Iteration 2 Progress Report

Ray Smith, Josh Duchene, Francesca Tyler, Matthew Duff,
Nathan Lenar, Daniel Owens, Nash Wellington

We ran into a small issue with github where 1) the .gitignore file was not ignoring .class files, leading to difficulties with merging which lead to 2) a major merge malfunction with our master branch which caused a massive headache. We resolved the .gitignore issue by removing cached .class files that had been stored in github which were being pulled every time we tried to interact with the repository.

We did not completely finish implementing the writing to .pdf files as writing to .pdf's ended up being much more challenging than we thought, and formatting them is a lot of work. We succeeded in generating a .pdf file and writing basic text and tables to it (seen below), but have not gotten into graphs yet, or had enough data processed to fill out a full metrics report. This is being pushed to Iteration 3.

2019-04-12 22:49

Case Number	Owner	Requestor	Date Requested	Description	Tokenized Description	Category

We also did not fully finish implementing the GUI, as some of it couldn't be done until we finished File_IO work for storing our processed datasets as Categorized objects, which got done later in the iteration than planned. It will be finished in Iteration 3 as well.

Everything else we planned to finish this iteration was completed on schedule.

We used JUnit to implement our unit testing and integration testing in this iteration of testing as planned. The tests we wrote this iteration focused on more thorough code coverage and adding test coverage to functions which previously did not have the backend support to test. We mostly stuck to the plan for testing although we did notice a lack of code coverage that we improved upon.

The neural network section of our code in the previous iteration lacked any testing functionality due to the nature of its outputs. In this iteration, we built upon the

functionality of “Neural” to the point where we are getting confidence ratings of individual tests after training the AI, so we are able to use asserts with confidence intervals to determine whether the AI’s categorizations are well done. This is the most thorough section of integration testing we have now.

Due to the nature of UI elements we have minimal automated testing on that section of the code. We are able to perform manual tests of the frontend parts of the code and may be able to add integration testing to some of the backend functionality of our UI, but for this iteration the priority is just building an interface that we can “interact with” to show how the program will function for our client.

Code Coverage Before:

Element	Coverage	Covered Instructi	Missed Instructio
Data-Analyzer	32.1 %	950	2,005
src/main/java	22.7 %	567	1,934
(default package)	0.0 %	0	901
Neural_Network	0.0 %	0	765
Objects	61.4 %	240	151
Case.java	62.5 %	212	127
Categorized.java	53.8 %	28	24
File_IO	73.6 %	327	117
Categorized_In.java	25.3 %	19	56
Categorized_Out.java	37.1 %	23	39
FileAccess.java	76.6 %	36	11
CSV_Out.java	92.5 %	98	8
CSV_In.java	97.9 %	138	3
Tokenization.java	100.0 %	13	0
src/test/java	84.4 %	383	71

Code Coverage After

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Data	85.3 %	2,588	447	3,035
src/main/java	84.1 %	1,889	357	2,246
Objects	74.1 %	343	120	463
Categorized.java	43.0 %	49	65	114
Case.java	84.2 %	294	55	349
File_IO	82.4 %	407	87	494
Categorized_In.java	44.2 %	46	58	104
Categorized_Out.java	85.1 %	57	10	67
CSV_Out.java	92.5 %	98	8	106
FileAccess.java	83.0 %	39	8	47
CSV_In.java	98.1 %	154	3	157
Tokenization.java	100.0 %	13	0	13
(default package)	0.0 %	0	76	76
Neural_Network	93.9 %	1,139	74	1,213
Neural.java	72.2 %	130	50	180
Run_Neural.java	92.1 %	152	13	165
k_Fold_Evaluation.java	98.9 %	267	3	270
Split_Data.java	96.7 %	89	3	92
Train_Neural.java	99.3 %	428	3	431
Cell.java	97.3 %	73	2	75
src/test/java	88.6 %	699	90	789
Objects	73.7 %	140	50	190
File_IO	90.9 %	240	24	264
Neural_Network	95.2 %	319	16	335

The code coverage data was honestly eye opening to us. We thought we were being very thorough with our testing until we were asked to output our code coverage. This is when we realized that there were some branches that were not even being tested by our test functions. We took advantage of the highlighted code to find gaps in our testing to cover some of the gaps in our testing implementations. We also were able to add integration testing to some sections which further helped cover our code coverage shortcomings.

See the README file or Instructions .pdf for how to run the program.