Data Science Use Case Retail Price Recommendation with Python

Github Repository Link : https://github.com/Askar2002/Data-Science-Use-Case-Retail-Price-Recommendation-with-Python.git

Mercari adalah suatu ritel belanja aplikasi besar di Jepang, untuk suatu permasalahan yang dalam. Mereka menawarkan rekomendasi suatu harga kepada penjual, tetapi ini sulit karena penjual mereka dapat menempatkan apa saja atau bundle barang apapun di toko Mercari.

Dalam machine learning project ini, saya membuat suatu model otomatis rekomendasi dengan harga produk. Dalam hal ini beberapa informasinya :
train_id — the id of the listing
name — the title of the listing
item_condition_id — the condition of the items provided by the sellers
category_name — category of the listing
brand_name — the name of the brand
price — the price that the item was sold for. This is target variable that we will predict
shipping — 1 if shipping fee is paid by seller and 0 by buyer
item_description — the full description of the item

Langkah Pertama : EDA (Exploratory Data Analysis)

Data set bisa di download dari https://www.kaggle.com/datasets/saitosean/mercari . Untuk mencari hasil yang valid saya hanya menggunakan the train.tsv.

```
import gc
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.sparse import csr_matrix, hstack
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error
import lightgbm as lgb

df = pd.read_csv('train.tsv', sep = '\t')
```

Randomly split the data into train and test sets. Saya menggunakan training set hanya untuk EDA.

```
msk = np.random.rand(len(df)) < 0.8
train = df[msk]
test = df[~msk]

train.shape, test.shape
```

((1185866, 8), (296669, 8))

```
train.head()
```

	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 & P...	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]...
5	5	Bundled items requested for Rule	3	Women/Other/Other	NaN	59.0	0	Banana republic bottoms, Candies skirt with ma...

Figure 1

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1185866 entries, 0 to 1482533
Data columns (total 8 columns):
train_id             1185866 non-null int64
name                 1185866 non-null object
item_condition_id    1185866 non-null int64
category_name        1180783 non-null object
brand_name           679496 non-null object
price                1185866 non-null float64
shipping             1185866 non-null int64
item_description     1185863 non-null object
dtypes: float64(1), int64(3), object(4)
memory usage: 81.4+ MB
```

Figure 2

Harga

```
train.price.describe()
```

```
count    1.185866e+06
mean     2.674116e+01
std      3.852606e+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
```

Figure 3

Harga barang miring, sebagian besar barang dihargai 10–20. Namun, barang paling mahal dibanderol pada 2009. Jadi kami akan melakukan transformasi log pada harganya.

```
plt.subplot(1, 2, 1)
(train['price']).plot.hist(bins=50, figsize=(12, 6), edgecolor =
'white', range = [0, 250])
plt.xlabel('price', fontsize=12)
plt.title('Price Distribution', fontsize=12)


plt.subplot(1, 2, 2)
np.log(train['price']+1).plot.hist(bins=50, figsize=(12,6),
edgecolor='white')
plt.xlabel('log(price+1)', fontsize=12)
plt.title('Price Distribution', fontsize=12)
```

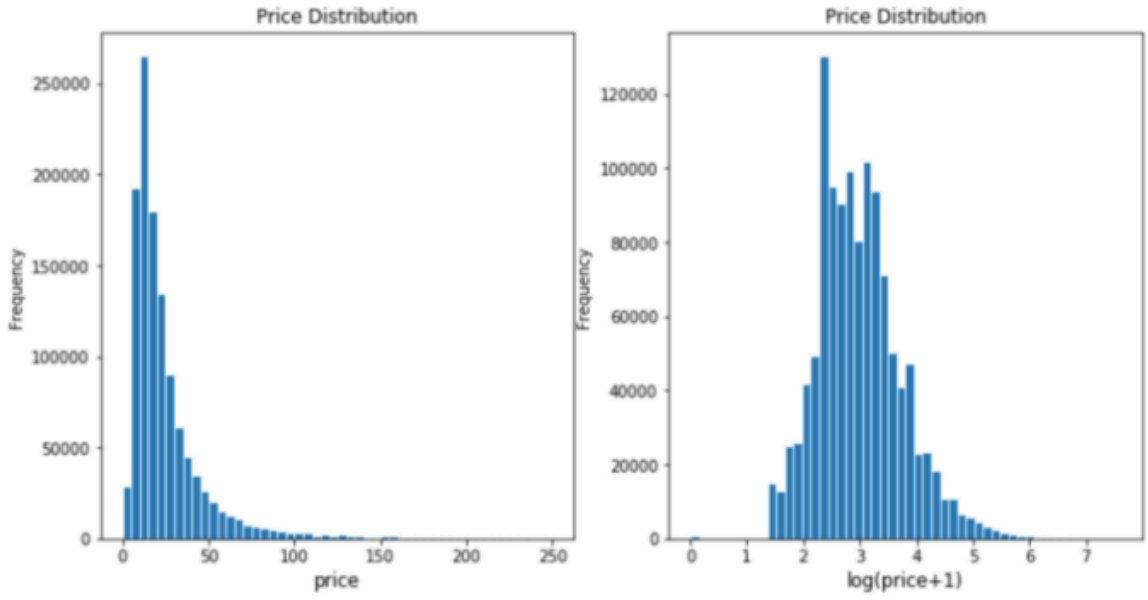

Figure 4

Shipping

Lebih dari 55% dari biaya pengiriman barang dibayar oleh pembeli.

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

```
0    0.552411
1    0.447589
Name: shipping, dtype: float64
```

Figure 5

Bagaimana shipping related dengan harga ?

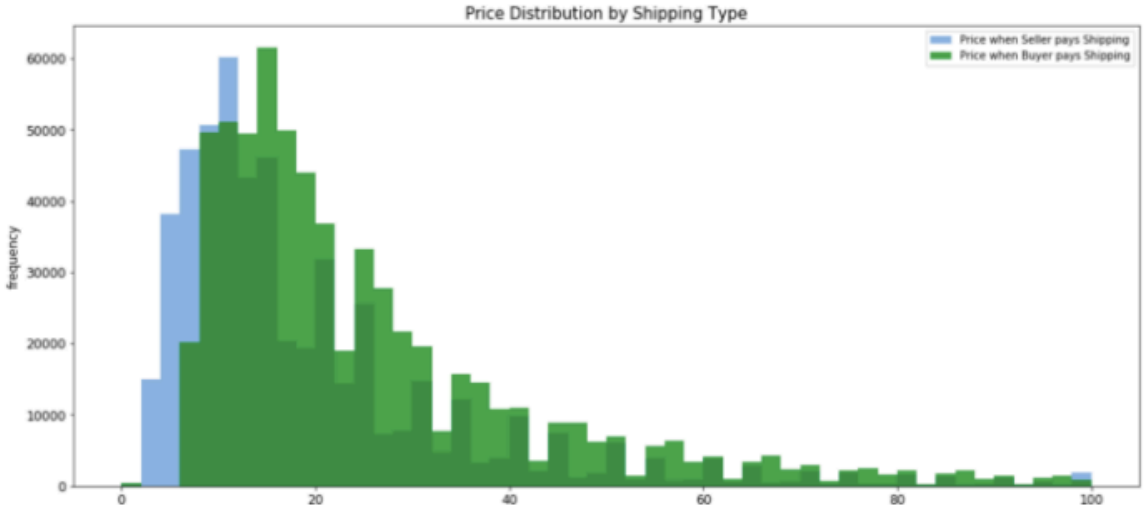

Figure 6

```
print('The average price is
{}'.format(round(shipping_fee_by_seller.mean(), 2)), 'if seller pays
shipping');
print('The average price is
{}'.format(round(shipping_fee_by_buyer.mean(), 2)), 'if buyer pays
shipping')
```

Rata rata harga adalah 22,58 jika seller membayar shipping

Rata rata harga 30,11 jika pembeli bayar shipping

Kemudian kita bandingkan dengan sesudah log transformation on the price.

```
fig, ax = plt.subplots(figsize=(18,8))
ax.hist(np.log(shipping_fee_by_seller+1), color='#8CB4E1', alpha=1.0,
bins=50,
       label='Price when Seller pays Shipping')
ax.hist(np.log(shipping_fee_by_buyer+1), color='#007D00', alpha=0.7,
bins=50,
       label='Price when Buyer pays Shipping')
plt.xlabel('log(price+1)', fontsize=12)
plt.ylabel('frequency', fontsize=12)
plt.title('Price Distribution by Shipping Type', fontsize=15)
plt.tick_params(labelsize=12)
plt.legend()
plt.show()
```

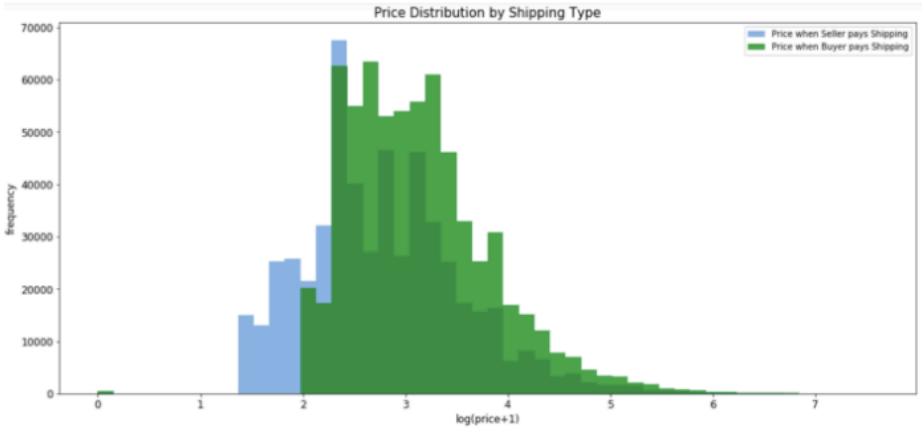

Figure 7

Jelas bahwa harga rata rata lebih tinggi ketika pembeli membayar shipping.

Category Names

```
print('There are', train['category_name'].nunique(), 'unique values in
category name column')
```

There are 1265 unique values in category name column

Top 10 most common category names :

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

```
Women/Athletic Apparel/Pants, Tights, Leggings                48028
Women/Tops & Blouses/T-Shirts                                 37173
Beauty/Makeup/Face                                            27438
Beauty/Makeup/Lips                                            23949
Electronics/Video Games & Consoles/Games                      21265
Beauty/Makeup/Eyes                                            20133
Electronics/Cell Phones & Accessories/Cases, Covers & Skins   19708
Women/Underwear/Bras                                          17065
Women/Tops & Blouses/Tank, Cami                               16225
Women/Tops & Blouses/Blouse                                   16200
Name: category_name, dtype: int64
```

Figure 8

Item condition vs Harga

```
sns.boxplot(x = 'item_condition_id', y = np.log(train['price']+1),
data = train, palette = sns.color_palette('RdBu',5))
```

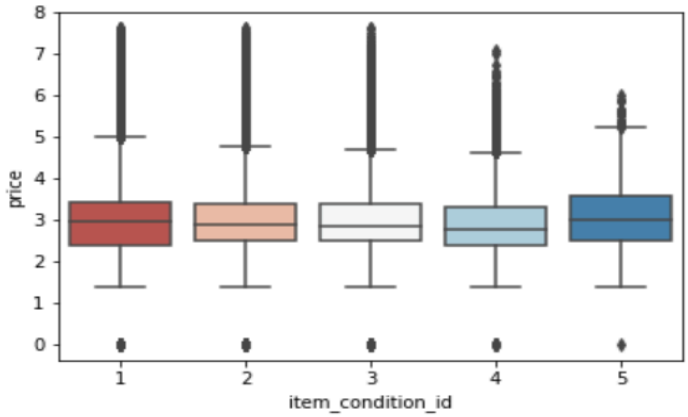

Figure 8

Ada perbedaan harga rata rata antara setiap item condition id

Setelah Exploratory Data Analysis, saya memutuskan untuk menggunakan semua fitru untuk membangun model.

Langkah Kedua : LightGBM

Dibawah payung proyek DMTK Microsoft, LightGBM adalah kerangka kerja peningkatan gradien yang menggunakan algoritma pembelajaran berbasis pohon. Ini dirancang untuk didistribusikan dan efisien dengan keuntungan sebagai berikut:

- Faster training speed and higher efficiency
- Lower memory usage
- Better accuracy
- Parallel and GPU learning supported
- Capable of handling large-scale data

Kemudian, kita melakukan percobaan.

Langkah Ketiga : General Setting

```
NUM_BRANDS = 4000
NUM_CATEGORIES = 1000
NAME_MIN_DF = 10
MAX_FEATURES_ITEM_DESCRIPTION = 50000
```

There are missing values in the columns that we have to fix :

```
print('There are %d items that do not have a category name.'
%train['category_name'].isnull().sum())
```

There are 5083 items that do not have a category name.

```
print('There are %d items that do not have a brand name.'
%train['brand_name'].isnull().sum())
```

There are 506370 items that do not have a brand name.

```
print('There are %d items that do not have a description.'
%train['item_description'].isnull().sum())
```

There are 3 items that do not have a description.

Helper function for LigthGBM :

```
def handle_missing_inplace(dataset):
    dataset['category_name'].fillna(value='missing', inplace=True)
    dataset['brand_name'].fillna(value='missing', inplace=True)
    dataset['item_description'].replace('No description
yet,''missing', inplace=True)
    dataset['item_description'].fillna(value='missing', inplace=True)


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_category = dataset['category_name'].value_counts().loc[lambda
x: x.index != 'missing'].index[:NUM_CATEGORIES]


def to_categorical(dataset):
    dataset['category_name'] =
dataset['category_name'].astype('category')
    dataset['brand_name'] = dataset['brand_name'].astype('category')
    dataset['item_condition_id'] =
dataset['item_condition_id'].astype('category')
```

Drop rows where price = 0

```
df = pd.read_csv('train.tsv', sep = '\t')
msk = np.random.rand(len(df)) < 0.8
train = df[msk]
test = df[~msk]
test_new = test.drop('price', axis=1)
y_test = np.log1p(test["price"])

train = train[train.price != 0].reset_index(drop=True)
```

Merge train and new test data.

```
nrow_train = train.shape[0]
y = np.log1p(train["price"])
merge: pd.DataFrame = pd.concat([train, test_new])
```

Langkah Keempat : Training Preparation

```
handle_missing_inplace(merge)
cutting(merge)
to_categorical(merge)
```

Count vectorize name and category name columns.

```
cv = CountVectorizer(min_df=NAME_MIN_DF)
X_name = cv.fit_transform(merge['name'])

cv = CountVectorizer()
X_category = cv.fit_transform(merge['category_name'])
```

TF-IDF Vectoize item_description column.

```
tv = TfidfVectorizer(max_features=MAX_FEATURES_ITEM_DESCRIPTION,
ngram_range=(1, 3), stop_words='english')
X_description = tv.fit_transform(merge['item_description'])
```

Label binarize brand_name column.

```
lb = LabelBinarizer(sparse_output=True)
X_brand = lb.fit_transform(merge['brand_name'])
```

Create dummy variables for item_condition_id dan shipping column.

```
X_dummies =csr_matrix(pd.get_dummies(merge[['item_condition_id',
'shipping']], sparse=True).values)
```

Create sparse merge.

```
sparse_merge = hstack((X_dummies, X_description, X_brand,
X_category, X_name)).tocsr()
```

Remove features dengan document frequency <=1.

```
mask = np.array(np.clip(sparse_merge.getnnz(axis=0) - 1, 0, 1),
dtype=bool)
sparse_merge = sparse_merge[:, mask]
```

Separate train dan test data dari sparse merge

```
X = sparse_merge[:nrow_train]
X_test = sparse_merge[nrow_train:]
```

Create dataset dari lightgbm

```
train_X = lgb.Dataset(X, label=y)
```

Specify our parameters as a dict.

```
params = {
'learning_rate': 0.75,
'application': 'regression',
'max_depth': 3,
'num_leaves': 100,
'verbosity': -1,
'metric': 'RMSE',
}
```

- Use ‘regression’ as application as we are dealing with a regression problem.
- Use ‘RMSE’ as metric because this is a regression problem.
- “num_leaves”=100 as our data is relative big.
- Use “max_depth” to avoid overfitting.
- Use “verbosity” to control the level of LightGBM’s verbosity (<0: Fatal).
- “learning_rate” determines the impact of each tree on the final outcome.

Langkah Kelima : Training Start

Training model memerlukan daftar parameter dan kumpulan data. Dan pelatihan akan memakan waktu cukup lama.

```
gbm = lgb.train(params, train_set=train_X, num_boost_round=3200,
verbose_eval=100)
```

Langkah Keenam : Predict

```
y_pred = gbm.predict(X_test, num_iteration=gbm.best_iteration)
```

Langkah Ketujuh : Evaluation

```
from sklearn.metrics import mean_squared_error
print('The rmse of prediction is:', mean_squared_error(y_test,
y_pred) ** 0.5)
```

The rmse of prediction is: 0.46164222941613137

Refrensi source code bisa ditemukan di https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Mercari%20Price%20Suggestion%20Lightgbm.ipynb

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