dataset_analyzis

June 20, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%load_ext autoreload
```

1 Load Data

[2]:
```
test_data = pd.read_csv('data/test_v2.csv')
train_data = pd.read_csv('data/train_v2.csv')
```

2 Data Pipeline

[3]:
```
%autoreload
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from include.DatetimeFromTimestamp import DatetimeFromTimestamp
from include.DataFrameDropper import DataFrameDropper
from include.DataFrameSelector import DataFrameSelector
from include.FilterNMostCommon import FilterNMostCommon


pipeline_normal = Pipeline([
    ('datetime_creator', DatetimeFromTimestamp()),
])

pipeline_1hot = Pipeline([
    ('dataframe_selector', DataFrameSelector(['product_category'])),
    ('filter_n_most_common', FilterNMostCommon(N=10, minRelFreq=0.01,␣
 ↪attribute_name='product_category')),
    ('1hot_encoder', OneHotEncoder(sparse = False))
])

pipeline_analysis = FeatureUnion(transformer_list=[
    ('pipeline_normal', pipeline_normal),
    ('pipeline_1hot', pipeline_1hot),
```

```
])
```

[4]:
```
train_data_treated = pipeline_analysis.fit_transform(train_data)
categories = pipeline_1hot.named_steps['1hot_encoder'].categories_
cols_train = list(train_data) + ['datetime'] + categories[0].tolist()
```

[5]:
```
test_data_treated = pipeline_analysis.transform(test_data)
cols_test = list(test_data) + ['datetime'] + categories[0].tolist()
```

2.0.1 Convert to DataFrame

Not necessary, we can always use the index from cols cols.index('C20')

[9]:
```
train_data_treated = pd.DataFrame(train_data_treated, columns = cols_train)
train_data_treated.head()
```

[9]:
```
   id      timestamp product_id product_department product_category   card_id  \
0   0  1413851531856   f3845767           1fbe01fe         28905ebd  ecad2386
1   1  1413851817483   f3845767           1fbe01fe         28905ebd  ecad2386
2   2  1413852597526   f3845767           1fbe01fe         28905ebd  ecad2386
3   3  1413851283020   f3845767           1fbe01fe         28905ebd  ecad2386
4   4  1413849935779   9166c161           fe8cc448         0569f928  ecad2386

    user_id  C15 C16   C17  …     C20  C21  amount isfraud  \
0  a99f214a  320  50  1722  …      -1   79  184.09       0
1  a99f214a  320  50  1722  …  100084   79  184.09       0
2  a99f214a  320  50  1722  …  100084   79  184.09       0
3  a99f214a  320  50  1722  …  100084   79  184.09       0
4  a99f214a  320  50  2161  …      -1  157  196.98       0

                    datetime 28905ebd 3e814130 50e219e0 Other f028772b
0  2014-10-21 00:32:11.856000        1        0        0     0        0
1  2014-10-21 00:36:57.483000        1        0        0     0        0
2  2014-10-21 00:49:57.526000        1        0        0     0        0
3  2014-10-21 00:28:03.020000        1        0        0     0        0
4  2014-10-21 00:05:35.779000        0        0        0     1        0

[5 rows x 22 columns]
```

[10]:
```
test_data_treated = pd.DataFrame(test_data_treated, columns = cols_test)
test_data_treated.head()
```

[10]:
```
         id      timestamp product_id product_department product_category  \
0  32263877  1414540656054   c4e18dd6           85f751fd         50e219e0
1  32263886  1414540614666   968765cd           6399eda6         f028772b
2  32263890  1414540692012   7e091613           e151e245         f028772b
3  32263895  1414540720045   7e091613           e151e245         f028772b
4  32263896  1414540641750   c4e18dd6           85f751fd         50e219e0
```

```
    card_id   user_id  C15 C16   C17  …  C19     C20  C21  amount  \
0  92f5800b  a99f214a  320  50  2424  …  161  100193   71  191.77
1  ecad2386  a99f214a  320  50  2526  …  167  100075  221  200.45
2  ecad2386  a99f214a  320  50  1872  …   39      -1   23  169.51
3  ecad2386  a99f214a  320  50  1872  …   39      -1   23  169.51
4  73206397  cc6c0613  320  50  2665  …   35      -1  221  191.77

                    datetime 28905ebd 3e814130 50e219e0 Other f028772b
0  2014-10-28 23:57:36.054000        0        0        1     0        0
1  2014-10-28 23:56:54.666000        0        0        0     0        1
2  2014-10-28 23:58:12.012000        0        0        0     0        1
3  2014-10-28 23:58:40.045000        0        0        0     0        1
4  2014-10-28 23:57:21.750000        0        0        1     0        0

[5 rows x 21 columns]
```

3 Overall analysis

- There are no null values in neither of the dataset, however, the are clearly values which are ***imputed to -1*** in the C20 attribute as stated in Kaggle. These were probably nulls.
- 17% of the entries in the data are fraudulent.

[56]:
```
test_data_treated.info(null_counts = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32369524 entries, 0 to 32369523
Data columns (total 21 columns):
id                    32369524 non-null object
timestamp             32369524 non-null object
product_id            32369524 non-null object
product_department    32369524 non-null object
product_category      32369524 non-null object
card_id               32369524 non-null object
user_id               32369524 non-null object
C15                   32369524 non-null object
C16                   32369524 non-null object
C17                   32369524 non-null object
C18                   32369524 non-null object
C19                   32369524 non-null object
C20                   32369524 non-null object
C21                   32369524 non-null object
amount                32369524 non-null object
isfraud               32369524 non-null object
datetime              32369524 non-null object
28905ebd              32369524 non-null object
3e814130              32369524 non-null object
```

