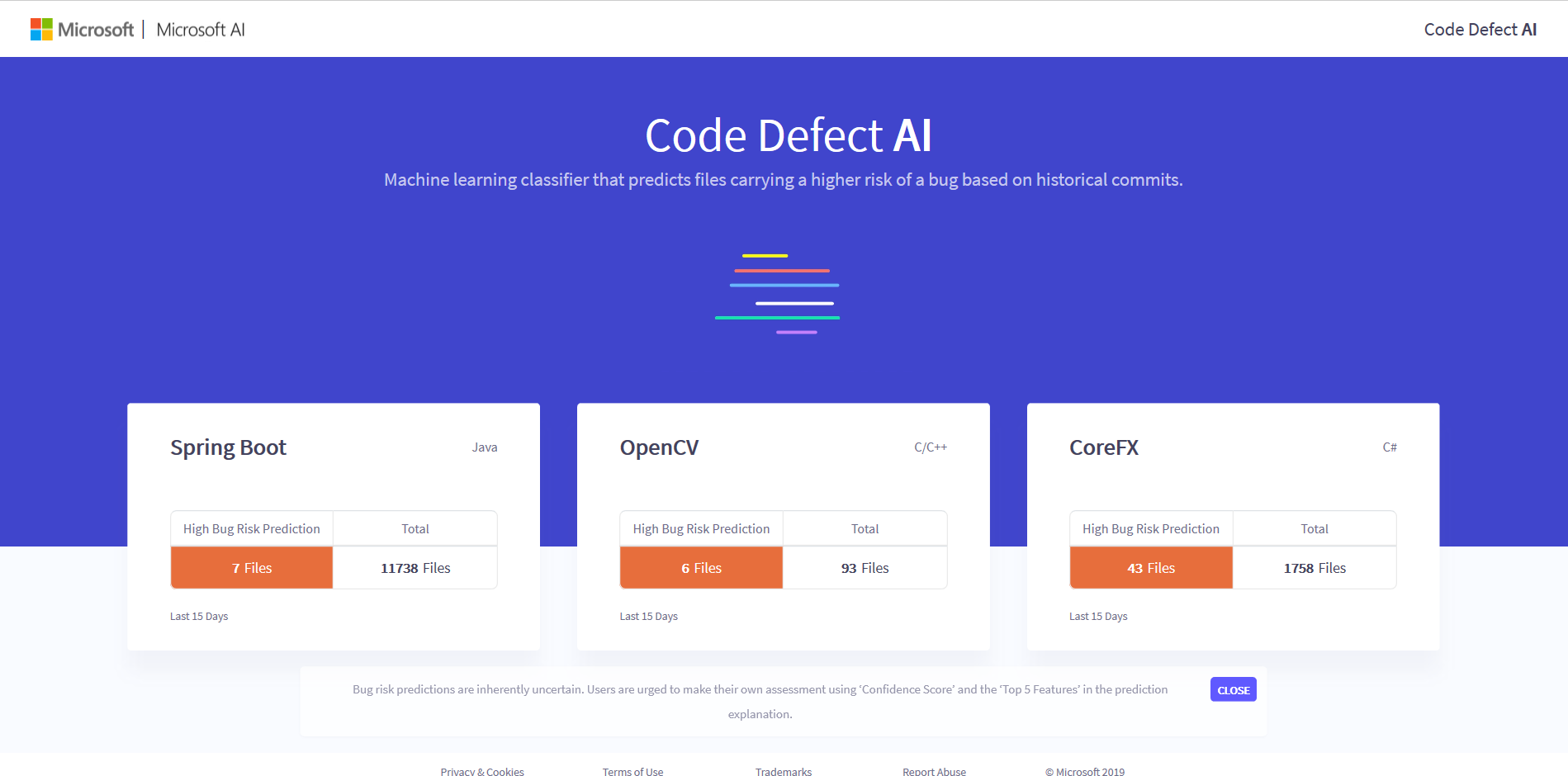
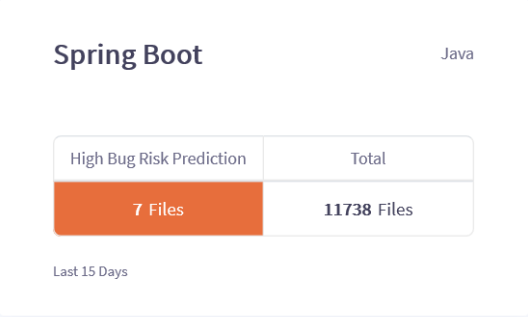


Figure 3: Project Dashboard

Programming language

Project name



Total number of checked-in files

Total Number of High Bug Risk Files

Time duration of data collected for prediction

Figure 4: Project Card

Note: values depicted are for illustration only. Values can vary on actual user interface

