Data Science Programming I Final Project

Introduction to Programming II

Team members： HaoQiu Yan 320180940440

YinRu Ye 320180940480

QiYuan Zhang 320180940541

WanFeng Zhu 320180940691

YuHao Zhang 320180940581

YunHao Kang 320180939851

YuMing Dong 320180939701

JiaChuan He 320180939771

Instructor: Dr. Nicholas Mc Guire

June 18, 2020

# Content

1. Introduction
   1. Abstract
   2. System
   3. Techniques used
   4. Requirement
2. Design
   1. Process
   2. Codes
   3. Operation
   4. Optimization
3. Feedbacks and reports
   1. Feedback
   2. Configuration
   3. Reports
   4. Data modeling
4. Background
   1. Hardware
   2. Software
5. Appemdix
   1. Model Test
   2. Coefficient Significance
   3. Residual Chart
   4. Decision Tree
   5. Summary

# 1 Introduction

## 1.1 Abstract

The quality of the code in the linux-stable library is uneven. We want to crawl through certain data to find out what the quality of the code is mainly related to.  
Based on the results of the panel discussion, we assume that the number of submissions and the average submission time point are the main factors affecting the quality of the code in the file  
This document mainly shows our development goals and process. All the results are based on the coding after team discussion and search, and the conclusion after the model is established.

## 1.2 System

·Describe

The system is based on python3.8 and linux-stable database, through machine learning to capture the required parameters and final modeling. The ultimate goal is to establish a reasonable model to verify the hypothesis and the mapping relationship established by the corresponding hypothesis based on the crawled data.

·Target

Collect the data in the linux-stable library to explore the factors and indicators related to the code quality of each file and the relative strength, at the same time verify the hypothesis and consolidate knowledge.

·Vision

Learning data collection, cleaning and modeling, and the writing of related documents are the practical applications of the theoretical knowledge in the class, and the training of data scientific analysis ability and application ability to machine learning

## 1.3 Techniques used

·Use python code try except basic loop, if statement and return statement

·Reference the POPEN and PIPE technologies of the subprocess library to store the results of the git command to the pipeline

·Referenced the re library for regular expressions to express the git command to be executed

·The pandas library is used to convert the results in the pipeline into a dataframe, and use its to\_csv attribute to save the results to csv

·The unicodedata library is used to encode the results in the pipeline in UTF8

·Cited the time format of the time library and converted it into a format that is easy to calculate

·Machine learning uses the linear regression model (LinearRegression) and regression decision tree model (DecisionTreeRegressor) of daol sklearn

·Modeling uses the train\_test\_split method and randomly generates the training set and test set

·The min\_max\_scaler = preprocessing.MinMaxScaler() method is used for data normalization, which is convenient for modeling and learning later

## 1.4 Requirement

**Non-Functional**

Req1 The system can continue to optimize

Req2 Data collection is the first step to be carried out, the lunix-stable library must be used

Req3 The system feeds back whether our assumptions are related to mapping through modeling

Req4 It should be possible to exclude the occasional interference of some abnormal data

Req5 The system is composed of multi-module design, multi-person division of labor cooperation

Req6 Most data sets should be valid after collection, cleaning and snoop

**Functional**

Req1 The system must be processed before modeling

Req2 The system provides text information collected by different codes

Req3 The system feeds back whether our assumptions are related to mapping through modeling

Req4 Multi-module combination work to improve efficiency

Req5 Errors reported by different modules can be optimized separately

**Technical**

Req1 The system is started based on python3.4 and machine learning

Req2 The data set needs a large amount to eliminate chance

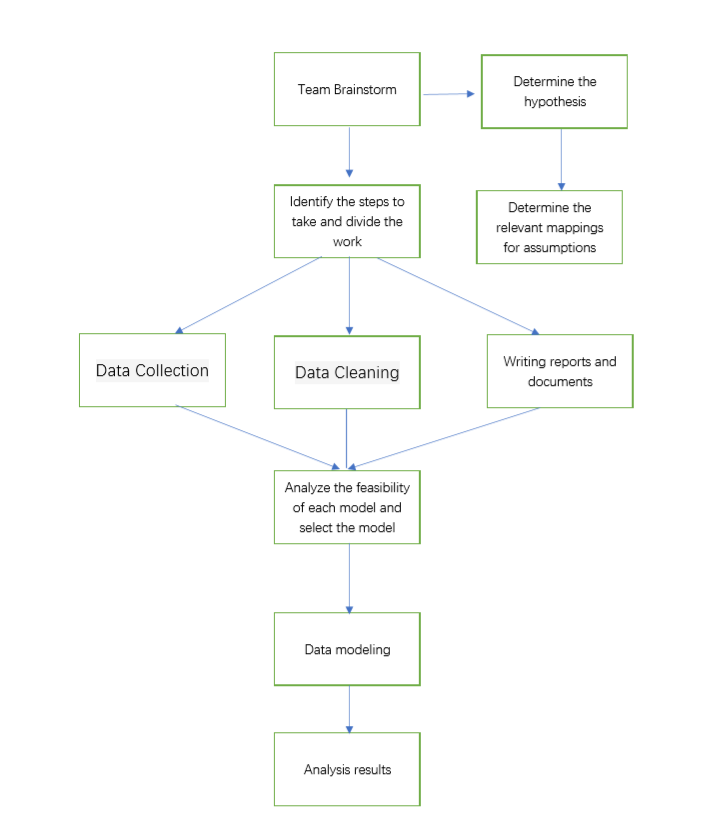
Req3 Only after all the modules can be modeled

Req4 The system can be used on computers with Windows and iOS

Req5 The system can collect monitoring data sets and conduct final modeling to query whether they meet the assumptions

# 2 Design

## 2.1 Process



|  |  |
| --- | --- |
| Member | Mission |
| JiaChuan He | Data harvesting concept and implementation  get the dataset from git kernel and save them into csv. |
| QiYuan Zhang | Data collection, get the required data set through git. Save the data set to a csv file |
| YunHao Kang | Data cleaning concept (analysis and ”ﬁxing”) and implementation  （eg.remove the repeat data, fill the null value, and handle the outlier value. |
| HaoQiu Yan | Technology selection and evaluation Select and evaluate appropriate data analysis techniques. (Eg. correlation analysis, decision tree, convolutional neural network) |
| YuMing Dong | Data analysis (according to algorithm training test verification) |
| YinRu Ye | Use appropriate techniques to process the cleaned data set. |
| YuHao Zhang | Interpret the results, data modeling Interpret the results of data analysis. (Eg. Assuming the establishment or not, reasonable or not) |
| WanFeng Zhu | Mapping report and documents(According to the analysis results and all the steps) |

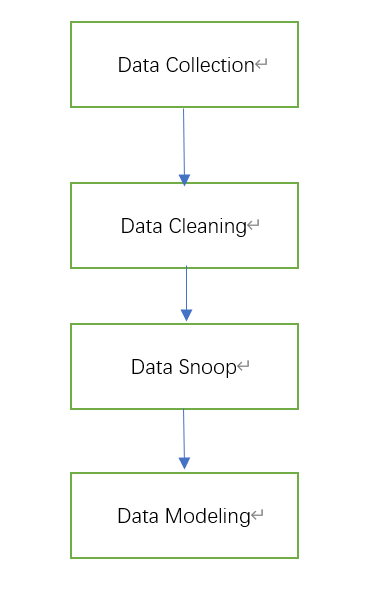
**Multi-module design**

The code is divided into four modules, namely data collection, data cleaning, data monitoring, and data modeling, allowing us to focus on different data sets or distribute the results of different parts of the code

Even if the code of a module is wrong, it does not delay the operation of other parts, only need to run the test results separately to find the point of error. Moreover, the multi-module design is helpful for the division of labor and cooperation. There is no need to look at the code completed by others. Take a look at the documentation to know the variable name and write the code for each step.

## 2.2 Codes

**Structure of code**

****

The system consists of four model folders (five python files).

**Data Collection**

Each file is taken as an object which belongs to StableFiles class.

1)StableFiles class has methods of blame() to return “git blame” results from git

2)and gitFixCommits() to get “git log” results from git

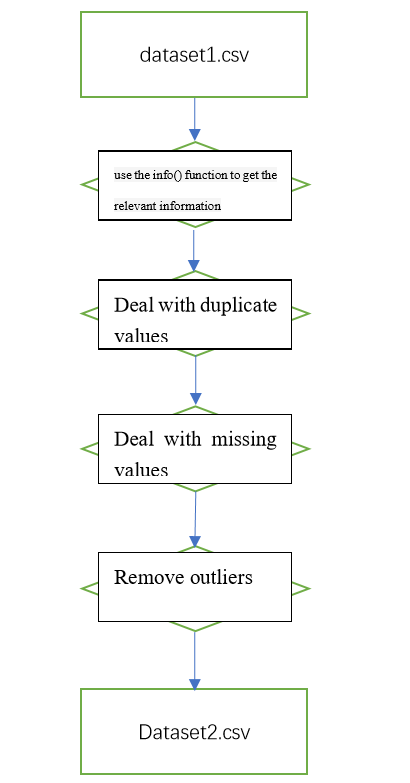
3)and count fixes\_percent, total\_shas and last\_fixes. count\_author() is used to count the number of persons who wrote these lines.

4)count\_commit() is used to count the number of shas among the existing lines of files.

5)count\_avetime() and count\_lines() are used to count the average time and file lins respectively.

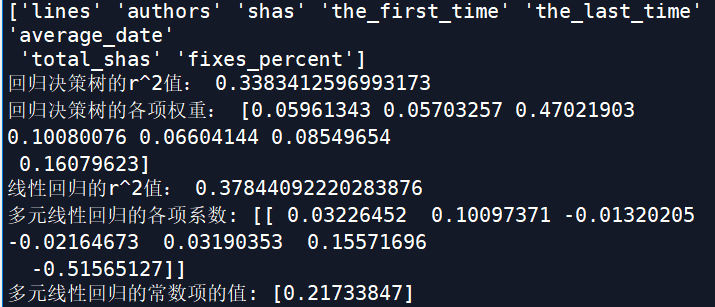
6)Then, run() function could create StableFiles objects, call StableFiles.getFeatures(), store the whole data sets as a dataframe and write into “results.csv”.

**Data Cleaning**



## 2.3 Operation

This project needs to put the code of all modules into a folder and then run it directly with python. You can get the processed data and the model diagram based on the data. It mainly includes 5 python files and 4 module folders.



## 2.4 Optimization

Data collection:

\*In order to reduce the running time, we write csv after the characteristics of all files are collected, instead of writing the characteristics of files one by one. Since we have such a large data set, reducing run time and accelerating data collection are crucial.  
\*After obtaining it from linux-stable, the characteristics of the file are stored as a dictionary and appended to a list of lines. The row list is written in "results".  
\*We deleted and saved the wrong files and lines in the "wrong\_file" list and "Badtime.txt".

During data cleaning, we believe that the following optimization points are as follows  
\*The value of "total\_shas" of some data is small (<5), which has a certain chance. You can consider deleting it in the later data preprocessing.  
\*When drawing a box diagram, I found that there are still many data above the upper limit of Q3+1.5IQR, and these data can also be regarded as outliers.

# 3 Feedbacks and Reports

## 3.1 Feedback

When writing the first part of the code, a total of four troubles were encountered.

The first is how to find the file name and the absolute path of the current file, but it was quickly solved.

The second one is just grabbing the C file, and it was quickly resolved.

The third problem is to change "\\" to "/". This is necessary because in Windows, file paths are usually separated by "\\", but in python, we only need "/".

The fourth difficulty is to simplify the file path so that collectv2.py can run smoothly and quickly.

In fact, first consider the method of changing the absolute path to a relative path. However, I found it very troublesome, so the slice file path was used in the end.

However, writing into a csv, after all the files’ features were collected, runs the risks of data lost. If the 7999th file meets an unexpected error and process finishes with an exit code, previous features of 7998 files will not be saved and be lost. To miss that maddening case, we use try…except to catch any Exception outside the for loop of sample file traversal. If process finishes with an exit code, previous features will be stored into “results.csv” safely.

Besides, it is difficult to extract shas, authors, and time from “git blame --no-merges v3.0..HEAD FileName”. Inquiry results are not uniform and the column indexes are dynamitic relative to the maximum length of authors and file paths. We find out the solution of String Splitting Twice. Firstly, inquiry results were read as strings line by line; then, we splitd a line with ‘)’ firstly and discard line contents at the end; after that, we located shas and time according to their length, which are constant; finally, we split the remaining secondly and joined them to get authors.

But we are confused that there is a little of files or lines in our sample which does not have time information term when using “git blame --no-merges v3.0..HEAD FileName”.

## 3.2 Configuration

The project is suitable for a variety of situations, the most important is to facilitate maintenance. What we explore is what qualities of excellent code in a large amount of code, and we will also standardize the code according to that standard. It is easy to maintain, git upload by multiple people, and constantly modify the log record code changes, regardless of whether the results meet our initial assumptions, and strive to standardize their own code and continuously optimize is our goal

## 3.3 Reports

Mapping report:

Assumption: In the linux-stable library, the number of commits in the file and the average commit time are the main factors that affect the quality of the code in the file  
Corresponding data: the number of submitted shas (equivalent to the submission code), the earliest submission time and the latest submission time in a file (used to calculate the average submission time point), and also grabbed for comparison Other indicators: number of developers, number of lines of code, number of all submissions

Collection data:

file\_name: Files’ relative path in linux-stable.git

lines: Files’ lines, which does include blanks lines

Authors: the number of persons who wrote these lines of the files, from v3.0 to now

Shas: the number of shas, among the existing lines of files, from v3.0 to now

the\_first\_time: the earliest time of commits, among the existing lines of files, from v3.0 to now

the\_last\_time: the latest time of commits, among the existing lines of files, from v3.0 to now

average\_date: the average time of commits, among the existing lines of files, from v3.0 to now

total\_shas: the total number of commits, from v3.0 to now

last\_fixes: the latest shas of commit with “Fixes: SHA (text)" line

fixes\_percent: the proportion of commit with “Fixes: SHA (text)" line against the total commit starting with “commit:”

In details, lines, authors, shas, the\_first\_time, the\_last\_time, average\_date were from command “git blame --no-merges v3.0..HEAD FileName”. fixes\_percent, total\_shas, last\_fixes were from "git log -p --no-merges --date-order v3.0..HEAD FileName”.

## 3.4 Data modeling

This part is about data modeling technology's implementation and applying the dataset to the model. It will include the explanation for each step, the code style and the meaning for each parameters' name.

**The Preprocession for Data**

After data cleaning, the first thing is to read the result csv file as a dataframe for using relative attributes and functions. The file invokes two csv file, one for acquiring the names for columns and the other for getting the specific x values and y values. The main method in this work is dataframe's "iloc" function. It can get a section of the dataframe by setting suitable index. All the processes need to import the package of pandas. Before we start to build a model,we found that the data in our dataset gathered at the corner of two axies, which is quite hard for computer to understande the relationships between different features. We import log from math package to get the logarithm result for each featurea and it works(we will clarify this in the following part). The method is to use dateframes' "apply" function.

In order to aviod value error and type error, we transform the datatype of the data into numpy's float. In the next block, we use the python package of seaborn to plot the pictures(each features versus y\_value) by writing for loop. We also draw a heat map to illustrate the relationships between each parameters by producing correlation matrix first. All the pictures are saved as png file.

**Generating Train Set and Test Set**

The main method used here is train\_test\_split, it will randomly generate test set and train set for x and y respectively after you set the test size and random state.

**Normalization**

Each feature differs greatly, and the greater its value is , the greater the impact will be caused on the prediction results. Limiting the values to the same magnitude ensures that all data are equally important. This step mainly use preprocessing.MinMaxScaler(), import preprocessing from sklearn. A linear transformation of the original data can make the resulting value map between [0, 1].

The algorithm is as follows:



Max is the maximum value in the sample, min is the minimum value.

**Techonology Selection**

Multi-linear Regression

The purpose of this final project is discussing the relation between the features and between each feature and the predictor. The multi linear regression is the first choice we think of. This method with less complex algorithm can build a model fast no matter how big the dataset is. Becasuse the weight coefficient can be displayed directly, the result of multi linear model is easily to be understood and explain. This model uses residual sum of squares (SSE).



