09RecommendationSystem

November 4, 2018

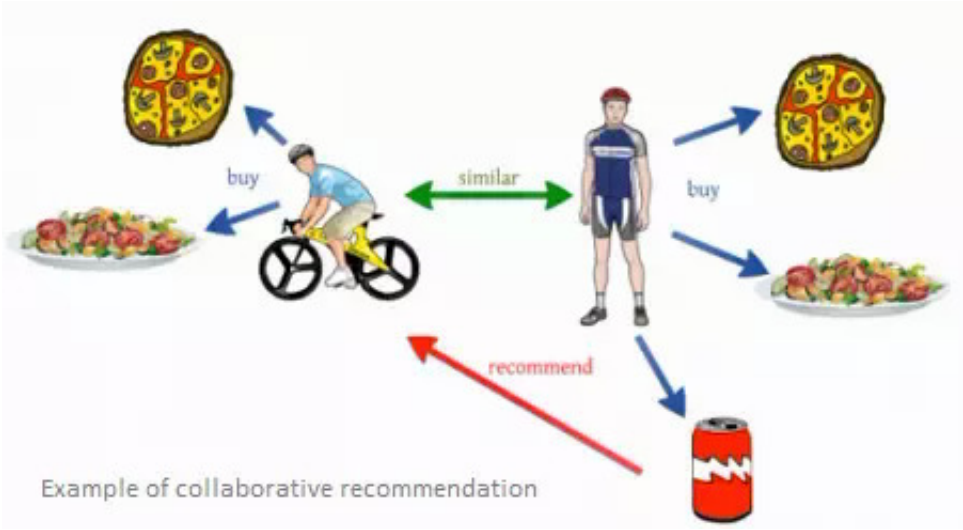
1. 推薦系統的種類
   - collabrative(協同推薦)
   - content-based(內容推薦)

   from quora

```
In [32]: import re
         import pandas as pd
         import numpy as np
         from collections import Counter
         from sklearn import preprocessing
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.decomposition import PCA
         from sklearn.model_selection import train_test_split
         from matplotlib import pyplot as plt
         import xgboost
```

```
In [2]: df_articles = pd.read_csv('shared_articles.csv')
        df_articles = df_articles[df_articles['eventType'] == 'CONTENT SHARED']
        df_articles.info()
        df_articles.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3047 entries, 1 to 3121
Data columns (total 13 columns):
timestamp          3047 non-null int64
eventType          3047 non-null object
contentId          3047 non-null int64
authorPersonId     3047 non-null int64
authorSessionId    3047 non-null int64
authorUserAgent    669 non-null object
authorRegion       669 non-null object
authorCountry      669 non-null object
contentType        3047 non-null object
url                3047 non-null object
```



collabrative

content-based

```
title              3047 non-null object
text               3047 non-null object
lang               3047 non-null object
dtypes: int64(4), object(9)
memory usage: 333.3+ KB
```

Out[2]:

	timestamp	eventType	contentId	authorPersonId
1	1459193988	CONTENT SHARED	-4110354420726924665	4340306774493623681
2	1459194146	CONTENT SHARED	-7292285110016212249	4340306774493623681
3	1459194474	CONTENT SHARED	-6151852268067518688	3891637997717104548
4	1459194497	CONTENT SHARED	2448026894306402386	4340306774493623681
5	1459194522	CONTENT SHARED	-2826566343807132236	4340306774493623681

	authorSessionId	authorUserAgent	authorRegion	authorCountry	contentType
1	8940341205206233829	NaN	NaN	NaN	HTML
2	8940341205206233829	NaN	NaN	NaN	HTML
3	-1457532940883382585	NaN	NaN	NaN	HTML
4	8940341205206233829	NaN	NaN	NaN	HTML
5	8940341205206233829	NaN	NaN	NaN	HTML

	url
1	http://www.nytimes.com/2016/03/28/business/dea...
2	http://cointelegraph.com/news/bitcoin-future-w...
3	https://cloudplatform.googleblog.com/2016/03/G...
4	https://bitcoinmagazine.com/articles/ibm-wants...
5	http://www.coindesk.com/ieee-blockchain-oxford...

	title
1	Ethereum, a Virtual Currency, Enables Transact...
2	Bitcoin Future: When GBPcoin of Branson Wins O...
3	Google Data Center 360ř Tour
4	IBM Wants to "Evolve the Internet" With Blockc...
5	IEEE to Talk Blockchain at Cloud Computing Oxf...