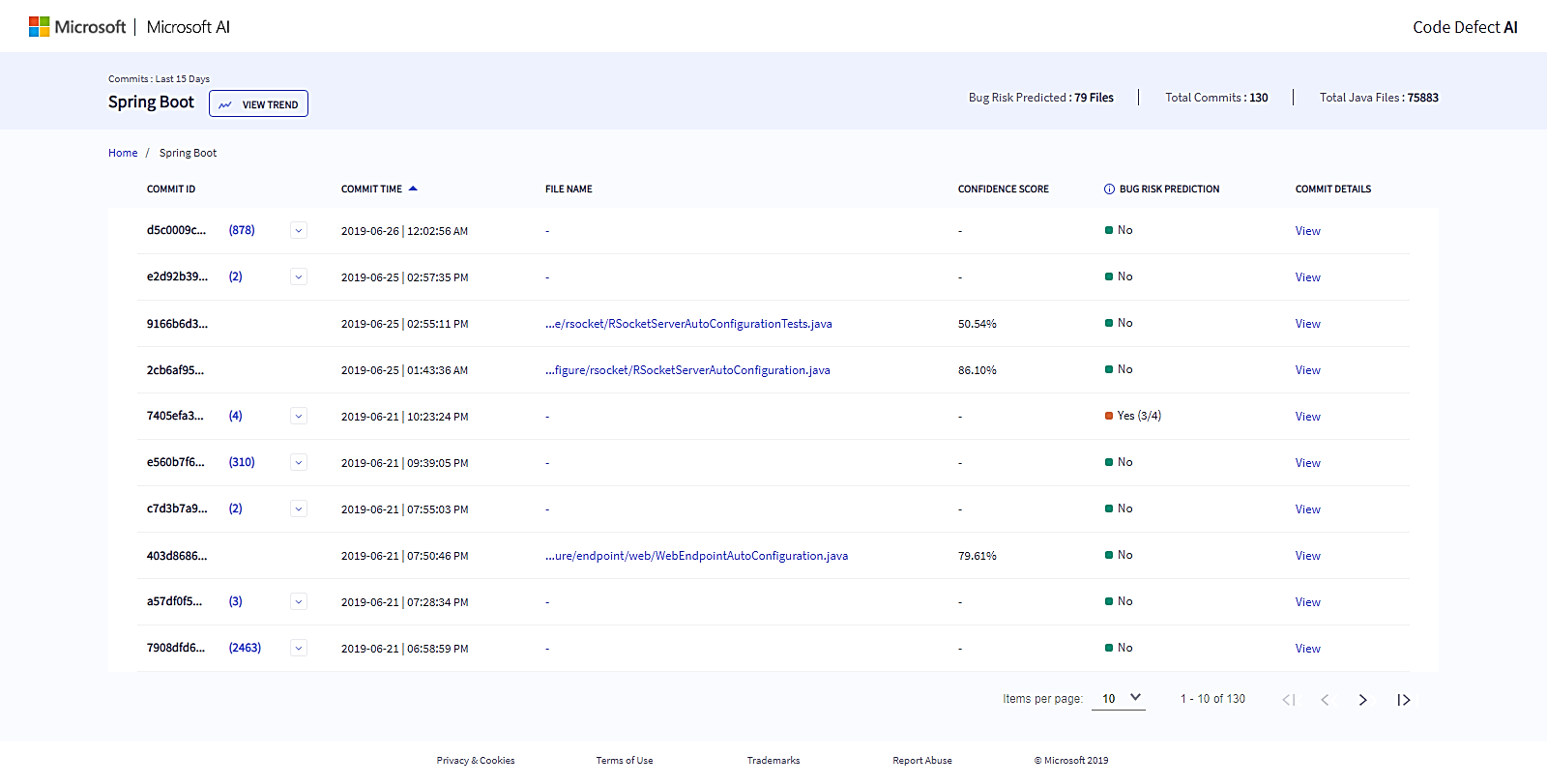
Figure 4 shows the project card showing the following details:

* Name of the project e.g. Spring Boot.
* Programming Language e.g. Java.
* Number of files predicted as carrying a high risk of bug shown under “High Bug Risk Prediction” heading.
* Total number of files checked-in in the specified time frame under heading “Total”.
* Time duration of data collected for prediction is Last 15 Days.

## Prediction Details Page

After clicking on any card on the Project Dashboard, user lands on Prediction Details page as shown in Figure 5. User can view the prediction for each of the files in the commits made.

By default, entire data is sorted in descending order of the time stamp of the commit. Sorting can be modified by the user by clicking on the column name of the required column. As soon as user clicks on the column name, a small purple icon appears besides header, depicting whether the sort order is ascending or descending.



Click here for project trend

File Prediction Details

Click here to view commit details

View Files for commit

For view data sorted on prediction

`

Figure 5: Prediction Details Page

Commit details page gives user a consolidated view of project commits details along with navigation to file prediction explanation, viewing all files in a commit the number of files is more than a single file and trends view screen.

## Commit Id

Commit ID refers to the revision number in the Git repository. This ID is 40 characters in length and consists of both letters and numbers. If multiple files are committed in the same commit, Commit ID can be expanded by selecting an “expandable caret” icon as shown in Figure 6.

