CS 212

Final Project

Clustering of Kernel developers data analysis report

Group 17

Jiacheng Shen shenjch18@lzu.edu.cn 320180949191

Ziyao Wang wangziyao2018@lzu.edu.cn 320180940361

Motong Tian tianmt18@lzu.edu.cn 320180940301

Yichen Wang ychwang2018@lzu.edu.cn 320180940341

Zixiang Song songzx18@lzu.edu.cn 320180940221

Ke Lei leik18@lzu.edu.cn 320180939861

Jiang Zhou zhoujiang18@lzu.edu.cn 320180940681

Wei Zhu zhuw2018@lzu.edu.cn 320180940701

1. Data background

Our data analysis program is based on Linux kernel, which can be considered to be a big, complex and fast changing data system. Data like the information of commits per version and authors can be extracted from it. We aim to use the data to find out the Linux development status, evaluating its development quality.

2. Modelling

Based on the data, the information of authors, we take author studying as our direction, and come up a model according to that.

2.1. Hypothesis

H0: All developers in kernel can be clustered in terms of contribution to kernel.
In other words, Linux Developers are either highly concerned to kernel development or not.

2.2. Test-metric

According to our analysis, we find two factors that may contributing to developer clustering: times of commits, commit cycle. As a clustering question, we choose Kmeans to cluster developers. And according to the factors, developers would be hopefully divided into classes that representing the contribution level to Linux kernel of developers.
Or otherwise, if Kmeans fail to divide the developers into clusters, we can only reject H0 and conclude that all developers are concerned about kernel development with the same attitude.

2.3. Data

The data responding to the metrics is the total commit time and the average time difference of commits of an author as a record. We choose to find the data in v4.1-v4.9, which include thousands of developers, and make the authors with the factors as records, then put them into a table.

2.4. Assumptions:

Comparability: the clusters are comparable between versions.
Metrics: the metrics used extract the same information from each item
Hidden variable: There are no hidden variables that are distorting the assumed underlying causal relation.

3. Requirements

3.1. Hypothesis and data selection

3.1.1. All factors in the hypothesis must be found
3.1.2. Must find out enough metric for each factor
3.1.3. The corresponding data of each metric must be found

3.1.4. Data must be found by ourselves

3.2. Data collection

3.2.1. Data collection program must be automated.
3.2.2. The data collection program should have a good utilization of time and space
3.2.3. Data must be extracted from Linux stable
3.2.4. Data extraction must be done by git command
3.2.5. Program must output a CSV table: column names are: author - Submission quantity - time average difference
3.2.6. Flow chart of program design must be provided

3.3. Data cleaning

3.3.1. The processing object must be the table provided by data collection
3.3.2. The final result should be an integrated form instead of multiple tables
3.3.3. Must use pandas for data cleaning
3.3.4. Must delete unnecessary columns (some columns only have intermediate effect, but they are not used in later models)
3.3.5. The final result must have no duplicate values (duplicate values need to be merged)
3.3.6. The final result must have no missing value (Nan does not exist in the dataframe)
3.3.7. The final result must have no outliers
3.3.8. The final result must have no format error
3.3.9. The final output data structure must be clear

3.4. Modeling

3.4.1. Must establish the mathematical model that can best prove the hypothesis
3.4.2. Must record the basis and advantages and disadvantages of the model
3.4.3. The input data and output results required by the model must be clear

3.5. Technology selection and data analysis

3.5.1. Must select the appropriate technology (eg. correlation analysis, decision tree, convolution neural network) according to the model and record the reasons
3.5.2. Technology must be implemented into code program
3.5.3. The data analysis program should have reasonable accuracy
3.5.4. Flow chart of program design must be provided

