kaggle

Kaggle BIPOC Final Showcase

Ms. Aye Hninn Khine
ayehninnkhine.nlp@gmail.com

Mentor: Mr. Qingchen Wang

June 17th, 2021

Mercari Price Suggestion Challenge

Presenter: Aye Hninn Khine @ Aye

Background

- 28 years old, Myanmar nationality
- Live in Hat Yai, Thailand
- 4th Year PhD Student in Computer Science (Prince of Songkla University)
- 2018 Google Women Techmakers Scholarship Recipient for Asia Pacific Region
- Current research - Medical Sentiment Analysis
- I love to share what I've learnt
    - Organized data science workshops
    - Give talks at local meetups (Google I/O Extended and GDG DevFest)

Project Definition

- Mercari Price Suggestion Challenge (predict selling price from given features)
- Python, Numpy, Pandas, matplotlib, Tensorflow, Keras, Wordbatch

Course	Course Provider	Status	Skills I've learned
Machine Learning Explainability	Kaggle	Completed	Using eli5 to visualize the prediction
Intro to AI Ethics	Kaggle	Completed	Types of Biases in AI
Feature Engineering	Kaggle	Completed	
Analytics for Manager	Qingchen Wang (The University of Hong Kong)	Completed	Data Analysis Statistical Analysis
MLOps Fundamentals	Coursera Google Cloud	Completed	MLOps, ML model deployment, GCP fundamental
Machine Learning with Tensorflow on GCP Specialization	Coursera Google Cloud	In Progress (three courses finished out of five)	GCP

Understanding the Data

The train set consists of over 1.4 million products and the phase 2 test set consists of over 3.4 million products.

Listing the field names in train/test data:

- *train_id* or *test_id* — unique id of the listing
- *name* — the product name by the seller. Note that to avoid data leakage the prices in this field are removed and represented as [rm]
- *item_condition_id* — here the seller provides the item conditions
- *category_name* — category listing for each item
- *brand_name* — the corresponding brands which each item belongs to
- *price* — this is our target variable and is represented in USD (column not present in test.tsv)
- *shipping* — 1, if the shipping fee is paid by the seller, and 0, otherwise
- *item_description* — Each item description is given here and the prices are removed and represented as [rm]

Sample Data

[8]:

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	[Men, Tops, T-shirts]	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	[Electronics, Computers & Tablets, Components ...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	[Women, Tops & Blouses, Blouse]	Target	10.0	1	Adorable top with a hint of lace and a key hol...
3	3	Leather Horse Statues	1	[Home, Home Décor, Home Décor Accents]	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	[Women, Jewelry, Necklaces]	NaN	44.0	0	Complete with certificate of authenticity

Evaluation

The evaluation of the competition is Root Mean Squared Loss Error (RMSLE)

RMSLE can be calculated as:

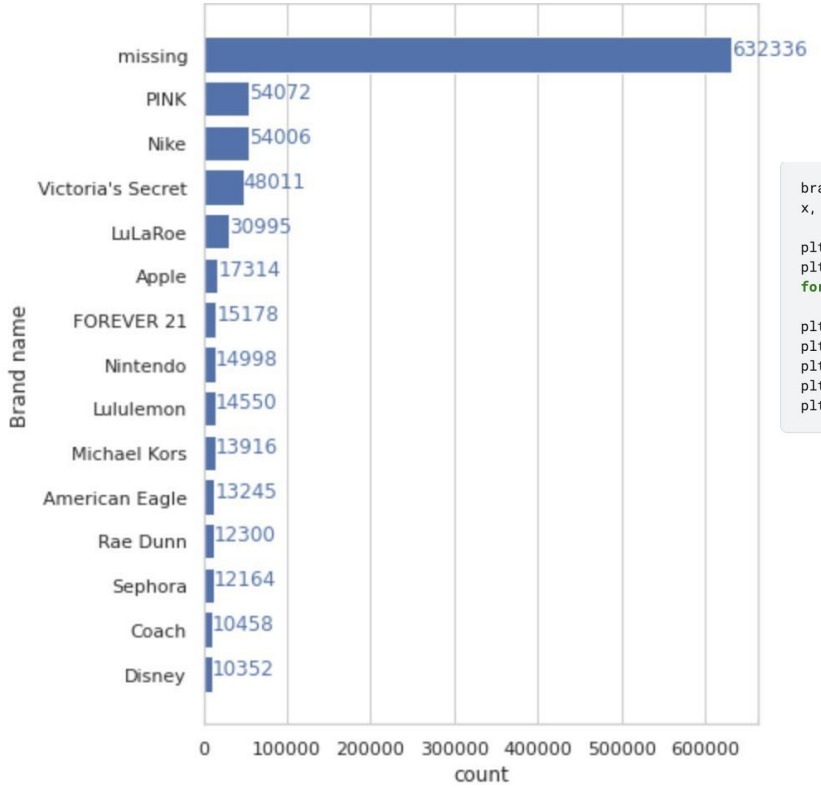
$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

The lower the error, the better the model

Exploratory Data Analysis (EDA)

```
condition_count = Counter(list(train.item_condition_id))
x, y = zip(*condition_count.most_common())
plt.figure(figsize=[8,6])
plt.bar(x, y, )
for i, val in enumerate(y):
          plt.annotate(val, (x[i], y[i]), color='b')
plt.xlabel('item condition')
plt.ylabel('count')
plt.grid(False, axis='x')
plt.show()
```

Cont'd: Exploratory Data Analysis (EDA)

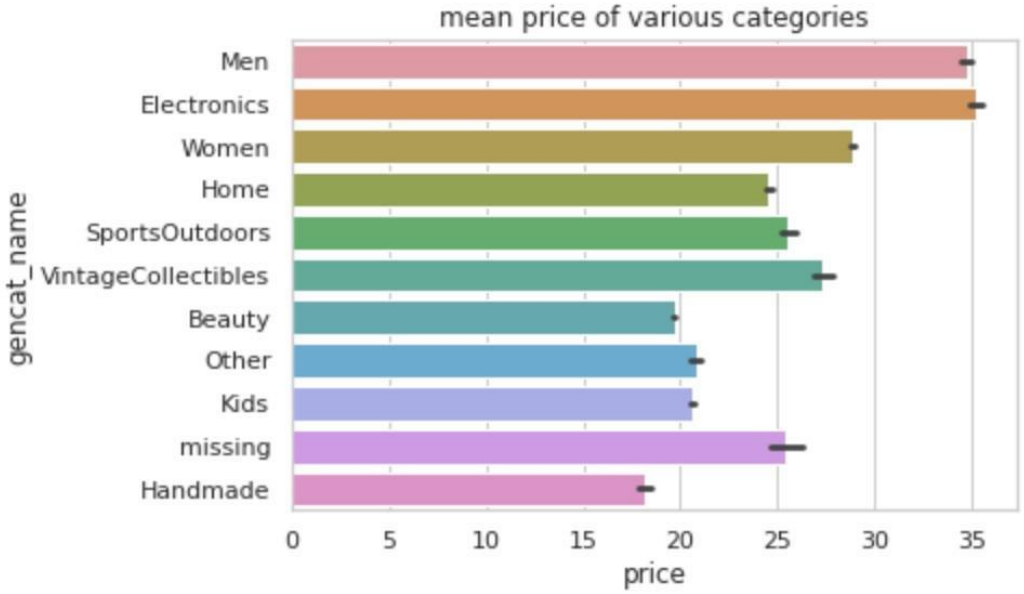



```
brand_count = Counter(list(train.brand_name.values))
x, y = zip(*brand_count.most_common(15))

plt.figure(figsize=[6,8])
plt.barh(x, y)
for i, val in enumerate(y):
          plt.annotate(val, (y[i], x[i]), color='b')
plt.gca().invert_yaxis()
plt.ylabel('Brand name')
plt.xlabel('count')
plt.grid(False, axis='y')
plt.show()
```

