Team Name: Test Bayes
Team Members:
Adavya Bhalla (bhalla)
Aditya Jhamb (aditya97)
Avidant Bhagat (avidant)
Ethan Mayer (emayer4)
Steven Austin (saaustin)

Test Bayes User Manual

Reason For Use: Test Bayes is a tool designed to speed up the testing process of development significantly. Testing is a static process, that can cause developers to waste a lot of time waiting for certain tests to fail. However, with Test Bayes, tests that are likely to fail on the next run are run with higher priority, meaning they will fail first, instead of later, saving the developer a lot of time testing. This speeds up the development process significantly. Test Bayes is built off of JUnit4, a very common Java test framework, making it compatible with many projects. Test Bayes uses a Bayes oriented mathematical model to determine probabilities of test failures, and orders them accordingly.

Use: Test Bayes utilizes the maven build system to get its information about tests. By specifying the test directories in the pom.xml file, Test Bayes can find all of the tests that are to be run. By including our project in the same directory as the pom.xml, Test Bayes can function properly. Since this project is a build-off of JUnit4, JUnit4 must also be installed in the project. Test Bayes is interfaced through a terminal. There are several commands that can be used to run tests, as follows:

```
tb                              : Runs all tests in pom file.
   -h                           : Display command help info.
   -d testDir1 testDir2 …       : Runs all tests in given
                                  directories instead of all
                                  tests.
   -u                           : Will not reorder tests to be
run.
   --avg=numTestRuns            : Set the number of test runs
                                  to consider in the calculations.
   --eps=doubleValue            : Set the failure probability
                                  difference between tests.
   -c                           : Clear all test data.
   -n                           : Treat new tests as if they are
                                  not similar in fail rate to
```

other	test runs of other methods

After running Test Bayes, test results will be output to the console (terminal) in the order they were run, or if the command was a parameter modification, no output will be produced. Test Bayes test runs are versioned, in that the output from test runs will be stored in files in the directory the tool is in, making the reordering exist across different machines running the code.

Customizable inputs-

You will be able to customize the following things according to need:

1. **New tests** - You can decide whether tests that have no previous runs will be run first, or just treated like tests with the most average stats out of all. This means that a new test will not be given a 50/50 chance of passing or failing, but instead will be treated as if it were similar to other tests, and will be given the average failure rate of all other tests about to be run. This would be specified with the `-n` flag in the command entered.
2. **Moving average -** You can decide what number of runs you want to include in the calculation of the moving average. While this is not the only parameter our code uses to compute the order of tests, it is certainly a big part of it, and we want to be able to adapt to your needs as much as possible. This can be set in the pom.xml file, through the `--avg` flag command, as described above. The default value for this moving average value is 20 test runs.
3. **Test Similarity -** You can finetune an epsilon parameter, which is a decimal value from 0.0 to 1.0, and determines how similar two tests' failure rates can be so as to order by time instead of failure rate. This can be set in the pom.xml file, through the `--eps` flag command, as described above. For example, if epsilon was set to 0.01, and two tests had failure rates of 0.94 and 0.95, but the second test was double as long, the first (shorter) test would be run. The default value for epsilon is 0.03.

Building our Project:

Building our project is simple. All dependencies of our project are built into the maven configuration, and will be imported on build, or are existent in the repository already. To run a build and run the tests, enter:

git clone https://github.com/avidant/test-bayes
cd test-bayes

```
mvn clean verify
```

This will output the success or failure of the build, and success or failure of the tests we've written for our classes.

Further reading -

Much of this work was based off of tutorials given by links 1-5. If the user wants to further understand the workings behind what we are doing a few good places to look at would be:

(1)https://junit.org/junit4/javadoc/4.12/org/junit/experimental/max/MaxCore.html
(2)http://automation-webdriver-testng.blogspot.com/2012/07/how-to-call-test-plans-dynamically.html
(3)http://beust.com/weblog/2008/03/29/test-method-priorities-in-testng/
(4)https://arxiv.org/pdf/1801.05917.pdf
(5)https://lib.dr.iastate.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=6493&context=etd
(6)https://dl.acm.org/citation.cfm?id=1116157
(7)http://spring.io/