```
50e219e0              32369524 non-null object
Other                 32369524 non-null object
dtypes: object(21)
memory usage: 5.1+ GB
```

[54]:
```
train_data_treated.info(null_counts = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32369524 entries, 0 to 32369523
Data columns (total 22 columns):
id                    32369524 non-null object
timestamp             32369524 non-null object
product_id            32369524 non-null object
product_department    32369524 non-null object
product_category      32369524 non-null object
card_id               32369524 non-null object
user_id               32369524 non-null object
C15                   32369524 non-null object
C16                   32369524 non-null object
C17                   32369524 non-null object
C18                   32369524 non-null object
C19                   32369524 non-null object
C20                   32369524 non-null object
C21                   32369524 non-null object
amount                32369524 non-null object
isfraud               32369524 non-null object
datetime              32369524 non-null object
28905ebd              32369524 non-null object
3e814130              32369524 non-null object
50e219e0              32369524 non-null object
Other                 32369524 non-null object
f028772b              32369524 non-null object
dtypes: object(22)
memory usage: 5.3+ GB
```

3.0.1 Value counts

[32]:
```
print("Train: ")
fraud_counts = train_data['isfraud'].value_counts()
print(fraud_counts)
print(fraud_counts / train_data.shape[0])
```

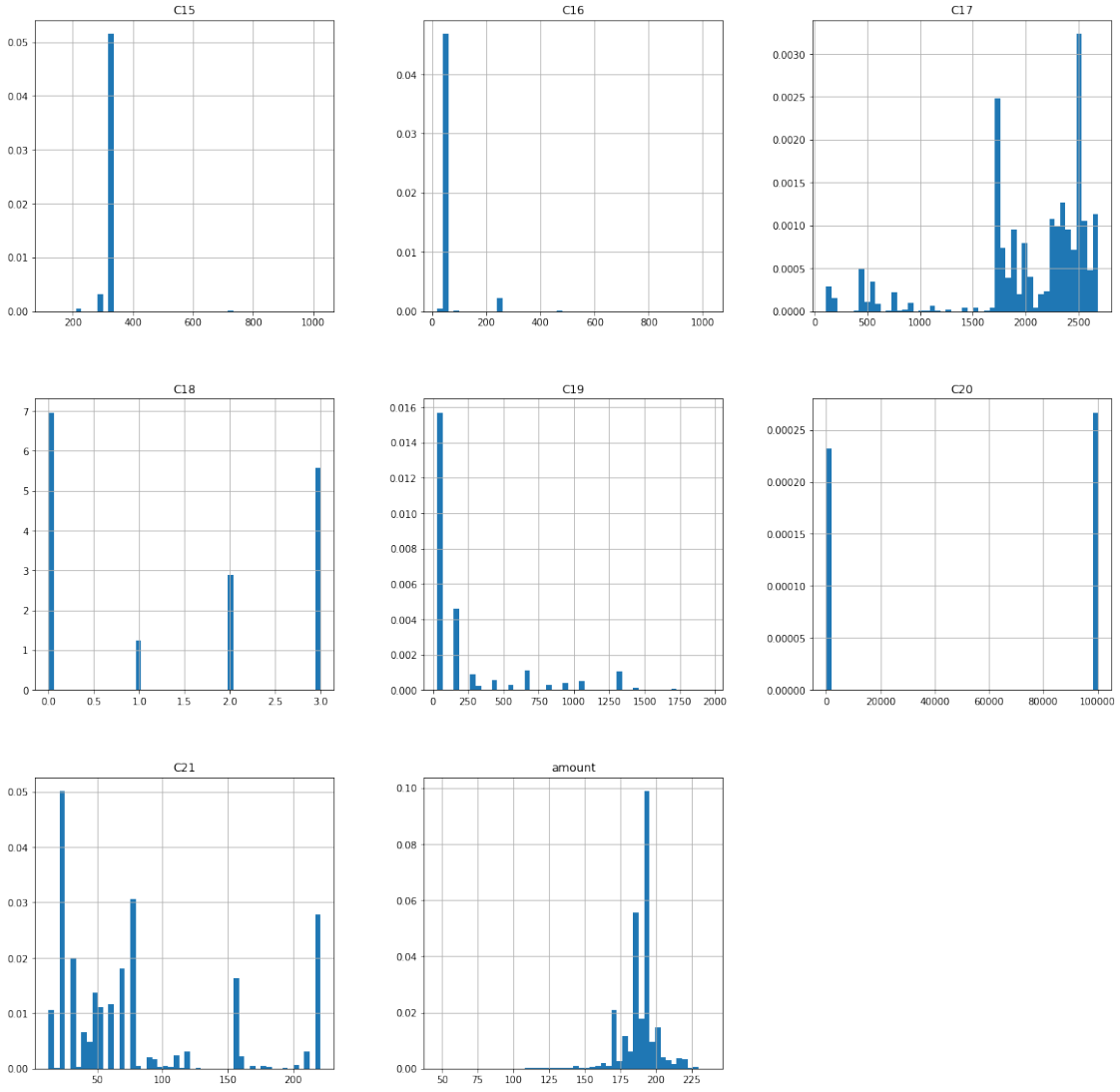
```
Train:
0    26819827
1     5549697
Name: isfraud, dtype: int64
0    0.828552
1    0.171448
Name: isfraud, dtype: float64
```

3.0.2 Histograms

[6]:
```
train_hist = train_data[['C15', 'C16', 'C17', 'C18', 'C19', 'C20', 'C21',␣
↪'amount']]
```

[7]:
```
train_hist.hist(bins = 50, figsize=(20,20), density=True)
plt.plot()
```

[7]: []



[63]:
```
train_data['C20'].value_counts()[:5]
```

[63]: -1 15059166
100084 2033777

```
 100111     1448778
 100148     1374808
 100077     1306586
Name: C20, dtype: int64
```

4 One by one analyzis

4.0.1 timestamp/datetime

- The train data contains data from 8 days, while the test data contains data from the 2 days after.
- Although there doesn't seem to exist a clear relevance of time of the day in the fraction of frauds, it should be used also.
- Thoughout the days the fraud rate is also the same.

[40]:
```
train_data_treated.loc[0,'datetime'], train_data_treated.iloc[-1]['datetime']
```

[40]:
```
(Timestamp('2014-10-21 00:32:11.856000'),
 Timestamp('2014-10-28 23:19:01.526000'))
```

[55]:
```
test_data_treated.loc[0,'datetime'], test_data_treated.iloc[-1]['datetime']
```

[55]:
```
(Timestamp('2014-10-28 23:57:36.054000'),
 Timestamp('2014-10-30 23:55:45.814000'))
```

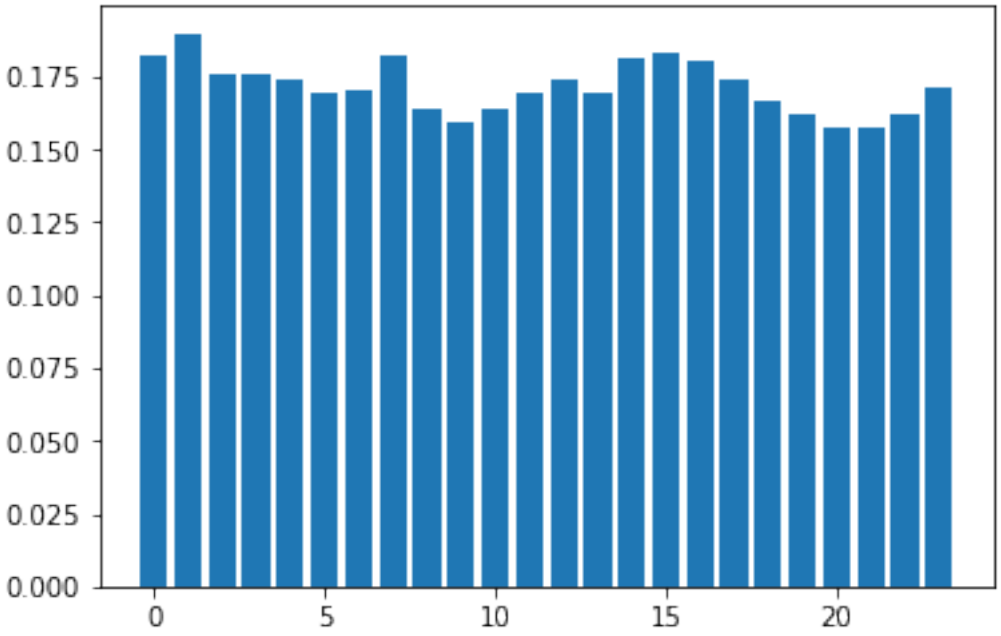
4.0.2 Hour

[11]:
```
datetime = train_data_treated['datetime']
train_data_treated['hour_of_day'] = [dt.time().hour for dt in datetime]
train_data_treated['hour_of_day'].value_counts(sort=False)
```

[11]:
```
0      658895
1      774563
2      983039
3     1049410
4     1404619
5     1605674
6     1427460
7     1472916
8     1729277
9     1894252
10    1709843
11    1670682
12    1775172
13    1944962
14    1720095
15    1646324
16    1661153
```

