PROJECT PHASE 4 REPORT

Precision, Recall, and F-1 for each matcher when tested on I: (ITERATION 1)

**Precision:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Numb folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DecisionTree (py_entitymatching_matcher.dmatcher.DTMatcher object at 0x0000000011C104E0)</td>
<td>5</td>
<td>0.636364</td>
<td>0.387692</td>
<td>0.333333</td>
<td>0.500000</td>
<td>0.333333</td>
<td>0.422145</td>
</tr>
<tr>
<td>1</td>
<td>RF (py_entitymatching_matcher.rmMatcher RFMatcher object at 0x0000000011C10550)</td>
<td>5</td>
<td>0.750000</td>
<td>0.555556</td>
<td>0.600000</td>
<td>0.500000</td>
<td>0.333333</td>
<td>0.547778</td>
</tr>
<tr>
<td>2</td>
<td>SVM (py_entitymatching_matcher.smMatcher SVMMatcher object at 0x0000000011C104A8)</td>
<td>5</td>
<td>0.500000</td>
<td>1.000000</td>
<td>0.250000</td>
<td>0.500000</td>
<td>1.000000</td>
<td>0.650000</td>
</tr>
<tr>
<td>3</td>
<td>LogReg (py_entitymatching_matcher.logRegMatcher LogRegMatcher object at 0x0000000011C105CD)</td>
<td>5</td>
<td>0.857143</td>
<td>0.666667</td>
<td>0.200000</td>
<td>0.750000</td>
<td>0.688889</td>
<td>0.538095</td>
</tr>
<tr>
<td>4</td>
<td>NB (py_entitymatching_matcher.nBMatcher NBMatcher object at 0x0000000011C10668)</td>
<td>5</td>
<td>0.538462</td>
<td>0.636364</td>
<td>0.250000</td>
<td>0.571429</td>
<td>0.416667</td>
<td>0.482854</td>
</tr>
</tbody>
</table>

**Recall:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Numb folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DecisionTree (py_entitymatching_matcher.dmatcher.DTMatcher object at 0x0000000011C104E0)</td>
<td>5</td>
<td>0.538462</td>
<td>0.444444</td>
<td>0.75</td>
<td>0.5</td>
<td>0.285714</td>
<td>0.503724</td>
</tr>
<tr>
<td>1</td>
<td>RF (py_entitymatching_matcher.rmMatcher RFMatcher object at 0x0000000011C10550)</td>
<td>5</td>
<td>0.230769</td>
<td>0.555556</td>
<td>0.75</td>
<td>0.3</td>
<td>0.428571</td>
<td>0.452979</td>
</tr>
<tr>
<td>2</td>
<td>SVM (py_entitymatching_matcher.smMatcher SVMMatcher object at 0x0000000011C104A8)</td>
<td>5</td>
<td>0.076923</td>
<td>0.222222</td>
<td>0.25</td>
<td>0.1</td>
<td>0.285714</td>
<td>0.186972</td>
</tr>
<tr>
<td>3</td>
<td>LogReg (py_entitymatching_matcher.logRegMatcher LogRegMatcher object at 0x0000000011C105CD)</td>
<td>5</td>
<td>0.461538</td>
<td>0.444444</td>
<td>0.25</td>
<td>0.3</td>
<td>0.285714</td>
<td>0.348339</td>
</tr>
<tr>
<td>4</td>
<td>NB (py_entitymatching_matcher.nBMatcher NBMatcher object at 0x0000000011C10668)</td>
<td>5</td>
<td>0.538462</td>
<td>0.777778</td>
<td>0.25</td>
<td>0.4</td>
<td>0.714286</td>
<td>0.536105</td>
</tr>
</tbody>
</table>