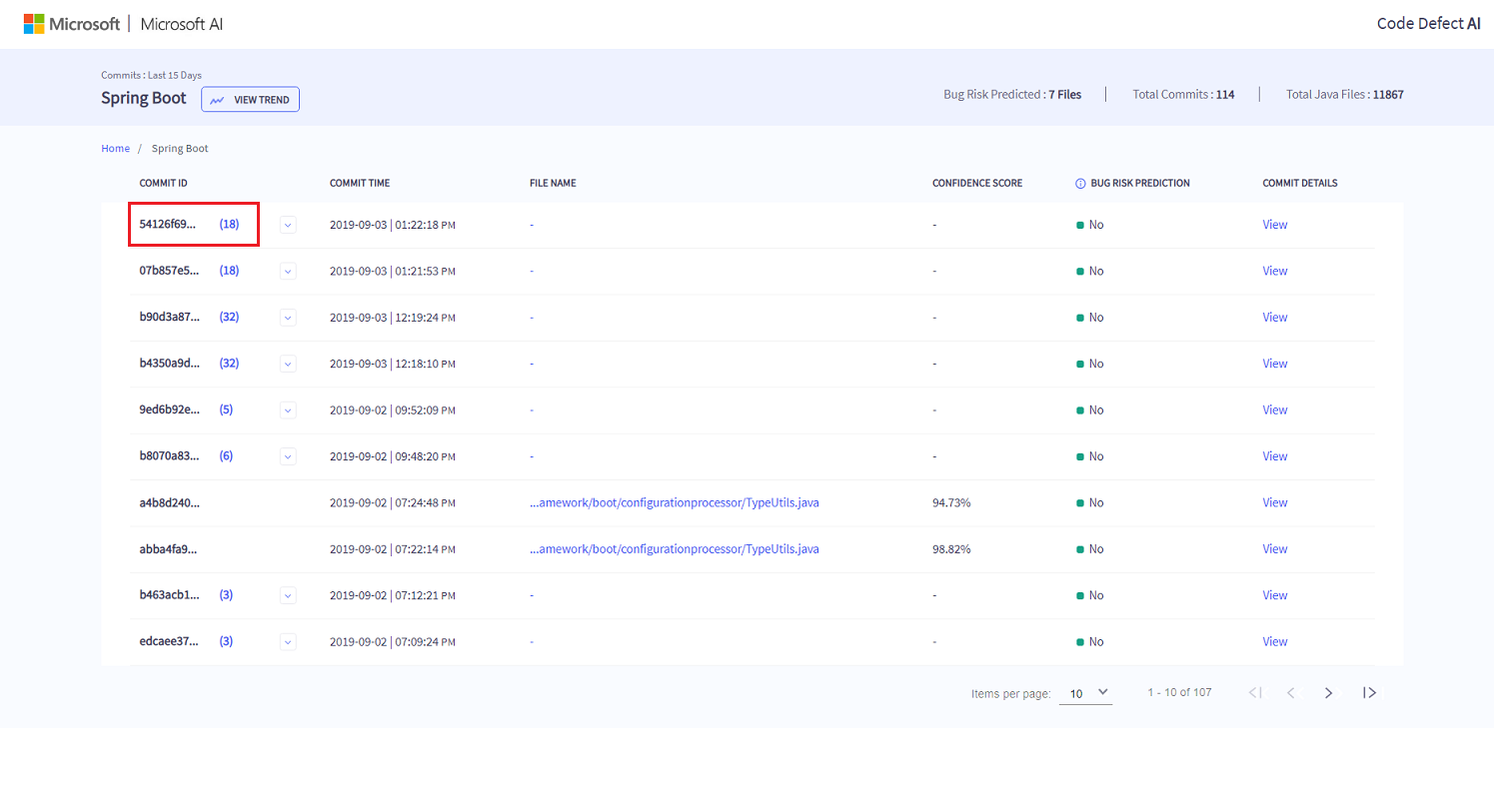


Figure 6: Expandable Caret Icon

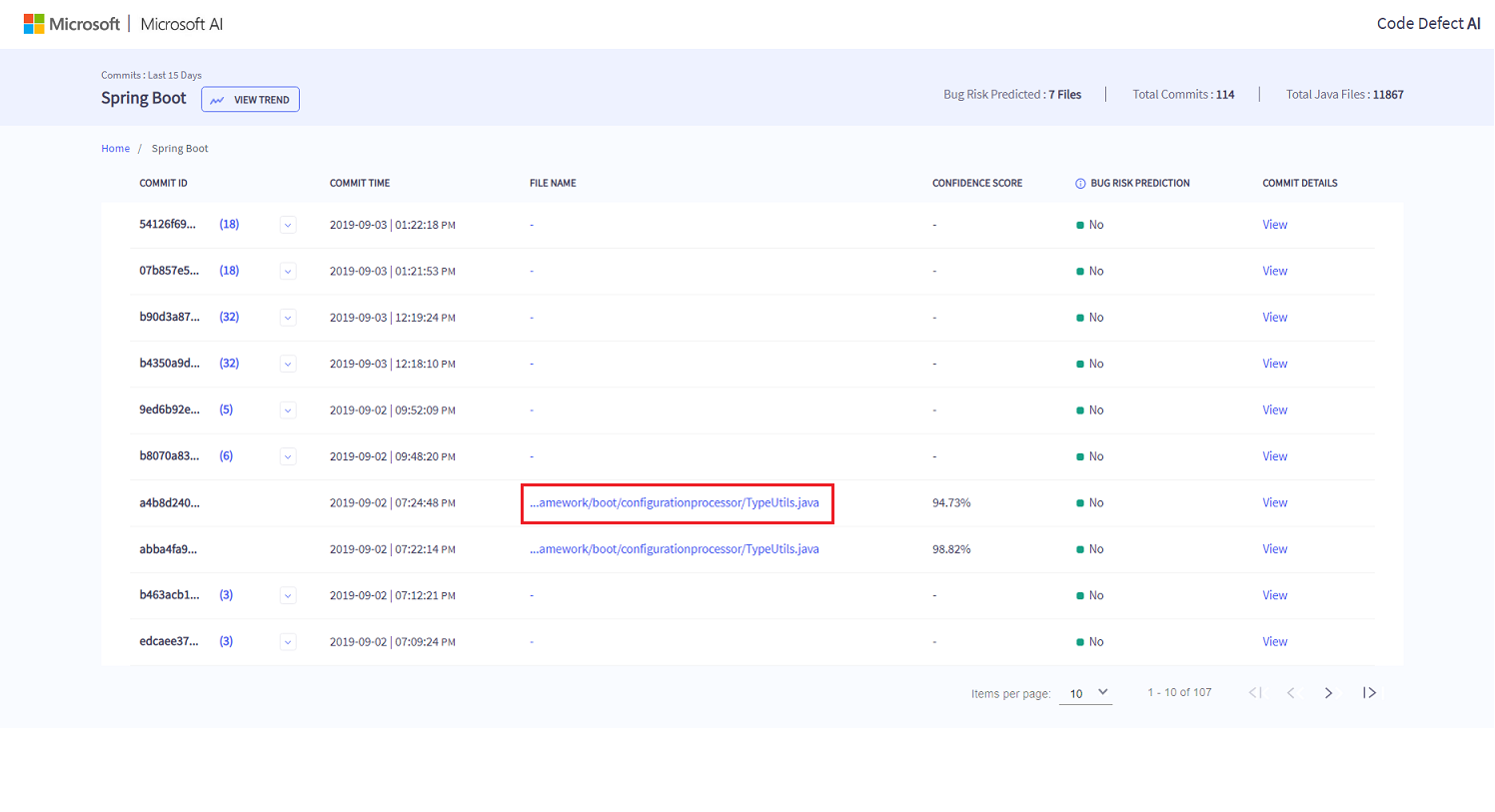


Figure 7: Single file commit

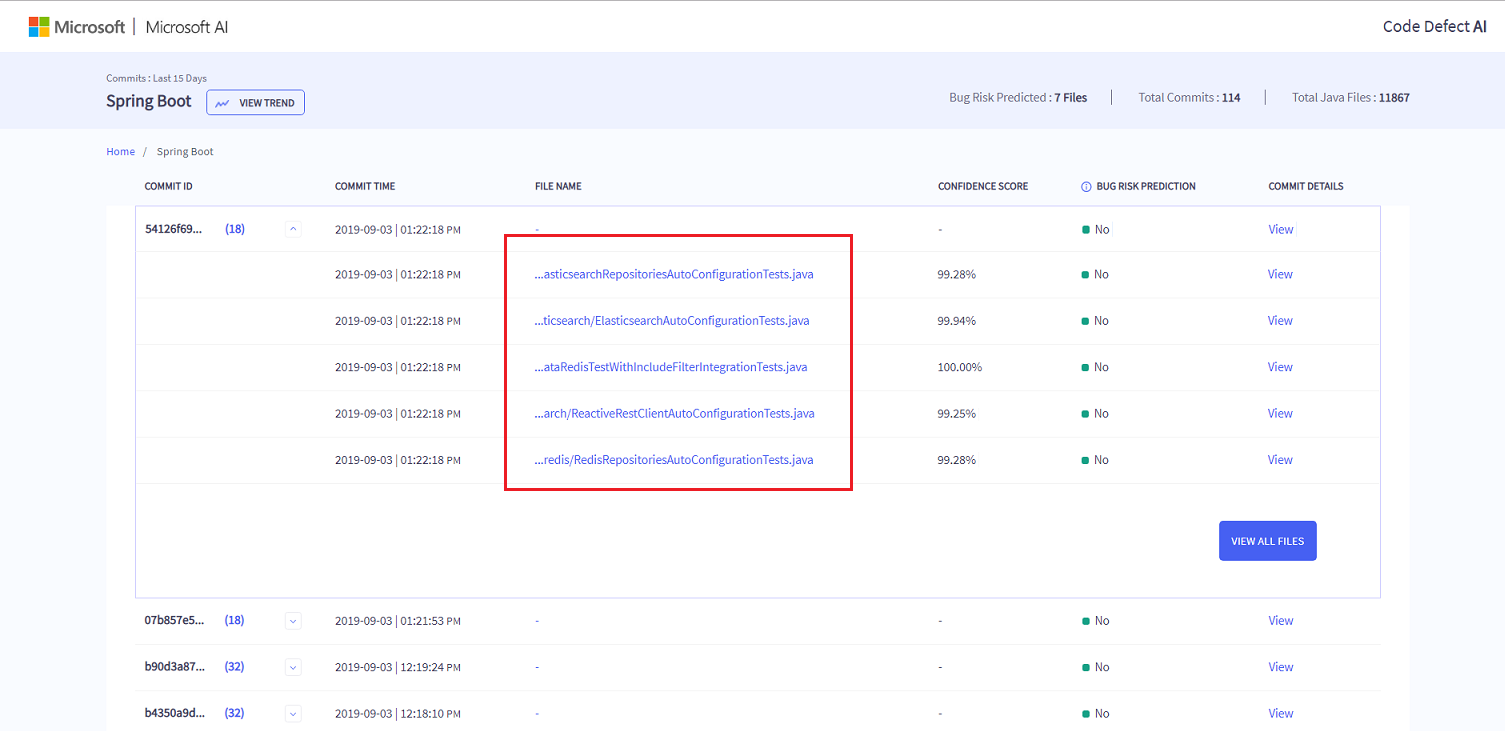


Figure 8: Multiple file for commit

Only five files for a Commit ID is displayed as shown in Figure 8. Clicking on “View All Files” button lists all committed files that are present in the commit as shown in Figure 9.

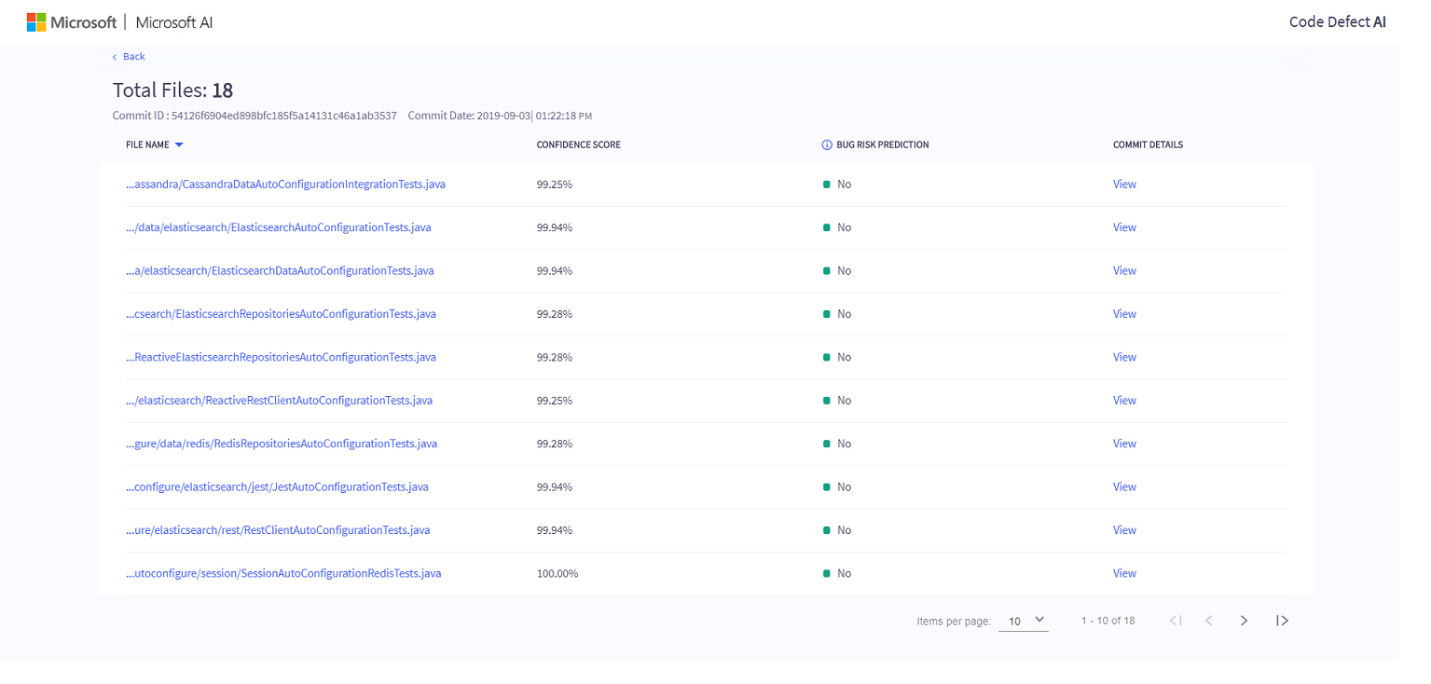


Figure 9: View All Files in a Commit ID

## Commit Time

Commit Time refers to the time at which the code is committed in the repository. By default, entire table is sorted in descending order of Commit Time so that the latest committed code is displayed at the top.

## File Name

File Name displays the name of the file with extension. The file name display is restricted to certain number of characters for alignment purpose. On hover, the complete file name with path is visible. On clicking on File Name, Bug Risk Prediction Explanation screen is shown as in Figure 10.

## Bug Prediction Explanation Screen

Despite widespread adoption, majority of the times machine learning models are considered black box model only. In Code Defect AI, Local Interpretable Model-agnostic Explanations (LIME)[[1]](#footnote-1) [1] is used to explain the Bug Risk Prediction in the Bug Risk Prediction Explanation screen as in Figure 10. The output of LIME is a list of explanations, reflecting the contribution of each feature to the prediction on a file.

Bug Risk Prediction Explanation Screen shows top five features which has maximum impact while making the bug risk prediction.

