Testing AI Program

# Introduction

In the following document, I will be going through the “Run 2” requirements, marking which elements have been implemented and providing notes on any problems that can be seen from just the implementation. After that, I will then go on to note issues that are present during the execution of code, such as values being returned that are not expected. I will also be providing the code that was used for these test cases.

The requirements for “Run 2” are as follow:

* R1 – Data Loading Error Handle
* R2 – Choose x and y columns (by int)
* R3 – Grid Search
* R4 – Single predictor
* R6 – Choose x and y columns (by string)
* R10 – Remove blanks

# Requirement Implementation

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Complete** | **Notes** |
| Data loading (R1) | Fully | Loading data from a file is present, and there is error management if the data file does not exist. |
| Choose column by index (R2) | Full | Using an integer to load data is present, the method returns an error code if the data is not present or returns the column if it is present.  This method checks if the column exists, and is invokable without using the load method as it did in the initial run. |
| Grid search (R3) | Partial | Grid search now implements try catch and shows an easy to understand message informing the user that the data is being worked with and that the program is still running. |
| Prediction of value (R4) | None | This method still is awaiting implementation |
| Choose column by string (R6) | Partial | Using a string to load data is also present, the method returns an error code if the data is not present or returns the column if it is present.  This method checks if the column exists, and is invokable without using the load method as it did in the initial run. |
| Remove blanks (R10) | Partial | There were no changes in this method to meet run 1 recommendations.  Recommendation is still to add column checking as to make sure the columns being used exist |

# Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Expected Result** | **Actual Result** | **Testing Method** | **Reason** |
| Data Loading  **Conditions:**   * File does exist | The method goes on to request the x and y columns | The method performs as expected | Data\_Loading\_Pass | Users in the real world make mistakes and accessing data that does not exist could cause the program to crash. Performing a check will ensure the data access is available.  Ensuring the method works whether a file exists, does not exist or the user makes repeated mistakes is vital as in the real world any of those situations could arise |
| Data Loading  **Conditions:**   * File does not exist | The method informs the user that there was an error and allows them to enter their file name again (for testing, this returns -1) | The method performs as expected (automated testing ignores second round and returns -1 showing the error was caught) | Data\_Loading\_Fail |
| Data Loading  **Conditions:**   * Repeated failure to enter a correct file name | The method should continue to loop, informing the user that there was an error and asking them to enter the correct name | The method performs as expected (actual input requests hidden for debugging) | Data\_Loading\_Fail |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Expected Result** | **Actual Result** | **Testing Method** | **Reason** |
| Choose column by Index, Choose column by String  **Conditions:**   * Data Exists * Column does not exist | Error informing the user that the column entered does not exist. Halts progress of program until a correct column is entered | The program catches the error and returns an empty array. Error handle implementation to be provided by the end user | Column\_Load\_Fail | When a user enters a column name, or number, they could easily hit a key by mistake or just spell the name incorrectly, or even type in a number for a column which does not exist. Ensuring that columns are only retrieved if they exist ensures that the rest of the program can function with the proper data it needs. Human error should be minimised by checking as soon as the data is provided whether it is correct. |
| Choose column by Index, Choose column by String  **Conditions:**   * Data does not exist | The method informs the user that there was an error and allows them to enter their file name again. | The method performs as expected (automated testing ignores second round and returns -1 showing the error was caught) | Data\_Loading\_Fail |
| Choose column by Index, Choose column by string  **Conditions:**   * Data exists * Column does exist | No errors, performing a “strip” successfully strips data from those columns that were empty (no film categories) | The method performs as expected | Column\_Load\_Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Expected Result** | **Actual Result** | **Testing Method** | **Reason** |
| Grid Search  **Conditions:**   * Text Pipeline exists * Grid parameters correctly formatted | The user will receive a notification informing them the grid search has completed and shown the accuracy | The user is shown the accuracy, however, the message shown for a success is not very user-friendly (Actual accuracy and prediction hidden in testing) | GridSearch\_Pass | Grid Search is a vital part of the program and, while we are keeping it locked behind an exposed API as to avoid issues, there is a chance that the user may accidentally modify the code. Informing the user that there is an error allows them to either fix it themselves or allow helplines or forums to help them more easily. |
| Grid Search  **Conditions:**   * Text pipeline not setup correctly * Grid parameters correctly formatted | The user will receive an error informing them there is an issue with their text pipeline/grid search parameters | The error is caught by the method and performs as expected | GridSearch\_Fail |
| Grid Search  **Conditions:**   * Text pipeline setup correctly * Grid parameters not correctly formatted | The user will receive an error informing them there is an issue with their text pipeline/grid search parameters | The error is caught by the method and performs as expected | GridSearch\_Fail |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Expected Result** | **Actual Result** | **Testing Method** | **Reason** |
| Prediction  **Conditions:**   * The data provided for the prediction is clear and easy to be identified | The user will be show what the category of the film they entered is that is accurate | The user is shown the raw array data, including information they would not need. This does, however, include the category.  The answer given is correct. (Actual answer hidden by test) | GridSearch\_Pass | Users using this system may not be proficient with IT systems and may need things to be more clearly communicated. They also may push the system beyond the data the data they provided or may even not think they have to re-analyse new data if they would like to use the system to work with different data. |
| Prediction  **Conditions:**   * The data is more difficult to guess, but within scope of the application | The user will be shown an answer that is at least somewhat accurate to the actual answer | The user is shown the raw array data, including information they would not need. This does, however, include the category.  The answer given is correct (Actual answer hidden by test) | Predict\_Test |
| Prediction  **Conditions:**   * The data provided is not within scope of the application | The user will be given an answer, and the program will not crash. | The program continues running and an inaccurate answer is given. | Predict\_Fail |