Analytics Vidhya JobAThon - data engineering challange

Introduction

This notebook contains the code developed for AV Jobathon challange.
Author: Ashish Agrawal
Email ID: ashish.agrawal1502@gmail.com
LinkedIn: https://www.linkedin.com/in/iashishagr

Approach

I started with loading tables first. I converted loaded data into appropriate datatype by transforming individual series. I added some other transformations to improve consistency & confirmed the same by calculating some stats for validation.

Next I defined some helper methods to prepare data as per submission file in Section 3. I call these method in Section 4 where I prepare the final submission file, populate it & finally save it.

Data preprocessing/ Data cleaning ideas

1. It was fun to work with date column (P.S. this was most challanging part :P). Eventually, separating the date values and null values & then applying datatype conversion worked. All credit goes to `mask` method.
2. Case-normalising to UPPER case helped to reduce redundancy for some columns like `Activity` and `OS`
3. Loops were *very* slow to assign values. Making whole series and updating to final table worked amazingly fast in comparison to former.

Tools Used

I used pandas for this data engineering tasks.

Future Scope of work

1. `VisitDateTime` can be imputed based on `UserID`, `webClientID` and location information.
2. `Country` column needs cleaning. At places it contains country name with city/region name.

In [1]:
```
import pandas as pd
import numpy as np
import csv
```

Loading *Visitor* Table

In [2]:
```
# loading the tables
# visitor = pd.read_csv('./data/VisitorLogsData.csv', parse_dates=[1], date_parser=date_handler)
visitor = pd.read_csv('./data/VisitorLogsData.csv')
visitor.head()
```

Out[2]:

	webClientID	VisitDateTime	ProductID	UserID	Activity	Browser	OS	City	Country
0	WI10000050298	2018-05-07 04:28:45.970	pr100631	NaN	NaN	Chrome Mobile	Android	Chennai	India
1	WI10000025922	2018-05-13 07:26:04.964	pr100707	NaN	NaN	Chrome	Windows	NaN	Taiwan
2	WI100000204522	2018-05-11 11:43:42.832	pr100030	NaN	click	Chrome	windows	Gurgaon	India
3	WI10000011974	2018-05-13 15:20:23.436	Pr100192	NaN	CLICK	Chrome	Windows		
4	WI100000441953	2018-05-08 20:44:25.238	Pr100762	NaN	click	Chrome	mac os x	Iselin	United States

Transforming VisitDateTime from multiple dates into one

In [3]:
```
# helper function for date transformation
def date_handler(col):
    '''
    It expects the dates in input column in 2 formats:
    1. One is in datetime format “2018-05-07 04:28:45.970”
    2. Another one is in unix timestamp nanosecond format “1527051855673000000”
```

```
    '''
    # converting dates of first format
    type1_dates = pd.to_datetime(col, format="%Y-%m-%d %H:%M:%S.%f", errors='coerce')

    # getting index where date in not like first format
    mask = type1_dates.isnull()

    # separating type 2 date format and nulls (if any)
    dates_with_nulls = col[mask]
    type2_dates = dates_with_nulls[~dates_with_nulls.isnull()]

    # converting type 2 date format i.e. timestamps into datetime
    type2_dates = pd.to_datetime(type2_dates.astype('int64'),unit='ns')

    # replacing null values in type1_dates with corresponding row value from type2_dates
    type1_dates = type1_dates.mask(pd.isnull, type2_dates)

    return type1_dates
```

In [4]:

```
# getting number of nulls before transformation
before_null = visitor.VisitDateTime.isnull()
print(before_null.sum())

# transforming VisitDateTime column
visitor.VisitDateTime=date_handler(visitor.VisitDateTime)

# getting number of nulls after transformation
after_null = visitor.VisitDateTime.isnull()
print(after_null.sum())

# comparing both before and after nulls
compare = (before_null == after_null)
print(compare.sum())
```

```
658915
658915
6588000
```

Hence, we have converted all the initially given not-null date values into datetime format, leaving existing null values as it is.

