Name: Bharath Samala

Email address: sai.jan001@gmail.com

Contact number: 9160290485

Anydesk address: **297 887 699**

Date: **19 July 2020**

Self Case Study -1: Mercari Price Suggestion Challenge

Overview:

Mercari is a Japanese community-powered shopping app where people can sell their stuff they don't use but still has value. A typical listing on the website has product name, brand, category, shipping, condition, product description, and price. Sellers are allowed to choose any price for their product. This gives rise to the problem of setting the right price for a product. If the price is too high the product won't find a buyer or if it's too low then the seller will be getting less profit. So, if we can suggest a price for the given product listing, it helps the seller to set a better price for the product.

For example, given sweater A and sweater B can we predict the price of the sweaters?

Sweater A:

"Vince Long-Sleeve Turtleneck Pullover Sweater, Black, Women's, size L, great condition."

Sweater B:

"St. John's Bay Long-Sleeve Turtleneck Pullover Sweater, size L, great condition"

The price of a product may depend on many factors like, brand name, demand and supply, type of product and even picture of the product. For example, electronic products of the same category and brand may have different pricing based on small changes in specifications.

The task at hand is to build a machine learning model that automatically predicts the price of a product when given a product name, category, brand, shipping, product conditions, and product description.

Problem statement:

Suggest price for a product, given name, brand name, category, shipping, item description, and condition of the product.

Constraints:

- This is a kernel only challenge, hence the solution should run within 60 min and use no more than 16gb RAM.
- Using external data is not allowed.

Metric:

- The evaluation metric for this competition is Root Mean Squared Logarithmic Error.

Data:

File names: train.tsv, test.tsv

The files consist of a list of product listings. These files are tab-delimited (‘\t’)

Columns:

train_id or test_id - the id of the listing

name - the title of the listing. Note that we have cleaned the data to remove text that looks like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]

item_condition_id - the condition of the items provided by the seller

category_name - a category of the listing

Brand_name - the name of the brand if available else nan

price - the price that the item was sold for. This is the target variable that you will predict. The unit is USD. This column doesn't exist in test.tsv since that is what you will predict.

shipping - 1 if the shipping fee is paid by seller and 0 by buyer

item_description - the full description of the item. Note that we have cleaned the data to remove text that looks like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]

First Cut Approach:

EDA:

- Check the format of each feature.
- Check the distribution of each feature.
- Check for missing values, invalid values in features.
- Check the distribution of each feature w.r.t target.

Questions:

- How does price vary with and without shipping?
- Does branded items cost more than non-branded items overall and with respect to different categories?
- Does better item condition result in higher price overall and with respect to different categories?
- What is the average price of an item across each category and subcategory?
- How does price vary for items with and without description?
- How does price vary for items with missing values in any column?
- Does the length of item description affect the target?
- Does the number of words in item description affect the price?
- Does presence of number in item description affect the price?
- Does the subjectivity of the item description have any correlation to the price?

Preprocess:

- Remove invalid values if present(e.g negative price or 0 prices)
- Assign a default value to missing values.
- Remove special characters, stopwords from text fields.
- Combine Name and brand name into one field.
- Combine item description, name, and category into one field.
- Min_max scale the item_condition field.
- Apply log1 on the target.

Split:

- Split data randomly in an 80:20 ratio for train and CV.

Vectorization:

- Use TFIDFVectorizer to vectorize name and text fields.
- Use Dictvectorizer to transform shipping and item_condition fields.

Models:

- Train 4 regression models, ridge regression, linear regression, lightgbm, and MLP on train data and measure the performance on test data.
- Take an ensemble of different combinations of these models and see the performance on test data in kaggle.

Experiments on features:

- For the item_description field, we can measure the subjectivity value. If the value is near to zero the text could be explaining facts like specifications of the product like 8 GB ram, stainless steel. If the value is near to 1 the text could be appealing to the emotion of the buyer like a lovely teddy bear, great add on to your living room.
- Number in the item description, the presence of numbers like a pack of 5 t-shirts could increase the price of the product.

Useful kernels/blogs:

1.Mercari Golf: 0.3875 CV in 75 LOC, 1900 s:

Link: https://www.kaggle.com/lopuhin/mercari-golf-0-3875-cv-in-75-loc-1900-s

Summary:

- This kernel uses simple MLP that works on sparse features and it trains 4 models on 2 datasets.
- They create a new data set which is binary during the execution with just one line of code `X.astype(bool).astype(float) instead of using binary BOW.`
- For feature engineering, they combine name and brand field into one column and name, item description, and category field into one.
- Final features used are name(name+brand), text (item description + name + category), shipping and item condition.
- Trained 4 models on 4 cores in parallel and each one overfit to a different minima.
- Batch size is doubled after each epoch to make models overfit.
- Average the predictions of the four models to get the final predictions.
- From this kernel we can use the same featurization techniques like combining text fields and using dictvectorizer on shipping and item condition instead of one hot encoding.

