Final Project Report

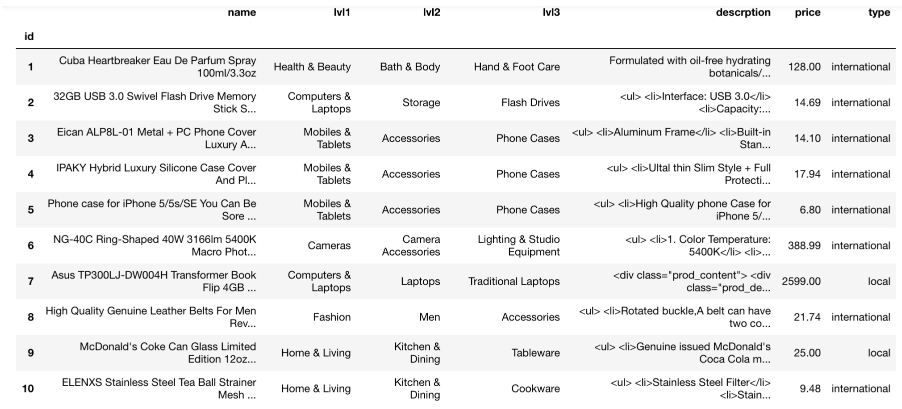
Chuqiao Dong

12-08-2018

1 Data preparation

1.1 preparing/cleaning data

The given data file contains seven columns and each of them containing seperate information. So before clean, the initial data file is presented in Figure 1.



id	name	lvl1	lvl2	lvl3	descrption	price	type
1	Cuba Heartbreaker Eau De Parfum Spray 100ml/3.3oz	Health & Beauty	Bath & Body	Hand & Foot Care	Formulated with oil-free hydrating botanicals/...	128.00	international
2	32GB USB 3.0 Swivel Flash Drive Memory Stick S...	Computers & Laptops	Storage	Flash Drives	<ul> <li>Interface: USB 3.0</li> <li>Capacity:...	14.69	international
3	Eican ALP8L-01 Metal + PC Phone Cover Luxury A...	Mobiles & Tablets	Accessories	Phone Cases	<ul> <li>Aluminum Frame</li> <li>Built-in Stan...	14.10	international
4	IPAKY Hybrid Luxury Silicone Case Cover And Pl...	Mobiles & Tablets	Accessories	Phone Cases	<ul> <li>Ultal thin Slim Style + Full Protecti...	17.94	international
5	Phone case for iPhone 5/5s/SE You Can Be Sore ...	Mobiles & Tablets	Accessories	Phone Cases	<ul> <li>High Quality phone Case for iPhone 5/...	6.80	international
6	NG-40C Ring-Shaped 40W 3166lm 5400K Macro Phot...	Cameras	Camera Accessories	Lighting & Studio Equipment	<ul> <li>1. Color Temperature: 5400K</li> <li>...	388.99	international
7	Asus TP300LJ-DW004H Transformer Book Flip 4GB ...	Computers & Laptops	Laptops	Traditional Laptops	<div class="prod_content"> <div class="prod_de...	2599.00	local
8	High Quality Genuine Leather Belts For Men Rev...	Fashion	Men	Accessories	<ul> <li>Rotated buckle,A belt can have two co...	21.74	international
9	McDonald's Coke Can Glass Limited Edition 12oz...	Home & Living	Kitchen & Dining	Tableware	<ul> <li>Genuine issued McDonald's Coca Cola m...	25.00	local
10	ELENXS Stainless Steel Tea Ball Strainer Mesh ...	Home & Living	Kitchen & Dining	Cookware	<ul> <li>Stainless Steel Filter</li> <li>Stain...	9.48	international

Figure 1: Data Frame Before Clean

After brief look through the initial data, this high level strategy has been made for cleaning:

· **name:** punctuation need to be removed; stop words need to be removed; numbers has little contribution and is better to remove.

· **lvl1:** punctuation need to be removed; rest categories need to be mapped.

· **lvl2:** punctuation need to be removed; rest categories need to be mapped and NAN is filled in 0.

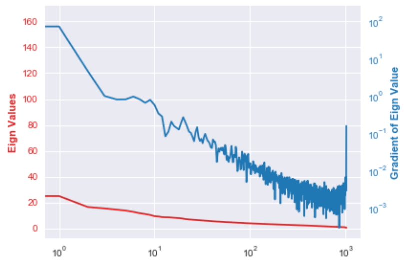
· **lvl3:** punctuation need to be removed; rest categories need to be mapped and NAN is filled in 0.

· **descrption:** punctuation need to be removed; html related symbols need to be removed; stop words need to be removed; numbers has little contribution and is better to remove.

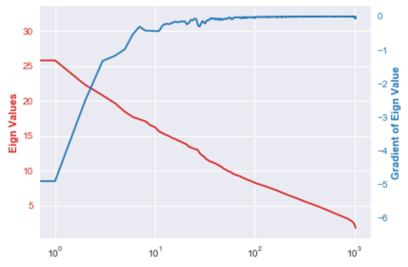
· **price:** negative values should be clipped to 0; abnormal large values should be clipped; rest price should be normalized.

· **type:** categories need to be mapped and NAN will be filled in 0.

