**CS 839 Data Science Project Stage 2 Report**

Sreejita Dutta, Deepanshu Gera, Rahul Jayan

**Data Description**

We have chosen books as the entity. We scraped this data from two web sources: walmart.com and amazon.com. The scraped table consists of the following attributes:

1. Name/Title of the book
2. Sale Price
3. Category/Genre
4. Author
5. ISBN
6. Pages
7. Publisher
8. Language
9. Dimension of the book
10. Weight of the book
11. User Rating

Though we have retained the attribute ISBN in the dataset, we have **not used ISBN** for blocking or entity matching.

amazon\_books.csv consists of **3212 tuples** and walmart\_books.csv consists of **3114 tuples**.

**Blocking to generate candidate pairs**

We have used three types of blockers for our blocking stage.

1. Black Box Blocker to handle single word ‘Name’
2. Word level Overlap Blocker on attribute ‘Author’ with overlap size 2
3. Word level Overlap Blocker on attribute ‘Name’ with overlap size 2
4. Rule based blocker with absolute norm compute on attribute ‘Pages’
5. Rule based blocker with absolute norm compute on attribute ‘Sales\_Price’

We obtained **802** tuple pairs after this blocking stage. We labelled **300** of these pairs for our learning stage.

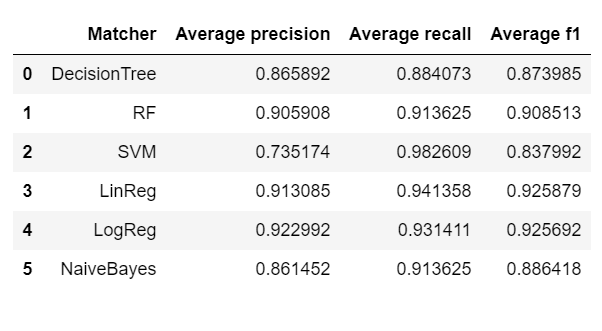
**Sampling and labeling the candidate set**

We labelled **300** of the blocked pairs for our learning stage.

**Matching**

1. **Initial Accuracy Metrics for development set**

Iteration 1: The results obtained on performing cross validation for the first time on set I is compiled in the table given below.

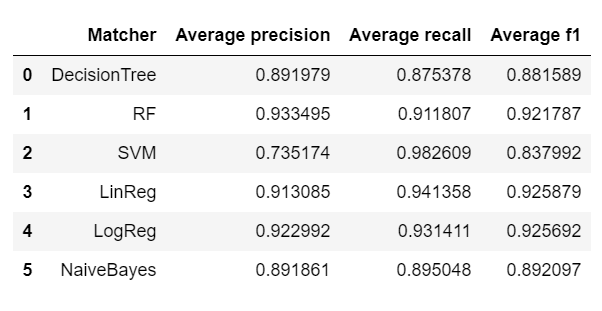


We selected Random Forest as the matcher since it was giving optimal precision and recall. Also, since Magellan provides debugger support for only Decision Tree and Random Forest matchers, we suspected that after debugging RF could outperform Linear Regression and Logistic regression matchers.

1. **Debugging the classifier**

Iteration 2: We decided to debug RF matcher for this iteration. On debugging we realized that a lot of false positives and false negatives were due to words indicating the type of print in the ‘Name’ of the book, for e.g. Paperback, Hardcover etc. Current features do not account for this and adding a feature incorporating this difference in format can potentially improve the accuracy metrics. So, we added a blackbox feature to address this issue.

After this step, we obtained the following accuracy on the matchers.

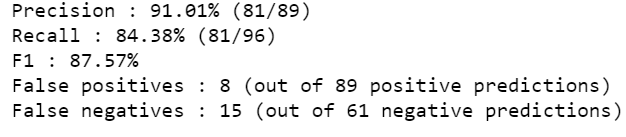


From these results we selected Random Forests matcher as the best one as it was giving the highest precision and optimal recall.

1. **Accuracy Metrics for evaluation set**

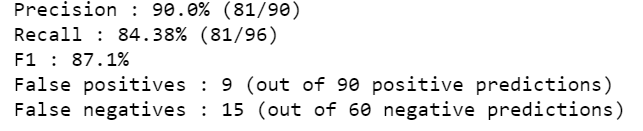
We selected Random Forests as the best matcher and we ran it on test dataset J. The results obtained on test dataset J is given below:

Random Forests

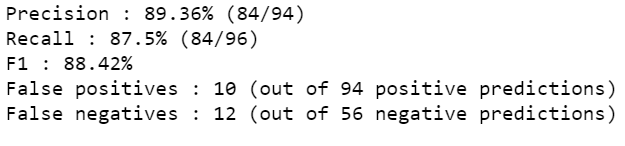


Result on test dataset for other 5 classifiers are given below:

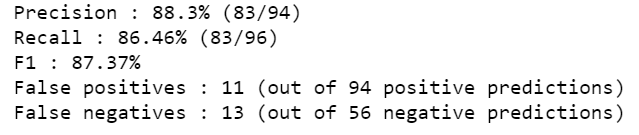
Decision Tree



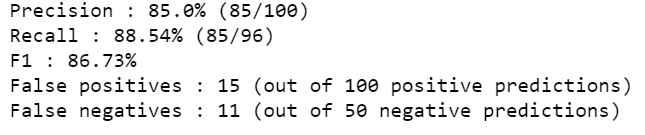
Logistic Regression



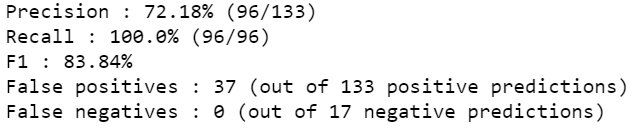
Linear Regression



Naïve Bayes



SVM



**Random Forests** gave the highest precision and recall on the evaluation set J

**Time Estimates**

1. Blocking: 4 hours
2. Labelling: 2 hours
3. Finding best matcher: 10 hours

**Discussion**

Matching books from the two web sources (amazon.com and walmart.com) was not straightforward. After looking at the dataset, we realized that books which had different names or authors or even ISBNs could represent the same entity. This makes it very hard for a classifier to distinguish a match from a mismatch for candidate pairs that look similar. For example, consider the candidate pairs:

* (“Sword Art Online, Vol. 2”, “Sword Art Online 10”)
* (“The Charm School Paperback”, “The Charm School”)

The first tuple pair is a mismatch, but the second tuple pair is a match. Since it is hard for the classifier to account for these minute differences, the second tuples pair is a false negative. Such cases lower the recall.

**Magellan points**

**Positives**

1. Good and clear documentation
2. Debugger for decision tree and random forests matcher is helpful

**Potential improvements**

1. Automatic feature extraction doesn’t seem to work for string attributes that vary in lengths
2. Availability of matchers for all learning-based algorithms would be helpful
3. GUI for labelling data could have option of hiding columns and persisting previously labelled columns