**F1:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Numb folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DecisionTree (py_entitymatching_matcher.dmatcher.DTMatcher object at 0x0000000011C104E0)</td>
<td>5</td>
<td>0.583333</td>
<td>0.363636</td>
<td>0.461538</td>
<td>0.500000</td>
<td>0.307692</td>
<td>0.443740</td>
</tr>
<tr>
<td>1</td>
<td>RF (py_entitymatching_matcher.rmMatcher RFMatcher object at 0x0000000011C10550)</td>
<td>5</td>
<td>0.352941</td>
<td>0.555556</td>
<td>0.666667</td>
<td>0.375000</td>
<td>0.375000</td>
<td>0.465033</td>
</tr>
<tr>
<td>2</td>
<td>SVM (py_entitymatching_matcher.smMatcher SVMMatcher object at 0x0000000011C104A8)</td>
<td>5</td>
<td>0.133333</td>
<td>0.363636</td>
<td>0.250000</td>
<td>0.166667</td>
<td>0.444444</td>
<td>0.271616</td>
</tr>
<tr>
<td>3</td>
<td>LogReg (py_entitymatching_matcher.logRegMatcher LogRegMatcher object at 0x0000000011C105CD)</td>
<td>5</td>
<td>0.500000</td>
<td>0.533333</td>
<td>0.250000</td>
<td>0.428571</td>
<td>0.400000</td>
<td>0.442381</td>
</tr>
<tr>
<td>4</td>
<td>NB (py_entitymatching_matcher.nBMatcher NBMatcher object at 0x0000000011C10668)</td>
<td>5</td>
<td>0.538462</td>
<td>0.700000</td>
<td>0.250000</td>
<td>0.470588</td>
<td>0.526316</td>
<td>0.497073</td>
</tr>
</tbody>
</table>
After realizing that these scores are low across the board, we decided to relabel the sample more carefully.

Our first technique was to add ‘ABV’ to the output of the blocking table so that we could use this attribute to better label beers that we might be unsure about. This helped us better decide on matches when the name of the beer might have differed slightly between the two.

We encountered situations where we couldn’t tell whether a beer pair was a match, specifically relating to any ‘Stone Epic Vertical’ beer, which is the same beer but aged for an extra one year, one month, and one day. That’s why each Stone Epic Vertical beer has either ‘010101’ or ‘020202’ and so on. The ABV of these beers can vary sometimes, but not uniformly. So some iterations of this beer have similar ABVs, but some are different. The company’s website claims that each different variety has its own ‘twists and turns’ but it’s totally subjective, making it hard to determine whether it’s a match or not. In the end, we decided to label Epic Vertical a match with each other to maintain consistency, but this would result in some False Negatives down the road.

We also had trouble with any beer that had a specification of ‘Barrel Aged’ usually with an added classifier of whether it was aged in a bourbon or wine barrel or etc. It was clear that these were different by looking at their ABV values, so we labeled them differently.

Once we relabeled the data under these new rules, we ran the matchers again.

**(ITERATION 2)**

**Precision:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Num folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 DecisionTree</td>
<td><code>&lt;py_entitymatching.matcher.dtmatcher.DTMatcher object at 0x0000000011C104E0&gt;</code></td>
<td>5</td>
<td>0.866667</td>
<td>0.933333</td>
<td>0.866667</td>
<td>0.875000</td>
<td>0.777778</td>
<td>0.863889</td>
</tr>
<tr>
<td>1 RF</td>
<td><code>&lt;py_entitymatching.matcher.rfmatcher.RFMatcher object at 0x0000000011C1050&gt;</code></td>
<td>5</td>
<td>0.923077</td>
<td>1.000000</td>
<td>0.866667</td>
<td>0.888889</td>
<td>0.875000</td>
<td>0.910726</td>
</tr>
<tr>
<td>2 SVM</td>
<td><code>&lt;py_entitymatching.matcher.svmmatcher.SVMMatcher object at 0x0000000011C104A8&gt;</code></td>
<td>5</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.777778</td>
<td>0.666667</td>
<td>0.600000</td>
<td>0.848889</td>
</tr>
<tr>
<td>3 LogReg</td>
<td><code>&lt;py_entitymatching.matcher.logregmatcher.LogRegMatcher object at 0x0000000011C105C0&gt;</code></td>
<td>5</td>
<td>0.888889</td>
<td>0.900000</td>
<td>0.846154</td>
<td>1.000000</td>
<td>0.777778</td>
<td>0.882564</td>
</tr>
<tr>
<td>4 NB</td>
<td><code>&lt;py_entitymatching.matcher.nbmatcher.NBMatcher object at 0x0000000011C10668&gt;</code></td>
<td>5</td>
<td>0.769231</td>
<td>0.900000</td>
<td>0.705862</td>
<td>0.714286</td>
<td>0.777778</td>
<td>0.773435</td>
</tr>
</tbody>
</table>
Recall:

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Num folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 DecisionTree</td>
<td>&lt;py_entitymatching.matcher.dtmacher.DTMatcher object at 0x0000000011C104E0&gt;</td>
<td>5</td>
<td>0.866667</td>
<td>0.933333</td>
<td>1.000000</td>
<td>0.875</td>
<td>0.875</td>
<td>0.910000</td>
</tr>
<tr>
<td>1 RF</td>
<td>&lt;py_entitymatching.matcher.rmatcher.RFMatcher object at 0x0000000011C104E0&gt;</td>
<td>5</td>
<td>0.800000</td>
<td>0.800000</td>
<td>1.000000</td>
<td>1.000</td>
<td>0.875</td>
<td>0.855000</td>
</tr>
<tr>
<td>2 SVM</td>
<td>&lt;py_entitymatching.matcher.svmmatcher.SVMMatcher object at 0x0000000011C104A8&gt;</td>
<td>5</td>
<td>0.600000</td>
<td>0.666667</td>
<td>0.538462</td>
<td>0.500</td>
<td>0.500</td>
<td>0.561026</td>
</tr>
<tr>
<td>3 LogReg</td>
<td>&lt;py_entitymatching.matcher.logregmatcher.LogRegMatcher object at 0x0000000011C105C0&gt;</td>
<td>5</td>
<td>0.533333</td>
<td>0.600000</td>
<td>0.846154</td>
<td>0.500</td>
<td>0.875</td>
<td>0.670897</td>
</tr>
<tr>
<td>4 NB</td>
<td>&lt;py_entitymatching.matcher.nbmatcher.NBMatcher object at 0x0000000011C10668&gt;</td>
<td>5</td>
<td>0.666667</td>
<td>0.600000</td>
<td>0.923077</td>
<td>0.625</td>
<td>0.875</td>
<td>0.737949</td>
</tr>
</tbody>
</table>

F1:

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Num folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 DecisionTree</td>
<td>&lt;py_entitymatching.matcher.dtmacher.DTMatcher object at 0x0000000011C104E0&gt;</td>
<td>5</td>
<td>0.866667</td>
<td>0.933333</td>
<td>0.928571</td>
<td>0.875000</td>
<td>0.823529</td>
<td>0.885420</td>
</tr>
<tr>
<td>1 RF</td>
<td>&lt;py_entitymatching.matcher.rmatcher.RFMatcher object at 0x0000000011C104E0&gt;</td>
<td>5</td>
<td>0.857143</td>
<td>0.888888</td>
<td>0.928571</td>
<td>0.941176</td>
<td>0.875000</td>
<td>0.898156</td>
</tr>
<tr>
<td>2 SVM</td>
<td>&lt;py_entitymatching.matcher.svmmatcher.SVMMatcher object at 0x0000000011C104A8&gt;</td>
<td>5</td>
<td>0.750000</td>
<td>0.800000</td>
<td>0.536364</td>
<td>0.571429</td>
<td>0.618385</td>
<td>0.674635</td>
</tr>
<tr>
<td>3 LogReg</td>
<td>&lt;py_entitymatching.matcher.logregmatcher.LogRegMatcher object at 0x0000000011C105C0&gt;</td>
<td>5</td>
<td>0.666667</td>
<td>0.720000</td>
<td>0.846154</td>
<td>0.666667</td>
<td>0.823529</td>
<td>0.744603</td>
</tr>
<tr>
<td>4 NB</td>
<td>&lt;py_entitymatching.matcher.nbmatcher.NBMatcher object at 0x0000000011C10668&gt;</td>
<td>5</td>
<td>0.714286</td>
<td>0.720000</td>
<td>0.800000</td>
<td>0.666667</td>
<td>0.823529</td>
<td>0.744896</td>
</tr>
</tbody>
</table>

Obviously, these scores are much improved. We selected the Random Forest matcher as that had the highest F1.

