Decision Tree

	precision	recall	f1-score	support
0	0.97	0.98	0.97	136
1	0.98	0.97	0.97	128
avg / total	0.97	0.97	0.97	264

SVM:

	precision	recall	f1-score	support
0	0.93	1.00	0.96	136
1	1.00	0.92	0.96	128
avg / total	0.96	0.96	0.96	264

Logistic Regression:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	136
1	1.00	0.93	0.96	128
avg / total	0.97	0.97	0.97	264

Random Forest:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	136
1	0.98	0.97	0.97	128
avg / total	0.97	0.97	0.97	264

Naive Bayes:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	136
1	0.96	0.97	0.97	128
avg / total	0.97	0.97	0.97	264

Initial CV Scoring Charts for 5 Different Types of Classifiers

Each classifier was first trained on the same segment of a 400 tuple sample.

This segment is referred to as our development set (dev_set.csv) which contains the associated feature vectors for this segment of the sample.

Similarly, we have segmented a portion of the 400 tuple sample to be our evaluation set for untainted testing further along in the matching process.

From these scores, we decided to move on with the **Decision Tree Matcher Model**

Sampling

After initially discovering a heavy skew of positive examples with our initial candidate set, we came to the realization that during the blocking step, due to the nature of our data, we had “over-blocked” (with too high of thresholds when blocking by title and artist) and probably didn’t require a learning based entity matching method. This “overblocking” reduced our candidate set to a set of tuple pairs with similarity scores of almost no variance. Thus, when taking a sample of these candidates for learning, our model will have little difference to learn from.

To counteract this skew, we relaxed our blocking rules for the artist attribute to a simple rule of having an equivalent primary character in the artist string. This increased the number of negative examples in our candidate set (candidates_final.csv) to a close to even split.

Note: This first sample produced a very accurate model; however, the corresponding decision tree (fig 1) was not incredibly interesting as it only used one attribute in our feature vectors. For this reason, we took an even more relaxed approach to blocking for an even larger second candidate set (candidates_final2.csv). The sample from this second candidate set produced a slightly less accurate tree (below), but with much more use of features from the feature corresponding feature vectors.

Debugging Decision Tree

As seen above, all of our CV scores surpassed our original desired accuracy 85%. After observing CV scores for all models, we raised this threshold to 95%. Because of our resulting high accuracy, we did not have to perform much debugging.

When we trained our Decision Tree matcher on our development set of data, we achieved a 97% accuracy on the first iteration. We analyzed our tree, and it was the resulting one below.

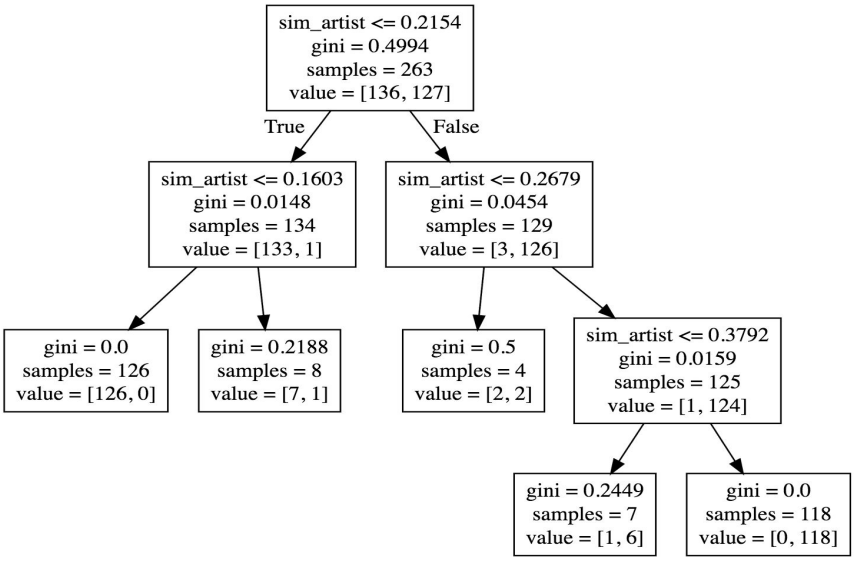

Fig. 1

Because this tree focuses only on one attribute of our feature vectors, we took action to create a more “interesting” matcher. We reblocked to create a second candidate set that was even more relaxed. To do this we lowered our song name threshold from 0.7 to 0.5, and used the same rule for artists. The resulting decision tree (fig 2) used all attributes given in the feature vectors but yielded a slightly lower F1 score (although still acceptable).