LIME assigns each feature a coefficient that depicts the weight of the feature in prediction decision by LIME algorithm. The weight is positive if the feature value supports the target class (Bug Risk Prediction) and negative if does not. To ease the understanding of the coefficient, output is depicted with horizontal bars with green and orange color coding representing ‘Low Bug Risk’ and ‘High Bug Risk’ respectively. To view the GitHub history for any file, user can click on “File History” button on Bug Risk Prediction Explanation page and the user will be redirected to GitHub link.

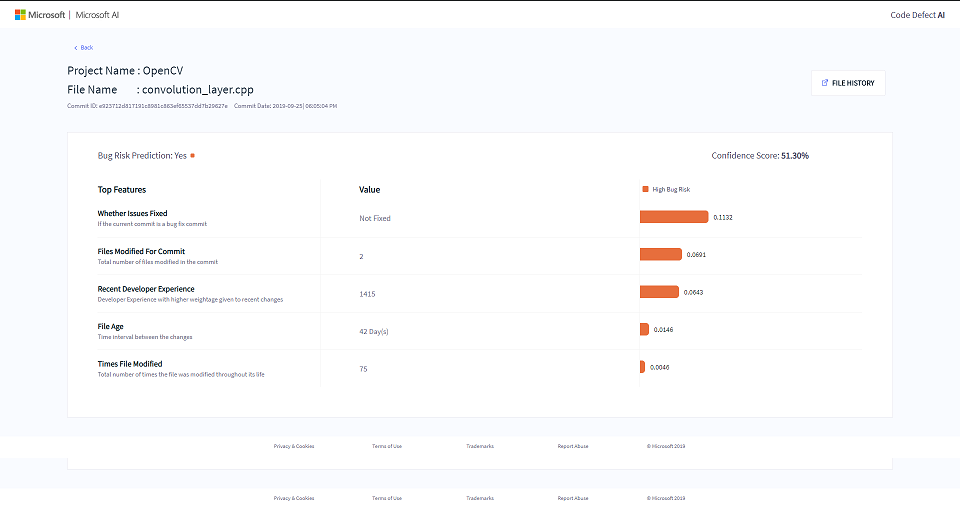


Figure 10: Bug Risk Prediction Explanation screen

## Confidence Score

Confidence score shown in Figure 11 depicts how confident the machine learning classifier in making the predicting.

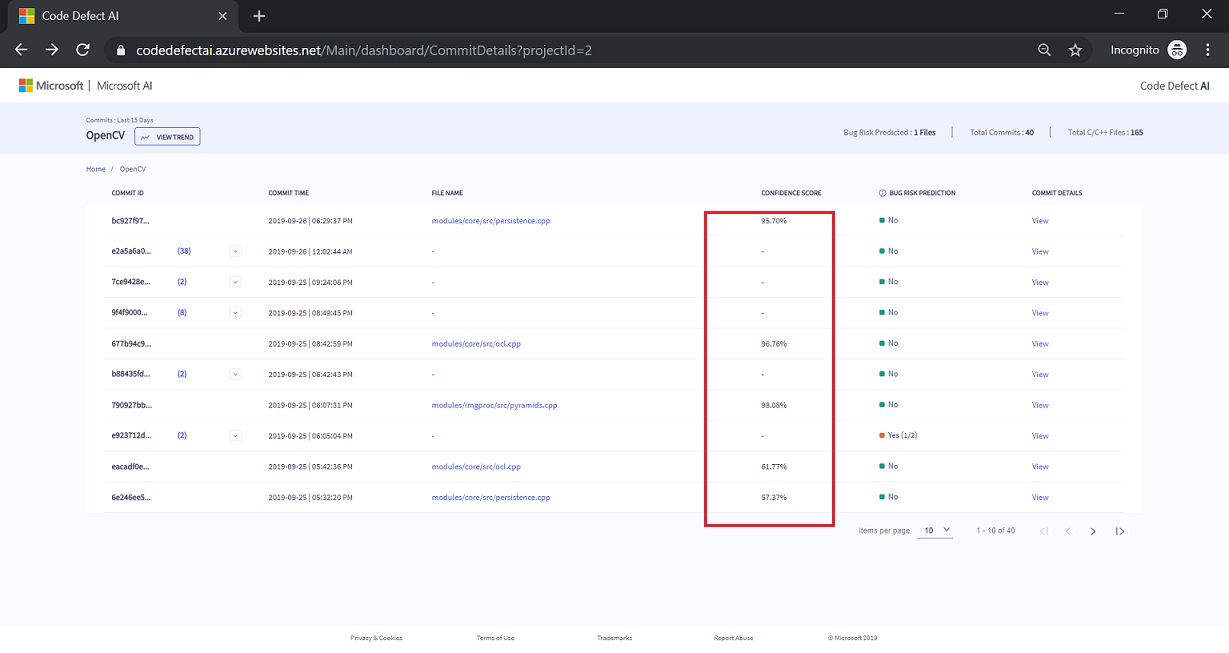


Figure 11: Confidence Score

For example:

1. If the Confidence Score is 81.0% and Bug Risk Prediction shows “Yes” then the model is 81.0% confident that code carries High Bug Risk.
2. If the Confidence Score is 81.0% and Bug Risk Prediction shows “No” then the model is 81.0% confident that code carries does not carry a High Bug Risk.

For commits that have more than one file, Confidence Score appears as “-” (hyphen) and when commit is expanded, confidence score is seen for each of the file in the commit.

