Stage 3 Report

I. Data Description

Entity: Music Albums

For this project we decided to gather information on music albums. We gathered this information from the following two web sources:

1. Metacritic.com - 3000 tuples

Metacritic aggregates reviews of music from leading critics. They calculate a metascore, which shows how well an item has scored from critics. They have a sizeable collection of music albums on their website that we decided to extract.

2. Wikipedia.com - 3189 tuples

For each year, Wikipedia has a page that lists music albums released during that year along with some information about each album. By extracting the information from several years of Wikipedia pages, we can gather a sizeable amount of music album data.

II. Blocker

The blocker we use is a combination of several attribute equivalence blockers. The blocker is the union of the following blockers:

1. Attribute equivalence on album title
2. Attribute equivalence on artist
3. Attribute equivalence on release date and at least 2 word overlap on album title
4. Attribute equivalence on release date and at least 1 word overlap on artist

The result is a candidate set of 3006 candidate tuple pairs. We debugged this blocking step and did not find any positive matches that this candidate set was missing.

We labelled 500 tuple pairs in the sample G out of 3006 tuple pairs to use for learning-based matching.

III. Cross Validation on I

We get the following results when running the 6 classifiers for the first time on set I:

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Decision Tree	0.934566	0.929703	0.929673

Random Forest	0.985714	0.958421	0.970322
SVM	1	0.224399	0.364633
Linear Regression	0.972115	0.953036	0.961021
Logistic Regression	0.973214	0.968421	0.968998
Naive Bayes	0.9566	0.912267	0.93285

After this initial cross validation step, we select the random forest learning method.

Because the initial results are satisfactory, we did not have to perform any formal debugging and cross validation iterations. As such, our final best matcher is random forest, with precision =.986, recall =.958, and F1 = .970.

IV. Results on J

The following table shows our results of training each matcher on set I and testing it on set J.

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Decision Tree	0.8605	0.9024	0.881
Random Forest	0.9589	0.8537	0.9032
SVM	0.9474	0.2195	0.3564
Linear Regression	0.963	0.9512	0.9571
Logistic Regression	0.9625	0.939	0.9506
Naïve Bayes	0.961	0.9024	0.9308

The matcher we choose from the cross validation step, the random forest, achieves satisfactory results by achieving precision > .9 (it achieves .96) and a high recall (.854). The table above shows that linear regression actually behaves slightly better, but both are within the requirements desired.

V. Time Estimates
These time estimates reflect the collective time spent by the 3 of us on each task.

Blocking: 2-3 hours
Labeling Data: 1 hour
Finding Best Matcher: 1-2 hours

VI. Recall Discussion
Our recall results are reasonably high. It would be possible to achieve higher recall results by labeling more data. We could also develop some post-processing rules in order to increase recall.

VII. BONUS

Good:
1. Documentation
The documentation, in and of itself, was well-written. It was easy to follow examples written in Jupyter notebook.

2. Debug tool
The debugging tool is a great addition to the package. However, it would be nice to include some sort of visualization, say, for the case of a random forest. This may help users better understand the decisions being made at each tree on top of the information displayed in table format.

Bad:
1. Feature names
The automated features are useful for running the learning-based matchers. However, it would be nice if they had more descriptive names to understand what the features are.

2. Installation
We had some initial trouble installing Magellan on mac computers. It was due to some conflicts with previously installed versions of dependencies. It would be nice if Magellan provided the conda environment file with required packages to simplify the installation process.

3. Simplified interface for training/testing
We liked the select_matcher method to run cross validation on each of the matchers at once. We think it would be great if there were a method like that where you could input each of the matchers, the training set I, and the test set J, and get all the results back in a table.