3.6. Report

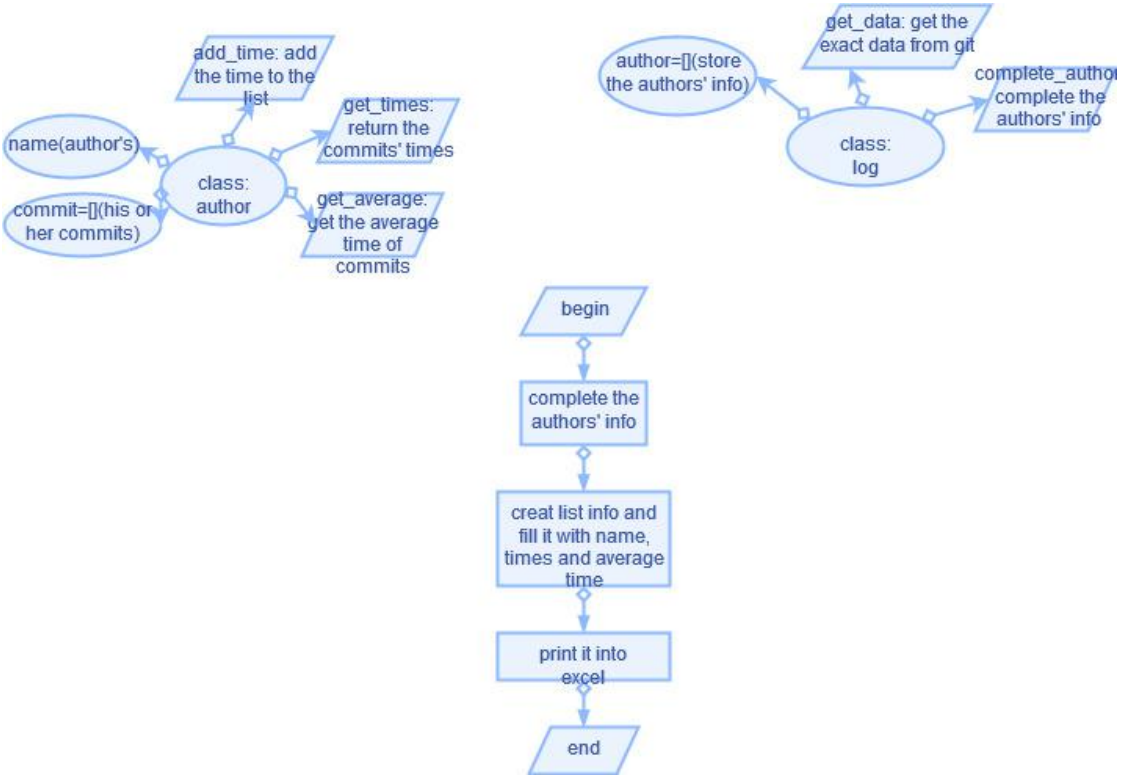
3.6.1. Report must reflect the main process of the project
3.6.2. Report must explain the project results and present a conclusion.

4. Design

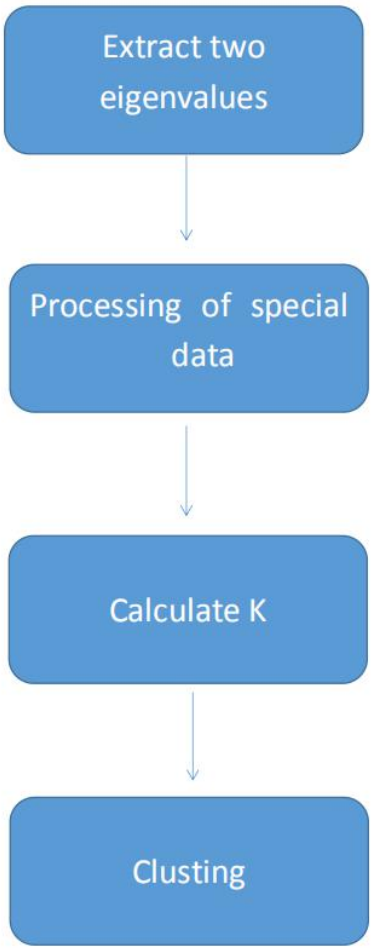
4.1. The main process of project

Finding out a direction: studying developers
-> coming up a hypothesis: developers in kernel can be clustered
-> Finding test-metrics: commit time and average commit cycle of developers
-> Mapping factors to data: three columns(author, commit time, average time diff) data table
-> Coming up a statistics or technology to manage the data: Kmeans clustering method
-> Finding proper git command for finding the data:
'git log -p --no-merges --pretty=format:\"%an:%ct" v4.1..v4.9'
-> Creating a program to automatically collecting data from Linux-stable
-> Creating a program to automatically clean the data output
-> Applying the technology to the data and realizing it with python
-> Analyzing the results and conclude it, writing them down in report.