In [5]:

```
# getting max and min date
print(visitor.VisitDateTime.max())
print(visitor.VisitDateTime.min())
```

```
2018-05-27 23:59:59.576000
2018-05-07 00:00:01.419000
```

This min max time values are in accordance with data description

Transforming Activity into upper case

In [6]:
```
visitor.Activity = visitor.Activity.str.upper()
```

Transforming ProductID into upper case

In [7]:
```
visitor.ProductID = visitor.ProductID.str.upper()
```

Transforming OS into upper case

In [8]:
```
visitor.OS = visitor.OS.str.upper()
```

Informative Stats

In [9]:
```
# Datatypes
visitor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6588000 entries, 0 to 6587999
Data columns (total 9 columns):
 #   Column         Dtype
---  ------         -----
 0   webClientID    object
 1   VisitDateTime  datetime64[ns]
 2   ProductID      object
 3   UserID         object
 4   Activity       object
 5   Browser        object
 6   OS             object
 7   City           object
```

```
 8   Country        object
dtypes: datetime64[ns](1), object(8)
memory usage: 452.4+ MB
```

In [10]:
```
# Nulls
visitor.isnull().sum()
```

Out[10]:
```
webClientID           0
VisitDateTime     658915
ProductID         527137
UserID           5937305
Activity          889446
Browser                0
OS                     0
City             2165831
Country           397693
dtype: int64
```

In [11]:
```
# Unique user IDs
visitor.UserID.unique().shape
```

Out[11]:
```
(34051,)
```

In [12]:
```
# Unique activity
visitor.Activity.value_counts(dropna=False)
```

Out[12]:
```
CLICK       3800629
PAGELOAD    1897925
NaN          889446
Name: Activity, dtype: int64
```

In [13]:
```
# Unique OS
visitor.OS.value_counts(dropna=False)
```

Out[13]:
```
WINDOWS          4245913
ANDROID           937713
MAC OS X          829915
LINUX             333623
IOS               135357
UBUNTU             90184
CHROME OS          10091
```

```
FEDORA              4270
OTHER                796
WINDOWS PHONE         46
TIZEN                 34
FREEBSD               28
SOLARIS                6
OPENBSD                6
NETBSD                 6
CHROMECAST             6
KINDLE                 3
BLACKBERRY OS          3
Name: OS, dtype: int64
```

In [14]:
```
# checking for country
visitor.Country.nunique()
```

Out[14]:
```
18914
```

In [15]:
```
visitor.Country.sample(10)
```

Out[15]:
```
6586059                      India
1268529                      India
5665547                    Morocco
5288453              Brazil Recife
72902                        China
3643736    United Kingdom Edinburgh
1770474               Turkey Izmir
2664343                  Singapore
3599895                      India
4605309     Taiwan Taoyuan District
Name: Country, dtype: object
```

This column needs cleaning as there can be only 193 countries

In [16]:
```
# checking for common webclient ids among records with & without user id

cidwithusers = visitor[~visitor.UserID.isnull()].webClientID.to_list()
cidwithoutusers = visitor[visitor.UserID.isnull()].webClientID.to_list()

z=np.intersect1d(cidwithusers,cidwithoutusers)

print(len(z))
```

```
0
```

If this is 0, then there is no any prior activity of any registered user

---

Loading *User* Table

In [17]:
```
user = pd.read_csv('data/userTable.csv', parse_dates=[1], infer_datetime_format=True)
user.head()
```

Out[17]:

	UserID	Signup Date	User Segment
0	U133159	2018-04-14 07:01:16.202607+00:00	C
1	U129368	2017-12-02 09:38:41.584270+00:00	B
2	U109654	2013-03-19 11:38:55+00:00	B
3	U108998	2018-01-18 08:29:51.627954+00:00	C
4	U131393	2018-03-27 08:05:28.806800+00:00	B

Informative Stats

In [18]:
```
# Datatypes
user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34050 entries, 0 to 34049
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   UserID        34050 non-null  object
 1   Signup Date   34050 non-null  datetime64[ns, UTC]
 2   User Segment  34050 non-null  object
dtypes: datetime64[ns, UTC](1), object(2)
memory usage: 798.2+ KB
```