```
17    1693217
18    1466573
19    1058561
20     892866
21     784973
22     716090
23     628908
Name: hour_of_day, dtype: int64
```

[12]:
```
train_data_treated['isfraud'] = train_data_treated['isfraud'].astype(int)
frauds_by_hour_of_day = train_data_treated[['hour_of_day', 'isfraud']].
 ↪groupby(by='hour_of_day').mean()
frauds_by_hour_of_day.index
plt.bar(frauds_by_hour_of_day.index, frauds_by_hour_of_day['isfraud'])
plt.show()
```



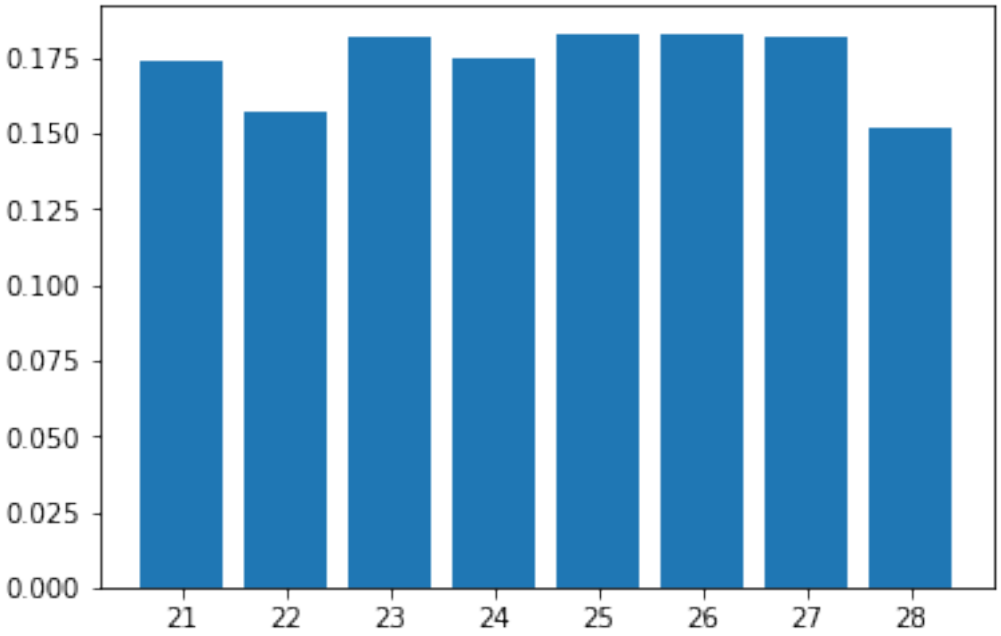
4.0.3 Day

[17]:
```
train_data_treated['day'] = [dt.day for dt in datetime]
train_data_treated['day'].value_counts(sort=False)
```

[17]:
```
21    4122995
22    5337126
23    3870752
24    3335302
```

