Data Science Stage 4

Anuja Golechha, Shruthi Nambiar, Ushmal Ramesh

[1]Dept of Computer Sciences
University of Wisconsin,Madison

1. Combining two tables:

We list below the steps followed in combining table A and table B:

- For our first step we block the two tables AB using overlap blocker. We focus on certain features like name and resolution while doing our blocking. After this step 1027 tuple pairs survive.
- We then use a Random Forest model we got in Stage 3, and run it on our complete blocked dataset.
- Our model then labels the candidate tuple pairs as matched and unmatched pairs. We use this list of matched and unmatched pairs to get our final merged Table E
- We begin by extracting the id's of all matched and unmatched pairs and their corresponding labels.
- Matched and unmatched pairs are grouped separately. We then run through the list of all pairs in order to eliminate duplicate entries. The algorithm to eliminate duplicates is explained below.
- After running the algorithm mentioned in subsection 1.1, we are left with id's of the tuple pairs we need to pick from table A and table B.
- We then use the id's as primary keys, extract the rows from Table A and B, and merge it into a single new Table E.

1.1. Eliminating Duplicates:

In order to ensure that no duplicate entities remain in the merged table, we do the following operations:

- We first separate the left_matched,left_unmatched,right_matched and right_unmatched into four different lists.
- A set operation is run on each of them that outputs only the unique entries
- Our next step is that if we tuple pairs match in two entries, for example
  If we have (10,20) and (10,30) as two matching tuple pairs, then only 10 will survive. This is because all the three refer to the same entity.
  Similarly if we have (10,20) and (10,30) as two unmatched entries we retain 10,20 and 30.
- We do a union of all the left entries and take the difference of all the right entries. We return the left table id's and right table id's that survive this step.

2. Statistics on table E

Schema of Table E: We list each of the features that the final merged table E contains. ID, 3.5mm jack, Battery Card slot, Clock Speed, GPS Support ,GPU, Internal Storage, NFC, Name, OS, Price, Primary Camera ,Resolution, SIM ,Secondary Camera, Sensors

Weight

Number of tuples are in Table E: 279 tuples
Sample tuples:
Example 1:

Yes 3000 microSD, up to 32 GB (dedicated slot) Quad-core 1.3 GHz Cortex-A7 Yes Mali-400MP2 8 GB, 2 GB RAM Micromax Canvas Juice 2 AQ5001 Android 5.0 (Lollipop) 154.7 8 MP, autofocus, LED flash 720 x 1280 pixels, 16:9 ratio (294 ppi density) Dual SIM (Micro-SIM, dual stand-by) 2 MP Accelerometer, proximity 170 g (6.00 oz)

Example 2:

Yes 3000 microSD, up to 256 GB Octa-core 2.0 GHz Cortex-A53 Yes Adreno 506 64 GB, 4 GB RAM or 32 GB, 3 GB RAM Yes Motorola Moto G5S Plus Android 7.1 (Nougat), planned upgrade to Android 8.0 (Oreo) 249.9 Dual: 13 MP, f/2.0, autofocus, dual-LED dual-tone flash, check quality 1080 x 1920 pixels, 16:9 ratio (401 ppi density) Single SIM (Nano-SIM) or Dual SIM (Nano-SIM, dual stand-by) 8 MP, f/2.0, LED flash Fingerprint (front-mounted), accelerometer, gyro, proximity 168 g (5.93 oz)

Example 3:

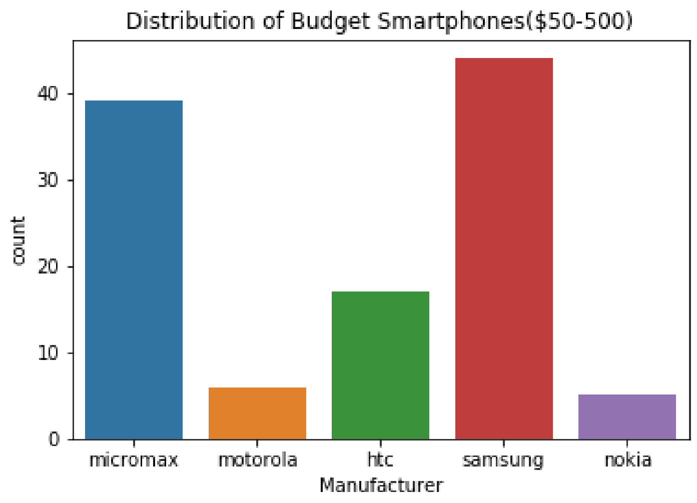
[2 Yes 2800 microSD, up to 256 GB (dedicated slot) Octa-core 1.3 GHz Cortex-A53 Yes Mali-T720MP3 16 GB, 2 GB RAM - Desire 72816 GB, 1.5 GB RAM - Desire 728G HTC Desire 728 dual sim Android 5.1.1 (Lollipop) 285.6 13 MP (f/2.2, 28mm), autofocus, LED flash 720 x 1280 pixels, 16:9 ratio (267 ppi density) Dual SIM (Nano-SIM, dual stand-by) 5 MP (f/2.8, 34mm), 1080p@30fps Accelerometer, proximity, compass 153 g (5.40 oz)]

Example 4:

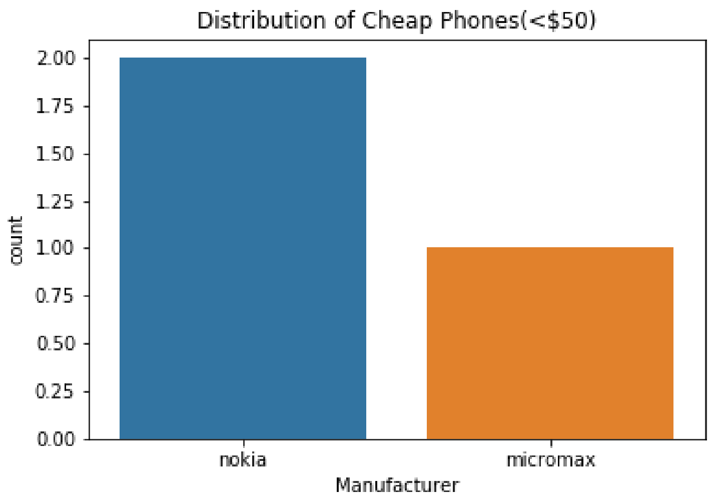
Yes 3050 microSD, up to 64 GB (dedicated slot) Quad-core 1.3 GHz Cortex-A53 Yes Mali-T720 16 GB, 3 GB RAM Micromax Canvas 2 Q4310 Android 7.0 (Nougat) 180 13 MP, f/2.0, autofocus, LED flash 720 x 1280 pixels, 16:9 ratio (294 ppi density) Dual SIM (Micro-SIM, dual stand-by) 5 MP, f/2.0 Fingerprint (front-mounted), accelerometer, proximity 160 g (5.64 oz)

3. Data Analysis

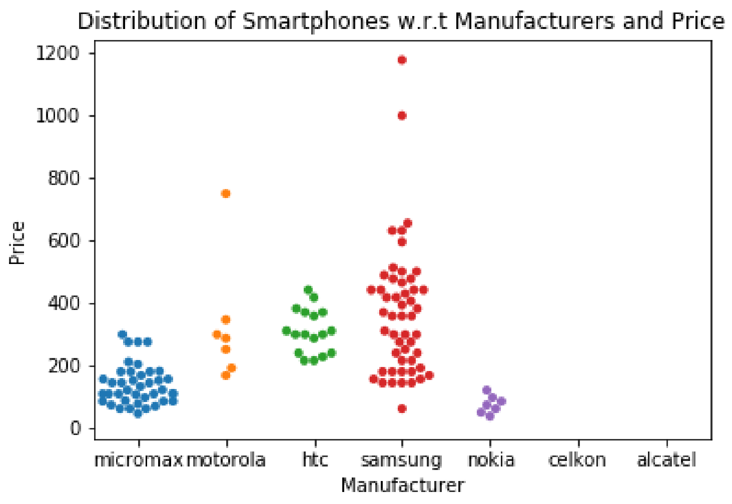
The graph below shows the distribution of Budget smartphones. The graph brings out the fact thet Micromax, which is an Indian company is emerging as a competitor to the behemoth Samsung in this crowded segment. Micromax aims to capture this lucrative market in India by mass producing budget phone with all the bells and whistles of flagship phones.