	text	lang
1	All of this work is still very early. The firs...	en
2	The alarm clock wakes me at 8:00 with stream o...	en
3	We're excited to share the Google Data Center ...	en
4	The Aite Group projects the blockchain market ...	en
5	One of the largest and oldest organizations fo...	en

In [3]:
```
def process_Agent(Agent):
    if "windows" in Agent:
        return 0
    elif "macintosh" in Agent:
        return 1
    elif "linux" in Agent:
```

```
            return 2
        else:
            return 3


    le_authorUserAgent = preprocessing.LabelEncoder()
    df_articles['authorUserAgent'] = le_authorUserAgent.fit_transform(df_articles['authorUs
    le_authorRegion = preprocessing.LabelEncoder()
    df_articles['authorRegion'] = le_authorRegion.fit_transform(df_articles['authorRegion']
    le_authorCountry = preprocessing.LabelEncoder()
    df_articles['authorCountry'] = le_authorCountry.fit_transform(df_articles['authorCountr
    le_contentType = preprocessing.LabelEncoder()
    df_articles['contentType'] = le_contentType.fit_transform(df_articles['contentType'].as
    le_url = preprocessing.LabelEncoder()
    df_articles['url'] = le_url.fit_transform(df_articles['url'].apply(lambda x:re.findall
    le_lang = preprocessing.LabelEncoder()
    df_articles['lang'] = le_lang.fit_transform(df_articles['lang'].astype(str).apply(lambd

    tfidf = TfidfVectorizer()
    tfidf.fit(np.hstack([df_articles['title'].values, df_articles['text'].values]))
    df_articles['title'] = list(tfidf.transform(df_articles['title'].values).toarray())
    df_articles['text'] = list(tfidf.transform(df_articles['text'].values).toarray())


    df_articles = df_articles[["contentId", "authorUserAgent","authorRegion","authorCountry
    df_articles.head(5)
```

Out[3]:

	contentId	authorUserAgent	authorRegion	authorCountry	\
1	-4110354420726924665	3	9	3	
2	-7292285110016212249	3	9	3	
3	-6151852268067518688	3	9	3	
4	2448026894306402386	3	9	3	
5	-2826566343807132236	3	9	3	

	contentType	url	lang	title	\
1	0	896	0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	
2	0	182	0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	
3	0	176	0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	
4	0	50	0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	
5	0	706	0	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	

	text
1	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
2	[0.048182400537760725, 0.0, 0.0, 0.0, 0.0, 0.0...
3	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
4	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5	[0.0, 0.03408482227280377, 0.0, 0.0, 0.0, 0.0,...

In [5]:
```
X_nlp_forPCA = np.concatenate([
```

```
            np.hstack(df_articles['title'].values).reshape(len(df_articles),-1),
            np.hstack(df_articles['text'].values).reshape(len(df_articles),-1)
        ], axis=1)

        pca = PCA(n_components=400, random_state=1212)
        df_articles['title'] = list(pca.fit_transform(X_nlp_forPCA[:, :int(X_nlp_forPCA.shape[
        df_articles['text'] = list(pca.fit_transform(X_nlp_forPCA[:, int(X_nlp_forPCA.shape[1],
        df_articles.head(5)
```

Out[5]:

	contentId	authorUserAgent	authorRegion	authorCountry
1	-4110354420726924665	3	9	3
2	-7292285110016212249	3	9	3
3	-6151852268067518688	3	9	3
4	2448026894306402386	3	9	3
5	-2826566343807132236	3	9	3

	contentType	url	lang	title
1	0	896	0	[-0.04075413153557972, -0.015353894409987887, ...
2	0	182	0	[-0.0012556766428318809, -0.06646168057041796,...
3	0	176	0	[0.11418731903476237, 0.17239345455221505, -0...
4	0	50	0	[0.08234964109707694, -0.13114805530673168, -0...
5	0	706	0	[0.04849331403920229, 0.012213744929441139, -0...