DecisionTree Regressor

Due to the complicated relations of the features, linear model as an assumption may not be fit. The DecisionTree Regressor regard mean square error(MSE) as basis on node splitting. The predicted value is the mean of the samples that fall within the leaf nodes. The calculation of DecisionTree Regressor is not very complex as well. It can process the continuous data without linear relation and high-dimension data. Most importantly, it is a supervised machine learning algorithm.

**Modeling**

The Multi-linear regression is based on the sklearn package's LinearRegression. The Regression decsion tree is based on DecisionTreeRegressor in the same sklearn package. The x\_test set is input and the output is the result of predict(x\_test). We choose r^2\_ score as the metric of judging the model. r^2 is SSR(Explained Sum of Squares)/SST(Sum of squares for total). The more closely it approach, the more accurate model is.

At last, we use lr\_coef for weight in linear regrssion model, \_importances for decsion tree regressor model to compare and conclude.

# 4 Background

## 4.1 Hardware

Setup:

•Laptop

•Linux System

## 4.2 Software

Version number used in the project:

matplotlib 3.2.1

numpy 1.18.2

pandas 1.0.3

scipy 1.4.1

setuptools 41.2.0

six 1.14.0

sklearn 0.0

subpreocess unknown

unicodedata unknown

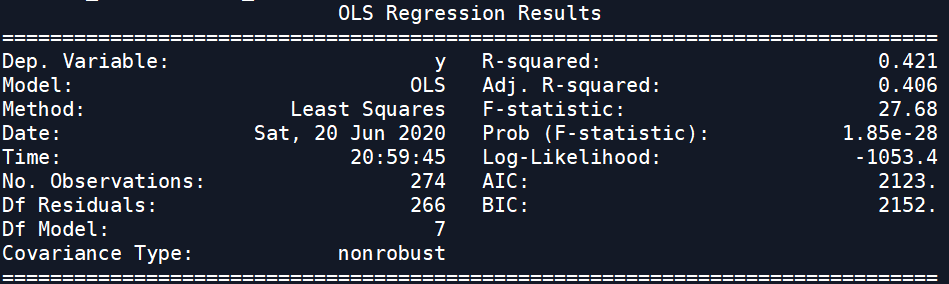
time unknown

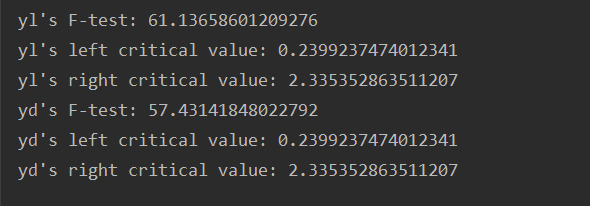
math unknown

seaborn unknown

# 5 Appendix:

Multiple Linear Regression





## 5.1 Model Test

In ftest.py, we take a Two Tailed Test of the multiple linear regression. Outputs are attached with the document as pictures.

1. Hypothesis (Two Tailed Test)

H0: b1 = b2 = b3 =b4 = b5 = b6 = b7 = 0

H1: At least there is a non-zero coefficient.

1. Decision Rule

First, We choose f\_test as test statistics, which is in F Distribution with freedom of (7, 275-7-1).

Second, the critical values at the significance of 0.05 are:

P(f<f(left critical value)) = 0.025, P(f>f(right critical value)) = 0.025

Then, we import f of scipy.stats. The left critical value is about 0.23992, and the right critical value is about 2.33535.

1. Test Statistics

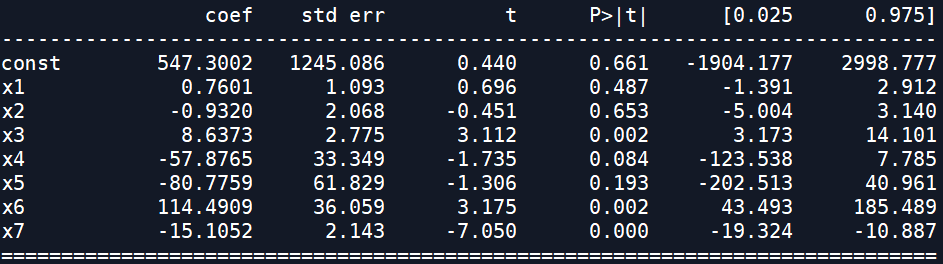
f\_test could be calculated from (SSR / p) / (SSE / (n - p - 1)). SSR is Sum of Squares due to Regression, SSE is Sum of Squares due to Error.