4.2. Design of the program for collecting data



4.3. Design of the analysis program



5. Analysis of technology

5.1. Why Kmeans?

Kmean algorithm is used for clustering operation, two classification and unsupervised learning. The authors are clustered according to the number of submissions and the average time difference. Because our group put forward the assumption that the authors can be divided into different classes, and decision tree, linear regression and other methods can not classify unknown classification, only clustering algorithm can automatically cluster the input. So we choose clustering algorithm.

5.2. The advantages&disadvantages of Kmeans

Advantages:

K-means algorithm is a classical algorithm to solve the clustering problem. The algorithm is simple and fast. For processing large data sets, the algorithm is relatively scalable and efficient. This algorithm often ends with local optimum. The algorithm tries to find the K partitions which minimize the square error function. When the cluster is dense, spherical or clustered, and the difference between clusters is obvious, its clustering effect is very good.

Disadvantages:

It is sensitive to outlier and noise. If a noise point is added to the dataset, the noise point is separated into a class. If k = noise, the rest of the data can only be classified into one class if they are convex points. The so-called convex data set refers to each pair of points in the set, and each point on the line segment connecting the two points is also in the set. Generally speaking, the shape of K-means clustering can only be spherical, which can not be generalized to arbitrary shape.

6. Analysis of data

After collection of data using the program ‘get_author_info.ipynb’ , we get the data table as expected.

6.1. Data snooping

Use the snooping tools in pandas:

```
>>>df = pd.read_csv('result.csv')
>>>df.shape
(5052, 3)
>>>df.isnull().sum()
author        0
times         0
average       0
dtype: int64
>>>df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5052 entries, 0 to 5051
Data columns (total 3 columns):
 #    Column     Non-Null Count    Dtype
---   ------     --------------    -----
 0    author     5052 non-null     object
 1    times      5052 non-null     int64
 2    average    5052 non-null     int64
dtypes: int64(2), object(1)
memory usage: 118.5+ KB
```

```
>>>df[0:3]
                author   times     average
0    Linus Torvalds        174      266246
1    Hauke Mehrtens         59      853094
2    Luuk Paulussen          2    26549028
```

Shows that the table is consist of 5052 records, and each record is consist of author(name of a developer), times(time count of commits of a developer) and average(average commit cycle in seconds), moreover, there is no invalid data like 'NaN', which means data is easy to clean.

Snooping the data in 'average':

```
>>>df["average"].describe()
count       5.052000e+03
mean        1.985631e+06
std         4.169225e+06
min         0.000000e+00
25%         0.000000e+00
50%         4.195405e+05
75%         2.043568e+06
max         4.555536e+07
Name: average, dtype: float64
>>>(df["average"] == 0).sum()
1816
```

It can be found that, there are some '0' average commit cycle. There are two possible situations of it:

- Some developers have ever committed only once.
- Some are meaningless records that need to be deleted.

Selecting the columns that 'average' is zero, and inspecting their 'times' to see whether all equal to '1':

```
>>>df1 = df[df['average'].isin([0])]
>>>(df1['times']!=1).sum()
181
```

Unfortunately, there are 181 records considered to be meaningless and need to be delete. So drop these useless data:

```
import pandas as pd


df = pd.read_csv('result.csv')
df1 = df[df['average'].isin([0])]
df1 = df1[~df1['times'].isin([1])]
index = df1.index
for i in index:
```

