ECE/CS 498 DS HW 1
Spring 2020

Name: **Instructor Solution**

Netid: **Instructors**

Registration Status: **Registered** ☺

A. Data Structure to Parse Raw Log File

- Provide a (i) diagram and (ii) brief explanation of the data structure you used to parse the raw log file

Example (one of many acceptable ones – make sure they have some sort of diagram/table and reasonable explanation)

A list of lists in Python was used to parse the raw log file

Each entry of the outer list was itself a list that contains all the information about a certain page fault. All entries in the inner lists were string elements (to better facilitate writing to the csv file afterwards). We will fill out an example for the first page fault from the raw log file.

The first seven cells of each inner list was a “header”, structured as follows. Note that the page fault address was first converted to an int from a hex string, and then back into a string.

Time	**Process Name**	**PID**	**Page Fault Address**	**RW Access**	**Major/ Minor**	**Resolve Time**
'1506816069251'	'firefox'	'13179'	'10773289646'	'R'	'minor'	'50'

The remaining entries *3n* elements in each inner list contained 3 elements for each of the *n* backtrace (BT) entries for that page fault, structured as follows. Note that again all the elements are strings, but each address and offset field were first converted from hex string into an int and then back to a string. For our example, *n*=5.

BT Entry 1 lib	**BT Entry 1 Address**	**BT Entry 1 Offset**	**…**	**BT Entry *n* lib**	**BT Entry *n* Address**	**BT Entry *n* Offset**
'/usr/lib/x86_64-linux-gnu/libcairo.so.2.11400.10'	'16727808'	'686943'		'/lib/x86_64-linux-gnu/libc-2.26.so'	'16767232'	'7501'

B.a. Time Range Covered By Data

- Start Time: 10/01/2017 at 12:01:09.251 AM
- End Time: 01/07/2018 at 6:59:50.839 PM
- Total Duration: 98 Days, 18 hours, 58 minutes, and 41.588 seconds

B.b. Unique Processes

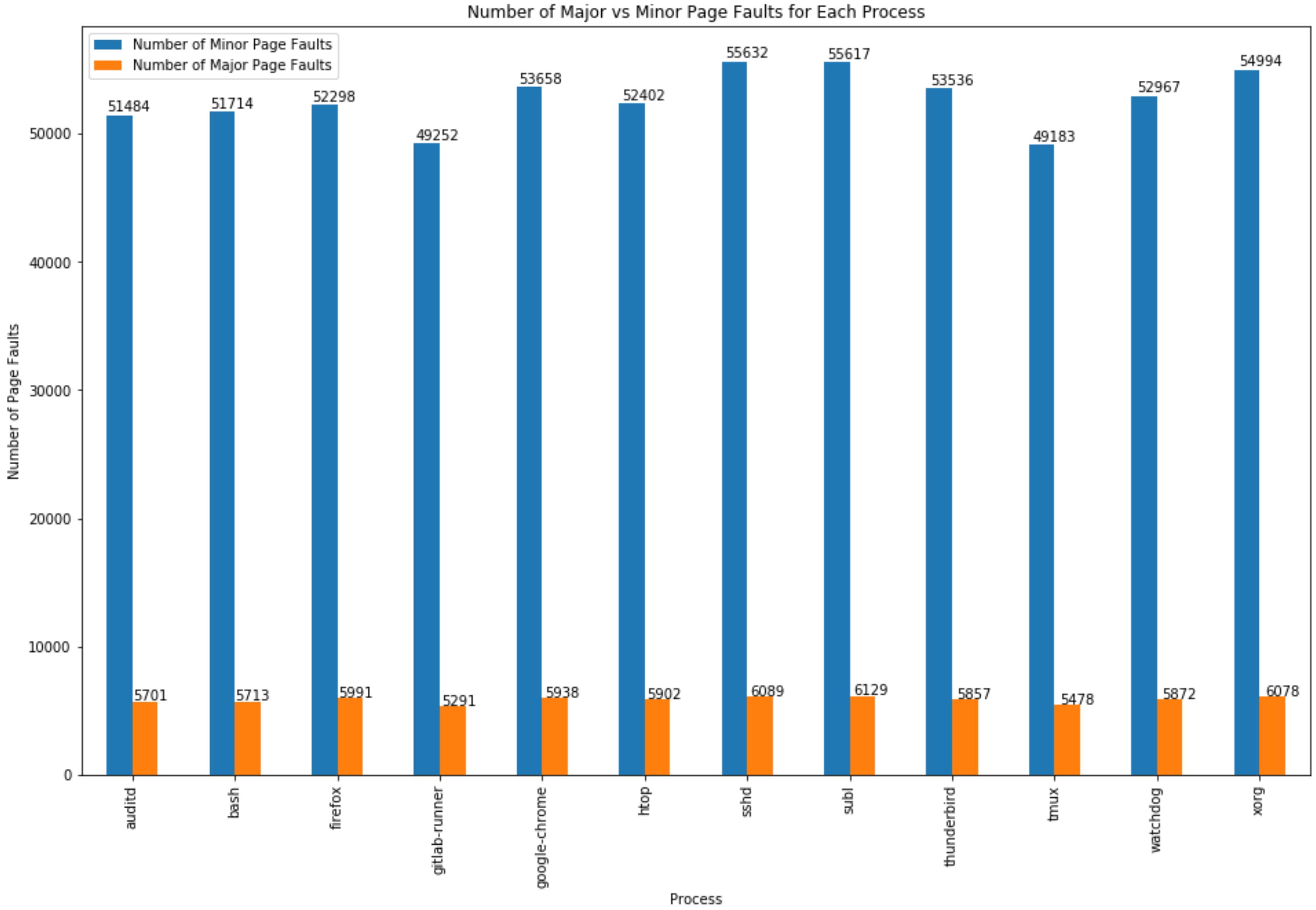
- Include
  - The number of unique processes: 12
  - The name of each process: ‘auditd', 'bash', 'firefox', 'gitlab-runner', 'google-chrome', 'htop', 'sshd', 'subl', 'thunderbird', 'tmux', 'watchdog', 'xorg’
  - The number of times each process was executed

Refer to the table on the right. The preferred answers lie in the ”Solution 1” column, though the answer in the “Solution 2” column will also be accepted. Only one of the two solution sets is allowed (i.e. students cannot submit both sets of answers).