2.More Effective Ridge LGBM Script (LB 0.44823):

Link: https://www.kaggle.com/tunguz/more-effective-ridge-lgbm-script-lb-0-44823

Summary:

- Categorical features like category, brand name are limited to a threshold number of popular values rest all assigned to single value ‘missing.
- This helps in reducing the dimensionality of data.
- Name and item description are featurized using BOW and TFidf respectively
- The author performed fit_transform operation on train and test together which should result in data leakage but it could save time.
- GC library is used to clear RAM when needed.
- Two ML models are trained on the data, ridge regression, and light gbm.
- The final prediction is the weighted sum of predictions of the two models.
- From this kernel we use the idea of training multiple models and combine the output of these models as final prediction.

3. What is LightGBM, How to implement it? How to fine-tune the parameters?

Link:
https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Summary:

- LightGBM is a gradient boosting framework that uses a tree-based learning algorithm.
- It grows trees leaf wise rather than level-wise like other algorithms.
- It is fast and uses less memory.
- It is prone to overfitting so it is best to use when we have a large dataset (10,000+).
- The blog explains the parameters of the lightgbm, implements lightgbm on a dataset, and also explains about hyperparameter tuning.
- This is a kernel-only competition, so there is a memory limit and a time limit of 60 min.
- We need a model that is powerful and also works with above constraints. Hence we choose lightgbm for this task.

4. ELI5: which features are important for price prediction¶

Link: https://www.kaggle.com/lopuhin/eli5-for-mercari

Summary:

- ELI5(Explain Like I'm 5) is an open-source library, to explain how each feature contributes to the prediction.
- This can work with models like xgboost, lightgbm along with linear models. One important requirement for this to work is, we have to use sklearn to vectorize features.
- If we can know how each feature is affecting the model performance we can make better decisions in changing featurization, preprocessing, or models to improve the performance.

There are two main ways to look at classification or a regression model:

1. We can get the feature importance of the model on overall data like below.

```
In [7]: eli5.show_weights(model, vec=vectorizer)
```

Out[7]:

y top features

Weight[?]	Feature
+2.971	<BIAS>
+1.840	item_description__14k
+1.829	item_description__limbs
+1.671	brand_name__david yurman
+1.662	item_description__carat
+1.617	name__dockatot
+1.606	item_description__unlocked
+1.530	name__tieks
+1.524	item_description__64gb
+1.468	brand_name__vitamix
+1.453	item_description__16gb
+1.445	item_description__500gb
+1.428	name__hatchimals
+1.422	name__hatchimal
+1.404	name__tria
+1.389	name__dyson
+1.378	name__movado
+1.362	name__007taz
+1.353	item_description__32gb
+1.344	item_description__imei
	... 81034 more positive ...
	... 74924 more negative ...

2.we can also get how each feature contributes to a prediction for a single data point.

```
In [9]: eli5.show_prediction(model, doc=train.values[100], vec=vectorizer)
```

Out[9]:

y (score **2.590**) top features

Contribution[?]	Feature
+2.971	<BIAS>
+0.248	item_description: Highlighted in text (sum)
-0.072	item_condition_id: Highlighted in text (sum)
-0.072	brand_name: Highlighted in text (sum)
-0.076	category_name: Highlighted in text (sum)
-0.156	name: Highlighted in text (sum)
-0.254	shipping: Highlighted in text (sum)

name: muscle t-shirt
category_name: women/tops & blouses/t-shirts
brand_name: missing
shipping: 0
item_condition_id: 3
item_description: what goes better with summer than tacos & tequila? chill out with friends sporting this great beachwear cover or wear as a stand alone with that oh so sexy bralette or bikini top! don't forget your cool shades! great condition! worn once, no stains, holes, rips or treats.

- We can look at errors in predictions, see what's going wrong with points with large errors using ELI5.

5. Extensive Text Feature Engineering:

Link: https://www.kaggle.com/shivamb/extensive-text-data-feature-engineering

Summary:

- This kernel implements different feature extraction techniques for text data along with other data on Donors choose dataset.
- Length of text, number of words, number of characters, number of punctuations, word density, stopword cout, article subjectivity etc.
- For measuring article subjectivity, the author uses the TextBlob library.
- In our dataset, we can apply these techniques on item_description column(text field).