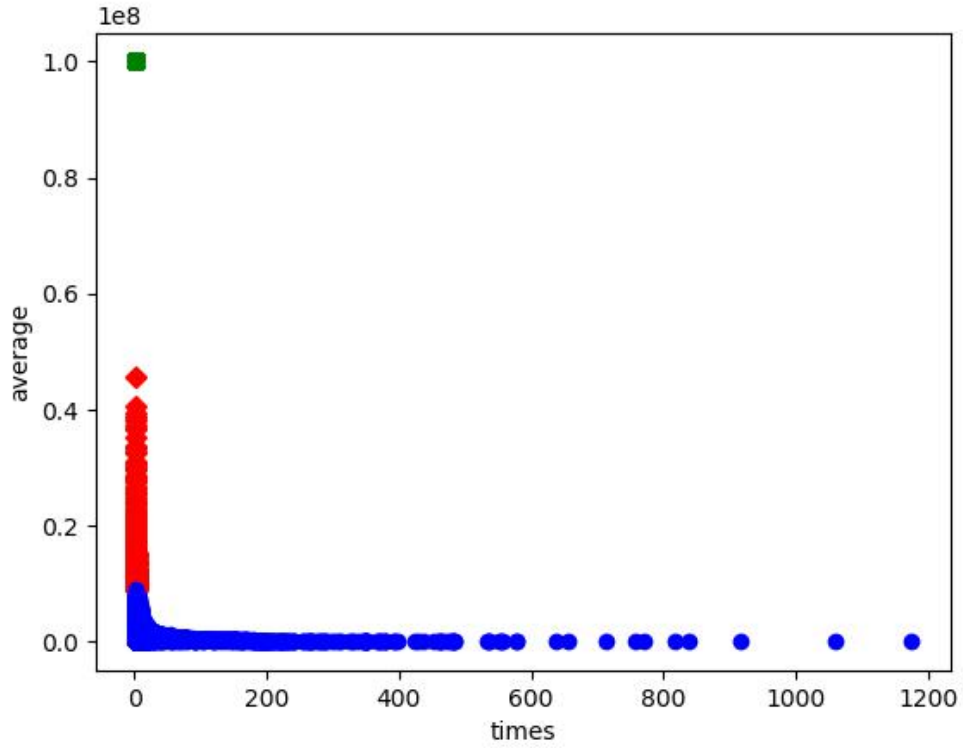
```
        df.drop([i], inplace=True)

df.to_csv('clean_result.csv')
```

Now the cleaned data is much better for model.

6.2. Result

Deciding 3 clusters for developers instead of 2 which was suggested by Sihouette coefficient, because those developers who only commit once can be some what considered as outliers.

By using 'Kmean.py' program, we get the cluster diagram:



Which shows that those whose commit times are more and average cycle are less can be considered as one clusters(the most diligent developers), and those who only commit once is another cluster(participants), the rest are in another cluster(normal developers).

7. Conclusion

7.1. Preliminary conclusion

As we get the diagram with developers in reasonable clusters as expect, we do not reject H0: All developers in kernel can be clustered in terms of contribution to kernel. Which means that in kernel development community, the contribution to kernel is different among developers. We divided them into three clusters and we would like to call them as Residents, Builders and Participants.

The conclusion above reflects the quality of kernel development that even as an open

source community, everyone can contribute to its development, but the development of kernel still relies heavily on the minority.

7.2. Data robustness

In the end, we analyse the data again and find out the the data we draw our conclusions on is not so robust. Here are some reasons:

- The nearly 5000 pieces of data is too short, the shortage of data not only contribute to the inrobustness of model, but also can not fully reflect the clustering of whole 20k+ developers.
- The data only from v4.1-v4.9, which is a relative short development period, maybe clustering of developers is highly influenced by patchlevel: at the beginning of the kernel development, all the developers highly contributed to development. Only the case in v4.1-v4.9 can not overcome the issue.
- The same developer could be different in different development period. A developer may be a good developer at the beginning, but due to the change of job or family, he/she fail to pay attention to kernel development. So the data here can only represent the developer clustering in one period.

So the conclusion is just reflect the problem from one small aspect, not the actual situation.