Project Stage 3

1, Describe the type of entity you want to match, briefly describe the two tables (e.g., where did you obtain these tables), list the number of tuples per table.
We chose to match **'Books'** entities from two different web sources - Goodreads (goodreads.com) and book depository (bookdepository.com).

Information about the tables

CSV Filename	Web Source	Number of tuples
bookdepository.csv	https://www.bookdepository.com	3968
goodreads.csv	https://goodreads.com	3794

The two tables had scheme shown below.

Attribute Name	Datatype	Description
Title	String	Title of the book (including the Edition number) as given in the source from which the book details are taken.
Authors	String	Names of all the persons listed under the author's section along with the authors. This includes the translator, the illustrator etc.
Genres	String	The category of books under which the book can be classified such as Horror, Comedy etc. A single book can be classified under multiple genres.
Publishing Date	String	Date of publishing.
Pages	Integer	Number of pages in the book.
Publisher	String	Name of the publisher(s) as listed in the source.
Language	String	Language.

2. Describe the blocker that you use and list the number of tuple pairs in the candidate set obtained after the blocking step.

Overlap blocker

We cascaded two overlap blockers to eliminate obvious non-matching tuple pairs.

- The first overlap blocker blocks tuple pairs performing 3-gram on '**Authors**' attribute with minimum of 5 shared tokens (overlap size).
- The second overlap blocker blocks tuple pairs performing 3-gram on '**Title**' attribute with minimum of 6 shared tokens (overlap size). We also ignored stop words from being considered as tokens.

We tried other blockers (Attribute Equivalence blocker) but preferred to use the Overlap blocker since AE blocker performed exact match, removed many matches due to minor differences in the attribute value.

3, List the number of tuple pairs in the sample G that you have labeled.
We sampled out **500 tuple pairs** from the candidate set and labeled them.

4, For each of the six learning methods provided in Magellan (Decision Tree, Random Forest, SVM, Naive Bayes, Logistic Regression, Linear Regression), report the precision, recall, and F-1 that you obtain when you perform cross validation for the first time for these methods on I.

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Decision Tree	0.919206	0.904099	0.906615
Random Forest	0.982213	0.896455	0.933770
SVM	0.942997	0.569838	0.695597
Linear Regression	0.947432	0.866567	0.892675
Logistic Regression	0.933237	0.981818	0.956224

5, Report which learning based matcher you selected after that cross validation.
From above metrics (F1 score), we chose the best matcher to be Logistic Regression.

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Logistic Regression	0.933237	0.981818	0.956224

6, Report all debugging iterations and cross validation iterations that you performed. For each debugging iteration, report (a) what is the matcher that you are trying to debug, and its precision/recall/F-1, (b) what kind of problems you found, and what you did to fix them, (c) the final precision/recall/F-1 that you reached. For each cross validation iteration, report (a) what matchers were you trying to evaluate using the cross validation, and (b) precision/recall/F-1 of those.

We performed only **one iteration** of cross-validation, since the matcher met the expectations, we did not perform any more steps.

7, Report the final best matcher that you selected, and its precision/recall/F-1.

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Logistic Regression	0.933237	0.981818	0.956224

8, For each of the six learning methods, train the matcher based on that method on I, then **report its precision/recall/F-1 on J.**

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Decision Tree	0.8	0.9263	0.8585
Random Forest	0.9111	0.8632	0.8865
SVM	0.9444	0.5365	0.6846
Linear Regression	0.9286	0.8211	0.8715
Logistic Regression	0.9020	0.9684	0.9340

9, For the final best matcher Y selected, train it on I, then **report its precision/recall/F-1 on J.**

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Logistic Regression	0.9020	0.9684	0.9340

10, Report approximate time estimates: (a) to do the blocking, (b) to label the data, (c) to find the best matcher.
Run time for each step
Blocking: 5 seconds
Labeling: 45 minutes
Finding the best matcher: 4 seconds

11, Provide a discussion on why you didn't reach higher recall, and what you can do in the future to obtain higher recall.
We didn't face any problem in getting high recall.

BONUS POINTS: provide comments on what is good with Magellan and what is bad, that is, as users, what else would you like to see in Magellan. Are there any features/capabilities that you would really like to see being added? Any bugs? Depending on how detailed and helpful these comments are, you can get bonus point from 1-10 (which will help with the final grade, not just with the project).

- The entire software can be containerized, meaning all dependencies and environment can be bundled as a single software so that user need not spend time in installing. Using Docker would be good way to go. (We spend more than 3 hrs in installing and configuring the software).
- Standard GCC in OS-X Sierra doesn't have all the dependency required. Hence "conda install -c uwmagellan py_entitymatching" doesn't work. Using Docker can eliminate this problem.