```
25    3363122
26    3835892
27    3225010
28    5279325
Name: day, dtype: int64
```

[18]:
```
frauds_by_day = train_data_treated[['day', 'isfraud']].groupby(by='day').mean()
frauds_by_day.index
plt.bar(frauds_by_day.index, frauds_by_day['isfraud'])
plt.show()
```



4.1 Product

- Almost every department is linked to a single category, which makes the category information very redundant when using the department.
- ID's in average are linked to 2.75 different `department_category` combinations, separating the ID information from this combination would also result in a lot of redudancy.
- Therefore, it is best to use the concatenated `cat_dep_id` attribute
- Different product categories have different fraud rates, hence it would probably be useful to label encode them with the values sorted by fraud probability

[25]:
```
len(train_data['product_category'].value_counts())
```

[25]: 26

[26]:
```
len(train_data['product_department'].value_counts())
```

[26]: 4551

[29]:
```
category_department = pd.Series([a + '-' + b for a, b in␣
↪zip(train_data['product_category'], train_data['product_department'])])
len(category_department.value_counts())
```

[29]: 4552

[27]:
```
len(train_data['product_id'].value_counts())
```

[27]: 7298

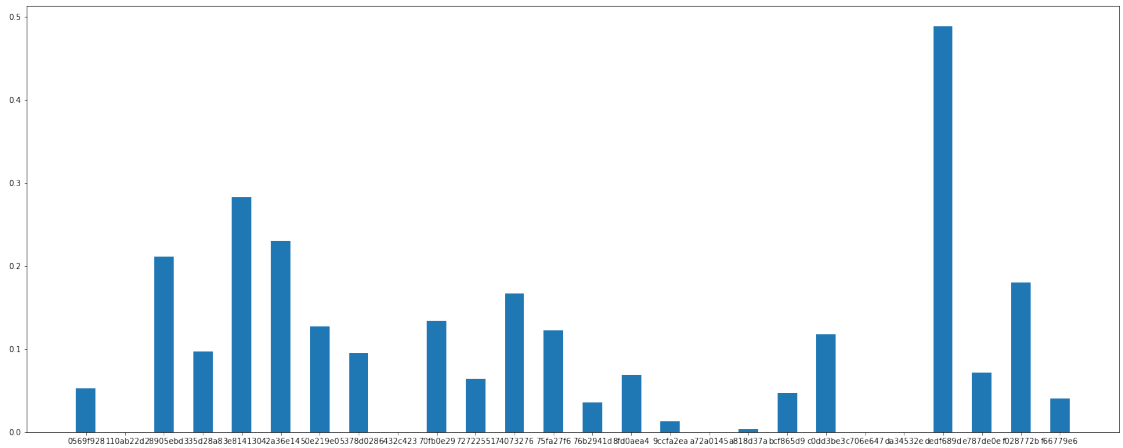
[28]:
```
category_department_id = pd.Series([a + '-' + b + '-' + c for (a, b), c in␣
↪zip(zip(train_data['product_category'], train_data['product_department']),␣
↪train_data['product_id'])])
len(category_department_id.value_counts())
```

[28]: 20036

[30]:
```
20036 / 7298
```

[30]: 2.745409701288024

[115]:
```
frauds_by_category = train_data[['product_category', 'isfraud']].
↪groupby(by='product_category').mean()
plt.figure(figsize=(25,10))
plt.bar(frauds_by_category.index, frauds_by_category['isfraud'], width=0.5)
plt.show()
```



4.2 User & Card

- We want to know if the same happens with `user_id` and `card_id`

- Each card is clearly almost always used only by one user, thus, we have a lot of redundant information again.
- Perhaps it would not make a lot of sense in a live system prediction through the `user_id` or `card_id`, however, it will likely be helpful here.

[119]:
```
len(train_data['user_id'].value_counts())
```

[119]: 2217213

[120]:
```
len(train_data['card_id'].value_counts())
```

[120]: 8094

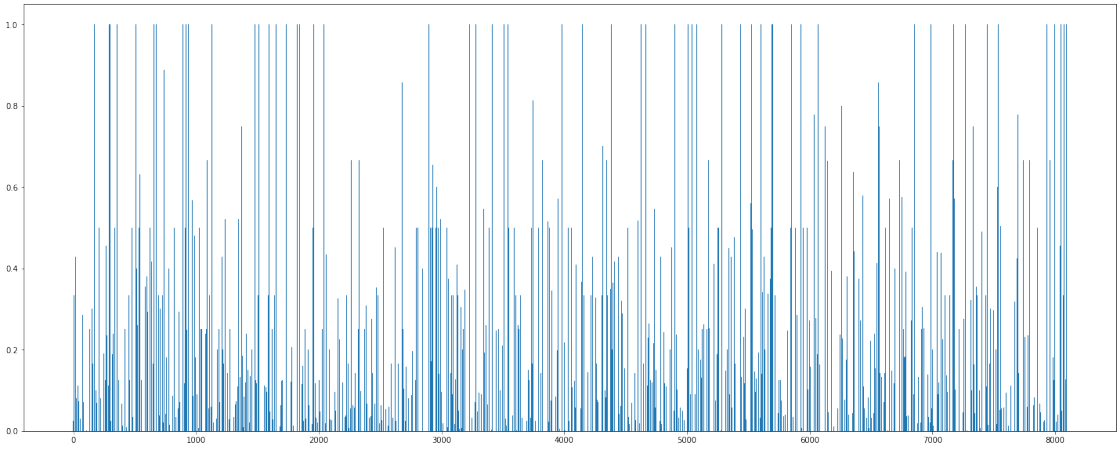
[121]:
```
card_user = pd.Series([a + '-' + b for a, b in zip(train_data['card_id'],␣
 ↪train_data['user_id'])])
len(card_user.value_counts())
```

[121]: 2316047

[123]:
```
2316047/2217213
```

[123]: 1.0445757804956042

[128]:
```
frauds_by_card = train_data[['card_id', 'isfraud']].groupby(by='card_id').mean()
plt.figure(figsize=(25,10))
plt.bar(list(range(len(frauds_by_card))), frauds_by_card['isfraud'])
plt.show()
```



4.3 Anonymized Categorical Variables

- In the kaggle page it said that the `Cxx` attributes are categorical. I would like to check if that is true

- They appear to be because they are all integeres and with no seamless connection correlation to the fraud rate.
- Given that `C18` has only 4 different values that are evenly enough distributed, perhaps one-hot encoding them would be good.
- `C15` and `C16` have the exact same amounts for a lot of values. After analyzing the concatenation of the 2 attributes we can see that it produces almost the same number of unique values, which means we can (and should) use them that way
- `C18` and `C17` also have 1.05 ratio of dependency

4.4 C15

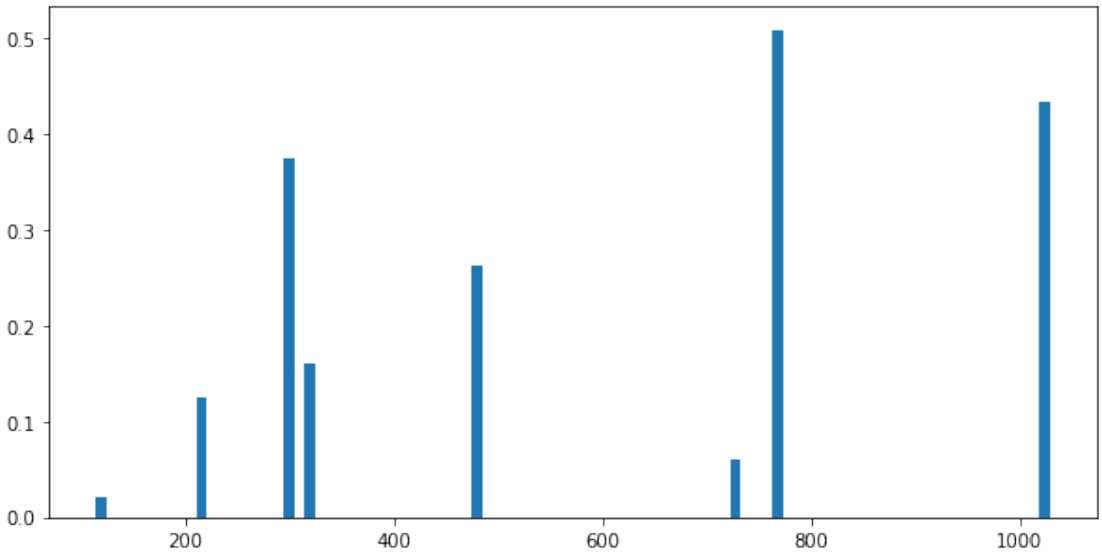
[129]:
```
train_data['C15'].value_counts()
```

[129]:
```
320     30164042
300      1852330
216       283371
728        63166
120         2450
480         1875
1024        1269
768         1021
Name: C15, dtype: int64
```

[142]:
```
x = train_data['C15']
x.min(), x.max()
```

[142]:
```
(120, 1024)
```

[35]:
```
frauds_by_C15 = train_data[['C15', 'isfraud']].groupby(by='C15').mean()
plt.figure(figsize=(10,5))
plt.bar(list(frauds_by_C15.index), frauds_by_C15['isfraud'], width=10)
plt.show()
```



4.5 C16

[130]:
```
train_data['C16'].value_counts()
```

[130]:
```
50      30440126
250      1487082
36        283371
480        89164
90         63166
20          2450
320         1875
768         1269
1024        1021
Name: C16, dtype: int64
```

[143]:
```
x = train_data['C16']
x.min(), x.max()
```

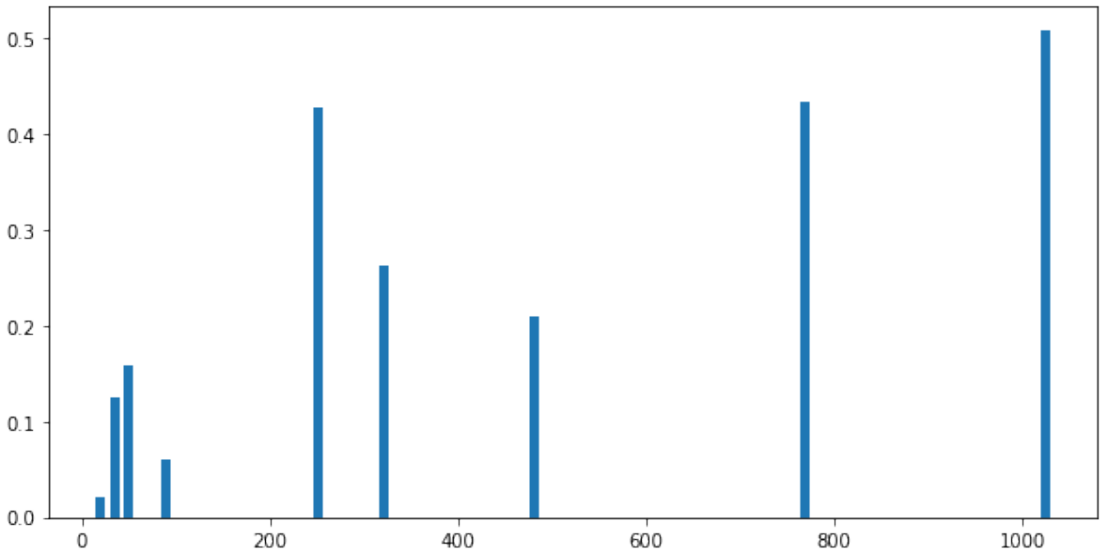
[143]:
```
(20, 1024)
```

[11]:
```
frauds_by_C16
```

[11]:
```
        isfraud
C16
20    0.020816
36    0.125037
50    0.159458
90    0.060301
```