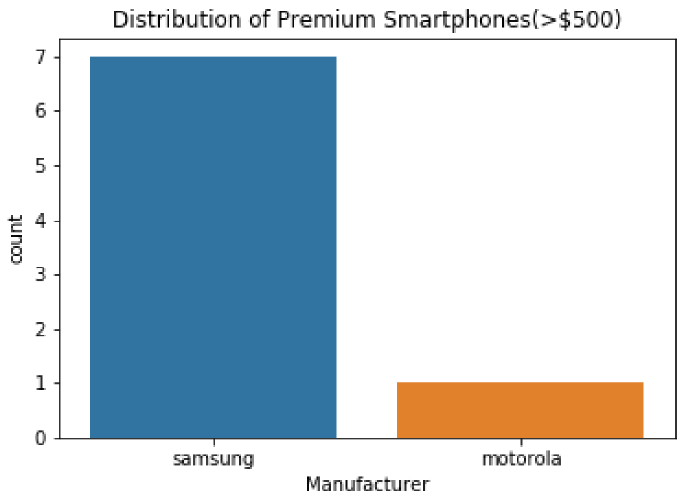
The graph below shows the distribution of cheap or 'feature' phones. The plot shows that Nokia is still the King of the hill in this area. The phones manufactured by the company are durable and one of the earliest models to be introduced. Its hard for any newcomer to capture this stagnated segment.

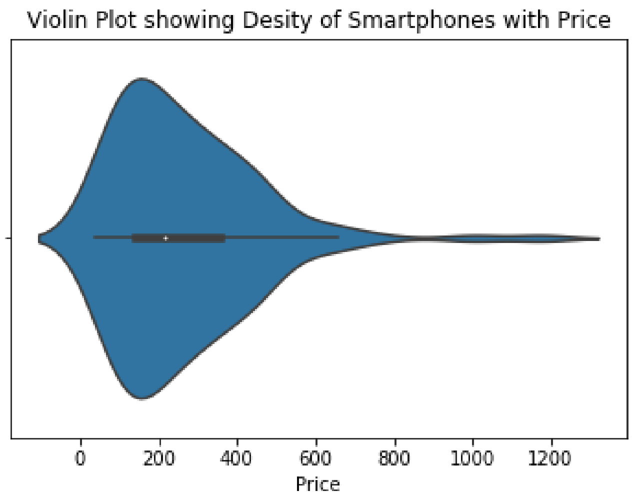


The above plot shows the distribution of phones with respect to all manufacturers. We can immediately spot that Samsung has products across a wide range of prices trying to cover every market segment. Companies like Micromax focus on niche segment of low end devices.



When it comes to premium smartphones, Samsung has a comfortable base with new versions and models coming out every year. The above plot highlight this dominance with Motorola coming in a distant second.

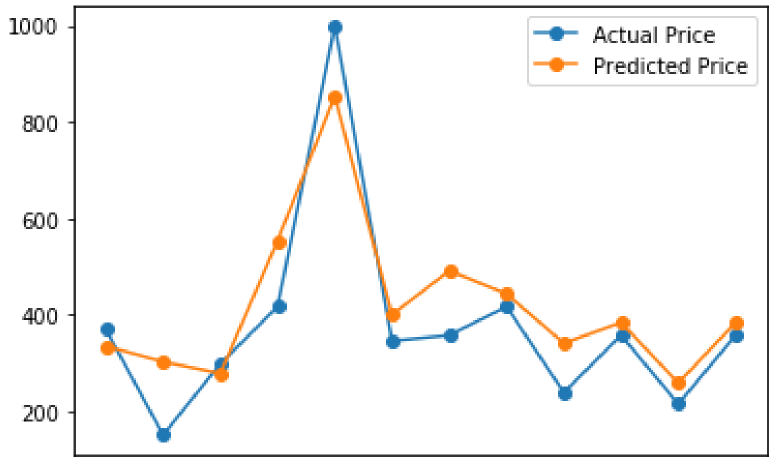




This violin plot shows that the bulk of smartphones are in mid range category.