Cont'd: Exploratory Data Analysis (EDA)

```
sns.barplot(y='gencat_name', x='price', data=train)
plt.title('mean price of various categories')
plt.show()
```



Initial Experiments (Training Data - 70:30 Ratio)

Experiment	Features							Regression Model	RMSLE			
	train_id	name	item_condition_i	category_name	brand_name	shipping	item_description					
1	N	N	Y	N	N	Y	N	Linear Regression	0.8020218377			
2								Lasso	0.8131697857			
3								Random Forest	0.7997092848			
4								Gradient Boosting	0.7997118819			
5								Decision Tree	0.7997108234			
6	N	Y	Y	Y	Y	Y	N	Linear Regression	0.7887819457			
7								Random Forest	0.5743769903			
8								Gradient Boosting	0.7031812532			
9								Decision Tree	0.690123101			
10								LGBM	0.648571535			
11								XGBoost	0.5758444262			
12	N	Y	Y	Y	Y	Y	Y	Linear Regression	0.6813909559			
13								Random Forest	0.7287321269			
14								Gradient Boosting	0.6916678986			
15								Decision Tree	0.6463054976			
16								LGBM	0.6085169918			
17								XGBoost	0.5391589671			
18	N	Y	Y	Y	Y	Y	Y	GRU	0.4898177916	No separation of categories		
19								GRU	0.48432997	Separation of categories		
20								GRU	0.490060755	Reduce the last layer		
21								LSTM	0.4911511571			
22								GRU	0.490110159	add one dense layer		
23								CNN	0.4648432119	change drop out value	epoch 5	
24								GRU	0.4648432119	change drop out value	epoch 10	

Cont'd: Initial Experiments (Training Data - 70:30 Ratio)

25	CNN+GloVE	0.5361306971	num word = 3000	
26		0.5668518041	num word = 10000	
		0.5291644091	parameter tuning	
27	CNN	0.4906866039	no category split	
28	RNN	0.51		
29	RNN+Ridge	RNN	0.439481143	
		Ridge	0.4939428807	
		Aggreagate	0.4413393618	
30	CNN	0.4381465847		Hyper-parameter tuning
	RNN	0.44	add word count features	
			stopword removal	

Preprocessing, Feature Extraction, and Vectorization

Preprocessing

1. Text Preprocessing (lowercase, stop words-NLTK)
2. Filling Missing Values

Feature Extraction

1. Split main and sub categories
2. Text length features (item description)

Feature Vectorization

1. Label encoder (NLTK) for categorical data
2. Bag-of-Words (BOW) for textual data

Preprocessing & Feature Extraction

In [7]:
```
def split_cat(text):
    try:
        return text.split("/")
    except:
        return ("missing", "missing", "missing")
```

In [8]:
```
def handle_missing_inplace(dataset):
    dataset['general_cat'].fillna(value='missing', inplace=True)
    dataset['subcat_1'].fillna(value='missing', inplace=True)
    dataset['subcat_2'].fillna(value='missing', inplace=True)
    dataset['brand_name'].fillna(value='missing', inplace=True)
    dataset['item_description'].fillna(value='No description yet', inplace=True)
```

In [9]:
```
def cutting(dataset):
    pop_brand = dataset['brand_name'].value_counts().loc[lambda x: x.index != 'missing'].index
[:NUM_BRANDS]
    dataset.loc[~dataset['brand_name'].isin(pop_brand), 'brand_name'] = 'missing'
    pop_category1 = dataset['general_cat'].value_counts().loc[lambda x: x.index != 'missing'].i
ndex[:NUM_CATEGORIES]
    pop_category2 = dataset['subcat_1'].value_counts().loc[lambda x: x.index != 'missing'].inde
x[:NUM_CATEGORIES]
    pop_category3 = dataset['subcat_2'].value_counts().loc[lambda x: x.index != 'missing'].inde
x[:NUM_CATEGORIES]
    dataset.loc[~dataset['general_cat'].isin(pop_category1), 'general_cat'] = 'missing'
    dataset.loc[~dataset['subcat_1'].isin(pop_category2), 'subcat_1'] = 'missing'
    dataset.loc[~dataset['subcat_2'].isin(pop_category3), 'subcat_2'] = 'missing'
```

Preprocessing & Feature Extraction

In [11]:
```
# Define helpers for text normalization
stopwords = {x: 1 for x in stopwords.words('english')}
non_alphanums = re.compile(u'[^A-Za-z0-9]+')
```

In [12]:
```
# get name and description lengths
def wordCount(text):
    try:
        if text == 'No description yet':
            return 0
        else:
            text = text.lower()
            words = [w for w in text.split(" ")]
            return len(words)
    except:
        return 0
```

In [13]:
```
def normalize_text(text):
    return u" ".join(
        [x for x in [y for y in non_alphanums.sub(' ', text).lower().strip().split(" ")] \
         if len(x) > 1 and x not in stopwords])
```

In [14]:
```
def normalize_dataset_text(dataset):
    dataset['item_description'] = dataset['item_description'].apply(lambda x: normalize_text
(x))
    dataset['brand_name'] = dataset['brand_name'].apply(lambda x: normalize_text(x))
```

Preprocessing & Feature Extraction

In [15]:
```
def delete_unseen(dataset):
    dataset.loc[~dataset['brand_name'].isin(all_brand), 'brand_name'] = 'missing'
    dataset.loc[~dataset['general_cat'].isin(all_general_cat), 'general_cat'] = 'missing'
    dataset.loc[~dataset['subcat_1'].isin(all_subcat_1), 'subcat_1'] = 'missing'
    dataset.loc[~dataset['subcat_2'].isin(all_subcat_2), 'subcat_2'] = 'missing'
```

In [16]:
```
def text_length_feature(dataset, train = True):
    if train:
        dataset['desc_len'] = dataset['item_description'].apply(lambda x: wordCount(x))
        dataset['name_len'] = dataset['name'].apply(lambda x: wordCount(x))
        dataset[['desc_len', 'name_len']] = desc_normalizer.fit_transform(dataset[['desc_len',
'name_len']])
    else:
        dataset['desc_len'] = dataset['item_description'].apply(lambda x: wordCount(x))
        dataset['name_len'] = dataset['name'].apply(lambda x: wordCount(x))
        dataset[['desc_len', 'name_len']] = desc_normalizer.transform(dataset[['desc_len', 'nam
e_len']])
```

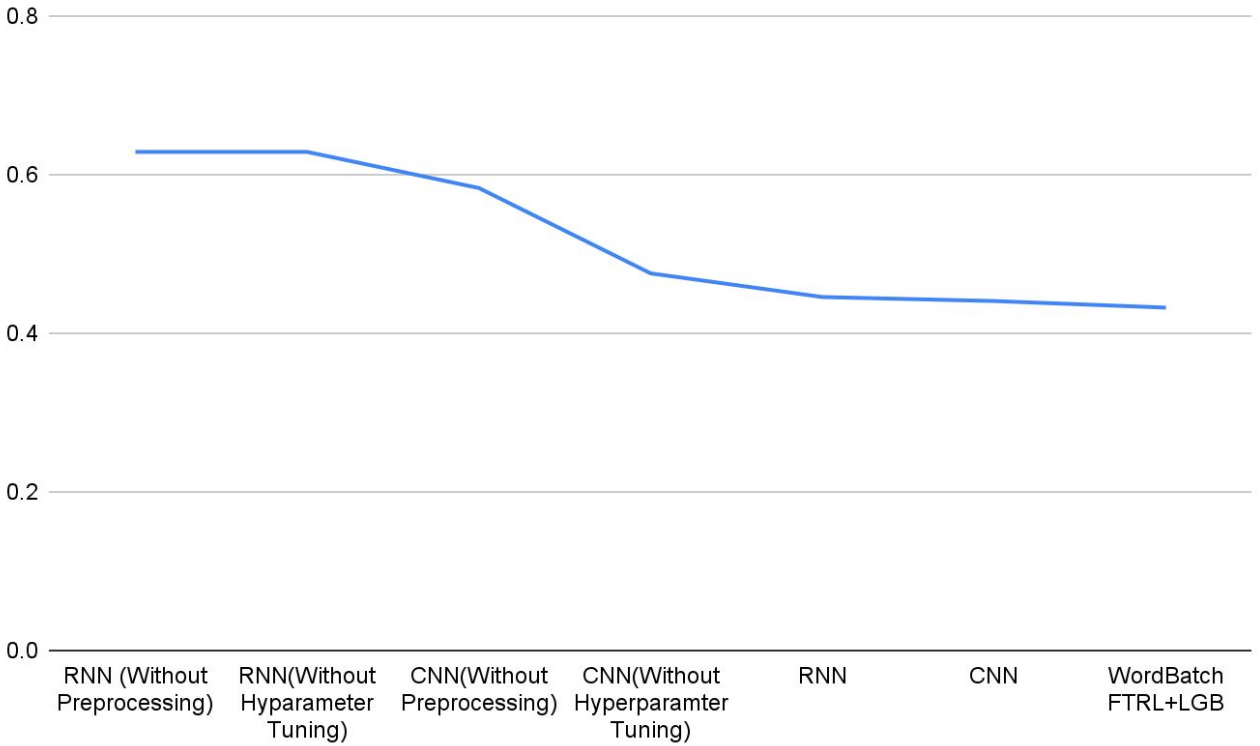
Prediction Models

1. CNN (Conv1D)
2. RNN
3. Wordbatch - FTRL (@anttip)

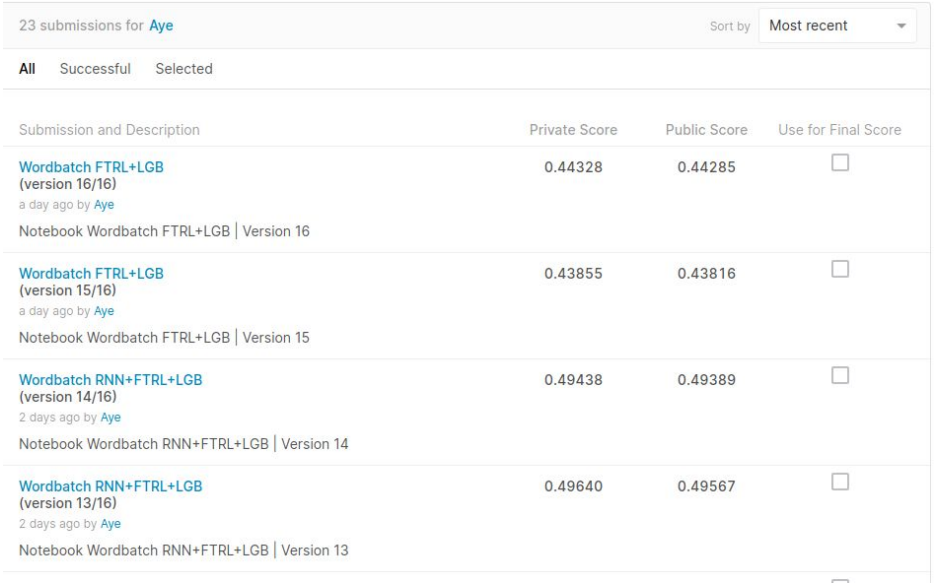
My Notebooks

1. https://www.kaggle.com/ayekhine/mercari-eda (EDA)
2. https://www.kaggle.com/ayekhine/neural-network-cnn-version-2 (CNN)
3. https://www.kaggle.com/ayekhine/mercari-price-rnn (RNN)
4. https://www.kaggle.com/ayekhine/wordbatch-ftrl-lgb (Word Batc

RMSLE Score


0.8
0.6
0.4
0.2
0.0
RNN (Without Preprocessing)
RNN(Without Hyparameter Tuning)
CNN(Without Preprocessing)
CNN(Without Hyperparamter Tuning)
RNN
CNN
WordBatch FTRL+LGB

My Submissions

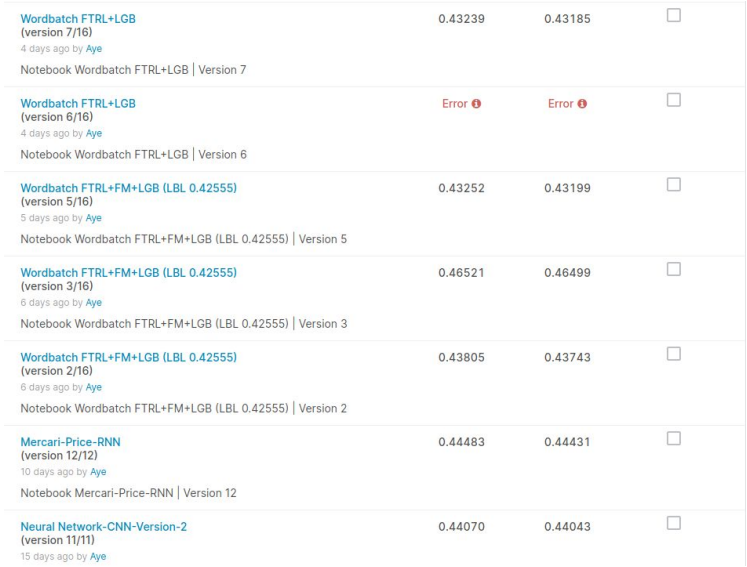


23 submissions for Aye

Sort by Most recent

All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
Wordbatch FTRL+LGB (version 16/16) a day ago by Aye Notebook Wordbatch FTRL+LGB | Version 16	0.44328	0.44285	☐
Wordbatch FTRL+LGB (version 15/16) a day ago by Aye Notebook Wordbatch FTRL+LGB | Version 15	0.43855	0.43816	☐
Wordbatch RNN+FTRL+LGB (version 14/16) 2 days ago by Aye Notebook Wordbatch RNN+FTRL+LGB | Version 14	0.49438	0.49389	☐
Wordbatch RNN+FTRL+LGB (version 13/16) 2 days ago by Aye Notebook Wordbatch RNN+FTRL+LGB | Version 13	0.49640	0.49567	☐



Wordbatch FTRL+LGB (version 7/16) 4 days ago by Aye Notebook Wordbatch FTRL+LGB | Version 7	0.43239	0.43185	☐
Wordbatch FTRL+LGB (version 6/16) 4 days ago by Aye Notebook Wordbatch FTRL+LGB | Version 6	Error	Error	☐
Wordbatch FTRL+FM+LGB (LBL 0.42555) (version 5/16) 5 days ago by Aye Notebook Wordbatch FTRL+FM+LGB (LBL 0.42555) | Version 5	0.43252	0.43199	☐
Wordbatch FTRL+FM+LGB (LBL 0.42555) (version 3/16) 6 days ago by Aye Notebook Wordbatch FTRL+FM+LGB (LBL 0.42555) | Version 3	0.46521	0.46499	☐
Wordbatch FTRL+FM+LGB (LBL 0.42555) (version 2/16) 6 days ago by Aye Notebook Wordbatch FTRL+FM+LGB (LBL 0.42555) | Version 2	0.43805	0.43743	☐
Mercari-Price-RNN (version 12/12) 10 days ago by Aye Notebook Mercari-Price-RNN | Version 12	0.44483	0.44431	☐
Neural Network-CNN-Version-2 (version 11/11) 15 days ago by Aye	0.44070	0.44043	☐

Limitations

Only 60 minutes to train and predict

System specification: 16GB RAM. 1GB disk, 4 Cores

Thank You
Kaggle, Julia and team, Qingchen Wang


kaggle