```
250    0.427809
320    0.263467
480    0.210017
768    0.433412
1024   0.508325
```

[36]:
```
frauds_by_C16 = train_data[['C16', 'isfraud']].groupby(by='C16').mean()
plt.figure(figsize=(10,5))
plt.bar(frauds_by_C16.index, frauds_by_C16['isfraud'], width=10)
plt.show()
```



4.5.1 C15 + C16

[83]:
```
C15_C16 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C15'],␣
 ↪train_data['C16'])])
C15_C16.value_counts()
```

```
10
```

[83]:
```
320-50      30074878
300-250      1487082
300-50        365248
216-36        283371
320-480        89164
728-90         63166
120-20          2450
480-320         1875
1024-768        1269
```

```
768-1024        1021
dtype: int64
```

4.6 C17

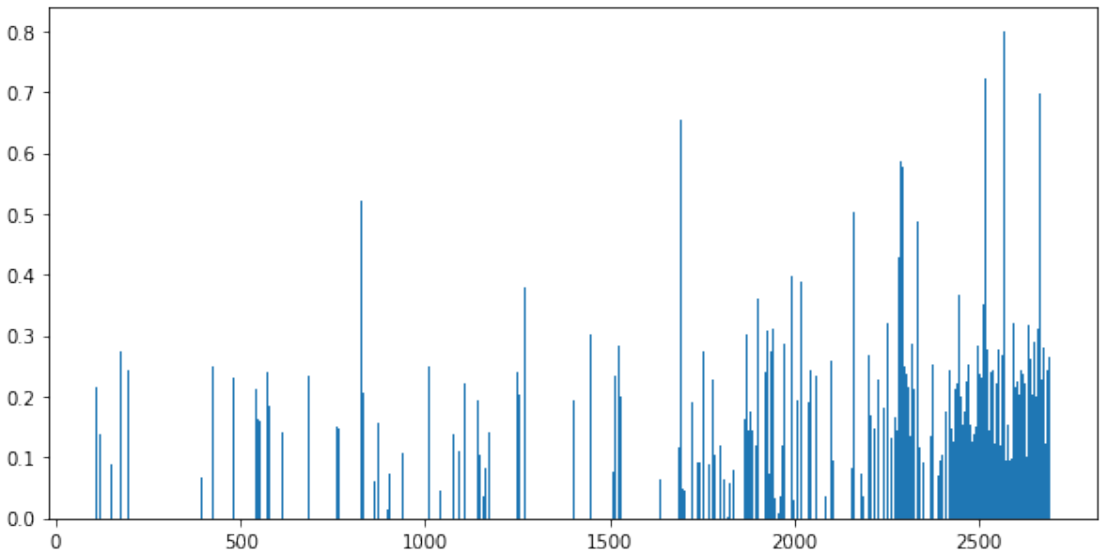
[144]:
```
len(train_data['C17'].value_counts())
```

[144]:
```
372
```

[146]:
```
x = train_data['C17']
x.min(), x.max()
```

[146]:
```
(112, 2688)
```

[38]:
```
frauds_by_C17 = train_data[['C17', 'isfraud']].groupby(by='C17').mean()
plt.figure(figsize=(10,5))
plt.bar(frauds_by_C17.index, frauds_by_C17['isfraud'], width=5)
plt.show()
```

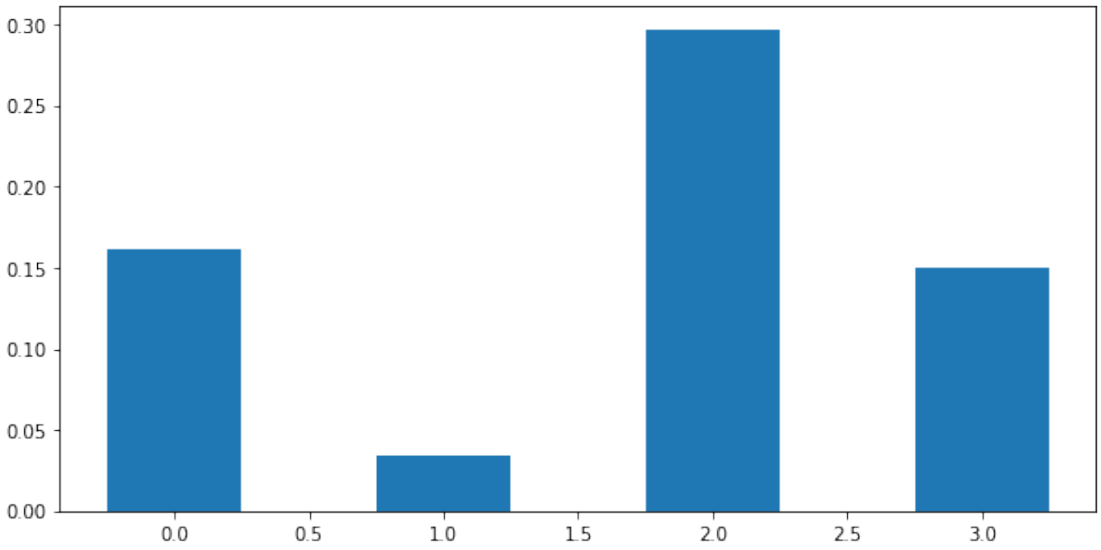


4.7 C18

[147]:
```
train_data['C18'].value_counts()
```

[147]:
```
0    13520676
3    10819205
2     5605622
1     2424021
Name: C18, dtype: int64
```

[45]:
```
frauds_by_C18 = train_data[['C18', 'isfraud']].groupby(by='C18').mean()
plt.figure(figsize=(10,5))
plt.bar(frauds_by_C18.index, frauds_by_C18['isfraud'], width=0.5)
plt.show()
```



4.8 C19

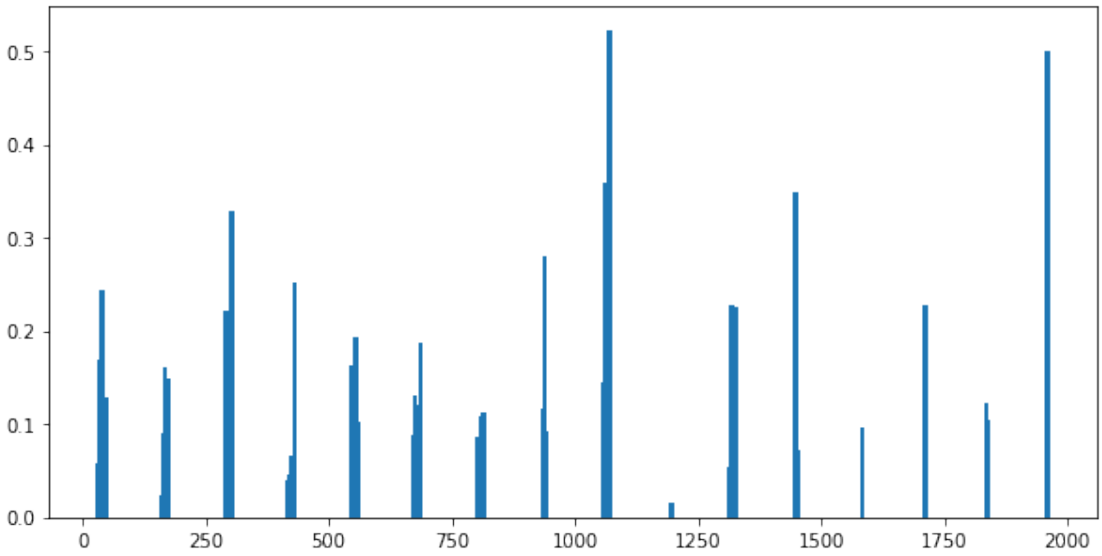
[8]:
```
len(train_data['C19'].value_counts())
```

[8]: 63

[152]:
```
x = train_data['C19']
x.min(), x.max()
```

[152]: (33, 1959)

[41]:
```
frauds_by_C19 = train_data[['C19', 'isfraud']].groupby(by='C19').mean()
plt.figure(figsize=(10,5))
plt.bar(frauds_by_C19.index, frauds_by_C19['isfraud'], width=10)
plt.show()
```



4.9 C20

[49]:
```
vc = train_data['C20'].value_counts()
print('\nSize: ',len(vc))
vc[:3]
```