## Bug Risk Prediction

Bug Risk Prediction displays the prediction of the machine learning classifier for each of the files in a commit as depicted in Figure 12. For example, if the column value is “No”, it signifies that file has been predicted as carrying a low bug risk. Whereas, and if the value is “Yes”, it means that file has been predicted as carrying a high bug risk.

For any commit which has more than one file, a consolidated view of prediction is given as in Figure 6: Expandable Caret Icon . The detailed prediction can be seen on expanding details for the commit.

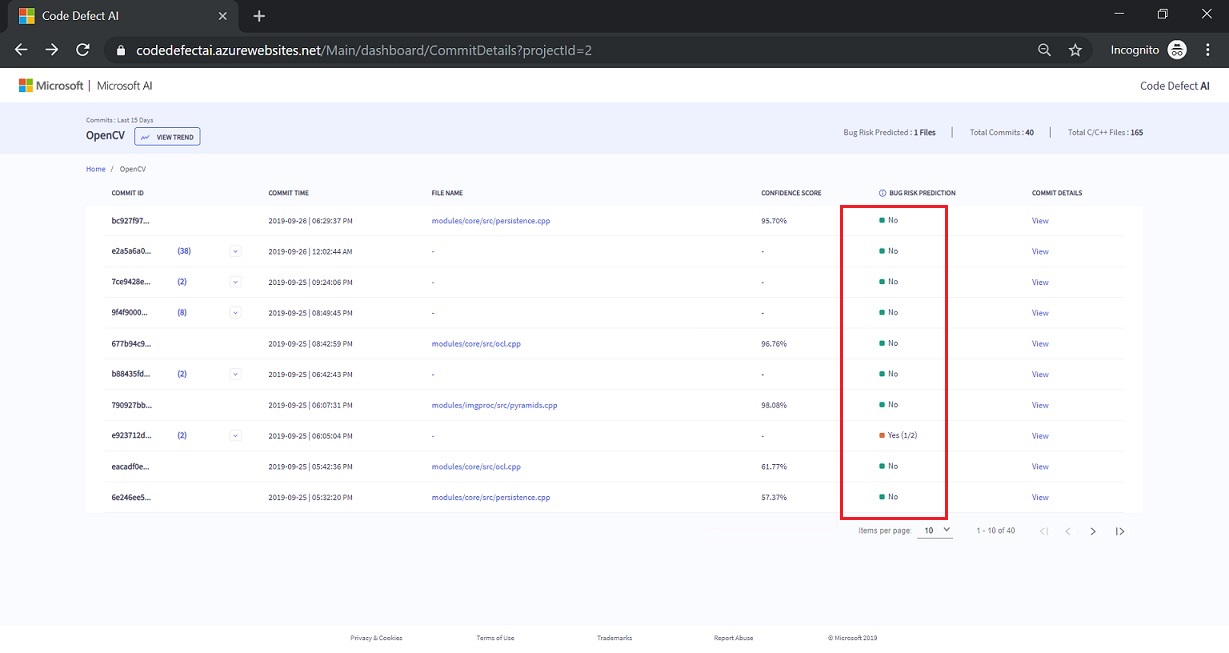


Figure 12: Bug Risk Prediction

## Commit Details

To view the GitHub changes made in any commit, user can click on the “View” hyperlink displayed under “Commit Details”. Please note that if any of commit or file has been deleted from GitHub history, that link will appear broken.

## View Trend – Multivariate Trend Analysis

The View Trend depicted in Figure 13 divides all features into two parts.

* Features that help predict project wide files carrying a high bug risk
* Features that help predict project wide files carrying a low bug risk.

Top 5 features are selected based on descending value of medians for LIME coefficients from each category and then plotted on boxplot. The feature on boxplot with highest median play an important role in the prediction. This can help a developer to take corrective action based on the feature importance. For instance, if Lines Modified is the feature which has more weight in determining the High Bug Risk, it can be deduced that the bug risk can be reduced with more emphasis on code review for files with number of lines modified per file greater than a certain threshold. Box plot depicts the 5-number summary of the feature; minimum, quartile 1, median (quartile 2), quartile 3 and maximum value.