From ftest.py, f\_test = 57.43142.

1. Consion

From b and c, we can find that f\_test (57.43142) is much larger than right critical (2.33535) value. It implies that the multiple linear regression based on seven predictors is bad, since we do not have 0.05 significance to refuse H0.

## 5.2 Coefficient Significance

Similarly, we take Two Tailed Tests of b1- b7, respectively. By importing and calling summary() from statsmodels.api, we got p-values of coefficients’ test statistics. Output is shown as follows. 

From column “P>|t|”, the significance of rejecting hypothesis “coefficient=0” is:

(line) x1—0.974

(authors) x2—0.694

(shas) x3—0.004

(first time) x4—0.168

(last time) x5—0.386

(average time) x6—0.004

(total\_shas) x7—0.000

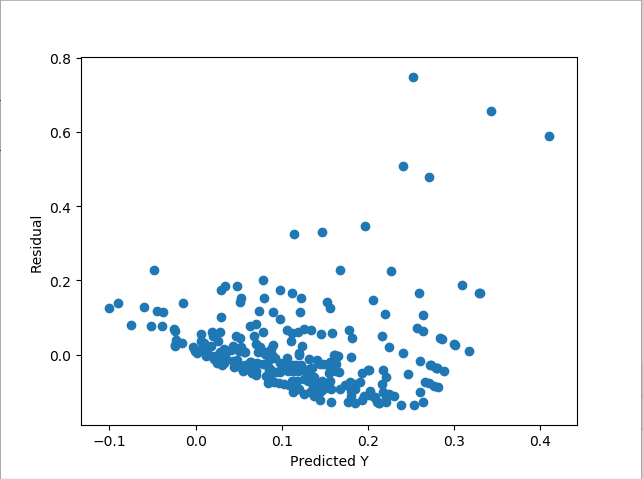
total\_shas has the minimal significance and is high relative to fixes percent. average time and shas come second in significance. On the other hand, the probability of line is closed to 1, which may imply that the file line has no relationship with fixes percent. authors also has relatively high significant in this multiple linear regression.

In conclusion, it suggests that total\_shas and average time are essential in linear regression, while lines and authors are redundant.

Besides, considering total shas represents the total commits of files from v3.0 to HEAD, shas is the number of commits among existing lines, and both have low significance, total\_shas and shas may have dependency, which needs to be explored furtherly.

## 5.3 Residual Chart

The residual chart is shown as follows.

Summary：

**Multiple Linear Regression**

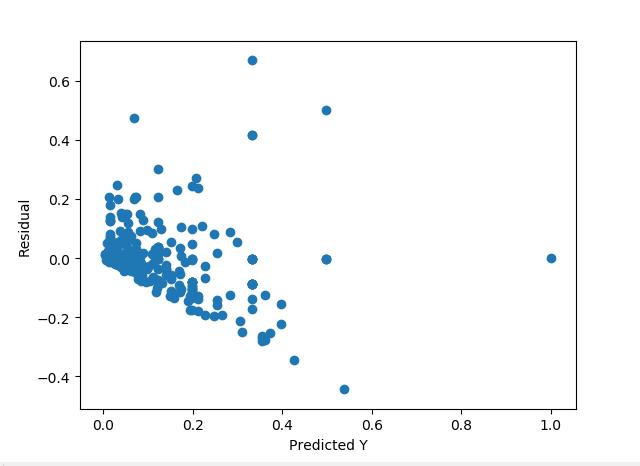
In this photo, the Residual and the Predicted Y show a non-linear trend and are not independent of each other.

Solution：

It would be better to change a new curve inertia instead of linear regression. Also, Y may have an autocorrelation.

## 5.4 Decision Tree

1) Residual Chart

The residual chart is shown as follows.

**Decision Tree**

## 5.5 Summary

This figure shows that the relationship between the residual value and the predicted value Y is linearly abnormal, and the variance of the residual is not the same, which increases as the predicted value of y increases.

Solution:

This requires transforming Y or other solving methods to eliminate this effect.