```
Size:  172
```

[49]:
```
-1         15059166
 100084     2033777
 100111     1448778
Name: C20, dtype: int64
```

[159]:
```
x = train_data['C20']
x.min(), x.max()
```

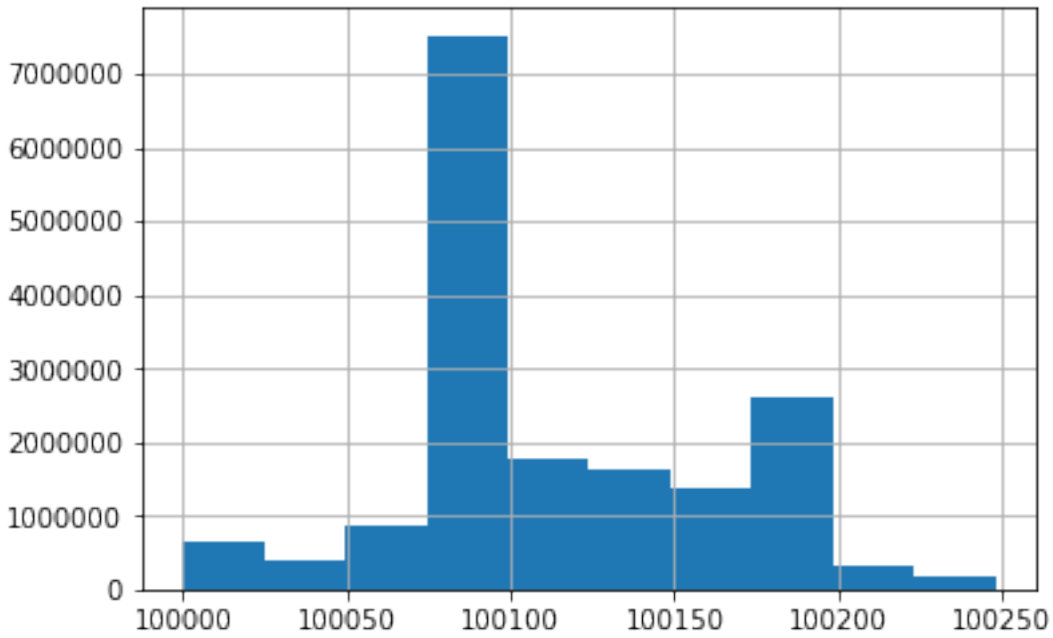
[159]:
```
(-1, 100248)
```

[160]:
```
x = x[x != -1]
x.min(),x.max()
```

[160]:
```
(100000, 100248)
```

[163]:
```
x.hist()
plt.plot()
```

[163]:
```
[]
```



[68]:
```
frauds_by_C20[:1]
```

[68]:
```
      isfraud
C20
-1   0.194456
```

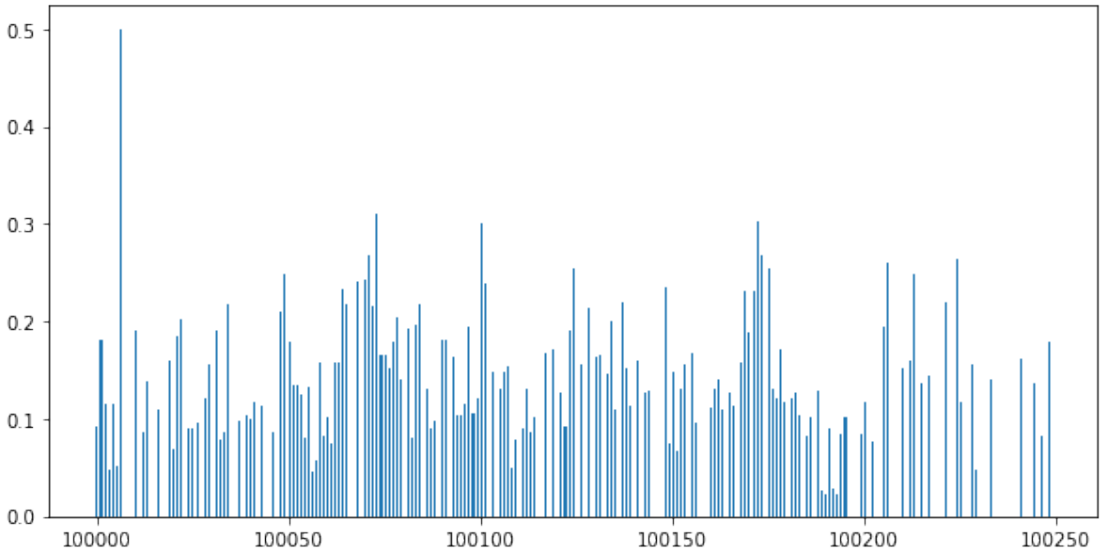
[10]:
```
frauds_by_C20
```

[10]:
```
           isfraud
C20
-1        0.194456
 100000   0.092080
 100001   0.180725
 100002   0.114857
 100003   0.047644
 100004   0.115081
 100005   0.050887
 100006   0.500000
 100008   0.000000
 100010   0.190141
 100012   0.086391
 100013   0.138319
 100016   0.109018
 100019   0.160583
 100020   0.069689
```