	text
1	[-0.25381908135912207, -0.03928945983356414, -...
2	[-0.10704562138903156, 0.0692688189810105, -0...
3	[-0.20080173580474742, 0.0334745050091199, 0.0...
4	[-0.29032820837142886, -0.08506507624478626, -...
5	[-0.16041997828540744, 0.057870128420256864, -...

```
In [6]: df_interactions = pd.read_csv('users_interactions.csv')
        df_interactions.info()
        df_interactions.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72312 entries, 0 to 72311
Data columns (total 8 columns):
timestamp      72312 non-null int64
eventType      72312 non-null object
contentId      72312 non-null int64
personId       72312 non-null int64
sessionId      72312 non-null int64
userAgent      56918 non-null object
userRegion     56907 non-null object
userCountry    56918 non-null object
dtypes: int64(4), object(4)
memory usage: 4.4+ MB
```

Out[6]:

	timestamp	eventType	contentId	personId	sessionId	userAgent	userRegion	userCountry
0	1465413032	VIEW	-3499919498720038879	-8845298781299428018	1264196770339959068	NaN	NaN	NaN
1	1465412560	VIEW	8890720798209849691	-1032019229384696495	3621737643587579081	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2...	NY	US
2	1465416190	VIEW	310515487419366995	-1130272294246983140	2631864456530402479	NaN	NaN	NaN
3	1465413895	FOLLOW	310515487419366995	344280948527967603	-3167637573980064150	NaN	NaN	NaN
4	1465412290	VIEW	-7820640624231356730	-445337111692715325	5611481178424124714	NaN	NaN	NaN

In [7]:
```
score_mapping = {'BOOKMARK':5,
                 'COMMENT CREATED':4,
                 'FOLLOW':3,
                 'LIKE':2,
                 'VIEW':1
                }
df_interactions['eventType'] = df_interactions['eventType'].apply(score_mapping.get)

le_userAgent = preprocessing.LabelEncoder()
df_interactions['userAgent'] = le_userAgent.fit_transform(df_interactions['userAgent'])
le_userRegion = preprocessing.LabelEncoder()
df_interactions['userRegion'] = le_userRegion.fit_transform(df_interactions['userRegion'])
le_userCountry = preprocessing.LabelEncoder()
df_interactions['userCountry'] = le_userCountry.fit_transform(df_interactions['userCountry'])

df_interactions = df_interactions[["contentId", "userAgent", "userRegion", "userCountry", "eventType"]]
df_interactions.head(5)
```

Out[7]:

	contentId	userAgent	userRegion	userCountry	eventType
0	-3499919498720038879	3	43	18	1
1	8890720798209849691	1	50	22	1
2	310515487419366995	3	43	18	1
3	310515487419366995	3	43	18	3
4	-7820640624231356730	3	43	18	1

In [8]:
```
df_merge = pd.merge(df_interactions, df_articles, on='contentId')
df_merge.head(5)
```

Out[8]:

	contentId	userAgent	userRegion	userCountry	eventType	\
0	-3499919498720038879	3	43	18	1	
1	-3499919498720038879	1	66	2	1	
2	-3499919498720038879	0	66	2	1	
3	-3499919498720038879	2	66	2	1	
4	-3499919498720038879	2	66	2	1	

	authorUserAgent	authorRegion	authorCountry	contentType	url	lang	\
0	3	9	3	0	569	0	
1	3	9	3	0	569	0	
2	3	9	3	0	569	0	
3	3	9	3	0	569	0	
4	3	9	3	0	569	0	

	title	\
0	[0.04436414967487065, -0.0828542054558259, -0...	
1	[0.04436414967487065, -0.0828542054558259, -0...	
2	[0.04436414967487065, -0.0828542054558259, -0...	
3	[0.04436414967487065, -0.0828542054558259, -0...	
4	[0.04436414967487065, -0.0828542054558259, -0...	

	text
0	[-0.2306608936690724, -0.04000880102084441, 0...
1	[-0.2306608936690724, -0.04000880102084441, 0...
2	[-0.2306608936690724, -0.04000880102084441, 0...
3	[-0.2306608936690724, -0.04000880102084441, 0...
4	[-0.2306608936690724, -0.04000880102084441, 0...