Process Name	Solution 1 (Preferred)	Solution 2 (Also Accepted)
auditd	57185	536
bash	57427	558
firefox	58289	550
gitlab-runner	54543	536
google-chrome	59596	567
htop	58304	556
sshd	61721	573
subl	61746	572
thunderbird	59393	553
tmux	54661	525
watchdog	58839	564
xorg	61072	573

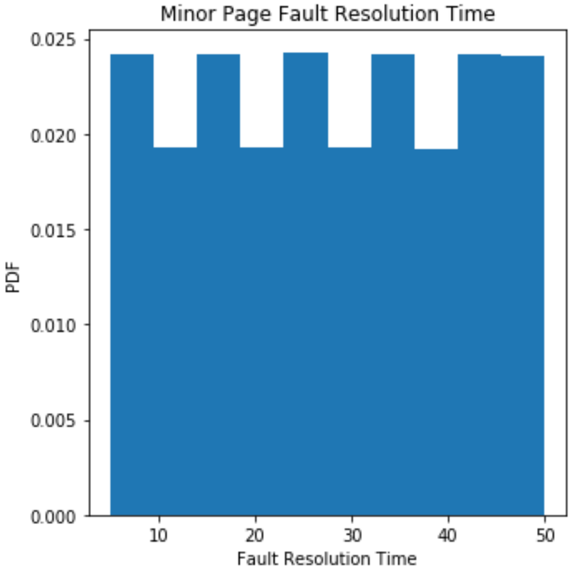
B.c. Major and Minor Page Faults

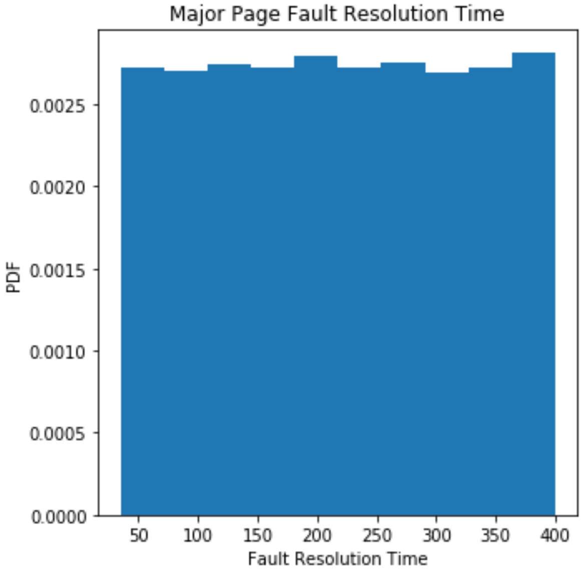
- Include a bar chart showing the number of major and minor page faults for each process
- Remember to include axes labels and a title!



B.d. Time to Resolve Page Faults

- Include
  - Histogram of time to resolve minor page faults
  - Histogram of time to resolve major page faults
  - Table with mean and standard deviations of times to resolve page faults for each process, separated by fault severity (i.e. major or minor)


Minor Page Fault Resolution Time


Major Page Fault Resolution Time

Process Name	Avg Minor Fault Res. Time	Stdev. of Minor Fault Resolution Time	Avg Major Fault Res. Time	Stdev. of Major Fault Resolution Time
auditd	27.520375	13.286257	217.744957	105.359066
bash	27.441138	13.274842	217.933310	105.217991
firefox	27.571360	13.276758	220.677850	104.520439
gitlab-runner	27.374746	13.263701	213.840484	106.110610
google-chrome	27.508144	13.230826	218.564163	104.980363
htop	27.427236	13.277471	218.407320	104.727800
sshd	27.506795	13.299934	216.405321	105.368912
subl	27.445493	13.255274	215.434655	105.697540
thunderbird	27.487018	13.252948	220.438450	107.001650
tmux	27.447289	13.276518	218.882986	105.851619
watchdog	27.599203	13.269930	217.280824	105.421466
xorg	27.534931	13.278453	217.523527	105.730484

C.a. Class Priors

- List the priors for all the classes

Process/Class	Prior
auditd	0.08137016631188317
bash	0.08171451500904983
firefox	0.08294107937664348
gitlab-runner	0.07761078921306362
google-chrome	0.08480084692704361
htop	0.0829624233041538
sshd	0.08782456999100709
subl	0.08786014320352431
thunderbird	0.08451199244140381
tmux	0.0777786947761449
watchdog	0.08372369005202226
xorg	0.08690108939406013

C.b. – C.c. : Predictions

- Given that the page fault was major, which process was it most likely caused by? subl

- Given that the page fault was from a read access, which process was it most likely caused by? subl

C.d. Appropriate Model

- In 2 sentences or less, explain which model taught in class could be used for classifying the process given information about the fault’s (i) severity and (ii) access type.

Example Answer (model must be either Naïve Bayes or Bayesian Network model):

One could use a Naive Bayes model to predict which process is most likely given the severity and access type of the page fault. The "class" node would be a node representing the process variable (which could take one of 12 values) and its two children nodes would be binary variables "se verity" (which could be major or minor) and "access type" (which could be read or write).