## 1 code

The code for this solution is at: https://github.com/camjm21/CS4400-FinalProjML

## 2 Solution outline

The solution includes five steps: (1) Data reading and EDA, (2) Blocking, (3) Feature engineering, (4) Model training and (5) Generating output.

### 2.1 Data reading and EDA

In this step, we read the left table and right table, as well as the training set. We explore the dataset to get some ideas of designing the solution. For example, we found that the left table has 2554 rows, and the right table has 22074 rows, so there are 2554*22074=56376996 pairs. Examining every pair is very inefficient, so we will need a blocking step to reduce the number of pairs that we will work on.

### 2.2 Blocking

We perform blocking on the attribute "brand", using an Attribute Equivalence Blocker from the entitymatching module. This generated a candidate set of id pairs where the two ids in each pair share the same brand. This is based on the intuition that two products with different brand are unlikely to be the same entity. Our blocking method reduces the number of pairs from 56376996 to 256340.

### 2.3 Feature engineering

I used the entitymatching module to generate features for the title of the product. This created a dataaframe with 5 features such as levenshtein, Cosine, jaccard, and monge_elkan. A similarity score is assigned to these and the model training is done upon the dataframe of this similarity score.

### 2.4 Model training

I first determined which model would be best by using a feature through entity_matching. Random Forest was the best model and provided me with an F1 score of .47, which is much better than the sample solution. I then trained the model on the training dataframe I set up and then predicted the candidate set matches. Since the number of non-matches is much more than the number of matches in the training set, we set class_weight="balanced" in random forest to handle this training data imbalance problem. We perform prediction on $X_c$ to get predicted labels $y_c$ for the candidate set.

### 2.5 Generating output

The pairs with $y_c = 1$ are our predicted matching pairs M. We remove the matching pairs already in the training set from M to obtain $M^-$. Finally, we save $M^-$ to output.csv. My final output final has 372 rows of matches.

## 3 Citations

I used the py_entitymatching module to help me block the candidate set, and also determine which matcher I wanted to use. I also used their workflows from online to develop my knowledge on the process of machine learning. I imported RandomForestClassifier from sklearn.ensemble to use as my matcher. And I imported numpy and pandas to help with overall data management.