```
  100021  0.184571
  100022  0.203259
  100024  0.089552
  100025  0.090951
  100026  0.096326
  100027  0.000000
  100028  0.120618
  100029  0.156818
  100031  0.190445
  100032  0.078293
  100033  0.086037
  100034  0.217997
  100037  0.097717
  100039  0.103116
  100040  0.099338
…              …
  100188  0.128405
  100189  0.026790
  100190  0.022487
  100191  0.089906
  100192  0.029060
  100193  0.021976
  100194  0.084030
  100195  0.102525
  100198  0.000000
  100199  0.084583
  100200  0.118162
  100202  0.076826
  100205  0.195389
  100206  0.259817
  100209  0.000000
  100210  0.152141
  100212  0.159697
  100213  0.248408
  100215  0.136744
  100217  0.144499
  100221  0.220731
  100224  0.265157
  100225  0.117903
  100228  0.156632
  100229  0.046944
  100233  0.140118
  100241  0.161784
  100244  0.136042
  100246  0.083333
  100248  0.179362
```

[172 rows x 1 columns]

[9]:
```
frauds_by_C20 = train_data[['C20', 'isfraud']].groupby(by='C20').mean()
frauds_by_C20_clean = frauds_by_C20[1:]
plt.figure(figsize=(10,5))
plt.bar(frauds_by_C20_clean.index, frauds_by_C20_clean['isfraud'], width=.5)
plt.show()
```



4.10 C21

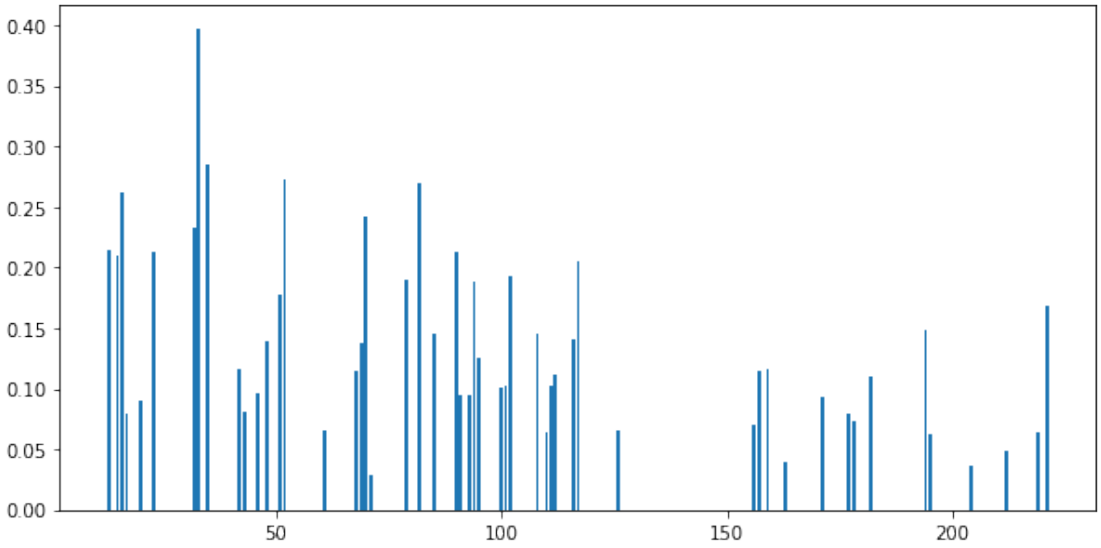
[86]:
```
len(train_data['C21'].value_counts())
```

[86]: 52

[156]:
```
x = train_data['C21']
x.min(), x.max()
```

[156]: (13, 221)

[77]:
```
frauds_by_C21 = train_data[['C21', 'isfraud']].groupby(by='C21').mean()
plt.figure(figsize=(10,5))
plt.bar(frauds_by_C21.index, frauds_by_C21['isfraud'], width=.75)
plt.show()
```



4.10.1 Combining C18 with others

[92]:
```
C18_C17 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C18'],␣
↪train_data['C17'])])
len(C18_C17.value_counts())
```

[92]: 391

[93]:
```
C18_C19 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C18'],␣
↪train_data['C19'])])
len(C18_C19.value_counts())
```

[93]: 112

[95]:
```
C18_C21 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C18'],␣
↪train_data['C21'])])
len(C18_C21.value_counts())
```

[95]: 74

4.11 Checking combinations of C19, C20 and C21

[101]:
```
C19_C20 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C19'],␣
↪train_data['C20'])])
len(C19_C20.value_counts())
```

[101]: 1918

[102]:
```
C19_C21 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C19'],␣
↪train_data['C21'])])
len(C19_C21.value_counts())
```

[102]: 186

[103]:
```
C21_C20 = pd.Series([str(a) + '-' + str(b) for a, b in zip(train_data['C21'],␣
↪train_data['C20'])])
len(C21_C20.value_counts())
```

[103]: 2159

4.12 Amount

5 Correlations

[165]:
```
corr_matrix = train_data[['C15', 'C16', 'C17', 'C18', 'C19', 'C20', 'C21',␣
↪'amount', 'isfraud']].corr()
corr_matrix
```

[165]:

	C15	C16	C17	C18	C19	C20	C21	\
C15	1.000000	-0.086409	-0.003177	0.019357	0.048154	0.009145	-0.011479	
C16	-0.086409	1.000000	0.064617	0.078688	-0.078528	-0.046388	-0.072418	
C17	-0.003177	0.064617	1.000000	-0.263515	-0.183348	0.026113	0.434140	
C18	0.019357	0.078688	-0.263515	1.000000	0.082197	0.020896	-0.582077	
C19	0.048154	-0.078528	-0.183348	0.082197	1.000000	0.103951	-0.215435	
C20	0.009145	-0.046388	0.026113	0.020896	0.103951	1.000000	-0.035945	
C21	-0.011479	-0.072418	0.434140	-0.582077	-0.215435	-0.035945	1.000000	
amount	0.016566	0.082174	0.257441	-0.054920	0.056989	0.073147	0.110930	
isfraud	-0.032106	0.133812	-0.056598	0.023604	0.002963	-0.056971	-0.065579	

	amount	isfraud
C15	0.016566	-0.032106
C16	0.082174	0.133812
C17	0.257441	-0.056598
C18	-0.054920	0.023604
C19	0.056989	0.002963
C20	0.073147	-0.056971
C21	0.110930	-0.065579
amount	1.000000	-0.098828
isfraud	-0.098828	1.000000

6 Observing frauds

[18]:
```
#a = train_data['user_id'].value_counts()
a[a < 30]
```

[18]:
```
4f2638e2    29
be91513c    29
ad5b1ded    29
a4d28d74    29
20419348    29
a9e5206e    29
7fce729a    29
caa5deaa    29
055acd14    29
1202cb62    29
79adbec1    29
b3e76e2e    29
fe6fe698    29
eff814d1    29
a2d145f0    29
03f85ebf    29
dd2ec838    29
fd222dd5    29
06239d2e    29
ce320dcb    29
cfe68cae    29
633dc08e    29
cb48f81d    29
cdb60770    29
09586b29    29
5d2df384    29
251c97ce    29
92454a51    29
b1e57fbd    29
d3490bf4    29
            ..
6f7ba7ea     1
62a60475     1
3b34d54d     1
899aecba     1
3bd06402     1
cb30953a     1
b233ebf5     1
30dfe019     1
477066d5     1
0150088a     1
ede6af8f     1
```

