Report on Kernel Quality Analysis

Group 15

1. Proposal

Hypothesis:

1.1 Systems with more developers are more stable.

1.2 The minor versions of the subordinate version have less substantial changes in the later period.

Metrics:

We can measure the amplitude and frequency of code modification to measure whether the system is stable. Stable means small modification amplitude and frequent modification.

2. Requirement analysis

To analyze linux kernel quality, we need to understand it deeply.

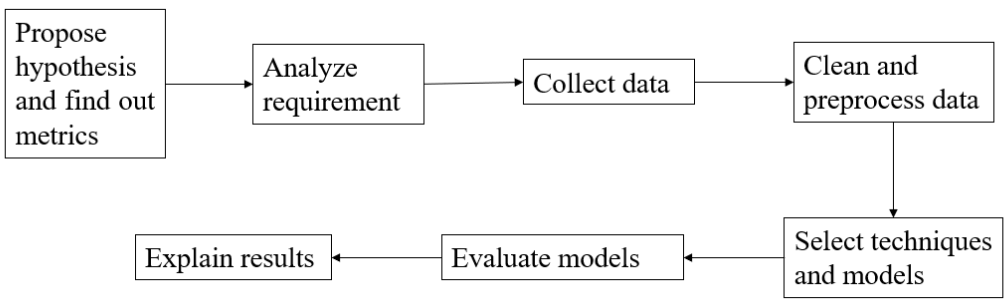
2.1 We shall collect proper data and do some preprocessing.

2.2 We shall analyze a few techniques and then select models for us.

2.3 We shall evaluate our models.

2.4 We shall make explanation on results.

Design:



3. Data acquisition

There are three types of data we need to obtain: corresponding to the code modification interval, code modification range, and the number of developers changes. And for those data, we obtained it by encapsulating git, and parse it by unicodedata after obtaining the data, and finally pass the data into csv for analysis.

As our project is aimed at the fourth version of Linux, the data we required are all from this version. After preliminary processing and analysis of all relevant data from 21 versions (V4.0-V4.20), it was found that there are certain difficulties in operation due to the excessive amount of data. After exporting the data of each version, it is found that there are 100-200 minor versions in V4.4, V4.9, V4.14, and V4.19 which are the suitable amount for analysis. Thus, these four minor versions are selected to pass Analyze and summarize some data to get the overall characteristics

Here are the detail of data acquisition:

① Corresponding to the code modification interval

· Acquisition process: By crawling the time of each fix_commit, sort it and make a difference between adjacent ones to get the interval of code modification time.

· Data range: V4.0-V4.20

② Code modification range

· Acquisition process: By obtaining the number of add and delete lines and the modified file path, and add the value of add and delete to get the code modification range.

· Data range: V4.0-V4.20

③ The number of developers changes

· By obtaining the numbers of developers and conduct comparative analysis to obtain changes in developers.

· Data range: V4.4, V4.9, V4.14, and V4.19

4. Data cleaning

This process standardizes the data set by deleting duplicate values, filling in missing values, modifying outliers, or formatting errors for later data analysis. The data trade-offs made in our project are mainly divided into three types:

① For code with a very short modification interval, we choose to keep it:

A lot of code is modified at a time interval of 1 second. At first, we thought that it was caused by multiple submissions by the same programmer, but after the code author crawled, it was found that the code with a short interval was not submitted by the same person, but by different programmers, they submitted in one second interval in different time zones, so these codes are not invalid, and they don't need to be cleaned.

② Delete data that does not contribute much to the modification:

After many files are crawled down, the path of the many data is the beginning of document or documentation. The role of these data is similar to readme, and they act as notes. So, they do not contribute much to the data modification and need to be cleaned.

③ Data with short intervals of addition and deletion are cleaned：

For this case, we believe that this is a refactoring of the code form rather than adding a new function or feature. In this case, we will find their index and delete them from the original list.

5. Techniques selection and evaluation

First, our group discussed six models, namely linear regression model, classification decision tree, naive Bayes, KNN algorithm and mixed Gaussian model and K-means.

For the linear model, for our project, the linear model has greater limitations, only few factors are applied, and the linear relationship is lacking.

For the Naive Bayes algorithm, its biggest feature is that the variables are required to be independent, but our variables may be correlated or influence each other. Hence, they are not completely independent.

For the decision tree algorithm, we have fewer indicators, which will lead to insufficient depth of the decision tree and lead to under-fitting. It is not easy to make a distinction.

For the KNN algorithm, an initial classification is required, but we lack the label of this classification and cannot obtain the initial classification.

Because we lack sufficient prior knowledge and it is difficult to manually label, we can only choose from unsupervised learning algorithms.

For unsupervised learning, the mixed Gaussian model requires that the data set must meet normality, but our data cannot guarantee normality, so we give up using the hybrid Gaussian algorithm.