In [18]:
```
X_part1 = df_merge[["userAgent", "userRegion", "userCountry", "authorUserAgent", "autl
X_part2 = np.concatenate([
    np.hstack(df_merge['title'].values).reshape(len(df_merge),-1),
    np.hstack(df_merge['text'].values).reshape(len(df_merge),-1)
], axis=1)
X = np.concatenate([X_part1, X_part2], axis=1)
Y = df_merge[['eventType']].values
Y = np.log(Y)
```

In [24]:
```
X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, test_size=1500, random_sta
print(X_train.shape)
print(X_valid.shape)
print(Y_train.shape)
print(Y_valid.shape)
```

```
(70769, 809)
(1500, 809)
(70769, 1)
(1500, 1)
```

In [27]:
```
# Set our parameters for xgboost
params = {}

# 請填入以下參數:
# 目標函數: 線性回歸
# 評價函數: rmse
# 學習速度: 0.01
# 最大深度: 5
# bst = xgboost.train(params, d_train, 3000, watchlist, early_stopping_rounds=50, ver
#=============your works starts===============#
params['objective'] = 'reg:linear'
params['eval_metric'] = 'rmse'
params['eta'] = 0.03
params['max_depth'] = 3
d_train = xgboost.DMatrix(X_train, label=Y_train)
d_valid = xgboost.DMatrix(X_valid, label=Y_valid)
watchlist = [(d_train, 'train'), (d_valid, 'valid')]
bst = xgboost.train(params, d_train, 3000, watchlist, early_stopping_rounds=10, verbos
Y_pred = bst.predict(xgboost.DMatrix(X_valid))
#==============your works ends================#
```

```
[0]        train-rmse:0.519924        valid-rmse:0.529289
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 10 rounds.
[10]        train-rmse:0.444217        valid-rmse:0.455455
[20]        train-rmse:0.396852        valid-rmse:0.409591
[30]        train-rmse:0.368338        valid-rmse:0.382182
[40]        train-rmse:0.35164        valid-rmse:0.36633
[50]        train-rmse:0.341998        valid-rmse:0.357203
[60]        train-rmse:0.33638        valid-rmse:0.351872
[70]        train-rmse:0.332787        valid-rmse:0.348679
[80]        train-rmse:0.330332        valid-rmse:0.34654
[90]        train-rmse:0.328617        valid-rmse:0.344939
[100]        train-rmse:0.327286        valid-rmse:0.343778
[110]        train-rmse:0.326319        valid-rmse:0.34307
[120]        train-rmse:0.325527        valid-rmse:0.342513
[130]        train-rmse:0.324792        valid-rmse:0.341901
[140]        train-rmse:0.324058        valid-rmse:0.341215
[150]        train-rmse:0.323312        valid-rmse:0.340539
[160]        train-rmse:0.322642        valid-rmse:0.339847
[170]        train-rmse:0.322034        valid-rmse:0.339191
[180]        train-rmse:0.321455        valid-rmse:0.338715
[190]        train-rmse:0.320809        valid-rmse:0.337999
[200]        train-rmse:0.320207        valid-rmse:0.337428
[210]        train-rmse:0.319681        valid-rmse:0.336853
[220]        train-rmse:0.319109        valid-rmse:0.336449
[230]        train-rmse:0.318613        valid-rmse:0.336072
```

```
[240]	train-rmse:0.318149	valid-rmse:0.335722
[250]	train-rmse:0.317682	valid-rmse:0.335274
[260]	train-rmse:0.317176	valid-rmse:0.334879
[270]	train-rmse:0.316659	valid-rmse:0.334438
[280]	train-rmse:0.316231	valid-rmse:0.334093
[290]	train-rmse:0.315796	valid-rmse:0.333558
[300]	train-rmse:0.315377	valid-rmse:0.333211
[310]	train-rmse:0.314968	valid-rmse:0.332847
[320]	train-rmse:0.314595	valid-rmse:0.33254
[330]	train-rmse:0.314226	valid-rmse:0.332288
[340]	train-rmse:0.313848	valid-rmse:0.331989
[350]	train-rmse:0.313469	valid-rmse:0.331619
[360]	train-rmse:0.313113	valid-rmse:0.331386
[370]	train-rmse:0.312714	valid-rmse:0.331124
[380]	train-rmse:0.312345	valid-rmse:0.330714
[390]	train-rmse:0.312033	valid-rmse:0.330414
[400]	train-rmse:0.311746	valid-rmse:0.33015
[410]	train-rmse:0.311353	valid-rmse:0.329825
[420]	train-rmse:0.31109	valid-rmse:0.329667
[430]	train-rmse:0.310857	valid-rmse:0.329464
[440]	train-rmse:0.310545	valid-rmse:0.329158
[450]	train-rmse:0.310306	valid-rmse:0.32884
[460]	train-rmse:0.309997	valid-rmse:0.328553
[470]	train-rmse:0.309733	valid-rmse:0.328297
[480]	train-rmse:0.309445	valid-rmse:0.327973
[490]	train-rmse:0.309188	valid-rmse:0.327688
[500]	train-rmse:0.308953	valid-rmse:0.327531
[510]	train-rmse:0.308686	valid-rmse:0.327254
[520]	train-rmse:0.308428	valid-rmse:0.327081
[530]	train-rmse:0.308173	valid-rmse:0.326834
[540]	train-rmse:0.30793	valid-rmse:0.326664
[550]	train-rmse:0.307708	valid-rmse:0.326537
[560]	train-rmse:0.307431	valid-rmse:0.326347
[570]	train-rmse:0.307207	valid-rmse:0.326062
[580]	train-rmse:0.306998	valid-rmse:0.325935
[590]	train-rmse:0.306776	valid-rmse:0.325834
[600]	train-rmse:0.306554	valid-rmse:0.32561
[610]	train-rmse:0.306367	valid-rmse:0.325422
[620]	train-rmse:0.306138	valid-rmse:0.325236
[630]	train-rmse:0.305954	valid-rmse:0.325087
[640]	train-rmse:0.30578	valid-rmse:0.324881
[650]	train-rmse:0.305603	valid-rmse:0.324768
[660]	train-rmse:0.305384	valid-rmse:0.324494
[670]	train-rmse:0.305163	valid-rmse:0.324424
[680]	train-rmse:0.304977	valid-rmse:0.324272
[690]	train-rmse:0.304799	valid-rmse:0.32413
[700]	train-rmse:0.304603	valid-rmse:0.324037
[710]	train-rmse:0.30444	valid-rmse:0.32384
```

```
[720]	train-rmse:0.304252	valid-rmse:0.323684
[730]	train-rmse:0.30409	valid-rmse:0.32354
[740]	train-rmse:0.303916	valid-rmse:0.323476
[750]	train-rmse:0.303723	valid-rmse:0.323322
[760]	train-rmse:0.303525	valid-rmse:0.323254
[770]	train-rmse:0.30339	valid-rmse:0.323171
[780]	train-rmse:0.303218	valid-rmse:0.323005
[790]	train-rmse:0.303038	valid-rmse:0.322929
[800]	train-rmse:0.302938	valid-rmse:0.32287
[810]	train-rmse:0.302773	valid-rmse:0.322666
[820]	train-rmse:0.302631	valid-rmse:0.322576
[830]	train-rmse:0.302435	valid-rmse:0.322342
[840]	train-rmse:0.302279	valid-rmse:0.322242
[850]	train-rmse:0.30215	valid-rmse:0.322117
[860]	train-rmse:0.301994	valid-rmse:0.321986
[870]	train-rmse:0.301834	valid-rmse:0.321899
[880]	train-rmse:0.301719	valid-rmse:0.321781
[890]	train-rmse:0.301582	valid-rmse:0.321687
[900]	train-rmse:0.301444	valid-rmse:0.32157
[910]	train-rmse:0.30129	valid-rmse:0.321441
[920]	train-rmse:0.301164	valid-rmse:0.321379
[930]	train-rmse:0.30105	valid-rmse:0.321303
[940]	train-rmse:0.300935	valid-rmse:0.321252
[950]	train-rmse:0.300832	valid-rmse:0.321213
[960]	train-rmse:0.300716	valid-rmse:0.321126
[970]	train-rmse:0.300626	valid-rmse:0.321078
[980]	train-rmse:0.300497	valid-rmse:0.321022
[990]	train-rmse:0.300369	valid-rmse:0.320985
[1000]	train-rmse:0.300283	valid-rmse:0.320907
[1010]	train-rmse:0.300139	valid-rmse:0.320755
[1020]	train-rmse:0.300026	valid-rmse:0.320625
[1030]	train-rmse:0.299899	valid-rmse:0.320548
[1040]	train-rmse:0.299762	valid-rmse:0.320421
[1050]	train-rmse:0.29964	valid-rmse:0.320361
[1060]	train-rmse:0.299526	valid-rmse:0.320286
[1070]	train-rmse:0.299403	valid-rmse:0.320201
[1080]	train-rmse:0.299287	valid-rmse:0.320123
[1090]	train-rmse:0.299177	valid-rmse:0.320095
[1100]	train-rmse:0.29906	valid-rmse:0.320035
[1110]	train-rmse:0.298957	valid-rmse:0.319947
[1120]	train-rmse:0.298827	valid-rmse:0.319908
[1130]	train-rmse:0.298757	valid-rmse:0.319851
[1140]	train-rmse:0.298645	valid-rmse:0.319754
[1150]	train-rmse:0.298533	valid-rmse:0.319729
[1160]	train-rmse:0.298416	valid-rmse:0.319595
[1170]	train-rmse:0.29831	valid-rmse:0.319565
[1180]	train-rmse:0.298179	valid-rmse:0.319455
[1190]	train-rmse:0.298087	valid-rmse:0.319405
```

```
[1200]        train-rmse:0.297997        valid-rmse:0.319362
[1210]        train-rmse:0.29792        valid-rmse:0.319272
[1220]        train-rmse:0.297799        valid-rmse:0.319207
[1230]        train-rmse:0.297683        valid-rmse:0.319106
[1240]        train-rmse:0.297601        valid-rmse:0.31909
[1250]        train-rmse:0.297486        valid-rmse:0.31897
[1260]        train-rmse:0.297371        valid-rmse:0.318893
[1270]        train-rmse:0.297289        valid-rmse:0.318832
[1280]        train-rmse:0.297193        valid-rmse:0.318759
[1290]        train-rmse:0.297091        valid-rmse:0.318677
[1300]        train-rmse:0.297007        valid-rmse:0.318641
[1310]        train-rmse:0.296886        valid-rmse:0.318586
[1320]        train-rmse:0.296802        valid-rmse:0.318542
[1330]        train-rmse:0.296741        valid-rmse:0.318499
[1340]        train-rmse:0.296636        valid-rmse:0.318439
[1350]        train-rmse:0.296528        valid-rmse:0.318351
[1360]        train-rmse:0.296424        valid-rmse:0.318293
[1370]        train-rmse:0.296342        valid-rmse:0.318247
[1380]        train-rmse:0.296277        valid-rmse:0.31814
[1390]        train-rmse:0.296215        valid-rmse:0.31807
[1400]        train-rmse:0.296091        valid-rmse:0.317988
[1410]        train-rmse:0.296006        valid-rmse:0.3179
[1420]        train-rmse:0.295902        valid-rmse:0.317872
[1430]        train-rmse:0.29583        valid-rmse:0.317857
[1440]        train-rmse:0.295744        valid-rmse:0.317798
[1450]        train-rmse:0.295658        valid-rmse:0.317751
[1460]        train-rmse:0.295588        valid-rmse:0.317711
[1470]        train-rmse:0.295494        valid-rmse:0.317604
[1480]        train-rmse:0.295418        valid-rmse:0.317545
[1490]        train-rmse:0.295316        valid-rmse:0.317502
[1500]        train-rmse:0.295247        valid-rmse:0.31748
[1510]        train-rmse:0.295183        valid-rmse:0.317466
[1520]        train-rmse:0.295086        valid-rmse:0.317349
[1530]        train-rmse:0.295001        valid-rmse:0.317293
[1540]        train-rmse:0.294898        valid-rmse:0.31717
[1550]        train-rmse:0.294795        valid-rmse:0.317105
[1560]        train-rmse:0.294709        valid-rmse:0.31706
[1570]        train-rmse:0.294617        valid-rmse:0.317046
[1580]        train-rmse:0.294539        valid-rmse:0.317037
Stopping. Best iteration:
[1577]        train-rmse:0.294558        valid-rmse:0.31702
```

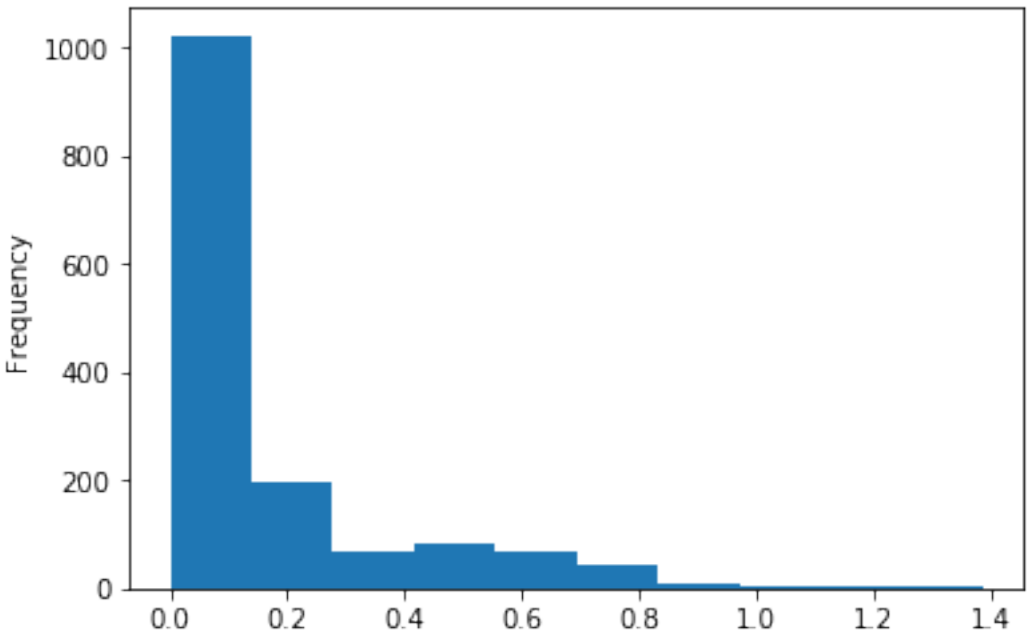
```
In [30]: df_result = pd.DataFrame()
         Y_pred = bst.predict(xgboost.DMatrix(X_valid))
         df_result['predict'] = np.exp(Y_pred)
         df_result['truth'] = np.exp((list(Y_valid)))
```

```
df_result['error'] = df_result.apply(lambda x:np.abs(x['predict'] - x['truth']) / x['t
df_result_sort = df_result.sort_values('truth')
df_result.head()
```

Out[30]:

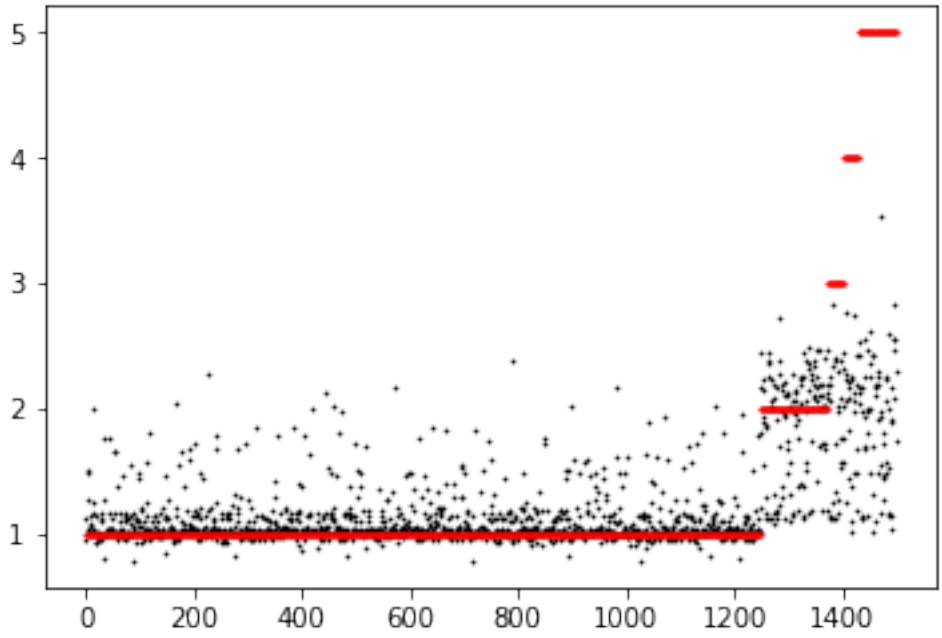
	predict	truth	error
0	1.124131	1.0	0.124131
1	1.018890	1.0	0.018890
2	1.061923	1.0	0.061923
3	1.198980	1.0	0.198980
4	1.017537	1.0	0.017537

In [35]:
```
df_result_sort['error'].plot('hist')
plt.show()
```



In [36]:
```
# 請使用 plt.scatter，以 0~len(df_result) 作為 x，預測值 (黑色) 與實際值 (紅色) 作為 y。
#!=============your works starts===============!#
plt.scatter(range(len(df_result)), df_result_sort['predict'].values, color='black', s=
plt.scatter(range(len(df_result)), df_result_sort['truth'].values, color='red', s=0.5)
#!=============your works ends===============!#

plt.show()
```



In [ ]: