Testing AI Program

# Introduction

In the following document, I will be going through the “Run 1” 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 1” 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)

# Requirement Implementation

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Complete** | **Notes** |
| Baseline implementation | Fully | The baseline for the program has been implemented and performs as expected. There is a notable slowdown when performing the model fitting which may be a note of concern, however, this is a onetime process. |
| 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) | Partial | Using an integer to load data is present, however, there is no error checking for this and.  I also recommend that this should be moved to a separate method rather than be included with the initial data loading. |
| Grid search (R3) | Fully | This method has fully been implemented and has increased accuracy from 47% to 67%. |
| Prediction of value (R4) | Partial | This has been implemented, however, expects the user to provide an array, providing a raw text input causes an error. The prediction also does not seem to output.  Recommendation is to add a method which takes a string input, and automatically forms an array from that value. Also recommend adding a “print” on the prediction |
| Choose column by string (R6) | Partial | Using a string to load data is also present, however, also does not check for the existence of the column.  I also recommend moving this into a separate method rather than being done at the start when the data is loaded. |

# 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. | The method performs as expected | 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 | Data\_Loading\_Fail |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Expected Result** | **Actual Result** | **Testing Method** | **Reason** |
| Choose column by Index  **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 continues operating, performing a “strip” results in an error. | 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  **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 | Data\_Loading\_Fail |
| 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 |
| 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 continues operating, performing a “strip” results in an error. | Column\_Load\_Fail |
| 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 | Data\_Loading\_Fail |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 | 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 method performs as expected, however, the message is not very user-friendly | 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 method performs as expected, however, the message is not very user-friendly | 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. | Predict\_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. | 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 |