Goal: review the false positive/false negative/unknown data and generate a list of potential modifications we can make to our learner's code. Much of this may fall into the "hand-coded rules" side of things. Once we have the list (or as we build it), we can try making these changes and note whether it has a positive or negative impact (obviously just want to keep the positive ones).

Observation: Based on unknowns.txt, we are throwing out a lot of pairs where the product name is identical. If the product name is identical, almost always this is a match (in fact, I have not seen a contradictory case).
Solution: If product name is identical between the pairs, do not classify as UNKNOWN. Also, no need to pass down to the model-based matcher. We can just hard-code in a MATCH in this case. Or, to make the code less complex we can just "trick" the matcher by setting identical string value of "DUMMY" in all other attributes for the product, so it will not be tossed out, will have perfect sim scores for the features and the classifier will determine it to be a MATCH.
Expected Result: Recall will definitely increase, precision may even increase as well.
Actual Result: **Old precision - 91.36, New precision - 92.5**
Old recall - 85.97, New recall - 86.3

Observation: I see from the first false negative example that this must be a case where the brand name is not in our dictionary. Otherwise it would have been extracted for the second product and this would have been a match.
Solution: We can enhance our brand name dictionary by extracting all brands from the products in set x, y, even the unlabelled dataset and add all of those brand names to variation_brand_names.txt
Expected Result: Recall will increase, we'll have to see what effect it has on precision, but unlikely to be negative.
Actual Result: **Precision and recall not significantly changed from original values, but did eliminate at least one false negative. Can't hurt to have a longer brand dictionary, so might as well include this change in the final matcher.**

Observation: The Product Long Description attribute has a bunch of HTML tags for some products. This is not useful for comparing this attribute for string similarity, and may throw off the string similarity measure since we should just be focusing on the words in this attribute.
Solution: Clean the data by removing all HTML tags from the product descriptions.
Expected Result: Recall will increase, unlikely to have a negative effect on precision.
Actual Result: **Precision and recall unchanged from the new precision and recall obtained from first observation**

Observation: There are some false positives where the product attributes are very, very similar and the only difference is the color. Color isn't actually an attribute but appears in the product name or description
Solution: We can generate a Color attribute for each product by combing through the name and description attributes. If two products have a color listed and it's different, automatically flag

them as a MISMATCH. If one has a color and the other doesn't, don't automatically flag them as a mismatch. Likewise, if the color *does* match, don't automatically flag them as a MATCH since they might still be completely different. We can do this again by altering the data before passing it to the classifier (for example put "aaa" as the value for all of product 1's attributes, "bbb" as the value for all of product 2's
Expected Result: Precision will increase. Recall will hopefully be unaffected. This would be a good one to test before/after.
Actual Result:

Observation: A number of false positives share identical values for most of the attributes except Product Name (and description).
Solution: Give our features that are based on Product Name more weight.
Expected Result (impact on precision/recall): Precision should increase, recall may decrease. The precision increase might not be worth the decrease in recall.
Actual Result: **Tried several weights (1-5) for Product Name based features, no effect was observed. Might be worth a try again.**

Observation: Some false positives have a unit of measurement in the product name or description which is the only difference between the two products. For example, two tablets might be identical except one is 7" screen and one has 10" screen.
Solution: We could try to extract this into a new attribute, and automatically classify a pair as MISMATCH if this attribute does not match for the two products in a pair.
Expected Result (impact on precision/recall): This would be tricky to implement but would increase precision, may dec
rease recall.
Actual Result:

Observation: For some false positives, the only difference between the products is a product ID value that is contained within the product name. Example: Panasonic KX-TG1062M vs. Panasonic KX-TS500W.
Solution: If we could extract this value into a new attribute, we could automatically classify any products with non-matching ID values as a MISMATCH.
Expected Result (impact on precision/recall): Would increase precision. Would likely be challenging to implement. May require complex regular expression or a very large dictionary of words.
Actual Result:

Observation: For some false positives, the only difference is the word "refurbished" somewhere in the product name or description.
Solution: If "refurbished" appears in one product's name or description but not the other's automatically declare a mismatch.
Expected Result (impact on precision/recall): Should improve precision slightly
Actual Result: Minor increase in precision; recall unchanged.

Template below (copy and paste above this line)

Observation:
Solution:
Expected Result (impact on precision/recall):
Actual Result: