In [0]:

```
# importing libraries
import os
import re
import time
import math
import nltk

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('whitegrid')


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDRegressor, Ridge
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split


from scipy.sparse import hstack
from scipy import sparse
from scipy.sparse import csr_matrix

from tqdm import tqdm
from nltk.corpus import stopwords
from prettytable import PrettyTable
from lightgbm import LGBMRegressor

import xgboost as xgb


import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import RMSprop, Adam
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: Fut
ureWarning: pandas.util.testing is deprecated. Use the functions in the publ
ic API at pandas.testing instead.
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: Fut
ureWarning: The sklearn.metrics.classification module is  deprecated in vers
ion 0.22 and will be removed in version 0.24. The corresponding classes / fu
nctions should instead be imported from sklearn.metrics. Anything that canno
t be imported from sklearn.metrics is now part of the private API.
  warnings.warn(message, FutureWarning)
```

In [0]:

```
# loading train
train = pd.read_csv('/content/drive/My Drive/Mercari/data/train.tsv', sep='\t')
train.shape
```

Out[2]:

```
(1482535, 8)
```

In [0]:

```
train.head(5)
```

Out[3]:

	train_id	name	item_condition_id	category_name	brand_name	price	shippin
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	NaN	35.0	
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	

In [0]:

```
train.dtypes
```

Out[4]:

```
train_id               int64
name                  object
item_condition_id      int64
category_name         object
brand_name            object
price                float64
shipping               int64
item_description      object
dtype: object
```

In [0]:

```
train.describe()
```

Out[5]:

	train_id	item_condition_id	price	shipping
count	1.482535e+06	1.482535e+06	1.482535e+06	1.482535e+06
mean	7.412670e+05	1.907380e+00	2.673752e+01	4.472744e-01
std	4.279711e+05	9.031586e-01	3.858607e+01	4.972124e-01
min	0.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00
25%	3.706335e+05	1.000000e+00	1.000000e+01	0.000000e+00
50%	7.412670e+05	2.000000e+00	1.700000e+01	0.000000e+00
75%	1.111900e+06	3.000000e+00	2.900000e+01	1.000000e+00
max	1.482534e+06	5.000000e+00	2.009000e+03	1.000000e+00

In [0]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1482535 entries, 0 to 1482534
Data columns (total 8 columns):
 #   Column             Non-Null Count    Dtype  
---  ------             --------------    -----  
 0   train_id           1482535 non-null  int64  
 1   name               1482535 non-null  object 
 2   item_condition_id  1482535 non-null  int64  
 3   category_name      1476208 non-null  object 
 4   brand_name         849853 non-null   object 
 5   price              1482535 non-null  float64
 6   shipping           1482535 non-null  int64  
 7   item_description   1482531 non-null  object 
dtypes: float64(1), int64(3), object(4)
memory usage: 90.5+ MB
```

In [0]:

```

```

In [0]:

```
# Loading test
test = pd.read_csv('/content/drive/My Drive/Mercari/data/test.tsv', sep='\t')
test.shape
```

Out[7]:

```
(693359, 7)
```

In [0]:

```
test.head()
```

Out[8]:

	test_id	name	item_condition_id	category_name	brand_name	shipping	item_de
0	0	Breast cancer "I fight like a girl" ring	1	Women/Jewelry/Rings	NaN	1	
1	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers	1	Other/Office supplies/Shipping Supplies	NaN	1	25 7.5" Bubb
2	2	Coach bag	1	Vintage & Collectibles/Bags and Purses/Handbag	Coach	1	Brand n bag. E [rm] at
3	3	Floral Kimono	2	Women/Sweaters/Cardigan	NaN	0	-flora ne lightw
4	4	Life after Death	3	Other/Books/Religion & Spirituality	NaN	1	Redisco after the

In [0]:

```
test.dtypes
```

Out[9]:

```
test_id               int64
name                 object
item_condition_id     int64
category_name        object
brand_name           object
shipping              int64
item_description     object
dtype: object
```

In [0]:

```
test.describe()
```

Out[10]:

	test_id	item_condition_id	shipping
count	693359.000000	693359.000000	693359.000000
mean	346679.000000	1.906102	0.447719
std	200155.646984	0.903378	0.497260
min	0.000000	1.000000	0.000000
25%	173339.500000	1.000000	0.000000
50%	346679.000000	2.000000	0.000000
75%	520018.500000	3.000000	1.000000
max	693358.000000	5.000000	1.000000

In [0]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 693359 entries, 0 to 693358
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   test_id            693359 non-null  int64 
 1   name               693359 non-null  object
 2   item_condition_id  693359 non-null  int64 
 3   category_name      690301 non-null  object
 4   brand_name         397834 non-null  object
 5   shipping           693359 non-null  int64 
 6   item_description   693359 non-null  object
dtypes: int64(3), object(4)
memory usage: 37.0+ MB
```

In [0]:

```

```

Checking for Null Values

In [0]:

```
train.isnull().any()
```

Out[12]:

```
train_id             False
name                 False
item_condition_id    False
category_name         True
brand_name            True
price                False
shipping             False
item_description      True
dtype: bool
```

In [0]:

```
test.isnull().any()
```

Out[13]:

```
test_id              False
name                 False
item_condition_id    False
category_name         True
brand_name            True
shipping             False
item_description     False
dtype: bool
```

In [0]:

```

```

Replacing Null Values

In [0]:

```
# Train

train['category_name'].fillna("Others", inplace=True)
train['brand_name'].fillna("Unknown", inplace=True)
train['item_description'].fillna("No description", inplace=True)

# Checking for Null values
train.isnull().any()
```

Out[14]:

```
train_id             False
name                 False
item_condition_id    False
category_name        False
brand_name           False
price                False
shipping             False
item_description     False
dtype: bool
```

In [0]:

```
# Test

test['category_name'].fillna("Others", inplace=True)
test['brand_name'].fillna("Unknown", inplace=True)
#test['item_description'].fillna("No description", inplace=True)

# Checking for Null values
test.isnull().any()
```

Out[15]:

```
test_id              False
name                 False
item_condition_id    False
category_name        False
brand_name           False
shipping             False
item_description     False
dtype: bool
```

In [0]:

```

```

Data Analysis

1. Price

In [0]:

```
train['price'].describe()
```

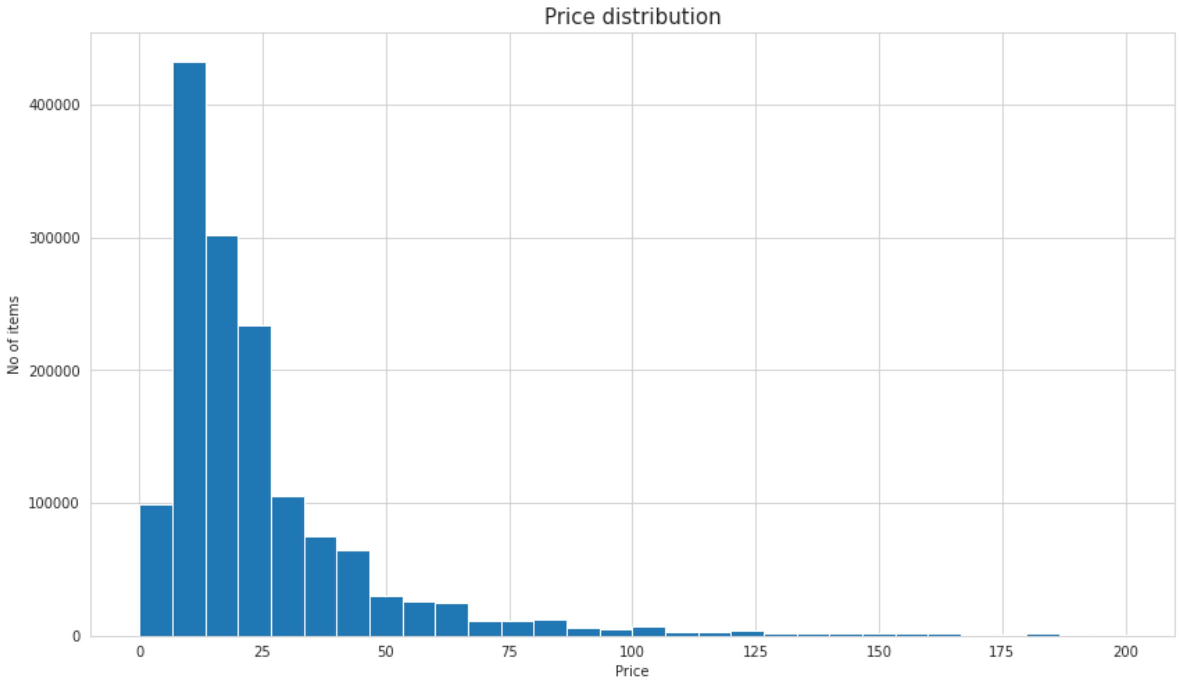
Out[16]:

```
count    1.482535e+06
mean     2.673752e+01
std      3.858607e+01
min      0.000000e+00
25%      1.000000e+01
50%      1.700000e+01
75%      2.900000e+01
max      2.009000e+03
Name: price, dtype: float64
```

In [0]:

```
#Histogram
fig, ax = plt.subplots(figsize=(14,8))
ax.hist(train['price'], bins=30, range=[0,200], label="Price")
plt.title('Price distribution', fontsize=15)
ax.set_xlabel('Price')
ax.set_ylabel('No of items')

plt.show()
```



We can clearly see that the most of the products price is in between 15 and 30

In [0]:

```
#We will add log(price) as a column in our train data
train["Log_Price"] = np.log1p(train["price"])
train.head()
```

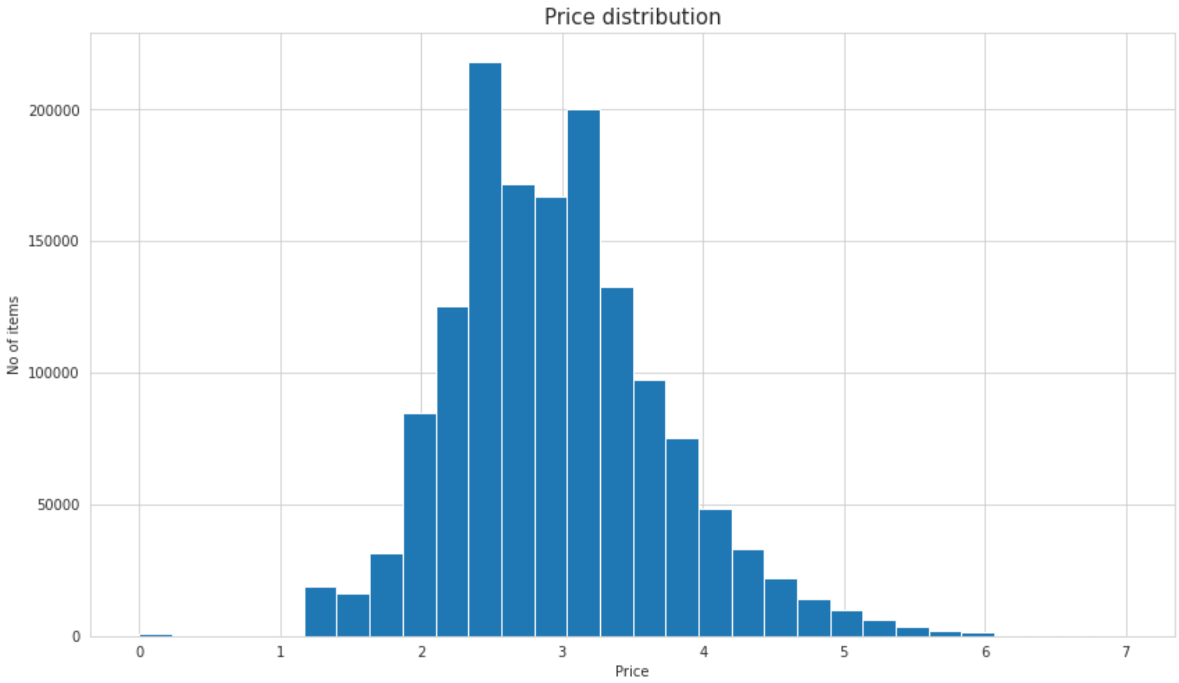
Out[18]:

	train_id	name	item_condition_id	category_name	brand_name	price	shippin
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	Unknown	10.0	
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	Unknown	35.0	
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	Unknown	44.0	

In [0]:

```
#Histogram
fig, ax = plt.subplots(figsize=(14,8))
ax.hist(train['Log_Price'], bins=30, range=[0,7], label="Price")
plt.title('Price distribution', fontsize=15)
ax.set_xlabel('Price')
ax.set_ylabel('No of items')

plt.show()
```



2. Shipping

In [0]:

```
train['shipping'].value_counts()
```

Out[20]:

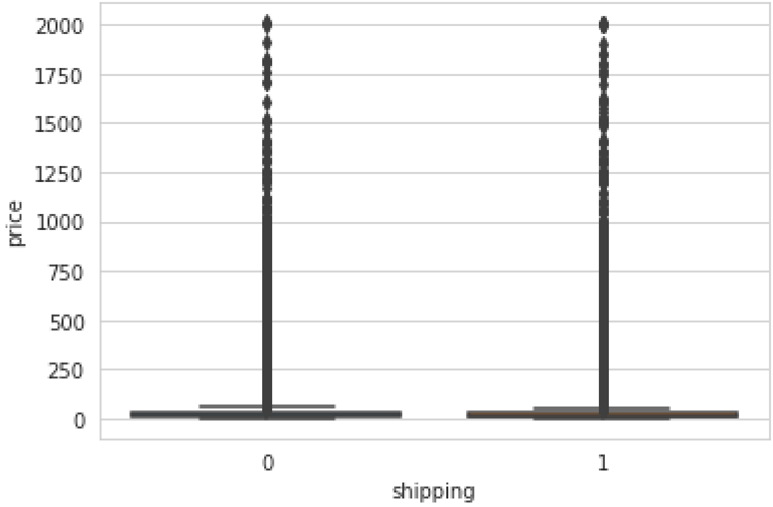
```
0    819435
1    663100
Name: shipping, dtype: int64
```

In [0]:

```
# Shipping vs Price
sns.boxplot(x=train['shipping'] ,y=train['price'],orient='v')
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f277744bb70>
```

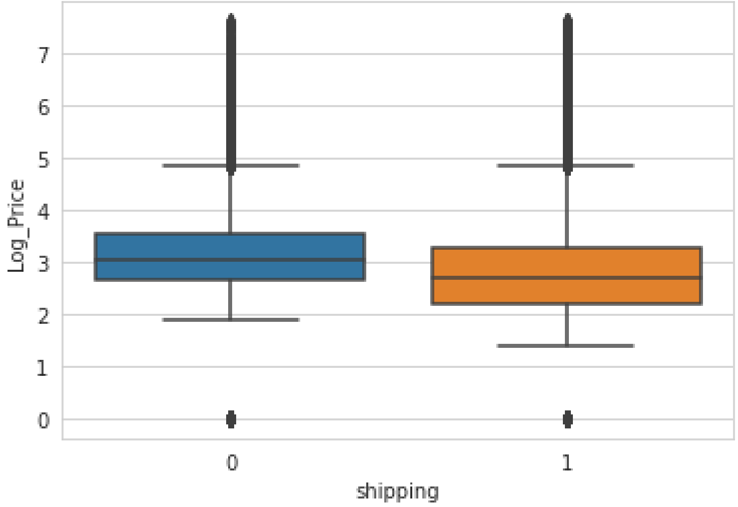


In [0]:

```
# Shipping vs Log_Price
sns.boxplot(x=train['shipping'] ,y=train['Log_Price'],orient='v')
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2777983a90>
```



3. item_condition_id

In [0]:

```
train['item_condition_id'].value_counts()
```

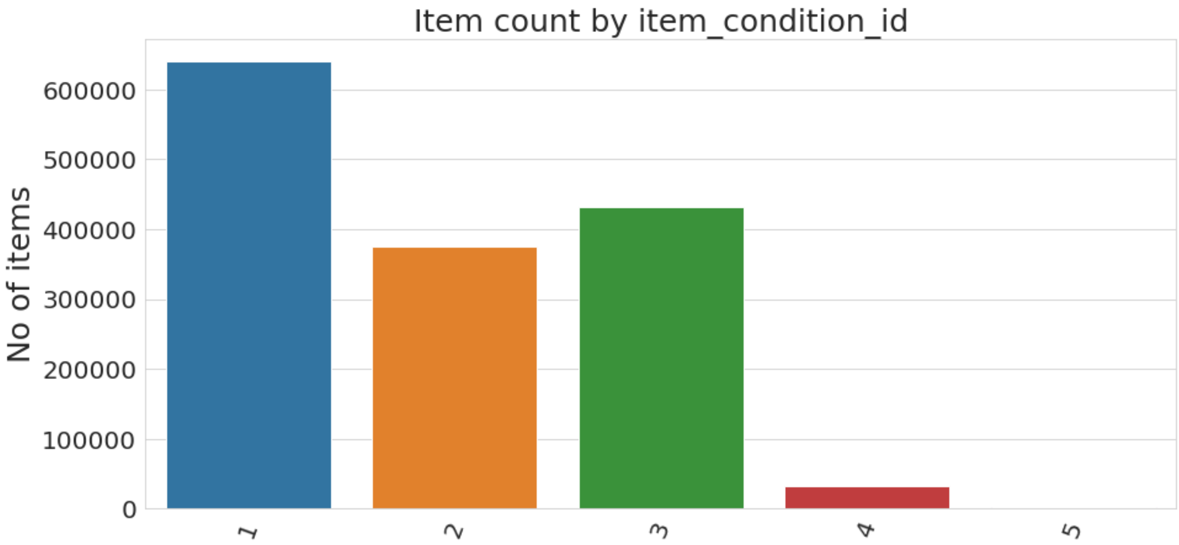
Out[23]:

```
1    640549
3    432161
2    375479
4     31962
5      2384
Name: item_condition_id, dtype: int64
```

In [0]:

```
fig, ax = plt.subplots(figsize=(15,7))
sns.countplot(x='item_condition_id', data=train, ax=ax)
plt.title('Item count by item_condition_id',fontsize=25)
plt.ylabel('No of items',fontsize=25)
plt.xlabel('')
plt.xticks(rotation=70,fontsize=20)
plt.yticks(fontsize=20)

plt.show()
```



In [0]:

```

```

4. category_name

In [0]:

```
len(train['category_name'].unique())
```

Out[25]:

```
1288
```

In [0]:

```
train['category_name'].value_counts()[:10]
```

Out[26]:

```
Women/Athletic Apparel/Pants, Tights, Leggings                 60177
Women/Tops & Blouses/T-Shirts                                  46380
Beauty/Makeup/Face                                             34335
Beauty/Makeup/Lips                                             29910
Electronics/Video Games & Consoles/Games                       26557
Beauty/Makeup/Eyes                                             25215
Electronics/Cell Phones & Accessories/Cases, Covers & Skins    24676
Women/Underwear/Bras                                           21274
Women/Tops & Blouses/Tank, Cami                                20284
Women/Tops & Blouses/Blouse                                    20284
Name: category_name, dtype: int64
```

In [0]:

```
train.head()
```

Out[27]:

	train_id	name	item_condition_id	category_name	brand_name	price	shippin
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	Unknown	10.0	
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	Unknown	35.0	
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	Unknown	44.0	

In [0]:

```
# Train
# Splitting categories into 3 features: main_cat, sub_cat_1, sub_cat_2

#main_cat, sub_cat_1, sub_cat_2 = train['category_name'].split('/')

main_cat = []
sub_cat_1 = []
sub_cat_2 = []

for row in tqdm(train['category_name']):
    try:
        main, sub_1, sub_2 = row.split('/')
        main_cat.append(main)
        sub_cat_1.append(sub_1)
        sub_cat_2.append(sub_2)
    except:
        main_cat.append("Others")
        sub_cat_1.append("Others")
        sub_cat_2.append("Others")
```

```
100%|██████████| 1482535/1482535 [00:01<00:00, 922202.01it/s]
```

In [0]:

```
print(len(main_cat))
print(len(sub_cat_1))
print(len(sub_cat_2))
```

```
1482535
1482535
1482535
```

In [0]:

```
# Adding these new features to the train dataframe
train['main_cat'] = main_cat
train['sub_cat_1'] = sub_cat_1
train['sub_cat_2'] = sub_cat_2

train.head()
```

Out[30]:

	train_id	name	item_condition_id	category_name	brand_name	price	shippin
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	Unknown	10.0	
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	Unknown	35.0	
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	Unknown	44.0	

In [0]:

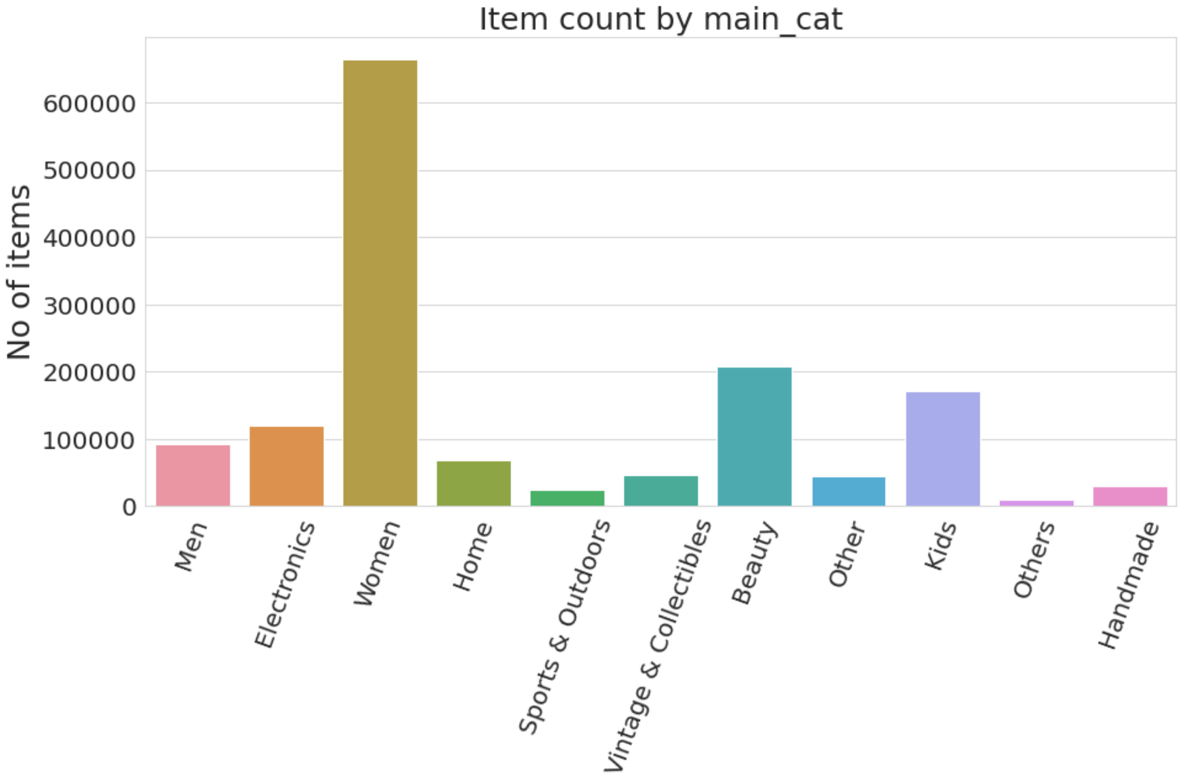
```
print("No of unique values in main_cat: ", len(train['main_cat'].unique()))
print("No of unique values in sub_cat_1: ", len(train['sub_cat_1'].unique()))
print("No of unique values in sub_cat_2: ", len(train['sub_cat_1'].unique()))
```

```
No of unique values in main_cat:  11
No of unique values in sub_cat_1:  113
No of unique values in sub_cat_2:  113
```

In [0]:

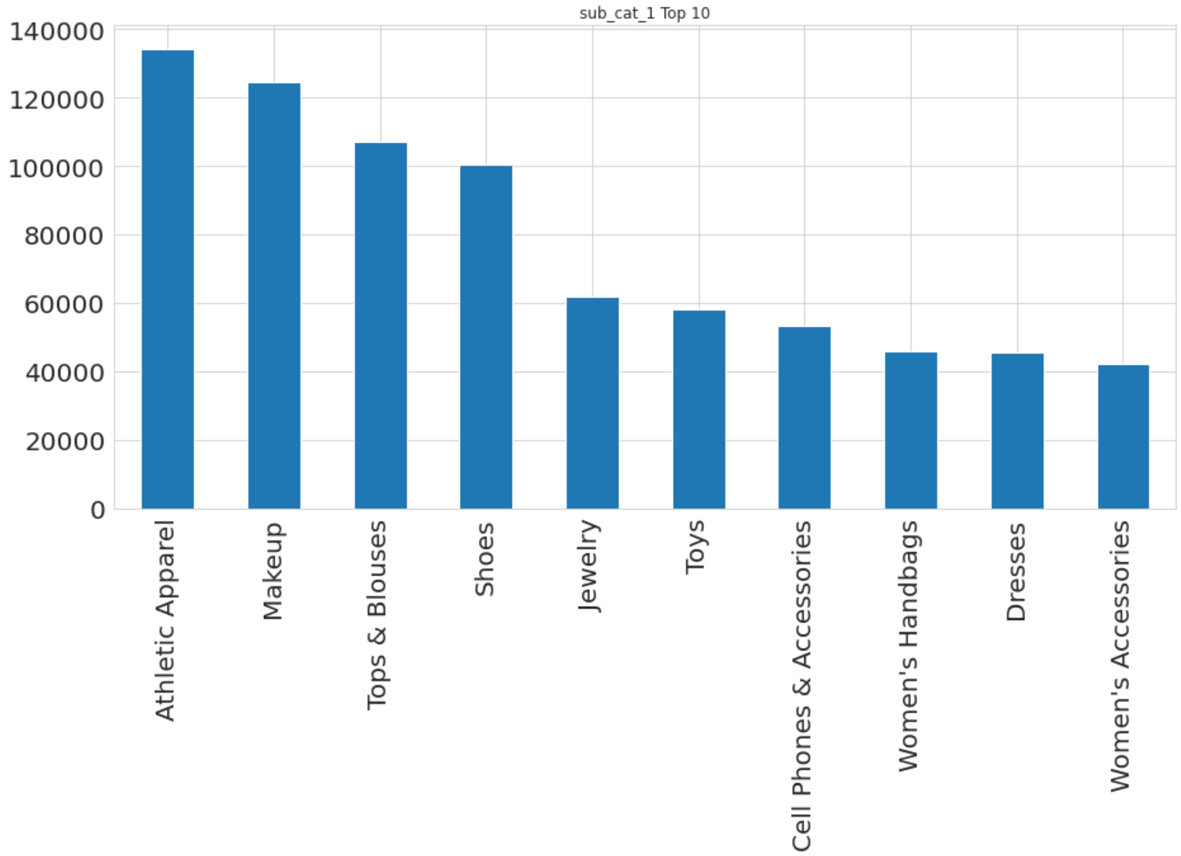
```
fig, ax = plt.subplots(figsize=(15,7))
sns.countplot(x='main_cat', data=train, ax=ax)
plt.title('Item count by main_cat',fontsize=25)
plt.ylabel('No of items',fontsize=25)
plt.xlabel('')
plt.xticks(rotation=70,fontsize=20)
plt.yticks(fontsize=20)

plt.show()
```



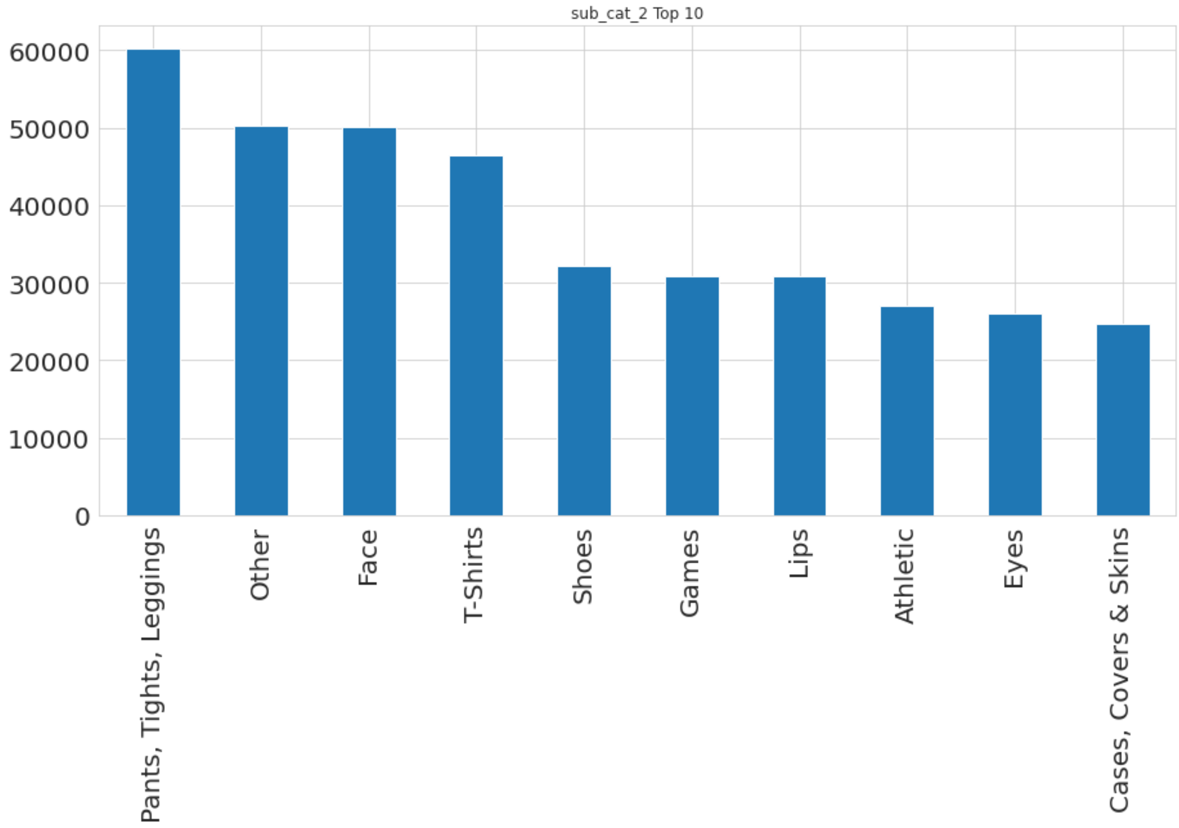
In [0]:

```
train['sub_cat_1'].value_counts()[:10].plot(kind='bar', figsize = (15,7), title="sub_cat_1
plt.show()
```



In [0]:

```
train['sub_cat_2'].value_counts()[:10].plot(kind='bar', figsize = (15, 7), title="sub_cat_2
plt.show()
```



In [0]:

```
# Test
# Splitting categories into 3 features: main_cat, sub_cat_1, sub_cat_2

#main_cat, sub_cat_1, sub_cat_2 = train['category_name'].split('/')

main_cat = []
sub_cat_1 = []
sub_cat_2 = []

for row in tqdm(test['category_name']):
    try:
        main, sub_1, sub_2 = row.split('/')
        main_cat.append(main)
        sub_cat_1.append(sub_1)
        sub_cat_2.append(sub_2)
    except:
        main_cat.append("Others")
        sub_cat_1.append("Others")
        sub_cat_2.append("Others")

print(len(main_cat))
print(len(sub_cat_1))
print(len(sub_cat_2))
```

```
100%|██████████| 693359/693359 [00:00<00:00, 842217.48it/s]

693359
693359
693359
```

In [0]:

```
# Adding these new features to the test dataframe
test['main_cat'] = main_cat
test['sub_cat_1'] = sub_cat_1
test['sub_cat_2'] = sub_cat_2

test.head()
```

Out[36]:

	test_id	name	item_condition_id	category_name	brand_name	shipping	item_de
0	0	Breast cancer "I fight like a girl" ring	1	Women/Jewelry/Rings	Unknown	1	
1	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers	1	Other/Office supplies/Shipping Supplies	Unknown	1	25 7.5" Bubb
2	2	Coach bag	1	Vintage & Collectibles/Bags and Purses/Handbag	Coach	1	Brand n bag. E [rm] at
3	3	Floral Kimono	2	Women/Sweaters/Cardigan	Unknown	0	-flora ne lightw
4	4	Life after Death	3	Other/Books/Religion & Spirituality	Unknown	1	Redisco after the

In [0]:

```
print("No of unique values in main_cat: ", len(test['main_cat'].unique()))
print("No of unique values in sub_cat_1: ", len(test['sub_cat_1'].unique()))
print("No of unique values in sub_cat_2: ", len(test['sub_cat_1'].unique()))
```

```
No of unique values in main_cat:  11
No of unique values in sub_cat_1:  113
No of unique values in sub_cat_2:  113
```

In [0]:

```

```

5. item_description

In [0]:

```
# description is text column. We can use the length of each row
```

In [0]:

```
# https://stackoverflow.com/questions/37335598/how-to-get-the-length-of-a-cell-value-in-pan
desc_length = train['item_description'].apply(len)
print(len(desc_length))
```

```
1482535
```

In [0]:

```
train['desc_length'] = desc_length
train.head()
```

Out[40]:

	train_id	name	item_condition_id	category_name	brand_name	price	shippin
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	Unknown	10.0	
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	Unknown	35.0	
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	Unknown	44.0	

In [0]:

```

```

6. brand_name

In [0]:

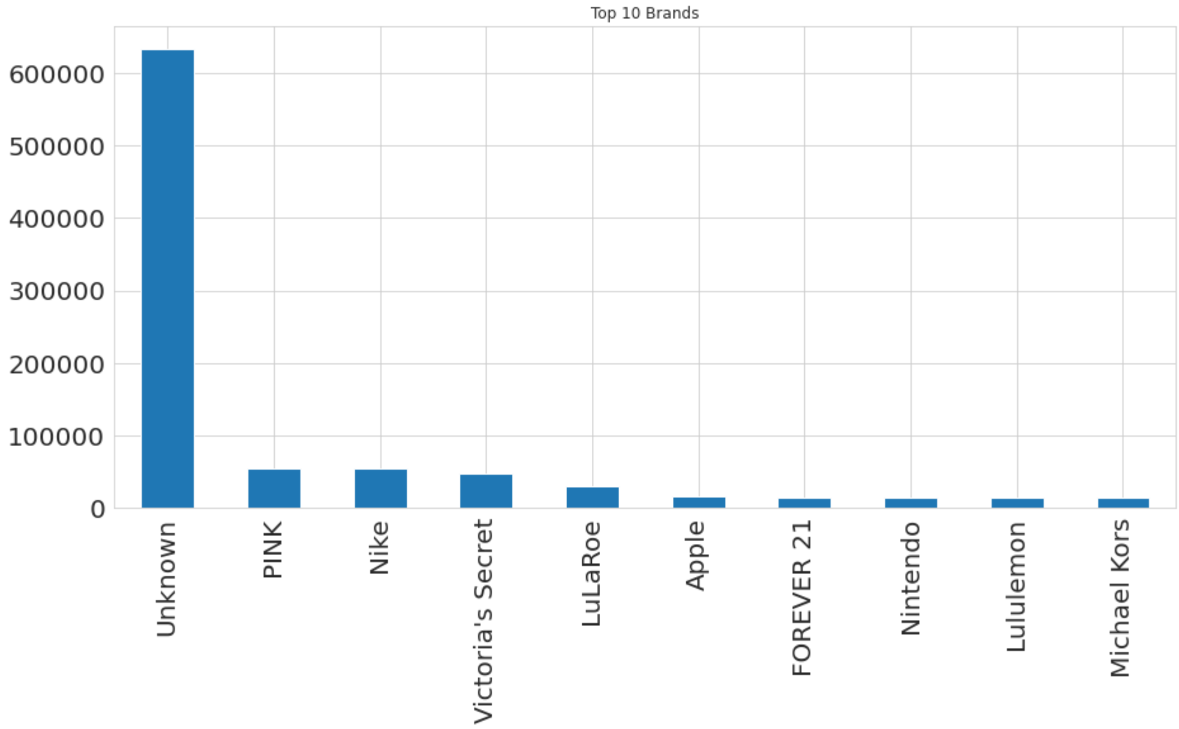
```
# Unique brands
len(train['brand_name'].unique())
```

Out[41]:

```
4810
```

In [0]:

```
train['brand_name'].value_counts()[:10].plot(kind='bar', figsize = (15, 7), title="Top 10 B
plt.show()
```



In [0]:

```
```

Data Preprocessing

In [0]:

```
# https://stackoverflow.com/a/47091490/4084039

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [0]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'c
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'dc
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:

```
```

Train: item_description

In [0]:

```
# Train['item_description']

preprocessed_item_description = []
# tqdm is for printing the status bar
for sentance in tqdm(train['item_description'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_item_description.append(sent.lower().strip())
```

```
100%|██████████| 1482535/1482535 [01:55<00:00, 12796.05it/s]
```

In [0]:

```
train['item_description'] = preprocessed_item_description
```

Test: item_description

In [0]:

```
# test['item_description']

preprocessed_item_description = []
# tqdm is for printing the status bar
for sentance in tqdm(test['item_description'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_item_description.append(sent.lower().strip())
```

```
100%|██████████| 693359/693359 [00:52<00:00, 13167.45it/s]
```

In [0]:

```
test['item_description'] = preprocessed_item_description
```

Train: title

In [0]:

```
# Train['name']

preprocessed_name = []
# tqdm is for printing the status bar
for sentance in tqdm(train['name'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_name.append(sent.lower().strip())
```

```
100%|██████████| 1482535/1482535 [00:34<00:00, 42389.22it/s]
```

In [0]:

```
train['name'] = preprocessed_name
```

Test: title

In [0]:

```
# Test['name']

preprocessed_name = []
# tqdm is for printing the status bar
for sentance in tqdm(test['name'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_name.append(sent.lower().strip())
```

```
100%|██████████| 693359/693359 [00:16<00:00, 42427.87it/s]
```

In [0]:

```
test['name'] = preprocessed_name
```

In [0]:

```
train.head()
```

Out[53]:

	train_id	name	item_condition_id	category_name	brand_name	price	shipping
0	0	mlb cincinnati reds shirt size xl	3	Men/Tops/T-shirts	Unknown	10.0	1
1	1	razer blackwidow chroma keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0
2	2	ava viv blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1
3	3	leather horse statues	1	Home/Home Décor/Home Décor Accents	Unknown	35.0	1
4	4	24k gold plated rose	1	Women/Jewelry/Necklaces	Unknown	44.0	0

In [0]:

```
test.head()
```

Out[54]:

	test_id	name	item_condition_id	category_name	brand_name	shipping	item_des
0	0	breast cancer fight like girl ring	1	Women/Jewelry/Rings	Unknown	1	
1	1	25 pcs new 7 5 x12 kraft bubble mailers	1	Other/Office supplies/Shipping Supplies	Unknown	1	25 pcs x12 kra mailers
2	2	coach bag	1	Vintage & Collectibles/Bags and Purses/Handbag	Coach	1	brand ne bag b coa
3	3	floral kimono	2	Women/Sweaters/Cardigan	Unknown	0	flora ne lightweigh
4	4	life death	3	Other/Books/Religion & Spirituality	Unknown	1	rediscov loss l tony c

In [0]:

```
train.shape
```

Out[55]:

```
(1482535, 13)
```

In [0]:

```
test.shape
```

Out[56]:

```
(693359, 10)
```

Train Validation Split

In [0]:

```
# Storing Log_Price in y_val
y_val = train['Log_Price']

# Dropping some columns
train.drop(['price', 'train_id','Log_Price'], axis=1, inplace=True)  # desc_length
```

In [0]:

```
train.drop('desc_length', axis=1, inplace=True)
```

In [0]:

```
train.head()
```

Out[59]:

	name	item_condition_id	category_name	brand_name	shipping	item_descripti
0	mlb cincinnati reds shirt size xl	3	Men/Tops/T-shirts	Unknown	1	no description
1	razer blackwidow chroma keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	0	keyboard gr condition wo like came box
2	ava viv blouse	1	Women/Tops & Blouses/Blouse	Target	1	adorable top h lace key h back pale pin
3	leather horse statues	1	Home/Home Décor/Home Décor Accents	Unknown	1	new tags leatl horses retail stand foot
4	24k gold plated rose	1	Women/Jewelry/Necklaces	Unknown	0	compl certific authentic

In [0]:

```
# Stroing the test_id into a seperate vaariable
test_ids = test['test_id'].values.astype(np.int32)

test.drop("test_id", axis=1, inplace=True)
```

In [0]:

```
test.head()
```

Out[61]:

	name	item_condition_id	category_name	brand_name	shipping	item_description
0	breast cancer fight like girl ring	1	Women/Jewelry/Rings	Unknown	1	size 7
1	25 pcs new 7 5 x12 kraft bubble mailers	1	Other/Office supplies/Shipping Supplies	Unknown	1	25 pcs new 7 5 x12 kraft bubble mailers lined ...
2	coach bag	1	Vintage & Collectibles/Bags and Purses/Handbag	Coach	1	brand new coach bag bought rm coach outlet
3	floral kimono	2	Women/Sweaters/Cardigan	Unknown	0	floral kimono never worn lightweight perfect h...
4	life death	3	Other/Books/Religion & Spirituality	Unknown	1	rediscovering life loss loved one tony cooke p...

In [0]:

```
print(train.shape)
print(test.shape)
```

```
(1482535, 9)
(693359, 9)
```

In [0]:

```

```

In [0]:

```
# Train Validation Split

x_train, x_val, y_train, y_val = train_test_split(train, y_val, test_size=0.2, random_state
```

In [0]:

```
print(x_train.shape, y_train.shape)
print(x_val.shape, y_val.shape)
```

```
(1186028, 9) (1186028,)
(296507, 9) (296507,)
```

In [0]:

Vectorizing for Text and Categorical Features

1. Name

In [0]:

```
# Name
vectorizer = CountVectorizer(max_features=500, min_df=5)

# Train
vectorizer.fit(x_train['name'].values)
train_name = vectorizer.transform(x_train['name'].values)

# Validation
val_name = vectorizer.transform(x_val['name'].values)

# Test
test_name = vectorizer.transform(test['name'].values)


print("Shapes:")
print("Train: ", train_name.shape)
print("Val: ", val_name.shape)
print("Test: ", test_name.shape)
```

```
Shapes:
Train:  (1186028, 500)
Val:  (296507, 500)
Test:  (693359, 500)
```

2. item_description

In [0]:

```
# item_description
vectorizer = CountVectorizer(max_features=5000, min_df=5)

# Train
vectorizer.fit(x_train['item_description'].values)
train_item_desc = vectorizer.transform(x_train['item_description'].values)

# Validation
val_item_desc = vectorizer.transform(x_val['item_description'].values)

# Test
test_item_desc = vectorizer.transform(test['item_description'].values)


print("Shapes:")
print("Train: ", train_item_desc.shape)
print("Val: ", val_item_desc.shape)
print("Test: ", test_item_desc.shape)
```

```
Shapes:
Train:  (1186028, 5000)
Val:  (296507, 5000)
Test:  (693359, 5000)
```

3. brand_name

In [0]:

```
# brand_name
vectorizer = CountVectorizer()

# Train
vectorizer.fit(x_train['brand_name'].values)
train_brand_name = vectorizer.transform(x_train['brand_name'].values)

# Validation
val_brand_name = vectorizer.transform(x_val['brand_name'].values)

# Test
test_brand_name = vectorizer.transform(test['brand_name'].values)


print("Shapes:")
print("Train: ", train_brand_name.shape)
print("Val: ", val_brand_name.shape)
print("Test: ", test_brand_name.shape)
```

```
Shapes:
Train:  (1186028, 4822)
Val:  (296507, 4822)
Test:  (693359, 4822)
```

4. main_cat

In [0]:

```
# main_cat
vectorizer = CountVectorizer()

# Train
vectorizer.fit(x_train['main_cat'].values)
train_main_cat = vectorizer.transform(x_train['main_cat'].values)

# Validation
val_main_cat = vectorizer.transform(x_val['main_cat'].values)

# Test
test_main_cat = vectorizer.transform(test['main_cat'].values)


print("Shapes:")
print("Train: ", train_main_cat.shape)
print("Val: ", val_main_cat.shape)
print("Test: ", test_main_cat.shape)
```

```
Shapes:
Train:  (1186028, 13)
Val:  (296507, 13)
Test:  (693359, 13)
```

5. sub_cat_1

In [0]:

```
# sub_cat_1
vectorizer = CountVectorizer()

# Train
vectorizer.fit(x_train['sub_cat_1'].values)
train_sub_cat_1 = vectorizer.transform(x_train['sub_cat_1'].values)

# Validation
val_sub_cat_1 = vectorizer.transform(x_val['sub_cat_1'].values)

# Test
test_sub_cat_1 = vectorizer.transform(test['sub_cat_1'].values)


print("Shapes:")
print("Train: ", train_sub_cat_1.shape)
print("Val: ", val_sub_cat_1.shape)
print("Test: ", test_sub_cat_1.shape)
```

```
Shapes:
Train:  (1186028, 141)
Val:  (296507, 141)
Test:  (693359, 141)
```

6. sub_cat_2

In [0]:

```
# sub_cat_2
vectorizer = CountVectorizer()

# Train
vectorizer.fit(x_train['sub_cat_2'].values)
train_sub_cat_2 = vectorizer.transform(x_train['sub_cat_2'].values)

# Validation
val_sub_cat_2 = vectorizer.transform(x_val['sub_cat_2'].values)

# Test
test_sub_cat_2 = vectorizer.transform(test['sub_cat_2'].values)


print("Shapes:")
print("Train: ", train_sub_cat_2.shape)
print("Val: ", val_sub_cat_2.shape)
print("Test: ", test_sub_cat_2.shape)
```

```
Shapes:
Train:  (1186028, 950)
Val:  (296507, 950)
Test:  (693359, 950)
```

7. item_condition_id

In [0]:

```
# item_condition_id
encoder = OneHotEncoder()

# Train
encoder.fit(x_train['item_condition_id'].values.reshape(-1, 1))
train_item_condition_id = encoder.transform(x_train['item_condition_id'].values.reshape(-1,

# Validation
val_item_condition_id = encoder.transform(x_val['item_condition_id'].values.reshape(-1, 1))

# Test
test_item_condition_id = encoder.transform(test['item_condition_id'].values.reshape(-1, 1))


print("Shapes:")
print("Train: ", train_item_condition_id.shape)
print("Val: ", val_item_condition_id.shape)
print("Test: ", test_item_condition_id.shape)
```

```
Shapes:
Train:  (1186028, 5)
Val:  (296507, 5)
Test:  (693359, 5)
```

8. shipping

In [0]:

```
# shipping
encoder = OneHotEncoder()

# Train
encoder.fit(x_train['shipping'].values.reshape(-1, 1))
train_shipping = encoder.transform(x_train['shipping'].values.reshape(-1, 1))

# Validation
val_shipping = encoder.transform(x_val['shipping'].values.reshape(-1, 1))

# Test
test_shipping = encoder.transform(test['shipping'].values.reshape(-1, 1))


print("Shapes:")
print("Train: ", train_shipping.shape)
print("Val: ", val_shipping.shape)
print("Test: ", test_shipping.shape)
```

```
Shapes:
Train:  (1186028, 2)
Val:  (296507, 2)
Test:  (693359, 2)
```

Stacking Data

In [0]:

```
x_train.head()
```

Out[74]:

	name	item_condition_id	category_name	brand_name	shipping	item_
1416089	lularoe kids l xl leggings	3	Kids/Boys (4+)/Bottoms	Unknown	1	wo
1423955	bundle 5 display mannequins	1	Other/Other/Other	Unknown	0	
403867	living proof perfect hair day dry shampo	1	Beauty/Hair Care/Styling Products	Unknown	0	listir bottles
701974	palazzo pants	2	Women/Pants/Casual Pants	Unknown	0	like ne pa
1124330	reserved ms jas pink boyshorts large	1	Women/Underwear/Panties	PINK	1	new ta se

In [0]:

```
# Stacking Train data
x_train_final = hstack((train_name, train_item_desc, train_brand_name, train_main_cat, trai
                        train_shipping)).tocsr()

# Stacking Validation Data
x_val_final = hstack((val_name, val_item_desc, val_brand_name, val_main_cat, val_sub_cat_1,


# Stacking Test Data
x_test_final = hstack((test_name, test_item_desc, test_brand_name, test_main_cat, test_sub_
                       test_shipping)).tocsr()
```

In [0]:

```
print('x_train_final shape: ', x_train_final.shape, "|||  y_train shape", y_train.shape)
print('x_val_final shape: ', x_val_final.shape, "|||  y_test shape", y_val.shape)
print('x_test_final shape: ', x_test_final.shape)
```

```
x_train_final shape:  (1186028, 11433) |||  y_train shape (1186028,)
x_val_final shape:  (296507, 11433) |||  y_test shape (296507,)
x_test_final shape:  (693359, 11433)
```

In [0]:

SGDRegressor

In [0]:

```
%%time
# Training the Model
model = SGDRegressor()
model.fit(x_train_final, y_train)
```

```
CPU times: user 4.86 s, sys: 101 ms, total: 4.96 s
Wall time: 4.86 s
```

In [0]:

```
# Predictions on Train Data
train_pred = model.predict(x_train_final)
sgd_train_RMSLE = np.sqrt(mean_squared_error(y_train, train_pred))
print("Train RMSLE: ", sgd_train_RMSLE)
```

```
Train RMSLE:  0.5171098022044266
```

In [0]:

```
# Predictions on Validation data
val_pred = model.predict(x_val_final)
sgd_val_RMSLE = np.sqrt(mean_squared_error(y_val, val_pred))
print("Validation RMSLE: ", sgd_val_RMSLE)
```

```
Validation RMSLE:  0.5216289416569649
```

In [0]:

```
# Predictions on Test data
test_pred_sgd = model.predict(x_test_final)
test_pred_sgd.shape
```

Out[81]:

```
(693359,)
```

In [0]:

Ridge

In [0]:

```
%%time
# Training the Model
model = Ridge()
model.fit(x_train_final, y_train)
```

```
CPU times: user 1min 2s, sys: 43.4 s, total: 1min 45s
Wall time: 53.9 s
```

In [0]:

```
# Predictions on Train Data
train_pred = model.predict(x_train_final)
ridge_train_RMSLE = np.sqrt(mean_squared_error(y_train, train_pred))
print("Train RMSLE: ", ridge_train_RMSLE)
```

```
Train RMSLE:  0.5070766597901347
```

In [0]:

```
# Predictions on Validation data
val_pred = model.predict(x_val_final)
ridge_val_RMSLE = np.sqrt(mean_squared_error(y_val, val_pred))
print("Validation RMSLE: ", ridge_val_RMSLE)
```

```
Validation RMSLE:  0.5135844086283321
```

In [0]:

```
# Predictions on Test data
test_pred_ridge = model.predict(x_test_final)
test_pred_ridge.shape
```

Out[85]:

```
(693359,)
```

In [0]:

XGBRegressor

In [0]:

```
%%time
# Training the Model
model = xgb.XGBRegressor()
model.fit(x_train_final, y_train)
```

```
[12:16:51] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:line
ar is now deprecated in favor of reg:squarederror.
CPU times: user 3min 35s, sys: 899 ms, total: 3min 36s
Wall time: 3min 37s
```

In [0]:

```
# Predictions on Train Data
train_pred = model.predict(x_train_final)
xgb_train_RMSLE = np.sqrt(mean_squared_error(y_train, train_pred))
print("Train RMSLE: ", xgb_train_RMSLE)
```

```
Train RMSLE:  0.6146981403019418
```

In [0]:

```
# Predictions on Validation data
val_pred = model.predict(x_val_final)
xgb_val_RMSLE = np.sqrt(mean_squared_error(y_val, val_pred))
print("Validation RMSLE: ", xgb_val_RMSLE)
```

```
Validation RMSLE:  0.6162700673572676
```

In [0]:

```
# Predictions on Test data
test_pred_xgb = model.predict(x_test_final)
test_pred_xgb.shape
```

Out[89]:

```
(693359,)
```

In [0]:

```

```

LGBMRegressor

In [0]:

```
%%time
# Training the Model
model = LGBMRegressor()
model.fit(x_train_final, y_train)
```

```
CPU times: user 2min 58s, sys: 158 ms, total: 2min 58s
Wall time: 2min 58s
```

In [0]:

```
# Predictions on Train Data
train_pred = model.predict(x_train_final)
lgbm_train_RMSLE = np.sqrt(mean_squared_error(y_train, train_pred))
print("Train RMSLE: ", lgbm_train_RMSLE)
```

```
Train RMSLE:  0.5425120498796302
```

In [0]:

```
# Predictions on Validation data
val_pred = model.predict(x_val_final)
lgbm_val_RMSLE = np.sqrt(mean_squared_error(y_val, val_pred))
print("Validation RMSLE: ", lgbm_val_RMSLE)
```

```
Validation RMSLE:  0.5445418450757752
```

In [0]:

```
# Predictions on Test data
test_pred_LGBM = model.predict(x_test_final)
test_pred_LGBM.shape
```

Out[94]:

```
(693359,)
```

In [0]:

```

```

Multilayer Perceptron

In [0]:

```
# Using sequential model
model = Sequential()

model.add(Dense(256, activation='relu', input_dim=11433))  # 11433 is the shape of x_train_
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())
model.add(Dense(64, activation='relu'))
model.add(Dense(1))
```

In [0]:

```
# compiling the model
model.compile(loss='mse',
                optimizer=RMSprop(0.001), #RMSprop(0.001),  # Adam(lr=0.001)
                metrics=['mae', 'mse'])
```

In [0]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 256)               2927104   
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
batch_normalization (BatchNo (None, 128)               512       
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65        
=================================================================
Total params: 2,968,833
Trainable params: 2,968,577
Non-trainable params: 256
_________________________________________________________________
```

In [0]:

```

```

In [0]:

```
'''
# https://intellipaat.com/community/19874/keras-sparse-matrix-issue
def batch_generator(X, y, batch_size):
    number_of_batches = X.shape[0]/batch_size
    counter=0
    shuffle_index = np.arange(np.shape(y)[0])
    np.random.shuffle(shuffle_index)
    X =  X[shuffle_index, :]
    y =  y[shuffle_index]

    while 1:
        index_batch = shuffle_index[batch_size*counter:batch_size*(counter+1)]
        X_batch = X[index_batch,:].todense()
        y_batch = y[index_batch]
        counter += 1

        yield(np.array(X_batch),y_batch)
        if (counter < number_of_batches):
            np.random.shuffle(shuffle_index)
            counter=0
'''
```

In [0]:

```
# https://stackoverflow.com/questions/41538692/using-sparse-matrices-with-keras-and-tensorf
def nn_batch_generator(X_data, y_data, batch_size):
    samples_per_epoch = X_data.shape[0]
    number_of_batches = samples_per_epoch/batch_size
    counter=0
    index = np.arange(np.shape(y_data)[0])
    while 1:
        index_batch = index[batch_size*counter:batch_size*(counter+1)]
        X_batch = X_data[index_batch,:].todense()
        y_batch = y_data[index_batch]
        counter += 1
        yield np.array(X_batch),y_batch
        if (counter > number_of_batches):
            counter=0
```

In [0]:

```
EPOCHS = 5
BATCH_SIZE = 128
```

In [0]:

```
# Training the Model
history = model.fit_generator(generator=nn_batch_generator(x_train_final, y_train.values, E
                    epochs=EPOCHS, validation_data =nn_batch_generator(x_val_final, y_val.v
                    steps_per_epoch=x_train_final.shape[0]/BATCH_SIZE, validation_steps=x_v
```

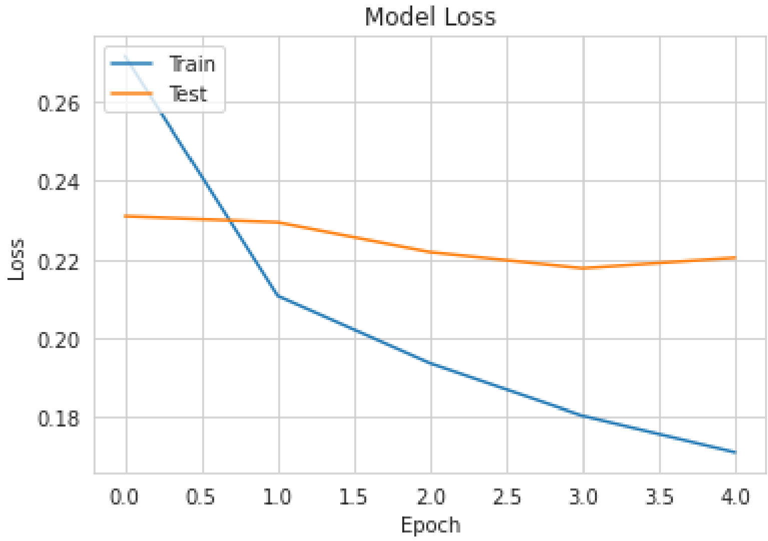
```
WARNING:tensorflow:From <ipython-input-100-85c441c7f0b7>:3: Model.fit_genera
tor (from tensorflow.python.keras.engine.training) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/5
9266/9265 [==============================] - 925s 100ms/step - loss: 0.2718
- mae: 0.3878 - mse: 0.2718 - val_loss: 0.2310 - val_mae: 0.3565 - val_mse:
0.2310
Epoch 2/5
9266/9265 [==============================] - 899s 97ms/step - loss: 0.2107 -
mae: 0.3400 - mse: 0.2107 - val_loss: 0.2294 - val_mae: 0.3594 - val_mse: 0.
2294
Epoch 3/5
9266/9265 [==============================] - 890s 96ms/step - loss: 0.1935 -
mae: 0.3229 - mse: 0.1935 - val_loss: 0.2218 - val_mae: 0.3511 - val_mse: 0.
2218
Epoch 4/5
9266/9265 [==============================] - 892s 96ms/step - loss: 0.1802 -
mae: 0.3098 - mse: 0.1802 - val_loss: 0.2177 - val_mae: 0.3447 - val_mse: 0.
2177
Epoch 5/5
9266/9265 [==============================] - 874s 94ms/step - loss: 0.1709 -
mae: 0.3002 - mse: 0.1709 - val_loss: 0.2204 - val_mae: 0.3466 - val_mse: 0.
2204
```

In [0]:

```
# Reference https://keras.io/visualization/

# Plot training & validation loss values

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [0]:

```

```

In [0]:

```
# Predictions on Train Data
train_pred = model.predict(x_train_final)
mlp_train_RMSLE = np.sqrt(mean_squared_error(y_train, train_pred))
print("Train RMSLE: ", mlp_train_RMSLE)
```

```
Train RMSLE:  0.39874828766074355
```

In [0]:

```
# Predictions on Validation data
val_pred = model.predict(x_val_final)
mlp_val_RMSLE = np.sqrt(mean_squared_error(y_val, val_pred))
print("Train RMSLE: ", mlp_val_RMSLE)
```

```
Train RMSLE:  0.46947987528516916
```

In [0]:

```
# Predictions on Test data
test_pred_MLP = model.predict(x_test_final)
test_pred_MLP = test_pred_MLP.flatten()
test_pred_MLP.shape
```

Out[106]:

```
(693359,)
```

In [0]:

In [0]:

Ensemble

In [0]:

```
# Not using XGBRegressor predictions for Ensembling.
# Using weighted ensemble by giving some weight values

final = (test_pred_sgd * 0.2 + test_pred_ridge * 0.3 + test_pred_LGBM * 0.1 + test_pred_MLP
final.shape
```

Out[110]:

```
(693359,)
```

In [0]:

In [0]:

```
# Making submission file

test_id = test_ids

submission = pd.DataFrame({
        'test_id': test_id,
        'price': final
        })
```

In [0]:

```
submission.to_csv("/content/drive/My Drive/Mercari/new_submission_1.csv", index=False)
```

Conclusion

In [0]:

```
x = PrettyTable()

x. field_names = ['Model', 'Train RMSLE', 'Validation RMSLE']

x.add_row(['SGDRegressor', round(sgd_train_RMSLE, 2), round(sgd_val_RMSLE, 2)])
x.add_row(['Ridge', round(ridge_train_RMSLE, 2), round(ridge_val_RMSLE, 2)])
x.add_row(['XGBRegressor', round(xgb_train_RMSLE, 2), round(xgb_val_RMSLE, 2)])
x.add_row(['LGBMRegressor', round(lgbm_train_RMSLE, 2), round(lgbm_val_RMSLE, 2)])
x.add_row(['Multilayer Perceptron', round(mlp_train_RMSLE, 2), round(mlp_val_RMSLE, 2)])

print(x)
```

```
+-----------------------+-------------+------------------+
|         Model         | Train RMSLE | Validation RMSLE |
+-----------------------+-------------+------------------+
|      SGDRegressor     |     0.51    |       0.52       |
|         Ridge         |     0.50    |       0.51       |
|      XGBRegressor     |     0.61    |       0.61       |
|     LGBMRegressor     |     0.54    |       0.54       |
| Multilayer Perceptron |     0.39    |       0.46       |
+-----------------------+-------------+------------------+
```