**Perform Analytics on a 5000 entry dataset.**

**CA5**

**B8IT105 Programming for Big Data**

**Janos Vajdics**

* + **Assignment 5 is based on transforming a large dataset in text format - over 5000 lines of text.**

**You will need to scrub (clean) the data and place it into the relevant holder/container objects.**

**Once in these objects you will see that there are 422 different sets of commit objects.**

**So your task will be to analyse these 422 objects that are in a list and come up with 3 interesting statistical pieces of information for this dataset with supporting evidence of "interestingness'**

**You code for calculating the analysis should be documented and tested.**

**Test should be in a separate file runnable from the command line.**

**Your statistical analytics conclusions should be in a word document explaining in approximately 500 words the information that you have gleamed from the dataset.**

Assignment is based on transforming a large database in text format- over 5000 lines of text (**changes\_python.log** text file).

First of all, I cleaned the data and place it into a relevant holder using the following files, after I transferred the database into Tableau and analysed it.

**Process \_changes.py**

# open the file - and read all of the lines.

changes\_file = 'changes\_python.log'

# use strip to strip out spaces and trim the line.

#my\_file = open(changes\_file, 'r')

#data = my\_file.readlines()

data = [line.strip() for line in open(changes\_file, 'r')]

# print the number of lines read

print(len(data))

sep = 72\*'-'

# create the commit class to hold each of the elements - I am hoping there will be 422

# otherwise I have messed up.

class Commit:

'class for commits'

def \_\_init\_\_(self, revision = None, author = None, date = None, comment\_line\_count = None, changes = None, comment = None):

self.revision = revision

self.author = author

self.date = date

self.comment\_line\_count = comment\_line\_count

self.changes = changes

self.comment = comment

def get\_commit\_comment(self):

return 'svn merge -r' + str(self.revision-1) + ':' + str(self.revision) + ' by ' \

+ self.author + ' with the comment ' + ','.join(self.comment) \

+ ' and the changes ' + ','.join(self.changes)

commits = []

current\_commit = None

index = 0

author = {}

while True:

try:

# parse each of the commits and put them into a list of commits

current\_commit = Commit()

details = data[index + 1].split('|')

current\_commit.revision = int(details[0].strip().strip('r'))

current\_commit.author = details[1].strip()

current\_commit.date = details[2].strip()

current\_commit.comment\_line\_count = int(details[3].strip().split(' ')[0])

current\_commit.changes = data[index+2:data.index('',index+1)]

#print(current\_commit.changes)

index = data.index(sep, index + 1)

current\_commit.comment = data[index-current\_commit.comment\_line\_count:index]

commits.append(current\_commit)

except IndexError:

break

print(len(commits))

commits.reverse()

for index, commit in enumerate(commits):

print(commit.get\_commit\_comment())

output\_file = 'changes.csv'

my\_file = open(output\_file, 'w')

my\_file.write('Revision, Author, Date # Lines, Comment, Files Changed \n')

for commit in commits:

my\_file.write(str(commit.revision) + '.' + commit.author + ',"' + commit.date + '",' + str(commit.comment\_line\_count) + ',"' + ''.join(commit.comment) + '",' + ' - '.join(commit.changes) + '\n')

my\_file.close()

After used **Read\_changes**.**py** file.

changes\_file = 'changes\_python.log'

data = [line.strip() for line in open(changes\_file, 'r')]

print(len(data))

sep = 72\*'-'

commits = []

index = 0

while index < len(data):

try:

details = data[index + 1].split('|')

commits.append(details)

index = data.index(sep, index + 1)

except IndexError:

break

output\_file = "changes.csv"

my\_file = open(output\_file, "w")

my\_file.write("Revision, Author,Date\n")

for details in commits:

my\_file.write(details[0] + ',' +

details[1] + ',' +

details[2] + '\n')

my\_file.close()

Then **simple.py** file

read\_file(changes\_file):

# use strip to strip out spaces and trim the line.

data = [line.strip() for line in open(changes\_file, 'r')]

return data

def get\_commits(data):

sep = 72\*'-'

commits = []

current\_commit = None

index = 0

while index < len(data):

try:

# parse each of the commits and put them into a list of commits

details = data[index + 1].split('|')

