Final project-1

December 13, 2019

1 Final Project: Predicting Red Hat Business Value

1.1 Friday section

Lu Wang 6568145789

Xinrong Lian 8266864389

Chenwei Yuan 7769615182

Xi gao 5366692125

[1]:
```
import numpy as np
import pandas as pd
from datetime import datetime
import seaborn as sns
sns.set(style="whitegrid", font_scale=1.5)
import matplotlib.pyplot as plt
# this allows plots to appear directly in the notebook
%matplotlib inline
```

[2]:
```
act_train_df = pd.read_csv('./act_train.csv', dtype={'people_id': np.str,␣
↪'activity_id': np.str, 'outcome': np.int8},
                           parse_dates=['date'])
act_test_df = pd.read_csv('./act_test.csv', dtype={'people_id': np.str,␣
↪'activity_id': np.str},
                            parse_dates=['date'])
ppl_df = pd.read_csv('./people.csv', dtype={'people_id': np.str, 'activity_id':␣
↪np.str, 'char_38': np.int32},
                       parse_dates=['date'])
```

[3]:
```
act_train_df.shape, act_test_df.shape, ppl_df.shape
```

[3]:
```
((2197291, 15), (498687, 14), (189118, 41))
```

[4]:
```
act_train_df.head()
```

[4]:
```
  people_id   activity_id       date activity_category char_1 char_2 char_3  \
0   ppl_100  act2_1734928 2023-08-26            type 4    NaN    NaN    NaN
```

```
1   ppl_100  act2_2434093 2022-09-27            type 2    NaN    NaN    NaN
2   ppl_100  act2_3404049 2022-09-27            type 2    NaN    NaN    NaN
3   ppl_100  act2_3651215 2023-08-04            type 2    NaN    NaN    NaN
4   ppl_100  act2_4109017 2023-08-26            type 2    NaN    NaN    NaN

  char_4 char_5 char_6 char_7 char_8 char_9  char_10  outcome
0    NaN    NaN    NaN    NaN    NaN    NaN  type 76        0
1    NaN    NaN    NaN    NaN    NaN    NaN   type 1        0
2    NaN    NaN    NaN    NaN    NaN    NaN   type 1        0
3    NaN    NaN    NaN    NaN    NaN    NaN   type 1        0
4    NaN    NaN    NaN    NaN    NaN    NaN   type 1        0
```

[5]:
```
act_test_df.head()
```

[5]:
```
    people_id   activity_id       date activity_category   char_1   char_2  \
0  ppl_100004   act1_249281 2022-07-20            type 1   type 5  type 10
1  ppl_100004   act2_230855 2022-07-20            type 5      NaN      NaN
2   ppl_10001   act1_240724 2022-10-14            type 1  type 12   type 1
3   ppl_10001    act1_83552 2022-11-27            type 1  type 20  type 10
4   ppl_10001  act2_1043301 2022-10-15            type 5      NaN      NaN

   char_3  char_4  char_5  char_6  char_7   char_8   char_9    char_10
0  type 5  type 1  type 6  type 1  type 1   type 7   type 4        NaN
1     NaN     NaN     NaN     NaN     NaN      NaN      NaN   type 682
2  type 5  type 4  type 6  type 1  type 1  type 13  type 10        NaN
3  type 5  type 4  type 6  type 1  type 1   type 5   type 5        NaN
4     NaN     NaN     NaN     NaN     NaN      NaN      NaN  type 3015
```

[6]:
```
ppl_df.head()
```

[6]:
```
    people_id  char_1      group_1  char_2       date   char_3   char_4  \
0     ppl_100  type 2  group 17304  type 2 2021-06-29   type 5   type 5
1  ppl_100002  type 2   group 8688  type 3 2021-01-06  type 28   type 9
2  ppl_100003  type 2  group 33592  type 3 2022-06-10   type 4   type 8
3  ppl_100004  type 2  group 22593  type 3 2022-07-20  type 40  type 25
4  ppl_100006  type 2   group 6534  type 3 2022-07-27  type 40  type 25

   char_5  char_6   char_7  ... char_29 char_30  char_31  char_32  char_33  \
0  type 5  type 3  type 11  ...   False    True     True    False    False
1  type 5  type 3  type 11  ...   False    True     True     True     True
2  type 5  type 2   type 5  ...   False   False     True     True     True
3  type 9  type 4  type 16  ...    True    True     True     True     True
4  type 9  type 3   type 8  ...   False   False     True    False    False

   char_34  char_35  char_36  char_37  char_38
0     True     True     True    False       36
1     True     True     True    False       76
2     True    False     True     True       99
3     True     True     True     True       76
```

```
4    False     True     True    False       84

[5 rows x 41 columns]
```

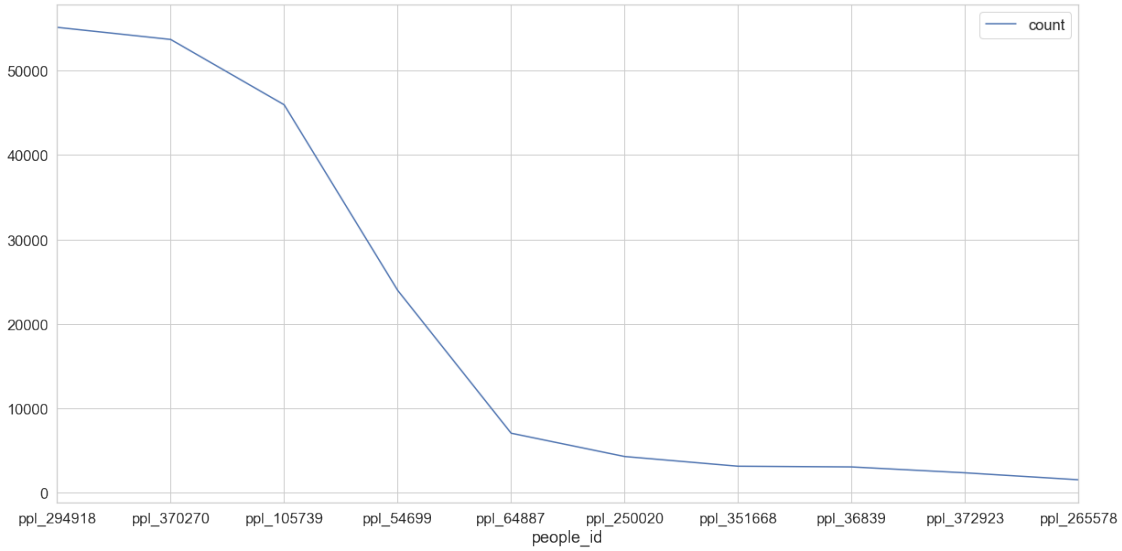
1.2 Initial Data visualization

[7]:
```
# Merging the test and train datasets with the people dataset
```

[8]:
```
train_visual = pd.merge(act_train_df, ppl_df, on='people_id')
test_visual = pd.merge(act_test_df, ppl_df, on='people_id')
```

