**Decision Tree Classifier**

The dataset used for this classifier included a table that contained the explanatory variables - version/book-count, genre\_id, number-of-ratings, rating and the class variable 1/0. Another worksheet genre\_id (Appendix 1 – Worksheets) held the genre names that were used and the corresponding id’s. The class variables were changed to 0 or 1, representing no or yes as the algorithm worked with numeric data.

The goal was to predict if a book would be made into a movie given the above explanatory data.

In Python, the algorithm used was sourced from <https://stackabuse.com/decision-trees-in-python-with-scikit-learn/>.

**Algorithm**

The dataset was read in and stored as ‘dataset’, the data was then prepared by dividing it into attributes and labels and further dividing the resulting data into training and test data sets. This allows the algorithm to be trained on one set of data and tested on a different set of data. This was done using the sklearn library that contained a function ‘train\_test\_split’. This function allows for the specification of the percentage of the dataset to use as the test data. For this analysis this value would be varied and ran multiple times to cover a larger set of testing data, as the ‘train\_test\_split’ method randomly splits the data.

The next step was to train the decision tree algorithm to make predictions. The Scikit-Learn library contains a tree library that contains methods for various decision tree library. For the classification task, the ‘DecisonTreeClassifier’ class was used as well as the ‘fit’ method that was used to train the data and the predict method to make predictions on the test data (Appendix 2-Decision Tree Classifier). (Appendix 2-Decision Tree Classifier). Finally, the algorithm was evaluated using the metrics listed below.

**Confusion Matrix:** also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.  Each row of the [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) represents the instances in a predicted class while each column represents the instances in an actual class

**Precision**, p: the number of correct positive results divided by the number of all positive results returned by the classifier).

**Recall,** r : the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive)).

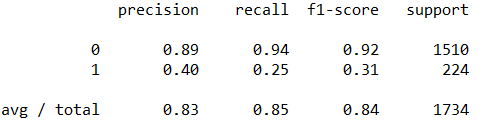
**F1-score** - a measure of a test's accuracy. It considers both the [precision](https://en.wikipedia.org/wiki/Precision_(information_retrieval)) *p* and the [recall](https://en.wikipedia.org/wiki/Recall_(information_retrieval)) *r* of the test to compute the score.

**Support:** The subset of the measurable space, number of rows for each class variable

**1st run with 20% of data as testing data(randomized)**

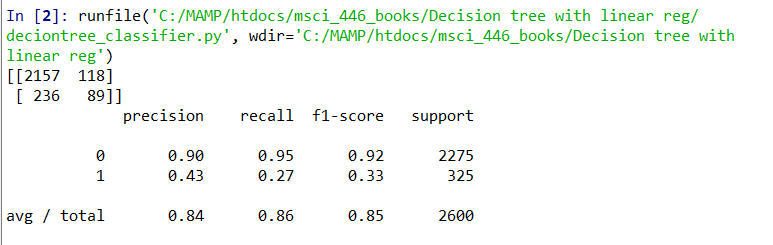


*The algorithm got 254 out of 1734 rows of data wrong, as seen from confusion matrix, which gives an 85.35% accuracy.*



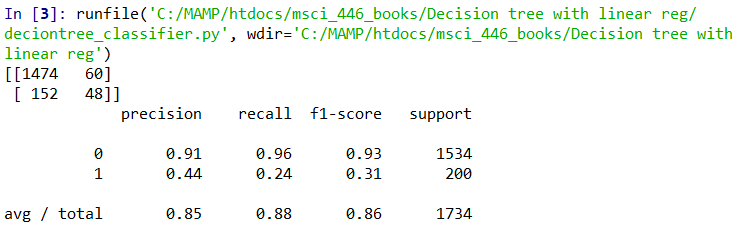
**Run 2 with test-size of 30%**

*The algorithm got 354 out of 2600 wrong, as seen from confusion matrix, this gives it an accuracy of 86.385% accuracy.*



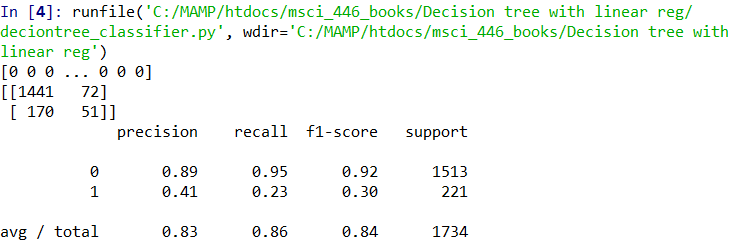
**Run 3 with 20% of data as testing data**

*The algorithm got 212 out of 1734 test data wrong, as seen from confusion matrix, giving it an accuracy of the 87.774%.*



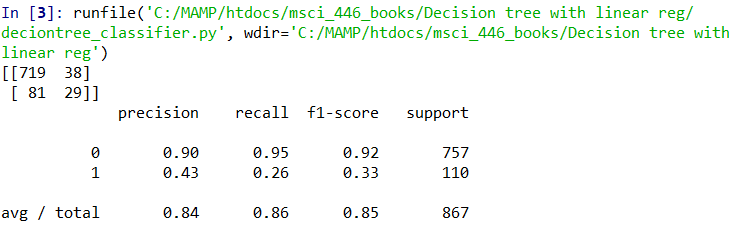
**Run 4 with 20% of data as testing data**

*The algorithm got 242 out of 1734 test data wrong, as seen from confusion matrix, giving it an accuracy of the 86.04%.*

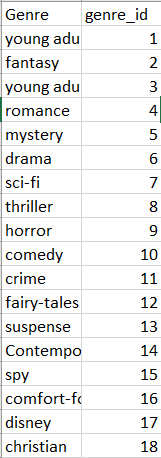


**Run 5 with 10% of data as testing data**

*The algorithm got 119 out of 867 test data wrong, as seen from confusion matrix, giving it an accuracy of the 86.275%.*



**Appendix, Data Worksheet**



*Genre Id to Genre for Decision Tree*