4. Accuracy measures

We also attempted to predict the price of a smartphone based on its features. We first carried out some feature engineering. We converted binary features like NFC, GPS Support etc to 0 or 1 and extracted numeric values from features like Primary Camera, Secondary Camera, Resolution, Battery etc. We dropped any rows containing missing values for this analysis. We built a linear regression model with regularization to avoid overfitting using half the data for training. We predicted the prices on the test data.



This plot shows the predicted prices versus the actual prices for the test data.

5. Conclusion

From the data we conclude that we can predict the price of a phone from features like NFC, GPS Support, Camera, Resolution and Battery. However, the prediction is made difficult by thee fact that some manufacturers try to bring down the price of a phone to capture a market segment by packing the phone full of features but still charging a reduced price. We analysed the data and found that Micromax focuses on niche segment of low end devices and is therefore a competitor for Samsung in this area. Samsung on the other hand has products spread over the entire price spectrum and has phones having a plethora of feature combinations to fit almost every price bracket.

6. Future work

We can augment our data set by adding data regarding pricing trends of smartphone components like CPU, GPU, Camera, Sensors etc. With this data, we can do analysis on what percentage of a smartphone's total cost each component occupies. This analysis can shed light on what each manufacturer thinks is crucial for a phone designed for a particular market segment. For instance, in the low end budget device category, we would expect that manufacturers would go for a mid range CPU and bump up the phone's storage and Battery as these are features that a budget user would care about more than device performance. On the contrary, in a premium phone the CPU and GPU would capture a lion's share of the phone's price.

The phone's Screen would also play an important part in this analysis. Since the screen is an important consideration regardless of the market segment, it would be interesting to see what percentage of total manufacturing cost is dedicated to the Screen across different price brackets. This data will highlight Design decisions that these manufacturers have to make while designing these products to ensure its success in the targeted segment.

6.1. Code

Below is the python code that we used to merge the two files:

```python

import py_entitymatching as em
A = em.read_csv_metadata('new1_1.csv', key='ID')#Files before
    cleaning
B = em.read_csv_metadata('new2_1.csv', key='ID')


rf= em.RFMatcher()
rf.fit(table=train_set, exclude_attrs=['_id', 'ltable_ID', '
    rtable_ID', 'Class'], target_attr='Class')
match_f = em.get_features_for_matching(A, B)
new = em.extract_feature_vecs(C3, feature_table=match_f,
    attrs_before = None)
new.fillna(value=0, inplace=True)


op = rf.predict(table=new,  exclude_attrs=['_id', 'ltable_ID', '
    rtable_ID'],  append=True, target_attr='Predicted_Labels')


df1=op[op['Predicted_Labels']==1]
df2=df1[['ltable_ID','rtable_ID']]
df2.to_csv("matching.csv")
df3=op[op['Predicted_Labels']==0]
unmatched_id=df3[['ltable_ID','rtable_ID']]


l_lmatched = set([93, 104, 272, 296, 361, 462, 492, 602, 705,
    850, 865, 915, 920, 1014, 1084, 1106, 1136, 1184, 1326, 1392,
     1474, 1511, 1561, 1625, 1775, 1926, 1932, 2038, 2072, 2108,
    2124, 2139, 2150, 2167, 2234, 2241, 2292, 2333, 2390, 2410,
    2470, 2534, 2591, 2594, 2600, 2666, 2680, 2695, 2710, 2754,
    2831, 2885, 3025, 3127, 3175, 3224])
r_matched = set([93, 104, 272, 296, 361, 462, 492, 602, 705, 850,
     865, 915, 920, 1014, 1084, 1106, 1136, 1184, 1326, 1392,
    1474, 1511, 1561, 1625, 1775, 1926, 1932, 2038, 2072, 2108,
    2124, 2139, 2150, 2167, 2234, 2241, 2292, 2333, 2390, 2410,
    2470, 2534, 2591, 2594, 2600, 2666, 2680, 2695, 2710, 2754,
    2831, 2885, 3025, 3127, 3175, 3224])
l_unmatched = set(list(unmatched_id['ltable_ID']))
r_unmatched = set(list(unmatched_id['rtable_ID']))
op1= l_lmatched.union(l_unmatched)
op2= r_unmatched.difference(r_matched)
out1 = A[A['ID'].isin(op1)]
out2 = B[B['ID'].isin(op2)]
datanew = out1.append(out2)
datanew.to_csv("final_list.csv")
```