[9]:
```
## Ten people_id with the most number of activities
ppl_distribution=pd.DataFrame()
ppl_distribution["count"]=train_visual.groupby('people_id')['outcome'].count().
 ↪sort_values(ascending=False)
ppl_distribution[:10].plot(figsize=(20, 10))
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2359a0f0>
```



[10]:
```
### Top Ten people_id with its outcome
ten_people_id=list(ppl_distribution[:10].index)
pp=[]
out=[]
for i in range(0,10):
    out.append(train_visual[train_visual["people_id"]==ten_people_id[i]].
 ↪outcome.unique()[0])
    pp.append(ten_people_id[i])
top_ten=pd.DataFrame()
top_ten["people_id"]=pp
top_ten["outcome"]=out
```

```
top_ten
# Top ten people with the most number of activities are all have big potential␣
 ↪business vlaue
```

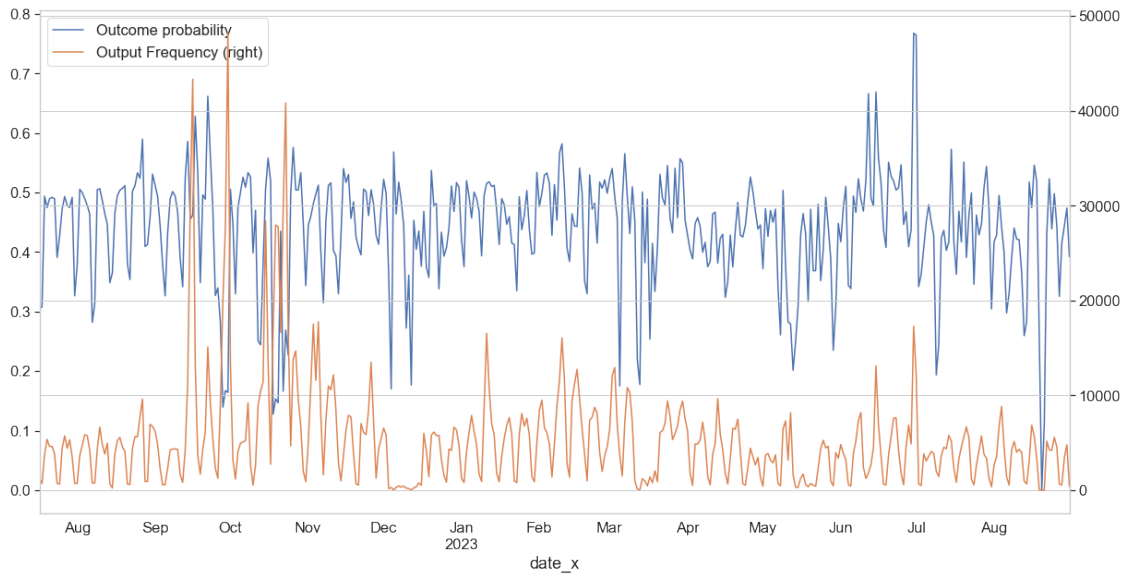
[10]:

	people_id	outcome
0	ppl_294918	0
1	ppl_370270	0
2	ppl_105739	0
3	ppl_54699	0
4	ppl_64887	1
5	ppl_250020	0
6	ppl_351668	0
7	ppl_36839	0
8	ppl_372923	0
9	ppl_265578	0

[11]:
```
# Time series plot for date_x.
```

[12]:
```
date_x = pd.DataFrame()
date_x['Outcome probability'] = train_visual.groupby('date_x')['outcome'].mean()
date_x['Output Frequency'] = train_visual.groupby('date_x')['outcome'].count()
date_x.plot(secondary_y='Output Frequency', figsize=(20, 10))
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a279a4c50>
```



[13]:
```
# Time series plot for date_y.
```

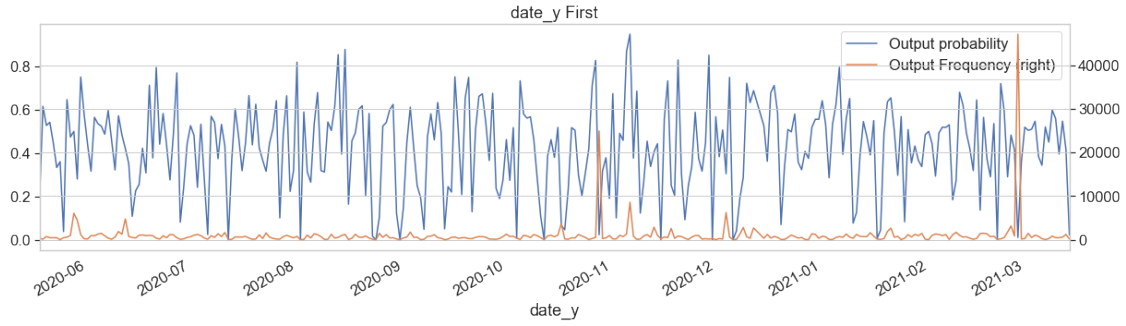
[14]:
```
# Because the time spand for date_y is too long, so we split it to four plot.
```

[15]:
```
date_y = pd.DataFrame()
date_y['Output probability'] = train_visual.groupby('date_y')['outcome'].mean()
```

```
date_y['Output Frequency'] = train_visual.groupby('date_y')['outcome'].count()
i = int(len(date_y) / 4)
```

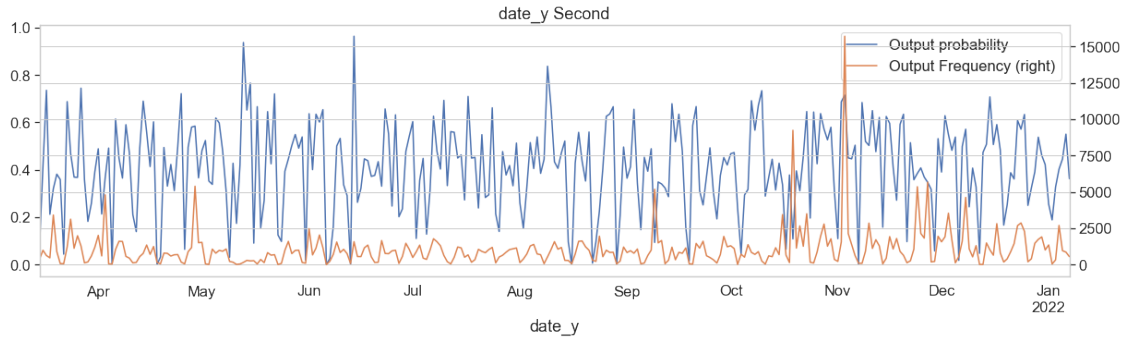
[16]:
```
date_y[:i].plot(secondary_y='Output Frequency', figsize=(20, 5), title='date_y␣
↪First')
```

[16]: <matplotlib.axes._subplots.AxesSubplot at 0x10ffe0da0>



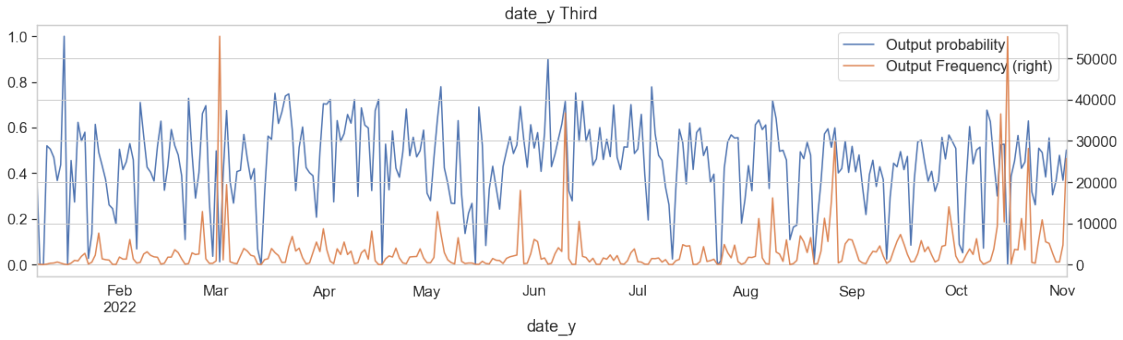
[17]:
```
date_y[i:2*i].plot(secondary_y='Output Frequency', figsize=(20, 5),␣
↪title='date_y Second')
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x10ffe93c8>



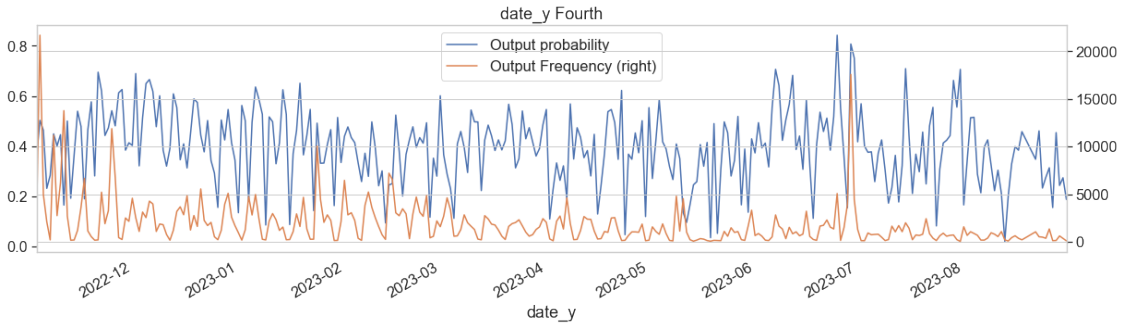
[18]:
```
date_y[2*i:3*i].plot(secondary_y='Output Frequency', figsize=(20, 5),␣
↪title='date_y Third')
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x10ffac8d0>



[19]:
```
date_y[3*i:4*i].plot(secondary_y='Output Frequency', figsize=(20, 5),␣
↪title='date_y Fourth')
```

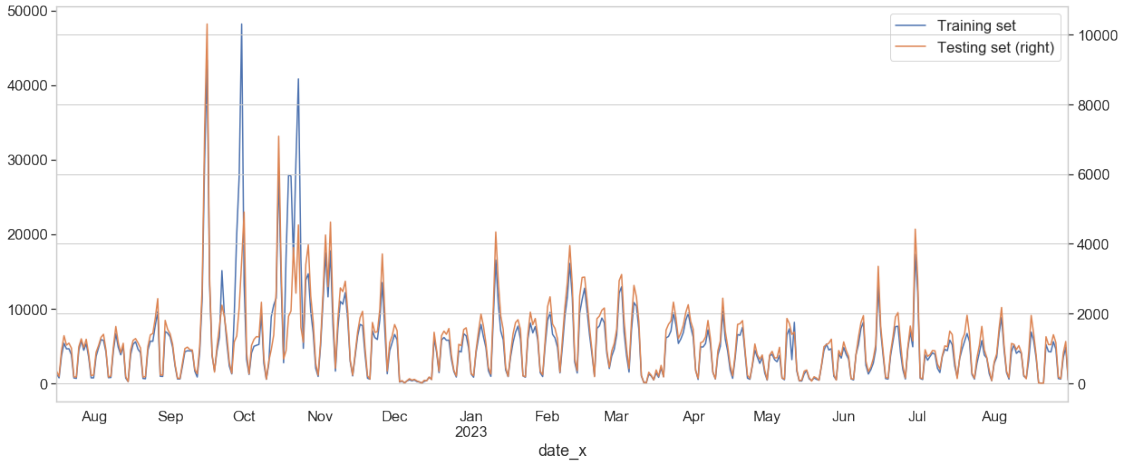
[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a25e3b668>
```



[20]:
```
#Comparison of date_x distribution between training/testing set.
```

[21]:
```
date_x_freq = pd.DataFrame()
date_x_freq['Training set'] = train_visual.groupby('date_x')['activity_id'].
↪count()
date_x_freq['Testing set'] = test_visual.groupby('date_x')['activity_id'].
↪count()
date_x_freq.plot(secondary_y='Testing set', figsize=(20, 8))
```

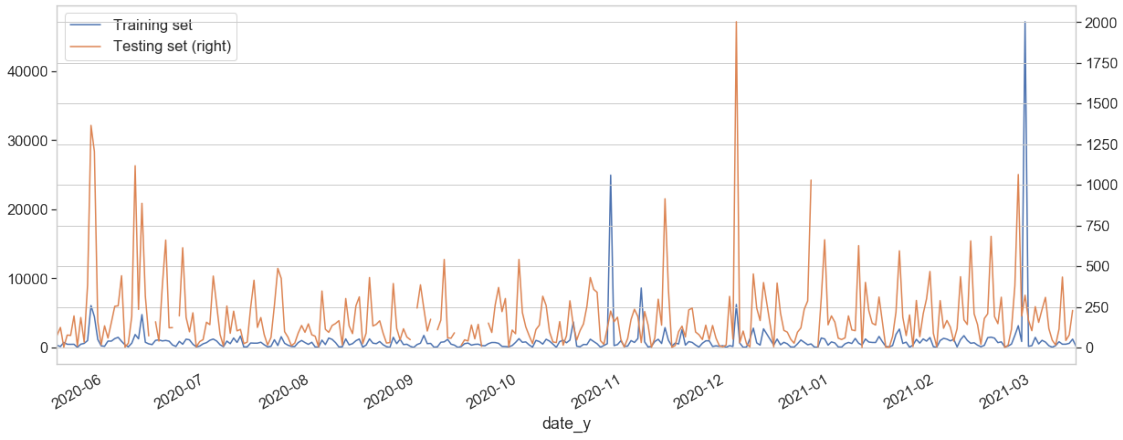
[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26094d68>
```

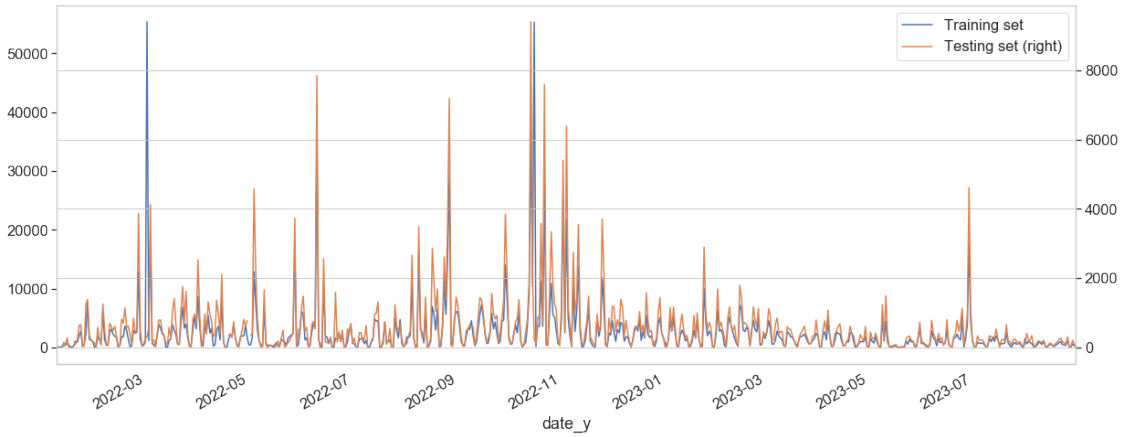


[22]:
```
#Comparison of date_y distribution between training/testing set.
```

[23]:
```
date_y_freq = pd.DataFrame()
date_y_freq['Training set'] = train_visual.groupby('date_y')['activity_id'].
 ↪count()
date_y_freq['Testing set'] = test_visual.groupby('date_y')['activity_id'].
 ↪count()
date_y_freq[:i].plot(secondary_y='Testing set', figsize=(20, 8))
date_y_freq[2*i:].plot(secondary_y='Testing set', figsize=(20, 8))
```

[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26a0e0f0>
```





1.3 Initial Date cleaning

[24]:
```
# adding year, month, day, weekend columns to replace date
```

[25]:
```
def dealing_with_date(df):
    df['year'] = df['date'].dt.year
    df['month'] = df['date'].dt.month
    df['day'] = df['date'].dt.day
    df['isweekend'] = (df['date'].dt.weekday >= 5).astype(int)
    df = df.drop('date', axis = 1)
    return df
```

[26]:
```
act_train_df = dealing_with_date(act_train_df)
act_test_df = dealing_with_date(act_test_df)
ppl_df = dealing_with_date(ppl_df)
```

[27]:
```
# Merging the test and train datasets with the people dataset
```

[28]:
```
train = act_train_df.merge(ppl_df, on='people_id', how='left', left_index=True)
test  = act_test_df.merge(ppl_df, on='people_id', how='left', left_index=True)
```

[29]:
```
train.shape, test.shape
```

[29]:
```
((2197291, 61), (498687, 60))
```

[30]:
```
# find categorical variables
```

[31]:
```
object_var = train.dtypes.loc[train.dtypes == 'object'].index
bool_var = train.dtypes.loc[train.dtypes == 'bool'].index
```

[32]:
```
object_var, bool_var
```

[32]:
```
(Index(['people_id', 'activity_id', 'activity_category', 'char_1_x', 'char_2_x',
        'char_3_x', 'char_4_x', 'char_5_x', 'char_6_x', 'char_7_x', 'char_8_x',
        'char_9_x', 'char_10_x', 'char_1_y', 'group_1', 'char_2_y', 'char_3_y',
        'char_4_y', 'char_5_y', 'char_6_y', 'char_7_y', 'char_8_y', 'char_9_y'],
```