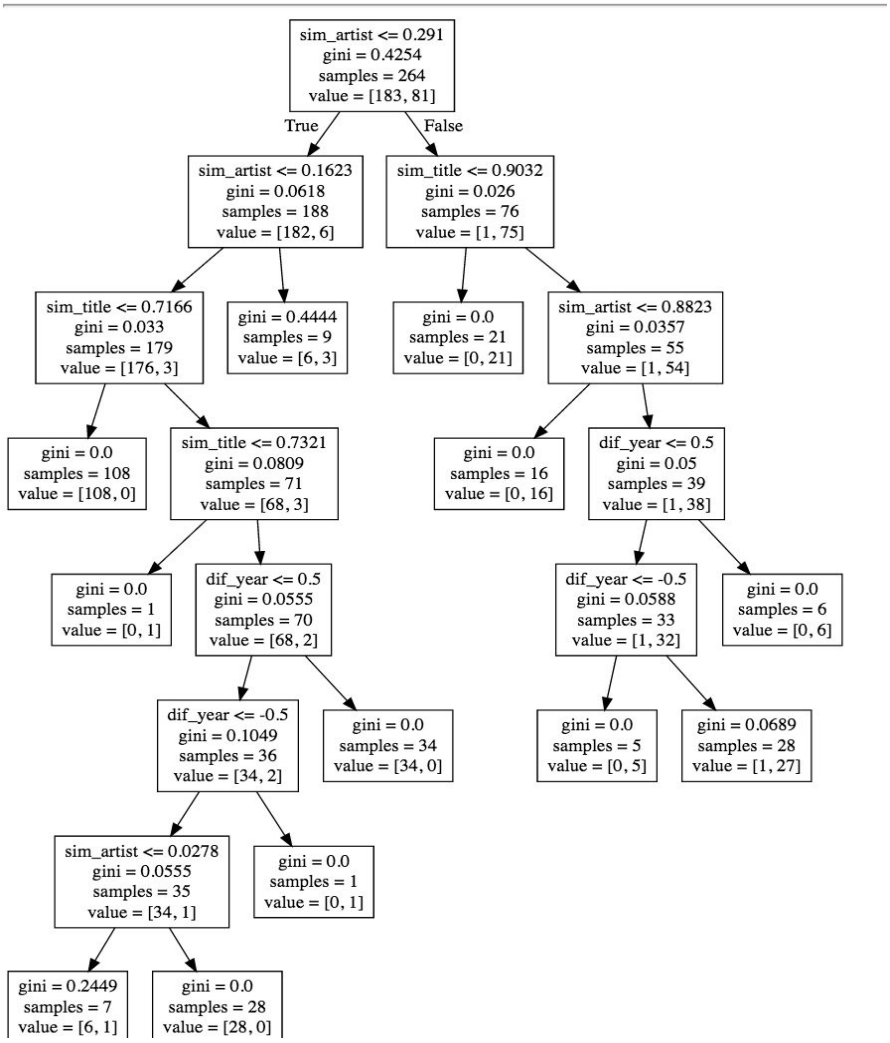

Fig. 2

Because our initial accuracy was over our desired accuracy, we ***continue with our decision tree matcher***

Evaluation

Predicting these models on the our evaluation set produces the following results:

Decision Tree:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	70
1	0.97	0.97	0.97	62
avg / total	0.97	0.97	0.97	132

SVM:

	precision	recall	f1-score	support
0	0.92	1.00	0.96	70
1	1.00	0.90	0.95	62
avg / total	0.96	0.95	0.95	132

Log Regression:

	precision	recall	f1-score	support
0	0.95	1.00	0.97	70
1	1.00	0.94	0.97	62
avg / total	0.97	0.97	0.97	132

Random Forest (avg over 10 trials) :

	precision	recall	f1-score	support
0	0.93	0.97	0.95	70
1	0.97	0.92	0.94	62
avg / total	0.95	0.95	0.95	132

Naive Bayes:

	precision	recall	f1-score	support
0	0.96	0.99	0.97	70
1	0.98	0.95	0.97	62
avg / total	0.97	0.97	0.97	132

We see that our **Naive Bayes** classifier performs best when predicting our evaluation set. Thus, we continue with our Naive Bayes classifier.

The **final set of features** we used were the similarity score of song name, similarity score of the artist name, and the difference in the years. We eliminated all of the missing values in the song name and artist name, but we did not remove all of the items that had missing years. In the left table, we had a policy set that every missing year would be set to 0. We then adopted that policy to the right right table, and set every missing year to 0.

Approximate Time Estimate

In the labelling process, we generated 400 random numbers in python (without replacement), and selected the corresponding tuples from the candidate set. We then split the resulting set of 400 tuples into 8 groups of 50, and each individually labeled each set of 50 and compared. This whole process took about 90 minutes. To find the best learning-based matcher, we spent about 30 minutes each and compared all of our output (which in theory should be the same, and in reality was the same). All of our learning-based matchers met the CV threshold of 90%, so we selected the most accurate one which was the Decision Tree.

Higher Precision Recall?

One reason that we couldn't reach a higher precision, recall, F-1 score is because of the way that we handled missing year values while generating our feature vectors. The policy we had in place was: replace any missing year value with 0. In the generation of the feature vectors, we set our year difference component to be determined by year score = year(x) - year(y), and if both years x and y were 0, set the difference to -1. Our intent was for -1 to be interpreted as an "ignore" value. However, the Decision Tree learned -1 to be a regular difference, and did not ignore the value. If we were to obtain a higher precision, recall, F-1 score, then we would have to develop a system to correctly handle the case where data is missing.

Another reason that we are not going to reach a higher precision, recall, and F-1, because we will eventually start to over fit our matcher specifically for our labeling data. This can cause problems by being too specific for a small amount of our data and failing to scale accurate to the rest of our data.