In [19]:
```
# Nulls
user.isnull().sum()
```

Out[19]:
```
UserID          0
Signup Date     0
User Segment    0
dtype: int64
```

In [20]:
```
# Unique user IDs
user.UserID.unique().shape
```

Out[20]:
```
(34050,)
```

In [21]:
```
# Max/min vintage
print(user['Signup Date'].max())
print(user['Signup Date'].min())
```

```
2018-05-20 07:06:47.531020+00:00
2010-12-01 03:11:51+00:00
```

In [22]:
```
# Getting timezone information
user['Signup Date'].dt.tz
```

Out[22]:
```
<UTC>
```

All of the sign up dates have same timezone - UTC

Helper Data preparation methods

In [23]:
```
merged = user.merge(visitor, on='UserID')
```

In [24]:
```
merged.columns
```

Out[24]:
```
Index(['UserID', 'Signup Date', 'User Segment', 'webClientID', 'VisitDateTime',
       'ProductID', 'Activity', 'Browser', 'OS', 'City', 'Country'],
      dtype='object')
```

In [25]:

```
def get_user_visits(user_table, visitor_table):
    date_7_days_ago = pd.Timestamp(2018,5,21)
    merged = user_table.merge(visitor_table, on='UserID').copy()
    last_7_days_activity_dates = merged[merged.VisitDateTime >= date_7_days_ago][['UserID','VisitDateTime']]
    last_7_days_activity_dates.VisitDateTime=last_7_days_activity_dates.VisitDateTime.map(lambda t: t.date())
    user_visits = last_7_days_activity_dates.groupby('UserID')['VisitDateTime'].nunique()
    return user_visits
```

In [26]:

```
def get_product_visits(user_table, visitor_table):
    date_15_days_ago = pd.Timestamp(2018,5,13)
    merged = user_table.merge(visitor_table, on='UserID').copy()
    last_15_days_productvisits = merged[merged.VisitDateTime >= date_15_days_ago][['UserID','ProductID']]
    product_visits = last_15_days_productvisits.groupby('UserID')['ProductID'].nunique()
    return product_visits
```

In [27]:

```
def get_user_vintage(user_table):
    user_table= user_table.copy()
    today = pd.Timestamp(2018,5,28)
    user_table['vintage'] =  user_table['Signup Date'].map(lambda t: (today.date()-t.date()).days)
    user_table.drop(columns=['User Segment','Signup Date'], inplace=True)
    user_table.set_index(['UserID'], inplace=True)
    return user_table
```

In [28]:

```
def get_most_viewed_product(user_table, visitor_table):
    date_15_days_ago = pd.Timestamp(2018,5,13)
    merged = user_table.merge(visitor_table, on='UserID').copy()
    last_15_days_vp = merged[(merged.VisitDateTime >= date_15_days_ago) & (merged.Activity=='PAGELOAD')][['UserID','ProductID','Vi
    last_15_days_vp['idx'] = last_15_days_vp.index
    product_views_count = last_15_days_vp.pivot_table(values=['idx','VisitDateTime'], index=['UserID','ProductID'], aggfunc={'idx'
    mvp_count = product_views_count.reset_index().sort_values(['UserID', 'idx','VisitDateTime'], ascending=[True, False, False])
    mvp_count.drop_duplicates(['UserID'], inplace=True)
    mvp_count.drop(columns=['idx','VisitDateTime'], inplace=True)
    mvp_count.set_index(['UserID'], inplace=True)
    return mvp_count
```

In [29]:

```
def get_most_active_os(user_table, visitor_table):
    merged = user_table.merge(visitor_table, on='UserID').copy()
    user_os = merged[['UserID','OS','VisitDateTime']]
    user_os['idx'] = user_os.index
```

```
    os_count = user_os.pivot_table(values=['idx','VisitDateTime'], index=['UserID','OS'], aggfunc={'idx':'count', 'VisitDateTime':
    maos_count = os_count.reset_index().sort_values(['UserID', 'idx','VisitDateTime'], ascending=[True, False, False])
    maos_count.drop_duplicates(['UserID'], inplace=True)
    maos_count.drop(columns=['idx','VisitDateTime'], inplace=True)
    maos_count.set_index(['UserID'], inplace=True)
    return maos_count
```

In [30]:
```
def get_recently_viewed_product(user_table, visitor_table):
    merged = user_table.merge(visitor_table, on='UserID').copy()
    product_views = merged[(merged.Activity=='PAGELOAD')][['UserID','ProductID','VisitDateTime']]
    rvp_count = product_views.sort_values(['UserID', 'VisitDateTime'], ascending=[True, False])
    rvp_count.drop_duplicates(['UserID'], inplace=True)
    rvp_count.drop(columns=['VisitDateTime'], inplace=True)
    rvp_count.set_index(['UserID'], inplace=True)
    return rvp_count
```

In [31]:
```
def get_user_pageloads(user_table, visitor_table):
    date_7_days_ago = pd.Timestamp(2018,5,21)
    merged = user_table.merge(visitor_table, on='UserID').copy()
    user_pageloads = merged[(merged.VisitDateTime >= date_7_days_ago) & (merged.Activity=='PAGELOAD')][['UserID','Activity']]
    user_pageload_count = user_pageloads.groupby('UserID')['Activity'].count()
    return user_pageload_count
```

In [32]:
```
def get_user_clicks(user_table, visitor_table):
    date_7_days_ago = pd.Timestamp(2018,5,21)
    merged = user_table.merge(visitor_table, on='UserID').copy()
    user_clicks = merged[(merged.VisitDateTime >= date_7_days_ago) & (merged.Activity=='CLICK')][['UserID','Activity']]
    user_click_count = user_clicks.groupby('UserID')['Activity'].count()
    return user_click_count
```

Preparing final *Submission* Table

Populating final *Submission* Table

In [33]:
```
submission = pd.read_csv('./submission_format.csv')
```

```
submission.head()
submission['UserIDidx'] = submission.UserID
submission.set_index('UserIDidx', inplace=True)
```

In [34]:
```
submission.head()
```

Out[34]:

UserIDidx	UserID	No_of_days_Visited_7_Days	No_Of_Products_Viewed_15_Days	User_Vintage	Most_Viewed_product_15_Days	Most_Active_OS	Recently_Vi
U100002	U100002	NaN	NaN	NaN	NaN	NaN	
U100003	U100003	NaN	NaN	NaN	NaN	NaN	
U100004	U100004	NaN	NaN	NaN	NaN	NaN	
U100005	U100005	NaN	NaN	NaN	NaN	NaN	
U100006	U100006	NaN	NaN	NaN	NaN	NaN	

In [35]:
```
# filling value for No_of_days_Visited_7_Days
submission.No_of_days_Visited_7_Days = get_user_visits(user,visitor)
submission.No_of_days_Visited_7_Days.fillna(0, inplace=True)
submission = submission.astype({'No_of_days_Visited_7_Days':'Int64'})

# asserting values are between 0-7
print(submission.No_of_days_Visited_7_Days.value_counts())
```

```
0    17878
1    10692
2     3217
3     1253
4      555
5      261
6      132
7       62
Name: No_of_days_Visited_7_Days, dtype: Int64
```

In [36]:
```
# filling value for No_Of_Products_Viewed_15_Days
submission.No_Of_Products_Viewed_15_Days = get_product_visits(user,visitor)
submission.No_Of_Products_Viewed_15_Days.fillna(0, inplace=True)
```

```
submission = submission.astype({'No_Of_Products_Viewed_15_Days':'Int64'})

print(submission.No_Of_Products_Viewed_15_Days.value_counts())
```

```
1     11832
0      8609
2      4843
3      2836
4      1640
5      1127
6       770
7       520
8       393
9       284
10      240
11      179
13      117
12      115
14       84
15       57
16       52
18       49
17       44
19       34
21       26
20       20
22       16
24       16
26       14
27       14
29       13
23       12
25       11
31       10
28        8
33        8
36        6
32        4
35        4
37        4
38        3
34        3
44        3
43        3
40        3
41        3
66        2
```

```
46        2
42        2
49        2
68        1
47        1
54        1
69        1
39        1
97        1
65        1
77        1
50        1
64        1
61        1
30        1
83        1
Name: No_Of_Products_Viewed_15_Days, dtype: Int64
```