```
987f1763     1
c71201e5     1
166e4b03     1
22f32cb7     1
d2ead847     1
4157c3b2     1
38c71af5     1
a6c29266     1
a6c61e60     1
05d62b17     1
9c8dbea1     1
3707bc8f     1
48dee450     1
7082ea3f     1
af74d90f     1
91829dd8     1
56d29f30     1
62274dfb     1
355cddd7     1
Name: user_id, Length: 2205396, dtype: int64
```

[19]:
```
train_data[train_data['user_id'] == '4f2638e2']
```

[19]:

	id	timestamp	product_id	product_department	\
2592636	2592636	1413894585084	8eb11125	39cffaa4	
2603166	2603166	1413899086392	e8cab48f	f17ebd97	
2666638	2666638	1413897322452	e8cab48f	f17ebd97	
6599112	6599112	1413975324564	e8cab48f	f17ebd97	
6646930	6646930	1413973524420	e8cab48f	f17ebd97	
26412721	26412721	1414435576305	19944fab	c6af341d	
29584925	29584925	1414498610271	c4e18dd6	c6af341d	
31210869	31210869	1414518736706	5888d1c7	f17ebd97	
31251551	31251551	1414516865868	5888d1c7	f17ebd97	
31306053	31306053	1414517602507	5888d1c7	f17ebd97	
31330627	31330627	1414518490623	c4e18dd6	5b787406	
31344355	31344355	1414517043692	5888d1c7	f17ebd97	
31376023	31376023	1414519211801	d9e5838f	ad639ab8	
31407295	31407295	1414521444412	f3ca2e42	5b787406	
31443318	31443318	1414522658619	f3ca2e42	5b787406	
31475533	31475533	1414520279515	f3ca2e42	5b787406	
31522367	31522367	1414519368240	f3ca2e42	5b787406	
31528076	31528076	1414520979622	f3ca2e42	5b787406	
31535696	31535696	1414520593642	f3ca2e42	5b787406	
31544951	31544951	1414520272091	f3ca2e42	5b787406	
31549483	31549483	1414519266017	f3ca2e42	5b787406	
31554229	31554229	1414521474458	f3ca2e42	5b787406	
31565599	31565599	1414521768433	f3ca2e42	5b787406	

```
31568985  31568985  1414521859079   f3ca2e42          5b787406
31569551  31569551  1414522772502   f3ca2e42          5b787406
31582014  31582014  1414520796748   f3ca2e42          5b787406
31582367  31582367  1414522413625   f3ca2e42          5b787406
31648994  31648994  1414523219166   8eb11125          39cffaa4
31649453  31649453  1414524678835   8eb11125          39cffaa4

         product_category   card_id   user_id  C15  C16   C17  C18   C19  C20  \
2592636          50e219e0  ecad2386  4f2638e2  320   50  2253    2   303   -1
2603166          50e219e0  ecad2386  4f2638e2  320   50  2253    2   303   -1
2666638          50e219e0  ecad2386  4f2638e2  320   50  2253    2   303   -1
6599112          50e219e0  ecad2386  4f2638e2  320   50  2476    2   167   -1
6646930          50e219e0  ecad2386  4f2638e2  320   50  2476    2   167   -1
26412721         50e219e0  ecad2386  4f2638e2  320   50  2512    2   291   -1
29584925         50e219e0  ecad2386  4f2638e2  320   50  2512    2   291   -1
31210869         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31251551         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31306053         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31330627         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31344355         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31376023         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31407295         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31443318         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31475533         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31522367         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31528076         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31535696         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31544951         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31549483         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31554229         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31565599         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31568985         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31569551         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31582014         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31582367         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31648994         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1
31649453         50e219e0  ecad2386  4f2638e2  320   50  2684    2  1327   -1

          C21  amount  isfraud
2592636    52  149.06        0
2603166    52  217.06        0
2666638    52  217.06        0
6599112    23  217.06        0
6646930    23  217.06        0
26412721   52  209.97        0
29584925   52  209.97        1
31210869   52  217.06        0
```

31251551	52	217.06	0
31306053	52	217.06	0
31330627	52	205.11	0
31344355	52	217.06	0
31376023	52	150.66	0
31407295	52	205.11	0
31443318	52	205.11	0
31475533	52	205.11	0
31522367	52	205.11	0
31528076	52	205.11	0
31535696	52	205.11	0
31544951	52	205.11	0
31549483	52	205.11	0
31554229	52	205.11	0
31565599	52	205.11	0
31568985	52	205.11	0
31569551	52	205.11	0
31582014	52	205.11	0
31582367	52	205.11	0
31648994	52	149.06	0
31649453	52	149.06	0