And in the source code, $'pandas'$ was used to load data and for this high level cleaning purpose. For columns to be labeled (i.e. **lvl1,2,3** and **type**), function $'MultiColumnLabelEncoder'$ was used to transfer the text information into labels. Since this labels are normal values, there should not be difference whether a certain category is labeled as 1 or 2. Thus, one-hot encoding was applied to the results from $'MultiColumnLabelEncoder'$. And other columns' values are ordinal information, so they will not be applied to one-hot encoder.
For '**price**' information, since it is already represented in floats, there is no need to transfer. But still some cells showing price as 'negative' values and abnormal large values which should not be the truth. Thus, in this case, I set clip to the orginal price values with the lower bound of 0 and upper bound of 5000. After this clip, I use the $'Normalizer'$ method by $'l2'$ norm transfer price to be in range [0,1].
Now the remaining are two text columns. For **descrption** and **name**, I tried with the $'CountVectorizer'$, $'TfidfVectorizer'$, and $'HashingVectorizer'$ methods, even though $'HashingVectorizer'$ performed better, but since the training input is still relatively small, so this word-vectorize methods cannot really do a good job. And sometimes, the most frequent occured words are not really meaningful for training (i.e., 'to'). So I choose some pre-trained model for word-vectorization. In this source code, I used $'en\_core\_web\_lg'$. In this model, the english words were pretrained based on many websites' information and it is more trustful for word-vectorization comparing with performing word count only based on provided data set. And after word-vectorization, a 300 columns' vector was returned for each data set. This is too large and we only need the most important information. This means we only need to few largest eigenvalues. Thus, a $'TruncatedSVD'$ was applied to the results from $'en\_core\_web\_lg'$. Dimensions for **descrption** (30) and **name** (15) were chosen based on the derivative of eigenvalue approach 0.1 and 0 respectively (Figure 2). And this is for cleaning data only.



(a) Description



(b) Name

Figure 2: Eigenvalues and Derivatives

1.2 feature engineering

After cleaning data, this is still not enough for the model to learn the data set enough, so I created some new features to get more information from the data set.

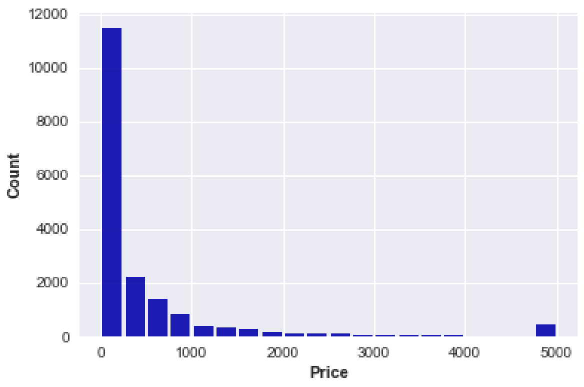


Figure 3: Data Frame Before Clean

For '**price**' column, even though I have already normalized the values, when plotting the histogram of the original price values (Figure 3), it is clear that majority of the values comes from a vary some range and the rest are very sparse. This is not a typical normal distribution so that I need to add one more feature to represent this biased distribution information. So I separated the total price range into 20 bins, and for each of the price values, it will be given a new feature based on which bin to locate and totally returns a 20 dimension results. And since each bin is a ordinal variable, so this will also not be applied to one-hot encoder.

For **descrption**, not only each word it contains is important, but how many sections it has (represents as the number of $< li >$). So in this case, I also count the section number as a new 1-D feature added to the overall features. And the word number as well as the string length are also important features in **descrption** and **name**. So 2-D feature containing these counts were also added for each of them (total 4-D). So, by combining all these features, the 'outfile' matrix was generated ready to train the model.
And also since the training data set is about the same size of the test data set, so, it is not reasonable to clean them separately. In this project, I concatenate them together for data cleaning and separate for training the model.

2 Models

2.1 successful models/algorithms

Successful models are: $'LogisticRegression'$, $'RandomForestClassifier'$, $'GradientBoostingClassifier'$, and $'NeuralNetwork'$.

2.2 failed models/algorithms

Failed models are: $'MultinomialNB'$, $'KNeighborsClassifier'$, simple $'DecisionTreeClassifier'$.

2.3 possible causes of success/failure

The reason for those models failed to predict the results mainly due to the number of noise is relatively large, and the features transferred to train the model is not too complicated for the simple decision tree classifier. And this is also the same reason those models performed good. Because they can capture high noised input data and they do some regression more or less, so that they have the ability to contain some complex features (not easy to be classified features).

3 Results

3.1 final results

My final results is based on the softmax ensemble voting of $'LogisticRegression'$, $'RandomForestClassifier'$, $'GradientBoostingClassifier'$ and then averaged with $'NeuralNetwork'$ results. The results raw data is attached (Leaderboard rank # 7).

3.2 interpretation of the results

Eventhough $'LogisticRegression'$, $'RandomForestClassifier'$, $'GradientBoostingClassifier'$ all have relatively good performance on this project, but they may still be trapped by some of the data points. And they may not likely to be trapped with the same data. So by using the softmax ensemble of results from these three methods, this rare mis-classification could be reduced. In addition, neural network may cause overfitting since the training data set is not large enough and may not perform well with the relatively large test data set. But neural network is very suitable to capture some complex features. So by averaging results from neural network and that ensemble vote from simple classifiers, this result would be able to perform well both in simple and complex features.

4 Lesson learned

The most important lesson I learned from this project is the importance of cleaning dirty data and how to choose the 'suitable' way to clean it. We need initially have a overview of the data set to decide the right way for cleaning. Moreover, features for training data need to be engineered before given to different training models. In order to do this, we need to manually put efforts by analyzing data and extract those information that could be ignored after simply clean. Take **decrption** as an example, for cleaning the text, we need to delete all the punctuation, but some punctuation may also contain information like section number information. In this case, we still need to working on the data engineering to gaining this information and add it separately to the feature matrix. Moreover, training-test data size may also influence the choice of model. And single model may not be able to perform good in every feature, so ensemble is important especially for noise and complex feature data.
For source code, please see attached source code file or in my GitHub repo.