In [37]:

```
# filling value for User_Vintage
submission.User_Vintage = get_user_vintage(user)
submission.User_Vintage.fillna(0, inplace=True)
submission = submission.astype({'User_Vintage':'Int64'})

print(submission.User_Vintage.value_counts())
```

```
53      311
48      308
49      303
47      302
59      293
       ...
2017      1
2051      1
2271      1
2085      1
2047      1
Name: User_Vintage, Length: 1998, dtype: Int64
```

In [38]:

```
# filling value for Most_Viewed_product_15_Days
submission.Most_Viewed_product_15_Days = get_most_viewed_product(user, visitor)
submission.Most_Viewed_product_15_Days.fillna('Product101', inplace=True)

print(submission.Most_Viewed_product_15_Days.value_counts())
```

```
Product101    12396
```

```
PR100017       1331
PR100166        764
PR100102        760
PR100390        499
               ...
PR100971          1
PR106397          1
PR105454          1
PR103383          1
PR103997          1
Name: Most_Viewed_product_15_Days, Length: 1885, dtype: int64
```

In [39]:

```
# filling value for Most_Active_OS
submission.Most_Active_OS = get_most_active_os(user, visitor)
submission.Most_Active_OS.fillna(submission.Most_Active_OS.mode(), inplace=True)

print(submission.Most_Active_OS.value_counts())
```

```
C:\Users\ashis\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
WINDOWS      20093
ANDROID       9798
MAC OS X      2426
LINUX          926
IOS            486
UBUNTU         295
CHROME OS       14
FEDORA          12
Name: Most_Active_OS, dtype: int64
```

In [40]:

```
# filling value for Recently_Viewed_Product
submission.Recently_Viewed_Product = get_recently_viewed_product(user, visitor)
submission.Recently_Viewed_Product.fillna('Product101', inplace=True)

print(submission.Recently_Viewed_Product.value_counts())
```

```
Product101    6056
PR100017      1294
PR100102      1201
PR100166       692
```

```
PR100200       549
               ...
PR102440         1
PR110569         1
PR102679         1
PR108029         1
PR101431         1
Name: Recently_Viewed_Product, Length: 2129, dtype: int64
```

In [41]:

```
# filling value for Pageloads_last_7_days
submission.Pageloads_last_7_days = get_user_pageloads(user,visitor)
submission.Pageloads_last_7_days.fillna(0, inplace=True)
submission = submission.astype({'Pageloads_last_7_days':'Int64'})

print(submission.Pageloads_last_7_days.value_counts())
```

```
0      20236
1       5934
2       2673
3       1502
4        873
       ...
89         1
313        1
122        1
93         1
95         1
Name: Pageloads_last_7_days, Length: 93, dtype: Int64
```

In [42]:

```
# filling value for Clicks_last_7_days
submission.Clicks_last_7_days = get_user_clicks(user,visitor)
submission.Clicks_last_7_days.fillna(0, inplace=True)
submission = submission.astype({'Clicks_last_7_days':'Int64'})

print(submission.Clicks_last_7_days.value_counts())
```

```
0      23441
1       2872
2       1620
3       1080
4        800
       ...
342        1
252        1
508        1
```

```
129        1
511        1
Name: Clicks_last_7_days, Length: 201, dtype: Int64
```

Checking submission file for nulls

In [43]:
```
submission.isnull().sum()
```

Out[43]:
```
UserID                          0
No_of_days_Visited_7_Days       0
No_Of_Products_Viewed_15_Days   0
User_Vintage                    0
Most_Viewed_product_15_Days     0
Most_Active_OS                  0
Recently_Viewed_Product         0
Pageloads_last_7_days           0
Clicks_last_7_days              0
dtype: int64
```

Exporting submission file

In [44]:
```
from datetime import datetime
```

In [45]:
```
name = 'sub_{0}.csv'.format(datetime.strftime(datetime.now(),"%Y-%m-%d_%H_%M_%S"))
```

In [46]:
```
submission.to_csv(name, index=False)
```