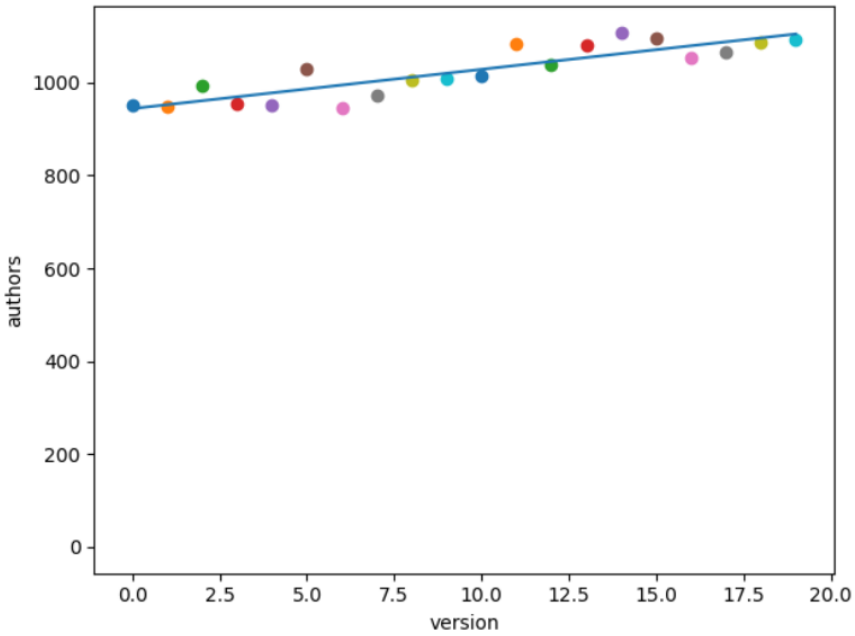
For the K-m We have no labels, we can only give the data to calculate a "centroid", then give the data, and calculate the distance between the point and the centroideans algorithm. It is suitable for our data and goal.

One disadvantage of the K-means algorithm is that the value of k is difficult to estimate. It is difficult to know in advance how many categories the given data set should be divided into. The automatic combination and splitting of the classes give us a reasonable number of types k. Also, there may be an offset in the “centroid”, which may cause the threshold value to be blurred.

6. Model selection and evaluation

① linear regression

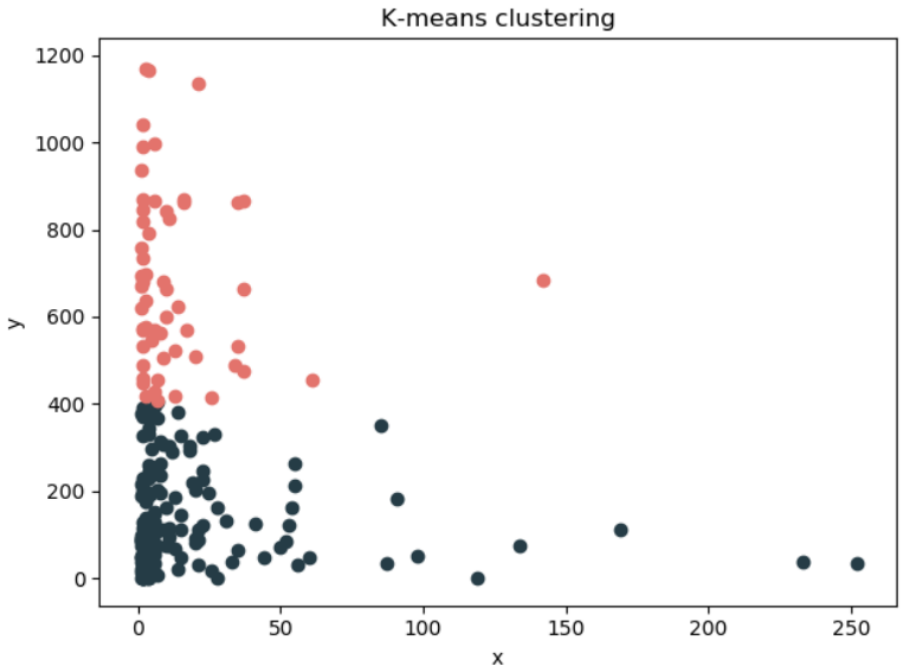
Our group used linear regression model to evaluate the number of developers for different versions. Through linear regression, we found that the slope of the fitted linear regression line was almost 0. That is to say, the number of developers remained at a steady level and has barely budged. This result violates the hypothesis that systems with more developers are more stable.



② K-means

Our group used K-means model to sort out the slightly modified versions. First, select k initial cluster centers. And calculate the distance between each object and these K centers, and allocate them to the nearest cluster according to the principle of minimum distance. Then, use the sample mean in each cluster as the new cluster center. Repeat previous steps until the cluster center no longer changes. At the end, k clusters are obtained.

Through k-means, our group got the version set with small time interval and large modification amplitude, and extracted them. Results verify that the minor version of the subordinate version has less substantial changes in the later period.



7. Result explanation

At the beginning with, our first hypothesis is systems with more developers are more stable. As the data processing analysis progressed, by analyzing the linear regression curve, we found that the number of technicians in each version fluctuated around 1000. It's too average to be a standard for judging the stability.

Our second hypothesis is the minor version of the subordinate version less substantial changes in the later period which has some distinctive features. According to the distribution of fix times with version_interval, as the number of revisions increases, the denser the line segments, the more stable the revision. So the hypothesis is a effective argumentation.