**Research question 1**

***RQ-1****: What source code properties characterize defective infrastructure as code scripts?*

To answer this question, we utilize the dataset obtained directly from the paper, whose properties are thoroughly analyzed and extracted based on grounded theory. According to the report, the overall numerical trend was explored, and several non-parametric hypothesis tests were applied. Specifically, the tests used are the MWU test and the Cliff’s Delta. After completing the hypothesis tests, the authors built a random forest model to make the prediction and ranked the feature importance.

The authors first observed the dataset distribution through a set of box plots to gain better insight into the dataset. According to the plot, we can observe that most properties have a relatively dense range. Conversely, we can also detect apparent differences and a wide value range in “Attribute,” “Comment,” “Hard\_coded\_string,” and “Lines\_of\_code.” The plots correspond to the one attached to the paper. By observing the plots, we are able to spot a trend that OpenStack has a broader range of values on most variable fields.

A screenshot of a computer screen

Description automatically generated

We then obtained two tables to gain general ideas on the dataset. From the first table, we illustrate the means and the maximum of the properties. A numerical comprehension can be established based on this table. As shown in the former box plot, OpenStack has lines of code over 1000, which, from the table, is accurately 1287.

A table with numbers and text

Description automatically generated

The second table shows the median values of each value for both defective and non-defective codes. “D” represents defective code, while “ND” indicates non-defective code. By observing the table, we found that each property count in defective code tends to be equal to or more than the corresponding count in non-defective code most of the time. The pattern is evident in “Lines\_of\_code” and “Attribute,” which indicates that a longer code may have a higher chance of being defective.

A table of data with text

Description automatically generated with medium confidence

 After obtaining a global understanding of the dataset, we are to apply the MWU test and the Cliff’s Delta. Following the settings in the paper, the null hypothesis is that the property is not different between defective and neutral files. To reject the null hypothesis using a one-sided MWU test, we should obtain a p-value smaller than 0.05. When measuring the difference using Cliff’s Delta, we follow the rule provided by Romano et al., that if the value is larger than 0.47, the difference is large; if between 0.33 and 0.47, the difference is medium; if between 0.14 and 0.33, the difference is small; otherwise, the difference is negligible.

With our result, we can reject the null hypothesis for all the properties except “Comment” and “URL.” The result shows us that the remaining 10 properties tend to have a larger value if they link with a defective code. We are not observing a property’s Cliff’s Delta value lying in the largest range in all organizations. However, the property “Lines\_of\_code” has 3/4 values larger than 0.47. Therefore, we can still identify that the differences in “Lines\_of\_code” between the two types of code are large. For most of the properties, the difference is medium.

A table of numbers and symbols

Description automatically generated

The paper did not offer details on the parameters of the random forest. Hence, we only apply the default random forest classifier provided by scikit-learn. The toolbox also offered access to the importance of each feature. Following the instructions, we train the classifier 10 times and collect feature importance for each model version. Medians are collected to rank the features. We show the ranking result in the following table. Since the random seeds are not fixed, the feature importance medians are not precisely the same as the ones in the article. Some of the rankings are not similar to the original table. We noticed that the ranking conflict mainly happens when several properties have importance that are close to each other. It can be understood that these properties are at the same importance level; therefore, they would shuffle their rank according to different random seed settings.

We take the first three lines as the most impacting properties. It is worth noting that only three properties appear in these three lines, respectively, “Lines\_of\_code,” Hard\_coded\_string,” and “Attribute.”

A table with text on it

Description automatically generated