# the author with spaces at end removed.

commit = {'revision': details[0].strip(),

'author': details[1].strip(),

'date': details[2].strip(),

'number\_of\_lines': details[3].strip().split(' ')[1]

}

# add details to the list of commits.

commits.append(commit)

index = data.index(sep, index + 1)

except IndexError:

break

return commits

if \_\_name\_\_ == '\_\_main\_\_':

# open the file def- and read all of the lines.

changes\_file = 'changes\_python.log'

data = read\_file(changes\_file)

commits = get\_commits(data)

# print the number of lines read

print(len(data))

#print(commits)

print(commits[0])

print(commits[1]['author'])

print(len(commits))

And **test\_simple.py** file.

import unittest

from simple import get\_commits, read\_file

class TestCommits(unittest.TestCase):

def setUp(self):

self.data = read\_file('changes\_python.log')

def test\_number\_of\_lines(self):

self.assertEqual(5255, len(self.data))

def test\_number\_of\_commits(self):

commits = get\_commits(self.data)

self.assertEqual(422, len(commits))

def test\_first\_commit(self):

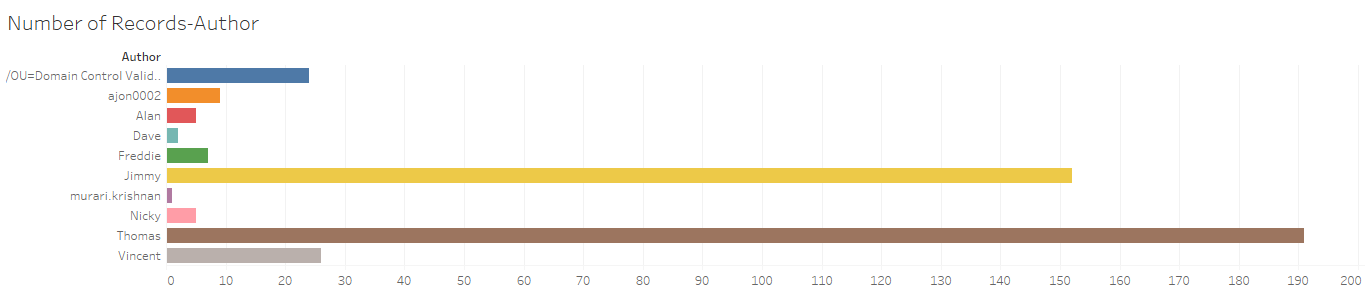
commits = get\_commits(self.data)

self.assertEqual('Thomas', commits[0]['author'])

self.assertEqual('r1551925', commits[0]['revision'])

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

In our large dataset, there is a large amount of text data available where the system recorded several authors daily activities regarding file changing, comments and revisions. The system also recorded this by date, exact time also that on which day was the comment or notification made.

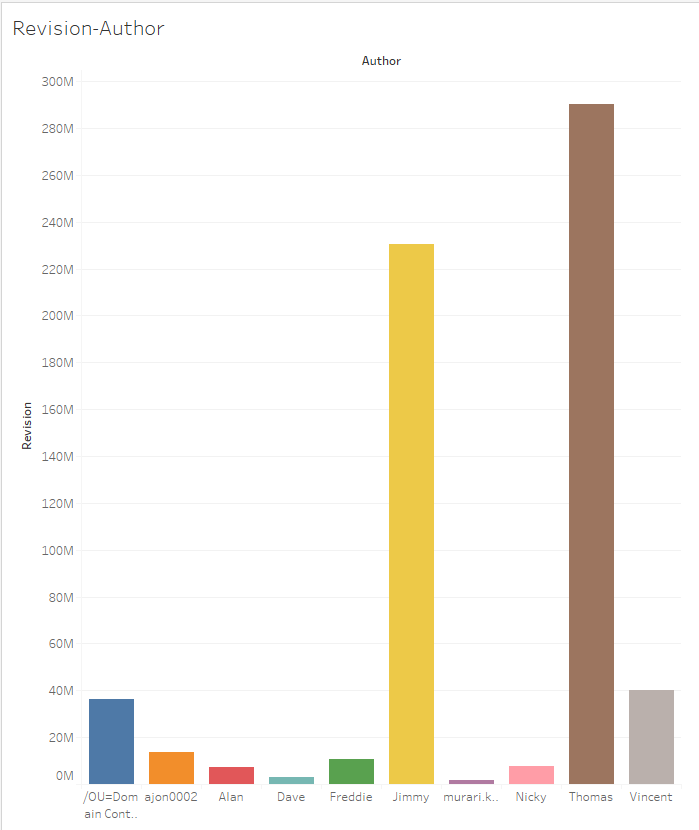
Our task is to transform, clean and then appropriately analyse 3 interesting statistical pieces of information for this dataset.