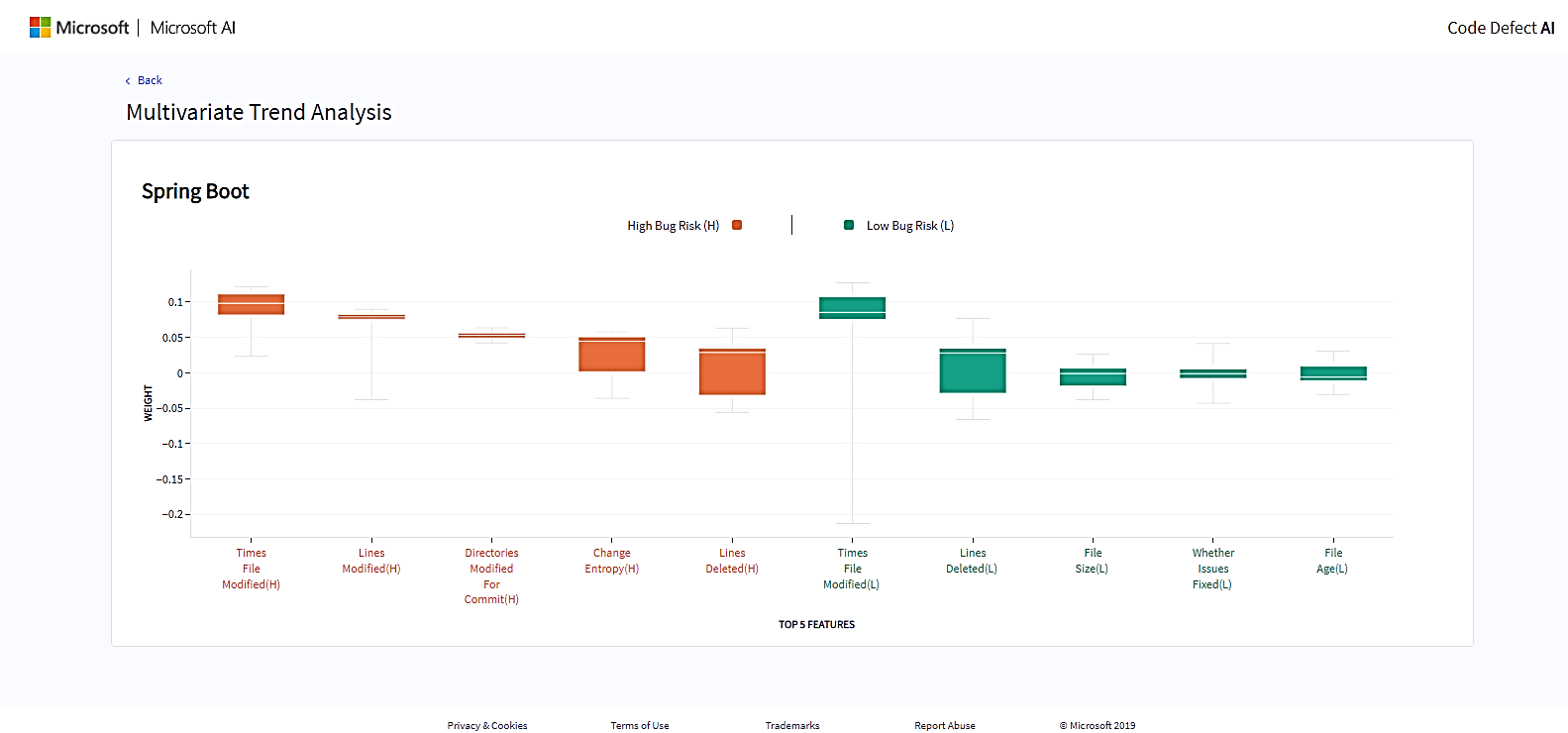


Figure 13: Multivariate Trend Analysis Page

When we hover over the graph we can see the following options:

|  |  |  |  |
| --- | --- | --- | --- |
| **S no.** | **Applicable Feature** | **Description** | **Snapshot** |
|  | Download a plot |  |  |
|  | Zoom |  |  |
|  | Pan |  |  |
|  | Zoom In |  |  |
|  | Zoom Out |  |  |
|  | Autoscale |  |  |
|  | Toggle Spike Lines |  |  |
|  | Show Closest data on Hover |  |  |
|  | Compare Data on Hover |  |  |

Table -14.2: Multivariate Trend Analysis Page

# Appendix

## GitHub Links

|  |  |  |
| --- | --- | --- |
| **GitHub Project** | **Programming Language** | **Project Git Hub URL** |
| SpringBoot | Java | <https://github.com/spring-projects/spring-boot> |
| OpenCV | C/C++ | <https://github.com/opencv/opencv> |
| CoreFX | C# | <https://github.com/dotnet/corefx> |

## Feature List

|  |  |  |
| --- | --- | --- |
| **UI Nomenclature** | **Units** | **Description** |
| Status | - | Depicts if the file is added, deleted, modified or renamed |
| Recent Developer Experience | - | Developer Experience with higher weightage given to recent changes |
| File Changes | - | Average lines changed per change block in the file |
| Files Modified for Commit | Number of File(s) | Total Number of files modified in the commit |
| File Age | Number of Day(s) | Time interval between the changes |
| Developers Count | Number of Developer(s) | Total number of developers that have changed the file |
| Times File Modified | Number of times | Total number of times the file was modified throughout its life |
| File Size | KB | Current size of the file in kilobyte (KB) |
| Developer Module Experience | Number of Commit(s) | Total number of commits made by the developer for the submodule |
| Developer Experience | - | Total number of commits made by the developer |
| Whether Issue Fixed | Boolean | If the current commit is a bug fix commit |
| Commit Type | Boolean | Merge or non-merge commit |

## Screen Footer links

|  |  |
| --- | --- |
| **Footer screen links** | **URL** |
| Privacy | <https://privacy.microsoft.com/en-US/privacystatement> |
| Terms of Use | <https://www.microsoft.com/en-us/legal/intellectualproperty/copyright/default.aspx> |
| Trademarks | <https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx> |
| Report Abuse | <https://support.microsoft.com/en-us/contactus/> |

1. "Why Should I Trust You?": Explaining the Predictions of Any Classifier,

   Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin, <https://arxiv.org/abs/1602.04938> [↑](#footnote-ref-1)