After debugging this to discover why it wasn’t higher, we found a few issues. Mainly, our inclusion of ABV in the previous step helped us decide on which pairs were matches and which weren’t, but because we didn’t include any feature vectors relating to ABV in our matchers, they did not consider ABV. So some of the beers that we classified because of ABV even though their names were similar (like the ‘Barrel Aged’ beers) were False Positives under the current matchers.

So, we wanted to add some feature vectors that matched based on ABV, but not all of that data was clean, some values were not numerical and some beers didn’t have alcohol in them and were thus listed on the website as ‘N/A’. After replacing every non-numerical value with 0 as a quick fix, we were able to use ABV feature vectors.
This addition brought a lot of improvement to our mean scores, especially for Decision Tree and Random Forest. There are some other small fixes we could have added if we had more time, like making each ABV value more accurate so that a beer without a reported ABV wouldn’t be 0. Then we could write our own feature vector. However, considering the data, we deemed the above values, especially for the Decision Tree and Random Forest, good enough.
Our selection for best matcher is the Random Forest. It’s precision, recall, and F1 from above are:

- Precision: 0.938611
- Recall: 0.973333
- F1: 0.953212

After testing it on testing set J, we got:

**Precision:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Num folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 RF</td>
<td>&lt;py_entitymatching.matcher:rmatcher.RFMatcher object at 0x000000001D4BF6D&gt;</td>
<td>5</td>
<td>0.928571</td>
<td>1.0</td>
<td>0.642857</td>
<td>0.509091</td>
<td>0.916667</td>
<td>0.879437</td>
</tr>
</tbody>
</table>

**Recall:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Num folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 RF</td>
<td>&lt;py_entitymatching.matcher:rmatcher.RFMatcher object at 0x000000001D4BF6D&gt;</td>
<td>5</td>
<td>1.0</td>
<td>0.666667</td>
<td>0.818182</td>
<td>0.742424</td>
<td>1.0</td>
<td>0.839827</td>
</tr>
</tbody>
</table>

**F1:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Matcher</th>
<th>Num folds</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 RF</td>
<td>&lt;py_entitymatching.matcher:rmatcher.RFMatcher object at 0x000000001D4BF6D&gt;</td>
<td>5</td>
<td>0.962963</td>
<td>0.8</td>
<td>0.72</td>
<td>0.8</td>
<td>0.956522</td>
<td>0.847897</td>
</tr>
</tbody>
</table>

These are lower than when it trained on I, which suggests some more fixes are needed that hadn’t manifested in the training data.

**FEATURE VECTORS**

Our final list of feature vectors is listed here:

8     brewery_brewery_lev_dist
9     brewery_brewery_lev_sim
10    brewery_brewery_nmww
11    brewery_brewery_sw
12    name_name_jac_qgm_3_qgm_3
13    name_name_cos_dlm_dcm_dlm_dcm
14    name_name_jac_dlm_dcm_dcm
15    name_name_mel
16    name_name_lev_dist
17    name_name_lev_sim
18    name_name_nmww
19    name_name_sw
28    abv_abv_exm
29    abv_abv_anm
30    abv_abv_lev_dist
31    abv_abv_lev_sim
Name: feature_name, dtype: object
TIME ESTIMATES:

It’s somewhat difficult to estimate time spent, since little mistakes tended to cause rather large time delays.

**Labeling: 6 hours.** We spent many hours at the beginning trying to figure out why our label data wasn’t working with Magellan, only to find out that there was a non-integer value in one of our tableA tuple’s ID field. Then we had to do more labeling once our first iteration revealed that our first run at labeling was very inconsistent.

**Finding best matcher: 4 hours.** After we got our data to run through the matchers with improved accuracy after the relabel. It was mainly an effort to clean the ABV data and devise a way to include feature vectors that take ABV into account. After iteration 3, we were happy to land on Random Forest as our best matcher.

Finding higher precision, recall, and F-1 is probably possible if we were to refine our data even more by better cleaning the ABV values to be all numerical and accurate. We could also add feature vectors that take style into account, although we avoided doing this because the different tables have different ways of writing the different styles (India Pale Ale vs IPA, etc). If we had more time, we could compile a dictionary of styles and convert each style value in each table into one of those styles, and then add feature vectors comparing those styles. At some point, it’d be impossible to improve F-1 anymore, because even us humans had trouble deciding whether some beer pairs really were ‘matches’. 