In the first graph, we analysed the number of records with the different authors.

The name of the authors can be seen on the vertical access with different colours while the number of the records can be found on the horizontal access. However, the domain control is included in this data, his activities are shown just for information purposes only on the very top of the graph. The authors namely are Ajon, Alan, Dave, Freddie, Jimmy, Murari Krishan, Nicky, Thomas and Vincent.

After the analysis of these two data we can see that the most active author is Thomas with almost 190 records, followed by Jimmy with 150 records. There is a big drop after these numbers as Vincent’s record is 25, followed by Ajon just above 10, then Freddie’s activity is just under 10 around seven or eight. The most inactive authors are Nicky (5), Alan (5), Dave (3) the Murari (1). Their numbers are shown in the brackets after their names.

In our second graph, we compared the authors with the number of revision.

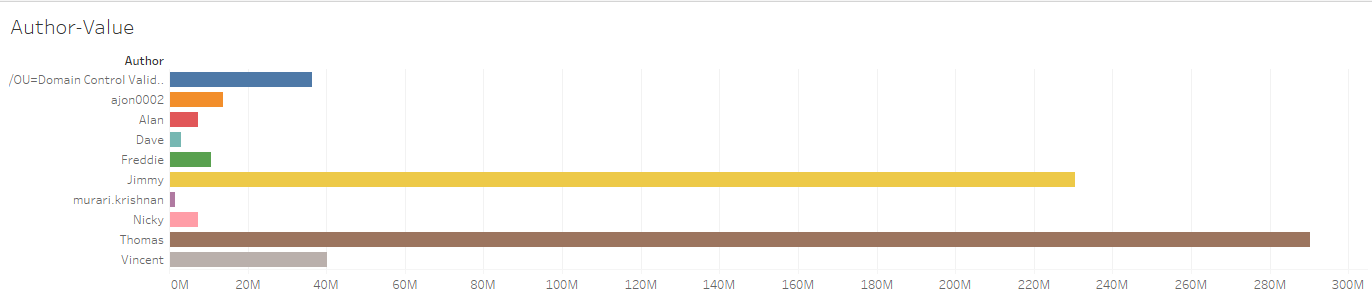
The name of the authors can be seen on the horizontal access with different colours while the number of the revisions can be found on the vertical access.

The colour code of each author remained the same as in the first graph to make an easier comparison and author detection. After the analysis of these two data we can see that the most active author is Thomas with almost 290 M revisions, followed by Jimmy with almost 230 M revisions. Just it could be seen in the first table there is a big decline in the number of revisions this time as well.

Vincent’s revision number is around 40 M, followed by Ajon with around 15 M, then Freddie’s activity is just above 10 M around 11-12 M of revisions. The most inactive authors are Nicky (10 M), Alan (8-9 M), Dave (5-6 M) the Murari (3-4 M). Their numbers are shown in the brackets after their names this time as well.

In our third graph, we created a graph where we can see how much value each of the authors are generating.

The name of the authors can be seen on the vertical access with the same colours as in the previous graphs while the number of the value can be found on the horizontal access.



Knowing the results of our two previous data analysis in the terms of records and revision numbers there’s a very clear evidence, that most active and most effective author who does the most valuable job is Thomas (290M), followed by Jimmy (230M). The third most productive author is Vincent with 40M, followed by Ajon just above 10M, then Freddie’s productivity is around 10M. The most inactive authors are Nicky (just under 10M), Alan (just under 10M), Dave (just under 5M) the Murari (1-2M). Their numbers are shown in the brackets after their names.

After the evidence based analysis of the large dataset we can make the following statements and conclusions:

* we analysed the number of records with the different authors, then
* we compared the authors with the number of revision.

These two numbers created a graph where it can be clearly seen which author does the most valuable job on the everyday basis:

1. Thomas
2. Jimmy
3. Vincent
4. Ajon
5. Freddie
6. Nicky
7. Alan
8. Dave
9. Murari