GeneSearcher Testing

Module: Model

I wrote unit tests for the Model module in `test_Genesearcher_model.py` that test functions of the module for correct output. I created 3 test classes using the python unittest library: `GenesearcherModelTestSettersGetters`, `GenesearcherModelTestProcessing`, and `GenesearcherModelTestReportOutput`.

`GenesearcherModelTestSettersGetters` tests the setter and getter methods which are trivial. Because these methods simply take in a value and assign it a member variable or return the value of a memeber variable, without any additional processing their is only one equivalency class the setters and getters. I tested them by setting up a model object, calling the setter with some test data, and then calling the coresponding getter and verifying that the returned value matches the object that was passed in to the setter.

`GenesearcherModelTestProcessing` tests the processing functions which are meant to generate a report. The following tests are provided

- `test_load_user_data_no_header`: tests that the model is able to read a users data file that is not formatted with a header.
- `test_load_user_data_with_header`: tests that the model is able to read a users data file that is formatted with a header
- `test_test_load_data_set_from_file`: tests that the model is able to load a data set from a file
- `test_load_data_set_from_server`: tests that the `load_data_set_from_server` function sets the data_set member of the object
- `test_generate_report`: given some user data, and a loaded data set, tests that the `generate_report` method produces the expected output.

`GenesearcherModelTestReportOutput`: Tests the output of the report generation functionality.

- `test_report_to_json`: Loads a report and and then writes it to a json file and checks that the output matches our expected output.
- 'test_report_to_csv': Loads a report and and then writes it to a csv file and checks that the output matches our expected output.

Manual Applicaton Testing

I tested the full applications funtionality through manual usage testing to verify functionality.

Test Scenarios:

Use the application under normal context to very basic functionality is in place:

User should be able to start the applicaton, select a data file, initiate data processing and then view their report in the text field. Additionally the user should be able to use the search bar to find key words of interest wich will be highlited in yellow and the application will snap to the next occurence of that word. Lastly the user should be able to export the report to a csv formatted file.

1. Start the GeneSearcher application:
    1. The application window should open
    2. The file selection field is not yet populated.
2. Click Upload Data to select a file.
    1. A file selection dialog appears and the user should selct a file from their system.
3. Click Begin Search
    1. The application should process the users information and print the results in the text field.
    2. The text field should be scrollable
4. Enter some word to find in the search bar then press find or hit enter.
    1. The application should move the text field to the first instance of that word and highlight it in yellow
    2. If clicking find or pressing enter again, the application should move to the next instance of that word
5. Click export
    1. The application will save a file titled report.csv to the directory from which it was run.
        1. The report should be a properly formated csv file.

All functions seem to work as expected under this scenario.

Using the application without internet connection:

I tested the application with out internet connection to see how it would handle being unable to download the dataset. The methodology was simply to disable my network interface and perform the test noted above.

Under this scenario an exception is thrown when attempting to load the dataset and the application fails to load.

Using the application with an outdated (improperly formatted) data set:

For this test scenario, i configured the firebase instance to deliver an old version of the dataset wich did not include all of the values that the program expects. The motivation for this test was to determine, to what degree some sort of dataset validation should be implemented. While unlikely that a properly up to date application and server pair would find itself in this scenario, it is possible that future improvements could include a feature such as caching of the dataset for offline operation. If the program was updated but the cached datafile was not, and then the user attempted to operate

the software in offline mode, it is possible that an improperly formatted data set would be loaded. The methodology was as follows. I configured firebase to deliver an outdated version of the dataset. I then attempted to load the application and follow the testing methodology listed in the first scenario.

In testing the application, opens and loads data without error. However when the user clicks to begin processing an exception is thrown as the data processing module attempts to access dictionary keys that do not exist and the program fails to generate a report. Exceptions are printed to the console. but the applicaton does not notify the user of error.

Using the applicaton with an improperly formatted user data file:

This test was intended to dertermine how the application would perform if the user chose to input an improperly formatted file. This is a highly likely scenario and it seems important to ensure that the application handles this scenario gracefully. The testing methodolgy was to perform the steps listed in the first scenario but, at step 2.1, select a file that is not a properly formated file provided by 23AndMe.

Under this scenario, the application attempts to work with the file as it would in normal operation. The program produces no errors or exceptions and the user is not notifed that anything might be wrong. The program simply appears to do nothing. As the data is primarily comprised of plain text, and the processing is performed mostly through text matching, this situation would be difficult to detect. From the programs perspective, it has data and everything is working correctly, but the user data fails to match any of the entries in the dataset so no information is returned in the report.