```
      dtype='object'),
 Index(['char_10_y', 'char_11', 'char_12', 'char_13', 'char_14', 'char_15',
        'char_16', 'char_17', 'char_18', 'char_19', 'char_20', 'char_21',
        'char_22', 'char_23', 'char_24', 'char_25', 'char_26', 'char_27',
        'char_28', 'char_29', 'char_30', 'char_31', 'char_32', 'char_33',
        'char_34', 'char_35', 'char_36', 'char_37'],
       dtype='object'))
```

[33]:
```
# dealing with people_id and activity_id first
# convert them to the format computer can understand
```

[34]:
```
for df in [train, test]:
    df['people_id'] = df['people_id'].apply(lambda x: x.split('_')[1]).
↪astype(np.float64).astype(np.int32)

    df['activity_id'] = df['activity_id'].apply(lambda x: x.split('act')[1]).
↪astype(np.str)
    df['activity_id'] = df['activity_id'].apply(lambda x: x.split('_')[0]).
↪astype(np.float64).astype(np.int32)
```

[35]:
```
train.head()
```

[35]:

	people_id	activity_id	activity_category	char_1_x	char_2_x	char_3_x	\
0	100	2	type 4	NaN	NaN	NaN	
0	100	2	type 2	NaN	NaN	NaN	
0	100	2	type 2	NaN	NaN	NaN	
0	100	2	type 2	NaN	NaN	NaN	
0	100	2	type 2	NaN	NaN	NaN	

	char_4_x	char_5_x	char_6_x	char_7_x	...	char_33	char_34	char_35	char_36	\
0	NaN	NaN	NaN	NaN	...	False	True	True	True	
0	NaN	NaN	NaN	NaN	...	False	True	True	True	
0	NaN	NaN	NaN	NaN	...	False	True	True	True	
0	NaN	NaN	NaN	NaN	...	False	True	True	True	
0	NaN	NaN	NaN	NaN	...	False	True	True	True	

	char_37	char_38	year_y	month_y	day_y	isweekend_y
0	False	36	2021	6	29	0
0	False	36	2021	6	29	0
0	False	36	2021	6	29	0
0	False	36	2021	6	29	0
0	False	36	2021	6	29	0

[5 rows x 61 columns]

[36]:
```
test.head()
```

[36]:

	people_id	activity_id	activity_category	char_1_x	char_2_x	char_3_x	\
3	100004	1	type 1	type 5	type 10	type 5	
3	100004	2	type 5	NaN	NaN	NaN	

```
5      10001            1            type 1  type 12   type 1   type 5
5      10001            1            type 1  type 20  type 10   type 5
5      10001            2            type 5       NaN      NaN      NaN

  char_4_x char_5_x char_6_x char_7_x  ... char_33 char_34 char_35  char_36  \
3   type 1   type 6   type 1   type 1  ...    True    True    True     True
3      NaN      NaN      NaN      NaN  ...    True    True    True     True
5   type 4   type 6   type 1   type 1  ...    True    True    True     True
5   type 4   type 6   type 1   type 1  ...    True    True    True     True
5      NaN      NaN      NaN      NaN  ...    True    True    True     True

   char_37  char_38  year_y month_y day_y isweekend_y
3     True       76    2022       7    20           0
3     True       76    2022       7    20           0
5     True       90    2022      10    14           0
5     True       90    2022      10    14           0
5     True       90    2022      10    14           0

[5 rows x 60 columns]
```

[37]:
```
# check if people_id make a difference (take some time)
duplicate_pplID =[]
for i in train.people_id.unique():
    if (test.people_id.unique()==i).any() == True:
        duplicate_pplID.append(i)
print(duplicate_pplID)
```

```
[]
```

[38]:
```
# returns an empty list, which means the people_id in testset are totally␣
 ↪different from that in trainset
# people_id column can be droped
```

[39]:
```
train = train.drop(['people_id'], axis=1)
test = test.drop(['people_id'], axis=1)
```

[40]:
```
object_var_update = train.dtypes.loc[train.dtypes == 'object'].index
bool_var_update = train.dtypes.loc[train.dtypes == 'bool'].index
```

[41]:
```
object_var_update, bool_var_update
```

[41]:
```
(Index(['activity_category', 'char_1_x', 'char_2_x', 'char_3_x', 'char_4_x',
        'char_5_x', 'char_6_x', 'char_7_x', 'char_8_x', 'char_9_x', 'char_10_x',
        'char_1_y', 'group_1', 'char_2_y', 'char_3_y', 'char_4_y', 'char_5_y',
        'char_6_y', 'char_7_y', 'char_8_y', 'char_9_y'],
       dtype='object'),
 Index(['char_10_y', 'char_11', 'char_12', 'char_13', 'char_14', 'char_15',
        'char_16', 'char_17', 'char_18', 'char_19', 'char_20', 'char_21',
        'char_22', 'char_23', 'char_24', 'char_25', 'char_26', 'char_27',
        'char_28', 'char_29', 'char_30', 'char_31', 'char_32', 'char_33',
```

```
       'char_34', 'char_35', 'char_36', 'char_37'],
      dtype='object'))
```

[42]:
```
# check the number of classes each categorical variable has
```

[43]:
```
categorical_var_update = object_var_update.tolist() + bool_var_update.tolist()
```

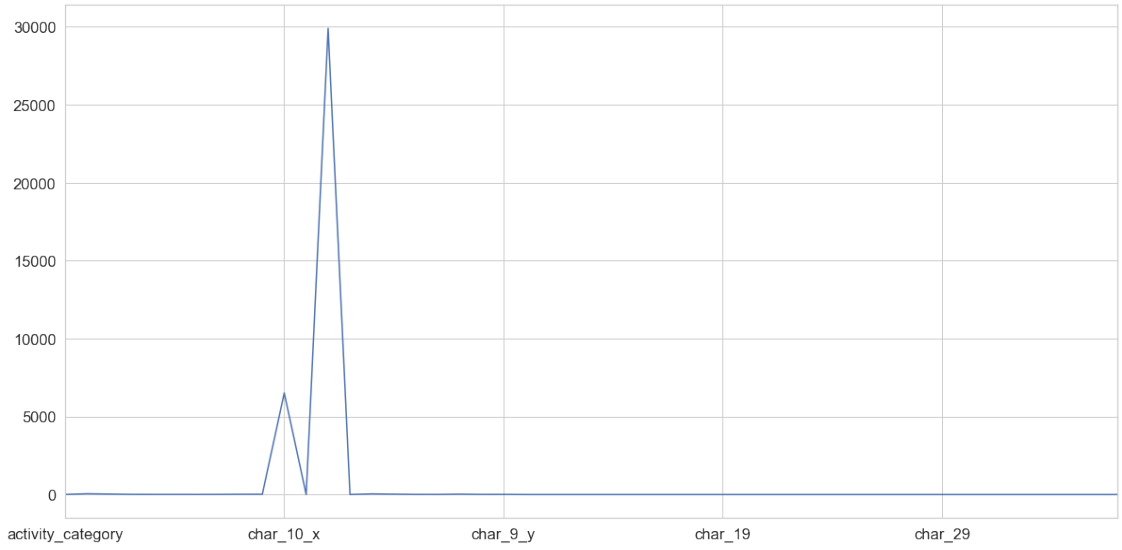
[44]:
```
train[categorical_var_update].apply(lambda x: len(x.unique()))
```

[44]:
```
activity_category        7
char_1_x                52
char_2_x                33
char_3_x                12
char_4_x                 8
char_5_x                 8
char_6_x                 6
char_7_x                 9
char_8_x                19
char_9_x                20
char_10_x             6516
char_1_y                 2
group_1              29899
char_2_y                 3
char_3_y                43
char_4_y                25
char_5_y                 9
char_6_y                 7
char_7_y                25
char_8_y                 8
char_9_y                 9
char_10_y                2
char_11                  2
char_12                  2
char_13                  2
char_14                  2
char_15                  2
char_16                  2
char_17                  2
char_18                  2
char_19                  2
char_20                  2
char_21                  2
char_22                  2
char_23                  2
char_24                  2
char_25                  2
char_26                  2
char_27                  2
char_28                  2
```

```
char_29                 2
char_30                 2
char_31                 2
char_32                 2
char_33                 2
char_34                 2
char_35                 2
char_36                 2
char_37                 2
dtype: int64
```

[45]:
```
train[categorical_var_update].apply(lambda x: len(x.unique())).
↪plot(figsize=(20,10))
```

[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a273e6c18>
```



[46]:
```
# check the class distribution of each categorical variables (plot: to do)
```

[47]:
```
train.char_10_x.value_counts(), test.char_10_x.value_counts()
```

[47]:
```
(type 1       904683
 type 23      200408
 type 2       116191
 type 61       35417
 type 452      23513
 type 489      23471
 type 52       19515
 type 481      18019
 type 433      17282
 type 8        16112
 type 3        14139
```

```
type 450      12824
type 649      11630
type 899      11427
type 400      10569
type 464      10368
type 55        8072
type 248       7860
type 257       7349
type 420       6719
type 201       6574
type 297       5145
type 600       4998
type 1058      4993
type 143       4760
type 1069      4624
type 110       4253
type 230       3875
type 1251      3798
type 585       3710
              ...
type 7692         1
type 4393         1
type 8417         1
type 7367         1
type 5702         1
type 8605         1
type 7166         1
type 4598         1
type 7103         1
type 8160         1
type 3942         1
type 8047         1
type 4477         1
type 5935         1
type 5140         1
type 4617         1
type 7118         1
type 7513         1
type 2072         1
type 6966         1
type 2922         1
type 8410         1
type 8405         1
type 2963         1
type 3745         1
type 9176         1
type 7972         1
```

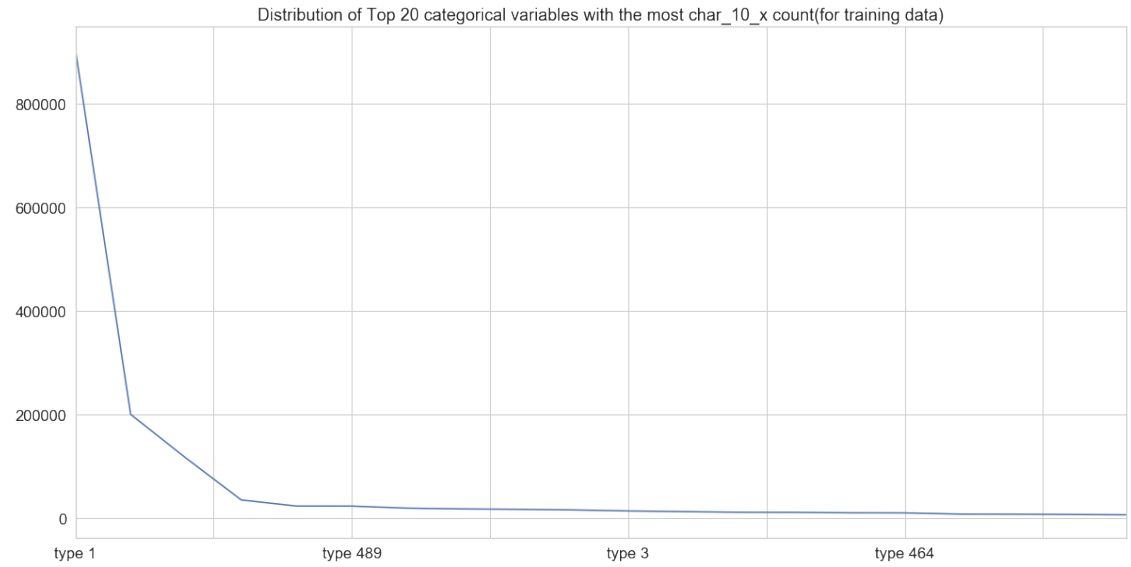
```
type 2179         1
type 2721         1
type 5558         1
Name: char_10_x, Length: 6515, dtype: int64, type 1       223164
type 2        30019
type 61        8667
type 452       6618
type 489       6284
type 52        4762
type 433       4592
type 481       4333
type 8         3806
type 3         3521
type 450       3505
type 464       2673
type 400       2532
type 649       2524
type 899       2416
type 23        2396
type 55        2123
type 248       1893
type 420       1717
type 201       1682
type 257       1642
type 1251      1331
type 297       1213
type 143       1139
type 230       1111
type 600       1087
type 110       1051
type 1058      1046
type 1069      1010
type 585        952
               ...
type 3892         1
type 540          1
type 8135         1
type 5934         1
type 1924         1
type 7144         1
type 2909         1
type 3577         1
type 1139         1
type 3882         1
type 1614         1
type 1627         1
type 7741         1
```

```
type 4369          1
type 8348          1
type 8484          1
type 3635          1
type 1148          1
type 4712          1
type 6562          1
type 1539          1
type 2261          1
type 4585          1
type 3992          1
type 2267          1
type 6790          1
type 4327          1
type 8281          1
type 8475          1
type 4502          1
Name: char_10_x, Length: 3961, dtype: int64)
```

[48]:
```
# Distribution of Top 20 categorical variables with the most char_10_x␣
↪count(for training data).
```

[49]:
```
train.char_10_x.value_counts().sort_values(ascending=False)[:20].
↪plot(figsize=(20,10),title="Distribution of Top 20 categorical variables␣
↪with the most char_10_x count(for training data)")
```

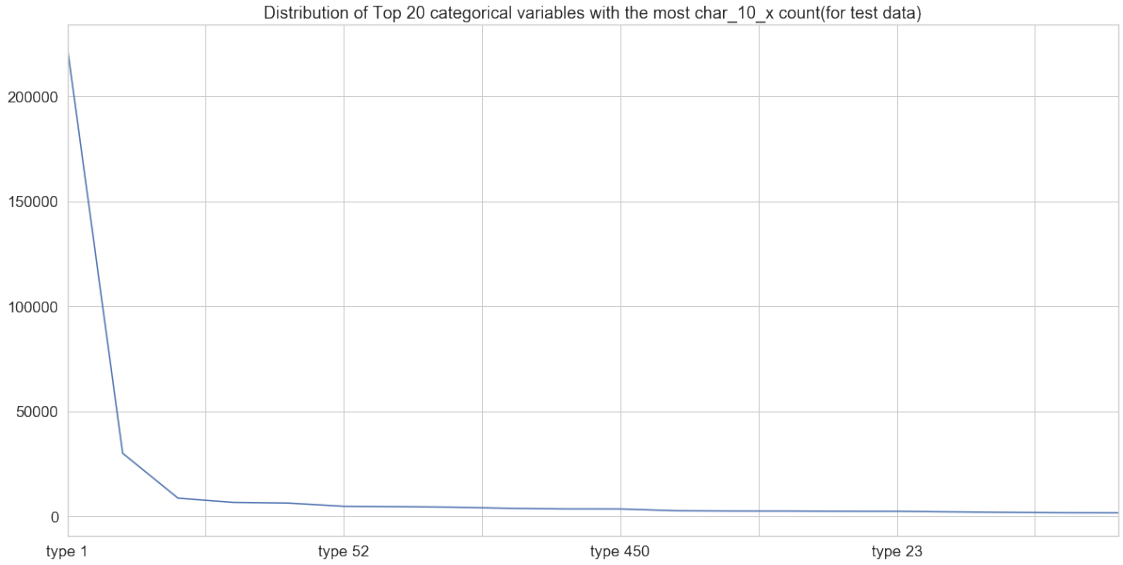
[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3e62e1d0>
```



[50]:
```
# Distribution of Top 20 categorical variables with the most char_10_x␣
↪count(for test data).
```

[51]:
```
test.char_10_x.value_counts().sort_values(ascending=False)[:20].
↪plot(figsize=(20,10),title="Distribution of Top 20 categorical variables␣
↪with the most char_10_x count(for test data)")
```

[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2742dbe0>
```



[52]:
```
# oberservation: char_10_x and group_1 has too many classes (may consider drop␣
↪them later)
# with highly imbalanced class distribution
```

[53]:
```
# dealing with missing values
```

[54]:
```
train.isnull().sum()
```

[54]:
```
activity_id              0
activity_category        0
char_1_x           2039676
char_2_x           2039676
char_3_x           2039676
char_4_x           2039676
char_5_x           2039676
char_6_x           2039676
char_7_x           2039676
char_8_x           2039676
char_9_x           2039676
char_10_x           157615
outcome                  0
year_x                   0
month_x                  0
day_x                    0
isweekend_x              0
```

```
char_1_y                0
group_1                 0
char_2_y                0
char_3_y                0
char_4_y                0
char_5_y                0
char_6_y                0
char_7_y                0
char_8_y                0
char_9_y                0
char_10_y               0
char_11                 0
char_12                 0
char_13                 0
char_14                 0
char_15                 0
char_16                 0
char_17                 0
char_18                 0
char_19                 0
char_20                 0
char_21                 0
char_22                 0
char_23                 0
char_24                 0
char_25                 0
char_26                 0
char_27                 0
char_28                 0
char_29                 0
char_30                 0
char_31                 0
char_32                 0
char_33                 0
char_34                 0
char_35                 0
char_36                 0
char_37                 0
char_38                 0
year_y                  0
month_y                 0
day_y                   0
isweekend_y             0
dtype: int64
```

[55]:
```
test.isnull().sum()
```

[55]:
```
activity_id              0
activity_category        0
char_1_x            458595
char_2_x            458595
char_3_x            458595
char_4_x            458595
char_5_x            458595
char_6_x            458595
char_7_x            458595
char_8_x            458595
char_9_x            458595
char_10_x            40092
year_x                   0
month_x                  0
day_x                    0
isweekend_x              0
char_1_y                 0
group_1                  0
char_2_y                 0
char_3_y                 0
char_4_y                 0
char_5_y                 0
char_6_y                 0
char_7_y                 0
char_8_y                 0
char_9_y                 0
char_10_y                0
char_11                  0
char_12                  0
char_13                  0
char_14                  0
char_15                  0
char_16                  0
char_17                  0
char_18                  0
char_19                  0
char_20                  0
char_21                  0
char_22                  0
char_23                  0
char_24                  0
char_25                  0
char_26                  0
char_27                  0
char_28                  0
char_29                  0
char_30                  0
```

```
char_31                 0
char_32                 0
char_33                 0
char_34                 0
char_35                 0
char_36                 0
char_37                 0
char_38                 0
year_y                  0
month_y                 0
day_y                   0
isweekend_y             0
dtype: int64
```

[56]:
```
nan_columns = ['char_1_x','char_2_x','char_3_x','char_4_x','char_5_x',
               'char_6_x','char_7_x','char_8_x','char_9_x','char_10_x']
for col in nan_columns:
    print(train[col].unique())
```

```
[nan 'type 3' 'type 36' 'type 24' 'type 2' 'type 5' 'type 12' 'type 23'
 'type 7' 'type 1' 'type 10' 'type 29' 'type 8' 'type 16' 'type 26'
 'type 15' 'type 17' 'type 13' 'type 41' 'type 11' 'type 9' 'type 25'
 'type 6' 'type 4' 'type 19' 'type 20' 'type 30' 'type 14' 'type 28'
 'type 22' 'type 35' 'type 40' 'type 33' 'type 43' 'type 18' 'type 27'
 'type 39' 'type 32' 'type 47' 'type 31' 'type 38' 'type 42' 'type 34'
 'type 21' 'type 49' 'type 46' 'type 37' 'type 44' 'type 50' 'type 48'
 'type 52' 'type 45']
[nan 'type 5' 'type 11' 'type 6' 'type 2' 'type 1' 'type 16' 'type 14'
 'type 4' 'type 8' 'type 3' 'type 10' 'type 25' 'type 26' 'type 9'
 'type 19' 'type 13' 'type 7' 'type 12' 'type 29' 'type 17' 'type 15'
 'type 18' 'type 20' 'type 21' 'type 24' 'type 22' 'type 27' 'type 23'
 'type 28' 'type 31' 'type 32' 'type 30']
[nan 'type 1' 'type 5' 'type 6' 'type 3' 'type 7' 'type 8' 'type 4'
 'type 9' 'type 2' 'type 10' 'type 11']
[nan 'type 1' 'type 3' 'type 2' 'type 4' 'type 6' 'type 5' 'type 7']
[nan 'type 6' 'type 1' 'type 5' 'type 2' 'type 4' 'type 3' 'type 7']
[nan 'type 3' 'type 1' 'type 2' 'type 4' 'type 5']
[nan 'type 3' 'type 1' 'type 4' 'type 2' 'type 5' 'type 6' 'type 7'
 'type 8']
[nan 'type 6' 'type 4' 'type 5' 'type 9' 'type 18' 'type 14' 'type 7'
 'type 3' 'type 8' 'type 11' 'type 1' 'type 13' 'type 10' 'type 15'
 'type 2' 'type 16' 'type 12' 'type 17']
[nan 'type 8' 'type 1' 'type 2' 'type 7' 'type 13' 'type 9' 'type 15'
 'type 4' 'type 6' 'type 3' 'type 12' 'type 10' 'type 17' 'type 18'
 'type 14' 'type 5' 'type 16' 'type 19' 'type 11']
['type 76' 'type 1' 'type 1727' ... 'type 7356' 'type 6865' 'type 7379']
```

[57]:
```
# have the insight to fill the NAN with 'type 0'
```

[58]:
```
# encoding of categorical features
```

[59]:
```
categorical_var = object_var_update.tolist() + bool_var_update.tolist()
```

[60]:
```
for df in [train, test]:
    for col in categorical_var:
        if df[col].dtype == 'object':
            df[col].fillna('type 0', inplace=True)
            df[col] = df[col].apply(lambda x: x.split(' ')[1]).astype(np.int32)
        elif df[col].dtype == 'bool':
            df[col] = df[col].astype(np.int8)
```

[61]:
```
train.head()
```

[61]:

	activity_id	activity_category	char_1_x	char_2_x	char_3_x	char_4_x	\
0	2	4	0	0	0	0	
0	2	2	0	0	0	0	
0	2	2	0	0	0	0	
0	2	2	0	0	0	0	
0	2	2	0	0	0	0	

	char_5_x	char_6_x	char_7_x	char_8_x	...	char_33	char_34	char_35	\
0	0	0	0	0	...	0	1	1	
0	0	0	0	0	...	0	1	1	
0	0	0	0	0	...	0	1	1	
0	0	0	0	0	...	0	1	1	
0	0	0	0	0	...	0	1	1	

	char_36	char_37	char_38	year_y	month_y	day_y	isweekend_y
0	1	0	36	2021	6	29	0
0	1	0	36	2021	6	29	0
0	1	0	36	2021	6	29	0
0	1	0	36	2021	6	29	0
0	1	0	36	2021	6	29	0

[5 rows x 60 columns]

[62]:
```
test.head()
```

[62]:

	activity_id	activity_category	char_1_x	char_2_x	char_3_x	char_4_x	\
3	1	1	5	10	5	1	
3	2	5	0	0	0	0	
5	1	1	12	1	5	4	
5	1	1	20	10	5	4	
5	2	5	0	0	0	0	

	char_5_x	char_6_x	char_7_x	char_8_x	...	char_33	char_34	char_35	\
3	6	1	1	7	...	1	1	1	
3	0	0	0	0	...	1	1	1	
5	6	1	1	13	...	1	1	1	

```
5          6          1          1          5  ...          1          1          1
5          0          0          0          0  ...          1          1          1

   char_36  char_37  char_38  year_y  month_y  day_y  isweekend_y
3        1        1       76    2022        7     20            0
3        1        1       76    2022        7     20            0
5        1        1       90    2022       10     14            0
5        1        1       90    2022       10     14            0
5        1        1       90    2022       10     14            0

[5 rows x 59 columns]
```

[63]:
```
# save cleaned data as train, test file for modeling later
```

[64]:
```
train.to_csv('./processing_data/train.csv', sep=',', header=True, index=False)
test.to_csv('./processing_data/test.csv', sep=',', header=True, index=False)
```

1.4 Cleaned Data Visualization

[66]:
```
train[['activity_category', 'outcome']].groupby(['activity_category'],␣
↪as_index=False).mean().sort_values(by='activity_category', ascending=False)
```

[66]:
```
   activity_category   outcome
6                  7  0.399747
5                  6  0.555843
4                  5  0.480243
3                  4  0.489205
2                  3  0.251989
1                  2  0.510324
0                  1  0.411325
```

[67]:
```
g = sns.FacetGrid(train, col='activity_category')
g.map(plt.hist, 'outcome', bins=20)
```

[67]:
```
<seaborn.axisgrid.FacetGrid at 0x1a26456b00>
```